DIAGNÓSTICO DE FALLAS INTERMITENTES EN SISTEMAS AUTOMÁTICOS DE MANUFACTURA USANDO MACHINE LEARNING

DIAGNOSIS OF INTERMITTENT FAILURES IN AUTOMATIC MANUFACTURING SYSTEMS USING MACHINE LEARNING

Samantha Acosta Ruiz
Tecnológico Nacional de México / IT de Aguascalientes, México
G21153070@aguascalientes.tecnm.mx

Elvia Ruiz Beltrán
Tecnológico Nacional de México / IT de Aguascalientes, México
eruiz@aguascalientes.tecnm.mx

Josué Antonio Prieto Olivares
Tecnológico Nacional de México / IT de Aguascalientes, México
r21153083@aguascalientes.tecnm.mx

Jorge Luis Orozco Mora
Tecnológico Nacional de México / IT de Aguascalientes, México
Jorge.om@aguascalientes.tecnm.mx

Jorge Octavio Valdés Valadez
Tecnológico Nacional de México / IT de Aguascalientes, México
Jorge.vv@aguascalientes.tecnm.mx

Recepción: 12/noviembre/2022 **Aceptación:** 19/marzo/2023

Resumen

En este trabajo, se proponen modelos de clasificación para el diagnóstico de fallas intermitentes en sistemas automatizados de manufactura, usando un enfoque de Machine Learning. Se implementaron varios modelos de aprendizaje automático como: Máquinas de Soporte Vectorial, Árboles de Decisión, Métodos de ensamble (Adaboost y RandomForest) y Redes Neuronales en Python. Se creó un escenario en el simulador de Factory I/O, así diseñar el sistema de manufactura y poder hacer la recolección de los datos del simulador. Para determinar el mejor modelo de clasificación se validaron los algoritmos con los datos de nuestro escenario.

Palabras clave: Aprendizaje máquina, diagnóstico de fallas, Factory I/O, fallas intermitentes, Industria 4.0.

Abstract

In this work, classification models are proposed for the diagnosis of intermittent failures in automated manufacturing systems, using a Machine Learning approach. Several machine learning models were implemented such as: Support Vector Machines, Decision Trees, Assembly Methods (Adaboost and RandomForest) and Neural Networks in Python. A scenario was created in the Factory I/O simulator, in order to design the manufacturing system and be able to collect the simulator data. To determine the best classification model, the algorithms were validated with the data from our scenario.

Keywords**:** *Factory I/O, failures diagnosis, Industry 4.0, intermittent failures, Machine Learning.*

1. Introducción

Hace tiempo que la automatización de procesos industriales pasó de ser una opción para convertirse en una necesidad para muchas empresas inmersas en procesos de mejora de su competitividad y en el mejoramiento de sus sistemas de producción. Debido a la creciente demanda de productos de la vida cotidiana los sistemas son sometidos a grandes cargas de trabajo, por lo que sufren un mayor desgaste, su vida útil empieza a decaer y su eficiencia empieza a disminuir. La automatización es clave en la industria 4.0 y es un valor en alza entre las empresas industriales. Según publica Fortune Businness Insights, el mercado de la automatización industrial a nivel mundial alcanzó los 157,04 mil millones de dólares en 2018 y se espera que el año 2026 alcance los 296,70 mil millones de dólares [Nexus, 2022]. La fabricación es ahora un negocio impulsado digitalmente, donde datos, perspectivas y la inteligencia, se han convertido en el activo más valioso que tiene una empresa que pueda capturar y automatizar la gestión de recetas en plantas [Vidal, 2018]. Un reciente estudio planteó la siguiente pregunta: “¿Cuáles considera que son los beneficios más importantes de un centro de computación automatizado o desatendido?”. Los beneficios principales de la automatización de operaciones que se citaron con mayor frecuencia fueron: reducción de costos, aumento de la productividad, disponibilidad, confiabilidad y rendimiento [Pat, 2022].

Cuando la industria recibe quejas sobre la calidad del producto, es un hecho que hay un problema en el proceso que está comprometiendo su excelencia. Esto significa que es necesario revisar los componentes que permiten la automatización industrial en la empresa. Cabe recordar que, si los problemas no se resuelven con buenas soluciones de automatización industrial, se comprometerá no solo el desempeño del equipo, que debe ser trasladado a otras actividades, sino también de un elemento indispensable para su competitividad: la calidad del producto. Aquí entraría en problemas la parte de la confiabilidad, que es el conjunto de atributos que permite cuantificar la calidad del servicio prestado y el grado de confianza que el usuario puede depositar en el sistema [Saiz, 2015]. Las amenazas de la confiabilidad son los fallos, errores y averías, que son circunstancias no deseadas que pueden tener como resultado la pérdida de la confianza en el sistema. Para evitarla, se dispone de distintos medios para conseguir la confiabilidad; prevención de fallos, tolerancia a fallos, eliminación de fallos y predicción de fallos. La prevención y la tolerancia a fallos tienen como objetivo proveer al sistema de la capacidad de entregar un servicio en el que se pueda confiar, mientras que la eliminación y la predicción de fallos tienen el objetivo de alcanzar la confianza en esta capacidad, justificando que las especificaciones funcionales, de confiabilidad y de seguridad son las adecuadas, y que el sistema las cumple con una determinada probabilidad [Saiz, 2015].

El enfoque de extracción y análisis de datos en tiempo real durante la fabricación, se transforman rápidamente en inteligencia accionable que reduce los costos y el exceso de trabajo, mejoran la productividad de la línea de producción, reducen los defectos del producto y aumentan la calidad, extienden la vida útil del equipo y comprenden mejor los requisitos del cliente.

Para reducir la pérdida económica, aumentar la seguridad del personal, disminuir el defecto en los productos y aumentar la calidad, es necesario implementar el diagnóstico continuo, lo que permite realizar un seguimiento del estado del sistema, identificando si se producen eventos atípicos o no, y ayuda con la detección de la fuente de incidentes [Vidal, 2018], es decir, las fallas que se pueden presentar en los sistemas.

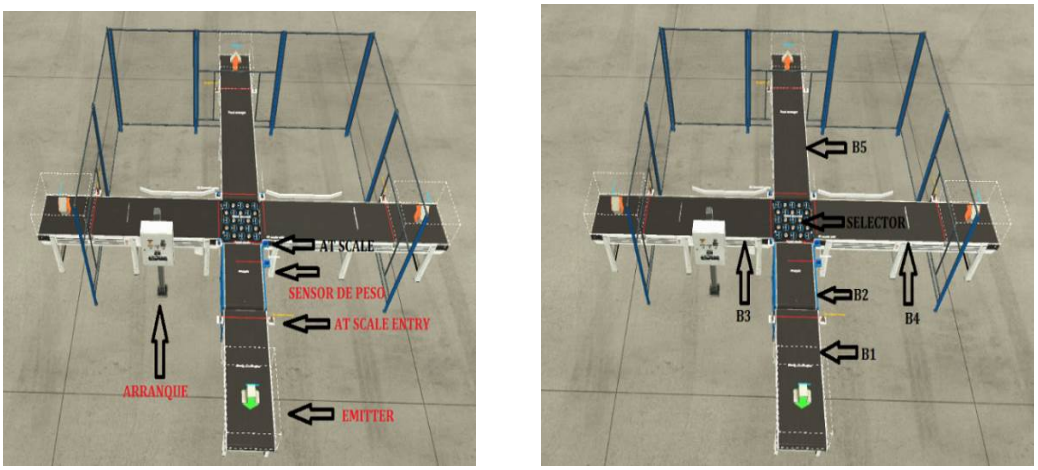
Existen modelos de detección de fallas en sistemas de manufactura usando enfoques de aprendizaje automático capaces de identificar la falla en tiempo real. En [Contreras, 2020], se genera un modelo de mantenimiento predictivo en motores de inducción, usando técnicas de vibraciones y con ello implementan algoritmos como: K-vecinos más cercanos (KNN), Máquinas de Soporte Vectorial (SVM) y Árboles de Decisión. Se observó que el mejor modelo generado fue el de Árboles de Decisión. En cambio, en [Guamán, 2019] se aplicó en un compresor reciprocante de simple efecto doble etapa para la predicción de fallas, donde usó aprendizaje profundo (redes neuronales recurrentes LSTM "Long Short Term Memory"), sobre el análisis de vibraciones. En [Inafuku, 2020] se basó en un sistema de diagnóstico de fallas para inspección y detección de fallas en los rodamientos en motores de inducción, los datos se obtuvieron a través de un brazo de robot móvil. Obtuvieron modelos usando redes neuronales convolucionales, como resultado obtuvieron que no es capaz de clasificar el tipo de falla a un porcentaje mayor al 90%, cabe resaltar que a diferencia de los trabajos anteriores aquí si se localiza con precisión el elemento que está en falla. En [Zúñiga, 2021], se generaron modelos basados en Árboles de Decisión, Bosques Aleatorios y Árboles potenciados, para la predicción de fallas en la producción de piezas en un sistema de manufactura orientado a la industria 4.0 (que se simuló en FESTO), el mejor modelo fue basado en Árboles de Decisión Potenciados. Una de las principales desventajas de los trabajos previos, es que los algoritmos no indican con precisión que elemento está experimentando una falla, si bien uno de ellos usa un robot móvil para monitorear eso no es posible en todos los sistemas de manufactura.

En este trabajo, se propone una metodología de detección de fallas intermitentes sobre motores y sensores de sistemas de manufactura, usando herramientas de aprendizaje automático, cuyos datos se obtuvieron de un sistema de manufactura simulado en Factory I/O. Se emplearon modelos como Árboles de Decisión, Máquinas de Soporte Vectorial, Métodos de ensamble (Adaboost y RandomForest) y Redes Neuronales Artificiales (RNA). Se generaron los modelos con un conjunto de datos de referencia de detección de fallas llamada "Machine Predictive Maintenance Classification", donde se observó que los mejores modelos en

clasificar las fallas fueron las redes neuronales multicapa. Esto mismo se aplicó a la base de datos generada en Factory I/O donde los resultados de las métricas muestran que las redes neuronales son menos precisas en la clasificación que los árboles de decisión potenciados (RandomForest).

2. Métodos

En este trabajo se creó un sistema de manufactura virtual en el simulador de Factory I/O [Prieto, 2020], el cual nos permite construir y controlar procesos industriales en tiempo real. La simulación puede ser controlada en tiempo real mediante la conexión de Factory I/O y equipos externos como controladores lógicos programables (PLC's) o microprocesadores. Se propuso el diseño de un sistema clasificador de cajas, figura 1, dicho sistema puede separar tres tipos de cajas: chica, mediana y grande.



a) Sensores principales. *b) Nomenclatura para banda transportadora.*

Figura 1 Elementos del sistema.

El sistema está conectado a un PLC para la interacción con el simulador Factory I/O, se analizan fallas intermitentes que pueden presentarse en los motores que controlan las bandas transportadoras y en sensores de peso que se usan para hacer la clasificación de las cajas. Las fallas se indujeron de forma manual al estar interactuando con el PLC que controla el sistema. Factory I/O genera los datos desde el momento en que se da de alta el comando para que haga su recolección,

y así los exporte en un formato de archivo CSV. En este archivo se realizó un filtrado sobre los datos que servirían para hacer el entrenamiento.

El proceso del sistema comienza al presionar el botón de arranque, figura 1, posteriormente se activa el evento que inicia la creación de una caja aleatoriamente a partir del alimentador (Emitter) y se deposita en la banda transportadora (B1). Al ser detectada por el primer sensor (At scale entry), este activa la siguiente banda (B2) donde al encontrarse en posición (At scale), es pesada la caja por el sensor analógico de peso. En la intersección de las bandas transportadoras se encuentra un selector que puede moverse a la izquierda, derecha y frontal, es donde se decide a donde enviar cada caja según el peso. Las cajas pequeñas van por la banda B3, las medianas por la banda B4 y las grandes por la banda B5, cada banda tiene su sensor (At left entry, At rigth entry y At forward entry, respectivamente) que envían una señal para activar la banda y así transportar la caja. Al terminar este proceso las cajas llegan a una zona (donde se indican con flechas naranjas) donde son censadas (At exit left, At exit right, At exit front), cuando el sensor detecta las cajas reinicia el sistema para generar una nueva caja y hacer el mismo proceso otra vez. Para más detalle de los elementos del sistema revisar en [Prieto, 2020].

Tecnologías y entornos de trabajo

Se creó un Notebook Jupyter en la plataforma de Google Colaboratory, las librerías de Python para implementar los modelos de aprendizaje máquina fueron: Scikit-Learn (aprendizaje automático), Numpy (funciones matemáticas), y por último TensorFlow (construir redes neuronales profundas).

3. Resultados

Modelado de los algoritmos

Antes de trabajar con los datos que se generaron de Factory I/O, se experimentó con un conjunto de datos llamado “Machine Predictive Maintenance Classification” de la página de Kaggel, el cuál fue creado para aplicar algoritmos de clasificación de fallas. Primeramente, se hizo este caso de estudio para observar que tipo de variables manejan para el diagnóstico de fallas en un mantenimiento predictivo y,

por otra parte, observar de todos los modelos de clasificación cual es el más eficiente para clasificar múltiples-fallas.

Como se observa en la figura 2, se consideraron 10000 datos y 10 variables, donde 9652 son datos sin falla y el resto de las fallas a analizar están divididas en 5 tipos: falla de disipación de calor (112), falla de sobreesfuerzo (78), falla en la alimentación (95), fallas aleatorias (18) y falla por desgaste de la herramienta (45). Estas fallas son referentes al proceso que analizaron del monitoreo de un sistema de manufactura y son las que se presentan en estos sistemas.

	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
0	1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
1	2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
2	3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure
3	4	L47183	L	298.2	308.6	1433	39.5	7	0	No Failure
4	5	L47184	L	298.2	308.7	1408	40.0	9	0	No Failure
...	...	...	...	...	...	...	...	...	...	...
9995	9996	M24855	M	298.8	308.4	1604	29.5	14	0	No Failure
9996	9997	H39410	H	298.9	308.4	1632	31.8	17	0	No Failure
9997	9998	M24857	M	299.0	308.6	1645	33.4	22	0	No Failure
9998	9999	H39412	H	299.0	308.7	1408	48.5	25	0	No Failure
9999	10000	M24859	M	299.0	308.7	1500	40.2	30	0	No Failure

10000 rows × 10 columns

Figura 2 Información del conjunto de datos del caso de estudio en clasificación de fallas.

Primeramente, se eliminaron las columnas del índice de muestreo y el ID del producto ya que no son variables relevantes. Después, se normalizaron los datos y las variables categóricas se pasaron a variables “dummy” a excepción de la variable tipo de falla (lo que se hizo es un diccionario para asignar a cada tipo de falla un índice para que se pueda identificar). Antes de entrenar cada modelo, se separaron los datos, donde el 80% para entrenar y el 20% son de prueba. Finalmente, se obtuvieron modelos con Árboles de Decisión, Máquina de Soporte Vectorial, Modelos de ensambles (Bagging, Boosting) y Redes Neuronales (multicapa y

recurrentes), ya que son los algoritmos más relevantes para la clasificación de fallas (de acuerdo al estado del arte que se revisó). En las tablas 1, 2 y 3 se muestran los resultados del modelo para la Máquina de Soporte Vectorial, según el tipo de kernel y con sus parámetros del valor de C, Grado y Gamma que se usaron.

Si se establece una C=25 ó 20, los resultados de predicción son iguales, y valores superiores no mejoran la predicción. Para un Kernel del tipo polinomial se observa, que los resultados para el polinomio de grado 4 da mejores resultados y el polinomio de grado 5 no mejora esa evaluación. Se recomienda que el tamaño del grado del polinomio, no se entrene mayor a 5, porque caerá en problemas de sobreajuste el modelo, ver tabla 2. Ahora, con un kernel de base radial (RBF), se observan los resultados en la tabla 3.

Tabla 1 Resultados de evaluación para los datos de test para un kernel lineal.

C	Precisión	Exhaustividad	F1-Score	Exactitud
2	0.6384	0.6065	0.5564	0.9845
5	0.7035	0.7253	0.7083	0.9915
10	0.7080	0.7232	0.7114	0.9915
15	0.7260	0.7420	0.7302	0.9925
20	**0.7520**	**0.7539**	**0.7457**	**0.993**
25	**0.7520**	**0.7539**	**0.7457**	**0.993**

Tabla 2 Resultados de evaluación para los datos de test para un kernel polinomial.

Grado	C	Precisión	Exhaustividad	F1-Score	Exactitud
3	2	0.7148	0.7315	0.7202	0.992
	5	0.7220	0.7295	0.7208	0.9925
	10	0.7220	0.7295	0.7208	0.9925
	15	0.7419	0.7587	0.7489	0.9935
	20	0.7419	0.7587	0.7489	0.9935
	25	0.7419	0.7587	0.7489	0.9935
4	2	0.7220	0.7295	0.7208	0.9925
	5	**0.7437**	**0.7711**	**0.7562**	**0.9935**
	10	0.7487	0.7503	0.7453	0.993
	15	0.7355	0.7212	0.7250	0.9925
	20	0.7319	0.7420	0.7351	0.9925
	25	0.7489	0.7420	0.7407	0.9925
5	2	0.7487	0.7503	0.7453	0.993
	5	0.7489	0.7420	0.7407	0.9925
	10	0.7407	0.7300	0.7305	0.992
	15	0.7389	0.7300	0.7304	0.992
	20	0.7554	0.7420	0.7450	0.993
	25	0.7554	0.74520	0.7450	0.993

Tabla 3 Resultados de evaluación para los datos de test para un kernel RBF.

Gamma	C	Precisión	Exhaustividad	F1-Score	Exactitud
10	2	0.6983	0.7024	0.6946	0.991
	5	0.6983	0.7024	0.6946	0.991
	10	0.6983	0.7024	0.6946	0.991
	15	**0.7072**	**0.7232**	**0.7116**	**0.9915**
	20	0.7088	0.7128	0.7035	0.9915
	25	0.7088	0.7128	0.7035	0.9915
50	2	0.6834	0.6753	0.6734	0.9895
	5	0.7053	0.6961	0.6935	0.9905
	10	0.7172	0.7170	0.7125	0.991
	15	0.7145	0.7065	0.7066	0.9905
	20	0.6920	0.6842	0.6856	0.9895
	25	0.6920	0.6842	0.6856	0.9895
100	2	0.7192	0.6983	0.7039	0.9905
	5	0.7357	0.7066	0.7182	0.991
	10	0.7127	0.6760	0.6920	0.9895
	15	0.7127	0.6760	0.6920	0.9895
	20	0.7127	0.6759	0.6919	0.989
	25	0.7223	0.6842	0.7014	0.9895

Las métricas para este modelo con un Gamma= 10 y con un C=15 tienen una mejor precisión al clasificar, tomando en cuenta que con un Gamma mayor puede entrar en problemas de sobreajuste. De los modelos generados con Máquina de Soporte Vectorial se concluye que el kernel de tipo polinomial de grado 4 con un C=5 da mejores resultados de clasificación de fallas. A continuación, se analizan los modelos con Árboles de Decisión, ver tablas 4 y 5.

Tabla 4 Evaluación para los datos de test para un árbol de decisión con criterio entropía.

Min_samples_leaf	Min_samples_split	Precisión	Exhaustividad	F1-score	Exactitud
1	**2**	**0.7687**	**0.7611**	**0.7642**	**0.992**
2	2	0.7448	0.7217	0.7293	0.9915
2	6	0.7448	0.7216	0.7293	0.991
3	6	0.7449	0.7424	0.7420	0.991
5	8	0.7502	0.7441	0.7429	0.9925
5	10	0.7502	0.7441	0.7429	0.9925

Tabla 5 Evaluación para los datos de test para un árbol de decisión con criterio GINI.

Min_samples_leaf	Min_samples_split	Precisión	Exhaustividad	F1-score	Exactitud
1	2	0.7140	0.6760	0.6920	0.988
2	2	0.7275	0.6998	0.7120	0.989
2	6	0.7203	0.7201	0.7172	0.99
3	**6**	**0.7285**	**0.7320**	**0.7270**	**0.9905**
5	8	0.7010	0.7292	0.7138	0.991
5	10	0.7010	0.7292	0.7138	0.991

En conclusión, respecto a los modelos que se analizaron de los distintos Árboles de Decisión se observó, que el mejor es empleando el criterio de entropía y en cuanto al diseño del árbol sería con un Leaf=1 y Split =2.

Otro de los modelos evaluados son el de Bagging (que el algoritmo sería RandomForest) y Boosting (que el algoritmo sería el Adaboost), ver tablas 6 y 7. Tanto AdaBoost y RandomForest, presentan buenos resultados de desempeño, aunque el que tuvo una mejor exactitud de clasificación fue RandomForest.

Tabla 6 Evaluación para los datos de test para un ensamble usando AdaBoost con un criterio de un árbol de decisión.

n-estimadores	Precisión	Exhaustividad	F1-score	Exactitud
40	**0.7905**	**0.7730**	**0.7806**	**0.9925**
60	0.7687	0.7612	0.7643	0.9925
100	0.7687	0.7613	0.7643	0.993
150	0.7727	0.7610	0.7653	0.9915
200	0.7836	0.7610	0.7706	0.9915
250	0.7629	0.7613	0.7610	0.993

Tabla 7 Evaluación para los datos de test para un ensamble usando RandomForest.

n-estimadores	Precisión	Exhaustividad	F1-score	Exactitud
40	**0.7694**	**0.7920**	**0.7790**	**0.9945**
60	0.7464	0.7697	0.7563	0.9935
100	0.7720	0.7852	0.7746	0.9935
150	0.7370	0.7614	0.7473	0.993
200	0.7401	0.7696	0.7528	0.993
250	0.7579	0.7733	0.7636	0.9935

Finalmente, se evaluaron modelos de aprendizaje profundo, ver tabla 8. Como se observa en los resultados de los diferentes modelos, un aspecto clave es que se dan mejores resultados teniendo 3 capas, de ahí que la última capa tiene 6 neuronas que se refieren a las 6 clases en cuanto a los tipos de fallas que se mencionaron previamente, que son nuestra variable objetivo.

En conclusión, cuando el total de parámetros al entrenar son menores, esto ayuda a un mejor rendimiento al entrenar la arquitectura. La mejor arquitectura posee 3 capas, con 60, 40 y 6 neuronas (funciones de activación Lineal, Tangente hiperbólica y Sigmoide, respectivamente); entrenando con 50 épocas, y un dropout en la segunda capa de 0.30.

Tabla 8 Resultados del modelo de una red neuronal multicapa para datos de test.

No de Capas	No de Neuronas	Función de Activación	Épocas	Precisión	Exhaustividad	F1-Score	Exactitud	Error
4	60	Sigmoide	50	0.6856	0.6940	0.6859	0.9900	0.0406
	40	Tangente hiperbólica						
	20	Tangente hiperbólica	100	0.7206	0.7565	0.7355	0.9920	0.0359
	6	Sigmoide						
3	60	Sigmoide	50	0.4292	0.5039	0.3995	0.9795	0.0554
	40	Tangente hiperbólica	100	0.7404	0.7586	0.7475	0.9935	0.0390
	6	Sigmoide						
3	60	Lineal	50	0.7306	0.7482	0.7374	0.9930	0.0367
	40	Tangente hiperbólica	100	0.7143	0.7244	0.7175	0.9920	0.0362
	6	Sigmoide						
3	**60**	**Lineal**	**50**	**0.7537**	**0.7920**	**0.7714**	**0.9940**	**0.0347**
	40, Dropout (0.30)	**Tangente hiperbólica**	100	0.7528	0.7795	0.7655	0.9940	0.0338
	6	**Sigmoide**						
3	60	Lineal	50	0.7382	0.7592	0.7470	0.9930	0.0364
	40	Lineal						
	6	Sigmoide	100	0.7440	0.7675	0.7550	0.9935	0.0345
3	60	Relu	50	0.6798	0.6437	0.6280	0.9865	0.0465
	40	Sigmoide	100	0.7228	0.7336	0.7233	0.9920	0.0379
	6	Softmax						
3	60	Relu	50	0.6911	0.7030	0.6947	0.9905	0.0388
	40	Tangente hiperbólica						
	6	Softmax						
4	60	Relu		0.6941	0.7003	0.6916	0.9910	0.0376
	40	Tangente hiperbólica						
	20	Tangente hiperbólica						
	6	Softmax						
4	60	Relu		0.6959	0.7009	0.6936	0.9910	0.0378
	40	Tangente hiperbólica	100	0.7126	0.7357	0.7211	0.9915	0.0371
	20	Sigmoide						
	6	Softmax						
4	60	Relu	50	0.7113	0.7211	0.7045	0.9905	0.0425
	40	Sigmoide						
	20	Sigmoide	100	0.7172	0.7440	0.7280	0.9920	0.0365
	6	Softmax						

También, se analizaron modelos con Redes Neuronales Recurrentes. En la tabla 9 se muestran las diferentes propuestas de arquitecturas con los resultados obtenidos. Por último, se probó el modelo de Redes Recurrentes LSTM, ver tabla 10. Al observar los resultados de la tabla 10, las métricas están muy bajas, apenas

con una arquitectura de 3 capas con una LSTM de 50, otra LSTM de 30 y una capa dense de 6 se obtuvo un mejor resultado.

Tabla 9 Resultados del modelo de una red neuronal recurrente para datos de test.

No de Capas	No de Neuronas	Función de Activación	Épocas	Precisión	Exhaustividad	F1-Score	Exactitud	Error
2	RNN, 50	Tangente hiperbólica	**50**	**0.7065**	**0.7336**	**0.7178**	**0.9915**	**0.0390**
	Dense, 6	Sigmoide						
	RNN,80	Tangente hiperbólica		0.6751	0.6196	0.6081	0.9860	0.0508
	Dense,6	Sigmoide	100	0.7145	0.7420	0.7170	0.9910	0.0428
	RNN,30	Tangente hiperbólica	50	0.7036	0.7238	0.7121	0.9910	0.0384
	Dense,6	Sigmoide						
	RNN,40	Tangente hiperbólica		0.6997	0.7238	0.7106	0.9910	0.0372
	Dense,6	Softmax	100	0.6777	0.6780	0.6716	0.9900	0.0421
3	RNN,50, return= true	Tangente hiperbólica	50	0.7076	0.7211	0.7029	0.9905	0.0430
	RNN,50	Tangente hiperbólica						
	Dense,6	Sigmoide	100	0.6973	0.7342	0.7144	0.9910	0.0453
	RNN,50, return= true	Tangente hiperbólica	50	0.6887	0.7024	0.6906	0.9905	0.0371
	RNN,20	Tangente hiperbólica						
	Dense,6	Sigmoide	100	0.6989	0.7253	0.7109	0.9910	0.0429
	RNN,80, return= true	Tangente hiperbólica	50	0.6775	0.6940	0.6780	0.9900	0.0443
	RNN,50	Tangente hiperbólica						
	Dense,6	Softmax						

Tabla 10 Resultados del modelo de una red recurrente LSTM para datos de test.

No de Capas	No de Neuronas	Función de activación	Épocas	Precisión	Exhaustividad	F1-Score	Exactitud	Error
2	LSTM, 50	Tangente hiperbólica	50	0.4250	0.4250	0.4013	0.9785	0.0544
	Dense, 6	Softmax						
	LSTM,30	Tangente hiperbólica		0.3968	0.4622	0.4186	0.9810	0.0533
	Dense,6	Sigmoide						
3	LSTM,50, return= true	Tangente hiperbólica		**0.6912**	**0.7357**	**0.7088**	**0.9910**	**0.0415**
	LSTM,30	Tangente hiperbólica						
	Dense,6	Sigmoide						
	LSTM,80, return= true	Tangente hiperbólica		0.6488	0.6488	0.6488	0.9895	0.0400
	LSTM,50	Tangente hiperbólica						
	Dense,6	Softmax						

En comparación con una Red Recurrente Normal se ve que es mejor en cuanto a clasificación con 2 capas, ya que con las LSTM de 3 capas empieza a mejorar un poco las métricas, pero siguen siendo bajas cuando se aumentan las capas ya que crece el número de parámetros que se van a entrenar.

Selección del modelo de aprendizaje

En conclusión, los modelos que mejores métricas generaron fueron: Árbol de Decisión, el método de ensamble (RandomForest) y la Red Neuronal Multicapa, ver, tabla 11. Se evaluó también su matriz de confusión para determinar la exactitud de la clasificación de cada falla. Por lo tanto, el mejor modelo se generó con RandomForest.

Tabla 11 Comparación de modelos.

Modelo	Precisión	Exhaustividad	F1-Score	Exactitud
RandomForest	0.7694	0.7920	0.7790	0.9945
Árbol de Decisión	0.7687	0.7611	0.7642	0.9920
Red Neuronal Multicapa	0.7228	0.7920	0.7714	0.9940

Implementación del modelo

La figura 3 muestra las variables del conjunto de datos que se obtuvieron del simulador de Factory I/O para analizar el sistema que hemos descrito anteriormente y ver qué relación tienen estas variables cuando ocurre una falla3.

	EMITTER	RPM BELT CONVEYOR	AT SCALE ENTRY	RPM FRONT CONVEYOR	RPM LEFT CONVEYOR	RPM RIGHT CONVEYOR	Weight/PESO	AT SCALE	LOAD SCALE	AT LEFT ENTRY	AT RIGHT ENTRY	AT FORWARD ENTRY	AT EXIT FRONT	AT EXIT LEFT	AT EXIT RIGHT	TIEMPO	TIPO DE FALLA
0	False	0	True	0	0	0	0.000000	False	False	False	False	False	True	True	True	0.000000	SIN-FALLA
1	False	0	True	0	0	0	0.000000	False	False	False	False	False	True	True	True	1.541667	SIN-FALLA
2	False	0	True	0	0	0	0.000000	False	False	False	False	False	True	True	True	3.058333	SIN-FALLA
3	False	0	True	0	0	0	0.000000	False	False	False	False	False	True	True	True	4.533332	SIN-FALLA
4	False	0	True	0	0	0	0.000000	False	False	False	False	False	True	True	True	6.041662	SIN-FALLA
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2733	False	10	True	0	0	0	0.153538	False	True	False	True	True	True	True	True	2464.344000	FALLA_BELTCONV
2734	False	10	True	0	0	2	0.007185	False	False	False	False	True	True	True	True	2465.839000	FALLA_BELTCONV
2735	False	10	True	0	0	2	0.000000	False	False	False	False	True	True	True	True	2467.343000	FALLA_BELTCONV
2736	False	10	True	0	0	2	0.000000	False	False	False	False	True	True	True	True	2468.863000	FALLA_BELTCONV
2737	False	10	True	0	0	2	0.000000	False	False	False	False	True	True	True	True	2470.334000	FALLA_BELTCONV

2738 rows × 17 columns

Figura 3 Conjunto de Datos del Sistema Clasificador de Cajas.

Las fallas intermitentes que inducimos al sensor de peso y a los motores de las bandas transportadoras (vayan a su máximo tope) ocurren bajo ciertas condiciones, al momento de correr el simulador y en ciertos instantes de tiempo se indujeron las fallas en esos elementos por lapsos de tiempo indefiniendo. Por lo que una de las variables a considerar es el tiempo, se estableció que cada 1.5 segundos guardara un nuevo dato.

El conjunto de datos obtenido tiene 2738 registros y 17 columnas que son: Emitter, RPM belt conveyor, At scale entry, RPM left conveyor, RPM front conveyor, RPM right conveyor, Peso, At scale, Load scale, At left entry, At right entry, At forward entry, At exit left, At exit front, At exit right, Tiempo, Tipo de falla. Se analizó el conjunto de datos para identificar los tipos de fallas, se observó que 1938 son datos sin falla y el resto son datos con fallas que están divididos en 5 tipos de fallas: falla en RPM belt conveyor (341), falla en RPM front conveyor (21), falla en RPM left conveyor (2), falla en RPM right conveyor (17) y falla en el sensor de peso (419). Inicialmente, se realizó el preprocesamiento de los datos para entrenar los modelos, tal como se describió en el caso de estudio, donde antes de entrenar cada modelo, los datos se separaron en entrenamiento y prueba usando el 80% de los datos para entrenar y el 20% de prueba. Después, se entrenó el modelo de Red Neuronal Multicapa tal como se muestra en la tabla 8. Además, los datos también se entrenaron con los modelos de Árboles de Decisión y RandomForest como en los descritos en la tabla 11. Las métricas de los modelos obtenidos se presentan en la tabla 12.

Tabla 12 Comparación de resultados con el conjunto de datos de nuestro sistema.

Modelo	Precisión	Exhaustividad	F1-Score	Exactitud
Red Neuronal Multicapa	0.9757	0.9275	0.9437	0.9252
Árbol de Decisión	0.9987	0.9943	0.9965	0.9945
RandomForest	0.9968	0.9939	0.9953	0.9927

Como se observa, el modelo de Red Neuronal Multicapa que se eligió para validar el proceso de clasificación con los datos del sistema, da buenas métricas, sin embargo, al compararlo con los otros dos modelos, se observa que las métricas aumentan más en particular con Árboles de Decisión, lo cual indica que es el mejor

modelo.También, se analizaron las matrices de confusión, se observó que el mejor clasificador es efectivamente el Árbol de Decisión.

Analizando con más profundidad las variables que estamos manejando, se concluyó que hay variables que no son relevantes al momento de clasificar los tipos de fallas, ya que se tomó en cuenta el P-valor para poder eliminar esas variables. Se optó por este procedimiento ya que se observaron que las Redes Neuronales tenían bajas métricas, en vista de ello se aplicó el P-valor para observar si eliminando ciertas variables se mejoraba el resultado. Por lo que se obtuvo la información de las variables (OLS) y aplicando el criterio de un $P > 0.05$, si la variable sobrepasaba ese valor se rechaza. Si $P < 0.05$ se acepta. Por lo tanto se eliminaron del conjunto de datos las variables: At exit right-true, At exit right-false, At exit left-true, At exit left-false, At exit front-true, At exit front-false, At forward entry-true, At forward entry-false, At right entry-true, At right entry-false, At left entry-true, At left entry-false, Load scale-true, Load scale-false, At scale-true, At scale-false, At scale entry-true, At scale entry-false, emitter-true, emitter-false, RPM belt conveyor. Con el nuevo conjunto de variables que quedaron se volvieron a entrenar los modelos y se observó que cambio surgía, tabla 13.

Tabla 13 Comparación de modelos eliminando variables que no son relevantes.

Modelo	Precisión	Exhaustividad	F1-Score	Exactitud
Red Neuronal Multicapa	0.9531	0.9099	0.9237	0.8996
Árbol de Decisión	1.0	1.0	1.0	1.0
RandomForest	1.0	1.0	1.0	1.0

Los resultados obtenidos hasta este momento prueban que los algoritmos que mejor resultados tienen en cuanto a clasificación de múltiple-clase de fallas son: Árboles de Decisión, RandomForest y Redes Neuronales Multicapa.También, a partir del sistema automático de manufactura en el software Factory I/O, se usó la información para hacer el diagnóstico de detección de fallas intermitentes que se indujeron al sistema. Empleando así los modelos que resultaron mejor en cuanto a clasificación múltiple-clase y se observó que para este problema los mejores clasificadores son: Árboles de Decisión y RandomForest.

4. Discusión

Es importante resaltar que con los resultados que obtuvo [Zúñiga, 2021], estamos comprobando que efectivamente en cuestiones de ambientes simulados el mejor clasificador de fallas es el árbol de decisión, a pesar de que Zúñiga solo maneja dos tipos de fallas (si existe o no falla). Pero en este artículo se extendieron a múltiples fallas para poder detectar y clasificar correctamente las fallas intermitentes en bandas transportadoras y en un sensor de peso. También, cabe resaltar que en el caso de [Contreras, 2020] se limitaron a probar solamente los algoritmos de entrenamiento de: K-vecinos más cercanos, Máquinas de Soporte Vectorial y Árboles de Decisión. Por lo que en este trabajo se amplió a una mayor cantidad de algoritmos de clasificación no solamente los de aprendizaje máquina sino también probar con aprendizaje profundo y sus variaciones de redes neuronales que existente hasta ahora. Todo esto con el fin de poder tener una comparación más precisa en cuanto a algoritmos de clasificación, enfocado a múltiples-clases.

5. Conclusión

Se comparó la eficiencia de los modelos de clasificación como: Máquina de Soporte Vectorial, Árboles de Decisión, Métodos de ensamblaje (Adaboost y RandomForest) y Redes Neuronales Artificiales (con sus variaciones) en fallas intermitentes en sensores y motores en un proceso de manufactura virtual creado en Factory I/O. Se observó que los modelos más eficientes en clasificar múltiples fallas fueron Árboles de Decisión y RandomForest.

Como trabajo a futuro se contempla generar más datos para validar los modelos y automatizar el proceso de clasificación de fallas y con ello llevar a cabo el diagnóstico en tiempo real y considerar predecir cuándo ocurrirá la falla y tener un diagnóstico más preciso de sistemas de manufactura.

6. Bibliografía y Referencias

[1] Contreras, J.L. Diseño de un modelo para mantenimiento predictivo en motores de inducción utilizando técnicas de la industria 4.0, Repositorio de Universidad Tecnológica del Perú. Lima, Perú. 2020.

[2] Guamán, A. Desarrollo de un modelo basado en datos a partir de señales de vibración para la detección de fallos en un compresor reciprocante de simple efecto doble etapa. Repositorio de la Universidad Nacional Mayor de San Marcos, Universidad del Perú, Decana de América. Lima, Perú. 2019.

[3] Inafuku, A.H. Diseño e Implementación de un Sistema de Diagnóstico de Fallas para la Inspección y Detección de Fallas en Componentes de Procesos Industriales utilizando un Robot Móvil y Algoritmos de Inteligencia Artificial. Repositorio de la Pontifica Universidad Católica del Perú. Perú. Septiembre, 2020.

[4] Nexus, I. (2022). 10 beneficios de contar con un sistema de automatización industrial. Mayo 14,2022, de nexus integra Sitio web: https://nexusintegra.io/es/10-beneficios-de-contar-con-un-sistema-de-automatizacion-industrial/.

[5] Pat, C. (2022). Automatización de procesos: 5 principales beneficios en empresas. Mayo 14, 2022: https://www.helpsystems.com/es/recursos/guias/automatizacion-de-procesos-5-principales-beneficios-en-empresas.

[6] Prieto, J. Detección de fallas en tiempo real mediante redes complejas en un sistema de manufactura 4.0. Pistas Educativas, Vol. 42, Núm. 136. México. Julio, 2020.

[7] Saiz, L. Fallos intermitentes: análisis de causas y efectos, nuevos modelos de fallos y técnicas de mitigación. Repositorio de la Universidad Politécnica de Valencia. Valencia. 2015.

[8] Vidal, E. (2018). 10 maneras en las que el monitoreo en tiempo real de la fabricación mejora la precisión y la calidad. Mayo 14, 2022, de Tecnología para la Industria. Sitio web: https://tecnologiaparalaindustria.com/10-maneras-en-las-que-el-monitoreo-en-tiempo-real-de-la-fabricacion-mejora-la-precision-y-la-calidad/.

[9] Zúñiga, L.M. Desarrollo de un modelo predictivo para un sistema de manufactura orientado a la industria 4.0. Repositorio de Pontificia Universidad Javeriana de Colombia. Bogotá, Colombia. 2021.