ISSN 1870-4069

Método de recomendación para pruebas eléctricas fallidas en la industria de manufactura aplicando aprendizaje automático

Maximiliano Ponce Marquez[1], Samuel González-López[1], Aurelio López-López[2], Guillermina Muñoz-Zamora[1]

[1] Instituto Tecnológico de Nogales, División de Estudios de Posgrado e Investigación, México

[2] Instituto Nacional de Astrofísica, Óptica y Electrónica, Coordinación de Ciencias Computacionales, México

{samuel.gl,M22340764,guilermina.mz}@nogales.tecnm.mx, allopez@inaoep.mx

Resumen. La aplicación de la inteligencia artificial en la industria manufacturera tiene grandes oportunidades. Este artículo presenta la comparación entre diferentes técnicas de aprendizaje automático y aprendizaje profundo para recomendar una posible causa raíz de una falla en un módulo electrónico para vehículos. Se experimentó bajo tres arquitecturas: la plataforma de aprendizaje automático de Microsoft, una red neuronal con Tensorflow y un modelo de clasificación con modelos preentrenados BERT-multilenguaje. Los modelos presentados sugieren una recomendación para el técnico de análisis y no una respuesta absoluta, pero puede ser de ayuda al personal inexperto o en entrenamiento. La herramienta de Microsoft obtuvo una precisión de 51.24 % con el algoritmo de LbfgsMaximumEntropyMulti, la red neuronal alcanzó una precisión del 43.13 %, mientras que el escenario de clasificación multiclase alcanzó un 65.40 %. La selección de estas 3 herramientas se basó en la utilidad y el tiempo de implementación, la herramienta de Microsoft es muy sencilla de utilizar a diferencia de la red neuronal artificial y el uso del modelo BERT. Sin embargo, la herramienta de Microsoft no se puede modificar a diferencia de las otras dos alternativas.

Palabras clave: Método de recomendación, aprendizaje automático, industria manufaturera, detección de causa de falla.

Recommendation Method for Failed Electrical Testing in the Manufacturing Industry Using Machine Learning

Abstract. The application of artificial intelligence in the manufacturing industry has great opportunities. This paper compares different machine learning and deep learning techniques to recommend a possible root cause of a failure in a

vehicle electronics module. It was experimented under three architectures: the Microsoft machine learning platform, a neural network with Tensorflow, and a classification model with pre-trained BERT-multilanguage models. The models presented to suggest a recommendation for the analysis of a technician and not an absolute answer, but they may be helpful to inexperienced personnel or trainees. The Microsoft tool achieved a precision of 51.24% with the LbfgsMaximumEntropyMulti algorithm, the neural network achieved a precision of 43.13%, while the multiclass classification scenario achieved 65.40%. The current selection of these three methods was based on the utility and time implementation, for example, the Microsoft AI wizard is easy to use compared to the artificial neural network programmed with Tensor Flow and the BERT model. However you can not modify the models utilized in the wizard unlike the other two alternatives.

Keywords: Recommendation method, machine learning, manufacturing industry, failure cause detection.

1. Introducción

Dentro de la industria manufacturera de partes para automóviles existen diferentes áreas encargadas de revisar las piezas producidas. Específicamente en el área de Análisis, de una empresa de partes de autos de la ciudad de Nogales Sonora, se encargan de revisar las piezas que fallaron en pruebas eléctricas en el piso de producción. Dado el fallo, el analista revisa el historial de la unidad y con base en la falla, dicho operario es capaz de determinar si es una falla falsa o si es una falla real. En este último caso, es necesario determinar la causa de la falla.

Generalmente en el caso de las fallas reales, se busca el componente en específico que causó la falla, para ello el analista debe usar sus capacidades técnicas y analíticas para determinar la causa de la falla. El analista cuenta con diversas herramientas como multímetro y diagramas eléctricos interactivos para encontrar la falla fácilmente.

Sin embargo, existen fallas que son muy difíciles de identificar y que en ocasiones causan mayores perdidas debido a que pueden salir unidades con la misma falla consecutivamente, es por ello la importancia de poder determinar la falla rápidamente.

Nuestro trabajo se enfocará principalmente en la herramienta de aprendizaje automático de Microsoft, el desarrollo de una red neuronal utilizando Tensorflow con Python en un entorno local y el ajuste con nuestros datos de un modelo pre-entrenado llamado BERT (Representación de Codificador Bidireccional de Transformadores) en su versión multilingual.

Se realiza la comparación entre las tres arquitecturas mencionadas haciendo énfasis en la la métrica Precisión. Cabe mencionar que los datos fuente tienen una variedad alta y desbalanceada de categorías.

2. Trabajo relacionado

El campo de la inteligencia artificial ha tenido grandes avances en los últimos años, en parte por las mejoras en el hardware y los desarrollos e investigaciones en nuevos

algoritmos de inteligencia artificial. En la actualidad se utiliza la inteligencia artificial para la mejora de procesos industriales.

Algunas de las aplicaciones son la optimización de la eficiencia general de los equipos (OEE por sus siglas en inglés) a través de la reparación y mantenimiento predictivo, además de implementaciones en el área de calidad y robótica [8].

Otra de las aplicaciones de la inteligencia artificial en la industria es la reducción de costos, un ejemplo es la implementación de un sistema inteligente que utiliza un sistema de visión e inteligencia artificial para optimizar los cortes en tuberías de metal, en el cual se utilizaron la técnica de análisis de componentes principales (PCA, un controlador PID que interactúa con un modelo de inteligencia artificial basado en máquinas de vectores de soporte (SVM) [7].

En [3] se habla de la posibilidad de poder utilizar los datos generados por una red de sensores para diseñar un modelo predictivo. En [4] los autores desarrollaron un modelo en un ambiente industrial real, el cual consistió en un modelo basado en el algoritmo de aprendizaje Random Forest.

La complejidad de los sistemas de manufactura y la gran oportunidad que la inteligencia artificial ofrece para conseguir productos de mayor calidad es uno de los retos en la actualidad. La obtención de los datos relevantes para entrenar modelos, en algunas empresas, requiere de un preprocesamiento ya que no están estructurados para poder ser utilizados en un modelo de inteligencia artificial directamente [5]. Así el primer paso es el filtrado de la información, es decir obtener la información relevante para la tarea a resolver.

Hay empresas que por peticiones del cliente o por reglas normativas tienen que utilizar o implementar un sistema de trazabilidad de sus productos, para poder detectar la causa raíz de problemas. Estos sistemas ofrecen una gran cantidad de información que puede ser utilizada para construir modelos de predicción.

El trabajo realizado en [6] propone un marco de trabajo de inteligencia artificial utilizando redes bayesianas, tomando el conocimiento de expertos para encontrar desviaciones de calidad y facilitar el análisis de la causa raíz, y se demuestra cómo afecta el conocimiento de expertos en el modelo.

En el trabajo [10] desarrollaron un análisis factorial para determinar la ventana de predicción, que es el tiempo de antelación para prever un fallo. Los autores utilizaron tres algoritmos ML para evaluar la ventana de predicción: bosque aleatorio, máquinas de soporte vectorial y regresión logística.

La falta de datos es un reto en el mantenimiento predictivo, en [9] los autores desarrollaron un método de entrenamiento para transferir el conocimiento aprendido de una falla en otra. En nuestro trabajo, el reto es la gran variedad de los datos que se genera en la industria automotriz.

3. Antecedentes

Anteriormente, dentro de la empresa que se analiza en este trabajo, se desarrolló un sistema web para registrar las fallas que el equipo de análisis encuentra diariamente, para que así posteriormente el analista pueda hacer una búsqueda en el historial de qué componentes han causado cierta falla en específico.

Tabla 1. Datos antes y después del filtrado.

Descripción	Cantidad de datos
Data original	59574
Data filtrada	14311

El desarrollo de este sistema ofreció buenos resultados ya que el personal de reciente ingreso o que no está relacionado con ciertos productos puede buscar fallas fácilmente en el sistema y dar una pista del componente que puede ser la causa de la falla.

No obstante, este sistema aun tenía más oportunidades, con el paso del tiempo la base de datos de esta información fue creciendo, abriendo la posibilidad de crear un modelo para poder recomendar el componente que ocasionó la falla.

4. Propuesta

4.1. Procesamiento de los datos

El primer paso fue el pre-procesamiento de los datos antes de entrenar el modelo. Inicialmente existían 59,754 entradas en la base de datos, pero no todos estos datos son válidos para el modelo. En particular, las pruebas relacionadas con la soldadura de las unidades resultaron inútiles, ya que no proveen información de valor para poder determinar la causa raíz de la falla.

Adicionalmente hay pruebas de unos equipos en especial, llamados ICT (In Circuit Test, por sus siglas en ingles) que ya dan directamente la respuesta, por lo que ya no es necesario predecir estas fallas. También se extrajeron solo aquellos resultados en los que se registró un solo componente, esto debido a que es mas sencillo poder categorizar esta información y acelerar el procesamiento para el modelo.

El dataset cuenta con columnas que ofrecen información del modelo, familia del producto, número de serie, estación en la que se probó la unidad, descripción de la falla, valor de la prueba, componente que ocasiono la falla y un comentario que el analista agrega de manera opcional.

Para este trabajo se decidió trabajar con las columnas de descripción de la prueba eléctrica, valor numérico de la prueba, causa de la falla o componente y comentario, ya que con estos datos es suficiente para poder ofrecer un resultado.

Cada producto tiene su propio grupo de pruebas, por lo que no hay problema al omitir la columna del modelo en este trabajo. Con estos datos se plantearon tres escenarios de experimentación:

– En el primer escenario se utilizaron las columnas TestDescription, Value como entradas y Component como salida. Para esto se implementó la herramienta de Aprendizaje Automático de Microsoft.
– Posteriormente se realizó con la misma configuración del escenario uno, la utilización de una red neuronal con TensorFlow.
– Con la finalidad de obtener una alternativa de recomendación, se configuró el tercer escenario bajo un esquema de agrupamiento, donde se identificaron aquellos comentarios que tuvieran similitud. Por ejemplo: 'Arrancado pad dañado' y 'c1418 capacitor dañado' se encuentran en el mismo grupo o categoría.

Tabla 2. Ejemplo del conjunto de datos pre-procesado.

TestDescription	Value	Component	Comment
504090_AI_P0 Pull up FB	3.828	IC101	IC101 Corte por soldadura
504090_AI_P0 Pull up FB	3.94	IC101	IC101 Corte por soldadura
504090_AI_P0 Pull up FB	3.94	IC101	IC101 Corte por soldadura
503020_LIN Voltage	3.647	D401	D401 Danado
502020_PCHK X400_3 LED_CATHODE	36.34	D401	D401 Danado
509050_BST CHK Phase 3	36	R411	R411 Contaminado insuficiencia

Asi obtuvimos 10 categorías o grupos. Posteriormente se ajustó un modelo de BERT. En este escenario se busca predecir la categoría dependiendo de la prueba realizada. Así, el operador puede revisar la categoría predecida para ver posible causa-raíz.

4.2. Construcción del modelo utilizando herramienta de aprendizaje automático de Microsoft

Una vez obtenido un conjunto de datos limpio y listo, se optó primeramente por utilizar la herramienta de Microsoft disponible en Visual Studio 2022. Esta herramienta permite seleccionar un caso de uso, un dataset y en base a ello construir un modelo de inteligencia artificial.

Como se mencionó anteriormente se utilizaron las columnas de descripción de la falla y el resultado de la prueba como entrada para el modelo y la columna de componente como salida.

La herramienta de Microsoft se encarga por completo de construir el modelo, en el que va comparando diferentes modelos de aprendizaje de máquina preestablecidos y al final seleccionar el que tuvo el mejor porcentaje de aciertos.

En este proyecto nos enfocamos en la métrica de precisión ya que el modelo predice los resultados en base al historial registrado en la base de datos de análisis, y resulta imposible poder predecir una falla que nunca se ha registrado en la base de datos ya que es un proceso muy complejo que requeriría de muchísima mas información, como la construcción de la unidad, los circuitos eléctricos, interconexiones, envío de mensajes, etc. Además, son muchos productos los que son analizados.

Resultados del modelo de Microsoft. Se realizaron 13 iteraciones con diferentes modelos, obteniendo los siguientes resultados:

De las 13 iteraciones, estas son las 5 con los mejores resultados:

El mejor modelo que se pudo entrenar fue el de la iteración 11 con una precisión del 51.24 %.

Tabla 3. Resultados del modelo de Microsoft - 13 iteraciones.

Modelo	Precisión	Tiempo de entrenamiento (s)	#Iteración
SdcaMaximumEntropyMulti	0.3190	12,1	0
SdcaLogisticRegressionOva	0.0705	130.8	1
FastForestOva	0.4614	142.5	2
LightGbmMulti	0.36	4.6	3
LbfgsMaximumEntropyMulti	0.4936	159.8	4
FastForestOva	0.4674	151.6	5
FastTreeOva	0.4835	211.8	6
SdcaMaximumEntropyMulti	0.3559	11.4	7
LbfgsLogisticRegressionOva	0.4936	54.9	8
SdcaLogisticRegressionOva	0.3418	123.7	9
LightGbmMulti	0.0981	5.2	10
LbfgsMaximumEntropyMulti	0.5124	320.1	11
FastForestOva	0.4527	179	12

Tabla 4. 5 Mejores resultados del modelo de Microsoft.

Modelo	Precisión	Tiempo de entrenamiento (s)	#Iteración
LbfgsMaximumEntropyMulti	0.5124	320.1	11
LbfgsLogisticRegressionOva	0.4936	54.9	8
LbfgsMaximumEntropyMulti	0.4936	159.8	4
FastTreeOva	0.4835	211.8	6
FastForestOva	0.4674	151.6	5

Cabe mencionar que en este primer escenario la cantidad de componentes que pueden ser la causa-raíz son muchos, por que creemos que ese comportamiento nos produce un resultado relativamente bajo.

4.3. Desarrollo de red neuronal con TensorFlow

También se optó por implementar una red neuronal utilizando la librería Tensorflow en Python. Para empezar se importaron las siguientes librerías:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import tensorflow as tf
```

Después se importó el conjunto de datos de la siguiente manera:

```
dataset = pd.read_csv('filtered-dataset.csv')
X = dataset.iloc[:,:-1].values
y = dataset.iloc[:,-1].values
```

Importar el conjunto de datos no es suficiente, en el caso de la herramienta de Microsoft lo hacia por nosotros, pero ahora sera necesario categorizar la informacion de forma manual. Se utilizó la libreria de Scikit learn para esta tarea con las siguientes instrucciones:

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder

ctPruebas = ColumnTransformer(transformers=
  [('encoder', OneHotEncoder(sparse=False),[0])],remainder='passthrough')
X = np.array(ctPruebas.fit_transform(X))

ctUnidadesMedida = ColumnTransformer(
  transformers=[('encoder',
      OneHotEncoder(sparse=False),
      [-2])],remainder='passthrough')
X = np.array(ctUnidadesMedida.fit_transform(X))

ctPruebas = ColumnTransformer(transformers=[('encoder',
OneHotEncoder(sparse=False), [0])],remainder='passthrough')
wy = np.array(ctPruebas.fit_transform(y.reshape(-1,1)))
```

Estas instrucciones convierten las columnas categorías en una colección de 1s y 0s, por ejemplo, la prueba '503010 Current Consumption503010 Current Consumption' pudo ser transformada a un arreglo como [0,0,0,1,0,1]. En este caso, dado que hay 1853 fallas y 521 componentes posibles, los arreglos serán de los tamaños correspondientes.

Esto se puede comprobar con el código siguiente:

```
print(X[0].shape)
print(y[0].shape)
```

Después se divide el conjunto de datos en un conjunto de entrenamiento y un conjunto de pruebas, con una proporción de 80/20.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
```

Posteriormente se hace una normalización de los datos, especialmente para los valores de las pruebas, para que todos estén en el rango de 0 a 1

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Ahora que el conjunto de datos está listo, se comienza a construir la arquitectura de la red neuronal, en este caso se utilizó la siguiente selección de capas en la red neuronal artificial. A continuación, se presenta el código para construir la red neuronal.

```
ann = tf.keras.models.Sequential()
ann.add(tf.keras.layers.Dense(units = 1853, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 2000, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 2000, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 3000, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 3500, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 3500, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 2000, activation='relu'))
```

```
ann.add(tf.keras.layers.Dense(units = 2000, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 1500, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 1000, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 1000, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 800, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 800, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 700, activation='relu'))
ann.add(tf.keras.layers.Dense(units = 521, activation='softmax'))
```

Observe que se utilizó la función de activación ReLU para todas las capas con excepción de la capa de salida en la que se optó por softmax, esto a que es mas adecuado para datos categóricos como en este caso.

Finalmente se entrenó el modelo con los siguientes parámetros:

```
ann.compile(optimizer = 'adam', loss = 'categorical_crossentropy',
metrics = ['accuracy'])
ann.fit(X_train, y_train, batch_size = 32, epochs = 500)
```

El modelo descrito obtuvo una precisión de 43.13 %, un poco menor a los resultados obtenidos con la herramienta de Microsoft.

4.4. Implementación del modelo BERT-multilingual

A continuación mostramos algunas secciones de la codificación para lograr la tarea de clasificación multiclase.

```
import tensorflow as tf
print(tf.__version__)
import pandas as pd
df = pd.read_excel('data.xlsx')
df.head()
```

En la siguiente figura se muestra la distribución de las 10 categorías. Se puede observar que las categorías 3 y 6 tienen mayor cantidad de ejemplos (ver Fig. 1).

A continuación se muestra la sección del código para calcular las métricas (en este caso Precisión y Exactitud):

```
n_epochs = 10
METRICS = [
      tf.keras.metrics.CategoricalAccuracy(name="Acccuracy"),
      tf.keras.metrics.Precision( name="Precision"),
      balanced_recall,
      balanced_precision,
      balanced_f1_score
]
earlystop_callback = tf.keras.callbacks.EarlyStopping(monitor =
"val_loss", patience = 3, restore_best_weights = True)
model.compile(optimizer = "adam",
              loss = "categorical_crossentropy",
              metrics = METRICS)
model_fit = model.fit(x_train,
                      y_train,
                      epochs = n_epochs,
```

```
                        validation_data = (x_test, y_test),
                        callbacks = [earlystop_callback])
```

Los resultados en 10 épocas lo podemos encontrar en la siguiente tabla:
El valor mas alto para la precision se alcanza en el época 9 con un valor de 0.6540.

5. Resultados

Generalmente se utiliza la métrica de recuerdo o recall cuando queremos que el modelo detecte tantos casos reales como sea posible, y se utiliza la métrica de precisión o exactitud (accuracy) cuando necesitamos que el modelo sea lo mas correcto posible. En este caso debido a que se busca un modelo de recomendación y que está basado totalmente en el historial generado por el personal, se busca obtener la mejor precisión posible. Recordemos que en este modelo, predecir una falla que no está registrada en el conjunto de datos de entrenamiento no ofrecerá una respuesta válida ya que no hay forma de relacionar la respuesta.

La herramienta de Microsoft obtuvo una precisión de 51.24 % con el algoritmo de LbfgsMaximumEntropyMulti, la red neuronal personalizada una precisión del 43.13 %, mientras que el escenario de clasificacion multiclase alcanzó un 65.40 %.

Los otros algoritmos utilizados por la herramienta de Microsoft arrojaron unos resultados muy similares, sin embargo, su desventaja es que no permite realizar muchas configuraciones. La herramienta automáticamente va actualizando los parámetros y seleccionando otros algoritmos de acuerdo con los resultados y después de un determinado tiempo selecciona el que condujo al mejor resultado.

El otro enfoque, utilizando una red neuronal codificada desde 0, dió un resultado menor, pero permite ser configurado completamente, pudiendo cambiar la forma en que se preprocesan los datos e incluso cambiar la arquitectura de red.

Finalmente creemos que el escenario tres muestra lo que pudiera ser una solución, siempre y cuando se realice un filtrado en diferentes categorías por un experto. Tambien es necesario buscar el balanceo de las clases, algo complicado en la industria donde hay datos irregulares.

Los resultados parecen ser no muy favorecedores, pero aún hay oportunidades de mejora. Es posible enfocarse en un solo producto y utilizar más datos del sistema de producción. Sin embargo, con estos resultados el analista puede obtener una recomendación que le podría ayudar en su proceso de búsqueda de fallas.

6. Conclusiones

En otros trabajos relacionados con inteligencia artificial en la manufactura se obtiene resultados mucho mayores como el 99.95 % de precisión obtenido en el sistema de cortes para minimizar costos [7], sin embargo, la naturaleza de los sistemas es distinto.

Se decidio utilizar estas tres herramientas de manera exploratoria, principalmente por el tiempo que se puede utilizar para construir y entrenar el modelo.

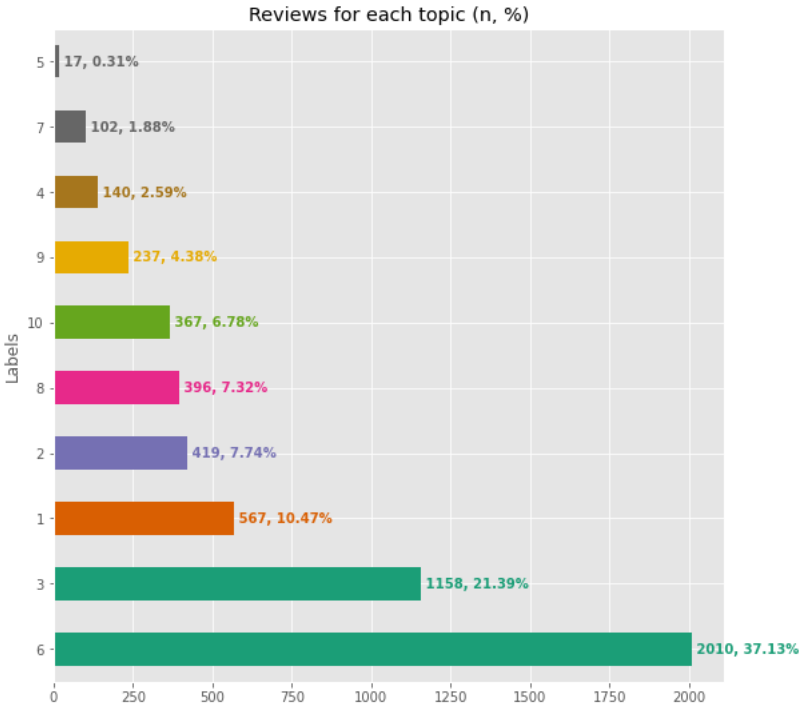

Fig. 1. Distribución de las clases.

Tabla 5. 10 épocas del modelo BERT-multilingual.

Época	Precisión	Exactitud
1	0.5140	0.3851
2	0.5856	0.4437
3	0.5986	0.4708
4	0.6215	0.4846
5	0.6208	0.4900
6	0.6276	0.5068
7	0.6354	0.5060
8	0.6407	0.5159
9	0.6540	0.5275
10	0.6442	0.5150

Recordemos que con forme pasan los días se van registrando nuevas fallas o componentes generadores de fallas, por lo que es necesario entrenar nuevamente el modelo de nuevo para que sea capaz de predecir o dar resultados en base a estas nuevas fallas.

En el caso del problema presentado en este trabajo se basa completamente en el historial generado por el personal de análisis, además, existe una gran variedad de fallas posibles para una gran diversidad de productos o modelos diferentes.

Los resultados obtenidos por los tres modelos reportados demuestran la complejidad del problema, sin embargo, es el inicio para explorar otras alternativas.

Es posible, por ejemplo, implementar un sistema mas robusto para obtener la información y probar otros algoritmos de aprendizaje. En base a los resultados se puede observar que en algunas ocasiones las soluciones mas sencillas son las mas adecuadas. Sin embargo, es importante realizar la exploración y experimentación.

En este caso un algoritmo de regresión logística multinomial ofreció un mejor resultado que una red neuronal programada con Tensor Flow. Se deben atacar las problemas con las herramientas adecuadas, pero para ello es necesario conocer el repertorio de herramientas que hay disponibles.

Agradecimientos. El primer autor es apoyado por Conacyt, becario 1244665.

Referencias

1. Ponce-Cruz, P.: Inteligencia artificial: Con aplicaciones a la ingeniería. Alpha Editorial (2010)
2. Ertel, W.: Introduction to artificial intelligence. Undergraduate Topics in Computer Science, Springer International Publishing (2017)
3. Sittón, I., Hernandez, E., Rodríguez, S., Santos, M. T., González, A.: Machine learning predictive model for industry 4.0. Knowledge Management in Organizations (KMO 2018), Communications in Computer and Information Science, vol. 877 (2018) doi: 10.1007/978-3-319-95204-8_42
4. Paolanti, M., Romeo, L., Felicetti, A., Mancini, A., Frontoni, E., Loncarski, J.: Machine learning approach for predictive maintenance in industry 4.0. In: 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), pp. 1–6 (2018) doi: 10.1109/MESA.2018.8449150
5. Wuest, T., Weimer, D., Irgens, C., Thoben, K. D.: Machine learning in manufacturing: Advantages, challenges, and applications. Production and Manufacturing Research, vol.4, no. 1, pp. 23–45 (2016) doi: 10.1080/21693277.2016.1192517
6. Lokrantz, A., Gustavsson, E., Jirstrand, M.: Root cause analysis of failures and quality deviations in manufacturing using machine learning. Procedia CIRP, vol. 72, pp. 1057–1062 (2018) doi: 10.1016/j.procir.2018.03.229
7. Acosta, P., Terán, H., Arteaga, O., Terán, M.: Machine learning in intelligent manufacturing system for optimization of production costs and overall effectiveness of equipment in fabrication models. Journal of Physics: Conference Series, vol. 1432 (2020) doi: 10.1088/1742-6596/1432/1/012085
8. Ng-Corrales, L. C., Lambán, M. P., Hernández-Korner, M. E., Royo, J.: Overall equipment effectiveness: Systematic literature review and overview of different approaches. Applied Sciences, vol. 10, no. 18, pp. 6469 (2020) doi: 10.3390/app10186469
9. Li, Z., Kristoffersen, E., Li, J.: Deep transfer learning for failure prediction across failure types. Computers and Industrial Engineering, vol. 172 (2022) doi: 10.1016/j.cie.2022.108521
10. Leukel, J., González, J., Riekert, M.: Machine learning-based failure prediction in industrial maintenance: Improving performance by sliding window selection. The International Journal of Quality and Reliability Management, vol. 40, no. 6 (2022) doi: 10.1108/IJQRM-12-2021-0439