Alarmas Para Evitar Fallas en Motores Eléctricos Mediante Mantenimiento Predictivo

Gabriel Rodríguez   
*Facultad de Ingeniería Eléctrica*  
*Universidad Tecnológica de Panamá*Ciudad de Panamá, Panamá  
gabriel.rodriguez10@utp.ac.pa

Juan Sánchez  
*Facultad de Ingeniería Eléctrica*  
*Universidad Tecnológica de Panamá*Ciudad de Panamá, Panamá  
juan.sanchez21@utp.ac.pa

*Abstract*—Los motores eléctricos son máquinas encargadas de transformar energía eléctrica en energía mecánica. En la realización de esta labor, los motores eléctricos sufren desgaste debido a su forma de operación, lo cual puede llevar a cabo una falla. Debido a esto, es sumamente importante ser capaz de asegurar un funcionamiento óptimo de las mismas, especialmente cuando se trabajan con cargas de gran tamaño. Para evitar eventos no deseados, se busca identificar el momento en el que ocurra la falla para ser capaz de realizar un mantenimiento predictivo que prevenga esta situación. Esta predicción se puede realizar mediante Aprendizaje Automatizado, o Machine Learning, y en este escrito se va a realizar una evaluación experimental de diferentes técnicas de Machine Learning para identificar fallas en motores eléctricos bajo diferentes condiciones.

Palabras clave—Diagnóstico de fallas, KNN, Motores eléctricos, Machine Learning, Random Forest, Tree Classifier

# Introducción

Existen diversos tipos de máquinas eléctricas que se emplean en sistemas eléctricos para generación y uso de la energía eléctrica. Estos sistemas poseen diversos componentes esenciales que permiten estas funciones. En el caso de los motores eléctricos, donde se convierte la energía eléctrica en energía mecánica mediante campos magnéticos, existen dos partes principales conocidas como el rotor y el estator [1]. Para el cumplimiento de sus funciones, se deben presentar las condiciones óptimas para un funcionamiento normal, pero el uso continuo puede llegar a producir desgastes o fallas, las cuales se buscan evitar, especialmente cuando se trata de motores de mayor capacidad. Las fallas en estos equipos pueden ser catastróficas, y se debe buscar la manera de evitar estos posibles accidentes. Esto se puede llevar a cabo mediante un mantenimiento predictivo, que es capaz de reconocer el momento adecuado en el que se debe realizar dicho mantenimiento para evitar que se produzcan estas fallas en los distintos componentes de los motores eléctricos [2]. Para comprender adecuadamente cómo realizar dichos mantenimientos predictivos, se utilizarán modelos de Machine Learning [3] [4], que serán capaces de analizar la información pertinente, y proporcionarán un resultado que permita comprender la necesidad de realizar mantenimientos en un motor eléctrico.

En un motor eléctrico, existen diversos parámetros que se deben tomar en cuenta para su funcionamiento nominal, y los modelos a utilizar deben ser capaz de analizar todos estos datos. Para que el modelo sea capaz de reconocer estos datos, primeramente, se debe “entrenar” el modelo, lo cual se realizará mediante un algoritmo inicial que introducirá las variables al modelo [5]. Los modelos que se emplearán en el desarrollo serán empleados bajo tres algoritmos de Machine Learning. Estos son: Decision Tree Classifier, Random Forest Classifier y K-Nearest Neighbors Classifier [6]. En este escrito se emplearán los algoritmos para crear el modelo de detección de fallas en motores eléctricos, y luego se compararán los resultados de cada algoritmo para reconocer el algoritmo más preciso para la detección temprana de fallas bajo las condiciones propuestas.

# Banco de información y datos experimentales

Los datos experimentales utilizados en los modelos se obtuvieron de distintos motores eléctricos proporcionados en una base de datos de libre acceso, considerando dos estados esenciales del motor eléctrico: en el caso de que el motor funciona con normalidad, y en el caso donde el sistema presenta una falla. La información utilizada para el modelo proviene de un dataset del Center for Machine Learning and Intelligent Systems [7]. Los datos proporcionados por el dataset son la temperatura del aire, la temperatura del proceso, la velocidad de rotación, el torque y el desgaste de las herramientas. El dataset presenta la información de diversos motores eléctricos con estos parámetros para luego determinar si, bajo las condiciones establecidas, puede presentarse una falla en el motor. Estos datos serán introducidos al algoritmo, y serán entrenados para ser utilizados en el proceso de predicción del modelo.

# Algoritmos de diagnósticos inteligentes de fallas

Para realizar el diagnóstico de los motores eléctricos mediante metodologías inteligentes, se utilizaron tres algoritmos de Machine Learning, asegurando que éstos estuvieran acorde al estado del arte [8].

## Decision Tree Classifier

El algoritmo de Decision Tree Classifier, o aprendizaje basado en árboles de decisión, es un clasificador de partición que se construye recursivamente dependiendo de los datos proporcionados al modelo [9]. Su objetivo principal es predecir una respuesta en función de sus covariables de acuerdo con unas reglas sucesivas donde se clasifica el estado final del sistema, en este caso, el motor eléctrico. Este algoritmo consiste en analizar las covariables con particiones donde el modelo debe tomar decisiones para escoger los diferentes caminos hacia la predicción.

## Random Forest Classifier

El algoritmo de Random Forest, o Árboles Aleatorios, es un estimador que acopla diferentes clasificadores de regresiones de árbol en varios resultados de un conjunto de datos y los promedia para mejorar la exactitud. Este método es capaz de controlar el sobredimensionamiento del modelo, produciendo resultados precisos [10]. De cada árbol se estima el error de estimación al promediar las diferentes ramas utilizando datos de prueba, aunque tiende a ser un poco más difícil de interpretar debido a su metodología de modelo de caja negra.

## K-Nearest Neighbors Classifier

El algoritmo de K-Nearest Neighbors (K-NN), o K-Vecinos más Cercanos, es un método de clasificación supervisada que sirve para estimar una función de datos de entrada con el grupo más próximo a los datos globales para clasificarlo de acuerdo con sus vecinos más cercanos [11]. En esta metodología, se escoge un número K de vecinos, luego se calcula la distancia Euclídea de los K vecinos y se cuentan los datos en cada categoría. Se estiman los valores de cada zona para proporcionar el modelo el sistema, de manera que permite agregar datos nuevos para optimizar el modelo.

# Realización de los modelos

Para el análisis de los datos, se utilizó un software de programación en *Python* [12], llamado *Jupyter Notebook* [13], junto con diversas librerías de la ciencia de los datos como *scikit-learn* [14]*,* que permiten el uso de distintos modelos de aprendizaje automático. El modelo se realizó utilizando la información del dataset aplicado en las tres metodologías detalladas anteriormente, y con cada método se determina la existencia o no de una falla en el motor eléctrico. El algoritmo empleado se presenta a continuación.

TABLA 1. Algoritmo del modelo

|  |
| --- |
| **Algoritmo 1** Declaración de las variables |
| **import** pandas **as** pd  **from** sklearn.model\_selection **import** train\_test\_split  #Para evaluar los resultados  **from** sklearn **import** metrics  **from** sklearn.metrics **import** mean\_absolute\_error  smd = pd.read\_csv("machines\_data.csv")  # Features  X = smd[['Air temperature [K]', 'Process temperature [K]','Rotational speed [rpm]','Torque [Nm]','Tool wear [min]']]  # Target  y = smd['Machine failure']  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, test\_size=0.4)  **print**(X\_train.shape)  **print**(X\_test.shape) |

TABLA 2. Algoritmo para el método A

|  |
| --- |
| **Algoritmo 2** K-Nearest Neighbors |
| #With Knearest neigbors  **from** sklearn.neighbors **import** KNeighborsClassifier  knn = KNeighborsClassifier(n\_neighbors=9)  knn.fit(X\_train, y\_train)  y\_pred = knn.predict(X\_test)  **print**(f'Accuracy Knn: {metrics.accuracy\_score(y\_test, y\_pred)}\nMean Absolute Error knn:{mean\_absolute\_error(y\_test, y\_pred)}') |

TABLA 3 Algoritmo para el método B

|  |
| --- |
| **Algoritmo 2** Decision Tree Classifier |
| #With DecisionTreeClassifier  **from** sklearn.tree **import** DecisionTreeClassifier  dtc = DecisionTreeClassifier(max\_depth=10)  dtc.fit(X\_train, y\_train)  y\_pred = dtc.predict(X\_test)  **print**(f'Accuracy Decision TreeClassifier: {metrics.accuracy\_score(y\_test, y\_pred)}\n Mean Absolute Error DecissionTree:{mean\_absolute\_error(y\_test, y\_pred)}') |

TABLA 4. Algoritmo para el método C

|  |
| --- |
| **Algoritmo 2** Random Forest Classifier |
| #With RandomForestClassifier  **from** sklearn.ensemble **import** RandomForestClassifier  forest\_model = RandomForestClassifier(max\_depth=10)  forest\_model.fit(X\_train, y\_train)  y\_pred = forest\_model.predict(X\_test)  **print**(f'Accuracy RandomForest: {metrics.accuracy\_score(y\_test, y\_pred)}\n Mean Absolute Error RandomForest:{mean\_absolute\_error(y\_test, y\_pred)}') |

# Resultados y discusión

Con el código programado, se crearon algoritmos utilizando los datos de entrenamiento del comportamiento de falla de los diferentes motores y se obtuvieron resultados detallando la existencia de la falla y la precisión del modelo. Los clasificadores fueron comparados para evaluar su rendimiento en base a los datos obtenidos que no se utilizaron en la sección de entrenamiento. Con los datos de prueba, se obtuvo el error cuadrático medio para conocer la desviación de los datos.

## Ajuste de los clasificadores

En esta etapa, el costo depende de la cantidad de parámetros y variables que deben ser optimizados y calculados en cada algoritmo inteligente. Los tres métodos de aprendizaje automático utilizados fueron optimizados debido a diferentes rangos para probar su rendimiento y precisión. Para el método de Decision Tree Classifier, se realizó el estudio del modelo con un parámetro de profundidad máximo establecido. Para el método de Random Forest Classifier, se realizó el estudio del modelo con una cantidad máxima de profundidad de árboles. Para el método de KNN, se escogieron un número de vecinos establecido para sus pruebas [6]. Esto se hizo a manera de establecer un límite fijo para comparar los tres métodos.

## Evaluación del rendimiento

Se evaluó su rendimiento y el error cuadrático medio para cada caso con sus debidas variaciones para verificar los diferentes casos de estudio y que los resultados obtenidos muestren su rendimiento correctamente. Se analizaron mediante gráficas para observar su comportamiento. Se comparó la profundidad del modelo respectivo y la exactitud con el error cuadrático medio.

Para el método de Decision Tree Classifier:

TABLA 5. Decision Tree Classifier

|  |  |  |
| --- | --- | --- |
| **Decision Tree Classifier** | **Square mean root** | **Accuracy score** |
| **Tdepth** |
| 4 | 0.026 | 0.974 |
| 5 | 0.02325 | 0.97675 |
| 6 | 0.02075 | 0.97925 |
| 7 | 0.019 | 0.981 |
| 8 | 0.0185 | 0.9815 |
| 9 | 0.01575 | 0.98425 |
| 10 | 0.0145 | 0.9855 |

Ilustración 1. Gráfico de error cuadrático medio vs. Profundidad

Ilustración 2. Gráfico de exactitud vs. Profundidad

Para el método de Random Forest Classifier:

TABLA 6. Random Forest Classifier

|  |  |  |
| --- | --- | --- |
| **Random Forest Classifier** | **Square mean root** | **Accuracy score** |
| **Tdepth** |
| 4 | 0.0235 | 0.9765 |
| 5 | 0.023 | 0.977 |
| 6 | 0.02075 | 0.97925 |
| 7 | 0.02125 | 0.97875 |
| 8 | 0.0175 | 0.9825 |
| 9 | 0.01875 | 0.98125 |
| 10 | 0.01675 | 0.98325 |

Ilustración 3. Gráfico de error cuadrático medio vs. Profundidad

Ilustración 4. Gráfico de exactitud vs. Profundidad

Para el método de K-NN:

TABLA 7. K-Nearest Neighbor Classifier

|  |  |  |
| --- | --- | --- |
| **k-Nearest Neighbors Classifier** | **Square mean root** | **Accuracy score** |
| **k** |
| 3 | 0.03275 | 0.96725 |
| 4 | 0.0305 | 0.9695 |
| 5 | 0.0295 | 0.9705 |
| 6 | 0.029 | 0.971 |
| 7 | 0.0285 | 0.9715 |
| 8 | 0.02825 | 0.97175 |
| 9 | 0.028 | 0.972 |

Ilustración 5. Gráfico de error cuadrático medio vs. k

Ilustración 6. Gráfico de exactitud vs. k

# Conclusión

En base a los resultados obtenidos, se observó que el método de Tree Classifier presentó la mayor precisión para identificar la falla en el sistema. Ahora bien, esto puede cambiar dependiendo de los datos introducidos en el sistema, ya que se pueden introducir más variables y esto puede ocasionar que el método varíe en tanto a su precisión. No obstante, el método de Tree Classifier, con los parámetros escogidos, fue el que presentó la mayor eficiencia en detectar la producción de una falla en el motor. Utilizando este método de Machine Learning, se puede predecir con certeza el momento en el que ocurrirá la falla, y realizar un mantenimiento predictivo para prevenir dicho suceso.

# Referencias

|  |  |
| --- | --- |
| [1] | S. Chapman, Máquinas Eléctricas, McGraw-Hill, 2012. |
| [2] | M. Ceraolo y D. Poli, Fundamentals of Electric Power Engineering: From Electromagnetics to Power Systems, Wiley, 2014. |
| [3] | S. Shalev-Shwartz y S. Ben-David, Understanding Machine Learning: From Theory to Algorithms, Cambridge University Press, 2014. |
| [4] | B. Shi y S. Iyengar, Mathematical Theories of Machine Learning - Theory and Applications, Springer Nature Switzerland, 2020. |
| [5] | R. Fernandes y M. Antonelli, Machine Learning: A Practical Approach on the Statistical Learning Theory, Springer International Publishing, 2018. |
| [6] | M. Kleppmann, Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems, O'Reilly Media, Inc., 2017. |
| [7] | S. Matzka, «AI4I 2020 Predictive Maintenance Dataset,» Center for Machine Learning and Intelligent Systems, 30 Ago 2020. [En línea]. Available: https://archive.ics.uci.edu/ml/datasets/AI4I+2020+Predictive+Maintenance+Dataset. |
| [8] | A. Glassner, Deep Learning: A Visual Approach, No Starch Press, 2021. |
| [9] | P. Gupta, «Decision Trees in Machine Learning,» towards data science, 17 May 2017. [En línea]. Available: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052. |
| [10] | T. Yiu, «Understanding Random Forest: How the Algorithm Works and Why it is so Effective,» towards data science, 12 Jun 2012. [En línea]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2. |
| [11] | O. Harrison, «Machine Learning Basics with the K-Nearest Neighbors Algorithm,» towards data science, 10 Sep 2018. [En línea]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761. |
| [12] | R. Kneusel, Practical Deep Learning: A Python-Based Introduction, No Starch Press, 2021. |
| [13] | Jupyter, «The Jupyter Notebook,» Jupyter Team, 2015. [En línea]. Available: https://jupyter-notebook.readthedocs.io/en/stable/. |
| [14] | F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot y É. Duchesnay, «Scikit-learn: Machine Learning in Python,» JMLR, 2011. [En línea]. Available: https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html. |