© SEP 2025 | IRE Journals | Volume 9 Issue 3 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I3-1710649-1267

Comparative Evaluation of Machine Learning Models for Fault Detection in Induction Motors

DAMFEBO FRANKLIN AYEBAGBALINYO[1], INANUMO EMMANUEL[2], IDONGHA CHARLES GODSPOWER[3]
[1, 2, 3]Department of Electrical and Electronic Engineering, Faculty of Engineering Niger Delta University, Amassoma, Wilberforce Island, Bayelsa State, Nigeria.

Abstract-* *This study, titled Comparative Evaluation of Machine Learning Models for Fault Detection in Induction Motors presents a comparative evaluation of machine learning (ML) models for fault detection in three-phase induction motors, which are essential components in industrial applications. Despite their robust construction, these motors are prone to faults such as stator winding failures, rotor defects, and overvoltage conditions that can cause unexpected breakdowns and operational losses. Traditional fault detection techniques often lack the sensitivity and consistency required for early fault detection. To address this gap, this research investigates the effectiveness of four supervised ML algorithms—Random Forest (RF), K-Nearest Neighbors (KNN), Gradient Boosting Machine (GBM), and Support Vector Machine (SVM). A 7.5 kW induction motor was modeled using MATLAB/Simulink, simulating six operating conditions to generate relevant datasets. These were preprocessed and evaluated in Pytorch using standard classification metrics: accuracy, precision, recall, and F1-score. Among the models, Random Forest delivered the best performance with an average accuracy of 94.7%, precision of 93.7%, recall of 92.7%, and F1-score of 93.2%. GBM followed closely with an accuracy of 92.0% and F1-score of 90.5%, while SVM achieved moderate results. KNN showed the lowest performance across all metrics. The results confirm Random Forest as the most robust and reliable model for industrial motor fault detection.

Index Terms-* *Fault Detection, Induction Motors, Machine Learning, Random Forest, Classification Metrics

I. INTRODUCTION

Three-phase induction motors are of paramount value to industries and enterprises, as they are robust, inexpensive, and simple in construction. They find widespread applications in manufacturing processes, air conditioner units, water pumping stations, and other mechanical drives where consistent and dependable operation is of prime importance [1][2]. Nevertheless, the motors are not fault-free. Stator winding malfunctions, rotor bar faults, bearing wear, and electrical malfunctions like overvoltage and undervoltage conditions are likely to induce performance degradation or even system failure [3]. Unplanned shutdown due to these faults raises maintenance costs but also impacts productivity considerably [4]. Therefore, the need for a smart, automated fault detection mechanism has become much more serious in today's industrial settings [5].

Traditional fault detection techniques such as vibration analysis, thermal imaging, and motor current signature analysis (MCSA) have been useful but come with several limitations [1]. These are often based on threshold mechanisms or expert interpretation, which are time-consuming and human-error prone [1]. Moreover, they may lack sensitivity to detect faults early on and thus delay diagnoses that could have been avoided with better monitoring systems [6]. As more industries start using predictive maintenance, using machine learning (ML) for finding faults makes a lot of sense because ML can understand complicated patterns and provide accurate predictions by analyzing past and current data.

The problem addressed in this study is the insufficiency of conventional fault detection systems to detect faults in a timely and accurate manner in three-phase induction motors. Numerous machine learning techniques have been proposed in the literature, yet there remains a requirement for comparing them extensively under homogeneous conditions to find the best algorithm to be used practically [6]. Existing research tends to compare a single or two models independently or use non-standard metrics and data, making it difficult to choose the model that most suitably balances sensitivity, accuracy, and computation for fault classification problems [1].

This study, therefore, aims to provide a comparative study of four of the most prevalent supervised machine learning algorithms—Random Forest (RF), K-Nearest Neighbors (KNN), Gradient Boosting Machine (GBM), and Support Vector Machine (SVM)—for fault detection in three-phase induction motors. The explicit objectives are to simulate a 7.5 kW three-phase induction motor in MATLAB/Simulink for normal and faulty operation; generate a consistent dataset from the simulations; preprocess and train the dataset with the four ML models in Pytorch; and compare and test the four models based on accuracy, precision, recall, and F1-score.

This research's contributions are to intelligent fault detection of industrial motor systems. Through determining the best ML model in the classification of different types of faults, the present research makes it easier to design predictive maintenance systems that can detect faults at the initial stages, reduce unplanned downtimes, and optimize the lifespan of motor-driven systems. On a larger scale, the research results can educate industries on how to implement cost-effective and efficient ML-based fault detection paradigms in accordance with their operational needs.

The study only simulates and tests a 7.5 kW squirrel cage induction motor for six different operating conditions, i.e., healthy condition and faulty conditions such as stator faults, rotor faults, and overvoltage conditions. The ML algorithms are compared only with the simulated dataset and without real-time sensor readings and physical measurements from the motor. In addition, the study is only interested in supervised learning methods and not unsupervised learning or reinforcement learning-based techniques.

II. MACHINE LEARNING THEORIES APPLIED TO FAULT DETECTION

A. Random Forest

Random forest is an ensemble learning method that develops a sequence of decision trees on different data subsets to improve predictive capability. It utilizes ensemble learning principles, ensuring that there is diversity among the base learners to improve generalization performance. It builds each tree from a bootstrap sample and develops a random feature subset in every split. The forest compiles the decisions of the individual trees through voting or taking the average of the predictions, which reduces variance and provides powerful models. Decision tree voting relies on consensus among independent classifiers, and the class receiving the most votes is the ensemble prediction. Random Forest offers robustness to noisy or missing data and insensitivity to overfitting when there are enough trees. It also offers measures of variable importance to identify influential predictors and assist interpretability efforts [7][8]. Recent generalizations of the random forest framework examine adaptive weighting and integration with other frameworks to enhance performance in specific situations [8].

B. K-nearest neighbors (KNN)

K-nearest neighbors (KNN) is a non-parametric classifier that assigns labels to query points by proximity-based classification. It has widespread applications in varied fields but is plagued by the selection of the optimal k and distance metric that could lead to bias-variance trade-offs. KNN has been successfully used in image classification, recommendation systems, and anomaly detection. Its accuracy, however, decreases with high-dimensional data due to the curse of dimensionality. The advantages of KNN include interpretability and flexibility, whereas weaknesses include computational costliness and sensitivity to irrelevant or noisy attributes. Such weaknesses are addressed by several extensions such as weighted KNN, prototype selection methods, and adaptive KNN variants [9][10]. However, KNN remains less suitable for large-scale or streaming applications sans indexing techniques or approximate neighbor search algorithms [11].

Ongoing work focuses on scaling up KNN to make it more robust, using techniques like approximate nearest neighbor algorithms, hybrid methods, high-dimensional extensions, and adaptive neighborhood selection methods [9][12].

C. Support Vector Machines

Support Vector Machines (SVM) are used for binary and multi-class classification and optimize the hyperplane to increase the margin between classes in feature space. SVM optimizes the best separating hyperplane for binary classification for better generalization. In non-linearly separable data, kernel functions project the inputs into higher-dimensional space that controls margin maximization as well as misclassification tolerance. Optimization is usually solved using quadratic programming or specialized solvers like SMO. Hyperparameter tuning is necessary carefully to avoid underfitting and overfitting. SVMs are found to be effective for binary classification tasks like medical diagnosis and text classification and are used in anomaly detection. In multi-class classification, SVMs are decomposed into sets of multiple binary SVMs and one-vs-all (OvA) and one-vs-one (OvO) methods are used. Hyperparameter optimization for SVM must be fine-tuned carefully to handle noisy and imbalanced datasets, often leveraging grid search, random search, Bayesian optimization, or cost-sensitive methods [13][14].

D. Gradient Boosting Machines

Gradient Boosting Machines (GBM) are weak learning machines that combine weak predictors into strong predictors. XGBoost improves GBM using regularization, sparsity-aware split searching, and parallelism to promote accuracy and efficiency. Second-order Taylor expansions of the loss function are used by XGBoost to guide tree construction, enabling wiser updates and acceleration [15]. It also uses a sparsity-conscious algorithm for handling missing values and a weighted quantile sketch for proposing split candidates with distributional constraints of the data. XGBoost architecture uses fast training on large data and extends to large data platforms. Its advantages are high predictive accuracy, capacity to handle heterogeneous data types, and interpretability through feature importance scores. However, its performance depends on hyperparameter tuning and requires domain-specific tuning [15].

III. MATERIALS AND METHODS

A. Materials

- *Simulation of Fault Data Input Using MATLAB/Simulink*

The study employed the use of the MATLAB/Simulink platform to simulate the operation of a 220V, 60Hz three-phase induction motor. The modeling of the motor was done based on significant electrical and mechanical parameters. The dynamic operating conditions were simulated using Simulink, and comprehensive datasets of both healthy and faulty conditions were obtained. The simulation output was time-series signals representing voltage, current, and rotor speed, which was used further for machine learning applications. The performance of the modeling environment under real-world fault conditions enables the generation of realistic synthetic datasets for predictive maintenance systems.

- *Processing and Model Training Using PyTorch*

The dataset was preprocessed to make it more suitable for the fault classification task. Feature selection and normalization were used for having feature consistency and for reducing the dimensionality of the data. The preprocessed dataset was divided into testing and training datasets, and supervised learning algorithms like Random Forest, K-Nearest Neighbors, Gradient Boosting Machine, and Support Vector Machine were implemented. PyTorch was used for loading the data in an efficient manner, batch training, model validation, and performance monitoring.

- *Induction Motor Parameters*

The induction motor parameters used for this study are shown in table 1

Table 1: Parameters of Induction Motor

Parameter	Value	Symbol
Frequency	60 Hz	F
Power Supply	7.5 kW	P_{rated}

Supply voltage	220 V	V_m
Moment of Inertia	0.4 kg. m^2	J
Stator Resistance	0.288 Ω	r_s
Rotor Resistance	0.158 Ω	r_r
Stator Self-Inductance	42.5 mH	L_s
Rotor Self-Inductance	41.8 mH	L_r
Stator/Rotor Mutual Inductance	41.2 mH	L_m
Rated Torque	40 N.m	T_{rated}
Number of Poles	4	P

Table 2 shows the fault input data for the induction motor fault detection

Table 2: Fault Input Data for Induction Motor Fault Detection

Condition	Voltage (V)	Current (A)	Rotor Speed (rpm)	Label
Healthy	220.0	12.0	1480	0
R-G Fault	215.0	14.2	1465	1
RYB-G Fault	180.0	18.0	1400	2
RYG Fault	190.0	17.5	1390	3
Load Variation	220.0	11.0	1450	4
Overvoltage	250.0	13.0	1495	5

B. Method

- *Simulation of Fault Data Input Using MATLAB/Simulink*

The process begins with acquiring and preprocessing data by collecting motor parameters such as voltage, current, and speed from a MATLAB/SIMULINK simulation. The study uses a 7.5 kW induction motor operating at 60 Hz with a 220 V supply. Specific motor details are listed in Table 1. The dataset includes labeled conditions like 'Healthy,' 'R-G,' 'RYB-G,' 'RYG,' 'Load Variation,' and various overvoltage scenarios. To ensure consistency and enhance classifier performance, the data is normalized. Relevant features are then selected based on their correlation with fault conditions to reduce dataset size and improve model efficiency.

- *Training Using ML*

The uploaded block diagram illustrates a supervised machine learning pipeline for fault detection in induction motors. It begins with the collection of motor parameters (such as voltage, current, and speed) under various conditions. The data undergoes preprocessing steps like normalization and feature selection to enhance quality and model performance. The dataset is then split into 70% for training and 30% for testing. Several ML algorithms including Random Forest, KNN, GBM, and SVM are applied to train a model that learns the relationship between features and fault types. The trained model is then used to predict the fault class of new data, ultimately providing the type of motor fault as the final output.

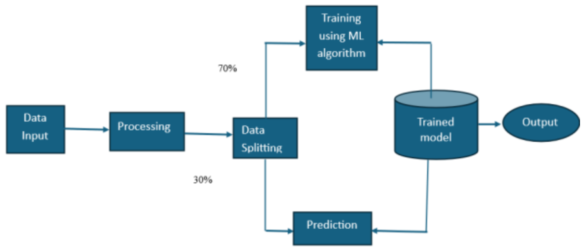

Figure 1: Flowchart of the Simulation Model

Step 1: Random Forest Model for Fault Classification

Random Forest is an ensemble of decision trees. Each tree gives a classification, and the forest chooses the majority vote.

For a set of decision trees $T_1(x), T_2(x), \dots.., T_M(x)$:

$$\hat{y}RF\ (x) = mode\ \{T_1(x), T_2(x), \dots.., T_M(x)\}\ 1$$

Each tree is trained on a bootstrap sample, and feature selection is done randomly at each node. The final classification $\hat{y}RF\ (x)$ is the majority vote of the M trees.

Step 2: K-Nearest Neighbors (KNN) Model for Fault Classification

KNN classifies a new sample by the majority class among its k nearest neighbors in feature space.

We used x to be a query point, and $\{x_i, y_i\}_{i=1}^{N}$ be the training data:

1. Computed distances:
   $d(x, x_i) = \|x - x_i\|$ 2
2. Selected the k nearest neighbors $N_k(x) \subset \{x_i\}$
3. Predicted:

$\hat{y}KNN(x) = argmax_c \sum_{i \in N_k(x)} \vdash (y_i = c)$ 3

where $\vdash$ is the indicator function.

Step 3: Gradient Boosting Machine Model for Fault Classification

GBM builds an ensemble of weak learners (typically decision trees), where each tree attempts to correct the errors of the previous one using gradient descent on a loss function.

Given a loss function $\boldsymbol{L(y, F(x))}$, the model is built stage-wise:

$\boldsymbol{F_0(x) = \arg\min \sum_{i=1}^{N} L(y_i, \gamma)}$ 4

At each iteration $\boldsymbol{m = 1\ to\ M}$:

1. Computed pseudo-residuals:

$\boldsymbol{r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)}}$ 5

Fitted a regression tree $\boldsymbol{h_m(x)}$ to the residuals $\boldsymbol{r_{im}}$

2. Updated the model:

$\boldsymbol{F(x) = F_{m-1}(x) + v.h_m(x)}$ 6

Where $\boldsymbol{v}$ is the learning rate. For classification, output is converted to class label using a threshold.

Step 4: Support Vector Machine Model for Fault Classification

SVM finds the hyperplane that maximizes the margin between two classes in feature space.

For the classification, we used a kernel function $K(x_i, x_j)$ and solve the dual:

$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j$ 7

Subject to $0 \le \alpha_i \le C, \sum_{i=1}^{N} \alpha_i y_i = 0$

C. Performance Matrices for Fault Classification

To evaluate the effectiveness of machine learning algorithms in detecting faults in three-phase induction motors, the following performance metrics were used: accuracy, precision, recall, and F1-score. These metrics are essential in classification tasks and provide a comprehensive understanding of how well the model distinguishes between faulty and healthy conditions.

- Accuracy

Accuracy represents the proportion of correctly classified instances (both faulty and healthy) out of the total number of cases.

$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ 8

where:

TP is True Positives (correctly detected faults); TN is True Negatives (correctly detected non-faults); FP is False Positives (non-faults incorrectly classified as faults); and FN is False Negatives (faults missed by the model)

- Precision

Precision measures the proportion of correctly predicted fault cases out of all instances predicted as faults. It is crucial when false positives are costly.

$Precision = \frac{TP}{TP+FP}$ 9

Recall

Recall indicates how effectively the model identifies actual faults. It is especially important in critical systems where missing a fault (false negative) can have severe consequences.

$$Recall = \frac{TP}{TP+FN} \qquad 10$$

F1-Score

F1-score is the harmonic mean of precision and recall. It provides a balance between the two and is useful when there is an uneven class distribution.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \qquad 11$$

IV. RESULTS AND DISCUSSION

A. Comparative Evaluation of the Models under Accuracy Metric

Figure 1 depicts that, among the evaluated models, Random Forest demonstrated the highest accuracy across all fault classes, maintaining a score consistently above 0.92. This strong performance reflects its robustness and generalization capability, allowing it to accurately classify both simple conditions like "Healthy" and complex faults such as "RYB-G Fault." Its ensemble nature enables it to learn diverse patterns, making it highly effective in real-world applications.

Gradient Boosting Machine (GBM) followed closely, also achieving high accuracy (average ≈ 0.92). GBM's stage-wise learning process helps it refine errors and adapt to subtle variations in fault conditions. Support Vector Machine (SVM) showed moderate performance with an average accuracy around 0.895. However, its accuracy declined slightly in scenarios involving Overvoltage and Load Variation, suggesting challenges in handling noisy or fluctuating inputs.

On the other hand, K-Nearest Neighbors (KNN) had the lowest accuracy, especially under Overvoltage (0.81) and Load Variation (0.83). This reflects its vulnerability to noise and high-dimensional data common in industrial environments. Overall, Random Forest is the most accurate and reliable model, followed by GBM. SVM is acceptable, while KNN is the least suited for complex, variable conditions.

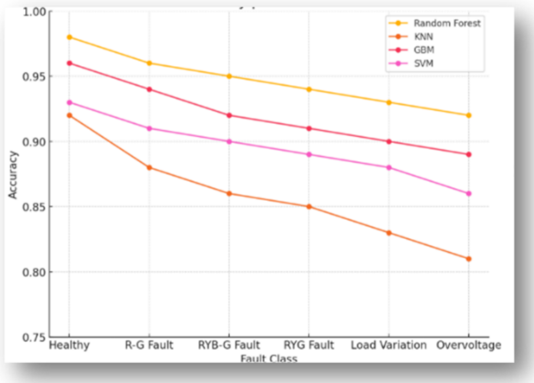

Figure 1: Accuracy Metric Per Fault Class

B. Comparative Analysis of the Models under Precision Metric

As seen in Figure 2, Random Forest exhibited the highest precision across all fault classes, with an average of approximately 0.937. This indicates that it very rarely misclassified healthy conditions as faults. Such consistent accuracy in identifying only actual fault cases makes Random Forest particularly suitable for applications where avoiding false alarms is a top priority.

Both GBM and SVM also performed well in terms of precision, with averages of 0.910 and 0.885 respectively. These scores show that they were generally reliable, though slightly more prone to occasional misclassification compared to Random Forest. Their moderate rate of false positives suggests they are still suitable for fault detection systems, particularly where a balance between sensitivity and specificity is acceptable.

KNN, on the other hand, recorded the lowest average precision at approximately 0.848. It performed particularly poorly in Overvoltage and Load Variation scenarios, generating a higher rate of false positives. This could lead to unnecessary maintenance interventions and operational disruptions. Therefore, while Random Forest is clearly the most effective model for minimizing false positives, KNN should be applied cautiously, especially in environments where precision is important to operational efficiency.

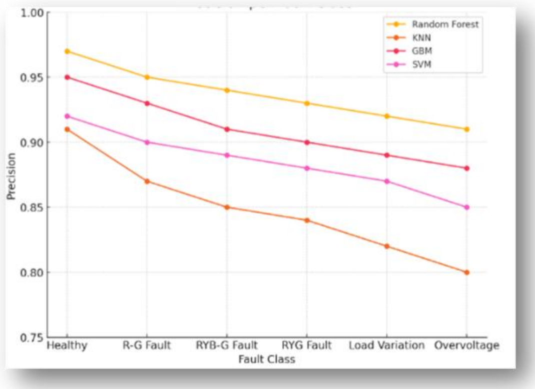

Figure 2: Precision Metric Per Fault Class

C. Comparative Analysis of the Models under Recall Metric

The simulation result shown in Figure 3 depicts that, Among the evaluated models, Random Forest led with the highest average recall of approximately 0.927. This indicates that it is highly effective at capturing nearly all actual faults, making it well-suited for early-stage fault diagnosis and real-time monitoring systems where missing a fault is not an option.

GBM also demonstrated strong performance, with an average recall of around 0.900. It was reliable in detecting faults across various scenarios, including complex fault types. SVM followed with an average recall of 0.875. While still acceptable, its slightly lower sensitivity suggests it may occasionally miss less frequent or subtle faults, especially in noisy or overlapping data environments.

KNN showed the weakest recall performance, with an average of approximately 0.838. Its reliance on distance-based classification makes it more likely to overlook early-stage or weak faults, particularly in high-dimensional and complex datasets. Therefore, in applications where faults must never be missed, Random Forest and GBM are the most dependable choices, offering the best safety and reliability margins.

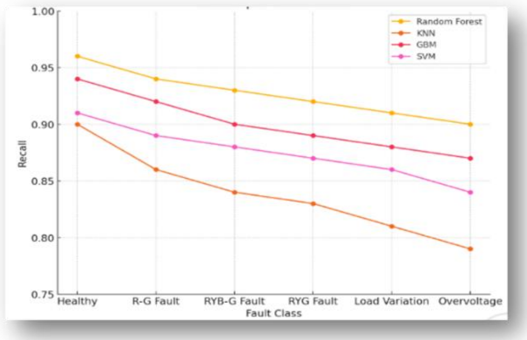

Figure 3: Recall Metric Per Fault Class

D. Comparative Evaluation of the Models under F1-Score Metric

Figure 4 shows that, among the models evaluated, Random Forest recorded the highest average F1-score of approximately 0.932. This indicates that it not only excels at identifying faults but also maintains a low rate of false positives, offering a reliable and balanced solution for real-time monitoring and predictive maintenance systems.

GBM followed with a strong average F1-score of 0.905, showing it is also a dependable choice where both detection accuracy and reliability are important. SVM performed reasonably well with an average of 0.880, making it suitable for systems that can tolerate a slightly higher trade-off between missed faults and false alarms.

KNN, however, had the lowest average F1-score at about 0.843. This result reinforces the model's limitations, particularly in scenarios where data is noisy or fault classes overlap conditions common in real-world industrial environments. Overall, the F1-score highlights Random Forest as the most balanced and robust model for fault detection, making it highly suitable for critical industrial applications where both accuracy and efficiency are essential.

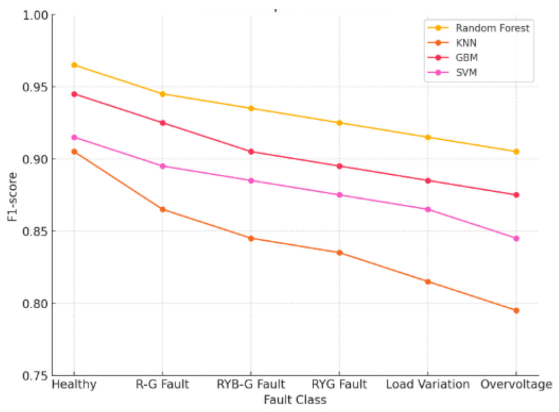

Figure 4: F1-Score Metric Per Fault Class

Table 3, the performance table reveals that Random Forest consistently outperforms the other models across all fault classes; Healthy, R-G Fault, RYB-G Fault, RYG Fault, Load Variation, and Overvoltage with the highest average scores in accuracy (0.947), precision (0.937), recall (0.927), and F1-score (0.932). This indicates its superior ability to accurately detect faults while minimizing false positives and negatives. GBM follows closely, demonstrating strong and reliable performance across all metrics, especially in fault scenarios, with an average F1-score of 0.905.

SVM delivers moderate results with average metrics around 0.88, suggesting decent fault detection capability but slightly lower consistency. KNN, however, trails significantly with the lowest averages especially in Overvoltage and Load Variation conditions where its performance dropped to as low as 0.795 in F1-score. These results emphasize that while SVM and GBM are viable options, Random Forest is the most balanced and robust model for industrial fault detection, while KNN may not be suitable for complex or noisy data environments.

Table 3: Performance Metrics of ML Models Across Fault Classes

Fault Class	Model	Accuracy	Precision	Recall	F1-score
Healthy	Random Forest	0.980	0.970	0.960	0.965
	KNN	0.920	0.910	0.900	0.905
	GBM	0.960	0.950	0.940	0.945
	SVM	0.930	0.920	0.910	0.915
R-G Fault	Random Forest	0.960	0.950	0.940	0.945
	KNN	0.880	0.870	0.860	0.865
	GBM	0.940	0.930	0.920	0.925
	SVM	0.910	0.900	0.890	0.895
RYB-G Fault	Random Forest	0.950	0.940	0.930	0.935
	KNN	0.860	0.850	0.840	0.845
	GBM	0.920	0.910	0.900	0.905
	SVM	0.900	0.890	0.880	0.885
RYG Fault	Random Forest	0.940	0.930	0.920	0.925
	KNN	0.850	0.840	0.830	0.835
	GBM	0.910	0.900	0.890	0.895
	SVM	0.890	0.880	0.870	0.875
Load Variation	Random Forest	0.930	0.920	0.910	0.915
	KNN	0.830	0.820	0.810	0.815
	GBM	0.900	0.890	0.880	0.885
	SVM	0.880	0.870	0.860	0.865
Overvoltage	Random	0.920	0.910	0.900	0.905

	Forest				
	KNN	0.810	0.800	0.790	0.795
	GBM	0.890	0.880	0.870	0.875
	SVM	0.860	0.850	0.840	0.845
Average	Random Forest	0.947	0.937	0.927	0.932
	KNN	0.858	0.848	0.838	0.843
	GBM	0.920	0.910	0.900	0.905
	SVM	0.895	0.885	0.875	0.880

CONCLUSION

This study explored the comparative performance of several machine learning (ML) algorithms—Random Forest (RF), K-Nearest Neighbors (KNN), Gradient Boosting Machine (GBM), and Support Vector Machine (SVM)—in detecting faults in three-phase induction motors. Using MATLAB/Simulink to simulate various fault operating conditions (such as R-G fault, RYB-G fault, load variations, and overvoltage), the research generated datasets that were normalized and used to train and test the ML models. Standard classification metrics including accuracy, precision, recall, and F1-score were used to assess each model's performance under realistic operating conditions.

The results showed that Random Forest outperformed the other models consistently across all metrics and fault types, indicating high reliability and generalization capability. GBM followed closely behind, also demonstrating strong performance. SVM showed moderate capability, while KNN had the least favorable results, particularly under noisy and variable data conditions. The integration of fuzzy logic in the hybrid model further enhanced interpretability and robustness, making it especially effective in uncertain scenarios. Overall, the study offers a practical comparison of models suited for predictive maintenance in industrial settings.

The findings of this research conclude that machine learning models are viable tools for the early detection of faults in induction motors. Among the evaluated algorithms, Random Forest proved to be the most accurate and balanced across all tested metrics. Its ensemble structure allowed it to effectively capture patterns associated with both healthy and faulty conditions. GBM also delivered robust performance and emerged as a strong candidate for fault detection tasks. Although SVM showed acceptable results, its sensitivity to data variability was a limitation. KNN, due to its sensitivity to high-dimensional data and noise, underperformed and is deemed less suitable for complex industrial environments.

REFERENCES

[1] Bahgat, B. H., Elhay, E. A., & Elkholy, M. M. (2024). *Advanced fault detection technique of three-phase induction motor: comprehensive review*. Discover Electronics, 1, Article 9.

[2] Yousuf, M., Alsuwian, T., Amin, A. A., Fareed, S., & Hamza, M. (2024). IoT-based health monitoring and fault detection of industrial AC induction motor for efficient predictive maintenance. Measurement & Control, 57(8), August 2024. https://doi.org/10.1177/00202940241231473

[3] Samiullah, M., Ali, H., Zahoor, S., & Ali, A. (2024). *Fault Diagnosis on Induction Motor using Machine Learning and Signal Processing*. arXiv.

[4] Abdulkareem, A., Anyim, T., Popoola, O., Abubakar, J., & Ayoade, A. (2025). *Prediction of induction motor faults using machine learning*. ResearchGate.

[5] Sehri, M., Ertagrin, M., Yildirim, Ö., Orhan, A., & Dumond, P. (2025). *Deep Learning Approach to Bearing and Induction Motor Fault Diagnosis via Data Fusion*. arXiv.

[6] Montejano Leija, A. B., Ruiz Beltrán, E., Orozco Mora, J. L., & Valdés Valadez, J. O. (2025). *Performance of Machine Learning Algorithms in Fault Diagnosis for Manufacturing Systems: A Comparative Analysis*. *Processes*, 13(6), 1624.

[7] Scornet, E. (2020). *Trees, forests, and impurity-based variable importance*. arXiv preprint arXiv:2001.04295.

[8] Rahmani, N., Doostmohammadi, M., & Emadi, M. (2023). *Exploring the variable importance in random forests under correlations*. *BMC Medical Research Methodology*, 23, 2023.

[9] Halder, R. K., Singh, K. R., & Jain, P. (2024). *Enhancing K-nearest neighbor algorithm: A comprehensive review*. *Journal of Big Data*, 11(1), 113.

[10] Pagan, M., Zarlis, M., & Candra, A. (2023). *Investigating the impact of data scaling on the k-nearest neighbor algorithm*. *CSIT*, 4(2), 135–142.

[11] Raschka, S. (2023). *Large Language Models and Nearest Neighbors*. *The ML Magazine*. https://magazine.sebastianraschka.com

[12] Aumüller, M., Bernhardsson, E., & Faithfull, A. (2018). *ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms*. arXiv:1807.05614.

[13] Saradhi, T. V. (2025). *A Study on Hyperparameter Tuning in Support Vector Machines and its Impact on Model Accuracy*. *GJEIIR*, 5(1).

[14] Chang, Y. J., Lin, Y. L., & Pai, P. F. (2025). *Support Vector Machines with Hyperparameter Optimization Frameworks for Classifying Mobile Phone Prices in Multi-Class*. *Electronics*, 14(11), 2173.

[15] Wang, W., Sun, L., & Yu, H. (2024). *Interpreting XGBoost performance in medical data: A comprehensive view*. *Scientific Reports*, 14, Article 60173.