**Transformer Fault Prediction using Machine Learning**

##### Priyanka Sarma

*Assistant Professor, Department of Computer Science & Electronics, University of Science & Technology, Meghalaya, India.*

##### Dr. Anwar ul Islam\*

*Associate Professor, Department of Computer Science & Electronics, University of Science & Technology, Meghalaya, India.*

##### Tony Bayan

*Assistant Professor, Department of Computer Science, K.C Das Commerce College, Gauhati University, India.*

*\*Corresponding author*

*e-mail:* [*atowar91626@gmail.com*](mailto:atowar91626@gmail.com)



## ABSTRACT

Distribution Transformers are integral parts of an infrastructure which ensures reliable distribution of electricity in a city across various households. These transformers albeit robust are subject to various internal as well as external factors which may affect it’s working and increase reactive maintenance costs. The increasing complexity and criticality of power systems demand proactive measures for ensuring their reliable operation. Transformer faults represent a significant concern in power infrastructure, leading to downtime, financial losses, and potential safety hazards. This paper explores the application of machine learning techniques for the prediction of transformer faults, aiming to enhance the resilience and efficiency of power distribution systems. The proposed approach leverages advanced machine learning algorithms. By analysing historical data from transformers, including oil levels, oil temperature, winding temperature, current and voltage levels the model learns patterns indicative of impending faults. The paper emphasises the development of a predictive maintenance system that can identify potential transformer issues before they escalate, thereby minimising downtime and reducing maintenance costs. Key component of the project include model training using historical fault data. The research contributes to the broader field of predictive maintenance in power systems, demonstrating the effectiveness of machine learning in addressing critical infrastructure challenges. The outcomes of this project are expected to offer utilities and power system operators an invaluable tool for improving the reliability and longevity of transformers. Additionally, the integration of machine learning in fault prediction aligns with the industry's shift towards smart grids and proactive maintenance strategies, contributing to a more sustainable and resilient energy infrastructure.

**1. INTRODUCTION**

Power transformers serve as critical components within electrical distribution systems, ensuring the efficient transmission and regulation of voltage across the grid. As key assets, transformers play a pivotal role in maintaining the reliability and stability of power networks. The occurrence of faults in these transformers can lead to severe consequences, including service interruptions, equipment damage, and costly downtime. Timely detection and prediction of potential faults are, therefore, imperative for implementing proactive maintenance strategies and mitigating the impact of transformer failures.

Traditional maintenance approaches often rely on scheduled inspections and routine assessments, which may not be sufficient for capturing subtle changes in transformer conditions. In light of these challenges, this research paper focuses on harnessing the power of machine learning to predict transformer faults without relying on real-time monitoring, leveraging historical data and periodic measurements for analysis.

Our approach seeks to bridge the gap between traditional maintenance practices and the emerging paradigm of condition-based monitoring. By harnessing historical operational data, load profiles, environmental variables, and diagnostic test results, we aim to develop a comprehensive understanding of the factors influencing transformer health. Machine learning algorithms will then be employed to identify hidden patterns, correlations, and anomalies within the data, with the ultimate goal of constructing predictive models capable of anticipating impending transformer faults.

The methodologies explored in this research paper encompass a diverse array of machine learning techniques, ranging from classical algorithms to more advanced models, to extract meaningful insights from the data. This comprehensive analysis intends to provide a nuanced understanding of the dynamic interplay between various operational parameters and the likelihood of transformer faults. Through the integration of machine learning into predictive maintenance strategies, our approach aims to enhance the reliability of transformer operation, reduce unplanned downtime, and optimise maintenance activities.

The significance of this paper lies in its potential to offer a practical and cost-effective alternative for transformer fault prediction. By enabling utilities and industries to adopt

proactive maintenance practices tailored to their specific operational environments, this research has the capacity to reshape conventional approaches to transformer reliability and contribute to the advancement of resilient electrical distribution systems.

##### 1.2 Literature Review

In a study conducted by Jawad Faiz et al. (2014) titled "Diagnosing Power Transformers Faults," the authors focused on ensuring a secure and reliable electrical energy supply by addressing faults in power transformers. The methodology presented in the research categorises the benefits into pros, including the enhancement of safe and reliable electrical energy supply, prevention of costly maintenance and repairs, improvement of power system stability, and swift interruption to minimise damages and prevent instability. The paper also identifies research gaps, emphasising the need to explore scalability issues, evaluate energy efficiency in off-grid locations, and further investigate the integration with smart grids. It recognizes the significant role of transformers in privatised and restructured power systems, emphasising the importance of precise diagnosis and rapid response to internal faults. The methodology involves statistical analysis, diagnostic techniques, and exploration of fault scenarios to develop effective strategies for fault prevention, contributing to the safe operation of power transformers in evolving power systems. Despite the merits of the proposed methodology, there are cons such as high initial implementation costs, the requirement for specialised technical expertise, dependence on stable internet connectivity, security concerns in data transmission, and potential challenges in remote areas. The paper suggests addressing these challenges through the development of advanced predictive maintenance models, establishment of standards for interoperability, and assessment of system interface user-friendliness.

In a recent study conducted by Hanane Hadiki et al. in July 2023, titled "Transformers Faults Prediction Using Machine Learning Approach," the authors employed a methodology to optimise transformer maintenance through Machine Learning-based fault prediction. The study utilised three-phase current and voltage measurements as a training dataset for various algorithms, including K-Nearest Neighbour and Decision Trees. Evaluation criteria focused on accuracy metrics

and cross-validation techniques to enhance therobustness of the chosen algorithms, ultimately identifying K-Nearest Neighbour and Decision Trees as the most accurate predictors. The pros of the methodology include its cost-effective alternative to traditional maintenance methods, utilisation of Machine Learning algorithms for accurate fault prediction, enhancement of efficiency through leveraging three-phase current and voltage measurements, and the high accuracy demonstrated by K-Nearest Neighbour and Decision Trees. Cross-validation techniques were highlighted for improving algorithm generalisation and robustness. However, the study also identifies several cons, such as dependency on the quality and representativeness of the training dataset, the potential need for technical expertise during initial implementation and algorithm fine-tuning, and the limited interpretability of Machine Learning models, posing challenges in understanding the reasoning behind predictions. To address research gaps, the paper emphasises the importance of standardised data collection processes, improved interpretability of Machine Learning models, and exploration of the approach's applicability to diverse transformer types and operating conditions. Continuous model adaptation to evolving transformer conditions and strategies for addressing rare or novel fault scenarios are also identified as crucial areas requiring further investigation to enhance the reliability and effectiveness of Machine Learning-based approaches in transformer maintenance.

In their 2019 study, "Transformer Fault Prediction Method Based on Multiple Linear Regression," Qin Jiafeng et al. employed a systematic methodology for analysing transformer fault evolution and establishing a robust forecasting model using multiple linear regression. The study's pros include a systematic approach to analyse fault evolution, a robust forecasting model that considers external factors, comprehensive correlation analysis between characteristic parameters and fault types, and the prediction of various faults and overall equipment failure. The methodology also offers real-time and accurate fault diagnosis. However, the study reveals research gaps, such as a lack of detailed exploration of specific external factors considered in the forecasting model, potential oversimplification through the assumption of linearity in the relationship between characteristic parameters and faults, and insufficient consideration of challenges associated with implementation.

The absence of explicit evaluation metrics formodel performance raises uncertainties about reliability. Cons associated with the methodology include a lack of detailed descriptions for each step, limited insight into specific external factors, the assumption of linearity in the relationship between characteristic parameters and faults, potential need for extensive data for accurate predictions, complexity posing challenges for implementation, and the omission of explicit evaluation metrics for model performance. The paper calls for future research to address these gaps to enhance the robustness and applicability of transformer fault prediction models. Closing these research gaps is crucial for ensuring the reliability and effectiveness of the proposed methodology in practical applications.

In the 2022 study titled "Electrical Faults Detection and Classification using Machine Learning" by Janarthanam K et al., a methodology is proposed for predicting faults in electrical power transmission lines using machine learning in the Spyder IDE. The study's pros include early fault detection, improved system reliability, enhanced efficiency in power transmission, and the utilisation of machine learning algorithms for comprehensive fault analysis. The research, aimed at addressing the increasing demand for electricity and stagnant transmission capacity development, focuses on identifying and classifying common faults. Multiple machine learning algorithms are implemented and tested with different input combinations to enhance accuracy. The ultimate goal is to develop a reliable approach for the early detection and classification of faults, contributing to improved system reliability and efficiency. However, the study identifies several cons and research gaps in the current landscape of machine learning-based fault prediction for electrical power transmission lines. Cons include dependence on accurate input data, implementation complexity, resource-intensive training processes, possible algorithmic biases, and initial setup and development costs. Research gaps encompass the optimization of input parameters, addressing challenges in real-time implementation and scalability, adapting to evolving grid architectures and operational conditions, assessing model generalisation across diverse configurations, and addressing potential ethical concerns and biases in algorithm deployment. The paper emphasises that closing these gaps is essential for advancing the reliability and applicability of fault prediction systems in power transmission

networks.

In light of the research publications discussed, there is a growing interest in utilising machine learning for transformer fault prediction, reflecting the potential benefits in enhancing the reliability and operational efficiency of power systems. However, these studies also reveal critical research gaps that need attention for the successful implementation of machine learning-based fault prediction in transformer systems. One prominent gap lies in the optimization of input parameters and the challenges associated with real-time implementation and scalability. Adapting to evolving grid architectures and operational conditions, assessing model generalisation across diverse transformer configurations, and addressing potential biases are identified as essential areas requiring further investigation. Additionally, the complexity and resource-intensive nature of the training process, along with possible algorithmic biases, present significant cons that need careful consideration. The initial setup and development costs further underscore the practical challenges associated with implementing machine learning-based approaches for transformer fault prediction. In conclusion, this research project aims to bridge these identified gaps by developing an intelligent system for transformer fault prediction using machine learning. The objective is to create a solution that is not only accurate and efficient but also addresses the challenges posed by real-world implementation. By focusing on these aspects, the research endeavours to contribute to the advancement of reliable and scalable machine learning applications in the domain of transformer fault prediction, ultimately providing a valuable tool for maintaining the integrity and functionality of power systems.

**2. METHODS :**

To develop a machine learning model, the following 6 steps are to be accomplished :

1. Data Collection and Preprocessing

2. Data Visualization

3. Data Splitting

4. Model Selection

5. Model Training

6. Model Evaluation

##### 

##### 2.1. Data collection and preprocessing :

Acquire historical records of transformer operational parameters, including oil levels, oil temperature, winding temperature, current and voltage levels This data should cover a sufficiently extended period to capture diverse operating conditions and potential fault scenarios.

**2.2. Data Visualization :**

Data Visualization involves creating graphical

representations of the dataset to extract meaningful insights. By visualising the data, the project team gains a better understanding of patterns, trends, and potential correlations between different variables. Visualisation aids in informed decisions about feature selection and model development.

**2.3. Data Splitting :**

It is one of the most crucial steps to assess the performance of machine learning models. This step involves dividing the preprocessed dataset into training and testing sets, allocating a significant portion for training the machine learning models and a separate portion for evaluation.

**2.4. Machine Learning Models :**

Machine learning models entail the process to decide on a machine learning model which most suits the application intended. Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbour (KNN), Decision Tree, Random Forest,Extra Tree Classifier, AdaBoost, XGB Classifier are few of the machine learning models which have been tested in this paper.

Furthermore to make the fault predictions robust we have implemented ensemble machine learning. Ensemble learning in machine learning is about combining various models to improve overall prediction accuracy and robustness. By leveraging the strengths of diverse models, ensemble techniques enhance performance, making predictions more reliable than individual models.

**2.5. Model Evaluation and Selection :**

Optimal model selection and evaluation revolve around the identification of the most fitting machine learning model, followed by a comprehensive assessment of its performance. Multiple machine learning models are to be selected and tested for their performance. Factors like accuracy, computational efficiency, and precision are considered while evaluating the performance of a model.

##### 2.7 Source codes

The codes are included below with the proper comments, describing the inputs, output, data transfer, and data acquisition rate.

2.7.1 Logistic Regression :

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay

from sklearn.linear\_model import LogisticRegression

import matplotlib.pyplot as plt

from matplotlib.Colours import ListedColormap

# Define a custom colormap with varied Colours

cmap = ListedColormap(['#0071C5', '#4EA2F2', '#A4C8FD', '#CDE3FD'])

logreg = LogisticRegression()

logreg.fit(X\_train, y\_train)

y\_pred\_lr = logreg.predict(X\_test)

log\_train = round(logreg.score(X\_train, y\_train) \* 100, 2)

log\_accuracy = round(accuracy\_score(y\_pred\_lr, y\_test) \* 100, 2)

print("Training Accuracy :", log\_train, "%")

print("Model Accuracy Score :", log\_accuracy, "%")

print("\033[1m--------------------------------------------------------\033[0m")

print("Classification\_Report: \n", classification\_report(y\_test,

y\_pred\_lr))

print("\033[1m--------------------------------------------------------\033[0m")# Calculate the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred\_lr)

# Display the confusion matrix

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=logreg.classes\_)

disp.plot(cmap=cmap)

plt.title('Confusion Matrix')

plt.show()

2.7.2 Support Vector Machine :

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

from matplotlib.Colours import ListedColormap

# Define a custom colormap with varied Colours

cmap = ListedColormap(['#FFA500', '#FFD700', '#FFA07A', '#FF4500'])

# Create and fit the SVM model

svc = SVC()

svc.fit(X\_train, y\_train)

y\_pred\_svc = svc.predict(X\_test)

svc\_train = round(svc.score(X\_train, y\_train) \* 100, 2)

svc\_accuracy = round(accuracy\_score(y\_pred\_svc, y\_test) \* 100, 2)

print("Training Accuracy :", svc\_train, "%")

print("Model Accuracy Score :", svc\_accuracy, "%")

print("\033[1m--------------------------------------------------------\033[0m")

print("Classification\_Report: \n", classification\_report(y\_test, y\_pred\_svc))

print("\033[1m--------------------------------------------------------\033[0m")

# Calculate the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred\_svc)

# Display the confusion matrix

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=svc.classes\_)

disp.plot(cmap=cmap)

plt.title('Confusion Matrix (SVM)')

plt.show()

2.7.3 K-Nearest Neighbours

from sklearn.neighbors

import KNeighborsClassifier

from sklearn.metrics import accuracy\_score, classification\_report,

confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

from matplotlib.Colours import ListedColormap

# Define a custom colormap with pink Colours

cmap\_pink = ListedColormap(['#FFC0CB', '#FF69B4', '#FF1493', '#DB7093'])

# Create and fit the KNN model

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(X\_train, y\_train)

y\_pred\_knn = knn.predict(X\_test)

knn\_train = round(knn.score(X\_train, y\_train) \* 100, 2)

knn\_accuracy = round(accuracy\_score(y\_pred\_knn, y\_test) \* 100, 2)

print("Training Accuracy :", knn\_train, "%")

print("Model Accuracy Score :", knn\_accuracy, "%")

print("\033[1m--------------------------------------------------------\033[0m")

print("Classification\_Report: \n", classification\_report(y\_test, y\_pred\_knn))

print("\033[1m--------------------------------------------------------\033[0m")

# Calculate the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred\_knn)

# Display the confusion matrix with a pink colour scheme

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=knn.classes\_)

disp.plot(cmap=cmap\_pink)

plt.title('Confusion Matrix (KNN)')

plt.show()

2.7.4 Decision Tree

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

from matplotlib.Colours import ListedColormap

# Define a custom colormap with purple colours

cmap\_purple = ListedColormap(['#F0E6F4', '#DDA0DD', '#800080', '#4B0082'])

# Create and fit the decision tree model

decision = DecisionTreeClassifier()

decision.fit(X\_train, y\_train)

y\_pred\_dec = decision.predict(X\_test)

decision\_train = round(decision.score(X\_train, y\_train) \* 100, 2)

decision\_accuracy = round(accuracy\_score(y\_pred\_dec, y\_test) \* 100, 2)

print("Training Accuracy :", decision\_train, "%")

print("Model Accuracy Score :", decision\_accuracy, "%")

print("\033[1m--------------------------------------------------------\033[0m")

print("Classification\_Report: \n", classification\_report(y\_test, y\_pred\_dec))

print("\033[1m--------------------------------------------------------\033[0m")

# Calculate the confusion matrix

cm = confusion\_matrix(y\_test, y\_pred\_dec)

# Display the confusion matrix with a pink colour scheme

disp = ConfusionMatrixDisplay(confusion\_matrix=cm, display\_labels=decision.classes\_)

disp.plot(cmap=cmap\_purple)

plt.title('Confusion Matrix (Decision Tree Classifier)')

plt.show()

2.7.5 Random Forest

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score,

classification\_report, confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

from matplotlib.Colours import ListedColormap

import pickle

# Define a custom colormap with purple Colours

cmap\_purple = ListedColormap(['#F0E6F4', '#DDA0DD', '#800080', '#4B0082'])

# Create and fit the Random Forest model

random\_forest = RandomForestClassifier(n\_estimators=100)

random\_forest.fit(X\_train, y\_train)

y\_pred\_rf = random\_forest.predict(X\_test)

random\_forest\_train = round(random\_forest.score(X\_train, y\_train) \* 100, 2)

random\_forest\_accuracy = round(accuracy\_score(y\_pred\_rf, y\_test) \* 100, 2)

print("Training Accuracy :", random\_forest\_train, "%")

print("Model Accuracy Score :", random\_forest\_accuracy, "%")

print("\033[1m--------------------------------------------------------\033[0m")

print("Classification\_Report: \n", classification\_report(y\_test, y\_pred\_rf))

print("\033[1m--------------------------------------------------------\033[0m")

# Calculate the confusion matrix

cm\_rf = confusion\_matrix(y\_test, y\_pred\_rf)

# Display the confusion matrix with a pink colour scheme

disp\_rf = ConfusionMatrixDisplay(confusion\_matrix=cm\_rf, display\_labels=random\_forest.classes\_)

disp\_rf.plot(cmap=cmap\_purple)

plt.title('Confusion Matrix (Random Forest Classifier)')

plt.show()

2.7.6 Extra Trees

from sklearn.ensemble import ExtraTreesClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

from matplotlib.Colours import ListedColormap

import pickle

# Define a custom colormap with shades of olive green

cmap\_olive = ListedColormap(['#556B2F', '#6B8E23', '#808000', '#8FBC8F'])

# Create and fit the Extra Trees model

et = ExtraTreesClassifier(n\_estimators=100)

et.fit(X\_train, y\_train)

y\_pred\_et = et.predict(X\_test)

et\_train = round(et.score(X\_train, y\_train) \* 100, 2)

et\_accuracy = round(accuracy\_score(y\_pred\_et, y\_test) \* 100, 2)

print("Training Accuracy :", et\_train, "%")

print("Model Accuracy Score :", et\_accuracy, "%")

print("\033[1m--------------------------------------------------------\033[0m")

print("Classification\_Report: \n", classification\_report(y\_test, y\_pred\_et))

print("\033[1m--------------------------------------------------------\033[0m")

# Calculate the confusion matrix

cm\_et = confusion\_matrix(y\_test, y\_pred\_et)

# Display the confusion matrix with shades of olive green

disp\_et = ConfusionMatrixDisplay(confusion\_matrix=cm\_et, display\_labels=et.classes\_)

disp\_et.plot(cmap=cmap\_olive)

plt.title('Confusion Matrix (Extra

Trees Classifier)')

plt.show()

2.7.7 AdaBoost

from sklearn.ensemble import AdaBoostClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

from matplotlib.Colours import ListedColormap

import pickle

# Define a custom colormap with shades of grey

cmap\_grey = ListedColormap(['#F0F0F0', '#D3D3D3', '#A9A9A9', '#808080'])

# Create and fit the AdaBoost model

ada = AdaBoostClassifier()

ada.fit(X\_train, y\_train)

y\_pred\_ada = ada.predict(X\_test)

ada\_train = round(ada.score(X\_train, y\_train) \* 100, 2)

ada\_accuracy = round(accuracy\_score(y\_pred\_ada, y\_test) \* 100, 2)

print("Training Accuracy :", ada\_train, "%")

print("Model Accuracy Score :", ada\_accuracy, "%")

print("\033[1m--------------------------------------------------------\033[0m")

print("Classification\_Report: \n", classification\_report(y\_test, y\_pred\_ada))

print("\033[1m--------------------------------------------------------\033[0m")

# Calculate the confusion matrix

cm\_ada = confusion\_matrix(y\_test, y\_pred\_ada)

# Display the confusion matrix with shades of grey

disp\_ada = ConfusionMatrixDisplay(confusion\_matrix=cm\_ada, display\_labels=ada.classes\_)

disp\_ada.plot(cmap=cmap\_grey)

plt.title('Confusion Matrix (AdaBoost Classifier)')

plt.show()

2.7.8 XGBoost

import xgboost as Xgb

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix, ConfusionMatrixDisplay

import matplotlib.pyplot as plt

from matplotlib.Colours import ListedColormap

import pickle

# Define a custom colormap with a unique colour

cmap\_unique = ListedColormap(['#FF6347', '#00FA9A', '#4682B4', '#FFD700'])

# Create and fit the XGBoost model

xgb = Xgb.XGBClassifier()

xgb.fit(X\_train, y\_train)

y\_pred\_xgb = xgb.predict(X\_test)

xgb\_train = round(xgb.score(X\_train, y\_train) \* 100, 2)

xgb\_accuracy = round(accuracy\_score(y\_pred\_xgb, y\_test) \* 100, 2)

print("Training Accuracy :", xgb\_train, "%")

print("Model Accuracy Score :", xgb\_accuracy, "%")

print("\033[1m--------------------------------------------------------\033[0m")

print("Classification\_Report: \n", classification\_report(y\_test, y\_pred\_xgb))

print("\033[1m--------------------------------------------------------\033[0m")

# Calculate the confusion matrix

cm\_xgb = confusion\_matrix(y\_test, y\_pred\_xgb)

# Display the confusion matrix with a unique colour

disp\_xgb = ConfusionMatrixDisplay(confusion\_matrix=cm\_xgb,

display\_labels=xgb.classes\_)

disp\_xgb.plot(cmap=cmap\_unique)

plt.title('Confusion Matrix (XGBoost Classifier)')

plt.show()

**3. RESULTS AND DISCUSSION:**

##### 3.1 Results

Several scenarios are used to test the smart farm monitoring system. The soil is tested for moisture under different climatic situations using a soil moisture sensor, and the results are properly interpreted. Specific Arduino code is written and imported into the microcontroller for sensing soil moisture, pH, and temperature. The wireless transmission is accomplished using Wi- Fi. The output values of the soil moisture sensor only depend on the soil’s resistivity. The sensor’s value at the start of a wet condition is 0. When this occurs, the motor pump turns off, and the measured value is communicated to the microcontroller through NodeMCU. The microprocessor activates the relay, and the motor turns on when the sensed value from the sensor surpasses the threshold value. The motor pump automatically turns ON and OFF when plants receive enough water. Below the given tables, Table 1, Table 2, Table 3, and Table 4, show the test result we obtained after a successful system test. Here the temperature, humidity, and soil moisture are measured in terms of degrees Celsius, and if the object is detected or not by the PIR sensor, the on/off status of the water motor and the buzzer will be displayed on the mobile screen of the user. To display content on the mobile application, the device first goes online after being turned on. After starting up quickly (within 10 to 15 seconds), it immediately begins sending data continually to mobile devices. After a system test was completed successfully, the test results are shown in the tables below. Here, the user’s mobile device will display the temperature in degrees Celsius, humidity, and soil moisture in percentages, the object’s PIR sensor’s ability to detect it or not, the water motor’s on/off status, and the buzzer’s st

##### Discussion

Figure 8 shows that as soon as the temperature threshold of 19.5’C is achieved, the water pump starts automatically to water the plants. Normally during the day time when the temperature is very high.

According to Figure 9, it has been observed that as humidity increases, it is not necessary to water the plants as the plants take up water vapor from the surface of the leaves under humid conditions.

From the results of Table 3 depicted in Figure 10, we conclude that with the decrease in moisture of the soil, the plants need to be watered.

After carefully examining the data from the experiment, it was concluded that the plants require watering when the temperature is high, the humidity is low, and the moisture content is low. The overall outcome of the data collected in January 2023 is displayed in Figure 11.

**4. CONCLUSIONS :**

In conclusion, this research has illuminated the potential of machine learning in revolutionising the predictive maintenance landscape for power transformers. By scrutinising historical operational data and employing a diverse array of machine learning algorithms, we endeavoured to create predictive models capable of discerning patterns indicative of potential faults. Our findings highlight the promise of machine learning in enhancing the reliability and efficiency of power distribution systems.

The results of our study emphasise the significant impact of adopting predictive maintenance strategies for transformers. The ability to foresee and address potential faults before they escalate is pivotal for minimising downtime, optimising resource utilisation, and ensuring the longevity of critical components within our electrical infrastructure.

Our proposed methodology, which operates without the constraints of real-time monitoring, presents a pragmatic and cost-effective solution applicable to various operational settings. This adaptability positions our approach as a viable option for widespread adoption, particularly in environments where real-time monitoring infrastructure may pose challenges.

Looking ahead, this research sparks the beginning of a new era in predictive maintenance for power transformers. The insights gained provide a foundation for continued refinement of fault prediction models, paving the way for more proactive and resilient electrical distribution systems. As our reliance on electricity grows, the significance of predictive maintenance, bolstered by machine learning, becomes increasingly vital for ensuring the seamless operation of power grids in the face of evolving challenges.

**5. Declarations :**

**5.1 Study Limitation :**

**5.2 Acknowledgement :**

**5.3 Funding Source :**

**5.4 Competing Interests :**

**5.5 Open Access :**

**6. References**