**Transformer Fault Prediction using Machine Learning**

**Devaansh Kakeri**

*Student, Department of Electronics and Telecommunication Engineering, Dr. D.Y. Patil Institute of Technology, Pune, India.*

**Abhishek Agasti**

*Student, Department of Electronics and Telecommunication Engineering, Dr. D.Y. Patil Institute of Technology, Pune, India.*

**Mrs. Vaishali Biradar**

*Assistant Professor, Department of Electronics and Telecommunication Engineering, Dr. D.Y. Patil Institute of Technology, Pune, India.*

*\*Corresponding author*

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

**ABSTRACT**

Distribution Transformers are an integral parts of the 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 analyzing 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 emphasizes the development of a predictive maintenance system that can identify potential transformer issues before they escalate, thereby minimizing 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 optimize maintenance activities.

The significance of this paper lies in its potential to offer a practical and cost-effective alternatives 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.

**1.1 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 categorizes 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 minimize damages and prevent instability. The paper also identifies research gaps, emphasizing 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 privatized and restructured power systems, emphasizing 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 specialized 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 optimize transformer maintenance through Machine Learning-based fault prediction. The study utilized three-phase current and voltage measurements as a training dataset for various algorithms, including K-Nearest Neighbor and Decision Trees. Evaluation criteria focused on accuracy metrics and cross-validation techniques to enhance the robustness of the chosen algorithms, ultimately identifying K-Nearest Neighbor and Decision Trees as the most accurate predictors. The pros of the methodology include its cost-effective alternative to traditional maintenance methods, utilization 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 Neighbor and Decision Trees. Cross-validation techniques were highlighted for improving algorithm generalization 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 emphasizes the importance of standardized 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 analyzing transformer fault evolution and establishing a robust forecasting model using multiple linear regression. The study's pros include a systematic approach to analyze 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 for model 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 Janarthanan 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 utilization 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 generalization across diverse configurations, and addressing potential ethical concerns and biases in algorithm deployment. The paper emphasizes 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 utilizing 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 generalization 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 endeavors 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 followed:-

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 visualizing the data, the project team gains a better understanding of patterns, trends, and potential correlations between different variables. Visualization 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 Neighbor (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:-**

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**

**Dataset used:-**

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameters** | **Classes** | **Training Slices** | **Testing Slices** |
| Kaggle | 2 | 16258 | 4063 |

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.colors import ListedColormap

# Define a custom colormap with varied colors

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()

|  |  |  |
| --- | --- | --- |
|  | **Actual 0** | **Actual 1** |
| **Predicted 0** | 3666 | 18 |
| **Predicted 1** | 216 | 193 |

**2.7.2 Support Vector Machine (SVM):-**

from sklearn.svm import SVC

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

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap

# Define a custom colormap with varied colors

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\_tain) \* 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()

|  |  |  |
| --- | --- | --- |
|  | **Actual 0** | **Actual 1** |
| **Predicted 0** | 3655 | 29 |
| **Predicted 1** | 153 | 256 |

**2.7.3 K-Nearest Neighbors (KNN):-**

from sklearn.neighbors import KNeighborsClassifier

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

import matplotlib.pyplot as plt

from matplotlib.colors import ListedColormap

# Define a custom colormap with pink colors

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 color scheme

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

disp.plot(cmap=cmap\_pink)

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

plt.show()

|  |  |  |
| --- | --- | --- |
|  | **Actual 0** | **Actual 1** |
| **Predicted 0** | 3597 | 87 |
| **Predicted 1** | 98 | 311 |

**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.colors import ListedColormap

# Define a custom colormap with purple colors

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 color scheme

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

disp.plot(cmap=cmap\_purple)

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

plt.show()

|  |  |  |
| --- | --- | --- |
|  | **Actual 0** | **Actual 1** |
| **Predicted 0** | 3653 | 31 |
| **Predicted 1** | 30 | 379 |

**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.colors import ListedColormap

import pickle

# Define a custom colormap with purple colors

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 color 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()

|  |  |  |
| --- | --- | --- |
|  | **Actual 0** | **Actual 1** |
| **Predicted 0** | 3662 | 22 |
| **Predicted 1** | 20 | 389 |

**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.colors 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()

|  |  |  |
| --- | --- | --- |
|  | **Actual 0** | **Actual 1** |
| **Predicted 0** | 3662 | 22 |
| **Predicted 1** | 18 | 391 |

**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.colors 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()

|  |  |  |
| --- | --- | --- |
|  | **Actual 0** | **Actual 1** |
| **Predicted 0** | 3630 | 54 |
| **Predicted 1** | 35 | 374 |

**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.colors import ListedColormap

import pickle

# Define a custom colormap with a unique color

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 color

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()

|  |  |  |
| --- | --- | --- |
|  | **Actual 0** | **Actual 1** |
| **Predicted 0** | 3659 | 25 |
| **Predicted 1** | 19 | 390 |

**3. Results and Discussion**

**3.1 Observation Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Parameters | Accuracy | | Precision (1) | Recall (1) | F1 Score |
| Training | Testing |
| 1. Logistic Regression | 94.24% | 94.23% | 0.91 | 0.47 | 0.62 |
| 1. Support Vector Machine (SVM) | 95.41% | 95.55% | 0.90 | 0.63 | 0.74 |
| 1. K-Nearest Neighbours | 98.08% | 95.48% | 0.78 | 0.76 | 0.77 |
| 1. Decision Tree Classifier | 100% | 98.46% | 0.92 | 0.92 | 0.92 |
| 1. Random Forest Classifier | 100% | 98.97% | 0.94 | 0.95 | 0.95 |
| 1. Extra Trees Classifier | 100% | 99.10% | 0.96 | 0.95 | 0.95 |
| 1. AdaBoost | 98.20% | 97.83% | 0.87 | 0.91 | 0.89 |
| 1. XGBoost | 100% | 98.92% | 0.94 | 0.95 | 0.95 |

1. **DISCUSSION**

Principal Component Analysis (PCA) is a statistical method used for dimensionality reduction in data analysis and machine learning. It transforms high-dimensional data into a lower-dimensional space while retaining as much of the original variability as possible. PCA is utilized for feature selection to reduce computational complexity, filter out noise, and identify the most informative features. It achieves this by decomposing the covariance matrix of the data into eigenvectors and eigenvalues, selecting principal components based on their corresponding eigenvalues, and projecting the original features onto these components to obtain a subset of features that capture the most significant patterns in the data.

In this study, we employed Principal Component Analysis (PCA) to discern the most influential features impacting the performance of transformers. Through PCA, we identified the crucial parameters that significantly affect transformer operation: Current Line 2 (IL2), Oil Level Indicator (OLI), Oil Temperature Indicator (OTI), and Phase Line 1 (VL1). These parameters play pivotal roles in ensuring the safe and efficient functioning of transformers. Our analysis revealed ideal ranges for these parameters, optimizing transformer operation and minimizing the risk of malfunctions. For instance, maintaining IL2 within the range of -25.83A to 138.91A ensures proper current flow, while keeping OTI between 8.66 and 51.51 maintains optimal oil temperature.

**Ideal Ranges Table**:-

|  |  |
| --- | --- |
| IL2 (Current Line 2) | -25.83A - 138.91A |
| OTI (Oil Temperature Indicator) | 8.66 - 51.51 |
| VL1 (Phase Line 1) | 222.57V - 259.54V |
| OLI (Oil Level Indicator) | 14.13% - 125.24% |

Furthermore, to determine the most effective predictive models for transformer fault detection, we evaluated five classifiers based on their accuracy. These models include Extra Trees Classifier (ETC), AdaBoost (AB), XGBoost (XGB), Support Vector Machine (SVM), and K-Nearest Neighbors (KNN). Notably, despite achieving 100% training accuracy across all models, Extra Trees Classifier emerged as the top performer in terms of testing accuracy. This selection is rationalized by the ensemble nature of ETC, which combines the robustness of decision trees and random forests. By leveraging the strengths of various models, ETC demonstrated superior generalization capability, making it an optimal choice for transformer fault prediction.   
  
Ensemble learning theory encompasses various methods such as bagging, boosting, voting, and stacking, each aimed at improving predictive accuracy by combining the outputs of multiple models. In our project, we utilize the voting method, which aggregates predictions from diverse models to determine the probability of fault occurrence and its severity. By selecting models with varying accuracy scores, we ensure a more robust and reasonable output, enhancing the reliability of our predictions.

To showcase our approach, we implemented the models and saved them as pickle files for deployment. Leveraging Streamlit, we developed an interactive web page where users can input relevant data and receive real-time predictions on fault probability and severity. This interactive interface enhances user experience and facilitates seamless interaction with the predictive model.

Looking forward, we envision expanding our project's scope to integrate real-time IoT devices using APIs. By directly interfacing with IoT devices, we can gather data streams in real-time, allowing for timely and proactive fault detection and management. This future enhancement will not only improve the responsiveness of our system but also enable predictive maintenance, reducing downtime and optimizing operational efficiency.

Overall, our findings underscore the criticality of feature selection and model evaluation in transformer maintenance. By pinpointing key parameters and selecting the most accurate predictive model, our study contributes to enhancing the reliability and efficiency of transformer operations in various industrial settings.

**4. CONCLUSIONS:-**

In conclusion, this research represents a significant endeavor to address the critical need for proactive measures in ensuring the reliable operation of distribution transformers within electrical distribution systems. The increasing complexity and importance of power infrastructure necessitate innovative approaches to fault prediction and maintenance, particularly in mitigating the adverse effects of transformer failures. By leveraging the power of machine learning, this study proposes a novel framework for predictive maintenance that harnesses historical operational data and periodic measurements to anticipate potential transformer faults.

The significance of this approach lies in its capacity to bridge the gap between traditional maintenance practices and the evolving landscape of condition-based monitoring. Through comprehensive analysis of historical data, load profiles, environmental factors, and diagnostic test results, machine learning algorithms can uncover hidden patterns and correlations indicative of impending faults. By developing predictive models capable of preemptively identifying transformer issues, utilities and power system operators can proactively address potential failures, thereby minimizing downtime and reducing maintenance costs.

The methodologies explored in this research encompass a diverse array of machine learning techniques, ranging from classical algorithms to more advanced models. By evaluating and selecting the top-performing models based on accuracy metrics, including Extra Trees Classifier, AdaBoost, XGBoost, Support Vector Machine, and K-Nearest Neighbors, we ensure the robustness and effectiveness of the predictive maintenance system. Notably, the superiority of the Extra Trees Classifier in testing accuracy underscores its suitability for real-world application in transformer fault prediction.

Furthermore, the outcomes of this research have broader implications for the field of predictive maintenance in power systems. By demonstrating the efficacy of machine learning in addressing critical infrastructure challenges, this study contributes to the advancement of resilient electrical distribution systems. The proposed framework offers utilities and industries an invaluable tool for improving the reliability and longevity of transformers, aligning with the industry's shift towards smart grids and proactive maintenance strategies.

In essence, this research represents a significant step towards enhancing the resilience and efficiency of power distribution systems through the integration of machine learning techniques. By leveraging historical data and advanced predictive models, utilities and power system operators can proactively identify and mitigate potential transformer faults, thereby ensuring the uninterrupted and reliable supply of electricity to consumers. Moving forward, continued research and implementation efforts in this area hold the promise of further enhancing the reliability and resilience of critical infrastructure systems worldwide.

**5. Declarations :**

**5.1 Study Limitation :**

**5.2 Acknowledgement :**

**5.3 Funding Source :**

**5.4 Competing Interests :**

**5.5 Open Access :**

**6. References**