Uniport Journal of Engineering and Scientific Research (UJESR)

Vol. 8, Issue 1, 2023, Page 190-198
ISSN: 2616-1192

(www.ujesr.org)

Transformer Predictive Maintenance Using Artificial Intelligence

Emmanuel, A.U
Department of Electrical and Electronic Engineering, Federal University Otuoke, Nigeria
Corresponding author's email: amasaeu@fuotuoke.edu.ng

Abstract
The proper functioning of transformers is crucial for the stable operation of power systems. However, the risk of unexpected failures poses significant challenges in ensuring uninterrupted power supply. Predictive maintenance techniques leveraging on Artificial Intelligence (AI) have emerged as a promising solution to address these challenges by predicting potential faults in transformers, thereby enabling proactive maintenance. This paper investigated and proposed a predictive maintenance framework for transformers utilizing AI techniques. The study encompassed the utilization of machine learning algorithms such as neural networks, support vector machines, and decision trees, to analyze historical data collected from transformers. The proposed framework integrates advanced AI algorithms to perform predictive analytics on the gathered data. By employing these algorithms, the model learns patterns and trends, enabling it to forecast potential faults or anomalies in transformer behaviour. Moreover, the system continuously adapts and improves its predictions through iterative learning, enhancing its accuracy and reliability over time. The implementation of this predictive maintenance system offers several advantages, including minimized downtime, improved safety, cost savings, and optimized maintenance schedules. By detecting potential issues in advance, utilities and maintenance teams can take preemptive actions, thereby preventing catastrophic failures and extending the lifespan of transformers. The proposed framework showcased promising prospects in revolutionizing maintenance practices, enhancing reliability, and ensuring the efficient performance of power systems.

Keywords: *Predictive maintenance, Transformer monitoring, Artificial intelligence, Machine learning*

Received: 10th December, 2023 *Accepted: 31st December, 2023*

1. Introduction

Transformers are critical components in power systems, facilitating the transmission and distribution of electrical energy. However, their reliability can be compromised by various factors such as aging, overloading, and environmental stresses. Unplanned downtime due to transformer failures not only results in substantial economic losses but also poses significant safety risks. Traditional maintenance approaches based on fixed schedules or condition monitoring often lack precision and efficiency, leading to suboptimal asset management.

The emergence of artificial intelligence (AI) has revolutionized maintenance strategies, particularly in the field of predictive maintenance. By harnessing the power of machine learning algorithms and data analytics, AI enables the development of sophisticated predictive models capable of forecasting equipment failures before they occur. In the context of transformers, implementing AI-driven predictive maintenance holds immense promise in enhancing reliability, minimizing downtime, and optimizing maintenance schedules.

Numerous studies such as Chen et al. (2022), Quiñones et al. (2022) and Vita et al. (2023) have highlighted the efficacy of AI techniques, such as neural networks, support vector machines, and deep learning, in predicting faults and degradation patterns in transformers. By leveraging on historical operational data, including load profiles, temperature variations, and insulation conditions, AI algorithms can learn complex patterns and correlations, enabling early detection of potential issues.

Hashemnia et al. (2015), carried out a study on how to improve power transformer winding fault detection using frequency response analysis (FRA) technique. The findings of this study support accurate qualitative and quantitative analysis of transformer FRA signatures by precisely simulating

winding axial displacement using transformer equivalent circuit.

An overview of fault diagnosis and condition monitoring methods for power transformers is provided in Zhang et al. (2016). It covers both cutting-edge methods like data mining, pattern recognition, and AI-based approaches, as well as conventional approaches like vibration analysis, partial discharge measurement, and dissolved gas analysis. The research addressed the benefits and drawbacks of each method and stress on the significance of combining various monitoring approaches for an all-encompassing evaluation of transformer health.

Chen et al. (2022) proposed the use of artificial intelligence (AI) methods for power transformer predictive maintenance. The research covered various AI algorithms used for fault diagnosis, remaining useful life estimation, and maintenance optimization, such as machine learning, deep learning, and evolutionary algorithms. The authors discussed opportunities and challenges in implementing AI-based predictive maintenance systems, as well as data preprocessing, feature extraction, and model training.

In light of these, the purpose of this study is to examine the use of AI-based predictive maintenance methods designed especially for transformers. This study aims to create a predictive maintenance framework that can accurately forecast transformer faults and degradation, enabling proactive maintenance interventions. It does this by combining real-time sensor data, historical maintenance records, and sophisticated machine learning algorithms. This research is important because it has the potential to enhance maintenance procedures in a way that will improve power systems' operational efficiency, cost-effectiveness, and reliability.

2. Materials and methods

2.1 System design

In this study, the waterfall design model was applied (Wojcicki and Dabrowski, 2018). The waterfall model is a sequential development approach where software design, implementation, testing, integration, deployment (or installation), and maintenance are seen as phases that development flows through gradually, like a waterfall as shown in Fig. 1. The methodology involved carrying out multiple experiments on datasets by utilising the Random Forest, Linear Regression, Naïve Bayes, and Support Vector Machine (SVM) algorithms. To achieve the best accuracy and precision, experiments were run on each algorithm separately as well as in combinations. The processes used to predict faults during the model-creation process involved first the gathering of the datasets, and data preprocessing was done, after which the Natural Language Toolkit was used to extract the data features, and finally the dataset was split and machine learnng algorithms used to produce the suggested classifier model. Following the application of Naïve Bayes, Random Forest, Binomial Linear Regression, and Support Vector Machine to the system response, a model containing the response message was generated. After testing was completed on a test dataset and the results confirmed, the precision was observed to ensure acceptance.

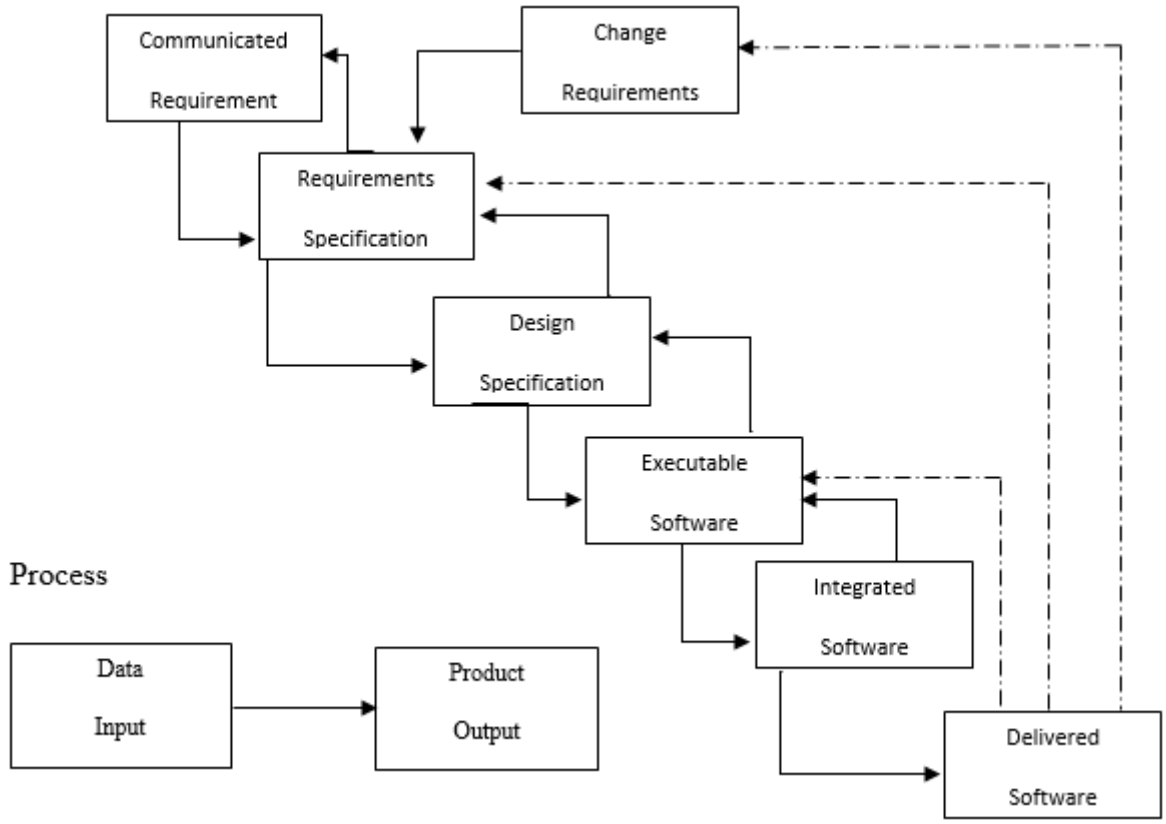


Fig. 1: System design methodology

2.2 Parameters to classify the transformer fault

In order to create a classification model that ranks the faults according to severity, a set of classification algorithms was applied. The model was then embedded in a Python application in order to forecast transformer faults. Support Vector Machine (SVM), Naive Bayes, Random Forests, and Binomial Linear Regression were the classification algorithms used in this model. When these algorithms were combined, averaged, and compared, they all produced results that are as reliable and accurate as possible. The dataset was ran through several algorithms to find errors. To arrive at the final result, the accuracy of the data gathered from each of them was examined as shown in Fig. 2.

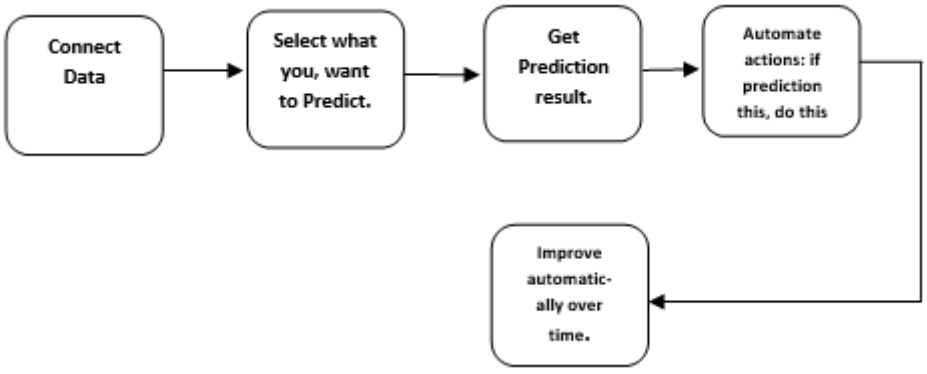


Fig. 2: The classification algorithms

Fig. 3 illustrates how the dataset was successfully preprocessed in the system following the application of the Natural Language Toolkit (NLTK), and how a message was generated for the purpose of applying algorithms on the trained portion. Following the application of Naïve Bayes, Random Forest, Binomial Linear Regression, and Support Vector Machine to the system response, a model containing the response message was generated. After testing was completed on a test

dataset and the results were confirmed, the precision was observed to ensure acceptance. Next, the model was applied to user-selected, unseen data. The model's reset accuracy was 50% since the full dataset was generated with half of the data being accurate and the other half not. From the accurate and inaccurate datasets, 80% of the data were chosen at random and used in the full dataset; the remaining 20% were set aside and used as a testing set once the model was done. Prior to applying a classifier to text data, preprocessing was necessary. Unnecessary characters were removed using the Natural Language Toolkit (NLTK). The resulting data was then encoded as integers and floating-point values in order for machine learning algorithms to accept it as an input.

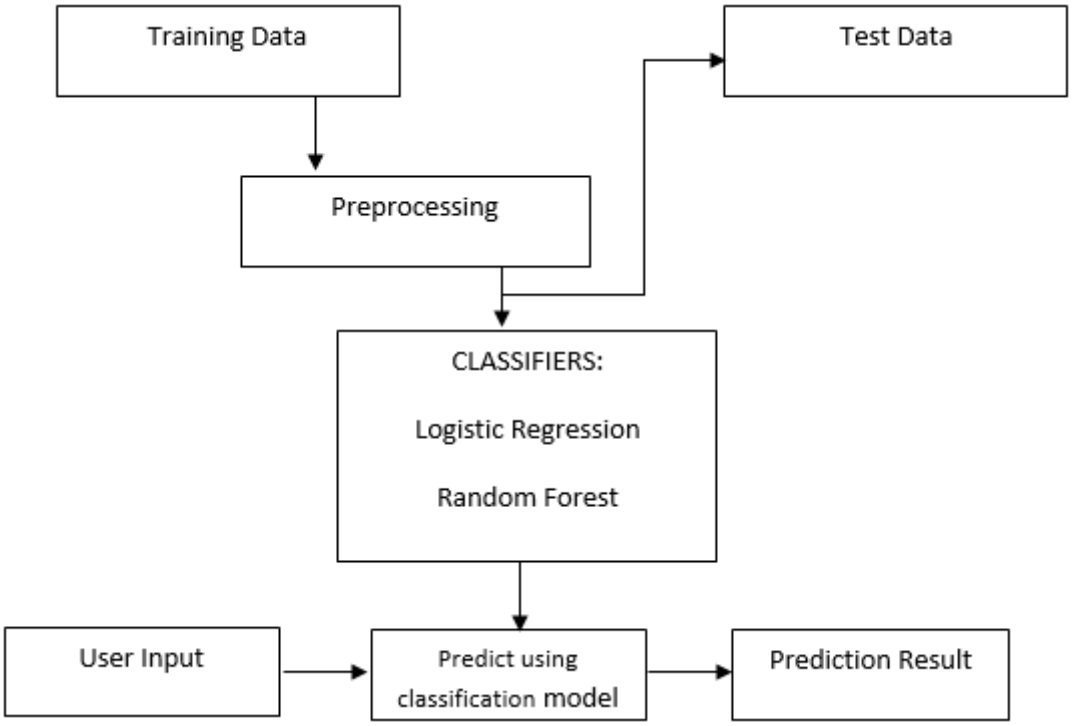


Fig. 3: Model of proposed system

2.3 Proposed system design

The suggested predictive fault classification system was designed as shown in Fig. 4. The dataset was first divided into training and testing of datasets in a comma-separated value (.csv) file after being pre-processed. The dataset was used for features extraction, and then followed by training the model with the training dataset. Finally, the title of the article obtained from the system's user interface was used for testing.

Fig. 4: The workflow of the system

2.4 Execution

Now that the designs and different machine learning-based predictive maintenance techniques for transformers have been shown. Next is to demonstrate the system's implementation in Python using two different approaches: analysis and design. To create the data needed to train the model, a script was written. The script creates a spreadsheet file with a CSV file extension, using pandas and the random function to generate random values for temperature and current, and adds a column to identify the fault. According to designated ranges for normal, overheating, short circuit, and oil leakage faults were labeled as shown in Fig. 5 and

Fig. 6. A total of 8000 data points are generated, with 2000 samples for each fault. The spreadsheet of the results is displayed in Fig. 6.

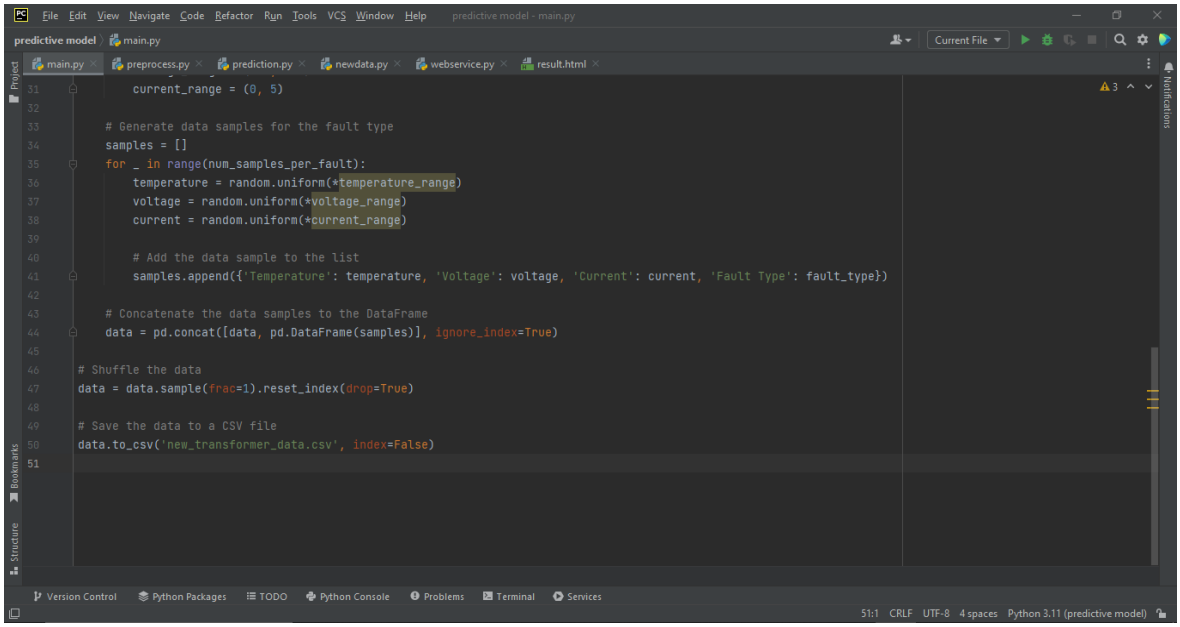


Fig. 5: Python script of trained model

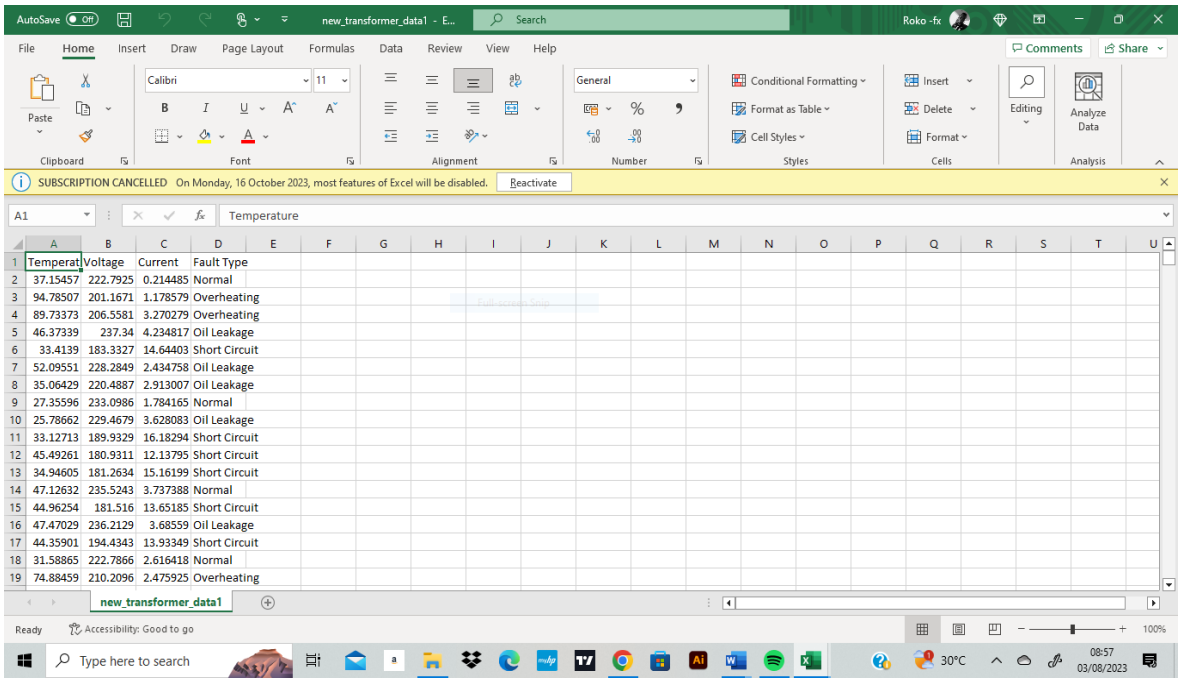


Fig. 6: Generated spreadsheet of trained model

2.5 Processing and training the datasets

Following the creation of the datasets, the data must be processed. To do this, the file must be read from its directory using the pandas function. All necessary processing packages, including tensor flow, sklearn. model_selection, and sklean. processing, were included. The processing steps involved importing the necessary processing packages, loading the data file, and separating the feature and target variable.

Step 1: The target variable must be encoded.

```
# Step 1: Encode the target variable
label_encoder = LabelEncoder()
target = label_encoder.fit_transform(target)
```

Step2: Scale the features to a specific range.

```
# Step 2: Scale the features to a specified range (e.g., 0 to 1)
scaler = MinMaxScaler()
features = scaler.fit_transform(features)
```

Step 3: Split the data into training and testing sets.

```
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, t
```

Step 4: Define the neural network architecture.

```
# Define the neural network architecture
model = tf.keras.Sequential([
    tf.keras.layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(len(label_encoder.classes_), activation='softmax')
])
```

Step 5: Compile the model.

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
# Compile the model
```

Step 6: Train model

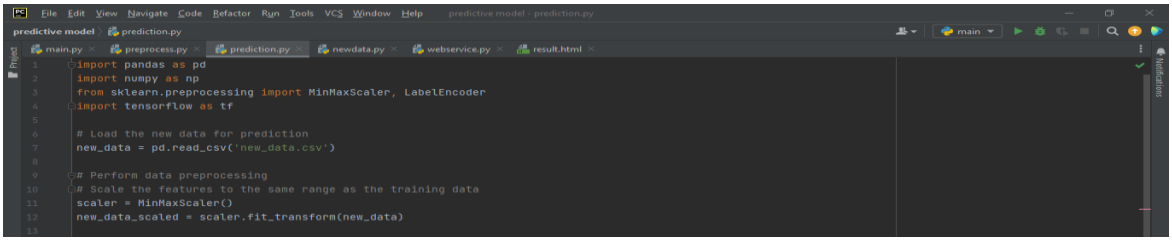
```
# Train the model
model.fit(X_train, y_train, validation_split=0.2, epochs=10, batch_size=32)
```

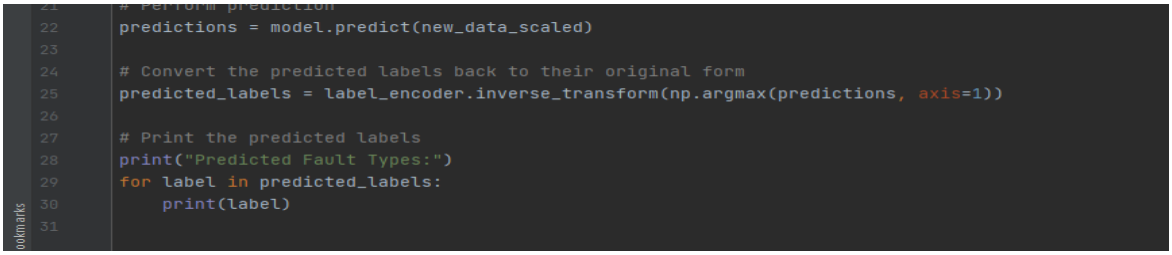
Step 7: Save the trained model.

```
#save the trained model
model.save('transformer_model.h5')
# Evaluate the model on the testing set
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')
```

2.6 Prediction with the new trained model

Following processing, a new set of fault data was added to the model so that it could predict faults using the trained model file. The prediction script is displayed below. The new data file is first read and processed to make it appropriate for the prediction model, after which all necessary modules are included. Following that, the new data was sent for prediction, the trained model was loaded, and the various faults were defined. Eventually, the newly created fault data file is read, examined, and fed into the prediction model. The outcomes are then displayed in the run terminal.





3. Results

A series of tests were conducted to assess the system performance, and the trained model was obtained by training on various data sizes. Once obtained, the accuracy of the model's predictions was recorded as shown in Tab. 1. A different model was created to test a new set of data, and the results were plotted using a pie chart, with images for models 1 through 5 displayed in that order, as can be seen from Fig. 7 to Fig. 11.

Fig. 7 shows the results of the first test category, which identified the various transformer defects. Historical data on various transformer fault categories was used to train the model. Following the test's execution the model identified a 10% oil leakage fault, which normally should be at 18% under typical circumstances. The model also detected 28% of overheating fault and 44% of short circuit fault.

Table 1: Generated result from the model

S/N	Size of Data Sample	Model generated	Test loses/accuracy (%)
1	8000	Transformer_model1.h5	Loss:0.0040, accuracy:1.0000
2	12000	Transformer_model2.h5	Loss:0.0019, accuracy:1.0000
3	16000	Transformer_model3.h5	Loss:0.0008, accuracy:1.0000
4	20000	Transformer_model4.h5	Loss:0.0011, accuracy:1.0000
5	24000	Transfomer_model5.h5	Loss: 0.0007, accuracy:0.9998

Fig. 8 shows the results of the second test, which identified the various transformer defects. Historical data on various transformer fault categories was used to train the model. Following the test's execution the model identified a 12% oil leakage fault, which usually should be at 18%. The model also detected 28% of overheating fault and 42% of short circuit fault. Fig. 9 shows the results of the third test, which identified the various transformer defects. Historical data on various transformer fault categories was used to train the model. Following the test's execution the model identified a 14% oil leakage fault, which normally should be at 16%. The model also detected 26% of overheating fault and 44% of short circuit fault. Fig. 10 shows the results of the fourth test category, which identified the various transformer defects. Historical data on various transformer fault categories was used to train the model. Following the test's execution the model identified a 16% oil leakage fault, which normally should be at 18% under typical circumstances. The model also detected 26% of

overheating fault and 40% of short circuit fault. Fig. 11 shows the results of the fifth test, which identified the various transformer defects. Historical data on various transformer fault categories was used to train the model. Following the test's execution the model identified a 14% oil leakage fault, which usually should be at 20%. The model also detected 26% of overheating fault and 40% of short circuit fault.

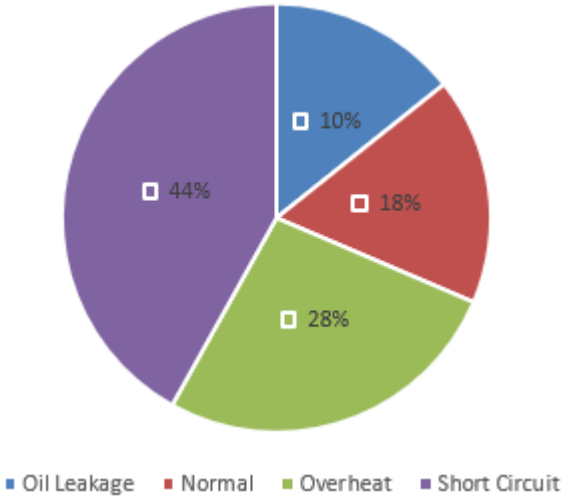


Fig. 7: Fault occurrences for first test

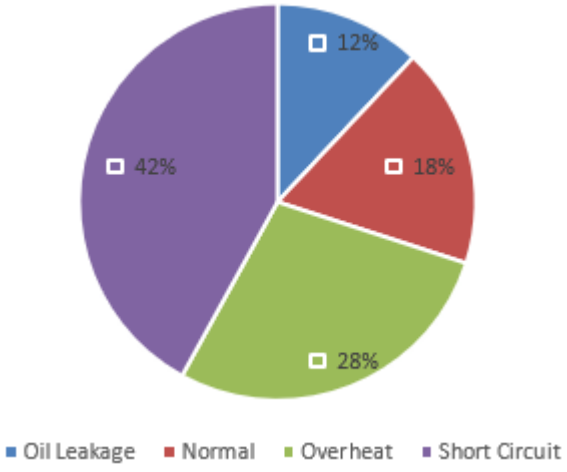


Fig. 8: Fault occurrences for second test

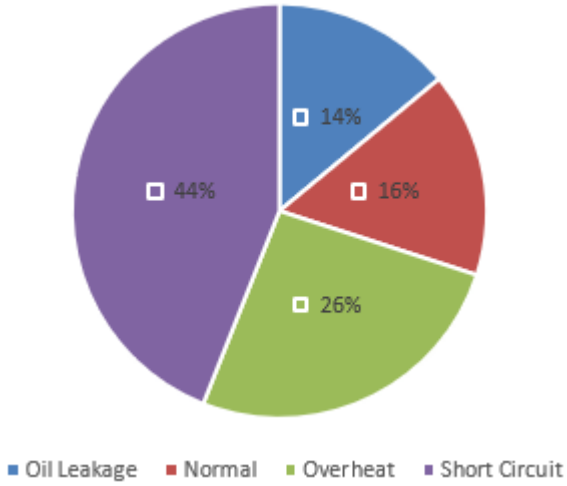


Fig. 9: Fault occurrences for third test

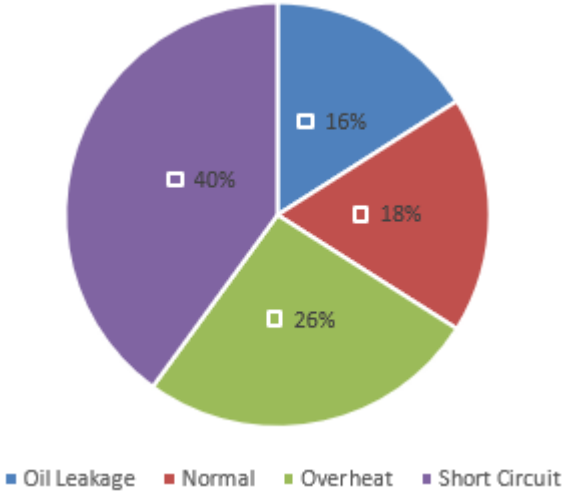


Fig. 10: Fault occurrences for fourth test

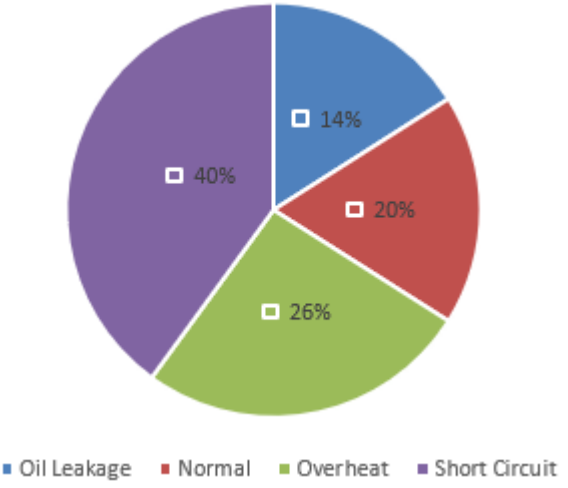


Fig. 11: Fault occurrences for fifth test

4. Discussion

The presented results show a range of outputs produced by all trained models from the first to the fifth iteration. These minute variations highlight how accurate the data-trained models are at predicting malfunctions. Furthermore, a pie chart plot effectively displays the most common issues that the transformer under review faced, as determined by the dataset. It is possible to improve the model's performance by continuing to train it further—a tactic that can be used on a transformer in the real world. In order to gather crucial data for this implementation, sensors and a microprocessor/micro-controller are deployed. This procedure will therefore produce the required data for improving the model's predictive abilities, leading to more precise fault prognosis and improved transformer maintenance procedures.

5. Conclusion

Significant improvements in fault detection and prevention are achieved when artificial intelligence (AI) is used for predictive maintenance of transformers. In order to accurately predict transformer faults, this study has shown how to successfully implement a number of classification algorithms, including Naïve Bayes, Random Forests, Binomial Linear Regression, and Support Vector Machine (SVM). One can proactively identify possible problems with transformers by utilizing AI, which enables prompt interventions and maintenance procedures. From dataset collection and preprocessing to feature extraction using the Natural Language Toolkit (NLTK), the model creation process involved a number of meticulous steps. Combining these procedures with the deliberate choice and implementation of algorithms resulted in the creation of a strong classifier model. The model's accuracy and precision were demonstrated through testing on specialized datasets, giving the confidence to rely on its predictions. Within the field of artificial intelligence (AI) predictive maintenance for transformers, this study has clarified the importance and possible advantages of utilizing AI-based techniques to protect the reliability and effectiveness of transformers.

References

Chen, Z., Zhou, Q., Du, B., Zhang, Y. and Ding, L. (2022) Electromagnetic Transient Calculation and Protective Measures of Transformers under Lightning Overvoltage. IEEE Trans. Dielectr. Electr.Insul, 718–726

Ciaburro, G. (2022) Machine fault detection methods based on machine learning algorithms: A review**.** Mathematical Biosciences and Engineering, 19(11): 11453–11490.

Hashemnia, N. and Abu-Siada, A. Islam, S. (2015) Improved power transformer winding fault detection using FRA diagnostics. in IEEE Transactions on Dielectrics and Electrical Insulation, 22(1): 556-563.

Quiñones, L.I.A., Lozano-Moncada, C.A., and Montenegro, D.A.B. (2022) Machine learning for predictive maintenance scheduling of distribution transformers. Journal of Quality in Maintenance Engineering, 29: 188–202.

Vita, V., Fotis, G., Chobanov, V., Pavlatos, C., and Mladenov, V. (2023) Predictive Maintenance for Distribution System Operators in Increasing Transformers' Reliability. Electronics, 12 (1356): 1-23

Wjcicki, B. and Dabrowski, R. (2018) Applying machine learning to software fault prediction. E-informatica Software Engineering Journal, 12(1): 199-216

Zhang, K., Yuan, F., Guo, J. and Wang, G.A. (2016) Novel Neural Network Approach to Transformer Fault Diagnosis Based on Momentum-Embedded BP Neural Network Optimized by Genetic Algorithm and Fuzzy c-Means, 3451–3461.

Zhang, L. and Zou, L. (2018) Analysis of winding vibration Characteristics of power transformers based on the Finite-Element Method. Energies, 9.