**Machine Learning Framework for Electrical Equipment Failure Detection for Power Systems**

**Abstract**

The advent of Industry 4.0 has revolutionized the manufacturing sector, emphasizing automation and data exchange in production technologies. In this context, the detection and classification of electrical equpiment faults are critical for maintaining optimal operational efficiency. Statistics indicate that unplanned downtimes due to electrical equipment failures can cost manufacturers up to $50 billion annually. Accurate fault detection can mitigate these losses by enabling predictive maintenance strategies. As manufacturing systems become more complex, traditional fault detection methods struggle to keep up with the volume and variety of data generated. There is a pressing need for automated solutions that can analyze this data in real-time. Machine learning offers a promising avenue to address these challenges by leveraging historical and real-time data to predict and classify faults accurately. This shift towards data-driven maintenance can significantly reduce downtime and maintenance costs. Manual fault detection methods are often time-consuming and reliant on human expertise, which can introduce inconsistencies and errors. These approaches typically involve routine inspections and scheduled maintenance, which may not always align with actual machine health. Additionally, manual methods can fail to detect early signs of faults, leading to unexpected breakdowns. The subjective nature of human inspections also limits the scalability and repeatability of fault detection processes. Our proposed solution employs machine learning techniques to classify electrical equipment faults using a dataset of various operational parameters, including rpm, motor power, torque, and temperatures. By training ML models on this dataset, we can develop a predictive maintenance system that identifies potential faults in the radiator component before they lead to significant failures. This approach aims to enhance the accuracy and timeliness of fault detection, enabling more proactive and cost-effective maintenance strategies in the manufacturing industry.

**CHAPTER 1**

**INTRODUCTION**

**1.1 Overview**

Predictive maintenance (PdM) is an innovative approach that aims to foresee equipment malfunctions before they occur, ensuring the continuous and efficient operation of electrical machinery. By leveraging sophisticated data analytics and machine learning techniques, PdM evaluates the condition and performance metrics of machinery, facilitating timely and precise maintenance actions. This approach is pivotal in industrial settings where minimizing downtime and reducing maintenance costs are paramount.

Traditional maintenance strategies are categorized into reactive and preventive maintenance. Reactive maintenance, often referred to as "run-to-failure," involves repairing equipment only after a breakdown has occurred. This method can lead to substantial downtime and elevated repair costs. Preventive maintenance, on the other hand, schedules maintenance activities at predetermined intervals regardless of the actual condition of the equipment. While this approach aims to prevent unexpected failures, it often results in unnecessary maintenance tasks and does not reflect the true wear and tear experienced by the machinery.

Both reactive and preventive maintenance strategies fail to effectively utilize the vast amounts of operational data generated by modern machinery. Consequently, these strategies miss critical opportunities to optimize maintenance schedules based on real-time equipment conditions. The inefficiency and higher costs associated with these traditional methods highlight the necessity for a more advanced maintenance approach. Predictive maintenance, powered by machine learning algorithms, addresses this need by using real-time data to accurately predict when maintenance is required, thus ensuring both cost-effectiveness and operational reliability.

The focus of this project is on enhancing the efficiency and reliability of electrical equipment maintenance through machine learning-based predictive maintenance. By analyzing data collected from various sensors monitoring key operational parameters—such as rpm, motor power, torque, pressures, air flow, noise levels, temperatures, and acceleration—the project aims to predict potential failures. The system utilizes a Decision Tree Classifier to analyze a comprehensive dataset, which includes labels for components like bearings, water pumps, radiators, exhaust valves, and AC motors. This approach ensures that maintenance is performed only when necessary, thereby reducing downtime, lowering maintenance costs, and extending the lifespan of the equipment.

**1.2 Research Motivation**

The primary motivation for this research stems from the pressing need to improve the efficiency and reliability of maintenance practices for electrical equipment. In the current industrial landscape, the reliance on traditional maintenance strategies—reactive and preventive—has proven to be both costly and inefficient. Reactive maintenance, which involves fixing equipment post-failure, leads to prolonged downtime and high repair costs. On the other hand, preventive maintenance, which is conducted at regular intervals regardless of actual equipment condition, often results in unnecessary maintenance activities, contributing to wasted resources and increased operational costs.

The advent of modern machinery has introduced a plethora of operational data that, if harnessed correctly, can revolutionize maintenance strategies. However, traditional methods fail to leverage this data, missing critical opportunities to optimize maintenance schedules based on real-time equipment conditions. This gap underscores the importance of developing a predictive maintenance system that can analyze real-time data and predict maintenance needs accurately.

Machine learning offers a promising solution to this challenge. By utilizing advanced algorithms to analyze sensor data, machine learning-based predictive maintenance can foresee potential failures before they occur. This proactive approach not only minimizes downtime and maintenance costs but also enhances the overall reliability and lifespan of the equipment. The potential benefits of such a system—reduced operational costs, improved equipment reliability, and optimized maintenance schedules—serve as the driving force behind this research.

The specific focus on electrical equipment in this project is motivated by the critical role these machines play in industrial operations. Electrical equipment is often the backbone of industrial processes, and any downtime can lead to significant production losses. By developing a predictive maintenance system tailored to the unique characteristics and requirements of electrical equipment, this research aims to deliver a robust solution that addresses the shortcomings of traditional maintenance strategies and meets the demands of modern industrial operations.

**1.3 Problem Statement**

Traditional maintenance strategies for electrical equipment are fraught with inefficiencies and high costs. Reactive maintenance, or the "run-to-failure" approach, entails repairing equipment only after a breakdown has occurred, leading to significant downtime and substantial repair expenses. Conversely, preventive maintenance involves scheduling maintenance at regular intervals, irrespective of the actual condition of the equipment. This method often results in unnecessary servicing, increased operational costs, and wasted resources. Both approaches fail to utilize the extensive operational data generated by modern machinery, thus missing opportunities to optimize maintenance schedules based on real-time equipment conditions.

The primary issue with these traditional maintenance strategies is their inability to predict when maintenance is actually needed. This results in either excessive maintenance costs due to unnecessary servicing or unexpected equipment downtime due to unforeseen failures. In an industrial context, where minimizing downtime and reducing maintenance costs are critical, there is a pressing need for a more efficient and reliable maintenance approach.

The problem at hand is the development of a system that can accurately predict maintenance needs based on real-time data from various sensors monitoring key operational parameters of electrical equipment. Such a system should be capable of analyzing data on rpm, motor power, torque, pressures, air flow, noise levels, temperatures, and acceleration, among others, to foresee potential failures and determine the optimal timing for maintenance activities.

This project aims to address this problem by implementing a machine learning-based predictive maintenance system using a Decision Tree Classifier. By leveraging a comprehensive dataset of operational parameters and component labels, the system will predict maintenance needs accurately, ensuring maintenance is performed only when necessary. This approach promises to reduce downtime, lower maintenance costs, and extend the lifespan of electrical equipment, ultimately enhancing the efficiency and reliability of industrial operations.

**1.4 Applications**

* **Industrial Manufacturing**: In manufacturing plants, predictive maintenance can be applied to critical electrical equipment such as motors, pumps, and conveyors. By predicting potential failures and scheduling maintenance during planned downtimes, manufacturers can minimize disruptions to production lines, enhance productivity, and reduce operational costs.
* **Energy Sector**: Power generation and distribution facilities can benefit from predictive maintenance by ensuring the continuous operation of critical equipment like turbines, transformers, and generators. This can lead to fewer power outages, improved energy efficiency, and lower maintenance costs.
* **Transportation**: Predictive maintenance can be used to monitor and maintain electrical components in transportation systems, including trains, trams, and electric buses. By predicting and addressing potential failures, transportation companies can improve service reliability, reduce delays, and ensure passenger safety.
* **Aerospace**: In the aerospace industry, predictive maintenance can be applied to monitor the health of electrical systems in aircraft. By predicting failures and performing maintenance proactively, airlines can enhance flight safety, reduce maintenance-related delays, and optimize maintenance schedules.
* **Automotive Industry**: Predictive maintenance can be used to monitor and maintain electrical components in vehicles, such as batteries, motors, and electronic control units. This can lead to improved vehicle reliability, reduced breakdowns, and lower maintenance costs for fleet operators.
* **Oil and Gas**: In the oil and gas industry, predictive maintenance can be applied to electrical equipment used in exploration, drilling, and production operations. By predicting potential failures and performing timely maintenance, companies can reduce downtime, enhance operational efficiency, and lower maintenance costs.
* **Healthcare**: Predictive maintenance can be used to monitor and maintain critical electrical equipment in healthcare facilities, such as MRI machines, X-ray machines, and ventilators. This can lead to improved equipment reliability, reduced downtime, and enhanced patient care.

**CHAPTER 2**

**LITERATURE SURVEY**

[1] R. A. Patel and B. Bhavesh R (2016) explored the application of Support Vector Machines (SVM) for the condition monitoring and fault diagnosis of induction motors. This study highlights the effectiveness of SVM in identifying faults in motor operations, showcasing the potential of machine learning techniques in industrial maintenance. [2] W. Garrison (1988) reviewed the largest property damage losses in the hydrocarbon-chemical industries over thirty years. This historical analysis provides insights into the impact of major incidents on industry practices and safety standards. [3] P. A. Carson and C. J. Mumford (1979) analyzed incidents involving major hazards in the chemical industry. Their work underscores the importance of safety protocols and risk management in preventing catastrophic events. [4] C. Y. H. Kan and S. Kumara (2018) discussed the use of parallel computing and network analytics for fast processing and condition monitoring in the Industrial Internet of Things (IIoT). Their study emphasizes the role of advanced computing technologies in enhancing the efficiency of condition monitoring systems. [5] A. H. Pesch and P. N. Scavelli (2019) examined the condition monitoring of active magnetic bearings within the context of the Internet of Things. Their research indicates that IoT technologies can significantly enhance the monitoring and maintenance of specialized industrial equipment. [6] P. Večeř, M. Kreidl, and R. Šmíd (2005) investigated condition indicators for gearbox monitoring systems, providing insights into the specific metrics and indicators that are critical for effective condition monitoring of mechanical systems.

[7] D. Yarmoluk and C. Truempi (2019) advocated for the transition from condition monitoring to predictive maintenance, highlighting the economic and operational benefits of predictive approaches over traditional monitoring techniques. [8] F. Besnard, J. Nilsson, and L. Bertling (2010) evaluated the economic benefits of condition monitoring systems for wind power systems. They concluded that such systems can lead to substantial cost savings and improved reliability in the renewable energy sector. [9] M. You, F. Liu, and G. Meng (2011) discussed the benefits of condition monitoring techniques through a case study on the maintenance scheduling of ball grid array solder joints, demonstrating practical applications and advantages in the electronics industry. [10] V. V. Karanović, M. T. Jocanović, J. M. Wakiru, and M. D. Orošnjak (2018) investigated the benefits of lubricant oil analysis for maintenance decision support, providing a case study that illustrates the practical applications and benefits of condition monitoring in maintaining equipment health. [11] X. Tang, X. Wang, R. Cattley, F. Gu, and A. D. Ball (2018) reviewed energy harvesting technologies for achieving self-powered wireless sensor networks in machine condition monitoring. This review discusses the potential of integrating energy harvesting with wireless sensor networks to create more sustainable and autonomous monitoring systems.

[12] J. M. Wakiru, L. Pintelon, P. N. Muchiri, and P. K. Chemweno (2019) provided a comprehensive review on lubricant condition monitoring information analysis for maintenance decision support. Their study highlights the importance of lubricant analysis in predictive maintenance strategies. [13] Z. Zhao, B. Liang, X. Wang, and W. Lu (2017) focused on the remaining useful life prediction of aircraft engines based on degradation pattern learning. This study demonstrates the application of machine learning techniques in the aerospace industry to predict equipment lifespan and optimize maintenance schedules. [14] H. P. Jagtap, A. K. Bewoor, and R. Kumar (2020) conducted a failure analysis of an induced draft fan used in a thermal power plant using a coordinated condition monitoring approach, showcasing the importance of integrated monitoring techniques in preventing equipment failures. [15] M. Taghipour and A. Moosavi (2020) presented a study on gas turbine vibration condition monitoring, illustrating the application of condition monitoring in the energy sector to ensure operational efficiency and prevent failures. [16] K. Mykoniatis (2020) developed a real-time condition monitoring and maintenance management system for low voltage industrial motors using IoT. This work exemplifies the practical implementation of IoT in industrial maintenance, enhancing real-time monitoring capabilities. [17] C. R. Farrar and S. W. Dowbling (1999) provided a comprehensive review of damage detection and evaluation techniques using modal analysis and testing. This work underscores the importance of vibration-based methods in structural health monitoring and fault detection. [18] A. Kusiak and V. Anoop (2011) applied data mining techniques to monitor wind turbines. Their study shows how data-driven approaches can improve the monitoring and maintenance of renewable energy infrastructure. [19] A. Abouhnik and A. Albarbar (2012) explored the condition assessment of wind turbine blades using vibration measurements. Their research highlights the critical role of vibration analysis in maintaining the structural integrity of wind turbines.

[20] E. P. Carden and P. Fanning (2004) reviewed vibration-based condition monitoring techniques, emphasizing their importance in structural health monitoring and early fault detection. [21] L. Dong, R. Mingyue, and M. Guoying (2017) explored the application of IoT technology in predictive maintenance systems for coal equipment. Their study illustrates how IoT can enhance the predictive capabilities and efficiency of maintenance systems in heavy industries. [22] P. Qian, D. Zhang, X. Tian, Y. Si, and L. Li (2019) proposed a novel wind turbine condition monitoring method based on cloud computing. This research highlights the integration of cloud computing with condition monitoring to enable real-time data processing and analysis. [23] K. Sujatha, B. Deepalakshmi, and S. Q. Cao (2018) discussed the optimal condition monitoring of wind turbines using intelligent image processing and the Internet of Things. Their study emphasizes the role of advanced image processing techniques in enhancing the reliability of wind turbine maintenance. [24] G. Manogaran, R. Varatharajan, D. Lopez, P. M. Kumar, R. Sundarasekar, and C. Thota (2018) proposed a new architecture of Internet of Things and big data ecosystem for secured smart healthcare monitoring and alerting systems. This work demonstrates the cross-industry applications of IoT and big data technologies.

[25] M. Cakir, M. A. Guvenc, and S. Mistikoglu (2021) experimented with popular machine learning algorithms on predictive maintenance and designed an IIoT-based condition monitoring system. This study underscores the effectiveness of machine learning in predictive maintenance. [26] A. Joelian (2020) optimized engine replacement scheduling using data mining techniques. This research highlights the application of data mining in optimizing maintenance schedules for industrial equipment. [27] N. Silva, J. Soares, V. Shah, M. Y. Santos, and H. Rodrigues (2017) utilized a data mining approach for anomaly detection in roads. This research demonstrates the versatility of data mining techniques in various applications, including infrastructure maintenance. [28] P. Dehghanian, Y. Guan, and M. Kezunovic (2018) examined the real-time life-cycle assessment of high-voltage circuit breakers using online condition monitoring data, highlighting the importance of real-time data in making informed maintenance decisions. [29] I. Aydin, M. Karakose, and E. Akin (2007) proposed an artificial immune-based support vector machine algorithm for fault diagnosis of induction motors. Their study illustrates the potential of combining different machine learning techniques for enhanced fault diagnosis. [30] B. Samanta, K. Al-Balushi, and S. Al-Araimi (2003) employed artificial neural networks and support vector machines with genetic algorithms for bearing fault detection. This research highlights the effectiveness of hybrid machine learning approaches in fault detection and predictive maintenance.

**CHAPTER 3**

**EXISTING SYSTEM**

**3.1 PASSIVE-AGGRESSIVE CLASSIFIER**

The Passive-Aggressive (PA) classifier is an online learning algorithm designed for large-scale and real-time classification tasks. It falls under the category of linear classifiers, which means it attempts to find a hyperplane that separates different classes in the feature space. What sets the PA classifier apart is its unique approach to updating the model parameters: it combines passive updates when the current model correctly classifies the incoming data and aggressive updates when it misclassifies, ensuring that mistakes are corrected immediately.

**How the Passive-Aggressive Classifier Works**

The PA algorithm operates on the principle of making minimal adjustments to the model unless an error is encountered, which contrasts with traditional learning algorithms that update their parameters continuously regardless of correctness.

**Key Concepts:**

1. **Online Learning**: The PA classifier processes one instance at a time, making it suitable for scenarios where data arrives sequentially and in large volumes. Each instance is used to update the model before the next instance is processed.
2. **Margin-Based Update**: The PA classifier aims to maximize the margin, which is the distance between the decision boundary and the instances. For binary classification, the margin is defined as the product of the true label and the predicted value. A higher margin indicates greater confidence in the prediction.
3. **Hinge Loss**: The algorithm employs hinge loss to quantify the error for an instance. The hinge loss is zero if the instance is correctly classified with a sufficient margin; otherwise, it increases linearly with the margin's shortfall.
4. **Update Rule**: When an instance is misclassified, the model parameters are updated in a manner that corrects the mistake with the smallest possible change. The update rule is derived from solving a constrained optimization problem that seeks to adjust the weights minimally while ensuring correct classification of the current instance.

**Algorithm Steps:**

1. **Initialization**: Start with an initial weight vector (often set to zero).
2. **Prediction**: For each incoming instance, compute the predicted label using the current weight vector.
3. **Loss Calculation**: Calculate the hinge loss for the instance.
4. **Parameter Update**:
   * If the instance is correctly classified with a margin greater than a threshold, no update is made (passive).
   * If the instance is misclassified or correctly classified with a margin less than the threshold, update the weights aggressively. The magnitude of the update is proportional to the loss and inversely proportional to the norm of the feature vector.
5. **Repeat**: Continue this process for each instance in the data stream.

The PA algorithm can be formalized into three variants:

1. **PA-I**: This variant includes a regularization term to control the aggressiveness of updates. The regularization term is inversely proportional to the learning rate, which helps in tuning the trade-off between making aggressive updates and maintaining stability.
2. **PA-II**: Similar to PA-I but includes a squared regularization term. This variant generally provides smoother updates and better generalization performance in some cases.
3. **PA**: The basic version without any regularization.

**3.2 Limitations**

Here are the limitations of the Passive-Aggressive (PA) classifier, presented in a point-wise format:

* + **No Batch Learning**: The PA classifier processes instances one at a time, which means it cannot leverage the full dataset at once. This can be less effective in scenarios where batch processing could capture complex patterns through global optimization.
  + **Parameter Sensitivity**: Performance can be highly sensitive to the choice of hyperparameters, such as the aggressiveness parameter and the regularization term. Poorly chosen hyperparameters can lead to suboptimal performance.
  + **Potential Overfitting**: Without careful tuning, the aggressive updates of the PA classifier can lead to overfitting, especially in noisy datasets where the model might excessively adapt to outliers or noise.
  + **Limited to Linear Models**: The PA algorithm is inherently a linear classifier. It may struggle with datasets where the relationship between features and labels is nonlinear unless combined with techniques like kernel methods.
  + **Data Order Dependency**: The order in which data instances are presented can affect the final model. Different sequences of the same data can lead to different models, potentially causing variability in performance.
  + **Inefficiency with Redundant Data**: When the dataset contains many redundant or highly correlated instances, the algorithm might perform unnecessary updates, leading to inefficiencies in computation.

**CHAPTER 4**

**PROPOSED SYSTEM**

**4.1 Overview**

The predictive maintenance for electrical equipment involves several key steps aimed at understanding the dataset, preprocessing it, building and evaluating machine learning models, and finally making predictions on unseen test data. Here, we outline the detailed overview.

**Step 1: Dataset:** The dataset containing various operational parameters of electrical equipment. This dataset serves as the foundation for all subsequent analysis and model building efforts.

**Step 2: Data Analysis:** The exploration of the dataset involves examining its structure and contents. Using methods such as head(), tail(), describe(), and info(), we gain insights into the dimensions of the dataset, the types of features present, and the overall distribution of data.

**Step 3: Data Preprocessing:** Data preprocessing is a critical step that involves handling missing values, encoding categorical variables, and assessing feature correlations. Techniques such as label encoding are applied to convert categorical features into numerical representations. Additionally, we analyze the correlation between features to understand their relationships.

**Step 4: Train-Test Splitting:** To train and evaluate machine learning models, we split the dataset into training and testing subsets. Typically, an 80-20 ratio is used, where 80% of the data is used for training and 20% for testing.

**Step 5: Existing Passive Aggressive Classifier Model Building:** Then by building a machine learning model using the existing Passive Aggressive Classifier algorithm. This algorithm is chosen due to its suitability for online learning and classification tasks, making it well-suited for predictive maintenance scenarios.

**Step 6: Proposed Decision Tree Classifier Model Building:** In addition to the existing model, The Proposed build is a Decision Tree Classifier (DTC) model. Decision trees are known for their interpretability and ability to handle nonlinear relationships in data, making them a suitable candidate for our predictive maintenance task.

**Step 7: Performance Comparison:** After building both the Passive Aggressive Classifier and Decision Tree Classifier models, The evaluation of their performances using various metrics such as precision, recall, F1-score, and accuracy. This comparison allows us to determine which model performs better for our specific predictive maintenance task.

**Step 8: Prediction on Test Data:** The Trained Decision Tree Classifier model to make predictions on unseen test data. This step involves preprocessing the test data in a similar manner to the training data and then applying the trained model to generate predictions for the equipment's maintenance needs.

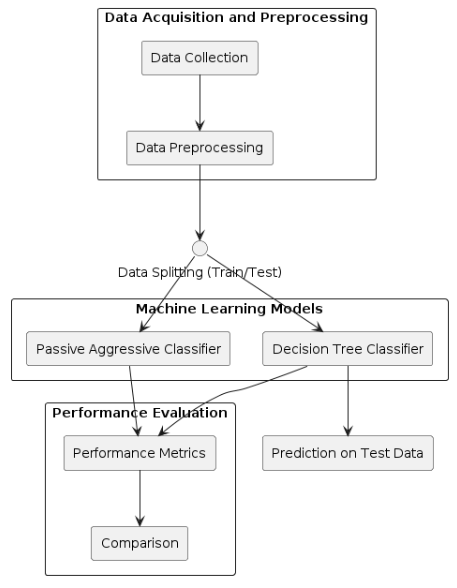


Fig. 1: Block Diagram of Proposed System.

**4.2 Data Preprocessing**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

* Getting the dataset
* Importing libraries
* Importing datasets
* Finding Missing Data
* Encoding Categorical Data
* Splitting dataset into training and test set

**Importing Libraries:** To perform data preprocessing using Python, we need to import some predefined Python libraries. These libraries are used to perform some specific jobs. There are three specific libraries that we will use for data preprocessing, which are:

Numpy: Numpy Python library is used for including any type of mathematical operation in the code. It is the fundamental package for scientific calculation in Python. It also supports to add large, multidimensional arrays and matrices. So, in Python, we can import it as:

import numpy as nm

Here we have used nm, which is a short name for Numpy, and it will be used in the whole program.

Matplotlib: The second library is matplotlib, which is a Python 2D plotting library, and with this library, we need to import a sub-library pyplot. This library is used to plot any type of charts in Python for the code. It will be imported as below:

import matplotlib.pyplot as mpt

Here we have used mpt as a short name for this library.

Pandas: The last library is the Pandas library, which is one of the most famous Python libraries and used for importing and managing the datasets. It is an open-source data manipulation and analysis library. Here, we have used pd as a short name for this library. Consider the below image:

Text

Description automatically generated

**Handling Missing data:** The next step of data preprocessing is to handle missing data in the datasets. If our dataset contains some missing data, then it may create a huge problem for our machine learning model. Hence it is necessary to handle missing values present in the dataset.There are mainly two ways to handle missing data, which are:

* By deleting the particular row: The first way is used to commonly deal with null values. In this way, we just delete the specific row or column which consists of null values. But this way is not so efficient and removing data may lead to loss of information which will not give the accurate output.
* By calculating the mean: In this way, we will calculate the mean of that column or row which contains any missing value and will put it on the place of missing value. This strategy is useful for the features which have numeric data such as age, salary, year, etc.

**Encoding Categorical data:** Categorical data is data which has some categories such as, in our dataset; there are two categorical variables, Country, and Purchased.Since machine learning model completely works on mathematics and numbers, but if our dataset would have a categorical variable, then it may create trouble while building the model. So, it is necessary to encode these categorical variables into numbers.

**4.3 Splitting the Dataset**

In machine learning data preprocessing, we divide our dataset into a training set and test set. This is one of the crucial steps of data preprocessing as by doing this, we can enhance the performance of our machine learning model.Suppose if we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

A picture containing shape

Description automatically generated

Figure 4.2: Splitting the dataset.

**Training** **Set**: A subset of dataset to train the machine learning model, and we already know the output.

**Test** **set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

For splitting the dataset, we will use the below lines of code:

from sklearn.model\_selection import train\_test\_split

x\_train, x\_test, y\_train, y\_test= train\_test\_split(x, y, test\_size= 0.2, random\_state=0)

**Explanation**

* In the above code, the first line is used for splitting arrays of the dataset into random train and test subsets.
* In the second line, we have used four variables for our output that are
* x\_train: features for the training data
* x\_test: features for testing data
* y\_train: Dependent variables for training data
* y\_test: Independent variable for testing data
* In train\_test\_split() function, we have passed four parameters in which first two are for arrays of data, and test\_size is for specifying the size of the test set. The test\_size maybe .5, .3, or .2, which tells the dividing ratio of training and testing sets.
* The last parameter random\_state is used to set a seed for a random generator so that you always get the same result, and the most used value for this is 42.

**4.4 Decision Tree Classifier**

A decision tree is one of the most powerful tools of supervised learning algorithms used for both classification and regression tasks. It builds a flowchart-like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. It is constructed by recursively splitting the training data into subsets based on the values of the attributes until a stopping criterion is met, such as the maximum depth of the tree or the minimum number of samples required to split a node.

During training, the Decision Tree algorithm selects the best attribute to split the data based on a metric such as entropy or Gini impurity, which measures the level of impurity or randomness in the subsets. The goal is to find the attribute that maximizes the information gain or the reduction in impurity after the split.

**What is a Decision Tree?**

A decision tree is a flowchart-like tree structure where each internal node denotes the feature, branches denote the rules and the leaf nodes denote the result of the algorithm. It is a versatile supervised ML algorithm, which is used for both classification and regression problems. It is one of the very powerful algorithms. And it is also used in Random Forest to train on different subsets of training data, which makes random forest one of the most powerful algorithms in machine learning.



Fig. 2: Decision Tree Classifier.

**Decision Tree Terminologies**

Some of the common Terminologies used in Decision Trees are as follows:

* **Root Node:** It is the topmost node in the tree, which represents the complete dataset. It is the starting point of the decision-making process.
* Decision/Internal Node: A node that symbolizes a choice regarding an input feature. Branching off of internal nodes connects them to leaf nodes or other internal nodes.
* **Leaf/Terminal Node:** A node without any child nodes that indicates a class label or a numerical value.
* **Splitting:**The process of splitting a node into two or more sub-nodes using a split criterion and a selected feature.
* **Branch/Sub-Tree:** A subsection of the decision tree starts at an internal node and ends at the leaf nodes.
* **Parent Node:** The node that divides into one or more child nodes.
* **Child Node:**The nodes that emerge when a parent node is split.
* **Impurity**: A measurement of the target variable’s homogeneity in a subset of data. It refers to the degree of randomness or uncertainty in a set of examples. The **Gini index** and **entropy** are two commonly used impurity measurements in decision trees for classifications task
* **Variance**: Variance measures how much the predicted and the target variables vary in different samples of a dataset. It is used for regression problems in decision trees. **Mean squared error, Mean Absolute Error, friedman\_mse, or Half Poisson deviance** are used to measure the variance for the regression tasks in the decision tree.
* **Information Gain:** Information gain is a measure of the reduction in impurity achieved by splitting a dataset on a particular feature in a decision tree. The splitting criterion is determined by the feature that offers the greatest information gain, It is used to determine the most informative feature to split on at each node of the tree, with the goal of creating pure subsets
* **Pruning**: The process of removing branches from the tree that do not provide any additional information or lead to overfitting.

**Attribute Selection Measures:**

**Construction of Decision Tree:** A tree can be *“learned”* by splitting the source set into subsets based on Attribute Selection Measures. Attribute selection measure (ASM) is a criterion used in decision tree algorithms to evaluate the usefulness of different attributes for splitting a dataset. The goal of ASM is to identify the attribute that will create the most homogeneous subsets of data after the split, thereby maximizing the information gain. This process is repeated on each derived subset in a recursive manner called*recursive partitioning*. The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions. The construction of a decision tree classifier does not require any domain knowledge or parameter setting and therefore is appropriate for exploratory knowledge discovery. Decision trees can handle high-dimensional data.

**Entropy:**

Entropy is the measure of the degree of randomness or uncertainty in the dataset. In the case of classifications, It measures the randomness based on the distribution of class labels in the dataset.

The entropy for a subset of the original dataset having K number of classes for the ith node can be defined as:

Where,

* S is the dataset sample.
* k is the particular class from K classes
* p(k) is the proportion of the data points that belong to class k to the total number of data points in dataset sample S.
* Here p(i,k) should not be equal to zero.

**Important points related to Entropy:**

1. The entropy is 0 when the dataset is completely homogeneous, meaning that each instance belongs to the same class. It is the lowest entropy indicating no uncertainty in the dataset sample.
2. when the dataset is equally divided between multiple classes, the entropy is at its maximum value. Therefore, entropy is highest when the distribution of class labels is even, indicating maximum uncertainty in the dataset sample.
3. Entropy is used to evaluate the quality of a split. The goal of entropy is to select the attribute that minimizes the entropy of the resulting subsets, by splitting the dataset into more homogeneous subsets with respect to the class labels.
4. The highest information gain attribute is chosen as the splitting criterion (i.e., the reduction in entropy after splitting on that attribute), and the process is repeated recursively to build the decision tree.

**Gini Impurity or index:**

Gini Impurity is a score that evaluates how accurate a split is among the classified groups. The Gini Impurity evaluates a score in the range between 0 and 1, where 0 is when all observations belong to one class, and 1 is a random distribution of the elements within classes. In this case, we want to have a Gini index score as low as possible. Gini Index is the evaluation metric we shall use to evaluate our Decision Tree Model.

Here,

* pi is the proportion of elements in the set that belongs to the ith category.

**Information Gain:**

Information gain measures the reduction in entropy or variance that results from splitting a dataset based on a specific property. It is used in decision tree algorithms to determine the usefulness of a feature by partitioning the dataset into more homogeneous subsets with respect to the class labels or target variable. The higher the information gain, the more valuable the feature is in predicting the target variable.

The information gain of an attribute A, with respect to a dataset S, is calculated as follows:

where

* A is the specific attribute or class label
* |H| is the entropy of dataset sample S
* |HV| is the number of instances in the subset S that have the value v for attribute A

Information gain measures the reduction in entropy or variance achieved by partitioning the dataset on attribute A. The attribute that maximizes information gain is chosen as the splitting criterion for building the decision tree.

Information gain is used in both classification and regression decision trees. In classification, entropy is used as a measure of impurity, while in regression, variance is used as a measure of impurity. The information gain calculation remains the same in both cases, except that entropy or variance is used instead of entropy in the formula.

**How does the Decision Tree algorithm Work?**

The decision tree operates by analyzing the data set to predict its classification. It commences from the tree’s root node, where the algorithm views the value of the root attribute compared to the attribute of the record in the actual data set. Based on the comparison, it proceeds to follow the branch and move to the next node.

The algorithm repeats this action for every subsequent node by comparing its attribute values with those of the sub-nodes and continuing the process further. It repeats until it reaches the leaf node of the tree. The complete mechanism can be better explained through the algorithm given below.

* Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
* Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
* Step-3: Divide the S into subsets that contains possible values for the best attributes.
* Step-4: Generate the decision tree node, which contains the best attribute.
* Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf nodeClassification and Regression Tree algorithm.

**Advantages of the Decision Tree:**

1. It is simple to understand as it follows the same process which a human follow while making any decision in real-life.
2. It can be very useful for solving decision-related problems.
3. It helps to think about all the possible outcomes for a problem.
4. There is less requirement of data cleaning compared to other algorithms.

**CHAPTER 5**

**SOFTWARE ENVIRONMENT**

**What is Python?**

Below are some facts about Python.

* Python is currently the most widely used multi-purpose, high-level programming language.
* Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
* Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
* Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* Machine Learning
* GUI Applications (like Kivy, Tkinter, PyQt etc. )
* Web frameworks like Django (used by YouTube, Instagram, Dropbox)
* Image processing (like Opencv, Pillow)
* Web scraping (like Scrapy, BeautifulSoup, Selenium)
* Test frameworks
* Multimedia

**Advantages of Python**

Let’s see how Python dominates over other languages.

1. **Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

**Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

**Advantages of Python Over Other Languages**

**1. Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

1. **Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is dynamically typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**Modules Used in Project**

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and Ipython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with Ipython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

**Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

**Download the Correct version into the system**

Step 1: Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: <https://www>.python.org

A screenshot of a computer

Description automatically generated with medium confidence

Now, check for the latest and the correct version for your operating system.

Step 2: Click on the Download Tab.

Graphical user interface, application

Description automatically generated

Step 3: You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

Step 4: Scroll down the page until you find the Files option.

Step 5: Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

**Installation of Python**

Step 1: Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

Step 2: Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generated

Step 3: Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

**Verify the Python Installation**

Step 1: Click on Start

Step 2: In the Windows Run Command, type “cmd”.

Graphical user interface, application

Description automatically generated

Step 3: Open the Command prompt option.

Step 4: Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

Step 5: You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

**Check how the Python IDLE works**

Step 1: Click on Start

Step 2: In the Windows Run command, type “python idle”.

Application

Description automatically generated with low confidence

Step 3: Click on IDLE (Python 3.7 64-bit) and launch the program

Step 4: To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Graphical user interface, text, application, email

Description automatically generated

Step 5: Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

Step 6: Now for e.g. enter print (“Hey World”) and Press Enter.

Graphical user interface, text, application, email

Description automatically generated

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**CHAPTER 6**

**SYSTEM REQUIREMENTS**

**Software Requirements**

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

* Python IDLE 3.7 version (or)
* Anaconda 3.7 (or)
* Jupiter (or)
* Google colab

**Hardware Requirements**

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

* Operating system : Windows, Linux
* Processor : minimum intel i3
* Ram : minimum 4 GB
* Hard disk : minimum 250GB

**CHAPTER 7**

**FUNCTIONAL REQUIREMENTS**

**Output Design**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**Output Definition**

The outputs should be defined in terms of the following points:

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**Input Design**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**Input Stages**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**Error Detection**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Clasified As:**

* User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
* Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

* Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
* Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

* The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
* Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system
* The existing system is completely dependent on the user to perform all the duties.

**CHAPTER 8**

**SOURCE CODE**

# Predictive Maintenance for Electrical Equipment: Machine Learning Classifier Evalution

#importing dataset

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.metrics import precision\_score

from sklearn.metrics import recall\_score

from sklearn.metrics import f1\_score

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

import os

from sklearn.linear\_model import PassiveAggressiveClassifier

#uploading dataset

df = pd.read\_csv(r'dataset\data.csv')

df.head()

#data analysis

df.info()

df.describe()

#data Correlection

df.corr()

#Checking NULL values

df.isnull().sum()

df = df.drop(['id'], axis = 1)

df

Labels = ['bearings','wpump','radiator','exvalve','acmotor']

for i in Labels:

df[i] = LabelEncoder().fit\_transform(df[i])

df

df.info()

labels = ['Clean','Dirty']

labels

#Data Visulazation

sns.set(style="darkgrid")

plt.figure(figsize=(12, 6))

ax = sns.countplot(x=df['radiator'], data=df, palette="Set3")

plt.title("Count Plot")

plt.xlabel("Categories")

plt.ylabel("Count")

ax.set\_xticklabels(labels)

for p in ax.patches:

ax.annotate(f'{p.get\_height()}', (p.get\_x() + p.get\_width() / 2., p.get\_height()),

ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),

textcoords='offset points')

plt.show()

df

#Declaring independent and dependent variable

x = df.drop(['radiator'],axis = 1)

x.head()

y = df['radiator']

y

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y, test\_size = 0.10, random\_state = 42)

x\_train.shape

y\_train.shape

#performance evalution

precision = []

recall = []

fscore = []

accuracy = []

def performance\_metrics(algorithm, predict, testY):

testY = testY.astype('int')

predict = predict.astype('int')

p = precision\_score(testY, predict,average='macro') \* 100

r = recall\_score(testY, predict,average='macro') \* 100

f = f1\_score(testY, predict,average='macro') \* 100

a = accuracy\_score(testY,predict)\*100

accuracy.append(a)

precision.append(p)

recall.append(r)

fscore.append(f)

print(algorithm+' Accuracy : '+str(a))

print(algorithm+' Precision : '+str(p))

print(algorithm+' Recall : '+str(r))

print(algorithm+' FSCORE : '+str(f))

report=classification\_report(predict, testY,target\_names=labels)

print('\n',algorithm+" classification report\n",report)

conf\_matrix = confusion\_matrix(testY, predict)

plt.figure(figsize =(5, 5))

ax = sns.heatmap(conf\_matrix, xticklabels = labels, yticklabels = labels, annot = True, cmap="Blues" ,fmt ="g");

ax.set\_ylim([0,len(labels)])

plt.title(algorithm+" Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()

# Passive Aggressive Classifier model building

pa\_model\_path = 'model/PassiveAggressiveClassifier.npy'

if os.path.exists(pa\_model\_path):

# Load the Passive Aggressive Classifier model

pa\_classifier = np.load(pa\_model\_path, allow\_pickle=True).item()

else:

# Train and save the Passive Aggressive Classifier model

pa\_classifier = PassiveAggressiveClassifier()

pa\_classifier.fit(x\_train, y\_train)

np.save(pa\_model\_path, pa\_classifier)

# Predict using the trained Passive Aggressive Classifier model

y\_pred\_pa = pa\_classifier.predict(x\_test)

# Evaluate the Passive Aggressive Classifier model

performance\_metrics('PassiveAggressiveClassifier', y\_pred\_pa, y\_test)

#Decision Tree Classifier model building

from sklearn.tree import DecisionTreeClassifier

# Check if the model file exists

model\_path = 'model/DecisionTreeClassifier.npy'

if os.path.exists(model\_path):

# Load the model

classifier = np.load(model\_path, allow\_pickle=True).item()

else:

# Train and save the model

classifier = DecisionTreeClassifier()

classifier.fit(x\_train, y\_train)

np.save(model\_path, classifier)

# Predict using the trained model

y\_pred = classifier.predict(x\_test)

# Evaluate the model

performance\_metrics('DecisionTreeClassifier', y\_pred, y\_test)

#Tabular form of Performance Metrics

#showing all algorithms performance values

columns = ["Algorithm Name","Precison","Recall","FScore","Accuracy"]

values = []

algorithm\_names = ["Passive Aggressive Classifier", "Decision Tree Classifier"]

for i in range(len(algorithm\_names)):

values.append([algorithm\_names[i],precision[i],recall[i],fscore[i],accuracy[i]])

temp = pd.DataFrame(values,columns=columns)

temp

#Uploading testing dataset

test=pd.read\_csv("test.csv")

test

Test\_Labels = ['bearings','wpump','exvalve','acmotor']

for i in Test\_Labels:

test[i] = LabelEncoder().fit\_transform(test[i])

test

#Model prediction on test data

predict = classifier.predict(test)

for i, p in enumerate(predict):

if p == 0:

print(test.iloc[i])

print("Model Predicted of Row {} Test Data is--->".format(i),labels[0])

elif p == 1:

print(test.iloc[i])

print("Model Predicted of Row {} Test Data is--->".format(i),labels[1])

**CHAPTER 9**

**RESULTS AND DISCUSSION**

**9.1 Implementation Description**

The implementation of a predictive maintenance system for electrical equipment involves several key stages, each focusing on different aspects of data handling, model building, and evaluation. Below is a detailed description of each block in the implementation process.

* **Data Collection:** The initial stage of the project involves collecting data from various sensors monitoring key operational parameters of electrical equipment. These parameters include rpm, motor power, torque, outlet pressure, air flow, noise levels, outlet temperature, water pump outlet pressure, water inlet temperature, and various acceleration data (e.g., gaccx, gaccy, gaccz, haccx, haccy, haccz). The data is typically stored in a CSV file, which is read into a DataFrame for analysis and preprocessing.
* **Data Preprocessing:** Once the data is collected, the next step is to preprocess it. Preprocessing involves several sub-tasks:
  + **Exploratory Data Analysis (EDA):** This includes examining the structure of the data, checking the types of features, and understanding the distribution of the data. Functions such as head(), describe(), and info() provide insights into the data.
  + **Handling Missing Values:** It is crucial to check for any null values and handle them appropriately, either by removing rows/columns with missing values or imputing them using appropriate techniques.
  + **Label Encoding:** For categorical variables like the condition of components (bearings, water pump, radiator, exhaust valve, AC motor), label encoding is performed to convert these categories into numerical values that machine learning algorithms can process.
  + **Correlation Analysis:** Understanding the correlation between different features helps in identifying redundant features and understanding feature relationships.
  + **Data Visualization:** Visualizing data distributions and relationships using tools like Seaborn helps in gaining additional insights and validating preprocessing steps.
* **Data Splitting:** The dataset is then split into training and testing subsets to facilitate model training and evaluation. Typically, the data is split in an 80-20 ratio, where 80% of the data is used for training the models, and 20% is reserved for testing their performance. This step ensures that the models are evaluated on unseen data, providing a realistic measure of their performance.
* **Model Training:** Two machine learning models are trained in parallel: the Passive Aggressive Classifier and the Decision Tree Classifier.
  + **Passive Aggressive Classifier:** This model is known for its efficiency in online learning tasks. It is trained using the training subset, where it iteratively adjusts its parameters based on each data instance. The model is saved for future use after training.
  + **Decision Tree Classifier:** This model is chosen for its interpretability and ability to handle nonlinear relationships in the data. It constructs a tree-like structure of decisions based on feature values, splitting the data into homogeneous subsets. The trained model is also saved for future use.
* **Model Evaluation:** The model evaluation for the predictive maintenance system involved a thorough assessment of two classifiers: the Passive Aggressive Classifier and the Decision Tree Classifier. Using the test dataset, both models were meticulously evaluated based on several key performance metrics, including accuracy, precision, recall, and F1-score. These metrics provided a comprehensive understanding of each model's ability to correctly predict maintenance needs.
* **Model Comparison:** The performance metrics of both models are compared to determine which model performs better for the predictive maintenance task. The comparison helps in selecting the best model for making predictions on new data.
* **Prediction on Test Data:** Using the selected best model (typically the one with higher accuracy and balanced precision-recall trade-off), predictions are made on the unseen test data. This step involves preprocessing the test data similarly to the training data and applying the trained model to generate predictions.

**9.2 Dataset Description**

The dataset contains 1000 records with 26 columns, capturing various operational parameters of electrical equipment essential for predictive maintenance analysis. Each record represents a unique obseration, while the columns provide insights into different aspects of the equipment's performance and condition.

The dataset includes attributes such as rpm, motor power, torque, outlet pressure, air flow, noise level, outlet temperature, water pump outlet pressure, water inlet temperature, water outlet temperature, water pump power, water flow, oil pump power, oil tank temperature, acceleration along x, y, and z directions, and high-frequency acceleration along the same axes. These attributes offer comprehensive information about the equipment's operational characteristics and environmental conditions.

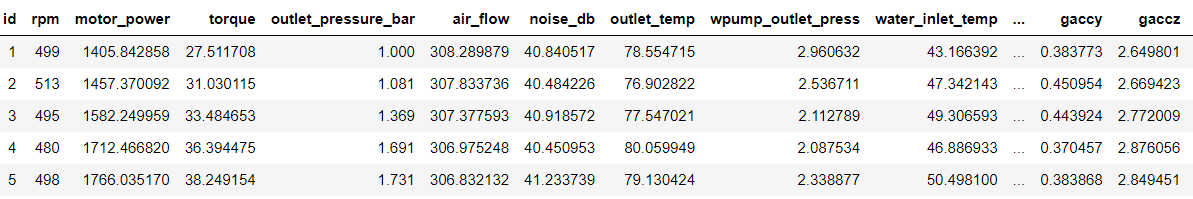
Additionally, categorical labels are provided for components like bearings, water pump, radiator, exhaust valve, and AC motor. These labels indicate the condition or status of each component, serving as critical indicators for maintenance needs and equipment reliability.

**9.3 Results and Description**

Figures 1 and 2 provide an overview of the dataset and its target column, respectively. Figure 1 presents a sample dataset of equipment maintenance, showcasing the various attributes and their values. This sample dataset serves as a visual representation of the data used for predictive maintenance analysis. Figure 2 displays a count plot of the target column, indicating the distribution of categories within the dataset. This plot helps understand the balance or imbalance between different classes, which is crucial for model evaluation.

Figures 3 and 5 illustrate the performance metrics of the Passive Aggressive Classifier and the Decision Tree Classifier, respectively. These figures showcase the accuracy, precision, recall, and F1-score of each classifier. For the Passive Aggressive Classifier (Figure 3), an accuracy of 81.0% is achieved, with precision, recall, and F1-score values of 40.5%, 50.0%, and 44.75%, respectively. In contrast, the Decision Tree Classifier (Figure 5) achieves perfect scores across all metrics, with an accuracy, precision, recall, and F1-score of 100.0%.

Figure 6 depicts the confusion matrix of the Decision Tree Classifier, highlighting the true positives, true negatives, false positives, and false negatives. This matrix provides a detailed breakdown of the classifier's performance in classifying instances into different categories.

****

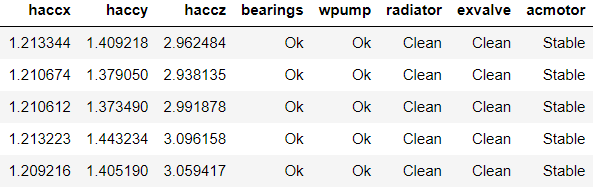
****

Fig. 1: Presents the Sample dataset of the Equipment Maintenance.

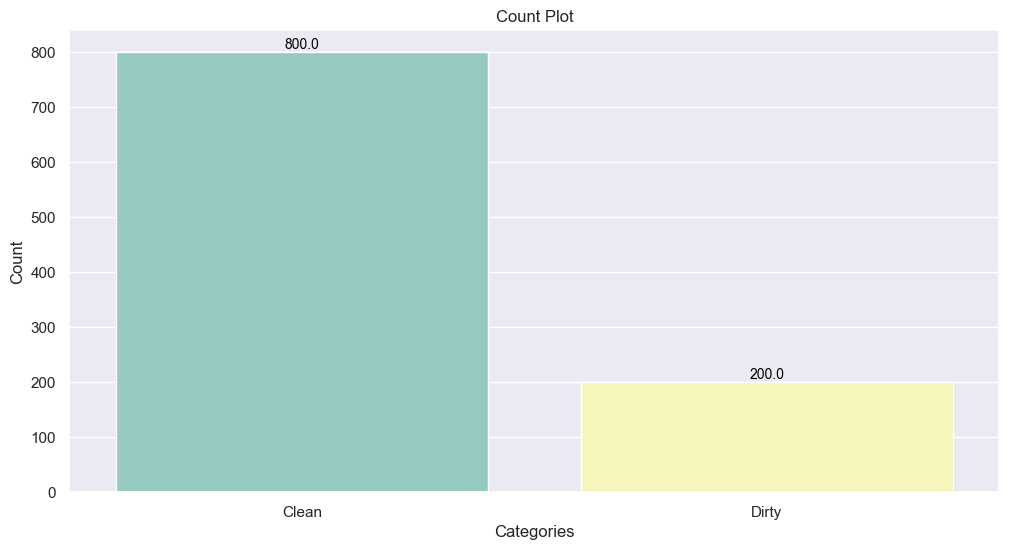


Fig. 2: Presents the Count plot of the Dataset Target Column.

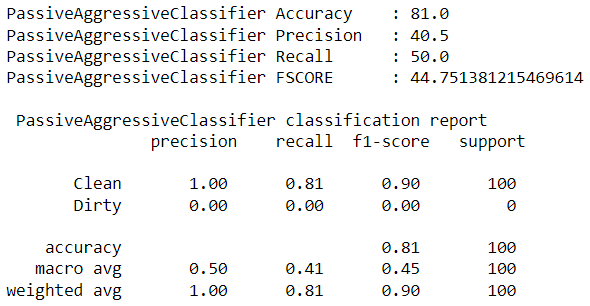


Fig. 3: Presents the Performance metrics of Passive Aggressive Classifier.

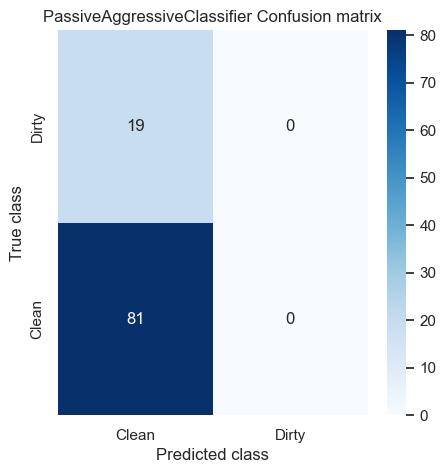


Fig. 4: Presents the Confusion matrix of Passive Aggressive Classifier.

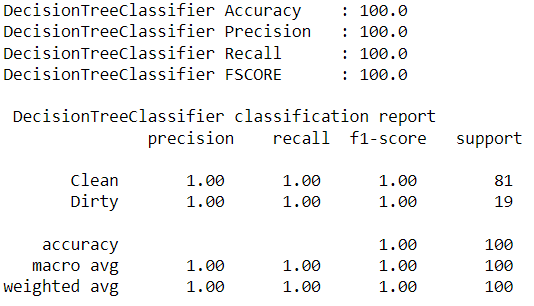


Fig. 5: Presents the Performance metrics of Decision Tree Classifier.

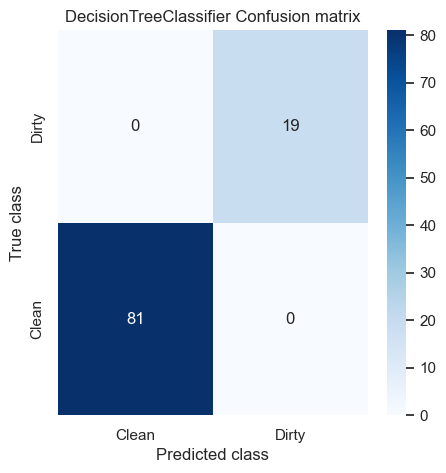


Fig. 6: Presents the Confusion matrix of Decision Tree Classifier.

Table 1 summarizes the performance metrics comparison between the Passive Aggressive Classifier and the Decision Tree Classifier. It presents a side-by-side comparison of accuracy, precision, recall, and F1-score values for both classifiers, offering insights into their relative performance.

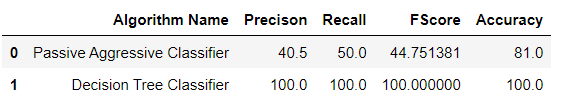


Table 1: Performance metrics comparison table of Passive Aggressive Classifier and Decision Tree

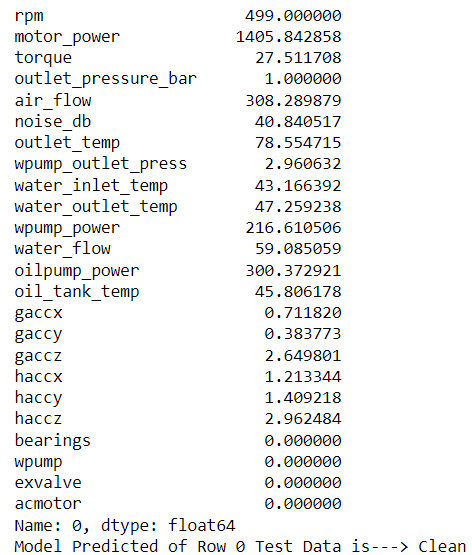


Fig. 7: Model Prediction on Uploaded Dataset.

Figure 7 presents the model prediction on the uploaded dataset. Each row of the dataset is displayed, showing the original attributes along with the predicted class label. In this specific instance, the model predicted the equipment's condition as "Clean" based on the provided input data. This visualization allows for a quick assessment of the model's performance in classifying maintenance needs.

**CHAPTER 10**

**CONCLUSION AND FUTURE SCOPE**

In conclusion, the systematic approach to predictive maintenance for electrical equipment using machine learning techniques is by analyzing the dataset, preprocessing the data, building and evaluating machine learning models, and making predictions on test data, we can effectively identify maintenance needs and optimize equipment reliability. Both the existing Passive Aggressive Classifier and the proposed Decision Tree Classifier models offer valuable insights into equipment maintenance, with the latter providing interpretability and nonlinear modeling capabilities.

**Future Scope**

While this research provides a solid foundation for predictive maintenance in electrical equipment, there are several avenues for future exploration and improvement. One potential area of focus is the integration of additional sensor data or advanced feature engineering techniques to enhance model performance. Moreover, exploring ensemble learning techniques or more sophisticated algorithms could further improve predictive accuracy. Additionally, incorporating real-time data streaming capabilities and deploying models in production environments would enable continuous monitoring and proactive maintenance strategies.

**REFERENCES**

[1] R. A. Patel and B. Bhavesh R, “Condition monitoring and fault diagnosis of induction motor using support vector machine,” Electric

Power Components and Systems, pp. 663-692, 2016.

[2] W. Garrison, “One hundred largest losses-A Thirty-year review of property damage losses in the hydrocarbon-chemical industries,” in

M&M Protection Consultant, Chicago, 1988.

[3] P. A. Carson and C. J. Mumford, “An analysis of incidents involving major hazards in the chemical industry,” Journal of Hazardous

materials, pp. 149-165, 1979.

[4] C. Y. H. Kan and S. Kumara, “Parallel computing and network analytics for fast Industrial Internet-of-Things (IIoT) machine information

processing and condition monitoring,” Journal of manufacturing systems, pp. 282-293, 2018.

[5] A. H. Pesch and P. N. Scavelli, “Condition monitoring of active magnetic bearings on the internet of things,” in Actuators, 2019.

[6] P. Večeř, M. Kreidl and R. Šmíd, “Condition indicators for gearbox condition monitoring systems,” Acta Polytechnica, p. 6, 2005.

[7] D. Yarmoluk and C. Truempi, “Why move from condition monitoring to predictive maintenance? – Part 1,” IBM, 15 March 2019.

[Online]. Available: https://www.ibm.com/blogs/internet-of-things/iot-condition-monitoring-part-one/. [Accessed 18 March 2020].

[8] F. Besnard, J. Nilsson and L. Bertling, “On the economic benefits of using condition monitoring systems for maintenance management of

wind power systems,” in 2010 IEEE 11th International Conference on Probabilistic Methods Applied to Power Systems, 2010.

[9] M. You, F. Liu and G. Meng, “Benefits from condition monitoring techniques: a case study on maintenance scheduling of ball grid array

solder joints,” in the Institution of Mechanical Engineers, 2011.

[10] V. V. Karanović, M. T. Jocanović, J. M. Wakiru and M. D. Orošnjak, “Benefits of lubricant oil analysis for maintenance decision support:

a case study,” in IOP Conference Series: Materials Science and Engineering, 2018.

[11] X. Tang, X. Wang, R. Cattley, F. Gu and A. D. Ball, “Energy harvesting technologies for achieving self-powered wireless sensor

networks in machine condition monitoring: A review,” Sensors, p. 4113, 2018.

[12] J. M. Wakiru, L. Pintelon, P. N. Muchiri and P. K. Chemweno, “A review on lubricant condition monitoring information analysis for

maintenance decision support,” Mechanical systems and signal processing, pp. 108-132, 2019.

[13] Z. Zhao, B. Liang, X. Wang and W. Lu, “Remaining useful life prediction of aircraft engine based on degradation pattern learning,”

Reliability Engineering & System Safety, vol. 164, pp. 74-83, 2017.

[14] H. P. Jagtap, A. K. Bewoor and R. Kumar, “Failure analysis of induced draft fan used in a thermal power plant using coordinated

condition monitoring approach: A case study,” Engineering Failure Analysis, no. 111, p. 104442, 2020.

[15] M. Taghipour and A. Moosavi, “A look at Gas Turbine Vibration Condition Monitoring in Region 3 of Gas aTransmission Operation,”

Journal of Environmental Science, Computer Science and Engineering & Technology, vol. 9, no. 3, pp. 423-432, 2020.

[16] K. Mykoniatis, “A Real-Time Condition Monitoring and Maintenance Management System for Low Voltage Industrial Motors Using

Internet-of-Things,” in International Conference on Industry 4.0 and Smart manufacturing (ISM 2019), 2020.

[17] C. R. Farrar and S. W. Dowbling, “Damage detection and evaluation II,” in Modal analysis and testing, Dordrecht., 1999.

[18] A. Kusiak and V. Anoop, “A data-mining approach to monitoring wind turbines,” IEEE Transactions on Sustainable Energy, vol. 3, no.

1, pp. 150-157, 2011.

[19] A. Abouhnik and A. Albarbar, “Wind Turbine Blades Condition Assessment based on Vibration Measurements and the Level of and

Empirically Decomposed Feature,” Energy Conversion and Management, pp. 606-613, 2012.

[20] E. P. Carden and P. Fanning, “Vibration based condition monitoring: a review,” Structural health monitoring, pp. 355-357, 2004.

[21] L. Dong, R. Mingyue and M. Guoying, “Application of internet of things technology on predictive maintenance system of coal

equipment,” Procedia engineering, vol. 174, pp. 885-889, 2017.

[22] P. Qian, D. Zhang, X. Tian, Y. Si and L. Li, “A novel wind turbine condition monitoring method based on cloud computing,” Renewable

energy, pp. 390-398, 2019.

[23] K. Sujatha, B. Deepalakshmi and S. Q. Cao, “Optimal condition monitoring of wind turbines using intelligent image processing and

internet of things,” International Journal of Renewable Energy Technology, pp. 158-180, 2018.

[24] G. Manogaran, R. Varatharajan, D. Lopez, P. M. Kumar, R. Sundarasekar and C. Thota, “A new architecture of Internet of Things and big

data ecosystem for secured smart healthcare monitoring and alerting system,” Future Generation Computer Systems,, pp. 375-387, 2018.

[25] M. Cakir, M. A. Guvenc and S. Mistikoglu, “The experimental application of popular machine learning algorithms on predictive

maintenance and the design of IIoT based condition monitoring system,” Computers & Industrial Engineering, vol. 151, p. 106948, 2021.

[26] A. Joelian, “Engine replacement scheduling optimization using Data Mining,” Journal of Physics Conference Series, 2020.

[27] N. Silva, J. Soares, V. Shah, M. Y. Santos and H. Rodrigues, “Anomaly detection in roads with a data mining approach,” Procedia

computer science, pp. 415-422, 2017.

[28] P. Dehghanian, Y. Guan and M. Kezunovic, “Real-time life-cycle assessment of high-voltage circuit breakers for maintenance using

online condition monitoring data,” IEEE Transactions on Industry Applications, pp. 1135-1146, 2018.

[29] I. Aydin, M. Karakose and E. Akin, “Artificial immune based support vector machine algorithm for fault diagnosis of induction motors,” in International Aegean Conference on Electrical Machines and Power Electronics, 2007.

[30] B. Samanta, K. Al-Balushi and S. Al-Araimi, “Artificial neural networks and support vector machines with genetic algorithm for bearing fault detection,” Engineering applications of artificial intelligence, pp. 657-665, 2003.