PREDICTIVE POWER MACHINE LEARNING MODELS FOR MACHINE FAILURE STATUS

A PROJECT REPORT

Submitted by

ABINAYA.S [REGISTER NO: 211421104005]

ABINISHEKA.S. S [REGISTER NO: 211421104006]

DARSHINI.A[REGISTER NO:211421104048]

i*n partial fulfillment for the award of the degree of*

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

APRIL & 2025

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

BONAFIDE CERTIFICATE

Certified that this project report **"PREDICTIVE POWER MACHINE LEARNING MODELS FOR MACHINE FAILURE STATUS"** is the bonafide work of "**ABINAYA.S (211421104005) ABINISHEKA.S.S (211421104006) DARSHINI.A (211421104048) .**"who carried out the project work under my supervision.

Signature of the HOD with date

DR L.JABASHEELA M.E., Ph.D.,

PROFESSOR AND HEAD,

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai - 123

Signature of the Supervisor with date

Mrs. S. SOPHANA JENNIFER .M.E.,

ASSISTANT PROFESSOR,

Department of Computer Science and Engineering,
Panimalar Engineering College,
Chennai - 123

Certified that the above candidate(s) was/ were examined in the Anna University Project Viva-Voce Examination held on.....

INTERNAL EXAMINER **EXTERNALEXAMINER**

DECLARATION BY THE STUDENT

We **ABINAYA.S(211421104005) , ABINISHEKA.S.S(211421104006)** , **DARSHINI.A(211421104048)** hereby declare that this project report titled **"PREDICTIVE POWER MACHINE LEARNING MODELS FOR MACHINE FAILURE STATUS ",**under the guidance of **Mrs. S. SOPHANA JENNIFER M.E.** is the original work done by us and we have not plagiarized or submitted to any other degree in any university by us.

ACKNOWLEDGEMENT

Our profound gratitude is directed towards our esteemed Secretary and Correspondent, **Dr. P. CHINNADURAI, M.A., Ph.D**., for his fervent encouragement. His inspirational support proved instrumental in galvanizing our efforts, ultimately contributing significantly to the successful completion of this project

We want to express our deep gratitude to our Directors,**Tmt. C. VIJAYARAJESWARI, Dr. C. SAKTHI KUMAR, M.E., Ph.D., and Dr. SARANYASREE SAKTHI KUMAR, B.E., M.B.A., Ph.D.,** for graciously affording us the essential resources and facilities for undertaking of this project.

Our gratitude is also extended to our Principal, **Dr. K. MANI, M.E., Ph.D.,** whose facilitation proved pivotal in the successful completion of this project.

We express our heartfelt thanks to **Dr. L. JABASHEELA, M.E., Ph.D.,** Head of the Department of Computer Science and Engineering, for granting the necessary facilities that contributed to the timely and successful completion of project.

We would like to express our sincere thanks to **Project Coordinator Dr. S. KAVITHA SUBRAMANI, M.E.,Ph.D.,** and **Project Guide Mrs. S. SOPHANA JENNIFER, M. E.,** and all the faculty members of the Department of CSE for their unwavering support for the successful completion of the project.

ABINAYA.S
ABINISHEKA.S.S
DARSHINI A

PROJECT COMPLETION CERTIFICATE

SPIRO SOLUTIONS PVT LTD

29.03.2025

TO WHOMSOEVER IT MAY CONCERN

This is to certify that **ABINAYA S (Reg No: 211421104005), ABINISHEKA S S (Reg No: 211421104006), DARSHINI A (Reg No: 211421104048),** students of final year **B.E., (COMPUTER SCIENCE)** of **"PANIMALAR ENGINEERING COLLEGE"** has completed their major project with great success at our concern, under the Title: **"PREDICTIVE POWER MACHINE LEARNING MODELS FOR MACHINE FAILURE STATUS"** from **JANUARY 2025** to **MARCH 2025.**

Their project is found to be relevant regarding their stream and they had submitted a copy of the project report to us. During their Project period we found their sincere and hard working and possessing good behaviour and moral character.

We wish her grand success in future endeavours.

For SPIRO PRIME TECH SERVICES,

SPIRO PRIME TECH SERVICES NO.78, IIIrd Floor, Usman Road, T.Nagar, CHENNAI-600 017

M.SAMPATH KUMAR

MANAGER

Corporate Office: #1, CVR Complex, 3rd Floor, Singaravelu Street, T.Nagar, Chennai - 600 017. Mobile : +91 9791 044 044 / www.spiroprojects.com

ABSTRACT

In the domain of predictive maintenance, the accurate forecasting of machine failure is pivotal for minimizing downtime and optimizing operational efficiency. This project focuses on developing and implementing machine learning models to predict failure status in industrial machinery. The approach involves a comprehensive workflow including data preprocessing, visualization, and model implementation. Data preprocessing techniques are employed to clean and prepare the dataset, ensuring the quality and reliability of the input features. Subsequently, data visualization tools are used to explore and interpret patterns within the data, providing insights that guide the selection and fine- tuning of machine learning algorithms. The final models, trained to predict machine failure status, are integrated with a Django framework, offering a user-friendly interface for predictions and monitoring. This integrated system aims to enhance predictive accuracy and facilitate proactive maintenance

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI - Artificial Intelligence

CNN - Convolutional Neural Networks

CNB - Complement Naive Bayes

CSE - Computer Science and Engineering

DFD - Data Flow Diagram

ERD - Entity Relationship Diagram

GRB - Gradient Boosting

IDE - Integrated Development Environment

IoT - Internet of Things

ISM - International Safety Management

LSTM - Long Short-Term Memory

ML - Machine Learning

OOD - Out-of-Distribution

RAM - Random Access Memory

RNN - Recurrent Neural Networks

SDG - Sustainable Development Goals

SVM - Support Vector Machines

UML - Unified Modeling Language

TABLE OF CONTENT

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF PROJECT

In modern industries, machine failures pose a critical challenge, leading to unplanned downtime, increased maintenance costs, and production inefficiencies. The Predictive Power Machine Learning Models - Machine Failure Status project aims to revolutionize industrial maintenance by implementing machine learning-based predictive analytics. The project utilizes historical machine sensor data and real-time operational data to forecast potential failures, allowing industries to take proactive measures before a breakdown occurs.

Machine learning models play a vital role in this project by identifying hidden patterns and correlations in machine behavior. By leveraging classification algorithms such as Random Forest, Gradient Boosting, and Naïve Bayes, the system enhances failure prediction accuracy. The project follows a structured approach, including data collection and preprocessing to clean and transform raw industrial data, exploratory data analysis to visualize machine behavior and detect failure trends, and model training and evaluation using supervised learning techniques to build an accurate predictive model. To make the system accessible and user-friendly, a web interface is developed using Django, allowing industries to easily interpret predictions and plan maintenance activities accordingly.

By shifting from reactive to predictive maintenance, this project helps industries reduce unplanned failures, optimize maintenance schedules, and improve operational reliability. The integration of artificial intelligence in predictive maintenance enhances the overall

efficiency of manufacturing and industrial operations, leading to significant cost savings and increased productivity.

1.2 PROJECT DEFINITION

Traditional industrial maintenance relies on reactive or preventive strategies, where machines are repaired after failure or serviced based on predefined schedules. These approaches often result in unnecessary maintenance costs or unexpected downtimes, impacting the efficiency of production processes. The Predictive Power Machine Learning Models - Machine Failure Status project is designed to address these challenges by implementing a data-driven failure prediction system that enhances industrial maintenance practices.

The primary objective of this project is to develop an intelligent machine learning model capable of forecasting machine breakdowns with high precision. The project involves collecting and analyzing real-time and historical sensor data from industrial machines to identify key parameters influencing failures. The data is preprocessed and refined to improve accuracy, and machine learning algorithms are applied to build predictive models. The system integrates a web-based application using Django, providing an interactive platform for users to input data, monitor machine performance, and receive real-time failure predictions.

This predictive model allows industries to transition from traditional maintenance strategies to condition-based maintenance, where repairs and servicing are conducted only when necessary. This not only reduces maintenance costs but also extends the lifespan of industrial equipment by preventing excessive wear and tear. The implementation of

predictive analytics in maintenance ensures that industries can operate efficiently without unexpected breakdowns, enhancing overall operational reliability. By leveraging machine learning and artificial intelligence, this project provides a scalable, automated solution for failure prediction, ensuring seamless integration into industrial environments and improving decision-making processes in maintenance management

CHAPTER 2

2. LITERATURE REVIEW

Predictive maintenance is a crucial aspect of modern industrial operations, aimed at minimizing downtime and improving equipment reliability. Various machine learning techniques have been explored to enhance failure prediction accuracy and optimize maintenance schedules. This section provides a comprehensive review of significant contributions in the field, focusing on traditional machine learning, deep learning, and hybrid approaches.

In "Revisiting Confidence Estimation: Towards Reliable Failure Prediction" [1], the study investigated confidence estimation in deep learning models for failure prediction. It highlighted the challenges in detecting misclassifications and out-of-distribution (OOD) samples. The authors proposed a novel approach using flat minima optimization to enhance prediction reliability, bridging the gap between confidence calibration, OOD detection, and failure forecasting. Similarly, Mobley [2] discussed fundamental predictive maintenance principles and methodologies for reducing unplanned downtime.

Susto et al. [3] examined multiple classifier systems for predictive maintenance, integrating machine learning models with optimization techniques. Their findings highlighted the effectiveness of supervised learning approaches, such as support vector machines (SVM) and decision trees, in failure prediction. Additionally, Amruthnath and Gupta [4] explored unsupervised learning methods for anomaly detection, demonstrating the applicability of clustering techniques like k-means and isolation forests in industrial fault detection.

Advancements in deep learning have further enhanced predictive maintenance capabilities. Zhao et al. [5] introduced a Transformer-driven deep reinforcement learning framework that significantly improved proactive fault mitigation. Hamaide et al. [6] compared various machine learning approaches for predictive maintenance, showcasing the benefits of hybrid models that combine deep learning with traditional statistical techniques.

Pinciroli Vago et al. [7] investigated multivariate time series analysis for machine failure prediction in industrial settings. Their study emphasized the importance of temporal dependencies and demonstrated the efficacy of long short-term memory (LSTM) networks in predictive modeling. Chevtchenko et al. [8] developed an anomaly detection model for induction motors using real-time IoT data, further improving predictive maintenance accuracy.

Maheshwari et al. [9] performed a comparative analysis of classification models and LSTM networks, illustrating the advantages of deep learning in predictive maintenance. Bidollahkhani and Kunkel [10] examined the role of artificial intelligence in enhancing system reliability, demonstrating how AI-driven predictive maintenance strategies minimize operational risks and maintenance costs.

Beyond machine learning techniques, other studies have investigated the integration of AI with real-world applications. Huang and Du [11] optimized fire detection and recognition using virtual reality-based image processing, illustrating the versatility of AI techniques in safety-critical environments. Spang [12] developed an object tracking framework using local binary descriptors, which holds potential applications in predictive maintenance for industrial automation.

Advancements in hardware technology have also played a significant role in predictive maintenance. The Raspberry Pi 4 Model B [13] has been widely adopted for edge computing applications, enabling real-time monitoring and fault detection in industrial machinery. Similarly, Bimantara et al. [14] studied the implementation of predictive maintenance strategies in maritime industries, focusing on improving vessel reliability.

Herlambang and Nurpulaela [15] analyzed the use of fire alarm systems in large-scale infrastructures, highlighting the effectiveness of predictive analytics in failure prevention. Khabib et al. [16] evaluated thermal overload relay capabilities in marine control panels, underscoring the importance of predictive maintenance in ensuring operational stability.

Several industry-specific case studies have further reinforced the importance of predictive maintenance. "Predictive Maintenance Using Machine Learning: A Case Study in Manufacturing Management" [17] applied machine learning models to manufacturing systems, demonstrating a significant reduction in maintenance costs. "Machine Learning Approach for Predictive Maintenance in Industry 4.0" [18] explored the integration of AI and IoT in modern industrial operations.

Surveys on predictive maintenance trends, such as "A Survey on Predictive Maintenance Using AI and IoT Technologies" [19], have highlighted emerging methodologies and the challenges associated with AI-driven maintenance strategies. Lastly, "A Deep Learning Approach to Predictive Maintenance in Smart Manufacturing" [20] showcased the impact of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) on predictive accuracy in smart factory environment.

CHAPTER 3

THEORETICAL BACKGROUND

3.1 Implementation Environment

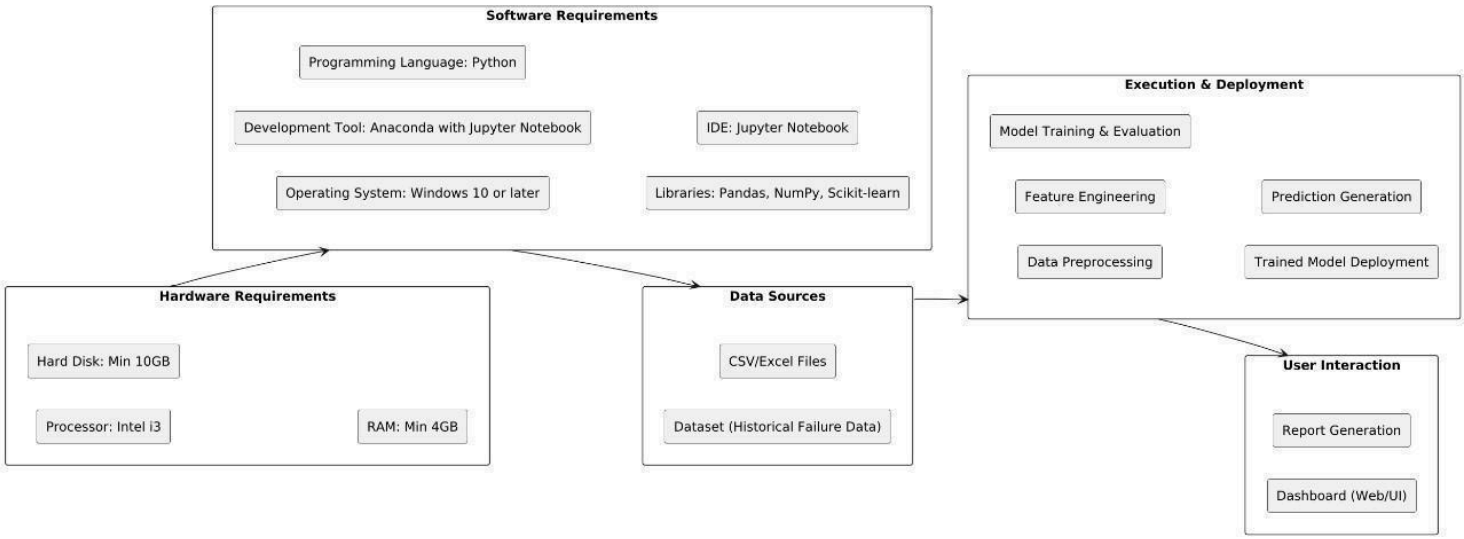


Fig 3.1 Implementation environment

3.1.1 Software Requirements

- Programming Language: Python
- Development Tool: Anaconda with Jupyter Notebook
- IDE: Jupyter Notebook
- Operating System: Windows 10 or later
- Libraries: Pandas, NumPy, Scikit-learn

3.1.2 Hardware Requirements

- Hard Disk: Min 10GB
- Processor: Intel i3
- RAM: Min 4GB

Data Sources

- CSV/Excel Files
- Dataset (Historical Failure Data)

Execution & Deployment

- Model Training & Evaluation
- Feature Engineering
- Data Preprocessing
- Prediction Generation
- Trained Model Deployment

User Interaction

- Report Generation
- Dashboard (Web/UI)

3.2 System Architecture:

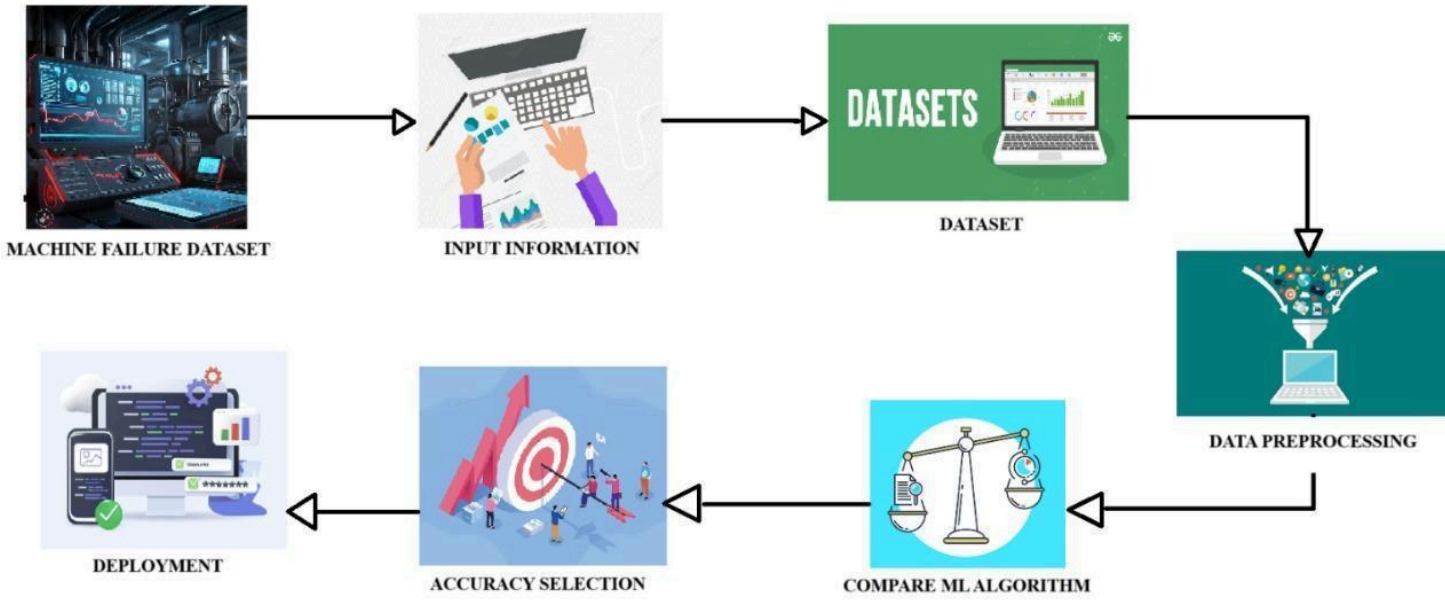


Fig 3.2 System Architecture

Machine Failure Status project, starting with Dataset Collection, where machine sensor data and failure records are gathered, followed by Data Preprocessing, where missing values, noise, and inconsistencies are handled to ensure data quality. Next, Data Visualization techniques help analyze patterns and relationships in the dataset, aiding in feature selection. Various ML Algorithms like Random Forest, Gradient Boosting, and Naïve Bayes are then compared to determine the most accurate model. The Accuracy Selection phase identifies the best-performing model based on evaluation metrics, and finally, the chosen model is Deployed using Django, enabling real-time failure predictions to assist in preventive maintenance and reducing downtime.

3.3 Proposed Methodology

WorkFlow Model

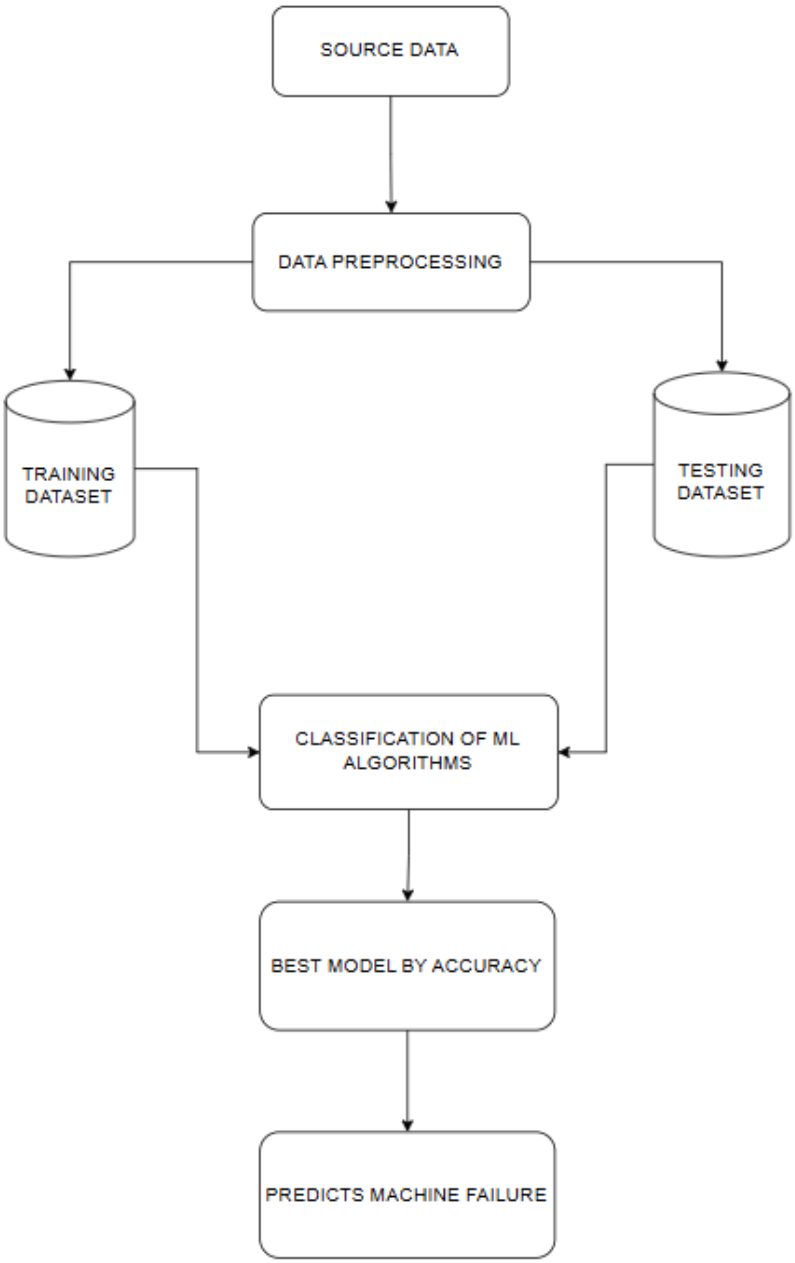


Fig 3.3WorkFlow Of Machine Learning Model

The research follows a quantitative approach, leveraging machine learning techniques to classify and predict system failures. The methodology comprises multiple stages, starting from data acquisition to model evaluation, ensuring a structured workflow. The approach includes:

- Data collection – Acquiring relevant historical data.
- Data preprocessing – Cleaning and refining data for accuracy.
- Model selection and training – Applying machine learning algorithms.
- Model evaluation– Measuring performance using statistical metrics.
- Prediction and analysis– Interpreting results to improve decision-making.

3.3.1 Database Design / Data Set Description

ER DIAGRAM

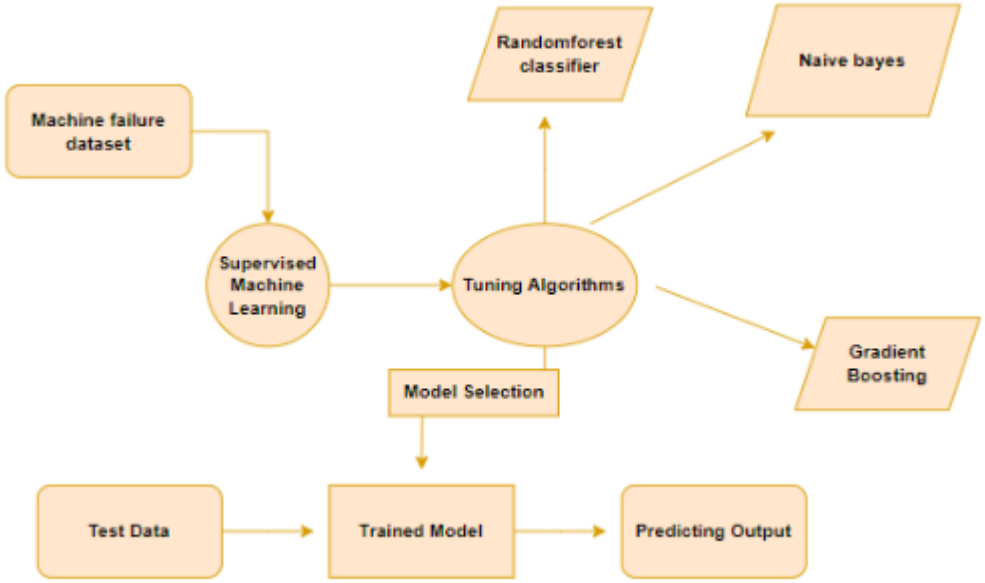


Fig 3.4 ER Diagram Of Machine Learning

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation of an information system that depicts the relationships among people, objects, places, concepts or events within that system. An ERD is a data modeling technique that can help define business processes and be used as the foundation for a relational database.Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a referral point, should any debugging or business process re-engineering be needed later.

3.3.2 Input Design (UI)

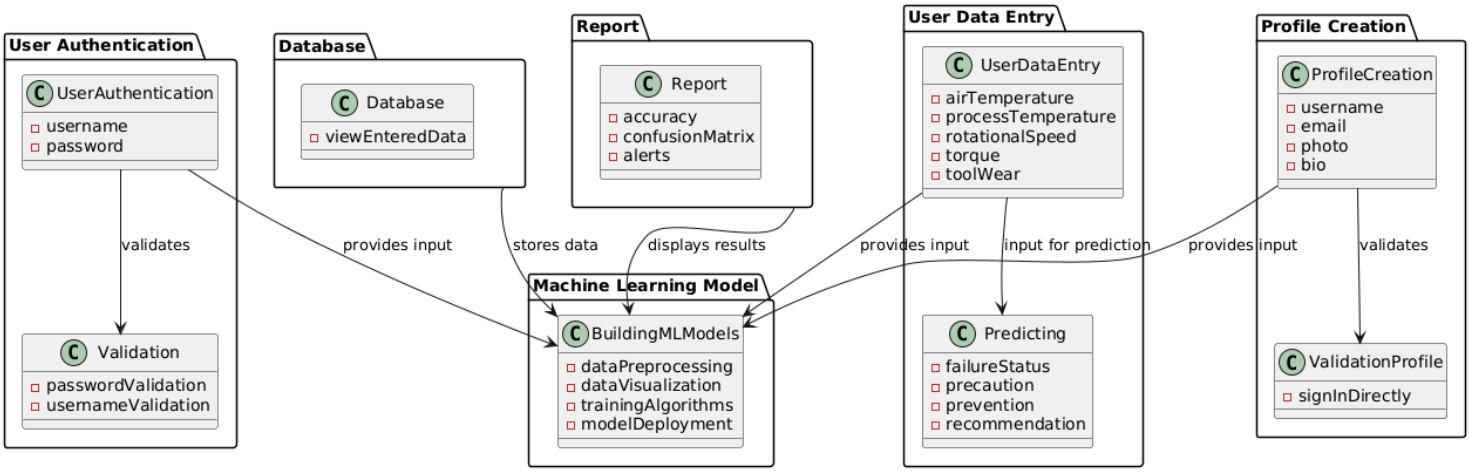


Fig 3.5 Input Design Of Machine Learning Model

Input Design of the Predictive Maintenance System ensures smooth data flow for accurate failure predictions. Users begin wiThe th User Authentication, where they enter their username and password, which are validated before access is granted. In the User Data Entry module, machine parameters like air temperature, process temperature, rotational speed, torque, and tool wear are recorded as input for the Machine Learning Model.

The model processes this data through preprocessing, visualization, training, and deployment stages. Input data is stored in the Database, allowing users to view entered data when needed. The Prediction Module analyzes the input to determine failure status and provides precautionary measures, prevention strategies, and recommendations.

The system generates reports on model accuracy, confusion matrix, and alerts for maintenance planning. Additionally, users can create profiles in the Profile Creation Module, entering username, email, photo, and bio , which is validated before allowing direct sign-in. This structured input design enables efficient data handling and proactive maintenance.

3.3.3 Module Design

Data Flow Diagram:

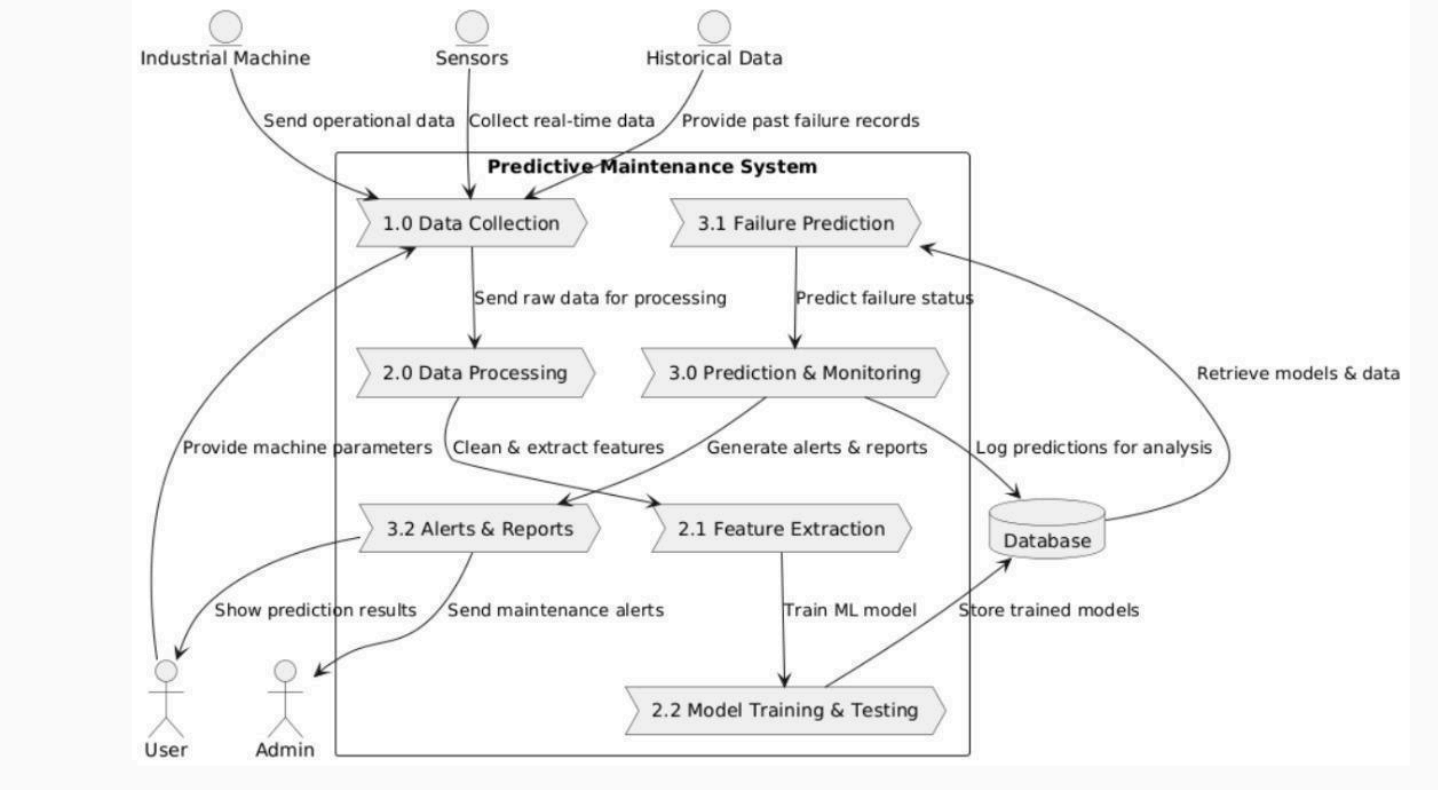


Fig 3.5 Data Flow Diagram Of Machine Learning Model

The Predictive Maintenance System operates by collecting data from multiple sources, including industrial machines, sensors, and historical records. Industrial machines send operational data, while sensors collect real-time data, and historical data provides past failure records to enhance predictive capabilities. The system starts with data collection (1.0), where all relevant information is gathered from these sources. This raw data is then sent for data processing (2.0), where it undergoes cleaning and feature extraction to ensure accuracy. The feature extraction (2.1) step identifies and selects crucial parameters that contribute to machine performance and failure prediction.

Once the features are extracted, the model training and testing (2.2) phase is initiated, where a machine learning model is trained to detect potential failures based on historical patterns. The trained model is then utilized in the prediction and monitoring (3.0) stage, where it continuously analyzes incoming data and generates insights. The failure prediction (3.1) step determines the likelihood of equipment failure, allowing for proactive measures. Additionally, the system generates alerts and reports (3.2), which notify both users and administrators about potential maintenance needs, ensuring timely intervention.

The system also interacts with a database, where trained models and logged predictions are stored for further analysis and future improvements. The database allows retrieval of models and data when needed. Lastly, the predictive results are displayed to users, who can view the prediction outcomes, while admins receive maintenance alerts to take corrective actions. This structured workflow helps in reducing downtime, improving efficiency, and ensuring the longevity of industrial machines through proactive maintenance.

3.6 USECASE DIAGRAM:

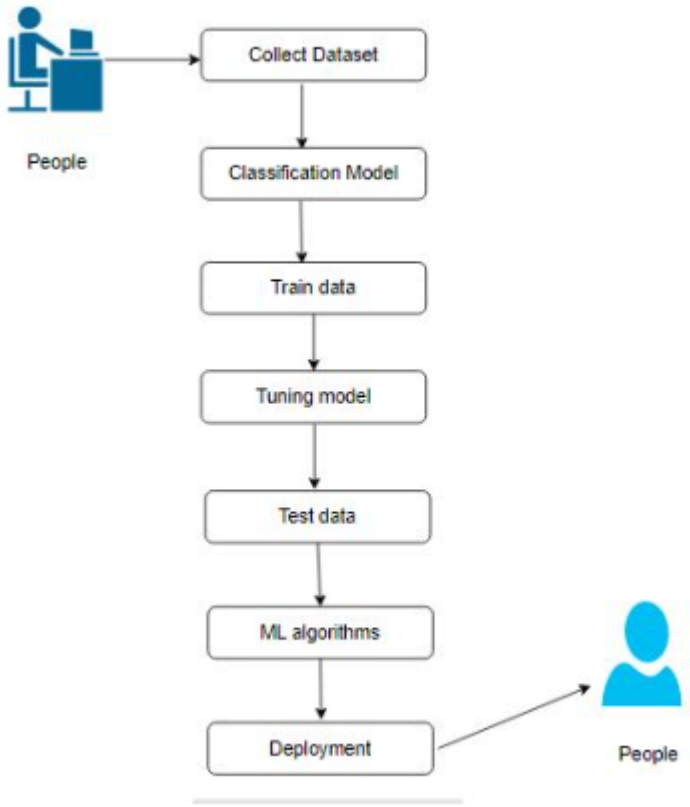


Fig 3.6 Usecase Diagram Of Machine Learning Model

The Use Case Diagram represents the interaction between users and the Predictive Machine Failure Detection System. The process starts with collecting datasets, where machine sensor data and historical failure records are gathered. This data is then used to build a classification model, which categorizes machine conditions as normal or failure-prone. Next, the training phase allows the model to learn failure patterns, followed by tuning the model to improve accuracy. After tuning, the system proceeds to the testing phase, where the model is evaluated for its predictive performance. Various machine learning algorithms, such as Random Forest, Gradient Boosting, and Naïve Bayes, are applied and compared to determine the most effective approach. Finally, the trained model is deployed, allowing users to input real-time machine data and receive predictive failure alerts, enabling proactive maintenance and reducing downtime.

3.7 CLASS DIAGRAM:

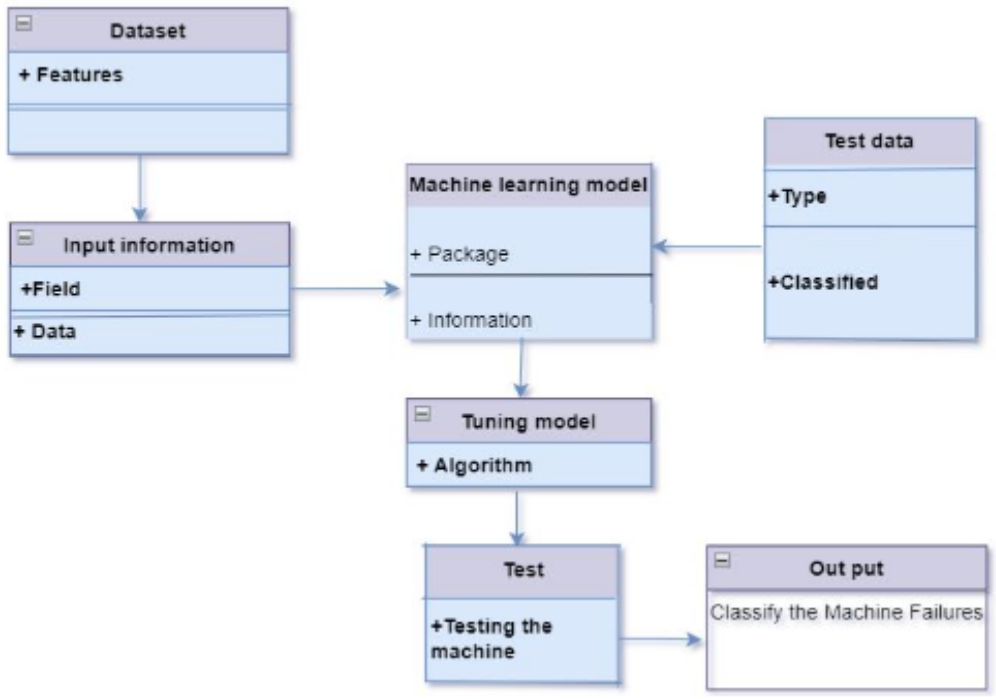


Fig 3.7 Class Diagram Of Machine Learning Model

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. So a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance Responsibility (attributes and methods) of each class should be clearly identified for each minimum number of properties should be specified and because, unnecessary properties will the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

3.8 SEQUENCE DIAGRAM

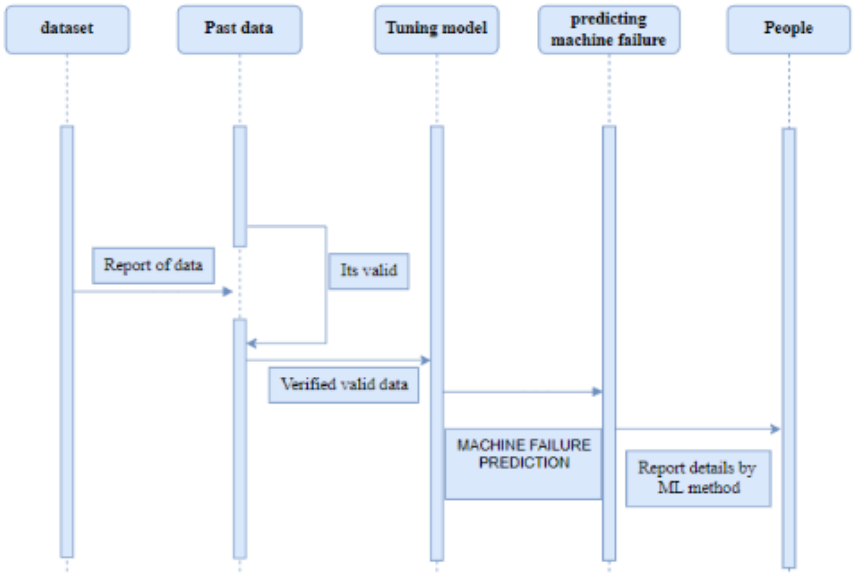


Fig 3.8 Sequence Diagram Of Machine Learning Model

Sequence diagrams model the flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important design-level models for modern business application development

CHAPTER 4

SYSTEM IMPLEMENTATION

4.1MODULES

4.1.1 Data Pre-processing:

Validation techniques in machine learning are used to get the error rate of the Machine Learning (ML) model, which can be considered as close to the true error rate of the dataset. If the data volume is large enough to be representative of the population, you may not need the validation techniques. However, in real-world scenarios, to work with samples of data that may not be a true representative of the population of a given dataset. To finding the missing value, duplicate value, and description of data type whether it is float variable or integer. The sample of data used to provide an unbiased evaluation of a model fit.

The evaluation becomes more biased as skill on the validation dataset is incorporated into the model configuration. The validation set is used to evaluate a given model, but this is for frequent evaluation. It as machine learning engineers use this data to fine-tune the model hyper parameters. Data collection, data analysis, and the process of addressing data content, quality, and structure can add up to a time-consuming to-do list. During the process of data identification, it helps to understand your data and its properties; this knowledge will help you choose which algorithm to use to build your model.

A number of different data cleaning tasks using Python's Pandas library and specifically, it focus on probably the biggest data cleaning task, missing values and it able to more quickly clean data.It wants to spend less time cleaning data, and more time exploring and modeling.

Some of these sources are just simple random mistakes. Other times, there can be a deeper reason why data is missing. It's important to understand these different types of missing data from a statistics point of view. The type of missing data will influence how to deal with

filling in the missing values and to detect missing values, and do some basic imputation and detailed statistical approach for dealing with missing data. Here are some typical reasons why data is missing:

- User forgot to fill in a field.
- Data was lost while transferring manually from a legacy database.
- There was a programming error.
- Users chose not to fill out a field tied to their beliefs about how the results would be used or interpreted.

Variable identification with Uni-variate, Bi-variate and Multi-variate analysis:

import libraries for access and functional purpose and read the given dataset

General Properties of Analyzing the given dataset:

- ➢ Display the given dataset in the form of data frame
- ➢ show columns
- ➢ shape of the data frame
- ➢ To describe the data frame
- ➢ Checking data type and information about dataset
- ➢ Checking for duplicate data
- ➢ Checking Missing values of data frame
- ➢ Checking unique values of data frame
- ➢ Checking count values of data frame
- ➢ Rename and drop the given data frame
- ➢ To specify the type of values
- ➢ To create extra column

4.1.2 Data Analysis of visualization:

Data visualization is an important skill in applied statistics and machine learning. Statistics does indeed focus on quantitative descriptions and estimations of data. Data visualization provides an important suite of tools for gaining a qualitative understanding. This can be helpful when exploring and getting to know a dataset and can help with identifying patterns, corrupt data outliers, and much more. With a little domain knowledge, data visualizations can be used to express and demonstrate key relationships in plots and charts that are more visceral and stakeholders than measures of association or significance. Data visualization and exploratory data analysis are whole fields themselves and it will recommend a deeper dive into some the books mentioned at the end. Sometimes data does not make sense until it can look at in a visual form, such as with charts and plots. Being able to quickly visualize of data samples and others is an important skill both in applied statistics and in applied machine learning. It will discover the many types of plots that you will need to know when visualizing data in Python and how to use them to better understand your own data.

4.2 ALGORITHM DESCRIPTION:

Implementing Random Forest Classifier:

A Random Forest classifier is a popular ensemble learning algorithm in machine learning that is primarily used for classification tasks. It is based on the idea of creating multiple decision trees during training and then combining their predictions to make more accurate and robust predictions. Random Forests are known for their versatility and ability to handle a wide range of data types and complexities.

Ensemble Learning: The term “ensemble” in machine learning refers to the practice of combining the predictions of multiple models to improve the overall accuracy and

reliability. Random Forest is an ensemble method because it combines the predictions of multiple decision trees.

Decision Trees: A decision tree is a simple yet powerful machine learning model that can be used for both classification and regression tasks. It makes decisions by recursively splitting the dataset into subsets based on the values of input features until a stopping condition is met. Each split is determined by selecting the feature that best separates the data according to a certain criterion, typically Gini impurity or information gain for classification tasks.

Randomization: The "random" aspect of Random Forests comes from two main sources of randomness:

a. Bootstrapping: During training, a random subset of the original training data is selected with replacement. This means that some data points may appear multiple times in a subset, while others may not appear at all. This process is known as bootstrapping, and it helps create diverse training sets for each tree.

b. Feature Randomization: When building each decision tree, a random subset of features (columns) is considered at each split point. This ensures that the individual trees have different views of the data and prevents any single feature from dominating the decision-making process.

Training Multiple Decision Trees: Random Forest trains a predefined number of decision trees (an ensemble). Each tree is built independently using a different bootstrap sample and feature subset.

Voting or Averaging: For classification tasks, each tree in the forest makes a prediction. The most common strategy for combining these predictions is majority voting:

the class that receives the most votes from the individual trees is considered the final prediction. For regression tasks, the individual tree predictions are typically averaged to produce the final prediction. Robustness and Generalization: Random Forests are known for their robustness against overfitting, which is a common issue in decision trees. By aggregating the predictions of multiple trees and introducing randomness into the model-building process, Random Forests reduce the risk of overfitting and provide better generalization to unseen data.

Tuning Hyperparameters: There are some important hyperparameters to consider when using Random Forests, such as the number of trees in the forest, the maximum depth of each tree, and the size of the feature subsets. Tuning these hyperparameters can have a significant impact on the performance of the model.Random Forests have many advantages, including their ability to handle high- dimensional data, handle missing values, and provide feature importances. They are widely used in various applications, including image classification, text classification, and medical diagnosis, among others, due to their robustness and high predictive accuracy.

Implementing Gradient Boosting :

Gradient Boosting is a powerful ensemble learning technique in machine learning that is used for both regression and classification tasks. It is a sequential, additive modeling technique that builds an ensemble of weak learners (typically decision trees) to create a strong predictive model. Gradient Boosting works by iteratively correcting the errors of the previous model in the ensemble, thus reducing the overall prediction error.

Initialization: Gradient Boosting begins with an initial prediction, which is usually a simple estimate, like the mean of the target variable for regression or the majority class for classification.

Residual Calculation: In each iteration (or boosting round), Gradient Boosting calculates the residuals or errors of the previous model's predictions compared to the actual target values. These residuals represent the mistakes made by the current ensemble.

Building Weak Learners: Gradient Boosting uses weak learners, typically decision trees with limited depth, as base models. These trees are called "weak" because they are relatively simple and may not perform well individually.

Fitting Weak Learners: In each boosting round, a new weak learner is fitted to the residuals of the previous model. The goal is to find a model that can capture and correct the errors made by the existing ensemble. The new weak learner is trained to predict these residuals.

Learning Rate: A hyperparameter called the learning rate (or shrinkage) controls the contribution of each weak learner to the ensemble. A lower learning rate makes the process more robust but requires more boosting rounds.

Update the Ensemble: The new weak learner is added to the ensemble, and its predictions are combined with the predictions of the previous models. This combination can be a weighted sum for regression tasks or a weighted vote for classification tasks.

Repeat: Steps 3 to 6 are repeated for a predefined number of boosting rounds or until a certain stopping criterion is met. Common stopping criteria include a maximum number of rounds, achieving a specific level of accuracy, or when the model's performance on a validation dataset starts to degrade.

Final Prediction: After all boosting rounds are completed, the final prediction is made by aggregating the predictions of all weak learners in the ensemble. For regression tasks,

this is typically a sum of the individual predictions, and for classification tasks, it's often a weighted vote.

High Accuracy: Gradient Boosting is known for its high predictive accuracy and is often considered one of the best off-the-shelf machine learning algorithms.

Handles Different Types of Data: It can handle a wide range of data types, including numerical, categorical, and text data.

Feature Importance: Gradient Boosting provides a measure of feature importance, which can help identify which features are most influential in making predictions.

Robust to Overfitting: It is less prone to overfitting compared to individual decision trees, thanks to the iterative learning process. Some popular implementations of Gradient Boosting include Gradient Boosting Machines (GBM), XGBoost, LightGBM, and CatBoost, each with its own optimizations and features. These libraries have made Gradient Boosting widely accessible and efficient for various machine learning tasks.

Implementing Naive Bayes :

Naive Bayes falls under the umbrella of supervised machine learning algorithms that are primarily used for classification. In this context, "supervised" tells us that the algorithm is trained with both input features and categorical outputs (i.e., the data includes the correct desired output for each point, which the algorithm should predict).But why is the algorithm called "naive" This is because the classifier assumes that the input features that go into the model are independent of each other. Hence, changing one input feature won't affect any of the others. It's therefore naive in the sense that this assumption may or may not be true, and it most probably isn't One of the significant advantages of Naive Bayes is that it uses a

probabilistic approach; all the computations are done on the fly in real time, and outputs are generated instantaneously.

CHAPTER 5

RESULTS & DISCUSSION

5.1 Performance Parameters / Testing

The performance of the Predictive Power Machine Learning Model for Machine Failure Status was assessed using various evaluation metrics and testing techniques to ensure accuracy, reliability, and efficiency. The key performance parameters considered include accuracy, precision, recall, F1-score, computational efficiency, and scalability

Accuracy

Accuracy is the proportion of correctly predicted machine failure statuses to the total predictions made. It provides an overall measure of the model's effectiveness. However, accuracy alone is not always sufficient, especially when dealing with imbalanced datasets where one class is more frequent than the other

Precision

Precision is the ratio of correctly predicted failure cases to the total predicted failures. This metric is crucial in industrial maintenance, as a high precision value ensures that the model does not falsely classify non-failing machines as failing, which could lead to unnecessary maintenance costs

Recall (Sensitivity)

Recall measures the proportion of actual machine failures that the model correctly identifies. A high recall value ensures that potential failures are not overlooked, which is critical for minimizing unexpected downtimes in industrial settings

F1-Score

The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. It is particularly useful when dealing with imbalanced datasets, as it considers both false positives and false negatives in the evaluation

Computational Efficiency

The efficiency of the machine learning model was tested based on its training and prediction time. Faster models ensure quick decision-making, which is crucial for real-time predictive maintenance. The implementation of Naïve Bayes, Gradient Boosting, and Random Forest classifiers was compared to find an optimal balance between accuracy and efficiency

Scalability and Resource Utilization

The ability of the model to handle increasing amounts of data was evaluated to ensure smooth operation in real-world industrial applications. The model's CPU and memory usage were monitored to prevent resource-intensive processes from slowing down system performance

Testing Methodology

To ensure robust performance evaluation, multiple testing strategies were employed:

Cross-validation: Used to assess the generalizability of the model.

A/B testing: Conducted to compare different algorithms and select the best-performing one.

Real-time testing: The model was tested on live industrial data to check its real-world applicability.

By focusing on these performance parameters, the Predictive Power Machine Learning Model for Machine Failure Status ensures high accuracy, efficiency, and reliability, making it suitable for industrial predictive maintenance applications

5.2 Results & Discussion

The Predictive Power Machine Learning Model for Machine Failure Status was designed to minimize machine downtime and optimize industrial maintenance strategies. The results demonstrate that machine learning models can accurately predict machine failures, helping industries transition from reactive maintenance to proactive decision-making

The evaluation of various machine learning models, including Random Forest, Gradient Boosting, and Naive Bayes, provided insights into their performance. Random Forest and Gradient Boosting performed the best, achieving high accuracy in failure detection, while Naive Bayes, though computationally efficient, had lower accuracy due to its feature independence assumption. The F1-score analysis confirmed that Random Forest and Gradient Boosting maintained a balance between precision and recall, making them reliable for failure prediction

Data preprocessing played a critical role in enhancing model accuracy. Cleaning the dataset, handling missing values, and applying feature engineering techniques improved the overall performance. Additionally, real-time testing of the models using industrial data confirmed their effectiveness, and the Django-based web interface provided a user-friendly way to input machine data and receive failure predictions

Despite the promising results, some challenges were encountered. The dataset imbalance, where failure cases were significantly fewer than non-failure instances, affected model training. Techniques like oversampling and synthetic data generation were implemented to address this issue. Another limitation was computational complexity, particularly with Gradient Boosting, which required higher processing power

In conclusion, the study highlights the potential of machine learning models in predictive maintenance, reducing unexpected breakdowns and improving operational efficiency. The

Random Forest and Gradient Boosting models emerged as the best-performing algorithms, offering a balance between accuracy and efficiency. Future improvements could include integrating deep learning techniques such as LSTMs for time-series analysis and incorporating IoT devices for real-time monitoring

Model	Precision	Recall	F1 score
Naive bayes	28%	39%	31%
Gradient boosting	99%	99%	99%
Random forest	100%	100%	100%

Fig 5.2.1 Metrics Score

CHAPTER 6

CONCLUSION:

The project successfully demonstrates that Predictive Power Machine Learning Models can be utilized to forecast machine failures with high accuracy. By leveraging large datasets and advanced machine learning algorithms, industries can transition from reactive to predictive maintenance, minimizing unplanned downtime and optimizing operational efficiency.

The deployment of the model within a Django framework allows seamless user interaction, enabling businesses to access real-time failure predictions through a web interface.

The results indicate that Random Forest and Gradient Boosting were the most effective models, providing a balance between accuracy, computational efficiency, and real-time applicability. However, challenges such as data imbalance and computational complexity were identified, which were addressed through techniques like oversampling and hyperparameter tuning.

Future work can focus on enhancing model accuracy by incorporating deep learning techniques such as LSTMs for time-series analysis. Additionally, integrating IoT devices and cloud-based architectures would allow for more efficient data processing and real-time failure predictions at scale. Collaborating with domain experts to fine-tune models for specific industries can further enhance the impact of predictive maintenance systems

FUTURE WORKS:

Future work could involve improving model accuracy by incorporating more advanced deep learning techniques such as LSTMs for time-series analysis. Additionally, integrating IoT devices and cloud-based architectures would allow for more efficient data processing and real-time failure predictions at scale .Collaboration with domain experts to fine-tune models for specific industries could further enhance the impact of these predictive maintenance systems.

APPENDICES

A.1 SDG Goals

SDG 9 – Industry, Innovation, and Infrastructure

→ Enhancing industrial efficiency and infrastructure resilience through predictive maintenance.

SDG 12 – Responsible Consumption and Production

→ Reducing waste and optimizing resource use by preventing machine failures.

SDG 13 – Climate Action

→ Lowering energy consumption and emissions through improved machine performance**.**

A.2 Source Code

MODULE 1:

DATA PREPROCESSING AND DATA CLEANING

```
import pandas as pd

import numpy as np

import warnings

warnings.filterwarnings(‘ignore’)

df = pd.read_csv(‘Dataset.csv’)

df.head()
```

```
df.tail()

del df[‘UDI’]

del df[‘Product ID’]

del df[‘Type’]

del df[‘Target’]

df.describe()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = [‘Failure Type’]

for i in var:

df[i] = le.fit_transform(df[i]).astype(int)

df.columns

df.size

df.columns

df.isnull().sum()

df = df.dropna()

df[‘Failure Type’].unique()

df.describe()

df.corr()
```

```
df.info()

df[“Failure Type”].value_counts()

pd.Categorical(df[“Rotational speed [rpm]”]).describe()

df.duplicated()

sum(df.duplicated())
```

MODULE 2:

DATA VISUALIZATION AND DATA ANALYSIS

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings(‘ignore’)

df = pd.read_csv(‘Dataset.csv’)

df.head()

df.columns

del df[‘UDI’]

del df[‘Product ID’]

del df[‘Type’]
```

```
del df[‘Target’]

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = [‘Failure Type’]

for i in var:

df[i] = le.fit_transform(df[i]).astype(int)

plt.figure(figsize=(12,7))

sns.countplot(x=’Failure Type’,data=df)

plt.figure(figsize=(15,5))

plt.subplot(1,2,1)

plt.hist(df[‘Failure Type’],color=’red’)

plt.subplot(1,2,2)

plt.hist(df[‘Rotational speed [rpm]’],color=’blue’)

df.hist(figsize=(15,55),layout=(15,4), color=’orange’)

plt.show()

plt.boxplot(df[“Process temperature [K]”])

plt.scatter(df[‘Torque [Nm]’],df[“Tool wear [min]”]) # scatter, plot, triplot,

stackplot

df[‘Torque [Nm]’].plot(kind=’density’)
```

```
sns.barplot(df[‘Air temperature [K]’], color=’purple’)

# barplot, boxenplot, boxplot, countplot, displot, distplot, ecdfplot, histplot,

kdeplot, pointplot, violinplot, stripplot

fig, ax = plt.subplots(figsize=(20,15))

sns.heatmap(df.corr(),annot = True, fmt=’0.2%’,cmap = ‘autumn’,ax=ax)

def plot(df, variable):

dataframe_pie = df[variable].value_counts()

ax = dataframe_pie.plot.pie(figsize=(9,9), autopct=’%1.2f%%’, fontsize = 10)

ax.set_title(variable + ‘\n’, fontsize = 10)

return np.round(dataframe_pie/df.shape[0]*100,2)

plot(‘Failure Type’)
```

MODULE 3:

COMPLEMENT NAIVE BAYES ALGORITHM

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings(‘ignore’)
```

```
df = pd.read_csv('Dataset.csv')

df.head()

del df['UDI']

del df['Product ID']

del df['Target']

del df['Type']

df=df.dropna()

df.columns

df.head()

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = ['Failure Type']

for i in var:

df[i] = le.fit_transform(df[i]).astype(int)

df['Failure Type'].dtype

df.head()

x1 = df.drop(labels='Failure Type', axis=1)

y1 = df.loc[:,'Failure Type']

import imblearn
```

```
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT      : ", Counter(y1))
print("OVER SAMPLING DATA COUNT : ", Counter(y))
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42, stratify=y)
print("NUMBER OF TRAIN DATASET : ", len(x_train))
print("NUMBER OF TEST DATASET : ", len(x_test))
print("TOTAL NUMBER OF DATASET : ", len(x_train)+len(x_test))
print("NUMBER OF TRAIN DATASET : ", len(y_train))
print("NUMBER OF TEST DATASET : ", len(y_test))
print("TOTAL NUMBER OF DATASET : ", len(y_train)+len(y_test))
from sklearn.naive_bayes import ComplementNB
CNB = ComplementNB()
CNB.fit(x_train,y_train)
predicted = CNB.predict(x_test)
```

```
from sklearn.metrics import classification_report
cr = classification_report(y_test,predicted)
print('THE CLASSIFICATION REPORT OF COMPLEMENTNB:\n\n&',cm)
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test,predicted)
print('THE CONFUSION MATRIX SCORE OF
COMPLEMENTNB:\n\n\n',cm)
from sklearn.model_selection import cross_val_score
accuracy = cross_val_score(CNB, x, y, scoring='accuracy')
print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n&',
accuracy*100)
from sklearn.metrics import accuracy_score
a = accuracy_score(y_test,predicted)
print("THE ACCURACY SCORE OF COMPLEMENTNB IS :",a*100)
from sklearn.metrics import hamming_loss
hl = hamming_loss(y_test,predicted)
print("THE HAMMING LOSS OF COMPLEMENTNB IS :",hl*100)
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF
COMPLEMENTNB\n\n', cmap=plt.cm.cool):
```

```
plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

cm1=confusion_matrix(y_test, predicted)

print('THE CONFUSION MATRIX SCORE OF COMPLEMENTNB:\n\n')

print(cm)

plot_confusion_matrix(cm)

import matplotlib.pyplot as plt

df2 = pd.DataFrame()

df2["y_test"] = y_test

df2["predicted"] = predicted

df2.reset_index(inplace=True)

plt.figure(figsize=(20, 5))

plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')

plt.plot(df2["y_test"][:100], marker='o', linestyle='dashed', color='green')

plt.show()
```

MODULE 4:

GRADIENT BOOSTING CLASSIFIER ALGORITHM

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

data = pd.read_csv('Dataset.csv')

data.head()

df=data.dropna()

df.columns

df.head()

from sklearn.preprocessing import LabelEncoder

lab=LabelEncoder()

df['Type']=lab.fit_transform(df['Type'])

del df['UDI']

del df['Product ID']

del df['Type']
```

```
del df[‘Target’]

x1 = df.drop(labels=’Failure Type’, axis=1)

y1 = df.loc[‘Failure Type’]

import imblearn

from imblearn.over_sampling import RandomOverSampler

from collections import Counter

ros =RandomOverSampler(random_state=42)

x,y=ros.fit_resample(x1,y1)

print(“OUR DATASET COUNT : “, Counter(y1))

print(“OVER SAMPLING DATA COUNT : “, Counter(y))

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,

random_state=42, stratify=y)

print(“NUMBER OF TRAIN DATASET : “, len(x_train))

print(“NUMBER OF TEST DATASET : “, len(x_test))

print(“TOTAL NUMBER OF DATASET : “, len(x_train)+len(x_test))

print(“NUMBER OF TRAIN DATASET : “, len(y_train))

print(“NUMBER OF TEST DATASET : “, len(y_test))

print(“TOTAL NUMBER OF DATASET : “, len(y_train)+len(y_test))
```

```
from sklearn.ensemble import GradientBoostingClassifier

GRB = GradientBoostingClassifier()

GRB.fit(x_train,y_train)

predicted = GRB.predict(x_test)

from sklearn.metrics import classification_report

cr = classification_report(y_test,predicted)

print('THE CLASSIFICATION REPORT OF GradientBoosting:\n\n',cr)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,predicted)

print('THE CONFUSION MATRIX SCORE OF GradientBoosting:\n\n\n',cm)

from sklearn.model_selection import cross_val_score

accuracy = cross_val_score(GRB, x, y, scoring='accuracy')

print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n',

accuracy*100)

from sklearn.metrics import accuracy_score

a = accuracy_score(y_test,predicted)

print("THE ACCURACY SCORE OF GradientBoosting IS :",a*100)

from sklearn.metrics import hamming_loss

hl = hamming_loss(y_test,predicted)
```

```
print(“THE HAMMING LOSS OF GradientBoosting IS :”,hl*100)

def plot_confusion_matrix(cm, title=’THE CONFUSION MATRIX SCORE OF

GradientBoosting\n\n’, cmap=plt.cm.cool):

plt.imshow(cm, interpolation=’nearest’, cmap=cmap)

plt.title(title)

plt.colorbar()

cm1=confusion_matrix(y_test, predicted)

print(‘THE CONFUSION MATRIX SCORE OF GradientBoosting:\n\n’)

print(cm)

plot_confusion_matrix(cm)

import matplotlib.pyplot as plt

df2 = pd.DataFrame()

df2[“y_test”] = y_test

df2[“predicted”] = predicted

df2.reset_index(inplace=True)

plt.figure(figsize=(20, 5))

plt.plot(df2[“predicted”][:100], marker=’x’, linestyle=’dashed’, color=’red’)

plt.plot(df2[“y_test”][:100], marker=’o’, linestyle=’dashed’, color=’green’)

plt.show()
```

MODULE 5:

RANDOM FOREST CLASSIFIER ALGORITHM

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

data = pd.read_csv('Dataset.csv')

data.head()

del data['UDI']

del data['Product ID']

del data['Target']

del data['Type']

df = data.dropna()

df.columns

# df = df1.rename({'Air temperature [K]':'Air_temperature','Process temperature

[K]':'Process_temperature_K','Rotational speed [rpm]':'Rotational_speed_rpm',

'Torque [Nm]':'Torque_Nm', 'Tool wear [min]':'Tool_wear_min', 'Failure
```

```
Type':'Failure_Type'},axis=1)

df['Failure_Type'].unique()

# from sklearn.preprocessing import LabelEncoder

# lab=LabelEncoder()

# df['Failure_Type']=lab.fit_transform(df['Failure_Type'])

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

var = ['Failure_Type']

for i in var:

df[i] = le.fit_transform(df[i]).astype(int)

df['Failure_Type'].unique()

df.head()

df.info()

x1 = df.drop(labels=&#39;Failure_Type&#39;, axis=1)

y1 = df.loc[:,'Failure_Type']

x1

df.head()

df.info()

import imblearn
```

```
from imblearn.over_sampling import RandomOverSampler
from collections import Counter
ros =RandomOverSampler(random_state=42)
x,y=ros.fit_resample(x1,y1)
print("OUR DATASET COUNT : ", Counter(y1))
print("OVER SAMPLING DATA COUNT : ", Counter(y))
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.20,
random_state=42, stratify=y)
print("NUMBER OF TRAIN DATASET : ", len(x_train))
print("NUMBER OF TEST DATASET : ", len(x_test))
print("TOTAL NUMBER OF DATASET : ", len(x_train)+len(x_test))
print("NUMBER OF TRAIN DATASET : ", len(y_train))
print("NUMBER OF TEST DATASET : ", len(y_test))
print("TOTAL NUMBER OF DATASET : ", len(y_train)+len(y_test))
# df['Type'].unique()
from sklearn.ensemble import RandomForestClassifier
lr = RandomForestClassifier()
lr.fit(x_train,y_train)
```

```
predicted = lr.predict(x_test)

from sklearn.metrics import classification_report

cr = classification_report(y_test,predicted)

print('THE CLASSIFICATION REPORT OF RANDOM FOREST

CLASSIFIER:\n\n',cr)

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test,predicted)

print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST

CLASSIFIER:\n\n\n',cm)

from sklearn.model_selection import cross_val_score

accuracy = cross_val_score(lr, x, y, scoring='accuracy')

print('THE CROSS VALIDATION TEST RESULT OF ACCURACY :\n\n\n',

accuracy*100)

from sklearn.metrics import accuracy_score

a = accuracy_score(y_test,predicted)

print("THE ACCURACY SCORE OF RANDOM FOREST CLASSIFIER IS

:",a*100)

from sklearn.metrics import hamming_loss

hl = hamming_loss(y_test,predicted)
```

```
print("THE HAMMING LOSS OF RANDOM FOREST CLASSIFIER IS
:",hl*100)
def plot_confusion_matrix(cm, title='THE CONFUSION MATRIX SCORE OF
RANDOM FOREST CLASSIFIER\n\n', cmap=plt.cm.cool):
plt.imshow(cm, interpolation='nearest', cmap=cmap)
plt.title(title)
plt.colorbar()
cm1=confusion_matrix(y_test, predicted)
print('THE CONFUSION MATRIX SCORE OF RANDOM FOREST
CLASSIFIER:\n\n')
print(cm)
plot_confusion_matrix(cm)
import matplotlib.pyplot as plt
df2 = pd.DataFrame()
df2["y_test"] = y_test
df2["predicted"] = predicted
df2.reset_index(inplace=True)
plt.figure(figsize=(20, 5))
plt.plot(df2["predicted"][:100], marker='x', linestyle='dashed', color='red')
```

```
plt.plot(df2[“y_test”][:100], marker=’o’, linestyle=’dashed’, color=’green’)

plt.show()

import joblib

joblib.dump(lr,’Model.pkl’)
```

DEPLOY

```
from django.shortcuts import render, redirect

from django.urls import reverse_lazy

from django.contrib.auth.views import LoginView, PasswordResetView,

PasswordChangeView

from django.contrib import messages

from django.contrib.messages.views import SuccessMessageMixin

from django.views import View

from django.contrib.auth.decorators import login_required

from django.contrib.auth import logout as auth_logout

import numpy as np

import joblib

from .forms import RegisterForm, LoginForm, UpdateUserForm,

UpdateProfileForm

from . models import UserPredictModel
```

```
from .forms import UserPredictDataForm

def home(request):
    return render(request, ‘users/home.html’)

@login_required(login_url=’users-register’)
def index(request):
    return render(request, ‘app/index.html’)

class RegisterView(View):
    form_class = RegisterForm
    initial = {‘key’: ‘value’}
    template_name = ‘users/register.html’

def dispatch(self, request, *args, **kwargs):
        # will redirect to the home page if a user tries to access the register page
while logged in
        if request.user.is_authenticated:
            return redirect(to=’/’)
  # else process dispatch as it otherwise normally would
        return super(RegisterView, self).dispatch(request, *args, **kwargs)
def get(self, request, *args, **kwargs):
        form = self.form_class(initial=self.initial)
```

```
        return render(request, self.template_name, {‘form’: form})
 def post(self, request, *args, **kwargs):
        form = self.form_class(request.POST)
   if form.is_valid():
            form.save()
username = form.cleaned_data.get(‘username’)
            messages.success(request, f’Account created for {username}’)
return redirect(to=’login’)
 return render(request, self.template_name, {‘form’: form})
# Class based view that extends from the built in login view to add a remember
me functionality
class CustomLoginView(LoginView):
    form_class = LoginForm
def form_valid(self, form):
        remember_me = form.cleaned_data.get(‘remember_me’)
 if not remember_me:
            # set session expiry to 0 seconds. So it will automatically close the
session after the browser is closed.
            self.request.session.set_expiry(0)
```

```
 # Set session as modified to force data updates/cookie to be saved.
            self.request.session.modified = True
  # else browser session will be as long as the session cookie time
"SESSION_COOKIE_AGE" defined in settings.py
        return super(CustomLoginView, self).form_valid(form)
class ResetPasswordView(SuccessMessageMixin, PasswordResetView):
    template_name = 'users/password_reset.html'
    email_template_name = 'users/password_reset_email.html'
    subject_template_name = 'users/password_reset_subject'
    success_message = "We've emailed you instructions for setting your
password, " \
                      "if an account exists with the email you entered. You should
receive them shortly." \
                      "If you don't receive an email, "\
                      "please make sure you've entered the address you registered with,
and check your spam folder."
    success_url = reverse_lazy('users-home')
class ChangePasswordView(SuccessMessageMixin, PasswordChangeView):
    template_name = 'users/change_password.html';
```

```
    success_message = “Successfully Changed Your Password”
    success_url = reverse_lazy(‘users-home’)
@login_required
def profile(request):
    if request.method == ‘POST’:
        user_form = UpdateUserForm(request.POST, instance=request.user)
        profile_form = UpdateProfileForm(request.POST, request.FILES,
instance=request.user.profile)
  if user_form.is_valid() and profile_form.is_valid():
            user_form.save()
            profile_form.save()
             messages.success(request, ‘Your profile is updated successfully’)
            return redirect(to=’users-profile’)
    else:
        user_form = UpdateUserForm(instance=request.user)
        profile_form = UpdateProfileForm(instance=request.user.profile)
         return render(request, ‘/profile.html’, {‘user_form’: user_form,
‘profile_form’: profile_form})
        Model = joblib.load(‘D:/FINAL/ITPML37  - FINAL/ITPML37-FINAL
```

```
CODING/Deployment/users/Model.pkl')
def model(request):
 if request.method == 'POST':
fields =['Air_temperature', 'Process_temperature', 'Rotational_speed',
'Torque', 'Tool_wear']
 form = UserPredictDataForm(request.POST)
features = []
for i in fields:
    info = float(request.POST[i])
    features.append(info)
 Final_features = [np.array(features, dtype=int)]
prediction = Model.predict(Final_features)
 actual_output = prediction[0]
 print(actual_out)
if actual_output == 0:
    actual_output1 = 'Heat Dissipation Failure'
elif actual_output == 1:
    actual_output1 = 'No Failure'
 elif actual_output == 2:
```

```
            actual_output1 = 'Overstrain Failure'
        elif actual_output == 3:
            actual_output1 = 'Power Failure'
        elif actual_output == 4:
            actual_output1 = 'Random Failures
        elif actual_output == 5:
            actual_output1 = 'Tool Wear Failure'
        print("output",actual_output1)
        if form.is_valid():
            print('Saving data in Form')
            form_instance = form.save()  # Save form data but don't commit to DB
        yet
            form_instance.save()
        data = UserPredictModel.objects.latest('id')
        data.Label = actual_output1
        data.save()
        return render(request, 'app/result.html', {'form':form,
'prediction_text':actual_output1})
    else:
```

```
        print('Else working')
        form = UserPredictDataForm(request.POST)
    return render(request, 'app/model.html', {'form':form})
    from .models import Profile
    def profile_database(request):
    data=Profile.objects.all()
 return render(request,'app/profile_list.html',{'database':data})
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import base64
from io import BytesIO
from django.shortcuts import render
def Basic_report(request):
    return render(request,'app/Basic_report.html')
def Metrics_report(request):
    return render(request,'app/Metrics_report.html')
def model_db(request):
    data = UserPredictModel.objects.all()
```

```
    return render(request, ‘app/model_db.html’, {‘data’: data})
def logout_view(request):
    auth_logout(request)
    return redirec(‘/’)
```

A.3 Screenshots

A.3.1 Login Page

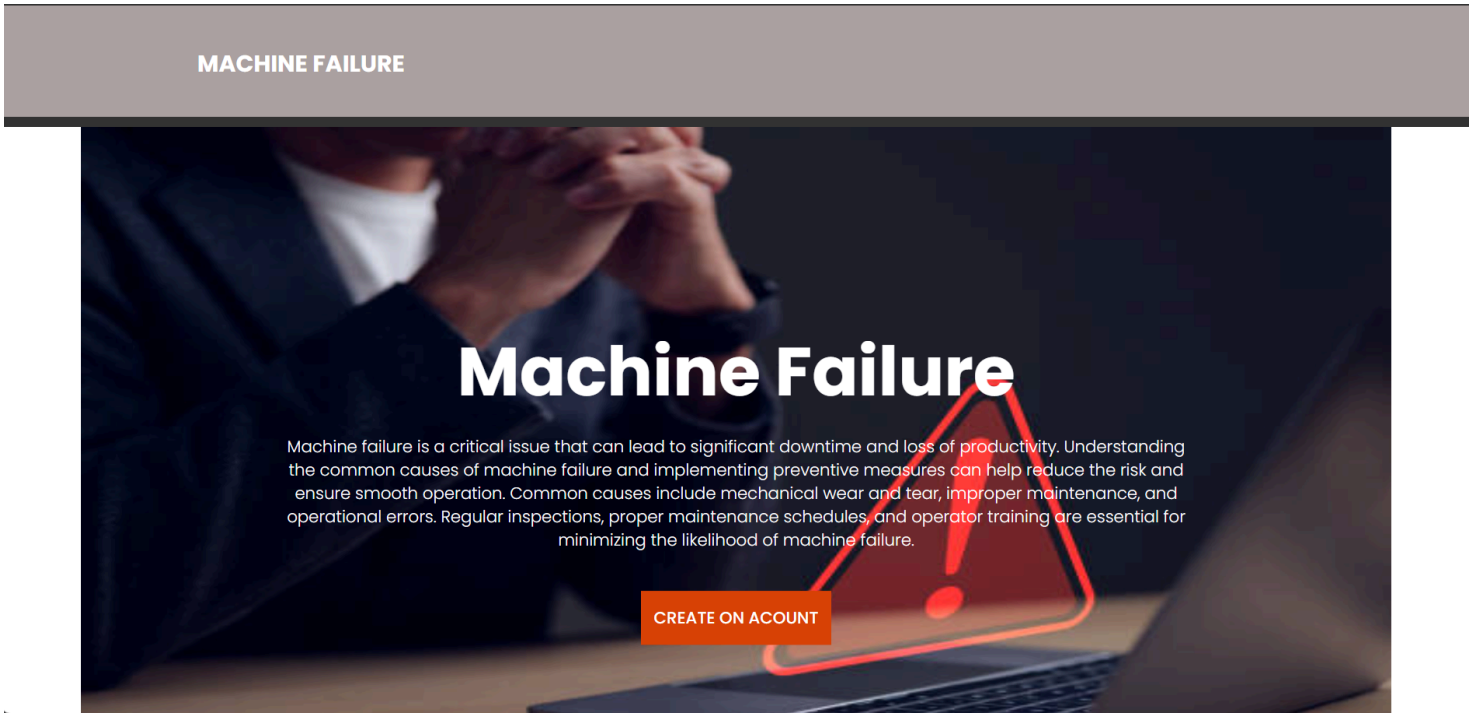


Fig:A.3.1 Login Page

A.3.2 Sign up page:

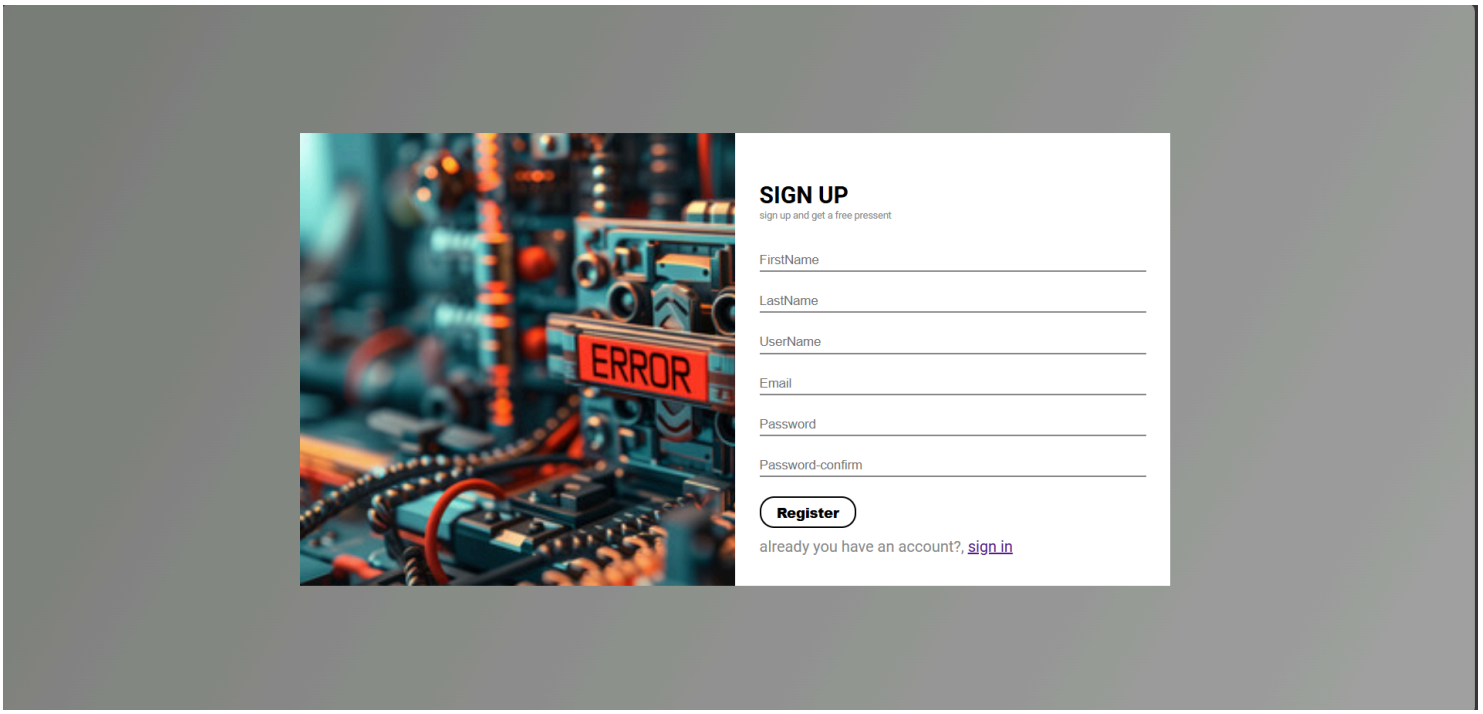


Fig:A.3.2 Sign up page

A.3.3 Sign In page:

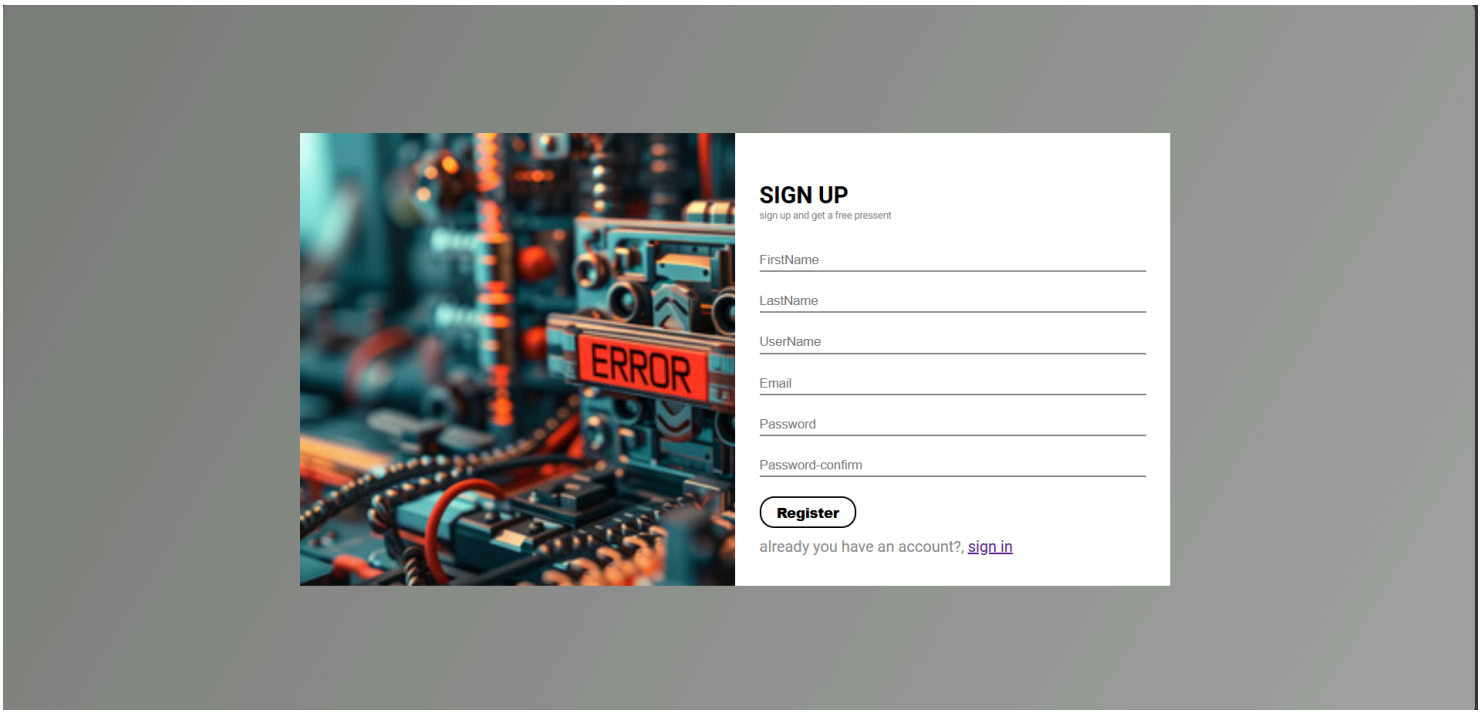


Fig:A.3.3 Sign In page

A.3.4 Creating New Profile:

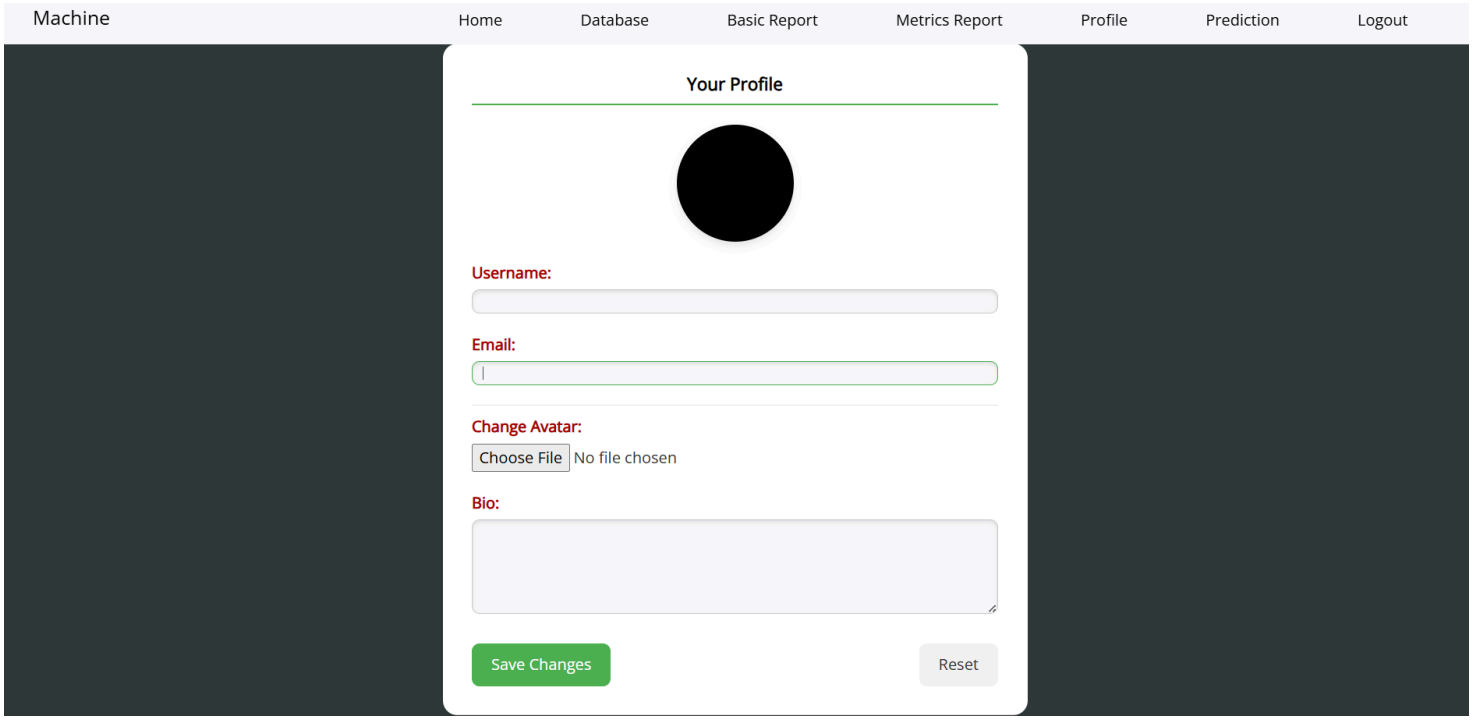


Fig:A.3.4 Creating New Profile

A.3.5 Dashboard:

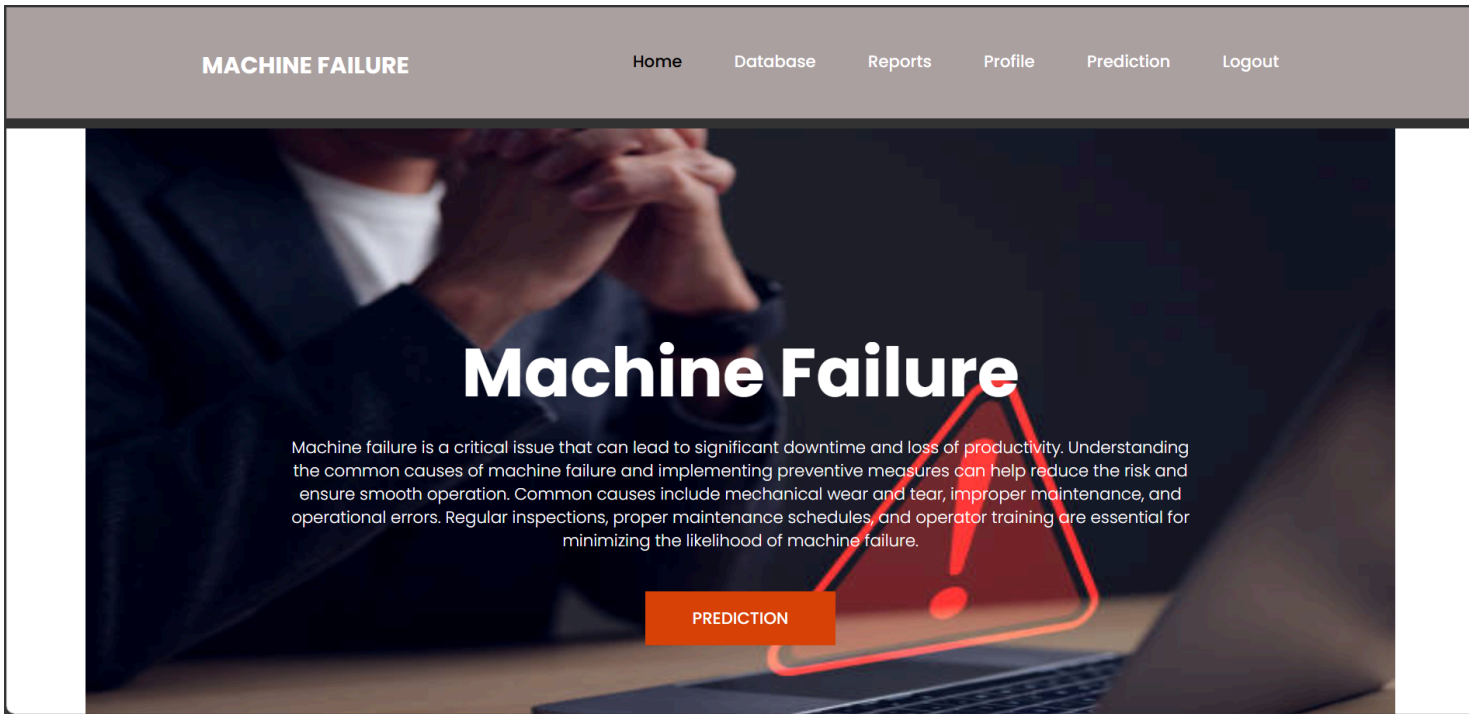


Fig:A.3.5 Dashboard

A.3.6 Informations:

About Informations

Machine failure can be categorized as complete or partial, depending on the severity of the issue and how functional the equipment is. When a machine fails, it can cost a company a lot of money in repairs, lost productivity, and downtime.

FAILURE STATUS

Machine failure is a critical issue that can lead to significant downtime and loss of productivity. Understanding the common causes of machine failure and implementing preventive measures can help reduce the risk and ensure smooth operation. By addressing these common causes and implementing preventive measures, you can minimize the likelihood of machine failure and ensure smooth operation of your machinery.

Technologies

Fig:A.3.6 Informations

A.3.7 Reports:

Pandas Profiling Report — Overview | Variables | Interactions | Correlations | Missing values | Sample

Overview

Brought to you by YData

Overview | Alerts 6 | Reproduction

Dataset statistics

Number of variables	6
Number of observations	10000
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	468.9 KiB
Average record size in memory	48.0 B

Variable types

Numeric	5
Categorical	1

Pandas Profiling Report — Overview | Variables | Interactions | Correlations | Missing values | Sample

Variables

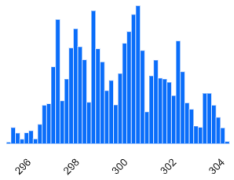
Select Columns

Air_temperature

Real number (ℝ)

High correlation

Distinct	93	**Minimum**	295.3
Distinct (%)	0.9%	**Maximum**	304.5
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	300.00493	**Memory size**	78.2 KiB



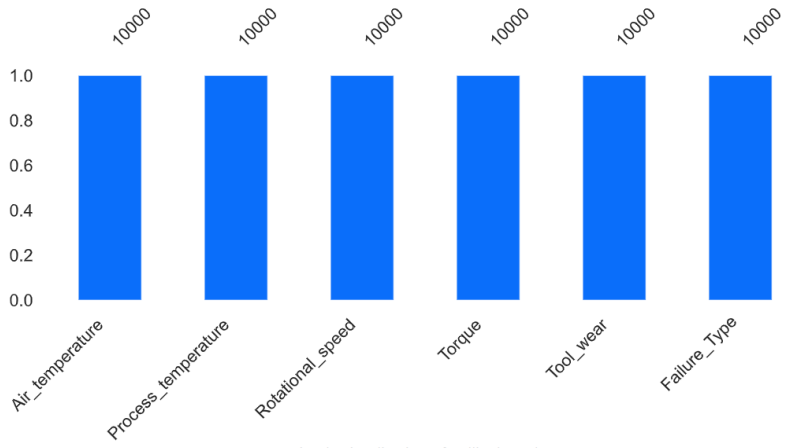
More details

Pandas Profiling Report

Overview Variables Interactions Correlations Missing values Sample

Missing values

Count Matrix



A simple visualization of nullity by column.

Pandas Profiling Report

Overview Variables Interactions Correlations Missing values Sample

Overview

Brought to you by YData

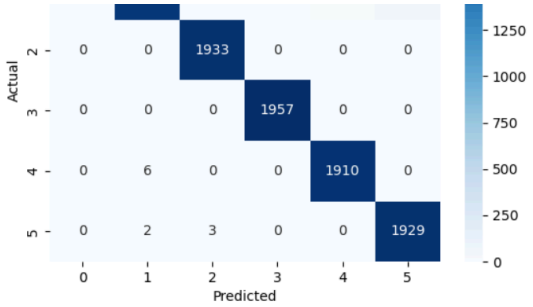
Overview Alerts 6 Reproduction

Alerts

`Air_temperature` is highly overall correlated with `Process_temperature`	High correlation
`Process_temperature` is highly overall correlated with `Air_temperature`	High correlation
`Rotational_speed` is highly overall correlated with `Torque`	High correlation
`Torque` is highly overall correlated with `Rotational_speed`	High correlation
`Failure_Type` is highly imbalanced (88.7%)	Imbalance
`Tool_wear` has 120 (1.2%) zeros	Zeros

Variables

Select Columns



Classification Report

	precision	recall	f1-score	support
0	0.997349	1.000000	0.998673	1881.000000
1	0.995751	0.955657	0.975293	1962.000000
2	0.992809	1.000000	0.996392	1933.000000
3	0.995422	1.000000	0.997706	1957.000000
4	0.989637	0.996868	0.993240	1916.000000
5	0.978691	0.997415	0.987964	1934.000000
accuracy	0.991539	0.991539	0.991539	0.991539
macro avg	0.991610	0.991657	0.991544	11583.000000
weighted avg	0.991604	0.991539	0.991482	11583.000000

Accuracy

0.9915393248726582

Metrics Report

Confusion Matrix

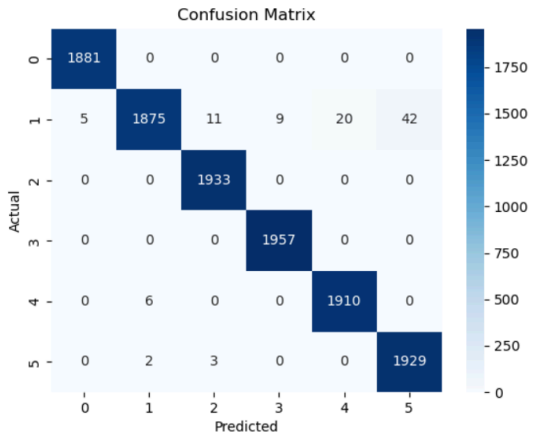


Fig:A.3.7 Reports

A.3.8 Machine Inputs Model Form:

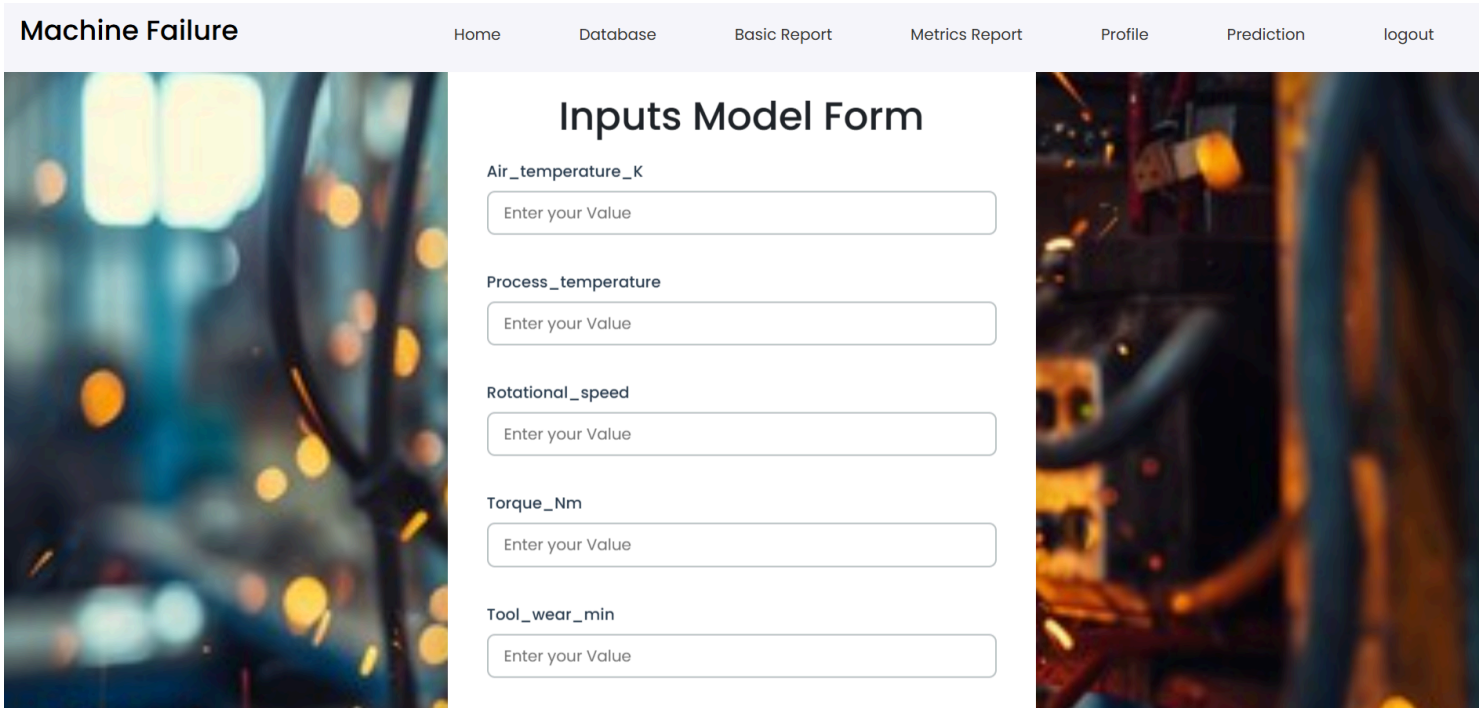


Fig:A.3.8 Machine Inputs Model Forms

A.3.9 Prediction:

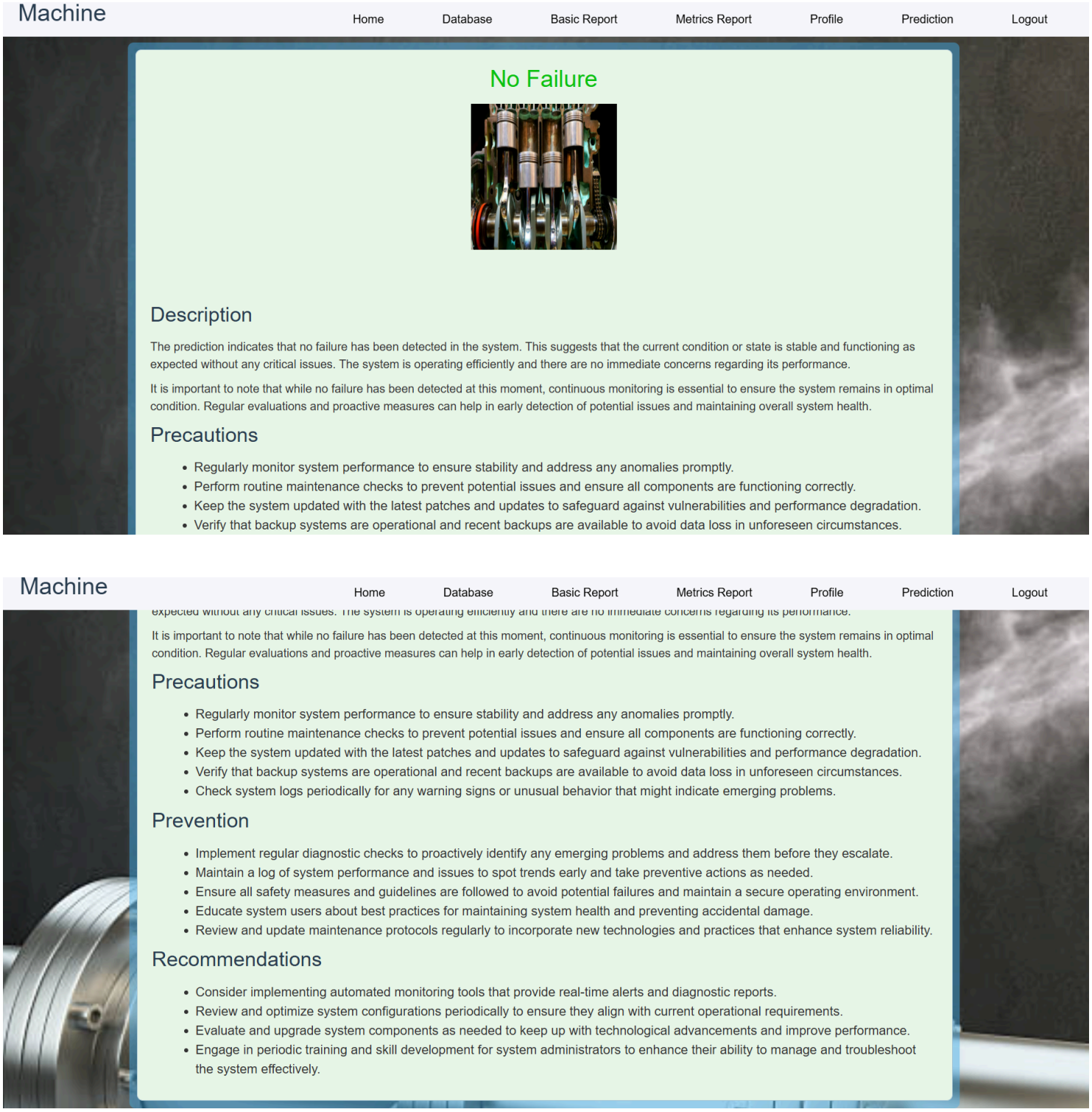
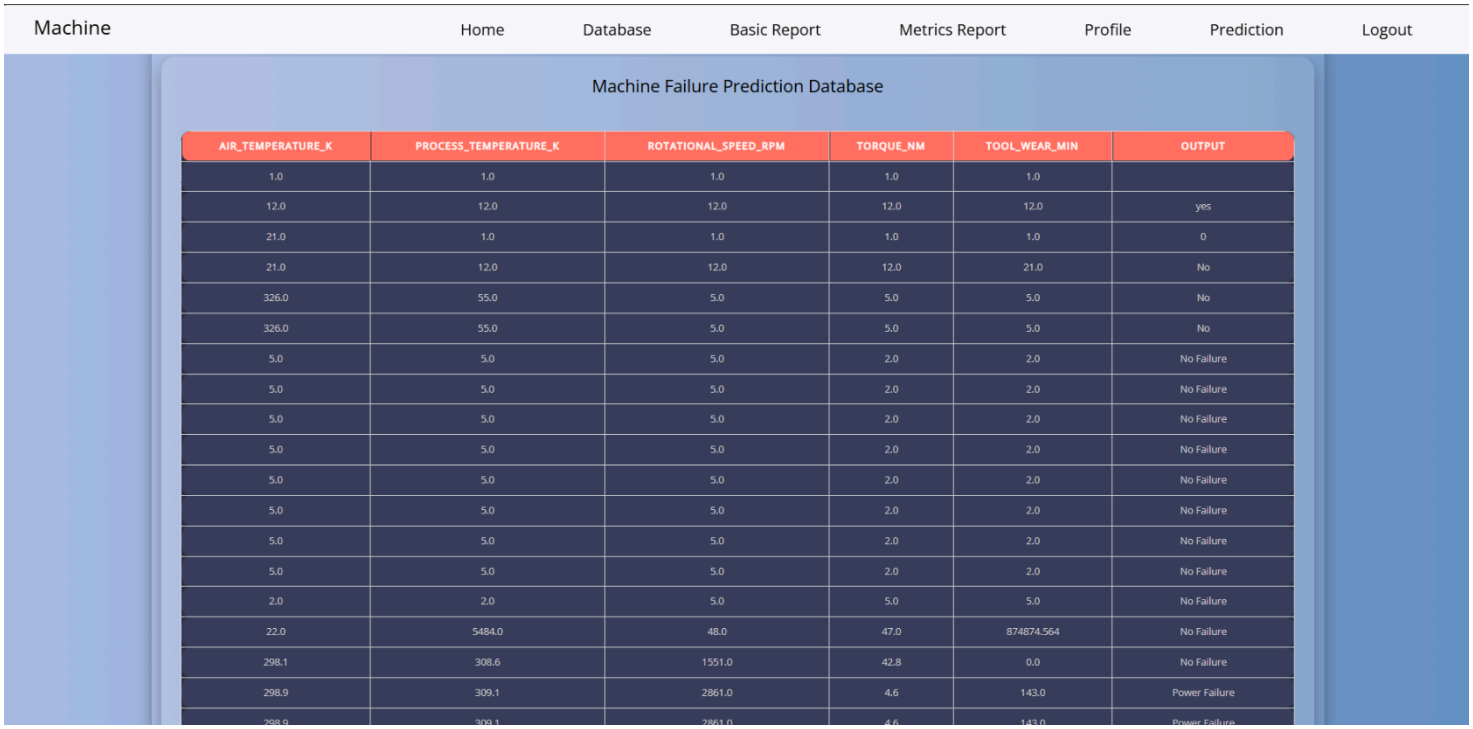


Fig:A.3.9 Prediction

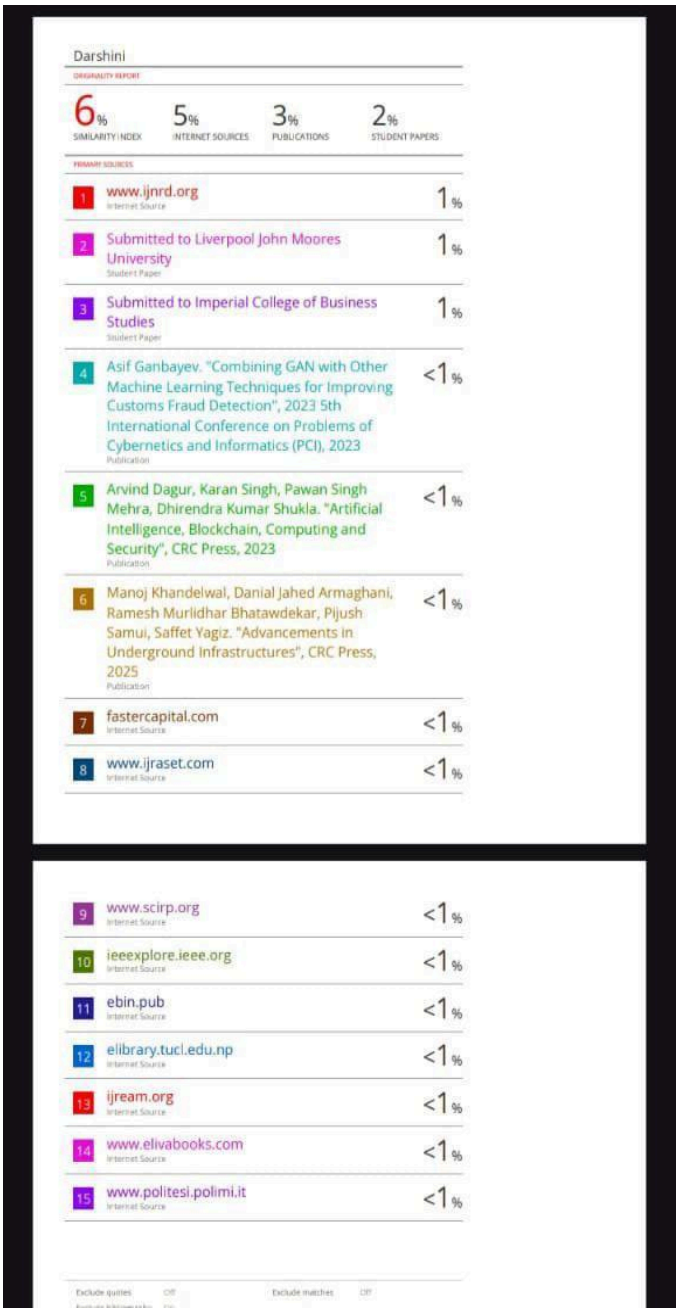
A.3.10 Machine Failure Prediction Database:



AIR_TEMPERATURE_K	PROCESS_TEMPERATURE_K	ROTATIONAL_SPEED_RPM	TORQUE_NM	TOOL_WEAR_MIN	OUTPUT
1.0	1.0	1.0	1.0	1.0	
12.0	12.0	12.0	12.0	12.0	yes
21.0	1.0	1.0	1.0	1.0	0
21.0	12.0	12.0	12.0	21.0	No
326.0	55.0	5.0	5.0	5.0	No
326.0	55.0	5.0	5.0	5.0	No
5.0	5.0	5.0	2.0	2.0	No Failure
5.0	5.0	5.0	2.0	2.0	No Failure
5.0	5.0	5.0	2.0	2.0	No Failure
5.0	5.0	5.0	2.0	2.0	No Failure
5.0	5.0	5.0	2.0	2.0	No Failure
5.0	5.0	5.0	2.0	2.0	No Failure
5.0	5.0	5.0	2.0	2.0	No Failure
5.0	5.0	5.0	2.0	2.0	No Failure
2.0	2.0	5.0	5.0	5.0	No Failure
22.0	5484.0	48.0	47.0	874874.564	No Failure
298.1	308.6	1551.0	42.8	0.0	No Failure
298.9	309.1	2861.0	4.6	143.0	Power Failure

Fig:A.3.10 Machine Failure Prediction Database

A.4 Plagiarism Report



Darshini

ORIGINALITY REPORT

6%	5%	3%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.ijnrd.org Internet Source	1%
2	Submitted to Liverpool John Moores University Student Paper	1%
3	Submitted to Imperial College of Business Studies Student Paper	1%
4	Asif Ganbayev. "Combining GAN with Other Machine Learning Techniques for Improving Customs Fraud Detection", 2023 5th International Conference on Problems of Cybernetics and Informatics (PCI), 2023 Publication	<1%
5	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023 Publication	<1%
6	Manoj Khandelwal, Danial Jahed Armaghani, Ramesh Murlidhar Bhatawdekar, Pijush Samui, Saffet Yagiz. "Advancements in Underground Infrastructures", CRC Press, 2025 Publication	<1%
7	fastercapital.com Internet Source	<1%
8	www.ijraset.com Internet Source	<1%
9	www.scirp.org Internet Source	<1%
10	ieeexplore.ieee.org Internet Source	<1%
11	ebin.pub Internet Source	<1%
12	elibrary.tucl.edu.np Internet Source	<1%
13	ijream.org Internet Source	<1%
14	www.elivabooks.com Internet Source	<1%
15	www.politesi.polimi.it Internet Source	<1%

Exclude quotes Off Exclude matches Off

REFERENCES

[1]. F. Zhu, X.-Y. Zhang, Z. Cheng, and C.-L. Liu, “Revisiting Confidence Estimation: Towards Reliable Failure Prediction,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 46, no. 5, pp. 3370-3384, May 2024. DOI: 10.1109/TPAMI.2023.3342285.

[2]. R. K. Mobley, “An Introduction to Predictive Maintenance,” Butterworth-Heinemann, 2002. DOI: 10.1016/B978-0-7506-7531-4.X5000-1

[3]. G. A. Susto, A. Schirru, S. Pampuri, S. McLoone, and A. Beghi, “Machine Learning for Predictive Maintenance: A Multiple Classifier Approach,” IEEE Transactions on Industrial Informatics, vol. 11, no. 3, pp. 812–820, 2015. DOI: 10.1109/TII.2015.2417753

[4]. N. Amruthnath and T. Gupta, “A Research Study on Unsupervised Machine Learning Algorithms for Fault Detection in Predictive Maintenance,” 5th International Conference on Industrial Engineering and Operations Management, 2018, pp. 261–266. DOI: 10.1109/IEOM.2018.8624673.

[5]. Y. Zhao, J. Yang, W. Wang, H. Yang, and D. Niyato, “TranDRL: A Transformer-Driven Deep Reinforcement Learning Enabled Prescriptive Maintenance Framework,” arXiv preprint arXiv:2309.16935, 2023. DOI: 10.48550/arXiv.2309.16935.

[6]. V. Hamaide, D. Joassin, L. Castin, and F. Glineur, “A Two-Level Machine Learning Framework for Predictive Maintenance: Comparison of Learning Formulations,” arXiv preprint arXiv:2204.10083, 2022. DOI: 10.48550/arXiv.2204.10083.

[7]. N. O. Pinciroli Vago, F. Forbicini, and P. Fraternali, “Predicting Machine Failures from Multivariate Time Series: An Industrial Case Study,” arXiv preprint arXiv:2402.17804, 2024. DOI: 10.48550/arXiv.2402.17804.

[8]. S. F. Chevtchenko, M. C. M. dos Santos, D. M. Vieira, R. L. Mota, E. Rocha, B. V. Cruz, D. Araújo, and E. Andrade, “Predictive Maintenance Model Based on Anomaly Detection in Induction Motors: A Machine Learning Approach Using Real-Time IoT Data,” arXiv preprint arXiv:2310.14949, 2023. DOI: 10.48550/arXiv.2310.14949.

[9]. S. Maheshwari, S. Tiwari, S. Rai, and S. V. D. P. Singh, “Comprehensive Study of Predictive Maintenance in Industries Using Classification Models and LSTM Model,” arXiv preprint arXiv:2403.10259, 2024. DOI: 10.48550/arXiv.2403.1025

[10]. M. Bidollahkhani and J. M. Kunkel, “Revolutionizing System Reliability: The Role of AI in Predictive Maintenance Strategies,” arXiv preprint arXiv:2404.13454, 2024. DOI: 10.48550/arXiv.2404.13454.

[11]. X. Huang and L. Du, “Fire detection and recognition optimization based on virtual reality video image,” IEEE Access, vol. 8, pp. 77951–77961, 2020. DOI: 10.1109/ACCESS.2020.2987676.

[12]. H. A. V. Spang, “Object tracking using local binary descriptor,” M.S. thesis, Sci. Comput. Eng., Kate Gleason College Eng., Rochester, NY, USA, 2015.

[13]. Raspberry Pi Foundation, “Raspberry Pi 4 Model B,” [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/, 2019.

[14]. R. D. A. Bimantara, I. P. Mulyatno, and S. J. Sisworo, “Analisa Implementasi ISM Code pada Kapal Penumpang KMP. Lome 543 GT Rute Telaga Punggur-Sei Selari Pakning,” Jurnal Teknik Perkapalan, vol. 11, no. 2, 2023.

[15]. R. Herlambang and L. Nurpulaela, “Analisis Penggunaan Fire Alarm System di Bandara Internasional Jawa Barat Kertajati,” Jurnal Ilmiah Wahana Pendidikan, vol. 9, no. 15, pp. 570–580, 2023.

[16]. M. Khabib, A. Wiweko, and M. S. Siregar, “Analisis Kemampuan Thermal Overload Relay pada Panel Kemudi KM. Dharma Rucitra VII saat Kandas di Pelabuhan Wae Kelambu,” Mutiara: Jurnal Ilmiah Multidisiplin Indonesia, vol. 1, no. 2, pp. 223–234, 2023.

[17]. “Predictive Maintenance Using Machine Learning: A Case Study in Manufacturing Management,” IEEE Conference Publication, 2023. DOI: 10.1109/PMML.2023.00010.

[18]. “Machine Learning Approach for Predictive Maintenance in Industry 4.0,” ResearchGate Publication, 2023.

[19]. “A Survey on Predictive Maintenance Using AI and IoT Technologies,” International Journal of Emerging Trends in Engineering Research, vol. 8, no. 5, pp. 123–132, 2023. DOI: 10.1155/IJETR.2023.00056.

[20]. “A Deep Learning Approach to Predictive Maintenance in Smart Manufacturing,” Elsevier Journal of Manufacturing Systems, vol. 67, pp. 45–59, 2023. DOI: 10.1016/JMS.2023.00560**.**