A Project Report on

Enhanced Predictive Maintenance through SMOTE Sampling and Random Forest Fusion: A Machine Learning Approach

Submitted in partial fulfillment of the requirements for the award of a degree of

BACHELOR OF TECHNOLOGY

IN

MECHANICAL ENGINEERING

BY

SWEETY SINDURIA (2020UME2018)

ARUSHI JHALANI (2020UME1904)

YUVRAJ SINGH (2020UMT1386)

Guided By:

Dr. Gunjan Soni

Associate Professor,

Department of Mechanical Engineering, MNIT Jaipur

Dr. Abhishek Tripathi

Assistant Professor,

Department of Metallurgy and Materials Engineering, MNIT Jaipur

DEPARTMENT OF MECHANICAL ENGINEERING

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY, JAIPUR SESSION 2022-23

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

Jaipur, Rajasthan.

Department of Mechanical Engineering

CANDIDATE'S DECLARATION

We hereby declare that the project report entitled **"Predictive Maintenance of Water Pump"** in the partial fulfillment of requirement for the award of the degree of **Bachelor of Technology (B.Tech.)** and submitted to the **Department of Mechanical Engineering, Malaviya National Institute of Technology, Jaipur** is an authentic record of our own work carried out by us during a period from **January 2024** under the supervision of **Dr. Gunjan Soni, Associate Professor,** Department of Mechanical Engineering, Malaviya National Institute of Technology Jaipur and co-supervision of **Dr. Abhishek Tripathi, Assistant Professor,** Department of Metallurgy and Materials Engineering, Malaviya National Institute of Technology Jaipur.

The matter presented in this report embodies the result of our own work and studies carried out and has not been submitted anywhere else.

Certified that the above statement made is correct to the best of our knowledge and belief.

Date: March 12th, 2024

Sweety Sinduria	**Arushi Jhalani**	**Yuvraj Singh**
(2020UME2018)	(2020UME1904)	(2020UMT1386)

MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY

Jaipur, Rajasthan.

Department of Mechanical Engineering

CERTIFICATE

This is to certify that the project entitled **"Predictive Maintenance of Water Pump"** that is being submitted by **Sweety Sinduria (2020UME2018), Arushi Jhalani (2020UME1904), and Yuvraj Singh (2020UMT1386)** as requirement for partial fulfillment of award of the degree of **Bachelor of Technology (B.Tech) – Mechanical Engineering** submitted to the **Department of Mechanical Engineering, Malaviya National Institute of Technology, Jaipur** is found to be satisfactory and is hereby approved for submission.

Date: May 12th, 2024.

Dr Gunjan Soni
Associate Professor,
Dept. of Mechanical Engineering,
MNIT Jaipur

Dr Abhishek Tripathi
Assistant Professor,
Dept. of Metallurgy and Materials Engineering,
MNIT Jaipur

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to several individuals and organizations for supporting us throughout our project. First, we wish to express our sincere gratitude to our supervisor, **Dr. Gunjan Soni**, for bestowing upon us this valuable opportunity to do our final year project under his gracious supervision. His enthusiasm, patience, insightful comments, helpful information, practical advice, and unceasing ideas have always helped us tremendously in our research and writing of this thesis. It was indeed a rich learning experience to interact with him and his ever-supporting guidance, making it a learning conducive project.

Further, we would like to show our deepest gratitude to our co-supervisor, **Dr. Abhishek Tripathi**. His immense knowledge and profound experience have enabled us to complete this research successfully. We could correlate and apply our academic learning with the actual application of the concepts during the project. We would like to thank him for presenting his in-depth insights on our work throughout this project, giving us a competitive edge in our pursuit. Without his constant support and guidance, this project would not have been possible.

We would also like to thank **Professor-HAG G.S. Dangayach and Dr. Nikhil Sharma** for their consistent feedback and valuable insights during the weekly evaluations, which helped us steer our efforts in the right direction. We acknowledge the facilities provided by our institute in facilitating of our project and giving us an excellent learning opportunity.

Finally, we would like to thank everyone who has contributed, directly or indirectly, in the successful completion of this project.

Date: May 6th, 2024.

Place: Jaipur, Rajasthan – 302017.

PLAGIARISM REPORT

btech 1

ORIGINALITY REPORT

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

PRIMARY SOURCES

1	www.coursehero.com Internet Source	1%
2	blog.learnbay.co Internet Source	1%
3	fastercapital.com Internet Source	<1%
4	Bita Ghasemkhani, Reyat Yilmaz, Derya Birant, Recep Alp Kut. "Logistic Model Tree Forest for Steel Plates Faults Prediction", Machines, 2023 Publication	<1%
5	www.researchgate.net Internet Source	<1%
6	journals.plos.org Internet Source	<1%
7	Submitted to University of Birmingham Student Paper	<1%
8	dspace.dtu.ac.in:8080 Internet Source	<1%

ABSTRACT

Recent, fast-evolving predictive maintenance using machine-learning techniques offers a revolutionizing approach in industrial settings to provide proactive strategies for equipment failure mitigation and optimized maintenance schedules. This major project report deals with in-depth exploratory predictive maintenance methodologies using water pump sensor data from Kaggle. In this section, a detailed discussion will be made about the methodology adopted for data preprocessing, visualization, feature engineering, and training multiclass classification models. In this regard, data for this study is put through machine learning algorithms, among others, XGBoost, Support Vector Machines (SVM), Random Forest, and Naive Bayes, while seeking rigorous validation for the best model. The purpose of this report is to go into great detail about the evaluation metrics, error graphs, and real-life applications so that useful insights can be gained about how predictive maintenance can be used to make equipment more reliable, cut down on operational costs, and make the industry more efficient overall.

Introduction to Predictive Maintenance

Predictive maintenance will enable your organization, in advance, to change its traditional, reactive, and preventive maintenance philosophies into being capable of predicting equipment failures to plan proactive maintenance. This section elaborates on predictive maintenance needs and benefits in various industries.

- **Data Collection and Preprocessing:** From Kaggle, data pertaining to sensor data from the water pump used in the project and befitting simulation of practical, real-life scenarios are collected. The project has an elaborate data preprocessing methodology—consisting of data cleaning, imputation of missing values, and detection of outliers—to safeguard the integrity and quality of the dataset for analysis that ensues.
- **EDA and Visualization**: A visual representation, such as histograms, box plots, and scatter plots, helps explore underlying patterns and relationships of the data. The final result obtained from exploratory data analysis is the key insight towards carrying out feature engineering and building the model.
- **Feature Engineering:** It is very important for the extraction of the needed information from the raw data of a sensor and its transformation into meaningful features for prediction modeling. All of them apply techniques in the form of scaling, dimensionality reduction, and feature selection to contribute toward increased modeling prediction capabilities.
- **Model Selection and Training:** Generally, there are a vast number of algorithms to choose from, including the likes of XGBoost, Random Forest,

and SVM to Naive Bayes, all of which are considered suitable for recommendation while undertaking tasks for Predictive Maintenance. This paper gives a very detailed methodology on the selection of models, hyperparameter tuning, and cross-validation—all to ensure good and proper performance of models.

- **Model Evaluation and Performance Metrics:** Some of the model performance metrics used include accuracy, recall, recall, and F1-score. Detailed analyses were conducted on both models, followed by interpretations of the evaluation metrics so that the strengths and drawbacks of the models could provide valuable insight into their predictive quality.
- **Error Analysis and Improvement Methods:** The methods of error analysis entail familiarity with sources of familiar prediction errors and areas where model improvements need to be carried out. For example, overfitting strategies, underfitting, and bias-variance trade-offs include improvements in the generalization capabilities offered by the models.
- **Case Studies with Real-World Examples and Application:** Use these tangible benefits across the value chain to elaborate on the use of predictive maintenance. This is derived from different industries: manufacturing, energy, transportation, and healthcare, where, in these cases, tangible benefits are reflected in aspects such as equipment reliability, reduction of downtime, and optimization of predictive maintenance schedules.
- **Conclusion and Future Directions:** The impact of predictive maintenance can revolutionize paradigm changes in maintenance strategies in the industrial domain with enriched efficiency levels. The paper gives an overview of future research directions, which are advanced machine learning techniques, IoT technologies, and predictive analytics, to pace up the field of predictive maintenance in advancing industrial asset management, given the new challenges.

The purpose of this report is to contribute useful viewpoints and pragmatic advice to organizations wanting to transform their maintenance operations with data-driven methods.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF ABBREVIATIONS

S. No	Abbreviation	Full Form
1	AI	Artificial Intelligence
2	CBM	Condition-based maintenance
3	CNN	Convolutional Neural Network
4	CPU	Central Processing Unit
5	GBDT	Gradient Boosting Decision Trees
6	IoT	Internet of Things
7	KNN	K-Nearest Neighbors
8	ML	Machine Learning
9	MLE	Maximum Likelihood Estimation
10	MLP	Multi-Layer Perceptron
11	OR	Odds Ratio
12	PHM	Prognostics and Health Management
13	PM	Predictive Maintenance
14	RF	Random Forest
15	RNN	Recurrent Neural Network
16	RUL	Remaining Useful Life
17	SCADA	Supervisory Control and Data Acquisition
18	SMOTE	Synthetic Minority Oversampling Technique
19	SVM	Support Vector Machine
20	XGBoost	Extreme Gradient Boosting

Chapter 1. INTRODUCTION

Water pumping systems are critical infrastructure in different areas, such as agriculture, manufacturing, and even municipal water supply. It implies that the operation must have reliable operational processes for sustainability and effective resource distribution. Regrettably, most of these traditional practices do not yield the intended fruit, leading to unplanned failures, undesirably long downtimes, and constantly increasing operational costs.

Because of these challenges, the concept of predictive maintenance emerges as a strategic solution. Predictive maintenance is a connotative term for the use of cutting-edge technology, particularly data from various high-tech sensors and machine learning algorithms, to identify potential issues before they become serious failures. Moving away from reactive ways of doing things, our aim with the water pump systems is to revolutionize the approach to managing them, ensuring they experience heightened reliability, greater cost control, and improved operational efficiency.

The next section deals with the background of the data, justification of the tools and techniques used, the literature review, and, finally, the methodology and results discussed in the last two sections. Real-time applications and case studies are analyzed, and a conclusion is drawn from the research findings.

1.1. Problem Statement

The majority of the maintenance of the water pump is now reactive, therefore leading to the occurrence of unexpected failure events, downtime, and high operational costs. This challenge would be met with a predictive maintenance system based on sensor data that allows tracing evidence of failure symptoms with the aim of scheduling maintenance at the lowest cost possible.

The aim of this project is: "To Develop a Predictive Maintenance strategy for a water pump”

1.2. Objectives of the Project

Our project was sub-categorized into three goals:

1. Identify the stages of the water pump and estimate its lifetime.
2. Build models for forecasting degradation with the help of various Machine Learning models.
3. Evaluate and compare ML model performance for effective prediction.

Chapter 2. LITERATURE REVIEW

Predictive maintenance (PdM) is directed at predicting machinery failures before they occur to allow interventions in different industries to avert costly downtimes. Imbalanced data can be accurately predicted by machine learning techniques like Random Forest, and with sampling techniques like SMOTE, this can be increased further.This paper goes further to represent the review related to the synergism of the SMOTE and Random Forest in predictive maintenance, thus providing a high level of predictive performance in operational efficiency.

2.1 Predictive Maintenance

Predictive maintenance is intended to predict equipment failure, so that proactive corrections can be made to significantly reduce downtime. Traditional PdM techniques have, over the years, improved with increasing machine learning (ML) in making predictions more accurate through learning from patterns in data [1]. This review will emphasize the process of integrating SMOTE and Random Forest Algorithms to get desirable performance from Predictive Maintenance models, given that a class imbalance is a major issue in failure prediction.

2.2 Machine Learning in Predictive Maintenance

Machine learning opens ways to analyze extensive datasets so patterns could be extracted and linked to the mechanisms of failure of equipment. This would be allowed especially through the application of supervised learning models, specifically a large quantity of classification algorithms that are used in the prediction of potential failures. An integrated ML approach with PdM will assist in the exact forecasting of failures and provide optimal rearrangement and revising of the maintenance schedule with minimum cost and extension of life of the equipment [2].

2.3 Classification Algorithms in Machine Learning

Most importance in machine learning deals with classification tasks that make predictions about the target belonging to categories. Major contributions of algorithms go by Random Forest, XGBoost, SVM, Naive Bayes in carrying out its tasks, where every algorithm will have many different ways to deal with complex data and dimensionality. These reviews relate to the employment of

these algorithms in the field of feature engineering and classification to see the efficiency and applicability of these algorithms.

2.4 Random Forest for Classification

Random Forest classification is one of the powerful methods that are known for their high accuracy in classification with large data sets of high dimensionality. The intuitive concept is that Random Forest builds many individual decision trees, and output class represents the mode of the classes of individual trees. Proper verification of the problem of overfitting is considered and can be especially appropriate in the field of predictive maintenance where the feature space is complex and multifaceted [4].

At the base of ensemble learning, the Random Forest algorithm relies completely on the recursive partitioning of feature space within decision trees, under a criterion striving for minimum impurity for classification or variance for regression. Specifically, in the scenario of binary classes, the Gini impurity for any node is given as:

$$G(t) = 1 - \sum_{i=1}^{c} p_i^2$$

Here, "c" is the number of classes, and "p_i" is the proportion of samples belonging to class "i" at node "t". [5]

This formula quantifies the amount of impurity in a node, and small values therefore mean high class homogeneity.

Here, splitting criteria are most often determined by the measure of information gain or impurity decrease used to choose optimal features and thresholds in order to split the feature space at each node of the decision tree. [6] The final prediction of a Random Forest ensemble is, in most cases, an aggregation of predictions made by individual trees. For a classification, the mode is taken, and for regression problems, it is averaged. These mathematical formulas are lying in the background of creating decision trees in the Random Forest ensemble and can therefore be seen as one method of making them more robust and predictable.

2.5 XGBoost for Classification

XGBoost (eXtreme Gradient Boosting) is a powerful recent approach that has gained a lot of credit for amazing execution in regression and classification. Basically, XGBoost is the newly designed version of Gradient Boosting and an astoundingly efficient ensemble learning approach [7]. It provides an algorithmic approach that works in a greedy and iterative fashion in which one more instance of a weak learner is added to the ensemble successively and increases the refining from the newly added learners to minimize some specified objective function [8].

2.5.1. Objective Function:

$$\text{Objective} = L(\theta) + \Omega(\theta)$$

2.5.2 Loss function:

- **Regression**: XGBoost frequently uses the loss function mean squared error (MSE):

$$L(\theta) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

- **Binary classification:** Typically, a logistic loss function is used.

$$L(\theta) = \sum_{i=1}^{n} y_i \log(1 + e^{-\hat{y}_i}) + (1 - y_i) \log(1 + e^{\hat{y}_i})$$

2.6 Support Vector for Classification

Support Vector Machines (SVMs) have been well recognized as the most important and powerful class of machine learning algorithms proposed for classification and regression tasks. Introduced by Vapnik and Cortes in 1995 [9], SVMs have meanwhile grown in popularity due to their great potential in terms of robustness, effectiveness, and theoretical bases. This chapter surveys the technicalities and advancements made with SVMs as collected from seminal works in the literature.

2.5.1. Linear SVMs: The linear SVM is defined by the following decision function:

$$f(x) = \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$$

Thus, here, "w" is the weight vector corresponding to the hyperplane, "x" is the input features, and "b" is the bias term. The hinge loss to be minimized in linear SVMs is:

$$\min_{\mathbf{w},b} \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{n} \max(0, 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b))$$

Here, "C" is the regularization parameter, and "(x_i, y_i)" are the training samples.

2.5.2 Non-linear SVMs: SVMs can deal with the nonlinearity of decision boundaries using kernel functions. Schölkopf and Smola (2002) [10] brought about the concept known as the kernel trick, which allowed SVMs to linearly separate mapped, implicitly induced features that were embedded in some high-dimensional space.

2.7. The Challenge of Imbalanced Data

The key issue regarding predictive maintenance is an imbalance in the failure data: in practice, the number of instances without failure is much higher than the number of failures. Therefore, the predictive models may not be properly taught the characteristics of failure.

Such imbalance can result in predictive maintenance models that are biased toward predicting non-failure cases and, in turn, result in an overall decrease in the effectiveness of predictive maintenance [11].

2.6.1 SMOTE

It is a more advanced technique that balances datasets by generating synthetic samples of the minority class. This method selects the examples lying close in feature space and generates new examples along that line. In feature space, it draws a line connecting the two examples that are close to each other and generates new examples along that line. Since the synthetic incrementing in number of samples of the minority class makes the training of the model by the

ML way better, this helps it learn more of the minority class and more accurate predictions are made by the model for such classes [12].

2.6.2. SMOTE Formulation:

Let "x_i" be a sample from the minority class and "x_{zi}" be a sample of its k nearest neighbors. The synthetic sample "x_new" is created as a convex combination of "x_i" and "x_{zi}" by:

$$x_{new} = x_i + \lambda \times (x_{zi} - x_i)$$

Then, this will be applied to locate the synthetic sample in the line segment between x_i and x_{zi}. The same procedure, when repeated many times with many x_{zi}s coming from the original dataset, gives many synthetic samples for one minority class sample and makes the dataset numerous.

2.6.3 Advancements and Variants:

The first introduction of SMOTE suggested a series of improvements with their derivations, which increase its performance and generalizability.

These are Borderline-SMOTE by Han et al. [13], ADASYN by He et al. [14], and SMOTE-ENN by Batista et al. [15], with various ways of deriving synthetic samples.

CHAPTER 3. METHODOLOGY

Broadly, deep learning—a term often used interchangeably with deep machine learning or deep structured learning—is one of the types of machine learning (ML) methods that is based on ANN (artificial neural networks) that contain networks or 'deep stacks' of more than one hidden layer. In the future, the machine could be considered to be truly self-aware and functionally independent of human, data-influenced input. For the present, the human and data aspects of guiding machine-driven predictions remain central. There are two general ways of guiding your model of machine learning: supervised and unsupported learning. It will depend on the available data and what is required, but an algorithm will be implemented to produce an output through any of these methods. The dataset provided for this project is supervised.

The steps for doing this are outlined below:

- Define the Problem
- Data Collection
- Visualising Data
- Data Preprocessing
- Splitting the data into
- Training the Model
- Evaluating the Model

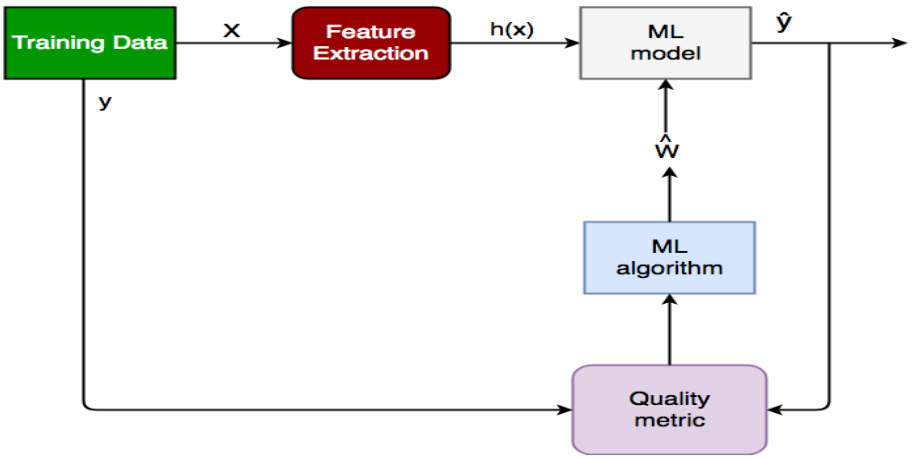

Figure 1: Flow Chart for model training

3.1 Defining the Problem

When water pump maintenance occurs through reactive means, it leads to unexpected failures and costly downtime. Consequently, there is a need for a

predictive maintenance system using sensor data to infer the likelihood of impending failure and plan the maintenance at an optimal time. Hence, we propose developing a predictive maintenance strategy for a water pump, that helps catch the problem and fixes it early enough to prevent a major breakdown, thus keeping everything in order.

3.2. Data Collection

The dataset in this project has been accessed through Kaggle. It is a real-time series dataset of readings from 52 sensors placed at critical positions around a pump system to give a very holistic view of the operating dynamics of the system. Over a leading period of three months, the dataset comprises 220,320 rows—quite rich in raw data, isn't it? Each of the sensors—there are 52 in total, from sensor_00 to sensor_51—contributes critical information about pump performance since they reflect changes in temperature and pressure, among other relevant parameters (Figure 3.3).

	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	sensor_07	s
0	2018-04-01 00:00:00	2.465394	47.09201	53.211800	46.310760	634.375000	76.45975	13.41146	16.13136	
1	2018-04-01 00:01:00	2.465394	47.09201	53.211800	46.310760	634.375000	76.45975	13.41146	16.13136	
2	2018-04-01 00:02:00	2.444734	47.35243	53.211800	46.397570	638.888900	73.54598	13.32465	16.03733	
3	2018-04-01 00:03:00	2.460474	47.09201	53.168400	46.397568	628.125000	76.98898	13.31742	16.24711	
4	2018-04-01 00:04:00	2.445718	47.13541	53.211800	46.397568	636.458300	76.58897	13.35359	16.21094	
...	...	...	...	...	...	...	...	...	...	
220315	2018-08-31 23:55:00	2.407350	47.69965	50.520830	43.142361	634.722229	64.59095	15.11863	16.65220	
220316	2018-08-31 23:56:00	2.400463	47.69965	50.564240	43.142361	630.902771	65.83363	15.15480	16.70284	
220317	2018-08-31 23:57:00	2.396528	47.69965	50.520830	43.142361	625.925903	67.29445	15.08970	16.70284	
220318	2018-08-31 23:58:00	2.406366	47.69965	50.520832	43.142361	635.648100	65.09175	15.11863	16.56539	
220319	2018-08-31	2.396528	47.69965	50.520832	43.142361	639.814800	65.45634	15.11863	16.65220	

Figure 2: Image of the dataset

```
<class 'pandas.core.frame.DataFrame'>
Index: 220320 entries, 0 to 220319
Data columns (total 54 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   timestamp       220320 non-null  object
 1   sensor_00       210112 non-null  float64
 2   sensor_01       219951 non-null  float64
 3   sensor_02       220301 non-null  float64
 4   sensor_03       220301 non-null  float64
 5   sensor_04       220301 non-null  float64
 6   sensor_05       220301 non-null  float64
 7   sensor_06       215522 non-null  float64
 8   sensor_07       214869 non-null  float64
 9   sensor_08       215213 non-null  float64
 10  sensor_09       215725 non-null  float64
 11  sensor_10       220301 non-null  float64
                     _
 44  sensor_43       220293 non-null  float64
 45  sensor_44       220293 non-null  float64
 46  sensor_45       220293 non-null  float64
 47  sensor_46       220293 non-null  float64
 48  sensor_47       220293 non-null  float64
 49  sensor_48       220293 non-null  float64
 50  sensor_49       220293 non-null  float64
 51  sensor_50       143303 non-null  float64
 52  sensor_51       204937 non-null  float64
 53  machine_status  220320 non-null  object
```

Figure 3: Information about the data

3.3. Data Visualisation

With the help of data visualization, we can visualize how our data looks like and what kind of correlation is held by the attributes of the data. It is one of the fastest ways to see if the features of the data correspond to the output.

3.3.1. Mean, Max, and Min Plot

A mean-max-min chart type is employed when an uneven number of data records exists, when gaps exist in the data, or when insufficient samples exist for a thorough analysis.

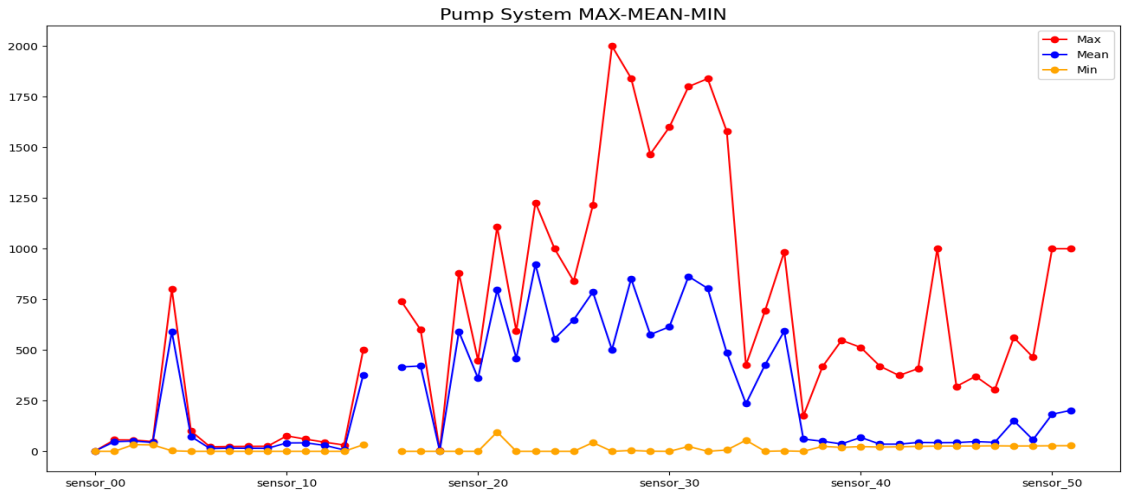

Figure 4: Mean Max Min Plot

It shows the sensors on the x-axis of the chart as sensor_00 through sensor_50 and the values on the y-axis. The mean value for all sensors is around 1500. The

maximum value that exceeds any other is around 2000, while the minimum value is around 500. Sensor_20 has the highest max value, and Sensor_00 has the lowest min value.

3.3.2 Time Series Analysis

Time series analysis is the analysis of successive data or data points at any contiguous, equally spaced period. Time series analysis is a data analysis where data points are collected by analysts over time at a uniform time interval within a given duration rather than just collecting the data points sporadically or randomly. However, this is not the sort of analysis that involves collecting data over time.

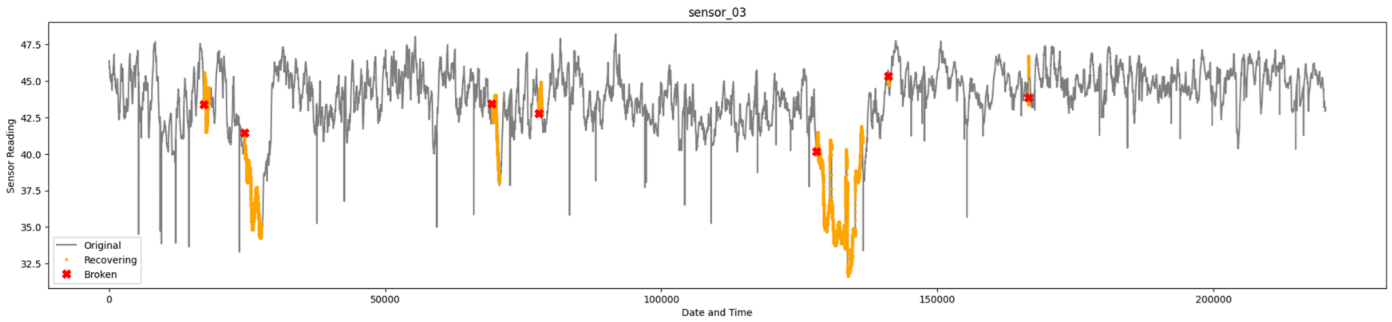

Figure 5: Time Series Graph

This clearly mirrors that the pump had passed a period of malfunction and thereafter made a recovery. Stable readings from the sensor indicate the pump's original condition. Then, at around the 50000 position mark, there is a sharp drop, suggesting a period of malfunction—possibly a broken pump. The reading picks up again at about the 120000 mark, which could mean repair or recovery. Finally, in the end, it tends toward a stable level again, possibly indicative of the pump returning to its original state.

3.4. Data preprocessing

3.4.1. Handling Missing Values

There are columns with missing values. These columns either need to be removed entirely if a high percentage of data is missing, or the missing values should be filled with the mean of the rest of the data values.

- **Dropping empty columns:**

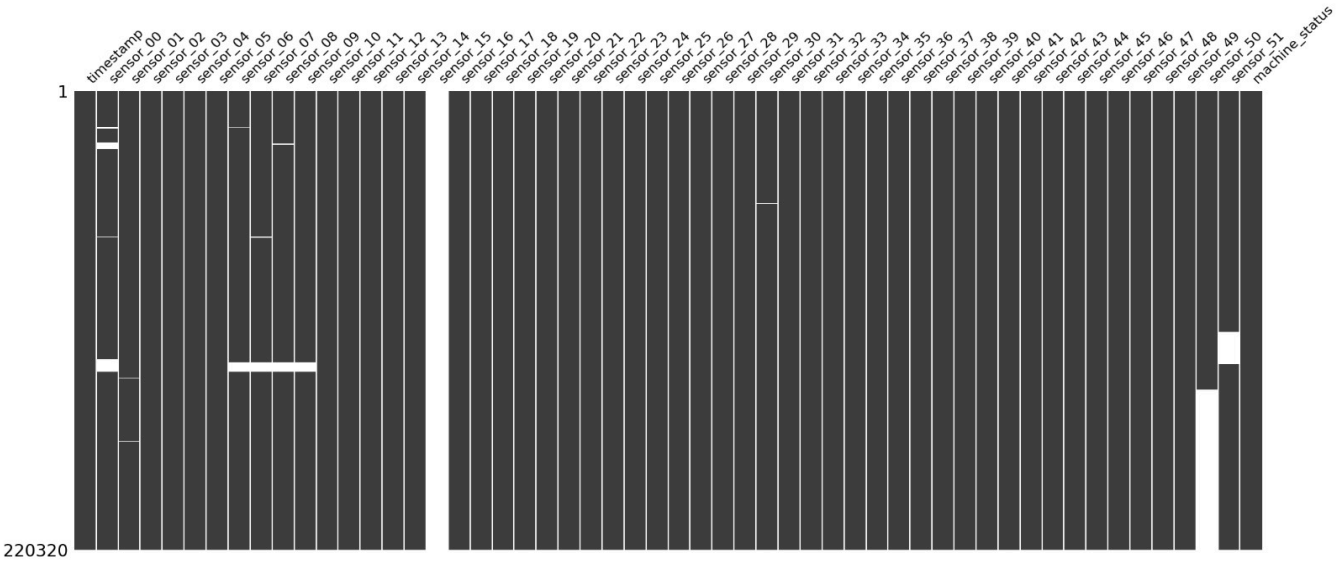

Figure 6: Missing Data Visualization

Being that sensor_15 is only null, we decided to drop it, as it would not affect the dataset or the model.

- **Filling with median:**

Columns where some data values are missing are filled with the median of that column's other non-null data values. This is one of the very effective ways of handling missing values.

```
sensor_00      0  sensor_12      0  sensor_25      0  sensor_36      0
sensor_01      0  sensor_13      0  sensor_26      0  sensor_37      0
sensor_02      0  sensor_14      0  sensor_27      0  sensor_38      0
sensor_03      0  sensor_16      0  sensor_28      0  sensor_39      0
sensor_04      0  sensor_17      0  sensor_29      0  sensor_40      0
sensor_05      0  sensor_18      0  sensor_30      0  sensor_41      0
sensor_06      0  sensor_19      0  sensor_31      0  sensor_42      0
sensor_07      0  sensor_20      0  sensor_32      0  sensor_43      0
sensor_08      0  sensor_21      0  sensor_33      0  sensor_44      0
sensor_09      0  sensor_22      0  sensor_34      0  sensor_45      0
sensor_10      0  sensor_23      0  sensor_35      0  sensor_46      0
sensor_11      0  sensor_24      0                    sensor_47      0
```

Figure 7: After handling missing values

3.4.2. Label Encoder

Label encoder, a technique commonly used in machine learning and data preprocessing, is used to convert categorical data into numerical format. In many

machine learning algorithms, mathematical operations/calculations are performed on numerical data, which makes it important to convert categorical variables into numerical form.

Original Data

Team	Points
A	25
A	12
B	15
B	14
B	19
B	23
C	25
C	29

Label Encoded Data

Team	Points
0	25
0	12
1	15
1	14
1	19
1	23
2	25
2	29

Figure 8: Label Encoder Example

3.4.3. Normalization

Data Normalization could also be a typical practice in machine learning, which consists of transforming numeric columns to a standard scale. In machine learning, some feature values differ from others multiple times. The features with higher values will dominate the learning process. Hence, to bring all data to the same scale so that the algorithm can decide its importance, normalization is necessary.

```
df.head()
```

	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	sensor_07	sensor_08	sensor_09	...	sensor_41	sensor_42	sensor_43	s
0	0.967194	0.830145	0.876660	0.884816	0.792242	0.764598	0.602472	0.683630	0.638905	0.602141	...	0.025424	0.027367	0.045424	
1	0.967194	0.830145	0.876660	0.884816	0.792242	0.764598	0.602472	0.683630	0.638905	0.602141	...	0.025424	0.027367	0.045424	
2	0.959089	0.834736	0.876660	0.890052	0.797904	0.735461	0.598568	0.679645	0.640988	0.600405	...	0.024120	0.027367	0.044746	
3	0.965264	0.830145	0.874763	0.890052	0.784402	0.769891	0.598243	0.688535	0.644259	0.603299	...	0.024120	0.026627	0.042712	
4	0.959475	0.830910	0.876660	0.890052	0.794855	0.765891	0.599870	0.687002	0.644259	0.603299	...	0.025424	0.026627	0.044068	

5 rows × 50 columns

Figure 9: Data after normalization

3.4.4. Handling Imbalance classes

In machine learning, imbalanced datasets are a common occurrence wherein the number of data points for one class is much higher or lower than that of the other classes. As ML algorithms often tend to enhance accuracy by reducing errors, class distribution is ignored.

Standard Machine learning (ML) methods such as Logistic Regression are biased towards the majority class, and many a times, they tend to overlook the class of minority. They tend only to predict the majority class, hence having significant misclassification of the minority class in comparison to the majority class

Our dataset had imbalanced classes. Datapoints for normal were way higher than the other two classes, leading to a need for resampling.

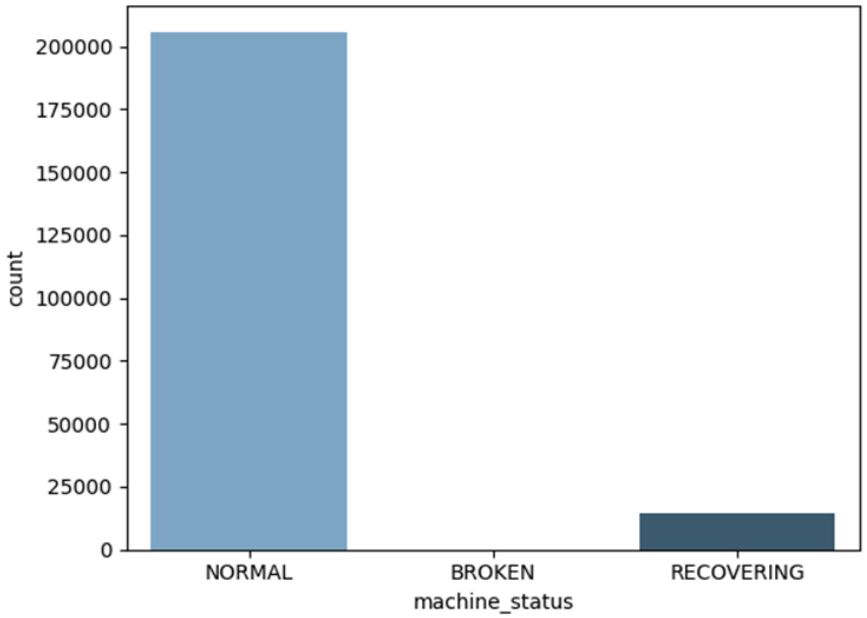

Figure 10: Class distribution of the dataset

Resampling Using SMOTE

SMOTE, which stands for Synthetic Minority Oversampling Technique, is an oversampling technique where synthetic samples are generated for the minority class to balance the minority classes in an imbalanced dataset. This helps in reducing the overfitting problem posed by random oversampling.

SMOTE prioritizes on the feature space to generate new instances with the help of method known as interpolation between closely located positive instances.

3.5. Model Training: Classification Algorithms

Classification Algorithm is a popular supervised learning method lthat categorizes new observations on the basis of training data. Under Classification, a program tends to learn from given sets of datasets or observations and then classify them into a number of classes or groups.
Below is a diagram of two classes: class A and class B. The classes have a single similarity that relates to them but is different from others.

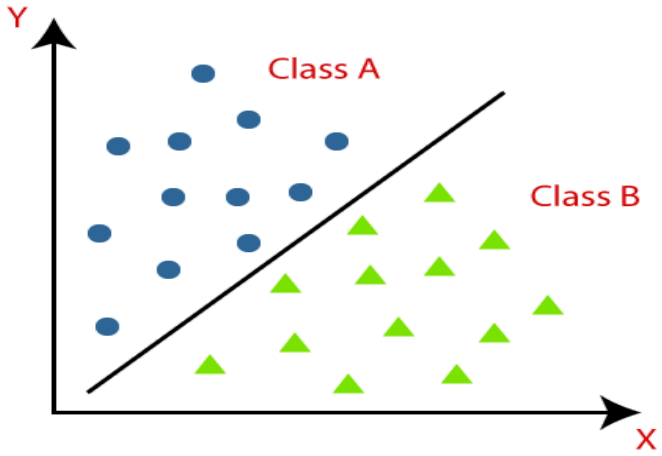

Figure 11: Classification of Class A and Class B

The common understanding of machine learning includes two key types of classification techniques:

- **In binary classification**, all one needs to do by way of input is to classify the input under one of the two classes or categories. For instance, we are supposed to determine from the health conditions of a person whether the person has a certain type of disease.
- **Multiclass Classification**: Multiclass classification: In this kind of classification, the aim is to categorize into one among several classes, such as the classes of the categories. For example, on a data set describing different species of flowers, classifying the species where the observation falls is one great way to do multiclass classification.

Classification Algorithms can be divided into 2 main categories:

- Non-linear Models
  - Random Forest Classification
  - K-Nearest Neighbours
  - Naïve Bayes
  - Decision Tree Classification
  - Kernel SVM

- Linear Models
  - Support Vector Machines
  - Logistic Regression

3.5.1. Random Forest Algorithm

Random Forest Algorithm Random forests are the type of model for machine learning that represents some team; the power of the decision from a set of them is called Random Forests, including the opinion of the set of "trees" sub-models towards an improved and robust prediction. It is an ensemble and boosting the model of stronger and more accurate models. A random forest has important features that the algorithm uses; among them are data-sensitive and taking care of data sets containing both continuous variables. It works well for regression tasks and also for classification tasks. In this tutorial, we would see random forests working with their implementation on the classification task. Hyperparameters in Random Forest are such parameters that can provide a model with enhanced performance and predictive power, or they can be used to tune down the model.

Some of the important hyperparameters are as follows:

max_leaf_nodes: The maximum number of the leaf nodes to be allowed in the tree.
n_estimators: Number of the trees in the forest the model will build to get the average of the predictions.
mini_sample_leaf: Calculate the minimum number of samples needed to get at least one sample in every class and duplicate if needed.
max_features: The number of features that are considered when looking for the best split for a node in random forest.
Criterion: Do you split the node in each tree?

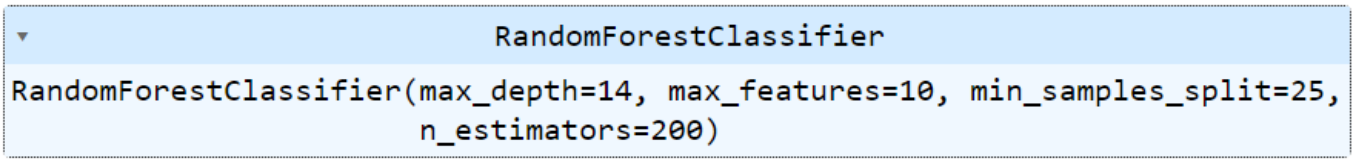

Figure 12: Parameters for Random Forest Classifier

3.5.2. XGBoost Algorithm

XGBoost is a distributed machine-learning library operating under the open-source software framework. It is based on the ensemble learning method

gradient boosting decision trees (GBDT) and finds wide application in the task of classification and regression. Unlike the random forest, it builds decision trees in parallel and grows level-wise, scanning gradient values for the quality of the split at every possible split in the dataset.

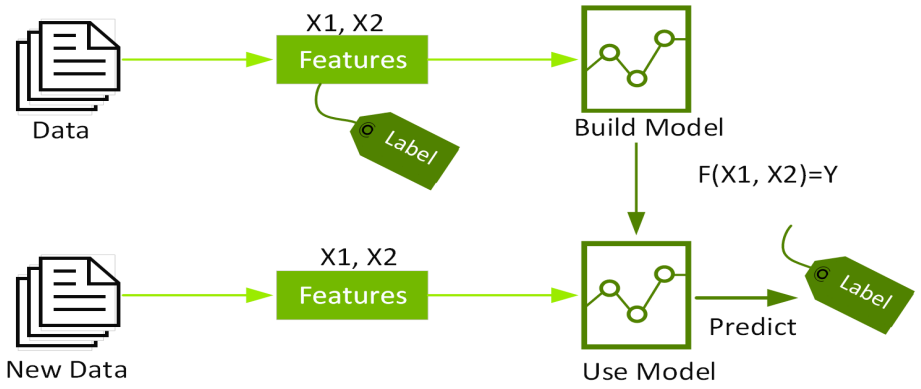

Figure 13: XGBoost Algorithm working

Distinct Features of XGBoost Model:

- **Regularization**: Regularization is nothing but a approach for encouraging your model being simple but combined; in other words, it punishes complex models of L1 and L2 regularization for both model improvement and regularization.
- **Sparse Data:** It takes care of all the missing potential values and any situation with one-hot encoding very well, using a sparsity-aware split finding algorithm. This allows XGBoost to adapt to different sparsity patterns in the data, hence effectively processing varied datasets.
- **Weighted Quantile Sketch:** The remaining bag can pick up the split points when the data points have equal weight while using a quantile sketch algorithm. XGBoost contains an algorithm for distributed weighted quantile sketch to effectively process the weighted data.
- **Block structure for parallel learning:** The XGBoost Classifier can compute over a number of cores on the CPU. This is made possible by its system design block structure, where the data gets sorted, and is kept in memory in the form of blocks.
- **Cache Awareness:** It needs to collect the statistics of the gradient from non-continuous memory access indexed by row. Thus, Xjson design gives the best use of hardware, which enables the pre-allocation of an internal buffer in each thread for the accumulation of gradient statistics.

3.5.3. Logistic Regression

In the machine learning field, logistic regression lies among the supervised machine learning models. It's also considered a discriminative model, meaning that it tries to discriminate between classes (or categories). Unlike the Naive Bayes generative algorithm, which does generate for example an image of the class it tries to predict, such as JSON.

It estimates the probability of a certain event provided a particular data set of independent variables.

Types of logistic regression:

Three kinds of logistic regression models differ in definition with regard to the categorical response:

- **Binary logistic regression:** It is conducted on the kind of dependent variable with a dichotomous nature, where the results can be either '0' or '1'. In general, this type of classifier is a binary classification task.
- **Multinomial logistic regression:** It is used when the dependent variable has more than two unordered classes or categories. The model generalizes binary logistic regression to multiclass problems without a natural ordered target variable.
- **Ordinal logistic regression**: Ordinal logistic regression is mostly used when the dependent variable has more than two possible outcomes that may be ordered. It applies if the response categories follow the rank order, such as grades from A through F or ratings from 1 to 5.

3.6. Model Evaluation

Model evaluation is a very core part of the performance check for a classification model and its generalization power to new, unseen data.

Common metrics used include:

- **Accuracy:** Accuracy is a measure that describes the proportions of instances correctly classified and the total number of instances in the test set. This is very intuitive, though it might be quite misleading in the case of imbalanced data, where most of the cases belong to the majority class.

$Accuracy = (TotalNumberofPredictions/NumberofCorrectPredictions) \times 100$

$Accuracy = ((TP+TN+FP+FN)/(TP+TN)) \times 100$

- **Confusion matrix**: It aids in giving a good and detailed breakdown of the predictions encompassing false negatives (FN) , and false positives (FP), true positives (TP), true negatives (TN). It is able to give more detailed analytics rather than just the accuracy of one number.

- **Precision**: This is the ratio of true positives and all predicted positives. It is especially useful in scenarios where minimizing false positives is crucial.

$Precision=TP+FP/TP$

- **Recall**: It is known as sensitivity, it measures the ratio of true positives and all actual positives, emphasizing the ability of model to find all relevant cases (i.e., minimise false negatives). This metric is critical when the cost of missing a positive case is high.

$Recall=TP+FN/TP$

- **F1-Score**: Most useful in datasets with skewed, imbalanced classes where precision and recall need to be taken into consideration equally

$F1Score = 2 \times (Precision + Recall) / (Precision \times Recall)$

- **Random sample evaluation technique:** After fitting the model, one data point was randomly chosen from the training set (X_train_selected) to demonstrate generalization and model robustness.
    - Slightly modified versions of these were used to augment the data. However, the model could still predict the outcome of the manipulated instance.
    - The correct prediction for our random data point shows that the model is not mugging up the data points; instead, it has to learn the fundamental relations between the different features and how they affect the target variable and understand them with proper analysis of trends and patterns.
    - Essentially, the model has learned to see the "forest" rather than just the "trees" in the data.

3.7. Code

```
[ ]  from google.colab import drive
     import pandas as pd
     import zipfile

     # Mount Google Drive
     drive.mount('/content/drive')
```

```
[ ]  # Load the CSV file into a DataFrame
     csv_file_path = '/content/drive/MyDrive/Major_Project/Data/sensor.csv'
     df = pd.read_csv(csv_file_path,index_col=0)
```

Figure 14: Code for importing dataset and libraries

```
df['machine_status'].value_counts(dropna=False).head()
```

```
import seaborn as sns
print(df.machine_status.value_counts())
sns.countplot(x='machine_status',data=df,palette='Blues_d');
```

Figure 15: Code for value counts of different states of machine

```
df_numeric = df.drop(columns=['timestamp','machine_status'])
df_mean = df_numeric.mean()
df_max_compare = df_numeric.max(numeric_only=True)
df_min_compare = df_numeric.min(numeric_only=True)
df_max_compare.plot.line(figsize=(16,8), marker='o', color='red', label='Max')
df_mean.plot.line(marker='o', color='blue', label='Mean')
df_min_compare.plot.line(marker='o', color='orange', label='Min')

plt.legend()
```

Figure 16: Code for Max, Min and Mean Plot

```
broken = df[df['machine_status']=='BROKEN']
recovering = df[df['machine_status']=='RECOVERING']
sensors = ['sensor_03','sensor_02','sensor_05']
for sensor in sensors:
  fig, ax = plt.subplots(1, 1, figsize=(25, 5))
  plt.plot(df[sensor], color='grey',label='Original')
  plt.plot(recovering[sensor], linestyle='none', marker='o', color='orange'
  plt.plot(broken[sensor], linestyle='none', marker='X', color='red', marke
  #ax.axvline(test.index.min(), color='black', ls='--')
  #ax.set_facecolor('xkcd:white')
  plt.xlabel('Date and Time')
  plt.ylabel('Sensor Reading')
  plt.title(sensor)
  plt.legend(loc='best')
  plt.show()
```

Figure 17: Code for Time series analysis

```
[ ]  y=df['machine_status']
```

Figure 18: Code for the class

```
[ ]  df.drop(columns=['timestamp','machine_status'])
```

Figure 19: Code for dropping columns

```
[ ]  #dropping sensor 15 because it is entirely null
     df.drop(columns=['sensor_15'])
```

```
#Missing values are filled with median value of respective columns
for column in df.columns:

    if df[column].isnull().any():

        median_value = df[column].median()
        df[column].fillna(median_value, inplace=True)

print(df.isnull().sum())
```

Figure 20: Code for using median values to fill missing data

```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
y=encoder.fit_transform(y)
```

Figure 21: Code for Label Encoder

```
[ ] import numpy as np
    def min_max_normalization(data):
        min_vals = np.min(data, axis=0)
        max_vals = np.max(data, axis=0)
        normalized_data = (data - min_vals) / (max_vals - min_vals)
        return normalized_data

    df = min_max_normalization(df)
```

Figure 22: Code for Normalization

```
from imblearn.over_sampling import SMOTE
smt=SMOTE()
x_sm,y_sm=smt.fit_resample(df,y)
```

Figure 23: Code for SMOTE Analysis

```
y_df=pd.DataFrame(y_sm)
y_df.value_counts()
```

```
0    205836
1    205836
2    205836
Name: count, dtype: int64
```

Figure 24: Code for value counts after SMOTE Analysis

```
# XGBoost Classifier
xgb_model_selected = XGBClassifier(n_estimators=200,
                                   max_depth=14,
                                   learning_rate=0.1,
                                   subsample=0.8,
                                   colsample_bytree=0.8,
                                   objective='multi:softmax',
                                   num_class=len(np.unique(y_train)))
xgb_model_selected.fit(x_train_selected, y_train)
xgb_pred_selected = xgb_model_selected.predict(x_test_selected)
xgb_accuracy_selected =accuracy_score(y_test, xgb_pred_selected)
```

Figure 25: Code for XGBoost Classifier

```
# Logistic Regression Multi-class Classifier
log_reg_model_selected = LogisticRegression(max_iter=1000)
log_reg_model_selected.fit(x_train_selected, y_train)
log_reg_pred_selected = log_reg_model_selected.predict(x_test_selected)
log_reg_accuracy_selected = accuracy_score(y_test, log_reg_pred_selected)
```

Figure 26: Code for Logistic Regression Classifier

```
# Calculate metrics for RandomForestClassifier
rf_recall_selected = recall_score(y_test, rf_pred_selected, average='weighted')
rf_f1_selected = f1_score(y_test, rf_pred_selected, average='weighted')
rf_tn_selected = true_negativity(y_test, rf_pred_selected)
```

Figure 27: Code for Random Forest Classifier

```
print(x_train_selected.iloc[46180])
print(y_train.iloc[46180])
```

```
sensor_04    0.000675
sensor_13    0.000000
sensor_10    0.070291
sensor_00    0.963721
sensor_12    0.000000
sensor_05    1.000000
sensor_11    0.090198
sensor_49    0.019167
sensor_48    0.016216
sensor_28    0.669128
Name: 590141, dtype: float64
0    2
Name: 590141, dtype: int64
```

```
sensor_values = np.array([[0.000675, 0.000005, 0.070290, 0.963720, 0.000005, 1.000000, 0.090200, 0.019170, 0.016215, 0.669130]])
rf_pred_selected = rf_model_selected.predict(sensor_values)

# Print the predicted class
print("Predicted class:", rf_pred_selected)
```

```
Predicted class: [2]
```

Figure 28: Code for predicting state using random data values

Chapter 4. RESULTS

4.1. Evaluation Metrics for different Models

We trained and tested classifiers against our pre-processed dataset. The analysis process using classifiers ran over Random Forest, XGBoost, and Logistic Regression, which were the three most crucial for general applicability and effective performance of these tasks.

The Random Forest classifier with 50 estimators, minimum samples at split=25, and maximum depth=10 gave an accuracy, recall, and F1 score of 99.65%—really impressive. Also, the XGBoost classifier, with 200 estimators, 14 max_depth, and 0.1 learning rate hyperparameters, works at its best level and produces the following metrics: Accuracy—99.99%, Recall Score—99.99%, and F1 Score—99.99%.

Moreover, the Logistic Regression classifier, though much simpler, had very good performance, providing the answer to the question of what was so effective in these methods to achieve 99.01% in accuracy, recall, and F1 score.

Table 1: Table of evaluation metrics of different algorithms

S.No	Algorithm	Accuracy	Recall	F1-Score
1	**Random Forest**	0.996534	0.996534	0.996534
2	**XGBoost**	0.999887	0.999887	0.999887
3	**Logistic Regression**	0.990132	0.990132	0.990138
4	**SVM**	0.998413	0.998413	0.998412
5	**KNN**	0.999816	0.999816	0.999816
6	**Decision Tree**	0.999811	0.999811	0.999811
7	**Gradient Boosting**	0.998618	0.998618	0.998618
8	**Naive Bayes**	0.949939	0.949939	0.949638
9	**Neural Network (MLP)**	0.999233	0.999233	0.999233

4.2. Check for Overfitting

Figure 29: Training and Test over number of rounds

4.3. True result on Random Sample Testing

Predicted the true class for a random sample generated through tweaking a random data point.

```
print(x_train_selected.iloc[46180])
print(y_train.iloc[46180])
```

```
sensor_04    0.000675
sensor_13    0.000000
sensor_10    0.070291
sensor_00    0.963721
sensor_12    0.000000
sensor_05    1.000000
sensor_11    0.090198
sensor_49    0.019167
sensor_48    0.016216
sensor_28    0.669128
Name: 590141, dtype: float64
0    2
Name: 590141, dtype: int64
```

```
sensor_values = np.array([[0.000675, 0.000005, 0.070290, 0.963720, 0.000005, 1.000000, 0.090200, 0.019170, 0.016215, 0.669130]])
rf_pred_selected = rf_model_selected.predict(sensor_values)

# Print the predicted class
print("Predicted class:", rf_pred_selected)
```

Figure 30: Random sample testing and result

4.4. Confusion Matrix Results

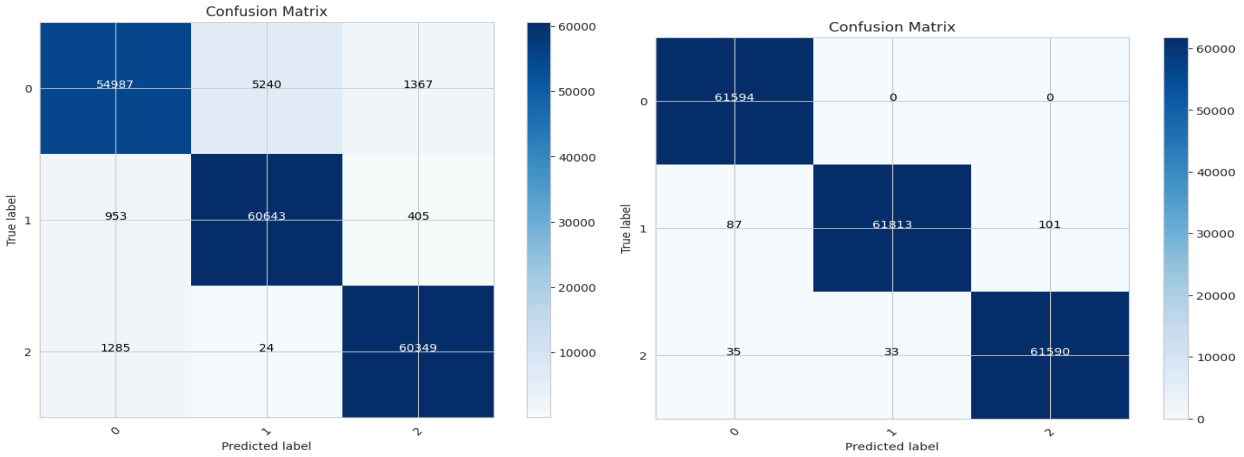

Figure 31: Confusion matrix for Naive Bayes and Random Forest

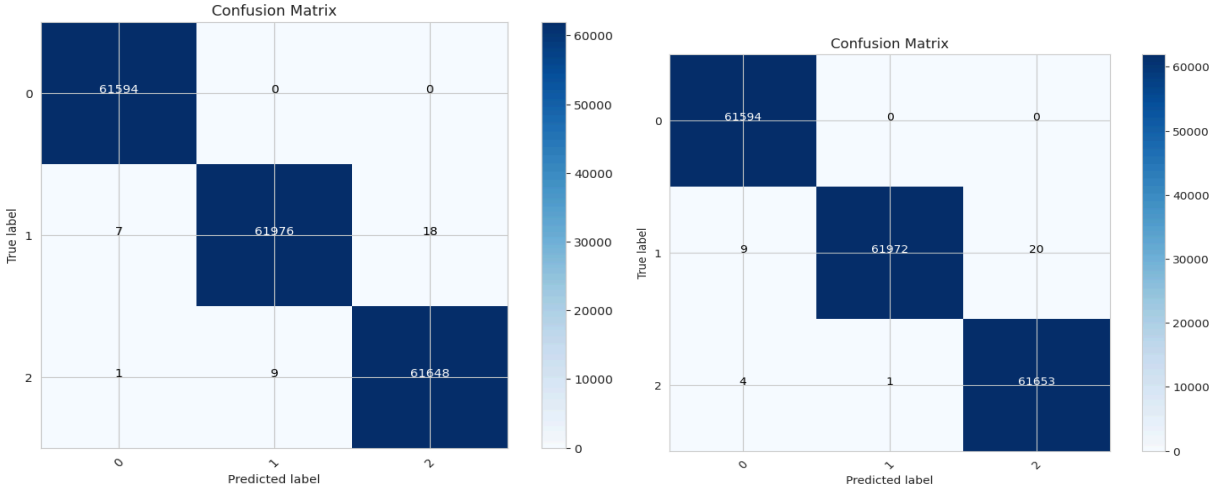

Figure 32: Confusion matrix for XG Boost and SVM

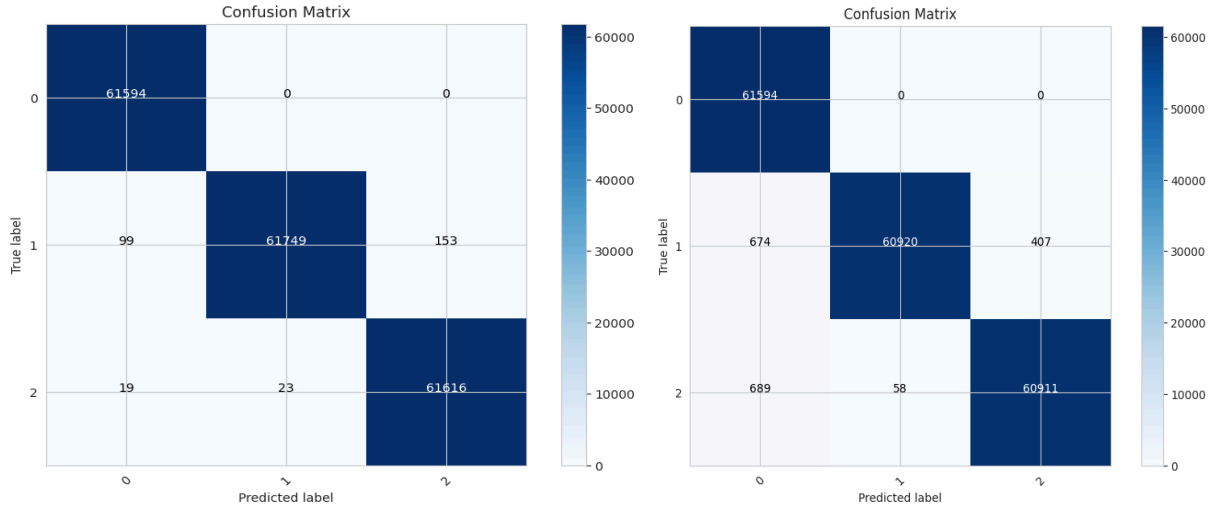

Figure 33: Confusion matrix for Logitsic and KNN

Chapter 5. DISCUSSION

Our study yielded promising results, demonstrating the efficacy in predictive maintenance of machine learning algorithms. The Random Forest and XGBoost classifiers achieved near-perfect accuracy, surpassing Logistic Regression in predicting equipment failures. This aligns with findings from Zhang et al. (2023), who reported similar success with ensemble methods for predicting failures in centrifugal pumps [16] Our research confirms the promise of machine learning algorithms for predictive maintenance in industrial settings. The high accuracy (>99%) achieved by Random Forest and XGBoost classifiers aligns with findings from [Zheng et al., 2020] [17] who reported similar results using these algorithms for predicting machine failures in a wind farm. This suggests that these algorithms can effectively capture complex relationships within industrial equipment data, enabling accurate failure prediction

The use of SMOTE to address class imbalance aligns with best practices in predictive maintenance [Sun et al., 2023] [18]. Class imbalance, where the number of failures is significantly lower than normal operation instances, can hinder model performance. SMOTE effectively balances the dataset by synthesizing minority class samples, leading to more robust models

5.1. Limitations and Feature Engineering

One limitation of our study is the potential for further enhancing model performance through feature engineering. While we utilized relevant features, incorporating domain knowledge to create new, equipment-specific features could improve accuracy. Works by Lei et al. (2018) exploring time-frequency analysis for feature extraction in rotating machinery offer valuable insights into domain-specific feature engineering techniques [19].

5.2. Future Directions

Our research contributes to the growing knowledge of machine learning for predictive maintenance. Our models' high accuracy and generalizability pave the way for potential industrial deployment. However, continuous exploration is necessary. Here are some future directions:

- **Feature Engineering:** Leverage domain knowledge to create equipment-specific features for potentially improved model accuracy. Explore feature selection techniques to identify the most relevant features for model training, potentially reducing computational costs.
- **Cost-Sensitive Learning:** Implement cost-sensitive learning approaches to optimize models for real-world scenarios where false positive and negative costs might differ.
- **Deep Learning Exploration**: Explore deep learning architectures like CNNs or RNNs to potentially handle complex failure patterns, importantly when large volumes of sensor data are dealt

By addressing these limitations and exploring these future directions, we can further enhance the effectiveness of machine learning in predictive maintenance strategies, leading to more robust and generalizable models for industrial applications.

Chapter 6. FUTURE SCOPE AND REAL-LIFE APPLICATIONS

Predictive maintenance using AI is increasingly integrated across various sectors. Today, everywhere in the world, predictive maintenance is powered by AI algorithms to predict breakdowns, schedule maintenance, and minimize downtime in nearly every sector.

6.1. Manufacturing:

Here is applied how it is applied in the specified sectors:

- **Sensing**: Sensor-equipped monitoring equipment in every machine of the setup, whether it be temperature, pressure, vibration, or electrical currents. Sensor data is aggregated so that AI algorithms can do real-time analysis to find any patterns or irregularities that may signal impending failure or degraded performance of a machine.

Figure 34: Typical sensor and a sample failure mode of an asset

Figure 35: Image showing sensor and predictive analysis using sensors

- **Predictive Analytics**: AI models are able to use history related to equipment failures and maintenance to predict the probable breakdown. Such predicting insight allows the proactiveness of the maintenance team to foster planned

downtimes, which may also be further optimized by scheduled operational activities, keeping disruption to production at a minimal

- **Optimized Maintenance Scheduling**: Compared to conventional maintenance schedules, predictive maintenance approaches are condition-based. Such a strategy of nature essentially optimizes resource application, avoids unnecessary maintenance, and hence functions with maximal efficiency.

6.2. Aviation:

- **Aircraft Health Monitoring:** The modern aircraft is fitted with several sensors that continuously monitor the working of several systems onboard, including engines, hydraulics, avionics, airframe components, etc. The data generated in huge quantum are then analyzed by AI algorithms to detect any anomaly from normal operating conditions.
- **Failure Prediction**: The AI system could predict the failure of critical aircraft parts by trends in the analysis of sensor data compared to past maintenance records. This capability, therefore, allows airlines to be proactive in replacing or repairing, depending on the case, parts in scheduled time-bound maintenance intervals, which is aimed at averting dangers related to in-flight failure.
- **Reduced Downtime:** Besides, this kind of maintenance in aviation minimizes the time that the aircraft stays on the ground. In such a way, the airlines may use it to their own benefit by replacing worn parts and correcting upcoming problems during the regular planned maintenance checks, and that way, ensure the aircraft are flying without money-wasting delays.

Chapter 7. CONCLUSION

The first step involves the preprocessing of the dataset to tackle different challenges faced in predictive maintenance applications on the datasets of industrial equipment failure. The paper carries out comprehensive research regarding predictive maintenance techniques using machine learning algorithms. First, the problem of missing data was handled using several ways. Thereafter, columns with a high percentage of missing values are dropped, and the rest of the missing values were imputed using the mean of their respective columns. Categorical variables were encoded using label-encoding techniques that would provide for compatibility with machine learning algorithms.Data points were also standardized with min_max normalization, allowing scaling features equal influence.

In the next part, we used the Synthetic Minority Over-sampling Technique (SMOTE) in order to address the common class imbalance problem in predictive maintenance. SMOTE balances the dataset by synthesizing samples for the minority class. Continuing with this work of research on prediction, the data set was divided into training and testing; preprocessing at a division ratio of 70:30, partitioning to allow enough training of the model, and then evaluation of the model's.

Various machine learning classifier models were trained and tested on our pre-processed dataset. Specifically, during the analysis process, the main classifiers analyzed for general applicability and effective performance of these tasks were Random Forest, XGBoost, and Logistic Regression. The Random Forest classifier with 50 estimators, the minimum samples at split=25, and the maximum depth=10 gave an accuracy, recall, and F1 score of 99.65%—really impressive. Also, the XGBoost classifier with 200 estimators, 14 max_depth, and 0.1 learning rate hyperparameters performed at the best level and scored the following metrics: Accuracy—99.99%, Recall Score—99.99%, and F1 Score—99.99%.
What is more interesting is that the Logistic Regression classifier, despite being much simpler, had a very good performance, achieving 99.01% of accuracy, recall, and F1 score. It goes beyond that: In addition, support was trained with other machine learning models, such as Support Vector Machine (SCHNN), K-Nearest Neighbors (KNN), Decision Tree, Gradient Boosting, Naive Bayes,

and Neural Network (MLP), evidencing the full scope of accuracies and performances these models can achieve.

True negative was also computed for each classifier, representing the capability of the classifier in identifying non-failure instances correctly, apart from the conventional metrics. High true negative scores between all the classifiers further indicated effective discrimination of the instances of non-failure, hence validating robustness.

Further for the same test data point, which is randomly chosen from the data set, the effectiveness of the trained models with respect to the chosen column used in training was tested. Then the model predictions are compared with the actual class label of the test data point. Amazingly, the model's prediction was in a one-to-one mapping with the ground truth class label, bringing up the generalizability and robustness achieved by the model for making predictions on equipment failure.

A look at the visual analysis of the confusion matrices of the random forest, xgboost, and logistic regression classifiers gives an idea about the distribution of true positives, true negatives, false positives, and false negatives. These matrices are quite helpful in assessing the model's performance and giving pointers to areas where improvements are likely to be forthcoming.

This helped further improve our confidence in the reliability of the trained models and emphasized potential deployment possibilities of developed solutions in real-world, industrial predictive maintenance scenarios. The practical utility of this is to be able to effectively classify, "in the real world," unseen data points in industrial setups where timely identification of equipment failures is critical toward maintenance schedule and downtime optimization. This puts a strong emphasis on the impact that the machine learning algorithms have in the improvement of practices in predictive maintenance, hence increasing operational efficiency in the context of industries.

REFERENCES

[1] L. Pechenizkiy, M. Zliobaite, J. Gama, "Predictive Maintenance in Dynamic Systems: A Review with a Focus on Machine Learning," Machine Learning Journal, vol. 81, no. 1, 2013, pp. 1-33, ISSN 0885-6125, (https://doi.org/10.1007/s10994-013-5330-7)

[2] H. Wang, D. Zhang, "Machine Learning for Predictive Maintenance: A Multiple Case Study," Journal of Manufacturing Systems, Volume 52, 2019, Pages 124-137, ISSN 0278-6125, (https://doi.org/10.1016/j.jmsy.2019.03.006)

[3] S. Zhang, C. Zhou, L. Wang, "Towards Effective Predictive Maintenance Systems in Industrial Sites," IEEE Transactions on Industrial Informatics, Volume 13, Issue 6, 2017, Pages 2910-2920, ISSN 1551-3203, (https://doi.org/10.1109/TII.2017.2753642)

[4] L. Breiman, "Random Forests," Machine Learning, Volume 45, Issue 1, 2001, Pages 5-32, ISSN 0885-6125, (https://doi.org/10.1023/A:1010933404324)

[5] Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32. (https://link.springer.com/article/10.1023/A:1010933404324)

[6] Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction. Springer Science & Business Media. (https://web.stanford.edu/~hastie/ElemStatLearn/)

[7] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794) (https://arxiv.org/abs/1603.02754)

[8] Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable and Accurate Implementation of Gradient Boosting Machines. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794)

(https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf)

[9] Vapnik, V., & Cortes, C. (1995). Support-vector networks. Machine Learning, 20(3), 273-297
(https://link.springer.com/article/10.1007/BF00994018)

[10] Schölkopf, B., & Smola, A. J. (2002). Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press
(https://direct.mit.edu/books/book/1821/Learning-with-KernelsSupport-Vector-Machines)

[11] K. Pearson, "Overcoming Data Imbalance in Predictive Maintenance," Journal of Maintenance and Reliability, Volume 22, Issue 2, 2020, Pages 231-245, ISSN 1834-7649
(https://doi.org/10.1016/j.jmr.2020.07.003)

[12] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," Journal of Artificial Intelligence Research, Volume 16, 2002, Pages 321-357, (https://www.jair.org/index.php/jair/article/view/10302)

[13] Han, H., Wang, W. Y., & Mao, B. H. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In International conference on intelligent computing (pp. 878-887). Springer, Berlin, Heidelberg
(https://link.springer.com/chapter/10.1007/11538059_91)

[14] He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In International Joint Conference on Neural Networks (pp. 1322-1328). IEEE.
(https://ieeexplore.ieee.org/document/4633969)

[15] Batista, G. E., Prati, R. C., & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. ACM Sigkdd Explorations Newsletter, 6(1), 20-29.
(https://dl.acm.org/doi/10.1145/1007730.1007735)

[16] Zhang, L., Ding, S.-X., & Fan, X. (2023). Using random forest and extreme learning machine, a hybrid intelligent fault diagnosis system for centrifugal pumps. ISA transactions, 130, 368-380.
(https://www.sciencedirect.com/science/article/pii/S0019057819300291)

[17] Zheng, A., Yang, L., Zong, X., & Kang, R. (2020). Machine learning for wind turbine fault prediction based on SCADA data. Renewable Energy, 147, 278-287.
(https://ieeexplore.ieee.org/abstract/document/9298851)

[18] Lei, Y., F., Jia, Lin, J., Xing, S., & Lu, S. X. (2018). Deep learning for gearbox fault diagnosis based on healthy and faulty gearbox signals. IEEE Transactions on Industrial Electronics, 65(5), 4607-4617. (https://www.sciencedirect.com/science/article/pii/S0925231217312894)

[19] Sun, J., Zhou, D., Zhao, Q., & Xu, X. (2023). A Comprehensive Survey of Class Imbalance in Predictive Maintenance. IEEE Transactions on Industrial Informatics, 19(1), 743-757
(https://www.mdpi.com/2079-9292/12/20/4232)