Fault Detection and Predictive Maintenance in IoT-based Building Management System using Machine Learning

by

Khaja Sheikh Imran Sabuj
16201071
Saifullah Chowdhury
17201057
Fahim Hasan Mehedi
17201108
Pritam Chakraborty
21141083
Shafarse Simeon Arnob
16101235

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
Brac University
January 2022

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Khaja Sheikh Imran Sabuj
16201071

Saifullah Chowdhury
17201057

Fahim Hasan Mehedi
17201108

Pritam Chakraborty
21141083

Shafarse Simeon Arnob
16101235

Approval

The thesis titled "Fault Detection and Predictive Maintenance in IoT-based building management system using machine learning" submitted by

1. Khaja Sheikh Imran Sabuj (16201071)
2. Saifullah Chowdhury (17201057)
3. Fahim Hasan Mehedi (17201108)
4. Pritam Chakraborty (21141083)
5. Shafarse Simeon Arnob (16101235)

Of Fall, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 18, 2022.

Examining Committee:

Supervisor:
(Member)

Amitabha Chakrabarty, PhD
Associate Professor
School of Data and Science
Department of Computer Science and Engineering
Brac University

Thesis Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson and Associate Professor
School of Data and Science
Department of Computer Science and Engineering
Brac University

Abstract

Infrastructures in the modern era are incorporating the Internet of Things (IoT) in everything from complex building automation systems (BAS) to individual small devices, emphasizing the importance of Predictive Maintenance. As a result, early fault detection is required, especially in sensitive and massive structures such as hospitals, industries, and multipurpose buildings. In such infrastructures, even minor failures can result in tragedies such as fires or slow down productivity. In our research, we have used several machine learning fault detection and diagnostics (FDD) algorithms in building fault detection data. We collected two datasets, MZVAV-1 (SET-A) and MZVAV-2-1 (SET-B), which were split into train-test sets to deploy LogisticRegression, KNearestNeighbours, Naive Bayes classifier, Support Vector Classifier, RandomForestClassifier, Decision Tree, MLP Classifier and Extra Tree Classifier. We achieved the highest accuracy of 98.91% using the Decision Tree classifier and the lowest accuracy of 14.17% from Naive Bayes classifier on the MZVAV-1 dataset. RandomForestClassifier and ExtraTree classifier outperformed all other algorithms with 99.91% accuracy on the MZVAV-2-1 dataset.

Keywords: Internet of Things, Fault Detection, Building Systems, Machine Learning, FDD Algorithms, Predictive Maintenance.

Acknowledgement

First and foremost, we give thanks to the Almighty God for allowing us to finish our thesis without any serious interruption. Next, we would like to express our gratitude to our supervisor, Dr. Amitabha Chakrabarty sir, for his constant support and advice during the journey of our research and we sincerely value his contribution. Lastly, without our parents' unconditional support, we may not be able to achieve our objectives. We are currently on the verge of graduating thanks to their generous support and prayers.

Table of Contents

List of Figures

List of Tables

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

AHU Air Handling Unit

$APAR$ AHU Performance Assessment Rules

BAS Building Automation System

DTC Decision Tree Classifier

FDD Fault Detection and Diagnostics

FN False Negative

FP False Positive

GAN Generative Adversarial Network

GMM Gaussian Mixture Model

GNB Gaussian Naive Bayes

$HVAC$ Heating, Ventilation, and Air Conditioning

IoT Internet of Things

KNN K-Nearest Neighbor

LR Logistic Regression

MLP Multilayer Perception

PCA Principal Component Analysis

RFC Random Forest Classifier

$SFTP$ Secure File Transfer Protocol

SVC Support Vector Classifier

TN True Negative

TP True Positive

VAV Variable Air Volume

Chapter 1

Introduction

For the past few years, energy consumption in building systems has been rapidly increasing. This industry accounts for 35 percent of the world's overall energy consumption each year [25]. HVAC, which stands for heating, ventilation and air-conditioning makes up the majority of energy usage in today's buildings. Inadequate maintenance and malfunctioning system components can result in energy waste, lowering HVAC system efficiency and rising operational expenses. As a result, a building automation system (BAS) with predictive maintenance and fault detection and diagnosis (FDD) capabilities is now necessary for efficiency.

The primary goal of FDD algorithms is to detect faults in the system as quickly as they occur. Different approaches can be taken to implement FDD algorithms in a system. When extensive knowledge about the system is available, rule-based FDD [5]–[7] and physical-model-based approaches [18] can be good options, but they are not feasible every time. In particular, when we deal with more complex systems such as the HVAC system in a large shopping mall, generating rules from expert analysis or creating a prediction model will be computationally expensive, which makes these approaches not feasible. Though system complexity is continuously increasing, industries are becoming more data-rich, making data-driven approaches more prevalent.

Building systems can be incorporated with enormous sensors to get data about systems, and we can connect those sensors to the internet. This concept can be connected to the concept of smart buildings and the Internet of Things (IoT). When devices, often known as "things," become automated and connected to the internet, it is known as Internet of Things (IoT). Building systems that are connected to the internet can simplify the control process, and FDD algorithms can easily be incorporated into this system. In our research, we will use an IoT architecture proposed by Yu et al. [19], and use various supervised learning algorithms on the datasets and analyze them to find the most efficient and accurate data-driven supervised FDD method.

1.1 Research Problem

For years, researchers have been concerned about predictive maintenance and FDD. While different researchers use different methods to solve the problem, there are some similar challenges that arise when using a fault detection and diagnosis (FDD) method in building systems.

First of all, the system must be capable of processing large amounts of data. The data generated by building systems is enormous. To deal with that, we'll need to come up with a framework that can handle such a large volume of data. The system must then be able to deliver results in real-time. It will not be a fruitful method to deal with predictive maintenance and fault detection and diagnosis (FDD) approaches if we do not detect the faults in the system in real-time. Fault detection must be done in real-time, and the system must be able to store and analyze historical data. This data should be used to train the system, making the detection process more precise and accurate based on the system's behavior. Another issue is that it is difficult to find an adequately labeled dataset to apply the supervised FDD algorithm. Despite the fact that multiple types of research have been conducted in this area, finding a perfect dataset for training classification algorithms remains challenging. The absence of proper datasets in this domain makes it difficult for researchers to carry out performance analysis of algorithms.

1.2 Research Objectives

In our research, we follow a data-driven approach of fault detection and diagnosis (FDD) methods and intend to find a proper supervised FDD algorithm to detect faults in building systems. For this, performance analysis on several algorithms is done using some labeled HVAC datasets. Through this research we are focusing on some objectives which are:

- Detecting faults in building systems in a quick and accurate manner.
- Describe an IoT ecosystem that will be integrated with the building system.
- Finding the best supervised FDD algorithm to detect faults.

Chapter 2

Literature Review

2.1 Predictive Maintenance and FDD Algorithms

Different types of faults can affect a building system's components, ranging from a single component to a whole system. It has the effect of lowering performance while also increasing energy consumption. As a result, detecting these faults as soon as possible is crucial. Predictive maintenance and fault detection and diagnostics (FDD) algorithms can be used to monitor those systems while also detecting irregularities or faults, which can reduce the operational cost of the systems.

Predictive maintenance keeps track of the current state of the components in order to notify the system and determine whether or not corrective maintenance is required [6]. With the increasing trend of smart buildings, it is a wise decision to integrate predictive maintenance or fault detection and diagnosis techniques into those systems. Fault detection and diagnosis is one of the industry's most well-accepted predictive maintenance principles, on which we can find extensive research in the literature. We can classify FDD into three generalized types based on the existing research in the literature, i.e., physical model-based, rule-based, and data-driven approaches [16]. The physical model-based approach uses physical principles and knowledge of the system's underlying fundamental behavior to create a physical model that represents a faultless state of the system. Then, this state is compared with the current state of the system to detect faults. It is simple to utilize a model-based approach if significant knowledge and information about the system is available [16], but obtaining sufficient knowledge about a system becomes costly and impractical if the system is complex. Rule-based modeling techniques utilize prior knowledge and domain expertise to create a rule space, which is then used to detect the fault symptoms in the system. According to the rule space, any present state of data that exceeds a specified threshold determined by expert analysis is considered a fault [16]. The data-driven method uses real data for analysis. To analyze and detect faults using a data-driven method, the system's historical data is necessary. The data availability of automated building systems is rapidly rising nowadays. As a result of the availability of appropriate datasets, data-driven methods are becoming more prevalent. Data-driven approaches that use historical datasets as a training model can also be classified into three categories, i.e., unsupervised approaches, supervised approaches, and hybrid approaches [26]. To analyze unlabeled data, an unsupervised learning strategy is required. When working with labeled data, on the

other hand, supervised learning approaches are used. We will mainly concentrate on detecting faults using data-driven supervised approaches in this paper.

2.2 Internet of Things (IoT)

The Internet of Things (IoT) is a network comprised of things, where things are wirelessly connected via smart sensors [10]. Devices that we previously used without being connected to the internet are now being connected to it. Smart air conditioners, smart refrigerators, and smart televisions are just a few examples. In addition, we can see that the concept of smart buildings has become increasingly popular in recent years. Control of building systems is being centralized, and those systems are being linked to the internet. Centralized HVAC systems, in particular, are becoming popular and are being adopted to building systems at a rapid pace. In this paper, we will use Yu et al.'s [19] IoT architecture to connect a building system to the cloud, where we will conduct our fault detection analysis.

2.3 Related Works

Different studies used different approaches in the domains of predictive maintenance and FDD. In the early stages of literature, most studies focused on model-based approaches, such as rule-based or physical model-based approaches. It is primarily due to the lack of appropriate datasets in this domain and less complexity of systems. However, the trend has steadily evolved toward a data-driven approach, owing to the increased availability of data regarding building systems in recent years.

Schein et al. [6] employed a rule-based approach to detect faulty air handling units (AHU) in their study. In the AHU controllers, they implemented AHU performance assessment rules (APAR). APAR comprises of some expert rules about the system. They gathered data from a variety of field sites, including an office building, a restaurant, a community college, and a university campus, and used APAR to analyze those data. In another study, Schein et al. [5] applied a hierarchical decision-making framework to create a hierarchical FDD algorithm that consists of rules for detecting system faults. Qin and Wang [4] used a study on faults in VAV terminals of a site survey and examined the results of it. They took a hybrid approach to detect faults. To detect faults, they used expert rules, performance indicators, and statistical process control models, which were backed up by a pattern recognition method. To get around fault isolation, they used two distinct pattern recognition indexes. A PCA-based technique was utilized to detect VAV terminal flow sensor biases and reconstruct the defective sensors. Lo et al. [7] developed an approach based on a fuzzy genetic algorithm (FGA). A fuzzy system is a fault detection method based on rules. It can be viewed as a classification problem, where the fuzzy system acts as a classifier, allowing it to distinguish between distinct faults based on rules. The precision of the fuzzy rules is important because it determines how accurately faults can be detected. Fault detection using fuzzy rules was once widely popular. Several other academics [1]–[3] have proposed fuzzy system-based fault detection and diagnostic systems.

In recent years, the focus of study in this field has shifted to data-driven approaches. The number of building systems is growing. As a result, it is now easy to obtain accurate data about systems in order to implement a data-driven method. According to Amruthnath and Gupta [13], unsupervised learning is preferable for fault detection because of the limited availability of historical data on the systems. To identify the right features and reduce the dimensionality of the dataset, they used principal component analysis (PCA). They then used the dataset to test a variety of unsupervised learning algorithms, including hierarchical clustering, K-means and Fuzzy C-means clustering, and the Gaussian mixture model (GMM). In recent research from Yan et al. [28], a framework that utilizes the generative adversarial network (GAN) is proposed for detecting faults in AHU to solve the imbalanced raw data problem. They first use GAN to rebalance the dataset, then use supervised methods to classify the faults in the system.

It is very difficult and time-consuming to model a complex and non-linear system, such as a large-scale building system, using expert analysis, mathematical functions, and physical models [8]. Rule-based approaches are only reliable with accurate expert analysis and comprehensive knowledge of the system. Nevertheless, if the system is complex, they become an expensive and unfeasible choice. As a result, adopting rule-based and physical model-based methodologies for complicated and large-scale projects is no longer a viable option. Data-driven models, on the other hand, are becoming increasingly popular in the field of FDD. Even in complex and large-scale systems, they are reliable, feasible, and powerful [8]. We are using supervised learning algorithms in our research since they are the best way to generate an optimal and efficient output when labeled data about a system is available.

Chapter 3

Methodology

Our model's initial task is to obtain data from the sensor deployed in the system based on the IoT ecosystem. It will generate a large amount of data, which will be stored in the cloud, from which we will extract the information. Then, using a variety of machine learning algorithms, we'll look for any potential flaws in the system. Even minor flaws in the system will be predicted by the model, and the user will be notified for preventative maintenance. Furthermore, the data will be used to improve the system and will aid in the reduction of energy waste and the smooth operation of the system.

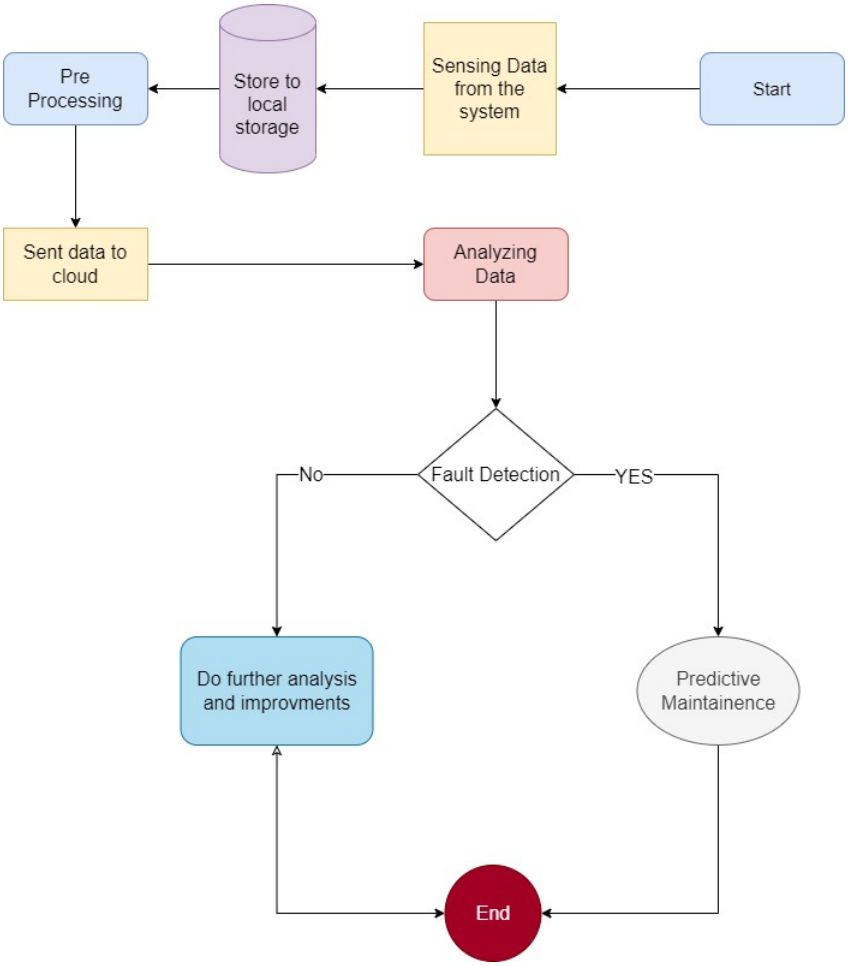

Figure 3.1: Work Flow of the proposed model system

3.1 IoT Ecosystem

We'll use Yu et al.'s IoT architecture [19]. They created an ecosystem to deal with big data issues in the IoT domain. The data ingestion layer, data management layer, data analytics layer, and data visualization layer are the layers that make up the architecture. Its architecture is based on edge nodes, cloud clusters, and data lakes (ECD-architecture). The data ingestion layer, which is the initial layer in the architecture, is responsible for sending a massive volume of sensor data to the Hadoop system. In this layer, data is stored on several servers and the metadata server keeps record of structured data measurement logs, whereas the OPC server handles numerical sensor data. They also use historical servers to store data and the OPC Collector to retrieve and output values from the OPC Server to a text file. After a predetermined time-interval, a compressed version of the text file is uploaded to the AWS cluster via Secure File Transfer Protocol (SFTP). A VPN gateway is deployed to secure the entire network.

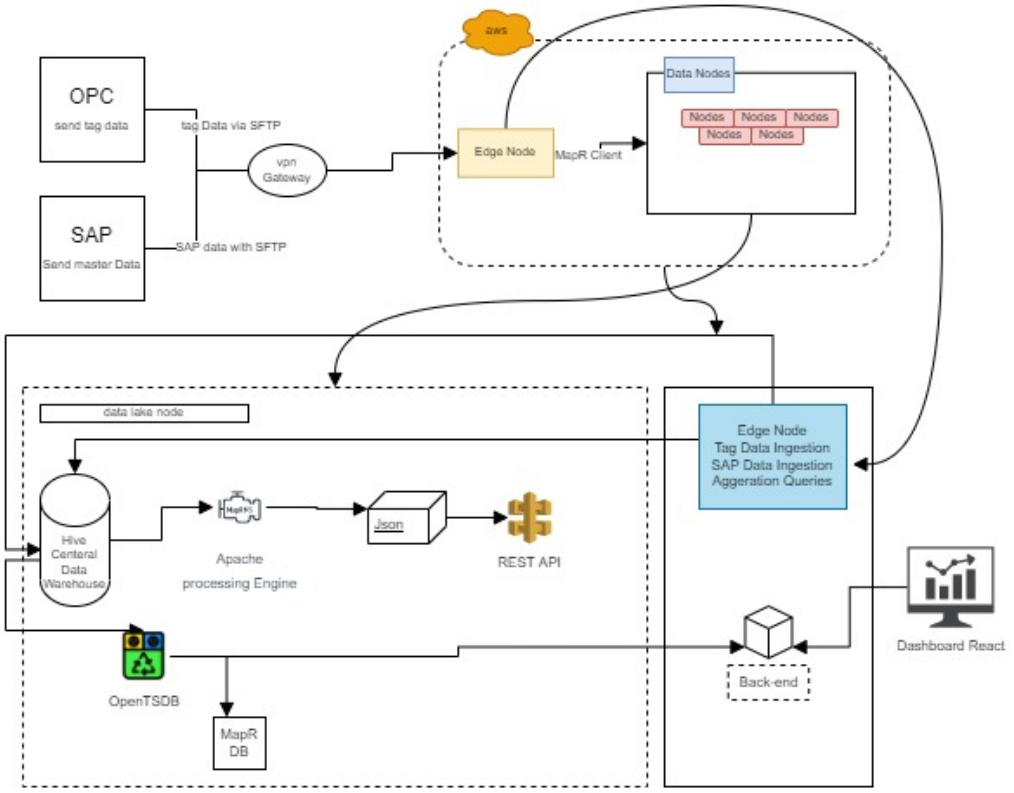

Figure 3.2: IoT Ecosystem (ECD)

The big data management layer is responsible for managing massive data and storing them efficiently [19]. Each batch of data in the edge node gets replicated three times, as per the architecture. After that, utilizing HDFS, these data are randomly spread out across the cloud nodes. In this layer, data lake is used to keep data from the edge node Data lake is a cloud-based data storage architecture, where every kind of data at any scale can be stored. After a preliminary data analysis and processing, structure data is obtained from the raw data and then converted into DataFrame format. Yu et al. [19] then use two different storages, namely Apache Hive, It serves

as a central data warehouse, and the MapR database, which is a NoSQL database. The stored data in the Apache Hive acts as historical data for data analysis, and the data stored in the MapR database is used for real-time analysis. Apache Spark is used as the data processing engine to process the massive volume of data in an efficient manner. In the analytics layer, we will use several algorithms to analyze the performance of different supervised learning algorithms.

3.2 Implemented Algorithms

In our proposed model we tried to implement 8 algorithms with testing and training the data we got from our described dataset. These 8 algorithms have different accuracy rates with different characteristics in terms of machine learning techniques. Here we will briefly discuss all the algorithms we used.

3.2.1 Logistic Regression

Logistic Regression is another supervised algorithm which is basically used for predicting the probability of outcome we desire from a model. This method basically follows the rules of the binary regression model and tries to find the desired output based on two classes. By the value 1 it shows the success ratio and by 0 it shows the failure part. Theoretically this algorithm runs its process based on binary regression but based on the categories of those numbers, it can be divided into two parts. One is binomial and the other one is multinomial. The binomial method focuses on the numerical process such as 0 and 1 through which it predicts the case as a win or loss. On the other hand, the multinomial process focuses on three or more variables which don't have any kind of numerical attributes. [17]

Algorithm Used: $P(Y-1)$ **as a function of X.**

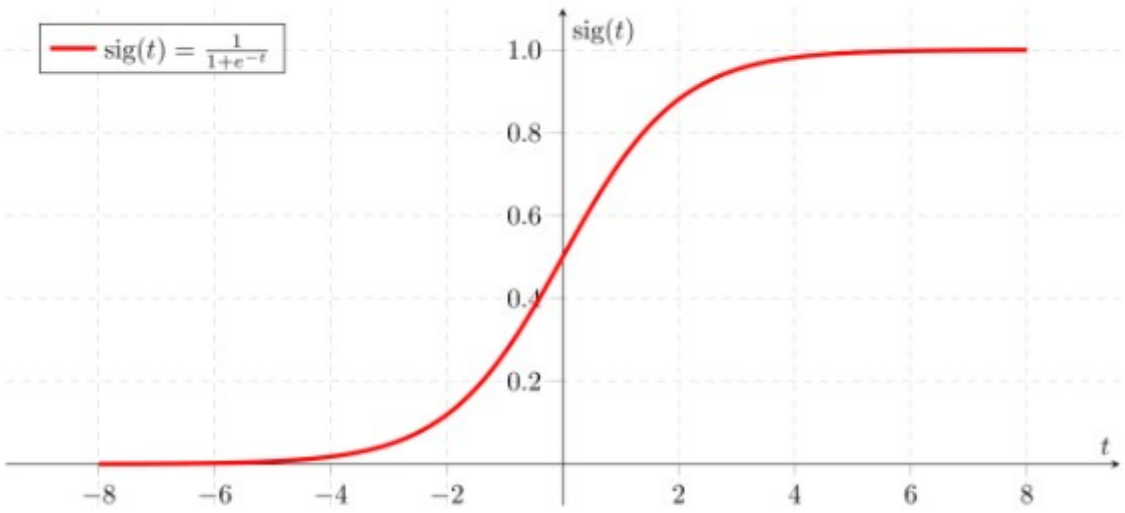

Figure 3.3: Logistic Regression.

3.2.2 K-Nearest Neighbors Classifiers

K-Nearest Neighbor also known as K-NN is another popular supervised algorithm technique used in machine learning models. This method is a simple and constructed method. This algorithm predicts the similarity in its stored data and shows its output based on the class it creates from its predictions. This algorithm is also used for regression and classification problems. This algorithm is also known as lazy learner algorithm because it can not train the dataset earlier rather than it takes its action while it's time for the classification. Afterwards, when it gets a new data related to the maximum type of the dataset then it predicts it through it. This method uses the euclidean distance process while processing the dataset. [15]

Algorithm Used: $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

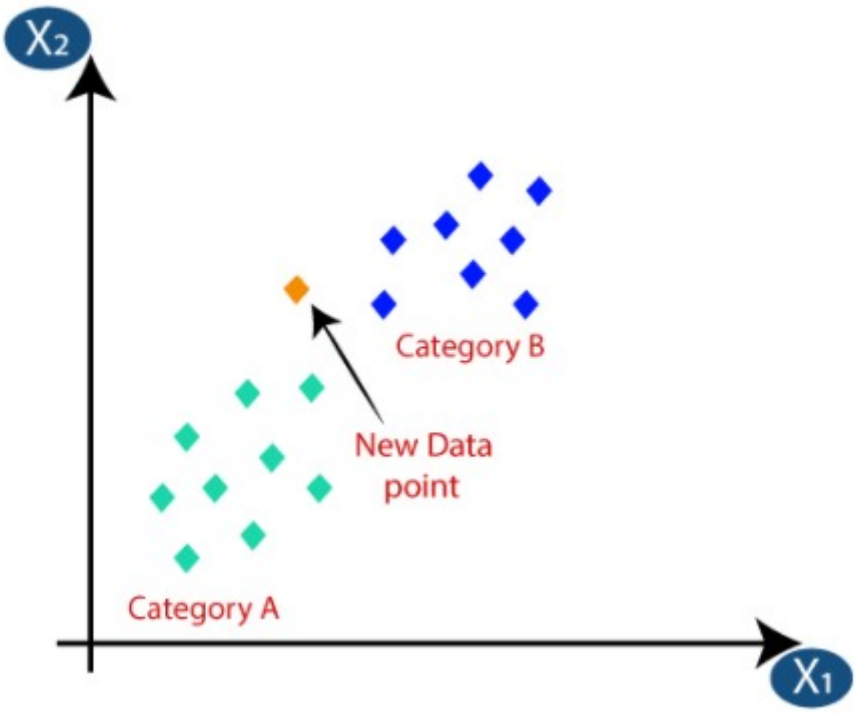

Figure 3.4: K-NN Classifier.

3.2.3 Naive Bayes Classifier

Another popular supervised algorithm which is used in many sectors of our daily life projects is called the Naive Bayes Classifier. This method is majorly used in text classification which requires a high dimensional training dataset. This algorithm is effective in terms of making fast machine learning models and predicting more accurately through them. This method predicts its outputs based on the objectified probability of any case. This method basically works on the basis of bayes theorem. In terms of the working methodology, this algorithm first tries to convert the dataset into a frequency table. Afterwards, it tries to make a similar table by predicting the existing table and its actions. Finally, it uses the bayes theorem to calculate the final prediction and output. [30]

Algorithm Used: $P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}$

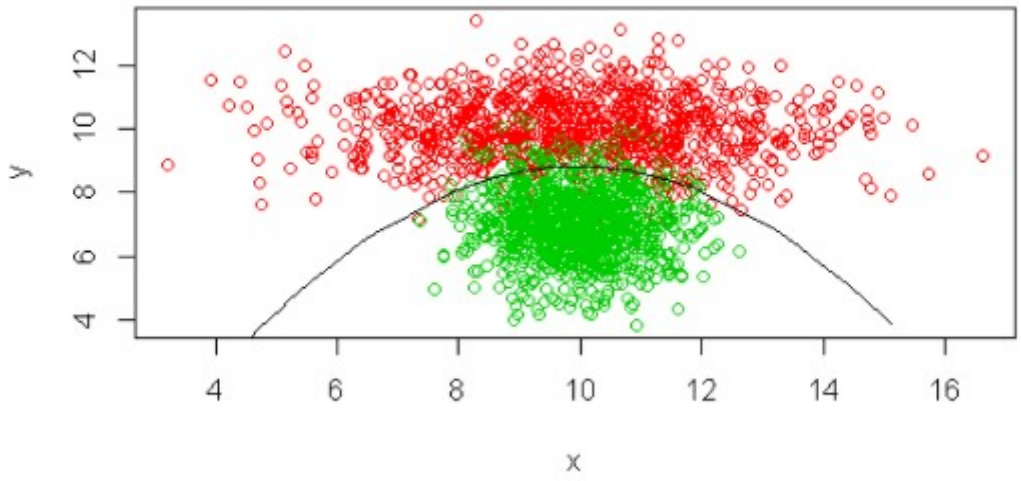

Figure 3.5: Naive Bayes Classifier.

3.2.4 Support Vector Machine Classifier

Support Vector Machine also known as SVM is another renowned supervised algorithm in the sector of machine learning. This algorithm is also being vastly used in terms of classification and regression problems. This method basically creates the best decision boundary and tries to fit in the data that are going to be predicted. The boundary line that is being created by the method is also known as hyperplane. This algorithm can be classified into two parts. One is linear SVM and the other one is the non-linear SVM. Linear SVM basically separates the data linearly where the non-linear SVM does the opposite action of linear SVM. While creating the hyperplane SVM tries to choose the highest vector point for the prediction purpose. [14]

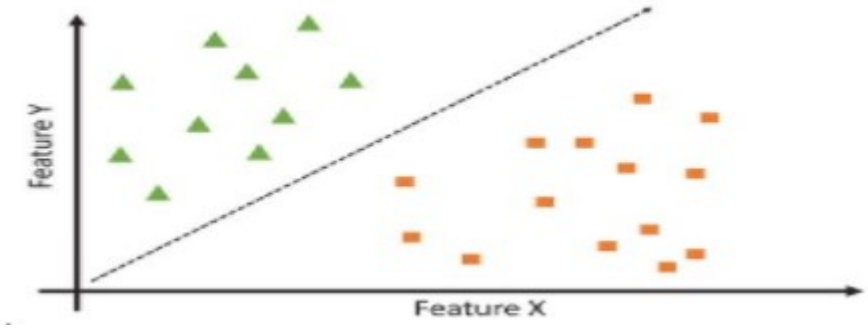

Figure 3.6: Support Vector Machine Classifier.

3.2.5 Random Forest Classifier

Random Forest Classifier is another popular classification based algorithm that is used for predicting and making accurate decisions about finding error data. This

method also belongs to the supervised learning model and works based on the idea of ensemble learning process. For solving problems of classification and regression it is vastly used. Random forest classifier combines multiple trees and predicts the output by a voting system from all the leaf nodes of those trees. Compared to other algorithms, Random forest classifier takes less time to detect the errors. Moreover, it shows accurate results even with large dataset. Additionally, this method shows accurate results even if there is loss of data in the dataset. In terms of its working procedure, this method first creates a random combination of several trees and predicts its output based on the maximum nodes of the same data. Hence, it repeats all the steps one by one and shows the predicted outcome. [29]

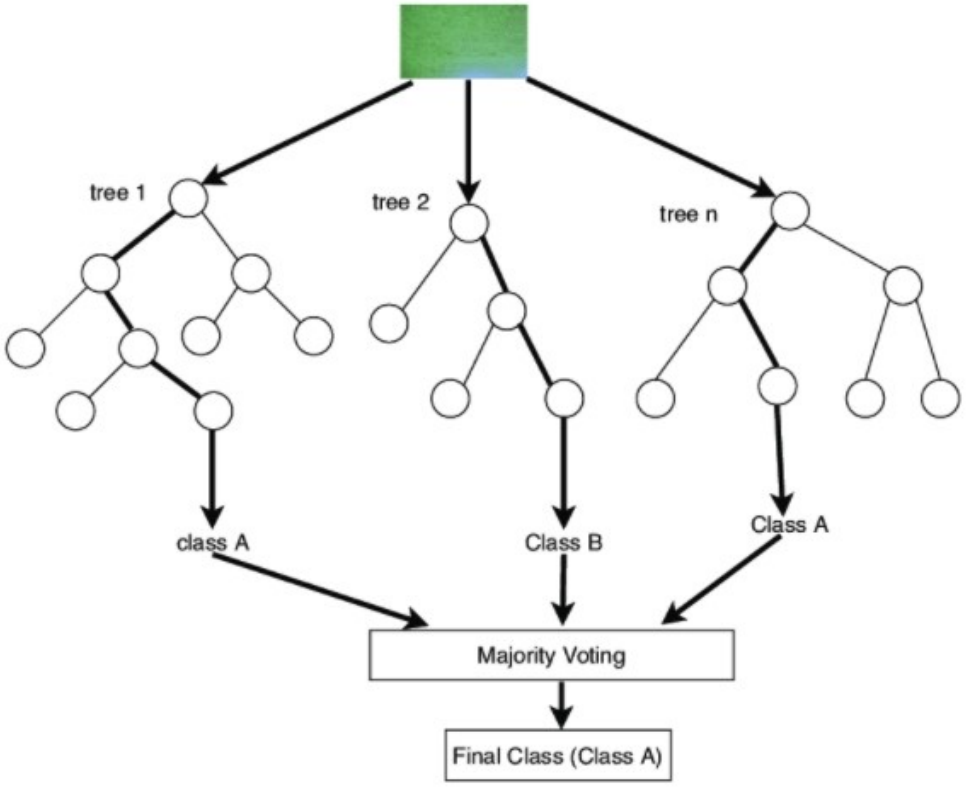

Figure 3.7: Random Forest Classifier.

3.2.6 Decision Tree Classifier

Decision Tree is one of the finest supervised machine learning techniques which is being used vastly all over the technical world. This algorithm is modeled like a tree shaped algorithm with branches, nodes and leafs. This method is basically used for classification and regression problems. To demonstrate the features of a dataset it uses the nodes, the decision rules are being represented via the branches and the outcome could be predicted via the leafs of the tree. The Decision Tree algorithm is distinguished among two parts. One is the decision node and the other one is the leaf node. The algorithm makes its decision via the decision node and its branches. It shows it outputs from the decisions via the leaf node. [21]

Algorithm Used: $E_p = TP - FP$

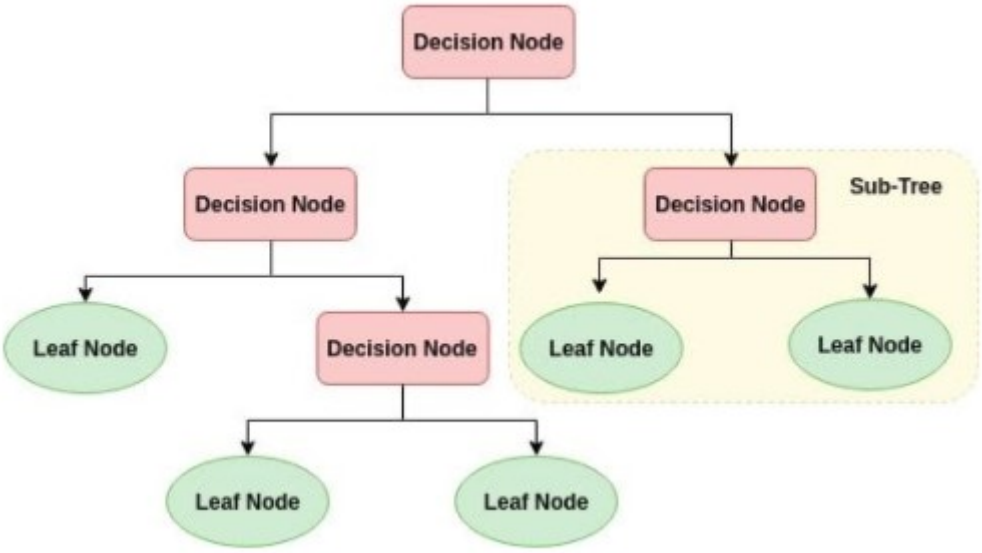

Figure 3.8: Decision Tree Classifier.

3.2.7 MLP Classifier

MLP Classifier which is elaborately known as multi layer perceptron is a learning of neural network algorithm. This method also solves the classification and regression based problems and predicts accuracy of any dataset. This algorithm is different from others because this algorithm uses underlying neural network techniques for its classification purpose. This method creates layers to predict its classifications. Such as, input layer, hidden layer, output layer etc. The depth of the hidden layer describes the depth of classification and attributes. [27]

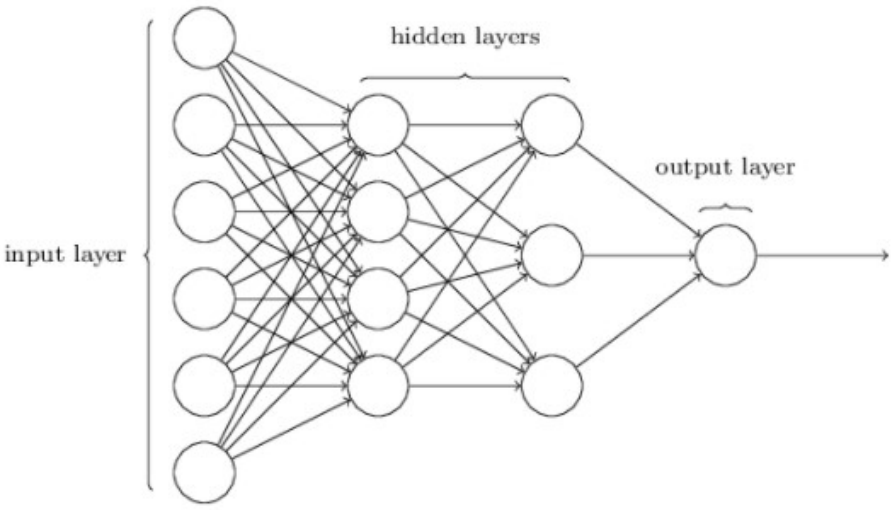

Figure 3.9: MLP Classifier.

3.2.8 Extra Tree Classifier

Extra tree classifier also known as extremely randomized tree classifier is a method which is also vastly used for solving classification and regression problems. This method tries to classify any dataset by creating a forest which stores related decision trees from a dataset. This method is an extended version of the random forest classifier with some different attributes. It creates every decision tree from the notation of the authentic training data and predicts the result combining those trees outputs. Afterwards, it is captured with some random sample of features from the dataset and it classifies the best and worst values from those samples. [24]

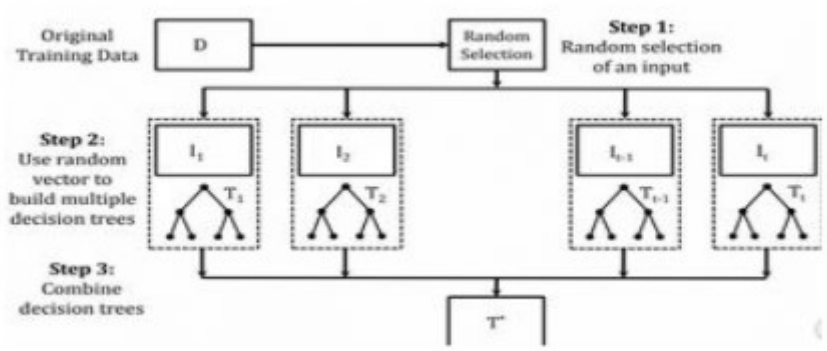

Figure 3.10: Extra Tree Classifier.

Chapter 4

Data Representation

4.1 Dataset description

Despite the widespread use of IoT in building management systems, there is a scarcity of publicly available data for fault detection and predictive maintenance. As a result, obtaining appropriate data to serve our purpose was difficult. Building fault detection data [23] was chosen for our model since it is designed for fault detection and diagnostics (FDD) methods. They've gathered data that's been sorted into two categories: occupied and vacant. The presence of humans inside a building determines its occupancy. Some of the datasets are gathered from real experiments, while others are computer simulations. We employed two datasets for our model: one is experimental data, and the other is a simulation result.

	Dataset Name	**Creation Process**
SET-A	Simulated MZVAV-1(Multi-zone variable air volume AHU dataset)	HVACSIM+ and an EnergyPlus-Modelica co-simulation were used to create this model.
SET-B	Experimental MZVAV-2-1 (Multi-zone variable air volume AHU dataset)	Created at the Lawrence Berkeley National Laboratory in Berkeley, California, using three experimental research facilities

Table 4.1: Used datasets and the approach of creation.

4.2 Exploratory Data Analysis

The datasets are created in different ways, but they both have the same characteristics. The values of various sensors are represented by each feature (or column). Both datasets have two classes: faulty and faultless, where '0' denotes no faults and '1' denotes data that is faulty. The purpose is to find errors in the 'Fault Detection Ground Truth' column. Figure 4.1 and 2.2 describes the features of the datasets respectively

	AHU: Supply Air Temperature	AHU: Supply Air Temperature Set Point	AHU: Outdoor Air Temperature	AHU: Mixed Air Temperature	AHU: Return Air Temperature	AHU: Supply Air Fan Status	AHU: Return Air Fan Status	AHU: Supply Air Fan Speed Control Signal	AHU: Return Air Fan Speed Control Signal
count	21600.000000	21600.000000	21600.000000	21600.000000	21600.000000	21600.000000	21600.000000	21600.000000	21600.000000
mean	62.996525	57.000463	58.220122	67.488618	72.611413	0.534722	0.533611	0.474657	0.398145
std	7.372700	4.000440	21.458405	7.875141	1.679548	0.498804	0.498881	0.261236	0.189342
min	0.000000	55.000000	-0.490000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	55.150000	55.000000	44.980000	64.100000	71.340000	0.000000	0.000000	0.200000	0.200000
50%	64.840000	55.000000	63.640000	68.660000	72.810000	1.000000	1.000000	0.620000	0.500000
75%	70.430000	55.000000	74.550000	72.590000	73.440000	1.000000	1.000000	0.720000	0.570000
max	79.690000	65.000000	91.850000	108.240000	79.120000	1.000000	1.000000	1.000000	0.800000

AHU: Exhaust Air Damper Control Signal	AHU: Outdoor Air Damper Control Signal	AHU: Return Air Damper Control Signal	AHU: Cooling Coil Valve Control Signal	AHU: Heating Coil Valve Control Signal	AHU: Supply Air Duct Static Pressure Set Point	AHU: Supply Air Duct Static Pressure	Occupancy Mode Indicator	Fault Detection Ground Truth
21600.000000	21600.000000	21600.000000	21600.000000	21600.000000	2.160000e+04	21600.000000	21600.000000	21600.000000
0.341757	0.331989	0.443025	0.168748	0.083330	1.400000e+00	0.738494	0.500000	0.133333
0.372642	0.368341	0.400821	0.252140	0.249985	5.278122e-13	0.695622	0.500012	0.339943
0.000000	0.000000	0.000000	0.000000	0.000000	1.400000e+00	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000	0.000000	1.400000e+00	0.000000	0.000000	0.000000
0.400000	0.400000	0.400000	0.000000	0.000000	1.400000e+00	1.340000	0.500000	0.000000
0.470000	0.470000	1.000000	0.290000	0.000000	1.400000e+00	1.390000	1.000000	0.000000
1.000000	1.000000	1.000000	1.000000	1.000000	1.400000e+00	2.370000	1.000000	1.000000

Figure 4.1: Description of the SET-A features.

	AHU: Supply Air Temperature	AHU: Supply Air Temperature Set Point	AHU: Outdoor Air Temperature	AHU: Mixed Air Temperature	AHU: Return Air Temperature	AHU: Supply Air Fan Status	AHU: Return Air Fan Status	AHU: Supply Air Fan Speed Control Signal	AHU: Return Air Fan Speed Control Signal
count	21600.000000	2.160000e+04	21600.000000	21600.000000	21600.000000	21600.000000	21600.000000	21600.000000	21600.000000
mean	53.797004	5.504000e+01	48.531414	65.089730	71.437411	0.672037	0.672037	0.482238	0.482238
std	2.557170	1.678341e-11	20.281195	7.390938	3.710836	0.469482	0.469482	0.353356	0.353356
min	45.010000	5.504000e+01	-14.190000	23.720000	55.090000	0.000000	0.000000	0.000000	0.000000
25%	52.010000	5.504000e+01	36.550000	57.180000	70.010000	0.000000	0.000000	0.000000	0.000000
50%	54.090000	5.504000e+01	52.590000	65.710000	72.010000	1.000000	1.000000	0.640000	0.640000
75%	55.040000	5.504000e+01	62.780000	72.060000	74.280000	1.000000	1.000000	0.670000	0.670000
max	71.760000	5.504000e+01	90.140000	78.460000	85.680000	1.000000	1.000000	1.000000	1.000000

AHU: Outdoor Air Damper Control Signal	AHU: Return Air Damper Control Signal	AHU: Cooling Coil Valve Control Signal	AHU: Heating Coil Valve Control Signal	AHU: Supply Air Duct Static Pressure Set Point	AHU: Supply Air Duct Static Pressure	Occupancy Mode Indicator	Fault Detection Ground Truth
21600.000000	21600.0	21600.000000	21600.000000	2.160000e+04	21600.000000	21600.000000	21600.00000
0.211304	0.0	0.043344	0.000046	4.000000e-02	0.034456	0.485463	0.85588
0.324680	0.0	0.082865	0.006804	9.534261e-15	0.032932	0.499800	0.35122
0.000000	0.0	0.000000	0.000000	4.000000e-02	-0.020000	0.000000	0.00000
0.000000	0.0	0.010000	0.000000	4.000000e-02	0.000000	0.000000	1.00000
0.000000	0.0	0.020000	0.000000	4.000000e-02	0.040000	0.000000	1.00000
0.240000	0.0	0.040000	0.000000	4.000000e-02	0.040000	1.000000	1.00000
1.000000	0.0	0.600000	1.000000	4.000000e-02	0.120000	1.000000	1.00000

Figure 4.2: Description of the SET-B features.

From figure 4.3 and 4.4 we get the idea about the Density of the various key features in both datasets is presented in this section.

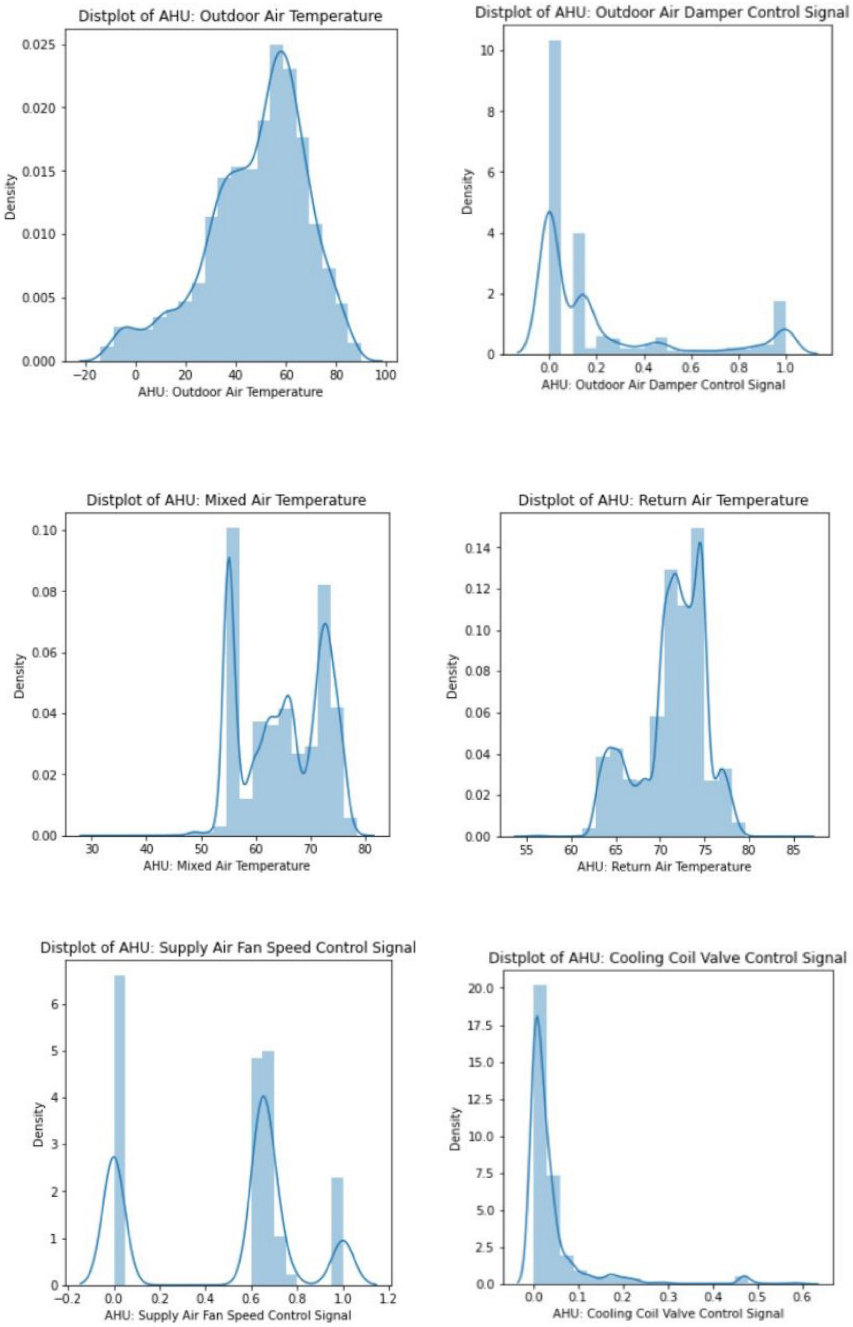

Figure 4.3: The density of different SET-A features that highly correlate to the two classes

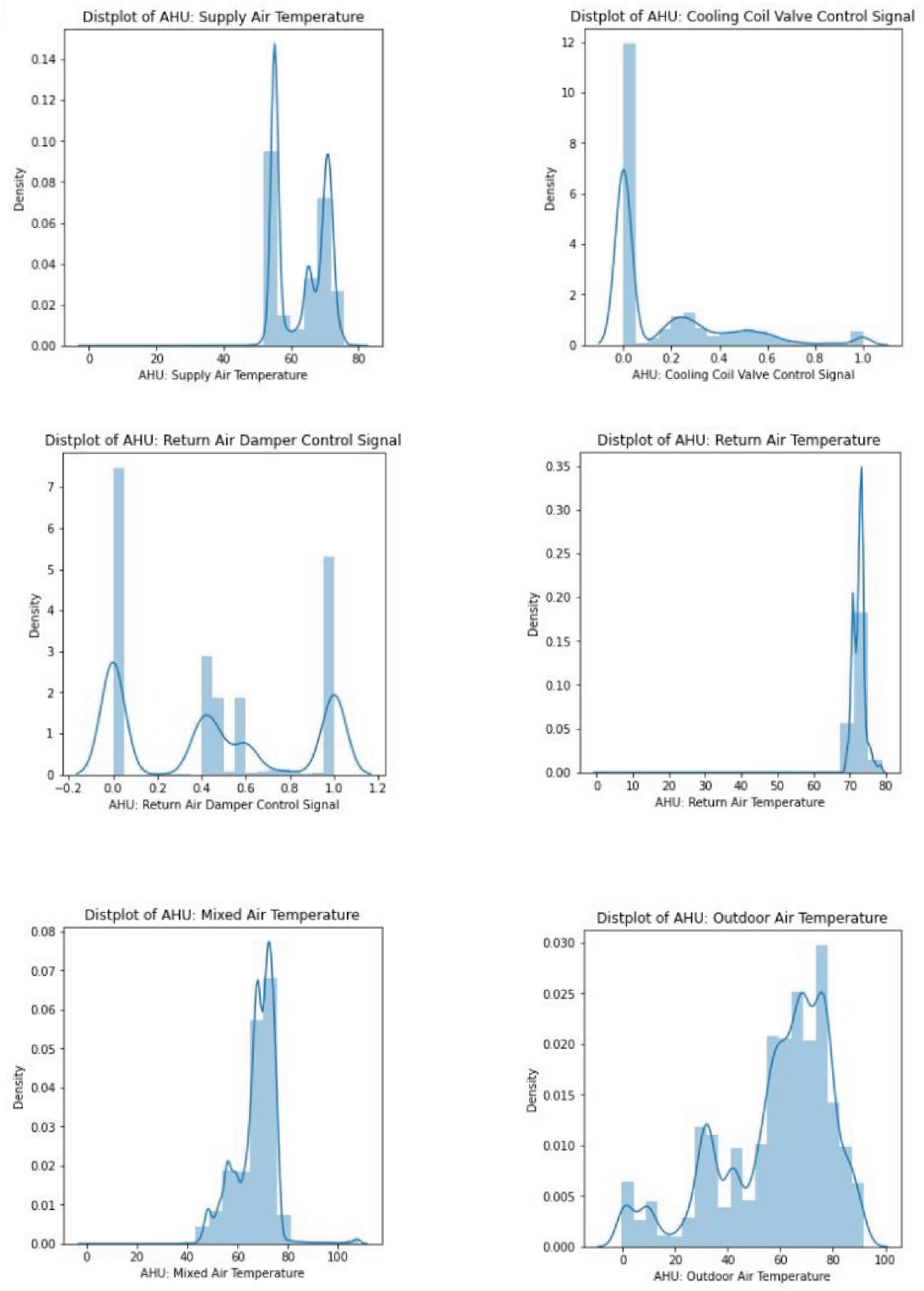

Figure 4.4: The density of different SET-B features that highly correlate to the two classes.

4.3 Data Pre-Processing

To begin, we are using Machine Learning models, and data preprocessing is the first step in the process. We will acquire the raw data generated by the sensors when sensors are integrated with real-world systems and value is retrieved from there. While processing, all of the data may not be required to reach our goal, or the data may be incomplete, contain irrelevant information, or be missing certain critical variables. As a result, we must preprocess or transform raw input into relevant data in order to achieve the most efficient outcome.

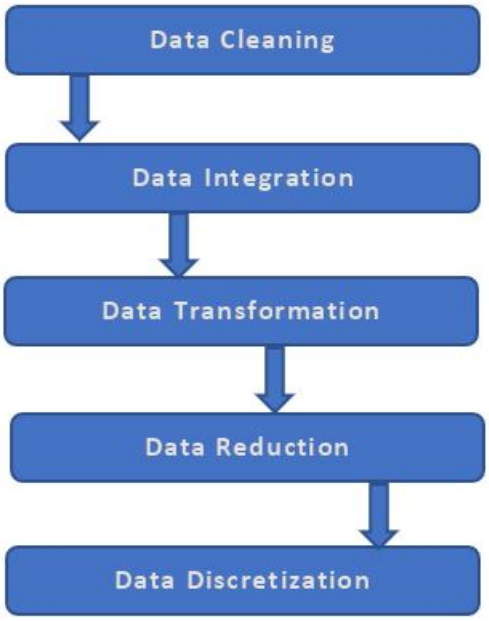

Figure 4.5: Steps of Data Preprocessing

The initial step was to clean up the data, removing extraneous features such as 'Datetime' from both datasets. The SET-A data rows with null values are eliminated. The SET-A dataset was created using summation and has 272160 rows, with 233280 rows belonging to the Faulty class (denoted by 1) and 38880 rows belonging to the Faultless class (denoted by 0). The SET-B dataset, on the other hand, is based on real-world experiments and contains 21600 sets of data, 18720 faultless and 2880 faulty. It demonstrates that 86% of the data in SET-A is faulty, causing an imbalance. We used the undersampling approach to solve this problem, which is a collection of techniques for balancing the class distribution. It removes examples from the training sample that correspond to the majority class in order to improve the class distribution's balance,for example, lowering the skew from 1:100 to 1:10, 1:2, or even 1:1. We reduced the SET 1 dataset to 21600 samples making it equal to the SET-B dataset for better comparison and results in few algorithms. We also used the StandardScaler and MinMaxScaler feature scaling algorithms to acquire more precise results. After adding MinMaxScaler, we were able to achieve better results.

4.4 Data Analysis Metrics

For a single algorithm, we calculated the training and testing scores, as well as the precision, recall, and accuracy metrics for the particular model. The key metrics of any pattern recognition technique that aids in the detection of a specific pattern in a given set of data are precision, recall, and accuracy.

4.4.1 Confusion Matrix

Confusion matrix compares the actual targets values with the machine learning model's predictions. This provides us with a comprehensive picture of how well our classification model is working and the types of errors it makes. Positive or Negative are the two possible values for the target variable. Where the columns reflect the target variable's actual values and the rows represent the target variable's predicted values [20].

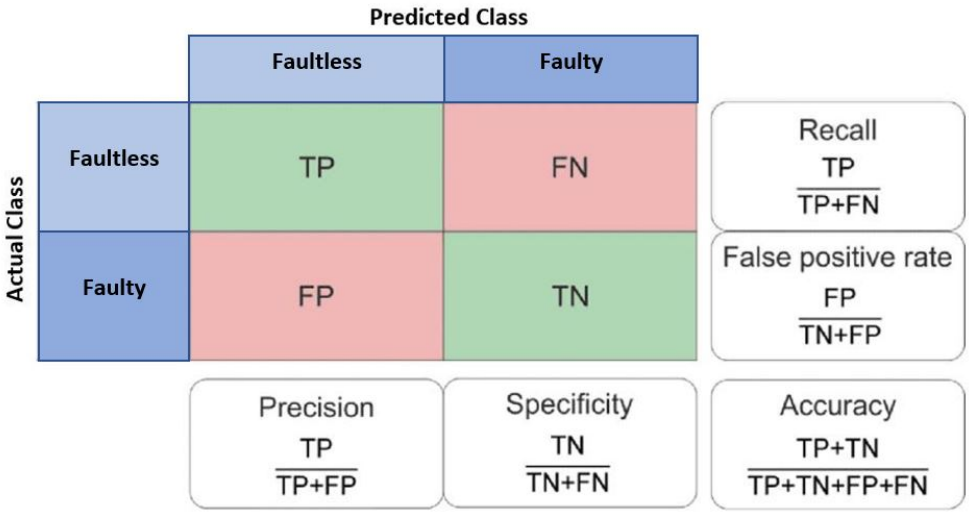

Figure 4.6: Confusion matrix with advanced classification metrics

The target variable has two values as displayed in figure 4.6, True Negative (TN) is an output that displays the number of correctly diagnosed negative situations. The terms FP and FN refer to False Positive and False Negative values, respectively. FP refers to the number of actual negative cases categorized as positive, while FN refers to the number of actual positive examples classified as negative. [22]

4.4.2 Advanced classification Metrics

Precision - - Precision basically gives the benefits in the times of high value of False Positive. It tries to demolish the highness of false positive value and show its result based on that. The ratio of accurately predicted positive observations to the total predicted positive observations is known as precision. [11] Mathematically it is

$$Precision = \frac{TP}{TP+FP}$$

Accuracy - For the assumption of the perfect training and testing set, accuracy is used. As the datasets contain scattered data, we have to train and test them for our prediction purpose. The most intuitive performance metric is accuracy, which is just the ratio of accurately predicted observations to total observations. [11]. Mathematically it is

$$Accuracy = \frac{TP+TN}{TP+TN.FP+FN}$$

Recall - Recall basically used for demolishing the high values of False Negative. The ratio of accurately predicted positive observations to all observations in the actual class is referred to as recall. [11]. Mathematically it is

$$Recall = \frac{TP}{TP+FN}$$

F1 Score - The F1 Score is a balance among precision and recall. The F1 score is often used to assess performance of binary classification, but it may also be applied to multi-class classifications. This score considers both false positives and false negatives. When the F1 score is 1, the model is deemed perfect, but when it is 0, the model is considered a complete failure. Although it is not as intuitive as accuracy, F1 is frequently more useful than accuracy, especially if the class distribution is unequal. When false positives and false negatives have equivalent costs, accuracy works well. Mathematically it is

$$F1Score = \frac{2*(Recall*Precision)}{(Recall+Precision)}$$

Specificity - Specificity basically meets the criteria by screening the test and detect the accurate True Negative value. It refers to the ratio of true negative to the summation of true negative and false positive. Mathematically it is

$$Specificity = \frac{TN}{TN+FP}$$

Receiver Operating Characteristic (ROC) Curve -A probability curve that compares the TPR against the FPR at different threshold levels.

$$TPR = Sensitivity \text{ and } FPR = 1 - Specificity.$$

Chapter 5

Analysis & Results

We will discuss the performance of our model on both simulated (SET-A) and experimental data(SET-B) in this section. After performing the necessary preprocessing, we trained our models with 70% of the data and left 30% for testing as shown in Table 5.1.

Dataset	**Category**	**Values**
SET-A	Faulty	233280
	Faultless	38880
	Training set	190512
	Testing set	81648
	Total	**272160**
SET-B	Faulty	2880
	Faultless	18720
	Training set	15120
	Testing set	6480
	Total	**21600**

Table 5.1: Dataset Description.

5.1 SET-A: MZVAV-1 (Simulation data)

In the SET-A simulation dataset, we initially acquired 83.7% accuracy for Logistic Regression (LR), however after applying standard scalar, the accuracy improved to 83.84%. As a result, it was unable to anticipate 13.6 percent of the erroneous data. LR has an precision rate of 85.94 percent, a recall rate of 97.05 percent, and a specificity of only 3.81%. Lastly, the f1 score is 91.16%.

We utilized 5 nearest neighbors for KNN, and SET-A yields a testing accuracy of 91.45% with the default parameters. After scaling the data, however, it increased

to 98.82. With a f1-score of 99.32%, the recall was 99.60 %. It had a precision of 99.03 % and a specificity of 94.16 %.

Again for SET-A dataset, the decision tree outperformed LR. Entropy was used to assess the quality of the data split. On the testing dataset, our model had a 98.91% accuracy rate. We reduced biases using undersampling method because the dataset was imbalanced with high proportion of faulty class, with 13.70% of samples classified as "faultless" and 85.22 % as "faulty,". obtained a f1-score of 99.37% and a recall of 99.42%. It had a 99.31% precision and a 95.87% specificity.

We achieved an accuracy of 98.89% using the Random Forest classifier model in the SET-A, with a precision level of 98.89 % While we had a recall of 99.82 %, we only had a specificity of 93.30 %. 99.35 % was the f1-score.

Using the Naive Bayes classifier model, we were able to attain very low accuracy of 14.17 % and a precision of 0 %. While its recall was also 0 %, its specificity was only 100 %. But The f1-score was 0%. The high amount of categorical data within that dataset is to account for the poor performance. We found that several of the columns are totally made up of ones and zeros. As a result, it lacks a normal distribution and GNB cannot perform effectively in the absence of normal distribution [9].

The results achieved when implemented on the SVC model were around 85.83 %; although SVC works better on a short dataset [12], we trimmed the dataset to get better results.There are no real negatives because when specificity is 0 percent, and all data without the condition is false positive. Since 100% of the TP was discovered, we received a recall of 1 where the F1-score is 92.38.

We also used the MLP classifier model and fetched an accuracy score of 85.83% with precision of 85.83%. It has a recall of 100% and specificity of 0%. It also obtained a f1-score of 92.38%.

Finally, we implemented the Extra Tree Classifier algorithm which gave us an accuracy of 98.10% with a precision of 98.21%. It has a recall of 99.60% and specificity of 89.13%. It ends up giving a f1-score of 98.90%.

5.2 SET-B: MZVAV-2-1 (Experimental data)

We initially achieved 94% accuracy for Logistic Regression (LR) in the SET-B dataset (Experimental data), however after Feature Scaling with StandardScaler, the accuracy improved substantially to 96.44 %. LR has an 83.56 % precision rate, a 91.2 % recall rate, and a specificity of 97.24 %.

With the default parameters, KNN produces a testing accuracy of 99.35 %, while SET-B generates a testing accuracy. However, after scaling the data, it increased to 99.65%. The recall was 98.38 %, with a f1-score of 98.67 %. It had a 99.95% precision and a 99.84 % specificity.

The decision tree model performs well in the SET-B (experimental dataset). With an accuracy of 99.74%, predicted from the test data. The recall and f1-score were respectively 98.96% and 99.02%. In this dataset, we discovered that random forest has a training accuracy of 100%, however the decision tree did not.
Random forest classifier performs incredibly well on testing dataset, with the best result of 99.91 % accuracy among the models. The recall, specificity, and f1-score of SET-B were all 99.42%, 99.98%, and 99.65%, respectively.

When using the SVC model to train and test, the results were 95.74% accurate. We achieved better results compared with SET-A because SVC works better on a little dataset. Recall is 100% percent, while specificity is 95.09%. A precision of 75.79 percent was achieved.

We were able to get an accuracy of 79.26 %, the lowest among the modes, and a precision of 39.13 percent using the Naive Bayes classifier model. While it had 100% recall, it only had 76.07 percent specificity. The f1-score came in at 56.25 %.

While implementing the MLP classification model, we achieved an accuracy of 95.05% with a precision of 73.15%. It shows a recall of 99.31% and specificity of 94.39%. It gives a f1-score of 84.24%.

Additionally, we used the Extra Tree Classification model and achieved an accuracy of 99.91% which is the best testing result of this project. It gives us a precision of 99.77% and recall of 99.54%. It rounds up its result analysis by giving a specificity of 99.96% and f1-score of 99.65%.

5.3 AUC-ROC Curve

On selected datasets, we used ROC and AUC curves to compare different algorithms. In the curves sensitivity is reported in the horizontal plane, and specificity is reported in the vertical plane. The term AUC refers to the area under the curve. The ROC curve is created by combining the points from plotting the ROC for various criteria. The algorithm performs better with a larger AUC.

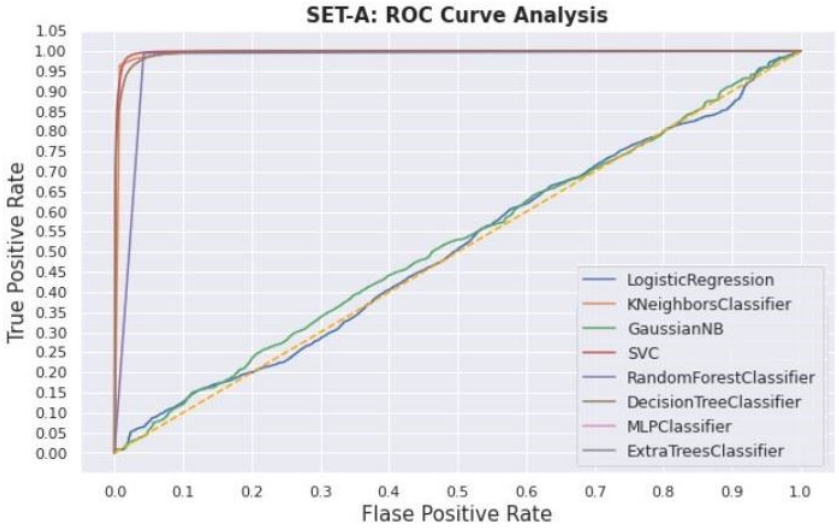

Figure 5.1: ROC Curve of SET-A

The AUC of all the algorithms we utilized in SET-A shown in Figure 5.1. We can observe that KNeighbors, Random forest, Decision tree, and Extra trees are all near the y-axis and thus provide the best possible results. In contrast Naive Bayes and MLP algorithms are falling behind in the curve denoting poor performance.

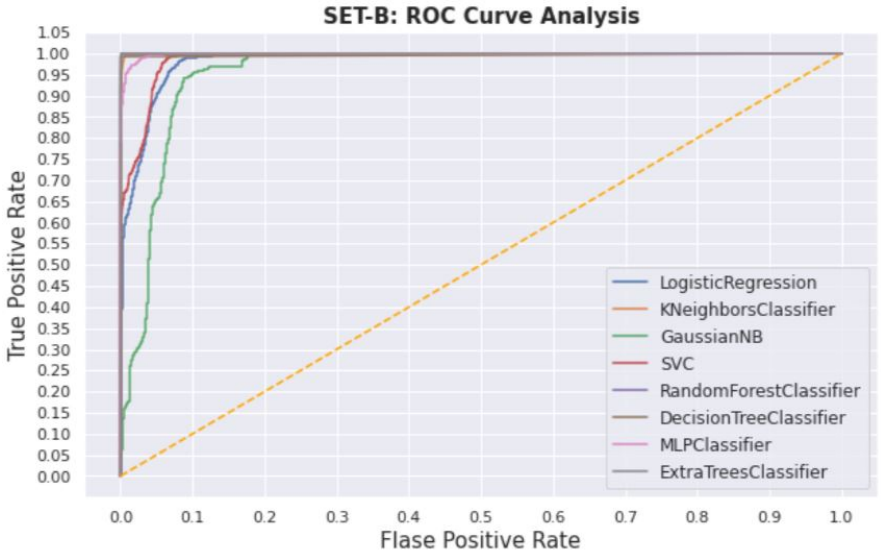

Figure 5.2: ROC Curve of SET-B

The AUC of all the algorithms we utilized in SET-A shown in Figure 5.1. We can observe that KNeighbors, Random forest, Decision tree, and Extra trees are all near the y-axis and thus provide the best possible results. In contrast Naive Bayes and MLP algorithms are falling behind in the curve denoting poor performance.

5.4 Confusion Matrices

Confusion matrices are used to compare the right / wrong estimations of labels "faultless" and "faulty" in fault detection ground truth values.

5.4.1 Logistic Regression Confusion Matrices

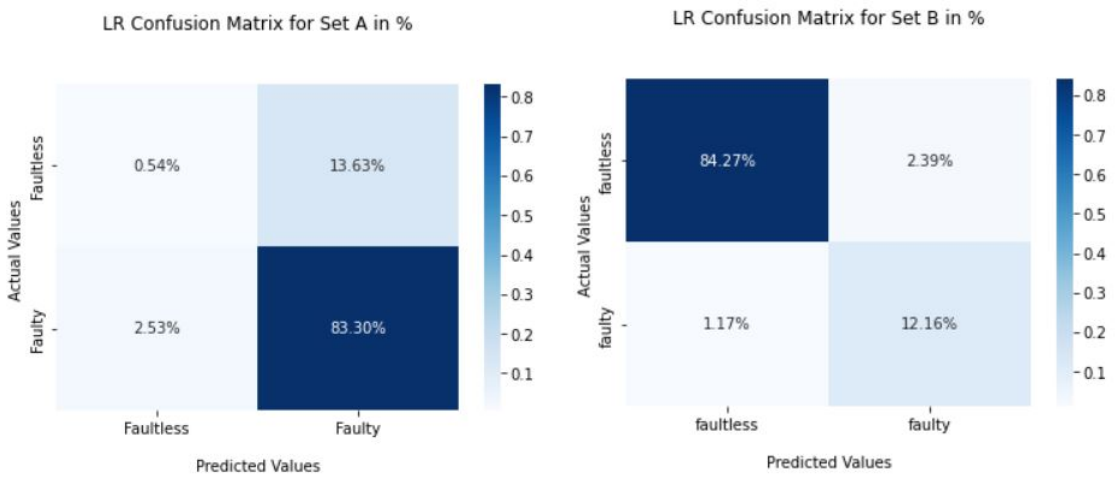

Figure 5.3: Confusion matrices of Logistic Regression (a) SET-A (b) SET-B

The confusion diagram obtained for the SET-A using the Logistic Regression (LR) classifier is shown in Figure 5.3(a). The LR model could only predict approximately 0.54% of the faultless values and 83.3% of the faulty values, resulting in a testing accuracy estimate of 83.7 %. Because the dataset was uneven, we calculated the recall and found an estimated value of 98.37 %. The confusion matrix for the LR model when training is performed on the SET-B is shown in Figure 5.3(b). In this case, around 12.16% of the faulty data and 84.27% of the faultless data were accurately predicted. The calculated recall was around 91.20%.

5.4.2 KNeighborsClassifier Confusion Matrices

When trained and evaluated on the SET-A Simulated dataset, the KNN model estimated that 83.49 % of the data is faulty and 8.94% of them are faultless, with a 98.82% accuracy rate, resulting in a recall score of around 96.82%, as shown in 5.4.(a). On the SET-B dataset we can see in 5.4.(b), it has 86.53% faultless labels and 13.12% fault labels, with 99.65% accuracy and recall score of 98.38%.

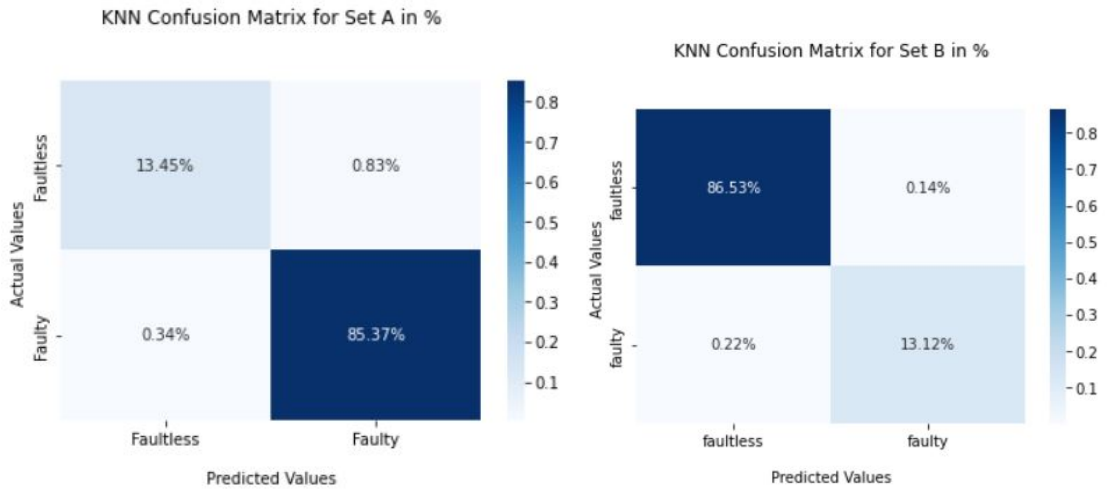

Figure 5.4: Confusion matrices of KNeighborsClassifier (a) SET-A (b) SET-B

5.4.3 Naive Bayes Classifier Confusion Matrices

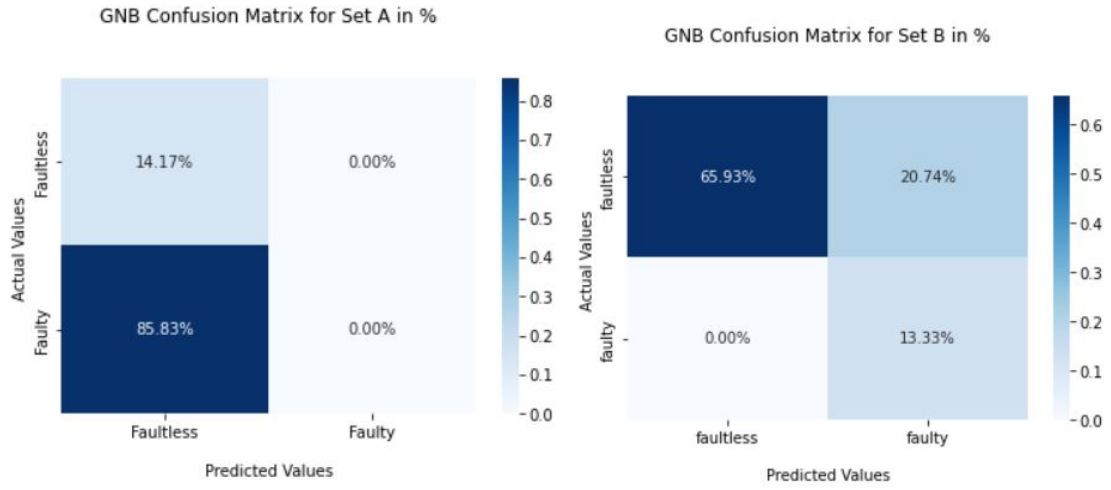

Figure 5.5: Confusion matrices of Naive Bayes (a) SET-A (b) SET-B

On the SET-A dataset, Naive Bayes performed the worst, and it likewise did poorly on the SET-B dataset. For 5.5(a), we can observe that 14.17% True Positive and 85.83% False Positive which leads to only 14.17% accuracy. SET-B, on the other hand, has 65.93 % TP and 13.33 % TN. As a result, we were able to achieve higher accuracy than SET-A.

5.4.4 Support Vector Classifier Confusion Matrices

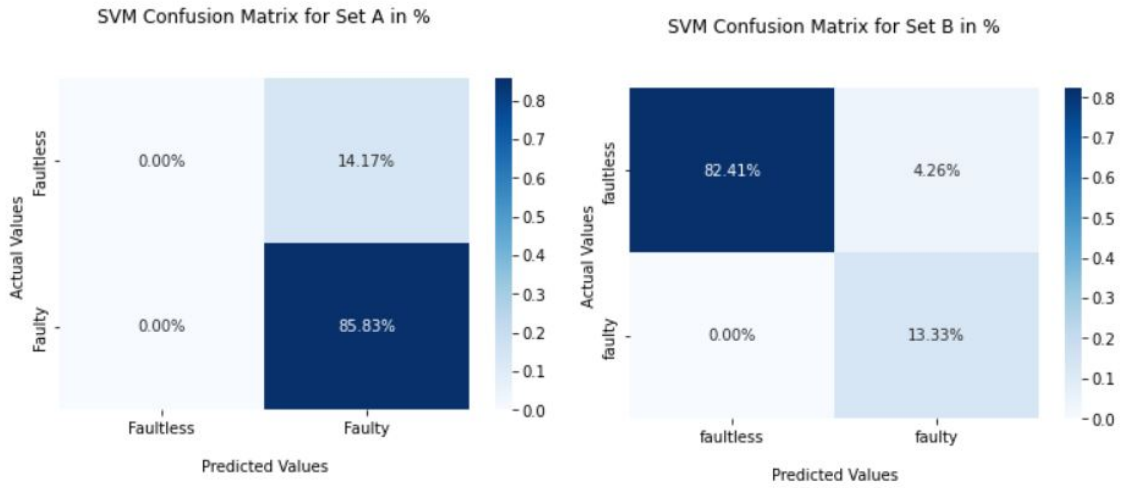

Figure 5.6: Confusion matrices of Support Vector Classifier (a) SET-A (b) SET-B

The SVC model's confusion matrices for SET-A and SET-B are exhibited in Figures 5.6(a) and 5.6(b).. When is was trained and evaluated on the SET-A dataset, displaying FN of 14.17 % and TN of 85.83 %, Therefore it could predict with an accuracy of 85.79% and recall of 100%. The SET-B on the other hand, has a FN of 4.26% and a TN of 13.33%, Moreover There is TP of 82.41%. As a result, it has a prediction accuracy of 87.22%, which is higher than SET-A.

5.4.5 RandomForestClassifier Confusion Matrices

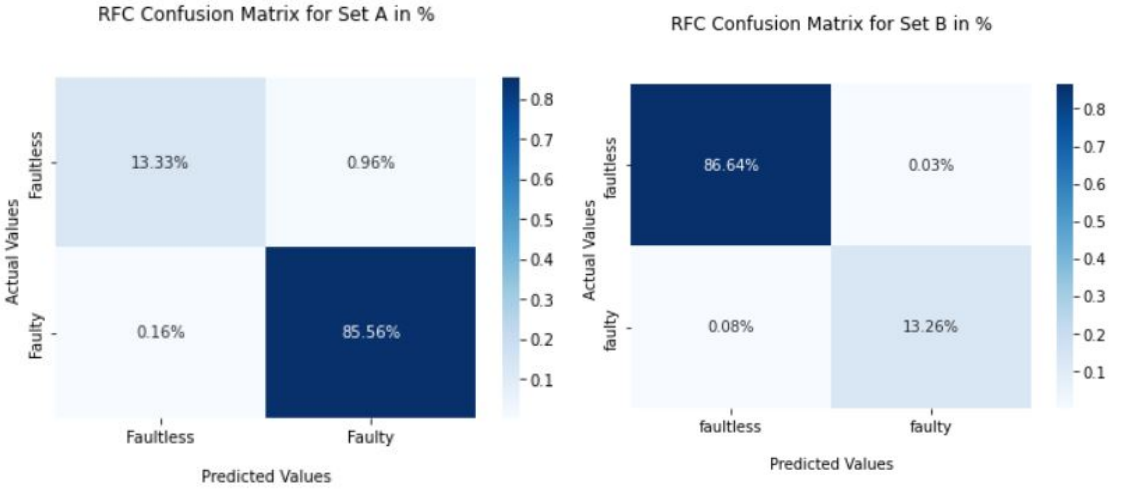

Figure 5.7: Confusion matrices of RandomForestClassifier (a) SET-A (b) SET-B

Figure 5.7(a) shows the comparisons for Random Forest Classifier in SET-A. The Confusion matrix indicated that RFC discovered 13.33 % True Positive values and 85.56% True Negative values. It had a False Positive rate of 0.16 % and a False Negative rate of 0.96 %. As a result The model provides an accuracy of 98.89%

and recall of 99.82%. In the case of experimental SET-B RFC predicted 86.64% TP, 13.26 TN, 0.08%FP and 0.03% FN. Thus the model came up with the highest accuracy of 99.91 %.

5.4.6 DecisionTreeClassifier Confusion Matrices

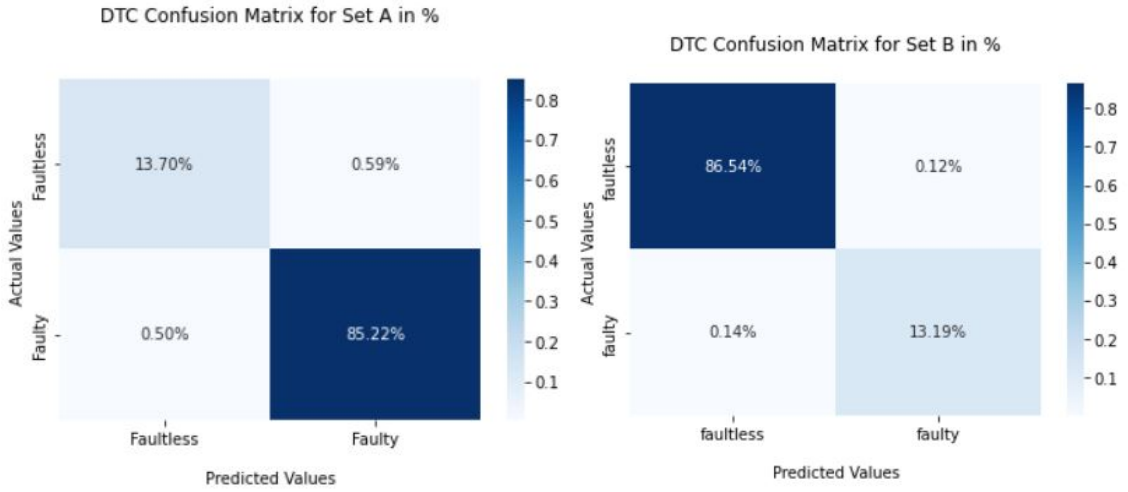

Figure 5.8: Confusion matrices of DecisionTreeClassifier (a) SET-A (b) SET-B

The SET-A confusion matrix employing the DecisionTreeClassifier is presented in Figure 5.8(a). Approximately 85.22 % of the faulty values were anticipated, while 13.70% of the faultless values were detected. False Positive and False Negative are respectively 0.50 % and 0.59 %. As a consequence, We received a 93.07% accuracy score, a 95.97 % recall, and a 95.96 % F1-score. On the other hand SET-B has 86.54% TP and 13.19% FP. Which results in 99.74% accurate prediction, Recall and f1-score were 98.96% and 99.02%, respectively.

5.4.7 MLP Classifier Confusion Matrices

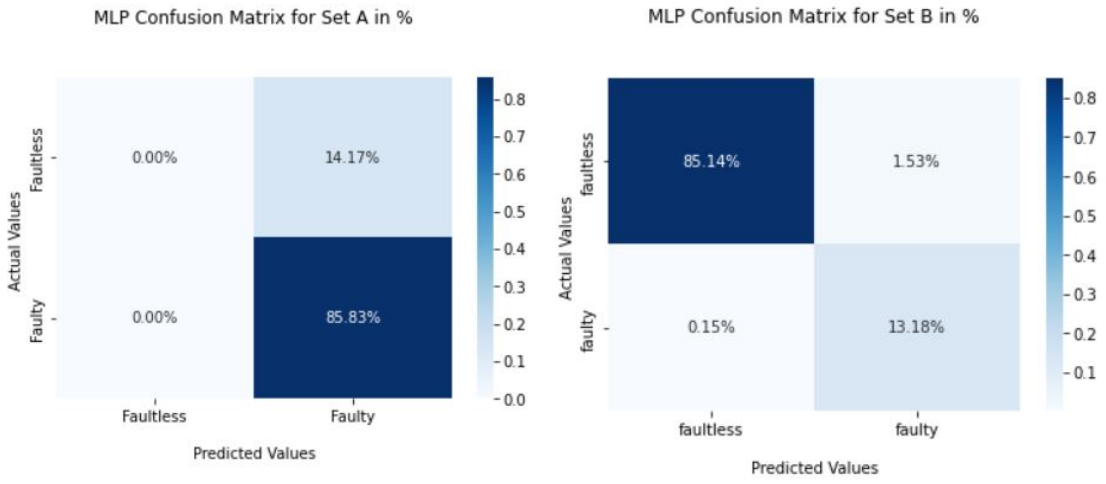

Figure 5.9: Confusion matrices of MLP Classifier (a) SET-A (b) SET-B

From the confusion matrix of 5.7(a) we can see that in SET-A the MLP Classification is showing us 14.17% faultless labels of data and 85.83% of faulty label data with accuracy of 85.83%, recall of 100% and f1-score of 92.38%. On the contrary, in 5.7(b) it is focused that in SET-B the method is showing 85.14% of the data are faultless and 13.18% of of the data are faulty. Where the accuracy score is 95.05%, recall score is 99.31% and f1-score is 84.24% which is better than SET-A.

5.4.8 Extra Tree Classifier Confusion Matrices

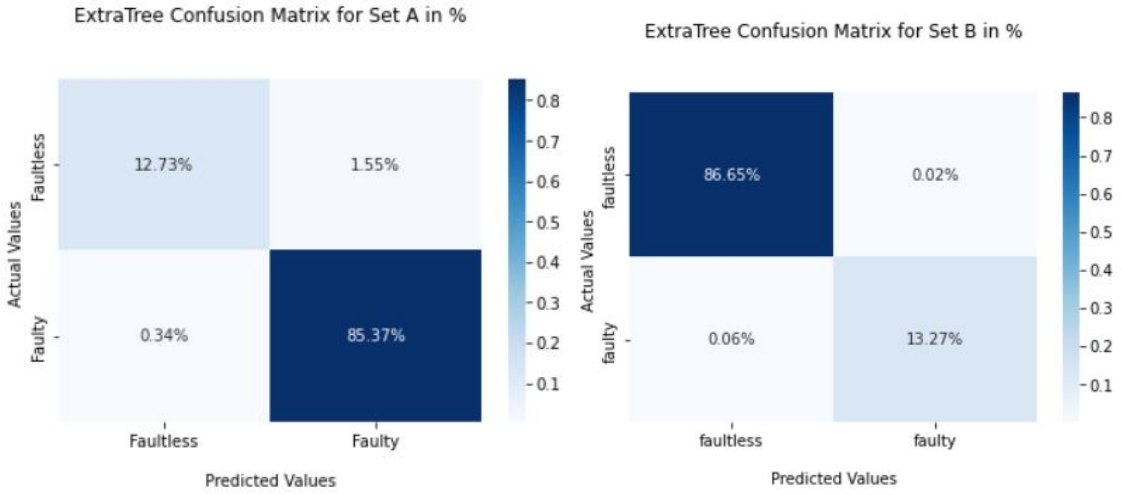

Figure 5.10: Confusion matrices of ExtraTreeClassifier (a) SET-A (b) SET-B

This confusion matrix visualizes from 5.10(a) that in SET-A the ExtraTree classifier. It suggests that 13.73% of the data are faultless and 85.37% of them are faulty data with an accuracy of 98.10%, recall of 99.60% and f1-score of 98.90%. Hence, from 5.10(b), in terms of SET-B we can observe that the method is giving us 85.65% faultless labels and 13.27% of faulty labels of data. Therefore, It has gained an accuracy score of 99.91%, where the recall is 99.54% and f1-score is 99.65% which is also better than SET-A.

5.5 Comparison among the models and datasets

In this section, we compare the results of modes for both SET-A (simulated data of) and SET-B (experimental data). We analyzed accuracy, precision, recall, specificity, and F1-score to assess our model's performance.We were able to determine that K-Nearest Neighbors, Random Forest, Decision Tree, and Extra Trees are most convenient in terms of accuracy rates and all other Data Analysis Metrics.

Figure 5.11 and table 5.2 demonstrate that Naive Bayes provides just 14.17% accuracy for the SET-A. The primary reason for this is because the dataset contains a large amount of category data. We noticed that several of the columns are made up entirely of zeros and ones (binary values). We see, SVC also couldn't perform well due to the size of the dataset (over two hundred thousand records). The outliers in the dataset are to account for the low performance of MLP. Moreover, due to

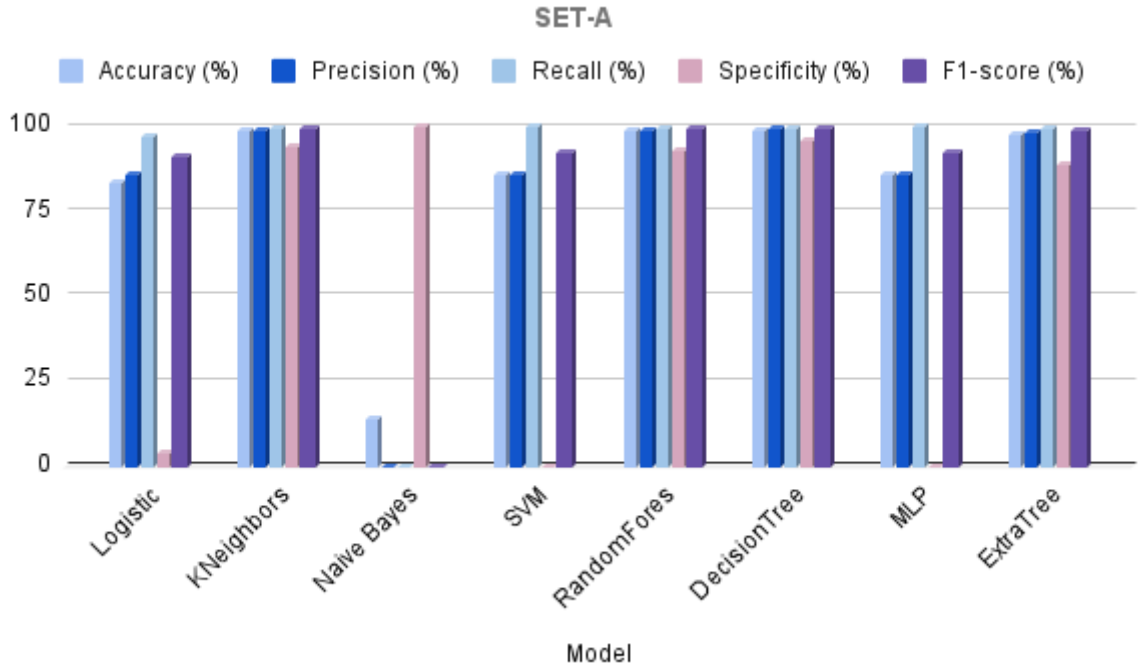

Figure 5.11: Column chart comparison of the models in SET-A

Model	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**Specificity** (%)	**F1-score** (%)
LR	83.84	85.94	97.05	3.81	91.16
KNN	98.82	99.03	99.60	94.16	99.32
GNB	14.17	0	0	100	0
SVC	85.83	85.83	100	0	92.38
RFC	98.89	98.89	99.82	93.30	99.35
DTC	98.91	99.31	99.42	95.87	99.37
MLP	85.83	85.83	100.00	0.00	92.38
ExtraTree	98.10	98.21	99.60	89.13	98.90

Table 5.2: SET-A Results

the imbalance in the dataset, the performance of Logistic regression was not up to the mark. Apart from the aforementioned algorithms, KNeighbors and tree-based methods DTC, RFC, and ExtraTree scored remarkably well.

The results of the SET-B experimental data are shown in Fig. 5.12 and Table 5.3. Despite the fact that the dataset contains a considerable amount of category data, Naive Bayes 79.26% accuracy (lowest among all). We see, SVM also performed better in this case due to the smaller size of the dataset. The performance of

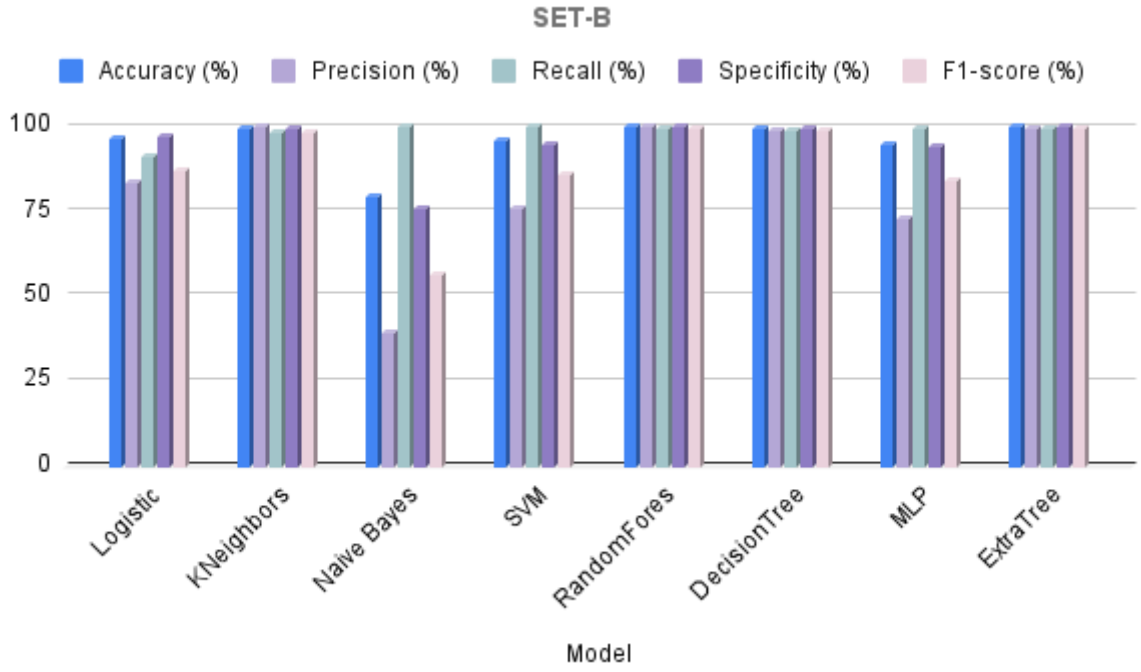

Figure 5.12: Column chart comparison of the models in SET-B

MLP also improved in the second dataset. Similarly, the performance of Logistic regression was better. For this lesser set of data, KNeighbors performed admirably. Beside the fact of using experimental data The tree-based methods DTC, RFC, and ExtraTree scored remarkably well and came up nealy 100% accurate. Because of their adaptability to anomalous distribution and their ability to represent any type of data, whether numerical or categorical.

Model	**Accuracy** (%)	**Precision** (%)	**Recall** (%)	**Specificity** (%)	**F1-score** (%)
LR	96.44	83.56	91.2	97.24	87.22
KNN	99.65	99.95	98.38	99.84	98.67
GNB	79.26	39.13	100	76.07	56.25
SVC	95.74	75.79	100.00	95.09	86.23
RFC	99.91	99.88	99.42	99.98	99.65
DTC	99.74	99.07	98.96	99.86	99.02
MLP	95.05	73.15	99.31	94.39	84.24
ExtraTree	99.91	99.77	99.54	99.96	99.96

Table 5.3: SET-B Results

Chapter 6

Conclusion and Future Research

6.1 Conclusion

Our research's initial objective was to develop a system that would make life easier in smart buildings or infrastructures by ensuring early detection of any system or device malfunctions and allowing for predictive maintenance. To do that we researched on IoT ecosystem and machine learning algorithms of fault detection and diagnostics. Precisely we worked on a couple of building fault detection datasets and predicted faulty data. Hence, we implemented eight different algorithms on the two datasets (SET-A and SET-B) as a model to detect and predict faulty data in any central IoT-based building management system. We did preprocessing, scaling and resizing the data where required. We worked with widely used as well as new algorithms among them K-Nearest Neighbors, Random Forest, Decision Tree, Extra Tree came out very useful with accuracy upto 99.91%, We have seen SET-B was more predictable which was from experimental data. Naive Bayes performed poorly with the simulated SET-A data, Support Vector Machine, Logistic Regression and MLP algorithms was average.With the simulated data, we discovered several limitations and set a goal to enhance the performance of algorithms with similar data in the future. We'll also focus on incorporating our model system into real-world building management systems to save energy, time, and cost.

6.2 Future Work

Implementing automated IoT systems in building management systems is becoming a must day by day. With our analyzed report, we can easily now detect the abnormality of any centrally installed device and can attempt to fix the issue within a lesser time. Moreover, in industrial building automation systems, this proposal can bring a diverse and impactful perspective because the devices that are used there are too sensitive and risky to integrate. Finally, we would like to expand the research and increase the chance of predicting more accurate data across format for the safety measurement of any smart building in near future.

References

[1] H. Schneider, "Implementation of a fuzzy concept for supervision and fault detection of robots," vol. 2, pp. 775–780, 1993.

[2] Sauter, Mary, Sirou, and Thieltgen, "Fault diagnosis in systems using fuzzy logic," 883–888 vol.2, 1994. DOI: 10.1109/CCA.1994.381205.

[3] R. Isermann, "On fuzzy logic applications for automatic control, supervision, and fault diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 28, no. 2, pp. 221–235, 1998.

[4] J. Qin and S. Wang, "A fault detection and diagnosis strategy of vav air-conditioning systems for improved energy and control performances," *Energy and buildings*, vol. 37, no. 10, pp. 1035–1048, 2005.

[5] J. Schein and S. T. Bushby, "A hierarchical rule-based fault detection and diagnostic method for hvac systems," *Hvac&r Research*, vol. 12, no. 1, pp. 111–125, 2006.

[6] J. Schein, S. T. Bushby, N. S. Castro, and J. M. House, "A rule-based fault detection method for air handling units," *Energy and buildings*, vol. 38, no. 12, pp. 1485–1492, 2006.

[7] C. Lo, P. Chan, Y.-K. Wong, A. B. Rad, and K. Cheung, "Fuzzy-genetic algorithm for automatic fault detection in hvac systems," *Applied Soft Computing*, vol. 7, no. 2, pp. 554–560, 2007.

[8] G. Xu, "Hvac system study: A data-driven approach," Ph.D. dissertation, The University of Iowa, 2012.

[9] S. Taheri and M. Mammadov, "Learning the naive bayes classifier with optimization models," *International Journal of Applied Mathematics and Computer Science*, vol. 23, no. 4, 2013.

[10] S. Li, L. Da Xu, and S. Zhao, "The internet of things: A survey. information systems frontiers," 2015.

[11] E. Solutions, *Accuracy, Precision, Recall F1 Score: Interpretation of Performance Measures*, Nov. 2016. [Online]. Available: https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/.

[12] M.-W. Huang, C.-W. Chen, W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "Svm and svm ensembles in breast cancer prediction," *PLOS ONE*, vol. 12, e0161501, Jan. 2017. DOI: 10.1371/journal.pone.0161501.

[13] N. Amruthnath and T. Gupta, "A research study on unsupervised machine learning algorithms for early fault detection in predictive maintenance," pp. 355–361, 2018.

[14] R. Gandhi, *Support Vector Machine — Introduction to Machine Learning Algorithms*, Jun. 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

[15] O. Harrison, *Machine Learning Basics with the K-Nearest Neighbors Algorithm*, Sep. 2018. [Online]. Available: https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761.

[16] W. Kim and S. Katipamula, "A review of fault detection and diagnostics methods for building systems," *Science and Technology for the Built Environment*, vol. 24, no. 1, pp. 3–21, 2018.

[17] S. Swaminathan, *Logistic Regression — Detailed Overview*, Mar. 2018. [Online]. Available: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc.

[18] C. Ren and S.-J. Cao, "Development and application of linear ventilation and temperature models for indoor environmental prediction and hvac systems control," *Sustainable Cities and Society*, vol. 51, p. 101 673, 2019.

[19] W. Yu, T. Dillon, F. Mostafa, W. Rahayu, and Y. Liu, "A global manufacturing big data ecosystem for fault detection in predictive maintenance," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 1, pp. 183–192, 2019.

[20] A. Bhandari, *Confusion matrix for machine learning*, en, https://www.analyticsvidhya.com/blog/2020/04/confusion-matrix-machine-learning/, Accessed: 2022-1-20, Apr. 2020.

[21] N. S. Chauhan, *Decision Tree Algorithm, Explained*, 2020. [Online]. Available: https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.

[22] *Confusion Matrix - an overview — ScienceDirect Topics*, 2020. [Online]. Available: https://www.sciencedirect.com/topics/engineering/confusion-matrix.

[23] figshare, *Metadata record for: Building fault detection data to aid diagnostic algorithm creation and performance testing*, Jul. 2020. [Online]. Available: https://springernature.figshare.com/articles/dataset/Metadata_record_for_Building_fault_detection_data_to_aid_diagnostic_algorithm_creation_and_performance_testing/11743074/2.

[24] GeeksforGeeks, *ML — Extra Tree Classifier for Feature Selection*, Jul. 2020. [Online]. Available: https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/.

[25] Global Alliance for Buildings and Construction (GlobalABC), *2020 GLOBAL STATUS REPORT FOR BUILDINGS AND CONSTRUCTION — Globalabc*, Dec. 2020. [Online]. Available: https://globalabc.org/news/launched-2020-global-status-report-buildings-and-construction.

[26] M. S. Mirnaghi and F. Haghighat, "Fault detection and diagnosis of large-scale hvac systems in buildings using data-driven methods: A comprehensive review," *Energy and Buildings*, p. 110 492, 2020.

[27] A. Nair, *A Beginner's Guide To Scikit-Learn's MLPClassifier*, Nov. 2020. [Online]. Available: https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/.

[28] K. Yan, J. Huang, W. Shen, and Z. Ji, "Unsupervised learning for fault detection and diagnosis of air handling units," *Energy and Buildings*, vol. 210, p. 109 689, 2020.

[29] *Machine Learning Random Forest Algorithm - Javatpoint*. [Online]. Available: https://www.javatpoint.com/machine-learning-random-forest-algorithm.

[30] *Naive Bayes Classifier in Machine Learning - Javatpoint*. [Online]. Available: https://www.javatpoint.com/machine-learning-naive-bayes-classifier.