Predictive Maintenance of HVAC System using supervised Machine Learning Algorithms

by

Md. Zubayer Ahmed Fahim
17201106
Tiash Roy
17241002
Mma Rakib
17241003
Shihab Sharar
19241029

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

Department of Computer Science and Engineering
Brac University
September 2021

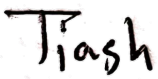
Declaration

It is hereby declared that

1. The thesis submitted is my/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:

Md. Zubayer Ahmed Fahim
17201106



Tiash Roy
17241002

Mma Rakib
17241003

Shihab Sharar
19241029

Approval

The thesis/project titled "Predictive Maintenance of HVAC System using supervised Machine Learning Algorithm" submitted by

1. Md Zubayer Ahmed Fahim (17201106)
2. Tiash Roy (17241002)
3. Mma Rakib (17241003)
4. Shihab Sharar (19241029)

Of Summer, 2021 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on September 25, 2021.

Examining Committee:

Supervisor:
(Member)

Amitabha Chakrabarty, PhD
Associate Professor
Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Dr. Md. Golam Rabiul Alam
Associate Professor
Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD
Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Technological advancements, especially rich data, being incorporated into the development of buildings is simultaneously elevating the importance of fault detection and diagnostics (FDD) of energy-efficient applications (for instance, an HVAC system). Consequently, the challenges lie in identifying the suitable FDD techniques and test methodologies and locating the appropriate dataset. Furthermore, identification and detection of fault during early stages and later stages determine how critical the problem is and what form of immediate maintenance is required (i.e., replacement of components, altering a given condition to restore normalcy, also known as negative feedback etc). This is especially important in shopping malls, educational and business institutions, where the malfunctioning of air conditioners or dehumidifiers can greatly affect productivity. The prediction models revolve around the maintenance of condition-based definitions for the ground truth of the system. At first, the preprocessing technique of normalising the data set was performed on it. Then the data was visualised to help identify categorical and important features influencing the binary values of the fault detection ground truth. Afterwards, the dataset was split into train-test sets with different machine learning algorithms, namely RandomForestClassifier, Decision Tree, KNearestNeighbours, Catboost, MLP Classifier etc. From our experiment, we achieved the best testing accuracies of 99.21% and 100% using the Random Forest classifier on the MZVAV-1 and SZCAV datasets respectively, 95.76% using the CatBoost classifier on the MZVAV-2-2 dataset, 99.91%, 100% and 99.71% using the Extra Trees classifier on the MZVAV-2-1 dataset, SZCAV and SZVAV datasets respectively.

Keywords: HVAC system, fault, machine learning, predictive maintenance

Acknowledgement

Firstly, all praise to the almighty God for whom our thesis have been completed without any major interruption.
Secondly, to our supervisor Amitabha Chakrabarty, PhD sir for his kind support and advice in our work. He helped us whenever we needed help.
And finally to our parents without their throughout support it may not be possible. With their kind support and prayer, we are now on the verge of our graduation.

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

AUC Area Under Curve

DT Decision Tree

FDD Automatic Fault Detection and Diagnosis

HVAC Heating, Ventilation and Air Conditioning

KNN K-Nearest Neighbours

LBNL Lawrence Berkeley National Laboratory

MLP Multi Layer Perceptron

MZVAV Multi Zone Variable Air Volume

ROC Receiver Operating Characteristic

SGD Stochastic Gradient Descent

SMOTE Synthetic minority oversampling technique

SVC Support Vector Clustering

SVM Support Vector Machine

SZCAV Single Zone Constant Air Volume

SZVAV Single Zone Variable Air Volume

TP True Positive

VAV Variable Air Volume

Chapter 1

Introduction

1.1 Background

Existing approaches to monitor and design cost-effective and energy-efficient systems have been evolving for decades, and the journey continues today as new technologies are introduced on a daily basis. Most of the projection-based reports estimate a significant amount of energy consumption from buildings and are anticipated to increase in the coming years, which indicates a very ominous future for upcoming generations. Approximately 40% of primary energy is used by buildings globally and account for 33% of direct and indirect carbon emissions from fuel combustion[20]. Studies reveal almost 30% of energy consumption can be reduced or completely eliminated with protocols to ensure proper detection, analysis and maintenance of existing monitoring systems[9]. This is where an HVAC system is much more efficient. Heating Ventilation and Air Conditioning (HVAC), allows more individual control over the system, much smaller equipment units are required so that space does not become an issue, and provides the potential for building conditioning in the case of a single unit failure. It provides a passive cooling strategy that utilizes high thermal mass with night ventilation, which is needed in a climate that is hot. Imagine you are in a supermarket shopping for upcoming festivities among hoards of people, and the ventilation system fails or malfunctions. As a result, you are forced to leave the premises, while the concerned authorities rally the maintenance team.

1.2 Motivation

Traditional cooling systems are more costly than HVAC systems[20]. Allocating resources for emergency and instant maintenance teams is not feasible for the long term, given traffic conditions in Bangladesh, proper financing, job loyalty etc and other factors. Most importantly, manually locating the root of the failure or fault of a ventilation system is time-consuming, and subsequently figuring out which course of action to take to mitigate the problem involves a number of core decisions to

be made. Also, humans are prone to error more than machines. Prior research has been conducted using an Artificial Immune Recognition System[5]. A machine, or more specifically a closely monitored system appropriately trained, monitored by people from either a remote location or on-site, with proper training, usually can effectively predict an outcome and carry out measures to prevent unfortunate system failures or outages. Proper training refers to both training of the system and people having very basic knowledge of how sensors work, and which factors - or readings from multiple sensors, usually comparable in a range to detect anomalous values - will directly affect the performance of the aforementioned system in the foreseeable future. As a result, preventive procedures can be set in motion long before the fault poses to be a catastrophic issue, such as causing a disruption in day-to-day activities, costing various institutions a substantial loss in resources. A semi-supervised approach has also been explored [11].

1.3 Hypothesis

An automated approach, via the help of an HVAC system, to address this issue has become quite necessary. While initial installation and execution will require hefty funding, future annual or even monthly maintenance costs will gradually decrease. This is because the prediction and classification of the fault will be performed by the HVAC system. An example of cost reduction is comparatively less manpower will be required during scheduled maintenance visits. Our research is on a rudimentary level, further cost analysis regarding contracts and installation, which complex machinery to deploy, which organisation is specialised in manufacturing replacement parts for faulty components etc., warrants further exploration. This paper aims to utilise certain machine learning algorithms to perform automated FDD on operational data to predict, identify and locate anomalies in various defined conditions of an HVAC system. Stated condition details are explained in the section 3. Both popular but old and recently developed algorithms have been utilised to compare and contrast the performances of different models. The models have also undergone pre-processing techniques, especially several trials with changes in parameters to extract the best possible training and testing accuracies for the classification of faulty and not faulty scenarios during result analysis. As a result, sensor readings can be altered, or thorough examination can be carried out to revert things to normal by checking if maintenance should be carried out at a convenient time. This HVAC system consists of a single-duct AHU-VAV system, where one AHU is connected with five VAV boxes - serving four exterior zones and one interior zone per floor of a building. Further information is explained using tabular formats in the section 3.

1.4 Thesis Outline

The thesis has been branched into a number of sections, with some sections having subsections. We begin by discussing prior research carried out on HVAC systems in the section 2, how smart cities and growing population demand smart technology

be utilised, how such a system is an efficient usage of energy as the world progresses to depending on renewable sources of energy etc. Then we move on to explaining the contents of the datasets we use in the section 3, mainly how each dataset was generated in which conditions and regions, how each dataset was similar and unique etc. Consequently, we talk about the prediction models we used in the chapter 4, a bit about how the machine learning algorithms work, some have diagrams to aid the description, mathematical equations used by the algorithms during computation etc. This leads to analysing the results of our research on faulty and not faulty outcomes, which algorithms performed better on which dataset etc. Hence, we provide a summary of our thesis in the section 6, what challenges were faced, what future research can be conducted for improvement etc.

Chapter 2

Related Work

The main objective of this paper is to execute Model Predictive Control (MPC) with the help of architecture developed by IoT for an HVAC system. As all of the sensors, subsystems and integrated controls are going to be linked with the internet, the use of a control unit can be used to process the HVAC system. The testing area was done at a campus building in Italy. The use of various actuators, a database server, a gateway and others are used to implement this project. The end-users will now be able to use the data to fit the means. The development of a model which shows how the system will optimize and provide comfort for the end-users which includes many HVAC modules, records sent to a gateway by the readings taken by the sensors of the environment, monitoring or the weather by an API, an IP device for connection to the end-user and many more. The readings collected by the sensors are sent to the database server and then with the help of the MPC, the implementation of what to be done to provide the best-energy saving mechanism can be gathered. The control algorithm touch is the predicted mean vote used for understanding the condition in which the group of people are in and then allowing to take a uniform thermal comfort for all. The experiment was conducted for four months. The system was always kept for conditions that are considered to be cold. The same time was kept for 2 minutes. The doors and other means of airflow were kept closed. The final results showed that this experiment kept the energy consumption low by 15-20% from previous results. However, this paper did not consider the cost of the system itself and still there are more scope that can be implemented by the help of MPC[18].

The paper first highlights how the faults and energy efficiency decrease throughout the lifetime of an HVAC system. This can be overcome by properly maintaining the system. Automated fault detection and diagnosis (FDD) can help us to understand when the fault may take place and if it is necessary for maintenance. The use of temperature, humidity and pressure is required to conduct this paper. There are various types of faults associated with the HVAC system which includes the fouling of the condenser and evaporator, the leaks in the compressor valve, the overcharge of the refrigerant and so on. The advantage of using embedded FDD is that it continuously monitors the system and thus helps to maintain it properly. Software developed specifically for sending feedback to a data server where all the results and faults are stored. The development of an individual system can also be done

with its very own parameters and faults. The advantage of using the FDD system is that there is no need for someone to be onsite. The system will help to save approximately 20% of energy[4].

The goal of this paper is to manage energy costs, to make sure the budgets of operation are kept intact, keeping maintenance costs to the minimum and to prevent any blocking of factory manufacturing. Technologies like Machine learning (ML), statistical techniques and data mining are used. The dataset used for this paper are from a sensor network which is implemented on buildings within 30 years of development. There were in total 8000 records of temperature for this experiment. The dataset was then divided into two different categories, one is for keeping the records for actual data and the other for keeping the normal and abnormal temperature. The data were divided for both training the algorithm of matching learning, as well as to test it. Both logistic regression and decision trees were used for the accuracy check. The training set used 70% of the total data provided and the rest 30% was used for testing. The results found have a precision percentage of only 64.2% and an accuracy of 66.7% [26].

The main objective of this paper is to help an HVAC system installed in trains. They were able to do this by coming up with a hybrid model-based approach. At first, the paper talked about sustainability with its importance, data analysis and types of technical approaches. MATLAB R2019b was used in order for them to come up with their model. Furthermore, the paper explains how the sensors in the physics-based model are allocated and how they are processed. The HVAC system was described thoroughly how it acts as a system of systems. The fault model described in this paper included compressors, carbon dioxide concentration sensors and many more. The synthetic data received from a previous physics-based model was used which was processed and then examined to come up with types of failures. The data (with new fault code assigned to them) that was later obtained was sent for training, validation and testing. Many features, including peak value ,shape factor and so on, were worked out by their model. While conducting the research many assumptions were made which excluded some sensors. The accuracy obtained on the validation set by Boosted trees was 87.8%. Adaboost was used to assist the classifier. 92.593% was the testing accuracy[24] .

Chapter 3

Dataset Details

Five datasets were used for our model. They were generated by the Building Technology and Urban Systems Division, Lawrence Berkeley National Laboratory. The datasets have two modes, occupied mode and unoccupied mode. The occupancy is determined by if any person is inside the building. Some datasets are experimental in type and the others are simulated. Again, based on air volume the datasets can be divided into two parts.

1. Constant air volume. Their dataset name ends with 'CAV'.
2. Variable air volume. Their dataset name ends with 'VAV'

From the table below we can see if a dataset is experimental or simulated.

Dataset Name	**Creation Method**
Simulated multi-zone variable air volume AHU data set(MZVAV-1)	Simulation
Experimental multi-zone variable air volume AHU data set(MZVAV-2-1.csv)	Experimental
Simulated multi-zone variable air volume AHU data set(MZVAV-2-2.csv)	Simulation
Experimental single-zone constant air volume AHU(SZCAV)	Experimental
Single zone variable air volume(SZVAV)	Experimental

Table 3.1: Dataset Collection Method

The aim of the datasets is to determine the Fault ground truth.We can get an idea about the frequency distribution about the 'Fault Ground Truth' column from the pie chart below.

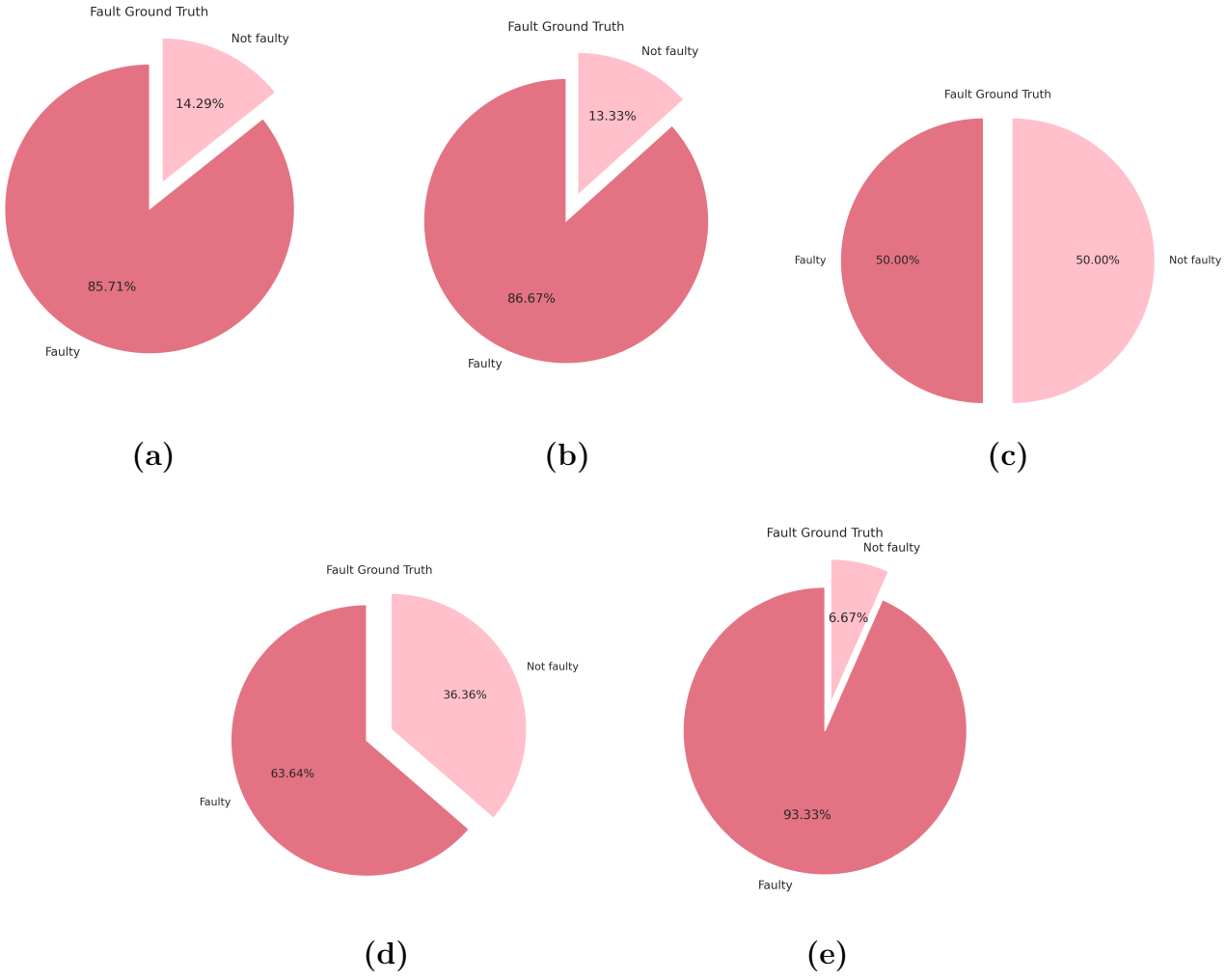


Figure 3.1: Fault Ground Truth Distribution (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

From 3.1(c) we can stipulate that MZVAV-2-2 is a perfectly balanced dataset. Whereas, 3.1(a), 3.1(b) and 3.1(e) suggests that MZVAV-1, MZVAV-2-1 and SZ-CAV are very imbalanced. 3.1(d) suggests that SZVAV is moderately balanced.

MZVAV-1 is developed using a co-simulation framework - building envelope model (using EnergyPlus) and building HVAC system model (using Dymola). The building has three floors. The air handling unit is located on the second floor. The data points were recorded at one minute intervals. Only outdoor air temperature sensor bias fault was injected into the data points.[20]. The outdoor sensors provided faulty measurement in this dataset.On the other hand, MZVAV-2-1 and MZVAV-2-1 designed based on the test facility at the energy resource station. It has three air handling units(AHU). AHU-1 is for the common area of the building and the remaining two is for the test systems. These two are identical to each other. Various

types of faults were injected into the data points into various components of the model. The types of fault are leaking valve of heating coil, stuck valve of heating coil, stuck valve of cooling coil and stuck OA damper[20]. This leak causes from the AC vibrations. The vibration causes the aluminium fins to rub the coppers and create holes on it. The last two datasets, SZCAV and SZVAV, were created in LBNL's Flexlab's test facility. Various types of faulted scenarios were created for the datasets. The types of fault are stuck and leaking OA damper, stuck and leaking valve of heating coil and lastly the same type of faults for the valve of cooling coil[20]. The job of OA damper is to prevent rain waters to enter the system. Leak on that damper causes outdoor rain water to enter the system.

3.1 Data pre-processing

Some of the datasets were made based on a simulation and some datasets were made from live environmental data. Understandable, the datasets were quite different from each other. In order to run the algorithm smoothly we had to pre-process the data. Our aim is to predict the "Fault ground truth".

In the MZVAV-1, MZVAV-2.2 and SZCAV dataset almost 85%-93% data were faulty or labeled 1. It was creating a bias while predicting. To avoid that bias we used the SMOTE(Synthetic minority oversampling technique) method to oversample the undersampled class. It generates synthetic data for the minority class by using interpolation. It will generate such a number of instances so that the ratio of both the classes are 1:1. At first, a random positive class instance is generated and KNN for that instance is obtained. To generate further instances the distance between the feature vector and its neighbour is calculated. Later it multiplies the distance vector with any random number in range of (0,1] and added to the previous feature vector [1]. We applied SMOTE on our training data only.

In the SZVAV and SZCAV datasets there were some NaN values and errored values. We handled them by dropping the rows containing those unwanted values. Some columns had the data type object. For the sake of our algorithm they were transformed into float64 data type. We also dropped the Timestamp column as it was irrelevant for our model. After being done with all the preprocessing we split the data into 75% training and 25% testing data. We used validation split for initializing catboost classifier with 80% training and 20% testing data samples. Subsequently, the training data samples were further divided into 75% training and 25% validation samples.

In all the datasets the values were in different ranges. That's why we normalized the data using the min-max scaler. Min-max scaler transforms all the values into [0,1] range. It increased the integrity and accuracy of our model.

Chapter 4

Prediction models

4.1 Decision tree

A decision tree introduced in [1] is a top-down greedy search approach with recursive partitioning. This algorithm has a wide area of usage for classification and regression tasks in machine learning. For predicting the presence of a fault in the HVAC system, a decision tree classifier can be a wise choice since it considers all possible outcomes and reaches a conclusion. The goal of the decision tree algorithm is to learn simple decision rules based on the given training data. For decision trees, we need to use some criteria for selecting different nodes. The most important steps in building a model as explained in [6] are splitting, stopping, and pruning. In our model, we used entropy for selecting attributes suitable for a particular level. The equation for calculating entropy is given as follows:

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{4.1}$$

For calculating the entropy of a group of attributes

$$E(T, X) = \sum_{c \in X} P(c)E(c) \tag{4.2}$$

The value of entropy calculated from equation 4.1 and 4.2 plays a vital role in decision tree classifier. The higher the entropy, the harder it is to further split the data based on a decision node. A value of $p_i = 0.5$ will result in a maximum entropy blocking the chance of reaching a conclusion. Information gain is a measure of how well the attributes can separate training sets in accordance with the target classification task. The attribute returning the highest information gain with the lowest entropy is prioritized first in the construction of a decision tree. Information gain in equation 4.3 is the difference between the entropy before split and average entropy after the splitting of dataset based on a given attribute.

$$\text{Information gain}(T, X) = \text{Entropy}(T) - \text{Entropy}(T, X) \tag{4.3}$$

The main advantage of using a decision tree classifier is that it does not require extensive pre-processing as it can work with missing values. Furthermore, a decision tree does not require data normalization and scaling.The general architecture of a decision tree is illustrated in figure 4.1.

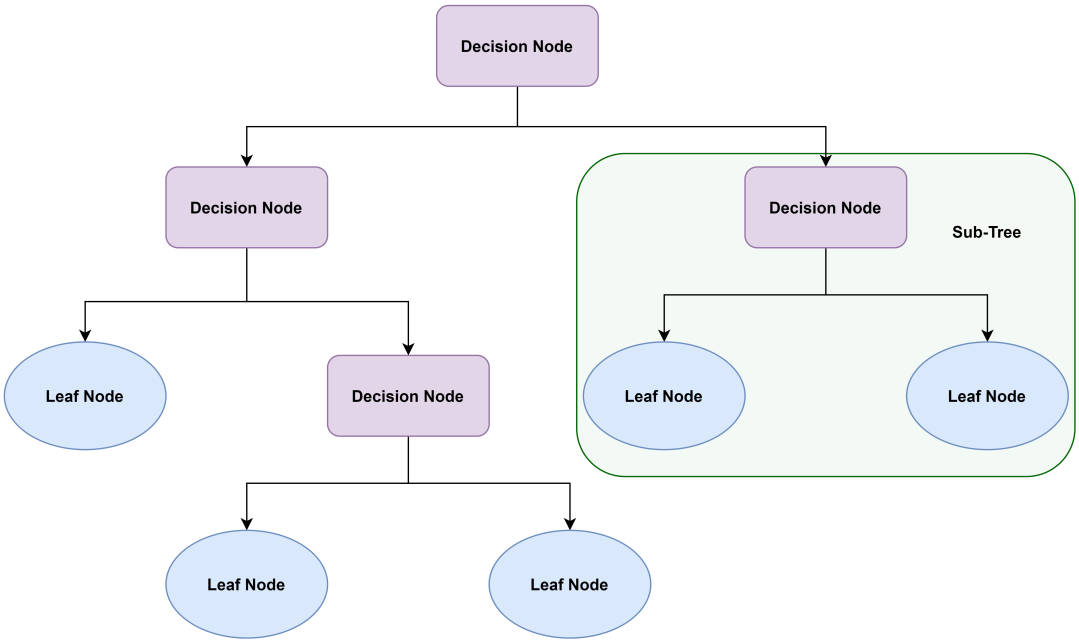


Figure 4.1: Decision Tree Classifier Architecture

4.2 Random Forest

Random forest was introduced in 2001 by L.Breiman[2]. A random forest is an ensemble learning method that fits a number of individual decision tree classifier on various splits and uses voting system for model's prediction. Since our datasets have only two classes in the target variable each decision tree in a random forest will predict either 0 (Not faulty) or 1 (Faulty). The class with the maximum votes from the output of the decision trees will be the prediction of our model. Random forest is a powerful algorithm which intends to improve predictive accuracy and helps in controlling over-fitting. The general structure of the working principle of a random forest classifier is illustrated in the figure 4.2.

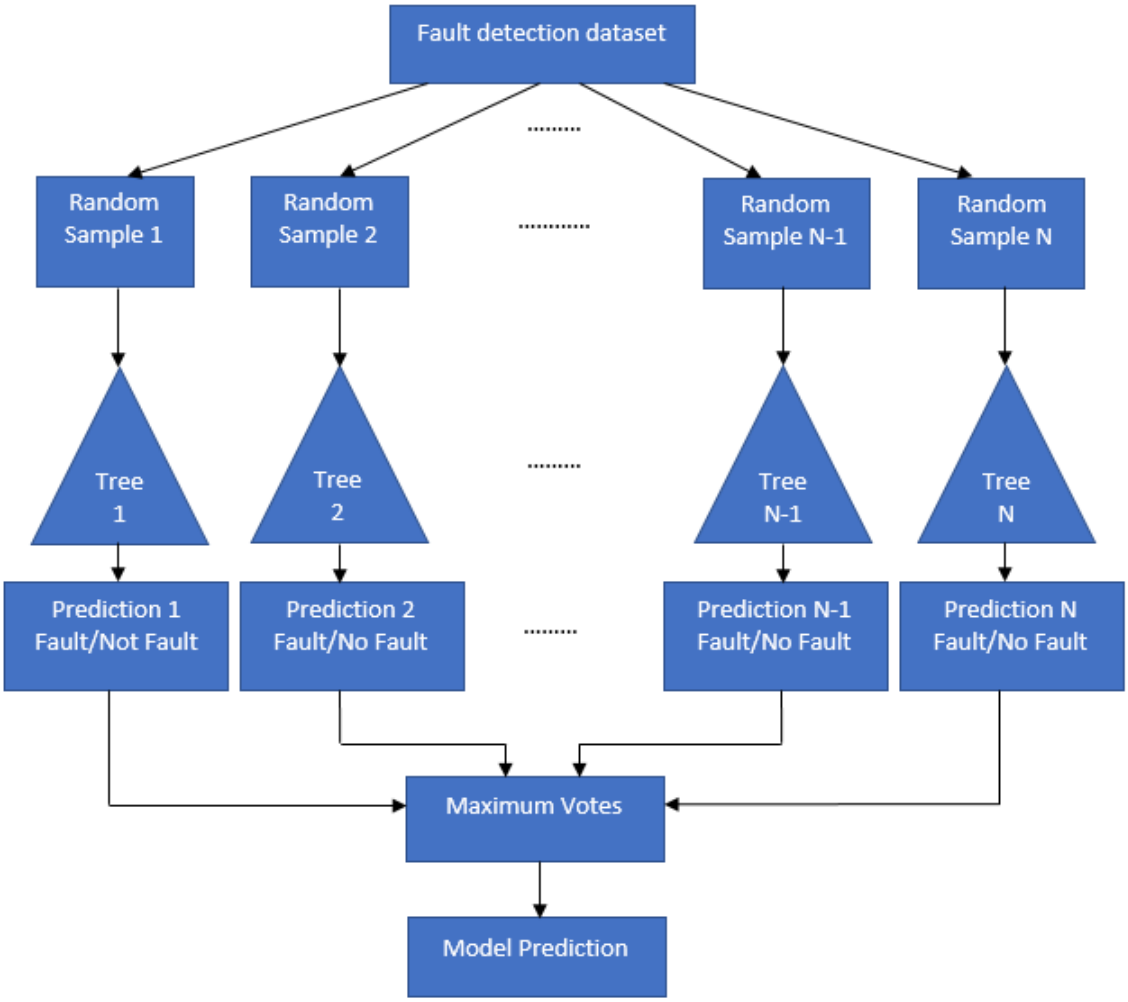


Figure 4.2: Random Forest Classifier Architecture

The dataset is at first divided into N random samples and passed through individual decision tree classifier. Predictions from the decision trees are then counted and the class with maximum votes are selected as the model's prediction output.

.

4.3 Extra Tree Classifier

Extremely Randomized Trees classifier (Extra Trees classifier) [3] is an ensemble tree-based technique for classification and regression task. It splits a tree node randomly selecting both attributes and cut points. Extra trees classifier is very similar to random forest classifier except the construction of the decision trees in the forests. Each tree is delivered with k features randomly. The model selects the best attribute/feature on which to split the data based on some mathematical criteria. Information gain can be used as a decision criterion. Decision tree inside

an extra trees classifier the model works with different features and generates an information gain. Total information gain is calculated afterwards for each attribute or feature. Based on the highest information gain an attribute can be considered to be the most important feature for reaching to a conclusion. The equations for calculating entropy and information gain are given in 4.1, 4.2 and 4.3. Extra trees classifier produced better accuracy for the prediction of credit scoring among all the ensemble classifiers [23]. Therefore, extra tree might be a good option for the HVAC datasets in predicting faults.

4.4 KNN

One of the popular non-parametric, lazy and supervised learning, instance-based algorithms is K-Nearest-Neighbors (KNN) [16]. The goal of KNN is to determine if a data point is in group one or two. The points are divided by determining the states of the points that are near to a particular point. In this algorithm, a model is not constructed. It is an algorithm that postpones the decision to generalize beyond the training examples till a new query is encountered. A sample of data is chosen and checked; if most of the points are in group one, then the data-point given in the question lies in group one rather than group two. The groups in question refer to the prediction of a certain outcome involving specific factors. Hence, KNN can be used for thermal comfort prediction activities [22].

The Minkowski equation for the distance metric parameter can be computed as follows:

$$(\sum_{i=1}^{n} |X_i - Y_i|^p)^{1/p} \tag{4.4}$$

When the value of p in equation 4.4 is set to one, the distance calculated is said to be the Manhattan distance. Increment the value of p by one (to two), the Euclidean distance can be determined. For values of p less than 1, the triangle inequality will not be satisfied hence the formula is rendered invalid.

Our prediction model utilises the KNeighborsClassifier in the scikit-learn package. The parameters (number of neighbours, distance metric and algorithm) was varied and a number of trials were carried out to compare accuracies obtained. Other parameters (were set to default) such as the weight function for prediction was kept uniform. Most datasets used had a class imbalance, which was handled using oversampling to reduce bias from the imbalance-learn package.

4.5 SVM

SVM (Support Vector Machine) is a supervised learning algorithm that tries to separate data points in the best possible way based on classes with a hyperplane. The main advantage of SVM is, it tends to perform well on high dimensional low amounts of data. When machine learning is considered, SVM performs efficiently since it is called the maximum margin hyperplane. SVM models can be used for linear and non-linear classification tasks, as well as regression. Also, these models are not sensitive to the feature scales so we don't need to scale the inputs, since there would be a scale for all the features for a comparable range.

What makes SVM so famous is the kernel trick [14]. It is a method of calculating the dot product of two products x and y in a highly dimensional feature space. Hence, the kernel functions are sometimes called "generalized dot product". The equation given in 4.5 and 4.6 can be used to aid the previously mentioned description [14]:

$$\text{maximise } \alpha \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K\left(X_i^T \cdot X_j\right) \tag{4.5}$$

$$\text{s.t } 0 \leq \alpha_i \leq C \text{ for all } i = 1, 2, \ldots, n \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0 \tag{4.6}$$

s.t = subjected to, C = value to establish optimization for preventing misclassification of training example.

There are 3 types of kernels: linear, polynomial and radial basis function (RBF)/ Gaussian kernels.

The linear and polynomial kernel [14] can be represented as:

$$K\left(X_1, X_2\right) = \left(a + X_1^T X_2\right)^b \tag{4.7}$$

b = degree of kernel, a = constant term and X space being 2 -dimensional

As the equation 4.7 suggests, the dot product is computed by increasing the power of the kernel.

RBF kernel is designed in a manner that its function will yield a value depending on the distance from a designated coordinate, an example can be the origin. Thus, it can be written as:

$$\begin{gathered} K\left(X_1 - X_2\right) = \text{exponent}\left(-\gamma \left\|\right| X_1 - X_2\right\|^2\right) \\ \|\text{X1} - \text{X2}\| = \text{ Euclidean distance between X1\&X2} \end{gathered} \tag{4.8}$$

The dot product (similarity) of X1 and X2 in equation 4.8 is calculated using the Euclidean distance in the original space [14].

In this thesis, the SVM models chosen had parameters with kernels set to linear, polynomial (degree 3) and RBF, with different values of C for each model and gamma (for the RBF parametric model). After training, oversampling was done to reduce bias. Several trials indicated the RBF model performed far better than the linear and polynomial SVM models. Hence, the results based below are on the RBF model (with parameters of C having a value of 1 and gamma having a value of 2).

4.6 AdaBoost

AdaBoost stands for "Adaptive Boosting" which is now one of the most useful classifiers out there. To know about adaboosting we must first know what boosting is. Boosting redefines many classifiers with low performance by combining their behavior to come up with a classifier that gives us much better performance [10].

Adabooster adjusts the weight required for weak classifiers to give us better output. It follows a repetitive method in which the weak classifiers are assigned with higher weights and the strong ones with lower weights. Keeping in mind that at the start , weights assigned to the classifiers are the same [10].

There are many parameters that are required to come up with a proper adabooster, and with a combination of SVC base estimator, we are able to reach a higher accuracy rate. However , in our model we did not use the SVC base estimator as we were able to reach high accuracy.

4.7 CatBoost

CatBoost stands for "Category" and "Boosting". As previously mentioned what boosting is, here we will only discuss what and how this type of boosting takes place. By using the concept of decision trees, catBoosting is developed from there on wards. It is specifically designed for gradient boosting [19].CatBoosting is expected to give output quicker and is capable of giving better results as it uses GPU implementation [21]. CatBoost is well known for its tackling ability against overfitting. It does this by choosing a tree structure based on its calculation of leaf values [12].

4.8 XGBoost

It is much better than other old algorithms in terms of speed and accuracy. XGBoost is based on gradient boosted decision trees. [17]. Gradient boosting is a technique that involves creating new models that forecast the residuals or errors of previous models, which are then combined to form the final prediction. Gradient boosting gets its name from the fact that it uses a gradient descent approach to minimize

loss when adding new models[8]. It can be implemented for both regression and classification problems. We used it for classification problems in our model.

XGBoost supports parallelization. Unlike other algorithms which run on CPU serially it can harness the power of multicore CPUs. This enables it to train very large datasets faster. It has a wide range of parameters. The used parameters for our model are:

1. Learning rate. It determines how quickly a model adapts to the problem. It ranges between [0,1].
2. Max depth. It determines the depth of the decision tree.
3. Objective. It determines the problem type. We used 'binary:logistic' because ours is a classification problem.
4. Colsample by tree. It determines the percentage of features used per tree.
5. N estimators. It determines the number of trees.
6. And lastly, alpha is the regularization parameter.

We used alpha as regularization parameter for all the datasets.

4.9 SGD

Stochastic gradient descent (SGD) can be thought of with the help of a function that has two variables. The function's inputs would be x and y coordinates on a two-dimensional map. The function's output would be the height of the land on the map at each point. A person who is walking on the land has the ability to make a choice and go in any direction. This function's "gradient" is the direction with the steepest slope. Gradient descent refers to following the gradient downhill. The reason for doing this is to locate the inputs that cause the function to output the lowest possible value. The term "stochastic" is defined to be random. So stochastic gradient descent is referring to a process that leads downhill and involves randomness.

The function which takes in the model and the data and measures how wrong - or how far away the predictions are from the actual result - is called the loss function. In the SGD classifier model used in this thesis, the loss parameter is set to hinge. Hinge loss is often used for models like SVM. By using the loss function and some calculus, SGD will tweak the parameters to produce a more accurate model. It will perform this operation until no more improvement can be made, hence optimising the model.

Optimising on a standard loss function may cause us to overfit the training data. Perfectly optimising on the data the model learns from may cause over-fitting of the training data. Perfectly optimising on the data the model learns from may cause it

to perform worse on more general data it has not seen before. The training data might just be noisy and learning from that noise might be counterproductive. These added terms to loss functions are often termed as regularization.

LASSO (or l1) penalty adds the sum of the absolute values of the parameters that are estimated to the loss. This means that if something does not really reduce the loss, the model will want its parameter to be zero. This can help introduce sparsity in the model - it will force useless information the model does not need out of the model. This thesis utilises the Ridge (or l2) penalty, which is a form of regularization. It adds the square of the parameters and will reduce the size of the given parameters. Parameters associated with uninformative features will usually not be forced to be exactly zero, but something very small.

In this thesis, all the SGD classifier models had their parameters set to hinge loss and the l2 penalty. SGD is widely used for optimisation techniques. Comfort detection utilising real time data for online learning approach was applied using SGD [7]. Controlled and efficient usage of energy was implemented using SGD in large buildings[13].

4.10 MLP classifier

MLP Classifier uses multilayer perceptron by propagation. It uses an underlying neural network to perform the classification. It usually has an input layer, multiple hidden layers and an output layer. Each neuron has its corresponding weight. During each iteration it adjusts the weight of the neuron in order to minimize the error. This weight update rule is based on the gradient descent method [15]. It continues epoch by epoch until it matches the stopping criteria. As it is a back propagation method, it has two steps:

1. Forward pass: runs the neural network and compute the error for each neuron at the output layer.
2. Backward pass: it starts at the output layer and passes the error neuron by neuron recursively calculating the local gradient of each neuron

The main advantage of this method is that higher accuracy can be achieved with a small amount of data. It also works for complex non linear problems. On the other hand, the main disadvantage is that it is not foreseeable that up to which extent the independent variable is affected by the dependent variable.

$$y = \varphi\left(\sum_{i=1}^{n} w_i x_i + b\right) = \varphi\left(\mathbf{w}^T\mathbf{x} + b\right) \tag{4.9}$$

Here in equation 4.9, W = Vector weights, X = vector of inputs, b = bias and φ = activation function

Multilayer Perceptron, which is mainly used in supervised learning problems, takes the approach of the perceptron to another level as it is made up of at least two perceptrons. Like a normal perceptron, it takes multiple or single inputs and gives us an output. The difference here is that the perceptron is more advanced as it has more hidden layers in between which generates biases, parameters and weights according to the problem at hand. Errors arise while conducting algorithms like this, and this problem can be fixed using the help of back propagation[27].

4.11 Naive Bayes

This classification technique is based on the Bayes theorem. The main assumption of this algorithm is that a particular feature of the model is completely unrelated to other features of that model[25]. It is a simple algorithm but very effective on large datasets. The biggest setback of the algorithm is that it is completely based on independent parameters. But in real life it is almost impossible to get independent parameters. As ours is a classification problem we used the Gaussian Naive Bayes.

$$P(c|x) = P(x|c)P(c)/P(x) \tag{4.10}$$

The notations given in equation 4.10 are given below,

1. $P(c|x)$ is the posterior probability of class (c, target) given predictor (x, attributes).
2. $P(c)$ is the prior probability of class.
3. $P(x|c)$ is the likelihood which is the probability of the predictor given the class.
4. $P(x)$ is the prior probability of the predictor.

Chapter 5

Result and analysis

Here, we will discuss our model's performance on the five datasets.

5.1 MZVAV-1 Dataset

Decision tree performed exceptionally well in multi-zone variable air volume (MZVAV-1) HVAC simulation dataset. The maximum depth of the decision tree classifier was chosen to be 50 for ensuring majority of the nodes are expanded till leaf. The quality of data split was measured using 'entropy'. Our model was able to achieve 99.14% accuracy on the testing dataset. Although the dataset MZVAV-1 was unbalanced with 86.67% of the samples belonging to the class "Not faulty" and 13.33% to "faulty" we were able to get rid of biases by implementing oversampling. On the other hand, random forest classifier is expected to perform better than Decision tree classifier since it generates decision tree on random samples of the dataset and uses maximum voting for generating the final output. The testing accuracy increased slightly from decision tree. The model was able to achieve 99.21% accuracy on the testing dataset with a recall of 98.09% and f1-score of 98.36%. Likewise, extra trees classifier is expected to perform better than random forest classifier since it uses the best features to predict the outputs. The decision criterion was selected to be 'gini'. The probability of an individual feature that was misclassified would be calculated using Gini index. In the MZVAV-1 dataset, the maximum number of features to consider was selected to be 5. However, the model's accuracy on the testing dataset is approximately equal to that of the random forest classifier.

For KNN, the parameters were set to k having a value of 7 (k is the number of nearest neighbors), distance metric set to euclidean and the algorithm for computing the nearest neighbors set to "ball_tree", the dataset gives training accuracy of 99.10% and testing accuracy of 97.84%. However, this dataset was the largest, while having an imbalance of ground truth values, leading to a correct prediction of approximately 84% of faulty values and 14% of not faulty values. The recall obtained was 98.37% with an f1-score of 95.80%. Likewise, when trained and tested on the SVM model, the results obtained were a training accuracy of 96.32% and a testing accuracy of

96.18%, which is understandable since SVM works better on a small dataset. On the contrary, Naive Bayes gives us the lowest accuracy on MZVAV-1. The main reason behind that is there is a lot of categorical data in this dataset. Some columns have binary values. That is why it does not have a normal distribution. And, naive bayes can not perform well without normal distribution. That is why it is performing very poorly with testing accuracy of only 14.2%. Lastly, for MLP the accuracy is poor for MZVAV-1. It is at 65.25%. The reason behind the poor performance is the outliers in the dataset. We added one input layer, one output layer and hidden layer size is $(50, 50, 50)$. We used Relu as an activation function. The main advantage of using relu is that it can avoid the vanishing gradient problem. As it is a large dataset we have used adam optimizer. It works very well on large datasets. The maximum iteration was set to 500.

From XGBoost, we get 93.77% testing accuracy whereas the training accuracy is 96.18%. It clearly indicates that the model did not overfit on this algorithm. The maximum depth of the tree was set to 25. Learning rate was 0.2 and the n estimator was set to 10. On the other hand, AdaBoost's performance is very low compared to XGBoost. Because adaboost is very sensitive to outliers, hence the testing accuracy is only 64.79%. Unlike Adaboost, CatBoost shows very high accuracy in this dataset the testing accuracy being at 90.45%. The SGD model performed poorly on this dataset, obtaining training and testing accuracies were 50.50% and 45.28% respectively.

5.2 MZVAV-2-1 Dataset

Decision tree model is showing extraordinary performance in experimental dataset of multi-zone variable air volume (MZVAV-1) HVAC dataset. The model was able to predict from the testing dataset with an over whelming 99.68% accuracy. The macro average recall and f1-score was 99.30% and 99.33% respectively. By the same token, the training accuracy is 100% for random forest in this dataset which is same as the decision tree but on testing dataset random forest classifier performs better with an accuracy of 99.85%. A recall of 99.74% and f1-score of 99.68% was acquired from the dataset. Extra trees classifier was able outperform random forest and vanilla decision tree to achieve an accuracy of 99.91% on the testing dataset with a recall of 99.83% and f1-score of 99.80%.

Using KNN (the parameters were set to k having a value of 7 and distance metric set to euclidean) we get a training accuracy of 99.73% and testing accuracy of 99.25% was obtained. The parameters were set to k having a value of 7 (k is the number of nearest neighbors), distance metric set to euclidean and the algorithm for computing the nearest neighbors set to "ball_tree". The recall and f1-score for this model was 99.28% and 98.46% respectively. Based on the performance we can say that it is one of our best models in this dataset. Among all the other datasets the SVM model performed the best on the this, achieving a training accuracy of 99.88% and a testing accuracy of 99.38%. Conversely, Naive Bayes gives one of the lowest accuracies on MZVAV-2-1. The main reason for that is the data is not normally distributed or the

curve is not a bell curve. We already know that naive Bayes works best on normally distributed data. The testing accuracy is only 64.92%. The testing accuracy with MLP classifier is 98.16% with this dataset using relu activation and adam solver incorporating maximum iteration of 500.

This dataset yielded the second best results for the SGD classifier model with a training accuracy of 92.04% and 88.57% for its testing accuracy. Using XGBoost on this dataset we perceived 99.61% testing accuracy and the training accuracy is 99.91%. They are almost overlapping which further proves this algorithm's efficiency. The max depth of the tree was set to 25. The learning rate was 0.3 and the n-estimator was set to 10. AdaBoost shows exceptional performance in this dataset because it does not have any outliers thus the accuracy is at 99.07%. Likewise, Catboost's accuracy for this dataset is at 99.83%. We can conclude that Adaboost and Catboost's performance are similar in this dataset.

5.3 MZVAV-2-2 Dataset

In this simulated dataset the performance of decision tree model was relatively poor compared to the previous two datasets. However, a maximum depth of 20 showed a better result for this dataset. The testing accuracy achieved with this model was 95.60%. The macro average recall and f1-score was 95.61% and 95.60% respectively. Likewise, the performance of random forest classifier on this MZVAV-2-2 dataset was moderate as it achieved a 93.97% testing accuracy with the same rcall and f1-score on testing data samples. Extra tree outperformed random tree classifier in this dataset but it could not surpass the performance of decision tree. The accuracy of this model on the testing dataset was 94.04% with approximately the same recall and F1-score.

Training and testing the KNN model on the "MZVAV-2-2" dataset, the best training accuracy of 95.73% and testing accuracy of 94.14% was obtained using the parameters Manhattan metric distance with a value of twelve for k(the number of neighbours). The model correctly predicted 46% of not faulty ground truth values and 49% of faulty values. Hence, it can be said the model predicted 95% of the values accurately when tested. Similarly, the SVM model yields a training accuracy of 95.76% and a testing accuracy of 93.73%. Again, as naive bayes is based on the assumption that all the features are unrelated to each other, we had to check each feature's correlation with Y. It was clear from the correlation matrix that with Y most of the features have 0 correlation. That is why naive bayes is suitable for MZVAV-2-2 as the features are independent.The accuracy is also very high which is 83.58%. For MLP we added one input layer, one output layer and the hidden layer size is($50x50x0$). Again, we used Relu as an activation function. Once again Adam solver was used here as tn optimizer. The maximum iteration was set to 500. The testing accuracy is 92.69%.

Implementing the SGD classifier model revealed training accuracy of 86.25% and a testing accuracy of 85.82% on this dataset. For XGBoost we get 94.08% testing accuracy. The training accuracy is 96.98%. The recall and F1-score is also very

high for this algorithm respectively at 94.09% and 94.07%. The max depth of the tree was set to 20. Learning rate was 0.1 and the n estimator was set to 10. The testing accuracy for Adaboost in this dataset is 94.22% and the training accuracy is 94.94%. For Catboost the accurcaies are 95.76% for testing and 96.88% for training data samples.

5.4 SZCAV Dataset

Likewise, in the MZVAV datasets, the decision tree classifier was able to hold on to its supremacy on the single-zone constant air volume dataset (SZCAV). The model with a maximum depth of 20 was able to achieve an accuracy of 99.91% on the testing dataset. The model's performance on this dataset surpassed all other accuracies attained by this model. The macro average recall and f1-score of decision tree classifier in SZCAV dataset was 99.68% and 99.62% respectively. Until now, we have achieved excellent performance on few datasets using random forest and decision tree classifiers. Accuracies for some datasets were close to 100%. However, in the SZCAV dataset, we achieved an accuracy of 100% in both training and testing data samples. This illustrates how well-suited the random forest classifier is for the prediction task using this dataset. Recall and f1-score for this dataset is also 100% for this dataset.In the same manner with random forest classifier, accuracy of extra trees classifier on SZCAV was 100%. The primary reason behind this state-of-the-art performance is that the dataset is imbalanced on testing sample. There are very few 'Not faulty' testing data samples in this dataset relative to 'Faulty' data samples. However, biasness of our model towards 'Faulty' data samples was avoided using oversampling on training data and thus it helped our model to classify all testing samples accurately.

For KNN, training accuracy of 99.93% and testing accuracy of 99.94% was obtained using the parameters set to euclidean for distance metric and algorithm for computing nearest neighbor set to "ball_tree" on the SZCAV dataset. The misclassification rate is less than 0.01. The model correctly predicted approximately 93% of the faulty values and an estimate of 6.5% for the not faulty ones. The SVM model achieved a training accuracy of 99.99% and a testing accuracy of 99.90% on the SZCAV dataset. In Naive Bayes the dataset shows moderate performance for SZCAV the model performs moderately well. Their testing accuracy is 55.53%. On a final note, we can say that among all the algorithms naive bayes performs the worst. That is because our datasets are not normally distributed which is a must for naive bayes to perform well. An accuracy of 96.15% was achieved through MLP classifier on the testing data samples.

A training accuracy of 93.45% and testing accuracy of 91.85% was obtained from this dataset using the SGD classifier model, the best results obtained when compared to the ones from the other datasets. For XGBoost we get 98.84% testing accuracy. The maximum depth of the tree was set to 25. Learning rate was 0.1 and the n-estimator was set to 10. Also, Adaboost shows tremendous performance in this dataset with 97.47% testing accuracy. Catboost performs even better with 99.90% accuracy. It

is the best performance for any algorithm in this dataset. In fact, it is one of the best performances across all five datasets.

5.5 SZVAV Dataset

In SZVAV testing dataset decision tree was able to attain an accuracy of 98.71%. The maximum depth of the decision tree was chosen to be 30. A recall of 98.69% and f1-score of 98.63% was achieved from the output of the model. For SZVAV dataset random forest classifier surpassed the accuracy of the decision tree classifier on testing data samples. The model achieved a testing accuracy of 99.27% with a recall of 99.34% and f-1 score of 99.22%. Therefore, after analyzing the performances of a decision tree classifier and a random forest classifier on all the datasets, we can conclude that a random forest classifier is a better choice for the predictive maintenance of an HVAC system. On the other hand, extra trees classifier was able to predict with an accuracy of 99.71% which is higher than both decision tree and random forest classifier. The recall and F1-score were 99.74% and 99.69% respectively.

Using KNN, the highest training accuracy of 99.38% and testing accuracy of 98.60% for dataset "SZVAV" was yielded using the parameters Manhattan metric distance, a k value of seven and the ball tree algorithm to compute the nearest neighbours. The model correctly predicted 61% of faulty values and 37% of the not faulty values. Hence, approximately 98% of the testing prediction is accurate. The precision and recall both obtained was 98%. The SVM model when trained and tested on the imbalanced SZVAV dataset produced a training accuracy of 99.76% and a testing accuracy of 98.76%. However, The accuracy is quite low for MLP. The training accuracy is 80.64% and the testing accuracy is 78.80%. We added one input layer, one output layer and three hidden layers of size 50. Relu was used as an activation function. The maximum iteration was set to 500. On the other hand, for SZVAV Naive Bayes performs moderately well considering the distribution is not normal. The testing accuracy is 66.99% . The SGD model performed the second most poorly on this dataset, having a training accuracy of 75.55% and a testing accuracy of 73.00%. For XGBoost, we get 97.08% testing accuracy. The max depth of the tree was set to 25. Learning rate was 0.1 and the n estimator was set to 10. Adaboot's testing and training accuracy for this dataset are respectively 91.56% and 89.85%. These number stipulates that the model did not overfit. As usual, Catboost again shows exceptional performance with 99.08% testing accuracy.

5.6 ROC and AUC Curve

ROC and AUC curves are usually used for comparing different algorithms on a single dataset. They are basically a summary of tons of confusion matrices. On the horizontal plane ROC sensitivity is given and on the vertical plane specificity is given.

$$TruePositiveRate = TruePositive/(TruePositive + FalseNegative) \quad (5.1)$$

$$FalsePositiveRate = FalsePositive/(FalsePositive + TrueNegative) \quad (5.2)$$

Initially, a classification problem calculates the probability for being in a specific class. The class is then decided by a distinct threshold. The threshold can be predefined or user defined based on the algorithm. ROC curve plots the True Positive Rate against False Positive Rate for different thresholds.

AUC stands for area under the curve. After plotting the ROC for different thresholds we get the ROC curve by adding those points. AUC is the area under that curve. The higher the AUC the better our algorithm. When the AUC is 1 the classifier is giving 100% accuracy and when AUC is 0 the classifier is predicting at 0% accuracy. AUC will always be inside 0 and 1. Below we can see the RUC curves for our best three and worst two classifications models across our five datasets:

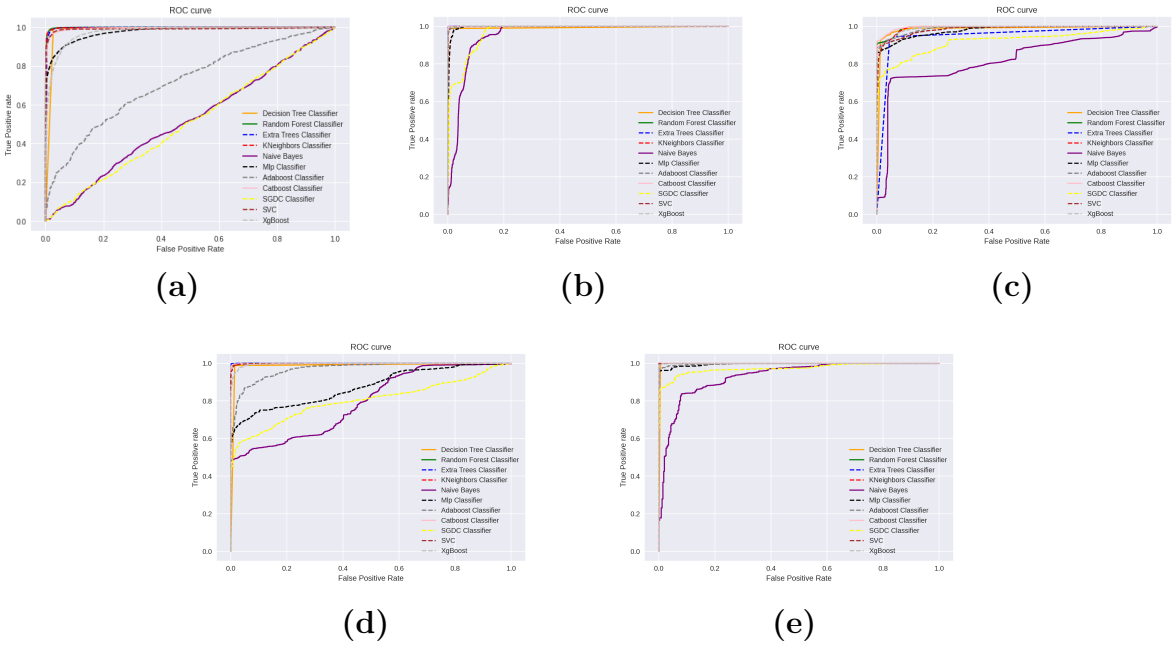


Figure 5.1: ROC Curve (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

For, figure 5.1(a) we can see that the AUC for all the algorithms except for AdaBoost, Naive Bayes and SGDC are close to 1. In fact, Decision tree, random forest, extra trees, SVC and KNN are in perfect symmetry with 1 in y-axis. Thus they are the best performing algorithms for this dataset. On the other hand, for figure 5.1(b) MLP classifier is showing a better performace compared to figure 5.1(a). The tree-based algorithms have AUC perfectly 1 which shows their superiority for this dataset. Moreover, Kneighbors classifier also performed exceptionally well here. However, SGD classifier and Naive bayes do have a decent amount of false positve rate in figure 5.1(b) and they were not able to perform well on prediction with this

dataset. In, figure 5.1(c) Naive Bayes shows the poorest performance compared to the other algorithms followed by SGD classifier. Unlike the above mentioned two datasets here, MLP performs better than the Extra tree classifier. Catboost classifier shows a remarkable performance with this balanced dataset. Except for SGD all the other algorithms are performing well as usual. If we analyze figure 5.1(d) closely we will be able to observe algorithms like Decision Tree, Random Forest, Extra Trees, KNeighbors, catboost, SVC and Xgboost have a pretty low or approximately zero false positive rate with their AUC being approximately 1. On the contrary, algorithms like Adaboost, MLP, SGDC and Naive Bayes were not able to keep up with the others showing their degraded performance. Lastly, for figure 5.1(e), algorithms like Extra trees and Random forest had an AUC score of exactly 1. SGDC and Naive bayes had a poor performance compared to other algorithms in this dataset as well. In addition, from closer observation it seems that false positive rate for MLP and Adaboost are slightly higher than the others.

5.7 Confusion Matrices

For better understanding of our models the confusion matrices of all the algorithms are given below.

Our data consists of one output of two different types. In order for us to showcase how the results have been classified, we will use a confusion matrix. The output can be represented in 4 parts. The concept is the same as 5.1 and 5.2 Here,

1. TN= True negative
2. FP= False positive
3. FN=False negative
4. TP= True positive

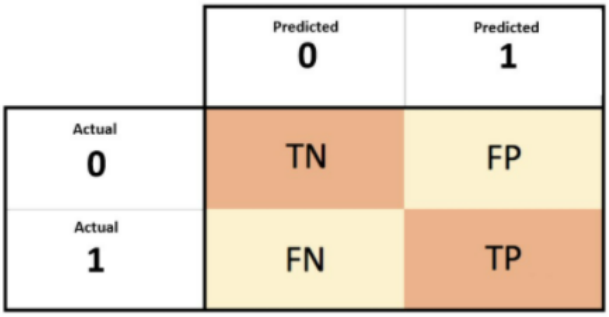


Figure 5.2: Confusion Matrix Architecture

5.7.1 Confusion Matrices of Decision Tree

Confusion matrices on all the datasets are:

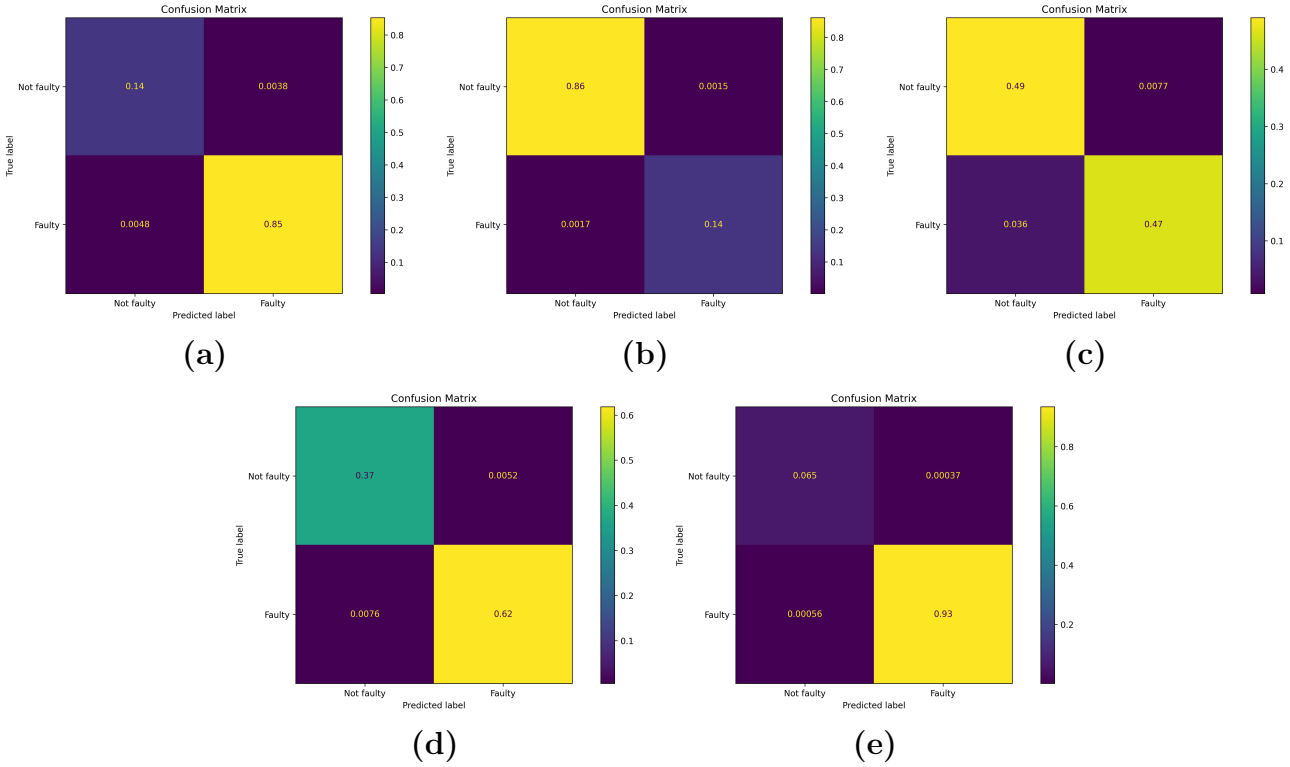


Figure 5.3: Confusion Matrices of Decision Tree (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

The confusion matrices were used to compare the accurate and inaccurate predictions of labels "faulty" and "not faulty" ground truth values. 5.3(a) represents the confusion diagram generated for the MZVAV-1 dataset using the decision tree classifier from the sci-kit learn package. The decision tree (DT) classifier model correctly predicted approximately 14% of the faulty values and approximately 85% of the not faulty values, hence leading to an estimated testing accuracy of 99.99%. Since the dataset is imbalanced, we calculated the recall and found an approximate value of 98.37%. The confusion diagram for the DT model when trained and tested on the imbalanced MZVAV-2-1 dataset can be seen in 5.3(b) . Here, approximately 14% of the faulty values and 86% of the not faulty values were predicted correctly, with inaccurate predictions of about 0.3% for both predicted labels. The recall calculated for 5.3(b) was about 99.30%. 5.3(c) refers to the confusion diagram of the DT model on the balanced dataset (MZVAV-2-2). The DT model classified approximately 47% of the faulty values and 49% of the not faulty values correctly. Again, the incorrect predictions (false positives and negatives) were a combined estimate of 4.37%. The DT model performed especially well on the imbalanced SZVAV and SZCAV datasets, whose confusion diagrams can be observed from 5.3(d) and 5.3(e) respectively. 5.3(d) shows the model correctly predicted approximately 62% of faulty and 37% not faulty labels, and 5.3(e) reveals accurate predictions of 93% of faulty values and 6.5% of not faulty values, showing how imbalanced datasets can create bias.

5.7.2 Confusion Matrices of Random Forest

Confusion matrices on all the datasets are:

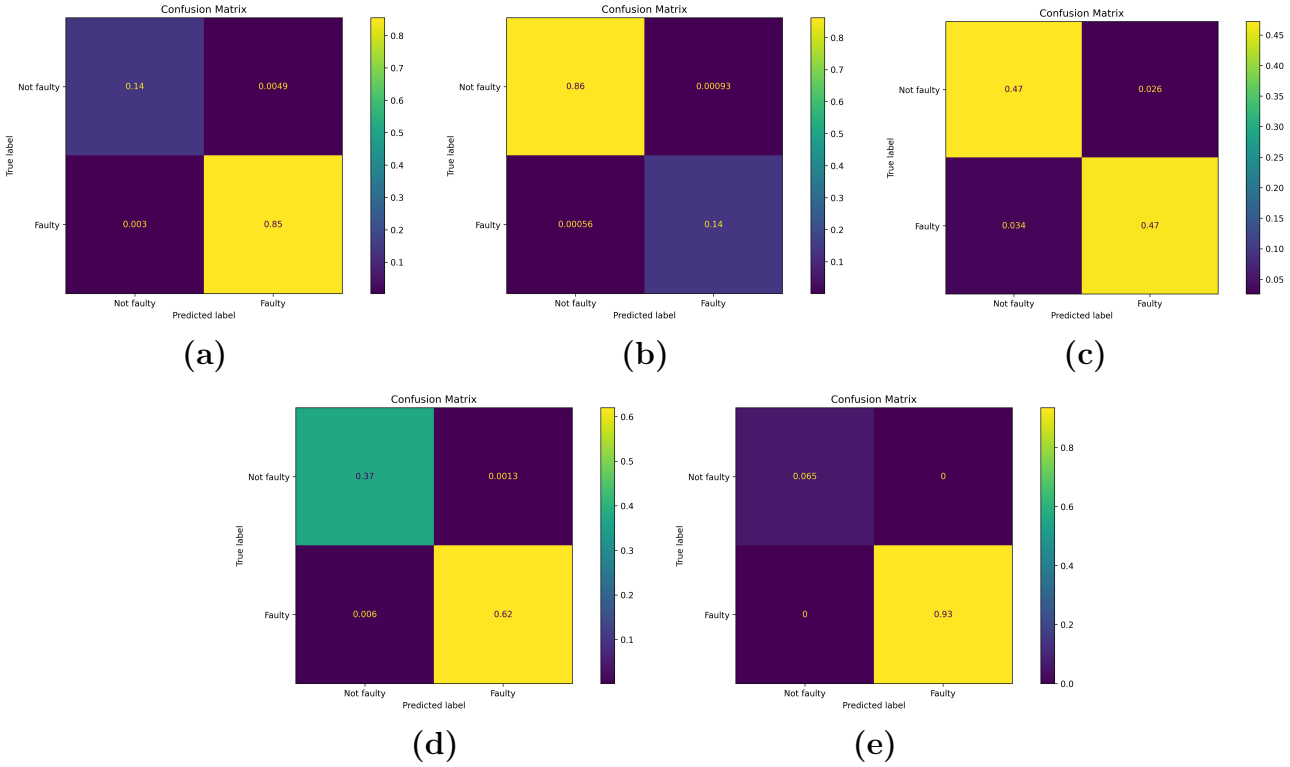


Figure 5.4: Confusion matrices of random forest classifier (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

The confusion matrix for the MZVAV-1 dataset using the Random Forest classifier from the sci-kit learn package is shown in 5.4(a). Approximately 85% of the faulty values and 14% of the not faulty values were predicted accurately, resulting in a recall of about 98.09%, and a misclassification rate of about 0.8%. This model performed well on the MZVAV-2-1 dataset as well, which is evident from the fact seen in the confusion diagram of 5.4(b) (about 86% and 14% of the true labels were predicted accurately, with an inaccurate prediction of about 0.1% of both labels and an estimated recall of 99.74%). Since MZVAV-2-2 was the most balanced dataset compared to the rest, the prediction model yielded much more accurate results, which can be seen in 5.4(c). 5.4(c) shows that the model correctly predicted 47% of both faulty not faulty labels, giving us a recall of about 93.97%. The imbalanced SZVAV yielded correct predictions of approximately 62% for faulty labels and approximately 37% for not faulty labels, a recall score of about 99.34%, and a combined false prediction rate of about 0.73%, which can be seen in 5.4(d). On the other hand, the imbalanced SZCAV displayed similar results in 5.4(e), producing accurate predictions of approximately 93% faulty values and 6.5% unfaulty values, leading to a misclassification rate of 0% and a recall score of 100%.

5.7.3 Confusion matrices of Extra Tree Classifier

Confusion matrices on all the datasets are:

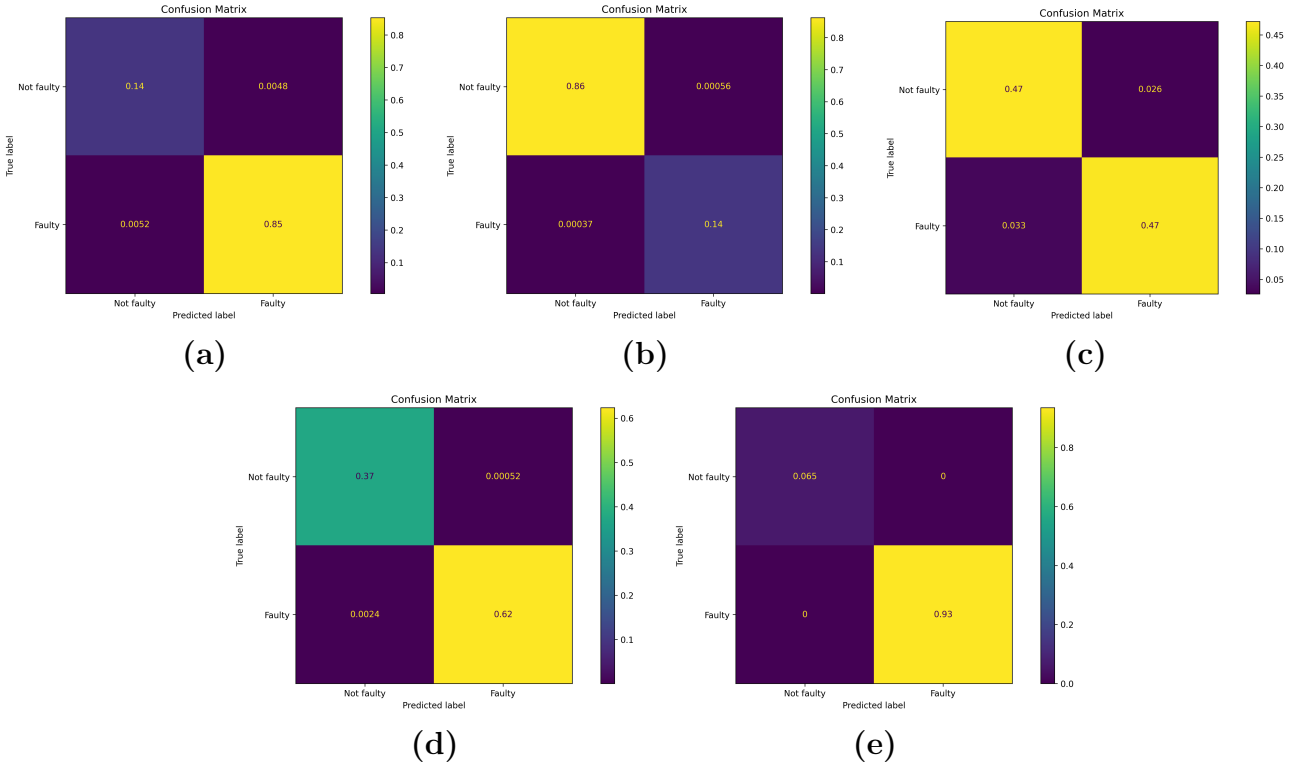


Figure 5.5: Confusion matrices of Extra Tree Classifier (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

The diagrams in 5.5 were used to compare the accurate and inaccurate predictions of labels “faulty” and “not faulty” ground truth values. The Extra Tree classifier model performed exceptionally well on all the datasets. The results can be interpreted from 5.5(a), where the model correctly predicted approximately 85% of the ”faulty” ground truth values and 14% of the ”not faulty” ones from the imbalanced MZVAV-1 dataset, giving a recall score of about 98.01%. When the model was trained and tested on the imbalanced MZVAV-2-1 dataset, approximately 86% and 14% of not faulty and faulty values were predicted correctly, misclassifying about 0.1% of the data, which is seen from 5.5(a). 5.5(c) demonstrates the correct prediction of approximately 47% of both faulty and not faulty labels, with a recall score of about 94.05%, when the model was applied on the balanced MZVAV-2-2. The model revealed a recall score of 99.74% when tested on the imbalanced SZVAV dataset, producing an accurate prediction of 62% faulty values and 37% of not faulty values, a misclassification rate of about 0.3%, which can be seen from 5.5(a). However, the model yielded a perfect 100% score for recall, with accurate predictions of approximately 93% of faulted labels and 6.5% of not faulted labels when tested on the SZCAV dataset, as shown in 5.5(e).

5.7.4 Confusion matrices of KNN

Confusion matrices on all the datasets are:

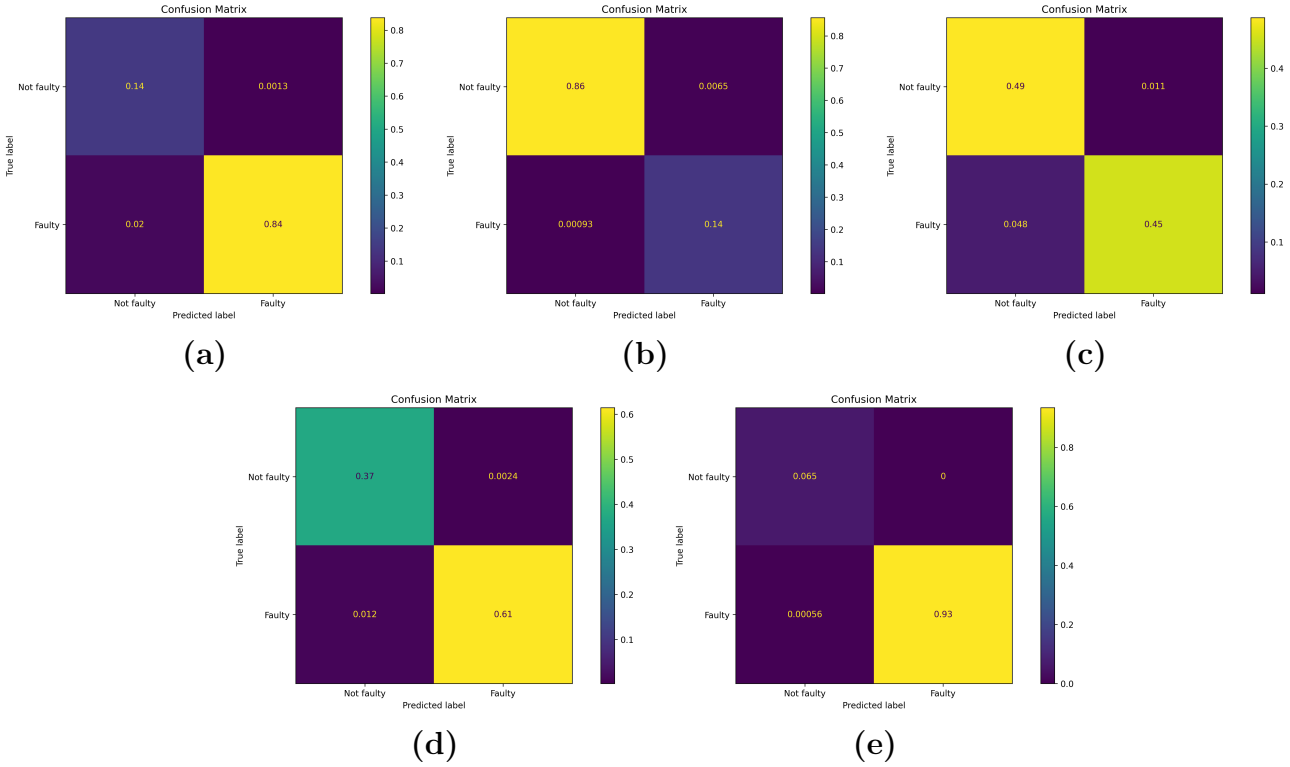


Figure 5.6: Confusion matrices of KNN (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

Parameters were tweaked for a few models of KNN on the different datasets, which is discussed in the "Results and Analysis" section. The KNN model when trained and tested on the imbalanced MZVAV-1 dataset yielded an estimate of 84% of faulty labels and 14% of unfaulty labels, with a misclassification rate of about 2%, leading to a recall score of approximately 98.37%, which is displayed in 5.6(a). 5.6(b) shows the confusion matrix of the model when applied to the imbalanced MZVAV-2-1 dataset, resulting in accurate predictions of approximately 14% of faulty labels and 86% of not faulty labels, a misclassification rate of about 0.74% and a recall score of about 99.28%. The model, when implemented on the balanced MZVAV-2-2 dataset, revealed correct predictions of about 45% for faulty labels and 49% for not faulty labels, a misclassification rate of almost 6%, and a recall score of 94.15%, which can be seen in 5.6(c). 5.6(d) demonstrates the results of the model when tested on the imbalanced SZVAV dataset, which obtained accurate predictions of about 61% faulty labels and 37% not faulty labels, with a misclassification rate of about 1.4%, and a recall score of about 98.76%. Lastly, the confusion matrix of the model when tested on the SZCAV dataset revealed the model's correct prediction of approximately 93% of faulted values and 6.5% of not faulted values, and a misclassification rate of about 0.06%, leading to a recall score of about 99.97%, as shown in 5.6(e).

5.7.5 Confusion matrices of SVM

Confusion matrices on all the datasets are:

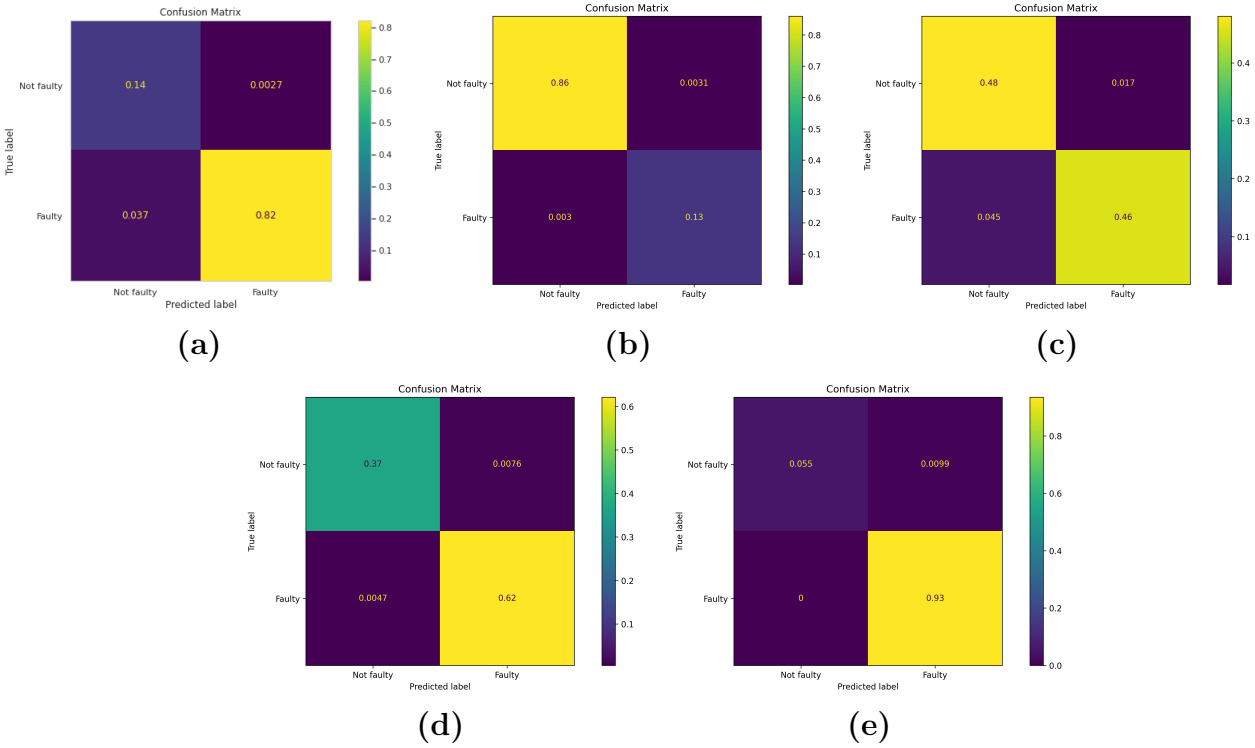


Figure 5.7: Confusion matrices of SVM (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

5.7(a) depicts the confusion matrix obtained when the SVM model was trained and tested on the MZVAV-1 dataset, revealing a correct prediction of approximately 82% of the faulty outcomes and 14% of the not faulty outcomes, with a misclassification rate of about 4%, leading to a recall of about 96.93%. The confusion matrix for the model implemented on the imbalanced MZVAV-2-1 dataset reveals an accurate prediction of approximately 13% of the faulty outcomes and 86% of the not faulty outcomes, with a misclassification rate of about 0.61%, leading to an approximate recall score of about 98.73%, which can be seen in 5.7(b). 5.7(c) shows that the model correctly predicted 46% of the faulty outcomes and 48% of the not faulty outcomes, with a misclassification rate of about 6.2%. 5.7(d) shows that the model correctly predicted 62% of the faulty outcomes and 36% of the not faulty outcomes when tested on the SZVAV dataset, with a combined misclassification rate of about 1.23%, resulting in a recall score of about 98.60%. The model correctly predicted approximately 93% of the faulty outcomes and 5.5% of the not faulty outcomes, with a combined misclassification rate of nearly 1%, resulting in a recall score of about 92.42% when tested on the SZCAV dataset, as shown in 5.7(e).

5.7.6 Confusion matrices of Adaboost

Confusion matrices on all the datasets are:

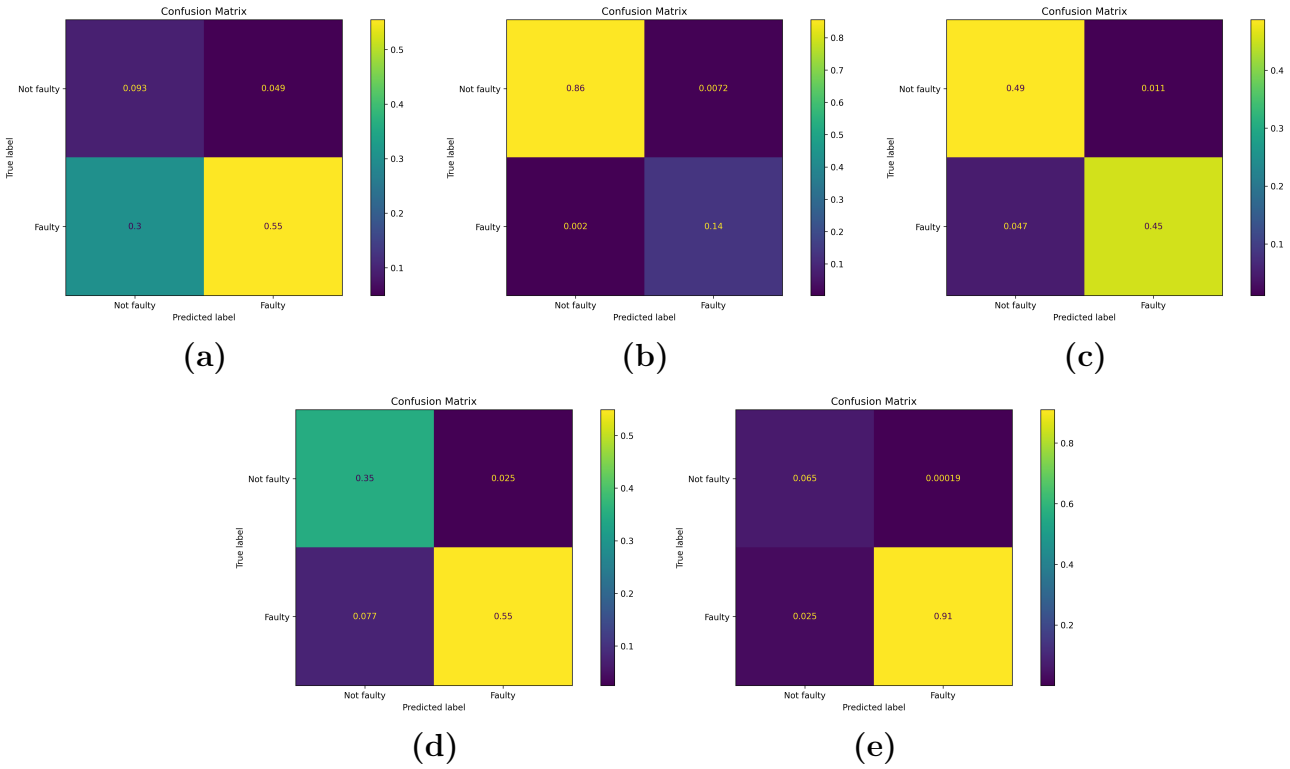


Figure 5.8: Confusion matrices of Adaboost (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

The diagrams were used to compare the accurate and inaccurate predictions of labels “faulty” and “not faulty” ground truth values. The model correctly predicted approximately 55% of the faulty outcomes and 9% of the not faulty outcomes, h when tested on the MZVAV-1 dataset, as shown in 5.8(a). 5.8(b) shows the confusion matrix of the model when tested on the MZVAV-2-1 dataset, which can be interpreted as accurate predictions of approximately 14% of the faulty scenarios and 86% of the not faulty scenarios. The confusion matrix obtained for the implementation of the model on the MZVAV-2-2 dataset can be observed in 5.8(c), which shows the correct prediction of 45% of faulty outcomes and 49% of the not faulty outcomes. 5.8(d) shows the confusion matrix of the model when tested on the SZVAV dataset, which can be interpreted as accurate prediction of approximately 55% of the faulty scenarios and 35% of the not faulty scenarios, with a misclassification rate of almost 9.4%. The confusion matrix obtained for the implementation of the model on the imbalanced SZCAV dataset can be observed in 5.8(e), which shows the correct prediction of 91% of faulty outcomes and 6.5% of the not faulty outcomes.

5.7.7 Confusion matrices of CatBoost

Confusion matrices on all the datasets are:

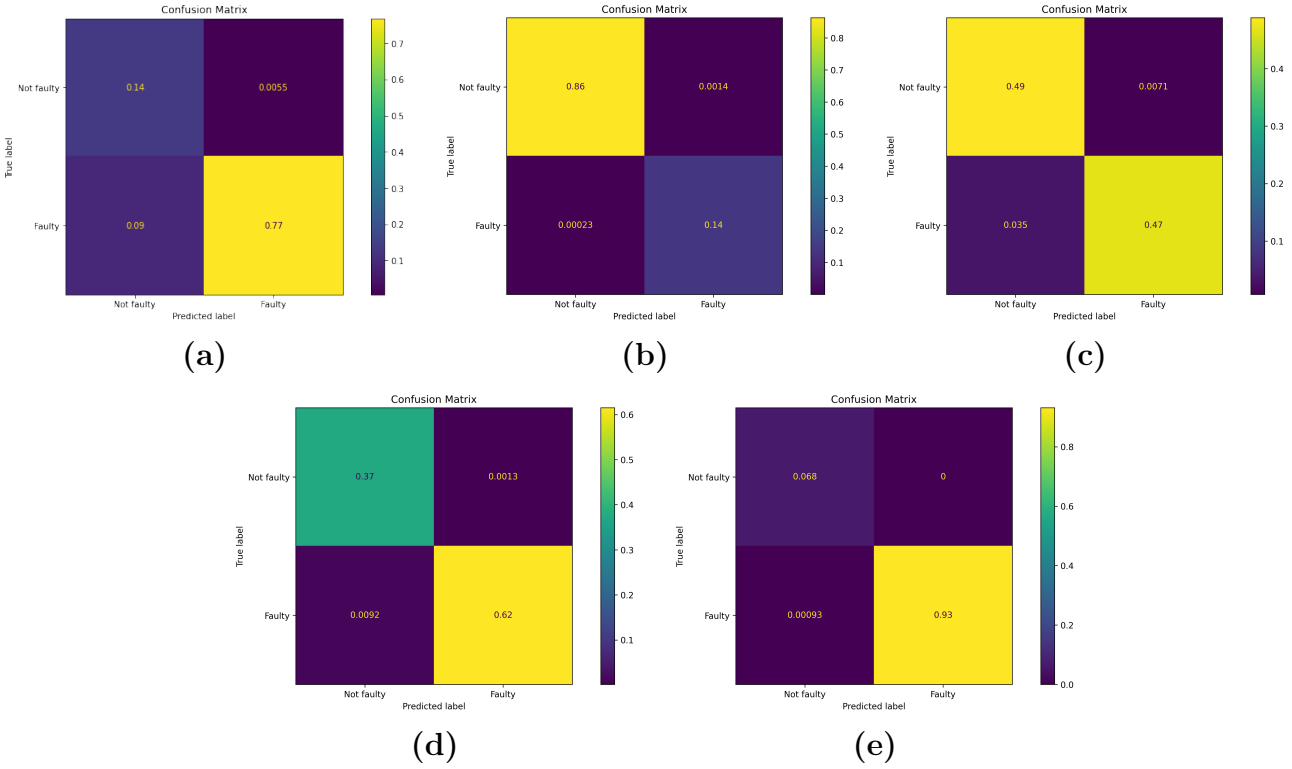


Figure 5.9: Confusion matrices of CatBoost (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

5.9(a) represents the confusion diagram generated for the MZVAV-1 dataset. From there we can see that TN= 0.14 and TP=0.77 and with this combination, we were able to reach an accuracy of approximately 91%. Similarly the confusion matrix for MZVAV-2-1 in 5.8(b) gave us an accuracy of more than 99% as TN=0.86 and TP=0.14. We can also use the combination of FP and FN to get the accuracy. For instance, the confusion matrix for MZVAV-2-2 in 5.8(c) has FP=0.0071 and FN=0.035, thus the addition of these gives us the inaccuracy (not in percentage). The process remains the same for the other two confusion matrices in Fig 5.8(d)(CatBoost implemented on SZVAV) and in 5.8(e)(CatBoost implemented on SZCAV) to carry out the accuracy rate.

5.7.8 Confusion matrices of XGBoost

Confusion matrices on all the datasets are:

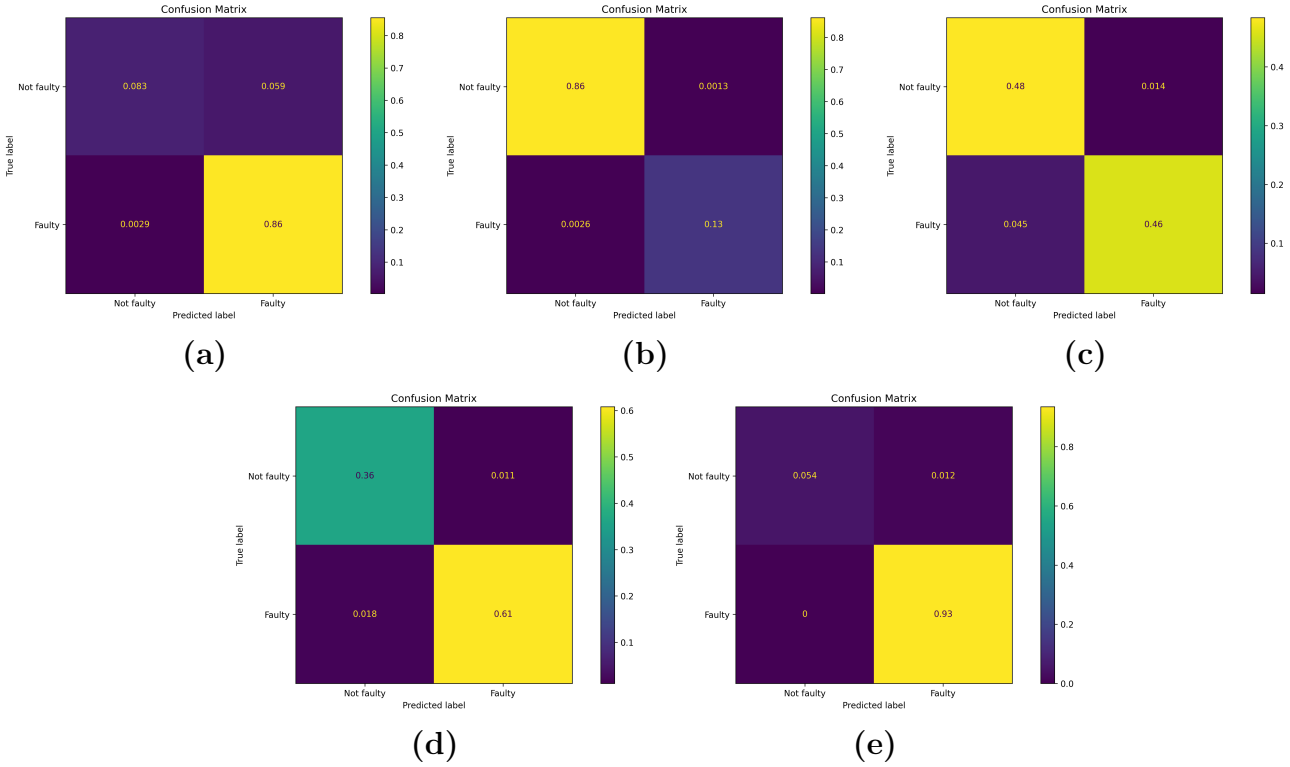


Figure 5.10: Confusion matrices of XGBoost (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

5.10(a) represents the confusion diagram generated for the MZVAV-1 dataset. Compared to other boosting techniques, XGBoost performed the best and this can be further illustrated by looking closely at the TN and TP values. However, examining the other confusion matrix shown in 5.10(b) (AdaBoost on MZVAV-2-1), 5.10(c) (AdaBoost on MZVAV-2-2) , 5.10(d) (AdaBoost on SZVAV) and 5.10(e) (AdaBoost on SZCAV) the addition of the values of TN and TP are greater . The values highlight that XGBoost performed second among boosting classifiers only to be beaten by CatBoost.

5.7.9 Confusion matrices of SGD

Confusion matrices on all the datasets are:

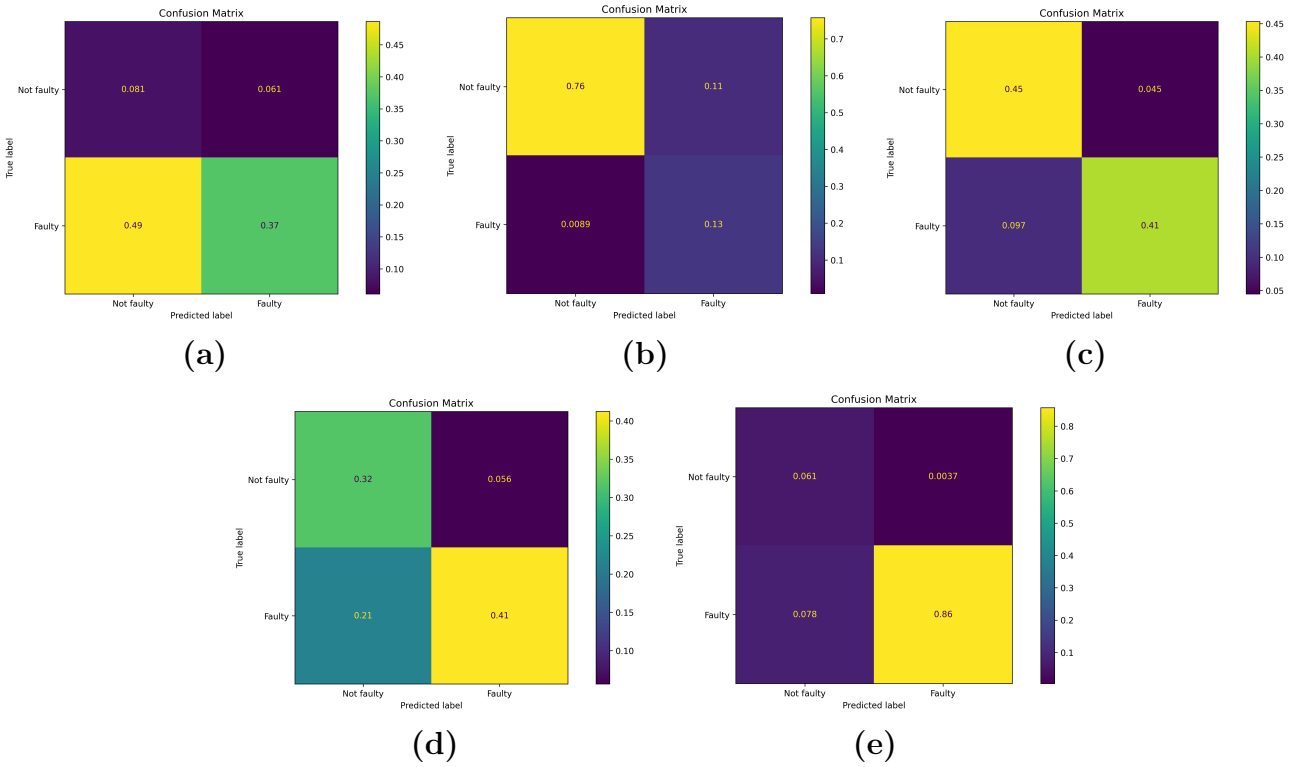


Figure 5.11: Confusion matrices of SGD (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

In 5.11(a) (SGD implemented on MZVAV-1), we can see that TN=0.081 and FN=0.37. This shows that this model performed very poorly on this dataset. If we look at the TP value in 5.11(b) (SGD implemented on MZVAV-2-1),it is 0.13 and TN = 0.76. The tesing accuracies for MZVAV-2-2 and SZCAV obtained are 85.82% and 91.85% respectively and their confusion matrices are represented in 5.11(c) and 5.11(e) respectively. On the other hand, the TP=0.41 and TN=0.32 in 5.11(d) (SGD on SZVAV), meaning the misclassification of labels were quite high (approximately 26.6%), thus making it one of the most unreliable models for this dataset.

5.7.10 Confusion matrices of MLP

Confusion matrices on all the datasets are:

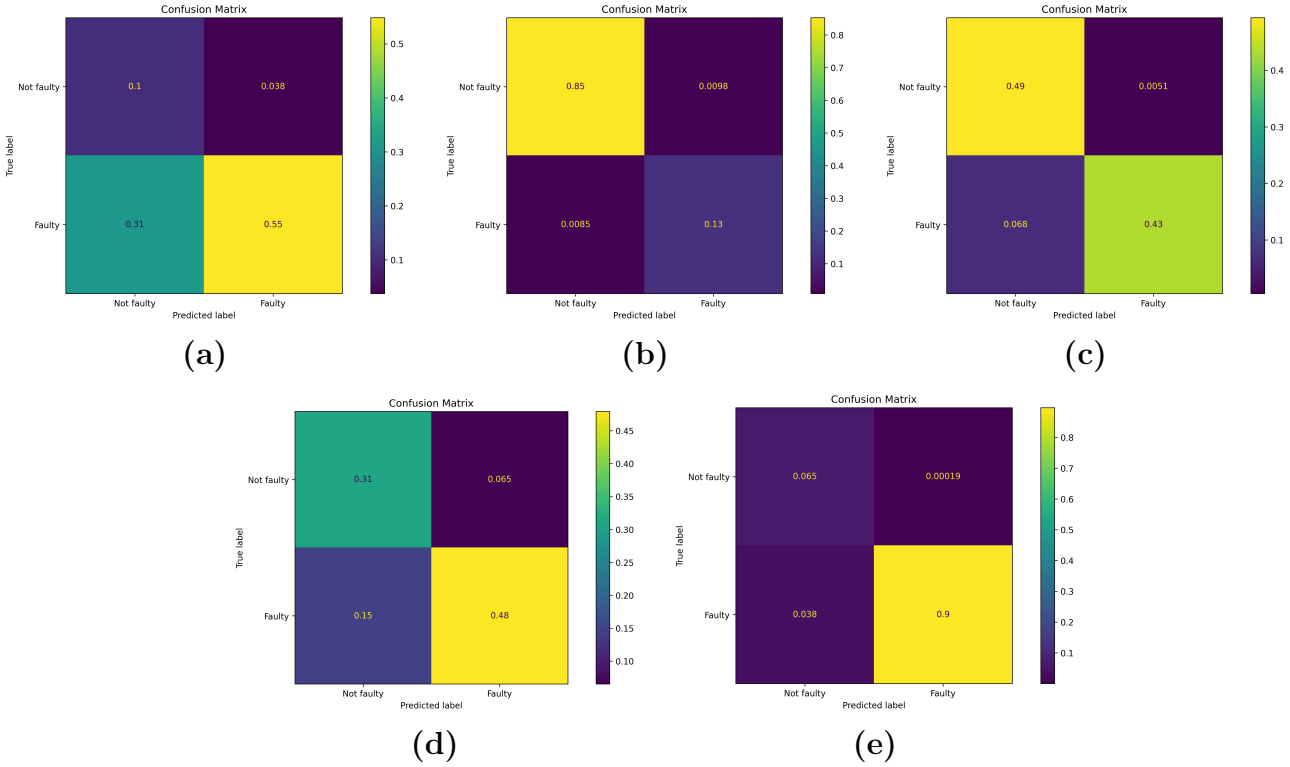


Figure 5.12: **C**onfusion matrices of MLP (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

In Fig 5.12(a) which is based on the implementation of MLP on the MZVAV-1 dataset. If we compare the performance of MLP on other datasets which are shown in Fig 5.12(b) (MLP implementation on MZVAV-2-1), 5.12(c) (MLP implementation on MZVAV-2-2), 5.12(c) (MLP implementation on MZVAV-2-2), 5.12(d)(MLP implementation on SZVAV) and 5.12(e)(MLP implementation on SZCAV), we can see that the results differ a lot and the reason behind this is the dataset consists of outliers. In Fig 5.11a, TN=0.1 and TP=0.55.

5.7.11 Confusion matrices of Naive Bayes

Confusion matrices on all the datasets are:

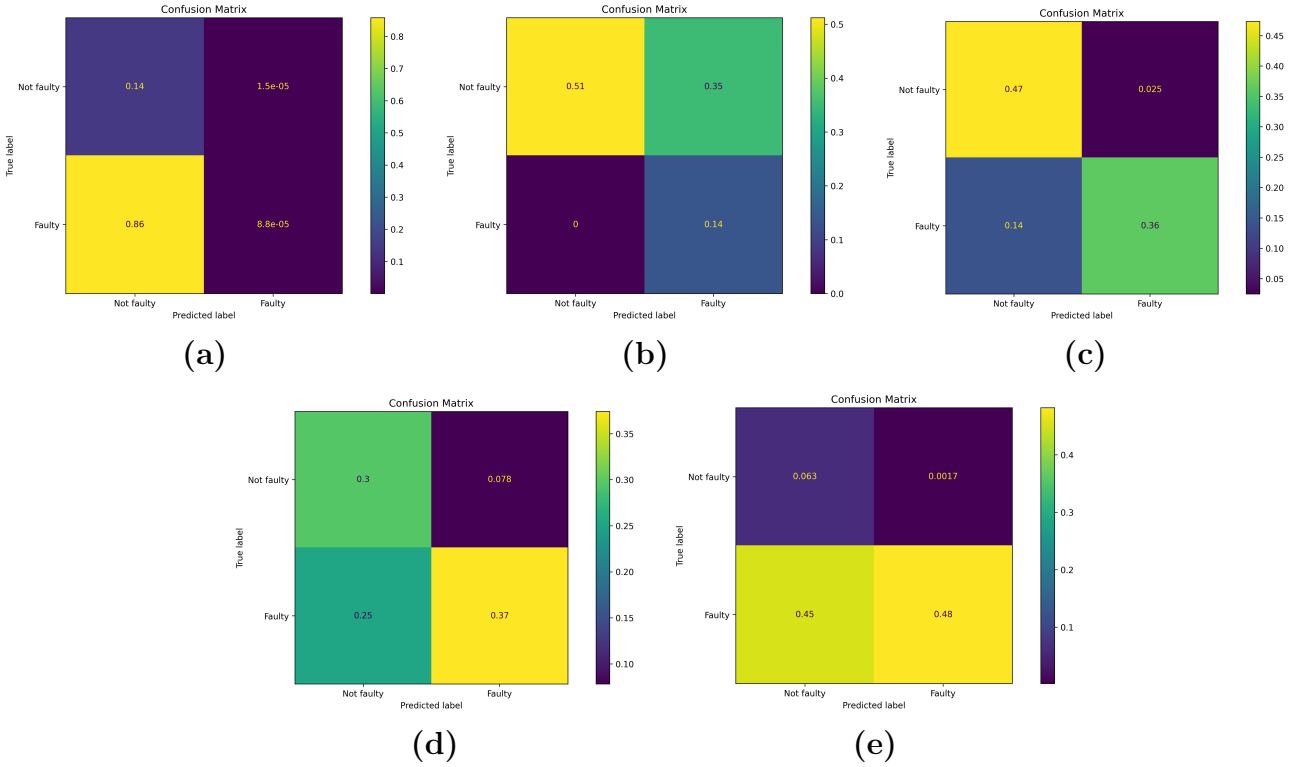


Figure 5.13: Confusion matrices of Naive Bayes (a) MZVAV-1 (b) MZVAV-2-1 (c) MZVAV-2-2 (d) SZVAV (e) SZCAV

Naive Bayes performed the worst on all the datasets. For further presentation on the insights, we have shown the confusion matrix. Here we can see for 5.13(a) (Naive Bayes implemented on MZVAV-1) TN=0.14 and TP=8.8e-05. Thus this shows us how the accuracy is only 14%. Similarly, 5.13(b) (Naive Bayes implemented on MZVAV-2-1) gave us disappointing accuracy. 5.13(c) (Naive Bayes implemented on MZVAV-2-2) is decent and with TN=0.47 and TP=0.36 we can see how a disappointing algorithm works on a well-balanced dataset. The accuracy for 5.13(d) (Naive Bayes implemented on SZVAV) and 5.13(e) (Naive Bayes implemented on SZCAV) are poor.

5.8 Comparison of different models used

For better understanding the performance of our model we provided recall and f1-socres beside training and testing accuracy. It is shown in the table 5.8.

1. Recall is the ratio of true positives to the sum of true positives and false negatives.
2. F1 is the weighted harmonic mean of precision and recall.
3. Testing and training accuracy specifies how accurate is model during the respective stages.

Model	**Dataset**	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Recall (%)**	**F1-Score (%)**
KNN	MZVAV-1	99.10	97.84	98.37	95.80
	MZVAV-2-1	99.73	99.25	99.28	98.46
	MZVAV-2-2	95.73	94.14	94.15	94.13
	SZCAV	99.93	99.94	99.97	99.77
	SZVAV	99.38	98.60	98.76	98.52
SVM	MZVAV-1	96.32	96.18	96.93	92.66
	MZVAV-2-1	99.88	99.38	98.73	98.71
	MZVAV-2-2	95.76	93.73	93.74	93.73
	SZCAV	99.99	99.90	92.42	95.64
	SZVAV	99.76	98.76	98.60	98.68
SGD	MZVAV-1	50.50	45.28	50.25	40.24
	MZVAV-2-1	92.04	88.57	90.65	81.09
	MZVAV-2-2	86.25	85.82	85.84	85.78
	SZCAV	93.45	91.85	92.98	77.78
	SZVAV	75.55	73.00	75.40	72.76
Decision tree	MZVAV-1	99.99	99.14	98.37	98.23
	MZVAV-2-1	100	99.68	99.30	99.33
	MZVAV-2-2	97.02	95.60	95.61	96.60
	SZCAV	100	99.91	99.68	99.62
	SZVAV	100	98.71	98.69	98.63
Random Forest	MZVAV-1	99.99	99.21	98.09	98.36
	MZVAV-2-1	100	99.85	99.74	99.68
	MZVAV-2-2	97.73	93.97	93.97	93.97
	SZCAV	100	100	100	100
	SZVAV	100	99.27	99.34	99.22

Model	Dataset	Training Accuracy (%)	Testing Accuracy (%)	Recall (%)	F1-Score (%)
Extra Trees	MZVAV-1	99.99	98.99	98.01	97.94
	MZVAV-2-1	100	99.91	99.83	99.80
	MZVAV-2-2	97.73	94.04	94.05	94.04
	SZCAV	100	100	100	100
	SZVAV	100	99.71	99.74	99.69
MLP	MZVAV-1	69.39	65.25	68.68	56.72
	MZVAV-2-1	96.65	98.16	96.32	96.14
	MZVAV-2-2	93.42	92.69	92.71	92.66
	SZCAV	97.78	96.15	97.80	87.51
	SZVAV	80.64	78.80	79.55	78.16
Naive Bayes	MZVAV-1	50.00	14.20	49.99	12.44
	MZVAV-2-1	79.19	64.92	79.67	59.19
	MZVAV-2-2	83.94	83.58	83.63	83.38
	SZCAV	75.02	54.53	74.48	44.88
	SZVAV	67.51	66.99	69.43	66.79
XGBoost	MZVAV-1	96.18	93.77	78.93	84.56
	MZVAV-2-1	99.91	99.61	98.98	99.17
	MZVAV-2-2	96.98	94.08	94.09	94.07
	SZCAV	99.42	98.84	91.14	94.83
	SZVAV	99.30	97.08	97.06	96.90
Adaboost	MZVAV-1	65.55	64.79	65.13	55.26
	MZVAV-2-1	99.70	99.07	98.83	98.07
	MZVAV-2-2	94.94	94.22	94.23	94.21
	SZCAV	98.77	97.47	98.51	91.16
	SZVAV	91.56	89.84	90.56	89.42
Catboost	MZVAV-1	93.64	90.45	92.80	84.04
	MZVAV-2-1	99.99	99.83	99.83	99.65
	MZVAV-2-2	96.88	95.76	95.79	95.76
	SZCAV	99.97	99.90	99.95	99.63
	SZVAV	99.70	99.08	99.14	99.02

Table 5.1: Comparison of different models used

Chapter 6

Conclusion

Our main objective was to develop a system that will help us ensure a pleasant experience while shopping or even while working in offices. In order to do that, we need to make sure the HVAC system works flawlessly.Our proposed system will be able to detect the slightest of anomalies. Those small anomalies are not detectable to us unless it turns into something bigger. By that time the whole system might start malfunctioning and maintenance becomes very expensive. Our system can detect tiny anomalies thus prevents large scale malfunctions by early maintenance. Even though the methodology we presented looks simple, the hard work behind it was intense. We had to search various dataset online and come up with one that would fulfill our requirement. Then properly examining and evaluating the dataset for any preprocessing required was also another important task for our project. However, not all dataset needed preprocessing. Splitting the dataset for training, validation and testing was also another major task. The algorithms we have presented are widely used and are even newly used in this field. The algorithms used on the dataset gave us different accuracy rates and from there we were able to figure out that tree-based algorithms (Decision tree, Random Forest, Extra Trees) are the superior ones for the problem at hand as the results were astonishing. However , not all of them performed up to the mark. If we look at the table shown in "Results and analysis", we can see that algorithms like Naive Bayes and SGD failed to give us our desired results. This is the only limitation we found in our project and we are determined to improve the performance of these algorithms as our future goal. Furthermore, the dataset we used is updating on a regular basis. Thus, our other goal is to keep implementing our algorithms on it and see whether we achieve similar results or are there any new tweaking needed. We will also try to come up with new algorithms that will help us lower down the time required for the results to be given out. Our thesis is just a part of something bigger. We believe that future enthusiasts will benefit vigorously from our research as they will have an insightful thought of what to expect from a system similar to ours.

Bibliography

[1] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[2] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006. DOI: 10.1007/s10994-006-6226-1.

[4] B. W. Olesen, "Using building mass to heat and cool," *Ashrae Journal*, vol. 54, no. 2, pp. 44–52, 2012.

[5] L. Chang, H. Wang, and L. Wang, "Fault detection and diagnosis of an hvac system using artificial immune recognition system," in *2013 IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)*, 2013, pp. 1–5. DOI: 10.1109/APPEEC.2013.6837247.

[6] Y.-Y. Song and L. Ying, "Decision tree methods: Applications for classification and prediction," *Shanghai archives of psychiatry*, vol. 27, no. 2, p. 130, 2015.

[7] Y. Zhou, D. Li, and C. J. Spanos, "Learning optimization friendly comfort model for hvac model predictive control," in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 2015, pp. 430–439. DOI: 10.1109/ICDMW.2015.119.

[8] T. Chen and C. Guestrin, "Xgboost," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016. DOI: 10.1145/2939672.2939785.

[9] N. E. Fernandez, S. Katipamula, W. Wang, Y. Xie, M. Zhao, and C. D. Corbin, "Impacts of commercial building controls on energy savings and peak load reduction," 2017. DOI: 10.2172/1400347.

[10] K. Kim and H.-I. Choi, "Adjusting initial weights for adaboost learning," in *2017 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, IEEE, 2017, pp. 1–5.

[11] M. Dey, S. P. Rana, and S. Dudley, "Semi-supervised learning techniques for automated fault detection and diagnosis of hvac systems," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, pp. 872–877. DOI: 10.1109/ICTAI.2018.00136.

[12] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: Gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[13] A. Naug and G. Biswas, "Data driven methods for energy reduction in large buildings," in *2018 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2018, pp. 131–138. DOI: 10.1109/SMARTCOMP.2018.00083.

[14] A. Yadav, *Support vector machines (svm)*, Oct. 2018. [Online]. Available: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589.

[15] E. Bisong, "The multilayer perceptron (mlp)," in. Sep. 2019, pp. 401–405, ISBN: 978-1-4842-4469-2. DOI: 10.1007/978-1-4842-4470-8_31.

[16] K. Taunk, S. De, S. Verma, and A. Swetapadma, "A brief review of nearest neighbor algorithm for learning and classification," in *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, 2019, pp. 1255–1260. DOI: 10.1109/ICCS45141.2019.9065747.

[17] J. Bao, "Multi-features based arrhythmia diagnosis algorithm using xgboost," in *2020 International Conference on Computing and Data Science (CDS)*, 2020, pp. 454–457. DOI: 10.1109/CDS49703.2020.00095.

[18] R. Carli, G. Cavone, S. Ben Othman, and M. Dotoli, "Iot based architecture for model predictive control of hvac systems in smart buildings," *Sensors*, vol. 20, no. 3, 2020, ISSN: 1424-8220. DOI: 10.3390/s20030781. [Online]. Available: https://www.mdpi.com/1424-8220/20/3/781.

[19] J. Ding, Z. Chen, L. Xiaolong, and B. Lai, "Sales forecasting based on catboost," in *2020 2nd International Conference on Information Technology and Computer Application (ITCA)*, IEEE, 2020, pp. 636–639.

[20] J. Granderson, G. Lin, A. Harding, P. Im, and Y. Chen, "Building fault detection data to aid diagnostic algorithm creation and performance testing," *Scientific Data*, vol. 7, no. 1, 2020. DOI: 10.1038/s41597-020-0398-6.

[21] Y. Li, Y. Mai, Z. Lin, and S. Liang, "Online transaction detection method using catboost model," in *2020 International Conference on Communications, Information System and Computer Engineering (CISCE)*, IEEE, 2020, pp. 236–240.

[22] F. H. Mohamed Salleh, M. b. Saripuddin, and R. bin Omar, "Predicting thermal comfort of hvac building using 6 thermal factors," in *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*, 2020, pp. 170–176. DOI: 10.1109/ICIMU49871.2020.9243466.

[23] A. Safiya Parvin and B. Saleena, "An ensemble classifier model to predict credit scoring - comparative analysis," in *2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, 2020, pp. 27–30. DOI: 10.1109/iSES50453.2020.00017.

[24] A. Gálvez, A. Diez-Olivan, D. Seneviratne, and D. Galar, "Fault detection and rul estimation for railway hvac systems using a hybrid model-based approach," *Sustainability*, vol. 13, no. 12, p. 6828, 2021.

[25] E. R. Setyaningsih and I. Listiowarni, "Categorization of exam questions based on bloom taxonomy using naïve bayes and laplace smoothing," in *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, 2021, pp. 330–333. DOI: 10.1109/EIConCIT50028.2021.9431862.

[26] I. Sittón-Candanedo, E. Hernández-Nieves, S. Rodrıguez-González, and F. de la Prieta-Pintado, "Fault predictive model for hvac systems in the context of industry 4.0," *information network*, vol. 9, p. 10,

[27] $Sklearn.neural_network.mlpclassifier$. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html.