上海交通大学硕士学位论文

基于虚拟传感器辅助 XGBOOST 模型的冷水机组制冷剂泄漏故障云诊断研究

硕士研究生：Burkay Anduv

学　　号：119020990024

导　　师：杜志敏 副教授

申请学位：工学硕士

学　　科：动力工程及工程热物理

所在单位：机械与动力工程学院

答辩日期：2022 年 1 月

授予学位单位：上海交通大学

Dissertation Submitted to Shanghai Jiao Tong University for the Degree of Master

CLOUD-BASED DIAGNOSIS OF REFRIGERANT LEAKAGE FAULT OF CHILLER USING VIRTUAL SENSOR RESIDUALS ASSISTED XGBOOST ALGORITHM

Candidate:	Burkay Anduv
Student ID:	119020990024
Supervisor:	A/Prof. Zhimin Du
Academic Degree Applied for:	Master of Engineering
Speciality:	Power Engineering and Engineering Thermophysics
Affiliation:	School of Mechanical and Power Engineering
Date of Defense:	January, 2022
Degree-Conferring-Institution:	Shanghai Jiao Tong University

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文《基于数字孪生的冷水机组故障检测与诊断研究》，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：2021年01月13日

上海交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密☑。

（请在以上方框内打“√”）

学位论文作者签名：

日期：2022年01月13日

指导教师签名：

日期：2022年01月

基于虚拟传感器残差辅助 XGBOOST 算法的冷水机组制冷剂泄漏故障云诊断研究

摘 要

近年来，传感器技术的普及和应用使得冷水机组运行的海量数据能够得以利用，进而促进了能够改善建筑能源效率的新型智能故障检测和诊断（FDD）方法的流行和通用。然而，尽管现有的数据处理方法非常强大，但冷水机组收集的实验数据通常难以覆盖数据驱动方法所需的完备工况，开发的故障诊断模型难于适应实际应用中的各类工况。另外，现有研究中预处理方式的缺陷或评估指标的不当选取很容易导致对模型的测试结果错误评价，进而导致当前 FDD 方法存在泛化能力低和过度拟合的问题。

本文重点研究了诊断模型错误评价原因，提出一种新的冷水机组制冷剂泄漏故障诊断框架来改进诊断模型的泛化性能。为此，首先收集工业冷水机组的故障和无故障运行数据。然后，在对原始数据的预处理后，利用无故障数据建立了虚拟传感器。提出使用 XGBoost 学习虚拟传感器与实际传感器的预测残差，构建虚拟传感器辅助的 XGBoost 分类器，实现对制冷剂的充注水平的预测。

提出的故障诊断与支持向量机（SVM）、多层感知器（MLP）、随机森林（RF）和纯极值梯度增强（XGBoost）等其他被广泛接受的传统 FDD 方法进行了比较。结果表明，传统方法在随机测试中的准确率均在 99%以上。然而，当对其进行 k 折交叉验证测试时，传统方法中性能最优的诊断模型也只能达到 54%、68%和 72%的诊断精度。本文所提出的故障诊断思路在随机测试中同样保持了 99%以上的准确率，并且对于 k 折交叉验证能够获得 68%、70%和 72%的准确率结果，具有较大的性能优势。

最后，构建了一个基于云端诊断的 FDD 框架，可在用户友好的网站上实时监测冷水机组的运行状况。FDD 模块单独使用一个专用服务器，基于冷水机组实时采集的传感器数据来预测制冷剂充注水平以辨识冷水机组的健康状态本文开发的故障诊断模型上传到云端诊断的 FDD 框架中，并通过在线的方式完成仿真和测试研究。

关键词：深度学习；极限梯度提升；预处理；暖通空调系统；故障诊断；回归

CLOUD-BASED DIAGNOSIS OF REFRIGERANT LEAKAGE FAULT OF CHILLER USING VIRTUAL SENSOR RESIDUALS ASSISTED XGBOOST ALGORITHM

ABSTRACT

Over the recent decades, improvements in the sensor technology have enabled researchers to access extensive operational data of HVAC chillers. This large amounts of data have facilitated the adaptation of novel smart fault detection and diagnosis (FDD) methods as a means to increasing the energy efficiency of buildings. However, although the data processing methods are very powerful, the experimental data collected from chillers generally lack the sufficient diversity and standardization that the data driven methods require to obtain a general model that can be applied in various cases. A deficiency in preprocessing or a misconception in the selection evaluation metric may easily result in a misinterpretation of the test results. If there is such misjudgment, the final FDD method would suffer heavily from low generalization capacity and overfitting.

This paper highlights the origins of common misconceptions and attempts to achieve a higher generalization performance than the conventional methods by constructing a novel procedure to diagnose the refrigerant leakage fault of chillers. To achieve this target, first the faulty and fault-free operation data of an industrial chiller is collected. Then, after conducting the relevant preprocessing steps, an algorithm that creates virtual sensors based on fault-free conditions is constructed. This algorithm is then used to calculate selected residuals of actual sensor readings and virtual sensor predictions. An optimized XGBoost algorithm is supplied with both normal and faulty conditions as inputs to make the final prediction on the refrigerant charge level.

The diagnosis results from the proposed method are compared with other widely-accepted methods such as Support Vector Machines (SVM), Multi-Layer Perceptron (MLP), Random Forest (RF) and pure Extreme Gradient Boosting (XGBoost). The results reveal that, all of the traditional methods have above 99% accuracy in random testing. Yet, when these methods are k-fold tested, even the best compared model could only achieve accuracies of 54%, 68% and 72%. The proposed method, on the other hand, achieved an exceptional generalization performance with accuracy results of 68%, 70% and 72%, while the method still retained above 99% accuracy in random testing.

As the final step, a cloud based FDD framework is constructed to track the status chiller health from a user friendly website. The framework uses a dedicated server for its FDD module, where the software makes judgements on chiller health by predicting the refrigerant load level from the sensor data obtained by the chiller in real-time. The proposed model is uploaded to the framework and the study is concluded with conducting simulations and tests on the online system.

Keywords: Deep learning; XGBoost; Preprocessing; HVAC system; FDD; Regression

Contents

Table of figures

List of tables

Nomenclature

T	temperature (°C)
P	pressure (Pa)
F	flow rate ($m^3\ h^{-1}$)
I	current (A)
Abbreviations	
SVM	support vector machine
MLP	multi-layer perceptron
RF	random forest
XGB/XGBoost	extreme gradient boosting
VSR	virtual sensor residuals
MW	moving window
MSE	mean-squared error
Greek Symbols	
Δ	differential value
Ω	regularization function
γ	regularization coefficient for number of tree leaves
λ	regularization coefficient for leaf weights

Subscripts and superscripts

i	inlet
o	outlet
r	refrigerant
cd	condenser
ev	evaporator
com	compressor
cw	chilled water
evp	evaporator refrigerant pool
ap	approaching
dis	discharge
sh	superheat
sc	sub-cooling
rpe	evaporator refrigerant pool
suc	compressor suction
ex	exhaust
amb	outdoor / ambient

Chapter 1

Introduction

1.1. Background

Chiller is one of the most important components of heating, ventilation and air conditioning (HVAC) systems, which are increasingly more essential to ensure thermal comfort in buildings. Since HVAC systems commit a large share of the total building energy consumption, researchers and engineers are pursuing for new techniques to limit this consumption [1]. Moreover, it is also crucial to prevent the economic loss that might be caused by some faults occurring in the HVAC systems over a lengthy period of time with use of relevant fault detection and diagnosis (FDD) tools. Refrigerant undercharge fault is one of the most frequent faults occurring in these systems, as 34% of the systems have this fault with various severities [2]. There are several causes of this type of fault such as, a failure to charge the refrigerant according to the requirements, leakage caused by improper maintenance or operation and insufficient air tightness of the system [3].

Similar to other common faults, refrigerant undercharge fault may also result in improper cooling performance, decreased efficiency of the chiller system and even may cause a total failure of the HVAC system. Additionally, such fault might also provoke the chiller expansion valve to fluctuate heavily, leading to instability and additional risks for both safety and the regulation capacity of the system [4]. To minimize such risks posed by refrigerant undercharge fault, it is therefore critical to develop a FDD strategy to eliminate the problems that might arise from the fault remaining undetected during the operation of chiller system.

Even though the complete FDD methods in the literature are proposed for detecting and classifying different faults or diagnosing varying fault severities of a specific fault, the core techniques and algorithms of the majority of the FDD methods are not fault specific. This statement is especially true for data-driven methods where the input data is collected from the same chiller system regardless of the inspected fault. Therefore, since different FDD methodologies can be used and incorporated interchangeably, this section starts with a general literature review of FDD methodologies, then proceeds with the specific diagnosis methods of refrigerant undercharge fault and concludes with the contributions of this study.

1.2. State of art

There are several ways to categorize FDD methods that can be found in the literature. Based on the work of Yang et al. [5], FDD methods can be classified into data-driven (DD) models, gray box models, and prior knowledge-based (rule-based) methods.

1.2.1. Model based FDD methods

The main idea of model based methods is to construct a dynamic process for signal and parameter estimation [6, 7]. First-principle method and the gray box method are the two most widely used approaches [8, 9, 10]. Both dynamic and static systems can be modeled with the first-principle method according to the systems' physical characteristics. However, since the time response is slow and the modeling can only be applied for steady-state conditions, this approach is not suitable for online FDD process. This drawback of slow response can be improved with exploiting the gray-box method, where it may stabilize the system by having a direct response to abrupt faults. On the contrary, using a gray-box model still require precise physical modeling coupled with regression techniques, thus there exists an uncertainty of oversimplifying the

complex HVAC system models [11]. Moreover, any modification on the system requires a construction of a completely new model since previous models are not usable anymore.

1.2.2. Data driven FDD methods

Zhang et al. [12] categorized the data driven approaches into qualitative and quantitative methods. In such categorization, qualitative-based methods include Expert systems, pattern recognition, frequency analysis and fuzzy logic. On the other hand, data-driven quantitative-based methods are divided into two sub categories, namely, statistical methods and neural networks. The common aspect of data-driven methods is that they use operational data which is collected from the system that is being studied. Contrastingly to the model-based approach, data-driven methods have more practicality for applying FDD to the complex HVAC systems since they are not concerned with system complexity and solely depend on the historical data of system operation. This real system operational data is the only requirement of data-driven methods and there are no additional needs such as expert knowledge or the physical models on for the HVAC system [13, 14].

Data-driven methods can also be divided into two groups such as data-mining based methods (or smart methods) and statistical methods. The statistical methods allow the FDD process to have flexible dimensionality reduction to reduce the amount of valuable data to be inspected. By employing dimensionality reduction, high weighted or manually selected features are selected as inputs for the chosen FDD method. Previous studies on data-mining based FDD employed various data-mining approaches to study the input and output data relationships to detect the abnormalities within the HVAC systems [12]. Utilization of pure data-mining methods or a combination statistical and data-mining methods is possible to develop a data-driven FDD process [15].

However, Ebrahimifakhar et al. [16] argued that the when overall accuracy of finding faults in a HVAC system is considered, employing data-mining-based classification methods such as support vector machines (SVM) generally yields better results than the statistical methods (i.e. linear discriminant analysis). Data-mining methods also can be divided into two sub-categories such as supervised and unsupervised methods. In the former, the reorganization of data patterns are based on the training set, and on the latter, the underlying relations in the dataset are summarized. In the recent years, various FDD methods are developed for HVAC systems utilizing data-driven supervised, semi-supervised and unsupervised data-mining techniques. Additionally, there are many hybrid methods in the literature where two or more of these techniques are employed to improve the accuracy and reliability of these methods.

1.2.3. Knowledge based FDD methods

Alzghoul et al. [17] defined knowledge-based method as the combination of a qualitative part of the model-based method, which includes fault trees, structural graphs, or qualitative physics, and data-driven qualitative subcategory including expert systems or fuzzy logic. In the circumstances where the physical or mathematical modeling of the system is too costly and computationally expensive, knowledge-based methods are generally preferred to be used for FDD. Moreover, usage of a knowledge-based method may be more preferable when modeling the chiller is not feasible due to a small number of inputs, outputs, and states of the system or when there is a requirement of specific domain knowledge to model the system [9, 18].

1.2.4. Combination of different methods

Combinations of different types of FDD methods can also be utilized to achieve a higher accuracy [19]. Models merging together model-based methods and supervised data-driven

methods to detect the system level faults are proposed by Ding [20] and Khorasgani et al. [21]. To successfully integrate two methods into a single FDD process, a new model structure is constructed. For example, two layers of Bayesian network containing fault and fault symptoms layers can be constructed which takes advantage of previous knowledge regarding the faults and the symptoms of other related systems [22]. The number of requirements for the attributes of modeling can be decreased by combining model-based and data-driven methods. Yet, there still exists some limitations for combining physical models with data-driven or knowledge-based methods. For instance, over-simplifying the modeling of large-scale HVAC systems may cause a high rate of false positives and modeling errors. Also, the process of simulating various faulty conditions can be very time-consuming and this type of combined method might need big data storage [23].

1.3. Data driven fault detection and diagnosis

1.3.1. Supervised FDD methods

First branch of supervised learning methods is the off-line supervised learning. Each observation in the training set for off-line learning process incorporates both input and output values (labels). After the training is completed, new data is then classified based on the model parameters which are learned from the training set. Supervised learning is a robust method in terms of training the model, also it can enables researchers to evaluate the performance of the FDD process by providing feedback such as the prediction accuracy and leaning results. This method is defined as backward approach in the data-mining, since the pre-defined output model is used. In fact, expert knowledge and accurate training data are critical for this method to initiate the initial training [24]. Second branch of supervised FDD methods is online supervised-learning, in which both training and testing procedures are conducted simultaneously for corresponding dataset. This

process involves two consecutive steps: prediction of labels and label correction. Thus, instead of initially separating of the dataset into train and test sections, a prediction is made in every iteration for each sample, which is followed by a correction of the predicted label in the subsequent step. This label correction is the core method that enables the improvement of the accuracy for future predictions [25].

In another approach, supervised learning methods can be categorized into two main groups: classification (which is for discrete target values) and regression. Some of the classification methods are Bayesian networks, Decision trees, and Support Vector Machines (SVM). On the other hand, the regression methods includes other methods such as lazy learners (e.g., K-nearest neighbor (KNN)), supervised neural networks, and ensemble methods. A top-down approach is employed by decision-tree methods. Where, in the first stage, the class labels are constructed, and these labels are utilized for classification of unseen data in the following stages.

SVM search for the optimal separating hyperplane with usage of support vectors, which makes it is a powerful tool for classification. Also, SVM can boost the generalization and minimize error by finding large margin separators which include training error and confidence level [26]. SVM has been widely used because of its performance of solving non-linear problems with relatively high accuracy. For instance, Namburu et al. [27] determined the most sensitive sensor for fault diagnosis in a chiller system with a genetic algorithm and then developed a data-driven generic FDD method. First, the fault is detected with SVM, PCA, and partial least-squares (PLS) methods. Then, the PLS is used again to assess fault severity. Han et al. [28] studied on Multi-label SVM (ML-SVM) for labeling multiple classes simultaneously. In their work, the more than two faults occurring in the system at the same time could be identified. They have extended their research with two other SVM-based FDD methodologies for chillers [29, 30]. Non-linear support vector

regression (SVR) is another way that can be applied to recognize fault patterns and detect faults. To resolve the mismatch of linear regression functionality and non-linear FDD problems, utilization of logistic regression is proposed. Anh et al. [31] constructed modified a least square support vector regression (LSSVR) model where the accuracy of the FDD method is improved. Besides, Anh et al. [32] also conducted a study where the performance of different regression based FDD algorithms are inspected. Moreover, they have proposed a least square-SVM (LS-SVM) regression-based method for fault detection in chillers, together with Han et al. [33]. The two studies show that the quality and quantity of labeled training data is a major concern for both LS-SVM and SVM methods. But, when the size of available faulty data is small, LS-SVM showed a slightly better performance with more accuracy than SVM.

There are also numerous studies where a neural-network approach is used to find faults in systems. In instance, Wang et al. [34] applied a kernel-based partial least square regression for detecting and diagnosing faults in heat exchangers and a SVR model is used by Bailey et al. [35] for FDD of a chiller. In the study, 28 inputs, seven outputs and two hidden layers for the artificial neural network (ANN) algorithm are defined to achieve an accurate result. Hou et al. [36] adopted a RS method for omitting redundant attributes of the ANN for FDD process of an air conditioning system.

Bayesian networks function is another method that can be applied in FDD methodologies which is based on the conditional probabilities theorem that predicts the response value of an observation set. For FDD in HVAC systems, fault labels are the response values. Even if complete information about the system is not available, Bayesian networks still have the ability to work well [35]. Since the naïve Bayesian assumption requires independence between observations, these models might have troubles in finding the indirect effects from faults in the case of dependencies,

thus there might be a degree of error in the network output. Several researchers combined the Bayesian network with The Gaussian naïve Bayesian network, which have the assumptions of a normal distribution for the response value where the weights of all prediction classes are identical. Some variations of Bayesian belief networks minimizing the effects of dependencies in the FDD process of air handling unit system is proposed by Zhao et al. [37, 38] in other studies.

Regression models are generally employed for numerical prediction. In regression models, logistic and linear regression methods are used for binary and numerical of regression, respectively [26]. Since the FDD methods primarily aim to construct intrinsic mathematical relationships between different attributes to detect the abnormalities [39], these two methods may also be applied in FDD process.

ANN-based methods have a great application potential for FDD in building HVAC components [40, 41]. ANN-based FDD methods mostly utilize on residuals and thresholds. Residuals contain the information about the deviation from system's normal operating conditions and can be calculated by subtracting the predictions from measured value [31, 42–44]. Moreover, a calculation of the residuals is crucial when the severity of a fault is estimated. ANN methods can be divided into two main methods based on the utilization of residual information. In the first group, ANN methods detect the faults by generating residuals and by comparing these residuals with predefined thresholds. In the second group, ANN methods categorize abnormal residuals in distinct fault groups [35]. The thresholds in the second group are normally decided with a trial-and-error method [45] or expert knowledge. To avoid frequent false alarms, there are two more ways to determine the most appropriate threshold besides statistical testing and norm-based residuals [46]. If there are different types of faults simultaneously present in the system, ANN

takes every residual as input to detect the residuals which are related to a specific fault. If a relationship is found, ANN classifies the faults in their corresponding group [47].

As a recent variant of ANN, deep-learning based FDD, is continually receiving more attention, since such methods generally provide higher accuracy and robustness compared to other supervised FDD methods when the available labeled data is limited [48]. In recent years, a number of deep neural network (DNN) methods are applied to HVAC systems for FDD [49]. In example, a recurrent neural network (RNN) is developed by Shahnazeri et al. [50] to find faults in an HVAC system. Even though their method did not require historical data, the limited capacity of their method to capture the nonlinearity have limited the accuracy of their results. While using DNN based FDD methods, automatic feature selection step is another factor that might influence the accuracy of the results. In the some situations, this selection may cause the loss of some essential data. This phenomena occurs when low weights are automatically assigned to some critical data during the training process. One of the common problem that significantly decrease the accuracy of the results is the collection of faulty samples from older components and training the model with these samples. Moreover, when the training of FDD model is done with the use of the data which is collected from the first months of the operating period of the components, the data lacks the systematic routine for the adjustment of FDD model to actual operating conditions [51]. Furthermore, it is also argued that the supervised FDD methods are more applicable to steady-state operating conditions rather than the transient state operation [52].

1.3.2. Semi-supervised FDD methods

A complete dataset is required to initialize the training process of supervised learning. However, the faults present in the system is usually not labeled in training obtained from HVAC

systems. Semi-supervised learning models aims to tackle this issue. The definition of semi-supervised learning is when only non-fault case labels are supplied to the training algorithm [52]. If only a few faulty samples are provided, semi-supervised learning models might have a better performance than supervised learning. The semi-supervised methods makes a comparison between each sample with a non-faulty class and the training set is expanded with new faults after each iteration. Therefore, with each iteration, the size of the training set with faulty samples is expanded. Subsequently, the algorithm evaluates the remainder of the testing data. However, it must be kept in mind that semi-supervised learning is computationally more costly compared to supervised learning.

A semi-supervised SVM for an FDD process of an AHU system is proposed by Yan et al. [53] and the method is validated by comparing it with other machine learning techniques such as random forest, semi-supervised KNN, and semi-supervised classification and regression tree (CART). It is demonstrated that semi-supervised SVM achieves a higher accuracy compared to other semi-supervised methods. Specifically, semi-supervised SVM retained 93% accuracy when the training is limited to using only 6.66% of faulty samples. Moreover, it is found out that increasing the number of faulty samples in the training dataset did not have an effect on the prediction accuracy. In their more recent study, they have also implemented generative adversarial network (GAN) extensions to their work, aiming to improve the accuracy [54]. The purpose of the addition of GAN is synthetically increasing the number of faulty training observations of the AHU system. Even though the proposed methodology achieves a higher accuracy in compared to classical supervised methods, the hyperparameters of the model still need to be configured according to different system configurations.

1.3.3. Unsupervised FDD methods

If there are no labels provided for all sets of training data, the learning method is called to be unsupervised [26]. This method enables the construction of indirect correlation between features of the dataset. Moreover, other useful relational knowledge and structural information can be extracted from unlabeled data via the usage of unsupervised learning. The most widely used methods that use unsupervised learning are: association rule mining (ARM) [24, 57, 58], clustering [26, 55, 56], motif discovery [59, 60], and self-organized neural networks [24].

ARM methods are employed to find structural patterns and relationships among variables, which makes it also an effective method to discover the internal regularities within the dataset. Furthermore, hidden patterns between features can be extracted in some cases [57]. An ARM method is implemented by Yu et al. [42] where daily and yearly data are collected and then compared to extract structural rules and finding the inherent faults of a mechanical ventilation system [42].

Another practical unsupervised approach for FDD is clustering. In clustering method, sets of samples are grouped into classes constructed from similar samples so that the samples existing in the same cluster are the ones that are more likely to each other, and the samples that are in different clusters have less similarities with one another. Clustering method also can be a powerful tool to understand time specific behavior of the building's elements, since it can extract intra-cluster relationships and other useful knowledge from these elements. For example, Reddy et al. [43] implemented clustering analysis to the hourly energy consumption of the building, where they aimed to predict the total short-term energy consumption. Xue et al. [61] utilized a combination of clustering and ARM to detect faults in a district heating system. First, different seasonal patterns

of operation in the heating system are classified via clustering. Then, associated rules for faults are found from in each cluster by employing ARM.

Some limitations also appear in the application of unsupervised methods. The results retrieved using such methods (i.e., rules from ARM) are usually very large and might include many unnecessary data. Therefore, post-processing methods are generally necessary to determine the correlations and automatically derive meaningful results from these large amount of returned results. Additionally, when clustering is employed to process the operational data of the system, the total number of clusters is generally defined based on the fault groups known from previous knowledge in some cases [62]. Thus, clustering method fails to diagnose all new faults which are not recognized as members to existing groups. Aside from this, contrary to supervised methods, the complicated correlations among multiple features may be difficult to detect for clustering analysis and ARM in the post-mining step [63].

Unsupervised ANN methods are comprehensively reviewed by Krarti et al. [64]. The FDD process can be enhanced with self-organized ANN algorithms to create an unsupervised model. The utilization of such algorithms can significantly simplify the process by reducing the number of iterations in the training process [65]. In other words, the speed of the FDD process can be considerably increased with self-organized ANN.

Pattern recognition and motif discovery is also another area of FDD methods that received attention in recent years. These methods are generally employed as a preprocessing algorithm or as a semi-supervised method used in combination with PCA methods [66]. For example, a combined PCA and symbolic aggregate approximation (SAX) pattern matching method [54] is employed in FDD for the whole building. First, a regression method together with a generic algorithms is implemented on the data to find the initial key features. Then, the key features

obtained in the feature selection stage are used in PCA. SAX produced a weather-based pattern threshold and this threshold is compared with the historical data to detect the faults. Although, motif finder and pattern recognition methods have major advantages when studying time series data, there is still room for improvement in application of these to large scale HVAC system FDD processes.

1.3.4. Hybrid FDD methods

There are many examples in the literature where the core FDD applications of different processes utilizing supervised, semi-supervised or unsupervised leaning are examined. However, in terms of minimizing the false alarms, the results obtained from these methods generally do not have sufficient precision in large-scale applications. The integrated approach consisting of both unsupervised and supervised learning techniques has been broadly employed in recent years to improve such precision. Du et al. [39] coupled subtractive clustering analysis and two consecutive neural networks (a combined neural network) to detect the faults and abnormities in the AHU. The first neural network detects the sensitivity of attributes and was utilized for feature selection. The selected features were later taken as targets to both first and second (auxiliary) neural network. To determine the corresponding weights target variables for each neural network, a PCA was also used. Then, these weights were used to calculate the combined relative error. If the defined error exceeds the threshold, the sample was labeled as a fault. Finally, the faults in the new data are diagnosed with subtractive clustering analysis.

A combination of CART, clustering, and EANN is employed to identify faults by detecting outliers in a electricity unit dataset in another study [41]. CART and K-means models are applied to find outliers in the first stage where a generalized extreme standard deviate (GESD) is used.

Then, the results from the k-means, CART, and GESD methods are compared with the noise from DBSCAN to find the best possible algorithm which minimizes number of false anomalies. The results revealed that, DBSCAN noise clustering is more applicable for grouping the data than the k-means, and the single neural network is found to be less robust compared to ensemble neural networks (EANN).

A multi-class-SVM based FDD methodology is proposed by Dey et al. [67] on data from fan-coil units. As an extension to the supervised algorithm, the data of a building's fan-coil units were also processed by x-means clustering to diagnose system level faults. Bayesian information criteria is used to automatically construct multiple clusters. Davies–Bouldin and silhouette techniques are employed to validate and assess the number of clusters. Finally, a Gaussian mixture model and hierarchical clustering are used to the comparatively validate of the results.

The amount studies on hybrid data-driven FDD processes are still limited when compared to the other methods present in the literature. Also, there are only a few researchers that utilize hybrid models for predicting system parameters. In example, Naseri et al. [68] combined clustering and a neural network to predict the daily peak load of electricity consumption with a hybrid model. The results reveal that data-driven had a more accuracy compared to the statistical methods. In another study, both the transient and non-transient operational data of an AHU are examined by Piscitelli et al. [69]. First, a temporal ARM algorithm is employed to diagnose the faults during the non-transient period and then classification models are implemented to assess the severity of the faults. It is possible for a hybrid model to contain limitations of both supervised and unsupervised techniques. However, a careful selection and combination of supervised and unsupervised learning methods may help to boost accuracy and decrease the training time of FDD process [70].

1.4. Refrigerant leakage fault detection

In addition to the studies previously mentioned, some researchers specifically focus on diagnosing refrigerant leakage fault. Following are some of the studies in the literature. Esbri[71] et al constructed a model based fault state observer using adaptive Kalman filter and genetic algorithm for diagnosing refrigerant leakage fault. The model has a significant performance on distinguishing the faulty operation with refrigerant leak from normal operating conditions, and it is also applicable for online FDD. Based on a rule-based method, a virtual sensor for the refrigerant charge level of several refrigeration and air-cooling systems is developed by Kim et al. [72], where the experimental results showed a considerable performance. Furthermore, Zhao [73] introduced an undercooling perfection coefficient as a principal fault characteristic variable to further increase the diagnosis accuracy of refrigerant undercharge fault. However, the models ability to represent the time-varying and extensive operating state of chillers is debatable, since the parameters of the model are selected based on past experience from singular data points. As a solid illustration of gray box models, a hybrid model to improve the prediction accuracy of the refrigerant undercharge fault is proposed by Zhu et al. [74].

1.5. Research objectives

Provided that the training data is sufficiently large, majority of the data driven methods claim to yield high accuracies in their corresponding dataset. However, many models experience difficulties when they are applied to another chiller than the specific chiller that the model have been trained for, and a migration scheme is therefore crucial to use the same model for a different chiller [75]. This phenomena urged some researchers to question the generalization capacities of the data driven methods. In fact, randomly sampling the test data is problematic due to the method

of experimental chiller data collection. Conventionally, only the steady state sensor data of the chiller at predefined operating conditions are taken as inputs to the data driven models. Although this is required to discard the noise resulting from the time variations and delays in the chiller system, collecting large amounts of data at a limited number of operating conditions causes the sensor data to form mini cloud clusters for respective conditions in the overall dataset. When the training and test datasets are randomly chosen, both training and test data points would co-exist in the same cloud for every cluster. Then, since the training and testing points would be stationed in very close proximity, the algorithm would train and test for basically the same points, thus test result would resemble nothing other than training loss. It is a common misconception that some studies claiming this random test result as a final accuracy for their proposed method. This mistake in choosing the proper evaluation metric might be the reason why these models have very low generalization capacity.

In many cases, it may not be practical to change the experimentation method, thus this paper proposes a change in evaluation method and utilizes k-fold cross validation method instead of random sampling as a testing scheme to correctly measure the accuracy of FDD models. Furthermore, comparative results of random and k-fold sampled tests of several conventional data-driven methods are included in this paper to clearly demonstrate the aforementioned phenomena. Conclusively, this paper proposes a novel methodology to create virtual sensor models and supplies the residuals of actual and virtual sensor values as inputs to an optimized XGBoost algorithm to considerably improve the generalization performance, and constructs a cloud-based framework to reduce the hardware requirements of FDD applications on chillers.

The following is the structure of this paper. First, experimental procedure with different fault scenarios are specified. Second, the initial data filtering and feature selection methods are

presented. Third the input data processing method using virtual sensor residuals is developed. Then, the selected virtual sensor residuals are supplied to the XGBoost algorithm and fault diagnosis results are obtained. Next, an online framework for real time fault diagnosis is developed. Finally, the proposed method is verified by comparing its results with the results from conventional algorithms. The Figure 1-1 demonstrates the overall structure of this study.

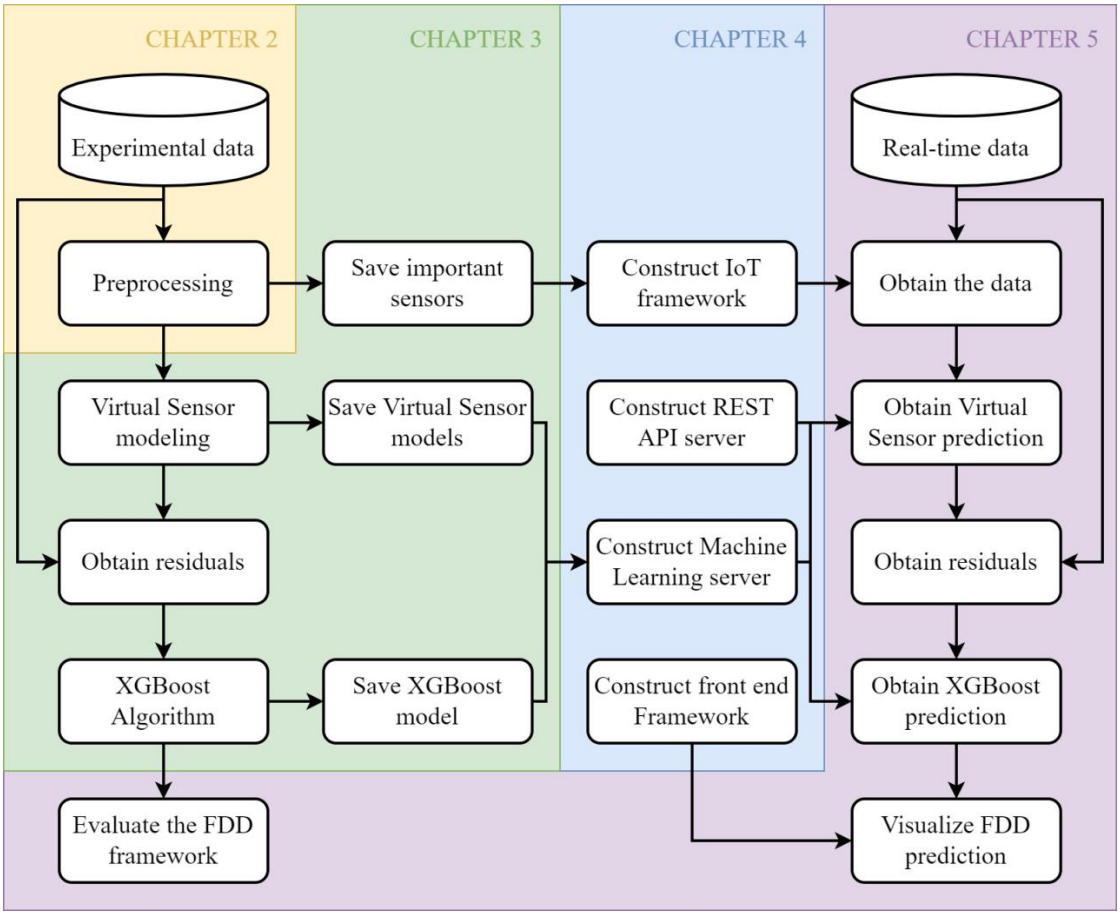


Fig. 1-1 Overall structure of this study

Chapter 2

Experimental procedure and data preprocessing

2.1. Description of the experimental system

The experiments are conducted on an air-cooled screw chiller which contains six groups of condensers and operates in double cycle air cooling operating mode. The chiller used in these experiments is shown in Figure 2-1. The chiller utilizes a semi closed screw compressor and the sliding valve of the compressor regulates the refrigerating capacity of the chiller for different cooling loads. The evaporator of the chiller is shell tube evaporator which employs water as the external heat exchange medium.

Fig. 2-1 The chiller used for experimental data collection

The experimented chiller is stationed in an industrial park in Taicang City, Jiangsu Province, China. The main purpose of the chiller installation is to supply an experimentation setup in the industrial park with constant inlet water temperature. The schematic of the experimental system and the locations of the sensors where the experimental data are collected is shown in Figure 2-2. The sensor data collected from the chiller controller are accessed using Modbus protocol and transferred to an internet of things intelligent terminal, which are explained in chapter 2.3 in detail.

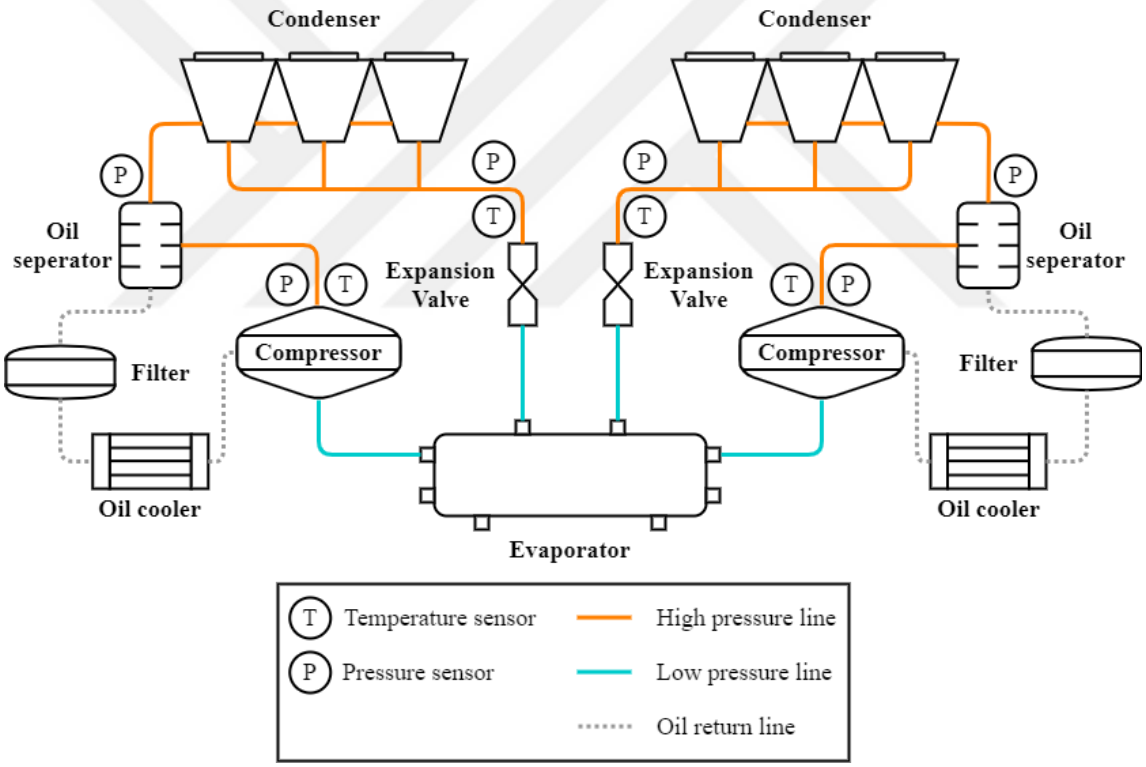


Fig. 2-2 Schematic of the experimental system

2.2. Experimental conditions

During the experiment, the chiller is tested for both fault-free and refrigerant undercharge fault conditions. The fault-free state test conditions simulate the normal operating conditions for the

inspected chiller. To test specific severities of refrigerant undercharge fault, the refrigerant already present in the chiller is first pumped out to vacuum state and then refilled to a set of specified weights in a standard enthalpy difference laboratory. During the experiment, different operating conditions of the chiller are reproduced by adjusting ambient temperature, setting the level of opening of the compressor slider valve and changing the outlet temperature of the evaporator. The sensor data measured during these sets of experiments are gathered from the intelligent terminal, the methodology of which is explained in the next section.

Table 2-1 features the experimental conditions for which the sensor data are collected. For all conditions, the ambient air temperature, the water flow rates and the water outlet temperatures of the chiller are set to the specified levels. By regulating on the compressor slide valve, the load rate of the chiller is fixed to 60%, 80% and 100%. The tested refrigerant charge levels are 100% (normal), 90%, 80% and 70%. The load rates and refrigerant charge levels are tested for all of other experimental conditions. Yet, because of the system limitations, outlet water temperatures and outdoor temperatures are tested for the specific ones shown on the table. In total, a set of 60 experiments are conducted.

Table 2-1 Experimental conditions

Ambient air temperature (℃)	Outlet water temperature of evaporator (℃)	Water flow (%)	Load rate (%)	Refrigerant load (%)
32	5	100%	60/80/100	70/80/90/100
32	9	100%	60/80/100	70/80/90/100

35	7	100%	60/80/100	70/80/90/100
38	5	100%	60/80/100	70/80/90/100
38	9	100%	60/80/100	70/80/90/100

2.3. Collection of sensor data

The intelligent terminal installed on the chiller refers to the Tracer UC800 controller produced by Trane Company [76]. The controller enables the access to the real-time sensor data through the communication interface via the Modicon Communication Bus (Modbus) protocol. Modbus is an application layer messaging protocol that was created by Schneider Electric Company for client/server communication between devices over various networks. In Modbus communication protocol, monitoring computer acts as the master station and the intelligent terminal acts as the slave station, and only the master can initiate transactions (queries). The address of the slave and the function to execute is contained in the query message. The bit length of this message may vary according to the different function codes. During the experiments the real time readings from all sensors are accessed by the monitoring computer through the aforementioned protocol and recorded in the relevant database.

After the successful collection of sensor data, the study is proceeded with the preprocessing to remove any potential noise and abnormality. The preprocessing is an integral part of machine learning methods to improve the quality of the data and the useful information that it contains. Similarly, the proposed method also require a proper preprocessing of the data achieve an efficient training process and to obtain healthy results. The Figure 2-3 presents the three main stages of preprocessing that are used in this study and the following sections examine these stages in detail.

Fig. 2-3 Three main stages of preprocessing

2.4. Data cleaning

In order to minimize the negative effects of the possible repetition, abnormality and noise present in the collected sensor data, an initial filtering is carried out. Besides such sensor level defects, chiller experimentations also typically have system level irregularities such as time variations and delays in the responses of the components to the alternating operating conditions. The time variations can be negated by filtering out the transient data and extracting the steady state sensor data as an input. Traditionally, the steady state detection is performed by setting a fixed threshold value. However, establishing a fixed threshold value might result in missed detection opportunities, a delay in detection or loss of valuable information. Therefore, Moving Window (MW) [77] is selected as the steady state judgement algorithm to filter out the transient data from the dataset. Moving window is used to analyze the dataset with a series of averages of different subsets of the collected data along the time axis. Figure 2-4 illustrates the concept of moving window with a length of n at the time k.

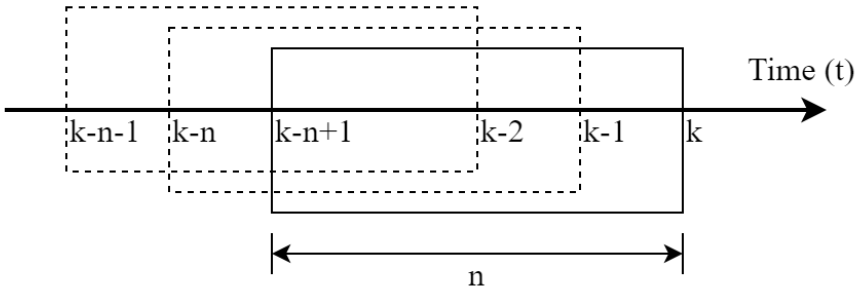


Fig. 2-4 Illustration of moving window

Assuming that a data sequence $\{x_1, x_2, \dots, x_N\}$ is collected in time *k*. A data window of length *n* would be $\{x_{k-n+1}, x_{k-n+2}, \dots, x_k\}$, and the mean of the window can be calculated as:

$$\bar{x}_k = \frac{1}{n}\sum_{i=k-n+1}^{k} x_i = x_{k-1} + \frac{1}{n}(x_k - x_{k-n}) \tag{2-1}$$

And the maximum average deviation for a data in the window is as follows:

$$\delta_{max} = \max |x_i - x_k| \tag{2-2}$$

The squared deviation (v_k) and mean squared deviation of such point within the window can be defined as:

$$v_k = \frac{1}{n}\sum_{i=k-n+1}^{k}(x_i - \bar{x}_k)^2 = \frac{1}{n}\sum_{i=k-n+1}^{k} x_i^2 - \bar{x}_k^2 \tag{2-3}$$

$$v_k = v_{k-1} + \frac{1}{n}(x_k^2 - x_{k-n}^2) - (\bar{x}_k^2 - \bar{x}_{k-1}^2) \tag{2-4}$$

$$\sigma_k = \sqrt{v_k} \tag{2-5}$$

If the data point *k* within the window satisfies the condition $\delta_{max} < 3\delta$ and $\sigma_k < \sigma$,where δ and σ denoting the maximum mean deviation and mean square deviation calculated from the historical data, it is considered to be in a steady state. On the other hand, if these conditions are not satisfied, the data is labeled as transient state and removed from dataset to prevent the training the FDD algorithm with erroneous data.

2.5. Initial sensor data selection

The data collected from chiller controllers generally contain many irrelevant features such as chiller control parameters and temperature set points for the targeted operation. There, it is necessary discard the irrelevant features are from the dataset. Also, to make the proposed

preprocessing method applicable to other chillers, more common and easier to collect features are chosen. Table 2-2 displays the sensor data and their abbreviations that are used as input features in this study.

Table 2-2 Selected sensor data

No	Name of chosen sensor data features	Abbreviation
1	Outdoor temperature	T_{amb}
2	Inlet temperature of chilled water	$T_{cw,i}$
3	Outlet temperature of chilled water	$T_{cw,o}$
4	Chilled water flow	F_{cw}
5	Evaporator saturation temperature	T_{ev}
6	Evaporator refrigerant pool temperature	T_{evp}
7	Evaporation pressure of evaporator	P_{ev}
8	Approaching temperature	T_{ap}
9	Compressor suction pressure	P_{suc}
10	Refrigerant pressure difference	P_r
11	Compressor mass flow	F_{com}
12	Compressor refrigerant discharge temperature	T_{dis}
13	Superheat	T_{sh}
14	Measured compressor current	I_{com}
15	Condensation temperature	T_{cd}
16	Condenser outlet temperature	$T_{cd,o}$
17	Condenser outlet sub-cooling temperature	T_{sc}
18	Condenser pressure	P_{cd}
19	Condenser differential refrigerant pressure	ΔP_{cd}
20	Condenser liquid line temperature	T_{ll}

21	Condenser liquid line pressure	P_{ll}
22	Exhaust air temperature	T_{ex}

2.6. Input feature correlation filtering

There might be some inter-correlations between some of the initially selected sensor data features because of two principal reasons. First, the pressure and temperature readings in the same components have very high chance to be correlated if there is no phase change in the working medium. Thus, when the data is normalized to a zero mean, unit variance and unitless state before training in the subsequent algorithm, the data from two features might become basically the same and one of these would lose significance as an input. Second, since all of the components of the chiller are connected to each other, there might be some relations in the sensor readings from connected components.

If the normalization processes from one or more features yield the same result, the duplicate features must be dropped from the input dataset. Neglecting this filtering might assign some unwanted weight factors to the correlated inputs in the dataset and cause distortion in the predictions of the algorithm. Thus, pairwise comparison of all selected features is necessary to enhance the precision of the model. Statistically, pairwise linear correlations between two features can be found by calculating the Pearson correlation coefficient [78], which is described in the following section.

2.6.1. Pearson correlation coefficient

There are many types of correlation formulas in the literature, which describe the strength of a relationship between two variables. These formulas usually return a value between [-1, 1], where,

a result of -1 represent a strong negative relationship, 0 indicates there is no relationship and 1 represents a strong positive relationship. Figure 2-5 illustrates the relationship between variables with different correlation coefficient values.

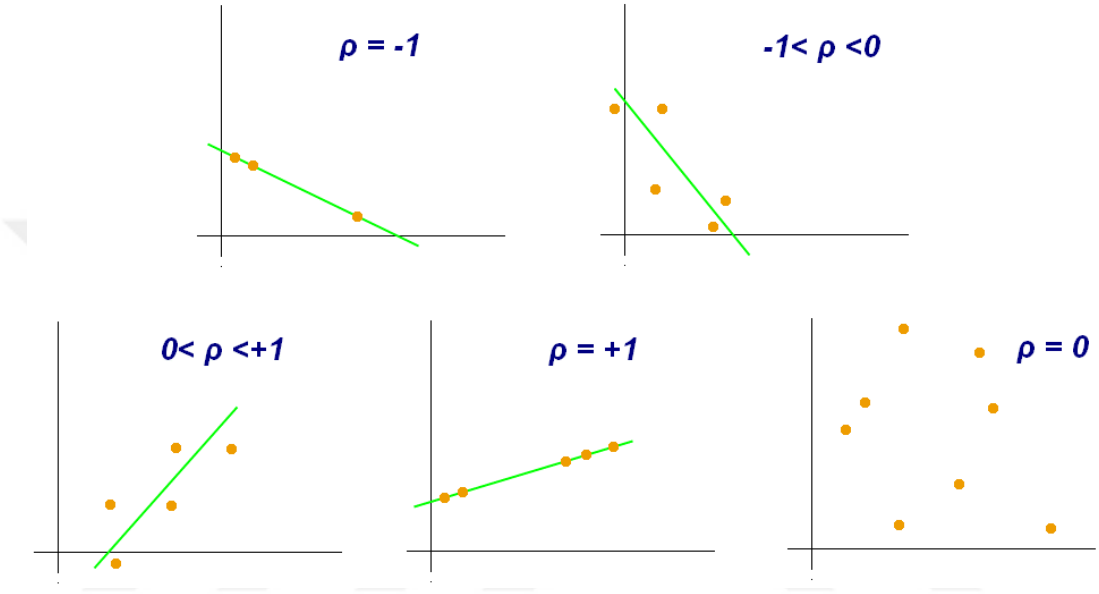


Fig. 2-5 Illustration of data with different correlation coefficients

As can be seen from the graphs, an absolute value of the correlation coefficients represents the strength of the relationship between the variables. In other words, the closer the absolute value of the correlation coefficient, the stronger the correlation between two variables. In statistics, Pearson's correlation coefficient formula, developed by Karl Pearson, is one of the most commonly used formulas to calculate the correlation coefficient, which is also known as, Pearson product-moment correlation coefficient (PPMCC), Pearson's r and the bivariate correlation.

When Pearson's correlation coefficient is applied to a population, it is represented by Greek letter ρ (rho) and referred as population Pearson correlation coefficient. Given the random variable pair (X, Y) the population Pearson correlation coefficient (ρ) is formulated as [78]:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \tag{2-6}$$

where cov represents the covariance, and σ_X, σ_Y represent the standard deviations of X and Y, respectively. The covariance can also be expressed in terms of the mean and expected values of the input variables such as:

$$cov(X,Y) = E[(X - \mu_X)(Y - \mu_Y)] \tag{2-7}$$

where μ_X, μ_Y represent the mean of X and Y, respectively, and E is the expectation. Therefore the equation for $\rho_{X,Y}$ becomes:

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \tag{2-8}$$

If the uncentered moments of the expressions are taken,

$$\mu_X = E[X] \tag{2-9}$$

$$\mu_Y = E[Y] \tag{2-10}$$

$$\sigma_X^2 = E[(X - E[X])^2] = E[X^2] - (E[X])^2 \tag{2-11}$$

$$\sigma_Y^2 = E[(Y - E[Y])^2] = E[Y^2] - (E[Y])^2 \tag{2-12}$$

$$\begin{aligned} E[(X - \mu_X) - (Y - \mu_Y)] &= E[(X - E[X])(Y - E[Y])] \\ &= E[XY] - E[X]E[Y] \end{aligned} \tag{2-13}$$

The formula for population Pearson correlation coefficient (ρ) becomes:

$$\rho_{X,Y} = \frac{E[XY] - E[X]E[Y]}{\sqrt{E[X^2] - (E[X])^2}\sqrt{E[Y^2] - (E[Y])^2}} \tag{2-14}$$

If the Pearson's correlation coefficient is applied for a sample, the estimated covariances and variances are substituted into the above formula and the Pearson's correlation coefficient $r_{x,y}$ for the pairs $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ is defined as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x - \bar{x})(x - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x - \bar{x})^2 \sum(x - \bar{y})^2}} \quad (2\text{-}15)$$

where n is the sample size and $\bar{x}$ and $\bar{y}$ are the sample means. The equation 2-10 is used in this research to calculate the correlation between the input data from two different sensor to obtain the knowledge about the relationship between sensor pairs. The interpretation of the resultant data from the correlation analysis is given in the section below.

2.6.2. Analysis results

From definition Pearson correlation coefficients of the feature pairs are resulted in the range of [-1, 1], where values closer to 1 and -1 signify perfectly positive and negative linear relationships, respectively. Since a correlation value of 0 signifies no linear relationship between features, it can be said that such two features contain independent and valuable information as inputs. A heatmap diagram of the absolute values of Pearson correlation coefficients of the initially selected features are given in Figure 2-6.

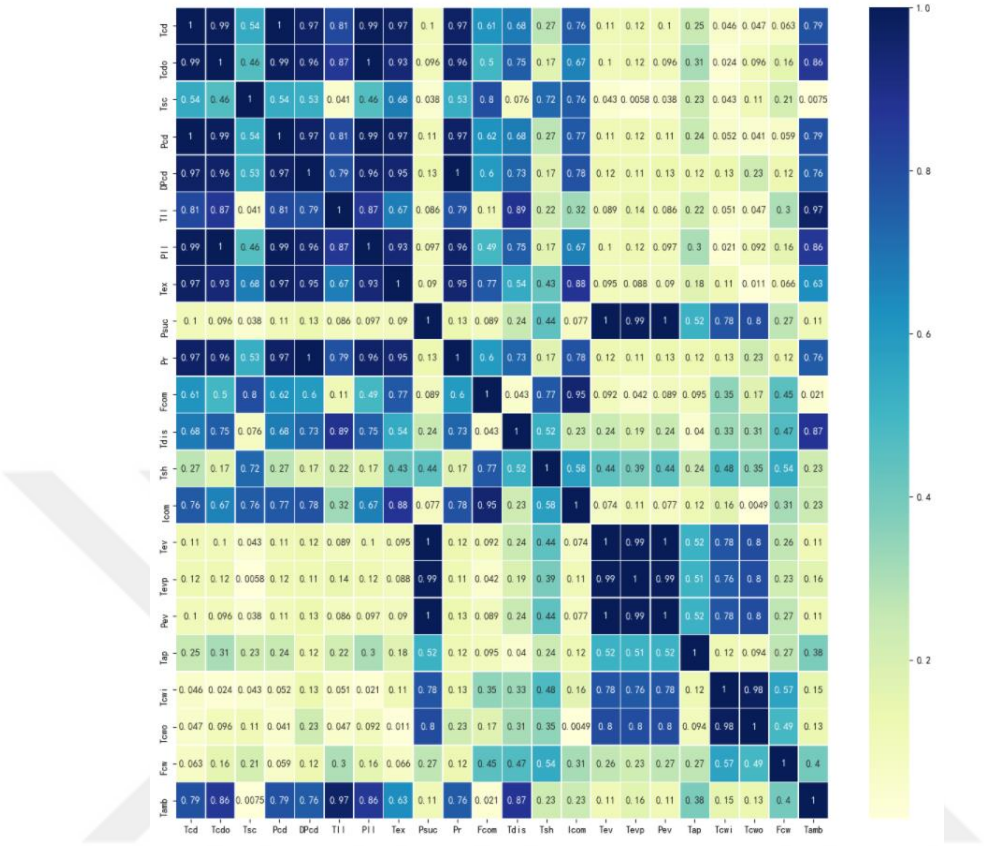


Fig. 2-6 Heatmap diagram of initial correlation coefficients

To eliminate the features which contain duplicate information, pairs with an absolute correlation coefficient value bigger than 0.95 are chosen for examination. The comparison graphs of Figure 2-7 shows the normalized data from some of the highly correlated variables. For every correlated feature pair, the decision to drop which feature is made based on expert knowledge. It should be noted that although the $T_{cw,i}$ and $T_{cw,o}$ pair had a correlation coefficient of 0.98, an exception of keeping both features in the dataset has been made to prevent a loss of critical diagnosis information. As a result of the correlation filtering process, the data from 11 features of T_{amb}, T_{cd}, T_{sc}, F_{com}, T_{dis}, T_{sh}, T_{ev}, T_{ap}, $T_{cw,i}$, $T_{cw,o}$ and F_{cw} are selected as the final inputs to the proposed method.

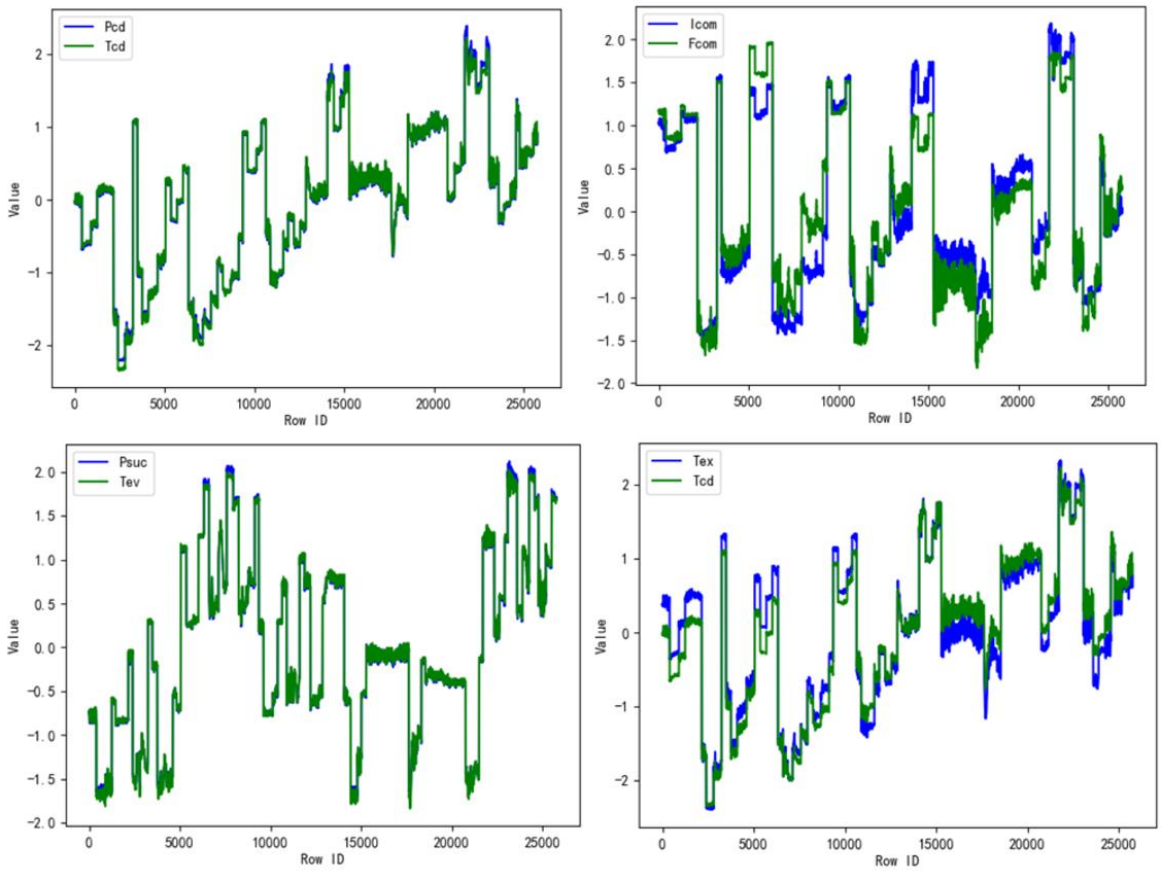


Fig. 2-7 Comparison graphs of some highly correlated data

2.7. Summary of data collection and preprocessing

The input data for the fault detection and diagnosis algorithm is collected from the experimental chiller setup under various fault experiment conditions as mentioned in the sections 2.1 and 2.2. The data gathered from the intelligent terminal of the chiller usually contains irregularities, defects, state transitions and many duplicate data which need to be corrected. Otherwise, such data will lead the main FDD algorithm to learn the incorrect data and therefore conduct an improper diagnosis on the faults. However, if this filtering of the sensor data is not done properly, valuable information in the dataset may also be lost and the prediction accuracy of

the FDD algorithm might decrease. In this research, three main stages of preprocessing are conducted as shown in Figure 2-8.

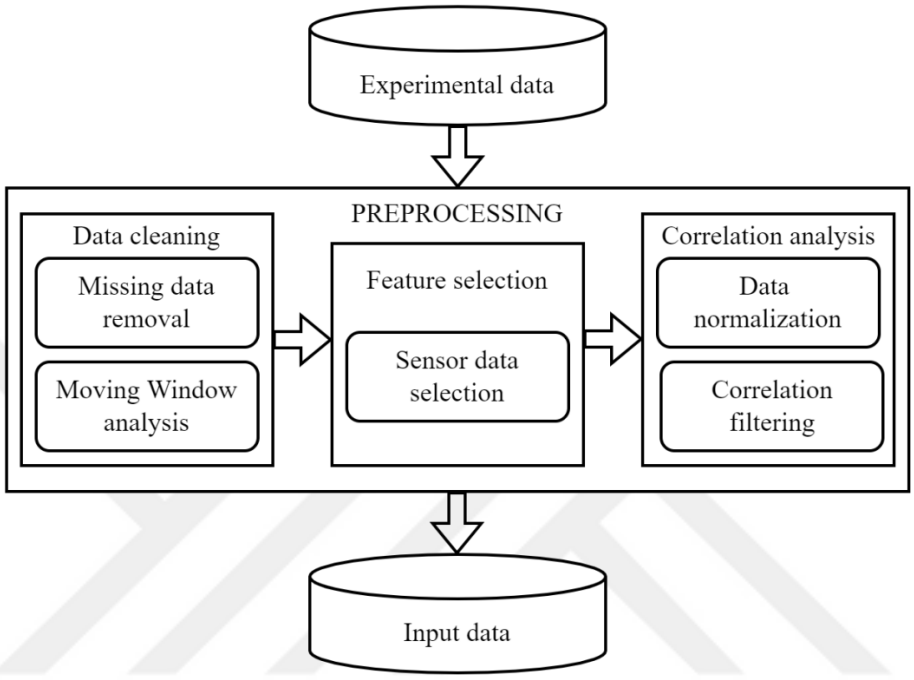


Fig. 2-8 Overall flowchart for preprocessing

In the data cleaning stage, the irregularities in the raw data is targeted to be filtered out from the dataset. The data is separated into two subsets as transient and steady data and the transient state data is removed. In the sensor data/feature selection stage, the sensors that historically contain valuable information are selected by expert knowledge to discard the unnecessary readings in the dataset such as temperature set-points and irrelevant chiller control information. In the correlation filtering stage the duplicate sensor data is removed from the dataset to prevent a bias or multiplier in the training of the FDD algorithm. In short, this section presented the methodology for filtering and processing of the large amount of data collected from the sensors on the chiller to provide the most valuable information to the subsequent FDD algorithm.

Chapter 3

Fault Detection and Diagnosis methodology

3.1. K-fold cross validation

K-fold cross validation/testing is used for both the construction of Virtual Sensor models and the validation of the overall FDD methodology in this research as a testing method, thus, the underlying principles of k-fold cross validation methodology is briefly introduced in this section. The k-fold sampling has many advantages over random sampling when testing the FDD algorithms. Even though the random sampling is the primary testing method of the FDD algorithms in the literature, such testing method have major defects in detecting the overfitting in the model when the input dataset have clusters.

As explained in the Chapter 2, the experiments on the chiller are conducted in 60 distinct experimental conditions and only the steady state data are filtered in the following stages as inputs to the FDD algorithm. Such processes cause the input dataset to contain internal clusters since there were many data points collected for these 60 steady state conditions. These steady state data forms 60 distinct mini data clouds in the experimental conditions input space. When the testing data is randomly selected, it is almost for sure that there will be a testing and training point existing in the same cloud. Thus, random testing with such conditions cannot detect any overfitting of the model and it will only provide the same information as training loss. Figure 3-1 illustrates an example dataset where the machine learning model with clustered input data results in an overfitting model and random testing fails to detect this overfitting.

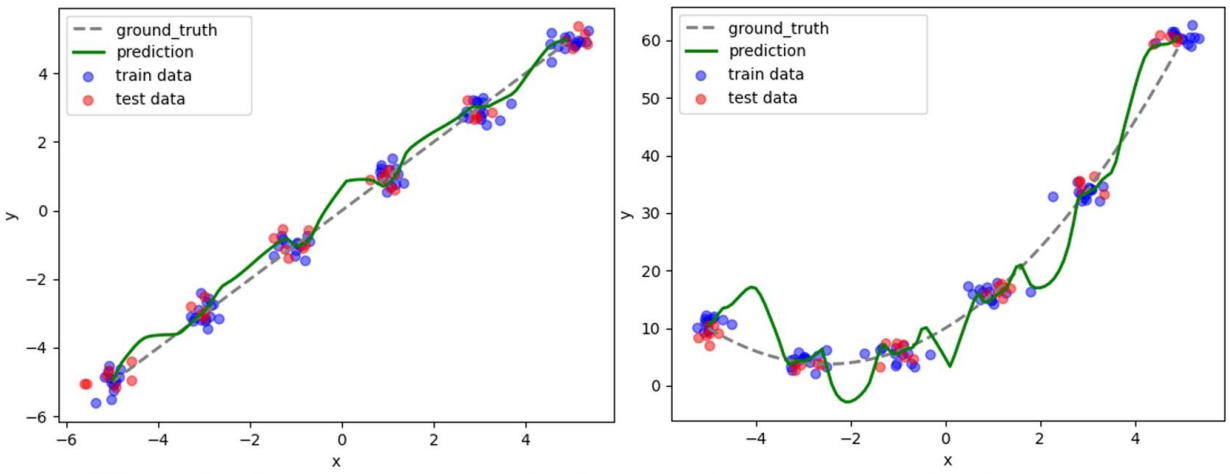


Fig. 3-1 Random sampling and overfitting in a clustered dataset

The above figures clearly illustrate an overfitting of the prediction model and the allocation of testing and training data on a clustered dataset. As can be seen from both figures, the resulting testing scores will be mistakenly very good since the models have sufficient performances within every single cluster. The case of training FDD models with the steady state data and random testing is very similar to these examples. The clouds data clouds will be formed for every single experimental condition and random testing would fail to detect any overfitting of the FDD model. However, since the inputs and outputs of these models are multi-dimensional, it is not possible to visualize the cloud formation for the data. Fortunately, even though the visualization of the clusters are not instantly possible for the multidimensional data, the overfitting of the models can still be statistically tested by using k-fold cross validation as the testing method.

K-fold cross validation is a resampling procedure used to evaluate machine learning models where the sample size of the data is limited. The parameter k, as included in the name, refer to the grouping number that the data samples would be split into. If the value of the k is determined prior to testing operation, such as if k=9, the model might also be referred as 9-fold cross validation.

The k-fold cross validation is primarily used to estimate the prediction capacity of the models on the previously unseen input data sections. This is achieved by limiting the training data samples in a particular way to estimate the models performance in general. In such procedure, the predictions are only made for these regions of data that is not used during the training of the model. This methodology owe its popularity to its simplicity and its ability to achieve less biased and less optimistic estimates of the model prediction capacity compared to other testing methods such as random testing.

The following is the general procedure for k-fold cross validation:

1) Randomly shuffle the dataset.

2) Split the dataset is into k groups.

3) For each group:

3.1) take the chosen group as the test dataset

3.2) take all remaining groups as the training dataset

3.3) fit a model on the training set and test the model on the test set

3.4) record the test score

4) Summarize the prediction capacity of the model using the previously collected test scores

Assigning each data point to their respective group prior to the k-fold testing algorithm and ensuring that these points remain in their respective group until all of the iterations are completed is a crucial factor in this methodology. This condition assures that each sample is used in testing 1 time and used in training the dataset k-1 times, thus produces an unbiased result. Figure 3-2 demonstrates an example training and testing set allocations on k-fold cross validation algorithm

where the 6-fold, 3-fold, and 2-fold cross validation methodologies for the same dataset are displayed from left to the right of the figure.

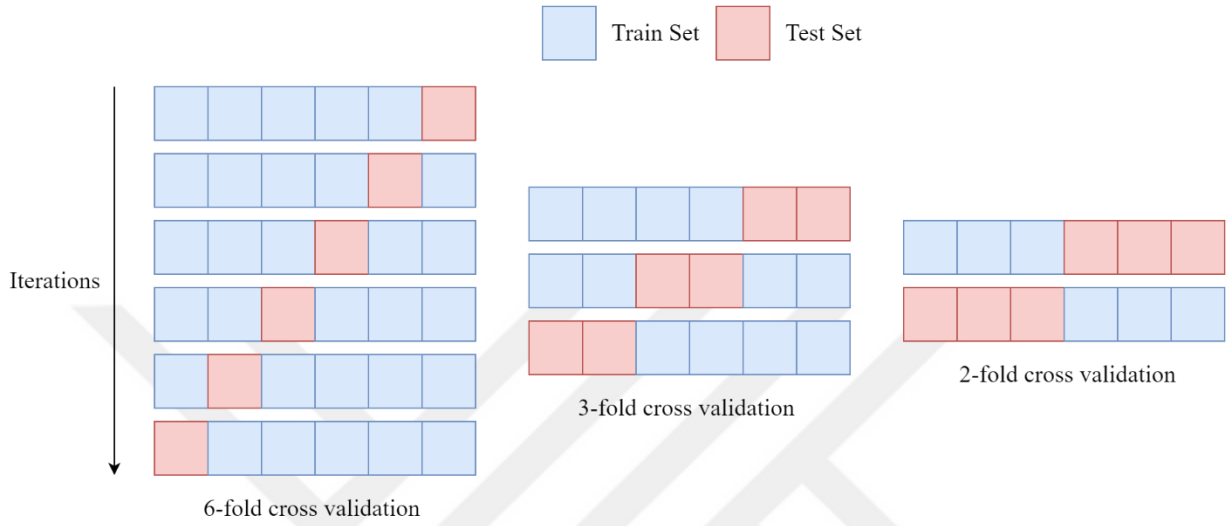


Fig 3-2 Train and test set allocation on k-fold cross validation

It should be noted that, for the case of this study, the initial random shuffling of the data is neglected because of the aforementioned defects of random sampling. Moreover, since clusters are assumed to be existent on the experimental condition inputs, the total points of clusters are assumed to be 60. Varying numbers of combinations of these clusters are administered as k-fold testing sets for both construction of virtual sensors and final testing of the FDD algorithm. The specific numbers (k) of k-fold test are explained with their specific reasoning in the following chapters.

3.2. Base logic of the Virtual Sensors

Flag variables are commonly used to control the initiation of subsequent logic trees or algorithms in rule-based and hybrid FDD methodologies. Typically, compressor power consumption is chosen as the main flag variable since it is intuitive that the power consumption of the chiller compressor would diverge from the normal operating condition values in case of an

existing fault in the chiller system. When the flag variable is determined, the next step is to detect the abnormalities in the data. In general, a fault-free model for the variable is constructed from historical data and then the difference between the fault-free model prediction and the actual value is calculated. If this value is above a certain threshold, the flagging condition is activated for the rest of the algorithm. Moreover, more than one flag variables can be chosen for the algorithm depending on the diagnosis requirements. In other words, there must be three aspects that must be determined in prior to use flag variables in fault diagnosis algorithm. First, the variables to be used as flags must be chosen. Second, the method to construct fault-free models must be specified. Third, the threshold to detect abnormalities must be established.

The proposed method does not utilize any flag variables or logic trees, however, virtual sensor residuals are based on the same core principles that the flag variables provide to FDD algorithm, which is the difference of normal condition prediction and the actual value. Instead of using a single flag variable, normal condition based virtual sensors are constructed for each input feature as linear combinations of the remaining sensor values. In other words, a virtual copy of each sensor is created based on the expectations from the remaining sensor readings of the system in fault-free operating conditions. Each virtual sensor is k-fold cross tested to determine whether that feature could be successfully modeled by the remaining sensors in the chiller. If the modeling performance of the virtual sensor is decent, the difference of the actual data and the virtual sensor is used as an input. If the modeling performance of a single feature is not sufficient, that virtual sensor is discarded and only the actual data is used for further sections. The differences between the processes of deciding flag variables and constructing virtual sensors are shown in Figure 3-3.

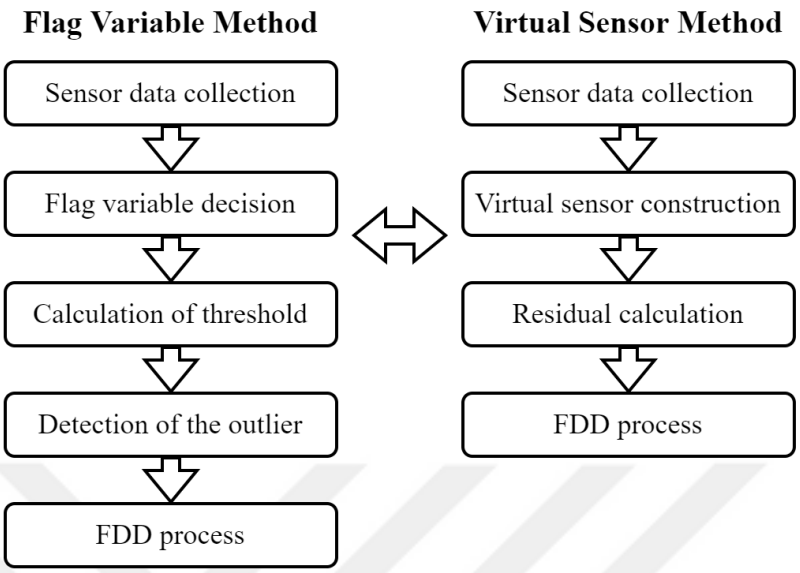


Fig. 3-3 Differences between flag variable decision and virtual sensor construction

This approach phases out two hyperparameters that must be defined in the methods using flag variables. First, since the selection of virtual sensors is based on modeling performance, there is no prior selection of variables for which the virtual sensors would be constructed. Second, the differences between measured values and virtual sensor predictions are directly provided to the following diagnosis algorithm, thus, there is no need to define an abnormality threshold.

The simplified version of the overall method is illustrated in Figure 3-5. First, the sensor data is preprocessed by the methods defined in the previous chapters. Second, the data is allocated into training and test sets. This sampling is conducted several ways such as random, 9-fold, 15-fold and 23-fold sampling to test the robustness of the algorithm. Figure 3-4 illustrates the input space sampling methodology for k-fold cross validation method used in this study.

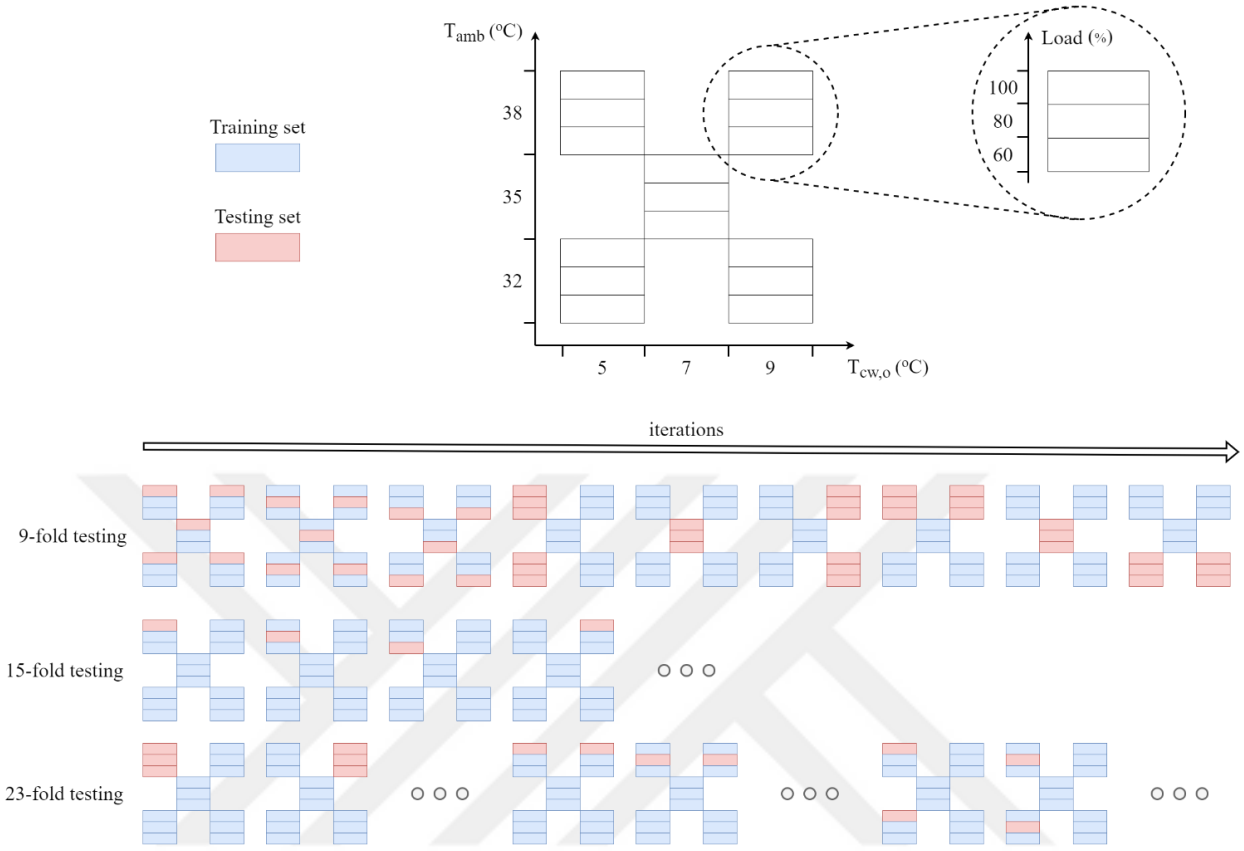


Fig. 3-4 9, 15 and 23-fold sampling for input space

As mentioned in section 2.2 the experimental conditions were four dimensional containing various refrigerant load, ambient temperature, chiller load and leaving water temperature values. For all k-fold tests different combinations of three dimensions such as ambient temperature, leaving water temperature and chiller load are sampled. In 15-fold cross validation, train-test division is made including combinations of all three dimensions, which contains 15 distinct sets in total. For instance, ambient temperature of 32, chiller load of 100% and leaving water temperature of 5 is designated as test set and the remaining data is chosen as training set. Then, the iteration is proceeded with changing the test set to the next combination and the training set is assigned as the respective remaining data. This testing methodology is repeated 15 times for all distinct sets and averages for 15-fold tests are obtained.

In 23-fold testing, two value combinations of these three dimensions are allocated as test sets. For example, ambient temperature of 38 and leaving water temperature of 9 are selected as test set and the remaining data is chosen as training set. For the two value combinations of the three dimensional experimental conditions, there are 9 + 9 + 5 = 23 unique sets in total. Likewise, the averages of 23 tests are obtained as the final 23-fold test results.

For 9-fold tests, each experimental condition is tested separately. For example, chiller load of 80% is selected as the test set and the chiller loads of 60% and 100% are chosen as the training set. Similarly, 3 + 3 + 3 = 9 tests are conducted in total to calculate the averages. In addition to these, the dataset is also randomly sampling to train and test sets with %80 - 20% testing ratio, respectively. Altogether, the algorithm is tested with 4 different sampling methods. The following figure illustrates a simplified flowchart for the proposed method.

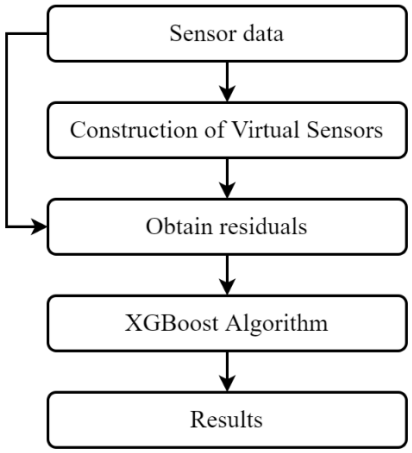


Fig. 3-5 Simplified flowchart of the proposed methodology

As illustrated Figure 3-5, the fault-free training data is initially utilized for constructing virtual sensor. A detailed explanation on the modeling of virtual sensors is given in the section 4.1. When a virtual sensor is created, predictions for both normal and faulty data are made by the virtual sensors. Then, all of the virtual sensor predictions are subtracted from the actual sensors' readings

and virtual sensor residuals are obtained. These residuals are supplied as training data to the XGBoost algorithm and the model parameters are saved. Finally, the test data is similarly utilized to obtain virtual sensor residuals from previously trained virtual sensor models and the saved XGBoost algorithm is tested to obtain the final results.

3.3. Modeling of Virtual Sensors

3.3.1. Mathematical formulation

A virtual copy of a particular sensor is primarily a regression model for the original sensor based on the remaining sensors in the chiller system. The training of the virtual sensor is performed using the fault-free conditions to extract the difference between normal condition expectation and the actual data. It should be noted that, not all of the remaining sensors are included for the virtual sensors' training and the sensors that are used for the models are precisely selected. Furthermore, if the virtual sensor modeling performance is not acceptable, the model output is fixed to zero, thus the residual would return only the actual value .The proposed virtual sensor models can be mathematically expressed as the following:

$$\hat{f} = A\,(\theta^T f + d) \tag{3-1}$$

where $\hat{f}$ represents virtual sensors (features) in the vector form, A is decision matrix, θ is coefficient matrix, f is the original sensors and d is the bias vector. Explicitly:

$$\begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \vdots \\ \hat{f}_n \end{bmatrix}$$

$$= \begin{bmatrix} a_1 & 0 & \dots & 0 \\ 0 & a_2 & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & a_n \end{bmatrix} \left(\begin{bmatrix} 0 & b_{2,1}c_{2,1} & \dots & b_{n,1}c_{n,1} \\ b_{1,2}c_{1,2} & 0 & & \vdots \\ \vdots & & \ddots & b_{n,n-1}c_{n,n-1} \\ b_{1,n}c_{1,n} & \dots & b_{n-1,n}c_{n-1,n} & 0 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} \right) \quad (3\text{-}2)$$

In the expression, $\hat{f}_i$ is a single virtual sensor, a_i is a binary cancellation parameter, $b_{j,i}$ are binary selection parameters for the remaining sensors, $c_{j,i}$ are the regression coefficients, f_j are the original sensors and d_i is the bias. For a single virtual sensor, the formula is reduced to:

$$\hat{f}_i = \begin{cases} a_1(0f_1 + b_{2,1}c_{2,1}f_2 + \cdots + b_{n,1}c_{n,1}f_n + d_1) & if\ i = 1 \\ a_n(b_{1,n}c_{1,n}f_1 + \cdots + b_{n-1,n}c_{n-1,n}f_{n-1} + 0f_n + d_n) & if\ i = n \\ a_i(b_{1,i}c_{1,i}f_1 + \cdots + 0f_i + \cdots + b_{n,i}c_{n,i}f_n + d_i) & otherwise \end{cases} \quad (3\text{-}3)$$

Since the parameters of a_i and $b_{j,i}$ are defined as binary, the formula for $\hat{f}_i$ reduces to a linear regression of some form after the initialization of parameters a_i and $b_{j,i}$. The weight coefficients $c_{j,i}$ and bias d_i are calculated by minimizing mean-squared error function.

Prior to the assignment the cancelation parameter a_i, the reduced linear regression equation must be constructed. Thus, the values for $b_{j,i}$ must be initially determined to calculate the coefficients of $c_{j,i}$. $b_{j,i}$ parameters are initialized with a for-loop through reducing the total number of $b_{j,i}$ parameters and then by conducting grid search. To reduce to total number of $b_{j,i}$ parameters, the specific $b_{j,i}$ value that would be assigned as zero in each iteration must be figured out. This

requires a sequence of parameters specifying the next $b_{j,i}$ that would be dropped out from the formula in the next iteration to be used in the for-loop. This sequence is obtained by the backward elimination algorithm and then provided to the for-loop. The separate calculation logic of virtual sensor models is given in the Figure 3-6 and the following section explains the working principles of the backward elimination algorithm in detail.

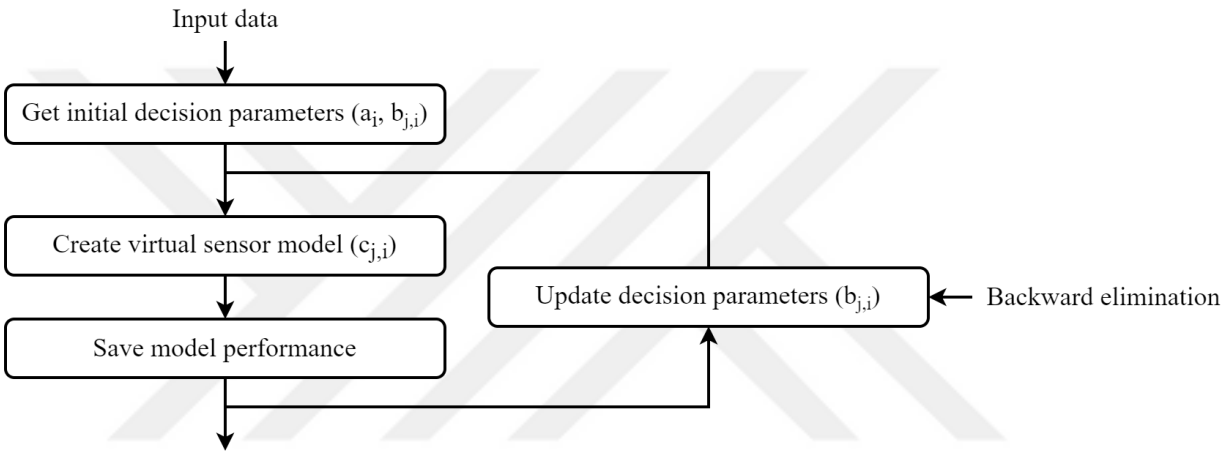


Fig. 3-6 Mathematical calculation flowchart of virtual sensor models

3.3.2. Backward Elimination algorithm

The main idea of backward elimination algorithm is to sort the remaining sensors according to the information that they contain about the inspected sensor. A detailed schematic for the backward elimination algorithm that is used for this purpose is given in Figure 3-7. The algorithm returns a sequence that sorts the remaining sensor by their information potential from lowest to highest. The backwards elimination algorithm is composed of the following steps.

First, the algorithm selects the normal operating condition data from the training dataset. Second, a for-loop is started for each feature in the dataset. For each iteration, the selected feature is the target and the remaining features are the inputs. Hence, the dimension of this input data (n)

is one less than the original training data dimension (m) at this stage ($n = m - 1$). Third, a secondary for-loop is initiated to iterate though this (m-1) dimensions in the algorithm, reducing the feature dimension each iteration. Fourth, a third for loop is started to discard a single feature from the input dataset for each iteration. Therefore, the dimension of input features is again reduced by one ($n = m - 2$). Now this input data with the absence of the discarded feature from the third for-loop is used to construct a linear regression model and tested with random sampling to predict the selected feature from the primary for-loop. After the test results are obtained, the mean-squared error of the prediction is saved, and the discarded feature is restored to the dataset.

Then, the third for loop proceeds with discarding the second feature and repeating the same steps. When the third for-loop ends, the saved MSE data would demonstrate the m - 2 dimensional modeling performance for the selected feature in absence of each dropped feature. The dropped feature with lowest MSE score signifies that feature contains the least information about the inspected sensor, thus the lowest MSE feature is saved in the removal sequence and removed from the input dataset reducing the feature dimension to m - 2. The secondary for loop advances to the next iteration for m - 2 dimensions. Again, the third for loop tests the models for m - 3 dimensions and records the lowest MSE features. The secondary for-loop iterations are carried out similarly until there is no feature left to discard and model the selected feature. Ultimately, a sequence of remaining features sorted in the order of lowest to highest information about the selected feature is returned. The same logic is repeated for each feature in the primary for loop iterations. In conclusion, the backward elimination algorithm produces m removal sequences about m features in the training dataset. An illustration of the usage of backward elimination algorithm to obtain the dropping sequence is given in Appendix-1.

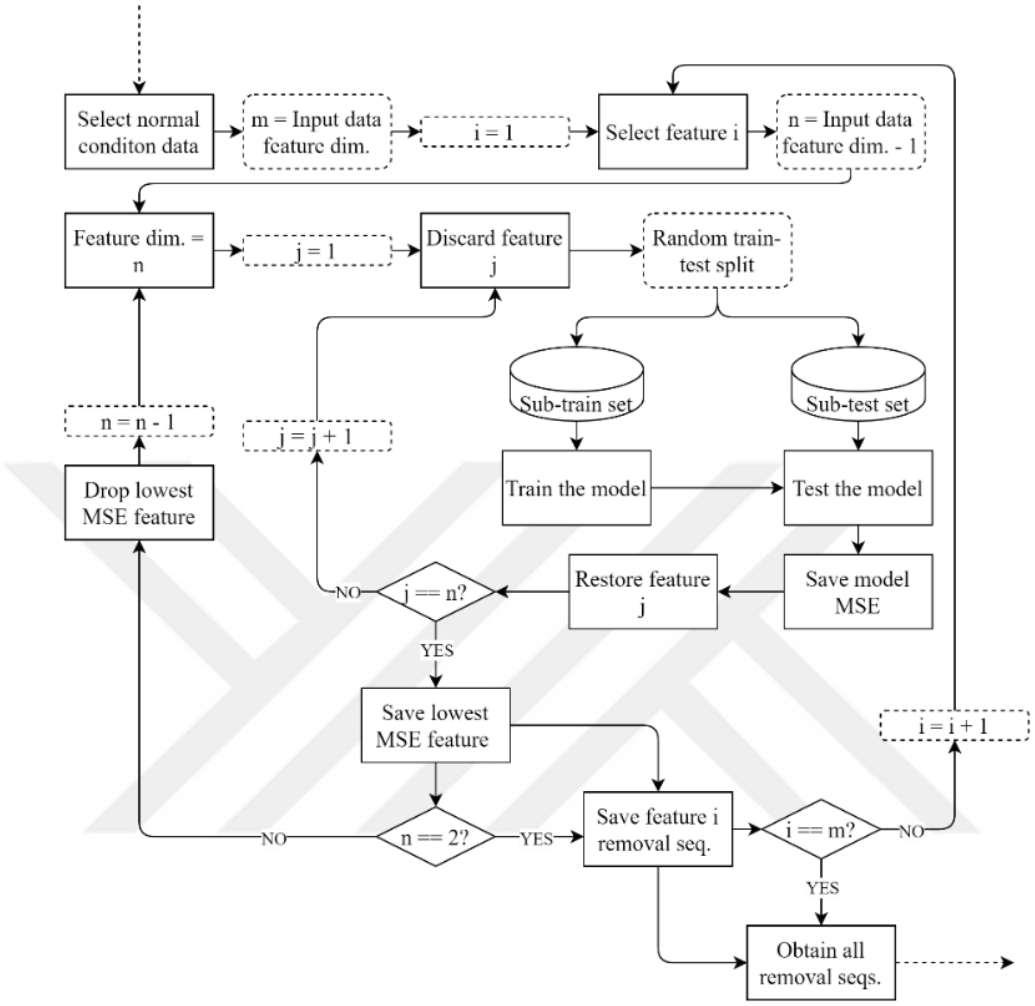


Fig. 3-7 Backward elimination algorithm

3.3.3. Virtual Sensor model construction

When the removal sequence information is obtained the main virtual sensor model construction algorithm is initiated. First, the fault free conditions are selected from the training data. Second, similarly to the backward elimination algorithm, first for-loop iterating through each feature is launched. Third, a secondary for-loop is started to iterate from the feature dimension *m* - 1 to 1 and a linear regression model is constructed in each iteration step. The model is 9-fold sampled to be trained and tested, as explained in the previous sections. The average MSE values of k-fold

tests are recorded and the iteration step is concluded. To reduce the input dimension of the models, the information of which feature to be dropped out of the dataset is obtained from the removal sequence returned from the backward elimination algorithm output. The next iteration of the secondary for-loop, which has feature dimension of m - 2, is initiated with the dropping of first feature supplied from the sequence. These iterations are continued until the model with input feature dimension 1 is tested. An illustration for the virtual model construction iterations for a single sensor and selection of the lowest MSE model is shown in Figure 3-8.

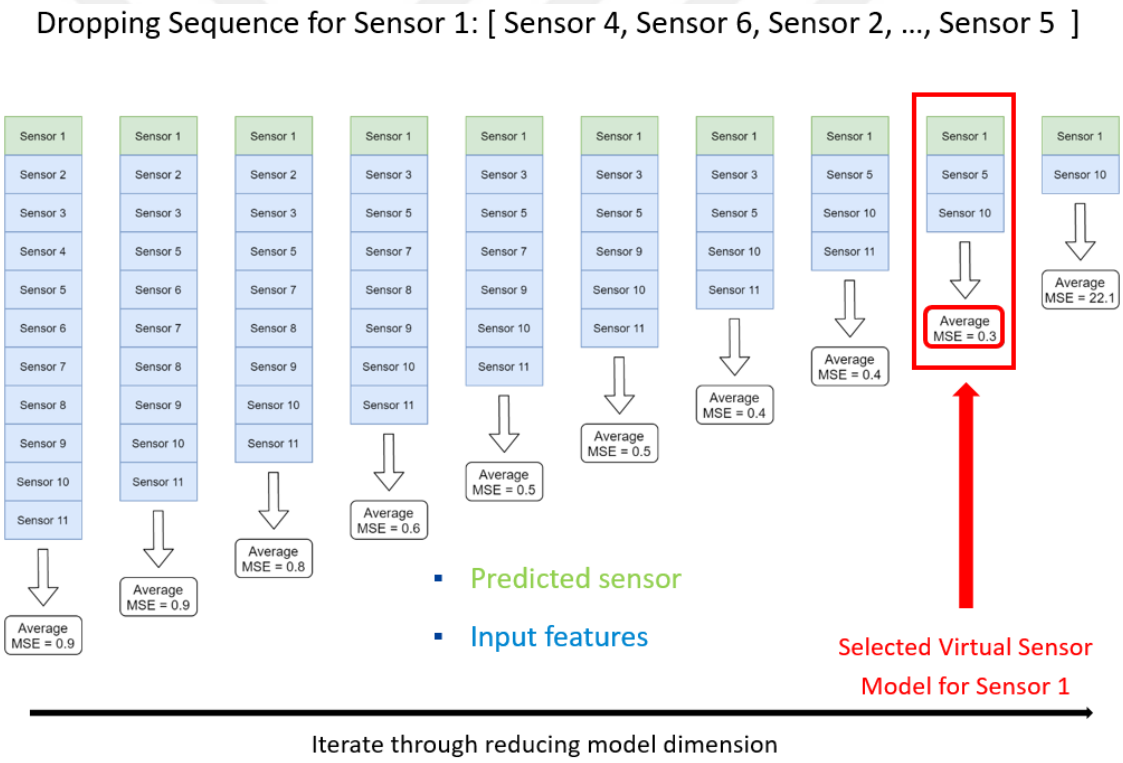


Fig 3-8 Iterations for constructing virtual model for a single sensor

When the secondary for-loop is concluded, the models of different dimensions are compared and the minimum MSE model is selected as the best fit for the primary for-loop selected feature. This step determines the coefficients $b_{j,i}$ specified in Equation 3-2. Then, the chosen model is trained and tested with random sampling, and the regression coefficients $c_{j,i}$ and R2 score of the

model is saved. If the R2 score yields a value higher than 0.95, the model is considered to be a well fit for the selected feature and regression coefficients are saved as the virtual sensor model. If the R2 score is less than 0.95, the model is evaluated as unsatisfactory and the coefficients of the model are saved as zero for virtual sensor output. This stage means that the coefficient a_i is also assigned. The same process is repeated for all features in the training dataset and to construct all of the virtual sensor models. The overall schematic that presents steps employed to determine the coefficients a_i and $b_{j,i}$ is shown in Figure 3-9.

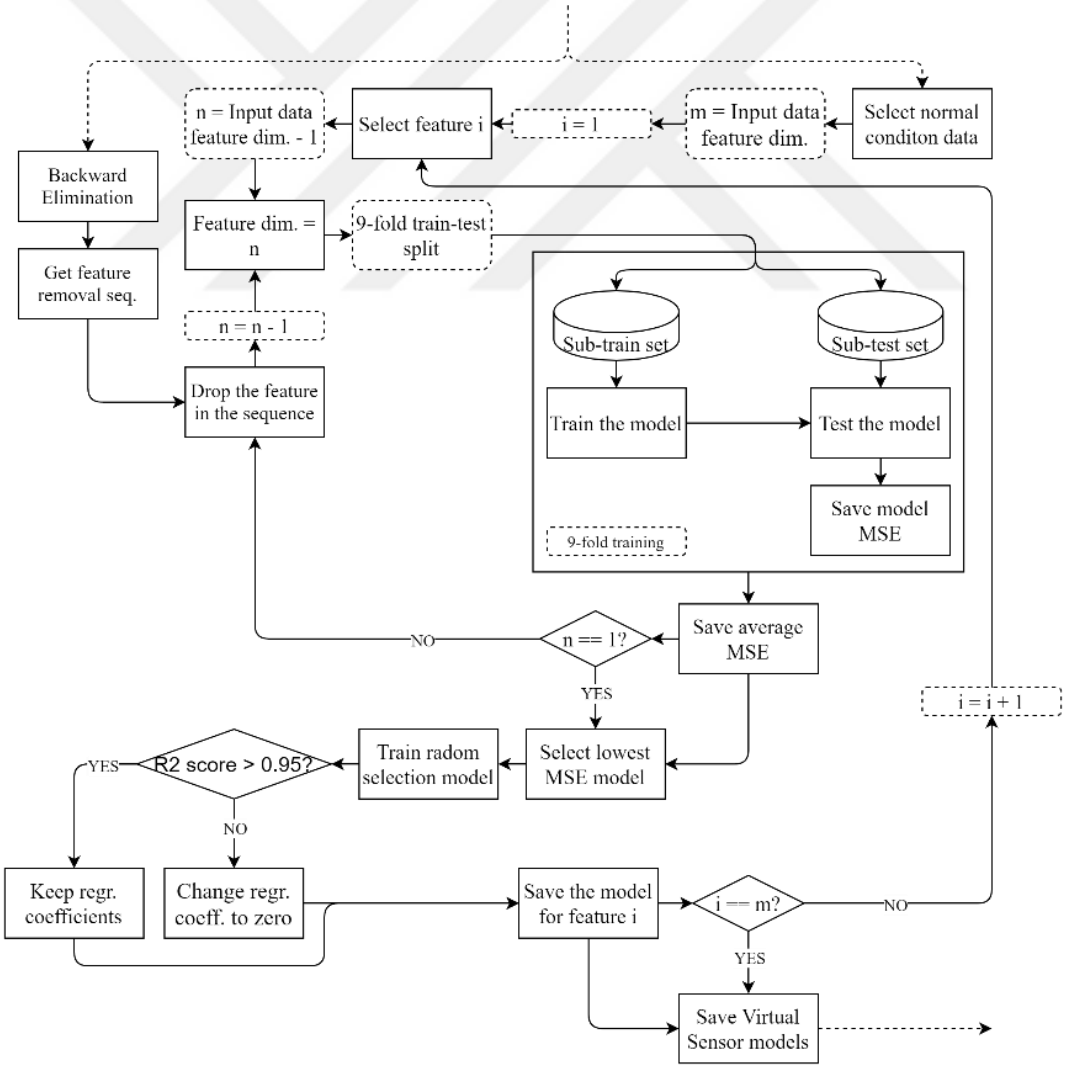


Fig. 3-9 Virtual Sensor model construction flowchart

3.4. Extreme Gradient Boosting algorithm

Artificial neural networks currently outperform all other frameworks of algorithms in prediction problems involving unstructured data, such as images, texts, etc. On the other hand, if small-to-medium size structured/tabular are used for regression/classification tasks, decision tree based algorithms generally perform better. Figure 3-10 displays the evolution of tree-based machine learning algorithms over the years and Table 3-1 summarizes the working principles of these algorithms [79-84] and Table 3-2 provides brief a comparison.

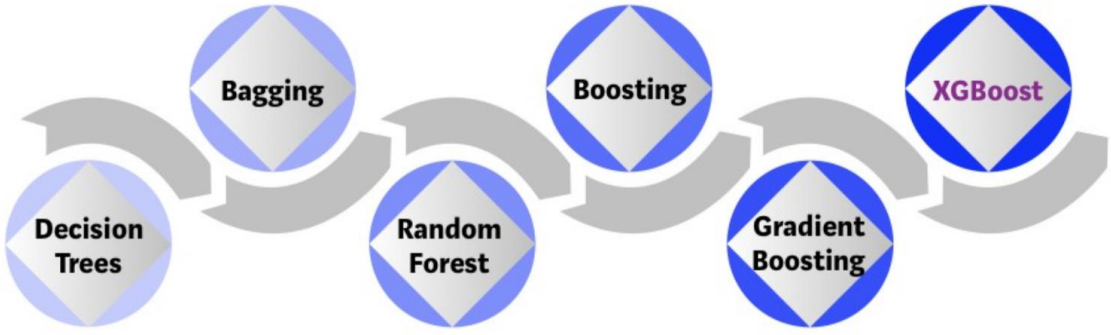


Fig. 3-10 Evolution of tree based machine learning algorithms

Table 3-1 Descriptions of tree-based machine learning algorithms

Algorithm	Description
Decision Trees	Possible solutions to a decision based on certain conditions is represented in a graphical format.
Bagging	Bagging is also known as bootstrap aggregating. It is an ensemble algorithm that combines predictions from multiple trees by a majority voting mechanism.
Random Forest	A subset of features from a bagging-based algorithm are selected in random to build a final forest or collection of decision trees.

Boosting	Sequential models are built by minimizing errors from previous models. During this stage the influence of high performance models are increased (or boosted).
Gradient Boosting	A gradient descent algorithm is employed to minimize the errors in sequential models.
XGBoost	An optimized Gradient Boosting algorithm with parallel processing where overfitting and bias is avoided using tree-pruning, handling of missing values and regularization.

Table 3-2 Comparison of tree-based machine learning algorithms

Algorithm	Weakness	Strength	Publication
Decision Trees	Low accuracy	Faster training time	1986
Bagging	Long training time	Wide range of application	1979
Random Forest	Long training time	High accuracy	2001
Boosting	Sensitivity to outliers	High accuracy	1996
Gradient Boosting	Overfitting	High accuracy	2001
XGBoost	Requires hyperparameter optimization	High accuracy without overfitting	2016

The Extreme Gradient Boosting (XGBoost) algorithm is a custom interpretation of gradient boosting algorithm, which is an ensemble machine learning method that can be used to conduct both classification and regression tasks. Ensemble methods are one of the sub-classes of decision tree models. In such methods, a new tree is added to the ensembles per iteration to increase the accuracy of previous predictions using the previous error residuals. This approach of ensemble

models is referred as boosting. Also, since the models employ a gradient descent algorithm to minimize the chosen loss function, the learning technique is called gradient boosting. The XGBoost algorithm is initially developed by Chen et al. [84] and the predictions of the algorithm can be formulated as following:

For a given data set containing *n* examples and *m* features $\mathcal{D} = \{(x_i,\ y_i)\}$ ($|\mathcal{D}| = n, x_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$), *K* additive functions are used in the tree ensemble model to predict to output.

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F} \tag{3-4}$$

where $\mathcal{F} = \{f(x) = w_{q(x)}\}(q : \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T)$ is the regression trees space, which is also known as CART. In the equation each f_k represents to an independent tree structure q with leaf weights *w*. To learn the function set used in the model, the following regularized objective function is minimized.

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_t) \tag{3-5}$$

where l represents the loss function that measures the difference between the actual and predicted value, and Ω symbolizes the regularization function which is given as:

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{3-6}$$

where *T* is the number of leaves in the tree.

It can be concluded from the formula that higher number of leaves and larger leaf weights are penalized by the regularization function. Since the ensemble model used in equation used in

equation 3-5 cannot be optimized in traditional Euclidean space and instead trained in an additive manner, the objective function must be modified. When $\hat{y}_i^{(t)}$ is defined as the prediction at i^{th} instance at iteration t, f_t should be added to minimize the following objective:

$$\mathcal{L}^{(t)} = \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) \tag{3-7}$$

which means that the f_t that improves the model the most (according to the equation 3-5) is greedily added. The objective in general settings can be optimized via using second-order approximation.

$$\tilde{\mathcal{L}}^{(t)} \simeq \sum_{i=1}^{n} \left[l\left(y_i, \hat{y}_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{3-8}$$

where $g_i = \partial_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$ and $h_i = \partial^2_{\hat{y}^{(t-1)}} l\left(y_i, \hat{y}_i^{(t-1)}\right)$ represent the first and second order gradient statistics of the loss function. The following simplified objective can be obtained by removing the constant terms at step t.

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{3-9}$$

By defining $I_j = \{i | q(x_i) = j\}$ as the instance set of leaf j. Equation 3-9 can be rewritten by expanding Ω as follows:

$$\begin{aligned} \tilde{\mathcal{L}}^{(t)} &= \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \gamma \mathrm{T} + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[\left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma \mathrm{T} \end{aligned} \tag{3-10}$$

for a fixed tree structure $q(x)$, the optimal weight w_j^* of leaf j can be computed by:

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda'} \tag{3-11}$$

and the corresponding optimal value is calculated as:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda'} + \gamma \mathrm{T} \tag{3-12}$$

The equation 3-11 is a valid objective function that can be used as the scoring function which measures the quality of a tree structure q. Since it is normally impossible to enumerate all possible trees, a greedy algorithm can be used instead, which iteratively sums the branches starting from a single leaf. Assuming that I_L and I_R are the instances of left and right node sets after the split, by letting $= I_L \cup I_R$, the loss reduction after the split can be rewritten as:

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda'} + \frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda'} - \frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda'}\right] - \gamma \tag{3-13}$$

The equation 3-13 is the final formula that is used to evaluate the split candidates in XGBoost algorithm. This evaluation of finding the best split is one of the key problems in tree learning, and the method exact greedy algorithm runs through every possible split. However, the greedy methods are not efficient considering the computational time and memory usage. Therefore, an approximate algorithm can also be used to improve the efficiency, where candidates can be split according to the percentiles of feature distribution. The pseudocodes for exact greedy and approximate algorithm are given in Appendix-2. XGBoost method uses weighted a modified approximate algorithm, namely, quantile sketch algorithm (Figure 3-11) for candidate split point proposal as default.

Additionally, the efficiency of training can also be decreased when the input *x* is sparse. The sparsity of the input can be caused by 1) missing points in data; 2) frequent zero entries; 3) one hot

encoding applied to the dataset. Since actual and virtual sensor residuals are used as inputs to the XGBoost algorithm in this study, there is a possibility that the input data contains zero entries when the virtual sensor predictions match the actual sensor readings. To prevent such efficiency decrease, the quantile sketch algorithm also include sparsity-awareness in XGBoost. In the sparsity-aware split finding algorithm, a default direction is added to each tree node, and the instance is classified towards this default direction when there is a value missing in the sparse input matrix. The optimum default directions are learnt using the input data during the iterative process. The pseudocode for the sparsity-aware split finding (quantile sketch) algorithm is given in Figure 3-11.

Algorithm: Sparsity-aware Split Finding

Input: I, instance set of current node
Input: $I_k = \{i \in I | x_{ik} \neq \text{missing}\}$
Input: d, feature dimension
Also applies to the approximate setting, only collect statistics of non-missing entries into buckets
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I}, g_i, H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  // enumerate missing value goto right
  $G_L \leftarrow 0,\ H_L \leftarrow 0$
  for j *in sorted*$(I_k,$ *ascent order by* $\mathbf{x}_{jk})$ **do**
    $G_L \leftarrow G_L + g_j,\ H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L,\ H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
  // enumerate missing value goto left
  $G_R \leftarrow 0,\ H_R \leftarrow 0$
  for j *in sorted*$(I_k,$ *descent order by* $\mathbf{x}_{jk})$ **do**
    $G_R \leftarrow G_R + g_j,\ H_R \leftarrow H_R + h_j$
    $G_L \leftarrow G - G_R,\ H_L \leftarrow H - H_R$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split and default directions with max gain

Fig. 3-11 Sparsity-aware split finding algorithm

In addition to these algorithmic enhancements, XGBoost algorithm contains several system optimization methods such as:

1) Parallelization:

   XGBoost implements a parallel process during the sequential tree building. This is possible because of the interchangeable and flexible nature of the base learner builder loops. In the base learner building process, the outer loop generates the leaf nodes of a tree, while the second inner loop enumerates the features. Normally, this nested loop structure presents a limit to parallelization since the completion of inner loop (which is more computationally demanding) is necessary to proceed to the outer loop. Thus, the order of loops are interchanged by initialization through a global scan where all instances are sorted using parallel threads to improve run time. This modification offsets any parallelization overhead in computation and significantly improves algorithmic performance

2) Tree Pruning:

   In traditional gradient boosting frameworks, the stopping criterion for tree splitting depends on the negative loss criterion and it is greedy in nature. XGBoost algorithm require the max_depth to be specified instead of this negative loss criterion, and implements backward tree pruning. This depth-first approach is one of the most significant factors that cause the high performance of the algorithm.

3) Hardware Optimization:

   The XGBoost algorithm also makes efficient use of the hardware resources. This optimization is achieved by creating cache aware internal buffer allocation in each thread where the gradient statistics are stored. Additionally, a further option to enable out-of-core

computing is provided for optimizing the available disk space during processing of big data frames which does not fit into the memory.

Moreover, an optimization of hyperparameters of the model is necessary to improve model reliability. The optimization of hyperparameters is performed by conducting a grid search on results of 9-fold tests of the models. 9-fold testing is chosen as the sampling method since its results would give the most critical information about the generalization capacity of the model compared to the random, 15-fold and 23-fold methods. This is because 9-fold testing dismisses the highest number of clusters from training and conducts tests in operating conditions totally unknown to the trained model. After the grid search, the hyperparameters of the XGBoost model are selected and the proposed virtual sensor residual assisted XGBoost method is trained and tested as previously mentioned. The final test results and its comparison with other methods are given in the 5^{th} chapter.

3.5. Summary of the overall FDD methodology

The overall FDD process proposed in this study can be briefly summarized as the following. First, the collected sensor data must be preprocessed as explained in Chapter 2, this preprocessing involves stages such as data cleaning, sensor selection and input data correlation filtering. This stage ensures that the input data for the FDD algorithm is free of defects and does not contain duplicate or irrelevant information. Then, the FDD algorithm is initiated, in which the main processes can be described as shown in Figure 3-12.

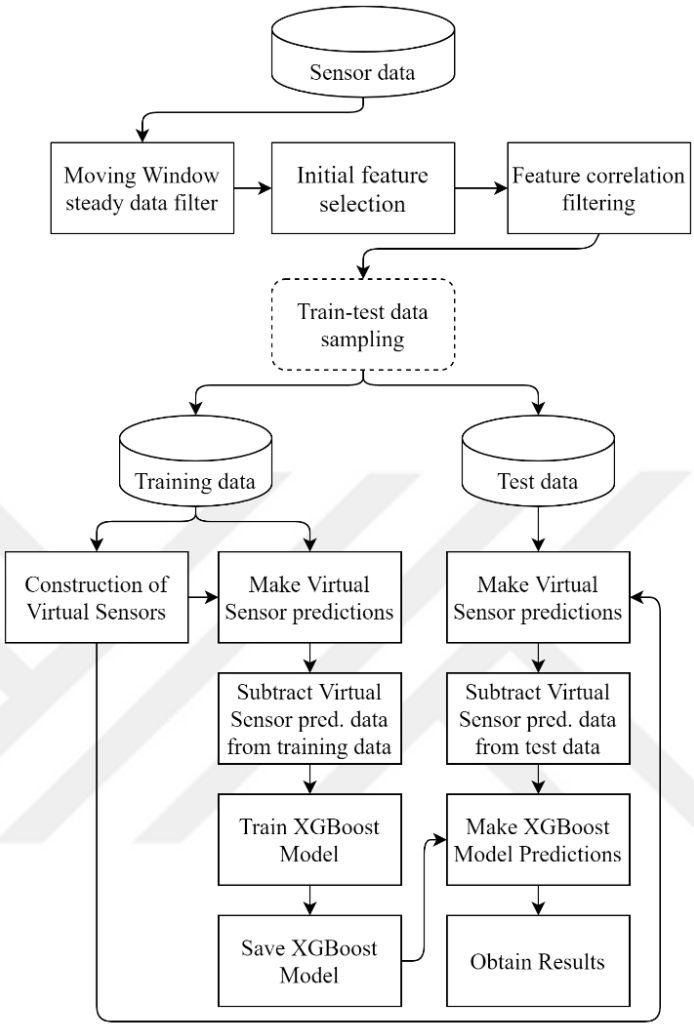


Fig. 3-12 Flowchart for the overall FDD methodology

The construction of virtual sensors itself include several sub-stages, where the intermediary results are validated using k-fold cross testing. The backward elimination process is the first sub-stage of virtual sensor construction where the importance of other sensors are ranked according to the information they contain about the tested sensor in an iterative manner. As a result of this stage, a dropping sequence is obtained, which provides the series of sensors to be removed in the main virtual sensor modeling algorithm iterations.

The virtual sensor modeling algorithm is the second sub-section of the virtual sensor construction algorithm. In this stage virtual models for actual sensors are constructed and tested using k-fold cross validation. The iterations proceed with removal of single sensors from the virtual models per iteration according to each dropping sequence. As a result, the best models with highest generalization capacities are selected as virtual sensor models.

In the last stage the differences between the actual sensors and their virtual models are supplied as inputs to the XGBoost algorithm. The XGBoost algorithm predicts the FDD result of data using its optimized gradient boosting framework. The training and testing of the FDD framework follows the same flowchart with different data which is split prior to the training. The prediction performance results of the overall FDD framework are given in Chapter 5.

Chapter 4

Cloud-based fault diagnosis framework

4.1. Front end framework

Front end layer of the fault diagnosis framework refers to the user interface where the users directly interact with the software. As per request of browsers, web pages return a Hyper Text Markup Language (HTML) [85] file that visualize the content and allow user-software interactions. HTML was created in 1991 by Berners-Lee, and HTML 2.0 was published as the first standard HTML specification in 1995. The current version of HTML is HTML 5.0, which is used in every modern web content. Aside from HTML, the contents appearance/presentation is managed by Cascading Style Sheets (CSS) [86] and the functionality/behavior is handled by JavaScript (JS) [87]. CSS is a declarative language that is used to style the HTML elements and display them properly. With the use of selectors, any element's style can be altered by changing its properties and values. JavaScript is a lightweight and interpreted programming language that is used as the scripting language for web content. It is currently the most popular programming language in the world and it is integrated with HTML. The JavaScript code of the HTML is run in the client's browser. However, there are many non-browser environments that can also run the JavaScript code.

Aside from the HTML-CSS-JavaScript basis, the frontend of the project is constructed using React.js library [88]. React is a JavaScript library that is used to create reusable user interfaces. It is released by Facebook in 2013. React aim to offer simpler programming model and improved performance by creating a virtual Document Object Model (DOM) which is a JavaScript object.

The library recommends use of JSX which is the JavaScript syntax extension. Different parts of the webpage can be constructed as different components, therefore the readability and rendering speed is increased. Also, the data flow in the virtual DOM is unidirectional, therefore any state change of the component is rendered on the virtual DOM, and only the updated result will be shown on the browser DOM. Thus, there is no need for a page refresh. The rendering principle of React virtual DOM is illustrated in Figure 4-1.

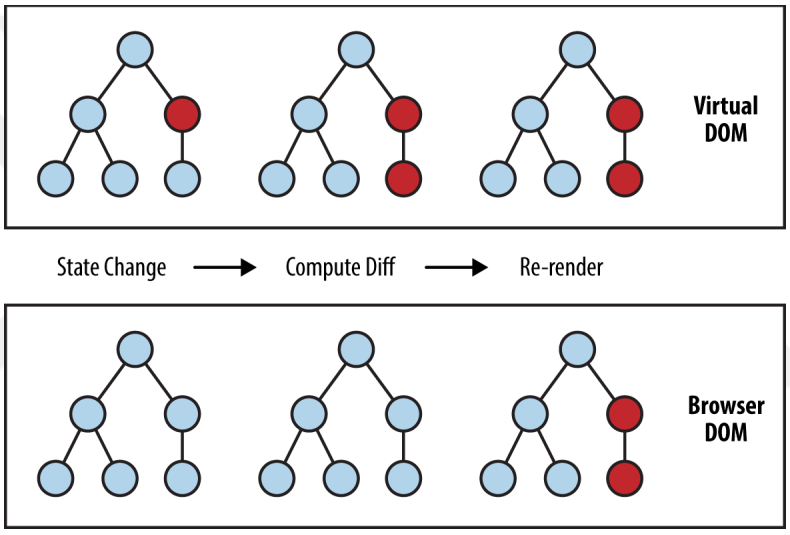


Fig. 4-1 State change principle of virtual DOM

In addition to React.js, Ant Design UI [89] library is used for the UI elements and Mapbox API [90] is used for the representing the locations of the chillers on an interactive map. And such elements of the website are styled using Less [91], which is a customizable CSS preprocessor. All of these libraries are installed via Yarn [92] package manager to the root folder of the project. The file tree of the frontend framework of this project is given in Appendix-1 and the simplified page and component tree is given in Figure 4-2. As can be seen from the figure, the website consists of 6 pages namely, login, home, atlas, graph, operator and unavailable pages. Also, the components are separated for their respective pages with the corresponding file names. It should be noted that

the components that are not classified in a folder with a page name are the ones that are used in multiple pages such as page header and page footer components.

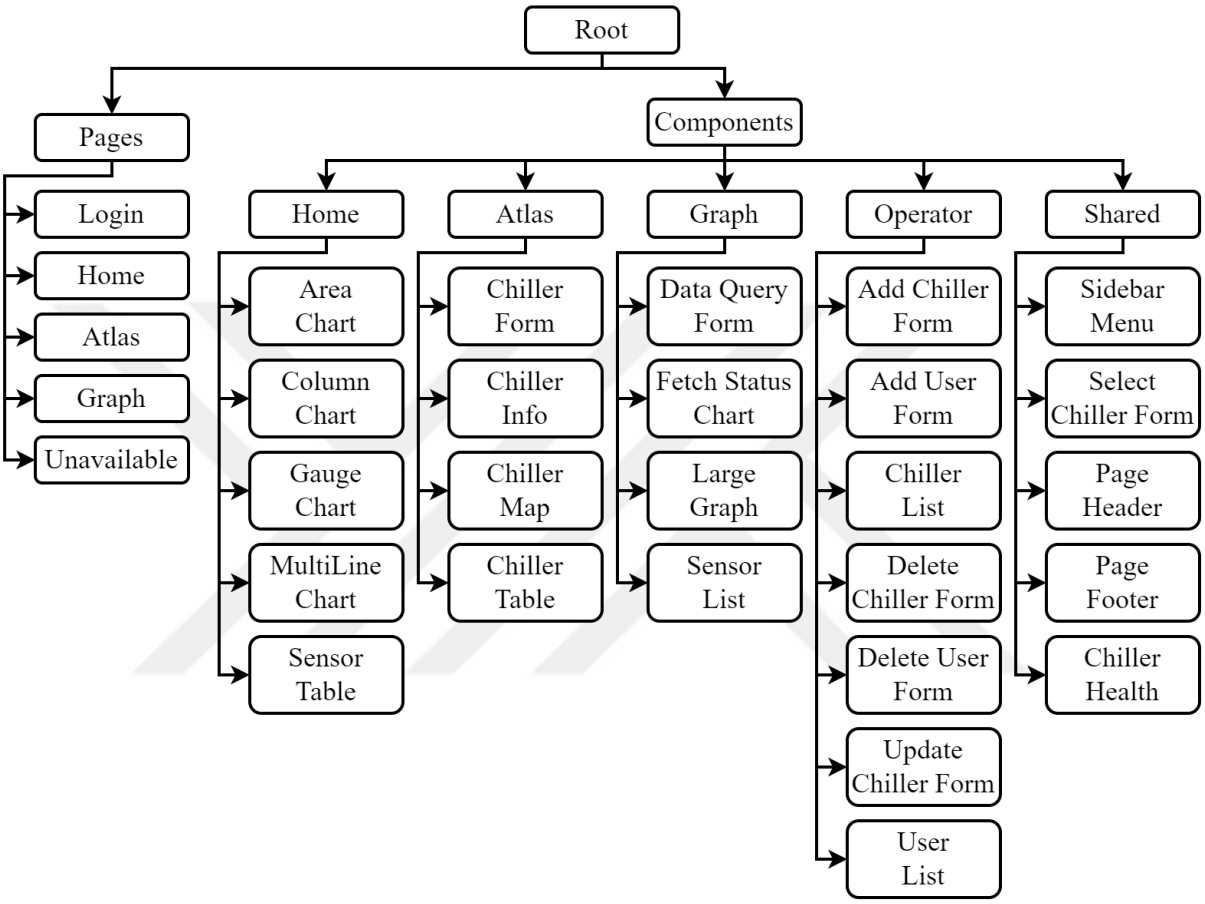


Fig. 4-2 Page and component tree of the frontend framework

In the login page, the user is required to enter his account name and password to proceed to the home page. If it is a new user, the user can redirect to the register page and register a new account by entering username, email and password. Moreover, visitors can also access the website with default credentials test: pass. When the user is authenticated, the homepage will directly display the chillers that are registered to the specific user.

In the atlas page, users can interact with the first page of the dashboard. The center of the page displays the map section, where the registered chillers are displayed in the real world locations. Additionally, the users can click the chiller pinpoints on the map to change the currently selected chiller to display additional information. The left part of this page contains the chiller information board, which illustrates the information regarding chiller name, location, connection status, and the chiller manager. Chiller managers are the users who are authorized to change the chiller information such as, chiller name and chiller users. A manager can add or remove other users to view the chiller with or without admin rights. On the bottom of the left panel is the chiller administration form, where such chiller information changes can be conducted. On the right side, top panel shows the FDD prediction on chiller health and can be refreshed on user request. Below, that panel, all chillers registered to the current user are displayed in a list format, where their connection and health status are also included. Lastly, homepage also contain the navigation bar on top to conduct pagination and user information changes, similarly to other pages of the dashboard.

Second page of the dashboard is the home page where chiller specific information are demonstrated in detail. The main system schematic is a Support Vector Graphic (SVG) file which is styled with CSS and made interactive with JavaScript element selectors. From the schematic user can click a desired component or sensor to view the respective data in the other components of the page. On the left panel, there is a line graph showing the data from the selected sensor and there is a table containing all sensor data. Below the main panel, there is a card named system information, which contains the information from compressor and the same system health data containing the FDD prediction on the chiller. On the right panel, there are cooling performance gauges, a multi-line graph and a bar chart about power consumption of the chiller.

The third page of the dashboard is comprised of the real-time line graph of the chosen sensor data. From the left panel user can change the selected chiller and choose any data that is desired to be displayed in real time. Moreover, user can query the past data between the defined dates from the bottom right form. The refreshing of the data is conducted with predefined 5 second intervals.

The remaining pages are the operator and unavailable pages. In the operator page, website admin can add a new chiller to the system by defining its name, location and users. Also, current users and chillers registered in the system can be viewed in a tabulated format. On the other hand, unavailable page is served to the newly registered user when there is no chiller assigned to the particular user. These users may ask the chiller admins to assign the chiller to their account to pass the unavailable page error and access the dashboard.

4.2. Back end framework

4.2.1. REST API server

The first part of the backend framework is the representational state transfer application programming interface (REST API), also known as RESTful API. In this server, how devices and applications communicate with each other is defined with a set of rules. The REST structure is first defined by computer scientist Dr. Roy Fielding in 2000, and since became the most common connection method for components and applications.

The backend server runs in Node.js [93], which is a cross-platform JavaScript runtime environment that runs the V8 JavaScript engine. The Node.js applications do not create a new process per incoming request, and instead runs in a single process. Moreover, it can utilize a set of asynchronous I/O primitives to prevent the blocking behavior of JavaScript code. Therefore,

Node.js servers can handle a large number or concurrent requests without a need for concurrent thread management.

In this project Express [94] framework is used in the Node.js server to set up middlewares to respond requests and to define the routing table. The file tree of the Node.js server is given in Appendix-2. In the root folder, the index.js file is where the main code for the server runs and the required packages and modules are imported. The models folder includes the relevant database models where the types and shapes of the data to be written to the database are defined. The routes folder defines the REST API routes where the front end framework accesses to interact with the Node.js server. The schematic for the routes are given in Figure 4-3.

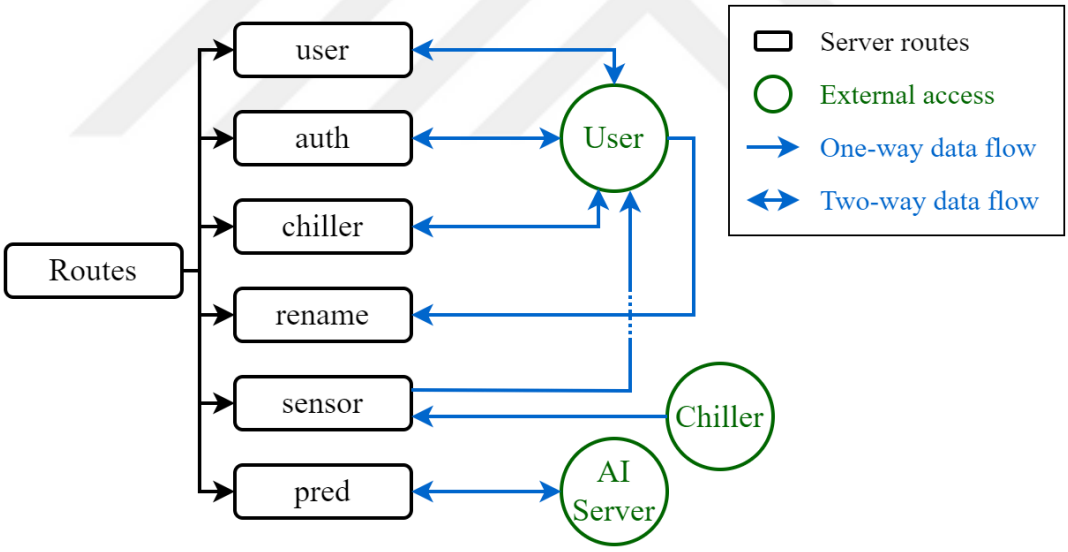


Fig. 4-3 Node.js server routes schematic

The names of the routes are in fact self-explanatory. In example, the auth route is accessed for user authentication, user, chiller and sensor routes are accessed for reading, writing and editing user, chiller and sensor information, and pred route is accessed for making predictions on the chiller health. On the pred route, the FDD predictions are in fact obtained by sending request to

another standalone machine learning server. The following chapter introduces the working principles of the machine learning server used in this project.

4.2.2. Machine learning server

The machine learning server of this project uses the Flask [95] web application framework which is written in Python. Flask was developed by Armin Ronacher in 2012 and consists of Web Server Gateway Interface (WSGI) toolkit and Jinja2 template engine. WSGI is a standard specification for Python web application development interface and Jinja2 is a template rendering engine for web pages. The choice of using Flask in this project as a machine learning server framework is made because of the freedom that it provides in the construction of the server structure. Because of this freedom, the FDD algorithm could be integrated in the server to make chiller health predictions on request.

The proposed algorithm is constructed using the PyTorch [96] machine learning framework and incorporated into the Flask server. The FDD models are first trained on a local computer and the saved models are then uploaded to the server. Figure 4-4 shows the separated processes of training and prediction incorporated into the machine learning server. Since training the models are very CPU/GPU intensive, the limited CPU capacity of the server is only used to make predictions form the loaded models. The main Node.js server requests predictions from the Flask server periodically, or on the user request. The file tree of the machine learning server is given in Appendix-3.

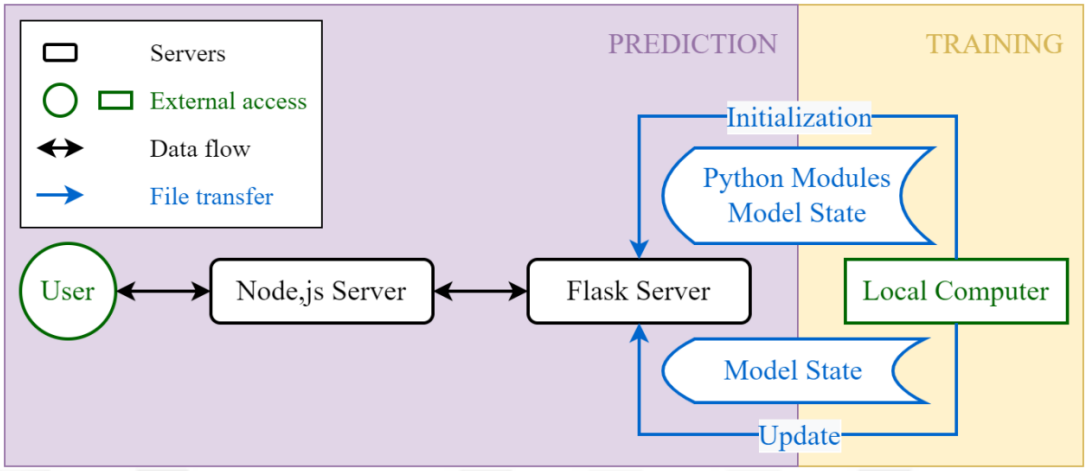


Fig. 4-4 Separation of training and prediction processes in machine learning server

As can be seen from the file tree, the app.py file in the root folder is responsible for running the server and importing the necessary modules for the FDD algorithm. The codes for the FDD algorithm (including virtual sensor modeling) are placed in the files such as fdd_main.py, ffm_predict.py and utils.py. These files import the pre-trained model states from the folders model_state and feature_data. These states of the pre-trained model can be updated by accessing the servers root folder and changing the loaded .sav and .bin files of the FDD model.

4.3. Cloud database

For the database of this this project MongoDB Atlas [97] is used, which is the company's cloud database service. MongoDB is considered to be a document-oriented NoSQL database, where collections and documents are used in contrary to the tables and rows of the traditional relational databases. The documents are comprised of key-value pairs as their basic unit and the collections consist of sets of documents. The models for the collections and documents used in this project are illustrated in Figure 4-5.

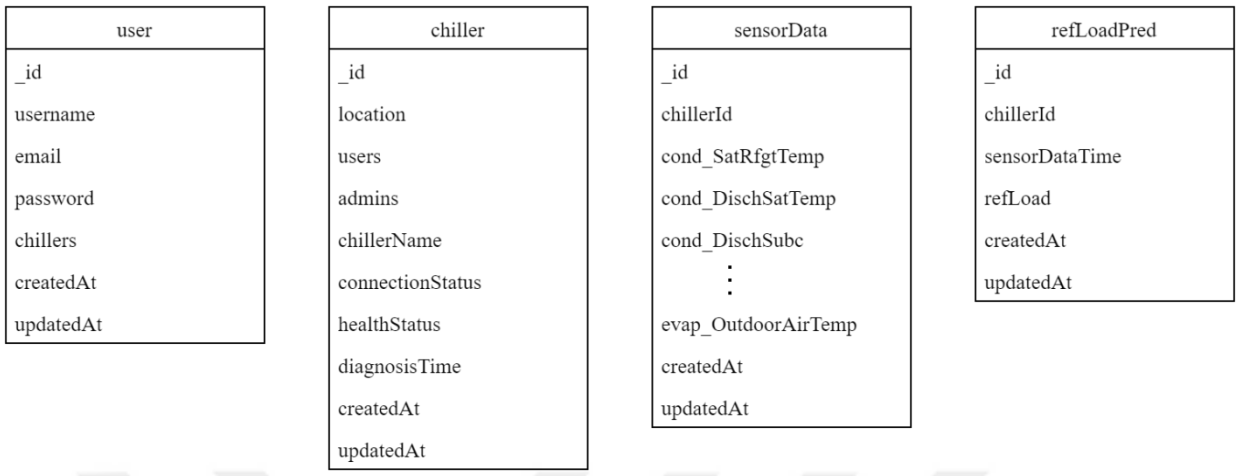


Fig. 4-5 Database collection and document models

As can be seen from the figure, user, chiller, sensorData and refLoadPred collections store the relevant documents containing the information about users, chillers, sensor data and the refrigerant load predictions. The chillers data field of user documents and the users data field of the chiller documents ensure the many-to-many relationship between users and the chillers. Additionally, chillerId data fields of the sensorData and refLoadPred documents establish the one-to-many relationship between chillers and sensor/prediction data. Even though the FDD predictions are obtained from the Flask server, these predictions are initially transferred to the Node.js server, and then written to the database. Therefore, only the Node.js server can directly access the cloud database.

4.4. Overall online framework

Overall, the cloud framework of the proposed methodology can be divided into three main sections such as the database, the backend and the frontend. The frameworks of these sections are explained in detail in the previous sections, and a diagram describing how these sections of the

overall online framework are connected to each other is given in Figure 4-6. The database, backend and frontend sections are represented with colors red, green and blue, respectively.

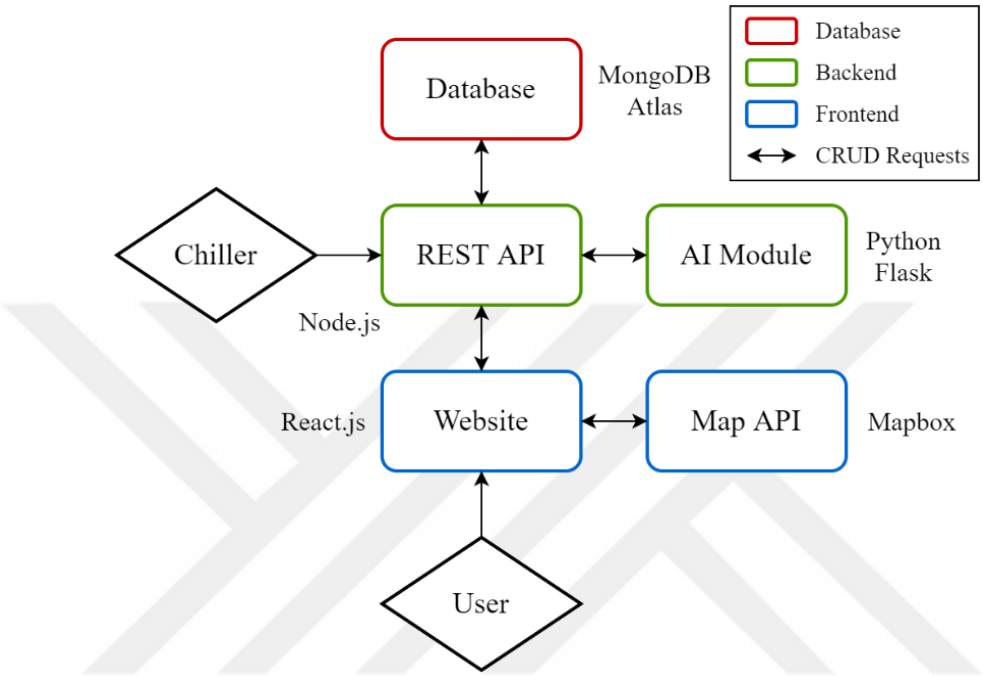


Fig. 4-6 Structure of the cloud framework of the proposed methodology

The connections between the cloud framework sections are represented by arrows in the diagram. These connections are conducted with HTML requests using the necessary access keys for security purposes. Such keys are stored in .env files of each server, as shown in the Appendix. As shown in the figure, the users access the Node.js server by sending HTML requests through their interactions with the front end website, and chillers can directly access the server from a dedicated route. This dedicated route is only used for sensor data collection. The database is then accessed from the Node.js server with the access key and the necessary create-read-update-delete (CRUD) operations are conducted. Either periodically, or upon user request, the main Node.js server requests FDD predictions from the Flask server by getting the sensor data from the database and sending it to the Flask server. Then the Node.js writes the FDD prediction response data from

the Flask server to the cloud database. The aforementioned processes as a whole summarize the cloud-based FDD framework used in this project.

4.5. Internet of Things framework

4.5.1. Hardware Setup

The chiller side access to the online framework is conducted with the use of Internet of Things (IoT) devices connected to the intelligent terminal of the chiller through Modbus protocol. In the applications where the FDD framework is not cloud based, Raspberry Pi [98] microcomputers are the most common choice of engineers to can run the FDD software locally and then send or store the fault prediction data for one or multiple chillers. However, when the FDD algorithm is handled in the cloud, there won't be a requirement for computational power in the local IoT device network endpoint anymore. The only task of the IoT device network is to collect and send sensor data to the online framework for the cloud-based fault diagnosis. Therefore, NodeMCU ESP8266 [99] development board has been selected in this study to perform the task of sensor data transfer to the online framework.

This board significantly reduces the cost of implementing FDD algorithms to chillers, compared with the systems that are using Raspberry Pi. Moreover, since there is no need for a single endpoint of the system to perform fault diagnosis in the proposed framework, every IoT device can access the Wi-Fi router independently. This modification also add a reliability advantage to the overall framework, since the data collection operation from remaining chillers would still continue in case of a failure in the communication of a single chiller in a multi chiller system. The pinpoint diagram of NodeMCU ESP8266 development board is given in Figure 4-7.

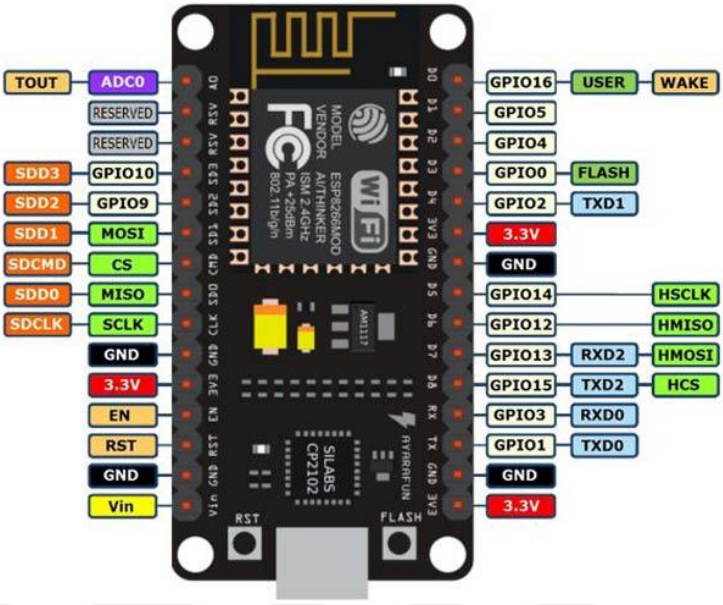


Fig. 4-7 Pinpoint diagram of NodeMCU ESP8266 board

The core of the NodeMCU ESP8266 development board is the ESP-12E module that contains the ESP8266 chip which includes Tensilica Xtensa 32-bit LX106 RISC microprocessor. The microprocessor supports RTOS and operates at 80-160MHz adjustable clock frequency. The NodeMCU board accommodates 128 KB RAM and 4MB of Flash memory for data and program storage. The relatively high processing power, built in Wi-Fi / Bluetooth module and deep sleep operating feature makes the board a popular choice for the IoT projects. The board can be powered by both using a Micro-USB jack and external power supply pin (VIN). Also the NodeMCU board supports UART, SPI and I2C interface for data transfer.

The complete specifications and features of the NodeMCU ESP8266 board are given in the following list and the exact locations of the components are shown in Figure 4-8:

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Input Voltage: 7-12V

- Operating Voltage: 3.3V
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- Digital I/O Pins (DIO): 16
- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- USB-TTL based on CP2102 is included onboard,
- PCB Antenna

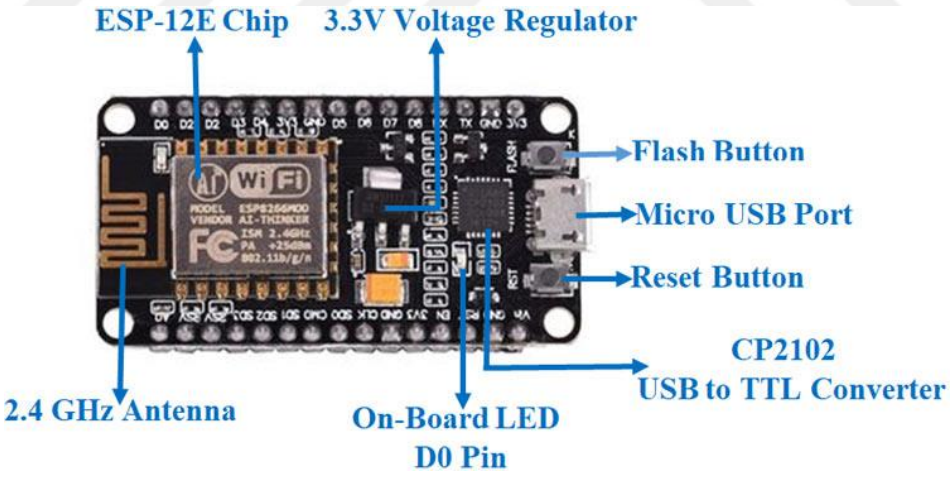


Fig. 4-8 Exact locations of components on the NodeMCU ESP8266 board

4.5.2. Board Programming

The NodeMCU ESP8266 board is programmed using MicroPython [100] in this study. MicroPython is a lean and efficient software implementation of Python3 programming language

which was originally developed by Australian theoretical physicist and programmer Damien George in 2013. It is written in C, optimized to operate on microcontrollers/constrained environments, and contains a small fraction of the Python3 library.

MicroPython compiles the Python code to bytecode and also compiles the runtime interpret of that bytecode to be executed on the microcontroller hardware. The users can also use the read-eval-print loop (REPL) interactive prompt to run supported commands. The bytecode that are generated by the MicroPython compiler has the ".mpy" file extension and allows the programmers to access to low level hardware. The logic flowchart for the MicroPython code that is used for programming the NodeMCU ESP8266 board is given in Figure 4-9.

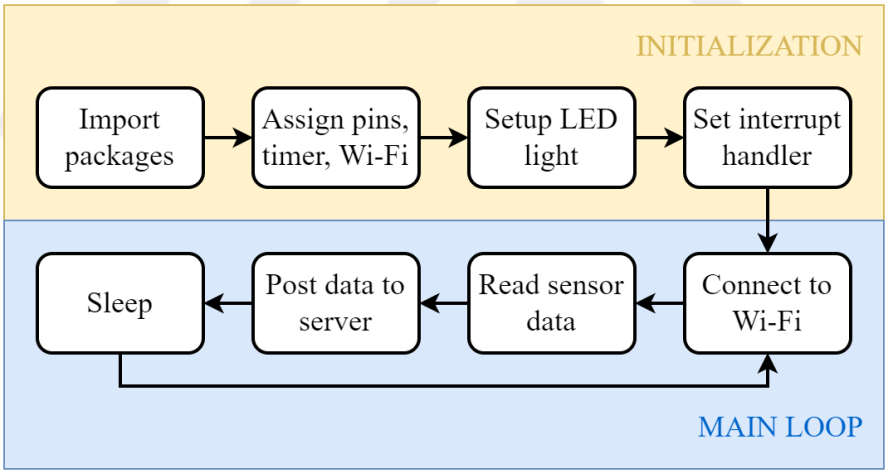


Fig. 4-9 Flowchart for the MicroPython code

As the first stage of the initialization, essential packages such as machine, time, network urequests and ujson are downloaded from the MicroPython library. Then the pin assignments are made for led light and switch button. The timer is initiated from the machine module to enable continuous led blinking during uninterrupted operation and the Wi-Fi module is launched in the station mode. Next, a led blinking function is defined to blink led light in different modes after

successful operations such as connecting to Wi-Fi router and posting data to server to give an output to the user. The last stage of initialization is the definition of the interrupt handler, which switches the device to standby mode on flash button press for debugging purposes.

After defining the time periods for the led blinking durations, the main loop of the device is initiated by connecting to Wi-Fi. If the board cannot connect to the Wi-Fi router, the software won't proceed and the connection loop would continuously try to connect the router with 5 second intervals. After successful connection, the board reads the sensor data and posts the data in JSON format to the REST API server by using urequests module. If the transfer is unsuccessful, the device will raise a non-blocking error that is displayed in the terminal if the device is connected to the PC. Proceeding the post request, the device sleeps for 5 seconds and repeats the main loop. In conclusion, the NodeMCU ESP8266 board sends the sensor data to the server every 5 seconds in an endless loop where the faults of the chiller are checked periodically. The next chapter evaluates the performance of the proposed FDD algorithm and the overall online system.

Chapter 5

Results and comparison

5.1. Virtual Sensor models

As mentioned in the previous sections, a new set of virtual sensor models are constructed every change in the training dataset. Which means, the binary parameters of the virtual sensors might be different for every unique set of training data, therefore the coefficients will be calculated again for each training of random, 9-fold, 15-fold and 23-fold testing. To illustrate the methodology and the changing test scores of the virtual sensor model construction, randomly sampled training data is inspected. Figure 5-1 demonstrates the mean MSE results of 9-fold tests of virtual sensor construction algorithm for three different input features as examples.

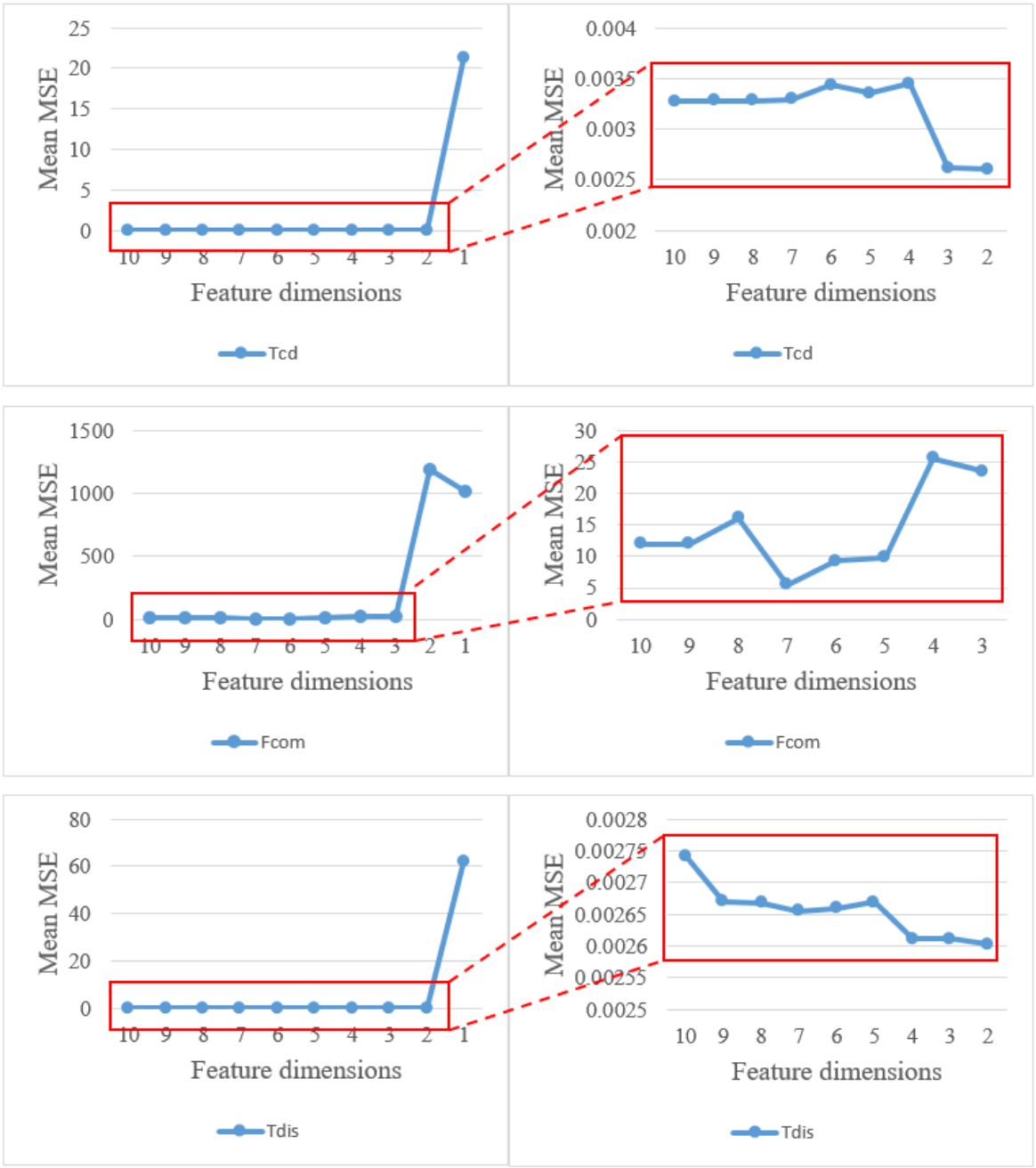
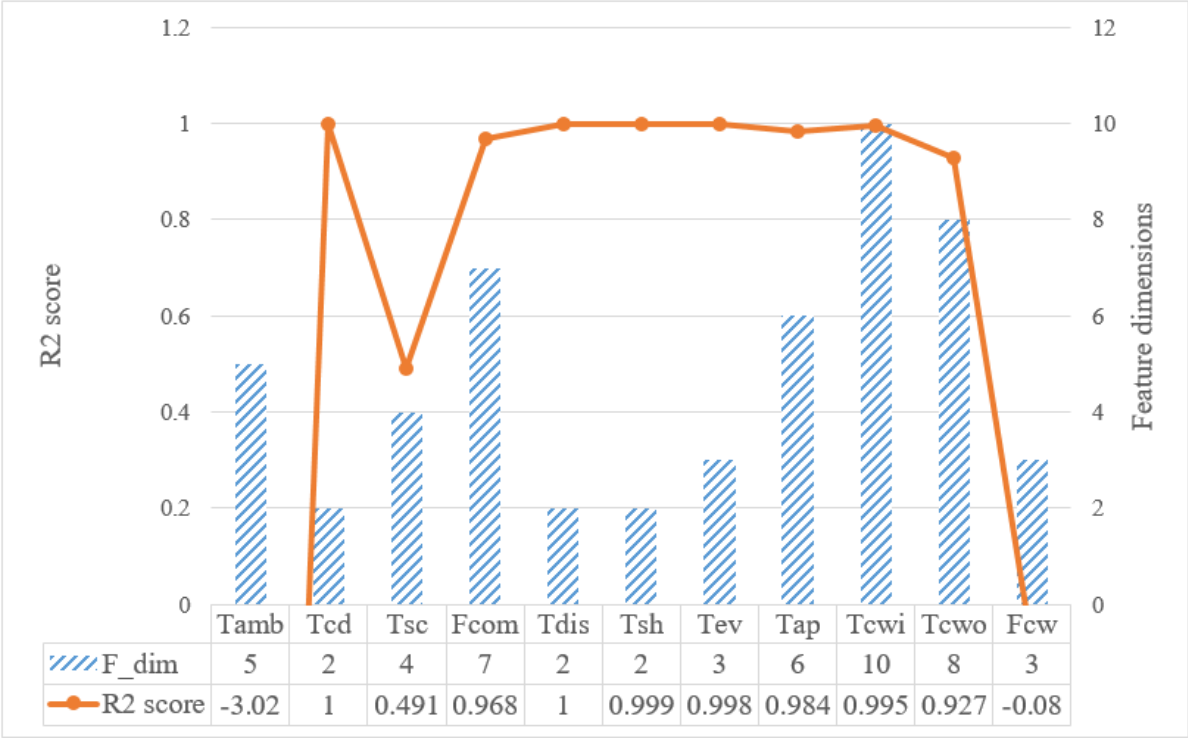


Fig. 5-1 9-fold mean MSE results of some virtual sensor models

As can be seen from the figures, the virtual sensor models of T_{cd}, F_{com} and T_{dis} are chosen to respectively include 2, 7 and 2 of the remaining features to obtain the highest generalization capacity models in fault-free conditions. This stage assigns the $b_{j,i}$ parameters in the formula.

Next is the training step with random sampling to obtain the $c_{j,i}$ linear regression coefficients and R2 score. The same processes are applied for all features in the dataset. Figure 5-2 illustrates the feature dimensions with minimum MSE and respective R2 scores of the selected models.



	Tamb	Tcd	Tsc	Fcom	Tdis	Tsh	Tev	Tap	Tcwi	Tcwo	Fcw
F_dim	5	2	4	7	2	2	3	6	10	8	3
R2 score	-3.02	1	0.491	0.968	1	0.999	0.998	0.984	0.995	0.927	-0.08

Fig. 5-2 Feature dimensions and R2 scores of virtual sensor models

There are two key points that can be obtained from this figure. First, majority of the models use a few remaining features to create virtual sensors with higher generalization. This is because when more features are added to the model, there might be some overfitting of the data and this would result in poor 9-fold test scores. For example, virtual condenser temperature model utilizes only the features T_{sh} and T_{dis}, whereas virtual evaporator saturation temperature uses $T_{cw,o}$, $T_{cw,i}$ and T_{ap} in their main linear regression models. Second, it is observed that the models for T_{amb}, T_{sc}, $T_{cw,o}$ and F_{cw} can't pass the R2 score requirement to be a valid virtual sensor. Meaning that the linear combinations of other features are not sufficient to model these features because of the non-

linearities within chiller the system, independence of the feature from the other sensors or the uniqueness of the information contained in the feature. As mentioned before, the respective a_i parameters of these features with lower R2 score are saved as zero. As a result, when calculating the virtual sensor residuals these parameters will not generate any residual value but directly provide the sensor information as an input to the XGBoost algorithm.

Figure 5-3 demonstrates the processing effect of virtual sensor residuals method on features F_{com} and $T_{cw,i}$ with histogram analysis. The actual and residual data shown in the figures include both training and test datasets used in the random testing iteration. Since the virtual sensor models are trained with fault-free data (100% refrigerant load), the residual values are distributed around 0 for 100% refrigerant load condition data in both models. Also, it can be seen from the models that the residuals get larger when the fault intensities get larger, as expected. Similar behavior can be observed for other features of virtual sensor models with specific characteristics in their respective histograms. Although the constructed virtual sensor models are changing at each iteration of random and k-fold training, the processing effect of virtual sensor residuals method is found out to be similar for every iteration. Thus, it can be declared that the virtual sensor residuals method process the raw input data and converts it into high quality information for the subsequent fault diagnosis algorithm.

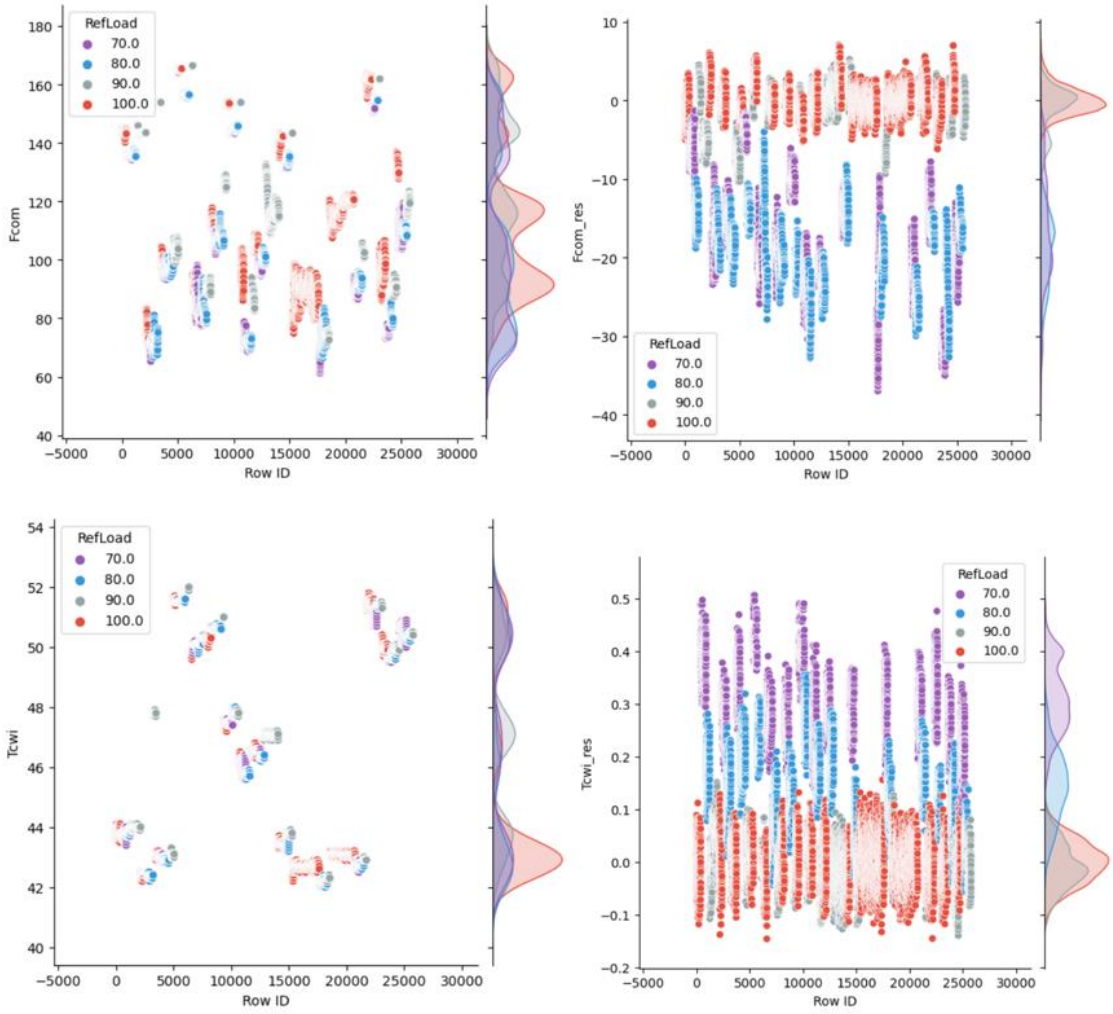


Fig. 5-3 Actual (left) and virtual sensor residuals (right) data for F_{com} (top) and $T_{cw,i}$ (bottom)

5.2. XGBoost hyperparameter optimization

The XGBoost algorithm contains various hyperparameters that can be tuned to increase the accuracy of the model. In this study, the parameters of the maximum depth per tree (max_depth), minimum child node weight (min_child_weight), the fraction of columns to be randomly sampled for each tree (colsample_bytree) and fraction of sampled observations (subsample) are optimized utilizing grid search technique. Considering time limitations, other hyperparameters such as learning rate, number of trees in the ensemble, alpha, lambda and gamma coefficients are set to

the default values of 0.3, 100, 1, 0, 1 and 0, respectively. The hyperparameters max_depth, min_child_weight, colsample_bytree and subsample are tested with the following sets of [8, 9, 10, 11, 12], [1, 32, 64, 128, 256, 512], [0.8, 0.9, 1] and [0.8, 0.9, 1]. Thus, in total, the model is 9-fold tested for 270 different points in the grid search space, however, since the grid search space is 4 dimensional, the results are visualized by demonstrating average mean MSE and the minimum mean MSE scores for each dimension of the tested hyperparameters in Figure 5-4.

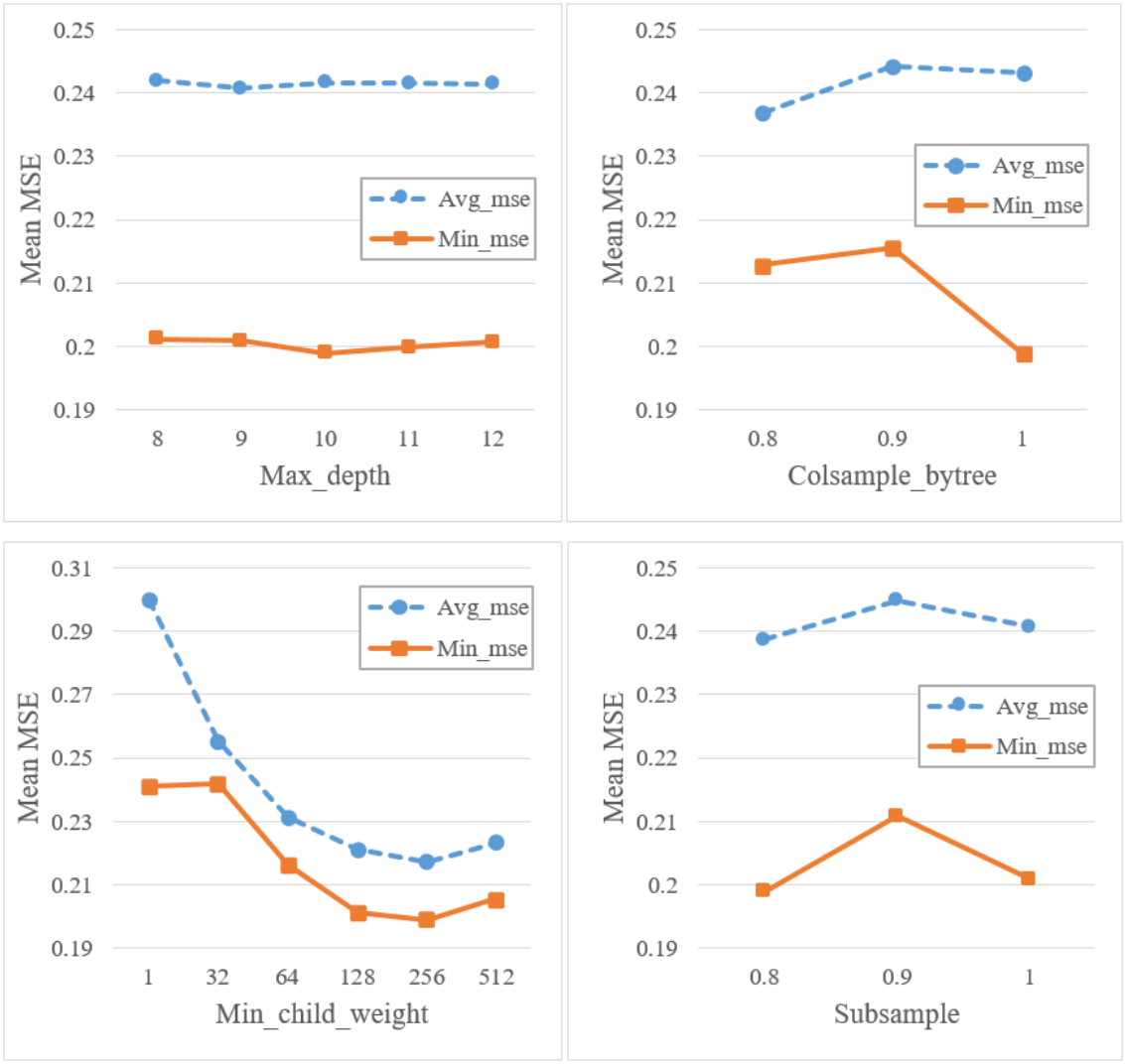


Fig. 5-4 XGBoost hyperparameter optimization

In Figure 5-4, points marked as avg_mse displays the average all samples for that value of the inspected parameter, whereas the min_mse shows a single sample. For example, avg_mse point at max_depth value of 8 takes into account of the average of 54 sample tests. For each sample, the model is tested 9-fold before the mean MSE are obtained. As a result of the grid search of the 270 samples, the hyperparamaters of the absolute minimum MSE sample are chosen as the optimized values. Which means, the final values 10, 256, 1 and 0.8 are selected for the hyperparameters max_depth, min_child_weight, colsample_bytree and subsample, respectively.

5.3. Fault diagnosis results of the proposed model

Predicting the refrigerant charge level in the chiller system is in fact a regression problem. However, the results of evaluation metrics of regression methods such as mean-squared error, is dependent on the input dataset and might not be directly intuitive. Therefore, the model outputs are rounded to the nearest dataset refrigerant load level to enable the use of classification methods' evaluation metrics. When the predictions are rounded, the proposed method can be treated as a multi-class classification algorithm with four classes, namely, 70%, 80%, 90%, 100% refrigerant load. All of the possible prediction outcomes for the k^{th} class c_k is given in Table 5-1.

Table 5-1 Possible outcomes in a multi-class classification problem

		Prediction		
		$c_0 \dots c_{k-1}$	c_k	$c_{k+1} \dots c_n$
Ground truth	$c_0 \dots c_{k-1}$	True Negative (TN)	False Positive (FP)	True Negative (TN)
	c_k	False Negative (FN)	True Positive (TN)	False Negative (FN)
	$c_{k+1} \dots c_n$	True Negative (TN)	False Positive (FP)	True Negative (TN)

According to the definitions given in Table 5-1, the following equations (5-1 to 5-4) are the used in this paper to measure the classification performance of the proposed model.

$$Accuracy = \frac{\text{TruePositive+TrueNegative}}{\text{TruePositive+TrueNegative+FalsePositive+FalseNegative}} \quad (5\text{-}1)$$

$$Precision = \frac{\text{TruePositive}}{\text{TruePositive+FalsePositive}} \quad (5\text{-}2)$$

$$Recall = \frac{\text{TruePositive}}{\text{TruePositive+ FalseNegative}} \quad (5\text{-}3)$$

$$F1\ score = \frac{2\times Precision\times Recall}{Precision+Recall} \quad (5\text{-}4)$$

It should be noted that while calculating precision, recall and F1 score, the metrics are calculated for each label first and then their unweighted average is taken. Also, the mean values of k tests are shown as results for the k-fold tests. After conducting random, 9-fold, 23-fold and 15-fold tests, the classification scores of the proposed model are obtained. The fault diagnosis results of the proposed method for respective tests are shown in Table 5-2 and Figure 5-5.

Table 5-2 Fault diagnosis results of virtual sensor residuals assisted XGBoost method

	Random (20%)	**9-fold**	**23-fold**	**15-fold**
Accuracy	0.998448	0.678124	0.705349	0.716063
Precision	0.998451	0.641901	0.730551	0.728765
Recall	0.998192	0.658418	0.729212	0.744636
F1	0.998321	0.610689	0.679501	0.677242

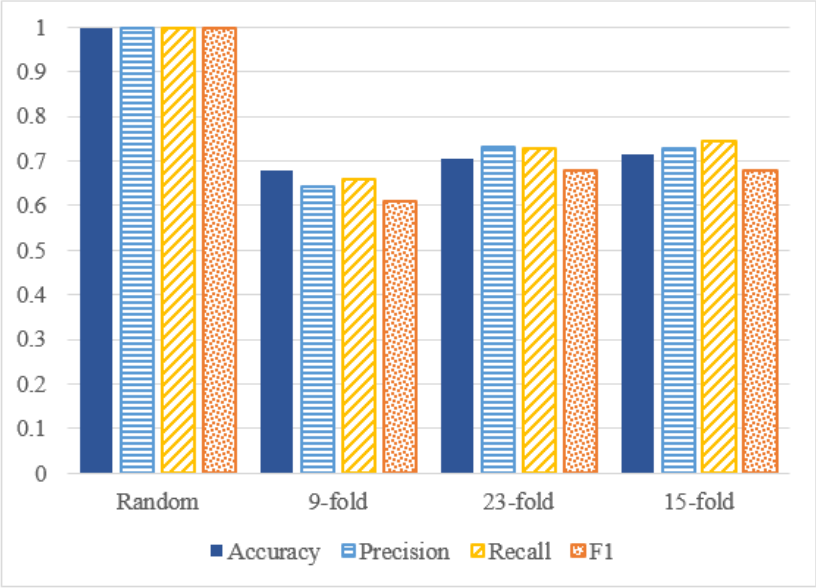


Fig. 5-5 FDD results graph for the proposed method

As shown in the graph, accuracy, precision, recall and F1 scores from random testing are all very high compared to other testing methods. The root cause of this phenomena is random sampling's tendency to train and test the same points in clustered datasets, which was explained in previous sections. Figure 5-6 shows a comparison of targets and the predictions from random testing and the first iteration of 9-fold testing for this model.

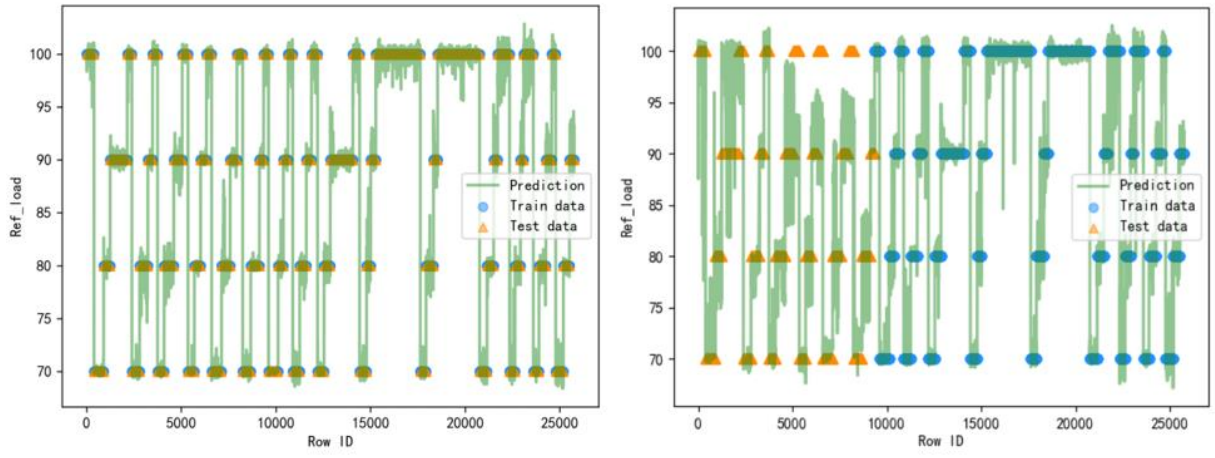


Fig. 5-6 Comparison of random testing (left) and the first iteration of 9-fold testing (right)

When the data is sampled randomly for a clustered and steady state filtered large datasets, training and testing datasets automatically overlap, therefore the results from random tests only contain the information of training loss. Raising the random allocation proportion to 30% or decreasing it to 10% would not have a significant effect since the overlapping phenomena would remain unchanged. On the other hand, in 9-fold tests, the models are tested for the operating conditions that were completely unknown in training. Thus, it is more challenging for the algorithm to make accurate predictions. Since the tested proportion of the dataset gets smaller and the trained proportion gets larger for 23-fold and 15-fold tests, the accuracies of predictions increase, as expected. The confusion matrixes for all testing methods are shown in Figure 5-7.

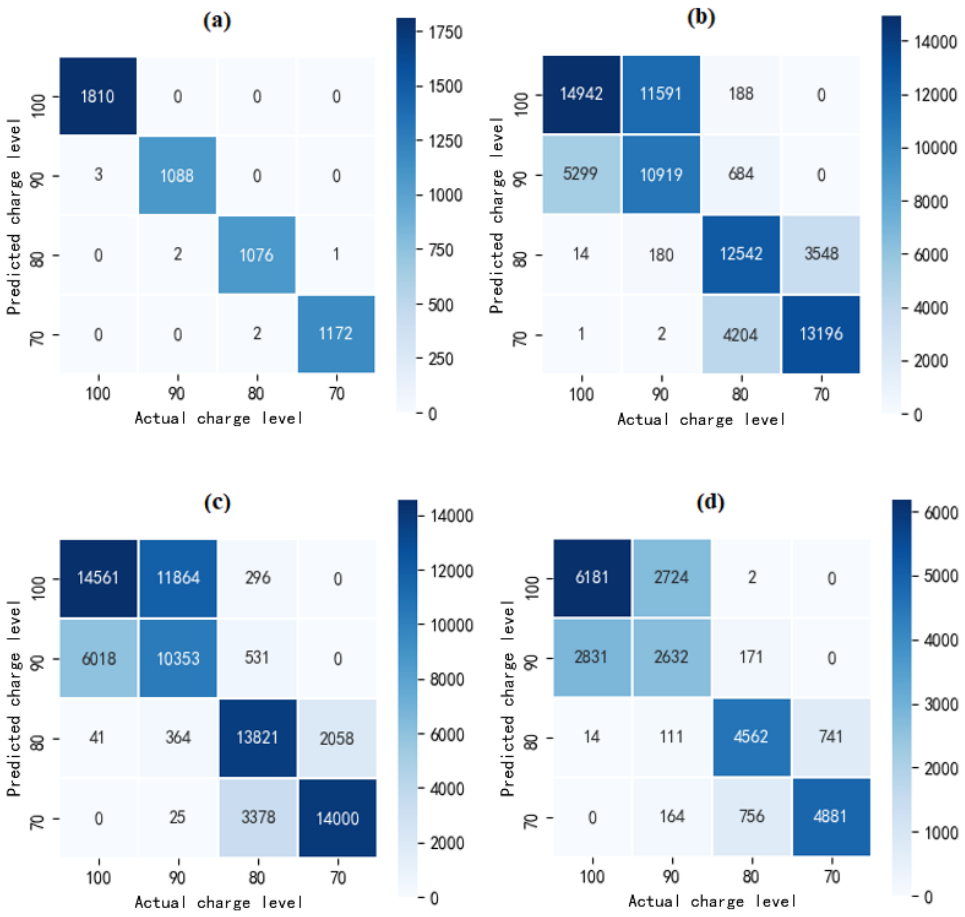


Fig. 5-7 Confusion matrixes for random (a), 9-fold (b), 23-fold (c) and 15-fold (d) tests

The confusion matrixes for k-fold tests represent the cumulative results of all k tests. Since the 9-fold and 23-fold tests scan through the dataset three times the total number of predictions are also three times of random and 15-fold tests. The k-fold results indicate that the classification between 100% and 90% refrigerant charge levels still needs to be improved. To check the validity of the proposed method, comparisons with conventional methods are made in the next section.

5.4. Comparison with conventional methods

The results from proposed virtual sensor residuals assisted XGBoost (VSR-XGB) method is compared with Support Vector Machines (SVM), Multi-Layer Perceptron (MLP), Random Forest (RF) and XGBoost (XGB) algorithms. Same stages of preprocessing are conducted for all algorithms before the models are trained to yield fault diagnosis results. The comparison graphs of the results are illustrated in Figure 5-8 and the numerical values for accuracies are shown in Table 5-3.

Fig. 5-8 Comparison of different methods

Table 5-3 Accuracy comparison for different methods

	SVM	MLP	RF	XGB	VSR-XGB
Random	0.992045	1	0.999806	0.999806	0.998448
9-fold	0.431478	0.578713	0.54971	0.594262	0.678124
23-fold	0.534013	0.640586	0.67972	0.639568	0.705349
15-fold	0.589648	0.696117	0.719671	0.683363	0.716063

In comparison to other methods, the proposed VSR-XGB model has greater accuracy results in 9-fold and 23-fold tests, whereas it is 0.2% behind of MLP in random testing and 0.4% behind

of RF in 15-fold testing. However, as explained before, random testing score only produces a reflection of training loss for the algorithm and any smart method can achieve results very close to 100% with optimized hyperparameters. Thus, the random testing score comparison should not be taken as a metric of comparison. Which means, the proposed model is actually only surpassed by RF algorithm in 15-fold testing by a very small margin. Nonetheless, if the 9-fold score of the RF model is inspected, it can be seen that the RF model performs poorly with only 54.97%. Moreover, 9-fold accuracy is intuitively a more important metric to measure generalization capacity compared to 15-fold score since a larger fraction data is hidden from the model during training. Yet, if the unweighted averages of k-fold scores are chosen as a metric to measure the generalization capacity of the models, VSR-XGB ranks the first and RF ranks the second best model. Therefore, it can be claimed that compared to the second best model, VSR-XGB improves the 9-fold, 23-fold test accuracies from 54.97% and 67.97% to 67.81% and 70.53% whereas only decreasing 15-fold test accuracy from 71.97% to 71.61%.

5.5. Deployment and testing of online framework

Following the tests, the final model is deployed to the dedicated Flask server hosted on Heroku [101]. Similarly, the REST API module which runs on Node.js is also deployed to its dedicated server on Heroku. The website build is hosted on Netlify [102], which can be accessible with the URL: “https://fdd-anduv.netlify.app/”. Both the user interfaces of the website and the backend access routes of the REST API are thoroughly tested. Moreover, real time data collection from the chiller is successfully simulated by both sending the previously collected data from a PC and an IoT device, namely, NodeMCU ESP8266 board. Source code of the whole project can be found in “https://github.com/burkayanduv/fdd-app” and the Figure 5-9 shows some screenshots from the final build of the website.

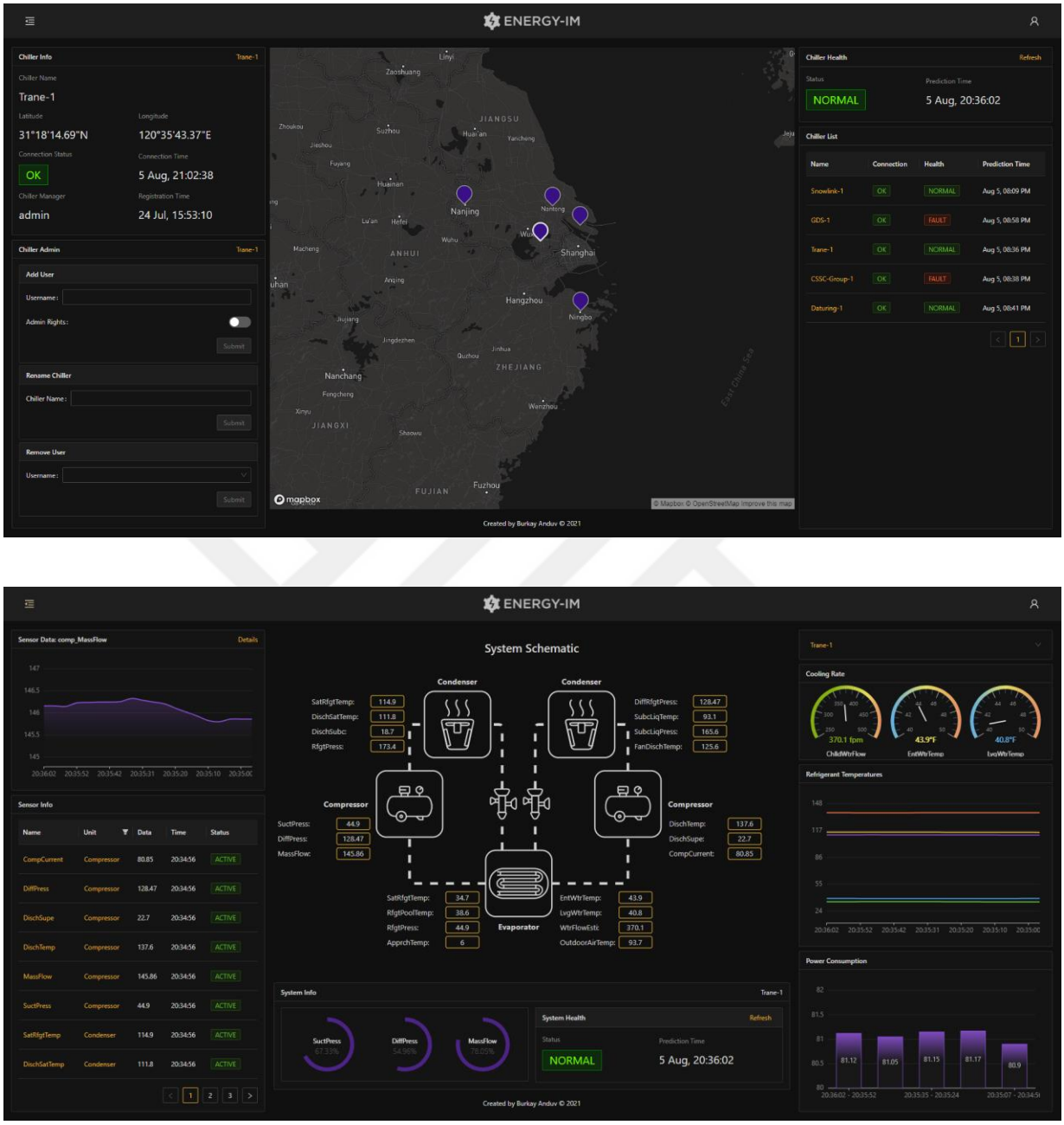


Fig. 5-9 Screenshots from the final online FDD framework

Chapter 6

Conclusion and future prospects

6.1. Summary

This study presented the methodology of virtual sensors construction and diagnosis of refrigerant leakage fault of chiller using an optimized XGBoost algorithm which utilizes the residual data from virtual and actual sensors. In addition, this methodology is uploaded to an online framework to make tracking the status of chiller health possible from a user friendly website. As the first step, various operating condition experiments are conducted on an industrial chiller to collect large amounts of experimental data. Then, the unique and information rich features are extracted from the dataset by necessary preprocessing algorithms. After the construction and training of the proposed method, the overall methodology is tested and the results are validated by comparing it with conventional machine learning methods.

The results reveal that, the chiller experiments' data collection framework causes the random testing results to be misleading, where all machine learning methods achieve very high classification accuracies. Therefore it is beneficial to use another evaluation metric such as k-fold cross validation to correctly measure the actual overall accuracy. A backward elimination algorithm is employed while constructing the virtual sensors to obtain the ideal number of input features. When the construction of virtual sensors are done and the residuals are calculated, the hyperparameters of the XGBoost algorithm are optimized utilizing grid search.

In comparison with the popular machine learning methods, the proposed method has shown a superior ability to make generalized predictions. If the mean accuracies are compared, the proposed method surpasses the second best model with 12.84% and 2.56% higher 9-fold and 23-fold accuracies, whereas only lacking behind 0.37% for the 15-fold accuracy. If it is compared with pure XGBoost algorithm, it can be seen that the accuracies are improved by 8.38%, 6.57% and 3.27% with the use of proposed model to the respective final values 67.81%, 70.53% and 71.61% for 9-fold, 23-fold and 15-fold tests. Thus, the virtual sensor residuals assisted XGBoost algorithm presents a significant improvement in making generalized predictions for refrigerant charge level, which means that it would have a considerable advantage in practical applications.

Finally, an online web application framework is constructed where the users can monitor the sensor readings and the health status of the chiller in real-time. The proposed model is uploaded to its dedicated server, where it would make predictions about the refrigerant charge level of the chiller according to the data and requests sent by the main server. The user interface and the data transfer routes of the online framework is thoroughly tested and the study is concluded with the deployment of the final online FDD web application.

6.2. Future prospects

Currently the model is developed only for diagnosing the refrigerant leakage fault of chillers, however, many faults can occur in the chiller after an extended service time. Hence, the model can be modified further to classify several types of faults and their corresponding fault severities. This task is in fact not very hard to achieve since the virtual sensor models are independent of fault type and severity, and the utilized XGBoost algorithm have already used the mode of multi-class classification to detect the fault severity. Therefore, with some minor adjustments, the model can

be easily modified to detect and diagnose multiple faults. The limiting factor in this study was the experiments that have been only conducted to collect the data for refrigerant undercharge fault of chiller. With the proper experimentation and data collection, the same model can be further applied detect and diagnose other faults.

Another area that the model can be applied for is the migration of the model to other chillers. Since the model have already achieved a solid generalization performance on different and unknown operating conditions of a chiller, the model can be also pushed further to be applied on another chiller. However, employing the same model on two or more different chillers requires an extensive preprocessing step which contains necessary standardization and domain adaptation steps for the input data. If the input data from different chillers can be sufficiently standardized, the model can be tested to obtain a general model which is able to detect and diagnose faults for many chillers.

The last area that can be improved further is the online framework. Firstly, since the applications are hosted on free shared servers, the REST API and the AI module are hosted separately. The free servers activates sleep mode if there are no incoming request for a while, and it takes a while for the servers to turn on when there is a new request. Therefore, the AI module sometimes have some delays to respond when a prediction request is made during the sleep mode. Moreover, the free cloud database currently operates in overseas servers, thus an access to database also adds additional delays to the process. As a solution to this, a dedicated commercial server can be hired to host all the backend and database software together. Which will drastically cut the communication times between the modules. Finally, the security of the framework can be further improved. Although the CRUD requests are currently secured with use of access tokens, an additional security layer can be added with use of JSON web tokens.

Acknowledgements

I have received support and assistance in various forms during my studies and at the time of writing this thesis. I would like to express my gratitude to everyone who were by my side in the meantime. No matter how many times I say thank you, it will never be enough.

Particularly, I would like to thank my supervisor, Professor Du Zhimin, for his continuous assistance and thorough guidance. It was his support and encouragement that inspired me to push myself for advancing my scientific knowledge and seeking for academic accomplishments. Furthermore, I would like the recognize the priceless knowledge and experience that I have learned from Professor Jin Xinqiao and Professor Gu Bo. My gratitude to them is beyond any words.

I'm sincerely grateful to my friends and colleagues: Zhu Xu, Chen Kang, Xue Yangfan, Li Pengcheng, Zhang Shuai, Huang Wei, Liu Zhurong, Lv Yuan, Yunus Emre Candir, Kadir Aslan, Mustafa Bozkurt Gursoy, Atakan Angun, Yilmaz Genc, Anil Karatopak, Canberk Gazioglu, Emre Berk Eski, Egemen Ertugrul, and countless others to mention. I'm deeply grateful to you all for sharing the joy, laughter, friendship and much more with me in my postgraduate life.

Lastly, I'm greatly thankful to my father, my mother, my brother and my beloved wife. I can never truly express how grateful I am to my family for their never-ending support in this long journey. My deepest thanks to you for always being with together me.

Publications

Published

1. Zhu X, Chen K, Anduv B, Jin X, Du Z. Transfer learning based methodology for migration and application of fault detection and diagnosis between building chillers for improving energy efficiency[J]. Building and Environment, 2021, 200:107957.
2. Zhang S, Zhu X, Anduv B, Jin X, Du Z. Fault detection and diagnosis for the screw chillers using multi-region XGBoost model[J]. Science and Technology for the Built Environment, 2021, 27(5):608-623.

Submitted

1. Li P, Anduv B, Zhu X, Jin X, Du Z. Diagnosis for the refrigerant undercharge fault of chiller using DBN enhanced extreme learning machine[J]. Sustainable Energy Technologies and Assessments, 2021.
2. Li P, Liu Z, Anduv B, Zhu X, Jin X, Du Z. Diagnosis for multiple faults of chiller using ELM-KNN model enhanced by multi-label learning and specific feature combinations[J]. Sustainable Energy Technologies and Assessments, 2021.

To Be Submitted

1. Anduv B, Zhu X, Li P, Jin X, Du Z. Refrigerant undercharge fault diagnosis of chiller using virtual sensor residuals assisted XGBoost model[J]. Applied Thermal Engineering, 2021.

2. Chen K, Zhu X, Anduv B, Jin X, Du Z. Digital twins model and its updating method for HVAC system using broad learning system algorithm[J]. Sustainable Energy Technologies and Assessments, 2021.

References

[1] Li H, Wang S. Coordinated optimal design of zero/low energy buildings and their energy systems based on multi-stage design optimization[J]. Energy. 2019, 189:116202.

[2] Proctor J A C. Performance Associated with AB970[C]. Presentation to the California Energy Commission, 2002.

[3] Evans N T. U.S. Patent Application No. 11/895[P], 2008, 154.

[4] Ahamed J U, Saidur R, Masjuki H H. A review on exergy analysis of vapor compression refrigeration system. Renewable and Sustainable Energy Reviews. 2011, 15(3):1593–1600.

[5] Yang H, Zhang T, Li H, Woradechjumroen D, Liu X. HVAC Equipment, Unitary: Fault Detection and Diagnosis[J]. Encyclopedia of Energy Engineering and Technology, 2014, 854–864.

[6] Isermann R. Model-based fault detection and diagnosis-status and applications[C]. IFAC Proceedings Volumes. 2004, 37(6):49–60.

[7] Babujee Jerome R. Pre-processing techniques for anomaly detection in telecommunication networks[D]. Aalto University, 2015.

[8] Araya D B, Grolinger K, ElYamany H F, Capretz M A M, Bitsuamlak G. An ensemble learning framework for anomaly detection in building energy consumption[J]. Energy and Buildings, 2017, 144:191-206.

[9] Chiang L H, Russell E L, Braatz R D. Fault detection and diagnosis in industrial systems[J]. Springer Science & Business Media, 2000.

[10] Xu G. HVAC system study: a data-driven approach[D]. The University of Iowa, 2012.

[11] Yu Y, Woradechjumroen D, Yu D. A review of fault detection and diagnosis methodologies on air-handling units[J]. Energy and Buildings, 2014, 82:550-562.

[12] Zhang Y, Jiang J. Bibliographical review on reconfigurable fault-tolerant control systems[C]. IFAC Proceedings Volumes, 2003, 36(5):257-268.

[13] Turner W J N, Staino A, Basu B. Residential HVAC fault detection using a system identification approach[J]. Energy and Buildings, 2017, 151:1-17.

[14] Iyengar S, Lee S, Irwin D, Shenoy P, Weil B. Watthome: A data-driven approach for energy efficiency analytics at city-scale[C]. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, July, pp.396-405.

[15] Zhang C, Cao L, Romagnoli A. On the feature engineering of building energy data mining[J]. Sustainable cities and society, 2018, 39:508-518.

[16] Ebrahimifakhar A, Kabirikopaei A, Yuill D. Data-driven fault detection and diagnosis for packaged rooftop units using statistical machine learning classification methods[J]. Energy and Buildings, 2020, 225:110318.

[17] Alzghoul A, Backe B, Löfstrand M, Byström A, Liljedahl B. Comparing a knowledge-based and a data-driven method in querying data streams for system fault detection: A hydraulic drive system application[J]. Computers in industry, 2014, 65(8):1126-1135.

[18] Angeli C. Diagnostic expert systems: From expert's knowledge to real-time systems[J]. Advanced knowledge based systems: Model, applications & research, 2010, 1:50-73.

[19] Tidriri K, Chatti N, Verron S, Tiplica T. Bridging data-driven and model-based approaches for process fault diagnosis and health monitoring: A review of researches and future challenges[J]. Annual Reviews in Control, 2016, 42:63-81.

[20] Ding S X. Data-driven design of model-based fault diagnosis systems[C]. IFAC Proceedings Volumes, 2012, 45(15):840-847.

[21] Khorasgani H, Farahat A, Ristovski K, Gupta C, Biswas G. A framework for unifying model-based and data-driven fault diagnosis.[C] In PHM society conference, 2018, September, 10(1).

[22] Chen Y, Wen J, Chen T, Pradhan O. Bayesian Networks for Whole Building Level Fault Diagnosis and Isolation[C]. International High Performance Buildings Conference, 2018, no.3214, p.10.

[23] Frank S, Heaney M, Jin X, Robertson J, Cheung H, Elmore R, Henze G. Hybrid model-based and data-driven fault detection and diagnostics for commercial buildings (No. NREL/CP-5500-65924)[R]. National Renewable Energy Lab.(NREL), Golden, CO (United States), 2016.

[24] Fan C, Xiao F, Li Z, Wang J. Unsupervised data analytics in mining big building operational data for energy efficiency enhancement: A review[J]. Energy and Buildings, 2018, 159:296-308.

[25] Shalev-Shwartz S, Ben-David S. Understanding machine learning: From theory to algorithms[M]. Cambridge university press, 2014.

[26] Jiawei H, Micheline K, Jian P. Data Mining Concepts and Techniques, Third Edit[M]. Waltham: Morgan Kaufmann, 2012.

[27] Namburu S M, Azam M S, Luo J, Choi K, Pattipati K R. Data-driven modeling, fault diagnosis and optimal sensor selection for HVAC chillers[J]. IEEE transactions on automation science and engineering, 2007, 4(3):469-473.

[28] Han H, Gu B, Hong Y, Kang J. Automated FDD of multiple-simultaneous faults (MSF) and the application to building chillers[J]. Energy and Buildings, 2011, 43(9):2524-2532.

[29] Han H, Gu B, Wang T, Li Z R. Important sensors for chiller fault detection and diagnosis (FDD) from the perspective of feature selection and machine learning[J]. International journal of refrigeration, 2011, 34(2):586-599.

[30] Han H, Gu B, Kang J, Li Z R. Study on a hybrid SVM model for chiller FDD applications[J]. Applied Thermal Engineering, 2011, 31(4):582-592.

[31] Tran D A T, Chen Y, Ao H L, Cam H N T. An enhanced chiller FDD strategy based on the combination of the LSSVR-DE model and EWMA control charts[J]. International Journal of Refrigeration, 2016, 72:81-96.

[32] Tran D A T, Chen Y, Jiang C. Comparative investigations on reference models for fault detection and diagnosis in centrifugal chiller systems[J]. Energy and Buildings, 2016, 133:246-256.

[33] Han H, Cui X, Fan Y, Qing H. Least squares support vector machine (LS-SVM)-based chiller fault diagnosis using fault indicative features[J]. Applied Thermal Engineering, 2019, 154:540-547.

[34] Wang P, Gao R X. Automated performance tracking for heat exchangers in HVAC[J]. IEEE Transactions on Automation Science and Engineering, 2017, 14(2):634-645.

[35] Bailey M B, Kreider J F. Creating an automated chiller fault detection and diagnostics tool using a data fault library[J]. ISA transactions, 2003, 42(3): 485-495.

[36] Hou Z, Lian Z, Yao Y, Yuan X. Data mining based sensor fault diagnosis and validation for building air conditioning system[J]. Energy Conversion and Management, 2006, 47(15-16):2479-2490.

[37] Fernandez N E, Katipamula S, Wang W, Xie Y, Zhao M, & Corbin C D. Impacts of commercial building controls on energy savings and peak load reduction (No. PNNL-25985)[R]. Pacific Northwest National Lab.(PNNL), Richland, WA (United States), 2017.

[38] Zhao Y, Wen J, Wang S. Diagnostic Bayesian networks for diagnosing air handling units faults–Part II: Faults in coils and sensors[J]. Applied Thermal Engineering, 2015, 90:145-157.

[39] Du Z, Fan B, Jin X, Chi J. Fault detection and diagnosis for buildings and HVAC systems using combined neural networks and subtractive clustering analysis[J]. Building and Environment, 2014, 73:1-11.

[40] Morisot O, Marchio D. Fault detection and diagnosis on HVAC variable air volume system using artificial neural network[C]. Proc. IBPSA Building Simulation, 1999.

[41] Capozzoli A, Lauro F, Khan I. Fault detection analysis using data mining techniques for a cluster of smart office buildings[J]. Expert Systems with Applications, 2015, 42(9):4324-4338.

[42] Yu Z J, Haghighat F, Fung B C, Zhou L. A novel methodology for knowledge discovery through mining associations between building operational data[J]. Energy and Buildings, 2012, 47:430-440.

[43] Jalori S, Reddy T A. A Unified Inverse Modeling Framework for Whole-Building Energy Interval Data: Daily and Hourly Baseline Modeling and Short-Term Load Forecasting[J]. ASHRAE Transactions, 2015, 121(2).

[44] Elnour M, Meskin N, Al-Naemi M. Sensor data validation and fault diagnosis using Auto-Associative Neural Network for HVAC systems[J]. Journal of Building Engineering, 2020, 27:100935.

[45] Du Z, Fan B, Chi J, Jin X. Sensor fault detection and its efficiency analysis in air handling unit using the combined neural networks[J]. Energy and Buildings, 2014, 72:157-166.

[46] Ding S X. Model-based fault diagnosis techniques: design schemes, algorithms, and tools[M]. Springer Science & Business Media, 2008.

[47] Magoulès F, Zhao H X, Elizondo D. Development of an RDP neural network for building energy consumption fault detection and diagnosis[J]. Energy and Buildings, 2013, 62:133-138.

[48] Jin B, Li D, Srinivasan S, Ng S K, Poolla K, Sangiovanni-Vincentelli A. Detecting and diagnosing incipient building faults using uncertainty information from deep neural networks[C]. In 2019 IEEE International Conference on Prognostics and Health Management (ICPHM), 2019, June, (pp. 1-8). IEEE.

[49] Guo Y, Tan Z, Chen H, Li G, Wang J, Huang R, ..., Ahmad T. Deep learning-based fault diagnosis of variable refrigerant flow air-conditioning system for building energy saving[J]. Applied Energy, 2018, 225:732-745.

[50] Shahnazari H, Mhaskar P, House J M, Salsbury T I. Modeling and fault diagnosis design for HVAC systems using recurrent neural networks[J]. Computers & Chemical Engineering, 2019, 126:189-203.

[51] Najafi M. Fault detection and diagnosis in building HVAC systems[D]. UC Berkeley, 2010.

[52] Beghi A, Brignoli R, Cecchinato L, Menegazzo G, Rampazzo M, Simmini F. Data-driven fault detection and diagnosis for HVAC water chillers[J]. Control Engineering Practice, 2016, 53:79-91.

[53] Yan K, Zhong C, Ji Z, Huang J. Semi-supervised learning for early detection and diagnosis of various air handling unit faults[J]. Energy and Buildings, 2018, 181:75-83.

[54] Yan K, Huang J, Shen W, Ji Z. Unsupervised learning for fault detection and diagnosis of air handling units[J]. Energy and Buildings, 2020. 210:109689.

[55] Mann A K, Kaur N. Review paper on clustering techniques[J]. Global Journal of Computer Science and Technology, 2013.

[56] Saxena A, Prasad M, Gupta A, Bharill N, Patel O P, Tiwari A, ..., Lin C T. A review of clustering techniques and developments[J]. Neurocomputing, 2017, 267:664-681.

[57] Ingle M G, Suryavanshi N Y. Association rule mining using improved Apriori algorithm[J]. International Journal of Computer Applications, 2015, 112(4).

[58] Yairi T, Kato Y, Hori K. Fault detection by mining association rules from house-keeping data[C]. In proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space, 2001, June, (Vol. 18, p. 21).

[59] Mueen A. Time series motif discovery: dimensions and applications[J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2014, 4(2):152-159.

[60] Fan C, Xiao F, Madsen H, Wang D. Temporal knowledge discovery in big BAS data for building energy management[J]. Energy and Buildings, 2015, 109:75-89.

[61] Xue P, Zhou Z, Fang X, Chen X, Liu L, Liu Y, Liu J. Fault detection and operation optimization in district heating substations based on data mining techniques[J]. Applied energy, 2017, 205:926-940.

[62] Li D, Hu G, Spanos C J. A data-driven strategy for detection and diagnosis of building chiller faults using linear discriminant analysis[J]. Energy and Buildings, 2016, 128:519-529.

[63] Zhao Y, Zhang C, Zhang Y, Wang Z, Li J. A review of data mining technologies in building energy systems: Load prediction, pattern identification, fault detection and diagnosis[J]. Energy and Built Environment, 2020, 1(2):149-164.

[64] Krarti M. An overview of artificial intelligence-based methods for building energy systems[J]. J. Sol. Energy Eng., 2003, 125(3):331-342.

[65] Kezunovic M, Rikalo I. Detect and classify faults using neural nets[J]. IEEE Computer Applications in Power, 1996, 9(4):42-47.

[66] Li S, Wen J. Application of pattern matching method for detecting faults in air handling unit system[J]. Automation in Construction, 2014, 43:49-58.

[67] Dey M, Rana S P, Dudley S. Smart building creation in large scale HVAC environments through automated fault detection and diagnosis[J]. Future Generation Computer Systems, 2020, 108:950-966.

[68] Amin-Naseri M R, Soroush A R. Combined use of unsupervised and supervised learning for daily peak load forecasting[J]. Energy conversion and management, 2008, 49(6):1302-1308.

[69] Piscitelli M S, Mazzarelli D M, Capozzoli A. Enhancing operational performance of AHUs through an advanced fault detection and diagnosis process based on temporal association and decision rules[J]. Energy and Buildings, 2020, 226:110369.

[70] Kalhori S R N. Improvement the accuracy of six applied classification algorithms through integrated supervised and unsupervised learning approach[J]. Journal of Computer and Communications, 2014, 2(04):201.

[71] Navarro-Esbri J, Torrella E, Cabello R. A vapour compression chiller fault detection technique based on adaptative algorithms. Application to on-line refrigerant leakage detection[J]. International Journal of Refrigeration, 2006, 29(5):716-723.

[72] Kim W, Braun J E. Performance evaluation of a virtual refrigerant charge sensor[J]. International journal of refrigeration, 2013, 36(3):1130-1141.

[73] Zhao Y, Wang S, Xiao F. A statistical fault detection and diagnosis method for centrifugal chillers based on exponentially-weighted moving average control charts and support vector regression[J]. Applied Thermal Engineering, 2013, 51(1-2):560-572.

[74] Zhu X, Du Z, Chen Z, Jin X, Huang X. Hybrid model based refrigerant charge fault estimation for the data centre air conditioning system[J]. International Journal of Refrigeration, 2019. 106:392-406.

[75] Zhu X, Chen K, Anduv B, Jin X, Du Z. Transfer learning based methodology for migration and application of fault detection and diagnosis between building chillers for improving energy efficiency[J]. Building and Environment, 2021, 200:107957.

[76] Trane Europe, Tracer UC800[EB/OL], trane.com/commercial/europe/hr/en/controls/equipment-controllers/trane-controllers/tracer-uc-800

[77] Wang X, Kruger U, Irwin G W. Process monitoring approach using fast moving window PCA[J]. Industrial & engineering chemistry research, 2005, 44(15):5691-5702.

[78] Freedman D, Pisani R, Purves R. Statistics: Fourth International Student Edition[M], 2020.

[79] Quinlan J R. Induction of decision trees[J]. Machine learning, 1986, 1(1):81-106.

[80] Breiman L. Bagging predictors[J]. Machine learning, 1996, 24(2):123-140.

[81] Breiman L. Random Forests[J]. Machine Learning. 2001, 45 (1):5–32.

[82] Freund Y, Schapire R E. Experiments with a new boosting algorithm[C]. In ICML, 1996, July, (Vol. 96, pp. 148-156).

[83] Friedman J H. Greedy function approximation: a gradient boosting machine[J]. Annals of statistics, 2001, 1189-1232.

[84] Chen T, Guestrin C. Xgboost: A scalable tree boosting system[C]. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, Aug, 13, (pp. 785-794).

[85] Lawson B, Sharp R. Introducing HTML5[M]. New Riders, 2011.

[86] Meyer E A. CSS: The Definitive Guide: The Definitive Guide[M]. O'Reilly Media, Inc., 2006.

[87] Guha A, Saftoiu C, Krishnamurthi S. The essence of JavaScript[C]. In European conference on Object-oriented programming. Springer, Berlin, Heidelberg, 2010, June, (pp. 126-150).

[88] React. A JavaScript Library for Building User Interfaces[EB/OL], reactjs.org/.

[89] Ant Design. The World's Second Most Popular React UI Framework[EB/OL], ant.design/.

[90] Mapbox. Maps, Geocoding, and Navigation Apis[EB/OL], mapbox.com/.

[91] Less. It’s CSS, with just a little more[EB/OL], lesscss.org/

[92] Yarn. Package Manager[EB/OL], yarnpkg.com/

[93] Node.js. A JavaScript runtime built on Chrome's V8 JavaScript engine[EB/OL], nodejs.org/.

[94] Express. Node.js Web Application Framework[EB/OL], expressjs.com/.

[95] Flask. Web Development, One Drop at a Time[EB/OL], flask.palletsprojects.com/en/2.0.x/.

[96] PyTorch. An open source machine learning framework[EB/OL], pytorch.org/

[97] MongoDB. Managed MongoDB Hosting: Database-as-a-Service[EB/OL], mongodb.com/cloud/atlas.

[98] Raspberry Pi. Tech Learn and Make with Raspberry Pi[EB/OL], raspberrypi.org/

[99] ESP8266. ESP8266 Wi-Fi MCU, Espressif Systems[EB/OL], espressif.com/en/products/socs/esp8266

[100] MicroPython. Python for microcontrollers[EB/OL], micropython.org/

[101] Heroku. Cloud Application Platform[EB/OL], heroku.com/

[102] Netlify. Develop & deploy the best web experiences in record time[EB/OL], netlify.com/

Appendix

1. Backward elimination algorithm iterations 1, 2 and n for obtaining the dropping sequence for a single sensor.

2. Exact Greedy and Approximate Algorithm for split finding

Algorithm 1: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node
Input: d, feature dimension
$gain \leftarrow 0$
$G \leftarrow \sum_{i \in I} g_i$, $H \leftarrow \sum_{i \in I} h_i$
for $k = 1$ ***to*** m **do**
  $G_L \leftarrow 0$, $H_L \leftarrow 0$
  for j *in sorted(I, by* $\mathbf{x}_{jk}$*)* **do**
    $G_L \leftarrow G_L + g_j$, $H_L \leftarrow H_L + h_j$
    $G_R \leftarrow G - G_L$, $H_R \leftarrow H - H_L$
    $score \leftarrow \max(score, \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{G^2}{H+\lambda})$
  end
end
Output: Split with max score

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ ***to*** m **do**
  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.
  Proposal can be done per tree (global), or per split(local).
end
for $k = 1$ ***to*** m **do**
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
end
Follow same step as in previous section to find max score only among proposed splits.

3. File tree of the front end framework (excluding node_modules and build folders).

```
client/
├─ public/
│  ├─ index.html
│  ├─ assets/
│  │  ├─ icon/
│  │  │  ├─ icon.jpg
│  │  ├─ images/
│  │  │  ├─ SJTU-bg.jpg
│  │  │  ├─ logo.jpg
├─ src/
│  ├─ index.jsx
│  ├─ App.jsx
│  ├─ views/
│  │  ├─ LoginPage.jsx
│  │  ├─ HomePage.jsx
```

```
16- │   │   ├─ AtlasPage.jsx
17- │   │   ├─ GraphPage.jsx
18- │   │   ├─ OperatorPage.jsx
19- │   │   ├─ UnavailablePage.jsx
20- │   ├─ components/
21- │   │   ├─ home/
22- │   │   │   ├─ AreaChart.jsx
23- │   │   │   ├─ ColumnChart.jsx
24- │   │   │   ├─ GaugeChart.jsx
25- │   │   │   ├─ MultiLineChart.jsx
26- │   │   │   ├─ RingProgressChart.jsx
27- │   │   │   ├─ SensorTable.jsx
28- │   │   ├─ atlas/
29- │   │   │   ├─ ChillerForm.jsx
30- │   │   │   ├─ ChillerInfo.jsx
31- │   │   │   ├─ ChillerMap.jsx
32- │   │   │   ├─ ChillerTable.jsx
33- │   │   ├─ graph/
34- │   │   │   ├─ DataQueryForm.jsx
35- │   │   │   ├─ FetchStatusChart.jsx
36- │   │   │   ├─ LargeGraph.jsx
37- │   │   │   ├─ SensorGraphList.jsx
38- │   │   ├─ operator/
39- │   │   │   ├─ OperatorAddChillerForm.jsx
40- │   │   │   ├─ OperatorAddUserForm.jsx
41- │   │   │   ├─ OperatorChillerList.jsx
42- │   │   │   ├─ OperatorDeleteChillerForm.jsx
43- │   │   │   ├─ OperatorDeleteUserForm.jsx
44- │   │   │   ├─ OperatorUpdateChillerForm.jsx
45- │   │   │   ├─ OperatorUserList.jsx
46- │   │   ├─ SidebarMenu.jsx
47- │   │   ├─ SelectChillerForm.jsx
48- │   │   ├─ PageHeader.jsx
49- │   │   ├─ PageFooter.jsx
50- │   │   ├─ ChillerHealth.jsx
51- │   ├─ context/
52- │   │   ├─ Context.jsx
53- │   │   ├─ Actions.jsx
54- │   │   ├─ Reducer.jsx
55- │   ├─ functions/
56- │   │   ├─ roundSensorData.jsx
57- │   │   ├─ useLocalStorege.jsx
58- │   ├─ assets/
59- │   │   ├─ schematic.svg
60- │   ├─ constants/
61- │   │   ├─ actionTypes.jsx
62- │   ├─ styles/
63- │   │   ├─ App.less
64- │   │   ├─ pageStyles/
65- │   │   │   ├─ homePage.less
66- │   │   ├─ componentStyles/
67- │   │   │   ├─ chillerForm.less
68- │   │   │   ├─ chillerMap.less
69- │   │   │   ├─ pageHeader.less
```

```
70- │   │   │   ├─ sidebarMenu.less
71- ├─ .env
72- ├─ .eslintrc.js
73- ├─ craco-config.js
74- ├─ package-lock.json
75- ├─ package.json
76- ├─ yarn.lock
```

4. File tree of the back end framework (excluding node_modules and build folders)

```
1- api/
2- ├─ models/
3- │   ├─ User.js
4- │   ├─ Chiller.js
5- │   ├─ SensorData.js
6- │   ├─ RefLoadPred.js
7- ├─ routes/
8- │   ├─ user.js
9- │   ├─ chiller.js
10- │   ├─ sensor.js
11- │   ├─ auth.js
12- │   ├─ pred.js
13- │   ├─ rename.js
14- ├─ index.js
15- ├─ .env
16- ├─ .eslintrc.js
17- ├─ package.json
18- ├─ yarn.lock
19- ├─ Procfile
```

5. File tree of machine learning module

```
1- ai-module/
2- ├─ app.py
3- ├─ fdd_main.py
4- ├─ ffm_predict.py
5- ├─ utils.py
6- ├─ model_state/
7- │   ├─ XGB_model.sav
8- │   ├─ ffm_model_DATA_COLUMN.sav*
9- │   ├─ input_data_scaler_ffm_DATA_COLUMN.bin*
10- │   ├─ input_data_scaler_ffm_k.bin
11- │   ├─ input_data_scaler_ffm_r.bin
12- │   ├─ input_data_scaler_r.bin
13- │   ├─ target_data_scaler_ffm_DATA_COLUMN.bin*
14- │   ├─ target_data_scaler_ffm_k.bin
15- │   ├─ target_data_scaler_ffm_r.bin
16- │   ├─ target_data_scaler_r.bin`
17- ├─ feature_data/
```

```
18-│   ├─ ffm_features.txt
19-│   ├─ kept_columms.txt
20-├─ Pipfile
21-├─ Pipfile.lock
22-├─ Procfile
```

* DATA_COLUMN denotes a compressed illustration of the 11 separate files, namely: comp_DiscSupe, comp_DischTemp, comp_MassFlow, cond_DischSubc, cond_SatRfgt_Temp, evap_ApprchTemp, evap_EntWtrTemp, evap_LvgWtrTemp, evap_OutdoorAirTemp, evap_Sat_RfgtTemp, evapWtrFlowEsti. These files would have the same prefix as shown in the file tree. Therefore, the actual file names are: ffm_model_comp_DiscSupe.bin, ffm_model_comp_DischTemp.bin, etc.

上海交通大学

学位论文原创性声明

本人郑重声明：所呈交的学位论文《基于数字孪生的冷水机组故障检测与诊断研究》，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：2021年01月13日

上海交通大学

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

保密□，在___年解密后适用本授权书。

本学位论文属于

不保密☑。

（请在以上方框内打“√”）

学位论文作者签名：

日期：2022年01月13日

指导教师签名：

日期：2022年01月

:9001989

上海交通大学 硕士 学位论文答辩决议书

119020990024

姓 名	BURKAY ANDUV	学 号	119020990024	所在学科	动力工程及工程热物理
指导教师	杜志敏	答辩日期	2022-1-12	答辩地点	机动A701
论文题目	Cloud-based Diagnosis of Refrigerant Leakage Fault of Chiller Using Virtual Sensor Residuals Assisted XGBoost Algorithm				

投票表决结果：5 / 5 （同意票数/实到委员数/应到委员数） 答辩结论：☑通过 ☐未通过

评语和决议：Theoretical value and social significance: The master's degree thesis of the oponse resolution is based on dignosis of refrigerant leakage fault of chiller, which is conducted via virtual sensor residuals assisted XGBoost algorithm. The proposed methodology have a higher generalization capacity compared to conventional methods. Moreover, a cloud based fault detection and diagnosis framework is developed to conduct the diagnosis operation online, which has theoretical and practical value.

Main content: First, experimental procedure with different fault scenarios are specified. Second, the initial data filtering and feature selection methods are presented. Third, the input data processing method using virtual sensor residuals is developed. Then, the selected virtual sensor residuals are supplied to the XGBoost algorithm and fault diagnosis results are obtained. Next, an online framework for real time fault diagnosis is developed. Finally, the proposed method is verified by comparing its results with the results from conventional algorithms.

The performance of the defense: The paper is well-justified, the workload is full, and the theoretical analysis and experimental results are credible. The thesis shows that the author already has the theoretical basis and relevant professional knowledge of the subject, has the ability of comprehensive analysis and scientific research, and meets the requirements of the master's degree thesis. Explain clearly in the course of the defense and answer the question correctly. All the members agree that the student should reply through the master's thesis and recommended the granting of a master's degree in engineering.

2022年 1月 12日

答辩委员会成员	职务	姓名	职称	单位	签名
	主席	黄永华	研究员	上海交通大学	
	委员	林文胜	副教授	上海交通大学	
	委员	丁国良	教授	上海交通大学	
	委员	张春路	教授	同济大学	
	委员	马涛	副教授	上海交通大学	
	秘书	柯霞	工程师	上海交通大学	