*energies*

MDPI

*Article*

# Fault Diagnosis of Tennessee Eastman Process with XGB-AVSSA-KELM Algorithm

**Mingfei Hu [1], Xinyi Hu [1], Zhenzhou Deng [2] and Bing Tu [3,*]**

[1] Qianhu College, Nanchang University, Nanchang 330031, China; mingfeihu@email.ncu.edu.cn (M.H.); xinyihu@email.ncu.edu.cn (X.H.)
[2] Information Engineering College, Nanchang University, Nanchang 330031, China; zzdeng@ncu.edu.cn
[3] Chongqing Research Institute, Nanchang University, Chongqing 402660, China
* Correspondence: tubing@ncu.edu.cn; Tel.: +86-158-0700-6813

**Abstract:** In fault detection and the diagnosis of large industrial systems, whose chemical processes usually exhibit complex, high-dimensional, time-varying and non-Gaussian characteristics, the classification accuracy of traditional methods is low. In this paper, a kernel limit learning machine (KELM) based on an adaptive variation sparrow search algorithm (AVSSA) is proposed. Firstly, the dataset is optimized by removing redundant features using the eXtreme Gradient Boosting (XGBOOST) model. Secondly, a new optimization algorithm, AVSSA, is proposed to automatically adjust the network hyperparameters of KELM to improve the performance of the fault classifier. Finally, the optimized feature sequences are fed into the proposed classifier to obtain the final diagnosis results. The Tennessee Eastman (TE) chemical process is used to verify the effectiveness of the proposed method through multidimensional diagnostic metrics. The results show that our proposed diagnosis method can significantly improve the accuracy of TE process fault diagnosis compared with traditional optimization algorithms. The average diagnosis rate for 21 faults was 91.00%.

**Keywords:** fault diagnosis; Tennessee Eastman process; KELM;XGBOOST; AVSSA; feature selection



**Citation:** Hu, M.; Hu, X.; Deng, Z.; Tu, B. Fault Diagnosis of Tennessee Eastman Process with XGB-AVSSA-KELM Algorithm. *Energies* **2022**, *15*, 3198. https://doi.org/10.3390/en15093198

Academic Editor: Ahmed Abu-Siada

Received: 15 March 2022
Accepted: 25 April 2022
Published: 27 April 2022



## 1. Introduction

Chemical process equipment has evolved over time to become more automated, complicated, and intelligent. Chemical processes frequently produce hazardous circumstances, such as toxic corrosion, as well as complicated equipment, making safety management difficult [1–3]. Furthermore, because the systems in the chemical process are intimately linked, a failure of one portion of the system can result in the entire system failing. Chemical process accidents result in numerous incidents of significant property and social losses each year [4–7]. As a result, there is a pressing need in the chemical industry to develop an intelligent fault detection and diagnosis system. However, the chemical process encompasses both the chemical industry and other process industries, a scope that is too broad to be discussed conveniently. Therefore, this paper chooses to focus on the TE process and to propose a fault diagnosis method applied to TE process.

There are three types of fault detection and diagnosis systems: quantitative, qualitative, and historical process data models [8–10]. Fault diagnostic methods based on historical process data, for example, process data or signals directly without the need for a complicated and exact mathematical model, avoiding a huge amount of prior information and time-consuming and difficult mathematical processes [11]. The amount of data available to researchers has expanded significantly as a result of the information and big data era, making it challenging for quantitative and qualitative methodologies to meet the requirements in terms of diagnosis accuracy and speed. As a result, data-driven fault diagnosis approaches are common in the chemical sector.

Data-based diagnostic methods and machine-learning-based diagnostic methods are the two broad categories of data-based diagnostic approaches [12]. Principal component

analysis [13], partial least squares [14], independent component analysis [15], and Fisher discriminant analysis [16] are examples of statistics-based approaches. Some dimension-reduction methods, such as multi-scale principal component analysis [17] and kernel Fisher discriminant analysis [18], have made advancements in terms of the highly nonlinear correlation between variables and very complicated processes in chemical systems. However, as the number of data dimensions grows, the complexity of these statistics-based methodologies grows exponentially, resulting in dimension disaster [19,20].

With the development of clever algorithms in recent years, machine learning methods have become increasingly popular in defect diagnosis systems, supporting artificial neural networks [21], support vector machines [22] and other machine learning methods.

In 1943, Mcculloch and Pitts demonstrated that a neuron is a simple gate-valued device that performs a logic function [23]. In 1958, Rosenblatt proposed the perceptron, the first feed-forward neural network, which is a binary linear classifier based on an artificial neural network [24]. In 1982, Hopfield, a physicist at Caltech, proposed the HNN model [25]. Backward propagation, the method of making vector sweeps of all neurons in a perceptron at the same time, was independently proposed by several scientists in 1986 [26]. Rumelhart and McClelland devised a theory based on parallel distributed processing and developed a multilayer network-based back-propagation learning technique [27].

Extreme learning machine was proposed by Huang in 2006 as a new feedforward neural network training approach [28]. Extreme learning machines can train hundreds of times faster than typical BP networks and support vector machines. Huang et al. created KELM in 2011, which uses kernel functions instead of feature mapping in the hidden layer of ELM to speed up learning and improve the learning model's stability and generalization [29].

Although progress has been achieved in the use of ELM in the detection of chemical process faults, there are still certain issues to be resolved. Some variables have a negligible effect on the results of the chemical process, while too many variables require computing resources. The correctness of the categorization model must be assessed after the process monitoring data have been obtained. According to long-term studies, many features are unrelated to the classification goals. Data features were sorted into three groups by John [30]: strongly correlated, weakly correlated, and irrelevant features. Feature selection involves finding a subset of features that can optimize and specify the evaluation criteria. Feature selection is applied in many fault diagnosis methods.

Among the commonly used feature selection algorithms, it is worth mentioning LDA, PCA, SVM, RFTB, GBDT, and so on. Ronald A. Fisher [31] in 1936 introduced the Linear Discriminant method, which is sometimes used to solve classification problems. The original linear discriminant was applied to binary classification problems and was later generalized in 1948 by C. R. Rao [32] to "multiclass linear discriminant analysis" or "multiple discriminant analysis".

Lau et al. [33] used multi-scale principal component analysis (MSPCA) and Deng et al. [34] used deep principal component analysis (DePCA) to improve the adverse effects of traditional PCA on feature selection.

Onel et al. [35,36] proposed a Nonlinear Support Vector Machine-based feature selection algorithm, and the nonlinear support vector machine model outperforms existing support vector machine models in the literature in terms of detection accuracy and latency. It also uses feature selection techniques with minimal information loss compared to feature extraction techniques such as PCA.

The random forest was introduced by Breiman [37] in 2001. Random forest is a classification tree-based algorithm that can be used for multivariate classification and regression. RF methods have several advantages over other more commonly used multiple regression or classification methods. For example, it can account for interactions and non-linearities between variables. In addition, it can provide information about the statistical weights of each variable in the overall results [38–40].

The GBDT algorithm was of interest to researchers for its powerful predictive performance when it was proposed with the gradient boosting principle by Friedman [41], but the GBDT algorithm is an extremely memory-consuming algorithm and speed became the most criticized aspect. After years of development, XGBoost was proposed by Chen [42], which is an efficient implementation of the GBDT algorithm with many algorithmic and engineering improvements. xGBoost has been widely used in major contexts. For example, in 2015, 17 of the 29 winning solutions posted on Kaggle's Blog 3 used XGBoost. Eight of these solutions used XGBoost alone to train their models, while most others combined XGBoost with neural networks. Further, the second most popular method, deep neural networks, was used in 11 solutions in the same year.

We felt it necessary to investigate the guiding implications of optimizing the KELM fault diagnosis feature selection sequence using the XGBoost algorithm, and the results of XGBoost, random forest tree bagger (RFTB) and Nonlinear Support Vector Machine (NSVM) feature extraction are later compared.

The Tennessee Eastman (TE) simulation platform is an open access, widely used platform to test control and troubleshooting models for complex industrial processes. Inspired by previous work, we propose a new method for chemical process fault diagnosis based on TE data. The method consists of three steps: feature extraction based on XGBOOST, the optimization-seeking algorithm AVSSA, and AVSSA-KELM fault detection based on fused depth features. The method flow is as follows:

(1) Facing the time-varying, strongly coupled and nonlinear chemical dataset, we use XGBOOST to remove redundant information while retaining most of the intrinsic and discriminative information to prevent features of different classes from overlapping in some regions of the feature space;
(2) A new algorithm AVSSA is proposed by introducing the adaptive variation process based on t distribution into SSA, which has a strong global search capability in the early iteration and a strong local search capability in the late iteration. We combine the AVSSA and KELM classifiers to automatically adjust the optimal network hyperparameters of KELM;
(3) Using the advantages of integrated learning and multi-core learning, AVSSA-KELM is proposed as the top classifier for fault diagnosis based on fused multi-domain features. The method has high generalization performance with good diagnostic capability and diagnostic speed.

This paper is structured as follows. Section 2 introduces the use of XGBoost for feature selection. Section 3 presents the methodology that introduces the principle of the AVSSA. Section 4 presents the methodology that introduces the principle of the KELM. Section 5 details the xgboost-based AVSSA-KELM fault diagnosis method. Section 6 describes the application of the proposed method to the TE process for verification. Section 7 compares the proposed method with other methods in terms of multidimensional indicators to demonstrate the superiority of the method. Section 8 provides a summary of this study.

## 2. eXtreme Gradient Boosting

Xgboost [43] (eXtreme Gradient Boosting) is a gradient-enhanced decision system whose base learner can be either a linear classifier or a tree, and in this paper we use the features of its tree model as a basis for quantifying the importance of each feature for feature selection.

### *2.1. Tree Model*

For a given dataset, in the process of tree model construction, each layer greedily selects a feature segmentation point as a leaf node to maximize the gain of the whole tree after segmentation, which means that the more times a feature is segmented, the greater the benefit of the feature to the whole tree and the more important the feature is. Similarly, the greater the average gain of the feature each time it is segmented, the more important

the feature is. The weight of each leaf node in the segmentation process can be expressed as $w(p_i, q_i)$, $p_i$ and $q_i$ are:

$$p_i = \partial_{\hat{z}^{(t-1)}} l\left(z_i, \hat{z}^{(t-1)}\right) q_i = \partial^2_{\hat{z}^{(t-1)}} l\left(z_i, \hat{z}^{(t-1)}\right), \tag{1}$$

where the training error $l(z_i, \hat{z}_i)$ represents the difference between the target value $z_i$ and the predicted value $\hat{z}_i$. In order to minimise the cost of the segmented tree, based on the weights of all leaf nodes and considering the gain of each feature as a segmentation point, there is:

$$\text{Gain } = \sum_{\text{left}} w + \sum_{\text{right}} w - \sum_{\text{nosplit}} w. \tag{2}$$

Equation (2) illustrates that, for each split point, its ground benefit can be expressed as the total weight after the split (the sum of the total weight of the left subtree of the leaf node and the total weight of the right subtree) minus the total weight of the leaf node before the split.

### *2.2. Combination of Tree Models*

As a non-parametric supervised learning model, the decision tree model is commonly used for classification, regression and feature extraction. The model does not require any a priori assumptions about the data and can quickly find decision rules based on the characteristics of the data. XGBoost uses an integration strategy based on the decision tree, using a gradient lift-off algorithm to continuously reduce the loss of the previously generated decision tree. XGBoost adds one tree at each iteration, and constructs a linear combination of K trees as:

$$\hat{z}_i^{(t)} = \sum_{k=1}^{K} f_k(x_i) = \hat{z}_i^{(t-1)} + f_t(x_i), f_k \in F, \tag{3}$$

where $F$ denotes the function space containing all trees, and $f_k(x_i)$ denotes the weight of the ith sample in the kth tree that is assigned to the leaf node where it is located.

### *2.3. Importance Metrics*

Importance metrics are a way to assess the importance of each feature in the feature set to which it belongs, and XGBoost constructs importance metrics based on the number of feature splits, FScore, the average gain value, AverageGain, and the average coverage of features, AverageCover. For the above three importance measures, there are:

$$\text{Weight} = |X|, \tag{4}$$

$$\text{Gain} = \sum \text{Gain}_X, \tag{5}$$

$$\text{Cover} = \sum \text{Cover}_X, \tag{6}$$

where $X$ is the set of the requested features classified to the leaf nodes: Weight is the node gain value at segmentation for each leaf node in $X$ obtained from Equation (2), and cover is the number of samples falling at each node in $X$.

### *2.4. XGBoost Feature Extraction Algorithm*

#### 2.4.1. Algorithm Construction

We classify each dataset according to XGBoost , which is constructed as follows.

(1) Starting from the root node. Traverse all features for each node;

(2) For feature $a_i \in A$, first sort by sample value $\left\{a_i^{(1)}, \cdots, a_i^{(n)}\right\}$,linearly scan a to determine the segmentation point with the best gain $\left(a_i^{(k)}, \text{gain}_i\right)$;

(3) Select the feature with the highest gain from all the selected feature segmentation points to segment $\max \text{Gain}\left(a_1^{(k_i)}, \cdots, a_m^{(k_n)}\right)$ and update the mapping relationship from the samples to the tree nodes;
(4) Segment all the way to the maximum depth, and calculate the residuals for constructing the next tree;
(5) Integrate all the generated trees to complete the construction of the final tree model;
(6) Derive the importance of the feature variables based on the number of decisions.

Pseudo-code for the basic steps, as shown in Algorithm 1.

**Algorithm 1:** XGBoost.

**Input:** Original set of features (X)
**Output:** Weight arrays of features (A), Gain array of features (B), Cover array of features (C)

depth ← 0
Initialize arrays A, B, C
Initialising the importance metric
Create root node
**while** *depth < Maximum tree depth* **do**
  **for** *for each* $x \in X$ **do**
    Calculating the best split point $x^k$;
    Create child nodes;
  **end**
  depth ← depth + 1;
**end**
Adding the created tree to the tree model according to equation (xxxx)
**for** *each tree* **do**
  **for** *each segmentation nodes* **do**
    Calculate the gain value (gain) and the number of samples (cover) that would have been generated by the split node
    Update the importance metric for each feature
    Weight ← Weight + 1, Gain ← Gain + gain, Cover ← Cover + cover.
  **end**
**end**

#### 2.4.2. The XGBoost Parameter and the Segmentation Criterion

The feature selection hyperparameters mainly include the XGBoost parameter and the segmentation criterion, which are specified as follows:

XGBoost parameter (default value if not specified):

max_depth: 7, this parameter is controlled as the maximum depth of the tree. This value is used to control overfitting. the larger the max_depth, the more specific the model learns.

n_estimators: 80, this parameter controls the maximum number of trees generated, and also the maximum number of iterations. This value has an effect on the number of features selected in feature selection, but this is largely avoided by the selection criteria of this algorithm.

learning_rate: 0.1, this parameter controls the step size of each iteration, too large for accuracy and too small for slow operation.

subsample: 1.0, this parameter controls the proportion of random samples for each tree. By decreasing the value of this parameter, the algorithm will be more conservative and avoid overfitting. However, if this value is set too small, it may lead to underfitting. In this paper, it is very easy to underfit with a smaller value, so we set it to 1.

colsample_bytree: 0.5, which controls the proportion of columns (each column is a feature) that are randomly sampled per tree.

Partitioning method:

The partitioning method uses weight arrays of features (A), gain arrays of features (B), and cover arrays of features (C) as the data source, with weight as the core and gain and cover as the auxiliary. The features are sorted by weight from highest to lowest, and the top N features with 80 percent of $\sum$Weight are taken as set $A_1$ . Then, gain and cover are sorted from smallest to largest, and the features with the lowest five percent of each are taken as sets $B_1$ and $C_1$.

Feature sequences after feature selection $= (A_1 \wedge \bar{B_1})V(A_1 \wedge \bar{C_1})$.

## 3. Adaptive Variation Sparrow Search Algorithm

The Sparrow Search Algorithm (SSA) is an intelligent bionic optimization algorithm proposed by Xue J and Shen B. In their study [44], they concluded that SSA has global search ability and strong adaptability by comparing SSA with algorithms such as GWO, PSO, GSA, etc. They also pointed out that most of the sparrows in most of the experiments on the tested functions clustered towards the global optimum and did not fall into local extremes. However, for example, with the Damavandi function, most of the sparrows converge on local minima. In order to increase the population diversity and avoid the algorithm easily falling into the local optimum, we can make the sparrow jump out of the local optimum position and improve the global search ability of the algorithm by introducing the mutation process of adaptive t-distribution. Thus, we propose our new algorithm called the Adaptive Variation Sparrow Search Algorithm (AVSSA).

### *3.1. Algorithmic Bionics Principles*

SSA is a new group intelligence optimization algorithm inspired by the foraging and anti-predatory behaviors of sparrows, and its bionic principle is as follows:

The sparrow population during foraging is divided into two parts: discoverers and followers, while scouts are also set up as an early warning mechanism. Discoverers are highly adapted and has a wide search range, guiding the population to search and forage. Followers follow the discoverers to forage for better adaptability. Meanwhile, sparrows also have anti-predatory behavior. When a sparrow searching on the periphery encounters a natural predator or other organism threatening the entire population, it will provide an early warning to the population to avoid the area and adjust its search location.

### *3.2. Sparrow Search Algorithm*

Discoverers are responsible for determining the direction and area of search for food, while foraging, and have a larger search area than followers and a larger proportion of the population, typically 60–70% of the population. Whether all sparrows can find food preferentially depends on the value of the fitness function of each individual. At the same time, discoverers are mostly at the periphery of the population and are responsible for early warnings about the surrounding environment, namely, when a predator is discovered, the discoverers need to lead the followers to a safe location to continue the search.

Discoverers' position updated:

$$x_{id}^{t+1} = \begin{cases} x_{id}^{t} \cdot exp(\frac{-i}{\alpha \cdot T}), R_2 < ST \\ x_{id}^{t} + Q \cdot L, R_2 \geq ST, \end{cases} \tag{7}$$

where $t$ is the current number of iterations; $T$ is the maximum number of iterations; $\alpha$ is a uniform random number of $(0, 1]$; $Q$ is the random number that obeys the standard normal distribution number of randomizers; $L$ is a matrix with $1 \times d$ elements of 1; $R_2 \in [0, 1]$ and $ST \in [0.5, 1]$ denote the warning value and safety value.

All remaining sparrows in the population acted as followers. If a discoverer has less food, namely a low fitness value, its current location is a poor foraging location, indicating that the current search direction is not suitable and it needs to change direction to another location. At the same time, as the followers follow the discoverer, when the discoverer finds food, followers can obtain some of the food or forage around the location.

Followers' position updated:

$$x_{id}^{t+1} = \begin{cases} Q \cdot exp(\frac{xw_d^t - x_{id}^t}{i^2}), i > \frac{n}{2} \\ xb_d^{t+1} + \left| x_{id}^t - xb_d^{t+1} \right| \cdot A^+ \cdot L, i \leq \frac{n}{2}, \end{cases} \tag{8}$$

where $x_{id}^t$ is the position of the sparrow in the $i$th dimension at the $t$th iteration; $xw_d^t$ is the worst position of the sparrow in the $d$th dimension at the $t$th iteration; $xb_d^{t+1}$ is the optimal position of the sparrow in the $d$th dimension at the $t+1$th generation of the population; $A$ is a $1 \times d$ matrix, where each element is randomly assigned to $-1$ or 1; $A^+ = A^T(AA^T)^{-1}$; $L$ is a matrix with $1 \times d$ elements of 1.

Some of the sparrows at the edge of the population are responsible for scouting, and they generally account for 10% to 20% of the population. In the face of the arrival of a threat, scouts can foresee the danger in time and send an alert to the population, at this time either the discoverers or followers need to adjust the search position away from the danger, into the safe area to search. At the same time, sparrows in the center of the population will move randomly, and when the alarm is issued, they will move closer to other sparrow positions to reduce the possibility of predation.

Scouts' position updated:

$$x_{id}^{t+1} = \begin{cases} xb_d^t + \beta(x_{id}^t - xb_d^t, f_i \neq fg) \\ x_{id}^t + K\frac{x_{id}^t - xw_d^t}{|f_i - f_w| + \epsilon}, f_i = f_g, \end{cases} \tag{9}$$

where $\beta$ is the control step parameter, obeying N(0, 1) random numbers; $K$ is a random number and $K \in [-1, 1]$, which indicates the orientation of the sparrow's movement, and is also a step control parameter; $\epsilon$ is a very small number to prevent the denominator from being 0; $f_i$ denotes the fitness value of the $i$th sparrow; $f_g$ and $f_w$ are the optimal and worst fitness values of the current sparrow population.

### *3.3. Adaptive Variation Process Based on t-Distribution*

The probability density function of the t-distribution is as follows:

$$p_t(y) = \frac{\Gamma\left(\frac{n+1}{2}\right)}{\sqrt{n\pi}\Gamma\left(\frac{n}{2}\right)}\left(1 + \frac{y^2}{n}\right)^{-\frac{n+1}{2}}, -\infty < y < \infty, \tag{10}$$

where $n$ is the degree of freedom.

In evolutionary planning, the Cauchy operator and Gaussian operator are the most commonly used variance operators, and the Cauchy and Gaussian distributions are special distributions at two boundary positions of the t-distribution. The standard Gaussian distribution density N (0, 1) has an expectation of 0 and a variance of 1. The standard Cauchy distribution probability density C (0, 1) has no expectation and the variance is infinite. According to the probability function of t distribution, t(n)→N(0, 1) when n→ ∞, namely, when n → ∞, t-distribution is standard Gaussian distribution; t(n)→C(0, 1) when $n = 1$, namely, when $n = 1$, t-distribution is the standard Cauchy distribution. So the advantage of t-distribution is that the t-operator $\varphi$ can be constructed, and the advantages of both types of operators can be exploited by bridging the Gaussian operator and the Cauchy operator through the t-operator.

In the variation operation, the original parameter values are replaced by a random number that fits a t-distribution with degrees of freedom equal to the number of iterations $k$. The variation formula is:

$$x_i^{'} = x_i + x_i \cdot \varphi(k), \tag{11}$$

where $x_i^{'}$ and $x_i$ are the positions of the $i$th sparrow before and after the t-distribution variation, respectively; $k$ is the number of iterations; $\varphi(k)$ is the t-operator with the number of iterations $k$ as the degrees of freedom.

The adaptive t-distribution mutation-based sparrow search algorithm takes the number of iterations $k$ of the sparrow algorithm is used as the degree of freedom of the t-distribution.

(1) When $k$ is small in the early iteration, the t-distribution is similar to the Cauchy mutation. The offspring produced by the Cauchy mutation have greater variability compared to the parent and are more perturbed than the Gaussian mutation, with a wider range of mutations, thanks to the Cauchy density function having a longer step length and a longer distribution at both ends giving it a higher probability of allowing the individual to escape the local optimum [45–47];

(2) When $k$ is large in the later iteration, the t-distribution resembles a Gaussian mutation. Since the peak of the Gaussian distribution curve is located where the mean value is, the search focuses on the local area near the original individual to enhance the local search capability [48–50].

This makes the t-distribution adaptive due to the change of iterations. Thus, we named this method the Adaptive Variation Sparrow Search Algorithm (AVSSA).

Based on the idealization and feasibility of the above model, the basic steps of the AVSSA can be summarized as the pseudo code shown in Algorithm 2.

**Algorithm 2:** The framework of AVSSA.

**Input:** $G$: the maximum iterations
$ND$: the number of discoverers
$NS$: the number of scouts
$R_2$: the alarm value
$n$: the number of sparrows
Initialize a population of $n$ sparrows and define its relevant parameters.
**Output:** $X_{best}$,$f_g$

**while** $t < G$ **do**
  Rank the fitness values and find the current best individual and the current worst individual.
  $R_2 = rand(1)$
  **for** $i = 1 : ND$ **do**
    Using Equation (1) update the sparrow's location;
  **end**
  **for** $i = ND + 1 : n$ **do**
    Using Equation (2) update the sparrow's location;
  **end**
  **for** $l = 1 : NS$ **do**
    Using Equation (3) update the sparrow's location;
  **end**
  Get the current new location;
  If the new location is better than before, update it;
  **for** $i = 1 : n$ **do**
    Using Equation (5) update the sparrow's location;
  **end**
  Get the current new location;
  If the new location is better than before, update it;
  $t = t + 1$
**end**
**return** $X_{best}$,$f_g$.

## 4. The Kernel Based Extreme Learning Machine

### *4.1. Extreme Learning Machine Overview*

A typical single implicit layer feedforward neural network structure consists of an input layer, an implicit layer and an output layer, with the input layer fully connected to the implicit layer and the implicit layer to the output layer neurons [51]. The input layer has $n$ neurons corresponding to $n$ input variables, the hidden layer has $l$ neurons, and the output layer has $m$ neurons corresponding to $m$ output variables. For the sake of generality, let the connection weights $\omega$ between the input and hidden layers be:

$$\omega = \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1n} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2n} \\ \dots & \dots & \dots & \dots \\ \omega_{l1} & \omega_{l2} & \dots & \omega_{ln}, \end{bmatrix} \tag{12}$$

where $\omega$ denotes the connection weight between the $i$th neuron in the input layer and the $j$th neuron in the hidden layer.

Let the connection weight between the implicit layer and the output layer be $\beta$ :

$$\beta = \begin{bmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1n} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2n} \\ \dots & \dots & \dots & \dots \\ \beta_{l1} & \beta_{l2} & \dots & \beta_{ln}, \end{bmatrix} \tag{13}$$

where self $\beta_{jk}$ denotes the connection weights between the $j$th neuron in the hidden layer and the $k$th neuron in the output layer.

Let the threshold value $b$ of the neuron in the hidden layer be:

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_l. \end{bmatrix} \tag{14}$$

Let the input matrix $X$ and output matrix $Y$ of the training set with $Q$ samples be:

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} \\ x_{21} & x_{22} & \dots & x_{2n} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nQ} \end{bmatrix} \tag{15}$$

$$Y = \begin{bmatrix} y_{11} & y_{12} & \dots & y_{1n} \\ y_{21} & y_{22} & \dots & y_{2n} \\ \dots & \dots & \dots & \dots \\ y_{m1} & y_{m2} & \dots & y_{mQ}. \end{bmatrix} \tag{16}$$

Let the activation function of the neurons in the hidden layer be $g(x)$, the output $T$ of the network is :

$$T = \begin{bmatrix} t_1 & t_2 & \dots & t_Q \end{bmatrix}_{n \times Q} \tag{17}$$

$$t_j = \begin{bmatrix} t_{1j} \\ t_{2j} \\ \dots \\ t_m j \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{t} \beta_{i1} g(\omega_i x_j + b_i) \\ \sum_{i=1}^{t} \beta_{i2} g(\omega_i x_j + b_i) \\ \dots \\ \sum_{i=1}^{t} \beta_{im} g(\omega_i x_j + b_i) \end{bmatrix}_{m \times 1}. \tag{18}$$

### *4.2. Kernel Based Extreme Learning Machine*

The Kernel Based Extreme Learning Machine (KELM) [52,53] is an improved algorithm based on the Extreme Learning Machine (ELM) combined with a kernel function.

ELM is a single implicit layer feedforward neural network whose learning objective function $F(x)$ can be represented by the matrix:

$$F(x) = h(x) \times \beta = H \times \beta = L, \tag{19}$$

where $x$ is the input vector, $h(x)$, $H$ is the output of the hidden layer nodes, $\beta$ is the output weight and $L$ is the desired output.

Turning the network training into a problem solved by a linear system, $\beta$ is determined according to $\beta = H^* \cdot L$, where $H^*$ is the generalised inverse matrix of $H$. To enhance the stability of the neural network, the regularisation factor $c$ and the unit matrix $I$ are introduced, so that the least squares solution for the output weights is:

$$\beta = H^T(HH^T + \frac{I}{c})^{-1}L. \tag{20}$$

Introducing the kernel function into the ELM, the kernel matrix is:

$$\Omega_{ELM} = HH^T = h(x_i)h(x_j) = K(x_i, x_j), \tag{21}$$

where $x_i$, $x_j$ is the test input vector, then Equation (19) can be expressed as:

$$F(x) = [K(x, x_1); \ldots; K(x, x_n)](\frac{I}{c} + \Omega_{ELM})^{-1}L, \tag{22}$$

where $(x_1, x_2, \ldots, x_n)$ is the given training sample, $n$ is the number of samples. $K()$ is the kernel function.

## 5. The Proposed XGB-AVSSA-KELM Method

We begin by utilizing the XGBoost to collect individual significance values to improve the model's efficiency and accuracy. Then, based on the results, features are selected with redundant attributes deleted to reduce the number of network element nodes. Due to the network structure, the configuration of the regularization coefficients $c$ and kernel function parameters $S$ will have an impact on the classification performance of KELM. AVSSA has a unique algorithmic structure compared to previous metaheuristics, which provides a new method for balancing exploration and exploitation in the optimization process, and it can retain a high population diversity while boosting convergence efficiency. As a result, the AVSSA technique can be used to discover the best $c$ and $S$ to improve the network's performance.

The process of the model is presented in Figure 1, and is described as follows:

Step 1: For training and prediction, input simulation data from the TE process into XGBoost;

Step 2: The importance values of XGBoost's features are ranked;

Step 3: Select features and retrieve the dataset for the input network based on the ranking results;

Step 4: Set up the Kernel-Based Extreme Learning Machine and input the random regularization factor $c$ as well as the kernel function parameter $S$;

Step 5: The optimal network hyper-parameters are obtained using the AVSSA algorithm;

Step 6: The optimal KELM diagnostic model is obtained by substituting the optimized regularization coefficients $c$ kernel function parameters $S$ into the KELM for training;

Step 7: The test samples are fed into the trained network to obtain the predicted output.

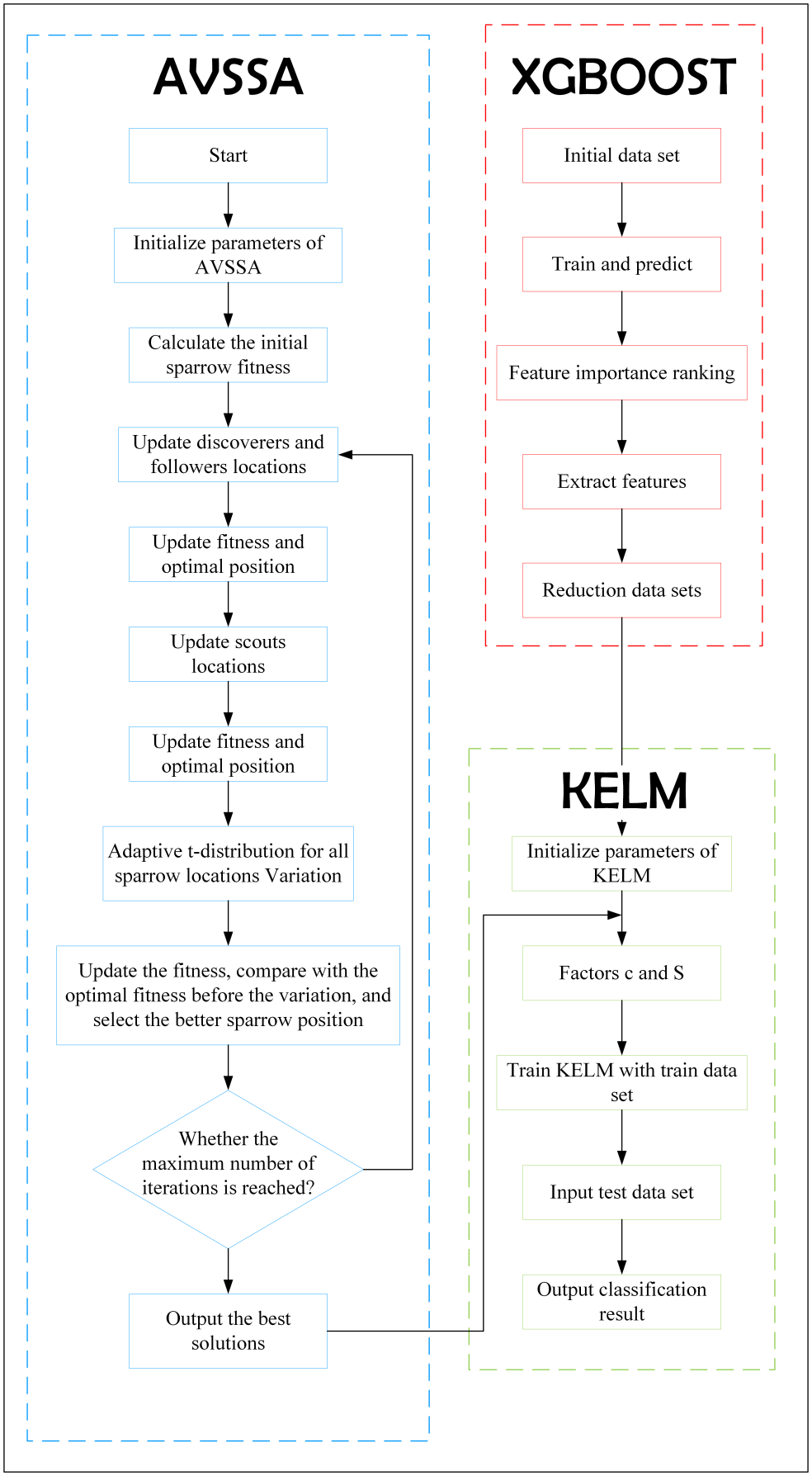


**Figure 1.** XGB-AVSSA-KELM algorithm flow chart.

## 6. Experiment

### *6.1. Model Establishment*

To detect and diagnose faults in the TE process database, the XGB-AVSSA-KELM model is used. Table 1 shows the errors found in the TE process database. Step changes in process variables, increased variability in process variables, and actuator faults are all linked to these faults (e.g., viscous valves). As a result, using a model, the data from the TE process simulation are utilized to diagnose and discover defects in the samples. In order to reduce redundant data and improve time series, the following significant variables of the TE process were chosen as assessment indicators for the diagnostic network in Table 4. The AVSSA algorithm was then integrated with KELM to produce the diagnostic model, which was trained using the optimized input data. We fed a test dataset into the trained diagnostic model in order to obtain classification results and confirm the model's accuracy.

**Table 1.** The faults description of TE process.

| Variable Number | Process Variable | Type |
|---|---|---|
| IDV(1) | A/C feed ratio, B composition constant (stream 4) | Step |
| IDV(2) | B comoosition, A/C ratio constant (stream 4) | Step |
| IDV(3) | D feed temperature (stream 2) | Step |
| IDV(4) | Reactor cooling water inlet temperature | Step |
| IDV(5) | Condenser c4mting water inIet temperature | Step |
| IDV(6) | A feed loss (stream 1) | Step |
| IDV(7) | C header pressure loss-reduced availability (stream 4) | Step |
| IDV(8) | A,B,C feed composition (stream 4) | Random variation |
| IDV(9) | D feed temperature (stream 2) | Random variation |
| IDV(10) | C feed temperature (stream 4) | Random variation |
| IDV(11) | Reactor cooling water inlet temperature | Random variation |
| IDV(12) | Condenser cooling water inlet temperature | Random variation |
| IDV(13) | Reaction kinetics | Slow drift |
| IDV(14) | Reactor cooling water valve | Sticking |
| IDV(15) | Condenser cooling water valve | Sticking |
| IDV(16)–(20) | Unknown | Unknown |
| IDV(21) | Valve (stream 4) | constant position |

### *6.2. Tennessee Eastman Process*

The Tennessee Eastman model is a simulation model built by Tennessee Eastman Chemical Company based on the actual chemical production process of the company [54]. Because it covers many typical phenomena in the actual complex chemical production process and contains common production failures in the chemical process, it is often used for troubleshooting studies in the industrial process field.

Figure 2 depicts the Tennessee Eastman process's approximate schematic, which includes five primary units: reactor, condenser, compressor, stripper, and separator. The TE process uses four reactants, A, C, D, and E, as well as two products, G and H. Inertia component B and by-product F are also present. The specific meaning of manipulated variables and process measurements in the legend of the picture can be found in Tables 2 and 3.

In the TE process, there are 12 manipulated variables and 41 process measurements. The dataset used for this experiment contains all 52 variables except for agitator speed, as this was not manipulated during the actual chemical process. Table 2 records the remaining 11 manipulated variables except for the agitator speed. Table 3 records the 41 process measurements. There are 22 training sets in all, including normal settings, for each fault. During a 24-h fault simulation, the fault training datasets were obtained. A 48-h running simulation was used to create the test datasets, with the problem being introduced at the 8-h mark. The sample time was set at three minutes. As a result, there are 480 failure training sets, and, consequently, fault training sets have a total of 480, while normal training sets have a total of 500. There are 960 samples in each fault test set, and the 161st sample is when they all start to fail.

**Table 2.** Manipulated variables.

| No. | Manipulates Variables | No. | Manipulates Variables |
|---|---|---|---|
| 1 | D feed flow valve | 7 | Separator pot liquid flow valve |
| 2 | E feed flow valve | 8 | Stripper liquid product flow valve |
| 3 | A feed flow valve | 9 | Stripper steam valve |
| 4 | Total feed flow valve | 10 | Reactor cooling water flow |
| 5 | Compressor recycle valve | 11 | Condenser cooling water flow |
| 6 | Purge valve | | |

**Table 3.** Process measurements.

| No. | Process Measurements | No. | Process Measurements |
|---|---|---|---|
| 1 | A feed | 22 | Separator cooling water outlet temperature |
| 2 | D feed | 23 | A in reactor is feed |
| 3 | E feed | 24 | B in reactor is feed |
| 4 | Total feed | 25 | C in reactor is feed |
| 5 | Recycle flow | 26 | D in reactor is feed |
| 6 | Reactor feed rate | 27 | E in reactor feed |
| 7 | Reactor pressure | 28 | F in reactor feed |
| 8 | Reactor level | 29 | A in reactor feed |
| 9 | Reactor temperature | 30 | B in reactor feed |
| 10 | Purge rate | 31 | C in reactor feed |
| 11 | Product separator temperature | 32 | D in reactor feed |
| 12 | Product separator level | 33 | E in reactor feed |
| 13 | Product separator pressure | 34 | F in reactor feed |
| 14 | Product separator underflow | 35 | G in reactor feed |
| 15 | Stripper level | 36 | H in reactor feed |
| 16 | Stripper pressure | 37 | D in product flow |
| 17 | Stripper underflow | 38 | E in product flow |
| 18 | Stripper temperature | 39 | F in product flow |
| 19 | Stripper steam flow | 40 | G in product flow |
| 20 | Compressor work | 41 | H in product flow |
| 21 | Reactor cooling water outlet temperature | | |

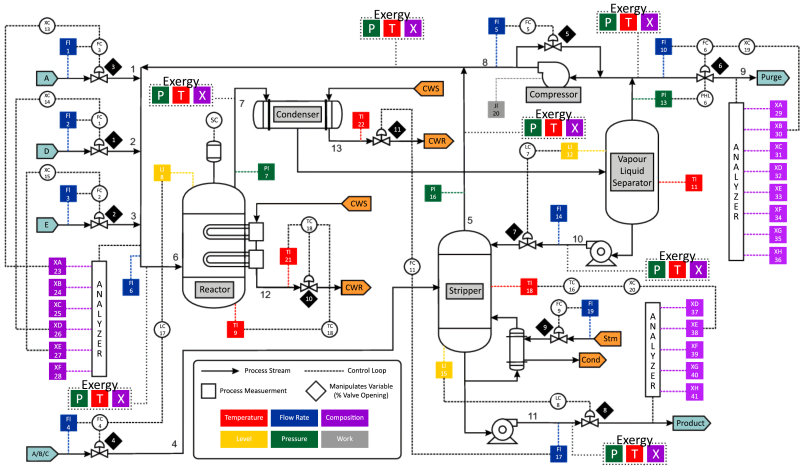


**Figure 2.** The TE process diagram.

### *6.3. Feature Selection*

Feature selection is a key topic in defect detection and classification, and it has an impact on the model's performance. The value of features in a model can be automatically

evaluated using a gradient-enhanced decision system developed by Xgboost in order to find non-linear feature interactions and consistently extract significant features. For feature selection, the relevance score of an individual decision tree is generated using the tree model's attributes as the basis for evaluating the value of each feature.

The feature selection result in fault 15 and fault 17, which have a normalized scale of 1, is shown in Figures 3 and 4.

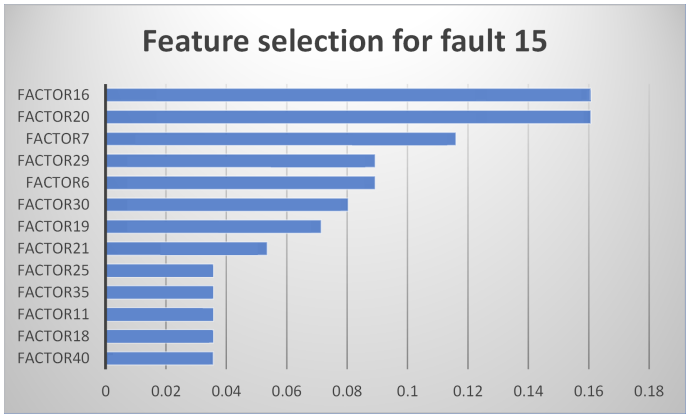


**Figure 3.** Feature selection for fault 15.

For each fault, XGBoost feature selection is performed, and we can obtain the feature selection sequence corresponding to each fault.

The feature selection sequence corresponding to each fault is shown in Table 4. The numbers 1 to 41 in the figure correspond to process measurements 1 to 41 in this Table 3; the numbers 42 to 52 in the figure correspond to manipulated variables 1 to 11 in Table 2.

**Table 4.** The result of feature selection.

| Fault | The Result of Feature Selection |
|---|---|
| 1 | 1, 16, 18, 39, 7, 13, 4, 6, 3, 20, 44, 9 |
| 2 | 10, 20, 47, 5, 7, 41, 15, 16, 9, 3, 49, 1, 25, 24, 30, 46 |
| 3 | 21, 16, 7, 6, 1, 40, 17, 18, 9, 47, 33, 14, 38, 4, 35, 52, 50, 29, 51, 34, 20, 13, 32, 46, 45, 12 |
| 4 | 18, 11, 7, 30, 51, 40, 38, 34, 20, 1 |
| 5 | 52, 16, 17, 20, 7, 11, 39, 18, 43, 14, 19, 28, 37, 13, 50, 41, 33, 1, 25, 35, 38, 8, 49, 3, 48, 9, 45, 12, 40, 4, 32, 36, 47, 24, 2 |
| 6 | 21, 11, 1, 20, 2, 13, 9, 14, 7, 28, 10, 19, 29, 44, 22, 46, 23 |
| 7 | 19, 38, 45, 46, 50 |
| 8 | 29, 1, 41, 16, 20, 44, 40, 23, 10, 47, 39, 7, 38, 14, 35, 19 |
| 9 | 7, 20, 16, 13, 3, 29, 11, 21, 1, 36, 37, 2, 33, 40, 10, 30, 46, 18, 26, 41, 25, 50, 34, 32, 19, 9, 6, 38, 39, 24, 35, 15, 23, 45, 12 |
| 10 | 16, 19, 7, 18, 50, 6, 38, 3, 25 |
| 11 | 51, 16, 20, 7, 6, 9, 22 |
| 12 | 18, 3, 7, 19, 13, 16, 11, 20, 2 |
| 13 | 19, 21, 50, 39, 7, 32, 51, 16, 1, 38 |
| 14 | 21, 7, 51, 16, 6, 9, 34 |
| 15 | 16, 20, 7, 29, 6, 30, 19, 21, 25, 35, 11, 18, 40 |
| 16 | 16, 50, 20, 7, 19, 6, 32, 18 |
| 17 | 16, 7, 21, 6 |
| 18 | 16, 7, 18, 19, 41, 11, 3, 50, 22, 1, 25 |
| 19 | 16, 7, 5, 20, 30, 46, 18, 6 |
| 20 | 16, 7, 13, 21, 20, 46 |
| 21 | 18, 16, 30, 6, 11, 7, 19 |

In the figure, we can see that, in general, the number of features in the feature selection sequence is generally in the range of five to fifteen. Comparing the fifty-two feature values

in the original feature sequence, the redundant and irrelevant features are effectively deleted and strongly correlated features are extracted, which reduces the computational resources required for fault diagnosis.

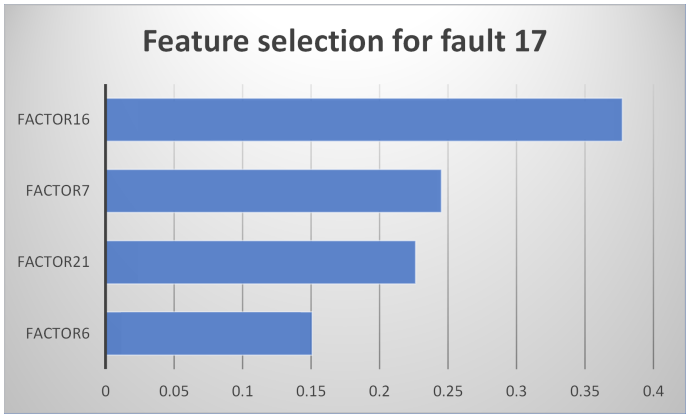


**Figure 4.** Feature selection for fault 17.

For faults 3, 5 and 9, we can see in the figure that the number of features in their feature selection sequence is much greater than for the other faults. It is well known that the difficulties in the TE process fault diagnosis include faults 3, 5 and 9, which correspond to the inability to effectively delete redundant and irrelevant features and extract strongly relevant features, resulting in too many features being selected for faults 3, 5 and 9.

## 7. Experimental Results

### *7.1. The Performance of Purposed Method*

#### 7.1.1. Effect of Feature Selection

To emphasize the significance of feature selection, using the t-SNE dimensionality reduction approach, the fault features were reduced to two dimensions and were visualized, with the results presented in Figures 5 and 6.

Figure 5 shows fault 1, 2 and 3 selected by XGBOOST features. Faults 1 and 2 are barely mixed with other faults and have high identifiability, corresponding to faults 1 and 2, while faults 3 and 4 are completely mixed together, demonstrating that fault 3 without feature selection has a high identification rate with other faults with a high degree of similarity, corresponding to the ultra-low fault diagnostic rate of fault 3.

Figure 6 shows faults 1, 2, 3, and 4 selected by XGBOOST features, in which the faults 1, 2, 3 and 4 are not mixed together, indicating that fault identifiability has improved after feature extraction, and that faults 1, 2, 3 and 4 are more closely aggregated, indicating that data similarity within the faults has been improved by feature extraction, and that the commonality of each fault data point has been well identified, which corresponds to the increased fault diagnosis rate after feature extraction.

#### 7.1.2. Fault Diagnostic Rate (FDR)

The values of true positive $TP$ and true negative $TN$ represent the number of observations representing the correct classification, while the false positive $FP$ and false negative $FN$ represent the number of misclassifications. The specific meanings they represent are shown in Table 5.

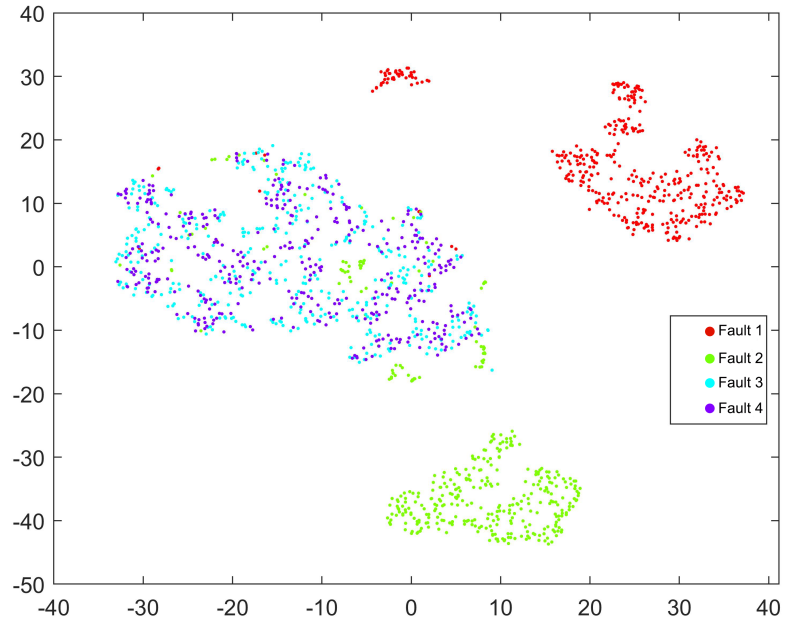


**Figure 5.** TSNE visual output of faults 1, 2, 3, 4.

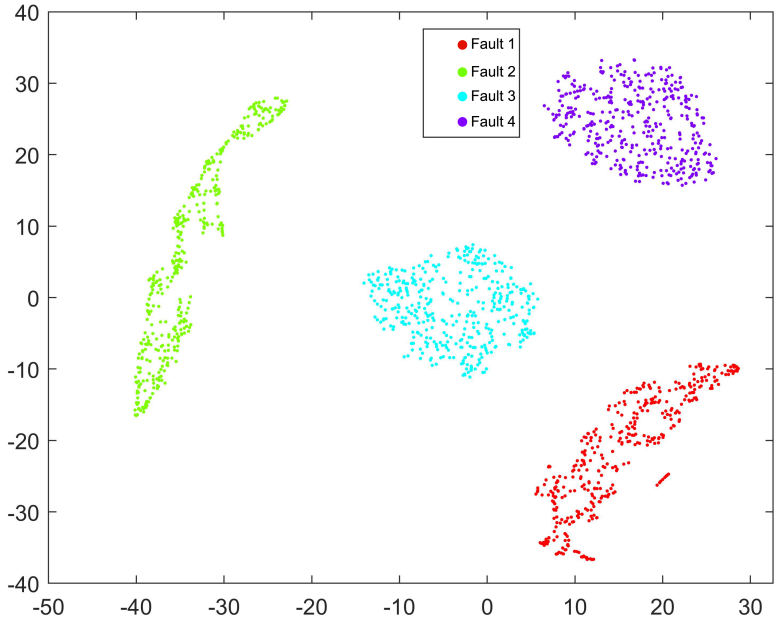


**Figure 6.** TSNE visual output of feature-selected faults 1, 2, 3, 4.

**Table 5.** Confusion Matrix.

| Confusion Matrix | | Prediction | |
|---|---|---|---|
| | | **Positive (p)** | **Negative (N)** |
| Real | True(T) | TP | FN |
| | Flase(F) | FP | TN |

$$FDR = \frac{TP + TN}{TP + TN + FN + FP} \tag{23}$$

As shown in Figure 7, the average FDR for the training dataset is 0.9992. This diagnostic model has exceptionally high diagnostic rates (above 0.99) for all faults in the training set,

demonstrating the algorithm's ability to spot the training set's characteristics. The average FDR in the test dataset was 0.9100, demonstrating that the model works not just with the training dataset, but also with the test dataset. Faults 3, 9, 15, and 16 have long been recognized as long-standing issues in process diagnosis. Except for faults 3, 9, 15, and 16, the diagnostic rate for all faults in this paper is above 85%, with an average diagnosis rate of 94.61%. For fault 16, the algorithm used in this research improves the diagnosis rate to 0.9177, which is a significant improvement to fault 16. This study additionally improves the diagnostic rate for faults 3, 9, and 15 to 0.7677, 0.7296, and 0.6167, respectively, while retaining the total diagnostic rate. One of the superiorities of this method is the increase in diagnosis rate.

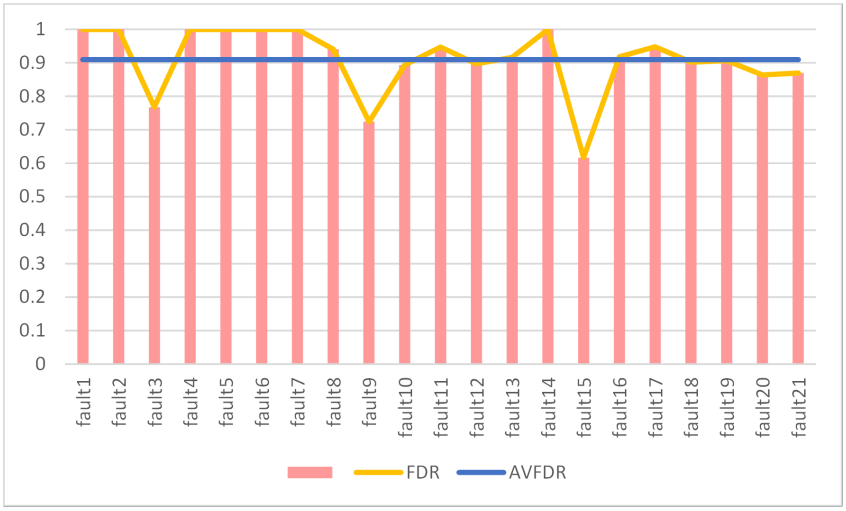


**Figure 7.** Fault diagnosis rate of XGB-AVSSA-KELM.

### *7.2. Performance Comparison*

#### 7.2.1. Convergence Speed Comparison

Figure 8 shows a comparison of the iteration curves for AVSSA (black line), Runge–Kutta [55] (red line), Hunger Games [56] (green line), African Vulture [57] (blue line) and Chameleon [58] (yellow line) for 21 faults. The data for all iteration curves are normalized using MAX-MIN and are presented in the figure as a line graph. Since the initial iteration curve was a straight line, it could not be normalized and is therefore assigned to a straight line with y = 0 in this work.

In Figure 8, it is clear that Chameleon (yellow line) and African Vulture (blue line) both use more iterations to find the optimal solution; Runge–Kutta (red line), AVSSA (black line), and Hunger games (green line) are all algorithms that find the optimal solution in fewer iterations, but they are all mixed together and difficult to see. Sparrow (Figure 9) and Hunger Games (Figure 10) are the fastest, Dragon Kuta (Figure 11) is in the center, while Chameleon (Figure 12) and African Vulture (Figure 13) are the slowest in terms of convergence speed. The AVSSA algorithm has a smoother iteration curve and fewer lines than the others (see Figures 9–11), which reflects the fact that the AVSSA algorithm has more cases of reaching the optimal value in the first iteration and demonstrates the AVSSA algorithm's unique advantage in terms of convergence speed.

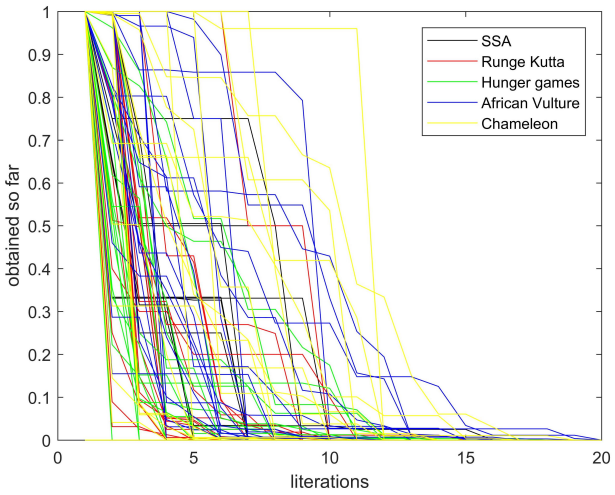


**Figure 8.** The iteration curves for the five methods.

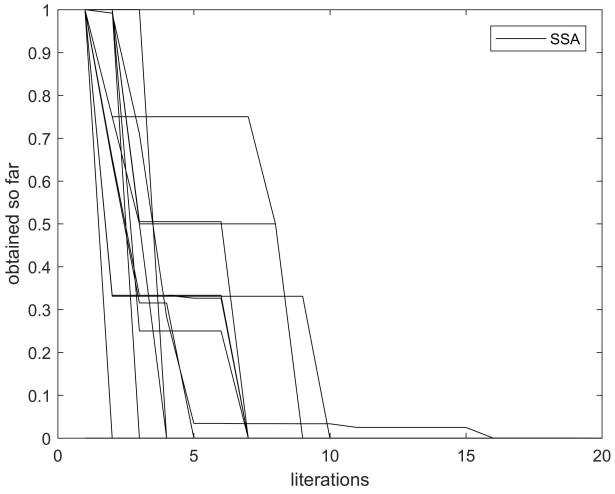


**Figure 9.** The iteration curves for AVSSA.

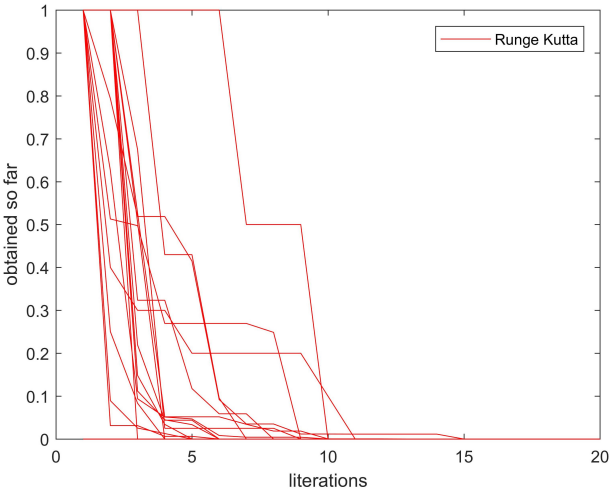


**Figure 10.** The iteration curves for the five methods.

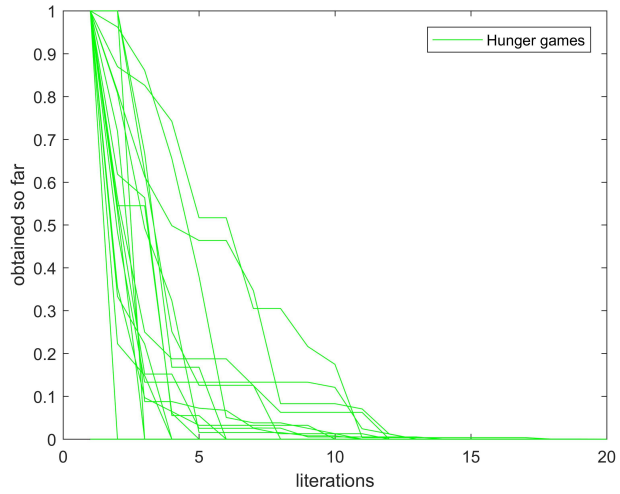


**Figure 11.** The iteration curves for AVSSA.

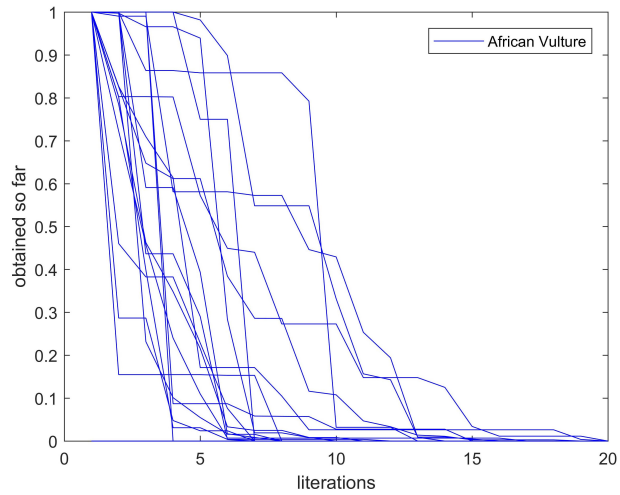


**Figure 12.** The iteration curves for the five methods.

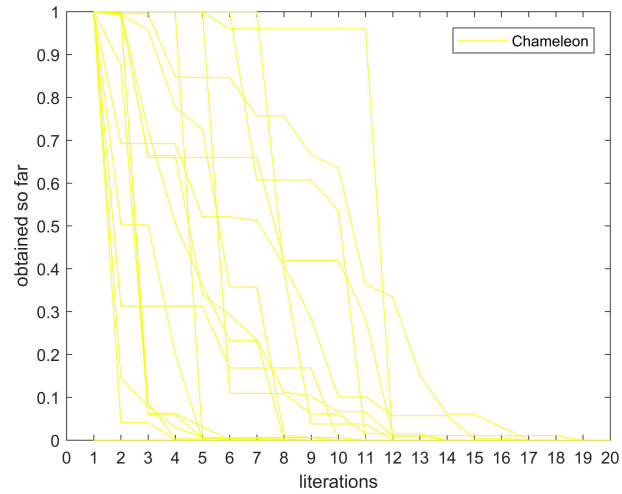


**Figure 13.** The iteration curves for AVSSA.

#### 7.2.2. Feature Selection Algorithm Comparison

NSVM-AVSSA-KELM, RF-AVSSA-KELM and XGB-AVSSA-KELM were the three sets of experiments in which NSVM did not provide feature extraction for faults 3, 9 and 15, marked with *.

The results of the comparison of the three different feature selection algorithms are shown in Table 6, where the XGB-AVSSA-KELM algorithm is the best of the three, proving the superiority of XGB in the feature selection algorithm.

**Table 6.** The feature selection algorithm comparison table.

| Fault | NSVM-AVSSA-KELM | RF-AVSSA-KELM | XGB-AVSSA-KELM |
|---|---|---|---|
| 1 | 0.9979 | 0.9969 | 1.0000 |
| 2 | 0.9729 | 0.9750 | 1.0000 |
| 3 | * | 0.5479 | 0.7677 |
| 4 | 1.0000 | 1.0000 | 1.0000 |
| 5 | 0.9687 | 0.9844 | 1.0000 |
| 6 | 1.0000 | 0.9844 | 1.0000 |
| 7 | 1.0000 | 1.0000 | 1.0000 |
| 8 | 0.9072 | 0.8198 | 0.9406 |
| 9 | * | 0.5198 | 0.7240 |
| 10 | 0.6625 | 0.7333 | 0.8938 |
| 11 | 0.8197 | 0.8354 | 0.9469 |
| 12 | 0.8375 | 0.8969 | 0.8969 |
| 13 | 0.7395 | 0.7896 | 0.9167 |
| 14 | 0.9114 | 0.9990 | 1.0000 |
| 15 | * | 0.5365 | 0.6167 |
| 16 | 0.7093 | 0.7594 | 0.9177 |
| 17 | 0.7437 | 0.8469 | 0.9479 |
| 18 | 0.9031 | 0.8823 | 0.9021 |
| 19 | 0.8000 | 0.8688 | 0.9063 |
| 20 | 0.7958 | 0.7969 | 0.8635 |
| 21 | 0.9708 | 0.6760 | 0.8698 |
| Average | 0.8745 * | 0.8309 | 0.9100 |

#### 7.2.3. Ablation Experiments

In order to verify the relative importance of both XGBOOST and AVSSA relative to the XGB-AVSSA-KELM algorithm, and the superiority of XGB feature selection, six sets of experiments are set up in this paper. They are KELM, AVSSA-KELM, NSVM-AVSSA-KELM, RF-AVSSA-KELM XGB-KELM, and XGB-AVSSA-KELM. NSVM and RP are the two feature selection methods used by Xie [59] and Onel [35] respectively, where the results of feature selection are cited and combined with AVSSA-KELM for fault diagnosis in this paper.

Table 7 shows the experimental outcomes. Overall, the KELM algorithm has the lowest FDR of the six sets of tests, with the rest of the algorithms being higher, demonstrating that all of the methods performed have improved the FDR to some extent.

First, we explore the effect of the AVSSA algorithm on the combined model. According to Table 7, when KELM is compared with AVSSA-KELM, the FDR of AVSSA-KELM is improved by 9.63% compared with KELM; when XGB- KELM is compared with XGB-AVSSA-KELM, the FDR of XGB-AVSSA-KELM is improved by 15.61% compared with XGB-KELM. This indicates that the network hyperparameters of AVSSA-optimized KELM can effectively improve the correct rate of fault diagnosis.

Then, we explore the role of XGBOOST (XGB) feature selection on the combined model. According to Table 7, when XGB-KELM and KELM are compared, the FDR of XGB-KELM is improved by 4.21% compared with KELM; when XGB-AVSSA-KELM and AVSSA-KELM are compared, the FDR of XGB-AVSSA-KELM is improved by 10.19% compared with AVSSA-KELM. This indicates that XGBOOST for feature selection can effectively improve the correct rate of fault diagnosis.

**Table 7.** Results of ablation experiments with preserved KELM.

| Fault | KELM | AVSSA-KELM | XGB-KELM | XGB-AVSSA-KELM |
|---|---|---|---|---|
| 1 | 0.9938 | 0.9990 | 0.9927 | 1.0000 |
| 2 | 0.9781 | 0.9813 | 0.9813 | 1.0000 |
| 3 | 0.5938 | 0.6354 | 0.4844 | 0.7677 |
| 4 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 5 | 0.4813 | 0.7146 | 0.9708 | 1.0000 |
| 6 | 0.9958 | 0.9958 | 0.9958 | 1.0000 |
| 7 | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| 8 | 0.6417 | 0.8635 | 0.6115 | 0.9406 |
| 9 | 0.7229 | 0.7240 | 0.6198 | 0.7240 |
| 10 | 0.6042 | 0.6552 | 0.5667 | 0.8938 |
| 11 | 0.5521 | 0.7865 | 0.6125 | 0.9469 |
| 12 | 0.4781 | 0.7375 | 0.4438 | 0.8969 |
| 13 | 0.6000 | 0.7031 | 0.6156 | 0.9167 |
| 14 | 0.8135 | 0.9552 | 0.9927 | 1.0000 |
| 15 | 0.4396 | 0.4865 | 0.6427 | 0.6167 |
| 16 | 0.6177 | 0.7291 | 0.6604 | 0.9177 |
| 17 | 0.6833 | 0.7625 | 0.6979 | 0.9479 |
| 18 | 0.9052 | 0.8990 | 0.8771 | 0.9021 |
| 19 | 0.6031 | 0.8333 | 0.7552 | 0.9063 |
| 20 | 0.7302 | 0.7385 | 0.7406 | 0.8635 |
| 21 | 0.5135 | 0.7698 | 0.5698 | 0.8698 |
| Average | 0.7118 | 0.8081 | 0.7539 | 0.9100 |

#### 7.2.4. Fault Diagnostic Rate Comparison

Table 8 and Figure 14 compare the performance of the proposed technique to that of other fault diagnostic methods in the TE process. Other methods are M-KFDA-SVM [60], EDBN-2 [61], GA-DCNN and RF-GA-CNN [62]. Because EDBN-2 does not include diagnostic rate data for faults 3, 9, and 15, and because faults 3, 9, and 15 are all difficult to diagnose in chemical processes, the FDR for all three is set to 60% in this work.

When compared to other fault diagnosis approaches, the model in this research performed the best, as shown in Table 8. AVSSA-KELM had the best overall classification performance based on XGBOOST feature selection, with a score of 0.91. The overall fault diagnosis rate in this paper was greater than 0.6.

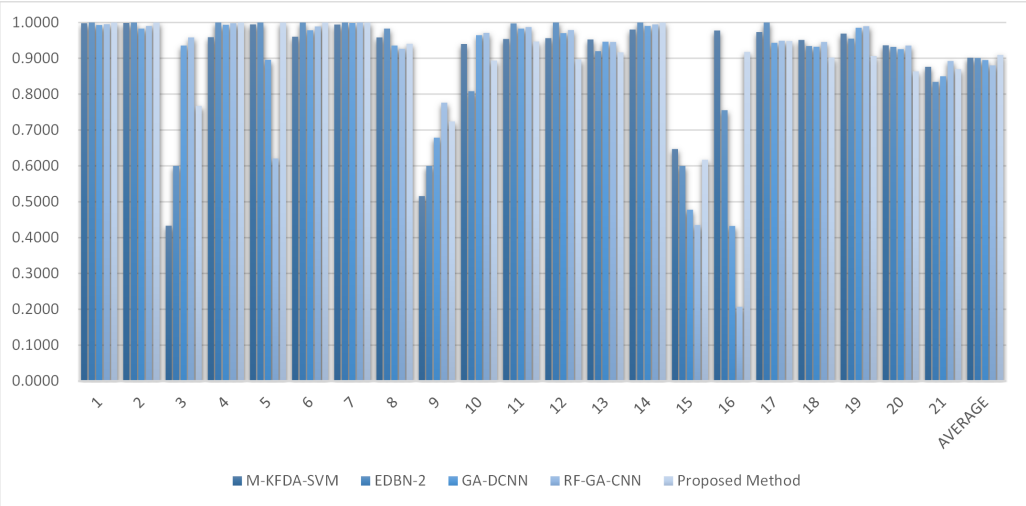


**Figure 14.** Comparison of FDR with other algorithms.

**Table 8.** Comparison of FDR with other algorithms.

| Fault | M-KFDA-SVM | EDBN-2 | GA-DCNN | RF-GA-CNN | Proposed Method |
|---|---|---|---|---|---|
| 1 | 0.9980 | 1.0000 | 0.9931 | 0.9956 | 1.0000 |
| 2 | 0.9987 | 1.0000 | 0.9825 | 0.9906 | 1.0000 |
| 3 | 0.4333 | 0.6000 | 0.9356 | 0.9581 | 0.7677 |
| 4 | 0.9589 | 1.0000 | 0.9938 | 0.9981 | 1.0000 |
| 5 | 0.9950 | 1.0000 | 0.8962 | 0.6206 | 1.0000 |
| 6 | 0.9602 | 1.0000 | 0.9786 | 0.9893 | 1.0000 |
| 7 | 0.9944 | 1.0000 | 0.9988 | 1.0000 | 1.0000 |
| 8 | 0.9581 | 0.9829 | 0.9356 | 0.9269 | 0.9406 |
| 9 | 0.5156 | 0.6000 | 0.6788 | 0.7762 | 0.7240 |
| 10 | 0.9399 | 0.8081 | 0.9644 | 0.9706 | 0.8938 |
| 11 | 0.9538 | 0.9974 | 0.9831 | 0.9875 | 0.9469 |
| 12 | 0.9563 | 1.0000 | 0.9700 | 0.9788 | 0.8969 |
| 13 | 0.9524 | 0.9198 | 0.9462 | 0.9456 | 0.9167 |
| 14 | 0.9805 | 1.0000 | 0.9906 | 0.9950 | 1.0000 |
| 15 | 0.6474 | 0.6000 | 0.4775 | 0.4350 | 0.6167 |
| 16 | 0.9780 | 0.7556 | 0.4325 | 0.2069 | 0.9177 |
| 17 | 0.9732 | 1.0000 | 0.9431 | 0.9488 | 0.9479 |
| 18 | 0.9515 | 0.9343 | 0.9325 | 0.9456 | 0.9021 |
| 19 | 0.9687 | 0.9553 | 0.9856 | 0.9900 | 0.9063 |
| 20 | 0.9363 | 0.9317 | 0.9256 | 0.9356 | 0.8635 |
| 21 | 0.8764 | 0.8344 | 0.8500 | 0.8930 | 0.8698 |
| Average | 0.9013 | 0.9009 | 0.8950 | 0.8804 | 0.9100 |

## 8. Conclusions

In this paper, a deep learning-based fault diagnosis method for chemical processes is proposed. The method combines feature selection and dataset optimization strategies with a KELM classifier. First, the dataset is optimized by removing redundant features using the XGBOOST model. Second, a new optimization algorithm, AVSSA, is proposed to automatically adjust the network hyper-parameters of KELM to improve the performance of the fault classifier. Finally, the optimized feature sequences are input into the proposed classifier to obtain the final diagnosis results. The proposed diagnosis method is applied to the TE process, and the experimental results verify the applicability and effectiveness of the model. Compared with several other deep learning models, this method has higher accuracy and achieves the same accuracy with fewer iterations.

The method has achieved some results, but there are still some limitations that need to be improved in future work. Our optimization of a large number of network hyperparameters of KELM has led to a significant improvement in the diagnostic performance of KELM, but the upper limit of the accuracy of the KELM classifier is still limiting the diagnostic accuracy. Future work is needed to carry out a deeper structural optimization of the classifier. It is also necessary to detect faults in real plants, compare the "time" required to solve them with traditional fault diagnosis processes and explore comprehensive optimization for practical applications. In addition, for the extracted features, compared with the pure data-driven diagnosis method, we can combine the characteristics of the chemical process itself and consider the characteristics of the TE process to explore the chemical connection between the feature variables, which will be a new cross-optimization direction.

**Author Contributions:** Conceptualization, M.H. In addition, X.H.; methodology, M.H.; software, M.H.; validation, M.H. In addition, X.H.; formal analysis, M.H. In addition, X.H.; investigation, X.H. In addition, B.T.; resources, M.H., X.H., B.T. In addition, Z.D.; data curation, M.H., B.T. In addition, Z.D.; writing—original draft preparation,M.H. In addition, X.H.; writing—review and editing, M.H. In addition, X.H.; visualization, M.H. In addition, X.H.; supervision, M.H., B.T. In addition, B.T.; project administration, M.H., B.T. In addition, Z.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References


1. Pagliaro, M. An industry in transition: The chemical industry and the megatrends driving its forthcoming transformation. *Angew. Chem. Int. Ed.* **2019**, *58*, 11154–11159. [CrossRef]
2. Stoessel, F. *Thermal Safety of Chemical Processes: Risk Assessment and Process Design*; John Wiley & Sons: Hoboken, NJ, USA, 2021.
3. Chen, C.; Reniers, G.; Khakzad, N. A dynamic multi-agent approach for modeling the evolution of multi-hazard accident scenarios in chemical plants. *Reliab. Eng. Syst. Saf.* **2021**, *207*, 107349. [CrossRef]
4. Yang, Y.; Chen, G.; Reniers, G.; Goerlandt, F. A bibliometric analysis of process safety research in China: Understanding safety research progress as a basis for making China's chemical industry more sustainable. *J. Clean. Prod.* **2020**, *263*, 121433. [CrossRef]
5. Chen, C.; Khakzad, N.; Reniers, G. Dynamic vulnerability assessment of process plants with respect to vapor cloud explosions. *Reliab. Eng. Syst. Saf.* **2020**, *200*, 106934. [CrossRef]
6. An, J.; Wang, Z.; Jiang, T.; Chen, P.; Liang, X.; Shao, J.; Nie, J.; Xu, M.; Wang, Z.L. Reliable mechatronic indicator for self-powered liquid sensing toward smart manufacture and safe transportation. *Mater. Today* **2020**, *41*, 10–20. [CrossRef]
7. Amin, M.T.; Imtiaz, S.; Khan, F. Process system fault detection and diagnosis using a hybrid technique. *Chem. Eng. Sci.* **2018**, *189*, 191–211. [CrossRef]
8. Venkatasubramanian, V.; Rengaswamy, R.; Yin, K.; Kavuri, S.N. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. *Comput. Chem. Eng.* **2003**, *27*, 293–311. [CrossRef]
9. Venkatasubramanian, V.; Rengaswamy, R.; Kavuri, S.N. A review of process fault detection and diagnosis: Part II: Qualitative models and search strategies. *Comput. Chem. Eng.* **2003**, *27*, 313–326. [CrossRef]
10. Venkatasubramanian, V.; Rengaswamy, R.; Kavuri, S.N.; Yin, K. A review of process fault detection and diagnosis: Part III: Process history based methods. *Comput. Chem. Eng.* **2003**, *27*, 327–346. [CrossRef]
11. Xinyi, C.; Xuefeng, Y. Fault diagnosis in chemical process based on self-organizing map integrated with fisher discriminant analysis. *Chin. J. Chem. Eng.* **2013**, *21*, 382–387.
12. Nor, N.M.; Hassan, C.R.C.; Hussain, M.A. A review of data-driven fault detection and diagnosis methods: Applications in chemical process systems. *Rev. Chem. Eng.* **2020**, *36*, 513–553. [CrossRef]
13. Dunia, R.; Qin, S.J.; Edgar, T.F.; McAvoy, T.J. Identification of faulty sensors using principal component analysis. *AIChE J.* **1996**, *42*, 2797–2812. [CrossRef]
14. Botre, C.; Mansouri, M.; Nounou, M.; Nounou, H.; Karim, M.N. Kernel PLS-based GLRT method for fault detection of chemical processes. *J. Loss Prev. Process. Ind.* **2016**, *43*, 212–224. [CrossRef]
15. Xu, Y.; Deng, X. Fault detection of multimode non-Gaussian dynamic process using dynamic Bayesian independent component analysis. *Neurocomputing* **2016**, *200*, 70–79. [CrossRef]
16. Yu, J. Localized Fisher discriminant analysis based complex chemical process monitoring. *AIChE J.* **2011**, *57*, 1817–1828. [CrossRef]
17. Shi, X.; Lv, Y.; Fei, Z.; Liang, J. A multivariable statistical process monitoring method based on multiscale analysis and principal curves. *Int. J. Innov. Comput. Inf. Control.* **2013**, *9*, 1781–1800.
18. Baudat, G.; Anouar, F. Generalized discriminant analysis using a kernel approach. *Neural Comput.* **2000**, *12*, 2385–2404. [CrossRef]
19. Wang, C.; Li, H.; Zhang, K.; Hu, S.; Sun, B. Intelligent fault diagnosis of planetary gearbox based on adaptive normalized CNN under complex variable working conditions and data imbalance. *Measurement* **2021**, *180*, 109565. [CrossRef]
20. Wang, N.; Li, H.; Wu, F.; Zhang, R.; Gao, F. Fault Diagnosis of Complex Chemical Processes Using Feature Fusion of a Convolutional Network. *Ind. Eng. Chem. Res.* **2021**, *60*, 2232–2248. [CrossRef]
21. Pirdashti, M.; Curteanu, S.; Kamangar, M.H.; Hassim, M.H.; Khatami, M.A. Artificial neural networks: Applications in chemical engineering. *Rev. Chem. Eng.* **2013**, *29*, 205–239. [CrossRef]
22. Mahadevan, S.; Shah, S.L. Fault detection and diagnosis in process data using one-class support vector machines. *J. Process. Control* **2009**, *19*, 1627–1639. [CrossRef]
23. McCulloch, W.S.; Pitts, W. Bulletin of Mathematical Biophysics. *Bull. Math. Biophys.* **1943**, *5*, 115–133. [CrossRef]
24. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386. [CrossRef] [PubMed]
25. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef] [PubMed]
26. Hopfield, J.J.; Tank, D.W. Computing with neural circuits: A model. *Science* **1986**, *233*, 625–633. [CrossRef] [PubMed]
27. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [CrossRef]
28. Huang, G.B.; Zhu, Q.Y.; Siew, C.K. Extreme learning machine: Theory and applications. *Neurocomputing* **2006**, *70*, 489–501. [CrossRef]

29. Huang, G.B.; Zhou, H.; Ding, X.; Zhang, R. Extreme learning machine for regression and multiclass classification. *IEEE Trans. Syst. Man Cybern. Part B* **2011**, *42*, 513–529. [CrossRef]
30. John, G.H.; Kohavi, R.; Pfleger, K. Irrelevant features and the subset selection problem. In *Machine Learning Proceedings 1994*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 121–129.
31. Fisher, R.A. The use of multiple measurements in taxonomic problems. *Ann. Eugen.* **1936**, *7*, 179–188. [CrossRef]
32. Rao, C.R. The utilization of multiple measurements in problems of biological classification. *J. R. Stat. Soc. Ser.* **1948**, *10*, 159–203. [CrossRef]
33. Lau, C.; Ghosh, K.; Hussain, M.A.; Hassan, C.C. Fault diagnosis of Tennessee Eastman process with multi-scale PCA and ANFIS. *Chemom. Intell. Lab. Syst.* **2013**, *120*, 1–14. [CrossRef]
34. Deng, X.; Tian, X.; Chen, S.; Harris, C.J. Deep principal component analysis based on layerwise feature extraction and its application to nonlinear process monitoring. *IEEE Trans. Control Syst. Technol.* **2018**, *27*, 2526–2540. [CrossRef]
35. Onel, M.; Kieslich, C.A.; Pistikopoulos, E.N. A nonlinear support vector machine-based feature selection approach for fault detection and diagnosis: Application to the Tennessee Eastman process. *AIChE J.* **2019**, *65*, 992–1005. [CrossRef] [PubMed]
36. Guzman, Y.A. Theoretical Advances in Robust Optimization, Feature Selection, and Biomarker Discovery. Ph.D. Thesis, Princeton University, Princeton, NJ, USA, 2016.
37. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
38. Speiser, J.L.; Miller, M.E.; Tooze, J.; Ip, E. A comparison of random forest variable selection methods for classification prediction modeling. *Expert Syst. Appl.* **2019**, *134*, 93–101. [CrossRef] [PubMed]
39. Vens, C.; Costa, F. Random forest based feature induction. In Proceedings of the 2011 IEEE 11th International Conference on Data Mining, Vancouver, BC, Canada, 11 December 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 744–753.
40. Rodriguez-Galiano, V.F.; Ghimire, B.; Rogan, J.; Chica-Olmo, M.; Rigol-Sanchez, J.P. An assessment of the effectiveness of a random forest classifier for land-cover classification. *ISPRS J. Photogramm. Remote Sens.* **2012**, *67*, 93–104. [CrossRef]
41. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
42. Chen, T.; Guestrin, C. Xgboost: A scalable tree boosting system. In Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 785–794.
43. Parsa, A.B.; Movahedi, A.; Taghipour, H.; Derrible, S.; Mohammadian, A.K. Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis. *Accid. Anal. Prev.* **2020**, *136*, 105405. [CrossRef]
44. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control Eng.* **2020**, *8*, 22–34. [CrossRef]
45. Wang, W.C.; Xu, L.; Chau, K.W.; Xu, D.M. Yin-Yang firefly algorithm based on dimensionally Cauchy mutation. *Expert Syst. Appl.* **2020**, *150*, 113216. [CrossRef]
46. Yang, Z.; Duan, H.; Fan, Y.; Deng, Y. Automatic carrier landing system multilayer parameter design based on Cauchy mutation pigeon-inspired optimization. *Aerosp. Sci. Technol.* **2018**, *79*, 518–530. [CrossRef]
47. Sapre, S.; Mini, S. Opposition-based moth flame optimization with Cauchy mutation and evolutionary boundary constraint handling for global optimization. *Soft Comput.* **2019**, *23*, 6023–6041. [CrossRef]
48. Song, S.; Wang, P.; Heidari, A.A.; Wang, M.; Zhao, X.; Chen, H.; He, W.; Xu, S. Dimension decided Harris hawks optimization with Gaussian mutation: Balance analysis and diversity patterns. *Knowl.-Based Syst.* **2021**, *215*, 106425. [CrossRef]
49. Zhang, X.; Xu, Y.; Yu, C.; Heidari, A.A.; Li, S.; Chen, H.; Li, C. Gaussian mutational chaotic fruit fly-built optimization and feature selection. *Expert Syst. Appl.* **2020**, *141*, 112976. [CrossRef]
50. Sarangi, A.; Samal, S.; Sarangi, S.K. Analysis of gaussian & cauchy mutations in modified particle swarm optimization algorithm. In Proceedings of the 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 15–16 March 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 463–467.
51. Huang, G.; Huang, G.B.; Song, S.; You, K. Trends in extreme learning machines: A review. *Neural Netw.* **2015**, *61*, 32–48. [CrossRef]
52. Yang, Z.; Ce, L.; Lian, L. Electricity price forecasting by a hybrid model, combining wavelet transform, ARMA and kernel-based extreme learning machine methods. *Appl. Energy* **2017**, *190*, 291–305. [CrossRef]
53. Yang, W.; Tian, Z.; Hao, Y. A novel ensemble model based on artificial intelligence and mixed-frequency techniques for wind speed forecasting. *Energy Convers. Manag.* **2022**, *252*, 115086. [CrossRef]
54. Downs, J.J.; Vogel, E.F. A plant-wide industrial process control problem. *Comput. Chem. Eng.* **1993**, *17*, 245–255. [CrossRef]
55. Ahmadianfar, I.; Heidari, A.A.; Gandomi, A.H.; Chu, X.; Chen, H. RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method. *Expert Syst. Appl.* **2021**, *181*, 115079. [CrossRef]
56. Yang, Y.; Chen, H.; Heidari, A.A.; Gandomi, A.H. Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts. *Expert Syst. Appl.* **2021**, *177*, 114864. [CrossRef]
57. Abdollahzadeh, B.; Gharehchopogh, F.S.; Mirjalili, S. African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems. *Comput. Ind. Eng.* **2021**, *158*, 107408. [CrossRef]
58. Braik, M.S. Chameleon Swarm Algorithm: A bio-inspired optimizer for solving engineering design problems. *Expert Syst. Appl.* **2021**, *174*, 114685. [CrossRef]
59. Xie, Z.; Yang, X.; Li, A.; Ji, Z. Fault diagnosis in industrial chemical processes using optimal probabilistic neural network. *Can. J. Chem. Eng.* **2019**, *97*, 2453–2464. [CrossRef]

60. Nor, N.M.; Hussain, M.A.; Hassan, C.R.C. Fault diagnosis and classification framework using multi-scale classification based on kernel Fisher discriminant analysis for chemical process system. *Appl. Soft Comput.* **2017**, *61*, 959–972. [CrossRef]
61. Wang, Y.; Pan, Z.; Yuan, X.; Yang, C.; Gui, W. A novel deep learning based fault diagnosis approach for chemical process with extended deep belief network. *ISA Trans.* **2020**, *96*, 457–467. [CrossRef] [PubMed]
62. Deng, L.; Zhang, Y.; Dai, Y.; Ji, X.; Zhou, L.; Dang, Y. Integrating feature optimization using a dynamic convolutional neural network for chemical process supervised fault classification. *Process. Saf. Environ. Prot.* **2021**, *155*, 473–485. [CrossRef]