electronics

Article

Feature Selection for Data Classification in the Semiconductor Industry by a Hybrid of Simplified Swarm Optimization

Wei-Chang Yeh and Chia-Li Chu *

Department of Industrial Engineering and Engineering Management, National Tsing Hua University, P.O. Box 24-60, Hsinchu 300, Taiwan; yeh@ieee.org
* Correspondence: chucurry@gmail.com

Abstract: In the semiconductor manufacturing industry, achieving high yields constitutes one of the pivotal factors for sustaining market competitiveness. When confronting the substantial volume of high-dimensional, non-linear, and imbalanced data generated during semiconductor manufacturing processes, it becomes imperative to transcend traditional approaches and incorporate machine learning methodologies. By employing non-linear classification models, one can achieve more real-time anomaly detection, subsequently facilitating a deeper analysis of the fundamental causes behind anomalies. Given the considerable dimensionality of production line data in semiconductor manufacturing, there arises a necessity for dimensionality reduction to mitigate noise and reduce computational costs within the data. Feature selection stands out as one of the primary methodologies for achieving data dimensionality reduction. Utilizing wrapper-based heuristics algorithms, although characterized by high time complexity, often yields favorable performance in specific cases. If further combined into hybrid methodologies, they can concurrently satisfy data quality and computational cost considerations. Accordingly, this study proposes a two-stage feature selection model. Initially, redundant features are eliminated using mutual information to reduce the feature space. Subsequently, a Simplified Swarm Optimization algorithm is employed to design a unique fitness function aimed at selecting the optimal feature subset from candidate features. Finally, support vector machines are utilized as the classification model for validation purposes. For practical cases, it is evident that the feature selection method proposed in this study achieves superior classification accuracy with fewer features in the context of wafer anomaly classification problems. Furthermore, its performance on public datasets further substantiates the effectiveness and generalization capability of the proposed approach.

Keywords: hybrid feature selection; simplified swarm optimization; semiconductor manufacturing

check for updates

Citation: Yeh, W.-C.; Chu, C.-L. Feature Selection for Data Classification in the Semiconductor Industry by a Hybrid of Simplified Swarm Optimization. *Electronics* **2024**, *13*, 2242. https://doi.org/10.3390/electronics13122242

Academic Editor: Frédérique Ducroquet

Received: 28 April 2024
Revised: 27 May 2024
Accepted: 28 May 2024
Published: 7 June 2024

1. Introduction

Semiconductor wafer manufacturing is a capital-intensive and technology-driven industry. Technological advancements, following Moore's Law, lead to exponential growth in circuit density over time [1]. As circuit density increases, semiconductor processes become more precise and intricate, escalating equipment and material costs. Undetected defective products on the production line consume resources without adding value, increasing costs and wasting capacity. Improving product yield is crucial for enhancing capacity and reducing costs in semiconductor manufacturing [2]. It also enhances product quality and reliability while alleviating pressures from R&D, equipment, and material costs.

Maintaining high yield requires robust quality control and real-time detection of production line anomalies. Semiconductor production is divided into front-end and back-end processes. The front-end process involves producing silicon wafers and fabricating integrated circuit components, while the back-end process includes the assembly, packaging, and final testing of individual dies. The front-end process is intricate, comprising hundreds of steps and about 80% of the total semiconductor manufacturing process [3]. After manufacturing, wafers undergo wafer acceptance tests (WAT) and chip probing (CP) to measure their electrical characteristics and functionality, determining their suitability for the back-end process.

This study focuses on the data analysis of parameters in the front-end semiconductor manufacturing process and wafer test measurements. Using WAT and CP data as input and identifying abnormal results from process machines as output, classification models predict abnormalities in specific processes during wafer manufacturing. The model analysis aims to identify key wafer test parameters relevant to specific processes, aiding production line optimization.

Traditionally, abnormality detection in wafer manufacturing employs statistical process control (SPC). Since its inception by Shewhart in the early 1930s, SPC has been widely used for process control and improvement in manufacturing. SPC uses control charts to monitor quality characteristics, setting upper and lower control limits to detect process abnormalities. However, due to the nonlinear nature of many abnormalities in semiconductor manufacturing, traditional linear SPC methods often fail to detect them timely. This leads to undetected issues at workstations, accumulating problems as wafers pass through multiple steps, resulting in unnecessary processing costs and ineffective judgments like overkill or underkill [4]. Consequently, additional manpower is required to redefine control limits and reclassify abnormal messages based on experience.

Overkill and underkill are key indicators in quality control, evaluating misjudgments during wafer testing. Overkill refers to normal wafers being classified as abnormal, while underkill refers to abnormal wafers being classified as normal. Both lead to losses for semiconductor manufacturers: overkill results in intercepted normal products, while underkill increases processing costs and wastes resources. Underkill can also impact the company's reputation by shipping defective products to customers, especially for those emphasizing high yield rates.

As market demands and customization requirements rise, coupled with the complexity of semiconductor processes involving numerous steps and parameters, managing nearly a million daily control charts becomes challenging. Wafers passing testing may later show anomalies reported by customers, requiring engineers to retrospectively search for process issues, incurring significant time costs and impacting reputation.

Given the high complexity and yield requirements in semiconductor manufacturing, traditional SPC methods struggle with non-linearity, high dimensionality, and class imbalance in production data [5]. Machine learning methods, introduced in recent decades, offer better predictions for anomaly detection by learning from massive datasets.

Early efforts to address SPC's limitations employed techniques like principal component analysis (PCA) and partial least squares (PLS) [4], but these methods struggled with real-time monitoring due to the need for manual adjustments in semiconductor processes. Machine learning methods like K-nearest neighbor (KNN) [6], support vector machine (SVM) [5], decision tree (DT) [7], and neural networks [8,9] have shown better performance in anomaly detection.

Feature engineering techniques are crucial for reducing data dimensionality and mitigating noise, which is essential for high-dimensional data. Feature selection, a subset of feature engineering, selects the most relevant features, maintaining interpretability and facilitating further analysis [10]. Traditional feature selection relies on engineers' domain expertise, but modern methods combine filter and heuristic algorithms for improved accuracy and efficiency. Hybrid methods use filter approaches to narrow features and heuristic algorithms to find optimal subsets within limited timeframes, enhancing accuracy and reducing computational time [11].

Class imbalance, which is common in semiconductor manufacturing, complicates classification problems. Methods to address this imbalance include undersampling, oversampling, feature engineering, and designing specialized algorithms [12,13]. Research focuses on predicting wafer yield, detecting machine failures, and analyzing key process parameters for real-time adjustments [14–16]. This study extends previous work by classifying quality into three categories, normal, abnormal, and at-risk, using WAT and CP data.

This study aims to use heuristic algorithm-based feature selection combined with machine learning to propose an anomaly detection model for high-dimensional, nonlinear, and imbalanced data. The objectives are as follows:

1. Propose a hybrid feature selection method combining mutual information (MI) with a simplified swarm optimization (SSO) algorithm using non-binary encoding. This method reduces data dimensionality and accurately selects key factors for anomaly prediction.
2. Develop an anomaly detection approach suitable for multivariate, imbalanced data that is applied to real-world cases for precise, real-time wafer quality management.

With over six hundred WAT parameters, identifying key parameters through feature selection helps engineers understand anomaly causes. This study leverages heuristic algorithms and machine learning to improve anomaly detection in semiconductor manufacturing.

2. Preliminary Issues

This study addresses the issue of abnormal wafer detection, which falls within the domain of imbalanced classification problems. This section elucidates the definition and implications of this issue through discussions on feature selection, imbalanced data classification models, and relevant simplified swarm optimization (SSO) algorithms, forming the foundational basis for the methodology employed in this research.

2.1. Feature Selection

Given the high complexity of semiconductor manufacturing processes, it is imperative to perform dimensionality reduction on the data. This allows the identification of the parameters that have the most significant impacts on analytical outcomes from among thousands of manufacturing and testing parameters. Feature selection is a common method for data dimensionality reduction that is widely applied in classification, data mining, and object detection [17]. It enhances the precision of machine learning models, prevents overfitting, and reduces computational costs [18]. Feature selection entails identifying a subset of features from the original set to maximize relevance and minimize redundancy. It can be categorized into three types: filter, wrapper, and embedded methods.

2.1.1. Filter Methods

Filters are among the earliest feature selection methods [19]. They evaluate features prior to model learning, focusing on the data's characteristics using statistical methods such as information gain, distance, consistency, similarity, or statistical metrics [20,21]. These methods are independent of classification models. For example, the Relief method, based on instance learning, uses the Euclidean distance to calculate feature scores [22], although it has limitations with binary classification and missing data. Improved versions like Relief-A, Relief-B, and Relief-F address these issues [23,24]. Filter methods based on information theory and correlation coefficients, such as information gain (IG), mutual information (MI), and joint mutual information (JMI), have been extensively researched [11,25–29].

2.1.2. Wrappers

Wrappers combine learning model feedback during feature selection. They estimate the impacts of adding or removing features based on the error rate or accuracy from the training model, searching the feature space for the best predictive performance [19]. Due to the NP-hard nature of feature selection [30], heuristic algorithms like genetic algorithms and particle swarm optimization are often used [31,32]. Although wrappers achieve higher accuracy, they have high time complexity and may overfit [33].

2.1.3. Embedded Methods

Embedded methods integrate feature selection with learning, optimizing both machine learning and feature selection parameters simultaneously [10]. These methods reduce features during model training using regularization or activation functions [18,34]. They balance accuracy and computational costs better than wrappers but still carry overfitting risks and require redesigning for specific algorithms [11].

2.1.4. Hybrid Methods

Hybrid methods combine filters with wrappers to enhance feature selection efficiency and accuracy. They first use filters to reduce dimensionality and then apply wrappers for precise searches. This approach has shown significant performance improvements in various domains like text classification and semiconductor manufacturing yield prediction [35–37].

In semiconductor manufacturing, feature selection methods are more appropriate than feature extraction for addressing anomalies in wafer yield, maintaining data interpretability after dimensionality reduction.

2.2. Classification Algorithms

Machine learning develops algorithms that simulate human intelligence, adjusting function structures dynamically through iterative learning. It can substitute repetitive tasks and identify data regularities overlooked by humans [38]. Machine learning is categorized into supervised, unsupervised, and semi-supervised learning. This study primarily employs supervised learning due to its wide applicability and good performance across various domains.

2.2.1. K-Nearest Neighbor (KNN)

The K-nearest neighbor algorithm (KNN) predicts the class of unknown data points based on nearby known samples. It calculates distances between a new data point and training samples, classifying the new point based on the major class of the nearest neighbors. KNN is straightforward, using parameters like the integer k, labeled data, and a distance formula. Despite high computational costs and sensitivity to noise, KNN is widely applied in domains such as data mining and image processing. In semiconductor manufacturing, KNN is used for online fault detection and failure detection [39].

This study aims to utilize heuristic algorithm-based feature selection methods combined with machine learning techniques to propose an anomaly detection model tailored to high-dimensional, nonlinear, and imbalanced data. The objectives are as follows:

When combining a filter with a heuristic algorithm—the simplified swarm optimization (SSO) algorithm—and adopting a non-binary encoding approach, we propose a hybrid feature selection method. This method not only reduces data dimensionality but also accurately selects the crucial influencing factors for anomaly prediction.

When integrating feature selection methods, we propose an anomaly detection approach suitable for multivariate, imbalanced data. This method is applied to real-world cases to achieve more precise and real-time wafer quality management.

2.2.2. Support Vector Machine (SVM)

Support vector machine (SVM), proposed by Boser, Guyon, and Vapnik in 1992 [40], has matured significantly over more than a decade of development. Today, SVM stands out as one of the most widely applied algorithms in machine learning due to its robustness, unique global optimal solution, and excellent generalization capability. Traditional linear algorithms may yield suboptimal results when faced with nonlinear data distributions. SVM addresses this issue by seeking the optimal hyperplane in the feature space for classification. By increasing the dimensionality of the feature space and then conducting segmentation, SVM achieves better classification results.

SVM is a kernel technique that uses kernel methods to map data into a higher-dimensional space. It treats machine learning tasks as convex function optimization problems, aiming to find the optimal solution through computation rather than heuristic algorithms. Upon completion of the SVM computation, support vectors are obtained, each defining the boundary of a hyperplane. The complexity of the problem affects the number of support vectors. SVM imposes additional constraints on optimization problems: the hyperplane must be positioned at the maximum distance between different classes, enhancing the generalization capability of SVM [41].

SVM has various variations that can be used to solve classification, regression, or distribution estimation problems. In classification tasks, C-Support Vector Classification (C-SVC) [42,43] is primarily employed. C-SVC transforms the classification problem into a primal optimization problem, assuming a set of training vectors $x_i \in R^n$, $i = 1, \ldots, l$ with corresponding class vectors $y \in R^l$, where $y_i \in \{1, -1\}$:

$$MIN_{w,\ b,\ \xi}\ \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i \tag{1}$$

$$subject\ to\ y_i\left(w^T\phi(x_i) + b\right) \geq 1 - \xi_i \tag{2}$$

$$\xi_i \geq 0,\ i = 1,\ \ldots,\ l$$

in which, w is a vector; ξ_i are slack variables, allowing for errors in classification; $\phi(x_i)$ is a function mapping the vector x_i into a higher-dimensional space; and C is a positive regularization parameter greater than 0.

Since w is typically high-dimensional, the above primal optimization problem is often converted into a dual problem as shown in Equations (3) and (4) as follows:

$$MIN_\alpha\ \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{3}$$

$$subject\ to\ y^T\alpha = 0 \tag{4}$$

$$0 \leq \alpha_i \leq C,\ i = 1,\ \ldots,\ l$$

where $e = [1, \ldots, 1]^T$ is a vector consisting of all ones, Q is a semi-positive definite matrix satisfying $Q_{ij} \equiv y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) \equiv \phi(x_i)^T\phi(x_j)$ is a kernel function.

Upon solving the dual problem, according to the primal–dual relationship, it can be inferred that the optimal solution for w satisfies Equation (5):

$$w = \sum_{i=1}^{l} y_i\alpha_i\phi(x_i) \tag{5}$$

The decision function is given by Equation (6):

$$sgn\left(w^T\phi(x_i) + b\right) = sgn\left(\sum_{i=1}^{l} y_i\alpha_i K(x_i,\ x) + b\right) \tag{6}$$

In the C-SVC algorithm, the variables that have the greatest impacts on the classification results are C and the choice of kernel function. As C increases, the emphasis is on minimizing the number of misclassified instances; conversely, reducing C allows for more classification bias. There is a wide variety of kernel functions available, with the most commonly used ones including the following:

- linear kernel—$K\left(\mathrm{x_i}, \mathrm{x_j}\right) = \mathrm{x_i}^T.\mathrm{x_j}$
- polynomial function—$K\left(\mathrm{x_i}, \mathrm{x_j}\right) = \left(\gamma \mathrm{x_i}^T \mathrm{x_j} + r\right)^q, q > 0$
- hyperbolic tangent, also known as sigmoid—$K\left(\mathrm{x_i}, \mathrm{x_j}\right) = \tan\mathrm{h}\left(\gamma \mathrm{x_i}^T \mathrm{x_j} + r\right)$
- Gaussian radial basis function, RBF—$K\left(\mathrm{x_i}, \mathrm{x_j}\right) = e^{-\gamma\|\mathrm{x_i} - \mathrm{x_j}\|^2}$

SVM has been widely employed across various research domains, including text classification, image classification, bioinformatics, facial recognition, and various predictive tasks [44]. Despite its solid theoretical foundations and strong generalization capabilities, SVM faces challenges such as high computational costs for large datasets and the need to convert multi-class classification problems into binary classification problems before processing.

2.2.3. Decision Tree and Random Forest

When a set of rules is used to partition the predictive target space and can be represented in a tree structure, it is referred to as a decision tree (DT) as shown in Figure 1. DTs consist of branches and nodes, with the root node containing the entire dataset. Each branch represents a rule, and the child nodes (decision nodes or leaf nodes) associated with the branch retain only the data that satisfy the condition of that branch. Rules are established based on the dataset contained within each decision node. Leaf nodes indicate the cessation of branching.

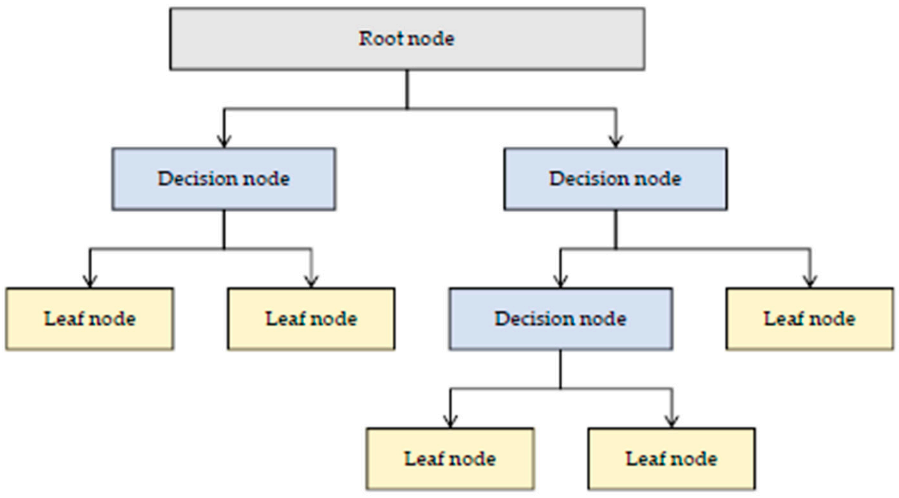

Figure 1. Decision tree structure.

Decision trees offer numerous advantages: they do not require data normalization, have a simple structure and fast computation time, and their models are easily interpretable. However, decision trees are highly sensitive to data, leading to poor robustness and susceptibility to overfitting, and in many applications, their accuracy still cannot match that of other machine learning methods. To address these limitations, Breiman introduced Random Forest (RF) in 2001 [45]. RF is an ensemble learning model based on decision trees, significantly improving prediction performance. It is still widely used today to solve various classification and regression problems.

In tree-based algorithms, the design of splitting rules and pruning strategies significantly influences model performance. Splitting rules directly impact classification outcomes, while pruning strategies determine when to terminate branching, avoiding overfitting due to overly detailed branching. The RF model adopts the Classification and Regression Tree (CART) algorithm [46] as its basis. CART restricts each decision node to perform binary splitting and employs the Gini coefficient as the splitting rule, along with cost-complexity pruning as the pruning rule.

The Gini coefficient (Gini index) is a metric used to measure information impurity. By calculating the extent to which the Gini coefficient decreases after data splitting, we can evaluate the quality of splitting rules and select the condition that maximally reduces impurity as the branching criterion. Suppose we have a dataset D, and the formula for calculating the Gini coefficient is as follows (Equation (7)):

$$Gini(D) = 1 - \sum_{i=1}^{c} (p_i)^2 \tag{7}$$

where p_i is a non-zero probability representing the ratio of data of class c in dataset D.

CART and other tree-based algorithms employ the same method of pre-pruning. During the tree branching process, a threshold is set to stop the growth of the decision tree, such as limiting the number of data points in leaf nodes or requiring the reduction of the Gini coefficient to be greater than 0.1. However, while pre-pruning is efficient, it carries a risk of over-pruning. Therefore, after the tree has completed its growth, CART introduces cost-complexity pruning to avoid generating leaf nodes with few samples and increase the decision tree's tolerance to noise. This is achieved by evaluating the tree using the following equation (Equation (8)):

$$R_\alpha(T) = R(T) + \alpha\,|\mathrm{t}|,\ \forall \mathrm{t} \in \mathrm{T} \tag{8}$$

where $R(T)$ represents the sum of residuals for all leaf nodes in a tree, $|t|$ denotes the number of leaf nodes in the tree, and α is a penalty factor. The tree structure with the maximum $R_\alpha(T)$ is selected as the model structure.

Random Forest (RF) constructs multiple CART trees based on these rules. Each tree originates from the same data distribution but operates independently. RF introduces two levels of randomness during training; before the generation of each tree, a random sample of data points is extracted from the original dataset, and a random subset of features is selected from the original feature space to form the tree model. During the prediction phase, for classification problems, RF employs a majority vote based on the results of each CART submodel to obtain the final result. The law of large numbers and randomness in RF effectively prevent overfitting and have shown good predictive accuracy for various classification datasets [45].

In summary, the KNN, SVM, and RF models are all machine learning algorithms and relatively simple non-linear models. These three supervised learning methods have demonstrated good performance across various classification problems [13]. Therefore, in this study, an experiment was designed to select the most suitable classification model from these three algorithms to serve as the evaluation model after feature selection.

2.3. Simplified Swarm Optimization

In numerous practical applications, the issues encountered tend to be relatively complex and frequently fall under the category of NP-hard problems, rendering the determination of optimal solutions within finite timeframes challenging. Consequently, many scholars resort to metaheuristic algorithms—a class of methods that, without guaranteeing feasibility or optimality, aim to find approximate optimal solutions at reasonable computational costs—to address such problems [47]. In the context of feature selection problems, metaheuristic algorithms such as genetic algorithms and particle swarm optimization (PSO) are commonly employed for solution derivation.

Simplified swarm optimization (SSO) was initially proposed by Yeh in 2009 [48] as an enhanced version of particle swarm optimization (PSO) [49], a stochastic optimization method based on swarm intelligence. Swarm intelligence-based approaches typically generate a set of solutions and then update them by following the lead of the best solutions within the swarm. Through continuous updating and iteration, these methods gradually approach the optimal solution.

The core concept of SSO lies in integrating the global best solution (gbest), local best solution (pbest), self-solution, and a random solution at each iteration. The updated position of the solutions is determined by random numbers and parameters predefined within the algorithm. This enables SSO to maintain solution diversity during the search process, thus avoiding convergence to local optima. The specific updating formula is as follows (Equation (9)):

$$x_{ij}^{t+1} = \begin{cases} gbest_j \; if \; \rho_{ij}^t \in [0, \, C_g) \\ pbest_{ij}^t \; if \; \rho_{ij}^t \in [C_g, \, C_P) \\ x_{ij}^t \; if \; \rho_{ij}^t \in [C_p, C_w) \\ x \; if \; \rho_{ij}^t \in [C_w, \, 1) \end{cases} \tag{9}$$

Here, x_{ij}^{t+1} represents the *j*th variable of the *i*th solution in the *t*th iteration; ρ is a uniformly distributed random number in the interval [0, 1]; (C_g, C_p, and C_w) are the hyperparameters of SSO, ranging in the interval (0, 1) with the constraint $C_g < C_p < C_w$; and x is a random number generated within predefined upper and lower bounds. If ρ falls in the interval [0, C_g), the variable x_{ij}^{t+1} is set to the *j*th variable corresponding to the *gbest* value. If ρ falls in the interval [C_g, C_p), the variable x_{ij}^{t+1} is set to the *j*th variable corresponding to the best historical solution *pbest* of the *i*th solution. If ρ falls in the interval [C_p, C_w), the variable remains the same as in the previous generation. If ρ falls in the interval [C_w, 1), a new variable is randomly generated. The introduction of the random variable x helps maintain diversity in the search process, preventing the results from being trapped in local optima. Furthermore, since (C_g, C_p, and C_w) directly influence the convergence speed and performance of SSO, past studies related to SSO often utilized Taguchi orthogonal arrays to select the optimal parameter combinations.

SSO addresses the shortcomings of PSO in solving discrete problems by designing step functions, thereby mitigating premature convergence and suboptimal performance. Extensive research across various domains, such as task allocation [50–52], facility location [53,54], and several practical areas [55–58], has consistently demonstrated that SSO outperforms traditional PSO or GA in both efficiency and solution quality. Moreover, SSO has been successfully applied to feature selection problems.

Chung and Wahid proposed a hybrid system integrating SSO with specialized preprocessing methods and local search strategies, demonstrating its effectiveness in network intrusion detection classification problems [59]. Lai et al. combined different filters and various wrapper methods derived from SSO, proving that hybrid selection methods significantly improve cancer classification problems [60].

3. The Proposed Approach

The objective of this study is to achieve optimal predictive accuracy with a minimal feature count. Feature selection is conducted in two stages. In the first stage, mutual information (MI) is employed as a filter to assess the relevance of features to the prediction target. Features with lower MI values are removed after ranking, significantly reducing the feature space. In the second stage, simplified swarm optimization (SSO) is utilized to search within the limited feature space. The fitness function is computed based on the results obtained from the classification algorithm. Through the iterative addition and removal of features within feature subsets, an optimal feature combination is identified. The hybrid feature selection method employed in this study is termed MI-SSO.

3.1. Mutual Information

Mutual information (MI) is employed to gauge the interdependence between two variables, utilizing an information entropy estimation based on K-nearest neighbor distances. It yields higher accuracy at lower computational costs compared to conventional bootstrap aggregation (bagging) methods. The specific formula for mutual information is as follows (Equation (10)):

$$\begin{aligned} MI(X, \, Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned} \tag{10}$$

In this context, X represents features, Y denotes the predictive target, H stands for information entropy, $H(X \mid Y)$ signifies the conditional entropy of X given Y values, and $H(X, Y)$ represents the joint entropy of X and Y. As the MI value increases, it indicates a higher correlation with the predictive target. Following the computation of MI values for all features in accordance with Equation (10), features can be sorted in descending order based on their MI values. Depending on the requirements, the top K features can be retained as initial candidate solutions for subsequent feature selection stages.

3.2. Feature Selection Based on SSO

In the second stage, simplified swarm optimization (SSO) selects the optimal feature subset from the reduced feature space. SSO uses swarm intelligence to explore efficiently, evaluating subsets based on predictive accuracy. By iteratively updating with global best, local best, and random solutions, SSO balances exploration and exploitation, reducing premature convergence and increasing the chance of finding the global optimum.

3.2.1. Particle Encoding Method

When solving the feature selection problem using heuristic algorithms, binary encoding is commonly employed [31,32,61], with the length of the solution equivalent to the number of features. However, when the number of features becomes excessive, binary encoding may result in prolonged solution times. Therefore, when there are constraints on the maximum number of feature subsets, the use of multivariate encoding can shorten the solution length and computational time.

The following description outlines the encoding and decoding process employed in this study: Initially, the K candidate features are sequentially numbered starting from 0 as integers. Under the constraint of selecting at most j features, the particle represents the numerical set of selected feature combinations, where the ith particle can be represented as: $X^i = \left(x^i_1, x^i_2, x^i_3, \ldots x^i_j\right)$, subject to the conditions (1) x^i_k is an integer and (2) $0 \le x^i_k < N$, $\forall k \le j$. For example (Figure 2), assuming there are 60 features in the dataset awaiting selection and the intention is to retain only 10% of the features, thus the solution length is set to 6. If a particle $X = (34, 5, 28, 5, 5, 11)$ is given, it indicates the selection of features numbered 34, 5, 28, and 11 from the original feature space, totaling 4 features. From this example, it can be observed that in this encoding method, the occurrence of duplicate values within particles is allowed. Although restricted to selecting at most 6 features, the obtained results may be less than the upper limit. The flexible design allows for the search of fewer feature quantities during the solving process.

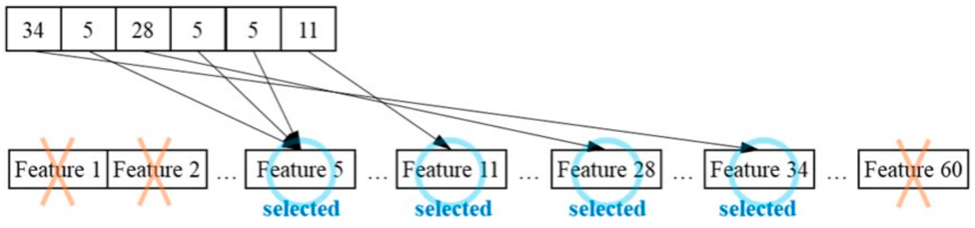

Figure 2. Particle encoding method for a simple case.

Shorter solution lengths can reduce the computational time, but excessively short lengths may overlook optimal solutions. When designing solution lengths, consider historical research and optimal feature quantities from other methods. Flexibility is crucial to avoid overly stringent restrictions. The fitness function controls the precise number of features using penalty functions to prevent excessive parameters.

3.2.2. Fitness Function

As described in Section 3.2.1, the classification performance varies with each feature subset inputted into the classifier. Therefore, a fitness function is needed to evaluate the performance of solutions generated during the iterative updating process of SSO. The fitness function in this study is designed to achieve the best model prediction accuracy with the fewest features. To address this dual-objective problem, we refer to [60] and employ a weighted approach, as shown in Equation (11):

$$Max\ \text{Fitness}(f) = \alpha\ \frac{\text{MCC}_{SCV=k}\left(\text{C}_f\right) + 1}{2} + (1-\alpha)\frac{\delta(F) - \delta(f)}{\delta(F)} \tag{11}$$

The fitness function consists of two parts: classification accuracy and the number of features. A higher fitness (f) indicates better solution quality. $\text{MCC}_{SCV}\left(\text{C}_f\right)$ is the average Matthews Correlation Coefficient (MCC) from stratified K-fold cross-validation (SCV) using classifier C with feature subset f. $\delta(F)$ and $\delta(f)$ denote the numbers of features in the original dataset and subset, respectively. The fewer features in the subset, the higher the fitness value. α is a weight between [0, 1], which is adjustable based on the importance of $\delta(f)$.

Given the dataset's imbalance, SCV ensures consistent class proportions in training and validation sets [62]. The fitness function uses MCC for evaluation, as it handles class imbalance better than accuracy and extends to multi-class problems through derivation, as shown in Equation (12) [63]:

$$\text{MCC} = \frac{cov(X,\ Y)}{\sqrt{cov(X, X)cov(Y,\ Y)}} \tag{12}$$

The specific calculation method for the underlying correlation coefficients is as follows (Equation (13)):

$$\begin{aligned} cov(X,Y) &= \textstyle\sum_{k=1}^{N} w_k\ cov(X_k,\ Y_k) \\ &= \tfrac{1}{N}\textstyle\sum_{s=1}^{S}\sum_{k=1}^{N}\left(X_{sk} - \overline{X}_k\right)\left(Y_{sk} - \overline{Y}_k\right) \end{aligned} \tag{13}$$

Let N denote the number of classes and s denote a single data instance. X_{sn} is a binary variable indicating the true class label of s, and Y_{sn} indicates the predicted class label. $\overline{X}_k$ and $\overline{Y}_k$ represent the proportions of the true and predicted classes k in the dataset.

The MCC value ranges from -1 to 1, with 1 indicating perfect prediction, 0 indicating random prediction, and -1 indicating complete disagreement between predictions and observations. The feature proportion in the fitness function ranges from 0 to 1. To align their numerical ranges, MCC is normalized [64]. After shifting and scaling, MCC values fall within 0 to 1.

3.2.3. Updating Steps

Aligned with the integer encoding approach, the basic SSO updating mechanism is employed. Initially, a set of solutions is randomly generated from the MI candidate solutions, and *gbest* and *pbest* are recorded based on the fitness function. During the updating process, each variable of the solution is updated according to the step function (Equation (9)). After updating, the fitness function is computed, and *gbest* and *pbest* are updated. This cycle iterates until convergence. The update process is illustrated in the flowchart as shown in below Figure 3.

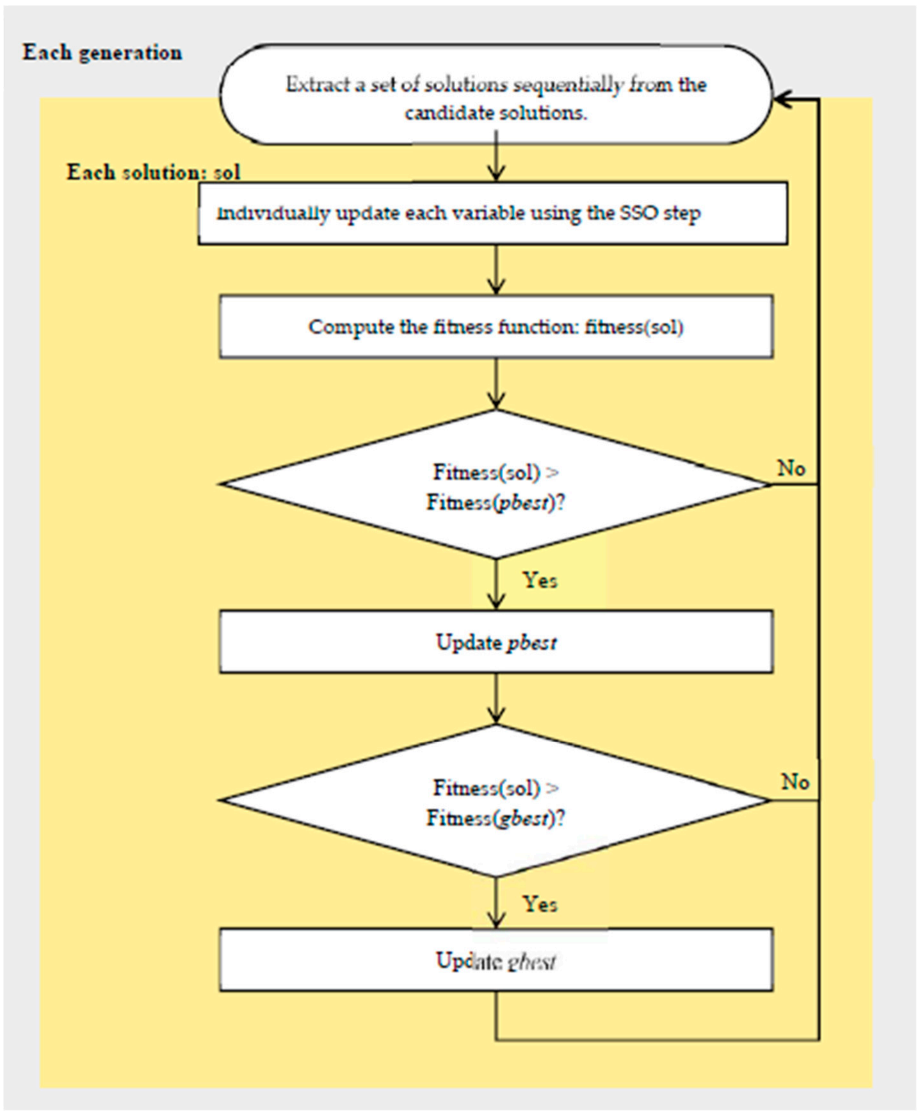
Each generation
Extract a set of solutions sequentially from the candidate solutions.
Each solution: sol
Individually update each variable using the SSO step
Compute the fitness function: fitness(sol)
Fitness(sol) > Fitness(*pbest*)?
No
Yes
Update *pbest*
Fitness(sol) > Fitness(*gbest*)?
No
Yes
Update *gbest*

Figure 3. Flowchart of SSO updating steps.

4. Experimental Results and Analysis

This section introduces the data used in the experiments (Section 4.1), explains the proposed feature selection method and hyperparameter configuration (Section 4.2), presents the experimental results from MI-SSO and compares its performance with other methods (Section 4.3), and details the application of MI-SSO in semiconductor anomaly detection (Section 4.4).

4.1. Description of the Datasets

Due to the confidential nature of semiconductor manufacturing, obtaining related datasets is challenging. Therefore, this study uses publicly available datasets for validating feature selection. To approximate wafer anomaly classification, selected datasets must have more features than data instances and at least half must be multi-class. The following five publicly available datasets [18,42,43,65,66] were chosen for experimentation as shown in Table 1:

Table 1. Basic description of the datasets.

Dataset Name	Number of Features	Number of Instances	Numbers of Classes and Proportions	Source
Brain2	10,367	50	4 (14:7:14:15)	[42]
Breast	24,481	97	2 (51:46)	[65]
Colon	2000	60	2 (40:22)	[43]
Lung	3312	203	5 (139:17:21:20:6)	[18]
MLL	12,582	72	3 (24:20:28)	[65]
Ovarian	15,154	253	2 (162:91)	[66]

For wafer anomaly classification, this study used data from a leading wafer foundry in Taiwan. After preprocessing, the dataset included 426 wafer batches with measurements for 672 test items, with features outnumbering instances by approximately 1.57 times. Domain experts categorized the wafers into three classes, normal, risk, and anomaly, with proportions of 380:23:14, respectively.

4.2. MI-SSO Parameter Configuration

The parameters pre-configured in this study include: the number of candidate solutions K selected by the MI-based feature selection in the first stage, SSO hyperparameters (C_g, C_p, and C_w) in the second stage, the SSO solution length N_{var}, the optimal α in the fitness function, and the classification algorithm for quality verification.

To determine the optimal parameter combination for MI-SSO, we conducted experiments using six publicly available datasets to assess various parameter settings' impacts on the prediction results. Given the longer computation times of heuristic algorithms, small-sample experiments were employed. Each parameter combination was tested 10 times, recording the Matthews Correlation Coefficient (MCC) and the number of selected features (#F). The average MCC and #F values after 10 repetitions evaluated classification performance.

Compared to other parameters, the choice of classifier significantly impacts the quality of classification results. Therefore, we first compared the effects of using three different algorithms, namely KNN, SVM, and RF, as classifiers on the feature selection results. In this experiment, the hyperparameters of SSO ($\mathrm{C_g}$, $\mathrm{C_p}$, and $\mathrm{C_w}$) were set to default values (0.4, 0.7, and 0.9), and the α parameter in the fitness function was set to 0.8 according to reference [60]. The number of candidate solutions K was set to 100, following the recommendation in Ref. [67]. The iteration number (*Ngen*) of SSO was set to 100, the number of solutions (*Nsol*) was set to 50, and k in SCV was set to four. All classifiers were retrained using grid search to search for model hyperparameters based on different datasets. According to the experimental results (Table 2) and considering the classification results across all six datasets, SVM achieved the highest classification accuracy, selected the fewest features, and required the lowest computational time. Therefore, in all subsequent experiments, SVM was chosen as the classifier for the MI-SSO method.

Table 2. Comparison of different classifiers (average).

Dataset		KNN	SVM	RF
Avg.	MCC	0.742544	0.758584	0.757617
	#F	21.552381	20.628571	21.666667
	Time (min)	6.666159	5.785098	11.083760

The classifier choice significantly impacts classification results. We compared KNN, SVM, and RF as classifiers for feature selection. SSO hyperparameters ($\mathrm{C_g}$, $\mathrm{C_p}$, and $\mathrm{C_w}$) were set to default values (0.4, 0.7, and 0.9), and α in the fitness function was set to 0.8 [60]. The number of candidate solutions K was set to 100 [67], the generation number N_{gen}

to 100, the number of solutions (N_{sol}) to 50, and k in SCV to four. All classifiers were retrained using grid search for hyperparameters. The experimental results (Table 2) showed that SVM achieved the highest accuracy, selected the fewest features, and required the lowest computational time. Therefore, SVM was chosen as the classifier for MI-SSO in all subsequent experiments.

After selecting the classifier, the next step involved finding the optimal combination of SSO hyperparameters through experimentation. The experimental design for hyperparameter combinations used the I9 orthogonal array from the Taguchi method, reducing the 27 sets of experiments with three levels and three factors to 9 sets. However, one combination did not meet the prerequisite condition $Cg \leq Cp \leq Cw$, resulting in eight different hyperparameter combinations.

Keeping other experimental conditions constant, the results (Table 3) reveal that although the combinations with the highest MCC or the fewest features vary across the six datasets, with the better-performing combinations showing similar performance. The difference in MCC is less than 0.05, and the variance in the number of selected features is less than one. Therefore, to determine the optimal SSO hyperparameter combination, the results from the six datasets were collectively considered, and the combination with the highest average fitness function value was chosen: (Cg, Cp, Cw) = (0.4, 0.7, 0.9).

Table 3. Comparison of classification performance with different combinations of SSO hyperparameters (average).

Dataset	**Cg**	**0.4**	**0.4**	**0.4**	**0.5**	**0.5**	**0.5**	**0.6**	**0.6**
	Cp	**0.5**	**0.6**	**0.7**	**0.5**	**0.6**	**0.7**	**0.6**	**0.7**
	Cw	**0.7**	**0.8**	**0.9**	**0.8**	**0.9**	**0.7**	**0.7**	**0.8**
Brain2	MCC	0.8796	**0.9025**	0.8937	0.8795	0.8954	0.8898	0.8913	0.8887
	#F	29.0	26.8	**23.1**	27.4	23.2	26.9	27.5	24.5
Breast	MCC	0.7024	0.7903	0.8278	0.8129	**0.8337**	0.8330	0.7787	0.7886
	#F	30.2	29.2	25.6	29.8	**25.4**	29.7	29.1	28.2
Colon	MCC	0.8682	0.8661	0.8825	0.8597	0.8623	0.8674	0.8715	**0.8763**
	#F	27.6	27.5	28.1	27.5	27.6	**27.3**	27.4	28.0
Lung	MCC	0.8471	0.8857	**0.9120**	0.8846	0.9113	0.8441	0.8538	0.8921
	#F	32.2	32.1	33.1	32.7	31.1	**30.4**	32.8	32.8
MLL	MCC	0.9920	**0.9952**	0.9933	0.9907	0.9939	0.9932	0.9924	0.9920
	#F	23.5	20.7	**15.7**	20.4	15.8	21.9	22.6	19.9
Ovarian	MCC	0.9979	0.9982	**0.9987**	0.9977	0.9977	0.9981	0.9969	0.9983
	#F	21.5	19.6	**14.4**	19.6	14.6	20.1	19.4	17.3
Avg.	MCC	0.8812	0.9063	**0.9180**	0.9042	0.9157	0.9043	0.8974	0.9060
	#F	27.3333	25.9833	23.3333	26.2333	**22.9500**	26.0500	26.4667	25.1167
	Fitness	0.9433	0.9553	**0.9608**	0.9542	0.9597	0.9543	0.9510	0.9551

After determining the classifier and SSO hyperparameters, only the number of candidate solutions K, the solution length N_{var}, and the hyperparameter α of the fitness function remain undecided. According to the experimental results of Dabba et al., when using MI as the first-stage feature selection method in microarray problems, K is set to 100 [67]. Both N_{var} and α affect the search direction and convergence speed of SSO. Given that classification accuracy is more important than the number of features, experiments were designed accordingly. Keeping other settings constant, the experimental results are shown in Table 4. Although MCC values under different combinations are close, the number of features varies significantly. This indicates that when using the MI-SSO method on different datasets, it is necessary to repeat the experiment to find the optimal feature combination rather than using fixed N_{var} and α combinations with the highest fitness function values chosen based on different datasets during experimentation.

Table 4. Impacts of alpha and the solution length on the classification results (average).

	α	**0.6**	**0.6**	**0.7**	**0.7**	**0.8**	**0.8**	**0.9**	**0.9**	**1**	**1**
Dataset	N_{var}	**30**	**50**	**30**	**50**	**30**	**50**	**30**	**50**	**30**	**50**
Brain2	MCC	0.9005	0.9061	0.9087	0.9021	0.9057	0.8976	0.9018	0.9080	0.9115	**0.9120**
	#F	**15.2**	23.2	16.7	22.4	16.5	22.2	16.3	22.8	22.7	32.5
	Fitness	0.9527	0.9553	0.9566	0.9534	0.9551	0.9513	0.9533	0.9561	0.9579	**0.9581**
Breast	MCC	**0.8644**	0.846246	0.815585	0.840691	0.8381	0.8524	0.8036	0.8521	0.7710	0.8522
	#F	17	25.4	16.4	27.3	**16.5**	26.6	17.1	26.1	22.9	34.7
	Fitness	**0.9355**	0.9269	0.9124	0.9243	0.9231	0.9298	0.9067	0.9297	0.8912	0.9297
Colon	MCC	**0.8793**	0.8718	0.8756	0.8714	0.8715	0.8750	0.8754	0.8746	0.8782	0.8710
	#F	20.7	28.3	19.3	27.3	**18.9**	27.3	20.3	28	23	29.8
	Fitness	**0.9422**	0.9384	0.9404	0.9382	0.9385	0.9399	0.9403	0.9397	0.9416	0.9380
Lung	MCC	0.9009	**0.9193**	0.9064	0.9157	0.9044	0.9061	0.9035	0.9146	0.9083	0.9135
	#F	23.1	31.8	23	33.6	**22.2**	33	24.1	33.3	26	37.3
	Fitness	0.9526	**0.9612**	0.9552	0.9594	0.9543	0.9549	0.9538	0.9589	0.9561	0.9583
MLL	MCC	0.9929	0.9941	0.9927	0.9943	0.9916	0.9936	0.9949	0.9930	0.9958	**0.9951**
	#F	10.7	16.6	10.7	15.8	10.3	16.9	**9.8**	16.6	21	31.9
	Fitness	0.9966	0.9971	0.9965	0.9972	0.9960	0.9969	**0.9975**	0.9966	0.9979	0.9976
Ovarian	MCC	0.9979	0.9974	0.9975	0.9984	**0.9988**	0.9980	0.9987	0.9984	0.9987	0.9987
	#F	8.9	15	8.5	14.6	**8.2**	14.8	8.3	15.1	21.8	31.2
	Fitness	0.9990	0.9987	0.9988	0.9992	**0.9994**	0.9990	0.9994	0.9992	0.9993	0.9993

Summarizing the numerous small-sample experiments conducted using publicly available datasets, it was found that MI-SSO, when employing SVM as the classifier and setting the parameters (Cg, Cp, Cw, and K) to (0.4, 0.7, 0.9, and 100), achieved higher classification accuracy with fewer features across different classification problems. However, the settings of N_{var} and α showed significant variations in feature selection results due to differences in data characteristics. Therefore, to achieve higher classification accuracy and fewer features, it is necessary to redesign experiments based on the specific characteristics of each dataset to find the optimal parameter combination.

4.3. Experimental Results

The encoding used in this study was implemented using Python 3.10.5 in Visual Studio Code. The experiments were conducted on a device equipped with an AMD Ryzen 5 5600G with Radeon Graphics processor running at 3.90 GHz and 16 GB of RAM. All datasets used in the experiments were preprocessed by imputing missing values with the mean and performing min–max normalization. Additionally, for each dataset, a grid search was employed to find the optimal hyperparameter combination for SVM.

According to the experimental results in Section 4.2, the experimental settings are as following Table 5:

Table 5. The experimental settings.

Hyperparameters	**Numerical Value**	**Illustrate**
K	100	Number of candidate solutions
N_{gen}	100	Number of iterations
N_{sol}	50	Number of solutions
N_{var}	30 or 50	Solution length, depending on the dataset
C_p	0.4	SSO hyperparameter
C_g	0.7	SSO hyperparameter
C_w	0.9	SSO hyperparameter
α	0.6, 0.7, 0.8, 0.9 or 1	Fitness function hyperparameters, depending on the dataset
k	4	Number of layers in SCV

In addition to MI-SSO, this study also conducted experiments using three other methods—MI, MI-GA, and MI-PSO—to compare them with the proposed method as follows:

- MI—This method uses only MI for feature selection. It sequentially selects the top K features based on their MI values (K = 1, 2, . . .) and evaluates their classification performance by incorporating them into the model. The process continues until there is no improvement in classification results for 100 consecutive solutions. This identifies the top K features that yield the best model performance.
- MI-GA—This method adapts SSO using a genetic algorithm (GA), with crossover and mutation probabilities set to 0.8 and 0.2, respectively [68].
- MI-PSO—This method adapts SSO using a particle swarm optimization (PSO) algorithm, with the inertia weight w set to 0.9 and the acceleration constants (c_1, c_2) set to (2, 2) [49].

The six public datasets were repeatedly tested using the specified parameters for 30 trials. The iteration number, number of solutions, and solution length for GA and PSO were the same as for SSO. The average performance of the four algorithms is shown in Table 6.

Table 6. Comparison of public datasets.

Dataset	Average	MI	MI-GA	MI-PSO	MI-SSO
Brain2	MCC	0.900086	0.853529	0.857373	**0.903063**
	#F	94	39.9	**28.5**	31.133333
Breast	MCC	0.833606	0.720417	0.535284	**0.848164**
	#F	90	25.933333	**14.466667**	16.333333
Colon	MCC	0.67774	0.76019	0.86627	**0.930007**
	#F	**18**	26.1	20.366667	19.633333
Lung	MCC	**0.953074**	0.874127	0.452402	0.918722
	#F	117	39.733333	**17.766667**	33.5
MLL	MCC	0.984438	0.972186	0.993398	**0.994614**
	#F	**4**	26.433333	16.466667	13.666667
Ovarian	MCC	**0.999289**	0.992263	0.998102	0.99863
	#F	53	25.933333	12.8	**8.633333**

MI-SSO achieved the highest MCC in five out of six datasets, indicating superior classification accuracy. However, for the Lung dataset, MI-SSO did not perform as well as MI, likely due to selecting too few features. Regarding the number of features, MI-SSO selected the fewest features only in the Ovarian dataset. In other datasets, MI-SSO selected relatively fewer features but not the fewest, showing a tendency to sacrifice some features for better accuracy, aligning with $\alpha > 0.5$ in the design.

In terms of runtime as shown in Table 7, MI had the shortest runtime, averaging 3 to 4 min depending on the dataset size. Among MI-GA, MI-PSO, and MI-SSO, MI-PSO was the fastest, followed by MI-GA and MI-SSO, with less than 8 s of difference among them.

Table 7. Algorithm running time (unit: minutes).

Dataset	MI-GA	MI-PSO	MI-SSO
SEMI	6.580849	6.904897	6.644581
Brain2	3.241753	3.209523	3.228779
Breast	4.874745	4.494404	4.755814
Colon	5.430743	5.18067	5.423277
Lung	3.115069	3.180372	3.13715
MLL	4.260329	4.080748	4.283061
Ovarian	4.92204	4.471951	4.887172
Average	4.632218	4.503224	4.622833

In the field of gene expression microarrays, numerous published papers explore the application of different feature selection methods. This study filtered five feature selection methods for comparison with MI-SSO based on the following criteria: published within the last four years, frequently cited, and tested using the same public datasets. The methods are MIM-mMFA [67], VS-CCPSO [68], MTPSO [69], TOPSIS-Jaya [70], and SARA-SVM [71]. Since accuracy is the primary evaluation criterion in gene expression microarrays, these five methods also use accuracy as the main benchmark and objective function. Therefore, to compare with other methods, the fitness function in MI-SSO was modified as follows:

$$Max\ \text{Fitness}(f) = \alpha\, \frac{\text{Accuracy}_{SCV=k}\left(C_f\right) + 1}{2} + (1-\alpha)\frac{\delta(F) - \delta(f)}{\delta(F)} \tag{14}$$

With the remaining experimental parameters unchanged, the experiment was repeated 30 times, and the results were recorded (Table 8). In terms of accuracy, MI-SSO achieved the best classification performance in two datasets (Breast and Ovarian). For the number of features, MI-SSO selected the fewest features in the Breast and Lung datasets. MIM-mMFA obtained the highest accuracy in three datasets (Brain2, Colon, and MLL), while SARA-SVM selected the fewest features in two datasets (Colon and Ovarian). MI-SSO balances maximum classification accuracy with the fewest features, and although it may not always outperform other techniques solely in terms of accuracy or feature count, it demonstrates competitiveness across the six gene expression microarray datasets and outperforms existing techniques in certain scenarios.

Table 8. Comparison of MI-SSO with other algorithms.

Dataset	Average	MIM-mMFA	VS-CCPSO	MTPSO	TOPSIS-Jaya	SARA-SVM	MI-SSO
Brain2	ACC #F	**1** **11.93**	0.8047 81.46	0.8540 1066.32			0.921967 31.3
Breast	ACC #F	0.868 25.9					**0.924472** **16.7**
Colon	ACC #F	**1** 26.3			0.9776 18.9	0.9702 **9**	0.952861 19.766667
Lung	ACC #F		**0.9791** 370.79	0.9740 343.24			0.956916 **32.6**
MLL	ACC #F	**1** 33			0.9962 **12.9**		0.996235 13.366667
Ovarian	ACC #F	0.9818 35.9			0.9952 18.5	0.9915 **6**	**0.999297** 8.7

4.4. Case Verification

After validating MI-SSO with publicly available data, this study applied MI-SSO to anomaly classification in semiconductor manufacturing. The data come from machine measurements and category labels annotated by engineers. The original dataset includes the following:

- Lot number—unique batch identifier;
- Wafer number—sequential number up to 25 wafers per batch;
- Parameter name—corresponding measurement parameters for each test;
- Measurement equipment—recorded equipment performing the test.
- Measurement points 1 to 5—floating-point numbers representing different characteristics at different positions on the same wafer;
- Label—category indicating wafer quality as good, bad, or risk.

Labels are assigned per batch as illustrated in Table 9, and so data from each batch (about 25 wafers) are aggregated into a single entry. The maximum and minimum values are selected based on measurement parameters and points. This results in a dataset with 426 entries and 672 features.

Table 9. Illustration of the case dataset.

Batch Number	p_1m_1 **Max**	p_1m_1 **Min**	p_1m_2 **Max**	p_im_j **Max/Min ...**	$p_{135}m_5$ **Max**	$p_{135}m_5$ **Min**	**Mark**
A	4.00636	3.88418	0.669097		4.99095	4.98854	good
B	3.96926	3.87155	0.611947	...	4.9946	4.99371	good
C	4.12133	3.88967	0.611947		4.98923	4.98752	bad
...	...	...	...		...	...	...

In the case study, MI-SSO used SVM as the classifier and SSO hyperparameters (C_g, C_p, and C_w) of (0.4, 0.7, and 0.9), with 100 candidate solutions (K). A small-sample experiment determined (N_{var}, α) = (30, 0.9) for the dataset (Table 10).

Table 10. The influences of different alpha values and solution lengths on the classification results for the semiconductor manufacturing dataset.

	α	**0.6**	**0.6**	**0.7**	**0.7**	**0.8**	**0.8**	**0.9**	**0.9**	**1**	**1**
Dataset	N_{var}	**30**	**50**	**30**	**50**	**30**	**50**	**30**	**50**	**30**	**50**
SEMI	MCC	0.9558	0.9604	0.9616	0.9610	0.9613	**0.9639**	0.9634	0.9607	0.9631	0.9562
	#F	**21.3**	29.5	23.3	31.9	23.5	32.6	22.5	35.2	25.9	38.7
	Fitness	0.9760	0.9754	0.9777	0.9749	0.9775	0.9759	**0.9787**	0.9738	0.9775	0.9710

After preprocessing (mean imputation, min–max normalization, and SVM hyperparameter tuning), feature selection and anomaly classification were performed using the MI-SSO, MI, MI-GA, and MI-PSO algorithms.

The results (Table 11) show that MI-SSO achieves the highest accuracy and fewest features for semiconductor anomaly classification. Although MI-SSO selects one more feature on average than PSO, it improves MCC by 0.02. MI-SSO outperforms GA and PSO-based methods and is more effective than MI alone.

Table 11. Comparison of the effectiveness of semiconductor manufacturing datasets.

Dataset	**Average**	**MI**	**MI-GA**	**MI-PSO**	**MI-SSO**
SEMI	ACC	0.993177	0.981726	0.990751	**0.994009**
	MCC	0.957388	0.882505	0.941981	**0.962464**
	#F	35	25.9	**22.64**	23.5
	Fitness	0.972539	0.945294	0.970054	**0.977992**

5. Conclusions

This study addresses the unique problem of wafer anomaly detection with a hybrid feature selection method combining mutual information (MI) and simplified swarm optimization (SSO). MI-SSO operates in two stages: MI filters out less relevant features, and SSO selects the most important subset from the reduced feature space, achieving precise classification with fewer features.

Experimental comparisons with MI, MI-GA, and MI-PSO show that MI-SSO achieves the highest classification performance with fewer features. SSO generally outperforms GA and PSO in convergence speed and consistently produces the best classification models with the smallest feature subsets.

MI-SSO not only enhances classification accuracy but also improves interpretability, helping us to understand the importance of certain features. For semiconductor manufacturing, MI-SSO helps intercept defective products, reduce processing costs, and detect hidden manufacturing problems early. The selected features provide valuable insights for engineers to optimize the measurement and manufacturing processes.

Author Contributions: Conceptualization, W.-C.Y. and C.-L.C.; methodology, W.-C.Y. and C.-L.C.; software, W.-C.Y. and C.-L.C.; validation, W.-C.Y.; formal analysis, W.-C.Y. and C.-L.C.; investigation, W.-C.Y.; resources, W.-C.Y. and C.-L.C.; data curation, W.-C.Y. and C.-L.C.; writing—original draft preparation, W.-C.Y. and C.-L.C.; writing—review and editing, W.-C.Y. and C.-L.C.; visualization, W.-C.Y.; supervision, W.-C.Y.; project administration, W.-C.Y.; funding acquisition, W.-C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science Council of Taiwan, R.O.C., under grant numbers MOST 110-2221-E-007-107-MY3.

Data Availability Statement: The data presented in this study are available in this article as the description in Section 4.1.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Moore, G.E. Cramming more components onto integrated circuits. *Proc. IEEE* **1998**, *86*, 82–85. [CrossRef]
2. Mack, C.A. Fifty years of Moore's law. *IEEE Trans. Semicond. Manuf.* **2011**, *24*, 202–207. [CrossRef]
3. Kikuchi, M. *Semiconductor Fabrication Facilities: Equipment, Materials, Processes, and Prescriptions for Industrial Revitalization*; Shimao: Taipei, Taiwan, 2016.
4. Kourti, T.; MacGregor, J.F. Process analysis, monitoring and diagnosis, using multivariate projection methods. *Chemom. Intell. Lab. Syst.* **1995**, *28*, 3–21. [CrossRef]
5. Baly, R.; Hajj, H. Wafer classification using support vector machines. *IEEE Trans. Semicond. Manuf.* **2012**, *25*, 373–383. [CrossRef]
6. He, Q.P.; Wang, J. Fault detection using the k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE Trans. Semicond. Manuf.* **2007**, *20*, 345–354. [CrossRef]
7. Piao, M.; Jin, C.H.; Lee, J.Y.; Byun, J.Y. Decision tree ensemble-based wafer map failure pattern recognition based on radon transform-based features. *IEEE Trans. Semicond. Manuf.* **2018**, *31*, 250–257. [CrossRef]
8. Shin, C.K.; Park, S.C. A machine learning approach to yield management in semiconductor manufacturing. *Int. J. Prod. Res.* **2000**, *38*, 4261–4271. [CrossRef]
9. Cheng, K.C.C.; Chen, L.L.Y.; Li, J.W.; Li, K.S.M.; Tsai, N.C.Y.; Wang, S.J.; Huang, A.Y.-A.; Chou, L.; Lee, C.-S.; Chen, J.E.; et al. Machine learning-based detection method for wafer test induced defects. *IEEE Trans. Semicond. Manuf.* **2021**, *34*, 161–167. [CrossRef]
10. Bolón-Canedo, V.; Sánchez-Maroño, N.; Alonso-Betanzos, A. *Feature Selection for High-Dimensional Data*; Springer: Berlin/Heidelberg, Germany, 2015.
11. Venkatesh, B.; Anuradha, J. A review of feature selection and its methods. *Cybern. Inf. Technol.* **2019**, *19*, 3–26. [CrossRef]
12. Fernández, A.; García, S.; Galar, M.; Prati, R.C.; Krawczyk, B.; Herrera, F. *Learning from Imbalanced Data Sets*; Springer: Berlin/Heidelberg, Germany, 2018.
13. Kaur, H.; Pannu, H.S.; Malhi, A.K. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–36. [CrossRef]
14. Jiang, D.; Lin, W.; Raghavan, N. A Gaussian mixture model clustering ensemble regressor for semiconductor manufacturing final test yield prediction. *IEEE Access* **2021**, *9*, 22253–22263. [CrossRef]
15. Fan, S.K.S.; Lin, S.C.; Tsai, P.F. Wafer fault detection and key step identification for semiconductor manufacturing using principal component analysis, AdaBoost and decision tree. *J. Ind. Prod. Eng.* **2016**, *33*, 151–168. [CrossRef]
16. Chien, C.F.; Wang, W.C.; Cheng, J.C. Data mining for yield enhancement in semiconductor manufacturing and an empirical study. *Expert Syst. Appl.* **2007**, *33*, 192–198. [CrossRef]
17. Eesa, A.S.; Orman, Z.; Brifcani, A.M.A. A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. *Expert Syst. Appl.* **2015**, *42*, 2670–2679. [CrossRef]
18. Li, J.; Cheng, K.; Wang, S.; Morstatter, F.; Trevino, R.P.; Tang, J.; Liu, H. Feature selection: A data perspective. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–45. [CrossRef]
19. Zebari, R.; Abdulazeez, A.; Zeebaree, D.; Zebari, D.; Saeed, J. A comprehensive review of dimensionality reduction techniques for feature selection and feature extraction. *J. Appl. Sci. Technol. Trends* **2020**, *1*, 56–70. [CrossRef]
20. Jović, A.; Brkić, K.; Bogunović, N. A review of feature selection methods with applications. In Proceedings of the 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 25–29 May 2015; pp. 1200–1205.

21. Dash, M.; Liu, H. Feature selection for classification. *Intell. Data Anal.* **1997**, *1*, 131–156. [CrossRef]
22. Kira, K.; Rendell, L.A. The feature selection problem: Traditional methods and a new algorithm. In Proceedings of the Tenth National Conference on Artificial Intelligence, San Jose, CA, USA, 12–16 July 1992; Volume 2, pp. 129–134.
23. Kononenko, I.; Šimec, E.; Robnik-Šikonja, M. Overcoming the myopia of inductive learning algorithms with RELIEFF. *Appl. Intell.* **1997**, *7*, 39–55. [CrossRef]
24. Kononenko, I. Estimating attributes: Analysis and extensions of RELIEF. In Proceedings of the European Conference on Machine Learning, Catania, Italy, 6–8 April 1994; pp. 171–182.
25. Yang, H.; Moody, J. Data visualization and feature selection: New algorithms for nongaussian data. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 29 November–4 December 1999; Volume 12.
26. Karegowda, A.G.; Manjunath, A.; Jayaram, M. Comparative study of attribute selection using gain ratio and correlation based feature selection. *Int. J. Inf. Technol. Knowl. Manag.* **2010**, *2*, 271–277.
27. Azhagusundari, B.; Thanamani, A.S. Feature selection based on information gain. *Int. J. Innov. Technol. Explor. Eng. (IJITEE)* **2013**, *2*, 18–21.
28. Alhaj, T.A.; Siraj, M.M.; Zainal, A.; Elshoush, H.T.; Elhaj, F. Feature selection using information gain for improved structural-based alert correlation. *PLoS ONE* **2016**, *11*, e0166017. [CrossRef] [PubMed]
29. Jadhav, S.; He, H.; Jenkins, K. Information gain directed genetic algorithm wrapper feature selection for credit rating. *Appl. Soft Comput.* **2018**, *69*, 541–553. [CrossRef]
30. Amaldi, E.; Kann, V. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theor. Comput. Sci.* **1998**, *209*, 237–260. [CrossRef]
31. Soufan, O.; Kleftogiannis, D.; Kalnis, P.; Bajic, V.B. DWFS: A wrapper feature selection tool based on a parallel genetic algorithm. *PLoS ONE* **2015**, *10*, e0117988. [CrossRef]
32. Vieira, S.M.; Mendonça, L.F.; Farinha, G.J.; Sousa, J.M. Modified binary PSO for feature selection using SVM applied to mortality prediction of septic patients. *Appl. Soft Comput.* **2013**, *13*, 3494–3504. [CrossRef]
33. Cai, J.; Luo, J.; Wang, S.; Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing* **2018**, *300*, 70–79. [CrossRef]
34. Guyon, I.; Weston, J.; Barnhill, S.; Vapnik, V. Gene selection for cancer classification using support vector machines. *Mach. Learn.* **2002**, *46*, 389–422. [CrossRef]
35. Sarkar, S.D.; Goswami, S.; Agarwal, A.; Aktar, J. A novel feature selection technique for text classification using Naive Bayes. *Int. Sch. Res. Not.* **2014**, *2014*, 717092. [CrossRef]
36. Bostani, H.; Sheikhan, M. Hybrid of binary gravitational search algorithm and mutual information for feature selection in intrusion detection systems. *Soft Comput.* **2017**, *21*, 2307–2324. [CrossRef]
37. Zhang, J.; Xiong, Y.; Min, S. A new hybrid filter/wrapper algorithm for feature selection in classification. *Anal. Chim. Acta* **2019**, *1080*, 43–54. [CrossRef] [PubMed]
38. Naqa, I.E.; Murphy, M.J. What is machine learning? In *Machine Learning in Radiation Oncology*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 3–11.
39. Zhou, Z.; Wen, C.; Yang, C. Fault detection using random projections and k-nearest neighbor rule for semiconductor manufacturing processes. *IEEE Trans. Semicond. Manuf.* **2014**, *28*, 70–79. [CrossRef]
40. Boser, B.E.; Guyon, I.M.; Vapnik, V.N. A training algorithm for optimal margin classifiers. In Proceedings of the Fifth Annual Workshop on Computational Learning Theory, Pittsburgh, PA, USA, 27–29 July 1992; pp. 144–152.
41. Awad, M.; Khanna, R.; Awad, M.; Khanna, R. Support vector machines for classification. In *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*; Apress: Berkeley, CA, USA, 2015; pp. 39–66.
42. Nutt, C.L.; Mani, D.R.; Betensky, R.A.; Tamayo, P.; Cairncross, J.G.; Ladd, C.; Pohl, U.; Hartmann, C.; McLaughlin, M.E.; Batchelor, T.T.; et al. Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. *Cancer Res.* **2003**, *63*, 1602–1607.
43. Alon, U.; Barkai, N.; Notterman, D.A.; Gish, K.; Ybarra, S.; Mack, D.; Levine, A.J. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proc. Natl. Acad. Sci. USA* **1999**, *96*, 6745–6750. [CrossRef]
44. Cervantes, J.; Garcia-Lamont, F.; Rodríguez-Mazahua, L.; Lopez, A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing* **2020**, *408*, 189–215. [CrossRef]
45. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
46. Breiman, L.; Friedman, J.; Olshen, R.; Stone, C.J. *Classification and Regression Trees*; Chapman & Hall/CRC: New York, NY, USA, 1984.
47. Beheshti, Z.; Shamsuddin, S.M.H. A review of population-based meta-heuristic algorithms. *Int. J. Adv. Soft Comput. Appl* **2013**, *5*, 1–35.
48. Yeh, W.C. A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems. *Expert Syst. Appl.* **2009**, *36*, 9192–9200. [CrossRef]
49. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.

50. Yeh, W.C.; Chuang, M.C.; Lee, W.C. Uniform parallel machine scheduling with resource consumption constraint. *Appl. Math. Model.* **2015**, *39*, 2131–2138. [CrossRef]
51. Yeh, W.C.; Wei, S.C. Economic-based resource allocation for reliable Grid-computing service based on Grid Bank. *Future Gener. Comput. Syst.* **2012**, *28*, 989–1002. [CrossRef]
52. Lee, W.C.; Chuang, M.C.; Yeh, W.C. Uniform parallel-machine scheduling to minimize makespan with position-based learning curves. *Comput. Ind. Eng.* **2012**, *63*, 813–818. [CrossRef]
53. Corley, H.W.; Rosenberger, J.; Yeh, W.C.; Sung, T.K. The cosine simplex algorithm. *Int. J. Adv. Manuf. Technol.* **2006**, *27*, 1047–1050. [CrossRef]
54. Yeh, W.C. A new algorithm for generating minimal cut sets in k-out-of-n networks. *Reliab. Eng. Syst. Safety* **2006**, *91*, 36–43. [CrossRef]
55. Luo, C.; Sun, B.; Yang, K.; Lu, T.; Yeh, W.C. Thermal infrared and visible sequences fusion tracking based on a hybrid tracking framework with adaptive weighting scheme. *Infrared Phys. Technol.* **2019**, *99*, 265–276. [CrossRef]
56. Bae, C.; Yeh, W.C.; Wahid, N.; Chung, Y.Y.; Liu, Y. A New Simplified Swarm Optimization (SSO) Using Exchange Local Search Scheme. *Int. J. Innov. Comput. Inf. Control* **2012**, *8*, 4391–4406.
57. Yeh, W.C. A new exact solution algorithm for a novel generalized redundancy allocation problem. *Inf. Sci.* **2017**, *408*, 182–197. [CrossRef]
58. Hsieh, T.J.; Yeh, W.C. Knowledge discovery employing grid scheme least squares support vector machines based on orthogonal design bee colony algorithm. *IEEE Trans. Syst. Man Cybern. Part B (Cybernetics)* **2011**, *41*, 1198–1212. [CrossRef] [PubMed]
59. Chung, Y.Y.; Wahid, N. A hybrid network intrusion detection system using simplified swarm optimization (SSO). *Appl. Soft Comput.* **2012**, *12*, 3014–3022. [CrossRef]
60. Lai, C.M.; Yeh, W.C.; Chang, C.Y. Gene Selection using Information Gain and Improved Simplified Swarm Optimization. *Neurocomputing* **2016**, *218*, 331–338. [CrossRef]
61. Song, X.F.; Zhang, Y.; Guo, Y.N.; Sun, X.Y.; Wang, L. Variable-size cooperative coevolutionary particle swarm optimization for feature selection on high-dimensional data. *IEEE Trans. Evol. Comput.* **2020**, *24*, 882–895. [CrossRef]
62. Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. *Int. Jt. Conf. Arti* **1995**, *14*, 1137–1145.
63. Gorodkin, J. Comparing two K-category assignments by a K-category correlation coefficient. *Comput. Biol. Chem.* **2004**, *28*, 367–374. [CrossRef] [PubMed]
64. Chicco, D.; Jurman, G. A statistical comparison between Matthews correlation coefficient (MCC), prevalence threshold, and Fowlkes–Mallows index. *J. Biomed. Inform.* **2023**, *144*, 104426. [CrossRef]
65. Zhu, Z.; Ong, Y.S.; Dash, M. Markov blanket-embedded genetic algorithm for gene selection. *Pattern Recognit.* **2007**, *40*, 3236–3248. [CrossRef]
66. Petricoin, E.F.; Ardekani, A.M.; Hitt, B.A.; Levine, P.J.; Fusaro, V.A.; Steinberg, S.M.; Mills, G.B.; Simone, C.; A Fishman, D.; Kohn, E.C.; et al. Use of proteomic patterns in serum to identify ovarian cancer. *Lancet* **2002**, *359*, 572–577. [CrossRef] [PubMed]
67. Dabba, A.; Tari, A.; Meftali, S.; Mokhtari, R. Gene selection and classification of microarray data method based on mutual information and moth flame algorithm. *Expert Syst. Appl.* **2021**, *166*, 114012. [CrossRef]
68. Heris, M.K. Practical Genetic Algorithms in Python and MATLAB—Video Tutorial. *Yarpiz*. 2020. Available online: https://yarpiz.com/632/ypga191215-practical-genetic-algorithms-in-python-and-matlab (accessed on 10 January 2024).
69. Chen, K.; Xue, B.; Zhang, M.; Zhou, F. Evolutionary multitasking for feature selection in high-dimensional classification via particle swarm optimization. *IEEE Trans. Evol. Comput.* **2021**, *26*, 446–460. [CrossRef]
70. Chaudhuri, A.; Sahu, T.P. A hybrid feature selection method based on Binary Jaya algorithm for micro-array data classification. *Comput. Electr. Eng.* **2021**, *90*, 106963. [CrossRef]
71. Baliarsingh, S.K.; Muhammad, K.; Bakshi, S. SARA: A memetic algorithm for high-dimensional biomedical data. *Appl. Soft Comput.* **2021**, *101*, 107009. [CrossRef]