A Graphical Model to Diagnose Product Defects with Partially Shuffled Equipment Data

Authors:

Gilseung Ahn, Sun Hur, Dongmin Shin, You-Jin Park

Date Submitted: 2020-01-20

Keywords: defect diagnosis, multi-source data fusion, equipment data analysis, graphical model, partially shuffled time series

Abstract:

The diagnosis of product defects is an important task in manufacturing, and machine learning-based approaches have attracted interest from both the industry and academia. A high-quality dataset is necessary to develop a machine learning model, but the manufacturing industry faces several data-collection issues including partially shuffled data, which arises when a product ID is not perfectly inferred and yields an unstable machine learning model. This paper introduces latent variables to formulate a supervised learning model that addresses the problem of partially shuffled data. The experimental results show that our graphical model deals with the shuffling of product order and can detect a defective product far more effectively than a model that ignores shuffling.

Record Type: Published Article

Submitted To: LAPSE (Living Archive for Process Systems Engineering)

Citation (overall record, always the latest version):	LAPSE:2020.0075
Citation (this specific file, latest version):	LAPSE:2020.0075-1
Citation (this specific file, this version):	LAPSE:2020.0075-1v1

DOI of Published Version: https://doi.org/10.3390/pr7120934

processes

MDPI

Article

A Graphical Model to Diagnose Product Defects with Partially Shuffled Equipment Data

Gilseung Ahn [1], Sun Hur [1,*], Dongmin Shin [1] and You-Jin Park [2]

[1] Department of Industrial and Management Engineering, Hanyang University, Ansan 15588, Korea; ahn.kilseung@gmail.com (G.A.); dmshin@hanyang.ac.kr (D.S.)
[2] Department of Industrial Engineering and Management, National Taipei University of Technology, Taipei 106, Taiwan; yjpark@mail.ntut.edu.tw
* Correspondence: hursun@hanyang.ac.kr; Tel.: +82-31-400-5265

Received: 26 October 2019; Accepted: 5 December 2019; Published: 8 December 2019

Abstract: The diagnosis of product defects is an important task in manufacturing, and machine learning-based approaches have attracted interest from both the industry and academia. A high-quality dataset is necessary to develop a machine learning model, but the manufacturing industry faces several data-collection issues including partially shuffled data, which arises when a product ID is not perfectly inferred and yields an unstable machine learning model. This paper introduces latent variables to formulate a supervised learning model that addresses the problem of partially shuffled data. The experimental results show that our graphical model deals with the shuffling of product order and can detect a defective product far more effectively than a model that ignores shuffling.

Keywords: partially shuffled time series; graphical model; equipment data analysis; defect diagnosis; multi-source data fusion

1. Introduction

The diagnosis of product defects is essential for improving productivity in manufacturing. The task involves deciding whether a product is defective, given product-related information (e.g., image of completed products, and equipment data). Thanks to large volumes of data collected from factories and the advent of machine learning algorithms, data-driven defect diagnosis is receiving increasing attention from both the industry and academia. Data-driven defect diagnosis involves the development of a classifier f by using a training dataset that consists of $\left(\boldsymbol{x}^{(i)},\ y^{(i)}\right)$ for $i = 1,\ 2,\ \cdots,\ n$, where $\boldsymbol{x}^{(i)}$ is a feature vector and $y^{(i)}$ is the label of product i, as well as using the classifier to diagnose product defects.

Data-driven defect diagnosis uses either a feature vector (i.e., input data), such as completed product data (e.g., the image and specification of products) or equipment (e.g., command and sensor) data. Equipment data, including the manufacturing process for each product, can be used to diagnose defects before a product is produced. For example, Manco et al. [1] applied outlier detection to predict door failures on metro trains, and Khalastchi et al. [2] proposed a hybrid approach to detect faults in robotic systems based on sensor readings, commands, and feedbacks from actuators. Song and Shi [3] proposed a quality-supervised double-layer method to detect quality-related faults in which the first layer uses principal component analysis (PCA) to detect the fault and the second layer finds the key variable orthogonal weight of the fault from the PCA. Fakhfakh et al. [4] developed a mathematical model for on-line fault diagnosis, resulting in the schedule delays of a flexible manufacturing system to minimize the total delays caused by faults. They also developed a constraint programming technique to solve this mathematical problem.

Multi-source data fusion that combines or merges data from multiple sources (e.g., multiple sensors and two or more machines) is necessary for realistic defect diagnosis with equipment data [5]. It can

be categorized as signal-level, feature-level, or decision-level fusion [6]. Among them, feature-level fusion extracts feature vectors from each data point and merges them based on product ID. It is used primarily to combine a dataset to train a defect diagnosis model [7,8].

However, multi-source data fusion cannot always be effectively conducted due to imperfections, correlations, inconsistency, disparateness, etc. [9]. In many cases (e.g., a specific subprocess such as milling in the manufacturing process), a product ID cannot be attached until the production process is finished. In such cases, inferring a product ID should be conducted first. However, it cannot be perfectly inferred when there are one or more workstations such that (1) two or more independent parallel machines reside in a workstation, (2) the processing time of each machine is variable, and (3) workstations cannot be observed from the outside due to safety issues. When equipment data are partially shuffled, there can be no guarantee that a record in data merged by feature-level fusion will indicate a product. We propose a graphical model to diagnose product defects with partially shuffled equipment data. The proposed graphical model calculates the probabilities that products are shuffled and the likelihood that a product is defective by considering the shuffling probability.

To the authors' best knowledge, this is the first study to address the real and common manufacturing situation in which each product cannot be perfectly traced because merged equipment data based on product IDs may be partially shuffled. This study proposes a simulation method to calculate the probability that product order is shuffled at each workstation based on production-line analysis. The method also includes the calculation of the maximum shuffling range R at workstation j and probability, $Q_{j,r}$, of two products are shuffled with shuffling range $r \leq R$. It also proposes a graphical model to diagnose product defects with partially shuffled equipment data with a consideration of product shuffling based on $Q_{j,r}$.

Section 2 of this paper describes the problem in detail, focusing on the situation in which partially shuffled equipment data are merged from multiple data sources. Section 3 develops a graphical model to diagnose defects with partially shuffled equipment data through three phases: (1) production-line analysis with simulation, (2) graphical model design, and (3) graphical model inference. Section 4 provides an example of a numerical application of the proposed graphical model, and Section 5 suggests future research directions.

2. Problem Description

Suppose there are serial workstations and that a product must visit each machine at every workstation to be completed. More formally, let product i be processed by machine k at workstation j from time t_1 to t_2. The feature for product i at workstation j, $\boldsymbol{x}_j^{(i)}$, is $\left(h_{j,k}^{(t_1)}, h_{j,k}^{(t_1+1)}, \cdots, h_{j,k}^{(t_2)}\right)$, where $h_{j,k}^{(t)}$ is the feature value collected from machine k at workstation j at time t and $t_1 + 1$ means time a unit time elapse (e.g., second and millisecond) after t_1. By expanding the problem to every product $i = 1, 2, \cdots, n$ and every workstation $j = 1, 2, \cdots, m$, a defect diagnosis dataset, $\left\{\left(\boldsymbol{x}^{(1)}, y^{(1)}\right), \cdots, \left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}$ can be generated in which $\boldsymbol{x}^{(i)} = \left(\boldsymbol{x}_1^{(i)}, \boldsymbol{x}_2^{(i)}, \cdots, \boldsymbol{x}_m^{(i)}\right)$ and $y^{(i)}$ are feature vector and a label of product i, respectively.

The problem is to train a classifier $f(\cdot)$ with training records $\left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$ and diagnose whether a given product is defective by using the trained classifier. It is a general approach to diagnose defects by using machine learning. However, this approach cannot be employed when some product-processing orders are partially shuffled and the shuffled orders are not known.

Consider a production line that consists of three workstations (W_1, W_2, W_3) where product orders maybe shuffled and shuffled orders are not perfectly inferred in W_2 because the workstation includes two parallel machines whose processing times are not constant and cannot be observed from outside, as illustrated in Figure 1. The circle indicates a machine, and the dotted rectangle indicates a workstation. Defect diagnosis for each product is conducted after the product is processed by W_3.

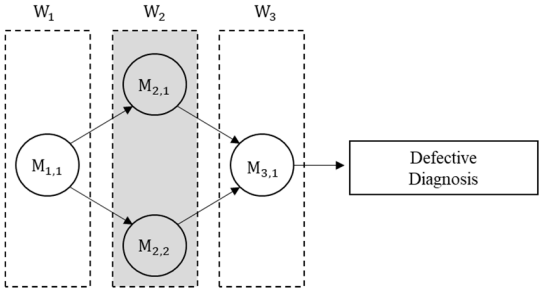

Figure 1. Example to illustrate processing orders that can be partially shuffled.

Suppose there are two products $i-1$, with i following the sequence $M_{1,1} \rightarrow M_{2,1} \rightarrow M_{3,1}$ and $M_{1,1} \rightarrow M_{2,2} \rightarrow M_{3,1}$. Suppose also that product $i-1$ enters $M_{1,1}$ ahead of product i, but product $i-1$ enters $M_{3,1}$ after product i. This situation can be illustrated in a Gantt chart (Figure 2), in which the arrow, rectangle, and colored rectangle denote the time to extract data, observed data, and shuffled observed data, respectively.

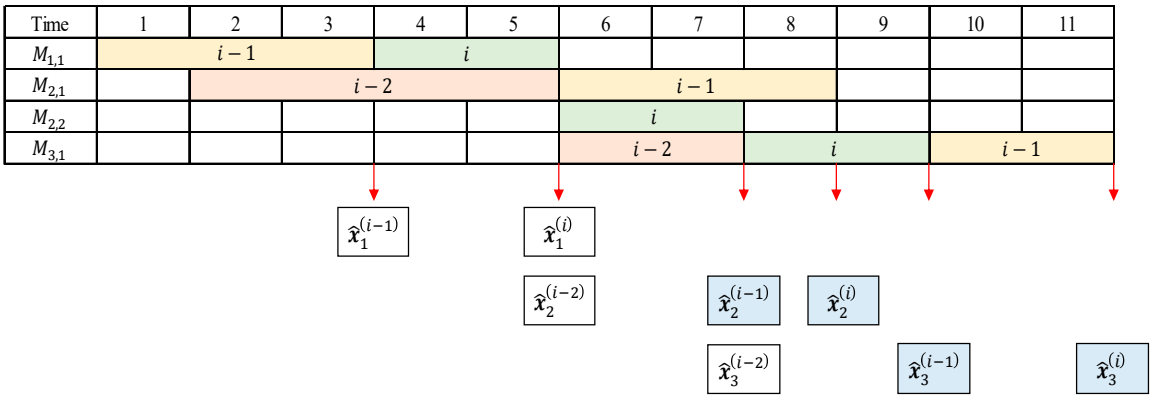

Figure 2. Gantt chart describing when product orders are shuffled.

As seen in Figure 2, equipment data for product $i-1$ are a concatenation of $M_{1,1}$'s data during [1,3], $M_{2,1}$'s data during [6,8], and $M_{3,1}$'s data during [10], and equipment data for product i is a concatenation of $M_{1,1}$'s data during [4,5], $M_{2,2}$'s data during [6,7], and $M_{3,1}$'s data during [8,9]. One cannot know which product is processed by machines $M_{2,1}$ and $M_{2,2}$ in W_2, because this workstation is not observed; one can only know if each machine is busy or idle. It is therefore impossible to identify which product is released at time 9 from W_3. In this situation, the feature vector for product i is not $\boldsymbol{x}^{(i)} = \left(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}\right)$, but $\hat{\boldsymbol{x}}^{(i)} = \left(x_1^{(i)}, x_2^{(i-1)}, x_3^{(i-1)}\right)$, which affects the training of the classifier $f(\cdot)$. In this paper, we call $\hat{\boldsymbol{x}}^{(i)}$ the observed feature vector for product i, and we called the equipment data collected for each product partially shuffled data.

3. Proposed Graphical Model

In this section, we construct a graphical model to diagnose defects with partially shuffled equipment data through three phases: (1) production-line analysis with simulation, (2) graphical model design, and (3) graphical model inference. In the first phase, we characterize the partially shuffled data by means of the shuffling probabilities saved in matrix Q. In the second phase, latent variables are introduced to design a graphical model structure. In the third phase, probabilistic supervised models for defect detection are trained, and an expectation of probability that a product is defective given the observed feature vector is calculated using the graphical model.

3.1. Production-Line Analysis with Simulation (Characteristics of Shuffled Data)

Let each $\widetilde{a}_j^{(i)}$ and $\widetilde{c}_j^{(i)}$ indicate the arrival order and completion order of product i at workstation j. In the absence of shuffling, $\widetilde{a}_j^{(i)} = \widetilde{c}_j^{(i)}$. If shuffling exists, $\widetilde{a}_j^{(i)} \neq \widetilde{c}_j^{(i)}$ for some i. Let us define the range of shuffling as $r \equiv |\widetilde{a}_j^{(i)} - \widetilde{c}_j^{(i)}|$ for product i. We can expect that the actual i^{th} completed product is either the $(i-r)^{th}$ or $(i+r)^{th}$ arriving product. Here, r is regarded as a random variable.

Let $Q_{j,\,r}$ denote the probability of a shuffle with a range of r at workstation j, which is formally expressed as:

$$Q_{j,r} = \Pr\left(\widetilde{c}_j^{(i_2)} - \widetilde{c}_j^{(i_1)} \in \{\gamma + r,\ \gamma - r\} | \widetilde{a}_j^{(i_1)} - \widetilde{a}_j^{(i_2)} = \gamma\right) \tag{1}$$

where $\widetilde{a}_j^{(i_1)} - \widetilde{a}_j^{(i_2)} = \gamma$ means that γ products arrive between the i_1^{th} and i_2^{th} arrival times at workstation j, and $\widetilde{c}_j^{(i_2)} - \widetilde{c}_j^{(i_1)} \in \{\gamma + r,\ \gamma - r\}$ implies that either $\gamma + r$ or $\gamma - r$ products are completed between the i_1^{th} and i_2^{th} completion at workstation j. That is, the i^{th} arrived product may not be the i^{th} completed product at the workstation because some products are completed faster or slower than expected. Maximum shuffling range R at workstation j is calculated as $R = \max\left\{r | Q_{j,\,r} > 0\right\}$, and $Q_{j,r}$ is defined for $r \leq R$.

Even though $Q_{j,r}$ plays an important role in developing the graphical model, it is not possible to analytically calculate it when the range of r is wide because there are many machines whose processing times are variables. However, we adopted a simulation method that could estimate $Q_{j,\,r}$. The first step is to generate simulated data that consist of simulated production routes for $\hat{n}$ products. Let $V^{(p)}$ be the processing route of simulated product p, which is a tuple of the machine, start time of processing, and the end time of processing at each workstation:

$$V^{(p)} = \left(\left(M_1^{(p)},\ a_1^{(p)},\ c_1^{(p)}\right),\ \left(M_2^{(p)},\ a_2^{(p)},\ c_2^{(p)}\right),\ \cdots,\left(M_m^{(p)},\ a_m^{(p)},\ c_m^{(p)}\right)\right) \tag{2}$$

where $M_j^{(p)}$, $a_j^{(p)}$, and $c_j^{(p)}$ indicate the machine that processes product p, its start time, and its end time (release time) at workstation W_j, respectively.

A processing route generation algorithm is presented as Algorithm 1. It repeats (1) randomly selecting one of the idle machines to process a product at a workstation, (2) sampling processing time from the distribution of the selected machine's processing time, (3) calculating the start and end time of the selected machine for the product, and (4) updating the busy/idle status of the selected machine for every product and workstation.

Algorithm 1 Product Route Generation Algorithm

Input	$\hat{n}$, $\mathrm{M}_{j,k}$, $\mathrm{F}_{j,k}$ for all j, k	
	01	Initialize $I_{j,k}^{(t)}$ as 1 for all $t,\ j,\ k$
	02	Initialize t as 1
	03	For $(p = 1; p \leq\ \hat{n},\ j++)$ do {
	04	For $(j = 1; j \leq m,\ j++)$ do {
Procedure	05	While $\left(\sum_{k=1}^{m_j} I_{j,k}^{(t)} = 0\right)$ do {Increase t by 1}
	06	$\hat{k} \overset{Random}{\leftarrow} \left\{k | I_{j,k}^{(t)} = 1\right\}$
	07	$a_j^{(p)} = t$
	08	$b_j^{(p)} \overset{sampling}{\leftarrow} \mathrm{F}_{j,k}(\cdot)$
	09	$c_j^{(p)} = t + b_j^{(p)}$
	10	$I_{j,\hat{k}}^{(t+\tau)} = 0$ for $\tau = 1,\ 2,\ \cdots, b_j^{(p)}$ }}
Output	$V^{(p)}$ for $p = 1,\ 2,\ \cdots,\ \hat{n}$	

A specific explanation of Algorithm 1 follows. Every machine's state for all t is initialized as idle (Line 01), and the time is also initialized (Lines 01–02). Product p is simulated (Lines 04~10): If there is no idle machine in workstation j at t, then t is increased by 1 until one or more machines become idle (Line 05); the product waits for processing when there is no idle machine. This product then enters a randomly selected idle machine $\hat{k}$ in workstation j (Line 06), and the start time for processing the product p is set to t (Line 07), and the processing time is sampled from its probability distribution (Line 8). The processing time determines the end time of processing product p. (Line 09), and the selected machine's state becomes busy (Line 10).

Using $V^{(p)}$ for $p = 1, 2, \ldots, \hat{n}$, obtained from Algorithm 1, we can calculate $Q_{j,r}$ in Equation (1). First, we calculate $\widetilde{a}_j^{(p)}$ and $\widetilde{c}_j^{(p)}$. based on $\left\{a_j^{(1)}, a_j^{(2)}, \cdots, a_j^{(\hat{n})}\right\}$ and $\left\{c_j^{(1)}, c_j^{(2)}, \cdots, c_j^{(\hat{n})}\right\}$ for all p, respectively. Then, we obtain the estimate of $Q_{j,r}$ from the ratio of pairs (i_1, i_2) that satisfies $\widetilde{c}_j^{(i_2)} - \widetilde{c}_j^{(i_1)} \in \{\gamma + r, \gamma - r\}$ among those $\widetilde{a}_j^{(i_1)} - \widetilde{a}_j^{(i_2)} = \gamma$.

3.2. Graphical Model Design (Graphical Model for Shuffled Data)

We can introduce a latent variable $\boldsymbol{z}^{(i)} = \left(z_1^{(i)}, z_2^{(i)}, \cdots, z_m^{(i)}\right)$ to indicate the ith completed product at each workstation. For example, if $z_j^{(7)} = 9$, then the 7th completed product at workstation j is the one that has entered into that workstation as the 9th. Actually, $z_j^{(i)}$ is unknown if workstation j is unobservable, but there can be only one i', such that $z_j^{(i')} = i$. By introducing these latent variables, we can develop a graphical model that represents the relationships between all $\boldsymbol{x}^{(i)}$ and $\hat{\boldsymbol{x}}^{(i)}$ values, and an inference to defect diagnosis is possible.

An exemplary structure of graphical model can be given with the help of latent variable $\boldsymbol{z}^{(i)}$, as presented in Figure 3 in the case $R = 2$.

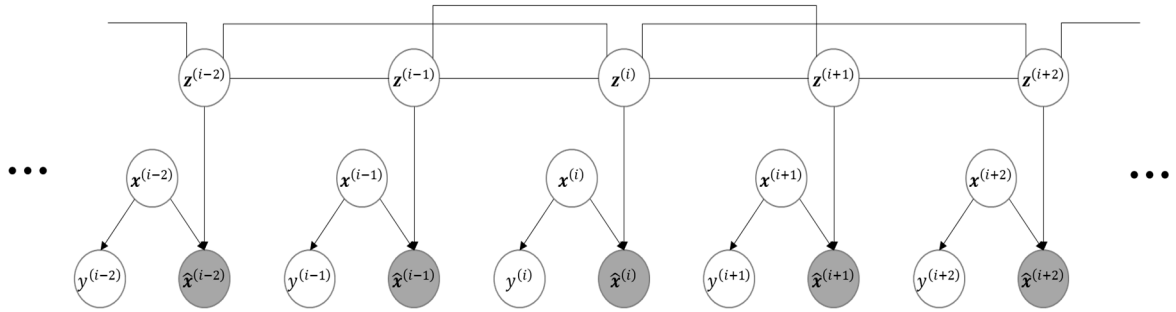

Figure 3. Structure of the graphical model (Case: $R = 2$).

In this figure, shaded circles indicate observable variables, while the others are unobservable. In general, node $\boldsymbol{z}^{(i)}$ is connected to $\boldsymbol{z}^{(i-R)}$, $\boldsymbol{z}^{(i-R+1)}$, $\cdots$, $\boldsymbol{z}^{(i+R)}$ because the variable $\boldsymbol{z}^{(i)}$ mathematically affects $\boldsymbol{z}^{(i-R)}$, $\boldsymbol{z}^{(i-R+1)}, \ldots, \boldsymbol{z}^{(i+R)}$, and $\hat{\boldsymbol{x}}^{(i)}$.

With the help of the graphical structure, $\hat{\boldsymbol{x}}^{(i)}$ given $\boldsymbol{z}^{(i)}$ can be expressed as:

$$\hat{\boldsymbol{x}}^{(i)} | \boldsymbol{z}^{(i)} = \left(\boldsymbol{x}_1^{(z_1^{(i)})}, \boldsymbol{x}_2^{(z_2^{(i)})}, \cdots, \boldsymbol{x}_m^{(z_m^{(i)})}\right) \tag{3}$$

$\boldsymbol{x}^{(i)}$ can be then inferred from $\hat{\boldsymbol{x}}^{(i')}$ and $\boldsymbol{z}^{(i')}$ for $i' = i - R, i - R + 1, \cdots, i + R$ as:

$$\boldsymbol{x}^{(i)} = \left(\sum_{i'=i-R}^{i+R} I\left(z_j^{(i')} = i\right) \times \hat{\boldsymbol{x}}_j^{(i')}\right)_{j=1}^{m} \tag{4}$$

where $I(condition)$ is 1 if the condition is satisfied, and 0, otherwise.

3.3. Inference (Defect Diagnosis)

The objective of this paper was to calculate the probability, $\Pr\left(y^{(i)}|\boldsymbol{x}^{(i)}\right)$, that product i is defective given its feature vector, which may be calculated by a probabilistic supervised model such as logistic regression. However, it cannot be directly obtained because $\boldsymbol{x}^{(i)}$ is unobservable, unlike the situation to which a supervised learning model is applied. Thus, we expanded $\Pr\left(y^{(i)}|\boldsymbol{x}^{(i)}\right)$ by means of the designed graphical model structure to consider all possible situations in which $\boldsymbol{x}^{(i)}$ is distributed in $\hat{\boldsymbol{x}}^{(i')}$ for $i' = i-R,\ i-R+1,\ \cdots,\ i+R$ as:

$$\Pr\left(y^{(i)}|\boldsymbol{x}^{(i)}\right) = \sum_{\widetilde{\boldsymbol{z}}^{(i)}} \Pr\left(y^{(i)}|\left(\sum_{i'=i-R}^{i+R} I\left(\widetilde{z}_j^{(i')} = i\right)\times \hat{\boldsymbol{x}}_j{}^{(i')}\right)_{j=1}^{m}\right)\times \Pr\left(\boldsymbol{z}^{(i)} = \widetilde{\boldsymbol{z}}^{(i)}\right) \tag{5}$$

Now, each part of the right-hand side of Equation (5) can be calculated.

(i) *Calculation of* $\Pr\left(\boldsymbol{z}^{(i)} = \widetilde{\boldsymbol{z}}^{(i)}\right)$

$\Pr\left(\boldsymbol{z}^{(i)} = \widetilde{\boldsymbol{z}}^{(i)}\right)$ in Equation (5) can be calculated using $Q_{j,\,r}$ from the production-line analysis in Section 3.1. When $R = 0$ (when there is no partial shuffling), then $\Pr\left(\boldsymbol{z}^{(i)} = i \times \mathbf{1}\right) = 1$ for all i. In this case, $\left(\sum_{i'=i-R}^{i+R} I\left(z_j^{(i')} = i\right)\times \hat{\boldsymbol{x}}_j{}^{(i')}\right)_{j=1}^{m}$ equals $\boldsymbol{x}^{(i)}$, implying that $\Pr\left(y^{(i)}|\boldsymbol{x}^{(i)}\right)$ is directly calculated from a probabilistic supervised model because $\boldsymbol{x}^{(i)} = \hat{\boldsymbol{x}}^{(i)}$. When $R \geq 1$, a conditional distribution of $z_j^{(i)}$ given $z_{j-1}^{(i)}$ can be obtained.

$$\Pr\left(z_j^{(i)} = z_{j-1}^{(i)}|z_{j-1}^{(i)}\right) = Q_{j,\,0} \tag{6}$$

$$\Pr\left(z_j^{(i)} = z_{j-1}^{(i)} + r|z_{j-1}^{(i)}\right) = \Pr\left(z_j^{(i)} = z_{j-1}^{(i)} - r|z_{j-1}^{(i)}\right) = \frac{Q_{j,r}}{2},\ \text{for } r = 1,\ \cdots,\ R. \tag{7}$$

$\boldsymbol{z}^{(i)}$'s distribution can be obtained recursively using Equations (6) and (7).

(ii) *Calculation of* $\Pr\left(y^{(i)}|\left(\sum_{i'=i-R}^{i+R} I\left(\widetilde{z}_j^{(i')} = i\right)\times \hat{\boldsymbol{x}}_j{}^{(i')}\right)_{j=1}^{m}\right)$

This probability is obtained by a supervised machine learning model with a training dataset composed of $\left(\hat{\boldsymbol{x}}^{(i)},\ y^{(i)}\right) = \left(\hat{\boldsymbol{x}}_1^{(i)},\ \hat{\boldsymbol{x}}_2^{(i)},\ \cdots,\hat{\boldsymbol{x}}_m^{(i)},\ y^{(i)}\right)$, i.e., the observed feature vector for product i and its label. Specifically, $\hat{\boldsymbol{x}}_j^{(i)},\ j = 1, 2, \cdots, m$ is the ith feature collected at workstation j; $y^{(i)}$ is 1 if the ith product is defective, and $y^{(i)}$ is 0 otherwise. Note that we do not know the exact value of $\boldsymbol{x}^{(i)}$, and it cannot be guaranteed that $\boldsymbol{x}^{(i)} = \hat{\boldsymbol{x}}^{(i)}$ because of possible shuffling. Instead, we utilized the value $\widetilde{\boldsymbol{z}}^{(i)}$ of the latent variable $\boldsymbol{z}^{(i)}$ to compose a dataset to train the model.

All terms of $I\left(\widetilde{z}_j^{(i')} = i\right)\times \hat{\boldsymbol{x}}_j{}^{(i')}$ $(i-R \leq i' \leq i+R)$ in the condition part of the probability vanish, except the one that satisfies $\widetilde{z}_j^{(i')} = i$. Therefore, only those observed feature vectors $\hat{\boldsymbol{x}}_j{}^{(i')}$ where $\widetilde{z}_j^{(i')} = i$ are required. This implies the feature vectors used for learning each model are determined by the distribution of $\boldsymbol{z}^{(i)}$. As an example, suppose that $\boldsymbol{z}^{(i)}$ can take either $(i,\ i+1,\ i)$, $(i,\ i-1,\ i)$ or $(i,\ i,\ i)$ when the shuffle range is $R = 1$. When $\boldsymbol{z}^{(i)} = (i,\ i+1,\ i)$ or $\boldsymbol{z}^{(i)} = (i,\ i-1,\ i)$, $\{\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_3\}$ is used as a feature set, but when $\boldsymbol{z}^{(i)} = (i,\ i,\ i)$, then the feature set is $\{\hat{\boldsymbol{x}}_1,\ \hat{\boldsymbol{x}}_2, \hat{\boldsymbol{x}}_3\}$.

4. Numerical Example

This section provides a numerical example to apply the proposed model. In addition, the classification performance of the defect diagnosis is compared between our model, which takes the possibility of shuffling of product order into consideration, and the model that ignores it and assumes $\boldsymbol{x}^{(i)} = \hat{\boldsymbol{x}}^{(i)}$ for all values of i. An F1 score, harmonic mean of the precision and recall, was employed as the classification performance measure to consider both cost of false positives (FPs) and false negatives (FNs). FN cost occurs when an actual defective product is not detected, and this cost covers selling defective products such as customer claim and safety accidents (note that the addressed product is a

brake disc, which is extremely safety-critical). FP cost occurs when a normal product is detected as defective, and this cost covers the costs of re-check, re-work, or disposal. Even though FN cost is much greater than FP cost, considering only one cost makes a classifier biased, so we employed an F1 score of 4.1. Line data

We generated an illustrative dataset by referring to a disk-brake processing line of a large-sized auto parts company in Korea for this numerical example's purpose. The generated data are very similar to the real data except for machine specifications and related parameters. Even though the data are artificial, we believe that the research result can be effectively applied to the real production field when a possible shuffle of production data exists because the main idea and methodology of the research focus on data-shuffling and its effect on the detection of defectives.

The processing line consists of seven workstations (S_1, S_2, $\cdots$, S_7) including rough grinding, drilling, and tapping. Figure 4 illustrates the processing line, where a circle, a rectangle, and a colored rectangle denote a machine, a workstation, and a workstation that is not observable, respectively.

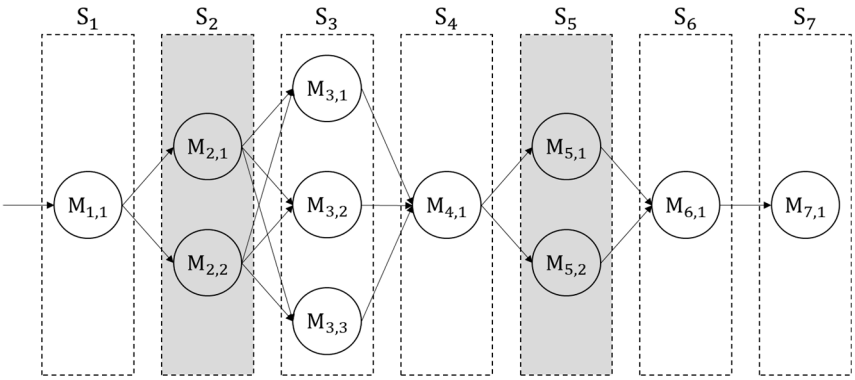

Figure 4. Layout of a disk-brake processing line in the numerical example.

As seen in this figure, S_2, S_3, and S_5 are composed of parallel machines, and the others include a single machine. We assumed that when a machine is abnormal the products processed by it are all defective.

Table 1 shows machine specifications, including probability density functions, of processing times and sensor values attached to each machine in normal and abnormal states. $\mathcal{N}\left(\mu,\ \sigma^2\right)$ denotes a normal distribution with mean μ and standard deviation σ, and $\mathrm{Exp}(\lambda)$ denotes an exponential distribution with mean $1/\lambda$. The numbers were all created artificially for the purpose of illustration.

Table 1. Machine specifications of the example.

Machine	Processing Time	Probability That the Machine Is Abnormal	Sensor Values	
			At Normal State	At Abnormal State
$M_{1,1}$	$\mathrm{Exp}(0.1)$	0.001	$\mathcal{N}\left(10,\ 3^2\right)$	$\mathcal{N}\left(15,\ 4^2\right)$
$M_{2,1}$ $M_{2,2}$	$\mathrm{Exp}(0.05)$	0.005	$\mathcal{N}\left(8,\ 3^2\right)$	$\mathcal{N}\left(12,\ 3^2\right)$
$M_{3,1}$ $M_{3,2}$$M_{3,3}$	$\mathrm{Exp}(0.1)$	0.003	$\mathcal{N}\left(9,\ 3^2\right)$	$\mathcal{N}\left(7,\ 2^2\right)$
$M_{4,1}$	$\mathrm{Exp}(0.2)$	0.010	$\mathcal{N}\left(10,\ 3^2\right)$	$\mathcal{N}\left(12,\ 3^2\right)$
$M_{5,1}$ $M_{5,2}$	$\mathrm{Exp}(0.15)$	0.001	$\mathcal{N}\left(10,\ 3^2\right)$	$\mathcal{N}\left(13,\ 3^2\right)$
$M_{6,1}$	$\mathrm{Exp}(0.1)$	0.001	$\mathcal{N}\left(12,\ 3^2\right)$	$\mathcal{N}\left(15,\ 4^2\right)$
$M_{7,1}$	$\mathrm{Exp}(0.1)$	0.001	$\mathcal{N}\left(12,\ 3^2\right)$	$\mathcal{N}\left(15,\ 4^2\right)$

4.1. Production-Line Analysis

One million simulated products were generated using Algorithm 1 to estimate partially shuffling probability $Q_{j,r}$. The result is summarized in Table 2.

Table 2. Partially shuffling probability matrix of the numerical example.

		Shuffle Range *r*								
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Work-Station *j*	1	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	2	0.6212	0.2272	0.0901	0.0360	0.0151	0.0063	0.0025	0.0010	0.0004
	3	0.7254	0.1808	0.0618	0.0207	0.0072	0.0026	0.0010	0.0003	0.0001
	4	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	5	0.8504	0.1204	0.0229	0.0050	0.0010	0.0003	0.0000	0.0000	0.0000
	6	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	7	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

With Table 2 and Equations (6) and (7), we can obtain a probability distribution of $\boldsymbol{z}^{(i)}$. For example, if we calculate the probability that the $i = 78$th product arrives at Workstation (1), and overtakes the 77th product at Workstation (2), and is overtaken by two products at Workstation (3) and keeps its order afterward, then it is given by as:

overtaken by two products at Workstation (3) and keeps its order afterward. The probability is then given by:

$$\begin{aligned}\Pr\left(\boldsymbol{z}^{(i)} = (i, i-1, i+1, i+1,\ i+1,\ i+1,\ i+1)\right) &= Q_{1,0} \times \frac{Q_{2,1}}{2} \times \frac{Q_{3,2}}{2} \times Q_{4,0} \times Q_{5,0} \times Q_{6,0} \times \\ Q_{7,0} &= 1.000 \times \frac{0.2272}{2} \times \frac{0.0901}{2} \times 0.7254 \times 1.000 \times 0.8504 \times 1.000 \times 1.000 = 0.00315699526\end{aligned} \tag{8}$$

4.2. Feature Data

A total of 100,000 values of $\boldsymbol{x}^{(i)},\ y^{(i)}$ were generated for all j (i.e., $\hat{n} = 100,000$), and they were artificially shuffled to get the observed feature values, $\hat{\boldsymbol{x}}_j^{(i)}$. Product i could then be expressed as a concatenation of the machine sequence, start and end times, machine state, feature vector, and label as follows:

$$\left(M_1^{(i)}, \cdots, M_7^{(i)},\ a_1^{(i)},\ \cdots, a_7^{(i)},\ c_1^{(i)},\ c_2^{(i)}, \cdots c_7^{(i)}, s_1^{(i)},\ s_2^{(i)},\ \cdots, s_7^{(i)},\ \boldsymbol{x}_1^{(i)},\ \boldsymbol{x}_2^{(i)},\ \cdots, \boldsymbol{x}_7^{(i)},\ y^{(i)}\right) \tag{9}$$

where $\boldsymbol{x}_j^{(i)}$ represents the feature of statistical values, including variance, skewness, mean, kurtosis, and median extracted from sensor values of the machine processing the product at workstation j [10], and $s_j^{(i)}$ indicates machine state (i.e., 0: normal or 1: abnormal). $y^{(i)}$ is the label (i.e., whether product i is defective or not), calculated as $\max_j\left(s_j^{(i)}\right)$. The number of defective products was 2,136; that is, the defective ratio during that period was 2.136%.

Among the 100,000 values of $\boldsymbol{x}^{(i)},\ y^{(i)}$, 70,000 randomly chosen values were used to train the proposed graphical model, as well as the one that ignores shuffling of product order. The remaining 30,000 values were reserved for the test of the two models.

4.3. Defect Diagnosis Result

A logistic regression model was employed as a classifier because it is one of the simplest probabilistic classifiers. Other probabilistic classification models may yield similar results. We compared the classification performance between our model and the model that classified the product without consideration of possible shuffling. The proposed model considered the probability that the feature values were shuffled among the products when classifying the product, but the other, "no shuffling," model assumed the order of collection of feature values was the same as that of the production. The preparation procedure for the dataset to be used for comparison is depicted in Figure 5.

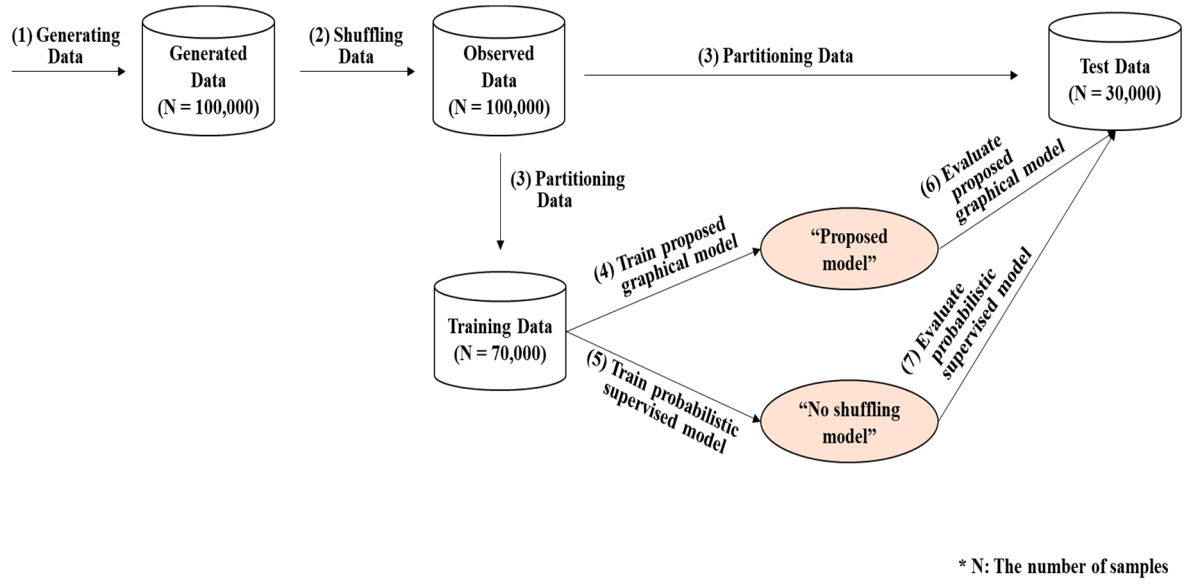

Figure 5. Preparation procedure for the dataset to be used for comparison.

Figure 6 shows the F1 scores of the proposed model and no shuffling model according to various cutoff values of threshold δ. The model decided whether a product was defective if $\Pr\left(y_i = 1|\hat{x}^{(i)}\right) \geq \delta$.

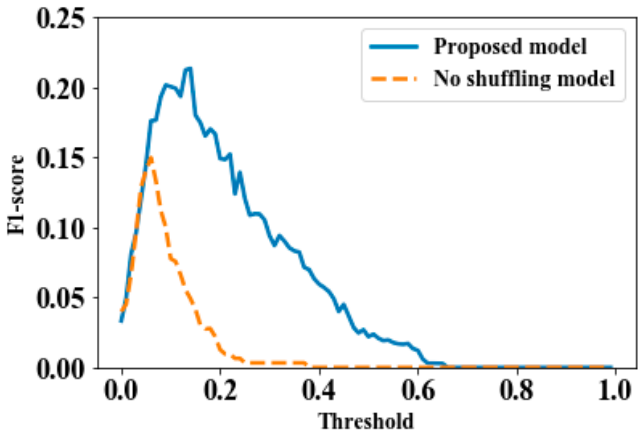

Figure 6. Defect diagnosis result: F1 scores of the proposed model and no shuffling model.

From Figure 6, we can see that the proposed model produced a much higher F1 score over all thresholds. This implies that our graphical model dealt with the shuffling of product order and could detect a defective product far more effectively than a model that ignores shuffling. As a result, one can expect an improved classification result by considering the shuffling when some workstations are unobservable from the outside and the shuffling of product order is expected. When the threshold was small, i.e., under $\delta = 0.1$, then the F1 scores of the two models were almost identical, which was expected because both models tried to judge almost all products as defective. Another implication is that the best performance of the "no shuffling" model appeared when the cutoff threshold was near $\delta = 0.08$, while the proposed model's performance was the best when $\delta = 0.17$. Therefore, the proposed model was less likely to produce a false alarm (classifying the non-defective as falsely defective) than the "no shuffling" model.

Figure 7 shows values of the F1 score, recall, precision and specificity of the proposed model for the range of $0.1 \leq \delta \leq 0.5$. As seen in this figure, the specificity was close to 1 irrespective of δ, implying that most non-defective products were diagnosed as non-defective (i.e., negative), which was natural because the number of negative records was much greater than the number of positive records.

Precision increased as δ got higher, while recall decreased. The F1 score was more affected by recall than precision because of very small values of recall. From this analysis, the following implications are drawn: First, specificity may not be a proper measure for defective diagnosis when classes are imbalanced. Second, precision and recall show the opposite trend as δ increases, implying that using only one of them may lead to wrong decision. Specifically, a classifier that determines most records as positive will get higher recall, and the precision of a classifier that regards most records as negative tends to be bigger. Since the F1 score of a highly imbalanced class dataset tends to show a similar trend to recall, a classifier which is not biased to negative class such as a naïve Bayes classifier with uniform prior can be suggested to reduce both misclassification costs.

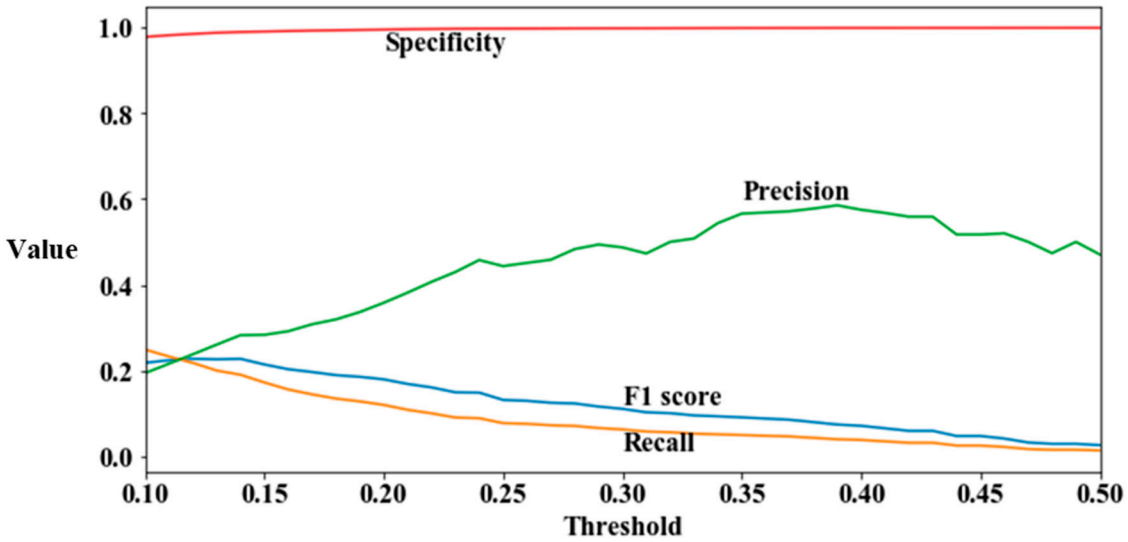

Figure 7. Performance measures of the proposed model according to threshold.

5. Conclusions

This paper addresses a realistic data quality problem of manufacturing equipment data when developing a product-defect diagnosis model—the partially shuffled data problem. Each record should correspond to a product in order to develop a proper defect diagnosis model, but most records in partially shuffled data do not correspond with a product. This problem results from one or more unobservable workstations that consist of parallel machines. One alternative may be to omit those features that may be partially shuffled if those workstations that produce partially shuffled data are near the end of the manufacturing process. However, if one or more of these workstations is near the front of the process, such as in the numerical example of this paper, omitting features obviously yields a very poor classifier with insufficient feature vector. This is because all the features collected from the workstations followed by that shuffling workstation may be possibly shuffled, and, therefore, we have to remove all features from then on. Thus, omitting features from the workstations cannot recover the shuffle of other features.

Therefore, this paper presents a probabilistic graphical model to diagnose product defects by considering the probabilities that product orders are partially shuffled. The graphical model was designed by means of a production-line analysis, and it was trained by partially shuffled data. From the numerical example, we verified that the graphical model exhibited a similar performance to a classifier trained by data that were not shuffled.

For future research, we will expand the graphical model to diagnose defects in products where the manufacturing sequence of some products varies due to machine breakdown, reworking, and other factors. In addition, we can apply this model to other fields such as traffic management and bio-informatics. Additionally, in order to enhance the model's diagnosis accuracy as well as to improve

its computational efficiency, it will be necessary to develop a proper method for extracting critical features with a shuffling strategy.

Author Contributions: Author contributions are as follows: data curation, D.S.; funding acquisition, S.H.; investigation, G.A. and D.S.; methodology, G.A. and S.H.; project administration, S.H. and D.S.; software, G.A.; validation, G.A.; writing—original draft, G.A.; writing—review and editing, S.H., D.S., and Y.-J.P.

Funding: This work has supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2019R1A2C1088255).

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

Nomenclature

Indices

i	Product index, $i = 1, 2, \cdots, n$
j	Workstation index, $j = 1, 2, \cdots, m$
k	Machine index, $k = 1, 2, \cdots, m_j$
t	Time index, $t = 1, 2, \cdots, T$
p	Simulated product index, $p = 1, 2, \cdots, \hat{n}$
r	Shuffle range index, $r = 0, 1, 2, \cdots, R$

Production Line-Related Terms

W_j	Workstation j
$\mathrm{M}_{j,k}$	Machine k at workstation j
$\mathrm{F}_{j,k}(\cdot)$	Probability distribution function of $M_{j,k}$'s processing time
$Q_{j,\,r}$	Probability that processing orders of two products, where r products are released between their release time at workstation $j-1$, are switched at workstation j
$M_j^{(i)}$	Machine that processes product i at workstation j
$a_j^{(i)}$	Start time of processing of product i at $M_j^{(i)}$
$b_j^{(i)}$	Processing time of product i by $M_j^{(i)}$
$c_j^{(i)}$	Completion time of processing of product i by $M_j^{(i)}$, $c_j^{(i)} = a_j^{(i)} + b_j^{(i)}$
$\widetilde{a}_j^{(i)}$	Arrival order of product i at workstation j
$\widetilde{c}_j^{(i)}$	Completion order of product i at workstation j
$I_{j,k}^{(t)}$	Binary variable indicating $\mathrm{M}_{j,k}$ is idle at t, $I_{j,k}^{(t)} = \begin{cases} 1, \text{if } \mathrm{M}_{j,k} \text{ is idle at } t \\ 0, \text{ otherwise} \end{cases}$

Model-Related Terms

$\boldsymbol{x}^{(i)}$	Feature vector for product i, $\boldsymbol{x}^{(i)} = \left(x_j^{(i)}\right)_{j=1}^{m} = \left(x_1^{(i)}, x_2^{(i)}, \cdots, x_m^{(i)}\right)$
$\boldsymbol{h}_{j,\,k}$	Time-series data collected from machine k at workstation j, $\boldsymbol{h}_{j,\,k} = \left(h_{j,\,k}^{(1)}, h_{j,\,k}^{(2)}, \cdots, h_{j,\,k}^{(T)}\right)$
$y^{(i)}$	Label of product i, $y^{(i)} = \begin{cases} 1, \text{ if product } i \text{ is defective} \\ 0, \text{ otherwise} \end{cases}$
$\hat{\boldsymbol{x}}^{(i)}$	Observed feature vector for product i
$f(\cdot)$	Classifier to diagnose whether each product is defective or not
$\boldsymbol{z}^{(i)}$	Latent variable of actual product whose data is collected as $\hat{\boldsymbol{x}}^{(i)}$

References

1. Manco, G.; Ritacco, E.; Rullo, P.; Gallucci, L.; Astill, W.; Kimber, D.; Antonelli, M. Fault detection and explanation through big data analysis on sensor streams. *Expert Syst. Appl.* **2017**, *87*, 141–156. [CrossRef]
2. Khaleghi, B.; Khamis, A.; Karray, F.O.; Razavi, S.N. Multisensor data fusion: A review of the state-of-the-art. *Inf. Fusion* **2013**, *14*, 28–44. [CrossRef]
3. Song, B.; Shi, H. Fault detection and classification using quality-supervised double-layer method. *IEEE Trans. Ind. Electron.* **2018**, *65*, 8163–8172. [CrossRef]
4. Fakhfakh, O.; Toguyeni, A.; Korbaa, O. On-line fault diagnosis of FMS based on flows analysis. *J. Intell. Manuf.* **2018**, *29*, 1891–1904. [CrossRef]

5. Ma, Z.; Marchette, D.J.; Priebe, C.E. Fusion and inference from multiple data sources in a commensurate space. *Stat. Anal. Data Min.* **2012**, *5*, 187–193. [CrossRef]
6. Hall, D.L.; Llinas, J. An introduction to multisensor data fusion. *Proc. IEEE* **1997**, *85*, 6–23. [CrossRef]
7. Chen, Z.; Li, W. Multisensor feature fusion for bearing fault diagnosis using sparse autoencoder and deep belief network. *IEEE Trans. Instrum. Meas.* **2017**, *66*, 1693–1702. [CrossRef]
8. Vazifeh, M.R.; Hao, P.; Abbasi, F. Fault diagnosis based on multikernel classification and information fusion decision. *Comput. Technol. Appl.* **2013**, *4*, 404–409.
9. Khalastchi, E.; Kalech, M.; Rokach, L. A hybrid approach for improving unsupervised fault detection for robotic systems. *Expert Syst. Appl.* **2017**, *81*, 372–383. [CrossRef]
10. Safizadeh, M.S.; Latifi, S.K. Using multi-sensor data fusion for vibration fault diagnosis of rolling element bearings by accelerometer and load cell. *Inf. Fusion* **2014**, *18*, 1–8. [CrossRef]