Distributed Data Filtering and Data-driven Modeling for Smart Manufacturing Networks

Yifu Li

Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
In
Industrial and Systems Engineering

Ran Jin, Chair
Kimberly P. Ellis
Doonyoon Lee
Subhash C. Sarin

July 22, 2020
Blacksburg, VA

Keywords: Data Filtering, Multi-task Learning, Similar-but-non-identical Processes, Smart manufacturing

Distributed Data Filtering and Data-driven Modeling for Smart Manufacturing Networks

Yifu Li

ABSTRACT

A smart manufacturing network connects machines via sensing, communication, and actuation networks. The data generated from the networks are used in data-driven modeling and decision-making to improve quality, productivity, and flexibility while reducing the cost. This dissertation focuses on improving the data-driven modeling of the quality-process relationship in smart manufacturing networks. The quality-process variable relationships are important to understand for guiding the quality improvement by optimizing the process variables. However, several challenges emerge. First, the big data sets generated from the manufacturing network may be information-poor for modeling, which may lead to high data transmission and computational loads and redundant data storage. Second, the data generated from connected machines often contain inexplicit similarities due to similar product designs and manufacturing processes. Modeling such inexplicit similarities remains challenging. Third, it is unclear how to select representative data sets for modeling in a manufacturing network setting, considering inexplicit similarities. In this dissertation, a data filtering method is proposed to select a relatively small and informative data subset. Multi-task learning is combined with latent variable decomposition to model multiple connected manufacturing processes that are similar-but-non-identical. A data filtering and modeling framework is also proposed to filter the manufacturing data for manufacturing network modeling adaptively. The proposed methodologies have been validated through simulation and the applications to real manufacturing case studies.

Distributed Data Filtering and Data-driven Modeling for Smart Manufacturing Networks

Yifu Li

GENERAL AUDIENCE ABSTRACT

The advancement of the Internet-of-Things (IoT) integrates manufacturing processes and equipment into a network. Practitioners analyze and apply the data generated from the network to model the manufacturing network to improve product quality. The data quality directly affects the modeling performance and decision effectiveness. However, the data quality is not well controlled in a manufacturing network setting. In this dissertation, we propose a data quality assurance method, referred to as data filtering. The proposed method selects a data subset from raw data collected from the manufacturing network. The proposed method reduces the complexity of modeling while supporting decision effectiveness. To model the data from multiple similar-but-non-identical manufacturing processes, we propose a latent variable decomposition-based multi-task learning model to study the relationships between the process variables and product quality variable. Lastly, to adaptively determine the appropriate data subset for modeling each process in the manufacturing network, we further proposed an integrated data filtering and modeling framework. The proposed integrated framework improved the modeling performance of data generated by babycare manufacturing and semiconductor manufacturing.

Acknowledgment

I have to first express my acknowledgment to Professor Ran Jin, my instructor, advisor, and collaborator for his supervision, support, and encouragement since the junior year of my undergraduate study. He is the one who brought me into the world of scientific research, and without his guidance, I could not complete my Ph.D. study and start my academic career. I would also like to thank my committee members, Dr. Kimberly P. Ellis, Dr. Doonyoon Lee, Dr. Subhash Sarin, for their great support and insights during my Ph.D. study.

I would like to thank the great support from members in the academic society of Manufacturing, Quality and Statistics and beyond, including Professor Jianjun Shi, Professor Chuck Zhang, and Dr. Kan Wang from Georgia Tech, Professor Yuan Luo from Northwestern University, and Professor Hongyue Sun from University of Buffalo.

I would also like to thank my fellow students, colleagues and friends for their support during my Ph.D. study, including but not limited to: Mr. Xiaoyu Chen, Mr. Jooneun Choi, Mr. Abhishek Kar, Mr. Qing Lan, Ms. Jingran Li, Mr. Chen'ang Liu, Mr. Jia Liu, Mr. Shuai Luo, Ms. Karuniya Mohan, Mr. Gongzhuang Peng, Ms. Zhangying Ren, Ms. Wenmeng Tian, Ms. Anqi Wang, and Mr. Lening Wang. Also, I would like to thank the National Science Foundation (NSF) and all my industrial sponsors for the support and great collaboration in my research.

Finally, I want to thank my parents and other family members for the support and courage, which helped me to come across the difficulties throughout my career and life.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

The Internet-of-Things (IoT) connects manufacturing processes and equipment into a network. Such an advanced manufacturing network creates a data-rich sensing environment that provides the capability to predict and monitor multiple similar-but-non-identical manufacturing processes. The "similar-but-non-identical" nature studied in this dissertation is reflected as similarities of process models in terms of model structures and parameters (i.e., parameters in the quality process models). Such similarities are often from the similarities of the manufacturing process, equipment, configuration, recipe, machine status, product design, and performance measure. The data from such similar-but-non-identical processes have been widely used to improve manufacturing operations, including production control (Yue et al. 2018) and quality assurance (Du, Zhang, and Shi 2018). However, the data collected from heterogeneous manufacturing processes are usually challenging for data-driven decision making (Sarin et al. 2016). Among these data-driven decision-making problems, there are several challenges which have prohibited various applications of data-driven decision making in manufacturing networks: 1) how does one ensure the data quality for data-driven modeling; 2) how does one model similar-but-non-identical manufacturing processes considering their inexplicit similarity. Here, the data quality is defined as the information richness, which is measured by entropy (Gray 2011) and the performance of the data analysis results given specific goals (e.g., R^2 of a regression modeling for a manufacturing process); 3) how to integrate data filtering and modeling to ensure that the data subsets filtered can meet the requirement for modeling different processes in a manufacturing network.

For data quality assurance, a conforming manufacturing process often produces data with little variation within certain time windows. If all raw data are directly used, it will not only introduce a lot of noise in computation but also pose high communication and computation

workload in the manufacturing network. There is a lack of methods to ensure high-quality sensing data with minimal redundancy. For example, the babycare manufacturing process generates more than three million sensor data samples from only one production line over a month. However, the readings of the sensor data have small variations, because these sensors are repeatedly measuring the conforming manufacturing process during most of the time and generate similar readings. The research question becomes how to generate a small but information-rich subset filtered from the massive but redundant raw data sets in data-driven modeling.

When modeling similar-but-non-identical manufacturing processes of the manufacturing network, data collected from a single process usually only carry a limited amount of information. One can greatly improve the generality and prediction accuracy of the models when considering their similarities (Wang et al. 2019). For example, in a silicon ingot growth manufacturing, each silicon ingot grows from a furnace using the same Czochralski (CZ) process (Fisher, Seacrist, and Standley 2012) with similar recipes. Multiple furnaces are typically connected to a manufacturing network to increase throughput. Different furnaces are subject to different degradation conditions and maintenance, which lead to a similar variable relationship in model structure or parameters (Jin et al. 2019). The heterogeneities among furnaces pose challenges to model such a manufacturing network for prediction and variation analysis of multivariate or profile responses. The research question becomes how to model the impact of the observable predictors on the quality response by recovering the similarity and the heterogeneity among the tasks.

When modeling multiple manufacturing processes in a network, each equipped with multiple sensors jointly collecting a large amount of *in situ* production data (Fisher, Seacrist, and Standley 2012), it is computationally challenging to support efficient on-time decision-making. Furthermore, as furnaces are producing the same type of products given the same recipe, the data

collected are similar-but-non-identical for different furnaces and significant data/information redundancy forms. Obtaining all data for analysis is often unnecessary (Wang, Yang, and Stufken 2019). The research question becomes how to ensure reasonable computational intensity when modeling multiple similar-but-non-identical processes in a manufacturing network.

1.1 State-of-the-Art

To generate a small but information-rich subset, the probability-based filtering for sample size reduction is widely used. It pre-defines the probability of preserving each observation. Two popular probability-based filtering methods are random sampling (Liu, Sadygov, and Yates 2004) and stratified sampling (Liberty, Lang, and Shmakov 2016). The randomly sampling is to apply uniform distribution on data to randomly select observations with equal probability. Stratified sample, using a time-based approach is to preserve a data observation for every few time steps, so that a representative observation is preserved within each window. However, the probability-based data filtering methods can overlook the inherent group structure of data and preserve excessive data from large clusters, while ignoring small data clusters. More importantly, there is only the *ad hoc* methods to determine the filtering ratio, which is the percentage of data preserved after filtering. This limits the effectiveness of filtering at preserving the data information and reducing the data size. The other research direction on data reduction is through data compression, and such research focuses on compressing the original data into formats, which can be reconstructed/recovered later on. Per Kaur, Sethi, and Singh (2015), the most important research directions under data compression include but not limited to Lempel-Ziv-Welch (LZW) Algorithm (Farach and Thorup 1998), Shannon-Fano coding (Shanmugasundaram and Lourdusamy 2011), bit reduction algorithm (Brar and Singh 2013), Huffman coding (Porwal et al. 2013), etc. However, the major limitation of applying the data compression methods is that the need for data

reconstruction significantly limits the minimum filtering ratio to be adopted while the reconstruction is not always needed for smart manufacturing network modeling.

When modeling similar-but-non-identical processes, there are two popular strategies. The first strategy is to model all processes with one universal model by assuming that all processes are the same. One could use linear regression and its extension on incorporating regularizers, such as Ridge Regression (Hoerl and Kennard 1970), LASSO (Tibshirani 1996), Elastic Net (Zou and Hastie 2005). However, these approaches fail to consider the heterogeneities of processes for modeling, and their performance will suffer. The other strategy, considered to be more suitable for modeling similar-but-non-identical processes, applies the same structure of the models but with different model parameters for different processes. Classic examples of such models are transferred learning (Pan and Yang 2009) and multi-task learning (MTL) (Kang, Grauman, and Sha 2011). Specifically, transfer learning transforms a source model, which is assumed to be accurate, to the target process by considering the process similarities (e.g., applying low-rank regularizer on model parameters). However, it can be difficult to find an accurate source model to be transferred to other processes, especially when all processes do not have an adequate amount of training samples.

1.2 Objectives

The objective of this dissertation is to develop a systematic data filtering and modeling framework to ensure the data quality and improve modeling performance in a manufacturing network. The proposed framework can effectively improve the data quality and capture the process heterogeneity for smart manufacturing network modeling. The objective of this research include:

1) investigating a methodology to reduce redundant data while ensuring the quality of the data sets for data analytics, such as data-driven modeling for variation analysis.

2) investigating a model to capture the process similarities and utilize the similar-but-non-identical data from multiple processes in a manufacturing network for quality modeling.

3) investigating a joint data filtering and modeling framework for quality modeling for multiple similar-but-non-identical processes in a network and build a fog-manufacturing testbed with the proposed method embedded.

This research is important because the proposed framework can effectively model similar-but-non-identical processes with reduced data redundancy in an information-rich environment. The data filtered from the massive raw data sets can effectively support the downstream modeling task and reduce the time latency in computation without significantly sacrificing the performance of the analysis.

This dissertation is important because:

- The data quality determines data analysis performance, such as modeling prediction accuracy. When there are data redundancies in the raw manufacturing data, an effective filtering method does not only support data-driven decision-making, such as monitoring, prognosis, and diagnosis but also greatly reduces the computational complexity.
- Recovering the similarity of the manufacturing processes will significantly improve the modeling performance, such as prediction accuracy, generality on future data, when there is only a limited amount of data collected from each process.
- Adaptively filtering and modeling similar-but-non-identical manufacturing processes in a network significantly reduces the communication and computation cost. Furthermore, building a Fog-manufacturing testbed based on the proposed method paves a promising way of adopting the method for IoT in manufacturing networks.

1.3 Organization of the Dissertation

The remainder of this dissertation is organized as follows. Chapter 2 propose to reduce redundant data while ensuring the quality of the reduced data for the continuous babycare manufacturing process. Chapter 3 proposed a multi-task learning (MTL) framework to capture the process similarities and utilize the similar-but-non-identical data for an ingot growth manufacturing process. Chapter 4 proposed a distributed data filtering and modeling framework to model similar-but-non-identical processes for a silicon ingot manufacturing network with a reduced computational and communicational load.

Chapter 2. Cluster-based Data Filtering for Manufacturing Big Data Processes

2.1 Introduction

With the advancement of sensor and communication technologies, Industrial Internet-based sensing processes can collect data in high frequency over a long period of time from various manufacturing environments and machine settings. While such a sensing process is capable of collecting a massive amount of data, the data may contain redundant information, which significantly limits the quality and efficiency of data analysis (Kenett and Shmueli 2014, Chen and Jin 2018). For example, a manufacturing process may generate a lot of sensor data from conformance manufacturing status, where the readings of the sensors have small variation through process modeling or control analysis, and little information can be extracted. As a result, selecting a meaningful or representative subset to reduce the overall size of the data while ensuring the quality of the subset is important to improve the efficiency of data analysis. Here the data quality is strongly related to the information richness, which can be measured by entropy (Gray 2011) and the performance of the resulting data analysis (e.g., R^2 of a trained model). It is expected that a small but information-rich subset filtered from the massive raw data sets can effectively support various data analysis tasks and reduce the time latency in computation without significantly sacrificing the performance of the analysis.

The objective of this work is to reduce the sample size of manufacturing big data while maintaining the amount of useful information for efficient data analysis. This work is motivated by a continuous babycare manufacturing process, where the data were continuously collected as multi-dimensional time-series data. The cloud-storage is typically expensive for data from such a process as there will be more than three million data points collected from a single production line over one month. Under this motivation, we focus on proposing a method to reduce the information

loss (i.e., entropy loss) when data is filtered/sampled for storage. After preliminary investigation, we found out that within a time window (e.g., several hours or days), the manufacturing process produces conforming products and the data collected may contain redundant information, e.g., constant values with little information for manufacturing analysis. Big data with redundancy pose challenges to efficient data storage and timely data analysis in a continuous manufacturing process. We believe that a properly filtered data set can preserve the majority of the original data set's useful information leading to computational saving and improved data analysis performance.

In the data reduction literature (Van Leeuwen 2007, Baraniuk 2007), reducing the size of data while preserving the information of data was a major focus of study. Among the most widely adopted approaches are probability-based filtering, which means that the probability of preserving an observation is pre-defined for filtering. Two popular probability-based filtering methods that are still widely adopted are random sampling (Clarkson and Shor 1989a, Liu, Sadygov, and Yates 2004) and stratified sampling (Trost 1986, Liberty, Lang, and Shmakov 2016). However, probability-based data filtering methods often overlook the inherent group structure or clustering patterns among data and may easily preserve excessive data from large clusters, while ignoring data from small but important clusters.

In manufacturing, similar processes and equipment can produce clusters of data having similar amounts of information in terms of statistical moments, like mean and variance (Yamaoka, Nakagawa, and Uno 1978). Filtering guided by such clustering patterns can effectively reduce the information redundancy by removing a large number of observations with similar information. A natural outcome is to incorporate data clustering into filtering (Thompson 1990), which encourages the selection of neighbourhood observations around the previously identified data clusters. Alternatively, Singh and Masuku (2014) first cluster the raw data, then select a number of large

data clusters to be preserved. However, one problem in the existing cluster-based filtering strategies is that they require clustering algorithms to be performed on the full data set and the size of data can easily become too large for clustering algorithms. Furthermore, as the size of data is reduced through filtering, the information preserved will inevitably degrade and the resulting data analysis performance becomes worse. In addition, existing methods do not optimize the selection of the filter ratio, by considering the information loss, to determine the proportion of data to be preserved.

We propose an unsupervised data filtering method along with a filtering information criterion (FIC) to automatically determine the proportion of data preserved in filtering. The proposed method aims to select representative subsets from raw data. Specifically, the unsupervised data filtering method includes two steps. The first step is to use certain index tags, such as time index tags, to segment the raw data into different segments (denoted as hubs) whenever there is a large gap between index tag values for two adjacent data observations. We assume that each hub has different characteristics compared with other hubs, as large index gaps usually indicate manufacturing events that impact the characteristics of *in situ* variables. An example of a large index gap can be caused by equipment shutdown, and the manufacturing process may run under different conditions after the equipment is restarted. The second step is to partition each hub into clusters, extract the centroid of each cluster, and perform cluster-wise random sampling. The proposed two-step method can recover the clustering pattern from raw data and help to better preserve the information by retaining the data from each cluster with the determined filtering ratio. Furthermore, the computational speed will be significantly accelerated by performing clustering within-hubs, instead of using the full data set.

To determine the best filtering ratio, we propose a filtering information criterion (FIC) to balance a trade-off between the information preserved and the size of the filtered data set. It is worth mentioning that the proposed sampling method does not rely on any data distribution assumption while deriving the analytical form of FIC relies upon data normality assumption. Furthermore, although we are not considering filtering for categorical or discrete variables, we suggest using dummy variable transformation on them before applying the proposed method. A babycare manufacturing case study is used to evaluate the filtering performance based on the optimal filtering ratio selected by FIC. We further conducted simulation studies to processatically evaluate the filtering method at multiple levels of filtering ratios. The numerical results from studies show the promising performance of the proposed filtering method, compared with the benchmark methods, such as random sampling and stratified sampling.

The rest of the paper is organized as follows. In Section 2.2, we introduce the proposed data filtering method and the FIC. In Section 2.3 , we perform a case study using a manufacturing data set to test the proposed filtering method with FIC. In Section 2.4 , we perform a simulation study inspired by a manufacturing data set for comparing the proposed data filtering method with other benchmark methods. In Section 2.5 , we provide a summary and discuss future work.

2.2 The Proposed Data Filtering Method

Denote the full manufacturing time-series data set as $X = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}'$ which contains n observations $\boldsymbol{x}_i = \left(x_{i1}, \dots, x_{ip}\right)', i = 1, \dots, n$. Here we assume that there are p variables of interest (e.g., signals from p sensors). Each observation $\boldsymbol{x}_i$ is often associated with its index tag, denoted as t_i. For example, the index tag can be the data collection timestamp (quantitative variable) or current machine operational status (qualitative variable). Note that although the size of data, n, is usually very large, the corresponding manufacturing process conforms to specification the

majority of time. $\hat{X}$ contains the raw data points or summary statistics (e.g., centroids of clusters for clustered data) from X, to preserve the information in a smaller size.

Here we formulate the data filtering,

$$\begin{aligned} &\min_{\hat{X}} L(\hat{X}, X) \\ &s.t. \frac{|\hat{X}|}{n} \leq r, \end{aligned} \tag{1}$$

where $L(\hat{X}, X)$ is a loss function, such as negative log-likelihood or entropy loss (Gray 2011), to quantify the information loss between the filtered data and the full data. The $|\cdot|$ is the cardinality of a data set, i.e., counting the sample size in the data set. Here $0 \leq r \leq 1$ is a tunable filter ratio, which represents the percentage of observations preserved after filtering.

To filter the raw data with data clustering patterns preserved, we propose to incorporate clustering into the loss function (1. For example, assume that X can be partitioned into k clusters, where $X = C_1 \cup ... \cup C_k$. The data between different clusters are heterogeneous in terms of means, variance, etc. A representative subset of such data, denoted as $X = \hat{C}_1 \cup ... \cup \hat{C}_k$, where $\hat{C}_i$ contains either the raw data or the summary statistic of C_i. Therefore, we incorporate the clustering into Equation (1) and propose cluster-based data filtering as

$$\begin{aligned} &\min_{\hat{X}, \boldsymbol{\mu}_i, C_i, \hat{C}_i} L(\hat{X}, X) + \lambda \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in C_i} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_i \right\|_2^2 \\ &s.t. \frac{\sum_{i=1}^{k} |\hat{C}_i|}{\sum_{i=1}^{k} |C_i|} \leq r, \end{aligned} \tag{2}$$

where $\boldsymbol{\mu}_i$ is the centroid of each data cluster and λ is the coefficient for the clustering term, which minimizes within-cluster distances. Here, we assume that the size of data extracted from each cluster is proportional to the sample size of the underlying cluster. Although the loss function (2) is well-defined, there are two major computational challenges when applying it to filtering. The first one is that the optimization problem involves solving data partitioning and sub-sampling, which is NP-hard (Burdakov, Kanzow, and Schwartz 2015). Furthermore, solving the multi-

objective problem presented in loss function (2) requiring the selection of λ, which can significantly increase computational cost if iterative parameter tuning procedures are adopted (e.g., 5-fold cross-validation).

To address these challenges, we propose a heuristic data filtering approach by combining index-based data partition and cluster-based data sampling to divide the data into clusters and extract subsets of data from clusters. In the first step, we adopt the index-based data partition, which is computationally fast, to partition the full data into hubs and pave an efficient way to enable cluster-wise data filtering. In the second step, within each hub, we perform clustering and randomly sample each cluster of data proportionally.

The index-based data partition consecutively partition the full data along the index tags into q data hubs as $X = H_1 \cup ... \cup H_q$. Here, H_j, where $j = 1, ..., q$, are consecutive and non-overlapping data subsets of X. The data partition (segmentation) technique has many variants, such as using a likelihood criterion (Guralnik and Srivastava 1999), minimum message length approximation (Fitzgibbon, Dowe, and Allison 2002), and landmark identification (Dikmen, Zhan, and Zhou 2012). Note that for manufacturing processes with index tags, it is not easy to assume certain probabilistic distribution properties for the index tag variable. Thus, segmentation based on likelihood and message length approximation are not applicable for our problem. Alternatively, we adopt the idea of finding landmarks (Perng et al. 2000) or perceptually important points (PIPs) (Zhang, Jiang, and Wang 2007) based on index tags for segmentation.

Note that the index tag in the manufacturing data often reflects the dynamics of a manufacturing process. A large gap between two consecutive index tags often reflects a change in the manufacturing process. For example, when data collection times are used as the index tags, a one-hour time gap between two consecutive observations may indicate that there is a product

change over. The aforementioned manufacturing events can significantly vary the *in situ* conditions of manufacturing processes and generate data in hubs with heterogeneous characteristics. Partitioning the data based on such gaps will save the need for executing a clustering algorithm over the full data set, which can be very time-consuming and even intractable. To incorporate the information on the gaps of index tags in the proposed method, we consider using the second-order tag gap at t_i, defined as $\delta_i = (t_{i+1} - t_i) - (t_i - t_{i-1})$.

When time tags of data collection are adopted as the index tags, the second-order tag gap can identify the large index tag gaps, either in the time domain (the first-order information) or the frequency domain (the second-order information) among index tags. We consider to form the segments by partitioning the raw data X at the following locations

$$\{j: \delta_j \geq q_{1-\alpha}(\delta_1, \dots, \delta_n)\}, \tag{3}$$

where $q_{1-\alpha}(\delta_1, \dots, \delta_n)$ is the $(1-\alpha)$ percentile of all $\delta_1, \dots, \delta_n$. It means that the gap values at these locations are larger than the $(1-\alpha)$ percentile of all δ_j's.

For each identified hub, we use clustering to further partition the data. That is, we would like to have k_j clusters for hub H_j as $H_j = C_1^{(j)} \cup \dots \cup C_{k_j}^{(j)}$. Here we adopt the k-means clustering method (MacQueen 1967) to form clusters $C_1^{(j)} \dots C_{k_j}^{(j)}$ for each hub. Specifically, the k-means clustering method minimizes the total within-cluster sum of squares for the clusters in a hub as

$$\min \sum_{s=1}^{k_j} \sum_{\boldsymbol{x}_j \in C_s^{(j)}} \left\| \boldsymbol{x}_j - \boldsymbol{\mu}_s^{(j)} \right\|_2^2, \tag{4}$$

where $\boldsymbol{\mu}_s^{(j)}$ is the centroid of the cluster s from the hub H_j, and $\|\cdot\|_2$ is the Euclidean norm. The number of clusters k_j for each hub is selected to maximize the average Silhouette distance of all data points in the hub (Rousseeuw 1987). Given any data point $\boldsymbol{x}_i$, the Silhouette distance calculates the difference between the average distance of $\boldsymbol{x}_i$ to the other data points in the same cluster and the average distance of $\boldsymbol{x}_i$ to the other data points in different clusters. Finally, we

generate samples $\hat{X}_s^{(j)}$ from each cluster $C_s^{(j)}$ to form the filtered data set $\hat{X} = \bigcup_{j=1}^{q} \bigcup_{s=1}^{k_j} \hat{X}_s^{(j)}$, where $\hat{X}_s^{(j)}$ consist of $100r\%$ data points randomly sampled from cluster $C_s^{(j)}$ and the cluster centroid $\boldsymbol{\mu}_s^{(j)}$. Here, the centroids are important summary statistics of clusters, and adding them to the filtered dataset is crucial for the proposed method to preserve representative information of clusters.

The filtering ratio r determines the degradation of data quality. For example, the entropy loss quantitatively measures the information loss between the raw and filtered data sets, which is defined as $EL = tr\left(\Sigma^{-1}\hat{\Sigma}\right) - \log\left(\det\left(\Sigma^{-1}\hat{\Sigma}\right)\right) - p$, where $tr(\cdot)$ is the trace operator, $det(\cdot)$ is the determinant operator, Σ is the sample covariance matrix of the raw data set, $\hat{\Sigma}$ is the sample covariance matrix of the filtered data set, and p is the number of variables in the data set.

Determining the filtering ratio r is not trivial. For example, in P&G babycare manufacturing data with a size of $n = 24915$ observations and $p = 32$ variables, the increase of loss due to the decrease of data size preserved is non-linear. In Figure 2-1, we compared the entropy loss acquired by random sampling under different filtering ratios. From Figure 2-1, we can see that the entropy loss remains stable when the filtering ratio is larger than $r = 0.2$, but increased dramatically when the filtering ratio was smaller than $r = 0.2$. As a result, $r = 0.2$ may be considered as a good choice of the filtering ratio, since it has a balanced small filtering ratio with a relatively small entropy loss. We will make use of this observation to test if our proposal can identify the optimal filtering ratio.

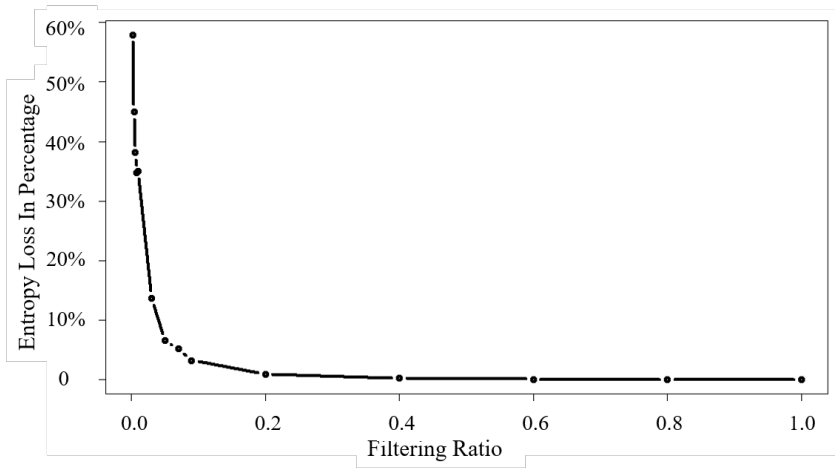


Figure 2-1.The increase of entropy loss as the filter ratio decreases.

In practice, the exact trade-off relationship between filter ratio and entropy loss among data sets may vary significantly. The filter ratio r can be determined by users based on their experience and judgment. However, there lacks statistical justification behind manual selection of the filter ratio for achieving a good balance between the information preserved in the filtered data $\hat{X}$ and the size of data preserved.

Here we provide a statistical guide on the selection of optimal value of r. Specifically, we propose a filtering information criterion, denoted as FIC, as the criterion to find an optimal value of r. The FIC is motivated and modified from Akaike information criterion (AIC), a statistical model selection method (Bozdogan 1987). Assuming that the data set X has n observations and follows the normal distribution $N(\boldsymbol{\mu}, \Sigma)$, with $\boldsymbol{\mu}$ as the mean and Σ as the covariance matrix, then the log-likelihood of the full data can be written as

$$L(X|\boldsymbol{\mu}, \Sigma) = -n\frac{1}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{\mu})'\Sigma^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}), \tag{5}$$

up to some constant and where

$$S = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})',$$

The original AIC is to consider the balance between the fit of the model to the data and the model complexity, which can be expressed as

$$AIC(\hat{\boldsymbol{\mu}}, \hat{\Sigma}) = -2L + 2k, \tag{6}$$

where L is the log likelihood function and k is the degree of freedom of the model.

We propose to use an AIC like measure to find the optimal ratio r. To do this, we estimate the mean and covariance matrix from the filtered data, which are the weighted sample mean and sample covariance matrix from the full data

$$\hat{\boldsymbol{\mu}} = \frac{1}{m}\sum_{\boldsymbol{x}_i \in \hat{X}} \boldsymbol{x}_i = \frac{1}{m}\sum_{i=1}^{n} w_i \boldsymbol{x}_i,$$

$$\hat{\Sigma} = \frac{1}{m}\sum_{\boldsymbol{x}_i \in \hat{X}} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})' = \frac{1}{m}\sum_{i=1}^{n} w_i (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})'$$

where $w_i \in \{0,1\}$ and $\sum_{i=1}^{n} w_i = m$, then a modified AIC, denoted as filtering information criterion (FIC), for evaluating the quality of the filtered data can be written as

$$\begin{aligned} FIC(\hat{\boldsymbol{\mu}}, \hat{\Sigma}) &= -l(X|\hat{\boldsymbol{\mu}}, \hat{\Sigma}) + 2|\hat{X}| \\ &= n log|\hat{\Sigma}| + \sum_{i=1}^{n} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})' \hat{\Sigma}^{-1} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}) + 2|\hat{\mathrm{X}}| \\ &= n\left[log|\hat{\Sigma}| + trace(\hat{\Sigma}^{-1} S^*) + 2\frac{|\hat{\mathrm{X}}|}{n}\right], \end{aligned} \tag{7}$$

where $S^* = \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}})'$

Given a pre-defined set $\boldsymbol{r} = \{r_1, \dots, r_m\}$, the optimal filter ratio r^* is chosen to have the smallest corresponding FIC value. In the case study and simulation that follows, we will evaluate the performance of the proposed filtering method and the filtering information criterion.

Algorithm 1. Pseudo code for the proposed filtering method

Step 1: Split raw data into hubs H_j, where $j = 1, \dots, q$, at second-order index tag gaps $\delta_j^* > P_h(\delta_i)$.

Step 2: Identify k_j clusters within each hub H_j, where $j = 1, \dots, q$, using k-means clustering and silhouette distance.

Step 3: Randomly sample from each cluster $C_s^{(j)}$ based on the optimal filter ratio r^*:

for each hub H_j, where $j = 1, \dots, q$, **do**

for each cluster $C_s^{(j)}$, where $s = 1, \dots, k_j$, **do**

Randomly sample $100r^*\%$ data points and the cluster centroid $\boldsymbol{\mu}_s^{(j)}$ from cluster $C_s^{(j)}$. Denote these observations collectively as $\hat{X}_s^{(j)}$.

Step 4: Combine the observations $\hat{X}_s^{(j)}$, $j = 1, \dots, \sum_{j=1}^{q} k_j$ and form the filtered data $\hat{X} = \bigcup_{j=1}^{q} \bigcup_{s=1}^{k_j} \hat{X}_s^{(j)}$.

We would remark that although the derivation of FIC starts with the normal assumption, one can use FIC for data from the exponential family distribution with proper mean and covariance matrix. Note that the formulation in Equation (7) can be viewed as an information discrepancy between $\hat{\Sigma}$ and S^*. Following this observation, it is possible to use FIC as a general guideline to select the optimal filtering ratio for data with proper mean and covariance matrix. The analysis of the case study provides further evidence on the use of FIC.

2.3 Case Study

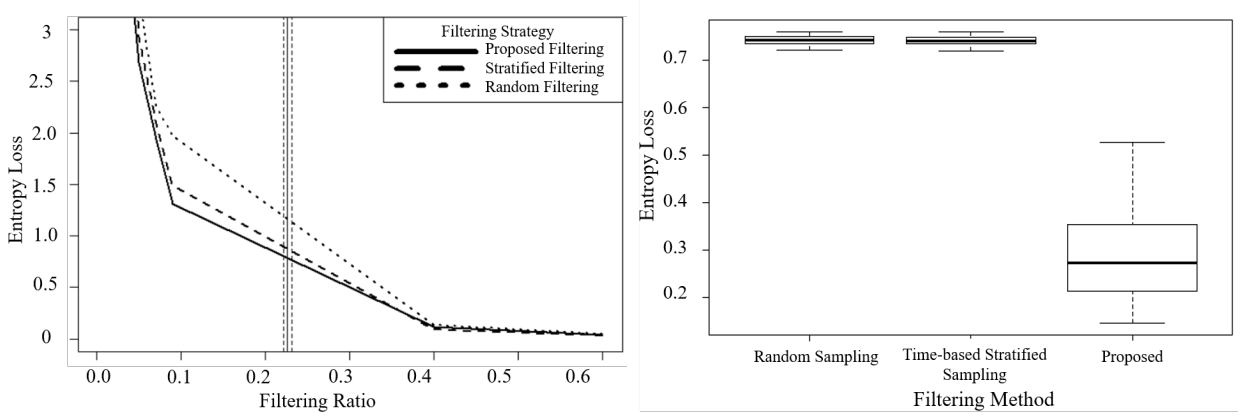
In collaboration with the P&G babycare manufacturing sector, we evaluate the performance of the proposed method. Data collected from the P&G production line has a size of $n = 24915$ observations and $p = 32$ variables, including the time tag. Although the physical meanings of variables are not disclosed here due to the data non-disclosure agreement, we summarized the summary statistics and the histogram of each centralized variable in the supplementary materials. Typically, a manufacturing big data process can have millions of observations for analysis. This selected data set is to provide a representative performance evaluation in a big data environment for two reasons. The first reason is that some engineering-driven partition can divide a big data set into sub-data with smaller sample sizes for filtering. The second reason is that production engineers often avoid waiting until the collected data for analysis becomes big sizes, which can cause a significant delay in decision-making.

The objective of this case study is two-fold. First, we demonstrate that an effective filtering ratio can be selected by FIC to achieve a good balance between the size of data preserved and the data quality. Second, we evaluate the performance of the proposed filtering method in comparison with several benchmark methods.

We used the entropy loss (Gray 2011) to evaluate the entropy loss of the filtered data sets compared with the raw data efficiently. Two benchmark methods denoted as BM_1 and BM_2 were included for comparison. The BM_1 was random sampling (Liu, Sadygov, and Yates 2004), which extracts observations randomly from the full data set. The second one was BM_2 (Liberty, Lang, and Shmakov 2016) which extracts the first observation within every equally spaced and non-overlapping time window, each of which contains sequential observations.

For example, given a filter ratio r as 0.1, we extracted the first observation of every 10 sequential observations in data. The threshold for picking the significant second-order index (time) gap was set at the 99.99 percentile of all the gaps in data. For example, given a filtering ratio r and n data points, we stratified the raw data into approximately nr consecutive and equally sized data segments and preserved the first observation of each segment.

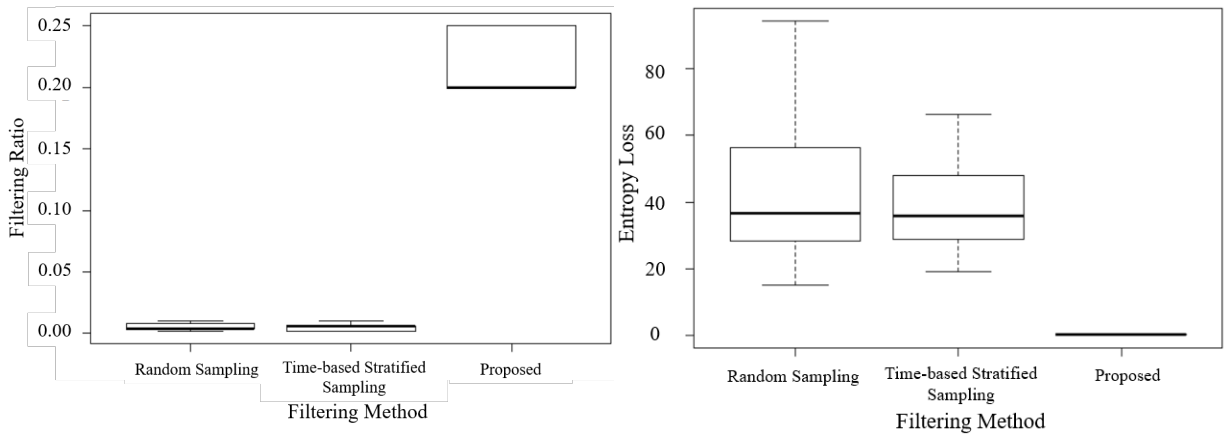
The threshold for forming hubs by segmentation in Equation (3) set $\alpha = 0.01$. To illustrate the impact of different α on gap identification, we compare the gaps above the (1- α) percentile with $\alpha = 0.02, 0.01, 0.001$ versus on the gaps identified in the natural log space in the Appendix A. To evaluate the performance of the proposed filtering method, 50 replications with 90% of randomly extracted data were performed on varying and fixed filtering ratios to ensure reproducibility. Here the time order of the randomly extracted data is preserved in each replication.



(a) (b)

Figure 2-2. (a) The mean value with standard errors (in vertical solid and dotted lines) of the optimal filter ratio determined over 50 replications versus mean entropy loss at different filter ratios for all methods over 50 replications. (b) The comparison in a boxplot of entropy loss for all methods using the fixed filtering ratio over 50 replications.

In Figure 2-2, we summarized the average entropy loss for each filtering method with varying (Figure 2-2 (a)) and fixed (Figure 2-2 (b)) filtering ratio. The mean value and the standard error of the optimal filter ratios selected by FIC for the proposed filtering method over 50 replications are drawn as solid and dashed vertical lines, respectively in Figure 2-2 (a). It can be seen from Figure 2-2 (a) that both performance measures did not improve significantly when the filter ratio is beyond 0.4 and dropped dramatically as the filter ratio went below 0.1. As a result, the desirable filter ratio, which balances the quality and the size of the filtered data, should be between 0.1 and 0.4. In Figure 2-2 (a), the average optimal filter ratio (the vertical dotted line) was 0.235 over 50 replications, which fit exactly into the desirable range (0.1-0.4) with very small standard errors. Furthermore, we observed that the cluster-based filtering method (the solid curve in Figure 2-2 (a) showed superior performance compared with the benchmark methods at the selected filter ratio. In this study, we also fixed the filtering ratio in each replication to be the optimal one determined by FIC using the cluster-based filtering method and compared all methods for 50 replications. As a summary, the boxplot in Figure 2-2 (b) shows that the cluster-based filtering method offers a significantly lower entropy loss than the benchmark methods did.



(a) (b)

Figure 2-3. (a) The boxplots of the optimal filter ratios determined over 50 replications. (b) The entropy losses obtained under the optimal filter ratio over 50 replications.

We also summarize the filtering performance for different methods when the filter ratios are automatically determined by FIC over 50 replications. The boxplots of the selected performance measures are shown in Figure 2-3. Figure 2-3 (a) shows the boxplot of the selected filter ratio over 50 replications. Figure 2-3 (b) shows the boxplot of the entropy loss obtained under the selected filter ratio over 50 replications. We can see that the proposed filtering method favored higher filter ratios, which were between 0.2-0.5, while the benchmark filtering methods favored much smaller filter ratios (0.01-0.05). As FIC jointly leverages the likelihood and the data size preserved, it is straightforward to see that the proposed filtering method can efficiently preserve the data likelihood when the filter ratio increases. On the other hand, a much smaller filter ratio is preferred by FIC for the benchmark filtering methods as they are less effective at preserving the likelihood where only decreasing the filter ratio will lead to a lower FIC score. Furthermore, although the cluster-based filtering method had a relatively higher filter ratio it offered much smaller entropy loss. We summarize that: (1). the filtering information criterion (FIC) can successfully help to identify the optimal filter ratio, balancing the trade-off between the selected

performance measures and the size of data preserved; (2). the cluster-based filtering method can more efficiently preserve the data likelihood compared to the benchmark filtering methods; and (3). the FIC will favor saving more data when the underlying filtering methods can better preserve the data likelihood.

Furthermore, to show that the proposed method can be easily applied to dataset with much larger size, we performed hub identification using $\alpha = 0.01$ on a dataset with $n = 24{,}915$ (the case study data) and a dataset with $n = 1{,}003{,}708$ data points from the P&G babycare manufacturing process. The result is that hub identification generated data hubs in a comparably average size of 199.902 and 196.1811 from two datasets. Such similar hub sizes indicate that the scale-up of the proposed method to new data sets to the same process can be straight-forward since hub identification partitions raw data in significantly different sizes into similar-sized hubs for further processing.

2.4 Simulation

Besides investigating the performance of the proposed filtering method based on the real manufacturing data set, we further evaluated its performance in a simulation study. In particular, we considered varying three simulation settings: the inherent number of data clusters, the signal-to-noise ratios, and the amount of model sparsity. To mimic the characteristics of the real manufacturing data set, the simulation data were generated as follows: for different settings on the inherent numbers of clusters, we respectively clustered the real data set into NC clusters using k-means clustering, each denoted as $\hat{X}_f$. Then we extracted the original time stamp for each observation in the cluster f as $\boldsymbol{t}_f$ and the means of all variables as $\boldsymbol{\mu}_f$, where $f = 1, \ldots, NC$ from real data. Then we simulated each data cluster with 1. $X_f \sim N(\boldsymbol{\mu}_f, \Sigma)$, 2. $X_f \sim N(\boldsymbol{\mu}_f, \Sigma, df = 10)$, 3. $X_f \sim Uniform(\boldsymbol{\mu}_f, \Sigma)$ in three different scenarios, X_f has same size as $\hat{X}_f$, and the regularized

covariance matrix $\Sigma = [\sigma_{ij}]$ with $\sigma_{ij} = 1$ when $i = j$, and $\sigma_{ij} = 0.3$ when $i \neq j$. To investigate the different impact of covariance structure, we can investigate the results from datasets following t-distribution with the same covariance matrix and Uniform distribution, which did not rely on covariance. Finally, we aggregated all simulated clusters X_f with the time stamps $\boldsymbol{t}_f$ to produce the simulated data matrix X.

We further created simulation models to evaluate the modeling performance. To generate the response variable $\boldsymbol{y}$ we used a linear model given by $\boldsymbol{y} = \tilde{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where $\boldsymbol{y} = (y_1, \dots, y_n)'$ was the response, $\tilde{X}$ was the simulated data matrix having all the main effect variables $(t, x_1, \dots, x_p)'$ in X and the two-factor interaction terms of the main effect variables in multiplication form $(x_t, x_1, \dots, x_{p-1}x_p)'$, $\boldsymbol{\beta} = (\beta_t, \beta_1, \dots, \beta_p, \beta_{t1}, \dots, \beta_{(p-1)p})'$ were the model parameters corresponding to main effect variables and the interaction terms, $\boldsymbol{\epsilon} = (\epsilon_1, \dots, \epsilon_n)'$ were the residual terms with $\epsilon_i = N(0, \sigma^2)$ as independent and identically distributed. To obtain the value of $\boldsymbol{\beta}$, we first trained a LASSO model using the real data set, and extracted the non-zero model coefficients as $\widehat{\boldsymbol{\beta}}$. Then, we randomly set $MS\%$ of main effect parameters $(\beta_t, \beta_1, \dots, \beta_p)'$ and 10% of interaction term parameters $(\beta_{t1}, \dots, \beta_{(p-1)p})'$ in $\boldsymbol{\beta}$ as significant (non-zeros), with values randomly chosen from $\widehat{\boldsymbol{\beta}}$. Furthermore, we varied the signal-to-noise ratio (STN) when generating the error term $\boldsymbol{\epsilon}$, where $STN = \frac{var(\tilde{X}\boldsymbol{\beta})}{var(\boldsymbol{\epsilon})}$, and $var(\cdot)$ represents the variance. In summary, two levels of the three factors are shown in Table 2-1.

Table 2-1. Simulation Design Table

Parameters	Low	High
Signal-to-Noise Ratio (STN)	3	10
Model Sparsity (MS)	0.3	0.7
Number of Clusters (NC)	30	3

[1]The same settings were applied on Normal, t, and Uniform distribution

We evaluated the performance of the proposed filtering method with 90% of data randomly extracted from the raw data over 50 replications to ensure reproducibility. Same as the case study, We normalized all the variables in this study. Instead of optimizing the filtering ratio, we evaluate the performance measures when different filtering ratios are used. Specifically, we selected three levels of the pre-determined filtering ratios to filter the training data in each replication. The α value for identifying the significant second-order index gap in Equation (3) is set to be $\alpha = 0.01$. Then the filtered data sets were used to estimate a linear model for the evaluation goodness-of-fit.

Five performance measures were used including entropy loss (James and Stein 1992), R^2, adjusted-R^2, and CPU time for the data filtering step and the modeling step. R^2 quantifies the goodness-of-fit for the model on the filtered training data (Cameron and Windmeijer 1996), while adjusted-R^2 (Gelman and Pardoe 2006) simultaneously evaluates the goodness-of-fit and the complexity of the model. Besides the previously adopted random sampling and stratified sampling methods, the third benchmark method was to directly use the full data set for modeling. Using the full data yielded zero entropy loss but not necessarily the best model performance. The simulation was performed on a workstation with CPU Xeon Processor E5-2687W, 3.10 GHz, 64 GB RAM.

For the simplicity of presentation, we only include the tables of the results with varying inherent number of clusters in the manuscript (Table 2-2 and Table 2-3), while presenting the rest of tables corresponding to the other scenarios in the supplementary materials. Both tables show

that the proposed method significantly outperformed the two benchmark methods on entropy loss as the filtering ratio decreased. Furthermore, the proposed method outperformed all methods in comparison on R^2 and adjusted-R^2. As the inherent number of clusters increased from 30 (Table 2-2) to 1000 (Table 2-3), each cluster contained less data and information. As a result, the clustering pattern is becoming less significant in data composition and the modeling performance is less affected by such a pattern. However, the proposed method always outperformed the benchmark filtering methods for the majority of performance measures in both scenarios. Although the proposed filtering method took a longer time at the filtering step, it costs less than 10 seconds on average and was deemed as a time-efficient method by production engineers. The rest of the results, corresponding to the Normal, t, and Uniform distribution, show that the proposed methods outperformed benchmark methods at reducing entropy loss and achieving better goodness-of-fit.

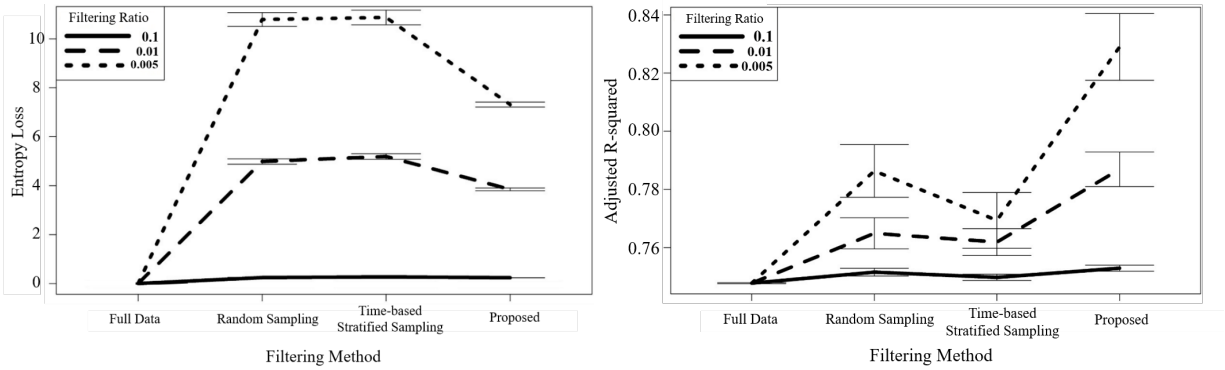
Additionally, we generated the Figure 2-4 based on the information from Table 2-2. It is seen that there was a significant reduction of entropy loss achieved by the proposed filtering method over the two benchmark filtering methods. The proposed filtering method outperformed the benchmark methods for the adjusted R^2 as well. Such improvements became more significant as the filtering ratio decreased.

Table 2-2. Mean and standard errors (within parenthesis) from 50 simulation runs for Normal distribution with $\mathrm{NC} = 30$, $\mathrm{MS} = 0.3$, and $\mathrm{STN} = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.748	0.748	–	39.664
		–	(0.000)	(0.000)	–	(0.220)
	BM_1	0.270	0.750	0.750	0.106	10.073
		(0.005)	(0.001)	(0.001)	(0.008)	(0.146)
	BM_2	0.247	0.752	0.752	0.123	10.292
		(0.004)	(0.001)	(0.001)	(0.009)	(0.223)
	Proposed	**0.237**	**0.753**	**0.753**	9.496	10.359
		(0.003)	(0.001)	(0.001)	(0.178)	(0.154)
0.01	Full Data	–	0.748	0.748	–	39.664
		–	(0.000)	(0.000)	–	(0.220)
	BM_1	5.194	0.764	0.762	0.052	0.725
		(0.111)	(0.005)	(0.005)	(0.004)	(0.013)
	BM_2	4.987	0.768	0.765	0.051	0.745
		(0.108)	(0.005)	(0.005)	(0.004)	(0.013)
	Proposed	**3.849**	**0.789**	**0.787**	9.622	0.717
		(0.056)	(0.006)	(0.006)	(0.179)	(0.017)
0.005	Full Data	–	0.748	0.748	–	39.664
		–	(0.000)	(0.000)	–	(0.220)
	BM_1	10.875	0.774	0.769	0.049	0.457
		(0.298)	(0.009)	(0.010)	(0.003)	(0.011)
	BM_2	10.790	0.791	0.786	0.068	0.484
		(0.281)	(0.009)	(0.009)	(0.007)	(0.010)
	Proposed	**7.312**	**0.833**	**0.829**	9.595	0.403
		(0.101)	(0.011)	(0.011)	(0.176)	(0.012)

Table 2-3. Mean and standard errors (within parenthesis) from 50 simulation runs for Normal distribution with $\mathrm{NC} = 1000$, $\mathrm{MS} = 0.3$, and $\mathrm{STN} = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.750	0.750	–	39.821
		–	(0.000)	(0.000)	–	(0.169)
	BM_1	0.319	0.753	0.753	0.104	7.680
		(0.013)	(0.001)	(0.001)	(0.007)	(0.104)
	BM_2	**0.261**	0.752	0.752	0.129	7.589
		(0.006)	(0.001)	(0.001)	(0.010)	(0.110)
	Proposed	0.302	**0.754**	**0.754**	10.019	7.705
		(0.013)	(0.001)	(0.001)	(0.337)	(0.096)
0.01	Full Data	–	0.750	0.750	–	39.821
		–	(0.000)	(0.000)	–	(0.169)
	BM_1	3.836	0.785	0.782	0.050	0.721
		(0.254)	(0.005)	(0.005)	(0.003)	(0.012)
	BM_2	3.992	0.772	0.769	0.062	0.743
		(0.245)	(0.005)	(0.005)	(0.007)	(0.012)
	Proposed	**3.203**	**0.792**	**0.789**	9.994	0.649
		(0.176)	(0.009)	(0.009)	(0.334)	(0.009)
0.005	Full Data	–	0.750	0.750	–	39.821
		–	(0.000)	(0.000)	–	(0.169)
	BM_1	6.812	0.791	0.786	0.054	0.431
		(0.478)	(0.008)	(0.008)	(0.005)	(0.010)
	BM_2	8.652	0.788	0.782	0.058	0.505
		(0.740)	(0.010)	(0.010)	(0.005)	(0.008)
	Proposed	**5.480**	**0.838**	**0.833**	9.997	0.371
		(0.363)	(0.013)	(0.013)	(0.330)	(0.009)



(a) Comparison of entropy loss. (b) Comparison of adjusted R^2.

Figure 2-4. (a) The means and standard errors (in black error bars) of two performance measures based on 50 simulation replications.

2.5 Discussion of Cluster-based Data Filtering

As sensor and communication technologies advance, Industrial Internet-based sensing processes are capable of collecting massive data for process modeling and control. However, such processes also generate a lot of redundant information, which significantly limits the quality and efficiency of the data analysis. As a result, extracting representative and high-quality data subsets is important to improve the efficiency of the data analysis. In this work, we proposed a filtering method and new criteria (FIC) to facilitate data analysis by selecting a small but effective data subset. Specifically, the proposed method partitions raw data into clusters and proportionally extracts a data subset from each cluster. The proposed cluster-based data filtering method outperformed the benchmark filtering methods on performance measures, such as information loss, the modeling goodness-of-fit in the case study, and the simulation. Furthermore, the new filtering ratio selection criteria (FIC) has shown its effectiveness in terms of balancing the trade-offs between the size of data preserved in filtering and the quality of the filtered data.

Although we focus on reducing the sample size in this paper, we can combine the proposed method with other variable screening methods when both the sample size and the number of

variables are large. State-of-the-art variable screening methods include LASSO (Tibshirani 1996), Elastic Net (Zou and Hastie 2005), sure independence screening (Fan and Lv 2008). This paper leads to a few future research directions. Data analysis under big data environments have become prevalent and even sometimes necessary because of the performance requirement. However, efficiently utilizing big data through analysis that is both time-efficient and accurate is still a challenging problem. The proposed method can also facilitate internet-of-things (IoT)-based data collection, communication, storage and analysis. For example, as the inline data de-duplication (real-time data filtering for continuous data streaming), has become more popular in recent years (Zhou, Liu, and Li 2013), future work will focus on the conversion of the current offline data filtering to online. Specifically, we need to investigate when to update the filter ratio due to process changes, product changes, etc. Furthermore, an integer programming heuristic or relaxation technique (Schrijver 1998) can be derived to help directly solve the data filtering using the objective function in (1) so that the filtering performance may be further improved. Lastly, we will investigate filtering for functional data (Sun, Huang, and Jin 2017) and imaging data (Li et al. 2019) as *in situ* measurement in manufacturing.

Chapter 3. Multi-task Learning with Latent Variation Decomposition for Multivariate Responses in a Manufacturing Network

3.1 Introduction

A manufacturing network connects multiple similar-but-not-identical manufacturing processes via sensing, computation, and actuation networks (Chen and Jin 2018). For example, in the silicon ingot production of wafer manufacturing, each silicon ingot grows from a furnace/crucible using the same Czochralski (CZ) process (Fisher, Seacrist, and Standley 2012). A graphical illustration of the production process is in Figure 3-1. However, an ingot typically requires over 50 hours to produce. Multiple furnaces are typically used as a connected manufacturing network to increase throughput. During the ingot growth, the ingot product specifications and process settings, such as material composition, size or weight of ingots, temperature, pulling speed, rotation speed, and processing time, vary from one ingot to another (Sun et al. 2016). Different ingot growth furnaces are subject to different degradation conditions and operating environments, which lead to a different variable relationship among the in situ process variables and product quality variables (Jin et al. 2019). Consequently, there are limited data from the identical CZ processes, while abundant data could be collected from multiple similar-but-not-identical ingots produced from different furnaces. On the other hand, each ingot growth process may generate in situ quality variables in the format of function or multivariate responses. These issues pose significant challenges to model such a manufacturing network for prediction and variation analysis of multivariate or profile responses.

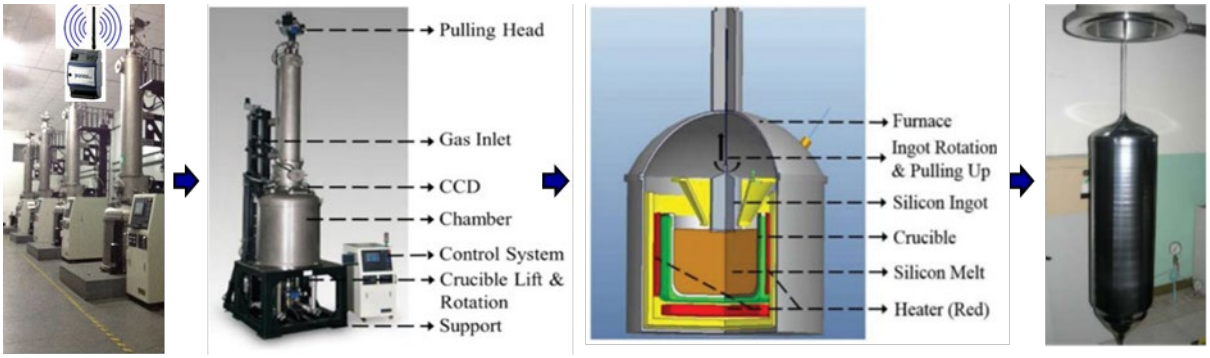


Figure 3-1. The illustration of a silicon ingot production process (Sun et al. 2016)

In literature, there are two major strategies to model similar-but-not-identical processes (e.g., furnaces). One strategy is to model all processes with the same model structure and model parameters. It assumes that all processes are identical. In the previous silicon ingot production, a logistic regression model was estimated by using the combined data sets from all ingots (Sun et al. 2016). In literature, various prediction models, such as linear regression model, which has various extensions including Ridge Regression (Hoerl and Kennard 1970), LASSO (Tibshirani 1996), Elastic Net (Zou and Hastie 2005), are estimated by using the identical process assumption. They do not consider the heterogeneities among the processes and cannot reflect the heterogeneities in the model structures or parameters.

The other strategy will use the same type of model with different model parameters for different machines. The model can be independently estimated for each machine based on its own data, but it may take a longer time to obtain sufficient samples. To model the data aggregated from different machines with heterogeneity, people proposed transferred learning and multi-task learning (MTL) framework. For transfer learning (Pan and Yang 2009), it designates a source model, which is trusted to be accurate for the source process and transfers the model with updated parameters to the target process by incorporating the process similarities. However, it is sometimes difficult to find a reliable source model, when all processes are experiencing a shortage of training

samples. As a result, MTL, which simultaneously models the data from all processes, is a more suitable framework to adopt under this case scenario.

Specifically, MTL treats modeling the data from each process/machine as a learning task. There are two major types of MTL methods, which are suitable for modeling of a manufacturing network: feature-based MTL and parameter-based MTL. Feature-based MTL focuses on jointly generating or selecting features for multiple tasks. It investigates how different processes can share identical or similar feature representations induced from the original feature set (Obozinski, Taskar, and Jordan 2010, Gong, Ye, and Zhang 2013). Parameter-based MTL focuses on jointly estimating the model parameters for multiple tasks. Parameter-based MTL first assumes that the variable sets are pre-defined and remain the same across different tasks. Then similar-but-non-identical parameters are estimated to reflect process heterogeneities, such as the low-rank regularization (Ando and Zhang 2005, Chen et al. 2009), cluster-wise model parameter regularization (Jacob, Vert, and Bach 2009, Kang, Grauman, and Sha 2011), task-relation learning (Evgeniou and Pontil 2004b, Parameswaran and Weinberger 2010), etc. For low-rank regularization, one assumes that the model parameters for multiple tasks will form a low-rank structure. A major limitation of the MTL methods is the assumption that the variation of the response variable can be fully explained by the observed predictors. When having multivariate response variables in different tasks, the covariance structure between the responses may not be fully explained by the observed variables, which makes MTL-based methods less effective.

To address the issue of the unexplained variation, many works focus on modeling the latent variations of the multivariate responses. For example, principal component analysis (PCA) and its various variants are commonly adopted, such as smoothed FPCA (Silverman 1996), sparse FPCA (James, Hastie, and Sugar 2000), sparse longitudinal FPCA (Yao, Müller, and Wang 2005), multi-

channel FPCA (MFPCA) (Paynabar, Zou, and Qiu 2016), tensor PCA (Yan, Paynabar, and Shi 2014), sparse multi-channel FPCA (SMFPCA) (Zhang et al. 2018), etc.. This type of PCA based methods discovers the latent variation patterns from data. However, the heterogeneity components in the process modeling are often ignored. In addition, these PCA methods are unsupervised learning methods, which cannot be used to model a process with a response directly.

In this work, we propose MTL with latent variation decomposition (LVD), called *MTL-LVD*, to integrate the feature-based MTL framework with PCA in modeling the multivariate responses of multiple similar-but-non-identical processes in a manufacturing network. We use the MTL to model the impact from the observable predictors to the multivariate response, due to the similarity and the heterogeneity among the tasks. Then we add the LVD from the PCA to model the impact of unobservable predictors to the multivariate response and explore the latent variable structures. Since the total variation of the multivariate response variable could be decomposed as the MTL based on the observable predictors, and LVD based on the latent variables, representing the common unobserved variation sources across different machines from the network, the proposed method will model a manufacturing network closely. The simulation study and the case study on the silicon ingot manufacturing data has shown that MTL-LVD can discover the latent variation patterns for the multivariate response variable and achieves significantly better product quality prediction accuracy.

The rest of the paper is organized as follows. Section 3.2 introduces the proposed MTL-LVD framework; Section 3.3 includes a simulation study, examining the performance of MTL-LVD on discovering the multi-dimensional latent variation terms and the model prediction accuracy; Section 3.4 performs a case study, illustrating the superior performance of MTL-LVD

on product quality forecasting based on the ingot manufacturing data; Section 3.5 draws the conclusion and discussed some future research directions.

3.2 Methodology

This section first introduces the MTL-LVD methodology under the supervised MTL framework using PCA for variation decomposition. Then, model parameter estimation and hyper-parameter tuning strategy are discussed.

Without the loss of generality, we can assume that there are m similar-but-non-identical processes and MTL-LVD is build based on the following assumption: 1) We assume that there are m similar-but-non-identical processes/machines in a manufacturing network and the j^{th} machine is producing n_j parts. 2) When building a linear model for each process, model parameters for multiple processes should be low-rank, due to the similarity of different machines in the manufacturing network. 3) There is typically latent variation in the process, which can lie in a low-dimensional linear space, which can be represented by a few unknown bases.

We denote the observed data for the j-th process as (x_j, y_j), where $j = 1, \dots, m$. The observable predictors x_j are process setting variables or scalar features of *in situ* process variables. Collectively, the r-th response variable of the j-th process, denoted as ${y_j}^{(r)}$, can be modeled as

$$ {y_j}^{(r)} = \boldsymbol{x}_j{}'\boldsymbol{\beta}_j + \epsilon_j \ , \ j = 1, \cdots, m. \tag{8} $$

Past works mainly focus on recovering $\boldsymbol{\beta}_j$ among similar-but-non-identical processes. To mitigate the shortcomings on the capability of modeling multivariate responses with MTL and improve the modeling performance under an incomplete set of variables, we first extend the Equation (8) to a multivariate response MTL framework with d responses. The observed data for each machine j become $(\boldsymbol{x}_j, \boldsymbol{y}_j)$, where $\boldsymbol{x}_j = \left({x_j}^{(1)}, \cdots, {x_j}^{(p)}\right) \in \mathbb{R}^{p \times 1}$, $\boldsymbol{y}_j = \left({y_j}^{(1)}, \cdots, {y_j}^{(d)}\right) \in \mathbb{R}^{d \times 1}, j = 1, \dots, m$. Furthermore, we propose to decompose the latent variation patterns into a low-

dimensional subspace with basis denoted by $\boldsymbol{v}^{(r)} = ({v^{(r)}}_1, \dots, {v^{(r)}}_q)'$, $r = 1, \dots, d$, and the corresponding PCA scores as $\boldsymbol{u}_j = (u_{1,j}, \dots, u_{q,j})'$, $j = 1, \dots, m$. In the proposed MTL-LVD model, we assume that the response variable ${y_j}^{(r)}$can be decoupled into the observable variations explained by the input variable $\boldsymbol{x}_j$ and latent variation patterns in the latent space with basis $\boldsymbol{v}^{(r)}$ as

$$ {y_j}^{(r)} = \boldsymbol{x}_j{}' {\boldsymbol{\beta}_j}^{(r)} + \boldsymbol{u}_j{}' \boldsymbol{v}^{(r)} + \epsilon_j^{(r)}, j = 1, \cdots, m. \tag{9} $$

Furthermore, $\boldsymbol{\beta}_j \in \mathbb{R}^{p \times 1}$ is the model coefficient and is different for each machine j. ϵ is assumed to follow the normal i.i.d distribution $\epsilon_j^{(r)} \sim N(0, \sigma^2)$. It worth noting that the assumption would not hurt the generalization of the proposed method since we can always use more latent loading vector $\boldsymbol{v}^{(r)}$ to model the non-i.i.d. variation patterns. To link the model coefficients for each machine $\boldsymbol{\beta}_j = 1, \cdots, m,$ we propose to borrow the framework from the parameter-based MTL, where the nuclear norm is used to enforce the low-rank structures on all model coefficients. Furthermore, without loss of generality, we can assume that the latent loading vectors $\boldsymbol{v}^{(r)}$ shared among all machines j and is orthonormal with each other.

Collectively, we would like to estimate the model parameters via the

$$ L = \sum_{j=1}^{m} \sum_{r=1}^{d} \left\| {y_j}^{(r)} - \boldsymbol{x}_j{}' {\boldsymbol{\beta}_j}^{(r)} - \boldsymbol{u}_j{}' \boldsymbol{v}^{(r)} \right\|^2 + \lambda_1 \|B\|_* \tag{10} $$

$$ s.t. V'V = I, $$

where $B = \begin{pmatrix} {\boldsymbol{\beta}_1}^{(1)} & \cdots & {\boldsymbol{\beta}_m}^{(1)} \\ \vdots & \ddots & \vdots \\ {\boldsymbol{\beta}_1}^{(d)} & \cdots & {\boldsymbol{\beta}_m}^{(d)} \end{pmatrix}$ and $V' = (\boldsymbol{v}^{(1)} \quad \cdots \quad \boldsymbol{v}^{(d)})$. Here, B represents model coefficient matrix corresponding to the predictors for d responses and j processes. Since processes under consideration are similar-but-non-identical, B is assumed to be low rank, subject to nuclear-

norm regularization $||B||_*$. Furthermore, V is the semi-orthogonal matrix realized from $\boldsymbol{v}^{(r)}$ for d responses, and the semi-orthogonality is enforced by the constraint $V'V = I$.

Estimating the parameters of MTL-LVD in Equation (10) is not trivial due to the combination of the nuclear norm, orthogonality constraint, and decomposed model coefficients (i.e. ${\boldsymbol{\beta}_j}^{(r)}$ and $\boldsymbol{v}^{(r)}$). We proposed the regularized regression approach together with the block coordinate descent algorithm to iteratively update ${\boldsymbol{\beta}_j}^{(r)}$, $\boldsymbol{u}_j$, and $\boldsymbol{v}^{(r)}$. The model hyper-parameters, including λ_1, which is the penalty coefficient of the nuclear-norm, and q as the dimension of the latent variation term V as shown in Equation (1), can be tuned via the five-fold cross-validation (CV).

Realizing the Equation (1) considering the observational data, we denote $X_j = (\boldsymbol{x}_{1,j}, \dots, \boldsymbol{x}_{n_j,j})'$ as the design matrix, $U_j = (\boldsymbol{u}_{1,j}, \dots, \boldsymbol{u}_{n_j,j})'$ as the generated PCA scores, and ${\boldsymbol{y}_j}^{(r)} = ({y_{1,j}}^{(r)}, \dots, {y_{n_j,j}}^{(r)})'$ to be the rth response vector for the jth process, where n_j is the number of samples for the jth process. The following constrained quadratic optimization problem can be used to estimate B, U, V.

$$L = \sum_{j=1}^{m}\sum_{r=1}^{d}\left\|{\boldsymbol{y}_j}^{(r)} - X_j{\boldsymbol{\beta}_j}^{(r)} - U_j\boldsymbol{v}^{(r)}\right\|_F^2 + \lambda_1||B||_* \qquad (11)$$
$$s.t. V'V = I.$$

Finally, Equation (11) can be further written by organizing the multivariate responses in a matrix format as:

$$L = \sum_{j=1}^{m} \| Y_j - X_jB_j - U_jV' \|^2 + \lambda_1||B||_* \qquad (12)$$
$$s.t. V'V = I,$$

where $Y_j = (\boldsymbol{y}_j{}^{(1)}, \dots, \boldsymbol{y}_j{}^{(d)})$ and $B_j = \begin{pmatrix} \boldsymbol{\beta}_j{}^{(1)} & \cdots & \boldsymbol{\beta}_j{}^{(d)} \end{pmatrix}$. To optimize, we propose to follow the block coordinate type of strategy to update B, U, V iteratively until convergence. To achieve this, we first prove the following propositions:

Proposition 1: Given B, U_j in Equation (12), V can be solved in a closed form via the singular value decomposition (SVD) as follows:

$$\begin{pmatrix} Y_1 - X_1 B_1{}^{k-1} \\ \vdots \\ Y_m - X_m B_m{}^{k-1} \end{pmatrix}' \begin{pmatrix} U_1{}^{k-1} \\ \vdots \\ U_m{}^{k-1} \end{pmatrix} = RDW', \qquad (13)$$

$$V = RW',$$

where the first q columns in R are used for $V = RW'$ if $q < d$. The proposition 1 is also named the Procrustes rotation procedure, and the detailed proof is shown in (Zou, Hastie, and Tibshirani 2006).

Proposition 2: Given B_j and V in Equation (12), U_j can be solved via

$$U_j = \left(Y_j - X_j B_j\right) V. \qquad (14)$$

The proof of Proposition 2 is shown in the Appendix B.

Proposition 3: Given U_j and V in Equation (12), B_j can be solved via the proximal gradient descent as follows:

$$B = P \,\mathrm{diag}(\hat{\sigma}_1 \dots \hat{\sigma}_m) Q', \qquad (15)$$

where $\hat{\sigma}_i = \begin{cases} \sigma_i - \lambda & \sigma_i \geq \lambda_1 \\ 0 & -\lambda_1 \leq \sigma_i \leq \lambda_1 \\ \sigma_i + \lambda & \sigma_i \leq -\lambda_1 \end{cases}$, and P, σ_i, Q can be obtained by SVD decomposition on $\tilde{B} = \begin{pmatrix} \tilde{\boldsymbol{\beta}}_1{}^{(1)} & \cdots & \tilde{\boldsymbol{\beta}}_m{}^{(1)} \\ \vdots & \ddots & \vdots \\ \tilde{\boldsymbol{\beta}}_1{}^{(d)} & \cdots & \tilde{\boldsymbol{\beta}}_m{}^{(d)} \end{pmatrix}$ as $\tilde{B} = Pdiag(\sigma_1 \dots \sigma_m) Q'$ and

$$\tilde{\boldsymbol{\beta}}_j{}^{(r)} = \boldsymbol{\beta}_j{}^{(r)} - t_1 \left(\nabla \left\| \boldsymbol{y}_j{}^{(r)} - X_j \boldsymbol{\beta}_j{}^{(r)} - U_j \boldsymbol{v}^{(r)} \right\|_F^2 / \nabla \boldsymbol{\beta}_j{}^{(r)} \right).$$

The derivation of $\nabla\left\|\boldsymbol{y}_j{}^{(r)} - X_j\boldsymbol{\beta}_j{}^{(r)} - U_jV'\right\|_F^2/\nabla\boldsymbol{\beta}_j{}^{(r)}$ is shown in the Appendix B.

By decomposing the optimization problem into three sub-problems via the block coordinate descent approach, the iterative algorithm is summarized as follows:

Algorithm 1. Optimization algorithm for solving MTL-LVD

initialize

B_j, U_j, V as $B^0, U_j{}^0, V^0$.

end

for $j = 1,2,3,\ldots$ **do**

Update V given B and U_j according to (13).

Update U_j given B and V according to (14).

Update B given U_j and V according to (15).

If $\frac{\left\|L^k - L^{k-1}\right\|_F^2}{L^{k-1}} \leq \epsilon$, **then**

Stop

end

end

Proposition 4: The proposed algorithm for solving Equation (12) can converge to a stationary point.

The convergence of the block coordinate descent algorithm (BCD) can be proved by the monotonic decrease of Equation (12) since each BCD update would only decrease the objective function. Furthermore, the objective function in Equation (12) is lower bounded by zero. Therefore, the BCD algorithm must converge to a stationary point.

Finally, we would like to analyze the complexity of the proposed algorithm. Recall that there are m similar-but-non-identical processes, the dimensionality of the response is d, the number of samples in each process are n_j, the number of observed variables is p, the number of latent variables is q. The computational complexity of updating all parameters in each iteration becomes O($qpnd + p^2nd + p^2md^2 + pm^2d$). The derivation can be seen in the Appendix B.

3.3 Simulation Study

To evaluate the performance of MTL-LVD, we performed a simulation study to evaluate two performance metrics as follows: 1) The accuracy that MTL-LVD is able to recover the simulated orthogonal loadings of the decomposed latent variation evaluated by the plot of simulated and recovered loadings. 2) The prediction error of the multivariate response variable evaluated by root mean squared prediction errors (RMSPEs) on the testing data set $(\boldsymbol{y}_j{}^{(r)}, X_j)$: $\sum_{j=1}^{m}\sum_{r=1}^{d}\left\|\boldsymbol{y}_j{}^{(r)} - X_j\boldsymbol{\beta}_j{}^{(r)} - U_j\boldsymbol{v}^{(r)}\right\|_F^2$ for m processes and d responses.

The settings to generate the simulation data set are summarized in Table 3-1. Assuming that there are eight similar-but-non-identical processes ($m = 8$). Within each process, there are eight observable variables in $\boldsymbol{x}_j = (x_{1,j}, \dots, x_{p,j})'$ $(p = 10)$, and three variables in the latent variation term $\boldsymbol{u}_j = (u_{1,j}, \dots, u_{q,j})'$ $(q = 3)$. Each process has 15 data points generated from $\boldsymbol{x}_j{}' \sim N(\boldsymbol{\mu}_X, \Sigma_X)$, and $\boldsymbol{u}_j{}' \sim N(\boldsymbol{\mu}_U, \Sigma_U)$, where $\boldsymbol{\mu}_X$, $\boldsymbol{\mu}_U \sim N(\mathbf{0}, I)$, and Σ_X, Σ_U are covariance matrix with diagonal elements as 1 and off-diagonal elements as 0.1. Additionally, each process has d multivariate responses as $d = 50$. The underlying model used to generate the response is

$$y_j{}^{(r)} = \boldsymbol{x}_j{}'\boldsymbol{\beta}_j{}^{(r)} + \boldsymbol{u}_j{}'\boldsymbol{v}^{(r)} + \varepsilon, \tag{16}$$

where $j = 1, \dots, 8$, $r = 1, \dots, 50$. The similarity of the process is simulated through the low-rank

structure of the model parameter matrix $B = \begin{pmatrix} \boldsymbol{\beta}_1^{(1)} & \cdots & \boldsymbol{\beta}_m^{(1)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\beta}_1^{(d)} & \cdots & \boldsymbol{\beta}_m^{(d)} \end{pmatrix}$ of Equation (10). More specifically, the parameters for the observed predictors of machine j, $\boldsymbol{\beta}_j = \left(\boldsymbol{\beta}_j^{(1)\prime} \quad \cdots \quad \boldsymbol{\beta}_j^{(d)\prime}\right)'$, corresponds to the column j of $B = \begin{pmatrix} \boldsymbol{\beta}_1^{(1)} & \cdots & \boldsymbol{\beta}_m^{(1)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\beta}_1^{(d)} & \cdots & \boldsymbol{\beta}_m^{(d)} \end{pmatrix}$, which was simulated with a random linear combination of n basis vectors out of m basis vectors (Rice 1966), where $n < m$. To simulate the basis matrix $V' = (\boldsymbol{v}^{(1)} \quad \cdots \quad \boldsymbol{v}^{(d)})$, where $\boldsymbol{v}^{(r)} = (v^{(r)}{}_1, \ldots, v^{(r)}{}_q)'$, which was randomly generated by pseudo-spline basis implemented in GAMSEL package (Chouldechova and Hastie 2015) with an input vector of $cosine(\frac{\boldsymbol{z}}{10})$ and $\boldsymbol{z} = (1, \ldots, 50)'$. The error term ε follows $N(0, \Sigma)$, and we set the signal-to-noise ratio ($\frac{variance\left(\boldsymbol{x}_j'\boldsymbol{\beta}_j^{(r)} + \boldsymbol{u}_j'\boldsymbol{v}^{(r)}\right)}{variance(\varepsilon)}$) as 10.

To evaluate the accuracy of the estimated MTL-LVD models, we compare the proposed method with three benchmark methods: linear regression models for every single process (Linear-S), one linear regression model for all processes (Linear-A), and MTL with nuclear norm (MTL-N). MTL-N has the revised loss function from Equation (10) as $\sum_{j=1}^{m}\sum_{r=1}^{d}\left\|y_j^{(r)} - \boldsymbol{x}_j'\boldsymbol{\beta}_j^{(r)}\right\|^2 + \lambda_1||B||_*$, which models the data combined from all processes but does not consider the latent variation, represented by $\boldsymbol{u}_j'\boldsymbol{v}^{(r)}$ in Equation (16).

Table 3-1. Settings of simulation data sets

Simulation Settings	Values
Number of Machines Simulated	8
Length of Response	50
Number of Samples for Each Machine	15
Number of Observed Predictors	10
Number of Latent Predictors	3

For all methods in the simulation and the case study in the next section, 5-fold CV was employed to tune the model hyper-parameters for all the models based on training data, including the penalty coefficient on the nuclear-norm (λ_1) of MTL-N and MTL-LVD. Furthermore, the dimension of the latent variation term V' for MTL-LVD is also tuned via the 5-fold CV. To ensure the reproducibility of the results, 50 replications were performed in the simulation study. Within each replication, the data are randomly partitioned in a 70-30 fashion for the training and testing datasets. In each replication, all models were trained based on the same training data set and tested for prediction accuracy in root mean squared errors (RMSEs) using the testing data set.

Table 3-2. Averages and standard errors of RMSPEs (or MEs) from 50 simulation runs

Method	Testing RMSE
Linear-S	1.670 (0.264)
Linear-A	0.334 (0.003)
MTL-N	0.348 (0.004)
MTL-LVD	0.299 (0.004)

Table 3-2 reports the average of RMSEs with standard errors (S.E.) shown in parenthesis based on 50 simulation replications. We can see that MTL-LVD offers a better prediction performance than all three benchmark models. Specifically, Linear-S has a significantly higher prediction error due to the limited amount of sample size for model training, since only the data from a single process are used. In the meantime, Linear-A has a much smaller prediction error by aggregating the samples from all similar-but-non-identical processes. However, neither of these two models consider the heterogeneity among processes when the data are aggregated. Using the MTL framework with nuclear norm (MTL-N) to fit different sets of model parameters for different processes, the prediction error can be further reduced. However, MTL-N was not capable of

capturing the unexplained latent variation, which was simulated by the $\boldsymbol{u}_j{}'\boldsymbol{v}^{(r)}$ term in data generator of Equation (16). As a result, the MTL-N was outperformed by the proposed method MTL-LVD, which achieves the lowest prediction error among all methods tested.

Another purpose of the simulation is to validate whether the MTL-LVD can discover the latent variation, which can only be explained by the latent variation term. Specifically, we would like to test if MTL-LVD can recover the true basis structure simulated in the latent variation term ($\boldsymbol{v}^{(r)}$ in Equation (9)), which has three simulated basis vectors aggregated as $V = (\boldsymbol{v}^{(1)} \quad \boldsymbol{v}^{(2)} \quad \boldsymbol{v}^{(3)})$. We generated Figure 3-2 to compare the true values of three basis vectors in V as solid lines and the recovered values by MTL-LVD as dotted lines. One can observe that MTL-LVD is able to recover the overall trends of all three basis vectors. In summary, the proposed MTL-LVD has the capability to achieve the best prediction accuracy while recovering the multi-dimensional latent variation.

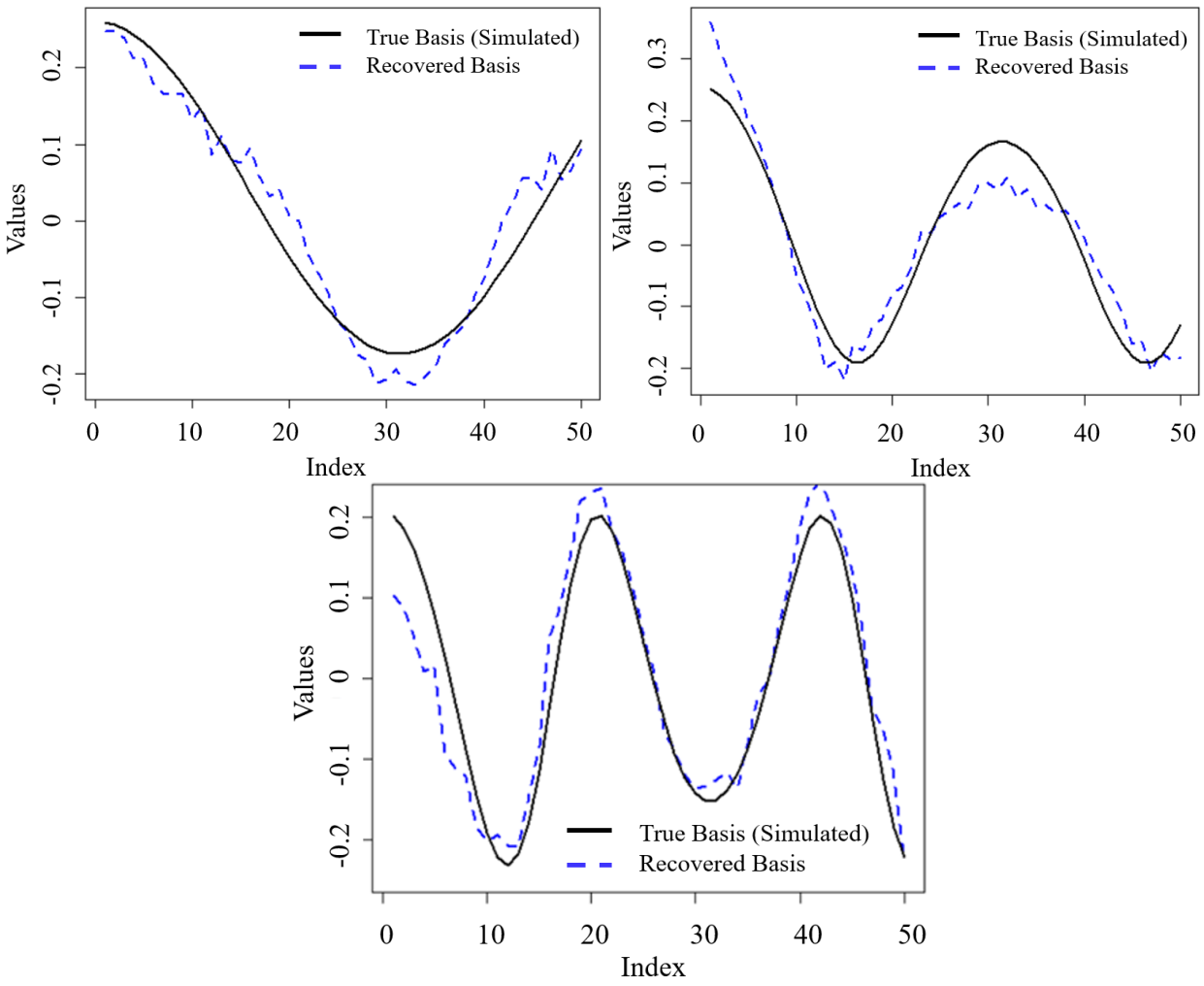


Figure 3-2. The true values of the three basis vectors simulated (the solid line) and the correspondingly recovered basis vectors by MTL-LVD (the dotted line)

3.4 Case Study

The CZ ingot growth process produces monocrystalline silicon ingots is both costly and time-consuming (Fisher, Seacrist, and Standley 2012). There can be various types of defects occurred on the ingots during manufacturing, such as the growth of a polycrystalline ingot when the monocrystalline ingot is desired (Sun et al. 2016). Therefore, it is important to model the relationship between the product quality variable and the process variables to understand how the process variables can impact the product quality. In this case study, the response variable is the

diameter of the continuously formed circular ingot surfaces, which are then moving-averaged into multivariate response vectors of length 50 for all ingot samples ($d = 50$ in Equation (9)). 10 process variables are generated as the summary statistics of the in situ process variables ($\boldsymbol{x}_j' \in \mathbb{R}^{10}$ in Equation (9)), with the variable names omitted due to the confidential concerns. Among the five different manufacturing processes (furnaces), we had the data of 13, 19, 19, 21, and 25 ingots, respectively. Each ingot is treated as a sample. Table 3-3 summarizes the data setting of the crystal silicon growth process case study.

Similar to the simulation study, we compare the MTL-LVD with three representative benchmarks: Linear-S, Linear-A, and MTL-N. 50 replications were performed in the case study with the data from 5 furnaces randomly partitioned in a 70-30 fashion into the training and testing datasets within each replication with the same 5-fold CV to select tuning parameters as in the simulation studies.

Table 3-3. Settings of data from the ingot growth manufacturing

Settings	Values
Length of Response	50
Number of Observed Predictors	10
Sample Size for Furnace 1	13
Sample Size for Furnace 2	19
Sample Size for Furnace 3	19
Sample Size for Furnace 4	21
Sample Size for Furnace 5	25

Table 3-4. Averages and standard errors of RMSPEs (or MEs) from 50 replications in the case study

Method	Testing RMSE
Linear-S	7.454 (1.478)
Linear-A	3.129 (0.143)
MTL-N	1.075 (0.119)
MTL-LVD	**0.565 (0.008)**

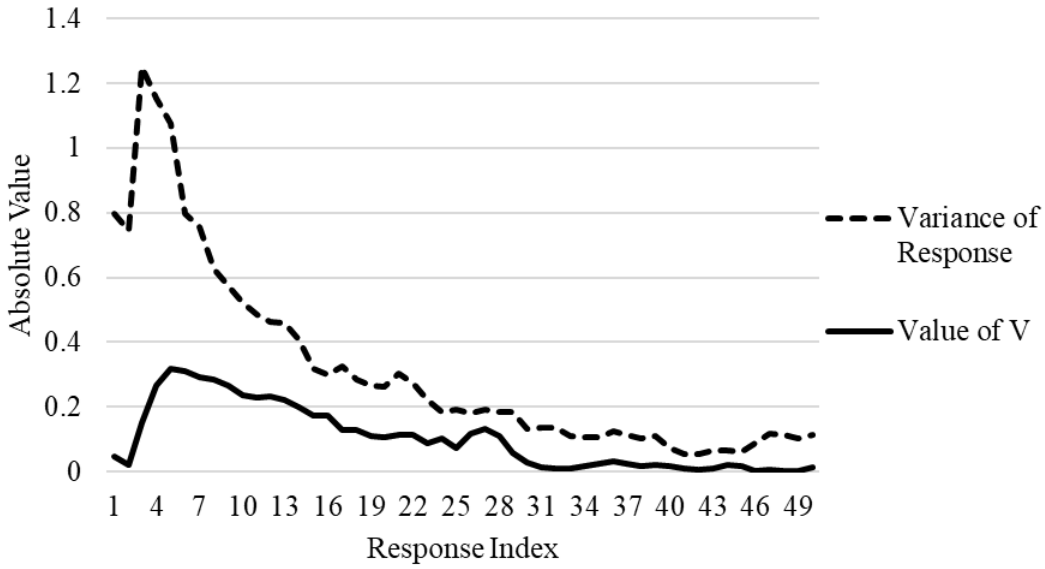


Figure 3-3. The variance of responses data across five manufacturing processes for 50 responses (index 1-50) versus the value of V (the model parameters for latent variation term)

Finally, we will still use RMSE to evaluate the prediction accuracy and the result of the prediction error is summarized in Table 3-4. The proposed MTL-LVD has a significantly better performance on predicting the diameter of the silicon ingot surface than the three benchmark models did over replications. Linear-S has the largest prediction error mainly because separately modeling each process will suffer from the effect of a small training data set. Linear-A has a much smaller prediction error comparing with Linear-S, since Linear-A aggregates the samples from

five similar-but-non-identical furnaces to increase the training sample size of the model. However, neither of these two models consider heterogeneities among different manufacturing processes. Using the MTL framework with nuclear norm (MTL-N), we can fit similar-but-non-identical models for multiple processes (furnaces) in the network. As a result, the prediction error was greatly reduced. Similar to the result of the simulation study, MTL-N could not capture the unexplained variation by the observed variables. As a result, introducing the latent variation term in MTL-LVD significantly reduced the prediction error on the testing data.

Furthermore, similar to the simulation study, to demonstrate the usefulness of the latent variation term in MTL-LVD, we show the relationship between the parameters of the latent variation term ($\boldsymbol{v}^{(r)}$ in Equation (9)) and variation of the multivariate response ($y_j{}^{(r)}$ in Equation (9)) in Figure 3-3. Specifically, $\boldsymbol{v}^{(r)}$ is determined to be one dimensional by 5-fold CV and becomes a scalar value for each response, so that in total, we have $V' = (v^{(1)}, \dots, v^{(50)})$ for 50 responses. We expect that $v^{(r)}$ should become a larger non-zero value as the variance of $y_j{}^{(r)}$ increases to help to capture the unexplained variance by the observable variables. In Figure 3-3, the variances of the first few responses are large, which shows that the variations introduced by different furnaces are large at the initial manufacturing stages (i.e., for the first a few hours of the process). The variation gradually decreased after the initial stages and stayed at a low level towards the end of the process. Correspondingly, we also observed that the parameters of the latent variation term were large at the initial stages and decreased afterward. The pattern of the parameter values ensembles the patterns of response variances. The instability at the beginning stages of ingot formation in the CZ process is perhaps due to an unstable polysilicon melting can lead to variations in oxygen concentration at the early stages of different ingot formation (Pascoa 2014). In summary, the latent variation term in MTL-LVD can help the observed variables to capture the variation

introduced during the data aggregation of multiple similar-but-non-identical manufacturing processes. As a result, the prediction performance of MTL-LVD can be significantly improved.

3.5 Conclusion of MTL-LVD

Although machine learning methods require sufficient training data samples to generate a model with good prediction performance, such a sample size requirement can hardly be fulfilled in many manufacturing applications. In this research, we studied a motivating case of modeling the crystal silicon ingot manufacturing, which requires the furnace to be maintained at a temperature of approximately 1500 degrees Celsius over 50 hours. Due to the high cost of the production, the in situ manufacturing data and silicon ingot quality data can only be collected from a few silicon ingots from a furnace at a time. In this work, we incorporated a variation decomposition approach to model the latent variation, which cannot be captured by the observable variables, into the feature-based MTL. Furthermore, we generalized the proposed methodology into a multivariate response and MTL model, which is capable of modeling multivariate responses monitored across multiple similar-but-non-identical processes (e.g., furnaces). The simulation study and the case study have shown that MTL-LVD can not only recover the latent variation among data of similar-but-non-identical processes but also offer significantly better accuracy on predicting the multivariate response.

There are several future research directions on MTL-LVD that deserve further investigations. The first one is incorporating different regularization terms into MTL-LVD based on different engineering perceptions/assumptions of the underlying problems. For example, l_1-norm can be applied to replace the nuclear norm in the MTL-LVD to perform variable selection and filter out the sensor signals, which are not strongly correlated with the responses (Tibshirani 1996). Another research direction can be using the model parameters of the latent variation term,

which reflects the commonality of machines, to perform process monitoring. Such a learned commonality can also be continuously updated and become more accurate with incoming samples.

Chapter 4. Distributed Data Filtering and Modeling for Fog Manufacturing

4.1 Introduction

The Cloud Computing (Wang, Zhang, and Jin 2020) has significantly improved manufacturing productivity and flexibility, ensured product quality, and reduced the manufacturing cost. Fog Computing extends the Cloud Computing paradigm close to the "Edge" of the manufacturing network (Wang, Zhang, and Jin 2020), i.e., near the manufacturing equipment, processes and data (Sun et al. 2016), thus improving the performance of runtime metrics, such as time latency of the computation services and the communication bandwidth utilization (Zhang et al. 2019). In this paper, Fog is defined as low-cost computation devices, including Edge and other devices between Edge and Cloud (Jin, Yan, and Lee 2020). A Fog Manufacturing process applies both Fog and Cloud Computing cooperatively collaboratively and integrates manufacturing equipment and processes into an interconnected network through sensing, actuation, and computation nodes (Zhang et al. 2019). Specifically, most of the computation services, such as process modeling, monitoring, diagnosis in manufacturing, are provided by using both Fog and Cloud to perform data analysis and facilitate decision-making in manufacturing. Such Fog Manufacturing (Zhang et al. 2019) provides promise on reliable and responsive computation services, with the potential for privacy preservation to process data in local computation units. However, due to the relatively limited communication bandwidth to Cloud and computation capabilities of Fog nodes, a large amount of data from the manufacturing network lead to significant computation time latency for data analysis (Bonomi et al. 2012).

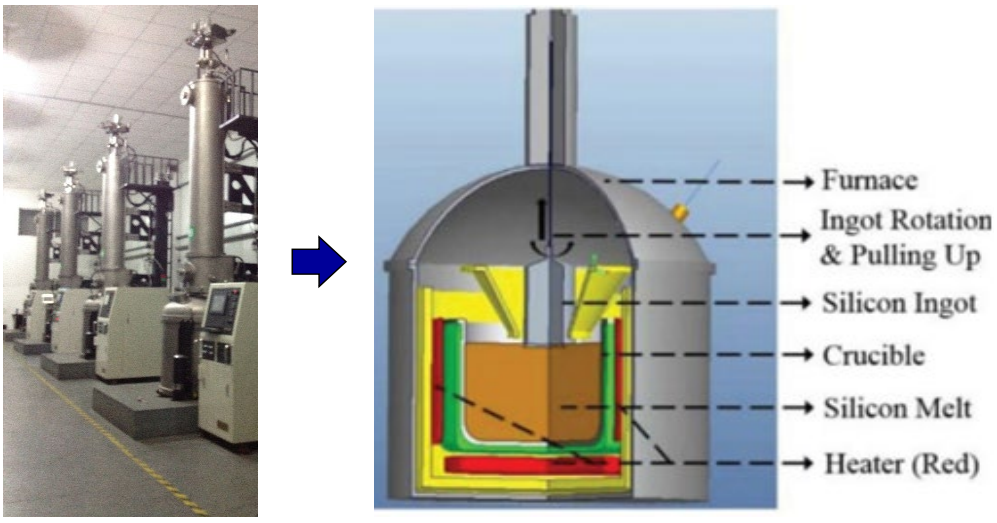


Figure 4-1. A schematic of a crystal growth furnace network: the network of multiple furnaces (left) and the internal structure of each one (right) (redrawn with authors' permission) (Sun et al. 2016)

Taking an ingot growth manufacturing network as an example (Sun et al. 2016), there are multiple ingot growth furnaces equipped with sensors jointly collecting a large amount of *in situ* data (Fisher, Seacrist, and Standley 2012). Figure 4-1 shows the facilities of the manufacturing. Specifically, the sensors on each furnace record more than thousands of multivariate observations per ingot. As more processes are connected in the manufacturing network, it is challenging to analyze all the data since the Fog nodes have limited computation capabilities, while the transmission of all data to Cloud will pose significant time latency. Thus, the current computation network cannot provide computation services in a timely manner. As a result, decisions in prediction, change detection, root cause diagnosis, cannot be completed by deadlines, and the manufacturing process yields poor quality and production control performances. Second, data privacy is another issue when sending data to the Cloud. Even with the 5G communication when the time latency is no longer a challenge in the near future (Cao et al. 2018), sending all data to Cloud will release manufacturing data containing Intellectual Property and Know-How of the

enterprises (e.g., process recipe, part design, and product quality). Third, the computation will become risky in the event of communication loss to Cloud, which cannot provide reliable computation results.

As a result, there is a need to generate small but informative data to provide affordable computational cost for Fog nodes and reduce the communication workload to Cloud. This game-changing opportunity lies in the similarity of the manufacturing processes, from the same ingot growth furnace configuration, comparable machine status, and maintenance, to similar recipe used in producing the same type of products. The data collected from such a "similar-but-non-identical" manufacturing process will have similarities of process models in terms of model structures and parameters (i.e., parameters in the quality process models) and significant information redundancy (Wang, Yang, and Stufken 2019). Such similarities are often from the similarity of the manufacturing process, equipment, configuration, recipe, machine status, product design, and performance measure. Reducing the same size by depressing the information redundancy becomes a potential solution.

There are existing data filtering methods, also known as "subsampling", studied as a useful tool to extract small but informative data subsets in manufacturing. However, obtaining a suitable sample size for analysis can be a nontrivial question. A large amount of data will create communication and computational challenges, while a small data set will affect the data analysis performance. Therefore, existing methods often apply data filtering before analysis without knowing if the sample size filtered is appropriate for analysis. As a result, there is no guarantee that the filtering will result in a suitably sized data subset for data analysis. Furthermore, it is still an open question to utilize the similarities of the same types of processes (e.g., ingot growth) to reduce the sample size required for estimating the model of each process.

The objective of this research is to integrate the data filtering and modeling for Fog Manufacturing with similar-but-non-identical processes. Other data analysis, such as process monitoring, root cause diagnosis, and prognosis, are out of the scope of this paper. Furthermore, although the same type of manufacturing processes may not be guaranteed to have similar input-output variables relationship, modeling the same type of processes with dramatically different variable relationships is out of the scope of this study. To tackle the knowledge gaps of the state-of-the-art, we proposed a distributed data filtering and modeling (DDFM) method. Specifically, the proposed DDFM method penalizes the sample size used for model estimation to reduce the computational cost. Additionally, the DDFM method penalizes heterogeneities between the individual model of each process and the network baseline model to incorporate the process similarity assumption. Here, the network baseline model reflects the commonalities in model structures among processes and warm-starts the model estimation of each process to reduce the data size required. To accelerate the computation of the proposed method, we propose a heuristic algorithm that distributes the filtering and modeling of each process in each Fog Node.

The DDFM method will effectively reduce the sample size and ensure computation performances because it automatically and adaptively determines the sample size extracted from each process with no prior knowledge required. Furthermore, the DDFM enables information sharing across the models of different processes in Fog Manufacturing in a sensible way for improved computation performance. The most informative samples from similar-but-non-identical processes will be identified to estimate a network baseline model.

A simulation study and a real Fog Manufacturing testbed for the ingot growth manufacturing have been used to validate the DDFM method. The benefit of the proposed method is two-fold. First, the proposed method is easy to be embedded in Fog Manufacturing for process

modeling with a reduced computational cost. Second, compared with all benchmark filtering algorithms, the proposed method generated the most cost-effective modeling performance (i.e., the smallest modeling error with the same sample size filtered from the Fog Manufacturing).

We organize the rest of the paper as follows. In Section 4.2, we review the existing work of data filtering and modeling. In Section 4.3, we introduce the proposed data filtering and modeling formulation and its estimation algorithm. In Section 4.4, we perform a simulation study to evaluate the performance of the proposed method under varying simulation settings. In Section 4.5, we perform a case study using a silicon ingot manufacturing network with the proposed algorithm implemented in Fog manufacturing testbed. In Section 4.6, we provide a summary and discuss future work.

4.2 Literature Review

Data analytics in manufacturing, such as process modeling, quality prediction, process monitoring, and process control, have been widely adopted. To meet the demands of various data analytics tasks, people have integrated a variety of data analytics algorithms as computational service platforms. Existing works on computational services include but not limited to the Berkeley data analysis system (BDAS) (Stoica et al. 2017), Cloud-based industrial automation (Hegazy and Hefeeda 2014), machine learning-based natural language processing (NLP) services (Staar et al. 2018), healthcare services (Sutton et al. 2018), etc.

Among the data analytics algorithms used for computation services, regression is one of the most popular methods to model manufacturing processes or networks. Along this direction, the state-of-the-art approaches include Ridge Regression (Hoerl and Kennard 1970), LASSO (Tibshirani 1996), Elastic Net (Zou and Hastie 2005), sure independence screening (Fan and Lv 2008), etc. However, these methods mainly concentrate on stand-alone processes, which are

inefficient when there are multiple same types of manufacturing processes connected in a network. Therefore, as the scope scales up to modeling the manufacturing network, we often assume that those connected processes have similar-but-non-identical model structures and parameters (Evgeniou and Pontil 2004a). Effectively capturing such similarities in modeling often leads to better modeling performance. To model those processes, researchers proposed transferred learning (Pan and Yang 2009) with applications in wafer defect-recognition (Yu, Shen, and Zheng 2020), wastewater treatment (Wang et al. 2018), and multi-task learning (MTL) (Obozinski, Taskar, and Jordan 2010), with applications in MRI data analysis (Gong, Ye, and Zhang 2013) and thermal field estimation for grain storage (Wang et al. 2019). As the sample size grows, however, the existing modeling methods often create significant computational and data transformation challenges since it requires to aggregate all information from each process together. Such challenges become greater as more processes are involved.

Significant efforts in modeling methods are to study how to incorporate data reduction for modeling when the sample size needs to be reduced before analysis. The most generally purposed sampling methods (not only for modeling) are uniform sampling, which extracts observations with a uniformly distributed probability. The most popular uniform sampling methods include random sampling (Clarkson and Shor 1989b, Liu, Sadygov, and Yates 2004, Chen et al. 2013) and stratified sampling (Trost 1986, Liberty, Lang, and Shmakov 2016, Wu et al. 2015). When a regression model is designated as the down-stream task following sampling or filtering, people study how to better preserve the information in the design matrix (i.e., input variable for regression) for model estimation with the least variation. The research along this direction includes information-based uniform sampling for regression (Drineas et al. 2011), leveraging-based sampling for regression (Ma and Sun 2015, Ma, Mahoney, and Yu 2015, Drineas et al. 2012), etc.

Among these works, the state-of-the-art method is information-based optimal subdata selection (IBOSS) (Wang, Yang, and Stufken 2019), which has low computational intensity and extract information-rich data subset for modeling. In the experimental study, IBOSS outperforms the past benchmark filtering methods on modeling performance. The idea of IBOSS is similar to the optimal experimental design (Kiefer 1959), which extracts the observations with extreme values to maximize the information preserved.

The major drawback of the existing filtering methods for modeling is that they are always performed before the modeling step. As a result, the pre-determined data subset sizes in the filtering step are weakly associated with modeling requirements. When using these filtering methods for modeling, the filtered data subset might not always yield satisfactory modeling performance, which should be addressed.

4.3 The Proposed Data Filtering Method

We proposed the DDFM method for modeling a manufacturing network in Fog Manufacturing that can automatically determine the filtering ratio, i.e., the percentage of observations filtered from the raw data, based on modeling performance. We assume that the K processes studied are similar-but-non-identical, so they hold similar process input-output relationships and result in similar model parameters. The commonalities among these processes, referred to as a network baseline model, can be estimated to serve as warm-start when estimating the individual model for each system to reduce the data usage in Fog Manufacturing. Denote predictors in observation i as $\boldsymbol{x}_i^k \in \mathbb{R}^{p+1}$ and the response as $y_i^k \in \mathbb{R}^1$ for Process $k = 1, \dots, K$, then we assume $\boldsymbol{x}_1^k$ and y_i^k follow a linear relationship as

$$y_i^k = \boldsymbol{x}_i^{k'} \boldsymbol{\beta}^k + \epsilon^k \, , i = 1, \dots, n_k, \tag{17}$$

where where $\boldsymbol{\beta}^k \in \mathbb{R}^{p+1}$ denote the model parameters for process k, and the residual $\epsilon^k \sim Normal(0, \sigma_k^2)$. We validated the normality assumptions in Figure C1 of the Appendix C and similarities among processes in Figure C2 of the Appendix C.

Let w_i be the indicator variable that equals 1 if observation $(\boldsymbol{x}_i^k, y_i^k)$ is filtered and 0 otherwise, the filtering for all K processes can be formulated as

$$\begin{aligned} &\min_{w_i^k, \boldsymbol{\beta}^k} \sum_{k=1}^{K} \sum_{i=1}^{n_k} w_i^k (y_i^k - \boldsymbol{x}_i^{k\prime} \boldsymbol{\beta}^k)^2 \\ &s.t. \frac{\sum_{i=1}^{n_k} w_i^k}{n_k} \leq r^k \text{ for } k = 1, \dots, K, \end{aligned} \tag{18}$$

where $0 \leq r^k \leq 1$ is the filtering ratio for Process k. However, several challenges exist when applying Problem (18) for modeling with filtered data. The first one is that practitioners have to manually define $r^k, k = 1, \dots, K$ based on their subjective judgment. Furthermore, the above formulation does not consider the similarities of the models estimated for different manufacturing processes. Enforcing the similarities for model parameter estimation often leads to improved modeling performance (Gross and Tibshirani 2016).

To address the above challenges, we propose to estimate model parameters based on filtered data with automatically determined filtering ratio r^k for different processes in the network. To enforce model similarities among processes, we propose to incorporate commonalities of the model parameters shared across processes into the estimation. We denote such commonalities as the network baseline model with parameters $\boldsymbol{\beta}' \in \mathbb{R}^{p+1}$. Then, we enforce the similarities between the model parameter vector of each process $\boldsymbol{\beta}^k$ and $\boldsymbol{\beta}'$, so that $\boldsymbol{\beta}'$ warm-starts estimation of $\boldsymbol{\beta}^k$ to

reduce the sample size required. In summary, we refer to the proposed method as the distributed data filtering and modeling (DDFM) with the objective function presented in Problem (18). The estimation for the network baseline model $\boldsymbol{\beta}'$ and the model for each process, $\boldsymbol{\beta}^1, \dots, \boldsymbol{\beta}^K$ can be written as:

$$\underset{\boldsymbol{\beta}', \boldsymbol{\beta}^1, \dots, \boldsymbol{\beta}^K}{\arg\min} \sum_{k=1}^{K} \sum_{i=1}^{n_k} w_i^k \, (y_i^k - \boldsymbol{x}_i^{k\prime} \boldsymbol{\beta}^k)^2 + \lambda_1 \sum_{k=1}^{K} \|\boldsymbol{\beta}^k - \boldsymbol{\beta}'\|_1 + \lambda_2 \sum_{k=1}^{K} \sum_{i=1}^{n_k} w_i^k \qquad (19)$$

where $|\cdot|_1$ is the l_1 norm, λ_1 and λ_2 are the tunable hyperparameters to balance the weight of three terms in Problem (19). Incorporating the l_1-norm enforces the difference between the network baseline model parameters ($\boldsymbol{\beta}'$) and individual process model parameters ($\boldsymbol{\beta}^k$).

Although Problem (19) is well-defined, several issues remain challenging. The first one is that obtaining an exact solution of Problem (19) can be computationally expensive or infeasible, mainly due to the presence of determining all binary variables of w_i^k (Burdakov, Kanzow, and Schwartz 2015). Furthermore, the tuning process of λ_1 and λ_2 can also become computationally expensive (e.g., through cross-validation). To address these challenges, we proposed a scalable heuristic, suitable for distributed computation in Fog Manufacturing, for the DDFM method.

The heuristic algorithm contains two steps: 1) estimating the network baseline model parameter $\boldsymbol{\beta}'$; and 2) estimating the individual model parameter $\boldsymbol{\beta}^k$ for each process. In both steps, the heuristic adopts IBOSS (Wang, Yang, and Stufken 2019), the state-of-the-art model-based filtering for modeling method, to reduce the computational complexity. For Process k, IBOSS proceeds as follows: For each variable, it selects $2s$ observations with s smallest and s largest values of the underlying variable, then it removes the previously selected samples and moves to

the next variable. IBOSS repeats such a selection process for all p variables (without the intercept term). If there are $\hat{n}^k$ amount of observations to be extracted from Process k, then $s = \frac{\hat{n}^k}{2p}$.

When estimating $\boldsymbol{\beta}'$, the heuristic uses data-shared Lasso (Gross and Tibshirani 2016), which proposes that any $\boldsymbol{\beta}^k$ can be generated as $\boldsymbol{\beta}^k = \boldsymbol{\beta}' + \Delta_k$. Such a formulation makes $\boldsymbol{\beta}'$ the ideal estimator for the network baseline model. Denote the filtered data set from each process as $(\hat{X}^k \in \mathbb{R}^{\hat{n}^k \times (p+1)}, \hat{\boldsymbol{y}}^k \in \mathbb{R}^{\hat{n}^k})$ from the raw data $(X^k \in \mathbb{R}^{n_k \times (p+1)}, \boldsymbol{y}^k \in \mathbb{R}^{n_k})$, $\boldsymbol{\beta}'$ is estimated as part of $\boldsymbol{\beta} = (\boldsymbol{\beta}', \boldsymbol{\beta}^1, \dots, \boldsymbol{\beta}^K)$ as

$$\boldsymbol{\beta} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2} \left\| \hat{y} - \hat{Z}\boldsymbol{\beta} \right\|_2^2 + \|\boldsymbol{\beta}\|_1, \tag{20}$$

where $\hat{Z} = \begin{pmatrix} \hat{X}^1 & \frac{1}{K}\hat{X}^1 & \dots & 0 \\ \vdots & & & \\ \hat{X}^K & 0 & \dots & \frac{1}{K}\hat{X}^K \end{pmatrix}$, $\hat{\boldsymbol{y}} = \left(\hat{\boldsymbol{y}}^{1\prime}, \dots, \hat{\boldsymbol{y}}^{K\prime}\right)'$, where is the data subset filtered from Process k. As data-shared Lasso adopts l_1-norm-based regularization, it only requires a small amount of data from each process to estimate $\boldsymbol{\beta}'$ (Gross and Tibshirani 2016). To estimate $\boldsymbol{\beta}'$ with minimal computational complexity, we adopt IBOSS with $r = 1$ for $(\hat{X}^k, \hat{\boldsymbol{y}}^k)$, where $k = 1, \dots, K$

Given $\boldsymbol{\beta}'$, the heuristic applies IBOSS to estimate $\boldsymbol{\beta}^k$ using Equation (21) as

$$\boldsymbol{\beta}^k = \underset{\boldsymbol{\beta}^k}{\operatorname{argmin}} \left\| \boldsymbol{y}^k - \hat{X}^k\boldsymbol{\beta}^k \right\|_2^2 + \|\boldsymbol{\beta}' - \boldsymbol{\beta}^k\|_1, \tag{21}$$

with iteratively increased values of s for IBOSS, until the change of $\boldsymbol{\beta}^k$ between two consecutive iterations drops below a pre-defined threshold (i.e., $(\boldsymbol{\beta}_s^k\text{-}\boldsymbol{\beta}_{s-1}^k)/\boldsymbol{\beta}_{s-1}^k < 1e^{-3}$ between iteration s and $s-1$ (Neese 2018)).

We summarized the heuristic in Algorithm 2. Here we observe that the estimation of $\boldsymbol{\beta}^k$ in Step 2 can be easily distributed to each individual process with the dedicated Fog nodes. Therefore, we can implement the proposed DDFM in the Fog Manufacturing, which will be illustrated in detail in Case Study

---

Algorithm 2. Pseudocode for the Distributed Filtering and Modeling Heuristic

---

Initialize $\boldsymbol{\beta}'$:

Step 1 (For $\boldsymbol{\beta}'$ of the network):

$$\left(\hat{X}^1, \hat{\boldsymbol{y}}^1\right), \ldots, \left(\hat{X}^K, \hat{\boldsymbol{y}}^K\right) = IBOSS(X^1, \boldsymbol{y}^1, 1), \ldots, IBOSS(X^K, \boldsymbol{y}^K, 1)$$

$$\hat{Z} = \begin{pmatrix} \hat{X}_1 & \frac{1}{K}\hat{X}_1 & \ldots & 0 \\ \vdots & & & \\ \hat{X}_K & 0 & \ldots & \frac{1}{K}\hat{X}_K \end{pmatrix}, \hat{\boldsymbol{y}} = \left(\hat{\boldsymbol{y}}^{1\prime}, \ldots, \hat{\boldsymbol{y}}^{K\prime}\right)'$$

$$\boldsymbol{\beta} = (\boldsymbol{\beta}', \boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^K) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \frac{1}{2}\left\|\hat{y} - \hat{Z}\boldsymbol{\beta}\right\|_2^2 + \|\boldsymbol{\beta}\|_1$$

Step 2 (For Process $k = 1, \ldots, K$):

$r = 1$

Do Until $(\boldsymbol{\beta}_r^k\text{-}\boldsymbol{\beta}_{r-1}^k)/\boldsymbol{\beta}_{r-1}^k < 1e^{-3}$:

$\hat{X}^k, \hat{\boldsymbol{y}}^k = IBOSS(X^k, \boldsymbol{y}^k, r)$

$\boldsymbol{\beta}^k = argmin_{\boldsymbol{\beta}}\left\|\boldsymbol{y}^k - \hat{X}^k\boldsymbol{\beta}\right\|_2^2 + \|\boldsymbol{\beta}' - \boldsymbol{\beta}\|_1$

$r = r + 1$

4.4 Simulation

The objective of the simulation is to evaluate the modeling performance of the DDFM method in various simulation settings and compare the performance with a few benchmark methods. In particular, we considered four settings by varying the sample size of the raw data in each process (n^k), the process similarities of the underlying model parameters (τ), the sparsity of model parameters (ρ), and the signal-to-noise ratios (γ), summarized in Table 4-1. Inspired by Wang, Yang, and Stufken (2019), for each process, we simulated input variables as $X^k \in \mathbb{R}^{n^k \times (p+1)}$, with the first p variables following $Normal(\boldsymbol{\mu}, \Sigma)$, and the additional intercept variable as all one. Each element in $\boldsymbol{\mu} \in \mathbb{R}^p$ follows $Uniform([0, 1))$ and the regularized covariance matrix $\Sigma = [\sigma_{ij}]$ with $\sigma_{ii} = 1$ and $\sigma_{ij} = 0.3$ when $i \neq j$.

To generate the response variable $\boldsymbol{y}^k \in \mathbb{R}^{n^k}$ for Process k, we adopt a linear model $\boldsymbol{y}^k = X^k\boldsymbol{\beta} + \boldsymbol{\epsilon}^k$, where $\boldsymbol{\beta}^k \in \mathbb{R}^{p+1}$, $\boldsymbol{\epsilon}^k \in \mathbb{R}^{n^k}$ as the residual term with each element following $N(0, \sigma_k^2)$ in independent and identically distributed. We used the similarity parameter τ to control the similarity of the model parameters among different processes. Specifically, we first simulated a network baseline model parameters $\boldsymbol{\beta'} \in \mathbb{R}^{p+1}$, with each element following $Normal(0,1)$. Then $\boldsymbol{\beta}^k = \boldsymbol{\beta'} + \frac{1}{\tau}\boldsymbol{s}$, where $\boldsymbol{s} = \left(sign(\beta_0')c_0, \ldots, sign(\beta_p')c_p\right)'$, $c_i \sim Uniform([0, 1))$, and $sign(\cdot)$ returns 1 when the value is larger than 0 and returns -1 otherwise.

F For each Process k, we set ρ^k as the model sparsity, representing the percentage of significant variables in the model (non-zero elements in $\boldsymbol{\beta}^k$). All processes share the same variable significance pattern as the connected processes are similar in manufacturing. When generating $\boldsymbol{\epsilon}^k$, we control the signal-to-noise ratio as $\gamma = \frac{var(X^k\boldsymbol{\beta}^k)}{var(\boldsymbol{\epsilon}^k)}$. In this simulation, $p = 20$, which is similar to the case study. The values of simulation setting variables are summarized in Table 4-1

Table 4-1. Simulation Design Table

Parameters	Low	High
Sample Size of Raw Data (n^k)	5000	10000
Process Similarity (τ)	5	10
Model Sparsity (ρ^k)	0.3	0.7
Signal-to-Noise Ratio (γ)	3	7

We compared the modeling performance of the proposed DDFM method with data randomly partitioned into training and testing sets in 90%-10% split for each process over 100 replications. Two performance measures on model parameter estimation error and prediction error on the testing sets are adopted. The first performance measure is the root-mean-squared errors between the simulated underlying true parameters and estimated for all processes. The second performance measure is root-mean-squared prediction error ($RMSPE$) when the trained model predicts the $\boldsymbol{y}^k$ of the 10% testing data in each process.

We considered several benchmark methods, which first extract data from one process at a time, and then estimate their model parameters with LASSO regression based on the filtered dataset (Tibshirani 1996). The proposed DDFM method can automatically determine the number of observations preserved during filtering. Therefore, to ensure a fair performance comparison, each benchmark method will preserve the same sample size. For example, if the proposed method extracts N observations in total from K processes, then each benchmark method extracts $\frac{N}{K}$ observations to generate as the filtered dataset. The benchmark methods are IBOSS (Wang, Yang, and Stufken 2019), random sampling (Liu, Sadygov, and Yates 2004), which extracts observations randomly from the full data set, and stratified sampling (Liberty, Lang, and Shmakov 2016), which partitions data into the equally spaced and non-overlapping windows, and then extracts the first observation in each window.

We summarize the results of model parameter recovery and response prediction accuracy in Table 4-2 and Table 4-3 with n^k and $\tilde{n}^k$ represents the full and the average filtered data size for each process. The best results in each simulation setting are highlighted in bold. Both tables show that the proposed method significantly outperformed all the benchmark methods in all simulation settings. It is because these benchmark methods do not consider the model similarities among different processes. For the impact of each setting, we can observe that the increase of sample size (n^k) slightly improves the performance of all methods as a larger raw data size reduces the variance of estimated model parameters (Wang, Yang, and Stufken 2019). When processes are more similar to each other (τ grew larger), the modeling errors were smaller for the proposed method, since it explicitly enforces the similarities of model parameters estimated for different processes. As we increased the model sparsity (ρ) and generated a more complex model, the errors grew larger for all methods. The proposed method, however, yielded superior performance on both levels of ρ's. Similarly, the increase of signal-to-noise ratio (γ) improved the performance for all methods, as there are less uncertainty in the data for improving modeling accuracy.

Lastly, as the proposed method enforces the process similarities, it may not work well when the processes are less similar to each other. To test the boundary of the proposed method in this simulation, we additionally introduce two extreme cases of when the processes are very different from each other: $\tau = 0.1$ and $\tau = 1$ ($n = 10000$, $\rho = 0.7$, $\gamma = 10$) with result summarized in Table C-1 and Table C-2 of Appendix C. We can see the proposed method either yielded comparable or worse performance of the benchmark method, which applies IBOSS and models the filtered data for one system at a time. When processes are very different from each other, we conclude that enforcing the model parameter similarities during estimation will have a negative impact on modelling performance.

Table 4-2. Mean and standard errors (within parenthesis) of model parameter recovery errors over 100 simulation replications

n	$\hat{n}$	τ	ρ	γ	Proposed	IBOSS	Random	Stratified	Cluster-Based
5000	86.32	5	0.30	3	**0.06 (<0.01)**	0.17 (<0.01)	0.20 (<0.01)	0.20 (<0.01)	0.19 (<0.01)
5000	86.68	5	0.30	7	**0.05 (<0.01)**	0.14 (<0.01)	0.17 (<0.01)	0.16 (<0.01)	0.16 (<0.01)
5000	84.80	5	0.70	3	**0.11 (<0.01)**	0.27 (<0.01)	0.31 (<0.01)	0.31 (<0.01)	0.31 (<0.01)
5000	85.64	5	0.70	7	**0.08 (<0.01)**	0.21 (<0.01)	0.24 (<0.01)	0.24 (<0.01)	0.24 (<0.01)
5000	82.12	10	0.30	3	**0.05 (<0.01)**	0.18 (<0.01)	0.21 (<0.01)	0.21 (<0.01)	0.21 (<0.01)
5000	81.64	10	0.30	7	**0.04 (<0.01)**	0.14 (<0.01)	0.16 (<0.01)	0.16 (<0.01)	0.17 (<0.01)
5000	80.92	10	0.70	3	**0.09 (<0.01)**	0.26 (<0.01)	0.30 (<0.01)	0.30 (<0.01)	0.30 (<0.01)
5000	81.60	10	0.70	7	**0.07 (<0.01)**	0.21 (<0.01)	0.24 (<0.01)	0.24 (<0.01)	0.24 (<0.01)
10000	85.68	5	0.30	3	**0.06 (<0.01)**	0.17 (<0.01)	0.20 (<0.01)	0.20 (<0.01)	0.20 (<0.01)
10000	85.56	5	0.30	7	**0.05 (<0.01)**	0.14 (<0.01)	0.17 (<0.01)	0.16 (<0.01)	0.17 (<0.01)
10000	83.76	5	0.70	3	**0.10 (<0.01)**	0.26 (<0.01)	0.31 (<0.01)	0.31 (<0.01)	0.31 (<0.01)
10000	84.52	5	0.70	7	**0.08 (<0.01)**	0.20 (<0.01)	0.24 (<0.01)	0.24 (<0.01)	0.24 (<0.01)
10000	81.08	10	0.30	3	**0.05 (<0.01)**	0.18 (<0.01)	0.20 (<0.01)	0.20 (<0.01)	0.20 (<0.01)
10000	82.00	10	0.30	7	**0.04 (<0.01)**	0.13 (<0.01)	0.15 (<0.01)	0.15 (<0.01)	0.15 (<0.01)
10000	81.36	10	0.70	3	**0.08 (<0.01)**	0.25 (<0.01)	0.30 (<0.01)	0.30 (<0.01)	0.30 (<0.01)
10000	81.80	10	0.70	7	**0.06 (<0.01)**	0.20 (<0.01)	0.23 (<0.01)	0.23 (<0.01)	0.23 (<0.01)

Table 4-3. Mean and standard errors (within parenthesis) of RMSPE over 100 simulation replications

n	$\hat{n}$	τ	ρ	γ	Proposed	IBOSS	Random	Stratified	Cluster-Based
5000	86.32	5	0.30	3	**1.19 (0.04)**	1.24 (0.04)	1.28 (0.04)	1.28 (0.04)	1.28 (0.04)
5000	86.68	5	0.30	7	**0.80 (0.02)**	0.84 (0.03)	0.86 (0.03)	0.86 (0.03)	0.86 (0.03)
5000	84.80	5	0.70	3	**1.83 (0.04)**	1.96 (0.05)	2.03 (0.05)	2.03 (0.05)	2.03 (0.05)
5000	85.64	5	0.70	7	**1.30 (0.03)**	1.39 (0.04)	1.44 (0.04)	1.44 (0.04)	1.44 (0.04)
5000	82.12	10	0.30	3	**1.11 (0.04)**	1.18 (0.04)	1.21 (0.04)	1.21 (0.04)	1.21 (0.04)
5000	81.64	10	0.30	7	**0.79 (0.02)**	0.84 (0.03)	0.87 (0.03)	0.87 (0.03)	0.87 (0.03)
5000	80.92	10	0.70	3	**1.71 (0.04)**	1.85 (0.05)	1.92 (0.05)	1.92 (0.05)	1.92 (0.05)
5000	81.60	10	0.70	7	**1.24 (0.03)**	1.34 (0.03)	1.40 (0.03)	1.40 (0.03)	1.40 (0.03)
10000	85.68	5	0.30	3	**1.13 (0.03)**	1.19 (0.04)	1.22 (0.04)	1.22 (0.04)	1.23 (0.04)
10000	85.56	5	0.30	7	**0.85 (0.02)**	0.89 (0.02)	0.92 (0.02)	0.93 (0.03)	0.92 (0.02)
10000	83.76	5	0.70	3	**1.78 (0.05)**	1.90 (0.05)	1.98 (0.05)	1.98 (0.05)	1.98 (0.05)
10000	84.52	5	0.70	7	**1.28 (0.03)**	1.37 (0.04)	1.43 (0.04)	1.43 (0.04)	1.43 (0.04)
10000	81.08	10	0.30	3	**1.17 (0.04)**	1.24 (0.04)	1.28 (0.04)	1.28 (0.04)	1.28 (0.04)
10000	82.00	10	0.30	7	**0.76 (0.02)**	0.80 (0.02)	0.83 (0.03)	0.83 (0.03)	0.83 (0.03)
10000	81.36	10	0.70	3	**1.71 (0.04)**	1.84 (0.05)	1.92 (0.05)	1.92 (0.05)	1.92 (0.05)
10000	81.80	10	0.70	7	**1.20 (0.03)**	1.30 (0.03)	1.35 (0.03)	1.35 (0.03)	1.35 (0.03)

4.5 Case Study

In the case study, we proposed and implemented testbed for an ingot growth manufacturing with a Fog Computing system (Zhang et al. 2019) using DDFM. Compared with Cloud Computing, Fog Computing allows various data analysis tasks to be distributed and executed at the local computation units (i.e., Fog nodes) of each manufacturing process. As a result, it can support the corresponding data analysis and decision-making with low computational time latency (Wang et al. 2020).

The architecture of the testbed is shown in Figure 4-2. There are three hierarchical layers in the testbed. From top to bottom, the first layer is the Cloud-based orchestrator, which directly controls the communication flow (i.e., data flow and computation flow) (Zhang et al. 2019). Moreover, the corresponding computation and communication utilization information can be recorded to monitor the status of the process on the orchestrator (Wang, Zhang, and Jin 2020). In the proposed testbed, a high-performance workstation (CPU: Intel i7-6700K) is employed as the orchestrator.

The second layer consists of a group of Fog nodes. In this layer, we adopt five laptops (CPU1: Intel i7-4720HQ; CPU2: i7-5600U; CPU3: i7-8705G; CPU4: i7-6600u; CPU5: i7-6820HQ) as five Fog nodes. Each Fog node can collect the data from the manufacturing process and store the raw data in a local database for further data analysis. The orchestrator can coordinate the Fog nodes to execute computation tasks with a specific dataset simultaneously (Pemmasani 2018).

The third layer is the raw data source. In this case study, we adopt the silicon ingot manufacturing network (Sun et al. 2016), containing the production data from five similar-but-non-identical manufacturing furnaces. To prevent the preliminary study from interrupting physical

manufacturing processes, we collected and stored data in advance at each perspective Fog node (Sun et al. 2016).

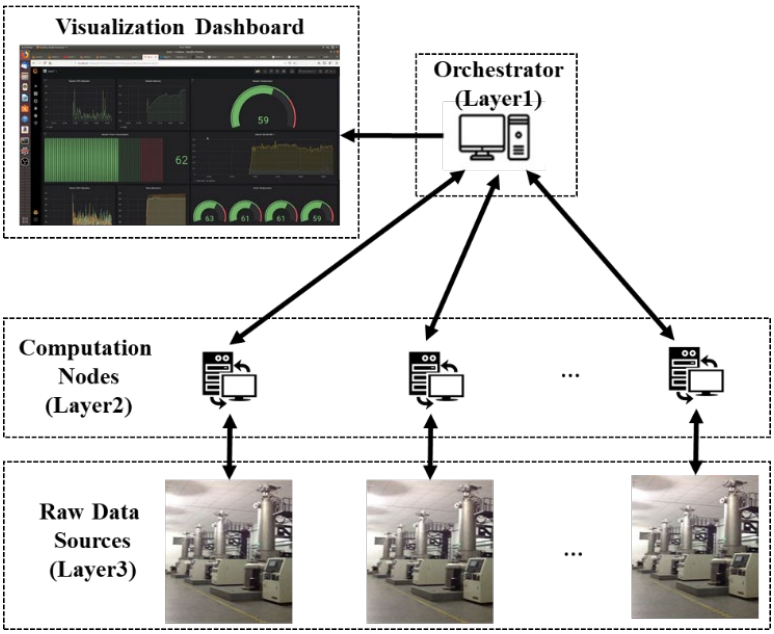


Figure 4-2 Architecture of the Fog Manufacturing Testbed (Wang, Zhang, and Jin 2020)

Figure 4-2 shows the schematic of the computation flow for the proposed DDFM method on the testbed. Step 1 of Algorithm 2 is implemented on the Cloud (the orchestrator) and Step 2 of Algorithm 2 is implemented at each individual Fog node for each process in parallel.

To evaluate the performance of the proposed method, we adopt the same benchmark methods from the simulation study, including IBOSS, random sampling, and stratified sampling. We evaluated the performance of the proposed filtering method with data randomly partitioned into training and test sets in 90%-10% split for each process over 100 replications. The prediction performance of the model is evaluated with root-mean-squared prediction error (RMSPE) and its standard error (SE) over 100 replications.

The prediction RMSPEs and corresponding SE from both Cloud and Fog scenarios are summarized in Table 4-4 The table shows that the proposed method yields the best performance compared with other benchmarks. Furthermore, the Cloud and the Fog computing yield

comparable performance as the Fog Computing executes the same computational tasks as the Cloud version does

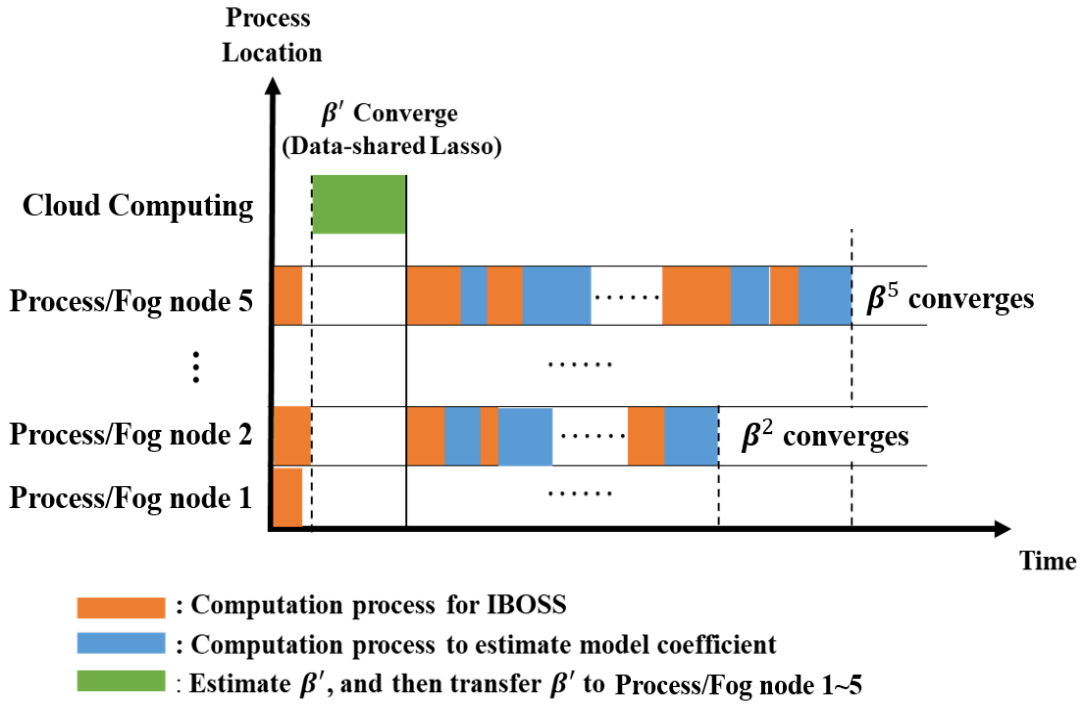


Figure 4-3. Schematic Diagram of Computation Flow

To demonstrate the advantages of Fog Computing compared with using the Cloud Computing only, the proposed method is also executed in the Cloud (i.e., the orchestrator of our testbed, which has better computing power than each Fog node does). In this case, all computation tasks for different processes will be executed sequentially. To compare the performance of these two approaches, six performance metrics are calculated following Zhang et al. (2019) in Figure 4-4: 1) Scaled RMSPEs; 2).Scaled standard error (SE) of RMSPEs among replications; 3) time latency; 4) redundancy; 5) computation utilization; 6) bandwidth utilization. The RMSPE and SE of the proposed methods are rescaled from Table 4-4 to a range between 0 and 1 for better illustration. Furthermore, Time latency represents the total time consumption to finish the corresponding computation task. Redundancy represents how fast a single process can perform its

assigned computing task twice so that it will complete the computing tasks during the downtime of other processes (Zhang et al. 2019). Computing utilization represents the average CPU utilization while performing the computing task. Bandwidth utilization represents how much communication bandwidth is required among Fog nodes and the orchestrator. Similar to the metric 1) and 2), metric 3) to metric 6) were also scaled to the same range from 0 to 1 representing the worst (the center) to the best values (the edge) in Figure 4-4 following the literature (Zhang et al. 2019).

From the results shown in Table 4-4, it can be observed that the scaled RMSPE and the scaled SE are comparable in two scenarios. Fog Computing has much better performance than Cloud Computing for both time latency and redundancy. This is because the computation task is assigned to the five Fog nodes and executed in parallel in the Fog Computing, which can provide a responsive computation service. On the other hand, Cloud Computing has better CPU utilization performance than the Fog computing, since the CPU on the orchestrator is much better than the CPU on the Fog nodes so that more resources can be preserved for other tasks. Moreover, Fog and Cloud computing have a comparable bandwidth utilization performance, because the total amount of data transmitted between Fog nodes and the orchestrator is negligible due to the application of the DDFS methods to reduce the sample size.

Table 4-4. Performances of Data Filtering

		Proposed	IBOSS	Random	Stratified	Cluster-based
Distributed	RMSE	**0.493**	0.575	0.527	0.516	0.527
	SE	**(0.001)**	(0.001)	(0.003)	(0.003)	(0.002)
Centralized	RMSE	**0.494**	0.572	0.516	0.520	0.523
	SE	**(0.001)**	(0.001)	(0.003)	(0.003)	(0.002)

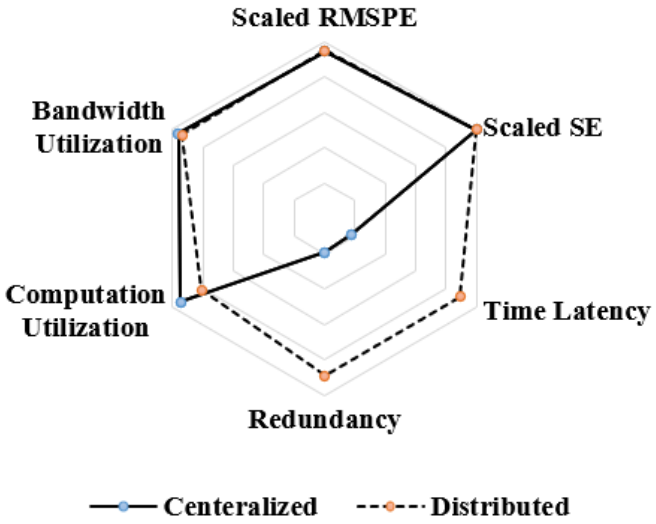


Figure 4-4. Radar Chart for Six Metrics

The real data generated from the case study also be utilized to validate the model assumptions in Appendix C. From Figure C1, we can observe that there is little variation pattern in the residuals to be explained and no presence of non-normality. We conclude that the selection of the linear model in the proposed method is appropriate and there is no violation of the normality assumption of the residuals. Moreover, Figure C2 shows the number of times that each variable was selected over 100 Replications, which validates that the processes modeled for ingot growth manufacturing are similar-but-non-identical. We can also observe that the Process 2 has different variable relationships from other processes. After investigation of manufacturing log, we found out that it suffers from failing motors, which produced non-conforming ingots and result in irregular *in situ* sensing data. Lastly, we found that Variable 17, which is the resistance of the furnace heater, has been frequently selected for all system. The resistance is the key factor in the ingot growth manufacturing (Jouini et al. 2012).

Although the proposed DDFS method in Fog Manufacturing outperforms in several runtime metrics, one limitation is that the DDFS method requires significantly more iterations to converge. As the Fog nodes have inferior computational power, the time latency for the whole

process can increase significantly due to the bottlenecks in some Fog nodes. One potential solution to address this issue is to offload the computation tasks to different Fog nodes based on their computation workload (Chen et al. 2018), such that there are more balanced computation workload in Fog Manufacturing.

4.6 Conclusion of Distributed Data Filtering and Modeling

A Fog Manufacturing applies both Fog and Cloud Computing collaboratively and integrates manufacturing equipment and processes into an interconnected network through sensing, actuation, and computation nodes. Therefore, most of the computation services in manufacturing can be provided by using both Fog and Cloud. This is important for future manufacturing, as Fog Manufacturing will provide tremendous promise on reliable and responsive computation services, with the potential for privacy preservation to process data in local computation units. However, due to the relatively limited communication bandwidth to Cloud and computation capabilities of Fog nodes, a large amount of data from the manufacturing network will lead to information redundancy and significant computation time latency for data analysis.

In this paper, we propose a distributed data filtering (i.e., subsampling) method to extract small but informative data subsets from the raw data, considering the similarity of the interconnected manufacturing processes. The filtered data from each manufacturing process will later be transmitted to Cloud for manufacturing process modeling. A simulation study and a real Fog Manufacturing testbed for an ingot growth manufacturing have been used to validate the proposed distributed data filtering and modeling method. The results indicate that the proposed distributed data filtering and modeling method will not only reduce the sample size and ensure the modeling performance, but also improve the performance of the runtime metrics of the computation, such as time latency and communication bandwidth utilization, in Fog

Manufacturing. We would like to emphasize that the proposed method relies on IBOSS to perform data reduction, which extracting the observations with extreme values (i.e. the corner points in design of experiment). In this case, if a model, such as Kriging (Van Beers and Kleijnen 2004), relies on data from space-filling design, instead of corner points, then the proposed is not guaranteed to have superior performance. When filtering does not work well or significantly impacts the modelling performance and prediction accuracy, then we may have to rely Cloud devices as Fog nodes may not have sufficient computational power. Furthermore, the extreme case in the simulation, when the processes are very different from each other, show that filtering and modeling each system individually yield the best result. Furthermore, we would like to re-iterate the observation from Gross and Tibshirani (2016) to remind readers that: no matter how similar that the processes are to each other, modelling a process based on its own data always yields the best performance given adequate amount of data collect.

There are several directions in future work. First, the DDFS method will be extended to other manufacturing data analytical methods, such as process monitoring, prognosis and diagnosis. For example, one can study how to jointly filter and monitor multiple manufacturing processes when they generate a large amount of data, simultaneously (Jin and Liu 2013). Second, we will integrate the filtering and modeling based on heterogeneous types of data sets in manufacturing, such as experimental data and observational data sets (Jin and Deng 2015), and qualitative and quantitative data sets (Deng and Jin 2015).

Chapter 5. Conclusion and Future Research

The advancement of Internet-of-Things (IoT) integrates manufacturing processes and equipment into a network. As there is an increasing amount of processes connected to the manufacturing network, we can quickly obtain a large amount of machine setting, *in situ* sensing, quality inspection data for analysis. However, the data collected from are often large and vary from each other due to the heterogeneities among processes. These characteristics often become challenges for efficient data-driven decision making (Sarin et al. 2016). Specifically, this dissertation addressed the following challenging in data modeling for smart manufacturing networks:

1) In Chapter 2. , we proposed a filtering method and new criteria (FIC) to facilitate data analysis by selecting a small but effective data subset. Specifically, the proposed method partitions raw data into clusters and proportionally extracts a data subset from each cluster based on the optimal filter ratio determined by FIC. The proposed cluster-based data filtering method outperformed the benchmark filtering methods on information loss, the modeling goodness-of-fit on training data, and prediction error on testing data sets for both the case study and the simulation. Furthermore, the new filtering ratio selection criteria (FIC) has shown its effectiveness in terms of balancing the trade-offs between the size of data preserved in filtering and the quality of the filtered data.

2) In Chapter 3, we incorporated a variation decomposition approach to model the latent variation, which cannot be captured by the observable variables, into the feature-based MTL. Furthermore, we generalized the proposed methodology into a multivariate response and MTL model, which is capable of modeling multivariate responses monitored across multiple similar-but-non-identical processes (e.g., furnaces). The simulation study and the case study have shown that MTL-LVD can not only recover the latent variation of data from similar-but-non-identical process but also offer significantly better accuracy on predicting the multivariate response.

3) Chapter 4 proposed a distributed data filtering algorithm for a manufacturing network to extract small but effective data subsets in different processes for data analysis. Specifically, this work combined the objectives of filtering and modeling in one unified framework. The proposed method iteratively incorporates the IBOSS and utilize the model similarities among processes for efficient data analysis. The proposed method generated better modeling performance with the same filtering ratio compared with benchmark methods. Furthermore, we proposed a Fog-based distributed filtering and modeling process based on the proposed algorithm. The case study from the ingot growth network shows that the proposed Fog-manufacturing generates an accurate predictive model while greatly reducing the computational time latency compared to centralized (Cloud) computing.

The proposed method can effectively model similar-but-non-identical processes in a smart manufacturing network, which can generate datasets that are too large to be analyzed efficiently in their raw sizes. The proposed methods have been validated in the advanced manufacturing processes, including babycare manufacturing, silicon ingot manufacturing.

There are several directions that deserve further investigation:

1) Data analysis under big data environments have become prevalent and even sometimes necessary because of the performance requirement. However, efficiently utilizing big data in an accurate and computationally efficient manner is still a challenging problem. The proposed method can also facilitate internet-of-things (IoT)-based data collection, communication, storage, and analysis. For example, as the inline data de-duplication (real-time data filtering for continuous data streaming), has become more prevalent in recent years (Zhou, Liu, and Li 2013), future work will focus on the conversion of the current offline data filtering to online.

2) Chapter 3 focuses on recovering model parameters for similar-but-non-identical processes with MTL-LVD. The model parameters reflect the commonality of machines, and they can serve as the baseline for monitoring process irregularities, such as the deviation of input-output relationships in production. It will be valuable research to investigate the joint process monitoring method for similar-but-non-identical processes in a smart manufacturing network.
3) We can study how to jointly filtering and monitoring for multiple manufacturing processes as there are a large amount of incoming streaming sensor data (Jin and Liu 2013). Such a networked process monitoring is beneficial for quality control in large-scale and continuous manufacturing networks, such as the previously investigated ingot growth process.

References

Ando, R. K., and Zhang, T. (2005), "A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data," *Journal of Machine Learning Research*, 6, 1817-1853.

Baraniuk, R. G. (2007), "Compressive Sensing," *IEEE Signal Processing Magazine*, 24, 1-9.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012), "Fog Computing and Its Role in the Internet of Things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13-16.

Bozdogan, H. (1987), "Model Selection and Akaike's Information Criterion: The General Theory and Its Analytical Extensions," *Psychometrika*, 52, 345-370.

Brar, R., and Singh, B. (2013), "A Survey on Different Compression Techniques and Bit Reduction Algorithm for Compression of Text Data," *International Journal of Advanced Research in Computer Science and Software Engineering (IJARCSSE)*, 3, 1-7.

Burdakov, O., Kanzow, C., and Schwartz, A. 2015. "On a Reformulation of Mathematical Programs with Cardinality Constraints." In *Advances in Global Optimization*, 3-14. Springer.

Cameron, A. C., and Windmeijer, F. A. (1996), "R-Squared Measures for Count Data Regression Models with Applications to Health-Care Utilization," *Journal of Business & Economic Statistics*, 14, 209-220.

Cao, J., Yu, P., Ma, M., and Gao, W. (2018), "Fast Authentication and Data Transfer Scheme for Massive Nb-Iot Devices in 3gpp 5g Network," *IEEE Internet of Things Journal*, 6, 1561-1575.

Chen, J., Tang, L., Liu, J., and Ye, J. (2009), "A Convex Formulation for Learning Shared Structures from Multiple Tasks," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 137-144.

Chen, W., Gao, S., Chen, C.-H., and Shi, L. (2013), "An Optimal Sample Allocation Strategy for Partition-Based Random Search," *IEEE Transactions on Automation Science and Engineering*, 11, 177-186.

Chen, X., and Jin, R. (2018), "Data Fusion Pipelines for Autonomous Smart Manufacturing," in *Proceedings of 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pp. 1203-1208.

Chen, X., Wang, L., Wang, C., and Jin, R. (2018), "Predictive Offloading in Mobile-Fog-Cloud Enabled Cyber-Manufacturing Systems," in *2018 IEEE Industrial Cyber-Physical Systems (ICPS)*, pp. 167-172.

Chouldechova, A., and Hastie, T. (2015), "Generalized Additive Model Selection," *arXiv preprint arXiv:1506.03850*.

Clarkson, K. L., and Shor, P. W. (1989a), "Applications of Random Sampling in Computational Geometry," *Discrete & Computational Geometry*, 4, 387-421.

Clarkson, K. L., and Shor, P. W. (1989b), "Applications of Random Sampling in Computational Geometry, Ii," *Discrete & Computational Geometry*, 4, 387-421.

Deng, X., and Jin, R. (2015), "Qq Models: Joint Modeling for Quantitative and Qualitative Quality Responses in Manufacturing Systems," *Technometrics*, 57, 320-331.

Dikmen, M., Zhan, Y., and Zhou, X. 2012. Joint Detection and Localization of Multiple Anatomical Landmarks through Learning. Google Patents.

Drineas, P., Magdon-Ismail, M., Mahoney, M. W., and Woodruff, D. P. (2012), "Fast Approximation of Matrix Coherence and Statistical Leverage," *The Journal of Machine Learning Research*, 13, 3475-3506.

Drineas, P., Mahoney, M. W., Muthukrishnan, S., and Sarlós, T. (2011), "Faster Least Squares Approximation," *Numerische mathematik*, 117, 219-249.

Du, J., Zhang, X., and Shi, J. (2018), "A Physics-Specific Change Point Detection Method Using Torque Signals in Pipe Tightening Processes," *IEEE Transactions on Automation Science and Engineering*, 16, 1289-1300.

Evgeniou, T., and Pontil, M. (2004a), "Regularized Multi--Task Learning," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 109-117.

Evgeniou, T., and Pontil, M. (2004b), "Regularized Multi-Task Learning," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 109-117.

Fan, J., and Lv, J. (2008), "Sure Independence Screening for Ultrahigh Dimensional Feature Space," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70, 849-911.

Farach, M., and Thorup, M. (1998), "String Matching in Lempel—Ziv Compressed Strings," *Algorithmica*, 20, 388-404.

Fisher, G., Seacrist, M. R., and Standley, R. W. (2012), "Silicon Crystal Growth and Wafer Technologies," *Proceedings of the IEEE*, 100, 1454-1474.

Fitzgibbon, L. J., Dowe, D. L., and Allison, L. (2002), "Change-Point Estimation Using New Minimum Message Length Approximations," in *Proceedings of 2002 Pacific Rim International Conference on Artificial Intelligence*, pp. 244-254.

Gelman, A., and Pardoe, I. (2006), "Bayesian Measures of Explained Variance and Pooling in Multilevel (Hierarchical) Models," *Technometrics*, 48, 241-251.

Gong, P., Ye, J., and Zhang, C. (2013), "Multi-Stage Multi-Task Feature Learning," *The Journal of Machine Learning Research*, 14, 2979-3010.

Gray, R. M. 2011. *Entropy and Information Theory*: Springer Science & Business Media.

Gross, S. M., and Tibshirani, R. (2016), "Data Shared Lasso: A Novel Tool to Discover Uplift," *Computational statistics & data analysis*, 101, 226-235.

Guralnik, V., and Srivastava, J. (1999), "Event Detection from Time Series Data," in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 33-42.

Hegazy, T., and Hefeeda, M. (2014), "Industrial Automation as a Cloud Service," *IEEE Transactions on Parallel and Distributed Systems*, 26, 2750-2763.

Hoerl, A. E., and Kennard, R. W. (1970), "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics*, 12, 55-67.

Jacob, L., Vert, P., and Bach, F. R. (2009), "Clustered Multi-Task Learning: A Convex Formulation," in *Proceedings of 2009 Advances in Neural Information Processing Systems*, pp. 745-752.

James, G. M., Hastie, T. J., and Sugar, C. A. (2000), "Principal Component Models for Sparse Functional Data," *Biometrika*, 87, 587-602.

James, W., and Stein, C. 1992. "Estimation with Quadratic Loss." In *Breakthroughs in Statistics*, 443-460. Springer.

Jin, R., and Deng, X. (2015), "Ensemble Modeling for Data Fusion in Manufacturing Process Scale-Up," *IIE Transactions*, 47, 203-214.

Jin, R., Deng, X., Chen, X., Zhu, L., and Zhang, J. (2019), "Dynamic Quality-Process Model in Consideration of Equipment Degradation," *Journal of Quality Technology*, 51, 1-13.

Jin, R., and Liu, K. (2013), "Multimode Variation Modeling and Process Monitoring for Serial-Parallel Multistage Manufacturing Processes," *Iie Transactions*, 45, 617-629.

Jin, R., Yan, H., and Lee, D. 2020. Fmrg: Pipeline Recommendation, Adaptive Optimization, and Privacy-Preserving Computation for Future Fog Manufacturing (under Review). NSF.

Jouini, A., Ponthenier, D., Lignier, H., Enjalbert, N., Marie, B., Drevet, B., Pihan, E., Cayron, C., Lafford, T., and Camel, D. (2012), "Improved Multicrystalline Silicon Ingot Crystal Quality through Seed Growth for High Efficiency Solar Cells," *Progress in Photovoltaics: research and applications*, 20, 735-746.

Kang, Z., Grauman, K., and Sha, F. (2011), "Learning with Whom to Share in Multi-Task Feature Learning," in *Proceedings of the 28th International Conference on Machine Learning*, pp. 521-528.

Kaur, A., Sethi, N. S., and Singh, H. (2015), "A Review on Data Compression Techniques," *International Journal of Advanced Research in Computer Science and Software Engineering*, 5, 1-7.

Kenett, R. S., and Shmueli, G. (2014), "On Information Quality," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 177, 3-38.

Kiefer, J. (1959), "Optimum Experimental Designs," *Journal of the Royal Statistical Society: Series B (Methodological)*, 21, 272-304.

Li, Y., Sun, H., Deng, X., Zhang, C., Wang, H.-P., and Jin, R. (2019), "Manufacturing Quality Prediction Using Smooth Spatial Variable Selection Estimator with Applications in Aerosol Jet® Printed Electronics Manufacturing," *IISE Transactions*, just-accepted, 1-17.

Liberty, E., Lang, K., and Shmakov, K. (2016), "Stratified Sampling Meets Machine Learning," in *Proceedings of 2016 International Conference on Machine Learning*, pp. 2320-2329.

Liu, H., Sadygov, R. G., and Yates, J. R. (2004), "A Model for Random Sampling and Estimation of Relative Protein Abundance in Shotgun Proteomics," *Analytical chemistry*, 76, 4193-4201.

Ma, P., Mahoney, M. W., and Yu, B. (2015), "A Statistical Perspective on Algorithmic Leveraging," *The Journal of Machine Learning Research*, 16, 861-911.

Ma, P., and Sun, X. (2015), "Leveraging for Big Data Regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, 7, 70-76.

MacQueen, J. (1967), "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pp. 281-297.

Neese, F. (2018), "Software Update: The Orca Program System, Version 4.0," *Wiley Interdisciplinary Reviews: Computational Molecular Science*, 8, e1327.

Obozinski, G., Taskar, B., and Jordan, M. I. (2010), "Joint Covariate Selection and Joint Subspace Selection for Multiple Classification Problems," *Statistics and Computing*, 20, 231-252.

Pan, S. J., and Yang, Q. (2009), "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, 22, 1345-1359.

Parameswaran, S., and Weinberger, K. Q. (2010), "Large Margin Multi-Task Metric Learning," in *Proceedings of 2010 Advances in Neural Information Processing Systems*, pp. 1867-1875.

Pascoa, S. S. 2014. *Oxygen and Related Defects in Czochralski Silicon Crowns*. MS Thesis, Norwegian University of Science and Technology, Department of Materials Science and Engineering.

Paynabar, K., Zou, C., and Qiu, P. (2016), "A Change-Point Approach for Phase-I Analysis in Multivariate Profile Monitoring and Diagnosis," *Technometrics*, 58, 191-204.

Pemmasani, G. (2018), "Dispy: Distributed and Parallel Computing with/for Python."

Perng, C.-S., Wang, H., Zhang, S. R., and Parker, D. S. (2000), "Landmark: A New Technique for Similarity-Based Pattern Querying in Time Series Databases," in *Proceedings of 16th International Conference on Data Engineering*.

Porwal, S., Chaudhary, Y., Joshi, J., and Jain, M. (2013), "Data Compression Methodologies for Lossless Data and Comparison between Algorithms," *International Journal of Engineering Science and Innovative Technology (IJESIT)*, 2, 142-147.

Rice, J. R. (1966), "Experiments on Gram-Schmidt Orthogonalization," *Mathematics of Computation*, 20, 325-328.

Rousseeuw, P. J. (1987), "Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis," *Journal of computational and applied mathematics*, 20, 53-65.

Sarin, S. C., Sherali, H. D., Varadarajan, A., and Liao, L. 2016. "Stochastic Scheduling for a Network of Flexible Job Shops." In *Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling*, 45-75. Springer.

Schrijver, A. 1998. *Theory of Linear and Integer Programming*: John Wiley & Sons.

Shanmugasundaram, S., and Lourdusamy, R. (2011), "A Comparative Study of Text Compression Algorithms," *International Journal of Wisdom Based Computing*, 1, 68-76.

Silverman, B. W. (1996), "Smoothed Functional Principal Components Analysis by Choice of Norm," *The Annals of Statistics*, 24, 1-24.

Singh, A. S., and Masuku, M. B. (2014), "Sampling Techniques & Determination of Sample Size in Applied Statistics Research: An Overview," *International Journal of Economics, Commerce and Management*, 2, 1-22.

Staar, P. W., Dolfi, M., Auer, C., and Bekas, C. (2018), "Corpus Conversion Service: A Machine Learning Platform to Ingest Documents at Scale," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 774-782.

Stoica, I., Franklin, M., Jordan, M., Fox, A., Joseph, A., Mahoney, M., Katz, R., Patterson, D., and Shenker, S. 2017. The Berkeley Data Analysis System (Bdas): An Open Source Platform for Big Data Analytics. University of California, Berkeley Berkeley United States.

Sun, H., Deng, X., Wang, K., and Jin, R. (2016), "Logistic Regression for Crystal Growth Process Modeling through Hierarchical Nonnegative Garrote-Based Variable Selection," *IIE Transactions*, 48, 787-796.

Sun, H., Huang, S., and Jin, R. (2017), "Functional Graphical Models for Manufacturing Process Modeling," *IEEE Transactions on Automation Science and Engineering*, 14, 1612-1621.

Sutton, J. R., Mahajan, R., Akbilgic, O., and Kamaleswaran, R. (2018), "Physonline: An Open Source Machine Learning Pipeline for Real-Time Analysis of Streaming Physiological Waveform," *IEEE journal of biomedical and health informatics*, 23, 59-65.

Thompson, S. K. (1990), "Adaptive Cluster Sampling," *Journal of the American Statistical Association*, 85, 1050-1059.

Tibshirani, R. (1996), "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, 58, 267-288.

Trost, J. E. (1986), "Statistically Nonrepresentative Stratified Sampling: A Sampling Technique for Qualitative Studies," *Qualitative sociology*, 9, 54-57.

Van Beers, W. C., and Kleijnen, J. P. (2004), "Kriging Interpolation in Simulation: A Survey," in *Proceedings of the 2004 Winter Simulation Conference, 2004.*

Van Leeuwen, F. 2007. *Hipparcos, the New Reduction of the Raw Data*. Vol. 350: Springer Science & Business Media.

Vasudevan, V., and Ramakrishna, M. (2017), "A Hierarchical Singular Value Decomposition Algorithm for Low Rank Matrices," *arXiv preprint arXiv:1710.02812*.

Wang, D., Liu, K., Zhang, X., and Wang, H. (2019), "Spatiotemporal Multitask Learning for 3-D Dynamic Field Modeling," *IEEE Transactions on Automation Science and Engineering*, 17, 708-721.

Wang, G., Qiao, J., Bi, J., Li, W., and Zhou, M. (2018), "Tl-Gdbn: Growing Deep Belief Network with Transfer Learning," *IEEE Transactions on Automation Science and Engineering*, 16, 874-885.

Wang, H., Yang, M., and Stufken, J. (2019), "Information-Based Optimal Subdata Selection for Big Data Linear Regression," *Journal of the American Statistical Association*, 114, 393-405.

Wang, L., Zhang, Y., Chen, X., and Jin, R. 2020. Online Computation Performance Analysis for Distributed Machine Learning Pipelines in Fog Manufacturing. In *IEEE International Conference on Automation Science and Engineering*: IEEE.

Wang, L., Zhang, Y., and Jin, R. (2020), "A Monitoring System for Anomaly Detection in Fog Manufacturing," in *2020 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pp. 88-93.

Wu, J., Chen, Y., Zhou, S., and Li, X. (2015), "Online Steady-State Detection for Process Control Using Multiple Change-Point Models and Particle Filters," *IEEE Transactions on Automation Science and Engineering*, 13, 688-700.

Yamaoka, K., Nakagawa, T., and Uno, T. (1978), "Statistical Moments in Pharmacokinetics," *Journal of Pharmacokinetics and Biopharmaceutics*, 6, 547-558.

Yan, H., Paynabar, K., and Shi, J. (2014), "Image-Based Process Monitoring Using Low-Rank Tensor Decomposition," *IEEE Transactions on Automation Science and Engineering*, 12, 216-227.

Yao, F., Müller, H. G., and Wang, J. L. (2005), "Functional Data Analysis for Sparse Longitudinal Data," *Journal of the American Statistical Association*, 100, 577-590.

Yu, J., Shen, Z., and Zheng, X. (2020), "Joint Feature and Label Adversarial Network for Wafer Map Defect Recognition," *IEEE Transactions on Automation Science and Engineering*.

Yue, X., Wen, Y., Hunt, J. H., and Shi, J. (2018), "Surrogate Model-Based Control Considering Uncertainties for Composite Fuselage Assembly," *Journal of Manufacturing Science and Engineering*, 140.

Zhang, C., Yan, H., Lee, S., and Shi, J. (2018), "Weakly Correlated Profile Monitoring Based on Sparse Multi-Channel Functional Principal Component Analysis," *IISE Transactions*, 50, 878-891.

Zhang, Y., Wang, L., Chen, X., and Jin, R. (2019), "Fog Computing for Distributed Family Learning in Cyber-Manufacturing Modeling," in *2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS)*, pp. 88-93.

Zhang, Z., Jiang, J., and Wang, H. (2007), "A New Segmentation Algorithm to Stock Time Series Based on Pip Approach," in *2007 International Conference on Wireless Communications, Networking and Mobile Computing*, pp. 5609-5612.

Zhou, R., Liu, M., and Li, T. (2013), "Characterizing the Efficiency of Data Deduplication for Big Data Storage Management," in *2013 IEEE international symposium on workload characterization (IISWC)*, pp. 98-108.

Zou, H., and Hastie, T. (2005), "Regularization and Variable Selection Via the Elastic Net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67, 301-320.

Zou, H., Hastie, T., and Tibshirani, R. (2006), "Sparse Principal Component Analysis," *Journal of Computational and Graphical Statistics*, 15, 265-286.

Appendix A

Table A-1. Mean and standard errors (within parenthesis) from 50 simulation runs for Normal distribution with $NC = 30$, $MS = 0.7$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.751	0.751	–	38.824
		–	(0.000)	(0.000)	–	(0.211)
	BM_1	0.272	0.754	0.754	0.111	10.664
		(0.005)	(0.001)	(0.001)	(0.009)	(0.191)
	BM_2	**0.247**	**0.755**	**0.755**	0.123	10.533
		(0.004)	(0.001)	(0.001)	(0.009)	(0.207)
	Proposed	**0.247**	0.754	0.753	9.015	9.707
		(0.008)	(0.001)	(0.001)	(0.160)	(0.144)
0.01	Full Data	–	0.751	0.751	–	38.824
		–	(0.000)	(0.000)	–	(0.211)
	BM_1	5.006	0.777	0.772	0.072	0.794
		(0.079)	(0.007)	(0.007)	(0.009)	(0.013)
	BM_2	4.987	0.791	0.787	0.056	0.705
		(0.108)	(0.006)	(0.006)	(0.006)	(0.010)
	Proposed	**3.908**	**0.802**	**0.799**	8.947	0.599
		(0.063)	(0.008)	(0.008)	(0.168)	(0.009)
0.005	Full Data	–	0.751	0.751	–	38.824
		–	(0.000)	(0.000)	–	(0.211)
	BM_1	10.523	0.726	0.718	0.060	0.548
		(0.321)	(0.025)	(0.025)	(0.006)	(0.009)
	BM_2	10.790	0.725	0.718	0.055	0.497
		(0.281)	(0.023)	(0.023)	(0.005)	(0.010)
	Proposed	**7.422**	**0.749**	**0.742**	8.886	0.355
		(0.117)	(0.033)	(0.033)	(0.160)	(0.007)

Table A-2. Mean and standard errors (within parenthesis) from 50 simulation runs for Normal distribution with $NC = 30$, $MS = 0.3$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.908	0.908	–	38.502
		–	(0.000)	(0.000)	–	(0.287)
	BM_1	0.271	0.909	0.909	0.099	6.299
		(0.005)	(0.001)	(0.001)	(0.007)	(0.074)
	BM_2	0.247	0.909	0.909	0.089	6.338
		(0.004)	(0.000)	(0.000)	(0.005)	(0.071)
	Proposed	**0.241**	**0.910**	**0.910**	9.048	5.864
		(0.005)	(0.000)	(0.000)	(0.175)	(0.069)
0.01	Full Data	–	0.908	0.908	–	38.502
		–	(0.000)	(0.000)	–	(0.287)
	BM_1	4.927	0.917	0.916	0.050	0.518
		(0.084)	(0.002)	(0.002)	(0.003)	(0.011)
	BM_2	4.987	0.915	0.914	0.050	0.505
		(0.108)	(0.002)	(0.002)	(0.003)	(0.008)
	Proposed	**3.825**	**0.924**	**0.923**	8.941	0.431
		(0.058)	(0.002)	(0.002)	(0.170)	(0.007)
0.005	Full Data	–	0.908	0.908	–	38.502
		–	(0.000)	(0.000)	–	(0.287)
	BM_1	10.694	0.925	0.923	0.068	0.386
		(0.314)	(0.003)	(0.003)	(0.007)	(0.011)
	BM_2	10.790	0.921	0.919	0.064	0.382
		(0.281)	(0.003)	(0.003)	(0.007)	(0.010)
	Proposed	**7.390**	**0.941**	**0.939**	8.907	0.291
		(0.150)	(0.004)	(0.004)	(0.153)	(0.007)

Table A-3. Mean and standard errors (within parenthesis) from 50 simulation runs for Normal distribution with $\mathrm{NC} = 30$, $\mathrm{MS} = 0.7$, and $\mathrm{STN} = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.909	0.909	–	38.491
		–	(0.000)	(0.000)	–	(0.108)
	BM_1	0.275	0.910	0.910	0.102	6.355
		(0.005)	(0.001)	(0.001)	(0.007)	(0.089)
	BM_2	0.247	**0.911**	**0.911**	0.088	6.509
		(0.004)	(0.001)	(0.001)	(0.004)	(0.096)
	Proposed	**0.236**	**0.911**	**0.911**	9.042	6.018
		(0.006)	(0.000)	(0.000)	(0.171)	(0.057)
0.01	Full Data	–	0.909	0.909	–	38.491
		–	(0.000)	(0.000)	–	(0.108)
	BM_1	4.950	0.920	0.918	0.050	0.522
		(0.109)	(0.003)	(0.003)	(0.003)	(0.011)
	BM_2	4.987	0.924	0.922	0.051	0.509
		(0.108)	(0.002)	(0.002)	(0.003)	(0.009)
	Proposed	**3.872**	**0.930**	**0.929**	8.955	0.442
		(0.053)	(0.003)	(0.003)	(0.167)	(0.005)
0.005	Full Data	–	0.909	0.909	–	38.491
		–	(0.000)	(0.000)	–	(0.108)
	BM_1	10.739	0.904	0.901	0.062	0.399
		(0.299)	(0.014)	(0.014)	(0.006)	(0.011)
	BM_2	10.790	0.919	0.916	0.054	0.392
		(0.281)	(0.012)	(0.012)	(0.005)	(0.009)
	Proposed	**7.367**	**0.924**	**0.922**	8.983	0.305
		(0.123)	(0.020)	(0.020)	(0.170)	(0.005)

Table A-4. Mean and standard errors (within parenthesis) from 50 simulation runs for Normal distribution with $\mathrm{NC} = 1000$, $\mathrm{MS} = 0.7$, and $\mathrm{STN} = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.750	0.750	–	38.612
		–	(0.001)	(0.001)	–	(0.206)
	BM_1	0.337	0.751	0.751	0.107	7.539
		(0.015)	(0.004)	(0.004)	(0.009)	(0.104)
	BM_2	**0.261**	**0.756**	**0.755**	0.115	7.766
		(0.006)	(0.002)	(0.002)	(0.009)	(0.078)
	Proposed	0.301	0.754	0.753	9.385	7.117
		(0.010)	(0.003)	(0.003)	(0.323)	(0.106)
0.01	Full Data	–	0.750	0.750	–	38.612
		–	(0.001)	(0.001)	–	(0.206)
	BM_1	3.669	0.765	0.761	0.048	0.756
		(0.168)	(0.012)	(0.012)	(0.003)	(0.015)
	BM_2	3.992	**0.782**	**0.779**	0.046	0.718
		(0.245)	(0.012)	(0.012)	(0.001)	(0.015)
	Proposed	**3.284**	0.770	0.766	9.443	0.716
		(0.206)	(0.011)	(0.011)	(0.318)	(0.016)
0.005	Full Data	–	0.750	0.750	–	38.612
		–	(0.001)	(0.001)	–	(0.206)
	BM_1	7.623	0.778	0.772	0.049	0.445
		(0.596)	(0.019)	(0.019)	(0.003)	(0.009)
	BM_2	8.652	**0.788**	**0.781**	0.049	0.431
		(0.740)	(0.017)	(0.018)	(0.003)	(0.009)
	Proposed	**5.696**	0.783	0.777	9.457	0.493
		(0.447)	(0.026)	(0.026)	(0.320)	(0.009)

Table A-5. Mean and standard errors (within parenthesis) from 50 simulation runs for Normal distribution with $\mathrm{NC} = 1000$, $\mathrm{MS} = 0.3$, and $\mathrm{STN} = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.908	0.908	–	39.449
		–	(0.000)	(0.000)	–	(0.642)
	BM_1	0.319	**0.910**	**0.910**	0.131	5.742
		(0.013)	(0.000)	(0.000)	(0.010)	(0.067)
	BM_2	**0.261**	0.909	0.909	0.090	5.767
		(0.006)	(0.001)	(0.001)	(0.006)	(0.059)
	Proposed	0.310	0.909	0.909	9.302	5.362
		(0.013)	(0.001)	(0.001)	(0.318)	(0.044)
0.01	Full Data	–	0.908	0.908	–	39.449
		–	(0.000)	(0.000)	–	(0.642)
	BM_1	4.223	0.923	0.922	0.067	0.586
		(0.251)	(0.002)	(0.002)	(0.007)	(0.011)
	BM_2	3.992	0.916	0.915	0.054	0.535
		(0.245)	(0.002)	(0.002)	(0.005)	(0.010)
	Proposed	**3.118**	**0.924**	**0.923**	9.247	0.434
		(0.197)	(0.003)	(0.003)	(0.303)	(0.005)
0.005	Full Data	–	0.908	0.908	–	39.449
		–	(0.000)	(0.000)	–	(0.642)
	BM_1	7.139	0.926	0.924	0.065	0.407
		(0.534)	(0.004)	(0.004)	(0.006)	(0.011)
	BM_2	8.652	0.929	0.927	0.053	0.386
		(0.740)	(0.003)	(0.003)	(0.005)	(0.010)
	Proposed	**5.448**	**0.945**	**0.944**	9.322	0.290
		(0.355)	(0.005)	(0.005)	(0.321)	(0.007)

Table A-6. Mean and standard errors (within parenthesis) from 50 simulation runs for Normal distribution with $\mathrm{NC} = 1000$, $\mathrm{MS} = 0.7$, and $\mathrm{STN} = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.910	0.910	–	38.367
		–	(0.000)	(0.000)	–	(0.098)
	BM_1	0.324	0.911	0.911	0.130	5.768
		(0.014)	(0.002)	(0.002)	(0.010)	(0.050)
	BM_2	**0.261**	**0.912**	**0.912**	0.126	5.769
		(0.006)	(0.001)	(0.001)	(0.010)	(0.070)
	Proposed	0.305	0.910	0.910	9.463	5.492
		(0.012)	(0.002)	(0.002)	(0.321)	(0.062)
0.01	Full Data	–	0.910	0.910	–	38.367
		–	(0.000)	(0.000)	–	(0.098)
	BM_1	4.429	**0.928**	**0.926**	0.067	0.566
		(0.328)	(0.004)	(0.004)	(0.007)	(0.015)
	BM_2	3.992	0.922	0.920	0.055	0.534
		(0.245)	(0.005)	(0.005)	(0.005)	(0.012)
	Proposed	**3.354**	0.922	0.921	9.335	0.477
		(0.214)	(0.004)	(0.004)	(0.316)	(0.010)
0.005	Full Data	–	0.910	0.910	–	38.367
		–	(0.000)	(0.000)	–	(0.098)
	BM_1	6.877	0.911	0.908	0.056	0.421
		(0.428)	(0.009)	(0.009)	(0.004)	(0.011)
	BM_2	8.652	0.925	0.922	0.060	0.382
		(0.740)	(0.009)	(0.009)	(0.006)	(0.010)
	Proposed	**5.614**	**0.943**	**0.941**	9.319	0.318
		(0.458)	(0.006)	(0.006)	(0.318)	(0.005)

Table A-7. Mean and standard errors (within parenthesis) from 50 simulation runs for t distribution (df = 10) with $NC = 30$, $MS = 0.7$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.750	0.750	–	46.454
		–	(0.000)	(0.000)	–	(0.492)
	BM_1	0.286	0.753	0.752	0.038	12.150
		(0.006)	(0.001)	(0.001)	(0.004)	(0.219)
	BM_2	0.263	0.751	0.750	0.048	13.054
		(0.005)	(0.001)	(0.001)	(0.004)	(0.298)
	Proposed	**0.254**	**0.754**	**0.754**	8.076	12.747
		(0.004)	(0.001)	(0.001)	(0.264)	(0.203)
0.01	Full Data	–	0.750	0.750	–	46.454
		–	(0.000)	(0.000)	–	(0.492)
	BM_1	4.468	0.785	0.782	0.032	0.770
		(0.078)	(0.006)	(0.006)	(0.005)	(0.011)
	BM_2	4.601	0.776	0.773	0.025	0.788
		(0.104)	(0.004)	(0.004)	(0.001)	(0.013)
	Proposed	**3.557**	**0.786**	**0.784**	7.652	0.719
		(0.063)	(0.005)	(0.005)	(0.246)	(0.016)
0.005	Full Data	–	0.750	0.750	–	46.454
		–	(0.000)	(0.000)	–	(0.492)
	BM_1	9.604	0.766	0.760	0.022	0.464
		(0.217)	(0.009)	(0.009)	(0.000)	(0.007)
	BM_2	9.725	0.785	0.779	0.026	0.463
		(0.239)	(0.009)	(0.010)	(0.002)	(0.010)
	Proposed	**6.650**	**0.811**	**0.807**	7.641	0.422
		(0.104)	(0.011)	(0.011)	(0.234)	(0.014)

Table A-8. Mean and standard errors (within parenthesis) from 50 simulation runs for t distribution (df = 10) with $NC = 30$, $MS = 0.7$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.909	0.909	–	44.390
		–	(0.000)	(0.000)	–	(0.089)
	BM_1	0.288	**0.910**	0.909	0.045	7.543
		(0.005)	(0.000)	(0.000)	(0.005)	(0.076)
	BM_2	0.263	0.909	0.909	0.040	7.672
		(0.005)	(0.000)	(0.000)	(0.001)	(0.101)
	Proposed	**0.250**	**0.910**	**0.910**	7.491	7.656
		(0.004)	(0.001)	(0.001)	(0.234)	(0.069)
0.01	Full Data	–	0.909	0.909	–	44.39
		–	(0.000)	(0.000)	–	(0.089)
	BM_1	4.507	0.919	0.918	0.028	0.491
		(0.078)	(0.002)	(0.002)	(0.004)	(0.008)
	BM_2	4.601	0.917	0.915	0.024	0.473
		(0.104)	(0.002)	(0.002)	(0.003)	(0.009)
	Proposed	**3.547**	**0.921**	**0.920**	6.289	0.445
		(0.051)	(0.002)	(0.002)	(0.212)	(0.010)
0.005	Full Data	–	0.909	0.909	–	44.39
		–	(0.000)	(0.000)	–	(0.089)
	BM_1	9.353	0.924	0.922	0.027	0.322
		(0.222)	(0.004)	(0.004)	(0.004)	(0.006)
	BM_2	9.725	0.926	0.924	0.025	0.316)
		(0.239)	(0.003)	(0.003)	(0.003)	(0.006)
	Proposed	**6.501**	**0.939**	**0.937**	6.201	0.279
		(0.087)	(0.004)	(0.004)	(0.214)	(0.009)

Table A-9. Mean and standard errors (within parenthesis) from 50 simulation runs for t distribution (df = 10) with $NC = 1000$, $MS = 0.3$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.747	0.747	–	44.663
		–	(0.000)	(0.000)	–	(0.077)
	BM_1	0.338	0.751	0.751	0.059	9.346
		(0.014)	(0.001)	(0.001)	(0.006)	(0.114)
	BM_2	**0.270**	**0.752**	**0.752**	0.067	8.870
		(0.006)	(0.001)	(0.001)	(0.007)	(0.116)
	Proposed	0.303	0.751	0.751	7.876	8.822
		(0.010)	(0.001)	(0.001)	(0.312)	(0.108)
0.01	Full Data	–	0.747	0.747	–	44.663
		–	(0.000)	(0.000)	–	(0.077)
	BM_1	3.574	0.790	0.787	0.024	0.746
		(0.164)	(0.005)	(0.005)	(0.001)	(0.010)
	BM_2	3.860	0.781	0.777	0.028	0.741
		(0.224)	(0.007)	(0.007)	(0.002)	(0.010)
	Proposed	**3.087**	**0.791**	**0.789**	7.731	0.707
		(0.125)	(0.006)	(0.007)	(0.321)	(0.014)
0.005	Full Data	–	0.747	0.747	–	44.663
		–	(0.000)	(0.000)	–	(0.077)
	BM_1	6.099	0.786	0.781	0.030	0.414
		(0.296)	(0.010)	(0.010)	(0.003)	(0.008)
	BM_2	8.231	0.777	0.771	0.028	0.422
		(0.688)	(0.012)	(0.012)	(0.002)	(0.008)
	Proposed	**5.160**	**0.822**	**0.818**	7.653	0.403
		(0.276)	(0.012)	(0.012)	(0.302)	(0.012)

Table A-10. Mean and standard errors (within parenthesis) from 50 simulation runs for t distribution (df = 10) with $NC = 1000$, $MS = 0.3$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.908	0.908	–	47.082
		–	(0.000)	(0.000)	–	(0.644)
	BM_1	0.339	0.909	**0.909**	0.041	6.686
		(0.013)	(0.001)	(0.001)	(0.004)	(0.054)
	BM_2	**0.270**	0.909	**0.909**	0.054	6.65
		(0.006)	(0.000)	(0.000)	(0.004)	(0.054)
	Proposed	0.309	**0.910**	**0.909**	6.475	6.211
		(0.010)	(0.001)	(0.001)	(0.291)	(0.049)
0.01	Full Data	–	0.908	0.908	–	47.082
		–	(0.000)	(0.000)	–	(0.644)
	BM_1	3.986	0.919	0.917	0.025	0.502
		(0.247)	(0.002)	(0.002)	(0.002)	(0.007)
	BM_2	3.860	0.916	0.915	0.027	0.510
		(0.224)	(0.002)	(0.002)	(0.001)	(0.007)
	Proposed	**3.114**	**0.928**	**0.926**	6.471	0.444
		(0.133)	(0.002)	(0.002)	(0.284)	(0.009)
0.005	Full Data	–	0.908	0.908	–	47.082
		–	(0.000)	(0.000)	–	(0.644)
	BM_1	7.625	0.928	0.926	0.028	0.326
		(0.549)	(0.004)	(0.004)	(0.003)	(0.007)
	BM_2	8.231	0.932	0.930	0.026	0.325
		(0.688)	(0.003)	(0.003)	(0.001)	(0.008)
	Proposed	**4.861**	**0.938**	**0.936**	6.451	0.287
		(0.238)	(0.005)	(0.005)	(0.276)	(0.010)

Table A-11. Mean and standard errors (within parenthesis) from 50 simulation runs for t distribution (df = 10) with $NC = 30$, $MS = 0.7$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.751	0.751	–	46.559
		–	(0.000)	(0.000)	–	(0.097)
	BM_1	0.275	0.756	0.755	0.043	11.451
		(0.005)	(0.001)	(0.001)	(0.005)	(0.182)
	BM_2	0.263	0.755	0.754	0.039	11.182
		(0.005)	(0.001)	(0.001)	(0.002)	(0.153)
	Proposed	**0.245**	**0.757**	**0.756**	6.687	10.713
		(0.004)	(0.001)	(0.001)	(0.247)	(0.204)
0.01	Full Data	–	0.751	0.751	–	46.559
		–	(0.000)	(0.000)	–	(0.097)
	BM_1	4.436	0.763	0.758	0.021	0.729
		(0.088)	(0.007)	(0.007)	(0.002)	(0.009)
	BM_2	4.601	0.752	0.747	0.022	0.725
		(0.104)	(0.018)	(0.018)	(0.001)	(0.013)
	Proposed	**3.540**	**0.796**	**0.792**	6.632	0.628
		(0.062)	(0.012)	(0.012)	(0.220)	(0.010)
0.005	Full Data	–	0.751	0.751	–	46.559
		–	(0.000)	(0.000)	–	(0.097)
	BM_1	9.528	**0.762**	**0.754**	0.024	0.432
		(0.262)	(0.017)	(0.017)	(0.003)	(0.008)
	BM_2	9.725	0.761	0.753	0.026	0.438
		(0.239)	(0.015)	(0.015)	(0.003)	(0.010)
	Proposed	**6.837**	0.716	0.709	6.995	0.407
		(0.132)	(0.026)	(0.026)	(0.249)	(0.013)

Table A-12. Mean and standard errors (within parenthesis) from 50 simulation runs for t distribution (df = 10) with $NC = 30$, $MS = 0.7$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.909	0.909	–	46.231
		–	(0.000)	(0.000)	–	(0.233)
	BM_1	0.281	0.911	0.911	0.035	7.335
		(0.005)	(0.000)	(0.000)	(0.004)	(0.082)
	BM_2	0.263	0.911	0.911	0.041	7.413
		(0.005)	(0.000)	(0.000)	(0.002)	(0.079)
	Proposed	**0.248**	**0.913**	**0.912**	6.668	7.175
		(0.004)	(0.001)	(0.001)	(0.229)	(0.072)
0.01	Full Data	–	0.909	0.909	–	46.231
		–	(0.000)	(0.000)	–	(0.233)
	BM_1	4.588	0.920	0.918	0.024	0.495
		(0.087)	(0.003)	(0.003)	(0.003)	(0.009)
	BM_2	4.601	0.926	0.924	0.022	0.486
		(0.104)	(0.002)	(0.002)	(0.001)	(0.008)
	Proposed	**3.559**	**0.931**	**0.929**	6.292	0.441
		(0.074)	(0.003)	(0.003)	(0.211)	(0.009)
0.005	Full Data	–	0.909	0.909	–	46.231
		–	(0.000)	(0.000)	–	(0.233)
	BM_1	9.298	0.912	0.909	0.019	0.353
		(0.244)	(0.016)	(0.016)	(0.001)	(0.008)
	BM_2	9.725	**0.929**	**0.926**	0.029	0.334
		(0.239)	(0.008)	(0.008)	(0.004)	(0.008)
	Proposed	**6.59**	0.914	0.911	6.315	0.285
		(0.091)	(0.020)	(0.020)	(0.223)	(0.009)

Table A-13. Mean and standard errors (within parenthesis) from 50 simulation runs for t distribution (df = 10) with $NC = 1000$, $MS = 0.7$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.748	0.748	–	46.052
		–	(0.001)	(0.001)	–	(0.143)
	BM_1	0.317	**0.754**	0.753	0.044	8.564
		(0.013)	(0.003)	(0.003)	(0.005)	(0.105)
	BM_2	**0.270**	**0.754**	**0.754**	0.058	8.466
		(0.006)	(0.002)	(0.002)	(0.005)	(0.117)
	Proposed	0.312	0.752	0.752	7.402	8.863
		(0.015)	(0.003)	(0.003)	(0.329)	(0.133)
0.01	Full Data	–	0.748	0.748	–	46.052
		–	(0.001)	(0.001)	–	(0.143)
	BM_1	3.700	0.756	0.752	0.025	0.759
		(0.186)	(0.011)	(0.011)	(0.002)	(0.014)
	BM_2	3.860	**0.781**	**0.777**	0.027	0.739
		(0.224)	(0.011)	(0.011)	(0.002)	(0.017)
	Proposed	**3.176**	0.764	0.760	7.113	0.706
		(0.151)	(0.010)	(0.010)	(0.318)	(0.014)
0.005	Full Data	–	0.748	0.748	–	46.052
		–	(0.001)	(0.001)	–	(0.143)
	BM_1	6.627	0.758	0.751	0.024	0.442
		(0.371)	(0.017)	(0.017)	(0.002)	(0.010)
	BM_2	8.231	0.782	0.775	0.025	0.419
		(0.688)	(0.017)	(0.017)	(0.001)	(0.009)
	Proposed	**5.044**	**0.793**	**0.786**	6.738	0.367
		(0.258)	(0.018)	(0.018)	(0.302)	(0.011)

Table A-14. Mean and standard errors (within parenthesis) from 50 simulation runs for t distribution (df = 10) with $NC = 1000$, $MS = 0.7$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.908	0.908	–	45.303
		–	(0.000)	(0.000)	–	(0.103)
	BM_1	0.375	**0.912**	**0.912**	0.039	6.685
		(0.015)	(0.002)	(0.002)	(0.003)	(0.072)
	BM_2	**0.270**	0.911	0.911	0.057	6.747
		(0.006)	(0.001)	(0.001)	(0.005)	(0.060)
	Proposed	0.317	0.909	0.908	6.928	6.507
		(0.013)	(0.002)	(0.002)	(0.284)	(0.071)
0.01	Full Data	–	0.908	0.908	–	45.303
		–	(0.000)	(0.000)	–	(0.103)
	BM_1	3.660	0.913	0.912	0.028	0.539
		(0.170)	(0.004)	(0.004)	(0.003)	(0.009)
	BM_2	3.860	**0.919**	**0.918**	0.031	0.502
		(0.224)	(0.004)	(0.004)	(0.003)	(0.011)
	Proposed	**3.020**	0.916	0.915	6.978	0.479
		(0.139)	(0.004)	(0.004)	(0.318)	(0.010)
0.005	Full Data	–	0.908	0.908	–	45.303
		–	(0.000)	(0.000)	–	(0.103)
	BM_1	7.145	0.920	0.917	0.021	0.342
		(0.553)	(0.007)	(0.007)	(0.001)	(0.007)
	BM_2	8.231	0.929	0.926	0.027	0.309
		(0.688)	(0.006)	(0.006)	(0.002)	(0.007)
	Proposed	**5.967**	**0.943**	**0.941**	7.139	0.307
		(0.491)	(0.005)	(0.005)	(0.317)	(0.009)

Table A-15. Mean and standard errors (within parenthesis) from 50 simulation runs for Uniform distribution with $NC = 30$, $MS = 0.3$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.749	0.749	–	44.373
		–	(0.000)	(0.000)	–	(0.239)
	BM_1	0.984	0.750	0.750	0.041	11.516
		(0.016)	(0.001)	(0.001)	(0.003)	(0.224)
	BM_2	0.917	0.751	0.751	0.041	11.442
		(0.016)	(0.001)	(0.001)	(0.003)	(0.177)
	Proposed	**0.867**	**0.753**	**0.753**	5.391	11.01
		(0.009)	(0.001)	(0.001)	(0.227)	(0.186)
0.01	Full Data	–	0.749	0.749	–	44.373
		–	(0.000)	(0.000)	–	(0.239)
	BM_1	24.862	0.758	0.755	0.023	0.671
		(0.793)	(0.007)	(0.007)	(0.001)	(0.008)
	BM_2	26.066	0.771	0.767	0.024	0.686
		(0.957)	(0.006)	(0.006)	(0.002)	(0.011)
	Proposed	**19.775**	**0.793**	**0.790**	5.408	0.620
		(0.413)	(0.006)	(0.006)	(0.228)	(0.009)
0.005	Full Data	–	0.749	0.749	–	44.373
		–	(0.000)	(0.000)	–	(0.239)
	BM_1	50.936	0.731	0.724	0.021	0.389
		(1.870)	(0.013)	(0.013)	(0.000)	(0.007)
	BM_2	55.008	0.735	0.729	0.026	0.387
		(1.829)	(0.012)	(0.012)	(0.003)	(0.008)
	Proposed	**36.723**	**0.788**	**0.783**	5.464	0.343
		(1.000)	(0.013)	(0.014)	(0.222)	(0.008)

Table A-16. Mean and standard errors (within parenthesis) from 50 simulation runs for Uniform distribution with $NC = 30$, $MS = 0.3$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.909	0.909	–	43.99
		–	(0.000)	(0.000)	–	(0.334)
	BM_1	0.983	0.910	0.910	0.051	6.959
		(0.018)	(0.000)	(0.000)	(0.005)	(0.070)
	BM_2	0.917	**0.911**	**0.911**	0.051	6.853
		(0.016)	(0.000)	(0.000)	(0.005)	(0.069)
	Proposed	**0.892**	**0.911**	**0.911**	5.514	7.021
		(0.012)	(0.000)	(0.000)	(0.224)	(0.059)
0.01	Full Data	–	0.909	0.909	–	43.99
		–	(0.000)	(0.000)	–	(0.334)
	BM_1	24.492	0.922	0.920	0.025	0.460
		(0.972)	(0.002)	(0.002)	(0.003)	(0.007)
	BM_2	26.066	0.925	0.923	0.024	0.458
		(0.957)	(0.003)	(0.003)	(0.002)	(0.007)
	Proposed	**19.881**	**0.932**	**0.930**	5.483	0.468
		(0.398)	(0.002)	(0.002)	(0.231)	(0.007)
0.005	Full Data	–	0.909	0.909	–	43.99
		–	(0.000)	(0.000)	–	(0.334)
	BM_1	50.982	0.924	0.922	0.034	0.307
		(1.849)	(0.004)	(0.004)	(0.005)	(0.005)
	BM_2	55.008	0.929	0.927	0.025	0.307
		(1.829)	(0.005)	(0.005)	(0.002)	(0.004)
	Proposed	**34.759**	**0.942**	**0.940**	5.444	0.302
		(0.669)	(0.005)	(0.005)	(0.225)	(0.008)

Table A-17. Mean and standard errors (within parenthesis) from 50 simulation runs for Uniform distribution with $NC = 1000$, $MS = 0.3$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.750	0.750	–	44.009
		–	(0.000)	(0.000)	–	(0.140)
	BM_1	0.577	**0.756**	**0.755**	0.046	9.199
		(0.015)	(0.001)	(0.001)	(0.004)	(0.114)
	BM_2	**0.512**	0.754	0.754	0.050	9.528
		(0.014)	(0.001)	(0.001)	(0.004)	(0.143)
	Proposed	0.558	0.755	0.754	6.644	9.814
		(0.023)	(0.001)	(0.001)	(0.413)	(0.127)
0.01	Full Data	–	0.750	0.750	–	44.009
		–	(0.000)	(0.000)	–	(0.140)
	BM_1	11.204	0.773	0.769	0.025	0.665
		(0.789)	(0.006)	(0.006)	(0.003)	(0.009)
	BM_2	11.937	0.771	0.767	0.024	0.685
		(0.904)	(0.006)	(0.006)	(0.002)	(0.010)
	Proposed	**8.219**	**0.795**	**0.791**	6.542	0.658
		(0.601)	(0.008)	(0.008)	(0.407)	(0.011)
0.005	Full Data	–	0.750	0.750	–	44.009
		–	(0.000)	(0.000)	–	(0.140)
	BM_1	20.993	0.755	0.749	0.021	0.396
		(1.668)	(0.013)	0.014)	(0.000)	(0.010)
	BM_2	25.337	0.769	0.763	0.028	0.368
		(2.151)	(0.014)	0.014)	(0.004)	(0.005)
	Proposed	**13.613**	**0.798**	**0.792**	6.625	0.358
		(0.848)	(0.016)	0.016)	(0.406)	(0.008)

Table A-18. Mean and standard errors (within parenthesis) from 50 simulation runs for Uniform distribution with $NC = 1000$, $MS = 0.3$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.909	0.909	–	45.936
		–	(0.000)	(0.000)	–	(0.394)
	BM_1	0.570	**0.910**	**0.910**	0.051	6.853
		(0.016)	(0.000)	(0.000)	(0.005)	(0.055)
	BM_2	**0.512**	**0.910**	**0.910**	0.052	6.896
		(0.014)	(0.000)	(0.000)	(0.006)	(0.049)
	Proposed	0.539	**0.910**	**0.910**	5.772	6.285
		(0.025)	(0.000)	(0.000)	(0.384)	(0.051)
0.01	Full Data	–	0.909	0.909	–	45.936
		–	(0.000)	(0.000)	–	(0.394)
	BM_1	11.017	0.924	0.922	0.024	0.486
		(0.704)	(0.002)	(0.002)	(0.002)	(0.007)
	BM_2	11.937	0.924	0.922	0.025	0.485
		(0.904)	(0.002)	(0.002)	(0.002)	(0.007)
	Proposed	**7.908**	**0.934**	**0.933**	5.927	0.443
		(0.566)	(0.002)	(0.002)	(0.385)	(0.007)
0.005	Full Data	–	0.909	0.909	–	45.936
		–	(0.000)	(0.000)	–	(0.394)
	BM_1	21.164	0.926	0.923	0.022	0.336
		(1.526)	(0.005)	(0.006)	(0.001)	(0.009)
	BM_2	25.337	0.922	0.919	0.026	0.327
		(2.151)	(0.006)	(0.006)	(0.003)	(0.007)
	Proposed	**13.792**	**0.943**	**0.940**	5.812	0.273
		(1.028)	(0.007)	(0.007)	(0.392)	(0.007)

Table A-19. Mean and standard errors (within parenthesis) from 50 simulation runs for Uniform distribution with $NC = 30$, $MS = 0.7$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.750	0.750	–	46.998
		–	(0.000)	(0.000)	–	(0.361)
	BM_1	0.943	0.751	0.750	0.049	11.213
		(0.018)	(0.001)	(0.001)	(0.005)	(0.194)
	BM_2	0.917	**0.755**	**0.755**	0.047	11.058
		(0.016)	(0.001)	(0.001)	(0.004)	(0.197)
	Proposed	**0.861**	0.754	0.754	4.882	10.420
		(0.010)	(0.001)	(0.001)	(0.213)	(0.165)
0.01	Full Data	–	0.750	0.750	–	46.998
		–	(0.000)	(0.000)	–	(0.361)
	BM_1	24.815	0.720	0.716	0.024	0.649
		(0.887)	(0.024)	(0.024)	(0.002)	(0.012)
	BM_2	26.066	0.735	0.731	0.028	0.691
		(0.957)	(0.027)	(0.027)	(0.003)	(0.012)
	Proposed	**21.119**	**0.794**	**0.791**	4.725	0.559
		(0.443)	(0.014)	(0.014)	(0.202)	(0.009)
0.005	Full Data	–	0.750	0.750	–	46.998
		–	(0.000)	(0.000)	–	(0.361)
	BM_1	48.571	0.722	0.714	0.022	0.395
		(1.869)	(0.028)	(0.028)	(0.001)	(0.007)
	BM_2	55.008	0.664	0.656	0.025	0.381
		(1.829)	(0.029)	(0.029)	(0.003)	(0.006)
	Proposed	**36.402**	**0.744**	**0.737**	4.727	0.284
		(0.857)	(0.024)	(0.024)	(0.215)	(0.006)

Table A-20. Mean and standard errors (within parenthesis) from 50 simulation runs for Uniform distribution with $NC = 30$, $MS = 0.7$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.909	0.909	–	43.161
		–	(0.000)	(0.000)	–	(0.089)
	BM_1	0.965	0.910	0.910	0.051	7.014
		(0.018)	(0.001)	(0.001)	(0.005)	(0.076)
	BM_2	0.917	**0.912**	**0.912**	0.047	6.972
		(0.016)	(0.000)	(0.000)	(0.004)	(0.06)
	Proposed	**0.883**	**0.912**	**0.912**	5.416	7.347
		(0.011)	(0.001)	(0.001)	(0.231)	(0.088)
0.01	Full Data	–	0.909	0.909	–	43.161
		–	(0.000)	(0.000)	–	(0.089)
	BM_1	24.662	0.835	0.833	0.021	0.449
		(0.878)	(0.031)	(0.031)	(0.001)	(0.008)
	BM_2	26.066	0.823	0.821	0.025	0.440
		(0.957)	(0.035)	(0.035)	(0.002)	(0.007)
	Proposed	**19.517**	**0.934**	**0.933**	5.246	0.454
		(0.383)	(0.004)	(0.004)	(0.230)	(0.009)
0.005	Full Data	–	0.909	0.909	–	43.161
		–	(0.000)	(0.000)	–	(0.089)
	BM_1	53.988	0.822	0.819	0.022	0.296
		(2.488)	(0.033)	(0.033)	(0.000)	(0.005)
	BM_2	55.008	0.802	0.798	0.024	0.309
		(1.829)	(0.031)	(0.031)	(0.002)	(0.007)
	Proposed	**35.883**	**0.912**	**0.910**	5.027	0.264
		(0.879)	(0.018)	(0.018)	(0.213)	(0.006)

Table A-21. Mean and standard errors (within parenthesis) from 50 simulation runs for Uniform distribution with $NC = 1000$, $MS = 0.7$, and $STN = 3$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.749	0.749	–	43.923
		–	(0.001)	(0.001)	–	(0.093)
	BM_1	0.586	0.750	0.750	0.039	9.917
		(0.016)	(0.004)	(0.004)	(0.002)	(0.183)
	BM_2	**0.512**	0.754	**0.754**	0.057	9.607
		(0.014)	(0.002)	(0.002)	(0.005)	(0.132)
	Proposed	0.531	**0.755**	**0.754**	6.355	9.618
		(0.019)	(0.003)	(0.003)	(0.419)	(0.132)
0.01	Full Data	–	0.749	0.749	–	43.923
		–	(0.001)	(0.001)	–	(0.093)
	BM_1	11.37	0.723	0.719	0.023	0.736
		(0.730)	(0.010)	(0.011)	(0.001)	(0.015)
	BM_2	11.937	**0.770**	**0.767**	0.025	0.673
		(0.904)	(0.011)	(0.012)	(0.002)	(0.015)
	Proposed	**7.511**	0.751	0.748	6.438	0.686
		(0.423)	(0.011)	(0.011)	(0.407)	(0.012)
0.005	Full Data	–	0.749	0.749	–	43.923
		–	(0.001)	(0.001)	–	(0.093)
	BM_1	19.249	0.728	0.719	0.022	0.393
		(1.501)	(0.014)	(0.015)	(0.001)	(0.007)
	BM_2	25.337	0.767	0.760	0.023	0.382
		(2.151)	(0.021)	(0.022)	(0.002)	(0.007)
	Proposed	**12.659**	**0.795**	**0.789**	6.494	0.362
		(0.718)	(0.021)	(0.022)	(0.412)	(0.007)

Table A-22. Mean and standard errors (within parenthesis) from 50 simulation runs for Uniform distribution with $NC = 1000$, $MS = 0.7$, and $STN = 10$

Filtering Ratio	Filtering Method	Entropy Loss	R^2	R^2 Adjusted	Filtering Time (in seconds)	Modeling Time (in seconds)
0.1	Full Data	–	0.909	0.909	–	43.618
		–	(0.000)	(0.000)	–	(0.085)
	BM_1	0.605	0.907	0.906	0.049	7.091
		(0.020)	(0.002)	(0.002)	(0.005)	(0.076)
	BM_2	**0.512**	**0.911**	**0.911**	0.048	6.932
		(0.014)	(0.001)	(0.001)	(0.004)	(0.049)
	Proposed	0.548	0.908	0.908	5.841	6.501
		(0.018)	(0.002)	(0.002)	(0.384)	(0.061)
0.01	Full Data	–	0.909	0.909	–	43.618
		–	(0.000)	(0.000)	–	(0.085)
	BM_1	12.092	0.915	0.914	0.023	0.455
		(1.035)	(0.005)	(0.005)	(0.002)	(0.010)
	BM_2	11.937	**0.918**	**0.917**	0.022	0.474
		(0.904)	(0.004)	(0.004)	(0.001)	(0.008)
	Proposed	**9.134**	0.915	0.914	5.689	0.434
		(0.778)	(0.004)	(0.004)	(0.379)	(0.010)
0.005	Full Data	–	0.909	0.909	–	43.618
		–	(0.000)	(0.000)	–	(0.085)
	BM_1	20.066	0.905	0.901	0.022	0.285
		(1.656)	(0.009)	(0.009)	(0.002)	(0.007)
	BM_2	25.337	0.914	0.911	0.021	0.302
		(2.151)	(0.008)	(0.008)	(0.001)	(0.008)
	Proposed	**13.94**	**0.931**	**0.929**	5.724	0.273
		(0.988)	(0.009)	(0.009)	(0.392)	(0.007)

Table A-23. Case Study Data Summary

Variable	Min	1st Quartile	Median	Mean	3rd Quartile	Max
time	-175949.000	-90425.000	24085.000	0.000	82908.000	135947.000
1	-3.148	-0.015	0.059	0.000	0.122	1.180
2	-2.044	-0.009	-0.004	0.000	0.001	4.529
3	-29.242	-1.715	-0.443	0.000	0.836	98.759
4	-0.600	-0.046	0.002	0.000	0.046	0.419
5	-21.157	-1.357	0.144	0.000	1.744	8.944
6	-6320.900	516.300	517.000	0.000	517.300	519.700
7	-599.050	-175.940	28.640	0.000	198.540	341.680
8	-4.991	-1.818	-1.818	0.000	1.225	17.551
9	-5.759	-1.875	-1.193	0.000	1.235	20.484
10	-3.645	-0.016	-0.008	0.000	0.000	4.285
11	-100.060	-1.660	-0.060	0.000	1.621	99.940
12	-503.970	-165.490	31.310	0.000	187.900	364.430
13	-0.533	-0.032	0.002	0.000	0.033	0.393
14	-17.833	-1.133	0.067	0.000	1.667	17.567
15	-3.689	-1.825	-0.320	0.000	1.214	8.756
16	-3.476	-1.619	-1.619	0.000	1.272	8.993
17	-1.915	-1.915	1.085	0.000	1.085	1.085
18	-1.375	-0.031	0.002	0.000	0.035	0.837
19	-1.121	-0.070	0.001	0.000	0.073	2.274
20	-5.464	-0.025	0.008	0.000	0.038	0.908
21	-36.783	-5.583	0.717	0.000	6.017	39.117
22	-24.440	-2.940	-0.040	0.000	3.160	27.360
23	-119.061	-39.461	-0.461	0.000	41.539	75.039
24	-29.354	-2.454	0.446	0.000	3.146	15.646
25	-1.467	-0.033	0.001	0.000	0.036	0.259
26	-109.399	-2.699	0.501	0.000	3.901	39.101
27	-20.571	-2.371	0.129	0.000	2.529	16.229
28	-1.992	-0.067	0.006	0.000	0.070	0.389
29	-1.119	-0.172	-0.007	0.000	0.161	1.179
30	-0.011	0.000	0.000	0.000	0.000	0.003
31	-42.300	-2.300	-0.300	0.000	2.100	21.100

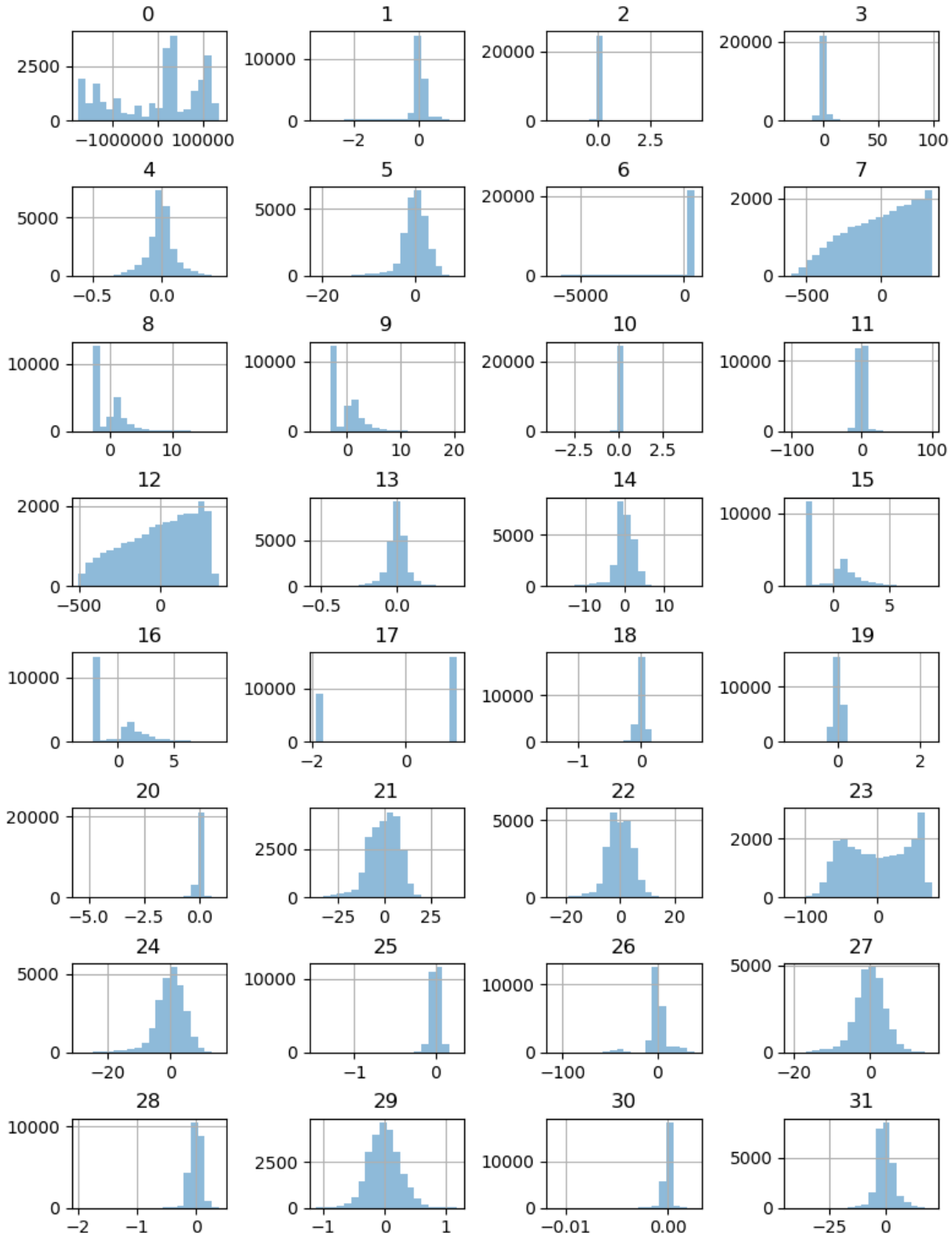


Figure A1. The Histogram of all variables

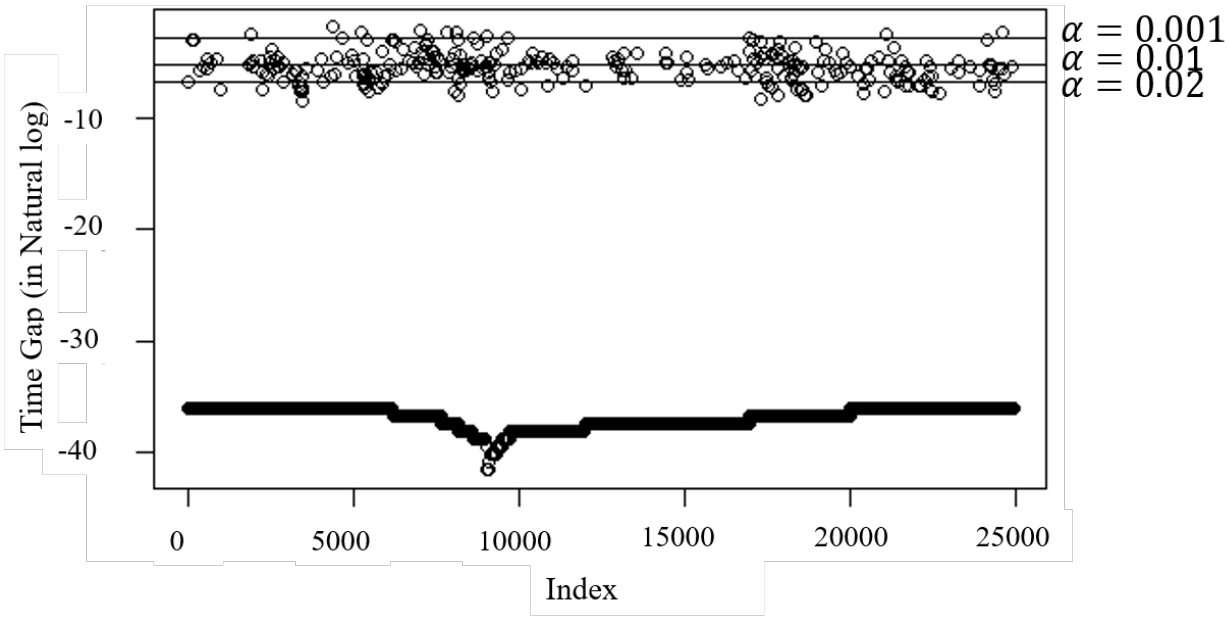


Figure A2. The gaps identified with $\alpha = 0.02,0.01,0.001$

Appendix B

Derivation of Proposition 2 of Chapter 3

Given B and V, U_j in re-written Equation (12)

$$L = \sum_{j=1}^{m} \| Y_j - X_j B_j - U_j V' \|^2 + \lambda_1 ||B||_*,$$

can be optimized at step k via

$$U_j{}^k = \left(Y_j - X_j B_j{}^{k-1}\right) V^k.$$

Derivation:

For the ease of representation, let's denote $(Y_j - X_j B_j)$ above as Y and V' above as B. The proof of Proposition 2 is equivalent to solving matrix U_j in the following matrix least square formulation:

$$\underset{A}{\operatorname{argmin}} \| Y - U_j B \|^2$$

where,

$$\begin{aligned} &\| Y - U_j B \|^2 \\ &= tr((Y - U_j B)^T (Y - U_j B)) \\ &= tr\left(Y^T Y + B^T U_j{}^T U_j B - Y^T U_j B - B^T U_j{}' Y\right) \\ &= tr(Y^T Y) + tr\left(B^T U_j{}^T U_j B\right) - 2tr\left(Y^T U_j B\right) \end{aligned}$$

By taking the first derivative of the above formulation and set it equal to zero, we have:

$$\begin{aligned} \nabla_{U_j} tr\left(U_j{}' U_j B B'\right) - 2\nabla_{U_j} tr\left(Y' U_j B\right) = 0, \\ 2U_j B B' - 2YB' = 0, \\ -2U_j B B^T = -2YB'. \end{aligned}$$

Therefore, the solution is

$$U_j = (YB^T)(BB^T)^{-1} = ((Y_j - X_j B_j)V)(V'V)^{-1},$$

and recall that $V'V = I$, then we have,

$$U_j = \left(Y_j - X_j B_j\right)V.$$

Derivation of the gradient in Proposition 3

$$\frac{\nabla\left(||\boldsymbol{y}_j{}^{(r)} - X_j\boldsymbol{\beta}_j{}^{(r)} - U_j{}^{k}\boldsymbol{v}^{(r)}||_F^2\right)}{\nabla\boldsymbol{\beta}_j{}^{(r)}} = \frac{\left(\boldsymbol{y}_j{}^{(r)} - X^j\boldsymbol{\beta}_j{}^{(r)} - U_j{}^{k}\boldsymbol{v}^{(r)}\right)'\left(\boldsymbol{y}_j{}^{(r)} - X^j\boldsymbol{\beta}_j{}^{(r)} - U_j\boldsymbol{v}^{(r)}\right)}{\nabla\boldsymbol{\beta}_j{}^{(r)}}$$
$$= \frac{-\boldsymbol{y}_j{}^{(r)\prime}X_j\boldsymbol{\beta}_j{}^{(r)} - \boldsymbol{\beta}_j{}^{(r)\prime}X_j{}'\boldsymbol{y}_j{}^{(r)} + \boldsymbol{\beta}_j{}^{(r)\prime}X_j{}'X_j\boldsymbol{\beta}_j{}^{(r)} + \boldsymbol{\beta}_j{}^{(r)\prime}X_j{}'U_j\boldsymbol{v}^{(r)} + U_j\boldsymbol{v}^{(r)\prime}X_j\boldsymbol{\beta}_j{}^{(r)}}{\nabla\boldsymbol{\beta}_j{}^{(r)}}$$
$$= -\left(\boldsymbol{y}_j{}^{(r)\prime}X_j\right)' - X_j{}'\boldsymbol{y}_j{}^{(r)} + 2X_j{}'X_j\boldsymbol{\beta}_j{}^{(r)} + X_j{}'U_j\boldsymbol{v}^{(r)} + X_j{}'U_j\boldsymbol{v}^{(r)}$$
$$= -2X_j{}'\boldsymbol{y}_j{}^{(r)} + 2X_j{}'X_j\boldsymbol{\beta}_j{}^{(r)} + 2X_j{}'U_j\boldsymbol{v}^{(r)}$$

The Computational Complexity of the Model Updating Algorithm

The model updating algorithm can be decomposed into three steps. At iteration k, we have:

22: Optimize V given B and U_j:

$$\begin{pmatrix} Y_1 - X_1 B_1{}^{k-1} \\ \vdots \\ Y_m - X_m B_m{}^{k-1} \end{pmatrix}' \begin{pmatrix} U_1{}^{k-1} \\ \vdots \\ U_m{}^{k-1} \end{pmatrix} = RDW',$$
$$V^k = RW'.$$

23: Optimize U_j for given B and V:

$$U_j{}^{k} = \left(Y_j - X_j B_j{}^{k-1}\right)V^k$$

24: Optimize B given U_j and V:

$$B^k = Pdiag(\hat{\sigma}_1 \dots \hat{\sigma}_m)Q',$$

Recall that there are m similar-but-non-identical processes, the dimensionality of the response is d ($Y_j \in \mathbb{R}^{n_j \times d}$), the number of samples in each process are n_j, the number of observed variables is p ($X_j \in \mathbb{R}^{n_j \times p}$), the number of latent variables is q ($U_j \in \mathbb{R}^{n_j \times q}$). Furthermore, $B_j =$

$\left(\boldsymbol{\beta}_j{}^{(1)} \quad \cdots \quad \boldsymbol{\beta}_j{}^{(d)}\right) \in \mathbb{R}^{p\times d}$, where $j = 1, \ldots, m$, $B = \begin{pmatrix} \boldsymbol{\beta}_1{}^{(1)} & \cdots & \boldsymbol{\beta}_m{}^{(1)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\beta}_1{}^{(d)} & \cdots & \boldsymbol{\beta}_m{}^{(d)} \end{pmatrix} \in \mathbb{R}^{p\times d}$, $V \in$ $\mathbb{R}^{d\times q}$ and we assume that $q < d$.

Since $Y_j - X_j B_j$ has a complexity of O($n_j pd + n_j d$)=O($(p+1)n_j d$), $\begin{pmatrix} Y_1 - X_1 B_1{}^{k-1} \\ \vdots \\ Y_m - X_m B_m{}^{k-1} \end{pmatrix}$ has a complexity of O($(p+1)nd$), where $n = n_1 +, \ldots, +n_m$. $\begin{pmatrix} Y_1 - X_1 B_1{}^{k-1} \\ \vdots \\ Y_m - X_m B_m{}^{k-1} \end{pmatrix}' \begin{pmatrix} U_1{}^{k-1} \\ \vdots \\ U_m{}^{k-1} \end{pmatrix}$ has a complexity of O($(p+1)nd + ndq$)= O($(p+q+1)nd$). Generating RDW' in Step 1 has a complexity of O(dq^2) (Vasudevan and Ramakrishna 2017), and $V^k = RW'$ has a complexity of O(dq^2). Collectively, Step 1 has a complexity of O($(p+q+1)nd + 2dq^2$)

Step 2 has a complexity of O($(p+1)n_j d + n_j dq$)= O($(p+q+1)n_j d$) for U_j, where $j = 1, \ldots, m$. To update all U_j, the complexity becomes O($(p+q+1)nd$).

In Step 3, we first use gradient to update un-thresholded B_j as $\tilde{B} = \begin{pmatrix} \widetilde{\boldsymbol{\beta}}_1{}^{(1)} & \cdots & \widetilde{\boldsymbol{\beta}}_m{}^{(1)} \\ \vdots & \ddots & \vdots \\ \widetilde{\boldsymbol{\beta}}_1{}^{(d)} & \cdots & \widetilde{\boldsymbol{\beta}}_m{}^{(d)} \end{pmatrix} \in$ $\mathbb{R}^{pd\times m}$, where $\widetilde{\boldsymbol{\beta}}_j{}^{(r)} = \boldsymbol{\beta}_j{}^{(r)} - t_1 \left(\nabla \left\| \boldsymbol{y}_j{}^{(r)} - X_j \boldsymbol{\beta}_j{}^{(r)} - U_j \boldsymbol{v}^{(r)} \right\|_F^2 / \nabla \boldsymbol{\beta}_j{}^{(r)}\right)$.

The computation of the gradient $\nabla \left\| \boldsymbol{y}_j{}^{(r)} - X_j \boldsymbol{\beta}_j{}^{(r)} - U_j \boldsymbol{v}^{(r)} \right\|_F^2 / \nabla \boldsymbol{\beta}_j{}^{(r)} = -2X_j{}' \boldsymbol{y}_j{}^{(r)} + 2X_j{}' X_j \boldsymbol{\beta}_j{}^{(r)} + 2X_j{}' U_j \boldsymbol{v}^{(r)}$ has a complexity O($n_j p + p + n_j p^2 + p^2 + p + n_j pq + pq + p$)= O($(n_j + 3 + n_j q + q)p + (n_j + 1)p^2$). Then, obtaining $\widetilde{\boldsymbol{\beta}}_j{}^{(r)}$ has the complexity of O($\left(n_j + 5 + n_j q + q\right)p + \left(n_j + 1\right)p^2$) and obtaining $\tilde{B}$ has the complexity of O($(n + 5m + nq + qm)dp +$

$(n+m)dp^2$). we assume that $pd < m$ and performing $\tilde{B} = Pdiag(\sigma_1 \dots \sigma_m)Q'$ has a complexity of O(pdm^2). Performing thresholding and reconstruct B as $B = Pdiag(\hat{\sigma}_1 \dots \hat{\sigma}_m)Q'$ has a complexity of O($(3+p^2d^2)m + pdm^2$). Collectively, Step 3 has a time complexity as O($(n+5m+nq+qm)dp + (n+m)dp^2 + pdm^2 + (3+p^2d^2)m + pdm^2$).

The overall computational complexity becomes O($2q^2d + 2pnd + 2qnd + 2nd + qnd + 5pmd + qpnd + qpmd + p^2nd + p^2md + pm^2d + 3m + p^2md^2 + pm^2d$).

n: the sample size

p: the number of observed variables

q: the number of latent variables

d: the dimensionality of the response

m: the number of similar-but-non-identical processes,

It is safe to assume $q < d$, $m < n$, We can simplify the complexity as O($qpnd + p^2nd + p^2md^2 + pm^2d$).

Appendix C

Table C-1. Mean and standard errors (within parenthesis) of model parameter estimation error over 100 simulation replications

n	$\hat{n}$	τ	ρ	γ	Proposed	IBOSS	Random	Stratified	Cluster-Based
10000	1862.4	1	0.7	10	**0.20 (<0.01)**	**0.20 (<0.01)**	0.23 (<0.01)	0.23 (<0.01)	0.24 (<0.01)
10000	2972.8	0.1	0.7	10	0.94 (0.02)	**0.68 (0.01)**	0.76 (0.01)	0.77 (0.01)	0.78 (0.01)

Table C-2. Mean and standard errors (within parenthesis) of prediction error over 100 simulation replications

n	$\hat{n}$	τ	ρ	γ	Proposed	IBOSS	Random	Stratified	Cluster-Based
10000	1862.4	1	0.7	10	1.85 (0.04)	**1.79 (0.04)**	1.83 (0.05)	1.82 (0.05)	1.83 (0.05)
10000	2972.8	0.1	0.7	10	7.81 (0.11)	**7.64 (0.11)**	7.72 (0.11)	7.72 (0.11)	7.71 (0.11)

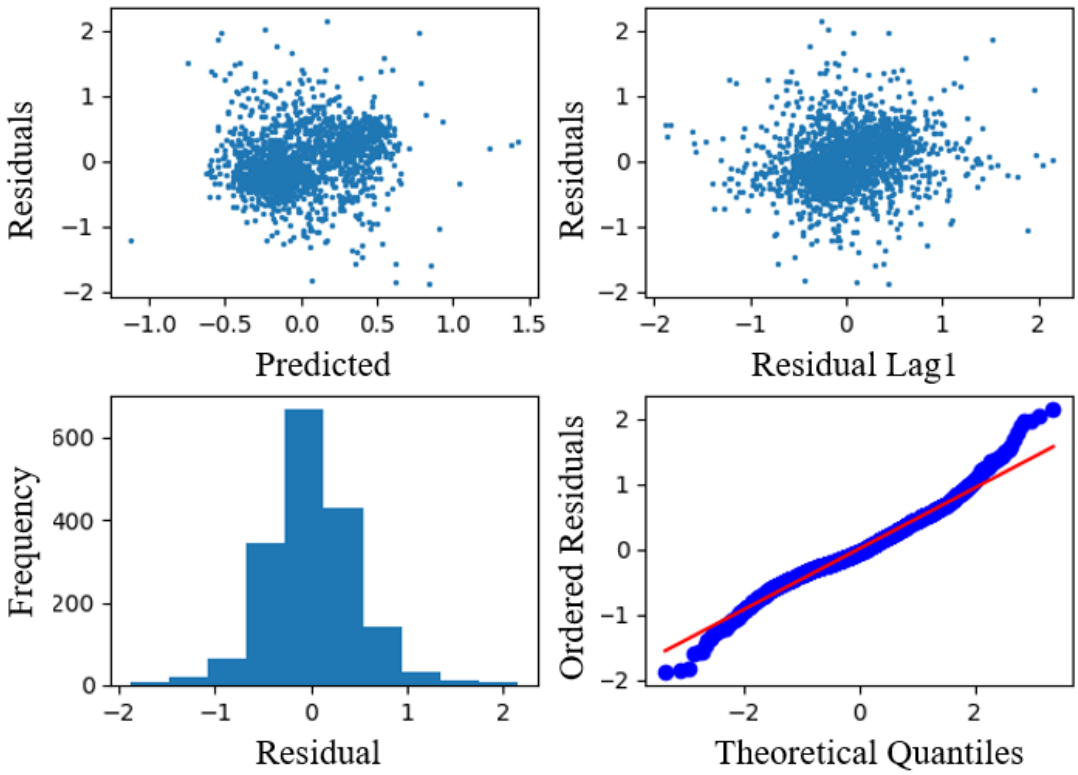


Figure C1. The assumption check of the modeling residuals for silicon ingot manufacturing case study

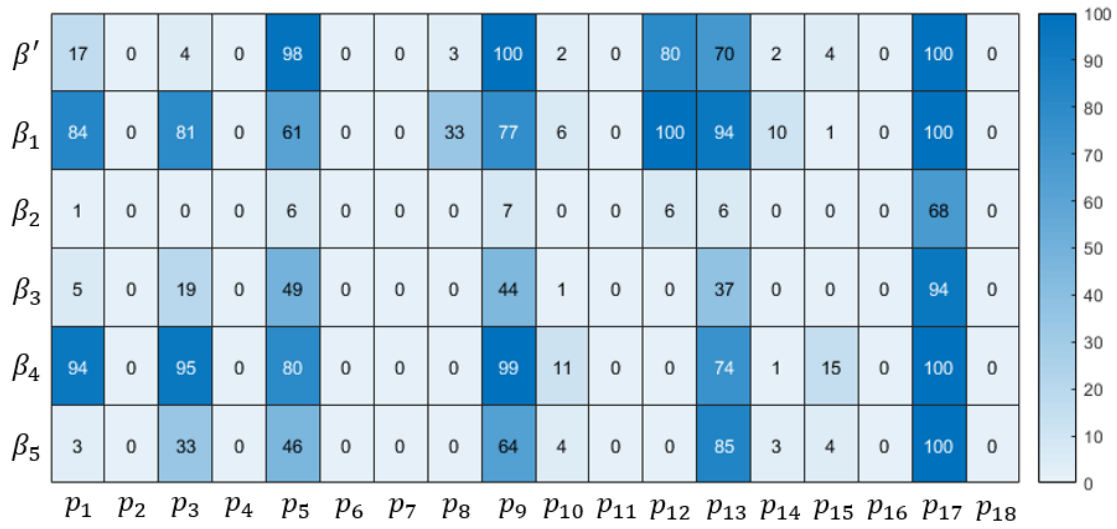


Figure C2. The number of times that each variable is selected over 100 replications for the silicon ingot manufacturing case study