

ABSTRACT

SHENG, JIEWEN. Privacy-Preserving Data Analytics for Industrial Prognostics. (Under the direction of Dr. Xiaolei Fang).

Industrial prognostics, defined as the prediction of failure times for systems or components, rely on substantial data analysis, primarily from sensors monitoring machinery health. This predictive capability is crucial for implementing proactive maintenance strategies, significantly impacting cost reduction and efficiency enhancement. However, the use of sensitive or proprietary data in these analyses introduces significant privacy concerns. Differential privacy and federated learning have emerged as leading frameworks to address these concerns, offering a means to perform analyses while protecting individual data contributions. Our research focuses on tailoring these privacy-preserving approaches for the industrial prognostics domain, with a particular emphasis on statistical methods that maintain the balance between privacy and the predictive utility of prognostic models.

The research in Chapter 2 introduces differentially private log-location-scale (DP-LLS) regression models, which incorporate differential privacy into LLS regression through the functional mechanism. The proposed models are established by injecting noise into the log-likelihood function of LLS regression for perturbed parameter estimation. The study in Chapter 3 introduces a privacy-preserving sensor selection approach for multi-sensor prognostics, utilizing LLS regression with group lasso for effective sensor selection. The regression models are optimized using the proximal gradient descent method and enhanced with differential privacy to protect sensitive information. In Chapter 4, we propose a federated learning framework incorporated with differential privacy for functional principal component analysis, allowing users with sparse data to benefit through collaboration while preserving privacy.

Privacy-Preserving Data Analytics for Industrial Prognostics

by
Jiewen Sheng

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Operations Research

Raleigh, North Carolina
2024

APPROVED BY:

Dr. Shu-Cherng Fang

Dr. Yunan Liu

Dr. Hong Wan

Dr. Xiaolei Fang
Chair of Advisory Committee

BIOGRAPHY

Prior to joining the PhD program in Operations Research at North Carolina State University, the author earned a Bachelor of Science in Mathematics from Linfield College in McMinnville, Oregon, and a Master of Engineering in Industrial and Systems Engineering from Lehigh University, Bethlehem, Pennsylvania.

TABLE OF CONTENTS

List of Tables	v
List of Figures	vi
Chapter 1 Introduction	1
1.1 Background	2
1.2 Motivations	3
1.3 Approaches	4
1.3.1 Differentially Private Log-Location-Scale Regression using Functional Mechanism	4
1.3.2 A Differentially Private Informative Sensor Selection Method	4
1.3.3 Privacy-Preserving Federated Functional Principal Component Analysis for Sparse Data	5
1.4 Outline of Dissertation	5
Chapter 2 Differentially Private Log-Location-Scale Regression using Functional Mechanism	7
2.1 Introduction	7
2.2 Preliminaries	11
2.2.1 LLS Regression	11
2.2.2 Differential Privacy	12
2.3 Differentially Private LLS Regression	14
2.3.1 Differentially Private SEV Regression	15
2.3.2 Differentially Private Logistic Regression	17
2.4 Simulation Study	20
2.4.1 Experimental settings	20
2.4.2 Results and Analysis	21
2.5 Case Study	27
2.5.1 Dataset and Experimental Settings	27
2.5.2 Results and Analysis	29
2.6 Conclusions	31
Chapter 3 A Differentially Private Informative Sensor Selection Method for Industrial Prognostics	33
3.1 Introduction	33
3.2 Regularized LLS Regression	36
3.2.1 Model Define	36
3.2.2 Sensor Selection Procedures	40
3.3 The Optimization Algorithm for Parameter Estimation	43
3.4 Differentially Private Sensor Selection	47
3.4.1 Differentially Private Proximal Gradient Descent	47
3.4.2 Revised Algorithm	52
3.5 Simulation Study	54

3.5.1	Experimental Settings	55
3.5.2	Results and Analysis	56
3.6	Case Study	63
3.6.1	Dataset and Experimental Settings	63
3.6.2	Results and Analysis	65
3.7	Conclusions	71
Chapter 4	Privacy-Preserving Federated Functional Principal Component Analysis for Sparse Data	74
4.1	Introduction	74
4.2	Functional Principal Components Analysis	79
4.3	A Federated Learning Approach of FPCA	85
4.3.1	Mean Function Estimate	86
4.3.2	Covariance Function Estimate	88
4.3.3	Measurement Error Variance Estimate	90
4.3.4	Late Stages of FL-FPCA	93
4.3.5	Differentially Private FL-FPCA	95
4.4	Simulation Study	97
4.4.1	Experimental Settings	97
4.4.2	Results and Analysis	100
4.5	Case Study	104
4.5.1	Dataset and Experimental Settings	104
4.5.2	Results and Analysis	105
4.6	Conclusions	109
Chapter 5	Conclusion	112
5.1	Summary of Contributions	112
5.2	Future Research Directions	114
5.3	Final Remarks	115
References	116
APPENDIX	120
Appendix A	Proofs in Chapter 2	121

LIST OF TABLES

Table 3.1	Case Study: Linear Model Results, DP Privacy Budget	66
Table 3.2	Case Study: SEV Model Results, DP Privacy Budget	67
Table 3.3	Case Study: Logistic Model Results, DP Privacy Budget	68
Table 4.1	Simulation: Boxplot Readings (Sparsity Level: 20% Missing)	101
Table 4.2	Simulation: Boxplot Readings (Sparsity Level: 70% Missing)	103
Table 4.3	Case Study: Boxplot Readings (Sparsity Level: 20% Missing)	107
Table 4.4	Case Study: Boxplot Readings (Sparsity Level: 70% Missing)	108

LIST OF FIGURES

Figure 2.1	SEV Regression: Prediction Error vs. Dimensionality d	22
Figure 2.2	Logistic Regression: Prediction Error vs. Dimensionality d	22
Figure 2.3	SEV Regression: Error vs. Sample Size n	24
Figure 2.4	Logistic Regression: Error vs. Sample Size n	25
Figure 2.5	SEV Regression: Error vs. DP privacy budget ϵ	26
Figure 2.6	Logistic Regression: Error vs. DP privacy budget ϵ	26
Figure 2.7	Illustrated degradation signals from one of the engines.	28
Figure 2.8	Error vs. Dimensionality d	29
Figure 2.9	Error vs. DP privacy budget ϵ	31
Figure 3.1	Simulation: Accuracy vs. Sample Size n , Linear Regression, Non-DP . . .	57
Figure 3.2	Simulation: Accuracy vs. Sample Size n , Linear Regression, DP	57
Figure 3.3	Simulation: Accuracy vs. Sample Size n , SEV Regression, Non-DP	58
Figure 3.4	Simulation: Accuracy vs. Sample Size n , SEV Regression, DP	58
Figure 3.5	Simulation: Accuracy vs. Sample Size n , Logistic Regression, Non-DP . .	58
Figure 3.6	Simulation: Accuracy vs. Sample Size n , Logistic Regression, DP	58
Figure 3.7	Simulation: Accuracy vs. Correlation ρ , Linear Regression, Non-DP	59
Figure 3.8	Simulation: Accuracy vs. Correlation ρ , Linear Regression, DP	59
Figure 3.9	Simulation: Accuracy vs. Correlation ρ , SEV Regression, Non-DP	60
Figure 3.10	Simulation: Accuracy vs. Correlation ρ , SEV Regression, DP	60
Figure 3.11	Simulation: Accuracy vs. Correlation ρ , Logistic Regression, Non-DP . . .	60
Figure 3.12	Simulation: Accuracy vs. Correlation ρ , Logistic Regression, DP	60
Figure 3.13	Simulation: Computational Cost vs. Privacy Budget	62
Figure 4.1	Example of the degradation data, $F = 2$, truncated at largest mutual t . . .	99
Figure 4.2	Simulation: Error of FPCA Models (Sparsity Level: 20% Missing)	100
Figure 4.3	Simulation: Error of FPCA Models (Sparsity Level: 70% Missing)	103
Figure 4.4	Case Study: Error of FPCA Models (Sparsity Level: 20% Missing)	106
Figure 4.5	Case Study: Error of FPCA Models (Sparsity Level: 70% Missing)	108

CHAPTER

1

INTRODUCTION

The integration of data-driven methodologies into industrial maintenance and prognostics is a critical evolution in the quest to enhance operational reliability and efficiency. Prognostics, defined as the prediction of failure times for systems or components, rely on substantial data analysis, primarily from sensors monitoring machinery health. This predictive capability is crucial for implementing proactive maintenance strategies, significantly impacting cost reduction and efficiency enhancement. However, the use of sensitive or proprietary data in these analyses introduces significant privacy concerns. Differential privacy has emerged as a leading framework to address these concerns, offering a means to perform analyses while protecting individual data contributions. Furthermore, federated learning has gained traction as a distributed learning paradigm that enables collaborative model training without the need for centralized data storage, enhancing data privacy. This dissertation focuses on exploring

differential privacy approaches tailored for the industrial prognostics domain, with a particular emphasis on statistical methods that maintain the balance between privacy and the predictive utility of prognostic models.

1.1 Background

Prognostics plays a pivotal role in the modern industrial maintenance paradigm, enabling organizations to predict failures before they occur. This predictive capability allows for the scheduling of maintenance activities in a manner that is both time-efficient and cost-effective, minimizing unplanned downtime and extending the operational life of machinery and equipment. The effectiveness of prognostic systems depends heavily on the precision of the statistical models used to predict equipment failure, which in turn relies on comprehensive data analysis.

Differential privacy introduces a structured methodology to quantify and mitigate the privacy loss associated with data analyses. It achieves this by ensuring that the output of a database query is insensitive to any single individual's data, effectively masking the participation of any individual in the dataset. The introduction of noise, calibrated to the sensitivity of the data analysis process, is a key mechanism through which differential privacy guarantees are achieved. This mechanism ensures that individual privacy is safeguarded while still allowing for the extraction of valuable aggregate information.

Federated learning is a distributed machine learning paradigm that enables training models on decentralized data without the need for data sharing. In federated learning, each participating entity, referred to as a worker or client, trains a local model on their own data. The local models are then aggregated to form a global model, typically by a central server. This approach allows for collaborative learning while keeping the data localized, thereby enhancing data privacy and security.

1.2 Motivations

The integration of differential privacy and federated learning into industrial prognostics is driven by a compelling need to reconcile critical objectives: maximizing the predictive accuracy of prognostic models, safeguarding the privacy of sensitive data, and enabling collaborative learning without centralized data storage. In industrial settings, the data harnessed for prognostics often encompass proprietary information about equipment performance, manufacturing processes, and operational efficiencies. Unauthorized access to or inference from this data could lead to competitive disadvantages, regulatory repercussions, or breaches of trust with stakeholders. At the same time, the effectiveness of prognostics in preventing unscheduled downtimes, extending equipment lifespans, and optimizing maintenance schedules is heavily reliant on the quality and comprehensiveness of the data analyzed. As such, there is a pressing need for methodologies that can utilize detailed operational data without compromising data confidentiality.

Differential privacy emerges as a solution to this dilemma by offering a systematic approach to quantifying and limiting privacy risks associated with data analysis. By embedding privacy considerations directly into the analysis of prognostic data, differential privacy enables the development of predictive models that are both effective and respectful of privacy constraints. This not only aligns with the ethical and legal imperatives to protect sensitive information but also helps in fostering trust in data-driven industrial processes.

Federated learning complements differential privacy by enabling collaborative model training without the need for centralized data storage. In industrial settings, data is often distributed across multiple entities, such as different factories or manufacturing lines. Federated learning allows these entities to contribute to the development of prognostic models without sharing their raw data, thereby enhancing data privacy and security. The combination of differential privacy and federated learning provides a powerful framework for privacy-preserving and collaborative learning in industrial prognostics.

The motivation for this research, therefore, stems from the need to develop privacy-preserving

and collaborative prognostic tools that can leverage the various data generated in industrial contexts, thereby unlocking new levels of operational efficiency and reliability without compromising on privacy and security.

1.3 Approaches

This dissertation investigates three specific privacy-preserving data analytics approaches designed for application in the industrial prognostics field:

1.3.1 Differentially Private Log-Location-Scale Regression using Functional Mechanism

This approach applies the functional mechanism, a method for achieving differential privacy in statistical models by adding noise to model coefficients, to log-location-scale regression models. These models are well-suited for predicting the lifespan of industrial components. The adaptation of the functional mechanism to these models allows for the generation of differentially private estimates of model parameters, thus facilitating privacy-preserving prognostic analysis.

1.3.2 A Differentially Private Informative Sensor Selection Method

A novel sensor selection methodology is introduced, incorporating a group lasso penalty into a log-location-scale regression model to facilitate sensor selection under privacy constraints. This methodology optimizes sensor usage by identifying the most informative sensors for prognostics while ensuring data privacy through the addition of Laplace noise during the proximal gradient descent optimization process. This not only preserves privacy but also enhances the cost-effectiveness and efficiency of the prognostic system by focusing on key sensors.

1.3.3 Privacy-Preserving Federated Functional Principal Component Analysis for Sparse Data

A federated learning framework is introduced to incorporate functional principal component analysis (FPCA) for industrial prognostics, specifically for decentralized, sparse, and irregular data. This approach addresses the challenges of applying FPCA directly to industrial prognostics data, which is often distributed across multiple entities and may be sparse or irregularly sampled. The proposed federated learning framework allows for collaborative learning of the FPCA model while keeping the data localized. Differential privacy mechanisms, such as gradient clipping and Laplace noise addition, are integrated into the federated learning process to ensure the privacy of individual workers' data. This approach enables the development of privacy-preserving and collaborative FPCA models for industrial prognostics, leveraging the power of distributed data while safeguarding data confidentiality.

1.4 Outline of Dissertation

The structure of this dissertation is designed to present a coherent and comprehensive exploration of differential privacy approaches in industrial prognostics, with each chapter representing a self-contained project that encompasses a literature review, the proposed approach, a simulation study, and a case study. This organization ensures a thorough examination of each methodological contribution within its relevant context, facilitating a deeper understanding of the challenges and solutions related to privacy-preserving prognostics. The dissertation is organized as follows:

Chapter 2 delves into the first project, focusing on the application of the functional mechanism to log-location-scale regression models for differential privacy. This chapter begins with a literature review that situates the approach within the current state of research, followed by a detailed presentation of the proposed approach, including theoretical justifications and algorithmic details. A simulation study then evaluates the approach's performance under various

conditions, and a case study demonstrates its applicability in an industrial context.

Chapter 3 presents the second project, which explores a novel sensor selection methodology incorporating a group lasso penalty into a log-location-scale regression model for differential privacy. Similar to Chapter 2, this chapter starts with a literature review to contextualize the approach, followed by an in-depth explanation of the methodology. A simulation study is conducted to assess the approach's efficacy, and a case study illustrates its practical value in enhancing the efficiency and privacy of industrial prognostics systems.

Chapter 4 focuses on the third project, which introduces a privacy-preserving federated learning framework for FPCA in industrial prognostics. The chapter begins with a literature review on FPCA, federated learning, and their applications in industrial prognostics. The proposed federated learning framework is then presented, detailing the integration of differential privacy mechanisms into the collaborative learning process. A simulation study is conducted to evaluate the performance of the proposed approach under various conditions, such as different levels of data sparsity and privacy budgets. A case study using real-world industrial prognostics data is then presented to demonstrate the practical applicability and benefits of the proposed framework.

Chapter 5 summarizes the work in this thesis, recaps the core ideas and concepts, and offers a summary with major experimental observations. We will wrap up with a few remarks and suggestions for future research.

CHAPTER

2

DIFFERENTIALLY PRIVATE LOG-LOCATION-SCALE REGRESSION USING FUNCTIONAL MECHANISM

2.1 Introduction

Log-location-scale (LLS) regression is a type of regression model that extends the standard linear regression model to accommodate situations where the response variable exhibits skewness or heteroscedasticity. It assumes that the response variable follows a distribution from the location-scale family, which includes a variety of specific distributions that cover most of the failure time distributions of engineering assets in industrial applications. Some

examples of LLS distributions include Normal, Log-Normal, Logistics, Log-Logistics, Smallest Extreme Value (SEV), and Weibull. Due to the versatility afforded by these distributions, LLS regression has been widely utilized in reliability engineering (Meeker and Escobar 2014; Doray 1994; Hong et al. 2010, 2015) and industrial prognostics (Fang et al. 2017, 2019a; Zhou and Fang 2023; Fang et al. 2019b).

Similar to other statistical learning models, LLS regression needs a considerable amount of data for model training to achieve a desirable performance. However, the available training data related to failures from a single company/factory is usually insufficient to reliably train an LLS model. To address this challenge, companies may purchase a well-trained prognostic model from Original Equipment Manufacturers (OEMs). This is attributed to the fact that OEMs frequently lease a substantial number of equipment to customers and own the condition monitoring data of these leased equipment, providing them with sufficient failure-related data to train a dependable prognostic model. The trained models can be provided as a standalone product through subscription, ensuring that the model is regularly updated as additional failure-related data becomes available. The subscription allows OEMs to deliver reliable prognostic models to customers while safeguarding the privacy of their data—customers have access to the trained models without knowledge of the OEM’s training data. However, recent research indicates that it is possible to infer information about the training data from the trained models by employing cyberattack techniques such as model inversion (Fredrikson et al. 2015; Wang et al. 2015; Hidano et al. 2017). To tackle this challenge, research has shown that differential privacy is an effective technique for safeguarding data privacy (Wang et al. 2015; Zhang et al. 2020; Abadi et al. 2016). In alignment with this perspective, this article proposes privacy-preserving LLS regression based on differential privacy.

Differential Privacy (DP) is a foundational concept in privacy-preserving data analysis that aims to protect data privacy while extracting meaningful insights from datasets. It provides a rigorous mathematical framework for ensuring that the inclusion or exclusion of any data point does not unduly impact the outcome of a computation. Fundamentally, this is achieved

by adding controlled noise to the results of statistical analyses. For example, OEMs may employ DP to perturb estimated regression coefficients with random noise before transmitting them to customers. The amount of noise introduced in DP is determined by two factors: *sensitivity* and *privacy budget*. The first factor, sensitivity, is determined by the training data. It measures the maximum fluctuation in the output (e.g., the maximum change of regression coefficients) when a single data point is removed from or added to the training dataset. It's evident that to safeguard the privacy of training data, a larger noise is required when the sensitivity is higher. The second factor, privacy budget, is chosen by those who train the model. It controls the upper limit of the probability of potential information disclosure (Dwork et al. 2006). This is necessary because, mathematically, DP (and many other privacy protection techniques) typically cannot guarantee a 100% prevention of data leakage. Instead, they use a parameter (i.e., privacy budget) to control the probability that the data will not be leaked. This probability can be low (close to zero) or high. If a lower probability is sought, a higher degree of noise needs to be introduced into the model. Consequently, the performance of the model will be compromised more. In contrast, when the probability is high, the level of added noise is low, resulting in the performance of the DP-based model closely resembling that of the model without DP applied. Therefore, there exists a tradeoff between the probability of data leakage and the extent to which the model's performance is compromised.

In DP, the algorithm or procedure through which the noise is incorporated is referred to as *mechanism*. One possible mechanism for regression analysis is to perturb the estimated regression coefficients with a random noise drawn from distributions like Laplace, Gaussian, or exponential. However, implementing this mechanism is often impractical because accurately determining the noise's amplitude relies on calculating the sensitivity, which usually is very challenging. To address this challenge, the commonly used mechanism in regression analysis is the *functional mechanism*, which introduces noise to the objective function of the optimization problem linked to regression parameter estimation, rather than directly perturbing the regression coefficients themselves (Zhang et al. 2012). In this article, the proposed

differentially private LLS regression is based on the functional mechanism. The parameter estimation of LLS regression is typically achieved by using maximum likelihood estimation (MLE), which solves an optimization problem whose objective function is the log-likelihood function constructed using the training data and unknown regression parameters. To integrate differential privacy into LLS regression, we begin by decomposing the objective function (i.e., the log-likelihood function) through Taylor expansion, truncating at the second order. Then, we introduce noise to perturb the weights of the Taylor series. Subsequently, we reconstruct the perturbed objective function using the modified weights. Finally, we optimize the perturbed objective function to obtain the parameters of the LLS regression. We will derive the sensitivity of weights, which determines the magnitude of the noise introduced. Additionally, we will prove that the proposed regression method complies with differential privacy.

Commonly employed distributions in LLS regression include normal, log-normal, logistic, log-logistic, smallest extreme value (SEV), and Weibull. When the response variable is from a normal, logistic, or SEV distribution, the model is referred to as location-scale regression. In contrast, when the response variable is from a log-normal, log-logistic, or Weibull distribution, it is called log-location-scale regression. Location-scale regression and log-location-scale regression can be easily transformed into each other. For example, log-normal regression can be converted into normal regression by applying the logarithm to its response variable. Similarly, Weibull (or log-logistic) regression can be transformed into SEV (or logistic) regression by taking the logarithm of its response variable. Given that Zhang et al. (2012) have explored differentially private normal regression, this article concentrates on the development of differentially private logistic regression and SEV regression (applicable to log-logistic regression and Weibull regression as well). It's important to clarify that the logistic regression investigated in this paper differs from logistic regression for classification (Wright 1995). In this article, logistic regression refers to the response variable following a logistic distribution, a member of the location-scale family. Logistic regression for classification, on the other hand, involves a response variable from a binary or binomial distribution.

The remainder of this chapter is organized as follows. Section 2.2 introduces the necessary preliminaries, encompassing LLS regression, differential privacy, and the functional mechanism. Section 2.3 outlines the proposed differentially private LLS regression. Following that, Sections 2.4 and 2.5 employ simulated and aircraft engine data from the NASA data repository to assess the performance of the proposed differentially private LLS regression. Finally, Section 2.6 provides concluding remarks.

2.2 Preliminaries

In this section, we first review (log)-location-scale regression and then revisit some fundamental concepts of differential privacy.

2.2.1 LLS Regression

Considering a dataset containing n samples, we denote the data as $\{\tilde{y}_i, \{x_{ij}\}_{j=1}^d\}_{i=1}^n$, where \tilde{y}_i is the dependent variable (i.e., response variable), which is corresponding to the time-to-failure (TTF) in reliability analysis and prognostics; $\{x_{ij}\}_{j=1}^d$ are the d -dimensional independent variables (i.e., predictors). Each \tilde{y}_i is assumed to be sampled from a distribution from the (log)-location-scale family. In other words, y_i is a random variable whose cumulative distribution function (CDF) is of the form

$$\Pr(y_i \leq t) = \Omega\left(\frac{t - \mu_i}{\sigma}\right), \quad (2.1)$$

where $y_i = \tilde{y}_i$ if \tilde{y}_i is from a distribution in the location-scale family and $y_i = \log(\tilde{y}_i)$ if \tilde{y}_i is from a distribution in the log-location-scale family. $\Omega(\cdot)$ is the CDF of a standard distribution in the location-scale family. μ_i is referred to as the location parameter, and σ is known as the scale parameter. Similar to the standard practices in regression analysis, it is assumed that the location parameter μ_i is determined by the independent variables, and the scale parameter σ is not associated with the predictors. Denote $\mu_i = \beta_0 + \sum_{j=1}^d x_{ij}\beta_j$, or $\mu_i = \sum_{j=0}^d x_{ij}\beta_j$ where

$x_{i0} = 1$ for simplicity, the CDF of y_i can be expressed as follows

$$\Pr(y_i \leq t) = \Omega\left(\frac{t - \sum_{j=0}^d x_{ij}\beta_j}{\sigma}\right), \quad (2.2)$$

where $\{\beta_j\}_{j=0}^d$ are the unknown regression coefficients. These regression coefficients and the scale parameter σ are usually estimated using maximum likelihood estimation (MLE). The likelihood function can be denoted as follows

$$L(\{\beta_j\}_{j=0}^d, \sigma) = \prod_{i=1}^n \frac{1}{\sigma} \omega\left(\frac{y_i - \sum_{j=0}^d x_{ij}\beta_j}{\sigma}\right), \quad (2.3)$$

where $\omega(\cdot)$ is the probability density function (PDF) of a standard distribution in the location-scale family. For example, $\omega(z) = 1/\sqrt{2\pi} \exp(-z^2/2)$ for normal distribution, $\omega(z) = \exp(z)/(1 + \exp(z))^2$ for logistic distribution, and $\omega(z) = \exp(z)(z - \exp(z))$ for SEV distribution. The estimation of $(\{\beta_j\}_{j=0}^d, \sigma)$ can be achieved by maximizing the log-likelihood function $\ell = \log(L(\{\beta_j\}_{j=0}^d, \sigma))$.

2.2.2 Differential Privacy

In differential privacy, a dataset can be seen as a collection of entries corresponding to individuals. For example, a dataset containing the TTF and predictors of n machines (i.e., $\{\tilde{y}_i, \{x_{ij}\}_{j=1}^d\}_{i=1}^n$) can be seen as n entries. Differential privacy guarantees that the modification (e.g., inclusion, exclusion, revision) of a single entry has no discernible statistical impact on the output. To define differential privacy, we first introduce the concept of neighboring datasets. Two datasets, D and D' , are considered neighboring if they have the same number of entries while differing in one entry (corresponding to the data of one sample). With this concept, ϵ -Differential Privacy is defined as follows.

Definition 1 (*ϵ -Differential Privacy (Dwork et al. 2006)*). An algorithm \mathcal{A} satisfies ϵ -differential

privacy, iff for any output O of \mathcal{A} and for any two neighbor databases D and D' , we have

$$\Pr[\mathcal{A}(D) = O] \leq e^\epsilon \cdot \Pr[\mathcal{A}(D') = O] \quad (2.4)$$

Here, the operator $\Pr[\cdot]$ represents probability. $\mathcal{A}(\cdot)$ represents the mechanism, which denotes the method employed to introduce noise. Loosely speaking, $\Pr[\mathcal{A}(D) = O]$ means the probability that the output is O using dataset D and mechanism $\mathcal{A}(\cdot)$. Taking regression analysis using the functional mechanism as an example, $\Pr[\mathcal{A}(D) = O]$ means that the probability that the noisy weights are O using training dataset D . Similarly, $\Pr[\mathcal{A}(D') = O]$ means that the probability that the noisy weights are also O using training dataset D' . Equation (2.4) quantifies the proximity of the probabilities to obtaining the same set of noisy weights when utilizing datasets D and D' . The proximity is controlled by e^ϵ , where ϵ is known as the *privacy budget*. A smaller value of ϵ enforces a stronger privacy guarantee. For example, as ϵ approaches 0, e^ϵ approaches 1. As a result, the ratio between the two probabilities is closer to 1, and thus the two datasets become more indistinguishable.

In the aforementioned example of regression using the functional mechanism, noise is incorporated into the weights. However, a crucial aspect that has not been addressed is determining the magnitude of the noise. The magnitude of the noise is determined by the dataset. Mathematically, it is associated with global sensitivity, the definition of which is provided below.

Definition 2 (*Global Sensitivity (Dwork et al. 2006)*) The global sensitivity of a function $f : D^n \rightarrow \mathbf{R}^d$

$$GS_f(D) = \max_{D, D'} \|f(D) - f(D')\|_1,$$

where D and D' are neighboring datasets. $f(\cdot)$ is a function that takes a dataset D or D' as the input, and the outputs are denoted as $f(D)$ and $f(D')$, respectively. In the functional mechanism, $f(D)$ and $f(D')$ are the values of the weights using dataset D and D' , respectively. Thus, $GS_f(D)$ is the maximum changes that could occur in the weights when one sample in D is replaced.

2.3 Differentially Private LLS Regression

In this section, we introduce differentially private (log)-location-scale regression. The commonly utilized distributions in the LLS family include normal, log-normal, logistic, log-logistic, SEV, and Weibull. Since Zhang et al. (2012) have investigated differentially private normal regression, this article focuses on the development of differentially private logistic regression and SEV regression, which can be applied to log-logistic regression and Weibull regression as well.

The proposed differentially private LLS regression relies on the functional mechanism, which works by perturbing the objective function used in the parameter estimation of LLS regression models. In other words, rather than solving the original log-likelihood function for parameter estimation, we optimize a perturbed version of it. This is achieved by first decomposing the objective function, specifically the log-likelihood function, through a Taylor expansion that is truncated at the second order. Then, noise is introduced to perturb the weights of the Taylor series. Subsequently, we reconstruct the perturbed objective function using the modified weights. Finally, optimization of the perturbed objective function is performed to obtain the parameters of the LLS regression. As mentioned earlier, the magnitude of the noise is directly linked to the sensitivity of the objective function. Thus, we will derive the sensitivity of the LLS regression models. Additionally, we will demonstrate that the proposed regression methods satisfy ϵ -differential privacy.

Before constructing the proposed differentially private LLS regression models, we begin by standardizing the data. Following the notation in Section 2.2.1, let's consider a dataset consisting of n samples, denoted as $\{y_i, \{\tilde{x}_{ij}\}_{j=1}^d\}_{i=1}^n$, where $\{\tilde{x}_{ij}\}_{j=1}^d$ are the d -dimensional predictors and y_i is the response (TTF or the logarithmic TTF). To scale the j th predictor, we use $x_{ij} = \frac{\tilde{x}_{ij} - \alpha_j}{(\beta_j - \alpha_j) \cdot \sqrt{d}}$, where $\alpha_j = \min\{\{\tilde{x}_{ij}\}_{i=1}^n\}$ and $\beta_j = \max\{\{\tilde{x}_{ij}\}_{i=1}^n\}$ denotes the minimum and maximum values of the j th predictor, respectively. This standardization ensures $\sqrt{\sum_{i=1}^d x_{id}^2} \leq 1$, which will be employed to determine the global sensitivity of the proposed differentially private LLS regression. Additionally, we standardize the response variable such that $y_i \in [-1, 1]$ for all i .

2.3.1 Differentially Private SEV Regression

Following the notations in the MLE function in Equation (2.3), the log-likelihood function of SEV regression can be expressed as follows

$$\ell(\{\beta_j\}_{j=0}^d, \sigma) = -n \log \sigma + \sum_{i=1}^n \frac{y_i - \sum_{j=0}^d \beta_j x_{ij}}{\sigma} - \sum_{i=1}^n \exp\left(\frac{y_i - \sum_{j=0}^d \beta_j x_{ij}}{\sigma}\right), \quad (2.5)$$

which is not concave. To make it a concave function, we apply the following transformation: $q = 1/\sigma$ and $p_j = \beta_j q$. As a result, the log-likelihood function can be re-expressed as

$$\ell(\{p_j\}_{j=0}^d, q) = n \log q + \sum_{i=1}^n \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right) - \sum_{i=1}^n \exp\left(y_i q - \sum_{j=0}^d p_j x_{ij} \right). \quad (2.6)$$

To integrate differential privacy with SEV regression, as shown in Proposition 1, we first employ Taylor expansion to expand the log-likelihood function in Equation (2.6) as a weighted combination of polynomial functions of $\{p_j\}_{j=0}^d$ and q .

Proposition 1 *By applying Taylor expansion and truncating it at the second order, Equation (2.6) can be re-written as follows*

$$\begin{aligned} \tilde{\ell}(\{p_j\}_{j=0}^d, q) = & -\frac{5n}{2} + 2nq - \frac{1}{2} \left(n + \sum_{i=1}^n y_i^2 \right) q^2 + \sum_{i=1}^n \sum_{j=0}^d y_i x_{ij} p_j q - \frac{1}{2} \sum_{i=1}^n \sum_{j=0}^d x_{ij}^2 p_j^2 \\ & - \frac{1}{2} \sum_{i=1}^n \left(\sum_{h=0, h \neq j}^d \sum_{j=0}^d x_{ij} x_{ih} p_j p_h \right). \end{aligned} \quad (2.7)$$

The proof is available in the Appendix. Let $w_1 = -\frac{5n}{2}$, $w_q = 2n$, $w_{q^2} = -\frac{1}{2} \left(n + \sum_{i=1}^n y_i^2 \right)$, $\{w_{p_j q} = \sum_{i=1}^n y_i x_{ij}\}_{j=0}^d$, $\{w_{p_j^2} = -\frac{1}{2} \sum_{i=1}^n x_{ij}^2\}_{j=0}^d$, and $\{w_{p_j p_h} = -\frac{1}{2} \sum_{i=1}^n x_{ij} x_{ih}\}_{j=0, h \neq j}^d$, Equation (2.7) can be rewritten as

$$\tilde{\ell}(\{p_j\}_{j=0}^d, q) = w_1 + w_q q + w_{q^2} q^2 + \sum_{j=0}^d w_{p_j q} p_j q + \sum_{j=0}^d w_{p_j^2} p_j^2 + \sum_{h=0, h \neq j}^d \sum_{j=0}^d w_{p_j p_h} p_j p_h.$$

To incorporate differential privacy into SEV regression, we then introduce noise to perturb the weights. As discussed earlier, the magnitude of the noise is determined by the sensitivity and the privacy budget. Proposition 2 provides an upper bound for the sensitivity of the polynomial coefficients of the function $\tilde{\ell}(\{p_j\}_{j=0}^d, q)$ in Equation (2.7).

Proposition 2 *Suppose D and D' be any two neighbor databases. Let $\tilde{\ell}_D(\{p_j\}_{j=0}^d, q)$, and $\tilde{\ell}_{D'}(\{p_j\}_{j=0}^d, q)$ be the objective functions on D and D' , respectively. Also, let the corresponding weights of $\tilde{\ell}_D(\{p_j\}_{j=0}^d, q)$ be $w_1^D, w_q^D, w_{q^2}^D, \{w_{p_j q}^D\}_{j=1}^d, \{w_{p_j^2}^D\}_{j=0}^d, \{\{w_{p_j p_h}^D\}_{j=0}^d\}_{h=0, h \neq j}^d$, and the weights of $\tilde{\ell}_{D'}(\{p_j\}_{j=0}^d, q)$ be $w_1^{D'}, w_q^{D'}, w_{q^2}^{D'}, \{w_{p_j q}^{D'}\}_{j=0}^d, \{w_{p_j^2}^{D'}\}_{j=0}^d, \{\{w_{p_j p_h}^{D'}\}_{j=0}^d\}_{h=0, h \neq j}^d$. Then, we have the following inequality:*

$$\begin{aligned} & \|w_1^D - w_1^{D'}\|_1 + \|w_q^D - w_q^{D'}\|_1 + \|w_{q^2}^D - w_{q^2}^{D'}\|_1 + \sum_{j=0}^d \|w_{p_j q}^D - w_{p_j q}^{D'}\|_1 \\ & + \sum_{j=0}^d \|w_{p_j^2}^D - w_{p_j^2}^{D'}\|_1 + \sum_{j=0}^d \sum_{h=0, h \neq j}^d \|w_{p_j p_h}^D - w_{p_j p_h}^{D'}\|_1 \leq 4 + 4\sqrt{d} + d. \end{aligned} \quad (2.8)$$

The proof of Proposition 2 can be found in the Appendix. Based on the inequality provided by Proposition 2 and let $\Delta = 4 + 4\sqrt{d} + d$, we perturb each weight in Equation (2.7) by adding $\text{Lap}(\Delta/\epsilon)$ to it, where $\text{Lap}(\Delta/\epsilon)$ represents a random variable drawn from the Laplace distribution with zero mean and scale Δ/ϵ , and ϵ is the privacy budget. We denote the perturbed weights as $\tilde{w}_1, \tilde{w}_q, \tilde{w}_{q^2}, \{\tilde{w}_{p_j q}\}_{j=0}^d, \{\tilde{w}_{p_j^2}\}_{j=0}^d$, and $\{\{\tilde{w}_{p_j p_h}\}_{j=0}^d\}_{h=0, h \neq j}^d$, where, for instance, $\tilde{w}_1 = w_1 + \text{Lap}(\Delta/\epsilon)$. As a result, the perturbed objective function for the parameter estimation of SEV regression can be denoted as follows:

$$\tilde{\ell}(\{p_j\}_{j=0}^d, q) = \tilde{w}_1 + \tilde{w}_q + \tilde{w}_{q^2} q^2 + \sum_{j=0}^d \tilde{w}_{p_j q} p_j q + \sum_{j=0}^d \tilde{w}_{p_j^2} p_j^2 + \sum_{j=0}^d \sum_{h=0, h \neq j}^d \tilde{w}_{p_j p_h} p_j p_h. \quad (2.9)$$

Optimizing $\max_{\{p_j\}_{j=0}^d, q} \tilde{\ell}(\{p_j\}_{j=0}^d, q)$ yields the estimated parameters $\{\{\hat{p}_j\}_{j=0}^d, \hat{q}\}$, which can be easily transformed back to the regression parameters in SEV regression: $\hat{\sigma} = 1/\hat{q}$ and $\hat{\beta}_j = \hat{p}_j \hat{\sigma}$. We summarize the parameter estimation algorithm for the differentially private SEV regression using the functional mechanism in Algorithm 1.

Algorithm 1 Parameter Estimation for Differentially Private SEV Regression

Require: Database $\{\tilde{y}_i, \{x_{ij}\}_{j=0}^d\}_{i=1}^n$, privacy budget ϵ

Ensure: ϵ -DP protected parameter estimates $\hat{\sigma}, \{\hat{\beta}_j\}_{j=0}^d$

- 1: Set $\Delta = 4 + 4\sqrt{d} + d$
 - 2: $\tilde{w}_1 = w_1 + \text{Lap}(\Delta/\epsilon)$, $\tilde{w}_q = w_q + \text{Lap}(\Delta/\epsilon)$, $\tilde{w}_{q^2} = w_{q^2} + \text{Lap}(\Delta/\epsilon)$
 - 3: $\{\tilde{w}_{p_j q} = w_{p_j q} + \text{Lap}(\Delta/\epsilon)\}_{j=0}^d$
 - 4: $\{\tilde{w}_{p_j^2} = w_{p_j^2} + \text{Lap}(\Delta/\epsilon)\}_{j=0}^d$
 - 5: $\{\{\tilde{w}_{p_j p_h} = w_{p_j p_h} + \text{Lap}(\Delta/\epsilon)\}_{j=0}^d\}_{h=0, h \neq j}^d$
 - 6: Solving $\{\{\hat{p}_j\}_{j=0}^d, \hat{q}\} = \max_{\{p_j\}_{j=0}^d, q} \tilde{\ell}(\{p_j\}_{j=0}^d, q)$, where $\tilde{\ell}(\{p_j\}_{j=0}^d, q)$ is defined in Equation (2.9)
 - 7: Set $\hat{\sigma} = 1/\hat{q}$, $\{\hat{\beta}_j = \hat{p}_j \hat{\sigma}\}_{j=0}^d$.
 - 8: Return $\hat{\sigma}, \{\hat{\beta}_j\}_{j=0}^d$
-

Proposition 3 shows that Algorithm 1 ensures ϵ -differential privacy, with the proof available in the Appendix.

Proposition 3 *Algorithm 1 satisfies ϵ -differential privacy.*

2.3.2 Differentially Private Logistic Regression

The log-likelihood function of logistic regression can be expressed as follows

$$\ell(\{\beta_j\}_{j=0}^d, \sigma) = -n \log \sigma + \sum_{i=1}^n \frac{y_i - \sum_{j=0}^d \beta_j x_{ij}}{\sigma} - 2 \sum_{n=1}^n \log \left(1 + \exp \left(\frac{y_i - \sum_{j=0}^d \beta_j x_{ij}}{\sigma} \right) \right). \quad (2.10)$$

which is also not concave. Thus, the following transformation is applied to make it concave $q = 1/\sigma$ and $p_j = \beta_j q$. As a result, the log-likelihood function can be re-expressed as

$$\ell(\{p_j\}_{j=0}^d, q) = n \log q + \sum_{i=1}^n \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right) - 2 \sum_{i=1}^n \log \left(1 + \exp \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right) \right). \quad (2.11)$$

Similar to SEV regression, to integrate differential privacy into the logistic regression, we first express the function $\ell(p_j = 0^d, q)$ in Equation (2.11) as a polynomial function of $\{p_j\}_{j=0}^d$ and q . This expansion is given in Proposition 4.

Proposition 4 *By applying Taylor expansion and truncating it at the second order, Equation (2.11) can be rewritten as*

$$\begin{aligned} \tilde{\ell}(\{p_j\}_{j=0}^d, q) = & -n\left(\frac{3}{2} + 2\log 2\right) + 2nq - \left(\frac{n}{2} + \frac{1}{4}\sum_{i=1}^n y_i^2\right)q^2 + \frac{1}{2}\sum_{i=1}^n \sum_{j=0}^d y_i x_{ij} p_j q \\ & - \frac{1}{4}\sum_{i=1}^n \sum_{j=0}^d x_{ij}^2 p_j^2 - \frac{1}{4}\sum_{i=1}^n \left(\sum_{h=0, h \neq j}^d \sum_{j=0}^d x_{ij} x_{ih} p_j p_h\right). \end{aligned} \quad (2.12)$$

The proof of Proposition 4 can be found in the Appendix. Denoting $w_1 = -n\left(\frac{3}{2} + 2\log 2\right)$, $w_q = 2n$, $w_{q^2} = -\left(\frac{n}{2} + \frac{1}{4}\sum_{i=1}^n y_i^2\right)$, $\{w_{p_j q} = \frac{1}{2}\sum_{i=1}^n y_i x_{ij}\}_{j=0}^d$, $\{w_{p_j^2} = -\frac{1}{4}\sum_{i=1}^n x_{ij}^2\}_{j=0}^d$, $\{\{w_{p_j p_h} = -\frac{1}{4}\sum_{i=1}^n x_{ij} x_{ih}\}_{j=0}^d\}_{h=0, h \neq j}^d$, Equation (2.12) can be rewritten as $\tilde{\ell}(\{p_j\}_{j=0}^d, q) = w_1 + w_q q + w_{q^2} q^2 + \sum_{j=0}^d w_{p_j q} p_j q + \sum_{j=0}^d w_{p_j^2} p_j^2 + \sum_{j=0}^d \sum_{h=0, h \neq j}^d w_{p_j p_h} p_j p_h$. To integrate differential privacy with logistic regression, we then introduce noise to the weights. To determine the amplitude of noise, we first provide an upper bound for the sensitivity of the polynomial coefficients of the function in Equation (2.12) in Proposition 5 below.

Proposition 5 *Suppose D and D' are any two neighbor databases. Let $\tilde{\ell}_D(\{p_j\}_{j=0}^d, q)$ and $\tilde{\ell}_{D'}(\{p_j\}_{j=0}^d, q)$ be the objective functions on D and D' , respectively. Also, let the corresponding weights of $\tilde{\ell}_D(\{p_j\}_{j=0}^d, q)$ be $w_1^D, w_q^D, w_{q^2}^D, \{w_{p_j q}^D\}_{j=0}^d, \{w_{p_j^2}^D\}_{j=0}^d, \{\{w_{p_j p_h}^D\}_{j=0}^d\}_{h=0, h \neq j}^d$ and the weights of $\tilde{\ell}_{D'}(\{p_j\}_{j=0}^d, q)$ be $w_1^{D'}, w_q^{D'}, w_{q^2}^{D'}, \{w_{p_j q}^{D'}\}_{j=0}^d, \{w_{p_j^2}^{D'}\}_{j=0}^d, \{\{w_{p_j p_h}^{D'}\}_{j=0}^d\}_{h=0, h \neq j}^d$. Then, we have the following inequality:*

$$\begin{aligned} & \|w_1^D - w_1^{D'}\|_1 + \|w_q^D - w_q^{D'}\|_1 + \|w_{q^2}^D - w_{q^2}^{D'}\|_1 + \sum_{j=0}^d \|w_{p_j q}^D - w_{p_j q}^{D'}\|_1 + \sum_{j=0}^d \|w_{p_j^2}^D - w_{p_j^2}^{D'}\|_1 \\ & + \sum_{j=0}^d \sum_{h=0, h \neq j}^d \|w_{p_j p_h}^D - w_{p_j p_h}^{D'}\|_1 \leq 2 + 2\sqrt{d} + \frac{1}{2}d. \end{aligned} \quad (2.13)$$

The proof of Proposition 5 is provided in the Appendix. According to Proposition 5, each weight in Equation (2.12) can be perturbed by adding a random value drawn from the Laplace distribution with zero mean and scale Δ/ϵ , where ϵ is the privacy budget, and $\Delta = 2 + 2\sqrt{d} + \frac{1}{2}d$. If we denote the perturbed weights as $\tilde{w}_1, \tilde{w}_q, \tilde{w}_{q^2}, \{\tilde{w}_{p_j q}\}_{j=0}^d, \{\tilde{w}_{p_j^2}\}_{j=0}^d$, and $\{\{\tilde{w}_{p_j p_h}\}_{j=0}^d\}_{h=0, h \neq j}^d$,

then $\tilde{w}_1 = w_1 + \text{Lap}(\Delta/\epsilon)$, $\tilde{w}_q = w_q + \text{Lap}(\Delta/\epsilon)$, \dots . As a result, the new objective function for the parameter estimation of logistic regression can be denoted as follows:

$$\tilde{\ell}(\{p_j\}_{j=0}^d, q) = \tilde{w}_1 + \tilde{w}_q + \tilde{w}_{q^2} q^2 + \sum_{j=0}^d \tilde{w}_{p_j q} p_j q + \sum_{j=0}^d \tilde{w}_{p_j^2} p_j^2 + \sum_{j=0}^d \sum_{h=0, h \neq j}^d \tilde{w}_{p_j p_h} p_j p_h. \quad (2.14)$$

Solving the optimization problem $\max_{\{p_j\}_{j=0}^d, q} \tilde{\ell}(\{p_j\}_{j=0}^d, q)$ yields the estimated parameters $\{\{\hat{p}_j\}_{j=0}^d, \hat{q}\}$, which can then be transformed back to the regression parameters in logistic regression: $\hat{\sigma} = 1/\hat{q}$ and $\hat{\beta}_j = \hat{p}_j \hat{\sigma}$. We summarize the parameter estimation algorithm for the differentially private logistic regression using the functional mechanism in Algorithm 2.

Algorithm 2 Parameter Estimation for Differentially Private Logistic Regression

Require: Database $\{\tilde{y}_i, \{x_{ij}\}_{j=0}^d\}_{i=1}^n$, privacy budget ϵ

Ensure: ϵ -DP protected parameter estimates $\hat{\sigma}, \{\hat{\beta}_j\}_{j=0}^d$

- 1: Set $\Delta = 2 + 2\sqrt{d} + \frac{1}{2}d$
 - 2: $\tilde{w}_1 = w_1 + \text{Lap}(\Delta/\epsilon)$, $\tilde{w}_q = w_q + \text{Lap}(\Delta/\epsilon)$, $\tilde{w}_{q^2} = w_{q^2} + \text{Lap}(\Delta/\epsilon)$
 - 3: $\{\tilde{w}_{p_j q} = w_{p_j q} + \text{Lap}(\Delta/\epsilon)\}_{j=0}^d$
 - 4: $\{\tilde{w}_{p_j^2} = w_{p_j^2} + \text{Lap}(\Delta/\epsilon)\}_{j=0}^d$
 - 5: $\{\{\tilde{w}_{p_j p_h} = w_{p_j p_h} + \text{Lap}(\Delta/\epsilon)\}_{j=0}^d\}_{h=0, h \neq j}^d$
 - 6: Solving $\{\{\hat{p}_j\}_{j=0}^d, \hat{q}\} = \max_{\{p_j\}_{j=0}^d, q} \tilde{\ell}(\{p_j\}_{j=0}^d, q)$, where $\tilde{\ell}(\{p_j\}_{j=0}^d, q)$ is defined in Equation (2.14)
 - 7: Set $\hat{\sigma} = 1/\hat{q}$, $\{\hat{\beta}_j = \hat{p}_j \hat{\sigma}\}_{j=0}^d$.
 - 8: Return $\hat{\sigma}, \{\hat{\beta}_j\}_{j=0}^d$
-

Proposition 6 indicates that Algorithm 2 ensures ϵ -differential privacy, and the proof can be found in the Appendix.

Proposition 6 *Algorithm 2 satisfies ϵ -differential privacy.*

2.4 Simulation Study

In this section, we conduct simulation studies to evaluate the performance of the proposed differentially private LLS regression models.

2.4.1 Experimental settings

We generate data for n samples (the specific values of n will be discussed later). First, each entry of the predictors of sample i , $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{id})^\top$, is randomly drawn from a standard normal distribution, $\mathcal{N}(0, 1)$. Here, d is the dimension of features, whose specific values will be discussed later as well. Next, we generate the regression coefficients $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_d)^\top$, each element of which is generated from a standard normal distribution, $\mathcal{N}(0, 1)$. Third, the response of sample i is generated using $\tilde{y}_i = (1, \mathbf{x}_i^\top)\boldsymbol{\beta} + \varepsilon_i$, where ε_i is a random number sampled from either an SEV or logistic distribution (depending on the regression model being evaluated), with the location parameter set to 0 and the scale parameter set to 1.

We use the simulated data to assess the performance of the proposed differentially private LLS regression, denoted as "DP." The benchmark employed in this study is the classical LLS regression, designated as "Non-DP." Our interest lies in investigating how the proposed method performs under different conditions, including (1) predictor dimensions (i.e., d), (2) training sample sizes (i.e., n), and (3) privacy budgets (i.e., ϵ). These factors are expected to influence the performance of the proposed DP method. We summarize the specific settings of these three factors used in this simulation study as follows.

- To examine the impact of predictor dimension on the performance of the proposed DP LLS regression, the tested values for d are 20, 22, 24, 26, 28, 30, 32, 34, 36 and 38. For SEV regression, we set $n = 1e4$. For logistic regression, $n = 5e3$. We have $\epsilon = 0.5$ for both models.
- To explore the influence of sample size, the values of n to be tested are $1e4$, $1.5e4$, $2e4$, $2.5e4$, $3e4$, $3.5e4$, $4e4$, $4.5e4$, $5e4$, $5.5e4$, and $6e4$ for SEV regression. For logistic regres-

sion, the tested n values are $5e3$, $6e3$, $7e3$, $8e3$, $9e3$, $1e4$, $2e4$, $3e4$, $4e4$, $5e4$ and $6e4$. We have $d = 35$ and $\epsilon = 0.5$ for both models.

- To assess the influence of the privacy budget, the tested values of ϵ are 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, and 2, while we set $d = 25$ for SEV regression and $d = 38$ for logistic regression. For both regression models, we set $n = 1e4$.

For each of the trials above, we randomly select 80% of the data for model training. The remaining 20% data are used for testing. The metric used to evaluate the performance is the absolute prediction error, which is computed using $|(\hat{y}_i - \tilde{y}_i)/\tilde{y}_i|$, where \hat{y}_i and \tilde{y}_i are the predicted and true response of sample i , respectively. Each trial will be repeated 100 times, by setting the random seed as the repetition number. The prediction errors of the 100 repetitions are reported in a single boxplot.

2.4.2 Results and Analysis

Prediction Error vs. Predictor Dimension

In the initial set of experiments, we investigate the performance of DP and non-DP models across various levels of feature dimensions. The dimensionality is crucial as it determines the sensitivity, influencing the amplitude of noise introduced to the objective function of parameter estimation. As discussed in Section 2, we introduce noise using $\text{Lap}((4 + 4\sqrt{d} + d)/\epsilon)$ for the SEV regression model and $\text{Lap}((2 + 2\sqrt{d} + \frac{1}{2}d)/\epsilon)$ for the logistic model, where d represents the feature dimension. Keeping the sample size constant at $1e4$ for SEV regression and $5e3$ for logistic regression, and setting the privacy budget to $\epsilon = 0.5$. For both regressions, we conducted tests for d values ranging from 20 to 38, in increments of 2. We present the prediction errors for SEV regression in Figure 2.1 and for Logistic regression in Figure 2.2.

In Figures 2.1 and 2.2, the horizontal axis is the dimension of predictors, and the vertical axis is the prediction error. For each predictor dimension, the left boxplot represents the non-DP model, while the right one corresponds to the DP model. Figures 2.1 and 2.2 reveal that the

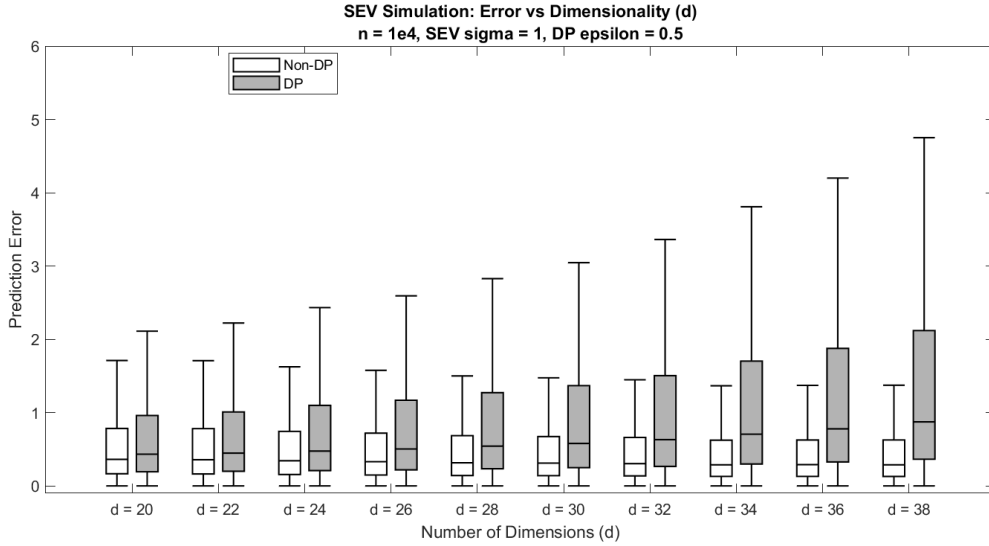


Figure 2.1: SEV Regression: Prediction Error vs. Dimensionality d

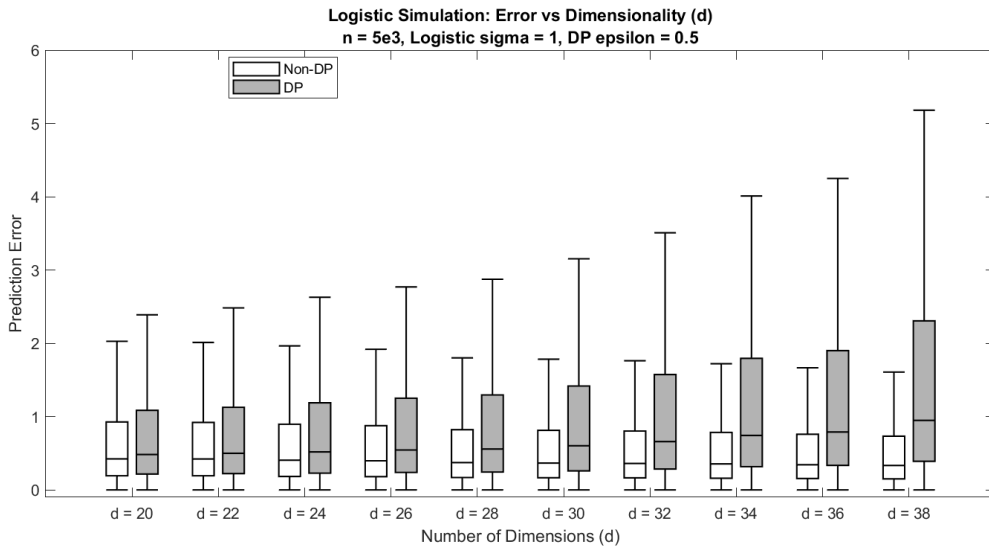


Figure 2.2: Logistic Regression: Prediction Error vs. Dimensionality d

prediction errors of the proposed DP models deteriorate with the increase in feature dimension, whereas the prediction errors of the non-DP models remain consistent with the rise in feature dimension. For instance, when $d = 20$, the median errors for both non-DP and DP models are approximately 0.45 for Logistic regression. However, as d expands to 26, the median error for the DP model increases to 0.54, while the non-DP model maintains a stable error of 0.40. This trend persists with further increases in dimensionality. At $d = 32$, the median error in the DP

model rises to 0.66, and at $d = 36$, it further grows to 0.79. In contrast, the errors in the non-DP model remain relatively unaffected by the changes in d , emphasizing the widening gap in prediction errors between DP and non-DP models as the feature dimensionality increases. This increase in prediction errors of DP models is anticipated and can be attributed to the escalating magnitude of Laplace noise added to the objective function for parameter estimation. Recall that the noise added to the SEV model is characterized by a Laplace distribution with a scale parameter $((4 + 4\sqrt{d} + d)/\epsilon)$, and the noise for the logistic model follows a Laplace distribution with a scale parameter $((2 + 2\sqrt{d} + \frac{1}{2}d)/\epsilon)$. In both cases, the magnitude of the noise increases with the feature dimension d .

Figures 2.1 and 2.2 also illustrate that the performance of the DP models remains comparable to that of the non-DP models when the dimension of features is smaller than or equal to 25, using the given fixed sample size. This suggests that it is feasible to safeguard data privacy using DP without significantly compromising the performance of the LLS regression models, particularly when the feature dimension is relatively small compared to the sample size. In real-world applications, it is beneficial to employ some dimension reduction or variable selection techniques to reduce the number of predictors before applying the differentially private LLS regression.

Prediction Error vs. Training Sample Size

In the following experiments, we investigate the correlation between prediction error and the size of training samples. For this purpose, we fix the feature dimension $d = 35$, and the privacy budget, ϵ , fixed at 0.5. The sample sizes investigated ranges from 5×10^3 to 6×10^4 , including various selected values for SEV and logistic regressions.

Figures 2.3 and 2.4 present the error of experiments with the increase in training sample size. It can be observed that the performance of non-DP models remains relatively stable, whereas DP models show improved performance that increasingly aligns with that of non-DP models as the dataset size grows. For example, in the results for the SEV regression, the median

error for the DP model is around 0.76 at $n = 1 \times 10^4$, while it is only 0.29 for the non-DP model. However, this discrepancy diminishes as n grows. At $n = 3 \times 10^4$, the median error for the DP model decreases to 0.35, while the non-DP model stands at 0.28, and it continues to decrease to 0.29 at $n = 6 \times 10^4$, with the non-DP model's error at 0.27. A similar pattern is observed in the results of the logistic regression. This is reasonable since a larger sample size enhances the representation of the underlying population, naturally reducing prediction error in regression models.

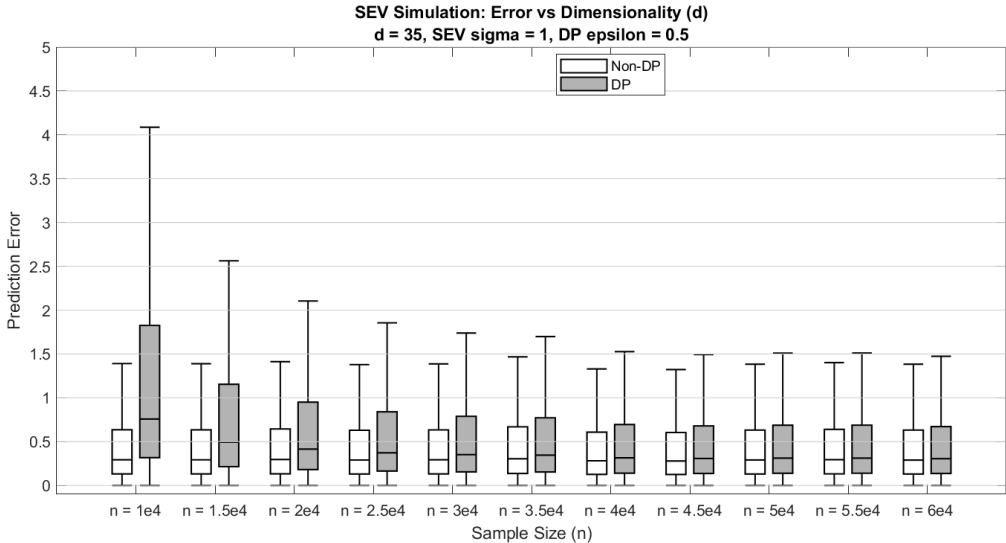


Figure 2.3: SEV Regression: Error vs. Sample Size n

In this set of experiments, we utilized a relatively large dimension number, 35, and based on the results, we can infer that a sample size of 4×10^4 and 2×10^4 is required for the proposed DP models to achieve results comparable to the non-DP models in terms of SEV and logistic regressions, respectively. Furthermore, we observed nearly identical error levels when the sample size reached 6×10^4 for SEV regression and 4×10^4 for logistic regression. In other words, the performance converged and did not continue to improve with further increases in sample size beyond it. It is expected that, with a smaller value of feature dimension d , the convergence requirements of n would become even more accommodating. Additionally, we observed that

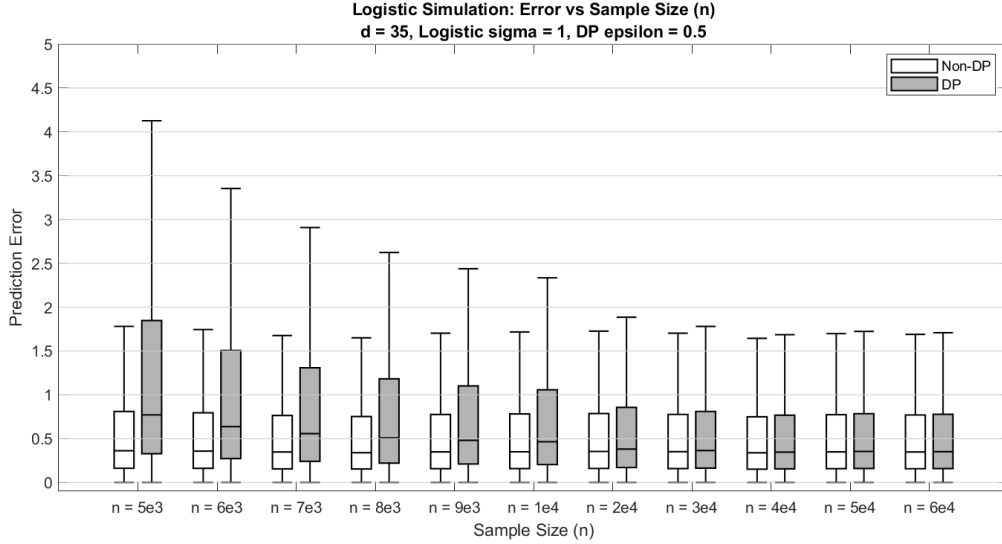


Figure 2.4: Logistic Regression: Error vs. Sample Size n

the non-DP models did not exhibit the same decreasing pattern of prediction error as the sample size increased within our experimental range. We attribute this to the fact that the smallest sample size in this study is 5×10^3 , which is already sufficiently large for the feature dimension of 35, indicating that the prediction performance has already converged.

Prediction Error vs. Privacy Budget

In the last set of experiments, our focus is on examining the impact of the privacy budget on the performance of the proposed LLS regression. This offers insights into the trade-offs between the level of privacy protection and prediction accuracy. Here, we conduct experiments across a range of ϵ values including 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, and 2, and fix the feature dimension at $d = 25$ for SEV regression, at $d = 38$ for logistic regression. The sample size is set as $n = 1 \times 10^4$ for both regressions.

Figures 2.5 and 2.6 display the performance of DP models with varying privacy budgets ϵ , with the rightmost boxplot representing the non-DP model. These figures illustrate that as the value of ϵ decreases (i.e., a stronger privacy guarantee is achieved), a higher prediction error is observed. Conversely, when the privacy budget becomes larger, the performance of the DP

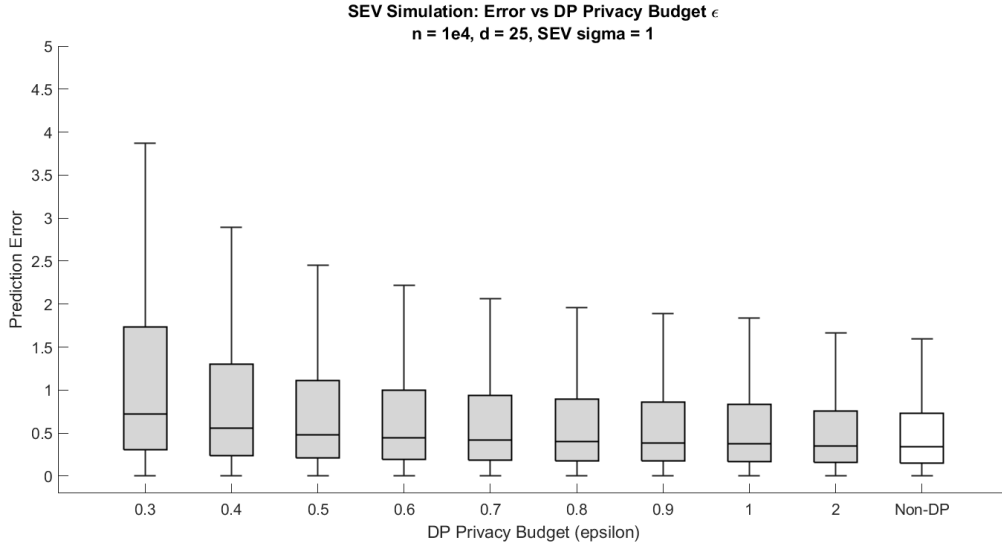


Figure 2.5: SEV Regression: Error vs. DP privacy budget ϵ

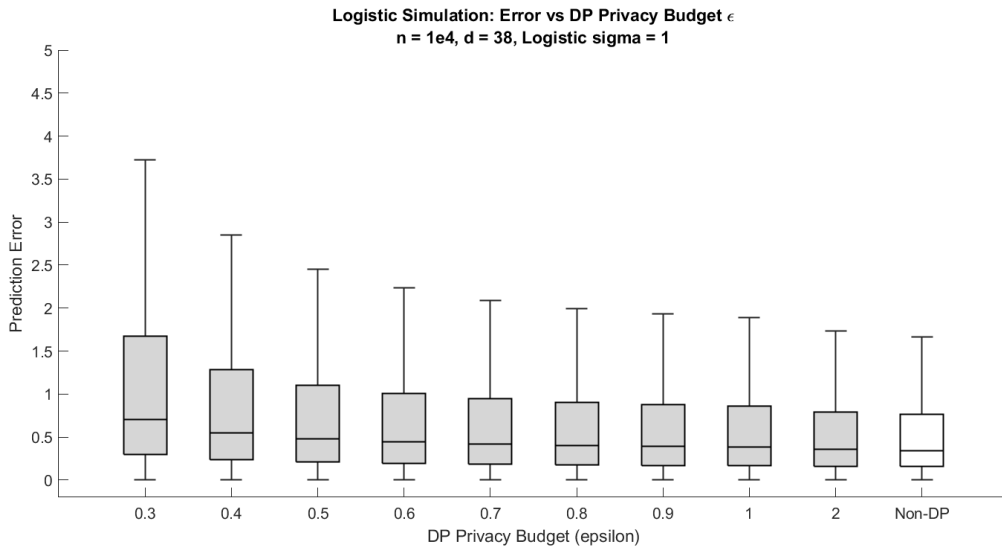


Figure 2.6: Logistic Regression: Error vs. DP privacy budget ϵ

models is improved. For instance, when $\epsilon = 0.3$, the median error for the differentially private SEV and Logistic regression models is approximately 0.72 and 0.70, respectively. When ϵ increases to 1, the median errors of the two models both decrease to 0.38. This is reasonable since the privacy budget affects the magnitude of the noise introduced to the LLS regression models. Recall that the noises added into the DP model are determined by $\text{Lap}((4 + 4\sqrt{d} + d)/\epsilon)$ for the SEV regression and $\text{Lap}((2 + 2\sqrt{d} + \frac{1}{2}d)/\epsilon)$ for Logistic regression. Therefore, when the value of

d is fixed, the smaller ϵ is, the larger the noises added, and consequently, the worse the model performs.

Figures 2.5 and 2.6 also demonstrate that the performance of the proposed differentially private LLS regression models converges with the increase in privacy budget. Both figures show that the prediction errors decrease rapidly when ϵ changes from smaller values to 1. However, when ϵ is larger than 1, the prediction errors do not exhibit a noticeable change and are almost identical to the non-DP models. We believe this is because the noise added to the regression model is sufficiently small (in comparison to the given sample size and feature dimension) when $\epsilon \geq 1$. As a result, the performance of the proposed models is not further compromised.

2.5 Case Study

In the case study, we utilize a dataset sourced from Saxena and Goebel (2008) to evaluate the performance of the proposed differentially private LLS regression.

2.5.1 Dataset and Experimental Settings

The dataset comprises multi-sensor degradation data collected from aircraft turbofan engines. It consists of degradation signals from 100 training engines that ran to failure, degradation signals from an additional 100 test engines with operations prematurely terminated at random time points before failure, and the actual times-to-failure (TTF) corresponding to the 100 test engines. Each engine was monitored by 21 sensors. As an illustration, the degradation signals from one of the engines in the training dataset are shown in Figure 2.7. Fang et al. (2017) have demonstrated that sensors 4, 17, and 20, along with SEV regression, are the most effective combination in modeling this dataset. Therefore, this study uses these three sensors and evaluates the performance of the proposed differentially private LLS regression under SEV distribution. It is known that the degradation signals in the training and test datasets have varying lengths due to failure truncation. Specifically, as machines are stopped for maintenance

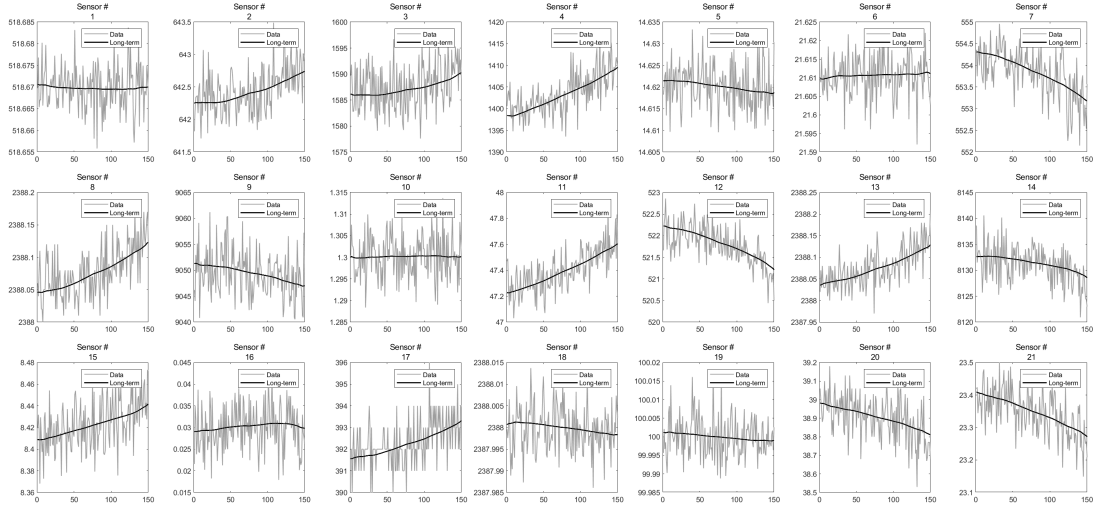


Figure 2.7: Illustrated degradation signals from one of the engines.

or replacement upon failure, the degradation signals can only be observed up to the time point of failure. Additionally, since different machines have different failure times, the length of degradation signals varies from one machine to another. To address this varying-length challenge, Fang et al. (2017) suggested the use of a time-varying framework, which truncates all degradation signals to the same length. In this study, we truncate the degradation signals from all engines by retaining their first 150 timestamps. Any engines with TTF smaller than 150 are excluded. Consequently, there are 94 engines left in the training dataset and 37 engines left in the test dataset.

Since the degradation signals from the three sensors have considerable correlations, we first apply Principal Component Analysis (PCA) to fuse them following the suggestion of Fang et al. (2017). Next, we regress the TTFs against the features extracted from PCA using LLS regression. Similar to the simulation study, we use a non-DP SEV regression as the benchmark. We will investigate the performance of the proposed differentially private SEV regression under different predictor dimensions (i.e., the number of principal components) and also different privacy budgets. Since the DP noise was drawn from the Laplace distribution randomly each time, in order to evaluate the robustness of the DP models against the randomness of

noise addition, the whole evaluation process was repeated 500 times for DP models, and the aggregated errors from all the repetitions are reported.

2.5.2 Results and Analysis

Prediction Error vs. Dimensionality

The first experiment focuses on investigating the performance of both DP and non-DP models with varying predictor dimensions. Specifically, we fix the privacy budget ϵ at 5, and try the first 3, 4, 5, and 6 principal components. Figure 2.8 shows the performance of both the non-DP model and the DP models.

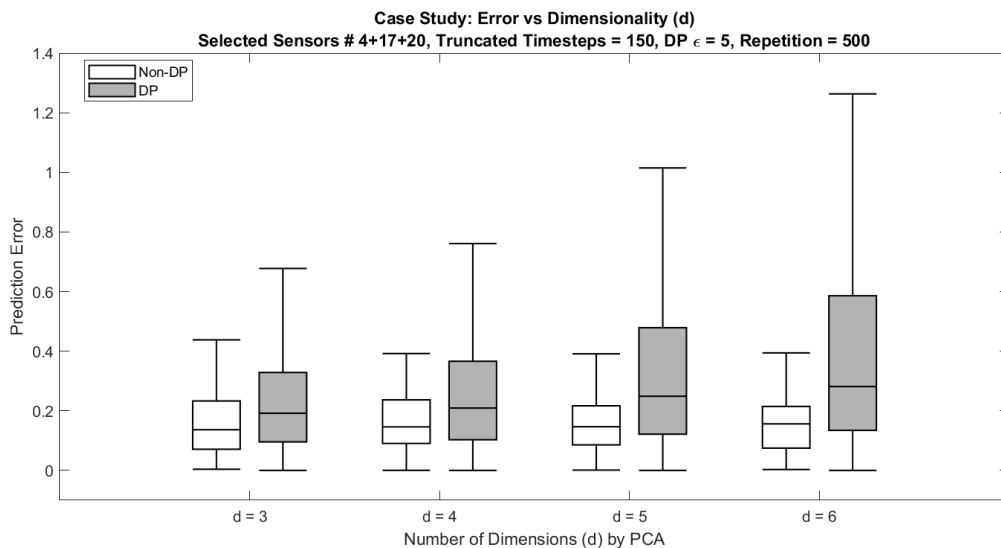


Figure 2.8: Error vs. Dimensionality d

Figure 2.8 illustrates that the prediction errors of the DP model increase with the expansion of predictor dimensions, while the performance of the non-DP model remains stable across the various predictor dimensions tested. For example, the median error (and Interquartile Range, IQR) of the DP model at $d = 3$ is around 0.19(0.24). However, it reaches 0.21(0.26) at $d = 4$ and 0.28(0.45) at $d = 6$. In contrast, the prediction errors (and IQR) of the non-DP model

when $d = 3, 4, 5$ and 6 do not have a significant change and remain around $0.15(0.15)$.

We attribute the performance degradation of the DP model with the increase in predictor dimension to the heightened magnitude of noise injected into the model. As the number of features increases, the error correspondingly grows, which compromises the model's performance. Recall that the noise added to the differentially private SEV model is from $\text{Lap}((4 + 4\sqrt{d} + d)/\epsilon)$, which is a function of d . Thus, the noise added to the model increases with the increase of d . Another factor contributing to the inferior performance of the DP model compared to the non-DP model is the limited sample size in the training dataset. The simulation studies in Section 2.4 have highlighted the significance of having sufficiently large training samples for DP models. In this study, the training sample size is 97 , which is relatively small.

Prediction Error vs. Privacy Budget

In this subsection, we explore the influence of the privacy budget ϵ on the performance of the DP model. Here, we set the dimension number at $d = 3$ and test a broad range of ϵ values, ranging from 0.3 to 1 that increment by 0.1 , and $1, 1.5, 2, 3, 4, 5$, and 10 . Our objective is to observe any trends in the prediction errors of the DP model and examine whether the errors of the DP model converge with those of the non-DP model as ϵ increases.

Figure 2.9 depicts the results of experiments on the DP budget ϵ , where the rightmost boxplot represents the error results associated with the non-DP model. It is evident that the errors of DP models consistently and clearly converge towards the non-DP errors as the value of ϵ increases. The median error values of the DP model are 0.63 at $\epsilon = 0.5$, decreasing to 0.45 at $\epsilon = 0.8$, and further to 0.38 at $\epsilon = 1$. It keeps decreasing to equal or less than 0.2 when $\epsilon \geq 5$, which is pretty close to the non-DP model. This observation suggests that the DP model with a larger ϵ value is more closely aligned with the performance of the non-DP model, while larger ϵ implies weaker privacy protection. This is reasonable since larger ϵ means less noise introduced to the differentially private LLS regression model.

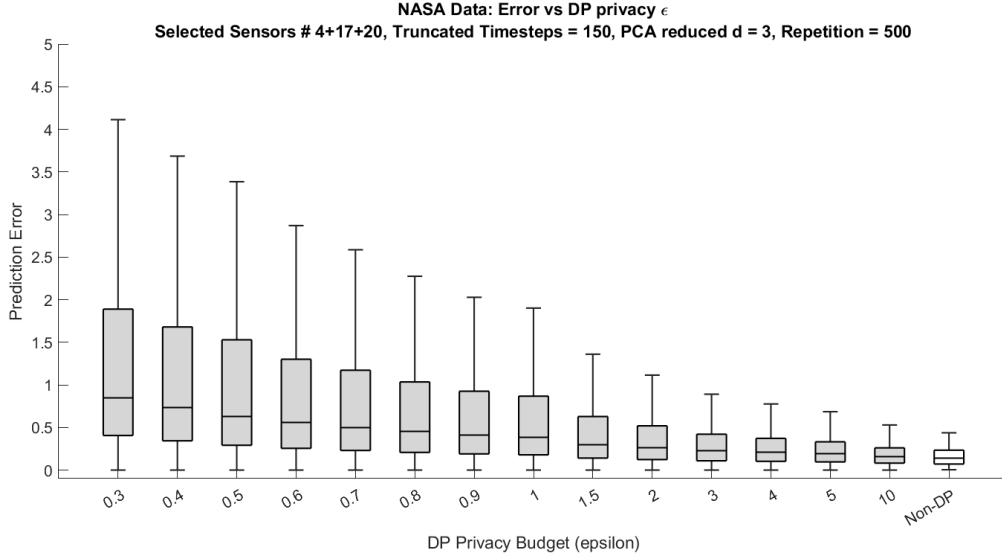


Figure 2.9: Error vs. DP privacy budget ϵ

2.6 Conclusions

This chapter proposed differentially private (log)-location-scale regression. We incorporated differential privacy into two specific LLS regression models, SEV and Logistics, which can also be applied to Weibull regression and Log-logistic regression. The integration of differential privacy into the aforementioned models is based on the function mechanism, which works as follows. First, we employ Taylor expansion to decompose the log-likelihood function of LLS regression as a weighted combination of polynomial functions, truncating it at the second order. Next, random noise from the Laplace distribution is injected into the weights. Third, a perturbed log-likelihood function is reconstructed using the noisy weights and the polynomial functions. Finally, the perturbed log-likelihood function is solved for parameter estimation. We derived the sensitivities of the proposed DP models, which determine the magnitude of noise injected into the weights. Also, we proved that the proposed DP-LLS regression models satisfy ϵ -differential privacy.

Extensive simulations and case studies were carried out to assess the performance of the proposed DP-LLS regression models. We focused on understanding how the size of the training dataset, sensor correlation, and privacy budget influence the DP method's effectiveness. The

findings highlighted the significant impact of all three factors on the performance of the DP-LLS. Specifically, the results indicate that performance degrades with an increase in predictor dimension when keeping the sample size and privacy budget fixed. Performance improves with an increase in training sample size and stabilizes if the sample size is large enough while keeping the other two factors fixed. Furthermore, performance worsens with a decrease in the privacy budget (indicating stronger privacy protection) when keeping the predictor dimension and sample size fixed. In general, a sufficiently large training dataset is needed to ensure satisfactory prediction performance of the DP-LLS regression models and a desired level of privacy protection.

CHAPTER

3

A DIFFERENTIALLY PRIVATE INFORMATIVE SENSOR SELECTION METHOD FOR INDUSTRIAL PROGNOSTICS

3.1 Introduction

Sensor selection is crucial in multi-sensor systems for industrial prognostics, directly impacting maintenance strategies and operational efficiency. By identifying the most informative sensors, predictive models can focus on data that significantly enhance prognostics accuracy, optimizing resource allocation and minimizing unnecessary maintenance efforts. Group lasso (Bakin et al. 1999; Cai 2001; Antoniadis and Fan 2001; Yuan and Lin 2006; Meier et al. 2008) is a commonly

used approach for group variable selection, leveraging its strength in promoting group sparsity and handling high-dimensional data efficiently. In the applications of multi-sensor system prognostics, studies have shown that the group lasso enables the identification of relevant sensor groups, enhancing model interpretability and prognostic performance by focusing on data that contribute to accurate failure predictions (Fang et al. 2017).

Meanwhile, the Log-Location-Scale (LLS) regression models represent an advancement over traditional linear regression by addressing scenarios where the dependent variable demonstrates asymmetry or non-constant variance. Notable distributions under the LLS umbrella include Normal, Log-Normal, Logistics, Log-Logistics, Smallest Extreme Value (SEV), and Weibull. LLS regression has found extensive application in reliability engineering (Meeker and Escobar 2014; Doray 1994; Hong et al. 2010, 2015) and industrial prognostics (Fang et al. 2017, 2019a; Zhou and Fang 2023; Fang et al. 2019b) on account of the adaptability provided by these distributions.

Integrating group lasso with LLS regression models offers a robust strategy for sensor selection and TTF prediction in industrial multi-sensor system prognostics. This synergy optimizes sensor selection by combining the sparsity and relevance of group lasso with the predictive accuracy of Log-Location-Scale regression, streamlining predictive maintenance processes. Such integration has been shown to significantly improve prognostic outcomes, enhancing maintenance decision-making and operational reliability (Fang et al. 2017).

A widely used method for solving group lasso problems is Proximal Gradient Descent (PGD), valued for its ability to induce sparsity through regularization. PGD works by updating model parameters in steps, using the loss function's gradient and a proximal operation to promote sparsity among groups of variables. However, PGD raises privacy concerns because these gradient updates can inadvertently reveal sensitive data details. This risk, known as gradient leakage, allows attackers to potentially extract private information from the training data by analyzing the gradient updates, typically through inference attacks and reconstruction attacks (Zhu et al. 2019; Geiping et al. 2020; Huang et al. 2021). To mitigate this risk and enhance data

confidentiality, this article proposes a privacy-preserving PGD approach based on differential privacy, and demonstrates its applications on various group lasso regularized LLS regression models.

Differential Privacy (DP) is a concept used to protect individual data points privacy when a data record is included in a database being analyzed (Dwork 2006). The idea is to add a bit of randomness, or noise, to the data or function output results so that it's hard to identify any single data point's information. This way, researchers can still gain insights from the data as a whole without risking individuals' privacy. The mechanism to achieve this involves carefully calculating how much noise to add based on the type of data and the queries being run, ensuring that the privacy of individuals is preserved while the overall data utility is maintained.

The sensitivity of a function and the privacy budget epsilon (ϵ) are two key concepts in DP. Sensitivity refers to how much the output of a function could change if one data point was added or removed from the dataset. This helps determine the amount of noise that needs to be added to protect privacy. On the other hand, epsilon (ϵ) is a measure of privacy loss in a dataset. A smaller epsilon means better privacy because it indicates that the presence or absence of an individual's data has a minimal impact on the outcome of a function (Dwork et al. 2006). Therefore, deciding on the value of epsilon is a critical step in implementing DP, as it balances the trade-off between data privacy and the accuracy of the function results.

As our major privacy concern lies with gradient leakage when using proximal gradient descent, we can use a differential privacy mechanism to add noise to protect the gradient. By introducing carefully calibrated noise to the gradient at each iteration, DP ensures that the optimization process remains privacy-preserving, preventing leakage of sensitive information. This addition of noise acts as a safeguard, making it significantly more challenging to infer individual data points from the model's parameters or its updates, thus striking a balance between model performance and privacy protection.

LLS regression utilizes some commonly used distributions like normal, log-normal, logistic, log-logistic, smallest extreme value (SEV), and Weibull. Models based on normal, logistic, or

SEV distributions are called location-scale regressions, whereas those using log-normal, log-logistic, or Weibull distributions are known as log-location-scale regressions. These two model types can transform into each other; for instance, converting log-normal to normal regression involves logging the response variable, and a similar approach applies to transforming Weibull or log-logistic into SEV or logistic regression. This study focuses on applying the DP-PGD approach for linear, logistic and SEV regressions with group lasso regularization, which also applies to log-normal, log-logistic and Weibull regressions. It's crucial to differentiate the logistic regression discussed here, which deals with a logistic-distributed response variable, from the logistic regression used in classification tasks (Wright 1995). The latter predicts binary or binomial outcomes, unlike the former, which is part of the location-scale family.

The structure of this chapter is laid out as follows: Section 3.2 presents the group lasso regularized log-location-scale regression models we will be using for sensor selection. Section 3.3 introduces the optimization method we will be using to solve the models. In section 3.4, we detail the differentially private approach we propose. Then, in Sections 3.5 and 3.6, we evaluate the effectiveness of this differentially private algorithm using both simulated data and real-world data from aircraft engines, sourced from the NASA data repository. The chapter concludes with Section 3.7, where we summarize our findings and reflections on the study.

3.2 Regularized LLS Regression

In this section, we will define our models for sensor selection. Specifically, we use group lasso regularized (log)-location-scale regression models. We will also introduce and explain the idea and basic concept behind this modeling.

3.2.1 Model Define

Given a dataset with n observations, s sensors, and each sensor has $\{d_k\}_{k=1}^s$ features. Without loss of generality, we assume each sensor has the same number of features d , which gives us a

total of sd features. We represent the data as $\{(\tilde{y}_i, \{x_{ij}\}_{j=1}^{sd})\}_{i=1}^n$, where \tilde{y}_i denotes the dependent variable, akin to the time-to-failure (TTF) in the context of reliability analysis and prognostics; $\{x_{ij}\}_{j=1}^{sd}$ symbolize the (sd)-dimensional predictors.

It is posited that each \tilde{y}_i originates from a distribution within the (log)-location-scale family. Specifically, y_i represents a random variable with its cumulative distribution function (CDF) given by

$$\Pr(y_i \leq t) = \Omega\left(\frac{t - \mu_i}{\sigma}\right), \quad (3.1)$$

where $y_i = \tilde{y}_i$ for distributions in the location-scale family, and $y_i = \log(\tilde{y}_i)$ when derived from the log-location-scale family. Here, $\Omega(\cdot)$ denotes the CDF of a standard form within this family. The parameters μ_i and σ represent the location and scale parameters, respectively, with μ_i being determined by the predictors as $\mu_i = \beta_0 + \sum_{j=1}^{sd} x_{ij}\beta_j$, or for convenience $\mu_i = \sum_{j=0}^{sd} x_{ij}\beta_j$ setting $x_{i0} = 1$. Thus, the CDF of y_i is reformulated as

$$\Pr(y_i \leq t) = \Omega\left(\frac{t - \sum_{j=0}^{sd} x_{ij}\beta_j}{\sigma}\right), \quad (3.2)$$

where $\{\beta_j\}_{j=0}^{sd}$ are the regression coefficients to be estimated. The estimation of these coefficients, alongside the scale parameter σ , typically employs the maximum likelihood estimation (MLE) method. The likelihood function is expressed as

$$L(\{\beta_j\}_{j=0}^{sd}, \sigma) = \prod_{i=1}^n \frac{1}{\sigma} \omega\left(\frac{y_i - \sum_{j=0}^{sd} x_{ij}\beta_j}{\sigma}\right), \quad (3.3)$$

where $\omega(\cdot)$ is the probability density function (PDF) corresponding to a standard distribution within the location-scale family, e.g., $\omega(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^2}{2}\right)$ for the normal distribution, $\omega(z) = \frac{\exp(z)}{(1+\exp(z))^2}$ for the logistic distribution, and $\omega(z) = \exp(z - \exp(z))$ for the SEV distribution. The optimal set of parameters $(\{\beta_j\}_{j=0}^{sd}, \sigma)$ is obtained by maximizing the log-likelihood function, denoted as $\ell = \log(L(\{\beta_j\}_{j=0}^{sd}, \sigma))$.

LLS regression is particularly well-suited for modeling industrial prognostic data, as it can

effectively capture the often skewed and non-normal distributions observed in real-world sensor measurements. By leveraging the flexibility of the LLS model, engineers can better represent the underlying degradation processes and improve the accuracy of remaining useful life predictions. This capability is crucial when selecting the most informative sensors for a multi-sensor prognostic system, as it ensures that the chosen sensors provide reliable and meaningful data for informed decision-making. Our next question is, how can we utilize LLS regression for sensor selection?

Variable selection is a critical process in the development of statistical models, especially in scenarios involving high-dimensional data. It not only aids in simplifying models to make them more interpretable but also in enhancing prediction accuracy by eliminating irrelevant or redundant predictors. Traditional methods like Lasso (Least Absolute Shrinkage and Selection Operator) have been widely used for variable selection by imposing an L1 penalty to encourage sparsity in the model coefficients (Tibshirani 1996).

Group Lasso extends this concept to accommodate the selection of groups of variables, rather than individual variables. This approach is particularly useful in situations where predictors are naturally grouped (e.g., genetic data where genes belong to different pathways), and the goal is to select relevant groups instead of individual predictors. Group Lasso was first introduced by Yuan and Lin (2006), who proposed a penalization method that considers the group structure of predictors by adding a penalty term proportional to the L2 norm of the coefficients within each group. This not only facilitates group-wise sparsity but also maintains the integrity of the group structure in the model.

Suppose we have s groups, each group has d variables. The group lasso regularization for regression models can be written as $\lambda \sum_{k=1}^s \|\boldsymbol{\beta}_k\|_2$, where $d \times 1$ vector $\boldsymbol{\beta}_k$ represents the vector of coefficients of k -th group, $\|\cdot\|_2$ denotes the L2 norm, and λ is the regularization parameter controlling the degree of sparsity.

The loss function for the common linear regression model with group lasso regularization for fitting normal distributed data has been well-studied and widely applied (Yuan and Lin

2006; Simon et al. 2013). We can write it as

$$\hat{l}(\boldsymbol{\beta}) = \frac{1}{2n} \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \sum_{k=1}^s \|\boldsymbol{\beta}_k\|_2 \quad (3.4)$$

where $\mathbf{x}_i^\top \boldsymbol{\beta} = \sum_{j=0}^{sd} \beta_j x_{ij}$, and $\boldsymbol{\beta}_k = (\{\beta_j\}_{j=(k-1)d+1}^{kd})$.

We would also like to define the model for other common LLS distributions, such as the SEV and logistic distributions, they can also be extended to Weibull and log-logistic distributions. To incorporate the group lasso regularization with other LLS regression models, recall that the objective function for the LLS regression can be set up as $\ell = \log(L(\{\beta_j\}_{j=0}^{sd}, \sigma))$, where the MLE function $L(\{\beta_j\}_{j=0}^{sd}, \sigma)$ is given in equation 3.2. To better define our objective functions, we take the negative of the log-MLE functions with group lasso regularization to be our loss functions, and construct minimizing problems of them. Therefore, the general form of our objective functions is:

$$\min_{\boldsymbol{\beta}, \sigma} \hat{\ell}(\boldsymbol{\beta}, \sigma) = -\log(L(\boldsymbol{\beta}, \sigma)) + \lambda \sum_{k=1}^s \|\boldsymbol{\beta}_k\|_2 \quad (3.5)$$

where $\boldsymbol{\beta} = (\{\beta_j\}_{j=0}^{sd})$, and $\boldsymbol{\beta}_k = (\{\beta_j\}_{j=(k-1)d+1}^{kd})$.

Based on this general form, we can write the objective function of SEV regression model with the group lasso penalty as

$$\hat{l}(\boldsymbol{\beta}, \sigma) = n \log \sigma - \sum_{i=1}^n \left(\frac{y_i - \mathbf{x}_i^\top \boldsymbol{\beta}}{\sigma} \right) + \sum_{i=1}^n \exp \left(\frac{y_i - \mathbf{x}_i^\top \boldsymbol{\beta}}{\sigma} \right) + \lambda \sum_{k=1}^s \|\boldsymbol{\beta}_k\|_2 \quad (3.6)$$

Similarly, the objective function for the logistic regression model for logistic distribution with group lasso penalty can be written as follows:

$$\hat{l}(\boldsymbol{\beta}, \sigma) = n \log \sigma - \sum_{i=1}^n \left(\frac{y_i - \mathbf{x}_i^\top \boldsymbol{\beta}}{\sigma} \right) + 2 \sum_{i=1}^n \log \left(1 + \exp \left(\frac{y_i - \mathbf{x}_i^\top \boldsymbol{\beta}}{\sigma} \right) \right) + \lambda \sum_{k=1}^s \|\boldsymbol{\beta}_k\|_2 \quad (3.7)$$

However, the above functions are not convex. To make them convex functions, we apply the

following transformation: $q = 1/\sigma$, and $p_j = \beta_j q$. As result, we define a new objective function for SEV model, which is a convex function, as follows:

$$\hat{l}(\mathbf{p}, q) = -n \log q - \sum_{i=1}^n (y_i q - \mathbf{x}_i^\top \mathbf{p}) + \sum_{i=1}^n \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}) + \lambda \sum_{k=1}^s \|\mathbf{p}_k\|_2 \quad (3.8)$$

And the convex objective function for the logistic model is as follows:

$$\hat{l}(\mathbf{p}, q) = -n \log q - \sum_{i=1}^n (y_i q - \mathbf{x}_i^\top \mathbf{p}) + 2 \sum_{i=1}^n \log(1 + \exp(y_i q - \mathbf{x}_i^\top \mathbf{p})) + \lambda \sum_{k=1}^s \|\mathbf{p}_k\|_2 \quad (3.9)$$

where $\mathbf{p} = (\{p_j\}_{j=0}^{sd})$, and $\mathbf{p}_k = (\{p_j\}_{j=(k-1)d+1}^{kd})$.

Sensor selection is essentially group variable selection if we treat all the features belonging to the same sensor as grouped variables. The core idea of group variable selection is that all the variables in the same group are either all selected or none. In our context, the features from the same sensor either all get selected or none are selected, i.e., the sensor gets selected or not. By utilizing LLS regression with group lasso regularization, we can better fit our data on LLS distributions, and promote sparsity between informative and non-informative sensors.

3.2.2 Sensor Selection Procedures

Applying group lasso for sensor selection involves treating the data from each sensor as a group of variables, which allows for selection at the group level. This means the model will either entirely include or exclude each sensor's data based on group relevance. The key points in implementing group lasso for sensor selection are as follows:

- **Data Preparation:** Organize each sensor's data into a group of variables. This initial step ensures that the subsequent modeling can efficiently process the data according to the group lasso methodology.
- **Modeling and fitting:** Fit a group lasso model by solving an optimization problem. To better deal with (log)-location-scale distributed data, we would like to set up the loss

function as an LLS regression model with a group lasso penalty term, which is the sum of the L2 norms of the coefficient vectors for each group. This penalty term is adjusted by a tuning parameter, λ , which influences the level of sparsity at the group level, encouraging entire groups of coefficients to be driven to zero simultaneously.

- **Tuning Parameter λ :** When fitting the models, the choice of λ is crucial as it balances the model's fit against its complexity. A larger λ leads to a sparser model with more groups (sensors) being excluded, while a smaller λ tends to keep more groups in the model.
- **Sensor Selection:** Once the group lasso regularized LLS regression model is fitted, sensors are selected based on the non-zero-ness of their corresponding groups. In this context, a group is considered non-zero if its coefficient vector's L2 norm is non-zero, indicating that the sensor provides valuable information and should be included in the model.

Let us address all these key points one by one. To start, the data preparation. In a multi-sensor system, we have s sensors, each of which may have a different number of features $\{d_k\}_{k=1}^s$. In practice, the feature numbers could be large, and it is essential for us to adopt some dimension reduction techniques, such as PCA, to reduce the feature number of each sensor. Assume each sensor has the same number of features d after PCA, which gives us sd total features for the whole data. Suppose we have n observations, after concatenate data from all sensors, we obtain a $n \times sd$ matrix as our \mathbf{X} , and a $n \times 1$ vector as our \mathbf{Y} , which makes a $n \times (sd + 1)$ dataset. We address each group of d features as features from one sensor.

Now let us discuss about modeling and fitting. We have defined our models in the previous part for linear, SEV, and logistic regressions. However, we need to be cautious about how to fit these models. The group lasso problem, due to its nature of incorporating both a differentiable part and a non-differentiable regularization for group-wise variable selection, requires sophisticated optimization techniques for effective solutions. A prominent approach for solving the Group Lasso problem is proximal gradient descent, which we will present in details in the next chapter.

Another issue we need to take into account for model fitting is the choice of the regularization parameter λ . It controls the strength of the penalty applied to the coefficients of the model and balances the trade-off between fitting the model to the data and enforcing sparsity in the solution. Choosing a λ that's too small can lead to a model that includes too many irrelevant variables (false positives), as the penalty for including additional variables in the model is not strong enough. A small λ value may also cause the model to fit the noise in the training data too closely, leading to poor generalization to new data. Conversely, a λ that's too large might exclude relevant variables from the model (false negatives) by imposing too strong a penalty on the coefficients. An excessively large λ can also lead to underfitting, where the model does not capture the underlying structure of the data, resulting in poor predictive performance.

To find an optimal λ value, generalized cross-validation (GCV) offers a practical solution. GCV systematically evaluates the model's performance across a range of λ values using cross-validation techniques. By assessing the model's predictive error on parts of the data not used for training, GCV helps identify a λ that achieves a favorable compromise between model complexity and its ability to generalize (Golub et al. 1979; Friedman et al. 2010). This approach ensures that the selected λ minimizes the risk of both overfitting and underfitting, leading to a model that is both sparse and performs well on new data.

To interpret the final sensor selection result, we obtain a vector of regression parameters, which also has the length of sd , i.e., the number of sensors (s) multiplied by the number of features (d) in each sensor. It can be easily reshaped into a $d \times s$ matrix, where each column represents coefficients for a sensor, $\{\mathbf{p}_k\}_{k=1}^s$. We say the k -th sensor is selected if $\mathbf{p}_k \neq 0$, and it is not selected if $\mathbf{p}_k = 0$. In practice, the condition check is often replaced by comparing the norm of the group parameters to a small threshold to deal with potential numerical issues in computation. Considering we applied transformation $p = \beta q$ on the original β when defining our objective functions, when interpreting the sensor selection results, we could use conditions $\|\frac{\mathbf{p}_k}{q}\|_2 \geq \delta$ means the sensor is selected and $\|\frac{\mathbf{p}_k}{q}\|_2 < \delta$ means the sensor will not be used.

After obtaining informative sensors, it is common practice to concatenate data from these

sensors and perform another PCA before proceeding to prediction. We will refrain from going into details about the prediction work after sensor selection, as it is not the focus of this paper's discussion.

3.3 The Optimization Algorithm for Parameter Estimation

Proximal Gradient Descent (PGD) represents a pivotal optimization technique in the context of sparse modeling, particularly for addressing the Group Lasso problem. The primary challenge in optimizing the Group Lasso objective lies in its non-differentiable penalty term, which prohibits the use of traditional gradient descent methods. Proximal gradient descent overcomes this by decomposing the optimization process into manageable steps, specifically a gradient descent step for the differentiable part of the loss function and a proximal mapping step for the non-differentiable penalty (Boyd et al. 2011; Simon et al. 2013; Parikh et al. 2014).

The gradient descent step involves updating the coefficient vector \mathbf{p} by moving in the opposite direction of the gradient of the differentiable part of the loss function $\hat{l}(\mathbf{p}, q)$. Suppose the differentiable part is called as function $g(\mathbf{p}, q)$, the coefficients update at iteration t by utilizing an intermediate coefficient vector $\mathbf{v}^t = \mathbf{p}^{(t)} - \alpha_t \nabla g(\mathbf{p}^{(t)})$, where α_t is the step size at iteration t . This \mathbf{v}^t will further proceed in the proximal step to update $\mathbf{p}^{(t+1)}$.

The proximal step addresses the non-differentiable penalty term of the group lasso through a proximal mapping. Given a vector \mathbf{v} and a non-negative regularization parameter λ , the proximal operator for each group in the group lasso method at \mathbf{v} is defined as $\text{prox}_{\lambda\|\cdot\|_2}(\mathbf{v}) = \arg \min_{\mathbf{p}} \left\{ \frac{1}{2} \|\mathbf{p} - \mathbf{v}\|_2^2 + \lambda \|\mathbf{p}\|_2 \right\}$. The minimization problem has a known closed-form solution, often derived using Karush–Kuhn–Tucker (KKT) conditions (Yuan and Lin 2006), leading to $\text{prox}_{\lambda\|\cdot\|_2}(\mathbf{v}) = \max\left(0, 1 - \frac{\lambda}{\|\mathbf{v}\|_2}\right) \mathbf{v}$. Therefore, in the proximal step, we update coefficients by $\mathbf{p}_k^{(t+1)} = \max\left(0, 1 - \frac{\lambda t}{\|\mathbf{v}_k^{(t)}\|_2}\right) \mathbf{v}_k^{(t)}$ for each group k . This step effectively shrinks the estimates towards zero, with group-wise adjustments based on the group sizes and the regularization parameter λ .

Proximal gradient descent combines gradient steps for differentiable components with proximal steps for non-differentiable parts within an objective function. For LLS regression models incorporating group lasso regularization, the approach involves distinct gradient steps for the model’s differential aspects, which vary across different distribution-based models, and a uniform proximal step for the non-differential part, namely the group lasso penalty. This means that while each LLS regression model requires a tailored gradient step reflecting its specific distribution, the proximal mapping applied to handle the group lasso penalty remains consistent across models.

Specifically, for our objective functions of LLS regression, we can see them as having two parts, e.g., $\hat{l}(\mathbf{p}, q) = g(\mathbf{p}, q) + h(\mathbf{p})$, where $g(\mathbf{p}, q)$ is a differentiable function, and $h(\mathbf{p})$ is a non-differentiable function. The differentiable part, $g(\mathbf{p}, q)$, will be taken care of by the gradient descent step, while the $h(\mathbf{p})$ part will be addressed later in the proximal mapping step. Since we will be using the same regularization term for all kinds of LLS regressions, the $h(\mathbf{p})$ function will be the same for all our models, i.e., the closed-form proximal mapping for group lasso. The remaining work we need to address is to derive the different gradients from the $g(\mathbf{p}, q)$ functions considering different LLS regressions.

Let us first take linear regression as an example. Since we don’t have to apply the transformation for linear regression’s objective function, we still have parameter $\boldsymbol{\beta}$ for this model. Particularly, the gradient of linear regression, $\nabla g = (\{\frac{\partial g}{\partial \beta_j}\}_{j=0}^{sd})$ can be computed by

$$\frac{\partial g}{\partial \beta_j} = \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})(-x_{ij}) \quad (3.10)$$

where $\mathbf{x}_i^\top \boldsymbol{\beta} = \sum_{j=0}^{sd} \beta_j x_{ij}$. And the algorithm for solving linear regression with group lasso penalty through proximal gradient descent can be given below as shown in algorithm 3.

In this algorithm, we first compute gradient according to equation 3.10 in line 3, and then in line 4, we perform the gradient step with an intermediate gradient \mathbf{v} . Notice that for line 5 in the algorithm, the proximal mapping step, we only consider parameters for each sensor, i.e.

$\{\boldsymbol{\beta}_k^{(t+1)}\}_{k=1}^s$, where $\boldsymbol{\beta}_k = (\{\beta_j\}_{j=(k-1)d+1}^{kd})$, which means β_0 does not proceed through the proximal mapping, thus we update β_0 in step 4 directly, and let $\mathbf{v} = (\{v_j\}_{j=1}^{sd})$, which updates from $\{\beta_j\}_{j=1}^{sd}$. We terminate the iterations when the gradient converges, and interpret the sensor selection results by looking at each $\boldsymbol{\beta}_k$ in the returned parameter $\boldsymbol{\beta}$.

Algorithm 3 Proximal Gradient Descent for Linear Regression with Group Lasso

Require: Database $\{y_i, \{x_{ij}\}_{j=1}^{sd}\}_{i=1}^n$, regularization parameter λ , learning rate α

Ensure: parameters: $\boldsymbol{\beta}$

1: Initialize parameters: set $\boldsymbol{\beta}^{(0)} = \mathbf{0}$

2: **for** $t = 0, 1, 2, \dots$, **do**

3: Compute gradient: refers to Equation 3.10 for computing $\nabla g(\boldsymbol{\beta}^{(t)})$

4: Perform gradient step:

$$(\boldsymbol{\beta}_0^{(t+1)}, \mathbf{v}^{(t)}) = \boldsymbol{\beta}^{(t)} - \alpha \nabla g(\boldsymbol{\beta}^{(t)})$$

where $\mathbf{v} = (\{v_j\}_{j=1}^{sd})$ updates from $(\{\beta_j\}_{j=1}^{sd})$

5: Perform proximal mapping for each sensor:

$$\{\boldsymbol{\beta}_k^{(t+1)}\}_{k=1}^s = \max\left(0, 1 - \frac{\lambda_t}{\|\mathbf{v}_k^{(t)}\|_2}\right) \mathbf{v}_k^{(t)}$$

where $\boldsymbol{\beta}_k = (\{\beta_j\}_{j=(k-1)d+1}^{kd})$, and $\mathbf{v}_k = (\{v_j\}_{j=(k-1)d+1}^{kd})$.

6: **end for**

7: Stop when $\|\boldsymbol{\beta}^{(t+1)} - \boldsymbol{\beta}^{(t)}\|_\infty < 10^{-5}$

8: **return** $\boldsymbol{\beta}^{(t+1)}$

Now, let us demonstrate the algorithm for general LLS regression models, take SEV and logistic regressions as examples. After the transformation, we have parameters (\mathbf{p}, q) . As noted in line 3 of algorithm 4, the gradient should be computed based on which type of LLS regression we are using.

The gradient $\nabla g(\mathbf{p}, q)_{\text{SEV}}$ for SEV regression is a vector of $sd + 2$ length, constructed by two parts, $\{\frac{\partial g}{\partial p_j}\}_{j=0}^{sd}$ and $\frac{\partial g}{\partial q}$. We can write the gradient as $\nabla g(\mathbf{p}, q) = \left(\frac{\partial g}{\partial p_0}, \frac{\partial g}{\partial p_1}, \dots, \frac{\partial g}{\partial p_{sd}}, \frac{\partial g}{\partial q}\right)^\top$. For each p_j , we have

$$\frac{\partial g}{\partial p_j} = \sum_{i=1}^n x_{ij} + \sum_{i=1}^n \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}) \cdot (-x_{ij}) \quad (3.11)$$

Algorithm 4 Proximal Gradient Descent for LLS Regression with Group Lasso

Require: Database $\{y_i, \{x_{ij}\}_{j=1}^{sd}\}_{i=1}^n$, regularization parameter λ , learning rate α

Ensure: parameters: parameters (\mathbf{p}, q)

1: Initialize parameters: set $\mathbf{p}^{(0)} = 0, q^{(0)} = 1$

2: **for** $t = 0, 1, 2, \dots$, **do**

3: Compute gradient depends on the specific LLS model

- For SEV model refers to Equation 3.11 and 3.12 for computing $\nabla g(\mathbf{p}^{(t)}, q^{(t)})$
- For logistic model refers to Equation 3.13 and 3.14 for computing $\nabla g(\mathbf{p}^{(t)}, q^{(t)})$

4: Perform gradient step:

$$(\mathbf{p}_0^{(t+1)}, \mathbf{v}^{(t)}, q^{(t+1)}) = (\mathbf{p}^{(t)}, q^{(t)}) - \alpha \nabla g(\mathbf{p}^{(t)}, q^{(t)}),$$

where $\mathbf{v} = (\{v_j\}_{j=1}^{sd})$ updates from $(\{p_j\}_{j=1}^{sd})$.

5: Perform proximal mapping for each sensor: For SEV/logistic models:

$$\{\mathbf{p}_k^{(t+1)}\}_{k=1}^s = \max\left(0, 1 - \frac{\lambda_t}{\|\mathbf{v}_k^{(t)}\|_2}\right) \mathbf{v}_k^{(t)},$$

where $\mathbf{p}_k = (\{p_j\}_{j=(k-1)d+1}^{kd})$, and $\mathbf{v}_k = (\{v_j\}_{j=(k-1)d+1}^{kd})$.

6: **end for**

7: Stop when $\|(\mathbf{p}^{(t+1)}, q^{(t+1)}) - (\mathbf{p}^{(t)}, q^{(t)})\|_\infty < 10^{-5}$

8: **return** $\mathbf{p}^{(t+1)}, q^{(t+1)}$

And for q , we have

$$\frac{\partial g}{\partial q} = -\frac{n}{q} - \sum_{i=1}^n y_i + \sum_{i=1}^n \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}) \cdot y_i \quad (3.12)$$

where $\mathbf{x}_i^\top \mathbf{p} = \sum_{j=0}^{sd} p_j x_{ij}$.

Similarly, the gradient for logistic regression, $\nabla g(\mathbf{p}, q)_{\text{Logistic}}$, is also a $sd + 2$ length vector composed by $\{\frac{\partial g}{\partial p_j}\}_{j=0}^{sd}$ and $\frac{\partial g}{\partial q}$. For each j , we have

$$\frac{\partial g}{\partial p_j} = \sum_{i=1}^n x_{ij} + 2 \sum_{i=1}^n (-x_{ij}) \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}) \cdot (1 + \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}))^{-1}. \quad (3.13)$$

And $\frac{\partial g}{\partial q}$ is given by

$$\frac{\partial g}{\partial q} = \sum_{i=1}^n x_{ij} + 2 \sum_{i=1}^n (y_i) \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}) \cdot (1 + \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}))^{-1} \quad (3.14)$$

After calculating the gradient in line 3, we can then use these gradients for proximal gradient descent. Notice that in line 4 of algorithm 4, the gradient step, we update p_0 and q directly, while the intermediate parameter vector $\mathbf{v} = (\{v_j\}_{j=1}^{sd})$ updates from $\{p_j\}_{j=1}^{sd}$ and goes to proximal mapping step for further updating $\{p_j\}_{j=1}^{sd}$.

One concern with proximal gradient descent is the risk of gradient leakage. In the modern world of industrial prognostics, it is often encouraged to do collaborative training to achieve better model performance, especially in federated learning. In a federated learning environment, users share their local gradient during training steps and update the local gradient based on publicly aggregated results. This leaves room for information leakage, known as gradient leakage, often achieved by inversion attacks or inference attacks.

Therefore, we need an efficient technique to protect our gradient while maintaining the accuracy of training results. In other words, we would like to keep the training process of our sensor selection private and keep our data safe, while obtaining an accurate selection result. This is where differential privacy (DP) comes into play.

3.4 Differentially Private Sensor Selection

In this section, we introduce a differentially private sensor selection approach by solving perturbed proximal gradient descent for group lasso regularized (log)-location-scale regression.

3.4.1 Differentially Private Proximal Gradient Descent

We will first introduce the basic definitions and ideas of differential privacy (DP). In the context of DP, a dataset is viewed as a compendium of individual entries, and DP aims to ensure that adding or removing a single dataset entry affects the output of a function in a minimally discernible way. This concept necessitates defining "neighboring datasets." Two datasets,

D and D' , are deemed neighboring if their difference is at most one data entry. The formal definition of ϵ -Differential Privacy, based on this premise, is given below.

Definition 3 (ϵ -Differential Privacy (Dwork et al. 2006)). A mechanism \mathcal{A} is said to adhere to ϵ -differential privacy if, for any output O produced by \mathcal{A} and for any pair of neighboring datasets D and D' , the following inequality holds:

$$\Pr[\mathcal{A}(D) = O] \leq e^\epsilon \cdot \Pr[\mathcal{A}(D') = O]. \quad (3.15)$$

In this definition, $\Pr[\cdot]$ denotes the probability, and $\mathcal{A}(\cdot)$ symbolizes the algorithm or mechanism used to inject noise into the dataset to preserve privacy. The equation quantitatively assesses how closely the probabilities of achieving the same noisy output from datasets D and D' are aligned, governed by the multiplier e^ϵ , where ϵ is called the privacy budget. The smaller ϵ is, the tighter the privacy constraints, with ϵ nearing zero making e^ϵ approach one, thereby rendering the datasets increasingly indiscernible.

In the context of DP, a specific algorithm or procedure that ensures privacy protection is called a mechanism. Mechanisms usually add calibrated noise to a function's output according to different metrics. The Laplace Mechanism is one of the most commonly used mechanisms. In the Laplace mechanism, noise is drawn from a Laplace distribution $\text{Lap}\left(\frac{\Delta}{\epsilon}\right)$ and incorporated into a function's output to secure privacy (Dwork et al. 2006). The Δ denotes the sensitivity of a function, which determines the maximal possible change in the function's output between any two neighboring datasets. And the ϵ is our privacy budget, which is a user-controlled parameter, and often reflects the desired level of the user's privacy need. Essentially, higher sensitivity or a smaller privacy budget leads to the addition of more noise.

To fit the group lasso model for sensor selection, we propose the use of differentially private proximal gradient descent in the prevention of gradient leakage, specifically, by adding Laplace noises to the gradient in the proximal gradient descent algorithm.

In the basic version of the Differentially Private Proximal Gradient Descent algorithm we

will be shown below, Laplace noise is introduced to the gradient calculation to ensure privacy. The magnitude of the Laplace noise added is determined by the gradient function's sensitivity, which varies across different regression models. This sensitivity measurement guides the scale of noise from the Laplace distribution to be added, ensuring that the addition is tailored to each model's specific characteristics while maintaining the differential privacy of the process.

Algorithm 5 DP-PGD for LLS Regression with Group Lasso (Naïve)

Require: Database $\{y_i, \{x_{ij}\}_{j=0}^{s_d}\}_{i=1}^n$, regularization parameter λ , learning rate α , privacy budget ϵ

Ensure: parameter vector \mathbf{p} , parameter q

- 1: Initialize $\mathbf{p}^{(0)} = 0, q^{(0)} = 1$
- 2: **for** $t = 0, 1, 2, \dots$, **do**
- 3: Compute gradient $\nabla g(\mathbf{p}^{(t)}, q^{(t)})$ depends on the specific LLS model
- 4: Compute sensitivity $\Delta g(\mathbf{p}^{(t)}, q^{(t)})$ as equation 3.26 depends on the specific LLS model
- 5: Add Laplace noise to the gradient:

$$\nabla g(\mathbf{p}^{(t)}, q^{(t)}) = \nabla g(\mathbf{p}^{(t)}, q^{(t)}) + \xi$$

where $\xi = (\{\xi_u\}_{u=0}^{s_d+1})$ and ξ_u i.i.d $\sim \text{Laplace}(0, \Delta g(\mathbf{p}^{(t)}, q^{(t)})/\epsilon)$

- 6: Perform gradient step: $(\mathbf{p}_0^{(t+1)}, \mathbf{v}^{(t)}, q^{(t+1)}) = (\mathbf{p}^{(t)}, q^{(t)}) - \alpha \nabla g(\mathbf{p}^{(t)}, q^{(t)})$
 - 7: Perform proximal operation for each sensor $\{\mathbf{p}_k^{(t+1)}\}_{k=1}^s = \max\left(0, 1 - \frac{\lambda_t}{\|\mathbf{v}_k^{(t)}\|_2}\right) \mathbf{v}_k^{(t)}$
 - 8: **end for**
 - 9: Stop when $\|(\mathbf{p}^{(t+1)}, q^{(t+1)}) - (\mathbf{p}^{(t)}, q^{(t)})\|_\infty < 10^{-5}$
 - 10: **return** $\mathbf{p}^{(t+1)}, q^{(t+1)}$
-

Previously, we have derived the gradient functions for linear, SEV, and logistic models. In order to add differentially private noises to the gradient by the Laplace Mechanism, we also need to know the sensitivity of gradients, which will be used as the parameter to draw i.i.d. noise from the Laplace distribution.

To compute the sensitivity of the gradient, which means we need a measurement of the maximal change the gradient could have if one arbitrary data point gets removed. Suppose we remove a data point from our dataset, we can get a new gradient $\nabla g'$, and obtain the maximal change between the original gradient and the new gradient by comparing the two gradients

by taking $\|\nabla g - \nabla g'\|_1$, where $\|\cdot\|_1$ denotes the L1 norm. Notice the change of the gradient is bounded by the L1 norm.

Since the gradient functions of each type of LLS regressions are different, we need to derive the $\nabla g'$ and the differences between ∇g and $\nabla g'$ for each type of LLS models.

For the linear regression model, assume we remove a data point x_r , then we have a new gradient $\nabla g' = \left(\left\{\frac{\partial g'}{\partial \beta_j}\right\}_{j=0}^{sd}\right)$ given by

$$\frac{\partial g'}{\partial \beta_j} = \frac{1}{n-1} \sum_{\substack{i=1 \\ i \neq r}}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})(-x_{ij}) \quad (3.16)$$

And we can obtain $\nabla g - \nabla g'$ by calculating $\left\{\frac{\partial g}{\partial \beta_j} - \frac{\partial g'}{\partial \beta_j}\right\}_{j=0}^{sd}$ as follows:

$$\begin{aligned} \frac{\partial g}{\partial \beta_j} - \frac{\partial g'}{\partial \beta_j} &= \frac{1}{n} \sum_{i=1}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})(-x_{ij}) - \frac{1}{n-1} \sum_{\substack{i=1 \\ i \neq r}}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})(-x_{ij}) \\ &= \frac{1}{n} (y_r - \mathbf{x}_r^\top \boldsymbol{\beta})(-x_{rj}) + \frac{1}{n} \sum_{\substack{i=1 \\ i \neq r}}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})(-x_{ij}) - \frac{1}{n-1} \sum_{\substack{i=1 \\ i \neq r}}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})(-x_{ij}) \quad (3.17) \\ &= \frac{1}{n} (y_r - \mathbf{x}_r^\top \boldsymbol{\beta})(-x_{rj}) - \frac{1}{n(n-1)} \sum_{\substack{i=1 \\ i \neq r}}^n (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})(-x_{ij}) \end{aligned}$$

where $\mathbf{x}_i^\top \boldsymbol{\beta} = \sum_{j=0}^{sd} \beta_j x_{ij}$ and $\mathbf{x}_r^\top \boldsymbol{\beta} = \sum_{j=0}^{sd} \beta_j x_{rj}$.

For SEV and logistic models, it is a little bit trickier. Recall that for both SEV and logistic models, their gradients $\nabla g(\mathbf{p}, q) = \left(\frac{\partial g}{\partial p_0}, \frac{\partial g}{\partial p_1}, \dots, \frac{\partial g}{\partial p_{sd}}, \frac{\partial g}{\partial q}\right)^\top$ are $sd+2$ length vectors composed by $\left\{\frac{\partial g}{\partial p_j}\right\}_{j=0}^{sd}$ and $\frac{\partial g}{\partial q}$. To calculate $\|\nabla g - \nabla g'\|_1$, implies we need to compute the differences between each pair of terms and put them together, i.e.,

$$\left\| \frac{\partial g}{\partial p_0} - \frac{\partial g'}{\partial p_0}, \frac{\partial g}{\partial p_1} - \frac{\partial g'}{\partial p_1}, \dots, \frac{\partial g}{\partial p_{sd}} - \frac{\partial g'}{\partial p_{sd}}, \frac{\partial g}{\partial q} - \frac{\partial g'}{\partial q} \right\|_1 \quad (3.18)$$

Let us take a look at the SEV regression model. Suppose we remove an arbitrary data point

where $i = r$ and obtain a new expression for $\{\frac{\partial g'}{\partial p_j}\}_{j=0}^{sd}$. For all j , we have

$$\frac{\partial g'}{\partial p_j} = \sum_{\substack{i=1 \\ i \neq r}}^n x_{ij} + \sum_{\substack{i=1 \\ i \neq r}}^n \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}) \cdot (-x_{ij}) \quad (3.19)$$

Therefore, the difference between two gradient term at each direction j is given by

$$\frac{\partial g}{\partial p_j} - \frac{\partial g'}{\partial p_j} = x_{rj} + \exp(y_r q - \mathbf{x}_r^\top \mathbf{p}) \cdot (x_{rj}) \quad (3.20)$$

And the difference between $\frac{\partial g}{\partial q}$ and $\frac{\partial g'}{\partial q}$ is given by

$$\frac{\partial g}{\partial q} - \frac{\partial g'}{\partial q} = -\frac{1}{q} - y_r + \exp(y_r q - \mathbf{x}_r^\top \mathbf{p}) \quad (3.21)$$

Similarly, for the logistic model, assume we remove a datapoint where $i = r$ and calculate the difference between two terms. After removing the datapoint, we have

$$\frac{\partial g'}{\partial p_j} = \sum_{\substack{i=1 \\ i \neq r}}^n x_{ij} + 2 \sum_{\substack{i=1 \\ i \neq r}}^n (-x_{ij}) \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}) (1 + \exp(y_i q - \mathbf{x}_i^\top \mathbf{p}))^{-1} \quad (3.22)$$

The difference between $\frac{\partial g}{\partial p_j}$ and $\frac{\partial g'}{\partial p_j}$ is

$$\frac{\partial g}{\partial p_j} - \frac{\partial g'}{\partial p_j} = x_{rj} + 2(-x_{rj}) \exp(y_r q - \mathbf{x}_r^\top \mathbf{p}) (1 + \exp(y_r q - \mathbf{x}_r^\top \mathbf{p}))^{-1} \quad (3.23)$$

And the difference between $\frac{\partial g}{\partial q}$ and $\frac{\partial g'}{\partial q}$ is given by

$$\frac{\partial g}{\partial q} - \frac{\partial g'}{\partial q} = -\frac{1}{q} - y_r + 2(y_r) \exp(y_r q - \mathbf{x}_r^\top \mathbf{p}) (1 + \exp(y_r q - \mathbf{x}_r^\top \mathbf{p}))^{-1} \quad (3.24)$$

By calculating $\{(\frac{\partial g}{\partial p_j} - \frac{\partial g'}{\partial p_j})\}_{j=0}^{sd}$ and $(\frac{\partial g}{\partial q} - \frac{\partial g'}{\partial q})$, we can obtain the change between ∇g and $\nabla g'$ on every direction.

However, the calculation of exact sensitivity has two major challenges. First, notice that all

the above expressions measure the change of one variable for one data point (x_r) . Since the data point to be removed is arbitrary, we need to find which point gives the maximal change for a given dataset. In other words, to obtain the maximal possible change of gradient, we need to first compute the change for every variable for all data points, then compute the norms of the vectors of differences, and finally find the maximal value among all candidate norms, which is

$$\max_{(x_i, y_i)} \left\| \frac{\partial g}{\partial p_0} - \frac{\partial g'}{\partial p_0}, \frac{\partial g}{\partial p_1} - \frac{\partial g'}{\partial p_1}, \dots, \frac{\partial g}{\partial p_{sd}} - \frac{\partial g'}{\partial p_{sd}}, \frac{\partial g}{\partial q} - \frac{\partial g'}{\partial q} \right\|_1 \quad (3.25)$$

Second, notice the parameters p_j and q in the above expressions. For every iteration, we are adding Laplace noise to the current gradient using the sensitivity of the current gradient, and the sensitivity should be derived based on the current parameters. For example, the sensitivity of the gradient for t -th iteration is given by

$$\Delta g(\mathbf{p}^{(t)}, \mathbf{q}^{(t)}) = \max_{(x_i, y_i)} \left\| \frac{\partial g}{\partial p_0^{(t)}} - \frac{\partial g'}{\partial p_0^{(t)}}, \frac{\partial g}{\partial p_1^{(t)}} - \frac{\partial g'}{\partial p_1^{(t)}}, \dots, \frac{\partial g}{\partial p_{sd}^{(t)}} - \frac{\partial g'}{\partial p_{sd}^{(t)}}, \frac{\partial g}{\partial q^{(t)}} - \frac{\partial g'}{\partial q^{(t)}} \right\|_1 \quad (3.26)$$

In summary, if we would like to use the exact value of the sensitivity of the gradient for our DP approach, we need to iterate over all data points in every descent iteration to solve for the current sensitivity to be used for DP, which is quite computationally inefficient.

It would be ideal if we could find a relatively fixed upper bound to be used instead of solving for the exact value of sensitivity. To solve this critical issue, we will next propose an enhanced version of our differentially private proximal gradient descent algorithm, incorporated with a practical technique called gradient clipping, along with other improvements.

3.4.2 Revised Algorithm

In practice, we can use gradient clipping to deal with this problem. Gradient clipping is a technique applied to control the magnitude of gradients by enforcing a maximum threshold. If the norm of a gradient exceeds this threshold, it is scaled down proportionally (Pascanu

et al. 2013). Suppose we have a clipping threshold C . Given gradient vector ∇g , to perform the gradient clipping, first we would like to calculate the norm of the gradient, $\|\nabla g\|$, and if $\|\nabla g\| > C$, scale the gradient down to the threshold, i.e., $\nabla g_{\text{clipped}} = \frac{C}{\|\nabla g\|} \nabla g$. Since $\frac{\nabla g}{\|\nabla g\|}$ is a unit vector, the new $\nabla g_{\text{clipped}}$ will have norm C after the scaling. And if $\|\nabla g\| \leq C$, we leave the gradient unchanged, i.e., $\nabla g_{\text{clipped}} = \nabla g$. In summary, gradient clipping ensures that the vector of gradient ∇g has a L2 norm at most C .

Algorithm 6 DP-PGD for LLS Regression with Group Lasso (Enhanced)

Require: Database $\{y_i, \{x_{ij}\}_{j=0}^{sd}\}_{i=1}^n$, regularization parameter λ , GCV tuned λ , privacy budget ϵ , clipping threshold C

Ensure: parameter vector \mathbf{p} , parameter q

- 1: Initialize $\mathbf{p}^{(0)} \sim \mathcal{N}(0, 1)$, $q^{(0)} = 1$
- 2: **for** $t = 0, 1, 2, \dots$, **do**
- 3: Compute gradient $\nabla g(\mathbf{p}^{(t)}, q^{(t)})$ depends on the specific LLS model
- 4: Clip gradient:

$$\nabla g(\mathbf{p}^{(t)}, q^{(t)}) = \nabla g(\mathbf{p}^{(t)}, q^{(t)}) / \max\left(1, \frac{\|\nabla g(\mathbf{p}^{(t)}, q^{(t)})\|_2}{C}\right)$$

- 5: Add Laplace noise to the gradient:

$$\nabla g(\mathbf{p}^{(t)}, q^{(t)}) = \nabla g(\mathbf{p}^{(t)}, q^{(t)}) + \xi$$

where $\xi = (\{\xi_u\}_{u=0}^{sd+1})$ and ξ_u i.i.d $\sim \text{Laplace}(0, 2C\sqrt{sd+2}/\epsilon)$

- 6: Perform gradient step: $(\mathbf{p}_0^{(t+1)}, \mathbf{v}^{(t)}, q^{(t+1)}) = (\mathbf{p}^{(t)}, q^{(t)}) - \alpha \nabla g(\mathbf{p}^{(t)}, q^{(t)})$
 - 7: Perform proximal operation for each sensor $\{\mathbf{p}_k^{(t+1)}\}_{k=1}^s = \max\left(0, 1 - \frac{\lambda_t}{\|\mathbf{v}_k^{(t)}\|_2}\right) \mathbf{v}_k^{(t)}$
 - 8: **end for**
 - 9: Stop when $\|\mathbf{p}^{(t+1)} - \mathbf{p}^{(t)}\|_\infty < 10^{-5}$
 - 10: **return** $\mathbf{p}^{(t+1)}, q^{(t+1)}$
-

To derive the parameter for drawing noise from $\text{Lap}(\frac{\Delta}{\epsilon})$ and satisfies ϵ -DP, we need to find some value Δ' where $\|\nabla g - \nabla g'\|_1 \leq \Delta'$ (Dwork et al. 2014). By the well-known inequality of L1 and L2 norms, for all $\mathbf{x} \in \mathcal{R}^d$, $\|\mathbf{x}\|_2 \leq \|\mathbf{x}\|_1 \leq \sqrt{d}\|\mathbf{x}\|_2$. In our case, $\|\nabla g\|_2 \leq C$, and the dimension of ∇g is $sd + 2$, therefore we must have $\|\nabla g\|_1 \leq C\sqrt{sd+2}$. Hence $\|\nabla g - \nabla g'\|_1 \leq 2C\sqrt{sd+2}$. In other words, drawing i.i.d noise from $\text{Lap}\left(\frac{2C\sqrt{sd+2}}{\epsilon}\right)$ and add to gradient as step 5 in algorithm

6 satisfies ϵ -DP.

In our enhanced algorithm 6, we would like to perform the gradient clipping before adding DP noise. Notice that in a non-DP context, introducing gradient clipping to the proximal gradient descent can potentially affect its convergence properties. Gradient clipping is primarily used to combat the exploding gradient problem, which is not typically a concern in the standard proximal gradient method because it inherently involves steps that prevent excessively large updates due to the proximal mapping. However, as our focus here is to apply DP method in protecting the gradient, implementing gradient clipping before the addition of DP noise serves two primary purposes: it reduces the sensitivity of the gradients, allowing for a lesser quantity of noise to be added while achieving the desired level of differential privacy; and it establishes a predetermined upper limit for the magnitude of gradient changes. This limit, determined by the clipping threshold C , can then be utilized in the replacement of Δ in the process of adding differential privacy noise. In a previous study of Abadi et al. (2016), the Gaussian mechanism with parameter σ comes from the clipping threshold C has been utilized to ensure (ϵ, δ) -DP for protecting the gradients in stochastic gradient descent.

In addition, we also add several other enhancements in algorithm 6 after numerical investigation with the naïve algorithm 5, including changes in initialization, and GCV-tuned lambda. Specifically, we notice it is better to use a small random vector rather than a zero vector when initial \mathbf{p}^0 .

We will next demonstrate the performance of our proposed approach using both simulations and case studies. We experimented with both the naïve and improved algorithms and will present the findings of the revised algorithm in the following sections.

3.5 Simulation Study

In this section, we conduct a simulation study to evaluate the performance of the proposed DP approach.

3.5.1 Experimental Settings

In our simulation, we created data from 3 sensors, with each sensor providing 2 features, resulting in a total of 6 features for n samples (we'll specify the number of samples later). These 6 features are organized into 3 groups corresponding to the 3 sensors. We designed the simulation so that sensors 1 and 2 contribute valuable information to the model, while sensor 3 does not.

Furthermore, sensors 2 and 3 are linked by a correlation factor, ρ , which we can adjust. Considering each sensor has 2 features, to implement the correlation, we let all four features from sensors 2 and 3 (features 3 to 6) correlate by utilizing Choleski factorization. To do so, we first generate an initial random $n \times 6$ matrix from a standard normal distribution, $\mathcal{N}(0, 1)$. The first two columns would be used as sensor 1's data directly. For sensors 2 and 3, we take the controlled parameter ρ to generate the correlation matrix and calculate Cholesky factor, then generate correlated data by multiplying columns 3 to 6 of the random matrix and the Cholesky factor. Keeping the first two columns from the initial random matrix and replacing columns 3 to 6 with the new correlated data, we obtained our desired data for \mathbf{X} .

Each of the regression coefficients is also drawn independently from a standard normal distribution. The response variable for each sample, \tilde{y}_i , is calculated by combining the first four feature values (from sensors 1 and 2) with regression coefficients, plus an error term ε_i . This error term is sampled from either a normal, SEV, or logistic distribution, chosen based on the specific regression model in question, with its mean set to 0 and standard deviation to 1.

We then use the simulated multi-sensor data to assess the performance of the proposed differentially private approach, denoted as "DP", and compared to results from classical LLS regression with group lasso, designated as "Non-DP", which we used as our benchmark in this study. We would like to investigate how the proposed approach performs in various conditions, including (1) training sample size (i.e., n), (2) correlation between sensors (i.e., ρ), and (3) privacy budgets (i.e., ϵ). These factors are expected to influence the performance of the proposed DP method. The specific settings of these three factors used in the simulation study

are summarized as follows:

- To evaluate the impact of sample size on our DP approach's performance, we test on different values of sample size n , including 1000, 2500, 5000, 7500, and 10000, under $\rho = 0.2, 0.5, 0.7, 0.8$ and 0.9 , while $\epsilon = 5$.
- To examine the influence of sensor correlation, we test on $\rho = 0.2$ to 0.9 , with an increase of 0.1 , under the same five levels of n as above, while $\epsilon = 5$.
- To explore different levels of privacy budget ϵ , we used $\epsilon = 1000, 100, 10, 1$ and 0.1 , while $n = 10000$ and $\rho = 0.5$.

The metric we will be using to evaluate the performance is the correct sensor selection percentage, which we will also refer to as selection accuracy, determined by the ratio of how many times we obtain a fully correct selection to the total number of experiment repetitions. We say there is a correct selection only if, in the final output of fitted β , all informative sensors (i.e., sensor 1 and sensor 2) are kept and the non-informative sensor (i.e., sensor 3) gets filtered out. In any other case, we say it is a false selection. We ran 100 repetitions for each different settings, and will report the percentage of correct selections under different experimental settings.

3.5.2 Results and Analysis

Accuracy vs. Training Data Size

In the first set of experiments, we investigate the performance of both non-DP and DP approaches regarding different levels of sample size. The sample size investigated ranged from 1000 to 10000, and we also looked into the impact of sample size under different sensor correlations. We present results for linear regression in Figures 3.1 and 3.2, SEV regression in Figures 3.3 and 3.4, and logistic regression in Figures 3.5 and 3.6. For each pair of plots, the left one shows the results from the non-DP approach, and the right one shows the results from our

DP approach. Within one plot, each different line corresponds to a different level of sensor correlation. The x-axis represents sample sizes, and the y-axis represents selection accuracy.

Based on the results, we can observe that the performance of both non-DP and DP approaches depends on the sample size. In general, the larger the sample size, the better performance we have. This makes sense because a better representation of the underlying population is achieved with larger sample sizes, which naturally lowers errors in regression models. We also have the following observations:

When the sample size is relatively smaller, e.g., at $n \leq 5000$, the non-DP approach has significantly better performance than the DP approach, especially for higher correlations. In other words, for the DP approach to reach comparable performance as the non-DP model, a much larger sample size is needed. For example, in the logistic regression models' results, when $\rho = 0.2$, the DP approach reaches 90% accuracy at $n = 5000$, while the non-DP approach reaches the same level of accuracy at $n = 2500$.

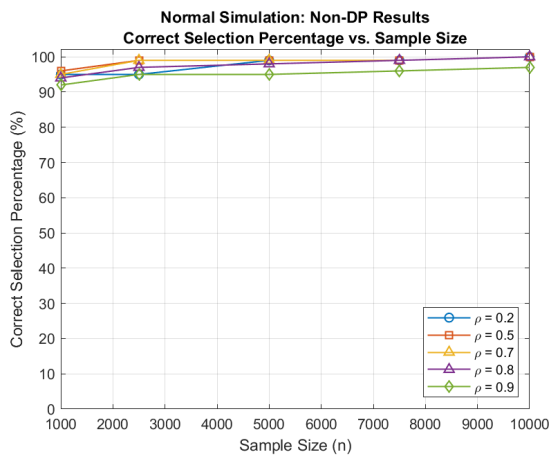


Figure 3.1: Simulation: Accuracy vs. Sample Size n , Linear Regression, Non-DP

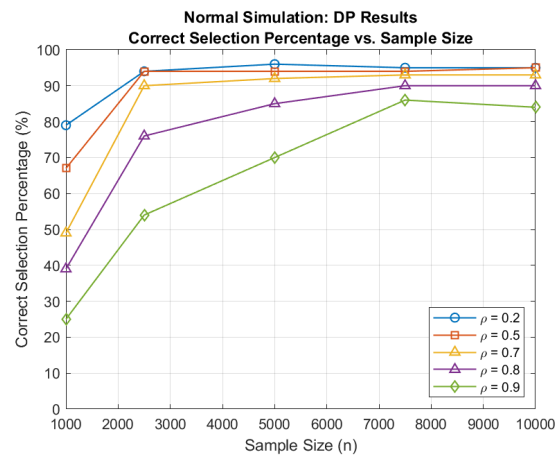


Figure 3.2: Simulation: Accuracy vs. Sample Size n , Linear Regression, DP

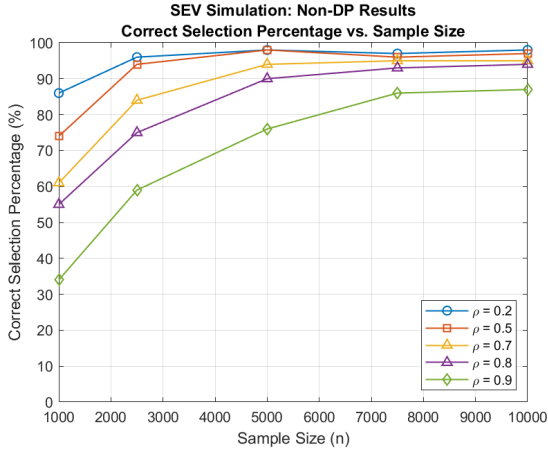


Figure 3.3: Simulation: Accuracy vs. Sample Size n , SEV Regression, Non-DP

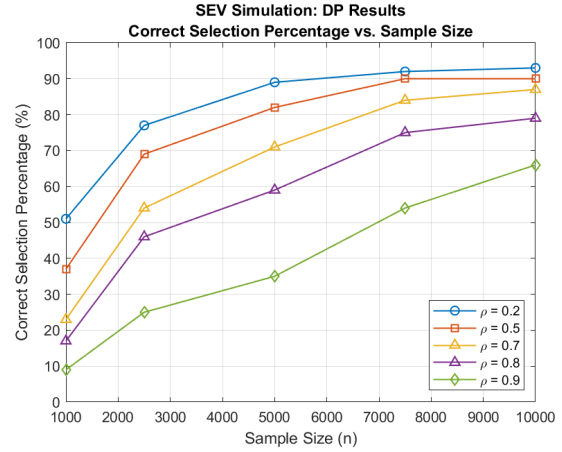


Figure 3.4: Simulation: Accuracy vs. Sample Size n , SEV Regression, DP

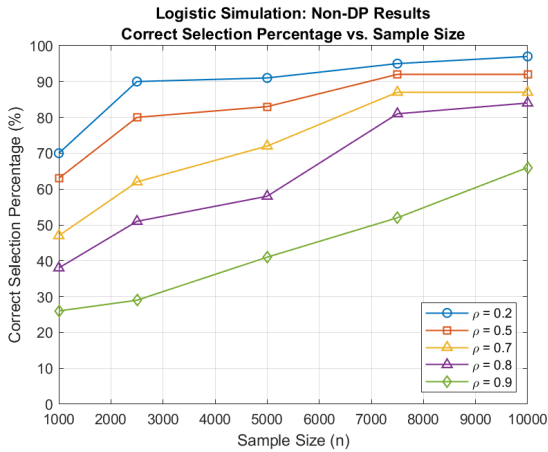


Figure 3.5: Simulation: Accuracy vs. Sample Size n , Logistic Regression, Non-DP

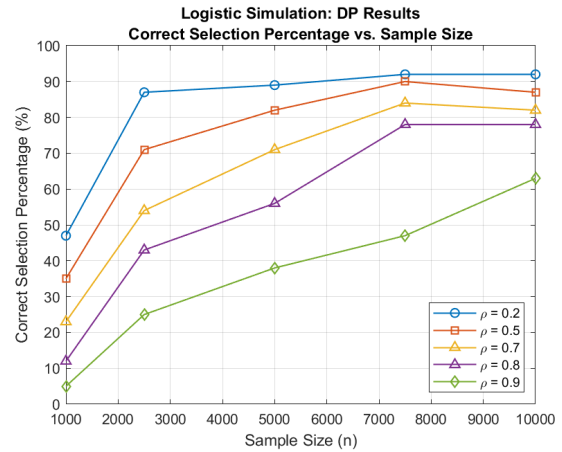


Figure 3.6: Simulation: Accuracy vs. Sample Size n , Logistic Regression, DP

In the case of non-DP linear regression, we observed stable and high accuracy across the chosen range of sample sizes and correlation levels, as shown in Figures 3.1 and 3.7. In our experiments, this model's accuracy showed a gradual increase within the sample size range of $100 \leq n \leq 1000$, and a rapid decline within the correlation coefficient range of $0.9 \leq \rho \leq 0.99$. Despite these variations, we maintained a consistent setting across our analysis to facilitate a

clear comparison of the results from other models.

Accuracy vs. Sensor Correlation

In the next, we will look into how the non-DP and DP approaches perform towards different levels of sensor correlation. The range of correlations we will evaluate ranges from 0.2 to 0.9, while the results are reported under different sample sizes. We present results for linear regression in Figures 3.7 and 3.8, SEV regression in Figures 3.9 and 3.10, and logistic regression in Figures 3.11 and 3.12. In each plot, the different lines represent different levels of sample size. The x-axis shows sensor correlation levels, while the y-axis shows selection accuracy.

Through observation of these results, we can conclude that both the non-DP approach and the DP approach are sensitive to sensor correlation. In general, a higher correlation leads to worse performance. This is easily understandable as we set up the correlation between an informative and a non-informative sensor, and we only consider a selection result to be correct when all sensors have been classified correctly, i.e., the correlated informative sensor and non-informative sensor must be distinguished.

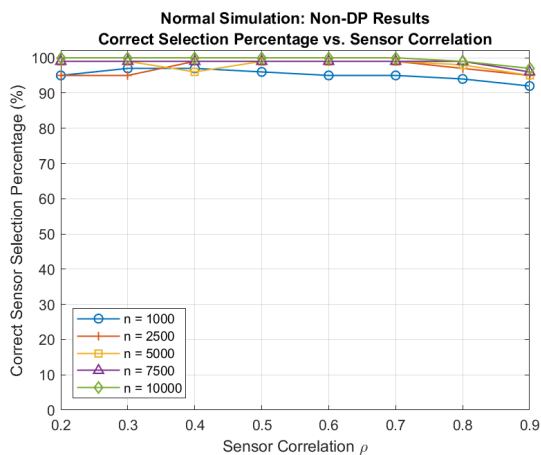


Figure 3.7: Simulation: Accuracy vs. Correlation ρ , Linear Regression, Non-DP

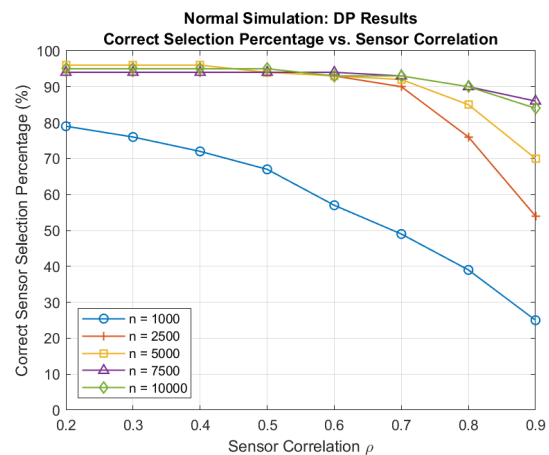


Figure 3.8: Simulation: Accuracy vs. Correlation ρ , Linear Regression, DP

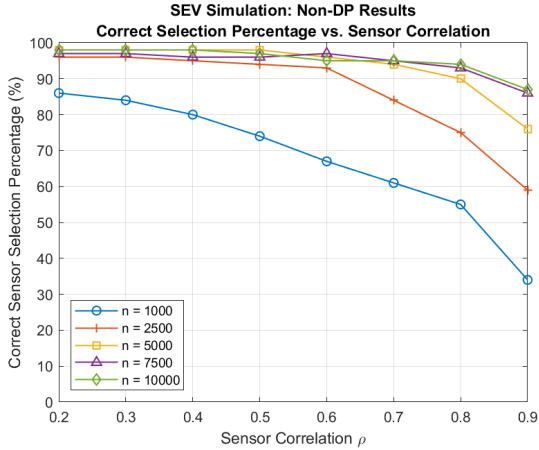


Figure 3.9: Simulation: Accuracy vs. Correlation ρ , SEV Regression, Non-DP

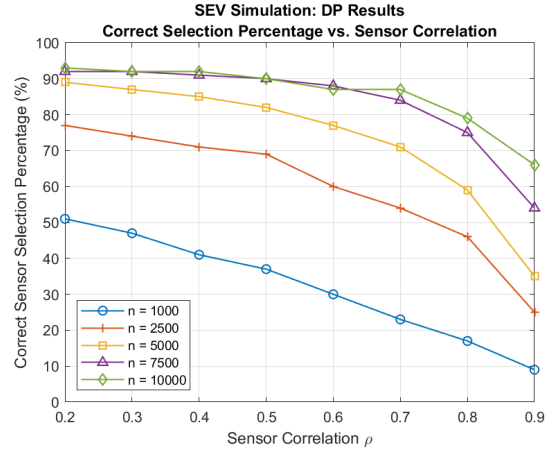


Figure 3.10: Simulation: Accuracy vs. Correlation ρ , SEV Regression, DP

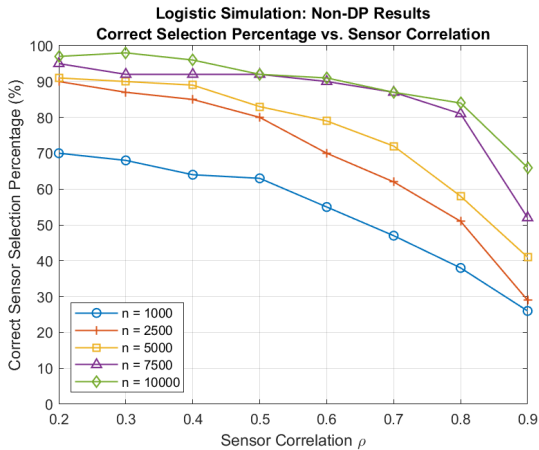


Figure 3.11: Simulation: Accuracy vs. Correlation ρ , Logistic Regression, Non-DP

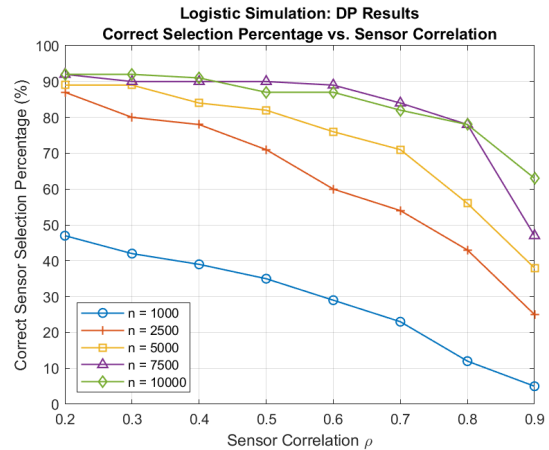


Figure 3.12: Simulation: Accuracy vs. Correlation ρ , Logistic Regression, DP

While similar patterns and trends can be seen in non-DP and DP results, we can see that the accuracy of the DP approach is in general lower than the non-DP approach. However, we also notice that both non-DP and DP approaches perform relatively stable for $\rho \leq 0.5$, especially when the sample size is large. For example, in the SEV regression's results, we can achieve an accuracy greater or equal to 90% at a relatively high correlation $\rho = 0.8$ when $n \geq 5000$ for the

non-DP approach. For the DP approach, we can only achieve similar accuracy when $\rho < 0.5$ and $n \geq 7500$.

DP Privacy Budget vs. Computational Cost

The performance evaluation of the DP privacy budget is different from the previous two results. Since the simulation data we set up is rather simple with only 3 sensors and two of them are informative, we notice that given enough computation iterations, different levels of privacy budget ϵ do not impact the final selection results much. It is reasonable since the DP noise we added is to the gradient step of proximal gradient descent, and the group variable selection is majored determined by the proximal mapping step.

However, we also notice that when the privacy budget is small, the program takes a significantly longer time to converge, as a smaller privacy budget implies larger noise added to the gradient and a stronger perturbation in the descent direction. As the privacy budget controls how strong our protection is on the gradient, we are also interested in learning the computational cost associated with different levels of privacy protection.

To investigate this trade-off, we conducted experiments on simulated data with a fixed sample size of $n = 10,000$ and a sensor correlation of $\rho = 0.5$. We varied the privacy budget from $\epsilon = 1000$ down to $\epsilon = 0.1$ and recorded the number of computation iterations required for the algorithm to converge under each setting. The results of this experiment are presented in Figure 3.13.

The plot in Figure 3.13 clearly illustrates the inverse relationship between the privacy budget and the computational cost. As the privacy budget decreases, the number of iterations required for convergence increases significantly. This behavior can be attributed to the increased level of noise introduced by smaller privacy budgets. When ϵ is small, the Laplace mechanism adds more noise to the gradient in each iteration of the proximal gradient descent algorithm. Consequently, the descent direction is perturbed more severely, leading to slower convergence and a higher number of iterations.

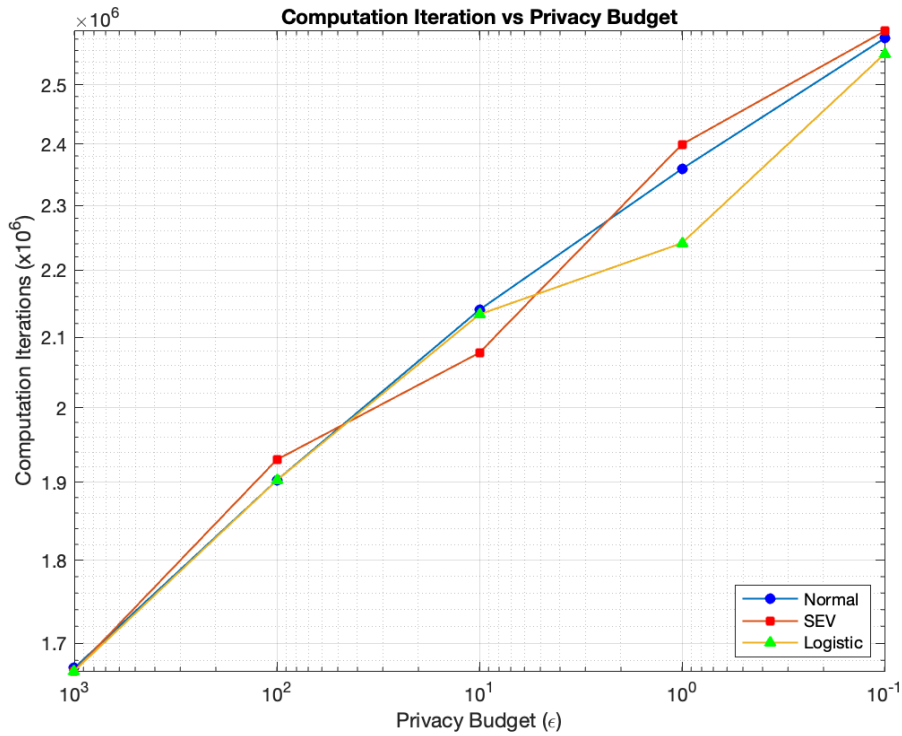


Figure 3.13: Simulation: Computational Cost vs. Privacy Budget

The increased computational cost associated with smaller privacy budgets has important implications for the practical application of our sensor selection approach. In real-world scenarios, where computational resources and time may be limited, it is crucial to strike a balance between privacy protection and computational efficiency. The choice of an appropriate privacy budget should take into account not only the desired level of privacy but also the available computational resources and the time constraints of the application.

It is worth noting that the privacy budget is not the only factor influencing the computational cost of our approach. Other factors, such as the sample size, the number of sensors, and the complexity of the underlying data distribution, can also impact the convergence rate and the number of iterations required. Further experiments and analyses could be conducted to investigate the interplay between these factors and the privacy budget in determining the computational cost.

The analysis of the relationship between the DP privacy budget and computational cost highlights the importance of considering the practical implications of privacy protection in sensor selection tasks. The inverse relationship between the privacy budget and the number of iterations required for convergence underscores the need for efficient optimization algorithms and parallel computing techniques to mitigate the computational burden of strong privacy guarantees.

3.6 Case Study

In our case study, we employ a dataset originally presented in Saxena and Goebel (2008) to assess the efficacy of our differentially private sensor selection approach.

3.6.1 Dataset and Experimental Settings

The dataset encompasses multi-sensor degradation data from aircraft turbofan engines, featuring two distinct sets of engines. The first set includes degradation data for 100 training engines that were observed until failure. The second set comprises data from another 100 test engines; these engines were not run to failure but instead had their operations halted at various, randomly determined points before failure. For these test engines, the dataset provides the actual times-to-failure (TTF). Monitoring of each engine was conducted using 21 distinct sensors, capturing a comprehensive range of degradation signals.

Fang et al. (2017) explored sensor selection techniques within the context of various Log-Location-Scale (LLS) regression models, addressing the inherent challenge posed by the variable lengths of degradation signals found in both training and test datasets. This variability in signal length is attributed to the practice of halting machine operation for maintenance or replacement upon reaching failure, thereby limiting the observable degradation signal to the failure point. Furthermore, the differing failure times across machines result in inconsistent signal lengths across the dataset. To navigate the issue of signal length variability, Fang et al.

(2017) proposed a time-varying framework that standardizes the length of degradation signals by truncating them to a uniform length. In our study, we apply this approach by truncating the degradation signals of all engines to the first 150 timestamps, thereby standardizing the dataset for analysis. Engines with a TTF less than 150 are excluded from consideration. Following this truncation process, the training dataset is reduced to 94 engines, while the test dataset is pared down to 37 engines. This adjustment facilitates a more structured and comparable analysis across the engine dataset, ensuring that the evaluation of the differentially private LLS regression model is conducted consistently.

In our analytical approach, each timestamp is regarded as an individual feature. Consequently, post-truncation, we are left with 150 features per sensor. Given the significant correlations observed among the degradation signals from the three sensors, we initially employ Principal Component Analysis (PCA) to reduce the data from each sensor into two principal features. This step effectively lowers the dimensionality of our dataset while retaining critical information. Then, we put together the smaller amounts of data from all the sensors. This converts the training dataset to 94 data points and 42 features, shown as a 94 by 42 matrix, and the testing dataset to 37 data points and 42 features, shown as a 37 by 42 matrix.

We then applied sensor selection methods to the prepared data, using the non-DP results as the benchmark for comparison, just like in the simulation study. Unlike the simulation study, we will not be able to experiment with sample size n and sensor correlation ρ due to the nature of the data for the case study. However, we noticed that for different privacy budgets, the DP approach yields different results. In this study, we will look at how well the different private sensor selection methods worked with different kinds of LLS regressions, including linear, SEV, and logistic, and in a range of privacy budget settings.

To better catch the impact of the privacy budget, we terminate the experiment after hitting a maximal number of iterations for all different levels of ϵ , instead of letting it run to convergence. And when interpreting the parameters, we use a small threshold instead of zeros to determine the informative or non-informative sensors. To understand how the DP models held up against

the randomness introduced by Laplace distribution-based noise, we will repeat the evaluation 100 times for each DP model and aggregate the voting results. The outcomes from these repeated tests will then be compiled to provide a comprehensive overview of the DP models' performance and their reliability in sensor selection with privacy considerations.

For each of the LLS regression models, we will use the result of the non-DP approach as our benchmark, and compare the results of different levels of privacy budget from the DP approach to the benchmark. Since the dataset is determinant and the only randomness is the DP noises, the result we obtain from non-DP models will be certain. Therefore, in the result tables, for the non-DP column, we will be shown the norm of each sensor's parameters. For DP results, we will be shown the percentage of each sensor's selection rate over the 100 repetitions.

3.6.2 Results and Analysis

Table 3.1 shows results for linear regression with group lasso. For non-DP results, we list the norm of group parameters for each sensor, and for DP results, we are showing the aggregated voting results for how many times each sensor has been selected out of 100 repetitions. Tables 3.2 and 3.3 represent results from SEV and Logistic models, respectively.

The results in the tables are as expected. When the privacy budget is relatively large, the noise added to the gradient does not impact the final selection much, as we have a proximal mapping step after the gradient step. However, the smaller the epsilon goes, the more noise we add to the gradient, and the more perturbation we have in the descent direction. As we terminate the experiment after hitting a certain number of iterations, we observe that for $\epsilon = 1$, it is hard to achieve a clear cut for informative and non-informative sensors directly from the solved parameters in one experiment. Nevertheless, the aggregated voting results can still reveal the information we wanted: find the most informative sensors. We notice that the highest-voted sensors in the strict privacy budget are aligned with non-DP results.

Table 3.1: Case Study: Linear Model Results, DP Privacy Budget

Sensor	Non-DP	epsilon = 10	epsilon =5	epsilon = 2	epsilon = 1
1	0	0%	0%	0%	37%
2	0	0%	0%	0%	47%
3	2.7468	100%	100%	100%	100%
4	2.6206	100%	100%	100%	100%
5	0	0%	0%	0%	35%
6	0	0%	0%	0%	40%
7	17.8215	100%	100%	100%	54%
8	0	0%	0%	0%	34%
9	0.2197	0%	0%	0%	37%
10	0	0%	0%	0%	36%
11	0	100%	0%	0%	33%
12	0	0%	0%	0%	32%
13	0	0%	0%	0%	41%
14	0	0%	0%	0%	34%
15	0	0%	0%	0%	39%
16	0	0%	0%	0%	28%
17	4.2496	0%	100%	100%	69%
18	0	0%	0%	0%	36%
19	0	0%	0%	0%	29%
20	0	0%	0%	0%	34%
21	0	0%	0%	0%	43%

Table 3.2: Case Study: SEV Model Results, DP Privacy Budget

Sensor	Non-DP	epsilon = 10	epsilon =5	epsilon = 2	epsilon = 1
1	0	0%	0%	0%	35%
2	0	0%	0%	1%	24%
3	0.2547	95%	54%	86%	72%
4	2.7508	96%	86%	87%	80%
5	0	0%	0%	0%	30%
6	0	0%	0%	0%	33%
7	0	0%	0%	1%	41%
8	0	0%	0%	0%	31%
9	2.3724	81%	86%	87%	79%
10	0	0%	0%	1%	28%
11	0	0%	0%	3%	23%
12	0	0%	0%	3%	38%
13	0	0%	0%	1%	27%
14	2.1828	96%	86%	83%	43%
15	0	0%	0%	0%	30%
16	0	0%	0%	1%	31%
17	0	0%	0%	2%	34%
18	0	0%	0%	0%	31%
19	0	0%	0%	1%	31%
20	0	0%	0%	0%	37%
21	0	0%	0%	2%	39%

Table 3.3: Case Study: Logistic Model Results, DP Privacy Budget

Sensor	Non-DP	epsilon = 10	epsilon =5	epsilon = 2	epsilon = 1
1	0	0%	0%	1%	31%
2	0	0%	0%	2%	34%
3	2.0952	100%	100%	83%	92%
4	1.9591	100%	100%	83%	92%
5	0	0%	0%	1%	32%
6	0	0%	0%	0%	30%
7	0	0%	0%	0%	42%
8	0	0%	0%	0%	37%
9	0.0083	0%	3%	16%	48%
10	0	0%	0%	0%	39%
11	0	0%	0%	2%	32%
12	0	0%	0%	0%	42%
13	0	0%	0%	0%	41%
14	0.6211	100%	100%	83%	88%
15	0	0%	0%	1%	31%
16	0	0%	0%	0%	40%
17	0	0%	0%	0%	36%
18	0	0%	0%	0%	29%
19	0	0%	0%	0%	39%
20	0	0%	0%	1%	39%
21	0	0%	0%	2%	33%

One interesting observation from the case study results is the consistency of sensor selection across different LLS regression models. In Tables 3.1, 3.2, and 3.3, we can see that sensors 3, 4, and 14 are consistently selected as informative sensors across all three regression models (linear, SEV, and logistic) when the privacy budget is relatively high ($\epsilon = 10$ and $\epsilon = 5$). This consistency suggests that these sensors are robust indicators of engine degradation, regardless of the specific regression model employed. The fact that our differentially private approach can identify these sensors consistently across different models highlights its effectiveness in capturing the underlying patterns in the data.

Moreover, the results reveal interesting dynamics between the privacy budget and the selection of informative sensors. As the privacy budget decreases, we observe a gradual decline in the selection frequency of some informative sensors. For example, in Table 3.2 (SEV regression), sensor 14 is selected 96% of the time when $\epsilon = 10$, but its selection frequency drops to 43% when $\epsilon = 1$. Similarly, in Table 3.3 (logistic regression), the selection frequency of sensor 14 decreases from 100% at $\epsilon = 10$ to 88% at $\epsilon = 1$. This decline can be attributed to the increased level of noise introduced by stronger privacy guarantees, which makes it more challenging to consistently identify informative sensors in individual runs.

However, it is important to note that even at the strictest privacy budget ($\epsilon = 1$), the most informative sensors still exhibit relatively high selection frequencies compared to the uninformative sensors. For instance, in Table 3.1 (linear regression), sensors 3 and 4 are selected 100% of the time, while the selection frequencies of uninformative sensors remain below 50%. This observation suggests that our differentially private approach can still effectively distinguish between informative and uninformative sensors, even in the presence of strong privacy protection.

Another noteworthy aspect of the case study results is the presence of some false positives, particularly at lower privacy budgets. In Table 3.1, sensor 11 is selected 100% of the time when $\epsilon = 10$, despite being uninformative according to the non-DP results. Similarly, in Table 3.2, sensor 9 is selected with high frequency across all privacy budgets, although it is not considered

informative in the non-DP setting. These false positives can be attributed to the noise introduced by differential privacy, which can occasionally lead to the selection of uninformative sensors. However, it is crucial to interpret these false positives in the context of the overall sensor selection patterns. While some uninformative sensors may be selected with higher frequencies at lower privacy budgets, the most informative sensors still exhibit consistent and high selection frequencies across different privacy levels. This suggests that our approach is robust to the presence of false positives and can still effectively identify the most important sensors for engine degradation monitoring.

Furthermore, the case study results highlight the importance of aggregating results over multiple differentially private runs. By considering the selection frequencies across 100 repetitions, we can mitigate the impact of noise and obtain a more reliable assessment of sensor importance. The voting mechanism employed in our approach allows us to filter out the noise and focus on the consistent patterns in sensor selection. This aggregation strategy is particularly valuable in scenarios with strict privacy requirements, where the noise level is higher, and individual runs may yield more variable results. In addition to the observations discussed above, the case study results also provide insights into the scalability and computational efficiency of our approach. The experiments were conducted on a dataset with 21 sensors and 150 timestamps, demonstrating the ability of our method to handle a reasonable number of sensors and data points. The fact that we were able to obtain meaningful results while preserving privacy suggests that our approach can be applied to real-world scenarios with similar data characteristics.

In conclusion, the detailed analysis of the case study results provides a deeper understanding of the behavior and performance of our differentially private sensor selection approach in a real-world scenario. The consistency of sensor selection across different regression models, the impact of privacy budgets on selection frequencies, the presence of false positives, and the importance of aggregation strategies are key insights that emerge from the analysis. These findings highlight the effectiveness and robustness of our approach, while also identifying potential

areas for future research and improvement. By leveraging the power of differential privacy and aggregation techniques, our method offers a promising solution for privacy-preserving sensor selection in various domains where data privacy is a critical concern.

3.7 Conclusions

In this chapter, we have proposed differentially private proximal gradient descent specifically designed for solving (log)-location-scale regression with group lasso penalty. We incorporated differential privacy into the gradient step of proximal gradient descent regarding three specific LLS regression models, Linear, SEV and Logistics, which can also be extended to Log-normal, Weibull, and Log-logistic regression. The integration of differential privacy is achieved through the Laplace mechanism, with gradient clipping employed to address the challenges associated with sensitivity analysis. Moreover, we have provided proof demonstrating that the proposed approach satisfies ϵ -differential privacy, ensuring a strong privacy guarantee.

To thoroughly evaluate the performance and behavior of our differentially private approach, we conducted comprehensive simulation studies and real-world case studies. In the simulation studies, we meticulously examined the influence of key factors such as training sample size, sensor correlation, and privacy budget on the accuracy of sensor selection. Our findings revealed intricate relationships between these factors and the performance of both non-private and differentially private methods. We observed that increasing the training dataset size generally enhances the performance of the DP method, highlighting the importance of having a sufficiently large and representative dataset to mitigate the impact of privacy-induced noise. However, it is worth noting that the DP approach typically requires a significantly larger training sample compared to its non-private counterpart to achieve comparable accuracy. This underscores the inherent trade-off between privacy and utility, as the noise introduced for privacy protection can degrade performance if not counterbalanced by a larger dataset.

Moreover, our simulation studies shed light on the sensitivity of the DP approach to sensor

correlations. We found that higher correlations between informative and uninformative sensors pose a greater challenge for accurate sensor selection, as the noise introduced by differential privacy can make it more difficult to distinguish between correlated sensors. This emphasizes the need for careful data preprocessing and feature engineering to mitigate the impact of sensor correlations when applying differentially private methods.

The privacy budget, which determines the strength of privacy protection, also emerged as a crucial factor in our analysis. We observed that smaller privacy budgets, corresponding to stronger privacy guarantees, necessitate a higher number of iterations for convergence, leading to increased computational costs. This is an expected consequence of the larger amount of noise added to the gradient, which perturbs the descent direction more significantly. However, it is reassuring to note that given a sufficient number of iterations, the final sensor selection results remained relatively robust to variations in the privacy budget.

Our case study on the real-world dataset of aircraft engine degradation provided valuable insights into the practical application of our differentially private approach. By applying PCA to reduce the dimensionality of the sensor data and considering various LLS regression models under different privacy budgets, we demonstrated the feasibility and effectiveness of our method in a realistic setting. The aggregation of results over multiple runs proved to be a valuable strategy for mitigating the impact of privacy-induced noise and reliably identifying the most informative sensors. This showcases the potential of our approach to be applied in real-world scenarios where privacy is a paramount concern.

The theoretical foundations and empirical evidence presented in this chapter contribute to the growing body of work on privacy-preserving statistical learning, specifically in the context of sensor selection and LLS regression. Our differentially private proximal gradient descent approach offers a valuable tool for researchers and practitioners aiming to build models that balance the competing objectives of data utility and individual privacy. The insights gained from our simulations and case studies can guide the design and implementation of differentially private sensor selection methods in various application domains.

Looking ahead, there are several promising avenues for future research. Further theoretical analysis of the statistical properties and convergence guarantees of our approach could provide additional insights and support its robustness. Extending the methodology to other regression models and penalty functions could broaden its applicability to a wider range of problems. Moreover, exploring strategies for adaptive privacy budget allocation and dynamic noise calibration could help optimize the privacy-utility trade-off in different settings.

In conclusion, our differentially private proximal gradient descent approach for sensor selection in LLS regression with group lasso penalty represents a significant step towards enabling privacy-preserving learning in sensitive domains. By providing a rigorous privacy guarantee and demonstrating its effectiveness through simulations and case studies, we contribute to the ongoing effort to develop statistical learning methods that respect individual privacy while still extracting valuable insights from data. As the demand for privacy-aware data analysis grows across industries and application areas, our work offers a promising direction for future research and practical implementation.

CHAPTER

4

PRIVACY-PRESERVING FEDERATED FUNCTIONAL PRINCIPAL COMPONENT ANALYSIS FOR SPARSE DATA

4.1 Introduction

Industrial prognostics is a critical aspect of modern manufacturing and maintenance processes. It involves predicting the future state of machinery and systems to estimate the Time to Failure (TTF) of equipment or components. By accurately forecasting when a machine is likely to fail, industries can schedule maintenance proactively, thereby reducing unexpected downtimes and optimizing operational efficiency. Effective prognostic models rely on the analysis of

historical data and current condition monitoring information to identify patterns and trends that indicate impending failures (Mobley 2002; Si et al. 2011; Lee et al. 2014). Research has shown that integrating various data sources can significantly enhance the accuracy of prognostic models (Vogl et al. 2019).

The increasing availability of data from sensors and monitoring systems has led to the adoption of advanced analytical methods in industrial prognostics. One such method is Functional Data Analysis (FDA), which is particularly useful for handling data that can be represented as smooth curves or functions. FDA provides tools for analyzing complex and high-dimensional data by focusing on the underlying functional nature of the data (Ramsay and Silverman 2005). This approach is well-suited for capturing the temporal dynamics inherent in industrial monitoring data, which often exhibit continuous and smooth variations over time. Work in the FDA for industrial applications has demonstrated its potential in improving predictive maintenance strategies (Liu et al. 2013; Fang et al. 2017; Yoo and Suh 2022).

A significant challenge in industrial prognostics is the issue of sparse and irregular data. In many industrial settings, collecting data at regular intervals can be costly and logistically challenging, resulting in datasets that are sparse and unevenly spaced (Wang et al. 2008; Loutas et al. 2013). Traditional statistical methods typically assume regular and dense observations, making them unsuitable for such sparse data scenarios (Eker et al. 2016). Sparse data can lead to incomplete and biased models if not properly addressed, highlighting the need for methodologies capable of effectively handling sparsity (Hong et al. 2014; Zhao et al. 2015).

Functional Principal Components Analysis (FPCA) is a powerful statistical tool designed to reduce the dimensionality of functional data. It captures the main modes of variation by transforming high-dimensional functional data into a lower-dimensional space, facilitating easier interpretation and analysis (Ramsay and Silverman 2005; Yao et al. 2005; Hall and Hosseini-Nasab 2006).

FPCA is an extension of traditional Principal Components Analysis (PCA) designed to handle functional data and provides a powerful framework for reducing the dimensionality

of such data while capturing the primary modes of variation (Ramsay and Silverman 2005). In traditional PCA, the goal is to transform a set of possibly correlated variables into a set of uncorrelated principal components (Jolliffe and Cadima 2016). Given a dataset consisting of n observations, each represented as a p -dimensional vector, PCA involves computing the covariance matrix of the data, finding its eigenvalues and eigenvectors, and projecting the original data onto the space spanned by the principal eigenvectors. This results in a lower-dimensional representation of the data that captures the most significant patterns of variation. FPCA, in contrast, deals with functional data where each observation is a function defined over a continuous domain. Instead of vectors, the observations are curves or surfaces. The primary objective of FPCA is to identify a few orthogonal functional principal components that capture the main modes of variation in these functions (Shang 2014).

FPCA offers several advantages over traditional PCA. One of the most significant benefits is its ability to handle data that are inherently continuous and possibly observed at irregular intervals. Traditional PCA requires data to be observed at the same set of discrete points for all observations, which is often impractical for functional data. FPCA, on the other hand, can accommodate functional data observed on different grids or even sparsely observed data (Yao et al. 2005). Moreover, FPCA provides a more natural and accurate representation of functional data. By modeling the data as continuous functions, FPCA preserves the intrinsic properties of the data, such as smoothness and continuity, which are often lost in traditional PCA. This leads to better interpretation and understanding of the underlying patterns in the data (Wang et al. 2016).

Based on the properties of FPCA, we can see that it is naturally important and useful in industrial prognostics, where it is critical to monitor and predict the condition and performance of machinery over time. Industrial prognostics involves analyzing time-series data collected from sensors embedded in machines, which often results in functional data due to continuous monitoring. These data are typically sparse and irregularly spaced, making traditional PCA unsuitable. By applying FPCA, we can effectively reduce the dimensionality of these functional

datasets while preserving essential patterns of variation. The mean function provides insights into the overall health and performance trends of the machinery, while the covariance function and its eigenfunctions reveal the dominant modes of variation and potential anomalies. The functional principal component scores can be used to predict future behavior, identify early signs of failure, and schedule maintenance more efficiently. FPCA's ability to handle irregularly spaced and sparse data makes it invaluable for industrial prognostics. It allows for robust analysis and prediction even when data points are missing or unevenly distributed. This capability ensures that industrial systems remain operational and efficient, reducing downtime and maintenance costs.

The fundamental work by Yao et al. (2005) on FDA for sparse data provides a comprehensive framework for addressing the challenges posed by sparsity. Yao et al. introduced a combination of FPCA and smoothing techniques to extract meaningful patterns from sparsely and irregularly observed functional data. This methodology offers robust techniques for analyzing sparse functional data, which is essential for developing accurate and reliable prognostic models.

Despite the advantages of FPCA, applying it to industrial prognostics presents additional challenges, particularly when dealing with distributed data sources. Industrial data is often generated across various sites and owned by different stakeholders, centralizing this data for FPCA is impractical and poses risks to data privacy and security. For example, the data collected from sensors embedded in machines are often distributed across multiple locations, such as various factories or field sites, and can be highly sensitive due to proprietary information. Federated Learning (FL), a decentralized approach that enables multiple participants to collaboratively train machine learning models while keeping their data localized, has emerged as a promising solution to these issues.

Introduced by the research of McMahan et al. (2017), federated learning addresses the challenges of privacy concerns by enabling the collaborative training of models across decentralized devices or servers while keeping the data localized. Unlike traditional centralized approaches, where data is aggregated on a central server, federated learning allows each node

to compute updates based on its local data and share only model parameters or gradients with a central server. This method ensures that the raw data remains on the local devices, thereby preserving privacy and reducing the risk of data breaches. Federated learning is particularly advantageous in industrial settings, where data is naturally decentralized and sensitive.

To further enhance privacy, differential privacy is integrated into federated learning. Differential privacy (DP), introduced by Dwork et al. (2006), provides strong mathematical guarantees on the privacy of individual data points. It ensures that the output of a data analysis algorithm is insensitive to any single data point. In federated learning, differential privacy mechanisms add carefully calibrated noise to the model updates before they are aggregated by the central server. This perturbation prevents the inference of sensitive information from the shared model parameters or gradients, ensuring robust privacy protection for individual data points.

Both federated learning and differential privacy are essential tools for modern industrial prognostics. The collaborative nature of federated learning allows for the creation of more accurate and robust predictive models by leveraging data from multiple sources. This holistic approach improves the ability to monitor and predict machinery performance, leading to better maintenance scheduling, reduced downtime, and increased operational efficiency. The integration of differential privacy ensures that these benefits are achieved without compromising the confidentiality of the underlying data.

This paper builds on the methodology proposed by Yao et al. (2005) by adapting it to the context of federated learning with differential privacy. Our framework leverages the strengths of FPCA, FL, and DP to enable robust and privacy-preserving industrial prognostics. By integrating these advanced methodologies, we aim to address the unique challenges of sparse data and privacy concerns in distributed industrial environments, providing a comprehensive solution for accurate and secure prognostic modeling.

The rest of this paper is organized as follows. Section 4.2 provides an overview of the functional principal component analysis (FPCA) of sparse data. In Section 4.3, we introduce our federated learning framework for applying FPCA to decentralized data. For each part, we dis-

Discuss the federated approach of each key step in the FPCA method, including the estimation of the covariance function and the computation of principal components using conditional expectation. The federated learning algorithm utilizes the communication between the local workers and the central server to average gradients, aggregate estimates, and make global computations without centralized datasets. We also present the incorporation of differential privacy into the federated learning process, explain the mechanisms for adding noise to the model updates, and discuss the privacy-utility trade-off. In Section 4.4, we conduct a simulation study to evaluate the performance of our proposed approach under various sparsity levels and privacy settings. In Section 4.5, we demonstrate the application of our method to a real-world industrial prognostic case study, showcasing its effectiveness in handling sparse data and preserving privacy. Finally, Section 4.6 concludes the paper, summarizes the main contributions, and discusses potential future research directions.

4.2 Functional Principal Components Analysis

In this section, we summarize the methodology for FPCA on sparse data as outlined in Yao et al. (2005), namely the Principal Component Analysis through Conditional Expectation (PACE) method. The primary goal of FPCA is to reduce the dimensionality of random trajectories by representing them using a few principal components, which capture the dominant modes of variation. The PACE method is designed to perform FPCA for sparse longitudinal data. This approach addresses the challenges posed by having only a few irregularly spaced measurements per subject, which makes traditional FPCA methods impractical. PACE provides a robust framework for estimating individual smooth trajectories and the dominant modes of variation in the data.

We start by modeling the sparse functional data as noisy sampled points from a collection of trajectories, assumed to be independent realizations of a smooth random function with unknown mean function $E[X(t)] = \mu(t)$ and covariance function $\text{cov}(X(s), X(t)) = G(s, t)$. The

domain of $X(\cdot)$ is a bounded and closed time interval \mathcal{T} . The trajectories can be expressed through an orthogonal expansion in terms of eigenfunctions ϕ_k and eigenvalues λ_k :

$$G(s, t) = \sum_{k=1}^{\infty} \lambda_k \phi_k(s) \phi_k(t), \quad s, t \in \mathcal{T}. \quad (4.1)$$

For the i -th random curve, we have:

$$X_i(t) = \mu(t) + \sum_{k=1}^{\infty} \xi_{ik} \phi_k(t), \quad t \in \mathcal{T}, \quad (4.2)$$

where ξ_{ik} are uncorrelated random variables with zero mean and variances $E[\xi_{ik}^2] = \lambda_k$.

To account for measurement errors, we extend the model to include uncorrelated errors with mean zero and constant variance σ^2 . Let Y_{ij} be the j -th observation of the random function $X_i(\cdot)$ made at a random time T_{ij} , with additional measurement errors φ_{ij} :

$$Y_{ij} = X_i(T_{ij}) + \varphi_{ij} = \mu(T_{ij}) + \sum_{k=1}^{\infty} \xi_{ik} \phi_k(T_{ij}) + \varphi_{ij}, \quad T_{ij} \in \mathcal{T}. \quad (4.3)$$

Here, $E(\varphi_{ij}) = 0$, $\text{Var}(\varphi_{ij}) = \sigma^2$, and the number of measurements N_i made on the i -th subject is random, reflecting sparse and irregular designs.

The first step in the PACE method involves estimating the mean function $\mu(t)$ of the underlying random trajectories. This is achieved by applying local linear smoothing. The mean function represents the average trend across all subjects and is crucial for understanding the overall behavior of the data. By fitting local lines to the observed data points and minimizing the weighted least squares, we obtain a smooth estimate of $\mu(t)$ that adapts to the sparsity and irregularity of the measurements. Specifically, the mean function $\mu(t)$ is estimated by solving $\arg \min_{\boldsymbol{\alpha}} \mathcal{S}_{\mu}$, where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1)$, and

$$\mathcal{S}_{\mu}(\boldsymbol{\alpha}) = \sum_{i=1}^n \sum_{j=1}^{N_i} \kappa_1 \left(\frac{T_{ij} - t}{h_{\mu}} \right) \{Y_{ij} - \alpha_0 - \alpha_1(t - T_{ij})\}^2 \quad (4.4)$$

where κ_1 is a kernel function with bandwidth h_μ . The desired estimated mean function $\hat{\mu}(t) = \hat{\alpha}_0(t)$.

Next, we would like to estimate the covariance function $G(s, t)$, which captures the variability and dependence structure of the trajectories around the mean function. To do so, we need first to obtain the raw covariances, which are computed from the data, removing the diagonal elements to account for measurement errors, as follows:

$$G_i(T_{ij}, T_{il}) = (Y_{ij} - \hat{\mu}(T_{ij}))(Y_{il} - \hat{\mu}(T_{il})), \quad j \neq l. \quad (4.5)$$

These raw covariances are then smoothed using a bivariate smoothing technique, resulting in a smooth estimate of the covariance surface. To estimate the covariance function $G(s, t)$ for $s \neq t$, we smooth the raw covariances by solving $(\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2) = \arg \min_{\beta_0, \beta_1, \beta_2} \mathcal{S}_G$, where $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$, and

$$\mathcal{S}_G(\boldsymbol{\beta}) = \sum_{i=1}^n \sum_{1 \leq j \neq l \leq N_i} \kappa_2 \left(\frac{T_{ij} - s}{h_G}, \frac{T_{il} - t}{h_G} \right) \{G_i(T_{ij}, T_{il}) - (\beta_0 + \beta_1(s - T_{ij}) + \beta_2(t - T_{il}))\}^2 \quad (4.6)$$

where κ_2 is a kernel function, h_G is the bandwidth parameter and β_0 , β_1 , and β_2 are the coefficients of the local linear surface approximation. They represent the intercept, the slope along the s -direction, and the slope along the t -direction, respectively. Here, $G_i(T_{ij}, T_{il})$ are the raw covariances, κ_2 is a bivariate kernel function, and h_G is the bandwidth parameter. The estimated $\widehat{G}(s, t)$ is then given by $\widehat{\beta}_0(s, t)$.

To account for measurement errors, the variance σ^2 of the errors is then estimated. This is done by fitting a local quadratic component along the direction orthogonal to the diagonal of the covariance surface and a local linear component along the diagonal. The estimated variance σ^2 reflects the additional noise present in the observations and is subtracted from the diagonal elements of the covariance surface to obtain a more accurate representation of the underlying variability in the data.

To obtain the adjusted estimate of $G(t, t)$ on the diagonal, denoted as $\tilde{G}(t)$, the approach suggested in Yao et al. (2005) is to first rotate both x -axis and y -axis by 45° clockwise and obtain the coordinates of (T_{ij}, T_{ik}) in the rotated axes, denoted by (T_{ij}^*, T_{ik}^*) , i.e.,

$$\begin{pmatrix} T_{ij}^* \\ T_{ik}^* \end{pmatrix} = \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 \\ -\sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix} \begin{pmatrix} T_{ij} \\ T_{ik} \end{pmatrix}. \quad (4.7)$$

The estimate can then be obtained by minimizing the weighted least squares. Let $(\hat{\gamma}_0, \hat{\gamma}_1, \hat{\gamma}_2) = \arg \min_{\boldsymbol{\gamma}} \mathcal{S}_R$, where $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, \gamma_2)$, and

$$\begin{aligned} \mathcal{S}_R(\boldsymbol{\gamma}) = \sum_{i=1}^n \sum_{1 \leq j \neq l \leq N_i} \kappa_2 \left(\frac{T_{ij}^* - s}{h_G}, \frac{T_{il}^* - t}{h_G} \right) \{ G_i(T_{ij}^*, T_{il}^*) - \\ (\gamma_0 + \gamma_1(s - T_{ij}^*) + \gamma_2(t - T_{ik}^*)) \}^2, \end{aligned} \quad (4.8)$$

then we have $\tilde{G}(t, t) = \hat{\gamma}_0(0, t/\sqrt{2})$ obtained with the rotated coordinates.

Denote the diagonal of the resulting surface estimate by $\tilde{G}(t)$, and a local linear smoother focusing on diagonal values $\{G(t, t) + \sigma^2\}$ by $\widehat{V}(t)$, obtained by replacing input with $\{G_i(T_{ij}, T_{ij})\}$ in the univariate kernel smooth function as \mathcal{S}_μ . To mitigate boundary effects, it is also recommended to cut off the two ends of the interval to get a more stable estimate (Staniswalis and Lee 1998). Let $|\mathcal{T}|$ denote the length of \mathcal{T} , and \mathcal{T}_1 be the interval $\mathcal{T}_1 = [\inf\{x : x \in \mathcal{T}\} + |\mathcal{T}|/4, \sup\{x : x \in \mathcal{T}\} - |\mathcal{T}|/4]$. The proposed estimate of σ^2 is

$$\hat{\sigma}^2 = \frac{2}{|\mathcal{T}|} \int_{\mathcal{T}_1} \{\widehat{V}(t) - \tilde{G}(t)\} dt \quad (4.9)$$

if $\hat{\sigma}^2 > 0$ and $\hat{\sigma}^2 = 0$ otherwise.

The eigenfunctions $\phi_k(t)$ and eigenvalues λ_k of the covariance function are then determined by solving the eigen-equations,

$$\int_T \widehat{G}(s, t) \hat{\phi}_k(s) ds = \hat{\lambda}_k \hat{\phi}_k(t), \quad (4.10)$$

subject to the constraints $\int_T \hat{\phi}_k(t)^2 dt = 1$, and $\int_T \hat{\phi}_k(t)\hat{\phi}_m(t)dt = 0$ for $m < k$. The eigenfunctions represent the principal modes of variation in the data, while the eigenvalues quantify the amount of variation explained by each mode. These components are estimated from the smoothed covariance surface and provide a basis for reconstructing individual trajectories. The eigenfunctions and eigenvalues are critical for dimension reduction, allowing the complex random trajectories to be represented using a small number of principal components.

The functional principal component scores ξ_{ik} for each subject are estimated using conditional expectation. This step involves predicting the principal component scores by conditioning on the observed data for each subject, borrowing strength from the entire sample. This approach provides the best linear prediction of the scores, combining information from both the individual subject and the population as a whole. The predicted scores ξ_{ik} are then used to reconstruct the individual trajectories by projecting them onto the estimated eigenfunctions.

Let $\tilde{\mathbf{X}}_i = (X_i(T_{i1}), \dots, X_i(T_{iN_i}))^T$, $\tilde{\mathbf{Y}}_i = (Y_{i1}, \dots, Y_{iN_i})^T$, $\hat{\boldsymbol{\mu}}_i = (\hat{\mu}(T_{i1}), \dots, \hat{\mu}(T_{iN_i}))^T$, and $\hat{\boldsymbol{\phi}}_{ik} = (\hat{\phi}_k(T_{i1}), \dots, \hat{\phi}_k(T_{iN_i}))^T$. Assuming that ξ_{ik} and φ_{ij} are jointly Gaussian, estimates for the FPC scores are obtained by conditional expectation. By substituting estimates of mean function $\hat{\boldsymbol{\mu}}_i$, eigenvalues $\hat{\lambda}_k$ and eigenfunctions $\hat{\boldsymbol{\phi}}_{ik}$, covariance matrix $\hat{\boldsymbol{\Sigma}}_{Y_i}$, the best prediction of ξ_{ik} given the observed data is:

$$\hat{\xi}_{ik} = \widehat{E}[\xi_{ik} | \tilde{\mathbf{Y}}_i] = \hat{\lambda}_k \hat{\boldsymbol{\phi}}_{ik}^T \hat{\boldsymbol{\Sigma}}_{Y_i}^{-1} (\tilde{\mathbf{Y}}_i - \hat{\boldsymbol{\mu}}_i). \quad (4.11)$$

The covariance matrix $\boldsymbol{\Sigma}_{Y_i} = \text{cov}(\tilde{\mathbf{Y}}_i, \tilde{\mathbf{Y}}_i) = \text{cov}(\tilde{\mathbf{X}}_i, \tilde{\mathbf{X}}_i) + \sigma^2 \mathbf{I}_{N_i}$ is a $N_i \times N_i$ matrix. The estimated covariance matrix $\hat{\boldsymbol{\Sigma}}_{Y_i}$ is given by

$$(\hat{\boldsymbol{\Sigma}}_{Y_i})_{j,l} = \widehat{G}(T_{ij}, T_{il}) + \hat{\sigma}^2 \psi_{jl} \quad (4.12)$$

for each (j, l) element, with $\psi_{jl} = 1$ if $j = l$ and $\psi_{jl} = 0$ if $j \neq l$. In other words, adding variance on the diagonal.

Following the assumption that the infinite-dimensional processes under consideration are well approximated by the projection on the function space spanned by the first K eigenfunc-

tions, the predicted trajectory for the i -th subject is given by:

$$\widehat{X}_i^K(t) = \hat{\mu}(t) + \sum_{k=1}^K \hat{\xi}_{ik} \hat{\phi}_k(t), \quad (4.13)$$

where K is the number of principal components retained.

The PACE method proposed by Yao et al. (2005) enables the prediction of individual smooth trajectories even with sparse data, as this method constructs accurate predictions of the underlying trajectories. This is particularly useful in industrial prognostics and other applications where data may be sparse and irregular. The PACE method also provides a framework for constructing pointwise and simultaneous confidence bands for the predicted trajectories, offering a measure of uncertainty in the predictions.

However, applying functional principal components analysis (FPCA) directly to industrial prognostics, particularly with sparse data, presents several significant challenges. Industrial data is often decentralized, and distributed across multiple locations such as different factories or machines, making it impractical to collect all the data in a central repository. FPCA traditionally requires pooling data to estimate the mean and covariance functions, which becomes infeasible in such a decentralized setting.

Moreover, industrial data frequently contains sensitive information related to proprietary processes or equipment conditions. Centralizing this data can lead to privacy concerns and potential legal issues concerning data sharing and security. Ensuring the confidentiality and integrity of the data while performing comprehensive analysis poses a substantial hurdle.

To address these challenges, we propose leveraging federated learning (FL) and differential privacy (DP), which enables model training across multiple decentralized devices or servers while keeping the data localized. This approach ensures data privacy and accommodates decentralized data sources effectively.

4.3 A Federated Learning Approach of FPCA

We propose a federated learning framework to address the challenges of applying FPCA directly to industrial prognostics. This approach ensures data privacy, accommodates decentralized sparse data, and improves knowledge of data insights for each participant.

Federated learning, introduced by McMahan et al. (2017), is a distributed machine learning paradigm that enables training models on decentralized data without the need for data sharing. The key idea is to train local models on each participant's data and then aggregate the model updates on a central server to obtain a global model. This approach has been successfully applied to various machine learning tasks, such as deep learning (Konečn'y et al. 2016) and support vector machines (Yang et al. 2019). One of the most well-known federated learning algorithms demonstrated in McMahan et al. (2017) lets each local worker compute the gradients of the model parameters based on its local data and send these gradients to a central server. The server then aggregates the gradients from all workers and updates the global model parameters. The updated global model is then sent back to the local workers for the next round of gradient computation.

In our federated learning framework for FPCA (FL-FPCA), each local worker computes gradient descent updates for estimating the mean function $\mu(t)$ and the covariance function $G(s, t)$ based on its local data. These local gradients are then sent to a central server, which aggregates them to obtain global gradients. These global gradients were sent to each participant for descending local mean and covariance estimates. The global variance of measurement error is obtained on the server in a similar manner, incorporating specific techniques purposed in Yao et al. (2005). Finally, the central server aggregates the covariance estimates from all workers and solves the global eigenfunctions and eigenvalues. This approach ensures that the eigenfunctions and eigenvalues reflect the overall structure of the data collected from all nodes, providing a comprehensive analysis without compromising data privacy.

In this section, we will go through the detailed federated learning framework of functional principal component analysis for sparse data. We will first introduce the federated algorithm

for estimating each step in the FPCA method, and then explain how we incorporate differential privacy to protect the gradients.

4.3.1 Mean Function Estimate

The first step is to estimate the mean function μ based on all available data. To estimate the mean function, we need to solve $\arg \min_{\boldsymbol{\alpha}} \mathcal{S}_{\mu}$, where $\boldsymbol{\alpha} = (\alpha_0, \alpha_1)$, and

$$\mathcal{S}_{\mu}(\boldsymbol{\alpha}) = \sum_{i=1}^n \sum_{j=1}^{N_i} \kappa_1 \left(\frac{T_{ij} - t}{h_{\mu}} \right) \{Y_{ij} - \alpha_0 - \alpha_1(t - T_{ij})\}^2 \quad (4.14)$$

The objective function \mathcal{S}_{μ} is designed to minimize the discrepancy between the observed data Y_{ij} and a local linear model $\alpha_0 + \alpha_1(t - T_{ij})$, weighted by the kernel function κ_1 . This function emphasizes data points close to the target point t by assigning higher weights to them. And when estimating $\mu(t)$, instead of using all $t \in \mathcal{T}$, we generate a uniform time grid on \mathcal{T} , denoted as $\mathcal{T}_{\text{grid}}$. This $\mathcal{T}_{\text{grid}}$ should span the same interval as \mathcal{T} , but with a limited number of uniformly distributed timestamps. Later, we will interpolate for $t \in \mathcal{T}$ when we need to use some $\hat{\mu}(T_{ij})$ which are not on the timestamps of $\mathcal{T}_{\text{grid}}$. Similar practices will also be used for estimating other model components in the following subsections.

The basic idea of the federated learning algorithm is that each local worker computes gradient updates for the smooth function using local data, and these updates are then aggregated on a central server. The aggregation can be done by averaging, weighted averaging, or summing up. The server then updates a global parameter vector by gradient descent, and sent the parameter back to local workers to calculate local gradient for the next round. The algorithm of the federated learning approach for estimating mean function is shown below in Algorithm 7.

The algorithm operates iteratively, with multiple rounds of local computations and global aggregations. Initially, each local worker m has access to its local dataset $\{(T_{ij}, Y_{ij})\}_{j=1}^{N_{im}}\}_{i=1}^{n_m}$, where i indexes the subjects and j indexes the observations within each subject. A univariate

Algorithm 7 FL-FPCA: Mean Function Estimate

Require: Local datasets $\{(T_{ij}, Y_{ij})\}_{j=1}^{N_{im}}\}_{i=1}^{n_m}$, univariant kernel function $\kappa_1(\cdot)$, bandwidth h_μ , learning rate η_α , local sample size of each worker $\{q_m\}_{m=1}^M$

Ensure: Estimated mean function in vector $\hat{\boldsymbol{\mu}}$

- 1: Initialize $\boldsymbol{\alpha}^{(0)}$
 - 2: **for** all t in $\mathcal{T}_{\text{grid}}$ **do**
 - 3: **for** each round $p = 1, 2, \dots, P$ **do**
 - 4: **for** every worker $m = 1, 2, \dots, M$ **in parallel do**
 - 5: Compute local gradients $\nabla \mathcal{S}_{\mu_m}^{(p)}$ according to (4.15).
 - 6: Send gradients $\nabla \mathcal{S}_{\mu_m}^{(p)}$ to the central server
 - 7: **end for**
 - 8: Aggregate gradients: $\mathbf{U}_\mu^{(p)} = \sum_{m=1}^M \nabla \mathcal{S}_{\mu_m}^{(p)}$
 - 9: Update $\boldsymbol{\alpha} = (\alpha_0, \alpha_1)$ by gradient descent: $\boldsymbol{\alpha}^{(p)} = \boldsymbol{\alpha}^{(p-1)} - \eta_\alpha \mathbf{U}_\mu^{(p)}$
 - 10: Broadcast $\boldsymbol{\alpha}^{(p)}$ to workers
 - 11: **end for**
 - 12: Obtain estimated mean function: $\hat{\boldsymbol{\mu}}(t) = \hat{\boldsymbol{\alpha}}_0(t)$
 - 13: **end for**
 - 14: **Return** vector $\hat{\boldsymbol{\mu}}$
-

kernel function $\kappa_1(\cdot)$ and a bandwidth parameter h_μ are specified, alongside a learning rate η_α for the gradient descent updates.

In each round of the algorithm, every worker computes the local gradients of the objective function \mathcal{S}_{μ_m} with respect to the parameters $\boldsymbol{\alpha} = (\alpha_0, \alpha_1)$. Suppose each worker m holds a local dataset $\{(T_{ij}, Y_{ij})\}_{j=1}^{N_{im}}\}_{i=1}^{n_m}$, where $i = 1, \dots, n_m$, $j = 1, \dots, N_{im}$. The gradient $\nabla \mathcal{S}_{\mu_m} = (\frac{\partial \mathcal{S}_{\mu_m}}{\partial \alpha_0}, \frac{\partial \mathcal{S}_{\mu_m}}{\partial \alpha_1})$ for worker m is given by

$$\begin{aligned} \frac{\partial \mathcal{S}_{\mu_m}}{\partial \alpha_0} &= -2 \sum_{i=1}^{n_m} \sum_{j=1}^{N_{im}} \kappa_1 \left(\frac{T_{ij} - t}{h_\mu} \right) \{Y_{ij} - \alpha_0 - \alpha_1 (t - T_{ij})\} \\ \frac{\partial \mathcal{S}_{\mu_m}}{\partial \alpha_1} &= -2 \sum_{i=1}^{n_m} \sum_{j=1}^{N_{im}} (t - T_{ij}) \kappa_1 \left(\frac{T_{ij} - t}{h_\mu} \right) \{Y_{ij} - \alpha_0 - \alpha_1 (t - T_{ij})\} \end{aligned} \quad (4.15)$$

After computing the local gradients, each worker m sends gradient $\nabla \mathcal{S}_{\mu_m}^{(p)}$ to the central server. The central server computes an aggregated gradient, which represents the combined gradient information from all workers, and then performs a gradient descent of the model parameters on the server. The global parameter $\boldsymbol{\alpha}^{(p)}$ is then broadcasted back to all workers.

Each worker uses the global parameter $\alpha^{(p)}$ to calculate their local gradients $\nabla \mathcal{S}_{\mu_m}^{(p+1)}$ for the next round of communication.

Once the specified number of rounds is completed, the central server holds an estimated intercept $\hat{\alpha}_{0_m}(t)$, which is the desired estimated mean function $\hat{\mu}(t)$.

Notice that this calculation needs to be done for all $t \in \mathcal{T}_{\text{grid}}$, and the server collects results of $\mu(t)$ for all $t \in \mathcal{T}_{\text{grid}}$, and obtains a vector $\hat{\boldsymbol{\mu}}$. When $\hat{\mu}(T_{ij})$ are required later for the following calculation, we interpolate $\hat{\mu}(T_{ij})$ based on obtained $\hat{\boldsymbol{\mu}}$.

4.3.2 Covariance Function Estimate

Following the estimation of the mean function $\mu(t)$, we need to estimate the covariance function $G(s, t)$ next. Notice that in practice, the result of the estimated mean function, $\hat{\boldsymbol{\mu}}$, is a vector, while the result for the estimated covariance function, $\widehat{\mathbf{G}}$, would be a matrix. When estimating G , we also use the uniform time grid on $\mathcal{T}_{\text{grid}}$ instead of using all $t \in \mathcal{T}$. Later, we will compute eigenfunctions and eigenvalues based on this $\widehat{\mathbf{G}}$ on $\mathcal{T}_{\text{grid}}$, and then interpolate eigenfunctions back to \mathcal{T} .

For now, the objective is to estimate the covariance surface, excluding the diagonal elements, by solving the minimization problem $\arg \min_{\boldsymbol{\beta}} \mathcal{S}_G$, where $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$ and the objective function \mathcal{S}_G is defined as:

$$\mathcal{S}_G(\boldsymbol{\beta}) = \sum_{i=1}^n \sum_{1 \leq j \neq l \leq N_i} \kappa_2 \left(\frac{T_{ij} - s}{h_G}, \frac{T_{il} - t}{h_G} \right) \left\{ G_i(T_{ij}, T_{il}) - (\beta_0 + \beta_1(s - T_{ij}) + \beta_2(t - T_{il})) \right\}^2 \quad (4.16)$$

where $G_i(T_{ij}, T_{il}) = (Y_{ij} - \hat{\mu}(T_{ij}))(Y_{il} - \hat{\mu}(T_{il}))$ are the raw covariances, $\kappa_2(\cdot, \cdot)$ is a bivariate kernel function, and h_G is the bandwidth parameter.

The algorithm of the federated learning approach for estimating the covariance surface function is shown in Algorithm 8. Similarly to the mean function estimation, the algorithm begins with local workers calculating raw covariances from their local data and the previously

Algorithm 8 FL-FPCA: Covariance Function Estimate

Require: Local datasets $\{(T_{ij}, Y_{ij})\}_{j=1}^{N_{im}}\}_{i=1}^{n_m}$, raw covariances $G_i(T_{ij}, T_{il}) = (Y_{ij} - \hat{\mu}(T_{ij}))(Y_{il} - \hat{\mu}(T_{il}))$, $j \neq l$ for each worker m , kernel function $\kappa_2(\cdot, \cdot)$, bandwidth h_G , learning rates η_β , local sample size of each worker $\{q_m\}_{m=1}^M$

Ensure: Estimated covariance surface function in matrix $\widehat{\mathbf{G}}$

- 1: Initialize $\boldsymbol{\beta}^{(0)}$
 - 2: **for** each pair of $(s, t) \in \mathcal{T}_{\text{grid}}$ **do**
 - 3: **for** each round $p = 1, 2, \dots, P$ **do**
 - 4: **for** every worker $m = 1, 2, \dots, M$ **in parallel do**
 - 5: Compute local gradients $\nabla \mathcal{S}_{G_m}^{(p)}$ according to (4.17)
 - 6: Send gradient $\nabla \mathcal{S}_{G_m}^{(p)}$ to the central server
 - 7: **end for**
 - 8: Aggregate gradients: $\mathbf{U}_G^{(p)} = \sum_{m=1}^M \nabla \mathcal{S}_{G_m}^{(p)}$
 - 9: Update $\boldsymbol{\beta}_m = (\beta_{0_m}, \beta_{1_m}, \beta_{2_m})$ by gradient descent: $\boldsymbol{\beta}_m^{(p)} = \boldsymbol{\beta}_m^{(p-1)} - \eta_\beta \mathbf{U}_G^{(p)}$
 - 10: Broadcast $\boldsymbol{\beta}_m$ to workers
 - 11: **end for**
 - 12: Obtain estimated covariance surface function $\widehat{G}(s, t) = \widehat{\beta}_0(s, t)$
 - 13: **end for**
 - 14: Convert estimated $\widehat{\mathbf{G}}$ to be a symmetric matrix: $\widehat{\mathbf{G}} \leftarrow \frac{1}{2}(\widehat{\mathbf{G}} + \widehat{\mathbf{G}}^\top)$
 - 15: **Return** matrix $\widehat{\mathbf{G}}$
-

estimated mean function $\hat{\mu}(t)$. A bivariate kernel function $\kappa_2(\cdot, \cdot)$, bandwidth h_G , and learning rate η_β are specified.

Each round, every worker computes the local gradient $\nabla \mathcal{S}_{G_m}^{(p)}$ based on the discrepancy between the observed raw covariances and the model $\beta_0 + \beta_1(s - T_{ij}) + \beta_2(t - T_{il})$. Suppose each worker has a local dataset $\{(T_{ij}, Y_{ij})\}_{j=1}^{N_{im}}\}_{i=1}^{n_m}$, for each local worker m , the gradient $\nabla \mathcal{S}_{G_m} = (\frac{\partial \mathcal{S}_{G_m}}{\partial \beta_0}, \frac{\partial \mathcal{S}_{G_m}}{\partial \beta_1}, \frac{\partial \mathcal{S}_{G_m}}{\partial \beta_2})$ is given by

$$\begin{aligned} \frac{\partial \mathcal{S}_{G_m}}{\partial \beta_0} &= -2 \sum_{i=1}^{n_m} \sum_{1 \leq j \neq l \leq N_{im}} \kappa_2\left(\frac{T_{ij} - s}{h_G}, \frac{T_{il} - t}{h_G}\right) (G_i(T_{ij}, T_{il}) - (\beta_0 + \\ &\quad \beta_1(s - T_{ij}) + \beta_2(t - T_{il}))) \end{aligned} \tag{4.17}$$

$$\begin{aligned} \frac{\partial \mathcal{S}_{G_m}}{\partial \beta_1} &= -2 \sum_{i=1}^{n_m} \sum_{1 \leq j \neq l \leq N_{im}} \kappa_2\left(\frac{T_{ij} - s}{h_G}, \frac{T_{il} - t}{h_G}\right) (s - T_{ij}) (G_i(T_{ij}, T_{il}) - \\ &\quad (\beta_0 + \beta_1(s - T_{ij}) + \beta_2(t - T_{il}))) \end{aligned}$$

$$\frac{\partial \mathcal{S}_{G_m}}{\partial \beta_2} = -2 \sum_{i=1}^{n_m} \sum_{1 \leq j \neq l \leq N_{i_m}} \kappa_2 \left(\frac{T_{ij} - s}{h_G}, \frac{T_{il} - t}{h_G} \right) (t - T_{ij}) (G_i(T_{ij}, T_{il}) - (\beta_0 + \beta_1 (s - T_{ij}) + \beta_2 (t - T_{il})))$$

These local gradients are then sent to the central server, which aggregates them and obtains a global gradient. The server then updates the global parameter $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2)$ through gradient descent and the parameter $\boldsymbol{\beta}^{(p)}$ is broadcasted back to all workers, who use them to calculate gradients for the next round.

After completing the specified number of rounds, the server has an estimated intercept $\hat{\beta}_0(s, t)$, which gives the desired estimated covariance surface $\widehat{G}(s, t) = \hat{\beta}_0(s, t)$. Such computation should be repeated for each pair of (s, t) for $s, t \in \mathcal{T}_{\text{grid}}$, and the server will collect the result \widehat{G} as a matrix. Notice that since each element in this matrix is estimated independently, to make it a covariance matrix, we need to transform it to a symmetric matrix, as mentioned in line 14 in algorithm 8.

4.3.3 Measurement Error Variance Estimate

After estimating the mean function and the covariance function, the next step in the PACE method is to estimate the measurement error variance σ^2 . This involves two main tasks: estimating the covariance surface function on the diagonal and obtaining a local linear smoother focusing on the diagonal values. The federated learning approach distributes these computations across local workers, and obtains a final estimate of σ^2 on the server. The complete procedures for estimating σ^2 are shown in Algorithm 9.

To estimate the covariance surface function on the diagonal, we solve the optimization problem $\arg \min_{\boldsymbol{\gamma}} \mathcal{S}_R$, where $\boldsymbol{\gamma} = (\gamma_0, \gamma_1, \gamma_2)$. The objective function \mathcal{S}_R is defined as

$$\mathcal{S}_R(\boldsymbol{\gamma}) = \sum_{i=1}^n \sum_{1 \leq j \neq l \leq N_i} \kappa_2 \left(\frac{T_{ij}^* - s}{h_R}, \frac{T_{il}^* - t}{h_R} \right) \left\{ G_i(T_{ij}^*, T_{il}^*) - (\gamma_0 + \gamma_1 (s - T_{ij}^*) + \gamma_2 (t - T_{il}^*)) \right\}^2 \quad (4.22)$$

Algorithm 9 FL-FPCA: Measurement Error Variance Estiamte

Require: Rotated local data $\{(T_{ij}^*, Y_{ij}^*)\}_{j=1}^{N_{im}}\}_{i=1}^{n_m}$, rotated raw covariances $G_i(T_{ij}^*, T_{il}^*) = (Y_{ij}^* - \hat{\mu}(T_{ij}^*))(Y_{il}^* - \hat{\mu}(T_{il}^*))$, $j \neq l$ for each worker m , local datasets $\{(T_{ij}, Y_{ij})\}_{j=1}^{N_{im}}\}_{i=1}^{n_m}$, raw covariances $G_i(T_{ij}, T_{il}) = (Y_{ij} - \hat{\mu}(T_{ij}))(Y_{il} - \hat{\mu}(T_{il}))$, $j \neq l$ for each worker m , kernel functions $\kappa_2(\cdot, \cdot)$ and $\kappa_1(\cdot)$, bandwidths h_R, h_V , learning rates η_γ and η_δ , sample size of each worker $\{q_m\}_{m=1}^M$

Ensure: Estimated measurement error variance $\hat{\sigma}^2$

- 1: Initialize $\boldsymbol{\gamma}^{(0)}$ and $\boldsymbol{\delta}^{(0)}$
- 2: **for** each $t \in \mathcal{T}_{\text{grid}}$ **do**
- 3: **for** each round $p = 1, 2, \dots, P$ **do**
- 4: **for** every worker $m = 1, 2, \dots, M$ **in parallel do**
- 5: Compute local gradients $\nabla \mathcal{S}_{R_m}^{(p)}$ and $\nabla \mathcal{S}_{V_m}^{(p)}$ according to equations (4.23) and (4.25), respectively.
- 6: Send gradients $\nabla \mathcal{S}_{R_m}^{(p)}$ and $\nabla \mathcal{S}_{V_m}^{(p)}$ to the central server
- 7: **end for**
- 8: Aggregate gradients:

$$\mathbf{U}_R^{(p)} = \sum_{m=1}^M \nabla \mathcal{S}_{R_m}^{(p)}, \quad \mathbf{U}_V^{(p)} = \sum_{m=1}^M \nabla \mathcal{S}_{V_m}^{(p)} \quad (4.18)$$

- 9: Update $\boldsymbol{\gamma}_m = (\gamma_{0_m}, \gamma_{1_m}, \gamma_{2_m})$ and $\boldsymbol{\delta}_i = (\delta_{0_m}, \delta_{1_m})$ by gradient descent:

$$\boldsymbol{\gamma}_m^{(p)} = \boldsymbol{\gamma}_m^{(p-1)} - \eta_\gamma \mathbf{U}_R^{(p)}, \quad \boldsymbol{\delta}_m^{(p)} = \boldsymbol{\delta}_m^{(p-1)} - \eta_\delta \mathbf{U}_V^{(p)} \quad (4.19)$$

- 10: Broadcast $\boldsymbol{\gamma}^{(p)}$ and $\boldsymbol{\delta}^{(p)}$ to workers
- 11: **end for**
- 12: obtain the estimate functions

$$\tilde{G}(t) = \hat{\gamma}_{0_m}(0, \frac{t}{\sqrt{2}}), \quad \widehat{V}(t) = \hat{\delta}_{0_m}(t) \quad (4.20)$$

- 13: **end for**
- 14: Compute variance:

$$\hat{\sigma}^2 = \max\left(0, \frac{2}{|\mathcal{T}|} \sum_{t \in (\mathcal{T}_{\text{grid}} \cap \mathcal{T}_1)} \{\widehat{V}(t) - \tilde{G}(t)\} \Delta(t)\right) \quad (4.21)$$

where $|\mathcal{T}| = \sup(\mathcal{T}) - \inf(\mathcal{T})$, and $\mathcal{T}_1 = [\inf(\mathcal{T}) + 0.25|\mathcal{T}|, \sup(\mathcal{T}) - 0.25|\mathcal{T}|]$, $\Delta(t)$ is the length of the interval between t 's in $\mathcal{T}_{\text{grid}}$.

- 15: **return** $\hat{\sigma}^2$
-

where T_{ij}^* and T_{il}^* are rotated time points, and $G_i(T_{ij}^*, T_{il}^*)$ are the raw covariances. Similar to previous smooth functions, κ_2 is a bivariate kernel function, while h_R is the bandwidth.

For each worker m , the gradient $\nabla \mathcal{S}_{R_m} = (\frac{\partial \mathcal{S}_{R_m}}{\partial \gamma_0}, \frac{\partial \mathcal{S}_{R_m}}{\partial \gamma_1}, \frac{\partial \mathcal{S}_{R_m}}{\partial \gamma_2})$ is given by

$$\begin{aligned}
\frac{\partial \mathcal{S}_{R_m}}{\partial \gamma_0} &= -2 \sum_{i=1}^{n_m} \sum_{1 \leq j \neq l \leq N_{im}} \kappa_2 \left(\frac{T_{ij}^* - s}{h_R}, \frac{T_{il}^* - t}{h_R} \right) \left\{ G_i(T_{ij}^*, T_{il}^*) - (\gamma_0 + \right. \\
&\quad \left. \gamma_1 (s - T_{ij}^*) + \gamma_2 (t - T_{ik}^*)^2) \right\} \\
\frac{\partial \mathcal{S}_{R_m}}{\partial \gamma_1} &= -2 \sum_{i=1}^{n_m} \sum_{1 \leq j \neq l \leq N_{im}} \kappa_2 \left(\frac{T_{ij}^* - s}{h_R}, \frac{T_{il}^* - t}{h_R} \right) (s - T_{ij}^*) \left\{ G_i(T_{ij}^*, T_{il}^*) - \right. \\
&\quad \left. (\gamma_0 + \gamma_1 (s - T_{ij}^*) + \gamma_2 (t - T_{ik}^*)^2) \right\} \\
\frac{\partial \mathcal{S}_{R_m}}{\partial \gamma_2} &= -2 \sum_{i=1}^{n_m} \sum_{1 \leq j \neq l \leq N_{im}} \kappa_2 \left(\frac{T_{ij}^* - s}{h_R}, \frac{T_{il}^* - t}{h_R} \right) (t - T_{ik}^*)^2 \left\{ G_i(T_{ij}^*, T_{il}^*) - \right. \\
&\quad \left. (\gamma_0 + \gamma_1 (s - T_{ij}^*) + \gamma_2 (t - T_{ik}^*)^2) \right\}
\end{aligned} \tag{4.23}$$

where each local worker m rotates their local dataset $\{(T_{ij}^*, Y_{ij}^*)\}_{j=1}^{N_{im}}\}_{i=1}^{n_m}$, computed according to Equation (4.7).

Similarly to the previous steps, each worker computes local gradients $\nabla \mathcal{S}_{R_m}^{(p)}$ for the diagonal covariance estimation and sends these gradients to the central server. The central server aggregates these gradients for a global one, and performs gradient descent on the global parameters. These parameters, $\boldsymbol{\gamma}^{(p)}$, are then broadcasted to all workers. Each worker calculates a new gradient using these parameters and their local data.

In parallel, we need to obtain a local linear smoother focusing on the diagonal values $\{G(t, t) + \sigma^2\}$ by $\widehat{V}(t)$. This involves solving $\arg \min_{\boldsymbol{\delta}} \mathcal{S}_V$, where $\boldsymbol{\delta} = (\delta_0, \delta_1)$ and the objective function \mathcal{S}_V can be defined as:

$$\mathcal{S}_V(\boldsymbol{\delta}) = \sum_{i=1}^n \sum_{j=1}^{N_i} \kappa_1 \left(\frac{T_{ij} - t}{h_V} \right) \left\{ \{G_i(T_{ij}, T_{ij})\} - \delta_0 - \delta_1 (t - T_{ij}) \right\}^2 \tag{4.24}$$

where $G_i(T_{ij}, T_{ij})$ are the diagonal values of the raw covariances.

Each worker also computes local gradients for the diagonal smoothing. For each worker m ,

the gradient $\nabla \mathcal{S}_{V_m} = (\frac{\partial \mathcal{S}_{V_m}}{\partial \delta_0}, \frac{\partial \mathcal{S}_{V_m}}{\partial \delta_1})$ is given by

$$\begin{aligned}\frac{\partial \mathcal{S}_{V_m}}{\partial \delta_0} &= -2 \sum_{i=1}^{n_m} \sum_{j=1}^{N_{im}} \kappa_1 \left(\frac{T_{ij} - t}{h_V} \right) \{ \{ G_i(T_{ij}, T_{ij}) \} - \delta_0 - \delta_1 (t - T_{ij}) \} \\ \frac{\partial \mathcal{S}_{V_m}}{\partial \delta_1} &= -2 \sum_{i=1}^{n_m} \sum_{j=1}^{N_{im}} (t - T_{ij}) \kappa_1 \left(\frac{T_{ij} - t}{h_V} \right) \{ \{ G_i(T_{ij}, T_{ij}) \} - \delta_0 - \delta_1 (t - T_{ij}) \}\end{aligned}\tag{4.25}$$

where each local worker m holds local dataset $\{ \{ (T_{ij}, Y_{ij}) \}_{j=1}^{N_{im}} \}_{i=1}^{n_m}$.

Then, each local worker m sends their gradients $\nabla \mathcal{S}_{V_m}^{(p)}$ to the central server, the server updates the global parameters $\boldsymbol{\delta} = (\delta_0, \delta_1)$ using the aggregated global gradients, and sends the global parameter $\boldsymbol{\delta}^{(p)}$ back to workers, for them to calculate gradients for the $(p + 1)$ round.

After completing the specified number of rounds, the central server has estimated intercepts $\hat{\gamma}_0(0, \frac{t}{\sqrt{2}})$ and $\hat{\delta}_0(t)$, which are how the server obtains global $\tilde{G}(t)$ and $\widehat{V}(t)$, respectively. Finally, the measurement error variance σ^2 is computed on the server by integrating the difference between $\widehat{V}(t)$ and $\tilde{G}(t)$ over a trimmed interval \mathcal{T}_1 to mitigate boundary effects, as shown in Equation (4.9). Notice that the t 's are discrete in practice, so we instead sum over the differences between $\widehat{V}(t)$ and $\tilde{G}(t)$ for all $t \in (\mathcal{T}_{\text{grid}} \cap \mathcal{T}_1)$, as shown in Equation (4.21) of Algorithm 9.

4.3.4 Late Stages of FL-FPCA

Following the estimation of the mean function, covariance function, and measurement error variance, the next step involves computing the eigenfunctions and eigenvalues of the estimated covariance function $\widehat{G}(s, t)$. Once the covariance surface $\widehat{G}(s, t)$ has been estimated and the central server aggregates covariance estimates from all nodes, the central server then solves the eigen-equations to obtain the global eigenfunctions $\hat{\phi}_k(t)$ and eigenvalues $\hat{\lambda}_k$ directly. This process involves solving the integral equations (4.10), where k indexes the principal components. The eigenfunctions $\hat{\phi}_k(t)$ represent the principal modes of variation, and the eigenvalues $\hat{\lambda}_k$ indicate the amount of variance explained by each mode.

The server also computes the PCA scores ξ_{ik} for each observation i by projecting the esti-

mated mean function-subtracted data onto the eigenfunctions. The PCA scores are calculated as (4.11). These FPC scores ξ_{ik} represent the coordinates of the local data in the reduced-dimensional space spanned by the eigenfunctions. These scores summarize the projection of the data onto the principal component space and are critical for reconstructing the functional data.

After calculating the eigenfunctions, eigenvalues, and PCA scores, the central server broadcasts these components to all workers. Each worker then uses the received eigenfunctions and PCA scores to perform the local projection. Each worker can reconstruct its functional data using the estimated mean function, eigenfunctions, and FPC scores. The reconstructed data for the i -th observation on a local worker is given by (4.13), where K is the number of retained principal components. The value of K could be determined by generalized cross-validation, or simply set as a fixed small value according to local worker's preference. In the common practice of FPCA, a small value of K , e.g., 2 or 3, performs well in most cases.

In summary, the central server performs eigenanalysis on the estimated covariance surface to obtain the global eigenfunctions and eigenvalues and computes the PCA scores. These components are then used by the workers to perform FPCA on their local data, project and reconstructing the functional data. This federated approach ensures that each worker obtains the global model components, eigenfunctions and eigenvalues while keeping its local data private. The server's role in computing the eigenfunctions and eigenvalues from the aggregated covariance function ensures that the principal components capture the overall structure of the data across all workers.

While this federated learning framework itself is a privacy-preserving approach, allowing workers share only gradients but not data points, there are also certain concerns about information leakage through the shared gradients. To further address this issue, we next present an enhanced protection layer for the gradients using differential privacy.

4.3.5 Differentially Private FL-FPCA

Let ω denotes the variable for smooth function \mathcal{S} . We have $\dim(\omega) = 2$ for smooth function with univariant kernel and $\dim(\omega) = 3$ for smooth function with bivariant kernel. Desired estimate functions given by $f(t) = \omega_0(t)$. We present the differentially private gradient update for the federated learning framework of FPCA in Algorithm 10.

The approach for adding DP noises to gradients shown in Algorithm 10 can be integrated into Algorithm 7, Algorithm 8, and Algorithm 9. Specifically, line 4 and line 7 in Algorithm 10 correspond to line 4 and line 5 in all three previous algorithms, respectively. To integrate DP, one can simply insert lines 5 and 6, gradient clipping and gradient perturbation, before sending gradients to central server in Algorithms 7, 8, and 9.

Algorithm 10 Differentially Private Gradient Update for FL-FPCA

Require: privacy budgets ϵ_m , clipping threshold C_{ω_m}

Ensure: parameter vector ω

1: Initialize $\omega^{(0)}$

2: **for** each round $p = 1, 2, \dots, P$ **do**

3: **for** every worker $m = 1, 2, \dots, M$ **in parallel do**

4: Compute gradient $\nabla \mathcal{S}(\omega)^{(p)}$ depends on the specific functions $S(\omega)$

5: Clip gradient:

$$\nabla \mathcal{S}(\omega)^{(p)} = \nabla \mathcal{S}(\omega)^{(p)} / \max\left(1, \frac{\|\nabla \mathcal{S}(\omega)^{(p)}\|_2}{C_{\omega_m}}\right)$$

6: Add Laplace noise to the gradient:

$$\nabla \mathcal{S}(\omega)^{(p)} = \nabla \mathcal{S}(\omega)^{(p)} + \rho_m$$

where $\dim(\rho_m) = \dim(\omega)$, and $\rho_{0_m}, \rho_{1_m}, (\rho_{2_m})$ i.i.d $\sim \text{Laplace}(0, 2C_{\omega_m} \sqrt{\dim(\omega)}/\epsilon_m)$

7: Send DP protected gradients $\nabla \mathcal{S}(\omega)^{(p)}$ to central server

8: **end for**

9: Aggregate gradients: $\mathbf{U}^{(p)} = \sum_{m=1}^M \nabla \mathcal{S}(\omega)^{(p)}$

10: Perform gradient descent: $\omega_m^{(p)} = \omega_m^{(p-1)} - \eta_\omega \mathbf{U}^{(p)}$

11: Broadcast $\omega^{(p)}$ to workers

12: **end for**

In each round of the federated learning process, each worker m computes the gradient

$\nabla \mathcal{S}(\boldsymbol{\omega})^{(p)}$ of the specific function $\mathcal{S}(\boldsymbol{\omega})$ based on their local data. To ensure differential privacy, the gradients are clipped using the clipping threshold C_{ω_m} . Gradient clipping is a technique that limits the magnitude of the gradients to a predefined threshold. If the L2 norm of the gradient exceeds the threshold, it is scaled down to have a norm equal to the threshold. This step helps to bound the sensitivity of the gradient and prevents individual data points from having a disproportionate impact on the model.

Laplace noise is then added to the clipped gradients to achieve differential privacy. The noise is drawn independently from the Laplace distribution with mean 0 and scale parameter $2C_{\omega_m} \sqrt{\dim(\boldsymbol{\omega})} / \epsilon_m$, where $\dim(\boldsymbol{\omega})$ equivalent to the dimension of the gradient and ϵ_m is the privacy budget allocated to worker m . The addition of Laplace noise provides the necessary privacy protection by obscuring the individual contributions to the gradients.

This DP algorithm simply concerns about the gradients. After adding DP noises to the gradient, the rest steps in the previous algorithms should be followed, i.e., each worker sends their differentially private gradients $\nabla \mathcal{S}(\boldsymbol{\omega})^{(p)}$ to the central server, which aggregates the received gradients and gets a global gradient. The aggregated gradient $\mathbf{U}^{(p)}$ is then used to perform gradient descent of the model parameter on the server. The parameter $\boldsymbol{\omega}^{(p)}$ is then broadcasted back to all the workers for the next round of updates. The algorithm repeats these steps for a predefined number of rounds or until convergence is reached.

The privacy budget ϵ_m controls the trade-off between privacy and utility. A smaller value of ϵ_m provides stronger privacy protection but may result in higher noise added to the gradients, potentially affecting the accuracy of the learned model. The choice of ϵ_m depends on the desired level of privacy and the sensitivity of the data. Each worker can decide their own privacy budget, depending on their privacy desires. However, the added noise on gradients will disturb the decent direction, and aggregate such gradients will gather such distortion. Therefore, the workers participating in the federated learning should follow a mutual policy to guide the decision of privacy budget ϵ , in order to achieve a mutual benefit from the aggregated results.

4.4 Simulation Study

We would like to conduct numerical studies to evaluate the performance of the proposed federated learning framework for functional principal component analysis (FL-FPCA), compared with traditional FPCA models. We first demonstrate a simulation study, and we will also explore a real-life case study in the next section.

4.4.1 Experimental Settings

We design a comprehensive simulation study to evaluate the performance of FL-FPCA in industrial prognostics. The study mimics the characteristics of real-world industrial prognostics data, which is often sparse, irregular, and distributed across multiple entities.

Consider a scenario with M workers, each worker m having q_m samples. Each data sample contains degradation signals and Time-to-Failure (TTF) data for a system, representing a machine or component in an industrial setting. The simulated dataset comprises a total of $\sum_{m=1}^M q_m$ subjects in the training dataset.

To mimic the distributed nature of real-world prognostic data, we assume that the data is collected by different workers, each assigned a subset of the training data. This setup reflects the scenario where prognostic data is collected and stored by various entities, such as factories, manufacturing lines, or sensors. In our study, we generate data for 200 systems and distribute them among three workers: worker 1 gets 20 systems, worker 2 gets 30 systems, and worker 3 gets 50 systems. The remaining 100 systems are designated as testing data.

The degradation trajectory of each system i is modeled using the equation $s_i(t) = -\frac{b_i}{\ln(t)}$, where b_i is a random variable drawn from a normal distribution $N(1, 0.25)$, with $0 \leq t < 1$ and $i = 1, \dots, 100$. The time to failure (TTF) for each system is determined as the first instance when the degradation trajectory $s_i(t)$ reaches or exceeds a predefined threshold $F = 2$. This condition is expressed mathematically as $\ln(\tilde{y}_i) = -\frac{b_i}{F}$, where \tilde{y}_i denotes the TTF of system i .

To account for data acquisition errors, we introduce noise to the true TTFs. The observed

TTFs are computed as $\ln(\tilde{y}_i) = -\frac{b_i}{F} + \zeta_i$, where ζ_i is a random variable following a normal distribution $N(0, 0.025)$. This noise component reflects the uncertainties and inaccuracies that may occur during the data collection process in real-world settings.

The observed degradation signals, which represent noisy discrete observations of the underlying degradation trajectories, are generated for each system i using the equation $x(\tau_i) = -\frac{b_i}{\ln(\tau_i)} + v(\tau_i)$, where $v(\tau_i)$ is a random variable following a normal distribution $N(0, 0.2)$, representing the random observation noise. Figure 4.1 illustrates the degradation signal data for systems 1 to 15, providing a visual representation of the simulated data.

By generating data in this manner, we aim to create a realistic simulation of industrial prognostics data that captures the distributed nature of data collection, the presence of noise and uncertainties, and the varying degradation trajectories and failure times of different systems. This simulated dataset will serve as the basis for evaluating the performance of our proposed methods and algorithms in the context of industrial prognostics.

We use FL-FPCA to predict the trajectories, while using standalone FPCA on centralized data and local FPCA on individual users as benchmarks. For FPCA on centralized data, we assume that the FPCA is conducted on a central dataset containing all local datasets from all workers, and we will refer to it as Non-FL FPCA. For local FPCA on individual users, we assume each user performs FPCA on their local data only, without participating in the collaborative federated learning. Moreover, we also evaluate DP-FL FPCA, in which we integrate differentially private gradient updates for each gradient descent algorithms in FL-FPCA.

The FL-FPCA model is trained using the federated learning approach described in the previous section, and we use the Gaussian kernel for smooth functions. We evaluate the performance of FL-FPCA at two different sparsity levels, by sampling upon different percentages of missing data, at 20% and 70%. We use the same fixed testing data contains 100 systems for all models, and the evaluation metric is the error vector $|(\tilde{\mathbf{Y}} - \mathbf{Y})/\mathbf{Y}|$. The reported error results are aggregated for 5 repetitions, each repetition uses a random seed aligned with the repetition number in assigning random subsets of data to workers.

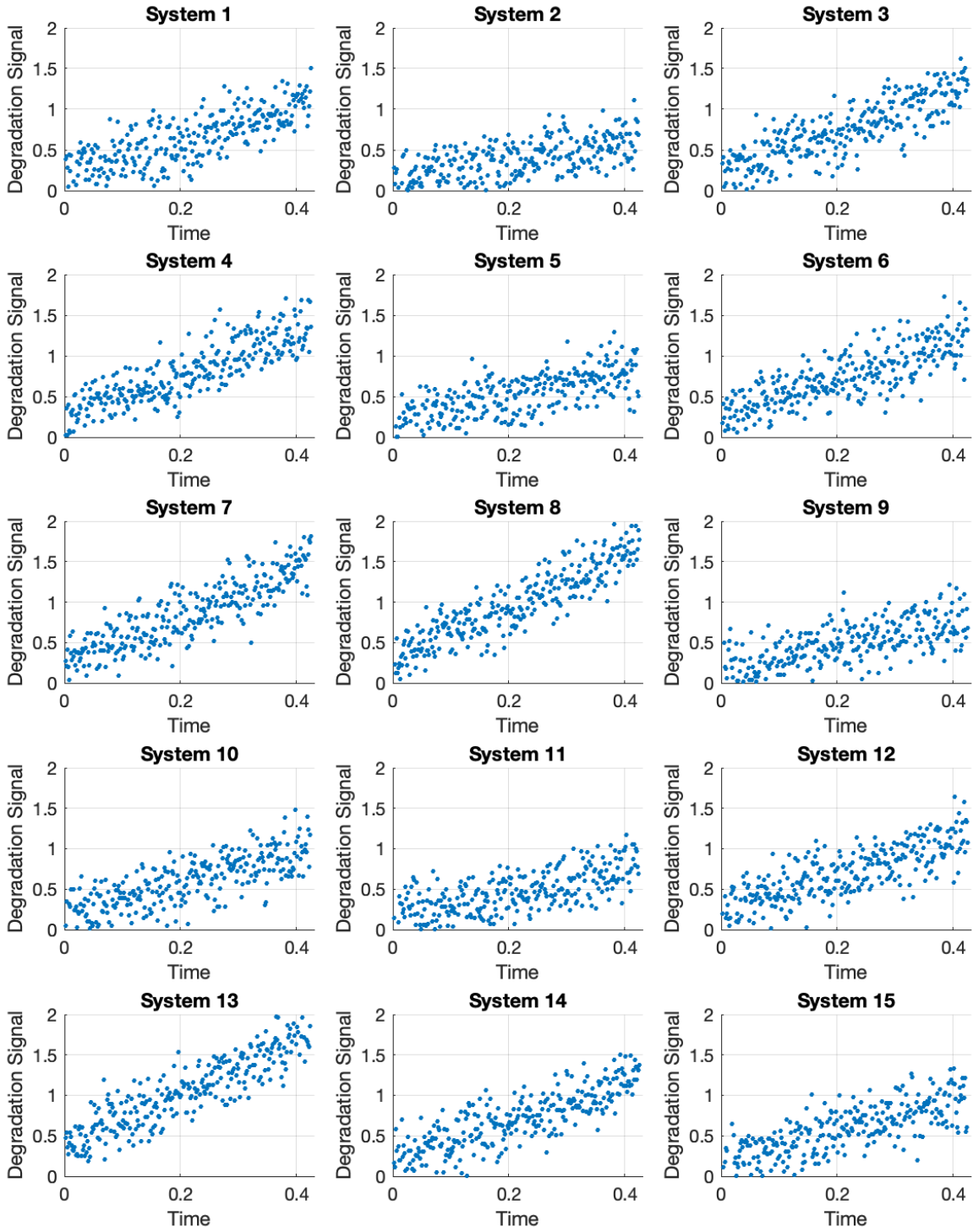


Figure 4.1: Example of the degradation data, $F = 2$, truncated at largest mutual t .

4.4.2 Results and Analysis

We report results at two different sparsity levels. The first set of experiments has 20% missing data, and the second set has 70% missing data. For each group of experiments, the results are reported in boxplots, as shown in Figures 4.2 and 4.3.

In these figures, the left three boxplots used local FPCA on individual worker’s local data. Recall that the first worker has 20 systems, the second worker has 30 systems, and the third worker has 50 systems. In other words, the model for “Worker 1” was trained using only the local $q_1 = 20$ systems, so were the others. The boxplot labeled with “Non-FL” represents the results obtained from FPCA on centralized data, where all $\sum_{m=1}^3 q_m = 100$ training data was used, combined from all three workers. The FL-FPCA model, on the other hand, has all 3 workers participate in the training process, each hold their decentralized data, and communicating with the central server on gradients and model parameters only.

Moreover, we also report DP-FL FPCA results. These models added differential privacy noise on gradients for each stage of FL-FPCA gradient descent, as stated in Algorithm 10. To evaluate the privacy guarantee brought by the DP approach, we evaluated DP-FL models at three different privacy budget levels, $\epsilon = 1, 0.7,$ and 0.5 .

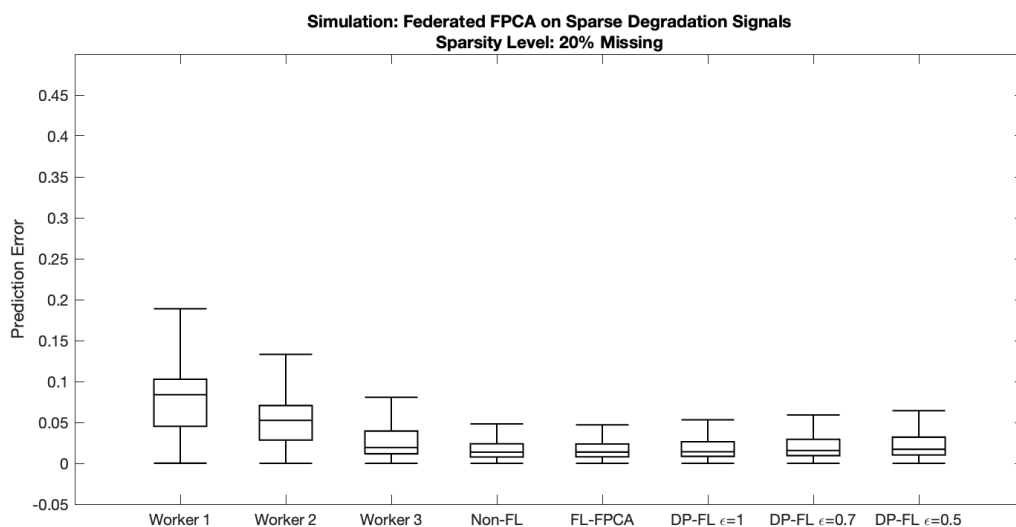


Figure 4.2: Simulation: Error of FPCA Models (Sparsity Level: 20% Missing)

From Figures 4.2 and 4.3, we can easily tell that the performance of FL-FPCA exceeds all individual FPCA on local data, and is close enough to non-FL FPCA with centralized data. Meanwhile, adding DP noises to gradients in FL-FPCA introduces larger error to the model prediction results.

Take Figure 4.2 with 20% missing values as example, Worker 1 which has only 20 systems has median error (and IQR) at 0.084(0.057), while worker 2 with 30 systems has 0.052(0.042). Worker 3 with the largest local dataset has a median error (and IQR) of 0.019(0.028), which is the smallest error among all three local models.

Table 4.1: Simulation: Boxplot Readings (Sparsity Level: 20% Missing)

Sparsity Level: 20% Missing				
Error	Q1	Median	Q3	IQR
Worker 1	0.045	0.084	0.103	0.057
Worker 2	0.028	0.052	0.071	0.042
Worker 3	0.012	0.019	0.039	0.028
Non-FL	0.008	0.014	0.024	0.016
FL-FPCA	0.008	0.014	0.024	0.016
DP-FL $\epsilon = 1$	0.009	0.015	0.026	0.018
DP-FL $\epsilon = 0.7$	0.009	0.016	0.029	0.020
DP-FL $\epsilon = 0.5$	0.010	0.017	0.032	0.022

Non-FL model with centralized data, as expected, has the lowest error, with the median (and IQR) at 0.014(0.016). The proposed FL-FPCA model is also at 0.014(0.016). The difference between the median and IQR of prediction errors of the FL-FPCA model and the non-FL model with centralized data is less than 0.0001, which is very close and suggests the two models converge to the same level of performance. This is reasonable as the FL-FPCA model leverages the collective knowledge from all workers without directly sharing the raw data. By aggregating the local updates from each worker, the FL-FPCA model can capture the underlying patterns and variations in the data, resulting in a performance comparable to the centralized data model.

For the DP-FL models shown on the three boxplot on the right side, we notice that they yield a slightly larger error than non-DP federated learning and centralized data models, while the error increases as the privacy budget ϵ decreases. When $\epsilon = 1$, the median error (and IQR) is 0.015(0.018), while it is 0.016(0.020) at $\epsilon = 0.7$, And when ϵ further decreases to 0.5, the median error increases to 0.017(0.022). These results demonstrate that these DP-FL models provide decent performance compared to FL-FPCA, and show improvement than local models.

A smaller privacy budget ϵ implies stronger privacy protection but also more noise added to the gradients. The noise across multiple stages of gradient descent can lead to a larger overall error in the final model. When the privacy budget is very small, the added noise can significantly impact the model's performance, resulting in higher errors. This highlights the trade-off between privacy and utility in differentially private learning: stronger privacy guarantees come at the cost of reduced model accuracy.

To achieve a closer performance to Non-DP FL-FPCA model, we also noticed that DP-FL models, in general, require more communication rounds, along with smaller learning rates. In other words, this also suggests a trade-off between privacy and computational cost. To conduct real-world applications using the DP-FL FPCA approach, the model should be carefully tuned according to the privacy demands of participants.

Meanwhile, we can observe there are differences between the results of sparsity levels at 20% missing values and 70% missing values. When the data contains more missing values, individual models of local workers show much larger errors than FPCA on centralized data and FL-FPCA collaborated by all workers. For example, at 70% missing level, the median error (and IQR) of worker 1 is 0.108(0.099), while the same worker at 20% missing level has a median error at 0.084(0.057). The non-FL model with centralized data and FL-FPCA model both show comparable performance 0.16(0.20) with the previous results from the lower sparsity level 0.14(0.16). This implies that the sparser and smaller the local dataset a worker has, the greater a worker could benefit from joining federated learning.

We also notice that the DP-FL models perform worse than the models in the same privacy

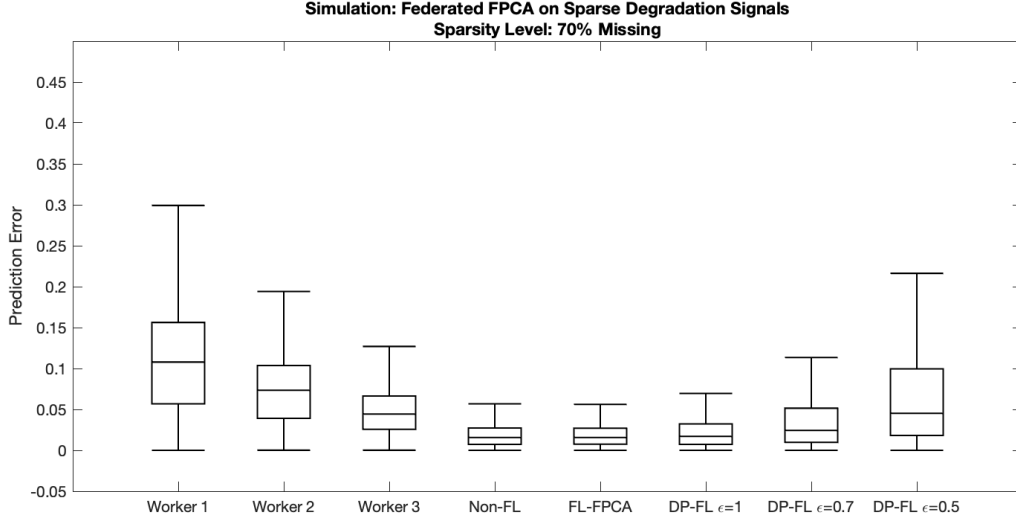


Figure 4.3: Simulation: Error of FPCA Models (Sparsity Level: 70% Missing)

budget as the experiments at lower sparsity level. The median (and IQR) of the prediction error for DP-FL models at $\epsilon = 1$ is 0.017(0.025), and increases to 0.024(0.042) at $\epsilon = 0.7$. When the privacy budget decreases to $\epsilon = 0.5$, the median of the DP-FL model increases to 0.045(0.081), which is slightly higher than the local model of Worker 3 with a median of 0.044(0.041). Therefore, to provide motivation for workers with relatively larger datasets to participate in federated learning, a sustainable privacy budget should be established.

Table 4.2: Simulation: Boxplot Readings (Sparsity Level: 70% Missing)

Sparsity Level: 70% Missing				
Error	Q1	Median	Q3	IQR
Worker 1	0.057	0.108	0.156	0.099
Worker 2	0.039	0.073	0.104	0.064
Worker 3	0.026	0.044	0.066	0.041
Non-FL	0.007	0.016	0.027	0.020
FL-FPCA	0.008	0.016	0.027	0.020
DP-FL $\epsilon = 1$	0.007	0.017	0.032	0.025
DP-FL $\epsilon = 0.7$	0.010	0.024	0.051	0.042
DP-FL $\epsilon = 0.5$	0.018	0.045	0.100	0.081

The simulation study demonstrates the effectiveness of the proposed FL-FPCA framework in handling sparse degradation signal prognostic data while preserving data privacy. The results show that the performance of FL-FPCA is comparable to that of centralized data FPCA, and individuals with limited local data can greatly benefit from joining federated learning. The study also highlights the trade-off between privacy and utility when using differentially private gradient updates in FL-FPCA. As the privacy budget decreases, the model's performance may degrade due to the added noise, but stronger privacy guarantees are achieved. Overall, the FL-FPCA framework provides a promising solution for collaborative learning in industrial prognostics while maintaining data confidentiality.

4.5 Case Study

We will then demonstrate the performance of our proposed federated learning approach of functional principal component analysis with an application on a real dataset.

4.5.1 Dataset and Experimental Settings

In this study, we utilize a dataset from Saxena and Goebel (2008) to assess the efficacy of our proposed federated learning method. The dataset includes multi-sensor degradation data from aircraft turbofan engines, encompassing 100 training engines that were run to failure and 100 test engines with operations halted prematurely at random times prior to failure. For the test engines, the dataset also provides the actual times-to-failure (TTF). Each engine's condition was tracked using 21 sensors.

Prior research, particularly the work by Fang et al. (2017), has indicated that sensor 20, analyzed with lognormal regression, offers the most valuable information for modeling this dataset. Given that our study centers on functional principal component analysis, which necessitates separate analysis for each sensor, we focus on the data from sensor 20 to assess the performance of our federated learning algorithm.

The degradation signals in both the training and test datasets exhibit varying lengths due to failure truncation. This variation arises because engines are stopped for maintenance or replacement upon failure, limiting the observation of degradation signals to the failure point. Moreover, since different engines have distinct failure times, the length of these signals varies across machines. To standardize the data, we truncate the degradation signals from all engines to their first 150 timestamps. Engines with TTF less than 150 are excluded, resulting in a training dataset of 94 engines and a test dataset of 37 engines.

The training data is then assigned to three different workers randomly: worker 1 receives 20 engines, worker 2 receives 30 engines, and worker 3 receives 44 engines. Similar to the simulation study, we train a total of eight models. For each of the three workers, we first train a local model using their respective local datasets. We then combine all data to train a non-federated standalone FPCA model (Non-FL). Following this, we train a federated learning FPCA (FL-FPCA) model assuming participation from all three workers. Lastly, we train three different differential privacy federated learning (DP-FL) models, each at a different privacy budget level, $\epsilon = 1, 0.7, \text{ and } 0.5$. For performance evaluation, we obtain the prediction error on the same testing dataset for all models, which contains 37 engines after truncation.

Since the training data assignment was split randomly, and the DP noise was drawn from the Laplace distribution randomly each time, in order to evaluate the robustness of the models against the randomness factors, the whole evaluation process was repeated 5 times for 7 out of 8 models, except the non-FL model with centralized data, with different random seeds aligned with the repetition number. The aggregated errors from all the repetitions are reported.

4.5.2 Results and Analysis

Similar to the simulation study, we report two sets of experiments for the case study with the NASA dataset. The first set uses a sparsity level of 20% missing values, with results reported in Figure 4.4, and the second set uses a sparsity level of 70% missing values, reported in Figure 4.5. From both figures, we can see that the general trends align with the previous simulation

results.

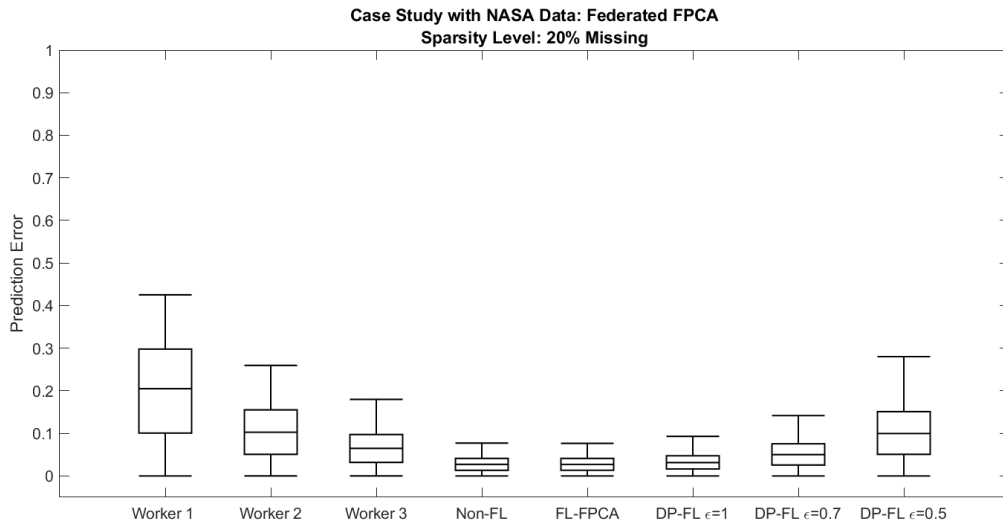


Figure 4.4: Case Study: Error of FPCA Models (Sparsity Level: 20% Missing)

In Figure 4.4, we can read the medians and IQRs for the prediction errors of different FPCA models, as shown in Table 4.3. Just like the simulation results, the performance of FL-FPCA converges to the Non-FL FPCA with centralized data. The median error (and IQR) for both the non-FL model and FL-FPCA is at 0.027(0.028), and the difference between two readings is less than 0.0001. Such a gap is not commonly noticeable and can be ignored in most cases, demonstrating the effectiveness of the federated learning approach in leveraging decentralized data while maintaining a performance level close to that of centralized data models.

The individual local models (Workers 1, 2, and 3) show higher prediction errors compared to the FL-FPCA and Non-FL models. This is expected, as the local models are trained on smaller subsets of the data, which may not fully capture the underlying patterns and variations. Among the local models, Worker 3, which has the largest local dataset (44 engines), achieves the lowest prediction error, with a median (and IQR) of 0.065(0.065). This highlights the importance of having sufficient local data for training accurate models.

The DP-FL models, which incorporate differential privacy, exhibit higher prediction errors

Table 4.3: Case Study: Boxplot Readings (Sparsity Level: 20% Missing)

Sparsity Level: 20% Missing				
Error	Q1	Median	Q3	IQR
Worker 1	0.101	0.205	0.298	0.197
Worker 2	0.051	0.103	0.155	0.104
Worker 3	0.032	0.065	0.097	0.065
Non-FL	0.013	0.027	0.041	0.028
FL-FPCA	0.013	0.027	0.041	0.028
DP-FL $\epsilon = 1$	0.016	0.031	0.047	0.031
DP-FL $\epsilon = 0.7$	0.026	0.050	0.076	0.050
DP-FL $\epsilon = 0.5$	0.051	0.100	0.151	0.100

compared to the non-private FL-FPCA and Non-FL models. As the privacy budget ϵ decreases, the prediction error increases. Among all three DP-FL models, the one with the largest privacy budget, $\epsilon = 1$, has a median error (and IQR) at 0.031(0.031). And when $\epsilon = 0.7$, the median is 0.050(0.050), which is higher than 0.027(0.028) of the non-DP FL-FPCA model, but still lower than the local model of Worker 3, which is 0.065(0.065). However, the DP-FL model with $\epsilon = 0.5$ has a significantly higher median error (and IQR) as 0.100(0.100). The overall trend is similar to the simulation study and demonstrates the trade-off between privacy and utility in differentially private learning. Stronger privacy guarantees (lower ϵ) come at the cost of reduced model accuracy due to the added noise in the gradients, and the DP-FL model should be carefully tuned in practice.

Figure 4.5 and Table 4.4 show the results for the case study with a sparsity level of 70% missing values. The overall trends are similar to those observed in the 20% missing values case, with FL-FPCA performing comparably to Non-FL FPCA and outperforming the individual local models. However, the prediction errors are generally higher compared to the 20% missing values case, indicating the impact of data sparsity on model performance. For example, the median error (and IQR) of Non-FL and FL-FPCA models are both 0.038(0.038) at the sparsity level of 70% missing values, while both Non-FL and FL-FPCA models at the sparsity level of 70% missing values have median at 0.027(0.028). The three local models also have higher prediction errors in the sparsity level of 70% missing values than 20% missing values.

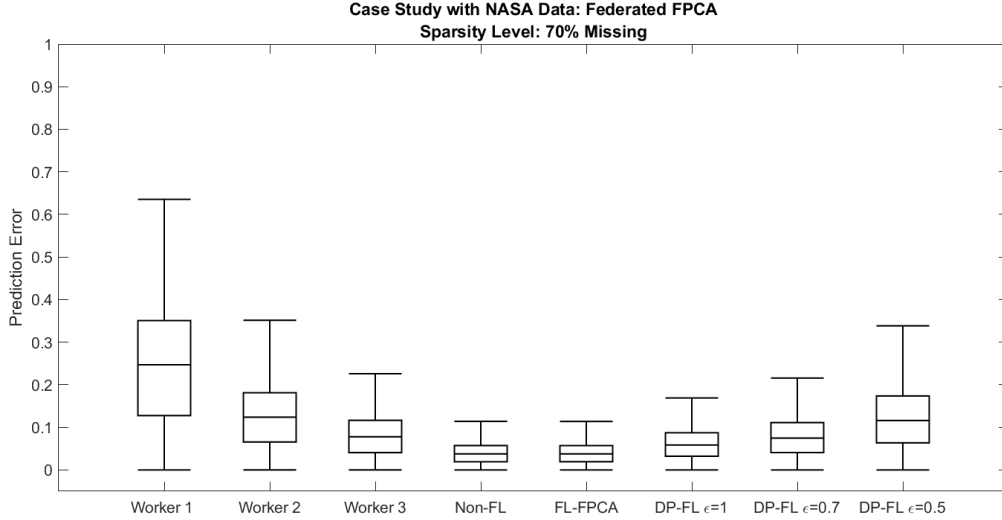


Figure 4.5: Case Study: Error of FPCA Models (Sparsity Level: 70% Missing)

Moreover, we notice that the DP-FL model at $\epsilon = 0.7$ has a median error (and IQR) of 0.075(0.070), which is already close to worker 3’s results, 0.078(0.076). At $\epsilon = 0.5$, the DP-FL model has a relatively high prediction error, which has a median of 0.116(0.110). This also verified the results from our previous investigation, and suggests a balance must be achieved by carefully model tuning when applying the DP-FL algorithm.

Table 4.4: Case Study: Boxplot Readings (Sparsity Level: 70% Missing)

Sparsity Level: 70% Missing				
Error	Q1	Median	Q3	IQR
Worker 1	0.128	0.247	0.351	0.223
Worker 2	0.066	0.124	0.181	0.116
Worker 3	0.041	0.078	0.117	0.076
Non-FL	0.019	0.038	0.057	0.038
FL-FPCA	0.019	0.038	0.057	0.038
DP-FL $\epsilon = 1$	0.032	0.059	0.087	0.055
DP-FL $\epsilon = 0.7$	0.041	0.075	0.111	0.070
DP-FL $\epsilon = 0.5$	0.064	0.116	0.174	0.110

Comparing the case study results with the simulation study, we can observe similar patterns.

Both studies demonstrate that FL-FPCA performs comparably to Non-FL FPCA with centralized data, highlighting the effectiveness of the federated learning approach. Additionally, individual local models have higher prediction errors compared to FL-FPCA and Non-FL models in both studies, emphasizing the benefit of collaborative learning. Furthermore, DP-FL models exhibit a trade-off between privacy and utility in both cases, with stronger privacy guarantees leading to higher prediction errors. Meanwhile, the prediction errors in the case study are generally higher than those in the simulation study, which could be attributed to the differences in the underlying data distribution and the specific characteristics of the NASA dataset.

The case study using the NASA dataset demonstrates the applicability and effectiveness of the proposed federated learning approach for FPCA in a real-world industrial prognostics setting. The results align with the findings from the simulation study, showing that FL-FPCA performs comparably to Non-FL FPCA with centralized data while maintaining data privacy. The study also highlights the benefit of collaborative learning for organizations with limited local data, suggests that when local datasets are limited or sparse, participating in federated learning can significantly improve model performance compared to training models solely on local data. We also investigated the trade-off between privacy and utility when incorporating differential privacy. Specifically, the choice of privacy budget ϵ in differentially private federated learning should be carefully considered based on the specific privacy requirements and the acceptable level of model accuracy. The case study provides valuable insights into the practical implementation of federated learning in industrial prognostics and supports the findings and conclusions drawn from the simulation study.

4.6 Conclusions

In this paper, we propose a federated learning framework for functional principal component analysis (FL-FPCA) to address the challenges of sparse data prognostics in industrial settings. The proposed framework allows for collaborative learning of the FPCA model while

keeping the data decentralized, thereby preserving data privacy and avoiding the need for data centralization.

The key contributions of this work are as follows: first, we introduce a federated learning approach for FPCA that enables the estimation of the mean function, covariance function, and eigenfunctions from decentralized and sparse data. Meanwhile, we also integrate differentially private gradient aggregation into the federated learning framework to enhance the privacy guarantees and protect against potential gradient leakage. Moreover, we conducted comprehensive numerical studies to evaluate the performance of FL-FPCA in various experimental settings, demonstrating its effectiveness in handling sparse and irregular prognostic data while preserving data privacy.

To evaluate the performance of the proposed FL-FPCA framework, we conduct a comprehensive simulation study and a case study using the NASA turbofan engine dataset. The simulation study mimics the characteristics of real-world industrial prognostics data, which is often sparse and distributed across multiple entities. The results demonstrate that FL-FPCA performs comparably to Non-FL FPCA with centralized data. This indicates that the federated learning approach can achieve similar accuracy while preserving data privacy and avoiding the need for data centralization. The study also highlights the importance of data quality and representativeness at each worker, rather than the number of workers itself, in determining the performance of the federated learning approach.

The case study using the NASA dataset further validates the effectiveness of the proposed FL-FPCA framework in a real-world industrial prognostics setting. The results align with the findings from the simulation study, showing that FL-FPCA performs comparably to Non-FL FPCA with centralized data while maintaining data privacy. The case study also highlights the benefit of collaborative learning for organizations with limited local data and the trade-off between privacy and utility when incorporating differential privacy.

The proposed FL-FPCA framework has significant implications for industrial prognostics. It enables organizations to leverage the power of collaborative learning while maintaining the

privacy and security of their data. This is particularly important in industrial settings, where data is often sensitive and proprietary, and sharing raw data across different entities is not feasible or desirable. By allowing for decentralized learning, FL-FPCA enables organizations to benefit from the collective knowledge and insights derived from the data without compromising data privacy.

Furthermore, the integration of differentially private gradient aggregation into the FL-FPCA framework provides an additional layer of privacy protection. Differential privacy ensures that the learned model does not reveal sensitive information about individual data points, even in the presence of adversarial attacks or gradient leakage. This is crucial in industrial prognostics, where the privacy and security of the data are of utmost importance.

In conclusion, the proposed FL-FPCA framework offers a powerful and privacy-preserving approach for collaborative learning in industrial prognostics. It enables organizations to leverage the benefits of decentralized data while maintaining data privacy and security. The simulation study and case study demonstrate the effectiveness of FL-FPCA in handling sparse and decentralized prognostic data, highlighting its potential for real-world applications. As industries continue to generate vast amounts of data and the need for privacy-preserving analytics grows, the FL-FPCA framework provides a promising solution for enabling collaborative learning while safeguarding sensitive information.

CHAPTER

5

CONCLUSION

5.1 Summary of Contributions

This dissertation has explored and advanced the field of privacy-preserving data analytics specifically applied to industrial prognostics. Through three distinct yet interconnected studies, we have developed approaches that integrate differential privacy and federated learning into statistical models for prognostics, ensuring the protection of sensitive data while maintaining predictive accuracy. The key contributions of this research are summarized as follows.

The first study introduced differentially private log-location-scale (DP-LLS) regression models by incorporating differential privacy into LLS regression through the functional mechanism. This approach involves injecting noise into the log-likelihood function of LLS regression to achieve perturbed parameter estimation. By doing so, we have been able to protect data privacy

without significantly compromising the predictive utility of the models. This represents an advancement in the application of differential privacy to regression models used in industrial prognostics, ensuring that sensitive data remains protected while still allowing for accurate failure time predictions.

In the second study, we developed a privacy-preserving sensor selection method for multi-sensor prognostics. This method utilizes LLS regression with group lasso for effective sensor selection, ensuring that only the most relevant sensors are used in the prognostic models. The regression models are optimized using the proximal gradient descent method and are enhanced with differential privacy to protect sensitive information. This study addresses the challenge of selecting the appropriate sensors for prognostic models in a manner that preserves privacy, thereby enabling more efficient and secure industrial prognostics.

The third study proposed a federated learning framework incorporated with differential privacy for functional principal component analysis (FL-FPCA). This framework allows users with sparse data to benefit from collaborative analysis while ensuring privacy. The incorporation of differential privacy within the federated learning paradigm provides a solution to address privacy concerns in industrial prognostics. By enabling collaborative data analysis without compromising individual data privacy, this framework facilitates more accurate and comprehensive prognostic modeling.

The integration of privacy-preserving techniques into industrial prognostics has important implications for both industry and academia. By ensuring the confidentiality of sensitive data, these methods facilitate the wider adoption of predictive maintenance strategies, which can lead to cost savings and efficiency improvements. Predictive maintenance relies heavily on the ability to analyze data from various sensors and systems to predict failures before they occur. However, the use of sensitive or proprietary data in these analyses introduces significant privacy concerns. The methodologies developed in this dissertation provide a foundation for future research and applications in the field, demonstrating that privacy and predictive accuracy can coexist.

The advancements made in this dissertation show that it is possible to protect sensitive data while still enabling effective prognostic modeling. This has the potential to change how industries approach maintenance and operational efficiency. By adopting privacy-preserving data analytics, industries can improve their predictive maintenance strategies without compromising the privacy of their data. This not only enhances operational efficiency but also builds trust with stakeholders by ensuring that sensitive information is protected.

5.2 Future Research Directions

While this dissertation has made progress in privacy-preserving industrial prognostics, several areas warrant further exploration. One area that requires attention is the scalability and performance of the proposed models. Future research could focus on enhancing the scalability and performance of the models to handle different datasets and more complex industrial systems. Efficient algorithms and computational techniques need to be developed to ensure that the models can be applied to real-time industrial environments. This will involve exploring new computational methods and optimizing existing ones to handle the various data forms and their complexity.

Another area for future research is the broader application of the privacy-preserving techniques presented in this dissertation. These techniques can be extended to other domains within industrial engineering and beyond. Exploring their applicability in different contexts will help generalize the methods and uncover new opportunities for privacy-preserving data analytics. For example, these techniques could be applied to healthcare, finance, and other fields where data privacy is a concern.

Improved privacy guarantees are also a crucial area for future research. Investigating advanced privacy-preserving mechanisms and their integration with existing models can provide stronger privacy guarantees. Incorporating the methods proposed in this dissertation with advanced privacy techniques from other fields may offer the potential to provide even greater

privacy protection while maintaining the accuracy and utility of the prognostic models.

Understanding the dynamics of user collaboration in federated learning environments is another important area for future research. Future work should explore incentive mechanisms to encourage participation while maintaining privacy. Additionally, developing robust methods to handle non-cooperative or malicious participants will be essential for practical implementations. Ensuring that all participants can trust the federated learning process is crucial for its success, and this will require addressing both technical and social challenges.

5.3 Final Remarks

In conclusion, this dissertation has made progress in addressing the privacy concerns inherent in industrial prognostics through the innovative application of differential privacy and federated learning. The methodologies developed herein not only protect sensitive information but also maintain the predictive power necessary for effective prognostic modeling. As industries continue to embrace data-driven approaches for maintenance and operational efficiency, the integration of privacy-preserving techniques will be important.

The work presented in this dissertation lays a foundation for future advancements in this area, ensuring that the benefits of predictive analytics can be realized without compromising data privacy. By demonstrating that privacy-preserving data analytics can be effectively applied to industrial prognostics, this research opens up possibilities for the use of predictive maintenance strategies across various industries. The methodologies developed here provide a roadmap for future research and applications, highlighting the potential for privacy-preserving techniques to contribute to the field of industrial prognostics and beyond.

REFERENCES

- Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.
- Antoniadis, A. and Fan, J. (2001). Regularization of wavelet approximations. *Journal of the American Statistical Association*, 96(455):939–967.
- Bakin, S. et al. (1999). Adaptive regression and model selection in data mining problems.
- Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122.
- Cai, T. T. (2001). Regularization of wavelet approximations: discussion. *Journal of the American Statistical Association*, 96(455):960–962.
- Doray, L. (1994). Ibrnr reserve under a loglinear location-scale regression model. In *Casualty Actuarial Society Forum*, volume 2, pages 607–652.
- Dwork, C. (2006). Differential privacy. In *International colloquium on automata, languages, and programming*, pages 1–12. Springer.
- Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006). Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*, pages 265–284. Springer.
- Dwork, C., Roth, A., et al. (2014). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407.
- Eker, O. F., Camci, F., and Jennions, I. K. (2016). Major challenges in prognostics: Study on benchmarking prognostics datasets. In *European Conference of the Prognostics and Health Management Society*, volume 7, pages 1–8.
- Fang, X., Paynabar, K., and Gebraeel, N. (2017). Multistream sensor fusion-based prognostics model for systems with single failure modes. *Reliability Engineering & System Safety*, 159:322–331.
- Fang, X., Paynabar, K., and Gebraeel, N. (2019a). Image-based prognostics using penalized tensor regression. *Technometrics*, 61(3):369–384.
- Fang, X., Yu, F., Yang, G., and Qu, Y. (2019b). Regression analysis with differential privacy preserving. *IEEE access*, 7:129353–129361.
- Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333.

- Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1.
- Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. (2020). Inverting gradients-how easy is it to break privacy in federated learning? *Advances in Neural Information Processing Systems*, 33:16937–16947.
- Golub, G. H., Heath, M., and Wahba, G. (1979). Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223.
- Hall, P. and Hosseini-Nasab, M. (2006). On properties of functional principal components analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):109–126.
- Hidano, S., Murakami, T., Katsumata, S., Kiyomoto, S., and Hanaoka, G. (2017). Model inversion attacks for prediction systems: Without knowledge of non-sensitive attributes. In *2017 15th Annual Conference on Privacy, Security and Trust (PST)*, pages 115–11509. IEEE.
- Hong, S., Zhou, Z., Zio, E., and Wang, W. (2014). An adaptive method for health trend prediction of rotating bearings. *Digital Signal Processing*, 35:117–123.
- Hong, Y., King, C., Zhang, Y., and Meeker, W. Q. (2015). Bayesian life test planning for log-location-scale family of distributions. *Journal of Quality Technology*, 47(4):336–350.
- Hong, Y., Ma, H., and Meeker, W. Q. (2010). A tool for evaluating time-varying-stress accelerated life test plans with log-location-scale distributions. *IEEE Transactions on Reliability*, 59(4):620–627.
- Huang, Y., Gupta, S., Song, Z., Li, K., and Arora, S. (2021). Evaluating gradient inversion attacks and defenses in federated learning. *Advances in Neural Information Processing Systems*, 34:7232–7241.
- Jolliffe, I. T. and Cadima, J. (2016). *Principal component analysis*. Springer.
- Konečn’y, J., McMahan, H. B., Yu, F. X., Richt’arik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*.
- Lee, J., Wu, F., Zhao, W., Ghaffari, M., Liao, L., and Siegel, D. (2014). Prognostics and health management design for rotary machinery systems—reviews, methodology and applications. *Mechanical systems and signal processing*, 42(1-2):314–334.
- Liu, K., Gebraeel, N. Z., and Shi, J. (2013). A data-level fusion model for developing composite health indices for degradation modeling and prognostic analysis. *IEEE Transactions on Automation Science and Engineering*, 10(3):652–664.
- Loutas, T. H., Roulias, D., and Georgoulas, G. (2013). Remaining useful life estimation in rolling bearings utilizing data-driven probabilistic e-support vectors regression. *IEEE Transactions on Reliability*, 62(4):821–832.

- McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR.
- Meeker, W. Q. and Escobar, L. A. (2014). *Statistical methods for reliability data*. John Wiley & Sons.
- Meier, L., Van De Geer, S., and Bühlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 70(1):53–71.
- Mobley, R. K. (2002). *An introduction to predictive maintenance*. Elsevier.
- Parikh, N., Boyd, S., et al. (2014). Proximal algorithms. *Foundations and trends® in Optimization*, 1(3):127–239.
- Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr.
- Ramsay, J. O. and Silverman, B. W. (2005). *Functional data analysis*. Springer.
- Saxena, A. and Goebel, K. (2008). C-MAPSS data set.
- Shang, H. L. (2014). A survey of functional principal component analysis. *AStA Advances in Statistical Analysis*, 98(2):121–142.
- Si, X.-S., Wang, W., Hu, C.-H., and Zhou, D.-H. (2011). Remaining useful life estimation—a review on the statistical data driven approaches. *European journal of operational research*, 213(1):1–14.
- Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. (2013). A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2):231–245.
- Staniswalis, J. G. and Lee, J. J. (1998). Nonparametric regression analysis of longitudinal data. *Journal of the American Statistical Association*, 93(444):1403–1418.
- Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288.
- Vogl, G. W., Weiss, B. A., and Helu, M. (2019). A review of diagnostic and prognostic capabilities and best practices for manufacturing. *Journal of Intelligent Manufacturing*, 30(1):79–95.
- Wang, J.-L., Chiou, J.-M., and Müller, H.-G. (2016). *Functional data analysis*. Springer.
- Wang, T., Yu, J., Siegel, D., and Lee, J. (2008). A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *2008 international conference on prognostics and health management*, pages 1–6. IEEE.
- Wang, Y., Si, C., and Wu, X. (2015). Regression model fitting under differential privacy and model inversion attack. In *IJCAI*, pages 1003–1009.

- Wright, R. E. (1995). Logistic regression.
- Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19.
- Yao, F., Müller, H.-G., and Wang, J.-L. (2005). Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100(470):577–590.
- Yoo, J. I. and Suh, J.-H. (2022). Remaining useful life prediction based on functional principal component analysis and artificial neural network. *Reliability Engineering System Safety*, 218:108171.
- Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67.
- Zhang, J., Zhang, Z., Xiao, X., Yang, Y., and Winslett, M. (2012). Functional mechanism: Regression analysis under differential privacy. *Proceedings of the VLDB Endowment*, 5(11).
- Zhang, Q., Ma, J., Xiao, Y., Lou, J., and Xiong, L. (2020). Broadening differential privacy for deep learning against model inversion attacks. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1061–1070. IEEE.
- Zhao, F., Tian, Z., Liang, X., and Zeng, M. (2015). An integrated prognostics method for failure time prediction of gears subject to the surface wear failure mode. *IEEE Transactions on Reliability*, 64(3):1077–1089.
- Zhou, C. and Fang, X. (2023). A supervised tensor dimension reduction-based prognostic model for applications with incomplete imaging data. *INFORMS Journal on Data Science*.
- Zhu, L., Liu, Z., and Han, S. (2019). Deep leakage from gradients. *Advances in neural information processing systems*, 32.

APPENDIX

APPENDIX

A

PROOFS IN CHAPTER 2

Proof of Proposition 1

Let $\log q = \sum_{r=1}^{\infty} \left[\frac{(-1)^{r+1}}{r} (q-1)^r \right]$ and expand it around 1 in Taylor expansion, and let $e^x = \sum_{r=0}^{\infty} \frac{x^r}{r!}$ and expand it around 0. We have

$$\begin{aligned}
 \tilde{\ell}(\{p_j\}_{j=0}^d, q) &= n \log q + \sum_{i=1}^n \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right) - \sum_{i=1}^n \exp \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right) \\
 &= n \sum_{r=1}^{\infty} \frac{(-1)^{r+1}}{r} (q-1)^r + \sum_{i=1}^n y_i q - \sum_{i=1}^n \sum_{j=0}^d x_{ij} p_j - \\
 &\quad \sum_{i=1}^n \sum_{r=0}^{\infty} \frac{\left(y_i q - \sum_{j=0}^d p_j x_{ij} \right)^r}{r!}
 \end{aligned}$$

which is a polynomial of p_j and q . To truncate it to the second order, we have

$$\begin{aligned}
& \bar{\ell}(\{p_j\}_{j=0}^d, q) \\
&= n \sum_{r=1}^2 \frac{(-1)^{r+1}}{r} (q-1)^r + \sum_{i=1}^n y_i q - \sum_{i=1}^n \sum_{j=0}^d x_{ij} p_j - \sum_{i=1}^n \sum_{r=0}^2 \frac{(y_i q - \sum_{j=0}^d p_j x_{ij})^r}{r!} \\
&= n \left(\frac{(-1)^2}{1} (q-1)^1 + \frac{(-1)^3}{2} (q-1)^2 \right) + \sum_{i=1}^n y_i q - \sum_{i=1}^n \sum_{j=0}^d x_{ij} p_j \\
&\quad - \sum_{i=1}^n \left(1 + \frac{(y_i q - \sum_{j=0}^d p_j x_{ij})^1}{1!} + \frac{(y_i q - \sum_{j=0}^d p_j x_{ij})^2}{2!} \right) \\
&= n \left((q-1) - \frac{1}{2} (q-1)^2 \right) + \sum_{i=1}^n y_i q - \sum_{i=1}^n \sum_{j=0}^d x_{ij} p_j - n - \sum_{i=1}^n \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right) \\
&\quad - \frac{1}{2} \sum_{i=1}^n \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right)^2 \\
&= -\frac{3n}{2} + 2nq - \frac{n}{2} q^2 + \sum_{i=1}^n y_i q - \sum_{i=1}^n \sum_{j=0}^d x_{ij} p_j - n - \sum_{i=1}^n y_i q + \sum_{i=1}^n \sum_{j=0}^d p_j x_{ij} \\
&\quad - \frac{1}{2} \sum_{i=1}^n \left(y_i^2 q^2 - 2y_i q \sum_{j=0}^d p_j x_{ij} + \left(\sum_{j=0}^d p_j x_{ij} \right)^2 \right) \\
&= -\frac{5n}{2} + 2nq - \frac{n}{2} q^2 - \frac{1}{2} \sum_{i=1}^n y_i^2 q^2 + \sum_{i=1}^n \left(y_i q \sum_{j=0}^d p_j x_{ij} \right) - \frac{1}{2} \sum_{i=1}^n \left(\sum_{j=0}^d p_j x_{ij} \right)^2 \\
&= -\frac{5n}{2} + 2nq - \frac{n}{2} q^2 - \\
&\quad \frac{1}{2} \sum_{i=1}^n y_i^2 q^2 + \sum_{i=1}^n \sum_{j=0}^d y_i x_{ij} p_j q - \frac{1}{2} \sum_{i=1}^n \left(\sum_{j=0}^d p_j^2 x_{ij}^2 + \left(\sum_{h=0, h \neq j}^d p_j p_h x_{ij} x_{ih} \right) \right) \\
&= -\frac{5n}{2} + 2nq - \frac{n}{2} q^2 - \frac{1}{2} \sum_{i=1}^n y_i^2 q^2 + \sum_{i=1}^n \sum_{j=0}^d y_i x_{ij} p_j q - \frac{1}{2} \sum_{i=1}^n \sum_{j=0}^d x_{ij}^2 p_j^2 - \\
&\quad \frac{1}{2} \sum_{i=1}^n \left(\sum_{h=0, h \neq j}^d \sum_{j=0}^d x_{ij} x_{ih} p_j p_h \right)
\end{aligned}$$

Proof of Proposition 2

Recall the polynomial function

$$\begin{aligned} \tilde{\ell}(\{p_j\}_{j=0}^d, q) = & -\frac{5n}{2} + 2nq - \frac{1}{2} \left(n + \sum_{i=1}^n y_i^2 \right) q^2 + \sum_{i=1}^n \sum_{j=0}^d y_i x_{ij} p_j q - \frac{1}{2} \sum_{i=1}^n \sum_{j=0}^d x_{ij}^2 p_j^2 \\ & - \frac{1}{2} \sum_{i=1}^n \left(\sum_{h=0, h \neq j}^d \sum_{j=0}^d x_{ij} x_{ih} p_j p_h \right). \end{aligned}$$

Notice we have $d + 2$ variables, $\{p_0, p_1, \dots, p_d, q\}$. Since our polynomial function was truncated by the second order, the polynomial basis of this function consists of zeroth-order, first-order, and second-order combinations of the set of variables. The zeroth-order basis is 1 and its corresponding coefficient is $w_1 = -\frac{5n}{2}$. The first-order basis is q and its corresponding coefficient is $w_q = 2n$. The second-order basis contains q^2 , $\{p_j q\}_{j=0}^d$, $\{p_j^2\}_{j=0}^d$, and the pairwise terms $\{\{p_j p_h\}_{h=0, h \neq j}^d\}$. Their corresponding coefficients are $w_{q^2} = -\frac{1}{2} \left(n + \sum_{i=1}^n y_i^2 \right)$, $\{w_{p_j q} = \sum_{i=1}^n y_i x_{ij}\}_{j=0}^d$, $\{w_{p_j^2} = -\frac{1}{2} \sum_{i=1}^n x_{ij}^2\}_{j=0}^d$, and $\{\{w_{p_j p_h} = -\frac{1}{2} \sum_{i=1}^n x_{ij} x_{ih}\}_{j=0}^d\}_{h=0, h \neq j}$, respectively.

Let D and D' be any two neighbor databases that differ by one data point. Without loss of generality, we assume that D and D' differ in the last sample. Let $(y_n, \{x_{nj}\}_{j=1}^d)$ be the last sample in D and $(y'_n, \{x'_{nj}\}_{j=1}^d)$ be the last sample in D' . Recall that $\tilde{\ell}_D(\{p_j\}_{j=0}^d, q)$ and $\tilde{\ell}_{D'}(\{p_j\}_{j=0}^d, q)$ are the objective functions of regression analysis on D and D' , respectively. In addition, the weights of $\tilde{\ell}_D(\{p_j\}_{j=0}^d, q)$ are $w_1^D, w_q^D, w_{q^2}^D, \{w_{p_j q}^D\}_{j=1}^d, \{w_{p_j^2}^D\}_{j=0}^d, \{\{w_{p_j p_h}^D\}_{j=0}^d\}_{h=0, h \neq j}$ and the weights of $\tilde{\ell}_{D'}(\{p_j\}_{j=0}^d, q)$ are $w_1^{D'}, w_q^{D'}, w_{q^2}^{D'}, \{w_{p_j q}^{D'}\}_{j=1}^d, \{w_{p_j^2}^{D'}\}_{j=0}^d, \{\{w_{p_j p_h}^{D'}\}_{j=0}^d\}_{h=0, h \neq j}$. Thus, we have

$$\begin{aligned}
& \|w_1^D - w_1^{D'}\|_1 + \|w_q^D - w_q^{D'}\|_1 + \|w_{q^2}^D - w_{q^2}^{D'}\|_1 + \sum_{j=0}^d \|w_{p_j q}^D - w_{p_j q}^{D'}\|_1 + \sum_{j=0}^d \|w_{p_j^2}^D - w_{p_j^2}^{D'}\|_1 + \\
& \quad \sum_{n=1, h \neq j}^d \sum_{j=0}^d \|w_{p_j p_h}^D - w_{p_j p_h}^{D'}\|_1 \\
& = 0 + 0 + \left\| -\frac{1}{2}y_n^2 + \frac{1}{2}y_n'^2 \right\|_1 + \sum_{j=0}^d \|y_n x_{nj} - y_n' x_{nj}'\|_1 + \sum_{j=0}^d \left\| -\frac{1}{2}x_{nj}^2 + \frac{1}{2}x_{nj}'^2 \right\|_1 + \\
& \quad \sum_{j=0}^d \sum_{h=0, h \neq j}^d \frac{1}{2} \left\| -x_{nj} x_{nh} + x_{nj}' x_{nh}' \right\|_1 \\
& \leq \left\| -\frac{1}{2}y_n^2 \right\|_1 + \left\| \frac{1}{2}y_n'^2 \right\|_1 + \sum_{j=0}^d \|y_n x_{nj}\|_1 + \sum_{j=0}^d \left\| -y_n' x_{nj}' \right\|_1 + \sum_{j=0}^d \left\| -\frac{1}{2}x_{nj}^2 \right\|_1 + \sum_{j=0}^d \left\| \frac{1}{2}x_{nj}'^2 \right\|_1 \\
& \quad \sum_{j=0}^d \sum_{h=0, h \neq j}^d \frac{1}{2} \left\| -x_{nj} x_{nh} \right\|_1 + \sum_{j=0}^d \sum_{h=0, h \neq j}^d \frac{1}{2} \|x_{nj}' x_{nh}'\|_1 \\
& \leq \frac{1}{2} + \frac{1}{2} + \left(1 + \frac{1}{\sqrt{d}} \times d\right) + \left(1 + \frac{1}{\sqrt{d}} \times d\right) + \frac{1}{2}(1+1) + \frac{1}{2}(1+1) + \\
& \quad + \frac{1}{2}(d \times (d-1) \times \frac{1}{d} + 2 \times d \times \frac{1}{\sqrt{d}}) + \frac{1}{2}(d \times (d-1) \times \frac{1}{d} + 2 \times d \times \frac{1}{\sqrt{d}}) \\
& = 4 + 4\sqrt{d} + d
\end{aligned}$$

Please notice that we have scaled the j th predictor by using $x_{ij} = \frac{\tilde{x}_{ij} - \alpha_j}{(\beta_j - \alpha_j)\sqrt{d}}$, where $\alpha_j = \min\{\{\tilde{x}_{ij}\}_{i=1}^n\}$ and $\beta_j = \max\{\{\tilde{x}_{ij}\}_{i=1}^n\}$ denotes the minimum and maximum values of the j th predictor, respectively. Thus, we have $\|x_{nj}\|_1 \leq \frac{1}{\sqrt{d}}$, $\|x_{nj}'\|_1 \leq \frac{1}{\sqrt{d}}$, $\|x_{nh}\|_1 \leq \frac{1}{\sqrt{d}}$, $\|x_{nh}'\|_1 \leq \frac{1}{\sqrt{d}}$, $\|x_{nj}^2\|_1 \leq \frac{1}{d}$, $\|x_{nj}'^2\|_1 \leq \frac{1}{d}$, $\sum_{i=1}^d \|x_{nj}^2\|_1 \leq 1$, $\|x_{nj} x_{nh}\|_1 \leq \frac{1}{d}$. Also, recall that we have scaled $y_i \in [-1, 1]$ for all i . Thus, $\|y_n\|_1 \leq 1$, $\|y_n'\|_1 \leq 1$, $\|y_n x_{nj}\|_1 \leq \frac{1}{\sqrt{d}}$, and $\|y_n' x_{nj}'\|_1 \leq \frac{1}{\sqrt{d}}$. Furthermore, we have not scaled the x_{n0} , which is the predictor corresponding to the intercept. Thus, $\|x_{n0}\|_1 = 1$.

Proof of Proposition 3

Let D and D' be any two neighbor databases that differ by one data point. Without loss of generality, we assume that D and D' differ in the last sample. Let $(y_n, \{x_{nj}\}_{j=1}^d)$ be the last sample in D and $(y_n', \{x_{nj}'\}_{j=1}^d)$ be the last sample in D' . Let $\mathcal{P}(\tilde{\ell}(\{p_j\}_{j=0}^d, q) | D)$ and $\mathcal{P}(\tilde{\ell}(\{p_j\}_{j=0}^d, q) | D')$

denote the probability that the output of the algorithm given dataset D and D' respectively. Let \tilde{w} denote polynomial weights after adding noises, i.e., $w + \text{Lap}(\frac{\Delta}{\epsilon})$. For example, $\tilde{w}_1 = w_1^D + \text{Lap}(\frac{\Delta}{\epsilon})$ using database D and $\tilde{w}_1 = w_1^{D'} + \text{Lap}(\frac{\Delta}{\epsilon})$ using database D' . Recall that the pdf of Laplace distribution with zero mean and scale $\frac{\Delta}{\epsilon}$ is $\text{pdf}(x) = \frac{\epsilon}{2\Delta} \exp(-\frac{\epsilon\|x\|_1}{\Delta})$ and $\Delta = 4 + 4\sqrt{d} + d$. Thus, we have the following:

$$\begin{aligned}
& \frac{\mathcal{P}(\tilde{\ell}(\{p_j\}_{j=0}^d, q) | D)}{\mathcal{P}(\tilde{\ell}(\{p_j\}_{j=0}^d, q) | D')} \\
&= \frac{\frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_1 - w_1^D\|_1}{\Delta}\right) \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_q - w_q^D\|_1}{\Delta}\right) \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_{q^2} - w_{q^2}^D\|_1}{\Delta}\right)}{\frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_1 - w_1^{D'}\|_1}{\Delta}\right) \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_q - w_q^{D'}\|_1}{\Delta}\right) \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_{q^2} - w_{q^2}^{D'}\|_1}{\Delta}\right)} \\
& \quad \frac{\prod_{j=0}^d \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_{p_j q} - w_{p_j q}^D\|_1}{\Delta}\right) \prod_{j=0}^d \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_{p_j^2} - w_{p_j^2}^D\|_1}{\Delta}\right)}{\prod_{j=0}^d \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_{p_j q} - w_{p_j q}^{D'}\|_1}{\Delta}\right) \prod_{j=0}^d \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_{p_j^2} - w_{p_j^2}^{D'}\|_1}{\Delta}\right)} \\
& \quad \frac{\prod_{j=0}^d \prod_{h=0, h \neq j}^d \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_{p_j p_h} - w_{p_j p_h}^D\|_1}{\Delta}\right)}{\prod_{j=0}^d \prod_{h=0, h \neq j}^d \frac{\epsilon}{2\Delta} \exp\left(-\frac{\epsilon\|\tilde{w}_{p_j p_h} - w_{p_j p_h}^{D'}\|_1}{\Delta}\right)} \\
& \leq \exp\left(\frac{\epsilon}{\Delta} (\|\tilde{w}_1^D - \tilde{w}_1^{D'}\|_1 + \|\tilde{w}_q^D - \tilde{w}_q^{D'}\|_1 + \|\tilde{w}_{q^2}^D - \tilde{w}_{q^2}^{D'}\|_1 + \sum_{j=0}^d \|\tilde{w}_{p_j q}^D - \tilde{w}_{p_j q}^{D'}\|_1 + \right. \\
& \quad \left. \sum_{j=0}^d \|\tilde{w}_{p_j^2}^D - \tilde{w}_{p_j^2}^{D'}\|_1 + \sum_{j=0}^d \sum_{h=0, h \neq j}^d \|\tilde{w}_{p_j p_h}^D - \tilde{w}_{p_j p_h}^{D'}\|_1)\right) \\
& \leq \exp\left(\frac{\epsilon}{\Delta} (4 + 4\sqrt{d} + d)\right) \\
& = \exp(\epsilon)
\end{aligned}$$

This implies that the computation of $\tilde{\ell}(\{p_j\}_{j=0}^d, q)$ ensures ϵ -differential privacy. Since the final result of Algorithm 1 is achieved by solving $\tilde{\ell}(\{p_j\}_{j=0}^d, q)$ without using any additional information from the original database, Algorithm 1 satisfies ϵ -differential privacy as well.

Proof of Proposition 4

Based on Taylor's expansion, we have

$$\log(1 + e^z) \approx \left[\log(2) + \frac{z}{2} + \frac{z^2}{8} \right]$$

Recall that $l(\tilde{\mu}, \sigma)$ can be written using as follows:

$$l(\{p_j\}_{j=0}^d, q) = n \log q + \sum_{i=1}^n \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right) - 2 \sum_{i=1}^n \log \left(1 + \exp \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right) \right)$$

To truncate it to the second order, we have:

$$\begin{aligned} & \tilde{l}(\{p_j\}_{j=0}^d, q) \\ = & n \sum_{r=1}^2 \frac{(-1)^{r+1}}{r} (q-1)^r + \sum_{i=1}^n y_i q - \sum_{i=1}^n \sum_{j=0}^d x_{ij} p_j \\ & - 2 \sum_{i=1}^n \left[\log 2 + \frac{1}{2} (y_i q - \sum_{j=0}^d p_j x_{ij}) + \frac{1}{8} (y_i q - \sum_{j=0}^d p_j x_{ij})^2 \right] \\ = & -\frac{3n}{2} + 2nq - \frac{n}{2} q^2 + \sum_{i=1}^n y_i q - \sum_{i=1}^n \sum_{j=0}^d x_{ij} p_j - 2n \log 2 \\ & - \sum_{i=1}^n y_i q + \sum_{i=1}^n \sum_{j=0}^d p_j x_{ij} - \frac{1}{4} \sum_{i=1}^n \left(y_i q - \sum_{j=0}^d p_j x_{ij} \right)^2 \\ = & -\frac{3n}{2} + 2nq - \frac{n}{2} q^2 - 2n \log 2 - \frac{1}{4} \sum_{i=1}^n \left(y_i^2 q^2 - 2y_i q \sum_{j=0}^d p_j x_{ij} + \left(\sum_{j=0}^d p_j x_{ij} \right)^2 \right) \\ = & -\frac{3n}{2} + 2nq - \frac{n}{2} q^2 - 2n \log 2 - \frac{1}{4} \sum_{i=1}^n y_i^2 q^2 + \frac{1}{2} \sum_{i=1}^n \left(y_i q \sum_{j=0}^d p_j x_{ij} \right) - \frac{1}{4} \sum_{i=1}^n \left(\sum_{j=0}^d p_j x_{ij} \right)^2 \\ = & -\frac{3n}{2} + 2nq - \frac{n}{2} q^2 - 2n \log 2 - \frac{1}{4} \sum_{i=1}^n y_i^2 q^2 + \frac{1}{2} \sum_{i=1}^n \sum_{j=0}^d y_i x_{ij} p_j q \\ & - \frac{1}{4} \sum_{i=1}^n \left(\sum_{j=0}^d p_j^2 x_{ij}^2 + \left(\sum_{h \neq j}^d \sum_{j=0}^d p_h p_j x_{ij} x_{ih} \right) \right) \end{aligned}$$

$$\begin{aligned}
&= -\frac{3n}{2} + 2nq - \frac{n}{2}q^2 - 2n \log 2 - \frac{1}{4} \sum_{i=1}^n y_i^2 q^2 + \frac{1}{2} \sum_{i=1}^n \sum_{j=0}^d y_i x_{ij} p_j q \\
&\quad - \frac{1}{4} \sum_{i=1}^n \sum_{j=0}^d x_{ij}^2 p_j^2 - \frac{1}{4} \sum_{i=1}^n \left(\sum_{h \neq j}^d \sum_{j=0}^d x_{ij} x_{ih} p_j p_h \right)
\end{aligned}$$

Proof of Proposition 5

Recall our polynomial function is

$$\begin{aligned}
\tilde{\ell}(\{p_j\}_{j=0}^d, q) &= -n \left(\frac{3}{2} + 2 \log 2 \right) + 2nq - \left(\frac{n}{2} + \frac{1}{4} \sum_{i=1}^n y_i^2 \right) q^2 + \frac{1}{2} \sum_{i=1}^n \sum_{j=0}^d y_i x_{ij} p_j q \\
&\quad - \frac{1}{4} \sum_{i=1}^n \sum_{j=0}^d x_{ij}^2 p_j^2 - \frac{1}{4} \sum_{i=1}^n \left(\sum_{h \neq j}^d \sum_{j=0}^d x_{ij} x_{ih} p_j p_h \right)
\end{aligned}$$

The zeroth-order basis 1 has coefficient $w_1 = -n \left(\frac{3}{2} + 2 \log 2 \right)$, and the first-order basis q has coefficient $w_q = 2n$. The second-order basis are q^2 , $\{p_j q\}_{j=0}^d$, $\{p_j^2\}_{j=0}^d$, and the pairwise combinations $\{\{p_j p_h\}_{j=0, h \neq j}^d\}$. Their corresponding coefficients are $w_{q^2} = -\left(\frac{n}{2} + \frac{1}{4} \sum_{i=1}^n y_i^2 \right)$, $\{w_{p_j q} = \frac{1}{2} \sum_{i=1}^n y_i x_{ij}\}_{j=0}^d$, $\{w_{p_j^2} = -\frac{1}{4} \sum_{i=1}^n x_{ij}^2\}_{j=0}^d$, and $\{\{w_{p_j p_h} = -\frac{1}{4} \sum_{i=1}^n x_{ij} x_{ih}\}_{j=0, h \neq j}^d\}$.

Let D and D' be any two neighbor databases that differ by one data point. Without loss of generality, we assume that D and D' differ in the last sample. Let $(y_n, \{x_{nj}\}_{j=1}^d)$ be the last sample in D and $(y'_n, \{x'_{nj}\}_{j=1}^d)$ be the last sample in D' .

Recall that $\tilde{\ell}_D(\{p_j\}_{j=0}^d, q)$ and $\tilde{\ell}_{D'}(\{p_j\}_{j=0}^d, q)$ are the objective functions of regression analysis on D and D' , respectively. In addition, the weights of objective function $\tilde{\ell}_D(\{p_j\}_{j=0}^d, q)$ are $w_1^D, w_q^D, w_{q^2}^D, \{w_{p_j q}^D\}_{j=1}^d, \{w_{p_j^2}^D\}_{j=0}^d, \{\{w_{p_j p_h}^D\}_{j=0, h \neq j}^d\}$ and the weights of objective function $\tilde{\ell}_{D'}(\{p_j\}_{j=0}^d, q)$ are $w_1^{D'}, w_q^{D'}, w_{q^2}^{D'}, \{w_{p_j q}^{D'}\}_{j=1}^d, \{w_{p_j^2}^{D'}\}_{j=0}^d, \{\{w_{p_j p_h}^{D'}\}_{j=0, h \neq j}^d\}$. Thus, we have

$$\begin{aligned}
& \|w_1^D - w_1^{D'}\|_1 + \|w_q^D - w_q^{D'}\|_1 + \|w_{q^2}^D - w_{q^2}^{D'}\|_1 + \sum_{j=0}^d \|w_{p_j q}^D - w_{p_j q}^{D'}\|_1 + \sum_{j=0}^d \|w_{p_j^2}^D - w_{p_j^2}^{D'}\|_1 + \\
& \quad \sum_{n=1, h \neq j}^d \sum_{j=0}^d \|w_{p_j p_h}^D - w_{p_j p_h}^{D'}\|_1 \\
& = 0 + 0 + \left\| -\frac{1}{4}y_n^2 + \frac{1}{4}y_n'^2 \right\|_1 + \frac{1}{2} \sum_{j=0}^d \|y_n x_{nj} - y_n' x_{nj}'\|_1 + \sum_{j=0}^d \left\| -\frac{1}{4}x_{nj}^2 + \frac{1}{4}x_{nj}'^2 \right\|_1 + \\
& \quad \frac{1}{4} \sum_{j=0}^d \sum_{h=0, h \neq j}^d \left\| -x_{nj} x_{nh} + x_{nj}' x_{nh}' \right\|_1 \\
& \leq \left\| -\frac{1}{4}y_n^2 \right\|_1 + \left\| \frac{1}{4}y_n'^2 \right\|_1 + \frac{1}{2} \sum_{j=0}^d \|y_n x_{nj}\|_1 + \frac{1}{2} \sum_{j=0}^d \left\| -y_n' x_{nj}' \right\|_1 + \sum_{j=0}^d \left\| -\frac{1}{4}x_{nj}^2 \right\|_1 + \sum_{j=0}^d \left\| \frac{1}{4}x_{nj}'^2 \right\|_1 \\
& \quad \frac{1}{4} \sum_{j=0}^d \sum_{h=0, h \neq j}^d \left\| -x_{nj} x_{nh} \right\|_1 + \frac{1}{4} \sum_{j=0}^d \sum_{h=0, h \neq j}^d \left\| x_{nj}' x_{nh}' \right\|_1 \\
& \leq \frac{1}{4} + \frac{1}{4} + \frac{1}{2} \left(1 + \frac{1}{\sqrt{d}} \times d\right) + \frac{1}{2} \left(1 + \frac{1}{\sqrt{d}} \times d\right) + \frac{1}{4}(1+1) + \frac{1}{4}(1+1) + \\
& \quad \frac{1}{4} \left(d \times (d-1) \times \frac{1}{d} + 2 \times d \times \frac{1}{\sqrt{d}}\right) + \frac{1}{4} \left(d \times (d-1) \times \frac{1}{d} + 2 \times d \times \frac{1}{\sqrt{d}}\right) \\
& = 2 + 2\sqrt{d} + \frac{1}{2}d
\end{aligned}$$

Please notice that we have scaled the j th predictor by using $x_{ij} = \frac{\tilde{x}_{ij} - \alpha_j}{(\beta_j - \alpha_j)\sqrt{d}}$, where $\alpha_j = \min\{\{\tilde{x}_{ij}\}_{i=1}^n\}$ and $\beta_j = \max\{\{\tilde{x}_{ij}\}_{i=1}^n\}$ denotes the minimum and maximum values of the j th predictor, respectively. Thus, we have $\|x_{nj}\|_1 \leq \frac{1}{\sqrt{d}}$, $\|x_{nj}'\|_1 \leq \frac{1}{\sqrt{d}}$, $\|x_{nh}\|_1 \leq \frac{1}{\sqrt{d}}$, $\|x_{nh}'\|_1 \leq \frac{1}{\sqrt{d}}$, $\|x_{nj}^2\|_1 \leq \frac{1}{d}$, $\|x_{nj}'^2\|_1 \leq \frac{1}{d}$, $\sum_{i=1}^d \|x_{nj}^2\| \leq 1$, $\|x_{nj} x_{nh}\|_1 \leq \frac{1}{d}$. Also, recall that we have scaled $y_i \in [-1, 1]$ for all i . Thus, $\|y_n\|_1 \leq 1$, $\|y_n'\|_1 \leq 1$, $\|y_n x_{nj}\|_1 \leq \frac{1}{\sqrt{d}}$, and $\|y_n' x_{nj}'\|_1 \leq \frac{1}{\sqrt{d}}$. Furthermore, we have not scaled the x_{n0} , which is the predictor corresponding to the intercept. Thus, $\|x_{n0}\|_1 = 1$.

Proof of Proposition 6

The proof is the same to that of the proof of Proposition 3 except that $\Delta = 2 + 2\sqrt{d} + \frac{1}{2}d$.