




# A statistical approach to quality evaluation of AI mislabel detection algorithm

Jiayi Lian<sup>a</sup>, Kexin Xie<sup>a</sup>, Kevin Choi<sup>b</sup>, Xueying Liu<sup>a\*</sup>, Balaji Veeramani<sup>b</sup>, Sathvik Murli<sup>b</sup>, Alison Hu<sup>b</sup>, Laura Freeman<sup>a</sup>, Edward Bowen<sup>b</sup>, and Xinwei Deng<sup>a</sup> 

<sup>a</sup>Department of Statistics, Virginia Tech, Blacksburg, Virginia; <sup>b</sup>AI Center of Excellence, Deloitte & Touche, New York, New York

## ABSTRACT

Testing and evaluating the quality of Artificial Intelligence (AI) algorithms is important for confidently deploying them in real applications. AI algorithms depend both on the hyper-parameters and the nature of the training data. A measure of the quality of an AI algorithm can be obtained by exhaustively evaluating the performance of the algorithm across hyper-parameters and data quality factors. However, such a procedure is challenging as it is not practical to enumerate all possible level combinations of the hyper-parameters and data quality factors. In this work, we present a principled framework using a statistical approach to systematically conduct quality evaluation of AI algorithms, named as QE-AI. The proposed framework consists of an efficient space filling design in a high-dimensional constraint space and an effective surrogate model using an additive Gaussian process to enable efficient quality evaluation of AI algorithms. We demonstrate the performance of the proposed approach for an AI mislabel detection algorithm.

## KEYWORDS

AI quality; constraint space; data quality; experimental design; space filling design; surrogate model

## 1. Introduction

### 1.1. Motivation

Great advancements in various scientific and industrial applications have been made possible by artificial intelligence (AI) algorithms (Ben Fredj et al. 2020; Ma et al. 2019). However, the quality assurance of these algorithms is still a significant concern. For example, a small malicious perturbation on data (Chen et al. 2020) or a certain amount of noise in the data (Patrini et al. 2017) can deceive AI algorithms and result in catastrophic failure of the model when applying them in real-life scenarios. Therefore, systematically evaluating the quality of AI algorithms is of great importance to safely leverage these algorithms for practical applications.

As an example, consider the ability of the AI mislabel detection (MLD) algorithm to detect data poisoning. In classification tasks, mislabeled responses can affect model training and undermine the performance of algorithms. These mislabels can be introduced in the data by simply assigning wrong labels to training data (Guan et al. 2011) by adversaries, or other non-malicious sources. Unintended mislabeling is common in applications due to crowd-sourcing or

differences in domain-expertise of annotators. MLD algorithms are used to detect mislabeled samples in the dataset. The performance of MLD algorithms is affected by many factors such as hyper-parameters in the algorithm and other data quality factors. Even though various MLD algorithms (Guan et al. 2011; Malossini, Blanzieri, and Ng 2006; Pulastya et al. 2021; Vu et al. 2019) have been developed in the literature, no comprehensive evaluation exists to assess their quality. Therefore, a systematic approach to investigate the quality of the MLD method is necessary to use it reliably, and to ensure the safety and assurance of it and the overall AI application.

The two major groups of factors affecting performance of AI algorithms (including MLD algorithms) are data quality and hyper-parameters. Data quality factors include class imbalance, dataset types used for training, noise, and the number of mislabeled observations in training data. Hyper-parameter factors of the algorithms are deeply related to the corresponding architectures. For MLD algorithms, the hyper-parameters include weights in the loss function and threshold values used for assigning samples to classes or those used to assign a particular sample to an out of distribution sample (or outlier).

**CONTACT** Xinwei Deng  [xdeng@vt.edu](mailto:xdeng@vt.edu)  Department of Statistics, Virginia Tech, Blacksburg, VA.

\*Xueying Liu has recently become an assistant professor at Department of Statistics of Baylor University.

To comprehensively evaluate the quality of the AI algorithms and obtain a trustworthy assessment on the quality of the algorithms, it is not practical to enumerate all possible level combinations of the hyper-parameters and data quality factors. It calls for a systematic approach to efficiently and effectively evaluate the quality of the AI algorithms. In this work, we propose a principled statistical framework, denoted as **QE-AI**, to conduct quality evaluation of the AI algorithm. Specifically, we propose a design-of-experimental framework to construct an efficient space-filling design in a high-dimensional constraint space and develop an effective surrogate model based on an additive Gaussian process to enable the emulation of the quality of AI algorithms. That is, we consider the design space consisting of both continuous and categorical factors with constraints. For such a high-dimensional constraint space, we adopt a constraint space-filling design (Joseph 2016) to systematically investigate how the data quality affects the quality of AI algorithm when internal structure of AI algorithms varies.

The proposed framework can set an evaluation protocol for AI algorithm practitioners to evaluate and rank the quality of AI algorithms, thus enhance the AI assurance of robustness, reproducibility, and transparency while deploying them in the real world. The proposed QE-AI framework is demonstrated by using the mislabel detection algorithm, but the developed framework can be widely used for other AI algorithms. It paves a foundation for understanding the uncertainty of AI algorithms, finding optimal configuration of hyper-parameters of the AI algorithms, and shedding a light on the assessment of AI algorithm's limitations.

## 1.2. Contribution

The main contribution of this work is as follows. First, we propose a principled framework of using a design-of-experimental approach to systematically evaluate the quality of AI algorithms. The developed framework is not restricted to the MLD algorithm against data poisoning, but can be used for evaluating other AI algorithms, especially when the data quality and algorithm structure (i.e., hyper-parameters) are intertwined. Second, to systematically investigate how the data quality affects the quality of AI algorithms when the internal structure of AI algorithms varies, we propose an effective design criterion with an efficient construction algorithm to obtain a space-filling design in a high-dimensional constraint space to

investigate the quality of AI algorithms in terms of appropriate metrics. We use detection accuracy and prediction accuracy as metrics for MLD algorithms. Specifically, we consider the design space consisting of three continuous factors without constraint, continuous factors of class proportions with linear constraint, and one binary factor for “data type.” The construction algorithm is efficient by leveraging the simplicity of coordinate descent in discrete optimization and constraint continuous optimization. Third, due to the complexity of design criterion and design space, an initial design is crucial to enable the design construction algorithm. Our method of initial design based on algebraic construction is very fast in computation with flexible run sizes. Fourth, we adopt an additive Gaussian process model as a surrogate model to emulate the quality of AI algorithms as a function of data quality factors and AI algorithm's hyper-parameters. The use of an additive Gaussian process can accommodate both continuous and categorical factors of interest, providing accurate prediction and uncertainty quantification of the quality of the algorithm.

## 2. Brief literature review

In the literature, Rushby (1988) discussed the idea of AI quality measurement and assurance several decades ago. AI reliability and robustness (current concepts in AI quality) became increasingly popular in both academic research and industrial applications (Dietterich 2017; Russell, Dewey, and Tegmark 2015; Virani, Iyer, and Yang 2020). For example, Lian et al. (2021) propose a design-of-experimental framework for investigating AI robustness as it relates to the class imbalance issue and distribution shift of classes between training and test data. Several works use experimental designs to tune hyper-parameters of AI algorithms (Balestrassi et al. 2009; Mutny, Kirschner, and Krause 2020; Packianather, Drake, and Rowlands 2000; Staelin 2003).

The evaluation of AI quality needs a meticulous design of experiment since the size of search space can be very large (Bell et al. 2022). The commonly used design for the search space is to have the space-filling property (Joseph 2016). Among various space-filling designs, the Latin hypercube design and its variants have received a lot of attention (Jung and Yum 2011). For example, Joseph, Gul, and Ba (2015) propose a so-called MaxPro design to maximize the design's space-filling capacity on all subspaces. Note that the design for the continuous factors of class proportions has a linear constraint, which is a mixture design in the

literature (Cornell 2011). Typical construction of mixture designs (Gomes, Claeys-Bruno, and Sergent 2018) is not applied for high dimensions due to the computational complexity. Quadratic and cubic linear models (Piepel and Cornell 1994) are used to analyze mixture designs in low dimensions.

Gaussian process (GP) models are effective as an efficient surrogate model for a complex system (Bernardo et al. 1998) in high dimensional space. Note that GP is usually valid for continuous variables (Cardelli et al. 2019; Ling, Low, and Jaillot 2016). When the data contains categorical input variables, one may use one-hot encoding to transfer a categorical variable to a continuous variable (Garrido-Merchán and Hernández-Lobato 2020). Additive Gaussian process (Deng et al. 2017) has been proposed to handle both qualitative and quantitative factors.

### 3. The proposed QE-AI framework

Figure 1 illustrates the motivation and general idea of the proposed QE-AI framework. It aims to investigate how various factors affect the performance of AI algorithms by a careful design of experiment (DoE) and a proper emulator. The use of DoE approach can mitigate the evaluation bias brought by the ad-hoc search (e.g., a grid search at a fixed data quality), which is commonly used among machine learning practitioners, and provide more accurate prediction and uncertainty quantification.

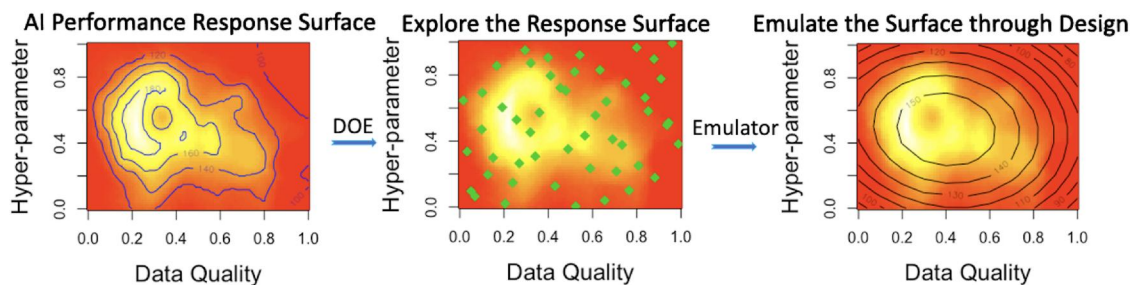
In this section, we discuss problem setup in Section 3.1 and describe a recent MLD algorithm from the literature that we use to demonstrate the framework in Section 3.2. Then we detail the proposed QE-AI framework with design factors and responses in Section 3.3, design construction in Section 3.4, and surrogate modeling in Section 3.5.

#### 3.1. Problem setup

Typically, AI algorithms are affected by two types of factors, factors affecting data quality and hyper-parameters

of the algorithms. All of these factors can be further grouped into three categories by the corresponding data types: continuous factor with constraint, continuous factor without constraint, and categorical factors. Assume the factor space  $\Omega = \omega_1 \times \omega_2 \times \omega_3$ , where  $\omega_1$  is the space for the continuous factor without constraint,  $\omega_2$  is the space for the continuous factor with constraints, and  $\omega_3$  is the space for the categorical factor. Consider the factor space  $\Omega$  as the input of the AI algorithm  $f(\cdot)$ , the output  $y = p(f(\Omega))$  is the consequent performance evaluation, where  $y \in Y$ ,  $Y$  is the response space,  $p(\cdot)$  is the performance measurement function. Because of the complexity of  $f(\cdot)$ , the relationship between input  $\Omega$  and output  $y$  is non-trivial. To infer the relationship, an emulator model  $g: \Omega \rightarrow Y$  is adopted. To verify the AI quality, the model should be capable of both generating a prediction and an uncertainty quantification. Once the emulator model is determined, we can collect experimental data to start the quality investigation. Due to the considerable computational expense involved in the training of the AI model  $f(\cdot)$ , obtaining performance evaluation on a wide range multi-dimensional grid is time consuming. Only considering a few hyper-parameters induces biased inference. Suppose only  $N_B = r \times N_{total}$  runs are approved as the budget for the experiment, where  $N_{total}$  is the number of the input factor combinations (i.e., The total number of experimental design settings.) and  $r$  is the number of replications for the individual input factor combinations, the budget should be wisely used to obtain a more comprehensive investigation of the AI system efficiently (key notations are summarized in Table 1).

In the following sections, we demonstrate the framework using an MLD algorithm as an example. After we briefly review the MLD algorithm, we describe the choice of design factors and response, space filling design construction in a constraint space, and surrogate modeling with additive Gaussian process.



**Figure 1.** An illustration of the motivation and the proposed QE-AI framework. It uses design of experiments, the green dots, to comprehensively explore the AI performance response surface, and builds an emulator model to make proper inference.

**Table 1.** A table of notation used in the design.

|             | Definition  |
|-------------|---|
| $z_1$       | Transformed weight ratio of the losses for classifier and VAE.              |
| $z_2$       | The deviation threshold $\alpha$ .  |
| $z_3$       | The percentage of mislabeling data.   |
| $z_4$       | The choices of benchmark datasets.  |
| $x_l$       | The percentage of class $l$ in the training data,<br>$l = 1, 2, \dots, m$ . |
| $N$         | The number of level combinations for $x_1, \dots, x_m$ .                    |
| $N_{total}$ | The total number of settings of all input factors.                          |
| $r$         | The number of replications for each<br>configuration of the input factors   |
| $N_B$       | The total number of runs.   |

### 3.2. Brief review of the MLD algorithm

In this section, we briefly review the MLD algorithm Pulastya et al. (2021) we use for presenting the proposed framework for evaluating the quality of the AI algorithm. The algorithm includes a variational autoencoder (VAE) (Kingma and Welling 2013) and a simple sigmoid classifier.

Without loss of generality, we consider the  $m$ -class classification problem with data  $(\mathbf{U}_i, c_i)$ , where  $\mathbf{U}_i$  represents an  $r$ -dimensional input data point  $i$  and  $c_i \in \{1, \dots, m\}$  represents its class label. Here  $\mathbf{U}_i = (u_{1(i)}, \dots, u_{r(i)})$  with  $u_{j(i)}$  to be  $j^{th}$  element of  $\mathbf{U}_i$ ,  $j = 1, \dots, r$ . The encoder architecture consists of convolutional layers and two dense layers to encode the mean and variance of the latent layer,  $\mathbf{v}$  of 100 dimensions. Then the output of the latent layer,  $\mathbf{v}$ , is provided as the input to the decoder for reconstructing the input. The output of the latent layer  $\mathbf{v}$  is also provided as the input to the sigmoid classification layer. The model is trained using a composite loss function as

$$L = w \cdot L_{ELBO} + (1 - w) \cdot L_{CL},$$

where  $0 \leq w \leq 1$  is the weight, the  $L_{ELBO} = E_{q(\mathbf{v}|\mathbf{U})}[\log(p(\mathbf{U}|\mathbf{v}))] - KL[q(\mathbf{v}|\mathbf{U})||p(\mathbf{v})]$  is the evidence lower bound loss (ELBO) function for the VAE, and  $L_{CL} = -\sum t(c_i)\log(p(\mathbf{U}_i|c_i))$  is the cross entropy loss function for the classifier. Here  $KL[\cdot]$  is the Kullback-Leibler divergence,  $t(\cdot)$  is the true distribution, and  $p(\cdot)$  is the predicted distribution. The layers in the VAE and the classifier are trained simultaneously. Using the trained structure, “mislabeling score” is constructed as follows. For a group of given classes  $\{\mathbf{U}_i : \forall i \text{ with } c_i = l\}$ , one calculates the median of absolute deviation from the reconstructed median as  $M_l = \text{median}\{|\mathbf{U}_{recon_i} - m_l| : \forall i \text{ with } c_i = l\}$ , where  $m_l$  is the median of all reconstructed input for the group. Count element-wise if the deviation from the reconstructed input to the median is greater than  $\alpha \times M_l$  (i.e.,  $\sum_{j=1}^r \mathbb{1}(|u_{j(recon_i)} - m_{j(l)}| > \alpha M_{j(l)})$ , where  $\mathbb{1}(\cdot)$  is an indicator function;  $u_{j(recon_i)}$ ,  $m_{j(l)}$ , and  $M_{j(l)}$

correspond to the  $j^{th}$  element of  $\mathbf{U}_{recon_i}$ ,  $m_l$ , and  $M_l$ ). If the count is greater than a threshold, the algorithm marks the item  $i$  as mislabeled. It is seen that  $\alpha$  is a hyper-parameter.

### 3.3. Design factors and response variables

In this work, we focus on investigating the effects of data quality factors and algorithm hyper-parameters for the MLD algorithm.

For the MLD algorithm, one influential factor is the weight in the loss function. Thus, the weight ratio  $\frac{w}{1-w}$  is considered as one factor. To symmetrize the weight ratio space and spread small values apart, we define the transformed weight ratio  $z_1$  as:

$$z_1 = \text{sgn}(w - 0.5) \times \frac{\max\{w, 1 - w\}}{\min\{w, 1 - w\}},$$

where  $\text{sgn}(\cdot)$  denotes the sign function. This definition ensures that  $z_1$  is positive when  $w > 0.5$ , negative when  $w < 0.5$ , and symmetrically reflects the imbalance between the two losses. The value of  $\alpha$  as the threshold of deviation is also critical for detecting the mislabeled data points. Consequently, we consider the hyper-parameter  $\alpha$  as another factor of interest, denoted as  $z_2$ . For the data quality factors, we mainly consider the proportion of mislabeling in the training data, the class imbalance in the training data, and the type of datasets. For the first important data quality factor, the percentage of mislabeled data is considered, denoted as  $z_3$ . Intuitively, when the proportion of mislabeled data is high in training data, it will be difficult to extract informative features, with inaccurate information on responses. The second data quality factor, the class imbalance, can undermine both detection accuracy and classification accuracy since sufficient information about the minority classes could be absent. It is important to investigate the robustness of the MLD algorithm with respect to the class imbalance in training data as well as the proportions of mislabeled data. We assume that the class imbalance corresponds to proportions of classes in the training set as  $x_1, x_2, \dots, x_m$  with constraint  $\sum_{l=1}^m x_l = 1$  and  $0 \leq x_l \leq 1$ ,  $l = 1, 2, \dots, m$ . For the third data quality factor, it is known that the MLD algorithm can have different performances on different types of benchmark datasets. Thus, we consider  $k$  benchmark datasets with  $k$  different types  $DS_1, \dots, DS_k$  as a categorical factor

$$z_4 = \begin{cases} 1 & \text{dataset for use is } DS_1, \\ \vdots & \\ k & \text{dataset for use is } DS_k. \end{cases}$$

We consider the performance metrics of the MLD algorithm, prediction accuracy  $y_1$  and detection accuracy  $y_2$  as the response variables. The prediction accuracy is the classification accuracy, which is the percentage of observations correctly classified based on the underlying class labels. For the MLD algorithm, detection accuracy is the percentage of correctly identified mislabeled cases, which is an important metric for assessing mislabeled data detection methods.

### 3.4. Design construction

Note that the factors we consider contain continuous factors  $z_1, z_2, z_3$  without constraint, a categorical factor  $z_4$ , and  $m$  continuous factors  $x_1, x_2, \dots, x_m$  with constraint. For such a complicated, high-dimensional constraint space, it is not practical to choose a large set of random design points (i.e., combination of factors) from the search space to investigate the quality of the MLD algorithm due to the limited computational resources. However, finding a set of representative design points in such a complicated space is also not trivial. To address this issue, we consider a space-filling design in the constraint space by leveraging the maximum projection from Joseph, Gul, and Ba (2015) under the setting of the constraint space.

First, we construct a space-filling design for  $x_1, x_2, \dots, x_m$  with constraint  $\sum_{l=1}^m x_l = 1$  and  $0 \leq x_l \leq 1, l = 1, 2, \dots, m$ . Denote the design with  $N$  points as  $\mathbf{X}_D = (x_{il})_{N \times m}, i = 1, \dots, N; l = 1, 2, \dots, m$ . We consider finding the space-filling design by

$$\begin{aligned} \mathbf{X}_D = \arg \min_{\mathbf{X}} & \sum_{i=1}^{N-1} \sum_{j=i+1}^N \frac{1}{\prod_{l=1}^m (x_{il} - x_{jl})^2} \\ \text{s.t.} & \sum_{l=1}^m x_{il} = 1, \sum_{l=1}^m x_{jl} = 1; \\ & 0 \leq x_{il} \leq 1, 0 \leq x_{jl} \leq 1, \forall i, j, m. \end{aligned} \quad (1)$$

Note that the above constraint  $\sum_{l=1}^m x_l = 1, 0 \leq x_l \leq 1, l = 1, 2, \dots, m$  defines a subspace where all dimensions have equal importance. It follows that one could consider the modified maximin criterion (Joseph, Gul, and Ba 2015)

$$f(\mathbf{X}|\boldsymbol{\theta}) = \sum_{i=1}^{N-1} \sum_{j=i+1}^N \left( \frac{1}{\sum_{l=1}^m \theta_l (x_{il} - x_{jl})^2} \right)^{p/2} \quad (2)$$

for finding the optimal design with  $\boldsymbol{\theta} = (\theta_1, \dots, \theta_m)'$  following a uniform distribution. The following theorem justifies that our adopted criterion is the modified maximin criterion in expectation.

**Proposition 1.** Suppose  $\mathbf{X} = (x_{il})_{N \times m}$  is a design on the subspace  $S_b = \{\mathbf{x} \in \mathcal{R}^m : \sum_{l=1}^m x_l = 1, 0 \leq x_l \leq 1, l = 1, 2, \dots, m\}$ . For  $f(\mathbf{X}|\boldsymbol{\theta})$  in (2) with  $p = 2m$  and  $\boldsymbol{\theta}$  following a uniform distribution in the region  $H = \{\boldsymbol{\theta} : 0 \leq \theta_l \leq 1, \sum_{l=1}^m \theta_l = 1\}$ . Then

$$\begin{aligned} E_{\boldsymbol{\theta}}(f(\mathbf{X}|\boldsymbol{\theta})) &= \int_H f(\mathbf{X}|\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= C \sum_{i=1}^{N-1} \sum_{j=i+1}^N \frac{1}{\prod_{l=1}^m (x_{il} - x_{jl})^2}, \end{aligned}$$

where  $C$  is a constant.

To implement the optimization in (1), one would use a nonlinear optimization with the number of parameters represented as  $Nm$ . Thus, the objective function can be very complex and the optimization process can be hard to execute. Although efficient candidate-generation methods exist (e.g., Huang, Joseph, and Ray (2021)), they often focus on inequality-constrained and full-dimensional domains, but not for our equality-constrained simplex. Therefore, we adopt the coordinate exchange from discrete optimization and coordinate descent from continuous optimization to solve the equality-constrained optimization efficiently. Algorithm 1 summarizes the procedure. Specifically, we randomly choose  $N$  runs from the candidate set  $\mathcal{A}$  as the initial design. Then we can find one run in the initial design that has the maximum  $\sum_{i \neq j} \prod_{l=1}^m \frac{1}{(x_{il} - x_{jl})^2}$ . Replace this run with one run in the candidate set  $\mathcal{A}$ . If the criterion is reduced, we can further optimize the criterion by traditional constraint optimization; If not, replace this run with another run in the candidate set until one run can reduce the criterion.

#### Algorithm 1 Optimization Procedure

**Input:** Candidate set  $\mathcal{A}$ , the number of design runs  $N$ , the number of redundant iterations  $t$ , the objective function  $f(\cdot)$  given by Eq. [1]

**Output:** Final design  $\mathbf{X}$

- 1: Let  $t = 0$ . Randomly choose  $N$  runs from the candidate set  $\mathcal{A}$  as the initial design  $\mathbf{X}$ . Initialize  $\mathbf{X}_{new}$ , s.t.  $|f(\mathbf{X}) - f(\mathbf{X}_{new})| > \epsilon$ ,
- 2: **while**  $t \leq 10000$  **do**
- 3: Replace one row of  $\mathbf{X}$  with one run  $x_a$  from  $\mathcal{A}$ , denote the new design as  $\mathbf{X}_{new}$ .
- 4: **if**  $f(\mathbf{X}) > f(\mathbf{X}_{new})$  **then**
- 5: Constraint continuous optimization to replace  $x_a$  with  $x_{opt}$ , update  $\mathbf{X}_{new}$ .
- 6: **if**  $(|f(\mathbf{X}) - f(\mathbf{X}_{new})| < \epsilon)$  **then**

```

7:       $X = X_{new}$ ; algorithm converges and break
      the loop.
8:  else
9:       $X = X_{new}$ .
10:  end if
11: else
12:      $t = t + 1$ 
13: end if
14: end while
15: return  $X$ 

```

Note that [Algorithm 1](#) requires a good candidate set in the constraint space. A proper candidate set should have points distributed uniformly on the subspace. A naive approach is to generate a space-filling design in the hypercube  $S = \{\mathbf{x} \in \mathcal{R}^m : 0 \leq x_l \leq 1, l = 1, 2, \dots, m\}$  and project the points into the hyperplane  $S_b = \{\mathbf{x} \in \mathcal{R}^m : 0 \leq x_l \leq 1, \sum_{l=1}^m x_l = 1\}$ . However, such an approach can have a high rejection rate, especially in the high-dimensional setting, since not all projected points will be in the constraint space  $S_b$ . To address this challenge, we propose an algebraic construction of the candidate set  $\mathcal{A}$  based on the simplex centroid design ([Scheffé 1963](#)).

Let the candidate set  $\mathcal{A}$  consist of all points in the simplex centroid design and points on the segment between the two points of the simplex centroid design. [Figure 2](#) illustrates an example of how we deploy the original candidate set to construct more samples for a candidate set of three dimensions. Note that such a construction of the candidate set is naturally space-filling based on the simplex centroid design, and is also very flexible on the run size and computationally fast. Its space-filling property also enhances robustness against the randomness induced by the initial design. Note that the proposed [Algorithm 1](#) is a combination of the coordinate-exchange algorithm and the gradient descent method. It does rely on the initial design. In our numerical study, it appears that the best result chosen is not that sensitive to the initial design but to the candidate set. One can use

multiple runs of the initial design to make the algorithm more robust to the choice of initial design.

With the space-filling design for the continuous factors  $x_1, \dots, x_m$  with linear constraint, we can then construct a cross array design between  $x_1, \dots, x_m$ , other continuous factors, and categorical factors as the complete design. Here the Latin hypercube design ([McKay, Beckman, and Conover 1979](#)) is used as the design construction for other continuous factors. In the next section, we will detail how we use the proposed design and collected responses to build a surrogate model for studying how the hyper-parameters and data quality factors impact the quality of the MLD algorithm.

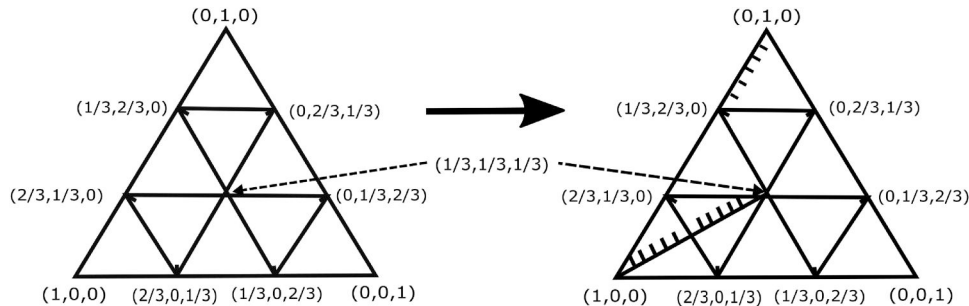
### 3.5. Surrogate modeling

With a large number of factors in our investigation, a quadratic linear regression model can contain too many model parameters while a first-order linear regression model can be too simple to emulate the intricate relationship between the factors and metrics. Therefore, a proper surrogate model is needed. Here, we consider the Gaussian process to be used for surrogate modeling. Note that our DoE contains both continuous factors and discrete factors. To address this challenge, we adopt the additive Gaussian process (AGP) used in [Deng et al. \(2017\)](#) as our surrogate.

Without loss of generality, suppose that the response is  $\tilde{y}$ , and the corresponding covariates are  $\tilde{\mathbf{w}} = (\tilde{\mathbf{x}}, \tilde{\mathbf{z}})$ , where  $\tilde{\mathbf{x}}$  is a continuous covariate vector, and  $\tilde{\mathbf{z}} = (\tilde{z}_1, \dots, \tilde{z}_{\tilde{q}})$  is a binary covariate vector with  $\tilde{q}$  dimensions. Then we consider AGP for modeling the response  $\tilde{y}$  as

$$\tilde{y}(\tilde{\mathbf{w}}) = \mu + \sum_{h=1}^{\tilde{q}} G_h(\tilde{z}_h, \tilde{\mathbf{x}}), \quad (3)$$

where  $G_h$ 's are independent Gaussian processes with mean zero and the covariance function  $\Sigma_h$ 's. Here  $\mu$  is the overall mean, and we set  $\mu = 0$  for simplicity. The



**Figure 2.** An illustration of the usage of 3D simplex centroid design to construct a candidate set. Left: Original 3D simplex centroid design. Right: Add more points between design points in the 3D simplex centroid design.

covariance function of  $G_h$  between a pair of observations  $\tilde{y}_i(\tilde{\mathbf{w}}_i)$  and  $\tilde{y}_j(\tilde{\mathbf{w}}_j)$  is defined as  $\Sigma_h(\tilde{\mathbf{w}}_i, \tilde{\mathbf{w}}_j) = \tau^2 \psi_h(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) \phi_h(\tilde{z}_{hi}, \tilde{z}_{hj})$  where

$$\begin{aligned}\psi_h(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) &= \exp\left(-\frac{\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|^2}{\vartheta_h}\right) + \eta \mathbb{1}(i = j), \\ \phi_h(\tilde{z}_{hi}, \tilde{z}_{hj}) &= \exp((1 - \mathbb{1}(\tilde{z}_{hi} \neq \tilde{z}_{hj})) \log \rho_h),\end{aligned}$$

where  $\mathbb{1}(\cdot)$  is an indicator function. Here  $\vartheta_h \geq 0$  and  $0 \leq \rho_h \leq 1$ . The parameter  $\eta \geq 0$  is a nugget effect. Thus, the overall covariance function becomes  $\Sigma(\cdot, \cdot) = \sum_{h=1}^{\tilde{q}} \Sigma_h(\cdot, \cdot)$ .

When using AGP for our design factors in Section 3.3, the continuous covariate vector  $\tilde{\mathbf{x}}$  contains  $x_1, \dots, x_m$  and  $z_1, \dots, z_3$ . Recall that the categorical factor  $z_4$  has  $k$  levels as  $k$  various types of datasets. While we consider the binary covariate vector  $\tilde{\mathbf{z}} = (\tilde{z}_1, \dots, \tilde{z}_{\tilde{q}})$  for AGP with  $\tilde{z}_h$  to be

$$\tilde{z}_h = \begin{cases} 0, & \text{DS}_1 \text{ is used;} \\ 1, & \text{DS}_{h+1} \text{ is used.} \end{cases}$$

where  $h = 1, 2, \dots, k-1$ . Now assume that the collected data are  $(\tilde{\mathbf{y}}, \tilde{\mathbf{W}})$ , where  $\tilde{\mathbf{y}}$  is the  $N$  dimensional vector of responses and  $\tilde{\mathbf{W}}$  the corresponding covariate matrix. Given a new design point  $\tilde{\mathbf{w}}_*$ , its corresponding response  $\tilde{y}_*$  can have

$$\begin{pmatrix} \tilde{\mathbf{y}} \\ \tilde{y}_* \end{pmatrix} \sim \mathcal{N}_{N+1} \left( \begin{pmatrix} \vec{0}_N \\ 0 \end{pmatrix}, \begin{pmatrix} \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}} & \Sigma_{\tilde{\mathbf{y}}, \tilde{y}_*} \\ \Sigma_{\tilde{y}_*, \tilde{\mathbf{y}}} & \Sigma_{\tilde{y}_*, \tilde{y}_*} \end{pmatrix} \right),$$

where  $\Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}$  is the covariance matrix for  $\tilde{\mathbf{y}}$ ,  $\Sigma_{\tilde{\mathbf{y}}, \tilde{y}_*}$  is the covariance vector between  $\tilde{\mathbf{y}}$  and  $\tilde{y}_*$  ( $\Sigma_{\tilde{y}_*, \tilde{\mathbf{y}}}$  is the covariance vector between  $\tilde{y}_*$  and  $\tilde{\mathbf{y}}$ ), and  $\Sigma_{\tilde{y}_*, \tilde{y}_*}$  is the variance of  $\tilde{y}_*$ . Therefore, it is obvious that the conditional distribution of  $\tilde{y}_*$  given  $\tilde{\mathbf{y}}$  can be obtained for prediction and uncertainty quantification as  $\tilde{y}_* | \tilde{\mathbf{y}} \sim \mathcal{N}(\mu_*, \sigma_*^2)$  where

$$\begin{aligned}\mu_* &= \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \Sigma_{\tilde{\mathbf{y}}, \tilde{y}_*} \tilde{\mathbf{y}}, \\ \sigma_*^2 &= \Sigma_{\tilde{y}_*, \tilde{y}_*} - \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \Sigma_{\tilde{\mathbf{y}}, \tilde{y}_*} \Sigma_{\tilde{y}_*, \tilde{\mathbf{y}}}.\end{aligned}\quad (4)$$

For simplicity, define  $\boldsymbol{\rho} = (\rho_1, \dots, \rho_{\tilde{q}})$  and  $\boldsymbol{\vartheta} = (\vartheta_1, \dots, \vartheta_{\tilde{q}})$ . To estimate unknown parameters  $\boldsymbol{\beta} = (\boldsymbol{\rho}, \boldsymbol{\vartheta}, \eta)$  and  $\tau^2$ , calculate  $\hat{\tau}^2 = \tilde{\mathbf{y}}^T \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \tilde{\mathbf{y}} / N$  firstly and then minimize the negative log-likelihood function given  $\hat{\tau}^2$  as

$$\hat{\boldsymbol{\beta}} = \arg \min_{\boldsymbol{\beta}} \left\{ \log(\det(\Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}})) + N \log(\tilde{\mathbf{y}}^T \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \tilde{\mathbf{y}}) \right\}.$$

The optimization can be solved by the derivative-based method. The estimation process is conducted in an iterative manner. See more details in the Appendix.

## 4. Numerical experiments

### 4.1. Design validation

The validity of the proposed space-filling design for the class proportions (i.e.,  $x_1, \dots, x_m$ ) is examined by comparing it with a benchmark method, the Kennard and Stone algorithm (Kennard and Stone 1969). Here we consider two performance measures “Coverage” and “Maxmin” to evaluate the space-filling property (Gomes, Claeys-Bruno, and Sergent 2018), denote as  $PM_1$  and  $PM_2$  as

$$\begin{aligned}PM_1 &= \frac{1}{\bar{d}} \sqrt{\frac{1}{N} \sum_{i=1}^N (d_i - \bar{d})^2}, \\ PM_2 &= \max(d_i),\end{aligned}$$

where  $d_i = \min_{j \neq i} \|x_i - x_j\|$  and  $\bar{d} = \frac{1}{N} \sum_{i=1}^N d_i$ . The bigger these two metrics are, the better for our purposes. The comparison results are reported in Table 2 with  $m = 10$  classes. It is seen that the proposed design has a consistent advantage over the benchmark method (K-S) in terms of *Coverage*. In terms of *Maxmin*, the proposed design is better than the K-S method when the design runs are relatively large.

### 4.2. Data collection

We conduct data analysis based on additive Gaussian process modeling with respect to prediction accuracy  $y_1$  and detection accuracy  $y_2$ . The range of the transformed weight ratio  $z_1$  is from  $-500$  to  $500$ . The range of  $z_2$  (i.e., the value of  $\alpha$ ) is from  $1$  to  $3$ . The range of the percentage of mislabeled data  $z_3$  is from  $10\%$  to  $50\%$ . Here we use MNIST (Deng 2012) and FashionMNIST (Xiao, Rasul, and Vollgraf 2017) as two types of benchmark datasets. The input images of these two datasets are both  $28 \times 28$ , and both have 10 classes. Thus, the proportions of classes are denoted  $x_1, \dots, x_{10}$ , such that  $0 \leq x_l \leq 1$ ,  $l = 1, \dots, 10$ . For each setting, we construct class-imbalanced splits by stratified sampling with replacement within each class from the original training (and, separately, test) sets so that the resulting class proportions match the target vector

**Table 2.** Compare our approach based on “maxpro” criterion of a subspace with Kennard and Stone (K-S) algorithm.

| Runs ( $N$ ) | Method          | $PM_1$        | $PM_2$        |
|--------------|-----------------|---------------|---------------|
| 50           | Proposed design | <b>0.1509</b> | 0.2962        |
|              | K-S             | 0.0287        | <b>0.3782</b> |
| 100          | Proposed design | <b>0.2149</b> | <b>0.4490</b> |
|              | K-S             | 0.0440        | 0.3403        |
| 150          | Proposed design | <b>0.1852</b> | <b>0.3926</b> |
|              | K-S             | 0.1579        | 0.3768        |
| 200          | Proposed design | <b>0.2099</b> | <b>0.3795</b> |
|              | K-S             | 0.2022        | 0.3672        |

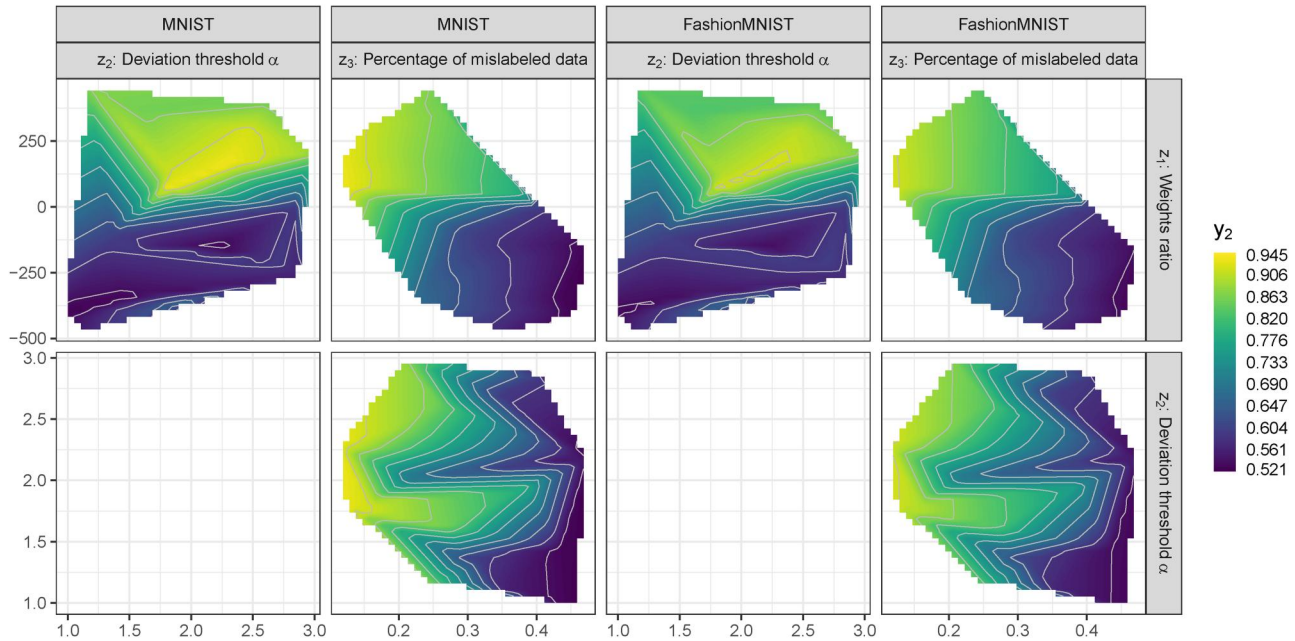
$(x_1, \dots, x_{10})$ , while keeping the total sample sizes fixed. For both datasets, there are 60, 000 observations of the training set and 10, 000 observations of the test set.

Training labels are later corrupted according to the specified noise mechanism. After class-imbalance sampling, we corrupt labels at a rate  $z_3$  using symmetric class-conditional noise. For each class  $i$  with  $n_i$  training samples, we select  $\lfloor z_3 n_i \rfloor$  samples uniformly at random (without replacement) and reassign their labels to a class  $j$  ( $j \neq i$ ) drawn uniformly from the other 9 classes. Validation and test labels remain clean. According to the design we obtained, we run 5 replicates for each setting out of 2, 000 in total (i.e., 10, 000 design runs in total). The detection accuracy and prediction accuracy are collected for further analysis.

#### 4.3. Visualization of findings

Here, we visualize the data from the experimental results with respect to detection accuracy  $y_2$  and prediction accuracy  $y_1$ . Figure 3 displays the contour plots of detection accuracy  $y_2$  with respect to the series of specific weights ratios  $z_1$ , the threshold of deviation  $z_2$ , and the proportion of mislabeled training data  $z_3$ , given the datasets MNIST (Figure 3(A)) and FashionMNIST (Figure 3(B)). Each figure is organized into a  $2 \times 2$  facet grid, and each axis represents one factor for comparison purposes. Across both datasets,

the top-right plot ( $z_1$  vs.  $z_3$ ) shows that the contour lines are generally vertical, indicating that the detection accuracy  $y_2$  is primarily controlled by  $z_3$  rather than  $z_1$ . However, a noticeable shift occurs at  $z_1 < 0$ , and the contour lines become denser and slightly curved, suggesting that the weight ratio begins to have a greater influence on the response. Despite this localized sensitivity to  $z_1$ , the overall trend remains dominated by the proportion of mislabeled data  $z_3$ . In the top-left plot ( $z_1$  vs.  $z_2$ ),  $y_2$  is positively associated with  $z_1$ , but the relationship with  $z_2$  is less straightforward. In the region of larger  $z_1$ , the presence of circular contour lines indicates a local peak or plateau, suggesting that there exists an optimal combination of  $z_1$  and  $z_2$  that maximizes the detection accuracy  $y_2$ . Conversely, at smaller  $z_1$  values, the contour lines are almost horizontal, indicating a negligible effect of  $z_2$  on detection accuracy. The bottom-right plot ( $z_2$  vs.  $z_3$ ) shows a more complex pattern. Detection accuracy  $y_2$  generally decreases with increasing  $z_3$ , but the contours kink sharply as they cross the mid-range of  $z_3$ , and they bunch tightly on the right-hand side, signaling abrupt drops as  $z_3$  exceeds 0.35. Bands of local high  $y_2$  appear at several moderate  $z_2$  values (around 2), suggesting that tuning the deviation threshold can partially mitigate the negative effects of mislabeled data. In terms of the datatype factor,  $z_4$ , the detection performance patterns are similar between the two datasets overall. However, FashionMNIST consistently



**Figure 3.** Detection accuracy  $y_2$  versus weights ratio  $z_1$ , the deviation of threshold  $z_2$ , and proportion of mislabeled training data  $z_3$  for (A) MNIST and (B) FashionMNIST datasets.

exhibits slightly lower peak  $y_2$  values compared to MNIST across the same parameter settings.

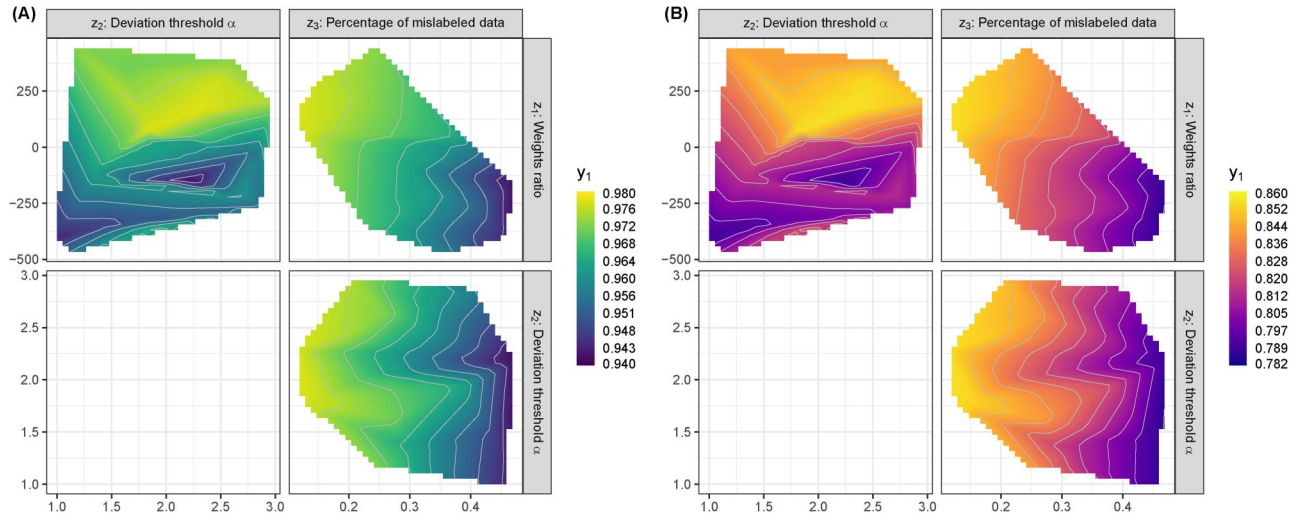
Now, consider prediction accuracy  $y_1$  as the performance measure. The patterns displayed in Figure 4 are similar to the patterns displayed by detection accuracy  $y_2$ . But there are several distinctions between them. First, the contour lines in the top-right plot ( $z_1$  vs.  $z_3$ ) are consistently shaped and gradually spaced across the entire domain, without any abrupt jump or sudden shift around  $z_1 = 0$ . Here, the prediction accuracy declines steadily as  $z_3$  increases, and although  $z_1$  still has a modest influence, its effect transitions smoothly without discontinuity. Second, for the top-left plot ( $z_1$  vs.  $z_2$ ), the drop in prediction accuracy  $y_1$  with decreasing  $z_1$  is more gradual than  $y_2$ . The ridge of high response that appears at large  $z_1$  is broader and flatter than  $y_2$ , suggesting a wider range of  $(z_1, z_2)$  combinations that yield near-optimal  $y_1$ . Third, for the bottom plot ( $z_2$  vs.  $z_3$ ), the prediction accuracy  $y_1$  declines more gradually with increasing  $z_3$ , and the interaction between prediction accuracy  $z_2$  and  $z_3$  is less volatile. Overall, the prediction accuracy  $y_1$  is less sensitive to small changes in either parameter, whereas the detection accuracy  $y_2$  penalizes departures from its optimum much more sharply.

Note that the scale of prediction accuracy of MNIST data is much greater than that of FashionMNIST data. This indicates that the data type  $z_4$  has a profound effect on the prediction accuracy  $y_1$ . When two convolutional neural network (CNN) models with consistent structures are well trained both on the clean training datasets of MNIST and

FashionMNIST, the CNN model trained on the MNIST dataset has a distinctly higher prediction accuracy than the model trained on FashionMNIST. In contrast, the detection accuracy  $y_2$  seems less sensitive to the data type  $z_4$  in Figure 3. This is because the detection algorithm may rely on the differences between mislabeled and correctly labeled data for given classes rather than primary information in the training set, whereas a CNN is more dependent on the primary information.

#### 4.4. Modeling results

To justify the proposed surrogate model, we partition all the experimental data into the training set (80%) and the test set (20%). The AGP is used as the surrogate model for the responses  $y_1$  (i.e., prediction accuracy) and  $y_2$  (i.e., detection accuracy), respectively. The parameters of AGP are estimated through the maximum likelihood estimation. Such a modeling framework allows for inference on the parameters and uncertainty quantification, such as confidence intervals. We have  $h = 1$  and  $\bar{q} = 1$ , because two datasets are used in the experiments. Consequently,  $\tau$  and  $\rho_{\bar{q}}$  can be regarded as one parameter  $\zeta = \tau \cdot \rho_{\bar{q}}$  in the estimation process. The maximum likelihood estimations of  $(\theta_1, \eta, \zeta)$  for  $y_1$  and  $y_2$  are  $(72.644, 0.001, 0.396)$  and  $(14.395, 0.002, 0.934)$  respectively. For comparison, we employ a linear regression model and a random forest model as two benchmarks. To evaluate the accuracy of the AGP model, we utilize the mean square error (MSE) and the Nash-Sutcliffe efficiency (NSE) (Kaufman et al. 2011) on the test set



**Figure 4.** Prediction accuracy  $y_1$  versus weights ratio  $z_1$ , the deviation of threshold  $z_2$ , and proportion of mislabeled training data  $z_3$  for (A) MNIST and (B) FashionMNIST datasets.

as metrics for assessing goodness-of-fit. The NSE is defined as

$$NSE = 1 - \frac{\sum_{\tilde{\mathbf{w}} \in \mathcal{W}_{pred}} (\hat{y}(\tilde{\mathbf{w}}) - \tilde{y}(\tilde{\mathbf{w}}))^2}{\sum_{\tilde{\mathbf{w}} \in \mathcal{W}_{pred}} (\tilde{y}(\tilde{\mathbf{w}}) - \bar{y})^2}, \quad (5)$$

where  $\hat{y}(\tilde{\mathbf{w}})$  and  $\tilde{y}(\tilde{\mathbf{w}})$  are the predicted response and underlying response, and  $\bar{y}$  is the average of the underlying response. As illustrated in Table 3, it is evident that the AGP models significantly outperform the linear regression and random forest models in predicting detection accuracy  $y_2$ . Moreover, the AGP model demonstrates comparable predictive ability to

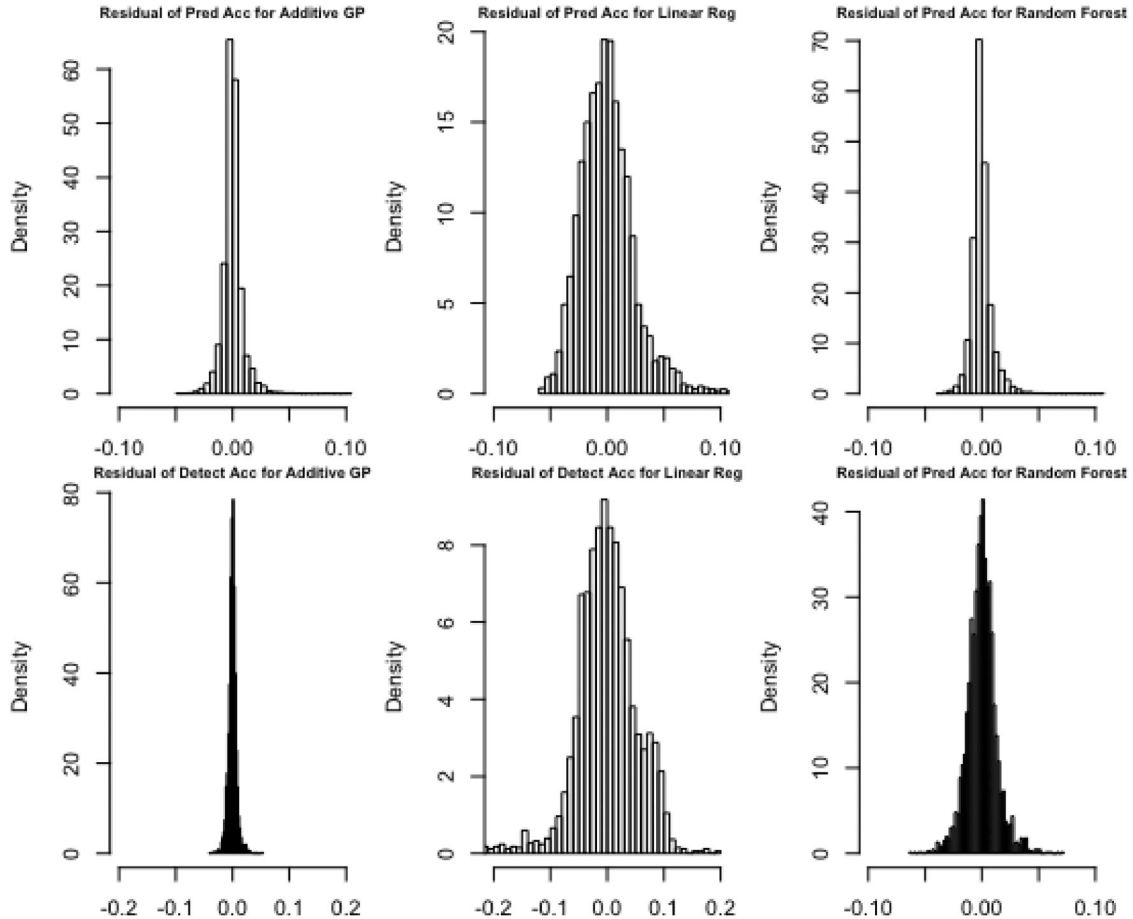
**Table 3.** Comparison results of the additive Gaussian process (AGP) model, linear regression (LM) and random forest (RF) models with respect to mean square error ( $MSE_1$  for  $y_1$  and  $MSE_2$  for  $y_2$ ) and Nash-Sutcliffe efficiency ( $NSE_1$  for  $y_1$  and  $NSE_2$  for  $y_2$ ).

| Metric  | AGP                     | LM     | RF                      |
|---------|-------------------------|--------|-------------------------|
| $MSE_1$ | $8.7147 \times 10^{-5}$ | 0.0028 | $8.3581 \times 10^{-5}$ |
| $NSE_1$ | 0.9864                  | 0.9019 | 0.9854                  |
| $MSE_2$ | $6.1186 \times 10^{-5}$ | 0.0006 | 0.0002                  |
| $NSE_2$ | 0.9980                  | 0.8996 | 0.9938                  |

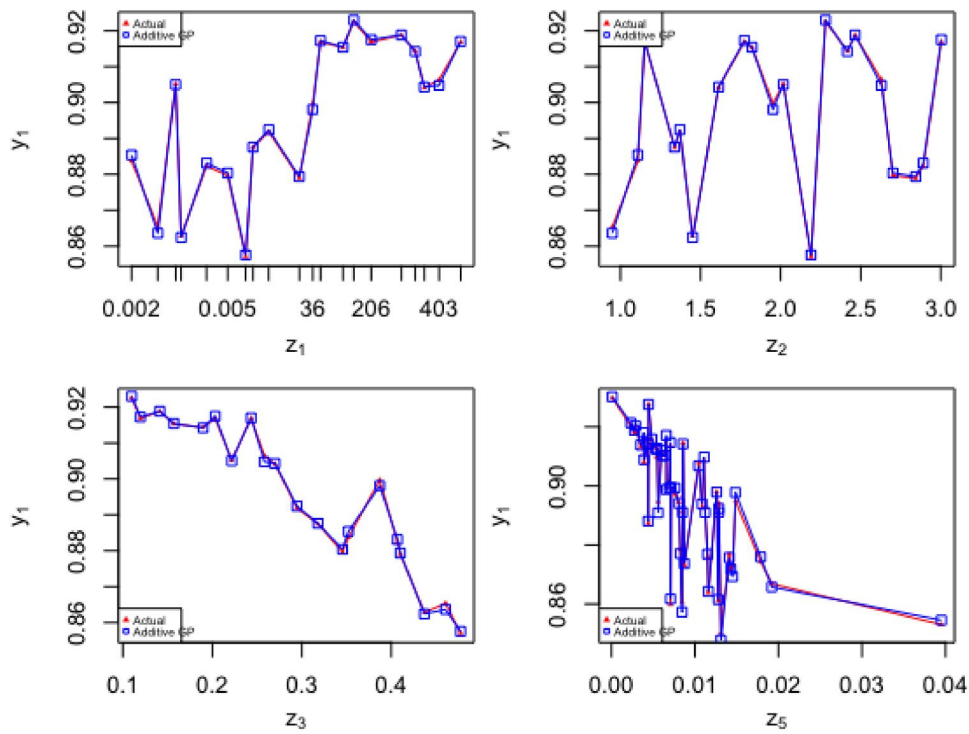
the random forest model and outperforms linear regression model.

Furthermore, this evidence is also supported by Figure 5, which compares the distributions of residuals of AGP, linear regression, and random forest models with respect to  $y_1$  and  $y_2$ . The residual distributions of AGP are much narrower than the corresponding linear regression models. The width of the residual distribution for AGP model is similar to the shape of the residual distribution for random forest model with respect to  $y_1$ , while the width of the residual distribution of AGP model is smaller than random forest model.

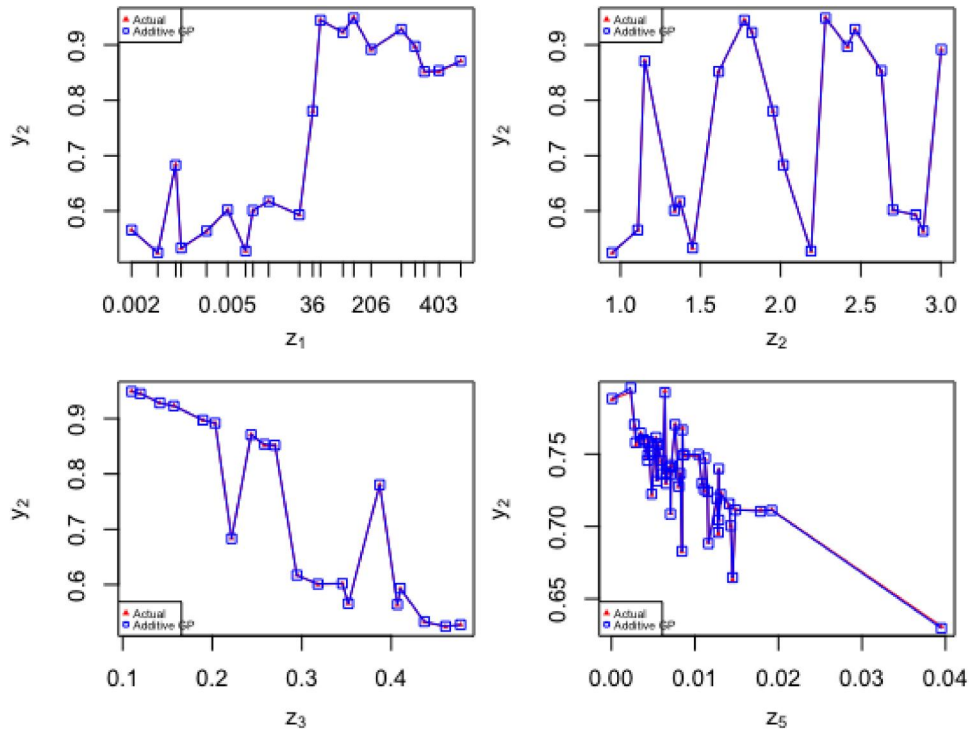
In Figure 6, we compare the average predicted response  $\hat{y}_1$  and underlying response  $\tilde{y}_1$  over weights ratio  $z_1$ , the deviation threshold  $z_2$ , percentage of mis-labeled data  $z_3$ , and variance of proportions of classes, denoted as  $z_5$ , respectively. The visual representation in this figure utilizes red triangles to mark the average experimental results while the blue squares represent the average AGP predictions. The observation of these red triangles overlapping with blue squares indicates a significant consistency in how the AGP method



**Figure 5.** Residual distributions of additive Gaussian (AGP) (left), linear regression model (linear reg) (mid), and random Forest (right) models with respect to prediction accuracy  $y_1$  (pred acc) and detection accuracy  $y_2$  (detect acc).



**Figure 6.** Comparison of average AGP outputs with the actual value of prediction accuracy  $y_1$  versus weights ratio  $z_1$ ,  $\alpha z_2$ , percentage of mislabeled data  $z_3$ , and variance of proportions of classes  $z_5$ .



**Figure 7.** Compare average AGP outputs with the actual value of detection accuracy  $y_2$  versus weights ratio  $z_1$ ,  $\alpha z_2$ , percentage of mislabeled data  $z_3$ , and variance of proportions of classes  $z_5$ .

predicts the actual response surface. It also reflects the AGP's quantification on the true underlying patterns or relationships between inputs and  $y_1$ . For Figure 7, its focus is on comparing the average predicted

response  $\hat{y}_2$  and underlying response  $\tilde{y}_2$  over the same inputs as Figure 6. Similar to the previous figure, the overlap between blue squares and red triangles demonstrates that the AGP method maintains its

effectiveness in accurately modeling the actual response surface for  $y_2$ . This overlapping indicates that the AGP method consistently performs well in terms of both metrics, showing its adaptability and robustness across various scenarios. This evidence is further demonstrated through separate comparisons conducted on MNIST and FashionMNIST test samples, as detailed in [Appendix D](#).

#### 4.5. Summary of findings

By comparing the results from the results of data visualization and the examination of the AGP modeling performance, we have found several interesting insights for the mislabeling detection algorithm studied in this article.

First, it is not surprising that the proportions of mislabeled data significantly affect the performance of detection accuracy. Moreover, changes in class imbalance can also diminish the detection accuracy. However, not all data quality factors have profound effects on the detection accuracy. In our study with two types of benchmark datasets (MNIST and FashionMNIST), the detection accuracy is robust against the type of data used in the proposed method. We observe that the optimal setting of two hyper-parameters, weights ratio  $z_1$  and threshold  $z_2$  are quite similar when the data types (datasets) are different. This indicates that the MLD algorithm has a similar capacity to derive the differences between malicious items and normal items when the datasets are not the same and the hyper-parameters of the detection algorithm do not have an interaction effect with data type. Second, the MLD algorithm turns conservative (detection accuracy in low variance) when its capacity to detect mislabeled data is poor. When the detection accuracy is relatively high, in our cases, roughly larger than 0.7, the stability and the average performance of the detection algorithm are positively associated. Third, the detection accuracy and the classification accuracy can be of different characteristics. A combination of both metrics can help distinguish if the algorithm is affected by the primary information in the dataset (data type factor).

### 5. Discussion and conclusion

In this work, we establish a QE-AI framework to comprehensively investigate how data quality factors (e.g., data type, percentage of mislabeled data, class imbalance in training data) and hyper-parameters of algorithms affect the quality of the AI algorithms. The

proposed QE-AI framework can set an evaluation protocol for AI algorithm practitioners to evaluate the quality of a set of AI algorithms, and enhance AI transparency with understanding of the uncertainty of AI algorithms.

There could be some limitations of the current work. First, we chose a mislabeled data detection algorithm and then evaluate it on two similar datasets to describe the proposed framework. However, the QE-AI framework is not limited to this particular MLD algorithm. It can be extended to other AI algorithms whose quality assurance is affected by various data quality factors and their internal structure. Besides, the evidence provided in the summary may need further validation across a broader range of usage scenarios in other datasets. Second, the number of classes in this study is around ten, not as large as the hundreds present in some benchmark data. For data with a large number of classes, such as the CIFAR100 dataset with 100 classes, the proposed design construct may encounter certain computational difficulties. Third, the AGP modeling technique does not inherently accommodate constraints on the inputs. Although this limitation does not substantially compromise the AGP's predictive performance, it will be interesting to examine such a limitation in a rigorous manner.

For future works, one can consider a sequential design based on surrogate modeling to handle the sensitivity concern and limitation of design runs. Note that we currently consider a cross array between the data quality factors and algorithm hyper-parameter factors. The cross array can lead to a large design size, which may not be very realistic in practice. The computation time of training the AGP model with a large run size can take several hours. To mitigate this issue, one can adopt the ideas of sliced Latin hypercube design to construct a small-size design with good projection properties (Ba, Myers, and Breneman 2015; Deng, Hung, and Lin 2015). Moreover, one can also consider the sparse Gaussian process to enhance the computational efficiency of surrogate modeling. Additionally, it is appealing to extend the QE-AI framework to handle more factors related data quality or hyperparameters for the other interesting investigations, such as multiple mislabeling detection algorithms on multiple types of datasets with different numbers of classes. For models that incorporate network architecture, parameters pertaining to the architecture can be considered as inputs to be integrated into the framework for gauging their effects.

## About the authors

**Jiayi Lian** is a former PhD student in the department of statistics at Virginia Tech. He recently joined in Wells Fargo as a quantitative analytics specialist.

**Kexin Xie** is a current PhD student in the department of statistics at Virginia Tech.

**Kevin Choi** is a data scientist at AI Center of Excellence (AI CoE) of Deloitte & Touche LLP.

**Xueying Liu** is a current PhD student in the department of statistics at Virginia Tech.

**Balaji Veeramani** is Associate Vice President at AI Center of Excellence (AI CoE) of Deloitte & Touche LLP.

**Sathvik Murli** is Lead Data Scientist/AI Engineer at AI Center of Excellence (AI CoE) of Deloitte & Touche LLP.

**Alison Hu** is a managing director at AI Center of Excellence (AI CoE) of Deloitte & Touche LLP.

**Laura Freeman** is a Professor of Practice in the department of statistics at Virginia Tech.

**Edward Bowen** is a managing director at AI Center of Excellence (AI CoE) of Deloitte & Touche LLP.

**Xinwei Deng** is a Professor in the department of statistics at Virginia Tech.

## Author contributions

CRedit: **Jiayi Lian**: Data curation, Formal analysis, Investigation, Methodology, Validation, Writing – original draft, Writing – review & editing; **Kexin Xie**: Data curation, Formal analysis, Investigation, Validation, Visualization; **Kevin Choi**: Conceptualization, Supervision, Validation, Writing – review & editing; **Xueying Liu**: Investigation, Supervision, Validation; **Balaji Veeramani**: Data curation, Investigation, Supervision, Validation, Writing – review & editing; **Sathvik Murli**: Investigation, Validation, Visualization, Writing – review & editing; **Alison Hu**: Investigation, Project administration, Supervision; **Laura Freeman**: Conceptualization, Supervision; **Edward Bowen**: Conceptualization, Investigation, Supervision, Validation; **Xinwei Deng**: Conceptualization, Investigation, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## ORCID

Xinwei Deng  <http://orcid.org/0000-0002-1560-2405>

## References

Ba, S., W. R. Myers, and W. A. Brennenman. 2015. Optimal sliced Latin hypercube designs. *Technometrics* 57 (4):479–87. doi: [10.1080/00401706.2014.957867](https://doi.org/10.1080/00401706.2014.957867).

Balestrassi, P. P., E. Popova, A. D. Paiva, and J. M. Lima. 2009. Design of experiments on neural network's training for nonlinear time series forecasting. *Neurocomputing* 72 (4–6):1160–78. doi: [10.1016/j.neucom.2008.02.002](https://doi.org/10.1016/j.neucom.2008.02.002).

Bell, S. J., O. P. Kampman, J. Dodge, and N. D. Lawrence. 2022. *Modeling the machine learning multiverse*.

Ben Fredj, O., A. Mihoub, M. Krichen, O. Cheikhrouhou, and A. Derhab. 2020. Cybersecurity attack prediction: A deep learning approach. In *13th International Conference on Security of Information and Networks, SIN 2020*, 1–6. Association for Computing Machinery.

Bernardo, J., J. Berger, A. Dawid, and A. Smith. 1998. Regression and classification using Gaussian process priors. *Bayesian Statistics* 6:475.

Cardelli, L., M. Kwiatkowska, L. Laurenti, and A. Patane. 2019. Robustness guarantees for Bayesian inference with Gaussian processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:7759–68.

Chen, J., D. Zhou, J. Yi, and Q. Gu. 2020. A Frank-Wolfe framework for efficient and effective adversarial attacks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:3486–94.

Cornell, J. A. 2011. *Experiments with mixtures: Designs, models, and the analysis of mixture data*, vol. 403. Hoboken, NJ: John Wiley & Sons.

Deng, L. 2012. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29 (6):141–2.

Deng, X., C. D. Lin, K.-W. Liu, and R. Rowe. 2017. Additive Gaussian process for computer models with qualitative and quantitative factors. *Technometrics* 59 (3): 283–92. doi: [10.1080/00401706.2016.1211554](https://doi.org/10.1080/00401706.2016.1211554).

Deng, X., Y. Hung, and C. D. Lin. 2015. Design for computer experiments with qualitative and quantitative factors. *Statistica Sinica* 1567–81. doi: [10.5705/ss.2013.388](https://doi.org/10.5705/ss.2013.388).

Dietterich, T. G. 2017. Steps toward robust artificial intelligence. *AI Magazine* 38 (3):3–24. doi: [10.1609/aimag.v38i3.2756](https://doi.org/10.1609/aimag.v38i3.2756).

Garrido-Merchán, E. C., and D. Hernández-Lobato. 2020. Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes. *Neurocomputing* 380:20–35. doi: [10.1016/j.neucom.2019.11.004](https://doi.org/10.1016/j.neucom.2019.11.004).

Gomes, C., M. Claeys-Bruno, and M. Sergeant. 2018. Space-filling designs for mixtures. *Chemometrics and Intelligent Laboratory Systems* 174:111–27. doi: [10.1016/j.chemolab.2018.01.013](https://doi.org/10.1016/j.chemolab.2018.01.013).

Guan, D., W. Yuan, Y.-K. Lee, and S. Lee. 2011. Identifying mislabeled training data with the aid of unlabeled data. *Applied Intelligence* 35 (3):345–58. doi: [10.1007/s10489-010-0225-4](https://doi.org/10.1007/s10489-010-0225-4).

Horn, R. A., and C. R. Johnson. 2012. *Matrix analysis*. Cambridge University Press.

Huang, C., V. R. Joseph, and D. M. Ray. 2021. Constrained minimum energy designs. *Statistics and Computing* 31 (6):80. doi: [10.1007/s11222-021-10054-2](https://doi.org/10.1007/s11222-021-10054-2).

Joseph, V. R. 2016. Space-filling designs for computer experiments: A review. *Quality Engineering* 28 (1):28–35. doi: [10.1080/08982112.2015.1100447](https://doi.org/10.1080/08982112.2015.1100447).

Joseph, V. R., E. Gul, and S. Ba. 2015. Maximum projection designs for computer experiments. *Biometrika* 102 (2): 371–80. doi: [10.1093/biomet/asv002](https://doi.org/10.1093/biomet/asv002).

Jung, J.-R., and B.-J. Yum. 2011. Artificial neural network based approach for dynamic parameter design. *Expert Systems with Applications* 38 (1):504–10. doi: [10.1016/j.eswa.2010.06.092](https://doi.org/10.1016/j.eswa.2010.06.092).

- Kaufman, C. G., D. Bingham, S. Habib, K. Heitmann, and J. A. Frieman. 2011. *Efficient emulators of computer experiments using compactly supported correlation functions, with an application to cosmology*.
- Kennard, R. W., and L. A. Stone. 1969. Computer aided design of experiments. *Technometrics* 11 (1):137–48. doi: [10.1080/00401706.1969.10490666](https://doi.org/10.1080/00401706.1969.10490666).
- Kingma, D. P., and M. Welling. 2013. *Auto-encoding variational Bayes*.
- Lian, J., L. Freeman, Y. Hong, and X. Deng. 2021. Robustness with respect to class imbalance in artificial intelligence classification algorithms. *Journal of Quality Technology* 53 (5):505–25. doi: [10.1080/00224065.2021.1963200](https://doi.org/10.1080/00224065.2021.1963200).
- Ling, C. K., K. H. Low, and P. Jaillet. 2016. Gaussian process planning with lipschitz continuous reward functions: Towards unifying Bayesian optimization, active learning, and beyond. In *Proceedings of the AAAI Conference on Artificial Intelligence* 30 (1). doi: [10.1609/aaai.v30i1.10210](https://doi.org/10.1609/aaai.v30i1.10210).
- Ma, Y., X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha. 2019. Trafficpredict: Trajectory prediction for heterogeneous traffic-agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6120–7.
- Malossini, A., E. Blanzieri, and R. T. Ng. 2006. Detecting potential labeling errors in microarrays by data perturbation. *Bioinformatics (Oxford, England)* 22 (17):2114–21. doi: [10.1093/bioinformatics/btl346](https://doi.org/10.1093/bioinformatics/btl346).
- McKay, M., R. Beckman, and W. Conover. 1979. A comparison of three methods for selection values of input variables in the analysis of output from a computer code. *Technometrics* 21 (2):239–45.
- Mutny, M., J. Kirschner, and A. Krause. 2020. Experimental design for optimization of orthogonal projection pursuit models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:10235–42.
- Packianather, M., P. Drake, and H. Rowlands. 2000. Optimizing the parameters of multilayered feedforward neural networks through Taguchi design of experiments. *Quality and Reliability Engineering International* 16 (6):461–73. doi: [10.1002/1099-1638\(200011/12\)16:6<461::AID-QRE341>3.0.CO;2-G](https://doi.org/10.1002/1099-1638(200011/12)16:6<461::AID-QRE341>3.0.CO;2-G).
- Patrini, G., A. Rozza, A. Krishna Menon, R. Nock, and L. Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1944–52.
- Piepel, G. F., and J. A. Cornell. 1994. Mixture experiment approaches: Examples, discussion, and recommendations. *Journal of Quality Technology* 26 (3):177–96. doi: [10.1080/00224065.1994.11979525](https://doi.org/10.1080/00224065.1994.11979525).
- Pulastya, V., G. Nuti, Y. K. Atri, and T. Chakraborty. 2021. Assessing the quality of the datasets by identifying mislabeled samples. In *Proceedings of the 2021 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM '21*, 18–22. Association for Computing Machinery.
- Rushby, J. 1988. *Quality measures and assurance for AI software*, vol. 18. National Aeronautics and Space Administration, Scientific and Technical Information Division.
- Russell, S., D. Dewey, and M. Tegmark. 2015. Research priorities for robust and beneficial artificial intelligence. *AI Magazine* 36 (4):105–14. doi: [10.1609/aimag.v36i4.2577](https://doi.org/10.1609/aimag.v36i4.2577).
- Scheffé, H. 1963. The simplex-centroid design for experiments with mixtures. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 25 (2):235–51. doi: [10.1111/j.2517-6161.1963.tb00506.x](https://doi.org/10.1111/j.2517-6161.1963.tb00506.x).
- Staelin, C. 2003. Parameter selection for support vector machines. Hewlett-Packard Company, Tech. Rep. HPL-2002-354R1, 1.
- Virani, N., N. Iyer, and Z. Yang. 2020. Justification-based reliability in machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:6078–85.
- Vu, H., T. D. Nguyen, T. Le, W. Luo, and D. Phung. 2019. Robust anomaly detection in videos using multilevel representations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:5216–23.
- Xiao, H., K. Rasul, and R. Vollgraf. 2017. *Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms*.

## Appendix A. Proof of proposition 1

**Proof.** Let  $p = 2m$  and  $\theta$  follow a uniform distribution in the region  $H = \{\theta : 0 \leq \theta_l \leq 1, \sum_{l=1}^m \theta_l = 1\}$ . Then  $dp(\theta) = \frac{(m-1)!}{\sqrt{m}} d\theta$ .

$$\begin{aligned}
 E_{\theta}(f(X|\theta)) &= \int_H f(X|\theta) dp(\theta) \\
 &= \int_H f(X|\theta) \frac{(m-1)!}{\sqrt{m}} d\theta \\
 &= \int_{S_{m-1}} \sqrt{mf} \left( X|\theta_1, \dots, \theta_{m-1}, 1 - \sum_{l=1}^{m-1} \theta_l \right) \\
 &\quad \times \frac{(m-1)!}{\sqrt{m}} d\theta_1 \dots d\theta_{m-1} \\
 &= (m-1)! \int_{S_{m-1}} f \left( X|\theta_1, \dots, \theta_{m-1}, 1 - \sum_{l=1}^{m-1} \theta_l \right) \\
 &\quad d\theta_1 \dots d\theta_{m-1} \\
 &= (m-1)! \sum_{i=1}^{N-1} \sum_{j=i+1}^N \int_{S_{m-1}} f_{(i,j)}(X|\theta_1, \dots, \theta_{m-1}) \\
 &\quad d\theta_1 \dots d\theta_{m-1},
 \end{aligned}$$

where  $S_{m-1} = \{\theta : 0 \leq \theta_1, \theta_2, \dots, \theta_{m-1} \leq 1, \sum_{l=1}^{m-1} \theta_l \leq 1\}$ . Here the expression of  $f_{(i,j)}(X|\theta_1, \dots, \theta_{m-1}) = \left( \sum_{l=1}^{m-1} \theta_l d_{(i,j)l} + \left( 1 - \sum_{l=1}^{m-1} \theta_l \right) d_{(i,j)m} \right)^{-m}$ ,  $d_{(i,j)l} = (x_{il} - x_{jl})^2$ . Suppose

$$Q_m(m, a) = \int_{S_{m-1}} \left( \sum_{l=1}^{m-1} \theta_l d_l + \left( 1 - \sum_{l=1}^{m-1} \theta_l \right) a \right)^{-m} d\theta_1 \dots d\theta_{m-1}.$$

For  $a \neq d_{m-1}$ ,  $Q_m(m, a) = \frac{1}{(m-1)(a-d_{m-1})} (Q_{m-1}(m-1, d_{m-1}) - Q_{m-1}(m-1, a))$ . It is easy to see that  $Q_2(2, a) = 1/(d_1 a)$ . Therefore,  $Q_m(m, a) = 1/\{(m-1)! d_1 \dots d_{m-1} a\}$ , then  $Q_m(m, d_m) = 1/\{(m-1)! d_1 \dots d_{m-1} d_m\}$  (Joseph, Gul, and Ba 2015). Here we prove that the expected objective function over  $H = \{\theta : 0 \leq \theta_l \leq 1, \sum_{l=1}^m \theta_l = 1\}$  is a scalar of the the expected objective function over  $S_{m-1} = \{\theta : 0 \leq \theta_1, \theta_2, \dots, \theta_{m-1} \leq 1, \sum_{l=1}^{m-1} \theta_l \leq 1\}$ . Therefore, the two expected objective function share the same kernel.  $\square$

## Appendix B. Validation of covariance matrix of additive Gaussian process

**Lemma 1** (Schur Product Theorem).

If  $\mathbf{D}$  is an  $N \times N$  positive semidefinite matrix with no diagonal entry equal to zero and  $\mathbf{E}$  is an  $N \times N$  positive definite matrix, then  $\mathbf{D} \circ \mathbf{E}$  is positive definite. If both  $\mathbf{D}$  and  $\mathbf{E}$  are positive definite, then  $\mathbf{D} \circ \mathbf{E}$  is positive definite as well (Horn and Johnson 2012).

**Proposition 2.** For the collected data  $(\tilde{\mathbf{y}}, \tilde{\mathbf{W}} = (\tilde{\mathbf{X}}, \tilde{\mathbf{Z}}))$ , where  $\tilde{\mathbf{X}}$  is the  $N \times \tilde{p}$  continuous covariates matrix and  $\tilde{\mathbf{Z}}$  is the  $N \times \tilde{q}$  binary covariates matrix. Then the covariance matrix of  $\tilde{\mathbf{y}}$  is a positive definite matrix.

*Proof.* The covariance matrix for  $\tilde{\mathbf{y}}$  is

$$\begin{aligned}\Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}} &= \sum_{h=1}^{\tilde{q}} (\Sigma_h(\tilde{\mathbf{w}}_i, \tilde{\mathbf{w}}_j))_{N \times N} \\ &= \sum_{h=1}^{\tilde{q}} (\tau^2 \psi_h(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j) \phi_h(\tilde{z}_{hi}, \tilde{z}_{hj}))_{N \times N} \\ &= \sum_{h=1}^{\tilde{q}} \tau^2 (\psi_h(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j))_{N \times N} \circ (\phi_h(\tilde{z}_{hi}, \tilde{z}_{hj}))_{N \times N},\end{aligned}$$

where  $\circ$  is Schur product,  $i, j = 1, 2, \dots, N$ . Denote  $(\psi_h(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j))_{N \times N}$  as  $\mathbf{C}_h$ ,  $(\phi_h(\tilde{z}_{hi}, \tilde{z}_{hj}))_{N \times N}$  as  $\mathbf{B}_h$ . Note that  $\mathbf{C}_h$ 's are positive definite matrices.  $\mathbf{B}_h$  can be obtained by

$$\mathbf{B}_h = \tilde{\mathbf{A}}_h \begin{pmatrix} 1 & \rho_h \\ \rho_h & 1 \end{pmatrix} \tilde{\mathbf{A}}_h^T,$$

where  $(\tilde{\mathbf{A}}_h)$  is a  $\tilde{q} \times 2$  matrix, and

$$(\tilde{\mathbf{A}}_h)_{i,1} = \begin{cases} 0, & \text{if } \tilde{z}_{hi} = 0; \\ 1, & \text{if } \tilde{z}_{hi} = 1. \end{cases} \quad (\tilde{\mathbf{A}}_h)_{i,2} = \begin{cases} 1, & \text{if } \tilde{z}_{hi} = 0; \\ 0, & \text{if } \tilde{z}_{hi} = 1. \end{cases}$$

Since  $\begin{pmatrix} 1 & \rho_h \\ \rho_h & 1 \end{pmatrix}$  is positive definite, then  $\mathbf{B}_h$  is at least positive semidefinite with all diagonal entries equal to 1. Followed by Lemma 1,  $\mathbf{B}_h \circ \mathbf{C}_h$  is positive definite. Therefore, the covariance matrix  $\Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}$  is positive definite.  $\square$

## Appendix C. Derivatives of the log-likelihood function

Given  $\tilde{\mathbf{y}}$  and  $\tilde{\mathbf{W}}$ , we build the log-likelihood function

$$\begin{aligned}l(\boldsymbol{\beta}, \tau^2) &= -\frac{N}{2} \log(2\pi) - \frac{N}{2} \log(\tau^2) - \frac{1}{2} \log(\det(\Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}})) \\ &\quad - \frac{1}{2\tau^2} \tilde{\mathbf{y}}^T \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \tilde{\mathbf{y}}.\end{aligned}$$

To maximize the log-likelihood function, firstly, we maximize it with respect to  $\tau^2$  given fixed  $\boldsymbol{\beta}$ . Take derivative with respect to  $\tau^2$

$$0 \stackrel{\text{set}}{=} \frac{\partial l(\tau^2 | \boldsymbol{\beta})}{\partial \tau^2} = -\frac{N}{2\tau^2} + \frac{1}{2(\tau^2)^2} \tilde{\mathbf{y}}^T \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \tilde{\mathbf{y}}.$$

Therefore,  $\hat{\tau}^2 = \tilde{\mathbf{y}}^T \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \tilde{\mathbf{y}} / N$ . Then we maximize the log-likelihood given  $\tau^2 = \hat{\tau}^2$ . Take derivatives with respect to  $\boldsymbol{\beta} = (\boldsymbol{\rho}, \boldsymbol{\eta})$

$$\begin{aligned}\frac{\partial \Sigma_h((\tilde{z}_{hi}, \tilde{\mathbf{x}}_i), (\tilde{z}_{hj}, \tilde{\mathbf{x}}_j))}{\partial \boldsymbol{\theta}_h} &= \Sigma_h((\tilde{z}_{hi}, \tilde{\mathbf{x}}_i), (\tilde{z}_{hj}, \tilde{\mathbf{x}}_j)) \\ &\quad \times \frac{\|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|}{\theta_h^2},\end{aligned}$$

$$\frac{\partial \Sigma_h((\tilde{z}_{hi}, \tilde{\mathbf{x}}_i), (\tilde{z}_{hj}, \tilde{\mathbf{x}}_j))}{\partial \eta} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases},$$

$$\frac{\partial \Sigma_h((\tilde{z}_{hi}, \tilde{\mathbf{x}}_i), (\tilde{z}_{hj}, \tilde{\mathbf{x}}_j))}{\partial \rho_h} = \begin{cases} 0 & i = j \\ \Sigma_h((\tilde{z}_{hi}, \tilde{\mathbf{x}}_i), (\tilde{z}_{hj}, \tilde{\mathbf{x}}_j)) & i \neq j \end{cases},$$

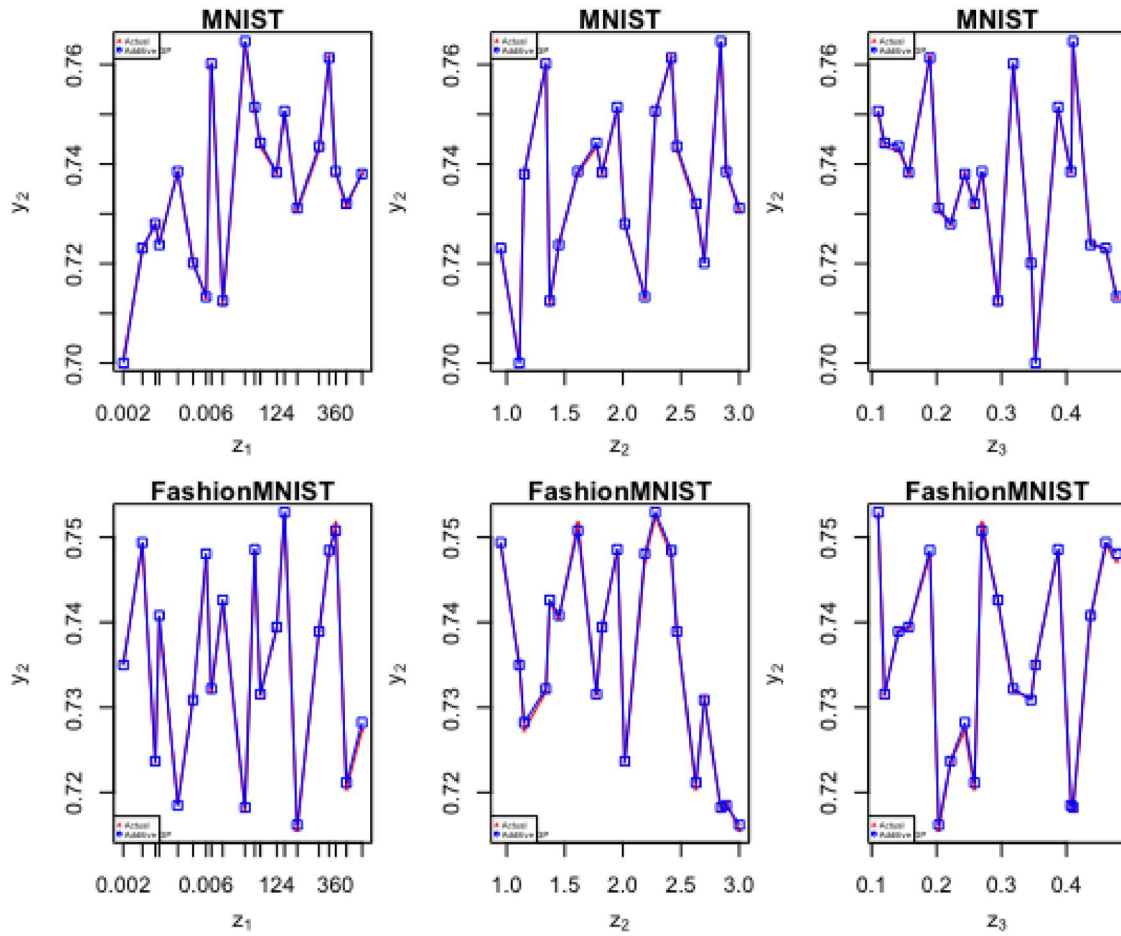
$$\frac{\partial \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1}}{\partial \boldsymbol{\beta}} = -\Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \frac{\partial \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}}{\partial \boldsymbol{\beta}} \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1},$$

$$\frac{\partial \log(\det(\Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}))}{\partial \boldsymbol{\beta}} = \text{tr} \left( \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \frac{\partial \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}}{\partial \boldsymbol{\beta}} \right),$$

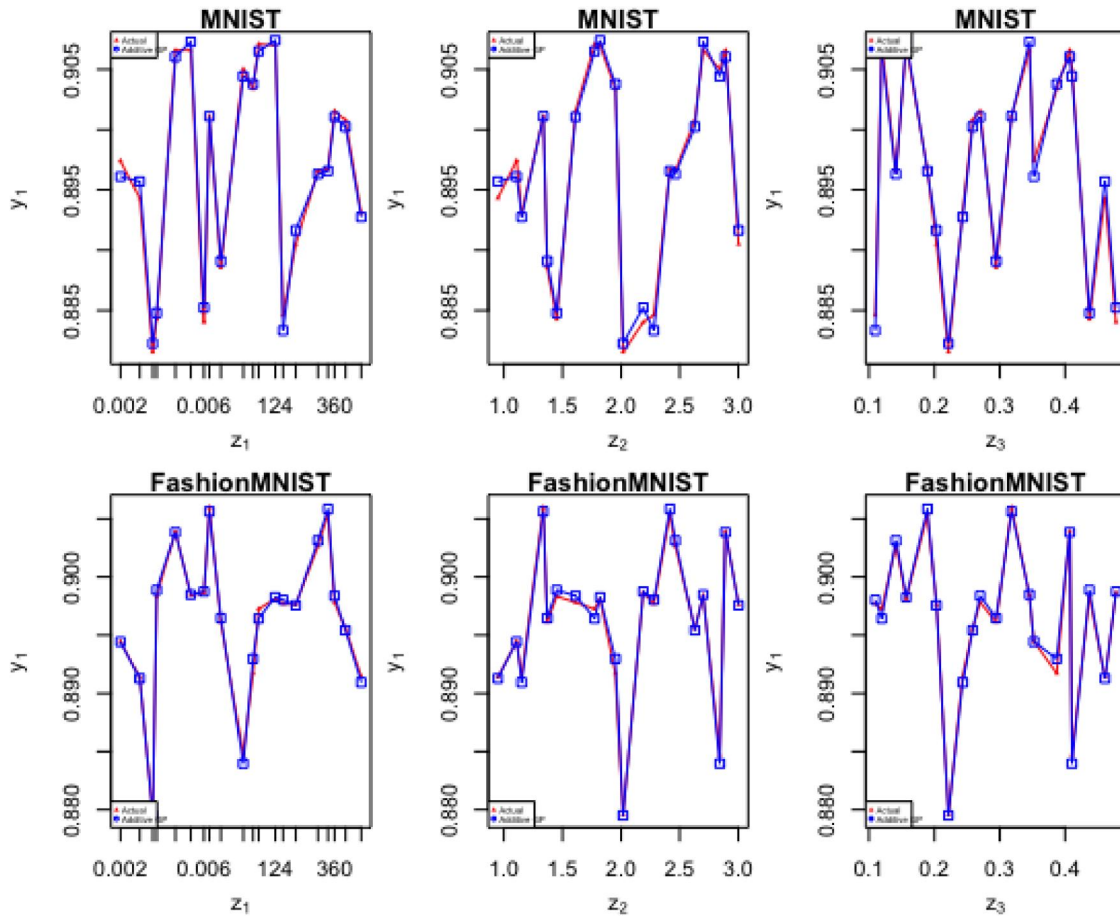
$$\begin{aligned}\frac{\partial l(\boldsymbol{\beta} | \hat{\tau}^2)}{\partial \boldsymbol{\beta}} &= -\frac{1}{2} \text{tr} \left( \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \frac{\partial \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}}{\partial \boldsymbol{\beta}} \right) \\ &\quad + \frac{N}{2} \frac{\tilde{\mathbf{y}}^T \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \frac{\partial \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}}{\partial \boldsymbol{\beta}} \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \tilde{\mathbf{y}}}{\tilde{\mathbf{y}}^T \Sigma_{\tilde{\mathbf{y}}, \tilde{\mathbf{y}}}^{-1} \tilde{\mathbf{y}}},\end{aligned}$$

and obtain  $\hat{\boldsymbol{\beta}}$  by minimizing the negative log-likelihood function given  $\hat{\tau}^2$  through derivative based optimization, such as Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. Repeatedly maximize the log-likelihood function with respect to  $\tau^2$  and  $\boldsymbol{\beta}$  until they converge.

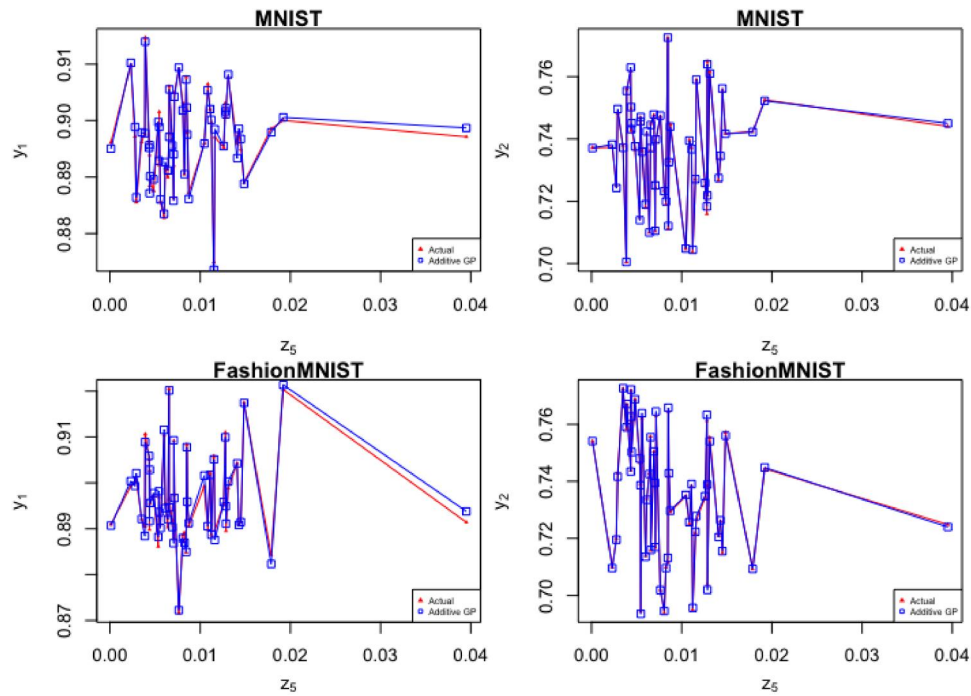
## Appendix D. Compare prediction of AGP with actual experimental results



**Figure D1.** Average predicted detection accuracy  $y_2$  versus weights ratio  $z_1$  (Left), the deviation of threshold  $z_2$  (mid), and proportion of mislabeled training data  $z_3$  (right) for MNIST (upper) and FashionMNIST (lower) datasets.



**Figure D2.** Average predicted prediction accuracy  $y_1$  versus weights ratio  $z_1$  (Left), the deviation of threshold  $z_2$  (mid), and proportion of mislabeled training data  $z_3$  (right) for MNIST (upper) and FashionMNIST (lower) datasets.



**Figure D3.** Average predicted prediction accuracy  $y_1$  (right) and detection accuracy  $y_2$  (Left) versus variance of proportions of classes  $z_5$  for MNIST (upper) and FashionMNIST (lower) datasets.