Supplemental Material of Mathematical Proofs and Four Datasets Used in Main Text for Data organization limits the predictability of binary classification

Fei Jing,[1] Zi-Ke Zhang,[2, *] Yi-Cheng Zhang,[3, †] and Qingpeng Zhang[1, ‡]

[1]*Musketeers Foundation Institute of Data Science, The University of Hong Kong, Hong Kong SAR, China*

[2]*Center for Digital Communication Studies, Zhejiang University, Hangzhou 310058, China*

[3]*Department of Physics, University of Fribourg, Chemin du Musée 3, 1700 Fribourg, Switzerland*

CONTENTS

* zkz@zju.edu.cn
† yi-cheng.zhang@unifr.ch
‡ qpzhang@hku.hk

I. PRELIMINARIES

Binary classification involves assigning elements of a set into one of two groups, or classes, based on a classification rule. This process is crucial in various applications such as medical diagnostics, quality control, and information retrieval. In machine learning and data analysis, binary classification is a supervised learning task aimed at predicting one of two possible outcomes for a given input.

In this context, the data is labeled as positive (often denoted by 1) or negative (denoted by -1). The objective is to construct a predictive model capable of accurately classifying new, unseen instances into these categories by learning from patterns in the training data. Initially, a labeled dataset is gathered, where each instance is associated with a known class label. The dataset is then divided into a training set for model development and a test set for evaluating performance on new data. During training, the model identifies patterns that distinguish between the classes. Techniques such as XGBoost, MLP, SVM, LR, DT, RF, KNN, and Naive Bayes can be utilized for binary classification.

Let's consider a binary classification scenario where the goal is to predict a binary label. An input-output pair is represented as $z = (x, y)$, where $x \in \mathcal{X}$ is the feature vector (input data) and $y \in \{1, -1\}$ is the class label. Given a training set $\mathcal{S} = \{(x_i, y_i) | i = 1, 2, \ldots, m\}$, we define $\mathcal{S}_+$ as the subset containing n_+ positive samples and $\mathcal{S}_-$ as the subset with n_- negative samples, such that the total number of instances is $m = |\mathcal{S}| = n_+ + n_-$. Let $\mathcal{P}(x_i)$ and $\mathcal{N}(x_i)$ denote the counts of positive and negative instances for a given feature vector x_i.

Classifiers are mappings that assign instances to specific classes. Some produce a continuous output, allowing various thresholds to define class membership—these are known as **continuous classifiers**. They combine a classification function $f(x) : \mathcal{X} \to \mathbb{R}$ with a threshold t to translate scores into binary classes. Others yield a discrete class label—these are **discrete classifiers** and are described by the function $f(x) : \mathcal{X} \to \{1, -1\}$. Logistic regression and neural networks are examples of continuous classifiers, while SVM, decision trees, and random forests are discrete classifiers.

Objective functions gauge the alignment between model predictions and actual labels. During training, the aim is to minimize the difference between these predictions and true labels. We discuss several objective functions for binary classification problems:

- Square loss function: $\min\limits_f \ \frac{1}{m} \sum\limits_{x_i} \Big(f(x_i) - y_i \Big)^2$;
- Logistic loss function: $\min\limits_f \ \frac{1}{m} \sum\limits_{x_i} -y_i \log f(x_i) - (1 - y_i) \log \Big(1 - f(x_i) \Big)$;
- Hinge loss function: $\min\limits_f \ \frac{1}{2m} \sum\limits_{x_i} \max \Big\{ 0, 1 - f(x_i) y_i \Big\}$;
- Softmax function: $\min\limits_f \ \frac{1}{m} \sum\limits_{x_i} - \log f(x_i) - \log \Big(1 - f(x_i) \Big)$.

Square and Logistic loss functions are typically suited for continuous classifiers, while Hinge and Softmax losses can be applied to both continuous and discrete classifiers.

For binary classification outcomes, we consider the instances to be either positive or negative, leading to four potential results from the classifier:

- TP (True Positive): $\sum\limits_{x_i \in \mathcal{S}} \frac{\mathcal{P}(x_i)}{2} \Big(f(x_i) + 1 \Big)$;
- FP (False Positive): $\sum\limits_{x_i \in \mathcal{S}} \frac{\mathcal{N}(x_i)}{2} \Big(f(x_i) + 1 \Big)$;
- FN (False Negative): $\sum\limits_{x_i \in \mathcal{S}} \frac{\mathcal{P}(x_i)}{2} \Big(1 - f(x_i) \Big)$;
- TN (True Negative): $\sum\limits_{x_i \in \mathcal{S}} \frac{\mathcal{N}(x_i)}{2} \Big(1 - f(x_i) \Big)$.

II. OBJECTIVE FUNCTIONS

The objective function in the training process of a classification model serves as a guide for parameter adjustment to fit the dataset. Here, we discuss four commonly used objective functions and show a direct correlation between their optimal solutions [1] and the dataset through discrete analysis.

A. Optimal square loss and statistical ensemble on square cost function

1. Optimal square loss

Given a dataset $\mathcal{S}$ with feature domain $\mathcal{X}$, and assuming $f(x)$ as a continuous classification function with parameters to be trained, the square error loss function is given by:

$$\min_f \frac{1}{m}\sum_{x_i}\Big(f(x_i)-y_i\Big)^2. \tag{1}$$

The optimal solution for this objective function is expressed as:

$$f^*_{\text{Square}} = \arg\min_f \frac{1}{m}\sum_{x_i}\Big(f(x_i)-y_i\Big)^2. \tag{2}$$

It's evident that:

$$\min_f \sum_{x_i}\Big(f(x_i)-y_i\Big)^2 \geq \sum_{x_i}\min_{f(x_i)}\Big(f(x_i)-y_i\Big)^2. \tag{3}$$

Moreover, equality holds:

$$\min_f \sum_{x_i}\Big(f(x_i)-y_i\Big)^2 = \sum_{x_i}\min_{f(x_i)}\Big(f(x_i)-y_i\Big)^2. \tag{4}$$

if and only if $f(x_i)$ and $f(x_j)$ are nearly independent for any $x_i \neq x_j \in \mathcal{S}$. In the binary classification context, the square loss function can be transformed to:

$$\begin{aligned}\min_f \sum_{x_i}\Big(f(x_i)-y_i\Big)^2 &= \sum_{x_i}\min_{f(x_i)}\Big(f(x_i)-y_i\Big)^2\\ &= \sum_{x_i}\min_{f(x_i)}\mathcal{P}(x_i)\Big(f(x_i)-1\Big)^2+\mathcal{N}(x_i)\Big(f(x_i)+1\Big)^2.\end{aligned} \tag{5}$$

The optimal solution becomes:

$$f^*_{\text{Square}}(x_i) = \arg\min_{f(x_i)}\mathcal{P}(x_i)\Big(f(x_i)-1\Big)^2+\mathcal{N}(x_i)\Big(f(x_i)+1\Big)^2 \tag{6}$$

and is simply:

$$f^*_{\text{Square}}(x_i) = \frac{\mathcal{P}(x_i)-\mathcal{N}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)} \tag{7}$$

for each $x_i \in \mathcal{S}$. Finally, we have

$$\text{minimum square loss} = \frac{4}{m}\sum_{x_i}\frac{\mathcal{P}(x_i)\mathcal{N}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)}. \tag{8}$$

2. *Statistical ensemble on square loss function*

In this context, every classifier f can be transformed into a n-dimensional vector

$$\pi^f = (x_1^f, x_2^f, \cdots, x_n^f) \tag{9}$$

in which $x_i^f = f(x_i)$ for each $x_i \in \mathcal{S}$. Next, the energy of π^f is defined as

$$\begin{aligned} E^{Square}(\pi^f) =& \frac{1}{m}\sum_{x_i}\Big(\pi_i^f - y_i\Big)^2 \\ =& \frac{1}{m}\sum_{x_i}\mathcal{P}(x_i)\Big(\pi_i^f - 1\Big)^2 + \mathcal{N}(x_i)\Big(\pi_i^f + 1\Big)^2. \end{aligned} \tag{10}$$

Let $\Pi(\mathcal{S}) = \mathbb{R}^n$. Then, the partition function is

$$\begin{aligned} Z^{Square} &= \sum_{\pi_f \in \Pi(\mathcal{S})} \exp(-\beta E^{Square}(\pi^f)) \\ &= \int \prod_{x_i^f} \mathrm{d}\pi_i^f \exp\left\{-\beta\left(\frac{1}{m}\sum_{x_i}\mathcal{P}(x_i)\Big(\pi_i^f - 1\Big)^2 + \mathcal{N}(x_i)\Big(\pi_i^f + 1\Big)^2\right)\right\} \end{aligned} \tag{11}$$

And, the ground state is calculated as

$$\begin{aligned} E_0^{Square} &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln Z^{Square} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \int \prod_{x_i^f} \mathrm{d}\pi_i^f \exp\left\{-\beta\left(\frac{1}{m}\sum_{x_i}\mathcal{P}(x_i)\Big(\pi_i^f - 1\Big)^2 + \mathcal{N}(x_i)\Big(\pi_i^f + 1\Big)^2\right)\right\} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \int \prod_{x_i^f} \mathrm{d}\pi_i^f \prod_{x_i} \exp\left\{-\frac{\beta}{m}\left\{\mathcal{P}(x_i)\Big(\pi_i^f - 1\Big)^2 + \mathcal{N}(x_i)\Big(\pi_i^f + 1\Big)^2\right\}\right\} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \prod_{x_i^f} \int \mathrm{d}\pi_i^f \exp\left\{-\frac{\beta}{m}\left\{\mathcal{P}(x_i)\Big(\pi_i^f - 1\Big)^2 + \mathcal{N}(x_i)\Big(\pi_i^f + 1\Big)^2\right\}\right\} \\ &= \sum_{x_i} \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \int \mathrm{d}\pi_i^f \exp\left\{-\frac{\beta}{m}\left\{\mathcal{P}(x_i)\Big(\pi_i^f - 1\Big)^2 + \mathcal{N}(x_i)\Big(\pi_i^f + 1\Big)^2\right\}\right\}. \end{aligned} \tag{12}$$

Utilizing Gaussian transformation, we finally obtain

$$E_0^{Square} = \frac{4}{m}\sum_{x_i}\frac{\mathcal{P}(x_i)\mathcal{N}(x_i)}{\mathcal{P}(x_i) + \mathcal{N}(x_i)}. \tag{13}$$

B. Optimal logistic loss and statistical ensemble on logistic cost function

1. *Optimal logistic loss*

The Logistic loss function, frequently used for binary classification, is defined for a dataset $\mathcal{S}$ and feature domain $\mathcal{X}$ as:

$$\min_f \ \frac{1}{m}\sum_{x_i} -y_i \log f(x_i) - \Big(1 - y_i\Big)\log\Big(1 - f(x_i)\Big), \tag{14}$$

with the optimal solution:

$$f^*_{\text{Logistic}} = \arg\min_f \ \frac{1}{m}\sum_{x_i} -y_i \log f(x_i) - \Big(1 - y_i\Big)\log\Big(1 - f(x_i)\Big). \tag{15}$$

Close to the optimal, the logistic loss function can be approximated as:

$$\min_{f} \ \frac{1}{m}\sum_{x_i}\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log f(x_i) - 2\mathcal{N}(x_i)\log\Big(1-f(x_i)\Big). \tag{16}$$

The corresponding optimal solution then is:

$$f^*_{\text{Logistic}}(x_i) = \arg\min_{f(x_i)}\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log f(x_i) - 2\mathcal{N}(x_i)\log\Big(1-f(x_i)\Big) \tag{17}$$

for each $x_i \in \mathcal{S}$. Finally, it can be simplified as

$$f^*_{\text{Logistic}}(x_i) = \frac{\mathcal{P}(x_i)-\mathcal{N}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)} \tag{18}$$

for each $x_i \in \mathcal{S}$.

2. Statistical ensemble on logistic cost function

In this context, the energy of π^f is defined as

$$E^{Logistic}(\pi^f) = \frac{1}{m}\sum_{x_i}\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log \pi^f_i - 2\mathcal{N}(x_i)\log\Big(1-\pi^f_i\Big). \tag{19}$$

Let $\Pi(\mathcal{S}) = \mathbb{R}^n$. Then, the partition function is

$$\begin{aligned} Z^{Logistic} &= \sum_{\pi_f\in\Pi(\mathcal{S})}\exp(-\beta E^{Logistic}(\pi^f)) \\ &= \int\prod_{x_i^f}\mathrm{d}\pi_i^f \exp\left\{-\beta\left(\frac{1}{m}\sum_{x_i}\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log\pi_i^f - 2\mathcal{N}(x_i)\log\Big(1-\pi_i^f\Big)\right)\right\}. \end{aligned} \tag{20}$$

And, the ground state is calculated as

$$\begin{aligned} E_0^{Logistic} &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln Z^{Logistic} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln\int_0^1\prod_{x_i^f}\mathrm{d}\pi_i^f \exp\left\{-\beta\left(\frac{1}{m}\sum_{x_i}\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log\pi_i^f - 2\mathcal{N}(x_i)\log\Big(1-\pi_i^f\Big)\right)\right\} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln\int_0^1\prod_{x_i^f}\mathrm{d}\pi_i^f \prod_{x_i}\exp\left\{-\frac{\beta}{m}\left\{\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log\pi_i^f - 2\mathcal{N}(x_i)\log\Big(1-\pi_i^f\Big)\right\}\right\} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln\prod_{x_i}\int_0^1\mathrm{d}\pi_i^f \exp\left\{-\frac{\beta}{m}\left\{\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log\pi_i^f - 2\mathcal{N}(x_i)\log\Big(1-\pi_i^f\Big)\right\}\right\} \\ &= \sum_{x_i}\lim_{\beta\to\infty} -\frac{1}{\beta}\ln\int_0^1\mathrm{d}\pi_i^f \exp\left\{-\frac{\beta}{m}\left\{\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log\pi_i^f - 2\mathcal{N}(x_i)\log\Big(1-\pi_i^f\Big)\right\}\right\} \\ &= \sum_{x_i}\min_{\pi_i^f\in(0,1)}\frac{1}{m}\left\{\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\log\pi_i^f - 2\mathcal{N}(x_i)\log\Big(1-\pi_i^f\Big)\right\} \\ &= \frac{1}{m}\sum_{x_i}\Big(\mathcal{N}(x_i)-\mathcal{P}(x_i)\Big)\ln\frac{\mathcal{P}(x_i)-\mathcal{N}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)} - 2\mathcal{N}(x_i)\ln\frac{2\mathcal{N}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)}, \end{aligned} \tag{21}$$

in which

$$f^*_{\text{Logistic}}(x_i) = \frac{\mathcal{P}(x_i)-\mathcal{N}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)} \tag{22}$$

for each $x_i \in \mathcal{S}$.

C. Optimal Hinge loss and statistical ensemble on Hinge cost function

1. Optimal Hinge loss

The hinge loss function is commonly used in Support Vector Machines (SVM) and other models employing maximum-margin classifiers, regardless of whether the classification task is discrete or continuous. It penalizes misclassified instances based on their distance from the decision boundary. The Hinge loss function for binary classification is described as follows:

$$\min_f \frac{1}{2m}\sum_{i=1}^{m}\max\{0, 1-y_i f(x_i)\}, \tag{23}$$

and its optimal solution is

$$f^*_{\text{Hinge}} = \arg\min_f \frac{1}{2m}\sum_{i=1}^{m}\max\{0, 1-y_i f(x_i)\}. \tag{24}$$

Unlike other objective functions, the output for any solution of the hinge loss function is discrete. We employ a similar technique to the hinge loss function as before,

$$\min_{f(x_i)\in\{1,-1\}} \mathcal{P}(x_i)\max\{0, 1-f(x_i)\} + \mathcal{N}(x_i)\max\{0, 1+f(x_i)\} \tag{25}$$

for any x_i, where its optimal solution can also be expressed as

$$f^*_{\text{Hinge}}(x_i) = \arg\max_{f(x_i)\in\{1,-1\}} \mathcal{P}(x_i)\max\{0, 1-f(x_i)\} + \mathcal{N}(x_i)\max\{0, 1+f(x_i)\} \tag{26}$$

for each $x_i \in \mathcal{S}$. Since $f(x_i)$ must be equal to 1 or -1, we can rewrite the optimal solution as

$$f^*_{\text{Hinge}}(x_i) = \begin{cases} 1, & \text{if } \mathcal{P}(x_i) \geq \mathcal{N}(x_i), \\ -1, & \text{if } \mathcal{P}(x_i) < \mathcal{N}(x_i), \end{cases} \tag{27}$$

for each $x_i \in \mathcal{S}$. And,

$$\text{minimum Hinge loss} = \frac{1}{m}\sum_{x_i}\min\Big\{\mathcal{P}(x_i), \mathcal{N}(x_i)\Big\}. \tag{28}$$

2. Statistical ensemble on Hinge cost function

In this context, the energy of π^f is defined as

$$E^{Hinge}(\pi^f) = \frac{1}{2m}\sum_{x_i}\mathcal{P}(x_i)\max\{0, 1-\pi^f_i\} + \mathcal{N}(x_i)\max\{0, 1+\pi^f_i\}, \tag{29}$$

where $\pi^f_i \in \{1,-1\}$ for each x_i. Let $\Pi(\mathcal{S}) = \{1,-1\}^n$. Then, the partition function is

$$\begin{aligned} Z^{Hinge} &= \sum_{\pi_f\in\Pi(\mathcal{S})}\exp(-\beta E^{Hinge}(\pi^f)) \\ &= \sum_{\pi_f\in\Pi(\mathcal{S})}\exp\left\{-\beta\left(\frac{1}{2m}\sum_{x_i}\mathcal{P}(x_i)\max\{0, 1-\pi^f_i\} + \mathcal{N}(x_i)\max\{0, 1+\pi^f_i\}\right)\right\}. \end{aligned} \tag{30}$$

And, the ground state is calculated as

$$\begin{aligned}
E_0^{Hinge} &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln Z^{Hinge}\\
&= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \sum_{\pi_f\in\Pi(\mathcal{S})} \exp\left\{-\beta\left(\frac{1}{2m}\sum_{x_i}\mathcal{P}(x_i)\max\{0,1-\pi_i^f\}+\mathcal{N}(x_i)\max\{0,1+\pi_i^f\}\right)\right\}\\
&= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \sum_{\pi_f\in\Pi(\mathcal{S})}\prod_{x_i} e^{-\frac{\beta}{2m}\left(\mathcal{P}(x_i)\max\{0,1-\pi_i^f\}+\mathcal{N}(x_i)\max\{0,1+\pi_i^f\}\right)}\\
&= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \prod_{x_i}\left(e^{-\beta\mathcal{N}(x_i)/m}+e^{-\beta\mathcal{P}(x_i)/m}\right)\\
&= \sum_{x_i}\lim_{\beta\to\infty} -\frac{1}{\beta}\ln\left(e^{-\beta\mathcal{N}(x_i)/m}+e^{-\beta\mathcal{P}(x_i)/m}\right)\\
&= \frac{1}{m}\sum_{x_i}\min\left\{\mathcal{P}(x_i),\mathcal{N}(x_i)\right\}.
\end{aligned} \tag{31}$$

D. Optimal Softmax loss and statistical ensemble on Softmax cost function

1. *Optimal Softmax loss*

The Softmax loss function, also known as the cross-entropy loss or log-likelihood loss, is essential in machine learning and deep learning, particularly for classification tasks. It measures the dissimilarity between predicted class probabilities and true class labels. The softmax function is typically used in conjunction with this loss function to convert raw model outputs into probability distributions over multiple classes. Mathematically, for a feature x, the softmax function computes the probability of the positive class as follows:

$$f(x) = \frac{e^{z_+}}{e^{z_+}+e^{z_-}}. \tag{32}$$

Here, z_+ (z_-) denotes the score that measures the likelihood of the data with feature x belonging to the positive (negative) class.

The softmax loss function is defined as the negative log-likelihood of the true class in binary classification tasks:

$$\min_f \ \frac{1}{m}\sum_{i=1}^{m}\big[-y_i\log f(x_i)-(1-y_i)\log(1-f(x_i))\big], \tag{33}$$

with the optimal solution being

$$f^*_{\text{Softmax}} = \arg\min_f \ \frac{1}{m}\sum_{i=1}^{m}\big[-y_i\log f(x_i)-(1-y_i)\log(1-f(x_i))\big]. \tag{34}$$

We can simplify the objective function as

$$\min_{f(x_i)} \ -\mathcal{P}(x_i)\log f(x_i)-\mathcal{N}(x_i)\log(1-f(x_i)), \tag{35}$$

and its optimal solution as

$$f^*_{\text{Softmax}}(x_i) = \arg\min_{f(x_i)} -\mathcal{P}(x_i)\log f(x_i)-\mathcal{N}(x_i)\log(1-f(x_i)), \tag{36}$$

for every $x_i\in\mathcal{S}$. By performing the necessary derivations, we find the optimal solution for the Softmax loss function as

$$f^*_{\text{Softmax}}(x_i) = \frac{\mathcal{P}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)}, \tag{37}$$

for each $x_i\in\mathcal{S}$.

2. Statistical ensemble on Softmax cost function

In this context, the energy of π^f is defined as

$$E^{Softmax}(\pi^f) = \frac{1}{m}\sum_{i=1}^{m} -\mathcal{P}(x_i)\log\pi_i^f - \mathcal{N}(x_i)\log(1-\pi_i^f), \tag{38}$$

Let $\Pi(\mathcal{S}) = \mathbb{R}^n$. Then, the partition function is

$$\begin{aligned} Z^{Softmax} &= \sum_{\pi_f\in\Pi(\mathcal{S})} \exp(-\beta E^{Softmax}(\pi^f)) \\ &= \int \prod_{x_i^f} \mathrm{d}\pi_i^f \exp\left\{-\beta\left(\frac{1}{m}\sum_{i=1}^{m} -\mathcal{P}(x_i)\log\pi_i^f - \mathcal{N}(x_i)\log(1-\pi_i^f)\right)\right\}. \end{aligned} \tag{39}$$

And, the ground state is calculated as

$$\begin{aligned} E_0^{Softmax} &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln Z^{Softmax} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln\int_0^1 \prod_{x_i^f} \mathrm{d}\pi_i^f \exp\left\{-\beta\left(\frac{1}{m}\sum_{x_i} -\mathcal{P}(x_i)\log\pi_i^f - \mathcal{N}(x_i)\log(1-\pi_i^f)\right)\right\} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln\int_0^1 \prod_{x_i^f} \mathrm{d}\pi_i^f \prod_{x_i} \exp\left\{-\frac{\beta}{m}\left\{-\mathcal{P}(x_i)\log\pi_i^f - \mathcal{N}(x_i)\log(1-\pi_i^f)\right\}\right\} \\ &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln\prod_{x_i}\int_0^1 \mathrm{d}\pi_i^f \exp\left\{-\frac{\beta}{m}\left\{-\mathcal{P}(x_i)\log\pi_i^f - \mathcal{N}(x_i)\log(1-\pi_i^f)\right\}\right\} \\ &= \sum_{x_i}\lim_{\beta\to\infty} -\frac{1}{\beta}\ln\int_0^1 \mathrm{d}\pi_i^f \exp\left\{-\frac{\beta}{m}\left\{-\mathcal{P}(x_i)\log\pi_i^f - \mathcal{N}(x_i)\log(1-\pi_i^f)\right\}\right\} \\ &= \sum_{x_i}\min_{\pi_i^f\in(0,1)} \frac{1}{m}\left\{-\mathcal{P}(x_i)\log\pi_i^f - \mathcal{N}(x_i)\log(1-\pi_i^f)\right\} \\ &= \frac{1}{m}\sum_{x_i} -\mathcal{P}(x_i)\ln\frac{\mathcal{P}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)} - \mathcal{N}(x_i)\ln\frac{\mathcal{N}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)}, \end{aligned} \tag{40}$$

in which

$$f^*_{\text{Logistic}}(x_i) = \frac{\mathcal{P}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)} \tag{41}$$

for each $x_i \in \mathcal{S}$.

III. EVALUATION MEASUREMENTS AND STATISTICAL ENSEMBLES

The evaluation of the performance of binary classifiers involves various metrics, among which the Receiver Operating Characteristic (ROC) curve and the Precision-Recall (PR) curve are prominent. These metrics offer insights into the effectiveness of a classifier at different threshold settings. This note delves into the optimal ROC and PR curves [2, 3], providing a mathematical exposition of their derivation and interpretation in the context of a given dataset.

A. Optimal ROC Curve and statistical ensemble on AR cost function

1. Optimal ROC Curve

The ROC curve represents the trade-off between the true positive rate (TPR) and the false positive rate (FPR) of a classifier. The curve is constructed by plotting TPR against FPR at various threshold levels. An optimal ROC curve approaches the top-left corner of the plot, indicating both high TPR and low FPR.

Consider a dataset $\mathcal{S} = \{x_1, x_2, \ldots, x_m\}$ with a feature domain $\mathcal{X}$. A classifier f assigns a score to each instance in $\mathcal{S}$, resulting in a sorted sequence $\mathcal{S}^f = \{x_1^f, x_2^f, \ldots, x_m^f\}$, where $f(x_i^f) \geq f(x_j^f)$ for $i < j$. The real label corresponding to x_i^f is denoted as y_i^f. To construct the ROC curve, one must calculate the TPR and FPR at various thresholds $t \in \{1, \ldots, m\}$ where each $t = k$ indicates that instances $x_1^f, \ldots, x_k^f$ are classified as positive. The TPR and FPR are given by:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{n_+} = \frac{1}{n_+} \sum_{i=1}^{k} \frac{1}{2}(1 + y_i^f), \tag{42}$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{n_-} = \frac{1}{n_-} \sum_{i=1}^{k} \frac{1}{2}(1 - y_i^f). \tag{43}$$

To find the optimal ROC curve, the objective is to maximize TPR and minimize FPR for each threshold k. This bi-objective optimization can be expressed as:

$$\begin{aligned} \max_f \quad & \sum_{i=1}^{k} \frac{1}{2}(1 + y_i^f), \\ \min_f \quad & \sum_{i=1}^{k} \frac{1}{2}(1 - y_i^f). \end{aligned} \tag{44}$$

Considering that the sum of true positives and false positives equals k, the optimization problem in Eq. (44) simplifies to:

$$\max \quad \sum_{i=1}^{k} \frac{1}{2}(1 + y_i^f). \tag{45}$$

The calculation of FPR for a fixed k includes instances that either exceed or equal the score $f(x_k^f)$. However, the optimization is primarily concerned with the maximization of true positives. Therefore we have

$$\sum_{i=1}^{k} \frac{1}{2}\left(1 + y_i^f\right) = \sum_{i=1}^{m} \frac{1}{2} \mathbb{1}\left(f(x_i) > f(x_k^f)\right)(1 + y_i) + \frac{\alpha}{2} \mathbb{1}\left(f(x_i) = f(x_k^f)\right)(1 + y_i), \tag{46}$$

in which $\alpha = \frac{k - \sum_{i=1}^{m} \mathbb{1}\left(f(x_i) > f(x_k^f)\right)}{\sum_{i=1}^{m} \mathbb{1}\left(f(x_i) = f(x_k^f)\right)}$ is the ratio of instances with score $f(x_k^f)$ in the subset $\{x_1^f, x_2^f, \cdots, x_k^f\}$ to all instances with score $f(x_k^f)$. Without loss of generality, we assume that $f(x_i) = f(x_j) \iff x_i = x_j$ for every $x_i, x_j \in \mathcal{S}$. Then we obtain that

$$\begin{aligned} & \sum_{i=1}^{k} \frac{1}{2}\left(1 + y_i^f\right) \\ = & \sum_{x_i \in \mathcal{S}} \mathbb{I}\left(f(x_i) > f(x_k^f)\right) \mathcal{P}(x_i) + \alpha \sum_{x_i \in \mathcal{S}} \mathbb{I}\left(f(x_i) = f(x_k^f)\right) \mathcal{P}(x_i) \\ = & \sum_{i=1}^{m} \mathbb{I}\left(f(x_i) > f(x_k^f)\right) \frac{\mathcal{P}(x_i)}{\mathcal{P}(x_i) + \mathcal{N}(x_i)} + \alpha \sum_{i=1}^{m} \mathbb{I}\left(f(x_i) = f(x_k^f)\right) \frac{\mathcal{P}(x_k^f)}{\mathcal{P}(x_k^f) + \mathcal{N}(x_k^f)} \\ = & \sum_{i=1}^{m} \mathbb{I}\left(f(x_i) > f(x_k^f)\right) \frac{\mathcal{P}(x_i)}{\mathcal{P}(x_i) + \mathcal{N}(x_i)} + \left(k - \sum_{i=1}^{m} \mathbb{I}\left(f(x_i) > f(x_k^f)\right)\right) \frac{\mathcal{P}(x_k^f)}{\mathcal{P}(x_k^f) + \mathcal{N}(x_k^f)} \\ = & \sum_{i=1}^{m} \frac{\mathcal{P}(x_i^f)}{\mathcal{P}(x_i^f) + \mathcal{N}(x_i^f)}. \end{aligned} \tag{47}$$

Denote $w_i = \frac{\mathcal{P}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)}$ as the non-negative weight of instance x_i. Hence, the optimization problem (45) can be regarded as a problem of how to choose a k-elements subset of $\mathcal{S}$ which satisfies that the total weight is maximum. And it is naturally rewritten in the form of combinatorial optimization as follows,

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} w_i z_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} z_i = k \\ & z_i \in \{0,1\}, \ i = 1, 2, \cdots, m \end{aligned} \tag{48}$$

Actually, this combinatorial optimization problem belongs to the classical 0-1 knapsack problems, which is the most common problem being solved. Noting that all weights are non-negative, simple greedy algorithm can reach the optimal solution if we select the top k instances with the highest weight. As a rule of how to select optimal k-element subset with arbitrary k, the optimal classifier for best ROC curve is the weight function, that is,

$$f^*_{\text{ROC}}(x_i) = \frac{\mathcal{P}(x)}{\mathcal{P}(x_i) + \mathcal{N}(x_i)} \tag{49}$$

for every $x_i \in \mathcal{S}$. Here, we denote that $\{x_1^*, x_2^*, \cdots, x_n^*\}$ is the sorted sequence of $\mathcal{S}$ in descending order of weight.

Naturally, the best ROC curve can be drawn sequentially from a series of data points in (FPR, TPR)-plane as follows,

$$\left\{ \left(\frac{1}{n_-} \sum_{i=1}^{k} \frac{\mathcal{N}(x_i^*)}{\mathcal{P}(x_i^*) + \mathcal{N}(x_i^*)}, \frac{1}{n_+} \sum_{i=1}^{k} \frac{\mathcal{P}(x_i^*)}{\mathcal{P}(x_i^*) + \mathcal{N}(x_i^*)} \right) \right\}_{k=0,1,\cdots,n}. \tag{50}$$

The optimal ROC curve can be regarded as a combination of n linear piecewise functions, whose derivatives are composed of the following sequence

$$\left\{ \frac{n_- \mathcal{P}(x_i^*)}{n_+ \mathcal{N}(x_i^*)} \right\}_{k=1,2,\cdots,n}. \tag{51}$$

It is easy to check that, the above sequence is monotonically decreasing, since $\{x_1^*, x_2^*, \cdots, x_n^*\}$ is sorted by descending order of weight and

$$\frac{\mathcal{P}(x_i^*)}{\mathcal{P}(x_i^*) + \mathcal{N}(x_i^*)} = \frac{1}{1 + \frac{1}{\frac{\mathcal{P}(x_i^*)}{\mathcal{N}(x_i^*)}}}. \tag{52}$$

Therefore, the best ROC curve is **concave**. Moreover, the area under the best ROC curve is the upper bound of AR ($textAR^u$). In other words, AR^u is equal to the area under the curve described as f^*_{ROC} in the following:

$$\begin{aligned} \text{AR}^u =& \sum_{i=1}^{m} \frac{1}{n_-} \frac{\mathcal{N}(x_i^*)}{\mathcal{P}(x_i^*) + \mathcal{N}(x_i^*)} \left(\frac{1}{n_+} \sum_{j<i} \frac{\mathcal{P}(x_j^*)}{\mathcal{P}(x_j^*) + \mathcal{N}(x_j^*)} + \frac{1}{2n_+} \frac{\mathcal{P}(x_i^*)}{\mathcal{P}(x_i^*) + \mathcal{N}(x_i^*)} \right) \\ =& \frac{1}{n_- n_+} \sum_{i>j} \frac{\mathcal{N}(x_i^*)\mathcal{P}(x_j^*)}{\Big(\mathcal{P}(x_i^*) + \mathcal{N}(x_i^*)\Big)\Big(\mathcal{P}(x_j^*) + \mathcal{N}(x_j^*)\Big)} + \frac{1}{2} \sum_{i=1}^{m} \frac{\mathcal{N}(x_i^*)\mathcal{P}(x_i^*)}{\Big(\mathcal{P}(x_i^*) + \mathcal{N}(x_i^*)\Big)^2} \\ =& \frac{1}{2n_- n_+} \sum_{i,j} \frac{\max\Big\{\mathcal{N}(x_i^*)\mathcal{P}(x_j^*), \mathcal{N}(x_j^*)\mathcal{P}(x_i^*)\Big\}}{\Big(\mathcal{P}(x_i^*) + \mathcal{N}(x_i^*)\Big)\Big(\mathcal{P}(x_j^*) + \mathcal{N}(x_j^*)\Big)} \\ =& \frac{1}{2n_- n_+} \sum_{i,j} \frac{\max\Big\{\mathcal{N}(x_i)\mathcal{P}(x_j), \mathcal{N}(x_j)\mathcal{P}(x_i)\Big\}}{\Big(\mathcal{P}(x_i) + \mathcal{N}(x_i)\Big)\Big(\mathcal{P}(x_j) + \mathcal{N}(x_j)\Big)} \\ =& \frac{1}{2n_- n_+} \sum_{x_i, x_j} \max\Big\{\mathcal{P}(x_i)\mathcal{N}(x_j), \mathcal{P}(x_j)\mathcal{N}(x_i)\Big\}. \end{aligned} \tag{53}$$

2. Statistical ensemble on AR cost function

For a given dataset $\mathcal{S} := \{x_1, x_2, \cdots, x_m\}$, one classifier f can be transformed into a ranking of S according to the score given by f:

$$\pi_f := \{x_1^f, x_2^f, \cdots, x_m^f\}, \tag{54}$$

where x_i^f is the i-th sample of the ordered set ranked by f in $\mathcal{S}$. Here, we define that $\Pi(\mathcal{S})$ as the rankings of all possible classifiers for $\mathcal{S}$. Then, we define the energy of π^f as follows,

$$\begin{aligned}
E^{AR}(\pi^f) =& 1 - AR(\pi^f) && (55)\\
=& 1 - \sum_{i<j} p_+(x_i^f)p_-(x_j^f) - \frac{1}{2}\sum_i p_+(x_i^f)p_-(x_i^f) && (56)\\
=& \sum_{i>j} p_+(x_i^f)p_-(x_j^f) + \frac{1}{2}\sum_i p_+(x_i^f)p_-(x_i^f) && (57)
\end{aligned}$$

Treating $\Pi(\mathcal{S})$ as an ensemble, we can write the following partition function,

$$\begin{aligned}
Z^{AR} &= \sum_{\pi^f \in \Pi(\mathcal{S})} \exp(-\beta E^{AR}(\pi)) \\
&= \sum_{\pi^f \in \Pi(\mathcal{S})} \exp\left\{-\beta\left(\sum_{i>j} p_+(x_i^f)p_-(x_j^f) + \frac{1}{2}\sum_i p_+(x_i^f)p_-(x_i^f)\right)\right\}.
\end{aligned} \tag{58}$$

And, the ground state energy E_0^{AR} $(1 - AR^u)$ can be defined as

$$E_0^{AR} = \lim_{\beta\to\infty} -\frac{1}{\beta}\ln Z^{AR}. \tag{59}$$

Next, we can solve E_0^{AR} as follows.

$$\begin{aligned}
E_0^{AR} &= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln Z^{AR} \\
&= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \sum_{\pi_f \in \Pi(\mathcal{S})} \exp\left\{-\beta\left(\sum_{i>j} p_+(x_i^f)p_-(x_j^f) + \frac{1}{2}\sum_i p_+(x_i^f)p_-(x_i^f)\right)\right\} \\
&= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \sum_{\pi_f \in \Pi(\mathcal{S})} \prod_{i>j} \exp\Big(-\beta p_+(x_i^f)p_-(x_j^f)\Big) \prod_{i=1}^{n} \exp\Big(-\frac{\beta}{2} p_+(x_i^f)p_-(x_i^f)\Big) \\
&= \lim_{\beta\to\infty} -\frac{1}{\beta}\ln \prod_{i,j}\left(e^{-\frac{\beta}{2}p_+(x_i^f)p_-(x_j^f)} + e^{-\frac{\beta}{2}p_+(x_j^f)p_-(x_i^f)}\right) \\
&= \lim_{\beta\to\infty} -\frac{1}{\beta}\sum_{i,j}\ln\left(e^{-\frac{\beta}{2}p_+(x_i^f)p_-(x_j^f)} + e^{-\frac{\beta}{2}p_+(x_j^f)p_-(x_i^f)}\right) \\
&= \sum_{i,j}\lim_{\beta\to\infty} -\frac{1}{\beta}\ln\left(e^{-\frac{\beta}{2}p_+(x_i^f)p_-(x_j^f)} + e^{-\frac{\beta}{2}p_+(x_j^f)p_-(x_i^f)}\right) \\
&= \frac{1}{2}\sum_{i,j}\min\Big\{p_+(x_j)p_-(x_k), p_+(x_k)p_-(x_j)\Big\} \\
&= \frac{1}{2n_+n_-}\sum_{x_i,x_j}\min\Big\{\mathcal{P}(x_i)\mathcal{N}(x_j), \mathcal{P}(x_j)\mathcal{N}(x_i)\Big\}.
\end{aligned} \tag{60}$$

Then, we know that

$$AR^u = 1 - E_0^{AR} = \frac{1}{2n_+n_-}\sum_{x_i,x_j}\max\Big\{\mathcal{P}(x_i)\mathcal{N}(x_j), \mathcal{P}(x_j)\mathcal{N}(x_i)\Big\}. \tag{61}$$

Further, it is worthy noting that

$$\begin{aligned}
AR^u =& \frac{1}{2}\sum_{i,j} \max\Big\{p_+(x_i)p_-(x_j), p_+(x_j)p_-(x_i)\Big\} \\
=& \sum_{i<j} \max\Big\{p_+(x_i)p_-(x_j), p_+(x_j)p_-(x_i)\Big\} + \frac{1}{2}\sum_i p_+(x_i)p_-(x_i) \\
=& \sum_{i<j} \max\Big\{p_+(x_i)\Big(1-p_+(x_j)\Big), p_+(x_j)\Big(1-p_+(x_i)\Big)\Big\} + \frac{1}{2}\sum_i p_+(x_i)p_-(x_i) \\
=& \sum_{i<j} \Big(\max\Big\{p_+(x_i), p_+(x_j)\Big\} - p_+(x_i)p_+(x_j)\Big) + \frac{1}{2}\sum_i p_+(x_i)p_-(x_i) \\
=& \sum_{i<j:\ p_+(x_i)>p_+(x_j)} \Big(p_+(x_i) - p_+(x_i)p_+(x_j)\Big) + \frac{1}{2}\sum_i p_+(x_i)p_-(x_i) \\
=& \sum_{i<j:\ p_+(x_i)>p_+(x_j)} p_+(x_i)p_-(x_j) + \frac{1}{2}\sum_i p_+(x_i)p_-(x_i).
\end{aligned} \tag{62}$$

Naturally, we construct a ranking $\pi^* = \{x_1^*, x_2^*, \cdots, x_m^*\}$ for $\mathcal{S}$ satisfying that $p_+(x_i^*) > p_+(x_j^*)$ for any $i < j$, whose AUC is equal to E_0^{AR} as follows,

$$E^{AR}(\pi_0) = \sum_{j<k:\ p_+(x'_j)>p_+(x'_k)} p_+(x'_j)p_-(x'_k) + \frac{1}{2}\sum_j p_+(x'_j)p_-(x'_j) = E_0 \tag{63}$$

It also means that, the optimal classifier f_{AR}^* corresponding to E_0^{AR} is

$$f_{AR}^*(x_i) = p_+(x_i), \quad \forall x_i \in \mathcal{S}. \tag{64}$$

B. Optimal PR Curve and statistical ensemble on AP cost function

1. Optimal PR Curve

The precision-recall (PR) curve is a widely utilized metric for evaluating binary classifiers, emphasizing the trade-off between precision and recall. It provides a detailed view of a classifier's performance at various decision thresholds. To plot a PR curve, one must compute precision and recall at numerous thresholds. This concept is analogous to the receiver operating characteristic (ROC) curve, which allows adjustment of the threshold to balance precision and recall.

For each threshold value, precision and recall are determined using the following formulas:

$$\text{precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}, \tag{65}$$

where TP is the number of true positives and FP is the number of false positives. Recall, also known as true positive rate (TPR), is defined as:

$$\text{recall} = \frac{\text{TP}}{\text{TP}+\text{FN}}, \tag{66}$$

where FN is the number of false negatives. Both the ROC and PR curves share a commonality in that they evaluate the classifier's performance by computing metrics at various thresholds. At a specific threshold k, the precision and recall can be expressed as:

$$\text{precision} = \frac{\text{TP}}{k} = \frac{n_+}{k} \times \text{TPR}, \tag{67}$$

$$\text{recall} = \text{TPR}, \tag{68}$$

where n_+ is the total number of positive samples.

The objective in seeking the optimal PR curve is to maximize both precision and recall for a given threshold k. This can be formulated as the following optimization problem:

$$\max_{f}\ \mathrm{TPR}, \tag{69}$$

which we have addressed in the previous section. The optimal classifier that achieves the best PR curve is identical to the one that optimizes the ROC curve and is given by:

$$f^*_{\mathrm{PR}}(x) = f^*_{\mathrm{ROC}}(x) = \frac{\mathcal{P}(x)}{\mathcal{P}(x)+\mathcal{N}(x)}, \tag{70}$$

where x belongs to the sample space $\mathcal{S}$. The optimal PR curve is constructed by plotting a series of data points in the (recall, precision)-plane, which are calculated as follows:

$$\left\{\left(\frac{1}{n_+}\sum_{i=1}^{k}\frac{\mathcal{P}(x_i^*)}{\mathcal{P}(x_i^*)+\mathcal{N}(x_i^*)}, \frac{1}{k}\sum_{i=1}^{k}\frac{\mathcal{P}(x_i^*)}{\mathcal{P}(x_i^*)+\mathcal{N}(x_i^*)}\right)\right\}_{k=0,1,\cdots,n} \tag{71}$$

Furthermore, the upper bound of the area under the PR curve (AP^u) can be calculated as follows:

$$\mathrm{AP}^u = \frac{1}{2n_+}\sum_{x_i^*} p_+(x_i^*)\left(\frac{1}{i-1}\sum_{j=1}^{i-1}p_+(x_j^*)+\frac{1}{i}\sum_{j=1}^{i}p_+(x_j^*)\right), \tag{72}$$

where $p_+(x_i)$ is the probability of a sample x_i being positive:

$$p_+(x_i) = \frac{\mathcal{P}(x_i)}{\mathcal{P}(x_i)+\mathcal{N}(x_i)}. \tag{73}$$

134 *2. statistical ensemble on AP cost function*

Akin to AR ensemble, we define that $\Pi(\mathcal{S})$ includes all possible classifiers for $\mathcal{S}$. For any classifier $\pi_f \in \Pi(\mathcal{S})$, we define its energy as $1 - AP(\pi_f)$. As a consequence, the partition function and ground state of AP ensemble can be written as

$$Z^{AP} = \sum_{\pi^f\in\Pi(\mathcal{S})}\exp\Big(-\beta\left(1-AP(\pi^f)\right)\Big) \tag{74}$$

and

$$E_0^{AP} = \lim_{\beta\to\infty} -\frac{1}{\beta}\ln Z^{AP}. \tag{75}$$

However, it is very difficult to directly calculate the above ground state because there is no explicit expression for AP, unlike AR. Alternatively, we have proved in Sec. III B 1 that, the optimal AR shares the same classifier (ranking) with the optimal AP, that is,

$$f^*_{AP}(x_i) = f^*_{AR}(x_i) = p_+(x_i), \quad \forall x_i \in \mathcal{S}. \tag{76}$$

Furthermore, we can write the AP^u in an indirect expression:

$$\mathrm{AP}^u = \sum_{i=1}^{m}\frac{p_+(x_i^*)}{2n_+}\left(\frac{1}{i-1}\sum_{j=1}^{i-1}p_+(x_j^*)+\frac{1}{i}\sum_{j=1}^{i}p_+(x_j^*)\right), \tag{77}$$

135 where $\pi^* = \{x_1^*, x_2^*, \cdots, x_m^*\}$ has been mentioned in Sec. III A 2.

C. Optimal Accuracy and statistical ensemble on AC cost function

1. *Optimal Accuracy*

Accuracy (AC) is a fundamental metric that quantifies the proportion of correctly predicted instances against the total number of instances within a dataset. This metric is universally applicable across classifiers. Given a dataset $\mathcal{S}$ and a feature space $\mathcal{X}$, accuracy can be computed with the following expression:

$$\mathrm{AC} = \frac{1}{2m} \sum_{x_i \in \mathcal{S}} \Big(1 + f(x_i) y_i\Big), \tag{78}$$

where m represents the total number of instances, $f(x_i)$ is the predicted label for instance x_i, and y_i is the true label. The accuracy increases by $\frac{1}{2m}$ for each correctly classified instance x_i; it remains unchanged for incorrect predictions.

While accuracy itself is not a conventional loss function, optimizing for the highest possible accuracy and determining the optimal classifier functions are critical endeavors in machine learning. The mathematical upper limit of accuracy, denoted as AC^u, can be formalized as:

$$\mathrm{AC}^u = \max_f \frac{1}{2m} \sum_{x_i \in \mathcal{S}} \Big(1 + f(x_i) y_i\Big), \tag{79}$$

with the corresponding optimal classifier being:

$$f^*_{\mathrm{AC}} = \arg\max_f \frac{1}{2m} \sum_{x_i \in \mathcal{S}} \Big(1 + f(x_i) y_i\Big). \tag{80}$$

Through further analysis, we can expand and simplify the equation by considering the relationship between probabilities associated with positive and negative instances:

$$\begin{aligned} \max_f \sum_{x_i \in \mathcal{S}} \frac{1}{2m}\Big(1 + f(x_i)y_i\Big) &= \max_f \sum_{x \in \mathcal{X}} \frac{\mathcal{P}(x)}{2}\Big(1 + f(x)\Big) + \frac{\mathcal{N}(x)}{2}\Big(1 - f(x)\Big) \\ &\leq \sum_{x_i \in \mathcal{S}} \max_{f(x)} \frac{\mathcal{P}(x_i)}{2}\Big(1 + f(x_i)\Big) + \frac{\mathcal{N}(x_i)}{2}\Big(1 - f(x_i)\Big), \end{aligned} \tag{81}$$

which leads us to redefine AC^u as:

$$\mathrm{AC}^u = \frac{1}{m} \sum_{x_i} \max_{f(x_i)} \frac{\mathcal{P}(x_i)}{2}\Big(1 + f(x_i)\Big) + \frac{\mathcal{N}(x_i)}{2}\Big(1 - f(x_i)\Big), \tag{82}$$

and the optimal classifier for each instance $x_i \in \mathcal{S}$ becomes:

$$f^*_{\mathrm{AC}}(x_i) = \arg\max_{f(x_i)} \frac{\mathcal{P}(x_i)}{2}\Big(1 + f(x_i)\Big) + \frac{\mathcal{N}(x_i)}{2}\Big(1 - f(x_i)\Big). \tag{83}$$

By enumeration, the solution for the optimal classifier is:

$$f^*_{\mathrm{AC}}(x_i) = \begin{cases} 1 & \text{if } \mathcal{P}(x_i) \geq \mathcal{N}(x_i), \\ -1 & \text{if } \mathcal{P}(x_i) < \mathcal{N}(x_i), \end{cases} \tag{84}$$

resulting in the upper limit of accuracy being:

$$\mathrm{AC}^u = \frac{1}{m} \sum_{x_i \in \mathcal{S}} \max \Big\{\mathcal{P}(x_i), \mathcal{N}(x_i)\Big\}. \tag{85}$$

2. *statistical ensemble on AC cost function*

Unlike *AR* and *AP*, *AC* focuses on the predicted binary label for every sample, rather than a ranking of all samples in $\mathcal{S}$. In this context, every classifier f can be transformed into a m-dimensional binary vector

$$\pi^f = (x^f_1, x^f_2, \cdots, x^f_m), \tag{86}$$

where $x_i^f \in \{-1, 1\}$ for $i = 1, 2, \cdots, m$. This also means that $\Pi(\mathcal{S}) = \{1, -1\}^m$ for AC cost function. And, the energy of π^f is defined as

$$\begin{aligned}
E^{AC}(\pi^f) =& 1 - AC(\pi^f) \\
=& 1 - \left(\frac{1}{m} \sum_i \frac{\mathcal{P}(x_i)}{2} \Big(1 + x_i^f\Big) + \frac{\mathcal{N}(x_i)}{2} \Big(1 - x_i^f\Big) \right) \\
=& \frac{1}{m} \sum_i \frac{\mathcal{P}(x_i)}{2} \Big(1 - x_i^f\Big) + \frac{\mathcal{N}(x_i)}{2} \Big(1 + x_i^f\Big) \\
=& \frac{1}{2} + \frac{1}{2m} \sum_i x_i^f \Big(\mathcal{N}(x_i) - \mathcal{P}(x_i)\Big).
\end{aligned} \tag{87}$$

Then, the partition function is

$$\begin{aligned}
Z^{AC} &= \sum_{\pi_f \in \Pi(\mathcal{S})} \exp(-\beta E^{AC}(\pi^f)) \\
&= \sum_{\pi^f \in \Pi(\mathcal{S})} \exp\left\{ -\beta \left(\frac{1}{2} + \frac{1}{2m} \sum_i x_i^f \Big(\mathcal{N}(x_i) - \mathcal{P}(x_i)\Big) \right) \right\}
\end{aligned} \tag{88}$$

And, the ground state is calculated as

$$\begin{aligned}
E_0^{AC} &= \lim_{\beta \to \infty} -\frac{1}{\beta} \ln Z^{AC} \\
&= \lim_{\beta \to \infty} -\frac{1}{\beta} \ln \sum_{\pi_f \in \Pi(\mathcal{S})} \exp\left\{ -\beta \left(\frac{1}{2} + \frac{1}{2m} \sum_i x_i^f \Big(\mathcal{N}(x_i) - \mathcal{P}(x_i)\Big) \right) \right\} \\
&= \lim_{\beta \to \infty} -\frac{1}{\beta} \ln \sum_{\pi_f \in \Pi(\mathcal{S})} e^{-\beta/2} \prod_i \exp\left(-\frac{\beta}{2m} x_i^f \Big(\mathcal{N}(x_i) - \mathcal{P}(x_i)\Big) \right) \\
&= \lim_{\beta \to \infty} -\frac{1}{\beta} \ln \left\{ e^{-\beta/2} \prod_i \left(e^{-\frac{\beta}{2m}(\mathcal{N}(x_i) - \mathcal{P}(x_i))} + e^{-\frac{\beta}{2m}(\mathcal{P}(x_i) - \mathcal{N}(x_i))} \right) \right\} \\
&= \frac{1}{2} - \lim_{\beta \to \infty} \frac{1}{\beta} \ln \left\{ \prod_i \left(e^{-\frac{\beta}{2m}(\mathcal{N}(x_i) - \mathcal{P}(x_i))} + e^{-\frac{\beta}{2m}(\mathcal{P}(x_i) - \mathcal{N}(x_i))} \right) \right\} \\
&= \frac{1}{2} - \sum_i \lim_{\beta \to \infty} \frac{1}{\beta} \ln \left\{ e^{-\frac{\beta}{2m}(\mathcal{N}(x_i) - \mathcal{P}(x_i))} + e^{-\frac{\beta}{2m}(\mathcal{P}(x_i) - \mathcal{N}(x_i))} \right\} \\
&= \frac{1}{2} + \frac{1}{2m} \sum_i \min \Big\{ \mathcal{N}(x_i) - \mathcal{P}(x_i), \mathcal{P}(x_i) - \mathcal{N}(x_i) \Big\} \\
&= \frac{1}{m} \sum_i \min \Big\{ \mathcal{P}(x_i), \mathcal{N}(x_i) \Big\}
\end{aligned} \tag{89}$$

Indeed,

$$AC^u = 1 - E_0^{AC} = \frac{1}{m} \sum_i \max \Big\{ \mathcal{P}(x_i), \mathcal{N}(x_i) \Big\}. \tag{90}$$

Naturally, the optimal classifier corresponding to AC^u is

$$f_{AC}^*(x_i) = \begin{cases} 1 & \text{if } \mathcal{P}(x_i) \geq \mathcal{N}(x_i) \\ -1 & \text{if } \mathcal{P}(x_i) < \mathcal{N}(x_i) \end{cases}, \tag{91}$$

for any $x_i \in \mathcal{S}$.

IV. UNIVERSAL OPTIMAL CLASSIFIER

The theoretical foundation of optimal classifiers demonstrates their alignment with the common objective functions including Square loss, Logistic loss, Hinge loss, and Softmax loss [4] in section II and section III. Specifically, the optimal classifiers for both the ROC and PR curves are congruent, denoted as $f^*_{\text{AR}} \equiv f^*_{\text{AP}}$. This implies that continuous binary classifiers converge to a unified optimal form for each feature vector $x_i \in \mathcal{S}$, symbolized as $g^*_{\text{Square}} \equiv g^*_{\text{Logistic}} \sim p_+(x_i) \equiv f^*_{\text{AR}} \equiv f^*_{\text{AP}}$, where g^* represents the objective function. Similarly, discrete binary classifiers are equivalent, denoted as $g^*_{\text{Hinge}} \equiv g^*_{\text{Softmax}} \equiv f^*_{\text{AC}}$.

From this, we can deduce a universal representation for any optimal classifier, encompassing both continuous and discrete forms, that achieves the best performance across various objective functions and evaluation metrics:

$$\mathcal{F}^*(x_i) = \begin{cases} 1 & f^*(x_i) \geq 0 \\ -1 & f^*(x_i) < 0 \end{cases}, \tag{92}$$

where $f^*(x_i) = \frac{\mathcal{P}(x_i) - \mathcal{N}(x_i)}{\mathcal{P}(x_i) + \mathcal{N}(x_i)}$ is the derived optimal classifier that minimizes loss and maximizes performance across various objective functions and evaluation metrics.

Eq. 92 underscores the synergy between the learning process, which is steered by objective functions, and the evaluation process within the realm of binary classification. The efficacy of a binary classifier, irrespective of its computational complexity or efficiency, is inherently bound by the selected objective function and the intrinsic properties of the dataset. This establishes a performance ceiling dictated by both the chosen evaluation metric and the data's innate characteristics.

V. SENSITIVITY ANALYSIS

In Sections 2 and 3, we established the mathematical relationships between the lower bound of the objective function with respect to the training set and the upper bound of the evaluation metrics related to the test set. This section extends that discussion within the out-of-sample framework by combining these relationships to investigate the dual concerns of training loss and generalizability in classification tasks.

Let us denote the training and test sets by $\mathcal{S}_{train}$ and $\mathcal{S}_{test}$, respectively. Furthermore, we denote $\mathcal{P}_{train}(x_i)$ and $\mathcal{N}_{train}(x_i)$ as the respective counts of positive and negative instances with feature x_i in the training set, and similarly, $\mathcal{P}_{test}(x_i)$ and $\mathcal{N}_{test}(x_i)$ for the test set.

For a discrete classifier $f(x)$, the boundary hinge loss within the training set $\mathcal{S}_{train}$ is defined as:

$$\sum_{x_i} \min\{\mathcal{P}_{train}(x_i), \mathcal{N}_{train}(x_i)\}. \tag{93}$$

Here, Δ_{train} represents the discrepancy between the classifier's hinge loss and the boundary hinge loss, expressed as:

$$\Delta_{train} = \sum_{x_i} \Delta_{train}(x_i),$$

where

$$\Delta_{train}(x_i) = \begin{cases} \mathcal{N}_{train}(x_i) - \min\{\mathcal{P}_{train}(x_i), \mathcal{N}_{train}(x_i)\} & \text{if } f(x_i) = 1, \\ \mathcal{P}_{train}(x_i) - \min\{\mathcal{P}_{train}(x_i), \mathcal{N}_{train}(x_i)\} & \text{if } f(x_i) = -1. \end{cases}$$

In parallel, the maximum accuracy achievable on the test set $\mathcal{S}_{test}$ is:

$$\sum_{x_i} \max\{\mathcal{P}_{test}(x_i), \mathcal{N}_{test}(x_i)\}.$$

Similarly, Δ_{test} quantifies the error between the classifier's actual accuracy and the maximum possible accuracy:

$$\Delta_{test} = \sum_{x_i} \Delta_{test}(x_i),$$

where

$$\Delta_{test}(x_i) = \begin{cases} \max\{\mathcal{P}_{test}(x_i), \mathcal{N}_{test}(x_i)\} - \mathcal{P}_{test}(x_i) & \text{if } f(x_i) = 1, \\ \max\{\mathcal{P}_{test}(x_i), \mathcal{N}_{test}(x_i)\} - \mathcal{N}_{test}(x_i) & \text{if } f(x_i) = -1. \end{cases}$$

Therefore, the summation of these two discrepancies represents their respective optimization potential and performance evaluation capacity. For any given feature x_i, the combined error is:

$$(\Delta_{train} + \Delta_{test})(x_i) = \begin{cases} \max\{\mathcal{N}_{train}(x_i) - \mathcal{P}_{train}(x_i), 0\} + \max\{\mathcal{N}_{test}(x_i) - \mathcal{P}_{test}(x_i), 0\} & \text{if } f(x_i) = 1, \\ \max\{\mathcal{P}_{train}(x_i) - \mathcal{N}_{train}(x_i), 0\} + \max\{\mathcal{P}_{test}(x_i) - \mathcal{N}_{test}(x_i), 0\} & \text{if } f(x_i) = -1. \end{cases}$$

We now introduce a lower bound for $(\Delta_{train} + \Delta_{test})$, defined as:

$$\Delta = (\Delta_{train} + \Delta_{test})_{\min} = \sum_{x_i} \Delta(x_i), \tag{94}$$

where

$$\Delta(x_i) = \begin{cases} 0 & \text{if } \mathcal{P}_{train}(x_i) \geq \mathcal{N}_{train}(x_i),\ \mathcal{P}_{test}(x_i) \geq \mathcal{N}_{test}(x_i) \\ 0 & \text{if } \mathcal{P}_{train}(x_i) < \mathcal{N}_{train}(x_i),\ \mathcal{P}_{test}(x_i) < \mathcal{N}_{test}(x_i) \\ \epsilon(x_i) & \text{if } \mathcal{P}_{train}(x_i) \geq \mathcal{N}_{train}(x_i),\ \mathcal{P}_{test}(x_i) < \mathcal{N}_{test}(x_i) \\ \epsilon(x_i) & \text{if } \mathcal{P}_{train}(x_i) < \mathcal{N}_{train}(x_i),\ \mathcal{P}_{test}(x_i) \geq \mathcal{N}_{test}(x_i) \end{cases}$$

and

$$\epsilon(x_i) = \min\{|\mathcal{P}_{train}(x_i) - \mathcal{N}_{train}(x_i)|, |\mathcal{P}_{test}(x_i) - \mathcal{N}_{test}(x_i)|\}.$$

As we observed, Δ is a general lower bound for the sum of training error and evaluation error regardless of the specific classifier in use. Naturally, $\Delta = 0$ if and only if

$$\begin{cases} \mathcal{P}_{train}(x_i) \geq \mathcal{N}_{train}(x_i) \\ \mathcal{P}_{test}(x_i) \geq \mathcal{N}_{test}(x_i) \end{cases} \text{ or } \begin{cases} \mathcal{P}_{train}(x_i) < \mathcal{N}_{train}(x_i) \\ \mathcal{P}_{test}(x_i) < \mathcal{N}_{test}(x_i) \end{cases}$$

for each $x_i \in \mathcal{S}$.

VI. RANDOM DIVISION

Random division method is a standard approach to splitting a dataset into training and testing subsets. In this method, each instance is independently assigned to the training subset with probability p and to the testing subset with probability $1-p$. For a given feature vector x_i, the counts $\mathcal{P}_{train}(x_i)$ and $\mathcal{N}_{train}(x_i)$ follow binomial distributions $\mathcal{B}(\mathcal{P}(x_i), p)$ and $\mathcal{B}(\mathcal{N}(x_i), p)$ respectively. Considering this, we can investigate the discrepancy Δ within the context of probabilistic partitioning. We define the expected value of Δ as $\mathbb{E}[\Delta] = \frac{1}{m}\sum_{x_i} \mathbb{E}[\Delta(x_i; p)]$, which is calculated as:

$$\begin{aligned} \mathbb{E}[\Delta(x_i; p)] &= \mathbb{E}\left[\min\left\{\mathcal{N}_{train}(x_i) - \mathcal{P}_{test}(x_i), \mathcal{P}_{train}(x_i) - \mathcal{N}_{test}(x_i)\right\}\right] + \\ &\quad \mathbb{E}\left[\max\left\{\mathcal{P}_{test}(x_i), \mathcal{N}_{test}(x_i)\right\}\right] - \mathbb{E}\left[\min\left\{\mathcal{P}_{train}(x_i), \mathcal{N}_{train}(x_i)\right\}\right] \\ &= \mathbb{E}\left[\max\left\{\mathcal{P}_{test}(x_i), \mathcal{N}_{test}(x_i)\right\}\right] + \mathbb{E}\left[\max\left\{\mathcal{P}_{train}(x_i), \mathcal{N}_{train}(x_i)\right\}\right] - \\ &\quad \max\left\{\mathcal{P}(x_i), \mathcal{N}(x_i)\right\}, \end{aligned} \tag{95}$$

where the expected maxima are determined by:

$$\mathbb{E}[\max\{\mathcal{P}_{train}(x_i), \mathcal{N}_{train}(x_i)\}] = \sum_{i=0}^{\mathcal{P}(x_i)} \sum_{j=0}^{\mathcal{N}(x_i)} \max\{i, j\} \binom{\mathcal{P}(x_i)}{i} \binom{\mathcal{N}(x_i)}{j} p^{\mathcal{P}(x_i)+\mathcal{N}(x_i)-i-j} (1-p)^{i+j},$$

and

$$\mathbb{E}[\max\{\mathcal{P}_{test}(x_i), \mathcal{N}_{test}(x_i)\}] = \sum_{i=0}^{\mathcal{P}(x_i)} \sum_{j=0}^{\mathcal{N}(x_i)} \max\{i, j\} \binom{\mathcal{P}(x_i)}{i} \binom{\mathcal{N}(x_i)}{j} p^{i+j} (1-p)^{\mathcal{P}(x_i)+\mathcal{N}(x_i)-i-j}.$$

It's important to note that Δ is symmetric; its value is invariant if we exchange the roles of the training and testing subsets $\mathcal{S}_{train}$ and $\mathcal{S}_{test}$. This symmetry is apparent in the equality $\mathbb{E}[\Delta(x_i;p)] = \mathbb{E}[\Delta(x_i;1-p)]$, as both expressions yield the same result.

We observe that Δ exhibits symmetrical behavior, as its value remains invariant under the interchange of the training set $\mathcal{S}_{train}$ and the test set $\mathcal{S}_{test}$. This symmetry property is further substantiated by the equality in expected values for complementary probabilities in the random partitioning process, expressed mathematically as $\mathbb{E}[\Delta(x_i;p)]$.

To enhance the interpretability of experimental outcomes, we derive mathematical representations for two key metrics: the expected maximum accuracy of the test set, denoted as AC^u, and the expected minimum hinge loss for the training set. These metrics, in the context of random partitioning, are defined by the following equations:

- For the expected maximum accuracy of the test set:

$$\mathbb{E}\mathrm{AC}^u = \frac{1}{m(1-p)} \sum_{x_i} \mathbb{E} \max \Big\{ \mathcal{P}_{test}(x_i), \mathcal{N}_{test}(x_i) \Big\} \tag{96}$$

- For the expected minimum hinge loss of the training set:

$$\mathbb{E}[\text{minimum hinge loss}] = \frac{1}{mp} \sum_{x_i} \mathbb{E} \min \Big\{ \mathcal{P}_{train}(x_i), \mathcal{N}_{train}(x_i) \Big\} \tag{97}$$

The expected minimum, present in the equation for hinge loss, is elucidated as follows:

$$\mathbb{E}[\min\{\mathcal{P}_{train}(x_i), \mathcal{N}_{train}(x_i)\}] = \sum_{i=0}^{\mathcal{P}(x_i)} \sum_{j=0}^{\mathcal{N}(x_i)} \min\{i,j\} \binom{\mathcal{P}(x_i)}{i} \binom{\mathcal{N}(x_i)}{j} p^{\mathcal{P}(x_i)+\mathcal{N}(x_i)-i-j} (1-p)^{i+j}. \tag{98}$$

In these expressions, $\mathcal{P}_{train}(x_i)$ and $\mathcal{N}_{train}(x_i)$ denote the number of positive and negative instances of x_i in the training set, respectively, and analogously for $\mathcal{P}_{test}(x_i)$ and $\mathcal{N}_{test}(x_i)$ in the test set. The binomial coefficients reflect the combinatorial possibilities for selecting i positives out of $\mathcal{P}(x_i)$ and j negatives out of $\mathcal{N}(x_i)$, factoring in the probability p of an instance belonging to the training set.

VII. OVERLAPPING AND BOUNDARY

Indeed, training loss and evaluation metrics inherently have a lower bound (0) and an upper bound (1). However, as analyzed in Suppleentary Notes 2 and 3, the precise boundaries do not always align with these natural limits. The reason is that positive and negative samples sometimes overlap, making it impossible for any classifier to achieve 100% prediction accuracy. Consequently, this section will further explore the quantifiable correlation between the degree of overlap among positive and negative samples in a dataset and the performance boundaries. However, prior to this exploration, our first step will be to establish a definition for the term "overlap" within the context of a dataset.

Given positive data distribution $\{\mathcal{P}(x_i)/n_+\}_{x_i \in \mathcal{X}}$ and negative data distribution $\{\mathcal{N}(x_i)/n_-\}_{x_i \in \mathcal{X}}$, these two probability distributions can be abbreviated as

$$\mathcal{P} := \{\hat{p}(x_i) | x_i \in \mathcal{S}\}$$

and

$$\mathcal{Q} := \{\hat{n}(x_i) | x_i \in \mathcal{S}\}$$

respectively, in which $\hat{p}(x_i) = \mathcal{P}(x_i)/n_+$ and $\hat{n}(x_i) = \mathcal{N}(x_i)/n_-$ for every $x_i \in \mathcal{S}$. Considering that Jensen-Shannon divergence is a symmetry and bounded measures to quantify the divergence degree between two probability distributions, we define the overlapping measures between $\mathcal{P}$ and $\mathcal{Q}$ as the complement of $J(\mathcal{P}||\mathcal{N})$, that is,

$$\begin{aligned} D_{\mathcal{S}} =& 1 - J(\mathcal{P}||\mathcal{N}) \\ =& 1 - \frac{1}{2}\Big(\mathrm{KL}(\mathcal{P}||\mathcal{M}) + \mathrm{KL}(\mathcal{N}||\mathcal{M})\Big) \\ =& 1 - \frac{1}{2}\left(\sum_{x_i \in \mathcal{S}} \hat{p}(x_i) \log_2\left(\frac{2\hat{p}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right) + \hat{n}(x_i) \log_2\left(\frac{2\hat{n}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)\right) \\ =& -\frac{1}{2}\left(\sum_{x_i \in \mathcal{S}} \hat{p}(x_i) \log_2\left(\frac{\hat{p}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right) + \hat{n}(x_i) \log_2\left(\frac{\hat{n}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)\right) \end{aligned} \tag{99}$$

where $\mathcal{M} = \frac{1}{2}\Big(\mathcal{P} + \mathcal{N}\Big)$ is a mixture distribution of $\mathcal{P}$ and $\mathcal{N}$, and $\mathrm{KL}(\mathcal{P}||\mathcal{N})$ is the Kullback–Leibler divergence of any two distributions $\mathcal{P}$ and $\mathcal{N}$. Here, $D_{\mathcal{S}}$ is also bounded by zero and one, in particular, $D_{\mathcal{S}} = 0$ means that they are completely separated; $D_{\mathcal{S}} = 1$ means that they are totally overlapped. Therefore, the specific value of $D_{\mathcal{S}} \in [0, 1]$ quantitatively characterizes the degree of overlap between $\mathcal{P}$ and $\mathcal{N}$.

Next, we plan to discover the quantitative relationship between $D_{\mathcal{S}}$ and AR^u through describing the fluctuation of AR^u given a fixed overlapping $D_{\mathcal{S}}$. At first, we define $\mathrm{AR}^u_{\max}(D_{\mathcal{S}})$ and $\mathrm{AR}^u_{\min}(D_{\mathcal{S}})$ as the maximum and minimum values of upper bound of AUC (AR^u) of dataset $\mathcal{S}$ with overlapping $D_{\mathcal{S}}$, respectively. In other words, $\mathrm{AR}^u_{\max}(D_{\mathcal{S}})$ and $\mathrm{AR}^u_{\min}(D_{\mathcal{S}})$ can be obtained through solving the two following optimization problems:

$$\begin{aligned}
\min \quad & \sum_{x_i, x_j \in \mathcal{S}} \max\Big\{\hat{p}(x_i)\hat{n}(x_j), \hat{n}(x_i)\hat{p}(x_j)\Big\} \\
\text{s.t.} \quad & \sum_{x_i \in \mathcal{S}} \hat{p}(x_i) \log_2\left(\frac{\hat{p}(x_i)}{\hat{p}(x_i) + \hat{n}(x_i)}\right) + \hat{n}(x_i) \log_2\left(\frac{\hat{n}(x_i)}{\hat{p}(x_i) + \hat{n}(x_i)}\right) + 2D_{\mathcal{S}} = 0 \\
& \sum_{x_i \in \mathcal{S}} \hat{p}(x_i) = 1 \\
& \sum_{x_i \in \mathcal{S}} \hat{n}(x_i) = 1 \\
& \hat{p}(x_i) \geq 0 \quad i = 1, 2, \cdots, m \\
& \hat{n}(x_i) \geq 0 \quad i = 1, 2, \cdots, m
\end{aligned} \tag{100}$$

and

$$\begin{aligned}
\max \quad & \sum_{x_i, x_j \in \mathcal{S}} \max\Big\{\hat{p}(x_i)\hat{n}(x_j), \hat{n}(x_i)\hat{p}(x_j)\Big\} \\
\text{s.t.} \quad & \sum_{x_i \in \mathcal{S}} \hat{p}(x_i) \log_2\left(\frac{\hat{p}(x_i)}{\hat{p}(x_i) + \hat{n}(x_i)}\right) + \hat{n}(x_i) \log_2\left(\frac{\hat{n}(x_i)}{\hat{p}(x_i) + \hat{n}(x_i)}\right) + 2D_{\mathcal{S}} = 0 \\
& \sum_{x_i \in \mathcal{S}} \hat{p}(x_i) = 1 \\
& \sum_{x_i \in \mathcal{S}} \hat{n}(x_i) = 1 \\
& \hat{p}(x_i) \geq 0 \quad i = 1, 2, \cdots, m \\
& \hat{n}(x_i) \geq 0 \quad i = 1, 2, \cdots, m
\end{aligned} \tag{101}$$

Now, we observed an interesting phenomenon that both AR^u and $D_{\mathcal{S}}$ remain unchanged if we swap the values of $\Big(\hat{p}(x_i), \hat{n}(x_i)\Big)$ and $\Big(\hat{p}(x_j), \hat{n}(x_j)\Big)$. Considering the computational process of AR^u, we assume that $\hat{p}(x_1)/\hat{n}(x_1) \leq \hat{p}(x_2)/\hat{n}(x_2) \leq \cdots \leq \hat{p}(x_m)/\hat{n}(x_m)$ without loss of generality. Under this assumption, the AR^u can be simplified as

$$\mathrm{AR}^u = \sum_{i=1}^{m} \sum_{j=1}^{i-1} \hat{p}(x_i)\hat{n}(x_j) + \frac{1}{2} \sum_{i=1}^{m} \hat{p}(x_i)\hat{n}(x_i). \tag{102}$$

Combined with the above assumption and equation, Eq. 100 and Eq. 101 can be rewritten as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m}\sum_{j=1}^{i-1}\hat{p}(x_i)\hat{n}(x_j)+\frac{1}{2}\sum_{i=1}^{m}\hat{p}(x_i)\hat{n}(x_i)\\
\text{s.t.} \quad & \sum_{i=1}^{m}\hat{p}(x_i)\log_2\left(\frac{\hat{p}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)+\hat{n}(x_i)\log_2\left(\frac{\hat{n}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)+2D_{\mathcal{S}}=0\\
& \hat{p}(x_1)/\hat{n}(x_1)\leq\hat{p}(x_2)/\hat{n}(x_2)\leq\cdots\leq\hat{p}(x_m)/\hat{n}(x_m)\\
& \sum_{i=1}^{m}\hat{p}(x_i)=1\\
& \sum_{i=1}^{m}\hat{n}(x_i)=1\\
& \hat{p}(x_i)\geq 0\quad i=1,2,\cdots,m\\
& \hat{n}(x_i)\geq 0\quad i=1,2,\cdots,m
\end{aligned}\tag{103}$$

and

$$\begin{aligned}
\max \quad & \sum_{i=1}^{m}\sum_{j=1}^{i-1}\hat{p}(x_i)\hat{n}(x_j)+\frac{1}{2}\sum_{i=1}^{m}\hat{p}(x_i)\hat{n}(x_i)\\
\text{s.t.} \quad & \sum_{i=1}^{m}\hat{p}(x_i)\log_2\left(\frac{\hat{p}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)+\hat{n}(x_i)\log_2\left(\frac{\hat{n}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)+2D_{\mathcal{S}}=0\\
& \hat{p}(x_1)/\hat{n}(x_1)\leq\hat{p}(x_2)/\hat{n}(x_2)\leq\cdots\leq\hat{p}(x_m)/\hat{n}(x_m)\\
& \sum_{i=1}^{m}\hat{p}(x_i)=1\\
& \sum_{i=1}^{m}\hat{n}(x_i)=1\\
& \hat{p}(x_i)\geq 0\quad i=1,2,\cdots,m\\
& \hat{n}(x_i)\geq 0\quad i=1,2,\cdots,m
\end{aligned}\tag{104}$$

respectively. According to the symmetry of the ranking assumption

$$\frac{\hat{p}(x_1)}{\hat{n}(x_1)}\leq\frac{\hat{p}(x_2)}{\hat{n}(x_2)}\leq\cdots\leq\frac{\hat{p}(x_m)}{\hat{n}(x_m)},$$

it follows that the feasible region is partitioned into $m!$ pairwise symmetric sub-regions. Within each sub-region, the mathematical formulation of $\text{AR}^{u}_{\max}(D_{\mathcal{S}})$ is invariant and possesses an identical maximum value. Consequently, we can disregard the ranking assumption in Eq. (104) and deduce a simplified version as follows:

$$\begin{aligned}
\max \quad & \sum_{i=1}^{m}\sum_{j=1}^{i-1}\hat{p}(x_i)\hat{n}(x_j)+\frac{1}{2}\sum_{i=1}^{m}\hat{p}(x_i)\hat{n}(x_i),\\
\text{s.t.} \quad & \sum_{i=1}^{m}\hat{p}(x_i)\log_2\left(\frac{\hat{p}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)+\hat{n}(x_i)\log_2\left(\frac{\hat{n}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)+2D_{\mathcal{S}}=0,\\
& \sum_{i=1}^{m}\hat{p}(x_i)=1,\\
& \sum_{i=1}^{m}\hat{n}(x_i)=1,\\
& \hat{p}(x_i)\geq 0,\quad i=1,2,\cdots,m,\\
& \hat{n}(x_i)\geq 0,\quad i=1,2,\cdots,m.
\end{aligned}\tag{105}$$

We employ the SLSQP solver [5] to resolve the optimization problem outlined in (105) and acquire the corresponding numerical solution for $\mathrm{AR}^u_{\max}$. In Fig. S8A, the AR^u curve versus $D_{\mathcal{S}}$ is depicted. Each datum point on the curve represents the numerical solution of the optimization problem under specific parameters. Notably, the curve converges swiftly as m increases, and attains the optimal $\mathrm{AR}^u_{\max}$ curve when $m \geq 10$.

Regarding the $\mathrm{AR}^u_{\min}$ curve, a heuristic approach is utilized to construct its optimal solution from the feasible region's boundaries. Specifically, we examine a particular case where $\hat{p}(x_1) = 1 - b$, $\hat{p}(x_2) = b$, $\hat{n}(x_1) = 0$, and $\hat{n}(x_2) = 1$, with b being a tunable parameter in the interval $[0, 1]$. In this scenario, the AR^u is expressed as

$$\mathrm{AR}^u = 1 - \frac{b}{2}, \tag{106}$$

and $D_{\mathcal{S}}$ is given by

$$\begin{aligned} D_{\mathcal{S}} &= -\frac{1}{2}\left(b \log_2 \frac{b}{b+1} + \log_2 \frac{1}{b+1}\right) \\ &= -\frac{1}{2}\left(b \log_2 b - (b+1)\log_2(b+1)\right). \end{aligned} \tag{107}$$

Furthermore, the numerical curve of $\mathrm{AR}^u_{\min}$ obtained through the SLSQP solver aligns closely with this heuristic solution as given in (107), providing partial validation for our heuristic approach (see Fig. S8B).

VIII. FEATURE ENGINEERING

In previous sections, we have discussed the boundaries of the objective function and evaluation metrics from the perspective of row data (feature vectors). In fact, column data (features) can also influence the degree of overlap and boundaries in a dataset through their impact on row data. In feature engineering, there are two classic methods of handling column data: feature selection and feature extraction. The former emphasizes adding or removing new features unrelated to existing ones, and the latter is based on extraction and mapping of original features. In this section, we will discuss these two methods separately. But before that, we need to discuss the simplest case first.

Suppose we have an original dataset $\mathcal{S} = \{(x_i, y_i) : i = 1, 2, \cdots, m\}$ with k features. This implies that the feature vector x_i can be expressed as $x_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,k})$. We also define that there are $\mathcal{P}(x_i)$ positive instances and $\mathcal{N}(x_i)$ negative instances with feature vector x_i in the entire dataset. By introducing a new feature into every instance, we can create a new dataset $\mathcal{S}'$. Using feature vector x_i as an example, these $\mathcal{P}(x_i) + \mathcal{N}(x_i)$ could be added into different feature values, which are included in $\{x_i^s = (x_{i,1}, x_{i,2}, \cdots, x_{i,k}, x_{i,k+1}^s) : s = 1, 2, \cdots, s_i\}$. For the sake of argument, we illustrate that the original feature vector is split into s_i pairwise distinct feature vectors after the addition of one row data, satisfying that

$$\begin{cases} \sum_{j=1}^{s_i} \mathcal{P}(x_i^j) = \mathcal{P}(x_i) \\ \sum_{j=1}^{s_i} \mathcal{N}(x_i^j) = \mathcal{N}(x_i) \end{cases}, \tag{108}$$

in which s_i is defined as the diversity for x_i. Consequently, we will proceed to prove the following lemma.

Lemma 1. *Upon the inclusion of a new feature into the original dataset, AR^u will either increase or remain constant, while $D_{\mathcal{S}}$ will either decrease or stay the same. These values will remain unchanged if, and only if, the diversity $s_i = 1$ for each x_i.*

Proof. The upper bound of AUC in the original dataset $\mathcal{S}$

$$\mathrm{AR}^u_{original} = \frac{1}{2n_-n_+} \sum_{i,j} \max\{\mathcal{P}(x_i)\mathcal{N}(x_j), \mathcal{P}(x_j)\mathcal{N}(x_i)\}, \tag{109}$$

and the new boundary is

$$\begin{aligned}\mathrm{AR}^u_{new} =&\frac{1}{2n_-n_+}\sum_{i,j}\sum_{i'=1}^{s_i}\sum_{j'=1}^{s_j}\max\{\mathcal{P}(x_i^{i'})\mathcal{N}(x_j^{j'}),\mathcal{P}(x_j^{j'})\mathcal{N}(x_i^{i'})\}\\ \geq&\frac{1}{2n_-n_+}\sum_{i,j}\max\left\{\sum_{i'=1}^{s_i}\mathcal{P}(x_i^{i'})\sum_{j'=1}^{s_j}\mathcal{N}(x_j^{j'}),\sum_{j'=1}^{s_j}\mathcal{P}(x_j^{j'})\sum_{i'=1}^{s_i}\mathcal{N}(x_i^{i'})\right\}.\\ =&\frac{1}{2n_-n_+}\sum_{i,j}\max\{\mathcal{P}(x_i)\mathcal{N}(x_j),\mathcal{P}(x_j)\mathcal{N}(x_i)\}\\ =&\mathrm{AR}^u_{original}\end{aligned} \tag{110}$$

Similarly, the original $D_{\mathcal{S}}$ can be written as

$$D_{\mathcal{S}}^{original} = -\frac{1}{2}\left(\sum_{x_i\in\mathcal{S}}\hat{p}(x_i)\log_2\left(\frac{\hat{p}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)+\hat{n}(x_i)\log_2\left(\frac{\hat{n}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)\right) \tag{111}$$

and the new overlapping is

$$\begin{aligned}D_{\mathcal{S}}^{new} =&-\frac{1}{2}\left(\sum_{x_i\in\mathcal{S}}\sum_{i'=1}^{s_i}\hat{p}(x_i^{i'})\log_2\left(\frac{\hat{p}(x_i^{i'})}{\hat{p}(x_i^{i'})+\hat{n}(x_i^{i'})}\right)+\hat{n}(x_i^{i'})\log_2\left(\frac{\hat{n}(x_i^{i'})}{\hat{p}(x_i^{i'})+\hat{n}(x_i^{i'})}\right)\right)\\ \leq&-\frac{1}{2}\left(\sum_{x_i\in\mathcal{S}}\sum_{i'=1}^{s_i}\hat{p}(x_i^{i'})\log_2\left(\frac{\sum_{i'=1}^{s_i}\hat{p}(x_i^{i'})}{\sum_{i'=1}^{s_i}\hat{p}(x_i^{i'})+\hat{n}(x_i^{i'})}\right)+\sum_{i'=1}^{s_i}\hat{n}(x_i^{i'})\log_2\left(\frac{\sum_{i'=1}^{s_i}\hat{n}(x_i^{i'})}{\sum_{i'=1}^{s_i}\hat{p}(x_i^{i'})+\hat{n}(x_i^{i'})}\right)\right)\\ =&-\frac{1}{2}\left(\sum_{x_i\in\mathcal{S}}\hat{p}(x_i)\log_2\left(\frac{\hat{p}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)+\hat{n}(x_i)\log_2\left(\frac{\hat{n}(x_i)}{\hat{p}(x_i)+\hat{n}(x_i)}\right)\right)\\ =&D_{\mathcal{S}}^{original}.\end{aligned} \tag{112}$$

They are equal to each other if and only if there exiSUD i' satisfying that $\hat{p}(x_i^{i'})=\hat{p}(x_i)$ and $\hat{n}(x_i^{i'})=\hat{n}(x_i)$ for every x_i, i.e., $s_i=1$. □

Actually, adding new features will cause the overlapping of the positive and negative samples of the dataset to decrease or remain unchanged, while reducing the original features will cause the overlapping to increase or remain unchanged. The same conclusion is also applicable to the boundaries of various evaluation indicators and loss functions, such as AR^u, AP^u and AC^u.

A. Feature Selection

Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. A feature selection algorithm can be seen as the combination of a search technique for proposing new feature subsets, along with an evaluation measure which scores the different feature subsets. The simplest algorithm is to test each possible subset of features finding the one which minimizes the error rate to reach the best performance. Actually, the exponential number of potential subset selection and the huge amount of training cost for any classifier make this process difficult. However, the boundary theory we proposed can better select subset of all features with high performance measured by AR, AP (best ranking) and AC and low computational cost (time complexity).

Given an integer k_0, the optimal k_0 feature subset is targeted by AR^u. This implies that we can directly compute the AR^u for each feature subset in the entire dataset $\mathcal{S}$ to assess the data quality and training potential, bypassing the need for a training process. This approach significantly conserves storage space and computational resources. We denote AR$^u_{k_0}$ and $D_{\mathcal{S}}^{k_0}$ as the optimal AR^u and $D_{\mathcal{S}}$, respectively, when we traverse all possible k_0 feature subsets. The corresponding feature subset is named by *optimal feature subset*. Based on Lemma 1 and the principle of recursion, it is evident that

Theorem 1. *$AR^u_{k_0}$ exhibits a monotonic non-decreasing trend and $D_{\mathcal{S}}^{k_0}$ is monotonic non-increasing when $k_0\in\{1,2,\cdots,k\}$. Here, k represents the total number of features in the entire dataset.*

Proof. Based on the principle of recursion, we only need to prove that $\text{AR}^u_{k_0} \leq \text{AR}^u_{k_0+1}$ and $D^{k_0}_{\mathcal{S}} \geq D^{k_0+1}_{\mathcal{S}}$ for any $k_0 \in \{1, 2, \cdots, k-1\}$. Assumed that $\mathcal{F}^*_{k_0} = \{f^*_1, f^*_2, \cdots, f^*_{k_0}\}$ is the optimal k_0-feature subset. Then we construct a new k_0+1-feature subset $\mathcal{F}_{k_0+1} = \mathcal{F}^*_{k_0} + \{f_i\}$, in which f_i is any selected feature not belonging to $\mathcal{F}^*_{k_0}$. According to Lemma 1, we know that the AR^u of $\mathcal{F}_{k_0+1}$ is higher than or equal to $\text{AR}^u_{k_0}$. At the same time, we also know that the AR^u of $\mathcal{F}_{k_0+1}$ is lower than or equal to $\text{AR}^u_{k_0+1}$ since the definition of $\text{AR}^u_{k_0+1}$. Therefore, we successfully proved that $\text{AR}^u_{k_0} \leq \text{AR}^u_{k_0+1}$. In a similar way, we can also prove $D^{k_0}_{\mathcal{S}} \geq D^{k_0+1}_{\mathcal{S}}$. □

Theorem 1 elucidates the direct correlation between the performance bounds and the overlapping index within a given dataset. Specifically, it reveals that an increase in the number of features (raw data) leads to a reduction in the overlap between the positive and negative sample distributions, which in turn enhances the performance boundaries. This insight informs the design of a feature selection algorithm that leverages the overlapping index, $D_{\mathcal{S}}$, to ensure the achievement of the highest possible performance upper limit.

Consider a dataset $\mathcal{S}$ composed of k features $\{F_1, F_2, \ldots, F_k\}$. The optimal feature subset of size k_0, denoted as the subset that minimizes $D_{\mathcal{S}}$ (or maximizes AR^u), can be defined where $k_0 = 1, 2, \ldots, k$. However, exhaustively evaluating all $\binom{k}{k_0}$ possible subsets may be computationally prohibitive for large k. To address this, approximation techniques like dynamic programming can be employed to devise an efficient approximation algorithm.

For a more practical example, consider the INE dataset with 13 features; it is possible to achieve the dataset's performance boundary using only 8 features, such that $D^8_{\mathcal{S}} = D_{\mathcal{S}}$. This indicates that the remaining five features are, to some extent, superfluous. We thus define the optimal feature selection dimension k^* as:

$$k^* = \min\{k_0 : D^{k_0}_{\mathcal{S}} = D_{\mathcal{S}}\}. \tag{113}$$

The feature subset corresponding to k^* is referred to as the *global optimal feature subset.*

B. Feature Extraction

Feature extraction is the procedure of deriving features (traits, properties, attributes) from raw data. It is seen as an equivalent transformation that creates new features from the original ones. In previous subsections, we deduced that the boundary of performance is dictated by the data structure. From a mathematical standpoint, feature extraction can be viewed as a mapping of original features (row data). If we incorporate new extracted data into the original dataset, the diversity for each feature vector is 1. In conjunction with Lemma 1, we can state,

Theorem 2. *The inclusion of extracted features into the original dataset does not alter the boundary of training loss, evaluation measures, and overlapping.*

Proof. Any feature extraction process can be regarded as a mapping from existing feature vectors, so the diversity is 1. Combined with Lemma 1, the boundaries and overlapping should be unchanged. □

C. Feature Generated from Other Samples

Both aforementioned two processes involve manipulating each sample's original feature vector through operations such as selection, transformation, or extraction. Drawing an analogy to clustering algorithms in unsupervised learning, we here explore from the perspective of rows: augmenting samples' features with information derived from other samples, rather than itself, based on agreed rules. E.g. propose a new indicator of a sample, neighbors' income, by counting all her/his neighborhoods within certain distance [6].

For each sample $x_i \in \mathcal{S}$, here we define its new feature vector is $x'_i = (x_{i,1}, x_{i_2}, \cdots, x_{i,2k})$, in which $(x_{i,1}, x_{i_2}, \cdots, x_{i,k})$ is its original k-feature vector and

$$(x_{i,k+1}, x_{i_k+2}, \cdots, x_{i,2k}) = f(\Lambda_r(x_i)). \tag{114}$$

And, $\Lambda_r(x_i)$ includes all samples whose distance from x_i is less than r in the original feature space, and f represents an arbitrary operator, such as a mean function.

We then construct a new dataset $\mathcal{S}'_r = \{x'_1, x'_2, \cdots, x'_n\}$ with a tunable parameter $r \in [0, \infty]$. Notably, $\mathcal{S}'_0$ is equivalent to $\mathcal{S}$. We can now state the following theorem:

Theorem 3. *The AR^u of $\mathcal{S}'_r$ is always equal to $\mathcal{S}$ regardless of r.*

Proof. Based on Theorem 1, it is evident that the AR^u of $\mathcal{S}_r'$ is at least as large as that of $\mathcal{S}$. Additionally, for any two samples with identical original feature vectors, their newly added features will also be identical. According to Lemma 1, we can say that the AR^u of $\mathcal{S}_r'$ cannot exceed that of $\mathcal{S}$. Hence, the theorem is proven. $\square$

In this section, we use the AR^u measure to characterize the predictability of a given dataset, as demonstrated in Lemma 1, Theorem 1, and Theorem 2. Notably, other measures of predictability, such as AP^u and AC^u, lead to the same conclusions. The equivalence of these different measures will be thoroughly explained in Section III.

IX. DATASETS

We utilized 4 datasets in the main text and 37 additional datasets from Kaggle (`https://www.kaggle.com`). The specifics of the four real-world datasets used in the main text are as follows:

- Airlines Delay Dataset (AID): comprised of 539,383 records across 8 distinct attributes, the objective is to forecast flight delays based on scheduled departure information.
- Heart Disease Dataset (HED): This dataset encompasses a wide range of cardiovascular risk factors, including age, gender, height, weight, blood pressure, cholesterol and glucose levels, smoking status, alcohol intake, physical activity, and presence of cardiovascular diseases, from over 70,000 individuals. It serves as a valuable asset for applying advanced machine learning methods to investigate the link between these factors and cardiovascular health, which could enhance disease understanding and prevention strategies.
- Income Classification Dataset (INE): This dataset features variables such as education, employment, and marital status to predict whether an individual earns more than $50K annually.
- Student Sleep Study Dataset (SUD): Originating from a survey-based analysis of US students' sleep patterns, this dataset utilizes factors like average sleep duration and phone usage time to infer adequate sleep among students.

We also conducted experiments on 37 real-world binary datasets to additionally validate the universality of the proposed theory. Detailed data descriptions and corresponding experimental results are shown at `https://github.com/Feijing92/binary`.

[1] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri, Neural Computation **16**, 1063 (2004).
[2] T. Fawcett, Pattern Recognition Letters **27**, 861 (2006).
[3] T. Yang and Y. Ying, ACM Computing Surveys **55**, 1 (2022).
[4] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The Elements of Statistical Learning* (Springer, 2009).
[5] D. Kraft, *A Software Package for Sequential Quadratic Programming*, Tech. Rep. (DFVLR-FB 88-28, 1988).
[6] F. Hedefalk and M. Dribe, Proceedings of the National Academy of Sciences, USA **117**, 14918 (2020).

SUPPLEMENTARY TABLES AND FIGURES

Table S1. Description of 4 real datasets we used in this Letter. It includes the number of instances (m), the number of positive instances (n_+), the number of negative instances (n_-), the number of features (k), the overlapping index ($D_{\mathcal{S}}$) and the optimal feature selection dimension (k^*).

Name	m	n_+	n_-	k	$D_{\mathcal{S}}$	k^*
AID	539383	299119	240264	7	0.4837	5
HED	70000	34979	35021	11	0.0015	5
INE	32561	7841	24720	13	0.0657	12
SUD	104	36	68	5	0.3181	5

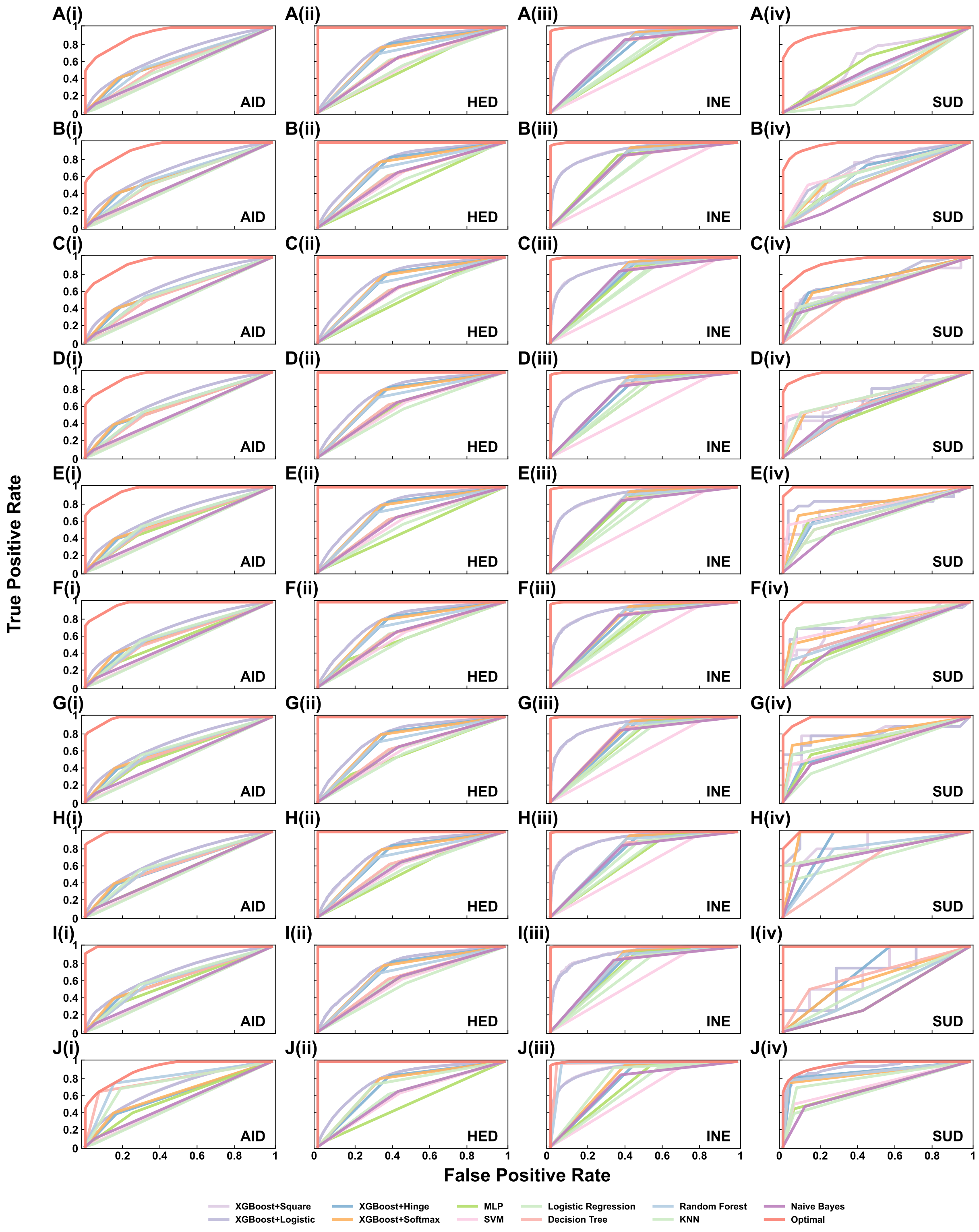

Figure S1. Exact upper bound of AUC and corresponding optimal ROC curves for four real-world datasets when $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.1$ (A), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.2$ (B), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.3$ (C), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.4$ (D), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.5$ (E), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.6$ (F), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.7$ (G), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.8$ (H), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.9$ (I), $|\mathcal{S}_{train}|/|\mathcal{S}| = 1$ (J). The binary classifiers we used in this experiment include XGBoost, MLP, SVM, Logistic Regresion, Decision Tree, Random Forest, KNN and Naive Bayes. Red curves represent the theoretical optimal ROC curves.

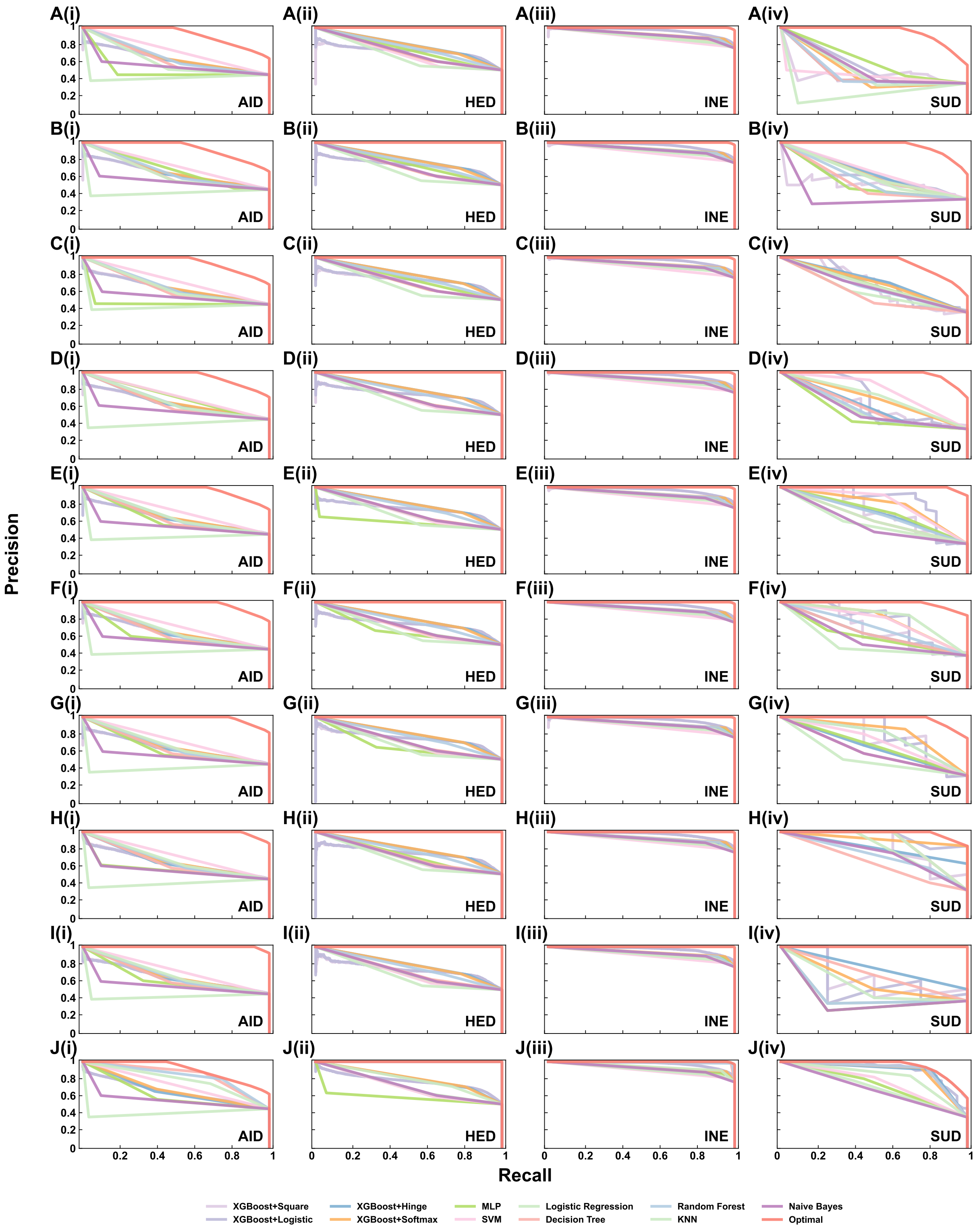

Figure S2. Exact upper bound of AP and corresponding optimal PR curves for four real-world datasets when $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.1$ (A), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.2$ (B), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.3$ (C), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.4$ (D), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.5$ (E), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.6$ (F), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.7$ (G), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.8$ (H), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.9$ (I), $|\mathcal{S}_{train}|/|\mathcal{S}| = 1$ (J). The binary classifiers we used in this experiment include XGBoost, MLP, SVM, Logistic Regresion, Decision Tree, Random Forest, KNN and Naive Bayes. Red curves represent the theoretical optimal PR curves.

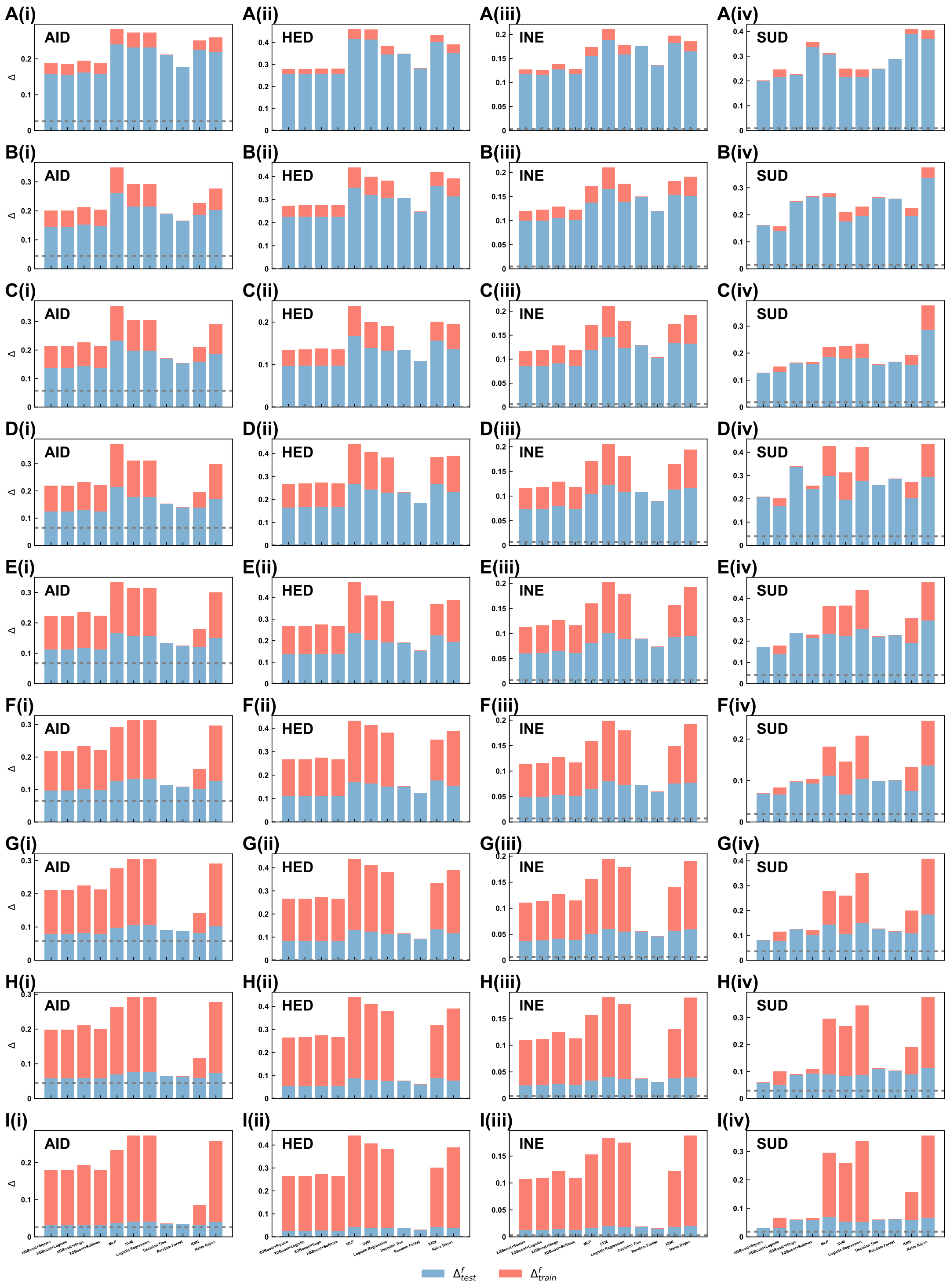

Figure S3. The loss errors of for four datasets in training (Δ^f_{train}) and test sets (Δ^f_{test}) when $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.1$ (A), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.2$ (B), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.3$ (C), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.4$ (D), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.5$ (E), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.6$ (F), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.7$ (G), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.8$ (H), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.9$ (I). Dash line represents the expected error of optimal classier based on Eq. 94.

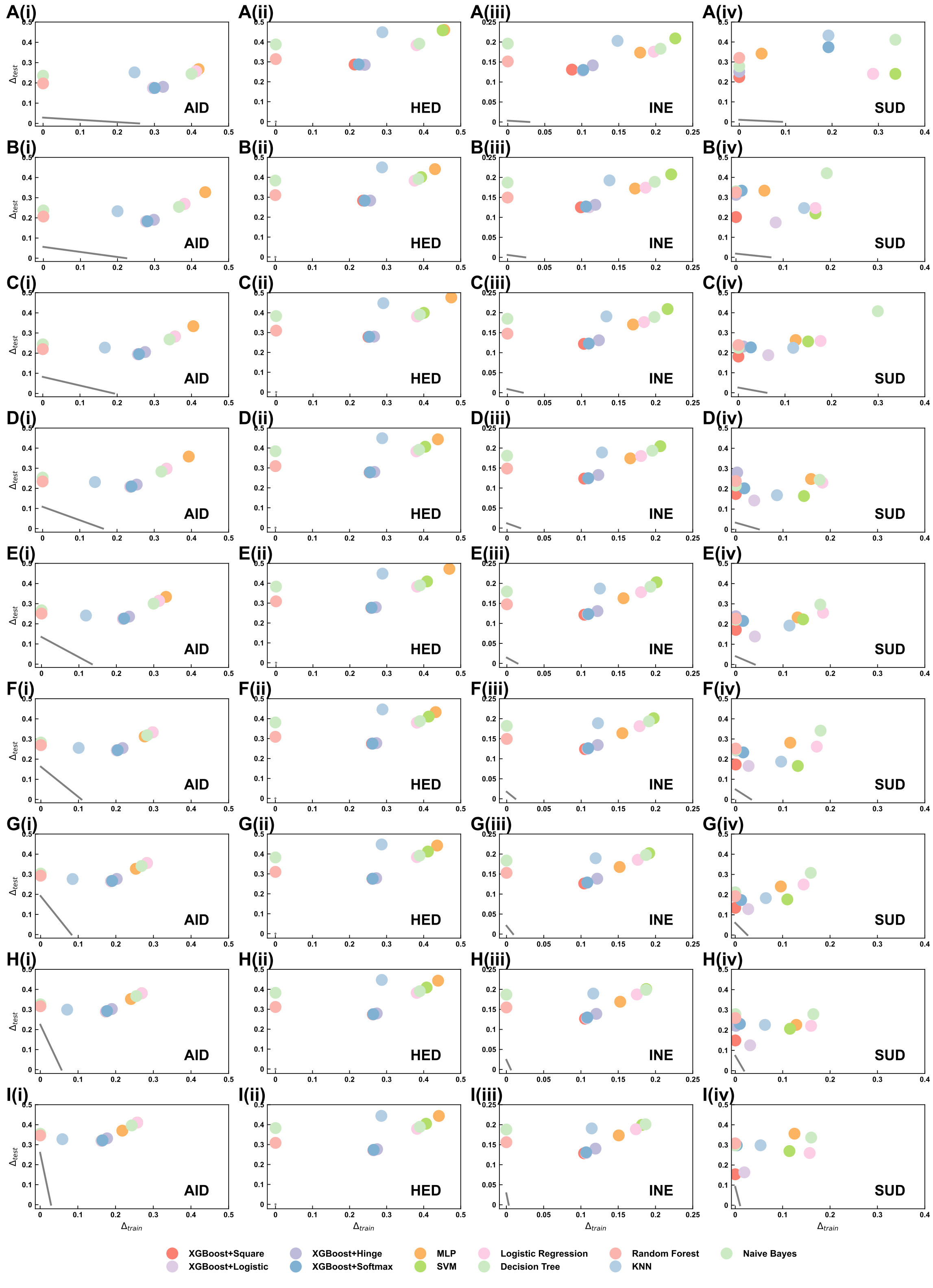

Figure S4. The loss errors of four datasets in training (Δ^f_{train}) and test sets (Δ^f_{test}) when $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.1$ (A), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.2$ (B), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.3$ (C), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.4$ (D), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.5$ (E), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.6$ (F), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.7$ (G), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.8$ (H), $|\mathcal{S}_{train}|/|\mathcal{S}| = 0.9$ (I). Gray line represents the expected error of optimal classier based on Eq. 94.

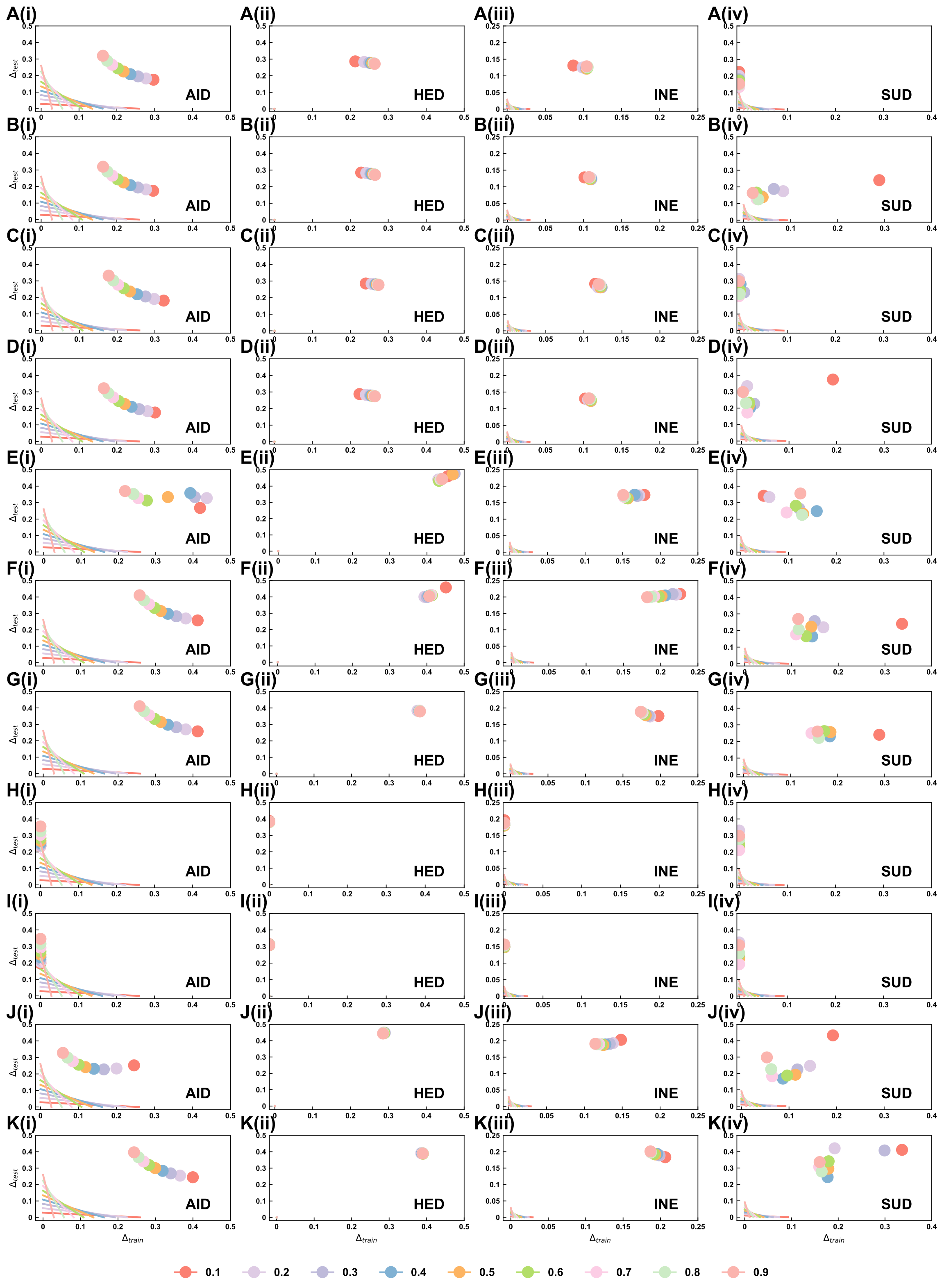

Figure S5. The loss errors of four datasets (AID, HED, INE and SUD) in training (Δ^{f}_{train}) and test sets (Δ^{f}_{test}) of different binary classifiers, including XGBoost with four classical objectives (A-D), MLP (E), SVM (F), Logistic Regression (G), Decision Tree (H), Random Forest (I), KNN (J). Colorful dots and lines represent different $|\mathcal{S}_{train}|/|\mathcal{S}|$ ranging from 0.1 to 0.9.

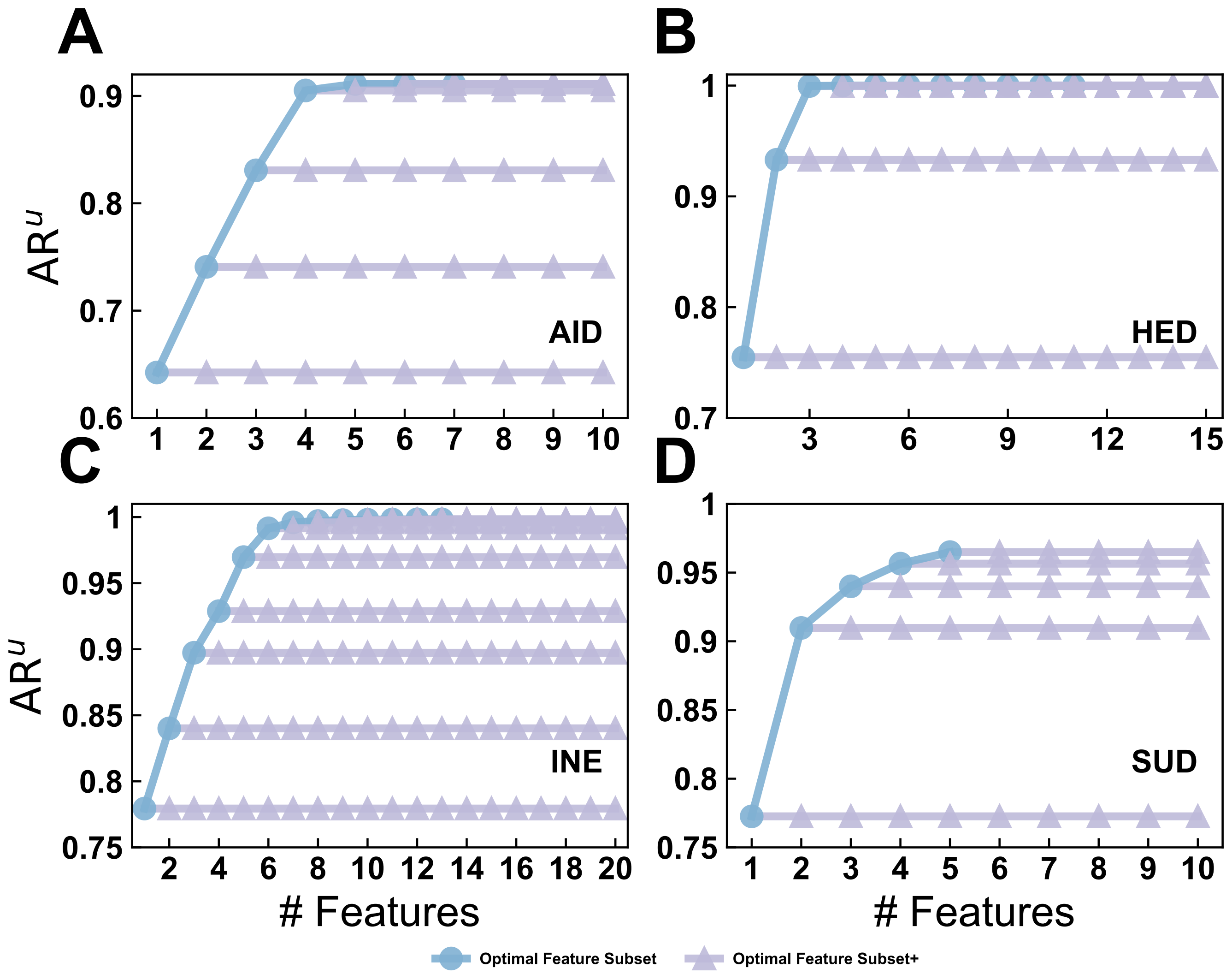

Figure S6. The $AR^u_{k_0}$ versus the optimal k_0 feature subset in feature selection (blue lines and dots). After we selected the optimal k_0 feature subset, we would use the feature extraction skill (LDA) to create new extracted features and add them into the original k_0 feature one by one (see red lines and dots). The datasets we used in this experiment includes AID (A), HED (B), INE (C) and SUD (D).

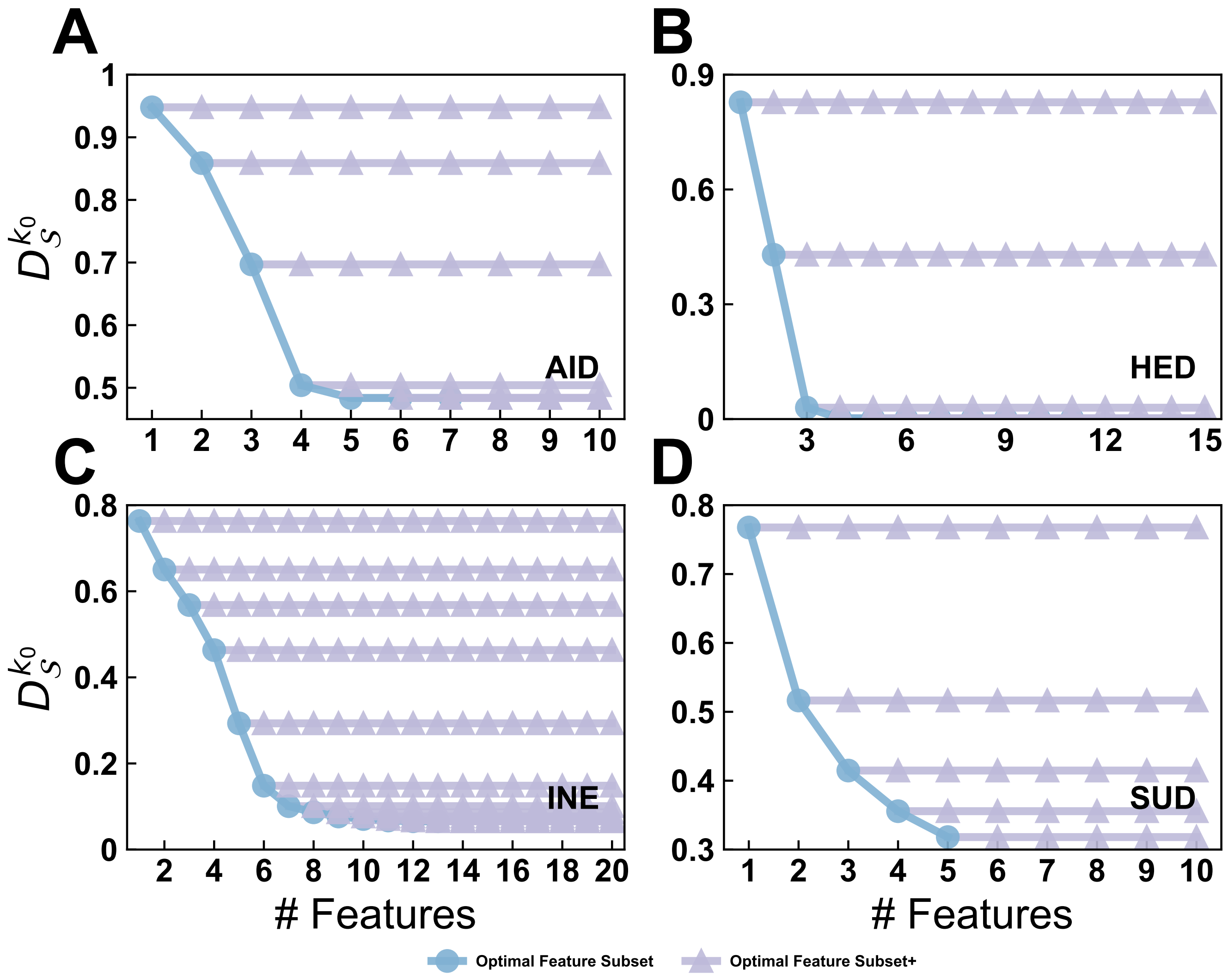

Figure S7. The $D_S^{k_0}$ versus the optimal k_0 feature subset in feature selection (blue lines and dots). After we selected the optimal k_0 feature subset, we would use the feature extraction skill (LDA) to create new extracted features and add them into the original k_0 feature one by one (see red lines and dots). The datasets we used in this experiment includes AID (A), HED (B), INE (C) and SUD (D).

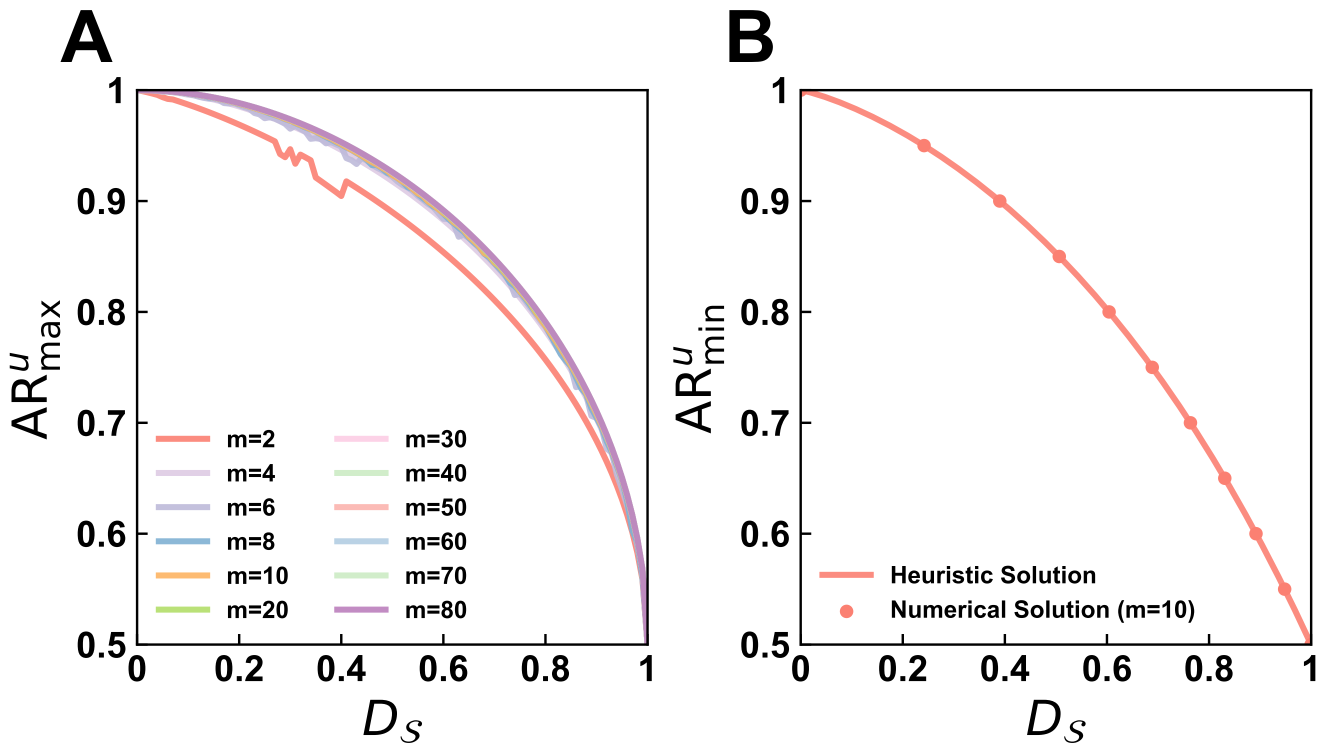

Figure S8. $\mathrm{AR}^u_{\max}(D_\mathcal{S})$ curve (A) and $\mathrm{AR}^u_{\min}(D_\mathcal{S})$ curve (B). (A) $\mathrm{AR}^u_{\max}(D_\mathcal{S})$ curves is numerically solved by the SLSQP solver when $m = 2, 4, 6, 8, 10, 20, 30, 40, 50, 60, 70, 80$. We find that these curves are quickly converged as m increases. Therefore the final $\mathrm{AR}^u_{\max}(D_\mathcal{S})$ curve can be approximated to the numerically solved curve when $m = 10$. (B) The heuristic curve for $\mathrm{AR}^u_{\min}(D_\mathcal{S})$ is calculated by (107). And the numerically approximated $\mathrm{AR}^u_{\min}(D_\mathcal{S})$ curve is derived by the SLSQP solver.

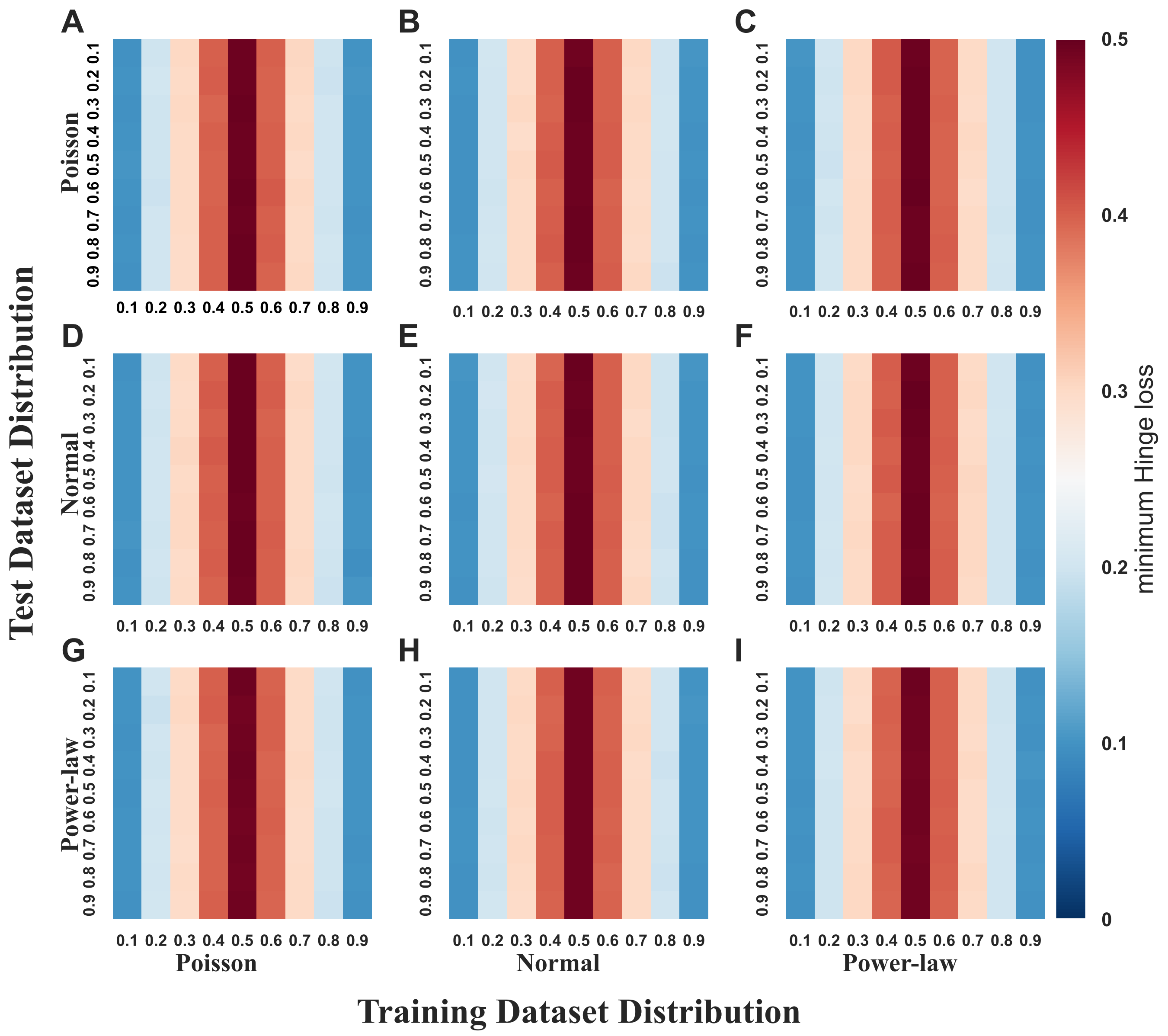

Figure S9. Minimum Hinge loss across synthesized datasets with varied label distributions. Each dataset is bifurcated into distinct training and testing subsets, synthesized using a trio of statistical distributions: Poisson, Normal (Gaussian), and Power-law. Feature vectors for training and testing are generated accordingly. The binary labels are assigned with a probability p for class 1 and $1-p$ for class 0, where p spans the set $\{0.1, 0.2, \ldots, 0.9\}$. This process yields nine unique dataset configurations, denoted as: Poisson & Poisson (A), Normal & Poisson (B), Power-law & Poisson (C), Poisson & Normal (D), Normal & Normal (E), Power-law & Normal (F), Poisson & Power-law (G), Normal & Power-law (H) and Power-law & Power-law (I). Each subset of this matrix is visualized as a heat map, charting the minimal Hinge loss achieved across varying label probabilities within the respective training and testing subsets, under specific distribution pairings. Notably, the minimal Hinge loss for the training subset is computed independently of the testing subset, resulting in identical columns for each heat map.

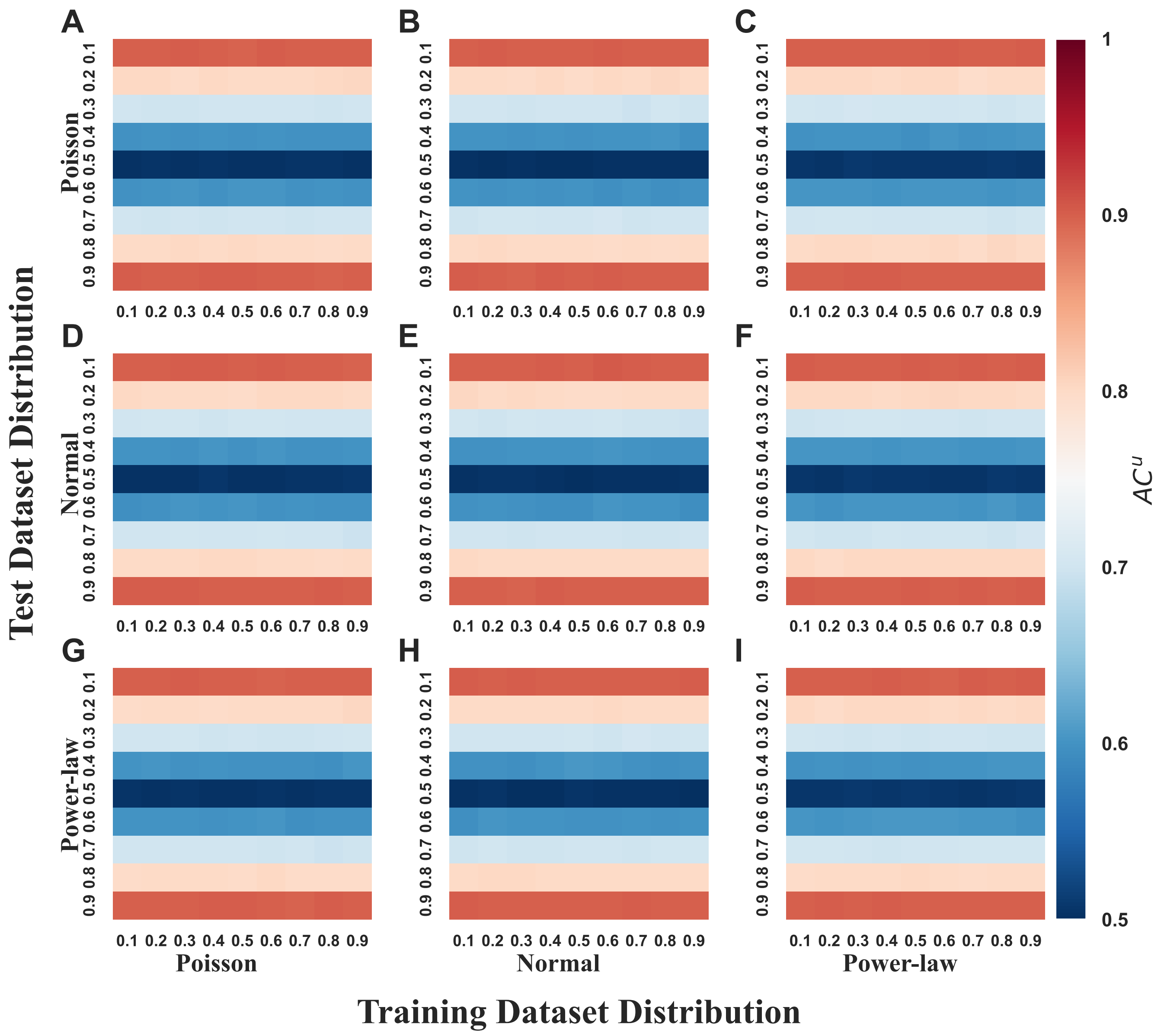

Figure S10. Maximum accuracy (AC^u) across synthesized datasets with varied label distributions. Each dataset is bifurcated into distinct training and testing subsets, synthesized using a trio of statistical distributions: Poisson, Normal (Gaussian), and Power-law. Feature vectors for training and testing are generated accordingly. The binary labels are assigned with a probability p for class 1 and $1-p$ for class 0, where p spans the set $\{0.1, 0.2, \ldots, 0.9\}$. This process yields nine unique dataset configurations, denoted as: Poisson & Poisson (A), Normal & Poisson (B), Power-law & Poisson (C), Poisson & Normal (D), Normal & Normal (E), Power-law & Normal (F), Poisson & Power-law (G), Normal & Power-law (H) and Power-law & Power-law (I). Each subset of this matrix is visualized as a heat map, charting the maximum accuracy (AC^u) achieved across varying label probabilities within the respective training and testing subsets, under specific distribution pairings. Notably, the maximum accuracy for the testing subset is computed independently of the training subset, resulting in identical rows for each heat map.

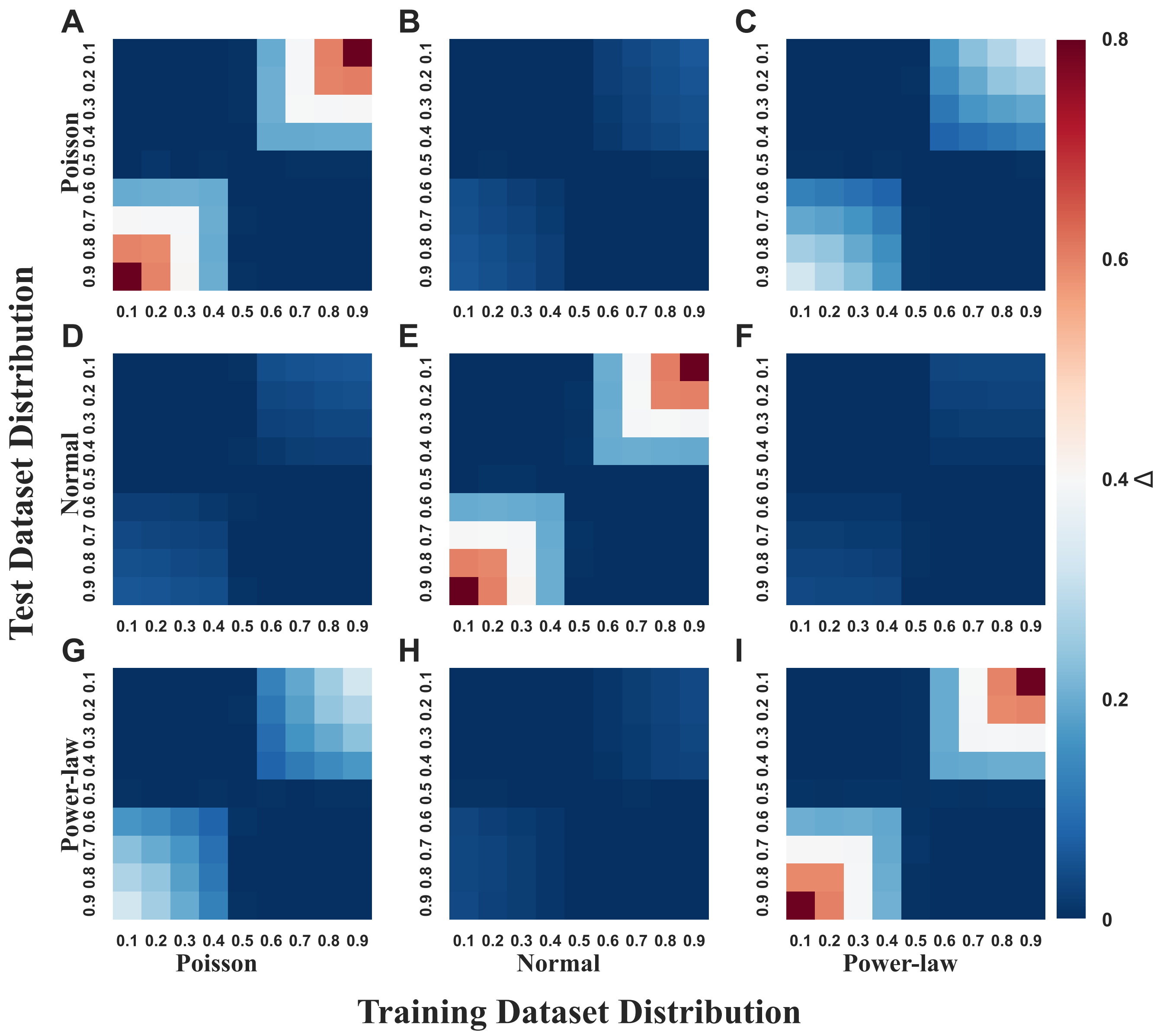

Figure S11. Joint error bound (Δ) across synthesized datasets with varied label distributions. Each dataset is bifurcated into distinct training and testing subsets, synthesized using a trio of statistical distributions: Poisson, Normal (Gaussian), and Power-law. Feature vectors for training and testing are generated accordingly. The binary labels are assigned with a probability p for class 1 and $1-p$ for class 0, where p spans the set $\{0.1, 0.2, \ldots, 0.9\}$. This process yields nine unique dataset configurations, denoted as: Poisson & Poisson (A), Normal & Poisson (B), Power-law & Poisson (C), Poisson & Normal (D), Normal & Normal (E), Power-law & Normal (F), Poisson & Power-law (G), Normal & Power-law (H) and Power-law & Power-law (I). Each subset of this matrix is visualized as a heat map, charting the joint error bound (Δ) achieved across varying label probabilities within the respective training and testing subsets, under specific distribution pairings. Notably, Δ for the training and testing subset is computed based on Eq. (54) for each heat map.