# Alioth: An Efficient and Secure Weight-of-Evidence Framework for Privacy-Preserving Data Processing

Ye Dong
National University of Singapore
Singapore, Singapore
dongye@nus.edu.sg

Xiangfu Song
Nanyang Technological University
Singapore, Singapore
xiangfu.song@ntu.edu.sg

W.j Lu
Independent Researcher
Hangzhou, China
fionser@gmail.com

Xudong Chen
Institute of Information Engineering, Chinese Academy of Sciences
Beijing, China
chenxudong@iie.ac.cn

Yaxi Yang
Nanyang Technological University
Singapore, Singapore
yaxi.yang@ntu.edu.sg

Ruonan Chen
Xidian University
Xi'an, China
chenruonan@xidian.edu.cn

Tianwei Zhang
Nanyang Technological University
Singapore, Singapore
tianwei.zhang@ntu.edu.sg

Jin-Song Dong
National University of Singapore
Singapore, Singapore
dcsdjs@nus.edu.sg

## ABSTRACT

Secure two-party computation (2PC)-based privacy-preserving machine learning (ML) has made remarkable progress in recent years. However, most existing works overlook the privacy challenges that arise during the data preprocessing stage. Although some recent studies have introduced efficient techniques for privacy-preserving feature selection and data alignment on well-structured datasets, they still fail to address the privacy risks involved in transforming raw data features into ML-effective numerical representations.

In this work, we present Alioth, an efficient 2PC framework that securely transforms raw categorical and numerical features into Weight-of-Evidence (WoE)-based numerical representations under both vertical and horizontal data partitions. By incorporating our proposed partition-aware 2PC protocols and vectorization optimizations, Alioth efficiently generates WoE-transformed datasets in secret. To demonstrate scalability, we conduct experiments on diverse datasets. Notably, Alioth can transform 3 million data samples with 100 features securely within half an hour over a wide-area network. Furthermore, Alioth can be seamlessly integrated with existing 2PC-based ML frameworks. Empirical evaluations on real-world financial datasets show Alioth improves both the predictive performance of logistic regression and 2PC training efficiency.

## 1 INTRODUCTION

Machine Learning (ML) has achieved remarkable success across a wide range of applications, including financial risk management [35], medical treatment [45], social analysis [15, 51], *etc.* ML training often requires collecting data from multiple entities. Contributing such data without privacy-preserving technologies may compromise the privacy of each entity. This conflicts with data security and privacy regulations [1, 2]. To alleviate these concerns, numerous studies [23, 31, 33, 37, 39, 43, 46] have introduced cryptographic technologies to enable Secure Two-Party Computation (2PC)-based ML. However, they mainly aim to preserve privacy during ML training and/or inference, but overlook the privacy requirements of data feature engineering, which is important for training high-quality models. Some works [16, 28, 32, 50] have proposed privacy-preserving feature selection solutions from well-formed datasets (c.f., Table 1). However, it is non-trivial to generalize them to address privacy concerns during the transformation of raw data features (*e.g.*, gender, salary) to ML-effective numerical representations.

Weight-of-Evidence (WoE) [17, 18, 52] is a Bayes' theorem based statistical method for binary classification, where each category of the predictor feature is transformed into a logarithm of the ratio of positive cases (c.f., § 2.1). This transformation can encode categorical and numerical features on a similar scale and create monotonicity with respect to the target. For instance, WoE transformation maps discrete categories into a continuous real-valued scale, instead of one-hot encoding, which can not only reduce the ML model size but also reflect their relative predictive strength concerning the target label. WoE is an important feature engineering step, and cannot be substituted by end-to-end model training. It is particularly effective with logistic regression models, making it a widely used technique in financial and credit risk analysis [35, 42].

However, WoE is originally developed for centralized datasets in plaintext, whereas real-world data are typically distributed across multiple parties and cannot be publicly shared. As illustrated in Figure 1, we consider both horizontal and vertical data partitions. Unfortunately, applying vanilla WoE to partitioned data introduces the following challenges: *i) In the horizontal setting, each party's WoE is likely to be biased because it relies solely on local sub-dataset rather than global statistics, leading to sub-optimal model performance. ii) In the vertical setting, the challenge is even more severe: parties without access to labels cannot compute WoE at all (e.g., $\mathcal{P}_A$ in Figure 1).* This highlights a critical gap in privacy-preserving ML:

***Although secure training frameworks have achieved remarkable progress on standard, well-structured datasets, they remain insufficient for real-world applications due to the lack of secure data processing methods (e.g., WoE), which are crucial for training high-quality models in practical tasks.***

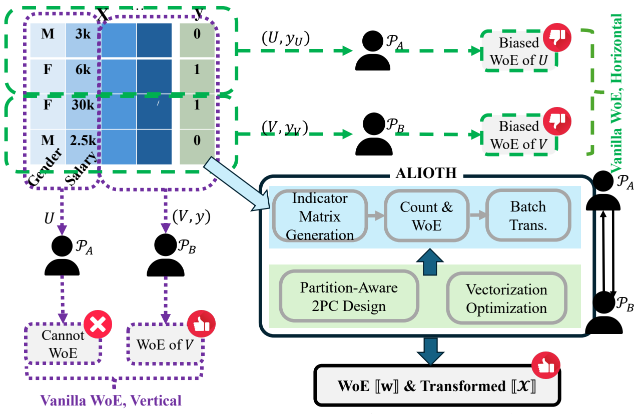


**Figure 1: Illustration of horizontal (dashed green) and vertical (dotted purple) data partitions and their vanilla WoE. Our Alioth outputs the WoE $[\![\mathrm{w}]\!]$ and transformed dataset $[\![\mathcal{X}]\!]$ in secret-shared form. Example features include categorical `Gender` (M/F) and numerical `salary` in X, with labels y.**

To bridge this gap, we propose Alioth, an efficient and secure two-party WoE-based feature transformation framework.

## 1.1 Technical Overview

As shown in Figure 1, we formulate the procedure as three phases: Indicator Matrix Generation, Count & WoE, and Batch Transformation. Besides, we design efficient 2PC protocols for them with the following key technical insights.

*1.1.1 Partition-Aware 2PC Design.* Recall that data are vertically and horizontally partitioned, and data owners also serve as the 2PC computing parties. Given this partitioning scheme, for the vertical setting and for categorical features in the horizontal setting (§ 4.1.1), the two parties independently generate their local indicator matrices and then securely merge them, either column-wise or row-wise, with free communication. For numerical features in the horizontal setting, although interaction is required for secure discretization and interval testing, we leverage the horizontal partition to construct a fully mergeable sketch-based [36] 2PC quantile protocol, which determines interval boundaries in only $O(\log_2 \ell)$ rounds, independent of the number of samples (§ 4.1.2). Similar techniques are also applied to 2PC statistics counting phase (§ 4.2).

*1.1.2 Vectorization Optimization.* Vectorization is critical for the efficiency of privacy-preserving ML. We leverage the same principle as follows: i) The indicator matrix generation of different features is independent. By generating indicator matrices and computing WoE in a vectorized manner, we achieve $O(\log_2 \ell)$ round complexity, independent of the number of samples $H$ and features $W$ (§ 4.1 and 4.2). ii) For data transformation, which involves $O(W)$ independent secure matrix–vector multiplications of varying contraction dimensions, we pad them to a uniform size and perform batched transformations using Homomorphic Encryption (HE)-based approach. This yields upto 30% speedup compared to the naïve approach of computing each multiplication individually in parallel (§ 4.3), along with reduced HE ciphertexts.

*1.1.3 Scopes & Extensions.* In this work, we focus on efficient 2PC WoE computation under vertical and horizontal data partitions,

**Table 1: Representative secure data processing frameworks. Trans. is for transformation and Sel. is for selection.**

| Framework | Settings | Statistics | Feature Ops. | | |
|---|---|---|---|---|---|
| | | | Trans. | Sel. | ML |
| Li *et al.* [28] | 3PC | Gini Impurity | ✘ | ✔ | ✔ |
| BMI [16] | 3PC | Mutual Information | ✘ | ✔ | ✔ |
| SeiFS [32] | 2PC in FL | Gini Impurity | ✘ | ✔ | ✔ |
| EPFS [50] | 2PC | Mutual Information | ✘ | ✔ | ✔ |
| Alioth (Ours) | 2PC | WoE | ✔ | ✔ | ✔ |

where the data owners also serve as one of the 2PC computing parties. For missing values, we replace them with a dedicated placeholder for categorical features and a special numeric constant for numerical features. We add $\texttt{fill\_value}=1e^{-6}$ to avoid the numerator or denominator of Equation (2) being zero due to low-frequency categories. Alioth is designed for two-party in arithmetic blackbox [19], allowing straightforward extension to the multi-party setting using MPC primitives. Since WoE demonstrates superior predictive performance with LR compared to other models, such as neural networks [44], this work focuses on LR training with WoE.

## 1.2 Our Contributions.

Alioth proposes the first secure and scalable WoE-based data processing framework, featuring a set of 2PC protocols and optimizations. Concretely, our main contributions are as follows:

- **Design of Framework Alioth.** We propose Alioth, the first efficient and secure WoE framework that supports both vertical and horizontal data partitions. Alioth directly outputs secret shares of the WoE-transformed dataset, enabling efficient and seamless integration with existing 2PC secure ML training and feature selection frameworks. To our best knowledge, this is the first work to provide secure WoE-based data processing.
- **Partition-Aware Protocols & Optimizations.** We provide a systematic analysis and formulation of the WoE workflow, decomposing it into three key procedures: indicator matrix generation, counts & WoE, and transformation. For each procedure, we design efficient partition-aware two-party protocols, including novel approximated quantiles in 2PC, and incorporate optimizations that leverage the specific characteristics of vertical and horizontal partitions. Some protocols and optimizations may be of independent interest.
- **Implementation & Evaluation.** We implement a prototype of Alioth on SecretFlow-SPU with a 2PC backend [23, 33] for evaluations: i) We report the end-to-end efficiency and profile the overhead of each building block under both data partitions. Specifically, we complete the WoE-based transformation of a dataset containing over 3 million samples and 100 features in 0.5 hour in WAN. ii) Furthermore, we evaluate Alioth on two widely used financial datasets and integrate it with two downstream applications: information value-based feature selection and logistic regression. In particular, Alioth improves the AUC of logistic regression from 0.507 to 0.735 on dataset HCDR [40] with reduced 2PC training overhead in 10 iterations.

**Organization.** § 2 introduces the background and preliminaries. § 3 presents our system overview. § 4 illustrates the detailed protocol design. § 6 reports the experimental evaluations and applications. § 7 summarizes the related works and § 8 concludes this work.

## 2 BACKGROUND & PRELIMINARIES

**Notations.** Bold uppercase $\mathbf{X}$ denotes matrices and bold lowercase $\mathbf{x}$ is for vectors. $\mathbf{X}_i$ (resp. $\mathbf{X}^j$) refers to the $i$-th row (resp. $j$-th column), and $\mathbf{X}_i^j$ indicates the $(i, j)$-th element. $\mathbf{x}_i$ denotes the $i$-th entry. $f_k^{(j)}$ refers to the $k$-th category of the $j$-th feature (*a.k.a.*, column) of $\mathbf{X}$. $\mathbf{B}^{(j)}$ denotes the indicator matrix for the $j$-th feature, and $\mathcal{B}$ represents the global ones of all features. $\mathbf{X} = \mathbf{U}||\mathbf{V}$ indicates $\mathbf{X}$ is vertically partitioned into $\mathbf{U}$ and $\mathbf{V}$, and $\mathbf{X} = \mathbf{U} \cup \mathbf{V}$ is for horizontal. $1\{C\}$ returns 1 if condition $C$ is true and 0 otherwise.

### 2.1 Weight of Evidence

Weight of Evidence (WoE) [17, 18, 52] is a statistical data-driven method and is defined for binary classification problems. Given dataset $(\mathbf{X}, \mathbf{y})$, $\mathbf{y}_i \in \{0, 1\}$ denotes the label of $\mathbf{X}_i$. Let the $j$-th feature be with $K_j$ possible categories $\{f_1^{(j)}, f_2^{(j)}, \ldots, f_{K_j}^{(j)}\}$, WoE for the $j$-th feature regarding the $k$-th category $f_k^{(j)}$ is defined as:

$$\mathbf{w}_k^{(j)} = \log_2\left(\frac{\Pr(\mathbf{X}^j = f_k^{(j)} \mid \mathbf{y} = 1)}{\Pr(\mathbf{X}^j = f_k^{(j)} \mid \mathbf{y} = 0)}\right), \forall k \in [1, K_j] \quad (1)$$

where $\Pr(\mathbf{X}^j = f_k^{(j)} \mid \mathbf{y} = 1) = \frac{\sum_i \mathbf{1}\{\mathbf{X}_i^j = f_k^{(j)} \wedge \mathbf{y}_i = 1\}}{\sum_i \mathbf{1}\{\mathbf{y}_i = 1\}}$, $\Pr(\mathbf{X}^j = f_k^{(j)} \mid \mathbf{y} = 0) = \frac{\sum_i \mathbf{1}\{\mathbf{X}_i^j = f_k^{(j)} \wedge \mathbf{y}_i = 0\}}{\sum_i \mathbf{1}\{\mathbf{y}_i = 0\}}$. So Eq. (1) can be expressed as:

$$\mathbf{w}_k^{(j)} = \log_2\left(\frac{\sum_i \mathbf{1}\{\mathbf{X}_i^j = f_k^{(j)} \wedge \mathbf{y}_i = 1\}}{\sum_i \mathbf{1}\{\mathbf{X}_i^j = f_k^{(j)} \wedge \mathbf{y}_i = 0)} \cdot \frac{\sum_i \mathbf{1}\{\mathbf{y}_i = 0\}}{\sum_i \mathbf{1}\{\mathbf{y}_i = 1\}}\right) \quad (2)$$

#### 2.1.1 *Categorical Feature.*

For a categorical feature $\mathbf{X}^j$ with categorical features (*e.g.*, `male` and `female` for `gender`), the WoE value for each category is computed using Eq. (1) or (2). A positive WoE value ($\mathbf{w}_k^j > 0$) indicates that $\Pr(\mathbf{X}^j = f_k^{(j)} \mid \mathbf{y} = 1) > \Pr(\mathbf{X}^j = f_k^{(j)} \mid \mathbf{y} = 0)$, meaning the category is more prevalent among the positive class ($\mathbf{y} = 1$). Conversely, $\mathbf{w}_k^j < 0$ implies the category is more associated with the negative class ($\mathbf{y} = 0$).

#### 2.1.2 *Numerical Feature.*

WoE cannot be directly applied to numeric features, since their values are from a continuous domain. We discretize numerical feature $\mathbf{X}^j$ into $K_j$ non-overlapping intervals:

$$\mathbb{I}_1^{(j)} = [I_0^{(j)}, I_1^{(j)}], \mathbb{I}_2^{(j)} = (I_1^{(j)}, I_2^{(j)}], \ldots, \mathbb{I}_{K_j}^{(j)} = (I_{K_j-1}^{(j)}, I_{K_j}^{(j)}],$$

where $I_0^{(j)} < I_1^{(j)} < \cdots < I_{K_j}^{(j)}$ are boundaries determined by a chosen method, such as an equal-width discretiser ($I_k^{(j)} = I_0^{(j)} + k \cdot (I_{K_j}^{(j)} - I_0^{(j)})/K_j$) or equal-frequency discretiser (quantile-based). Each interval $\mathbb{I}_k^{(j)}$ is then treated as a category $f_k^{(j)}$, so that: $\mathbf{X}_i^j \in \mathbb{I}_k^{(j)} \Leftrightarrow \mathbf{X}_i^j \mapsto f_k^{(j)}$. Afterwards, the WoE for $\mathbb{I}_k^{(j)}$ is computed using Eqs. (1) or (2). In ALIOTH, *we employ equal-frequency discretization, as it performs better across most scenarios* [17, 18].

#### 2.1.3 *Limitations of WoE.*

WoE is not defined when $\Pr(\mathbf{X}^j = f_k^{(j)} \mid \mathbf{y} = 1) = 0$ or $\Pr(\mathbf{X}^j = f_k^{(j)} \mid \mathbf{y} = 0) = 0$, which typically occurs for low-frequency categories. A common solution is to group such categories as one, *i.e.*, using `RareLabelEncoder` [17]. In this work, we add a small `fill_value=` $1e^{-6}$ to avoid 0 and omit it for brevity.

**Table 2: Well-established 2PC protocols.**

| Protocols | Description |
|---|---|
| $[\![\mathbf{X} \cdot \mathbf{y}]\!] = \Pi_{\mathsf{MV}}([\![\mathbf{X}]\!], [\![\mathbf{y}]\!])$ | Matrix-vector multiplication. |
| $[\![x \cdot y]\!] = \Pi_{\mathsf{Mul}}([\![x]\!], [\![y]\!])$ | Scalar multiplication. |
| $\langle x \le y \rangle = \Pi_{\mathsf{LE}}([\![x]\!], [\![y]\!])$ | Less-Equal. |
| $[\![x]\!] = \Pi_{\mathsf{B2A}}(\langle x \rangle)$ | Bool.-to-Arith. Conv. |
| $[\![\log_2 x]\!] = \Pi_{\mathsf{Log}}([\![x]\!])$ | Logarithm. |
| $[\![x/y]\!] = \Pi_{\mathsf{Div}}([\![x]\!], [\![y]\!])$ | Division. |

### 2.2 Cryptographic Primitives

#### 2.2.1 *Additive Secret Sharing.*

$[\![x]\!]$ denotes an additive sharing of the value $x \in \mathbb{Z}_L$. We write $[\![x]\!] = ([\![x]\!]^A, [\![x]\!]^B)$ where $\mathcal{P}_A$ holds $[\![x]\!]^A$ and $\mathcal{P}_B$ holds $[\![x]\!]^B$ with $x = [\![x]\!]^A + [\![x]\!]^B \bmod L$. We omit the party-specific superscript when it is clear from context. $L > 2$ (i.e., $L = 2^{64}$ or large prime) refers to Arithmetic share. $L = 2$ (a.k.a., bit-width $\ell = 1$) is for Boolean share $\langle x \rangle$, where addition/subtraction (resp. multiplication) is equivalent to logical XOR $\oplus$ (resp. AND $\wedge$).

#### 2.2.2 *Addition & Multiplication.*

Addition/subtraction and secret-public multiplication can be computed locally. Multiplying two secrets requires communication and can be realized through `HE`. Moreover, efficient `HE`-based solutions are proposed for secure matrix and vector multiplications [23, 33].

Similarly, for Boolean circuits, bitwise XOR is a local operation and AND is achieved by Oblivious Transfer (`OT`) [24, 54]. Two-party Less-Equal ($x \le y$) is also expressed using Boolean circuits and computed using the `OT`-based approach proposed by [23, 46].

#### 2.2.3 *Truncation & Others.*

For real value $\hat{x} \in \mathbb{R}$, we encode it as fixed-point value $x = \lfloor \hat{x} \cdot 2^f \rfloor$ under pre-defined precision $f > 0$ and then secretly share it. To prevent overflow in fixed-point multiplication, we use $\Pi_{\mathsf{Trunc}}$ from [23, 33, 34] to truncate $[\![x; 2f]\!]$ to $[\![x; f]\!]$ with 1 least-bit error. By default, we implicitly incorporate $\Pi_{\mathsf{Trunc}}$ following fixed-point multiplication and omit it for brevity.

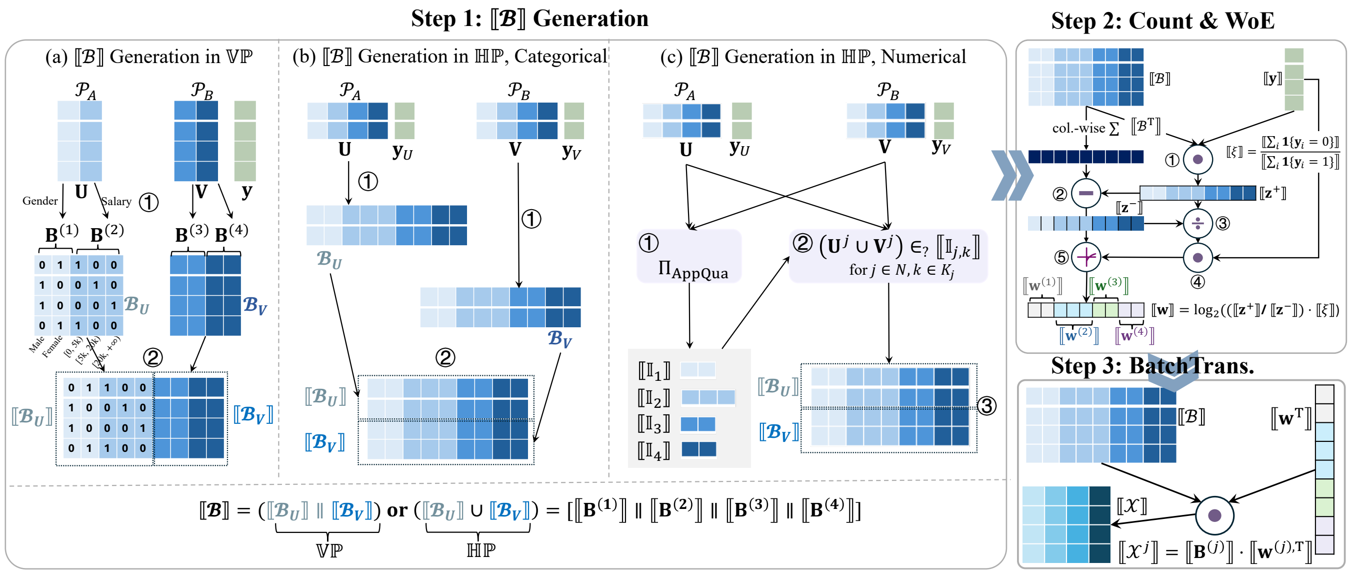
Additionally, ALIOTH is designed with several other 2PC protocols in Table 2, which have been securely realized from existing well-established 2-party secret sharing-family frameworks [23, 33, 46].

## 3 OVERVIEW OF ALIOTH

### 3.1 System Model

Let $\mathcal{P}_A$ and $\mathcal{P}_B$ be two parties holding their private inputs. We consider vertical and horizontal data partitions as follows:

- $\mathbb{V}$**ertical** $\mathbb{P}$**artition (**$\mathbb{VP}$**).** In this setting, two parties hold *disjoint subset* of features of *all* samples. Formally, dataset $\mathbf{X} \in \mathbb{R}^{H \times W}$ is split column-wise as $\mathbf{X} = \mathbf{U}||\mathbf{V}$. Party $\mathcal{P}_A$ holds its input $\mathbf{U} \in \mathbb{R}^{H \times W_1}$ and $\mathcal{P}_B$ holds $\mathbf{V} \in \mathbb{R}^{H \times W_2}$ (Figure 2, Step 1, a), with $W = W_1 + W_2$. Without loss of generality, we let $\mathcal{P}_B$ maintain the label $\mathbf{y}$. Note that $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{y}$ are already aligned between $\mathcal{P}_A$ and $\mathcal{P}_B$, *a.k.a.*, $\mathbf{X}_i = \mathbf{U}_i||\mathbf{V}_i$ and $\mathbf{y}_i$ is the corresponding label. This can be achieved through Private Set Intersection (PSI)-based protocols [48] and is orthogonal to our work.
- $\mathbb{H}$**orizontal** $\mathbb{P}$**artition (**$\mathbb{HP}$**).** Conversely, in $\mathbb{HP}$, each party holds *all* features of *a subset of samples* along with their labels (Figure 2,



$$⟦\mathcal{B}⟧ = (⟦\mathcal{B}_U⟧ \parallel ⟦\mathcal{B}_V⟧) \text{ or } (⟦\mathcal{B}_U⟧ \cup ⟦\mathcal{B}_V⟧) = [⟦\mathbf{B}^{(1)}⟧ \parallel ⟦\mathbf{B}^{(2)}⟧ \parallel ⟦\mathbf{B}^{(3)}⟧ \parallel ⟦\mathbf{B}^{(4)}⟧]$$

$$⟦\xi⟧ = \frac{⟦\sum_i \mathbf{1}\{\mathbf{y}_i = 0\}⟧}{⟦\sum_i \mathbf{1}\{\mathbf{y}_i = 1\}⟧}$$

$$⟦\mathbf{w}⟧ = \log_2((⟦\mathbf{z}^+⟧ / ⟦\mathbf{z}^-⟧) \cdot ⟦\xi⟧)$$

$$⟦\mathcal{X}^j⟧ = ⟦\mathbf{B}^{(j)}⟧ \cdot ⟦\mathbf{w}^{(j),\mathrm{T}}⟧$$

**Figure 2: The system design of Alioth. In this example, the dataset X has 4 samples and 4 features. In the 𝕍ℙ setting, each party holds two features across all samples, and $\mathcal{P}_B$ holds labels without loss of generality. In the ℍℙ setting, each party holds all features and corresponding labels of two samples. The categories (or intervals) of each feature are (2, 3, 2, 2). For each feature, the indicator vector of one sample is one-hot, and we present the examples for feature `gender` and `salary`.**

Step 1, b & c). Specifically, $\mathcal{P}_A$ holds a subset of rows of $\mathbf{X}$ and corresponding labels as ($\mathbf{U} \in \mathbb{R}^{H_1 \times W}, \mathbf{y}_U \in \{0,1\}^{H_1}$), while $\mathcal{P}_B$ holds the remaining samples ($\mathbf{V} \in \mathbb{R}^{H_2 \times W}, \mathbf{y}_V \in \{0,1\}^{H_2}$) with $H = H_1 + H_2$. Thus, $\mathbf{X} = \mathbf{U} \cup \mathbf{V}$ and $\mathbf{y} = \mathbf{y}_U \cup \mathbf{y}_V$.

After secure WoE-based transformation, $\mathcal{P}_A$ and $\mathcal{P}_B$ obtain the shares of transformed selected columns (*a.k.a.*, features) $⟦\mathbf{X}^J⟧$ ($J$ is for selected columns) or entire $⟦\mathbf{X}⟧$ ($J$ includes all features). We use the same notation (*e.g.*, $\mathbf{U}$) to denote the data held by each party in both partitions. The exact meaning is clear from context.

## 3.2 High-Level Workflow

As illustrated in Figure 2, we formulate Alioth as three main steps: Indicator Matrix Generation, Count & WoE, and Transformation.

- **Step 1: Indicator Matrix Generation.** Given a dataset $\mathbf{X} \in \mathbb{R}^{H \times W}$, let $\mathbf{x} \in \mathbb{R}^H$ denote an arbitrary column (*a.k.a.*, one feature across all samples). For each category of this feature (*e.g.*, category `male` for feature `gender`), we construct an indicator vector $\mathbf{b}$ such that $\mathbf{b}_i = 1$ if $\mathbf{x}_i$ belongs to the current category and 0 otherwise. In 𝕍ℙ setting, this can be computed locally by each party in plaintext (a-①). In contrast, in ℍℙ setting: i) For a categorical feature, $\mathcal{P}_A$ (resp. $\mathcal{P}_B$) locally generates its partial indicator vector $\mathbf{b}_U$ (resp. $\mathbf{b}_V$) based on its local data $\mathbf{U}$ (resp. $\mathbf{V}$) (b-①). ii) For a numerical feature, we first perform equal-frequency discretization with non-overlapping intervals. To securely determine the interval boundaries, we design a fast 2PC quantile protocol $\Pi_{\mathsf{AppQua}}$ (c-①). Once the boundaries are established, $⟦\mathbf{b}⟧$ can be obtained through 2PC interval test (c-②). Assuming the $j$-th feature has $K_j$ categories, the above procedure can be executed in parallel across all categories of all features, all indicator vectors are ultimately concatenated in column-wise (a-②) or row-wise (b-② & c-③) to compose the global indicator matrix $⟦\mathcal{B}⟧ \in \{0,1\}^{H \times \sum_{j=1}^{W} K_j}$ in a privacy-preserving manner.
- **Step 2: Count & WoE.** For each feature (*e.g.*, `gender`), we need to count the positive and negative samples for every category (*e.g.*, `male`) to compute Equation (2). This counting process is equivalent to the vector inner product. Concretely, for one feature category (or interval) with indicator vector $\mathbf{b} \in \{0,1\}^H$, as each sample is labeled with $\mathbf{y}_i = \{0,1\}$, the number of positive samples can be computed as $\mathbf{b} \cdot \mathbf{y}$. Then, the negative samples of the same category is $\sum_{i=1}^{H} \mathbf{b}_i - \mathbf{b} \cdot \mathbf{y}$.
  In general, given $⟦\mathcal{B}⟧$ and the label vector $⟦\mathbf{y}⟧$ of the dataset, we compute the number of positive samples for each category of every feature via matrix-vector multiplication (②) in vectorization. The negative counts are then obtained by column-wise summation of $⟦\mathcal{B}⟧$ followed by subtraction (②). With these statistics, WoE $⟦\mathbf{w}⟧$ is calculated as Equation (2) by invoking 2PC division (③), multiplication (④), and logarithm (⑤) in element-wise.
- **Step 3: Batch Transformation.** Finally, the original $\mathbf{X}$ is transformed into $⟦\mathcal{X}⟧$ by replacing each category (or interval for numerical feature) with its corresponding WoE value. For $\forall j$-th feature, $⟦\mathcal{X}^j⟧$ is computed through a matrix–vector multiplication between its indicator matrix and associated WoE vector.

The transformed dataset $⟦\mathcal{X}⟧$ is secret-shared between $\mathcal{P}_A$ and $\mathcal{P}_B$, regardless of whether the original input $\mathbf{X}$ is partitioned in either 𝕍ℙ or ℍℙ. Afterwards, two parties can jointly conduct various downstream privacy-preserving analytic and/or ML tasks on $⟦\mathcal{X}⟧$, such as information value (IV)-based feature selection and LR training.

When the parties aim at the WoE of a feature subset, $\mathbf{U} \| \mathbf{V}$ or $\mathbf{U} \cup \mathbf{V}$ includes only the selected columns, rather than the full $\mathbf{X}$.

## 3.3 Security Model

In Alioth, we preserve the privacy of feature and label values, as well as all intermediate values and transformed results. They are either maintained in secret-shared or privately retained by their owner. On the other hand, we do not protect the following public information: i) *Auxiliary knowledge naturally available in practical deployments*, including data partitioning scheme ($\mathbb{VP}$ or $\mathbb{HP}$), type of feature (categorical or numerical), and number of categories or intervals ($K_j$) of $\forall j$-th feature. For categorical features, all categories (*e.g.*, `male`, `female` for `gender`) are known to the feature owner. ii) *Information inherently inferred from the outputs*, namely the number of samples ($H$) and features ($W$).

Following prior works [23, 34], Alioth is secure against semi-honest static probabilistic polynomial time (PPT) adversaries. The adversary $\mathcal{A}$ corrupts one of $\mathcal{P}_A$ or $\mathcal{P}_B$ at the beginning of the protocol and follows the protocol specification honestly. We use the standard simulation-based security definition as follows.

Definition 1 (Semi-Honest Security). *Let $\Pi$ be a two-party protocol and $\mathcal{F} : (\{0,1\}^n)^2 \to (\{0,1\}^m)^2$ be the ideal randomized functionality. We say $\Pi$ securely computes $\mathcal{F}$ in presence of a semi-honest PPT adversary $\mathcal{A}$ if for every corrupted party $P_s$ ($s \in \{A, B\}$) and every input $\mathbf{x} \in (\{0,1\}^n)^2$, there exists an efficient simulator $\mathcal{S}$:*

$$\{\mathsf{view}_{s,\Pi}(\mathbf{x}), \mathsf{out}_\Pi(\mathbf{x})\} \overset{c}{\approx} \{\mathcal{S}(s, \mathbf{x}_s, \mathcal{F}_s(\mathbf{x})), \mathcal{F}(\mathbf{x})\},$$

*where $\mathsf{view}_{s,\Pi}(\mathbf{x})$ is the view of $\mathcal{P}_s$ in execution of $\Pi$ on $\mathbf{x}$, $\mathsf{out}_\Pi(\mathbf{x})$ is the output, $\mathcal{F}_s(\mathbf{x})$ denotes $\mathcal{P}_s$'s output of $\mathcal{F}(\mathbf{x})$, and $\overset{c}{\approx}$ represents computationally indistinguishability.*

Remark 1 (Malicious Security). *Most real privacy-preserving applications adopt the semi-honest assumption due to practical efficiency and laws/regulatory acceptance. Malicious extensions can be supported via malicious-secure primitives at $> 10\times$ cost.*

## 4 DESIGN OF ALIOTH

We formulate the ideal functionality $\mathcal{F}_{\text{Alioth}}$ in Figure 3. $\mathcal{F}_{\text{Alioth}}$ receives the vertically or horizontally partitioned dataset $\mathbf{U}$ and $\mathbf{V}$ from the two parties, together with the associated labels, and merges them column-wise or row-wise, depending on the partitioning scheme (Steps 1-5). For each feature $j$, $\mathcal{F}_{\text{Alioth}}$ constructs its indicator matrix $\mathbf{B}^{(j)}$, which encodes the category (or interval) membership of all samples for $j$-th feature (Steps 6). All such indicator matrices are concatenated column-wise to form the global indicator matrix (Step 7). Subsequently, $\mathcal{F}_{\text{Alioth}}$ counts the required statistics and computes WoE values for all features (Step 8). Finally, it transforms each feature according to its WoE and indicator matrix (Step 9), and outputs the secret-shared results (Steps 10-11).

## 4.1 Two-Party Indicator Matrix Generation

In § 4.1.1, we show the non-interactive indicator matrix generation for $\mathbb{VP}$ and categorical features under $\mathbb{HP}$. In § 4.1.2, we present the generation procedures for numerical features in $\mathbb{HP}$.

*4.1.1 Non-interactive $[\![\mathcal{B}]\!]$ Generation.* In $\mathbb{VP}$, $\mathcal{P}_A$ (resp. $\mathcal{P}_B$) generates the indicator matrix $\mathcal{B}_U \in \{0,1\}^{H \times \sum_{j=1}^{W_1} K_j}$ (resp. $\mathcal{B}_V \in \{0,1\}^{H \times \sum_{j=1}^{W_2} K_j}$) for $\mathbf{U}$ (resp. $\mathbf{V}$) locally, where $W = W_1 + W_2$. Then, the global indicator matrix can be concatenated column-wise as $[\![\mathcal{B}]\!] = [\![\mathcal{B}_U]\!] \| [\![\mathcal{B}_V]\!]$, where $\mathcal{B}_U$ and $\mathcal{B}_V$ are secret-shared via *lazy-sharing*[1] without communication.

**Functionality $\mathcal{F}_{\text{Alioth}}$**

**Parameters:** Two parties $\mathcal{P}_A$ and $\mathcal{P}_B$. In $\mathbb{VP}$, $\mathcal{P}_A$ holds $\mathbf{U}$ and $\mathcal{P}_B$ holds $(\mathbf{V}, \mathbf{y})$; In $\mathbb{HP}$, $\mathcal{P}_A$ holds $(\mathbf{U}, \mathbf{y}_U)$ and $\mathcal{P}_B$ holds $(\mathbf{V}, \mathbf{y}_V)$. The $j$-th feature is with $K_j$ categories (or intervals). The dataset is with $H$ samples and $W$ features.

**Functionality:**

```
1:  if $\mathbb{VP}$ is True then
2:      Receive $\mathbf{U}$ from $\mathcal{P}_A$ and $(\mathbf{V}, \mathbf{y})$ from $\mathcal{P}_B$, and $\mathbf{X} = \mathbf{U} \| \mathbf{V}$.
3:  else if $\mathbb{HP}$ is True then
4:      Receive $(\mathbf{U}, \mathbf{y}_U)$ from $\mathcal{P}_A$ and $(\mathbf{V}, \mathbf{y}_V)$ from $\mathcal{P}_B$, and
        union $\mathbf{X} = \mathbf{U} \cup \mathbf{V}$ and $\mathbf{y} = \mathbf{y}_A \cup \mathbf{y}_B$.
5:  end if
6:  Generate indicator matrix $\mathbf{B}^{(j)}$ for feature $j \in [1, W]$.
7:  Concatenate indicator matrices of all features to generate
    the global indicator matrix $\mathcal{B}$.
8:  Compute $\mathbf{z}^+ = \mathcal{B} \cdot \mathbf{y}$, $\mathbf{z}^- = \sum_{i=1}^{H} \mathcal{B}_i - \mathbf{z}^+$, $\xi = \frac{\sum_i^H 1\{\mathbf{y}_i = 0\}}{\sum_i^H 1\{\mathbf{y}_i = 1\}}$
    and WoE $\mathbf{w} = \log_2(\frac{\mathbf{z}^+}{\mathbf{z}^-} \cdot \xi)$.
9:  Parties extract $\mathbf{w}^{(j)} = \{\mathbf{w}_{K_{j-1}+1}, \ldots, \mathbf{w}_{K_{j-1}+K_j}\}$ from $\mathbf{w}$,
    and compute $\mathcal{X}^j = \mathbf{B}^{(j)} \cdot \mathbf{w}^{(j),\intercal}$ for feature $j \in [1, W]$.
10: Sample uniform random $\mathbf{R}$ with the same size as $\mathcal{X}$, let
    $[\![\mathcal{X}]\!]^A = \mathbf{R}$ and compute $[\![\mathcal{X}]\!]^B = \mathcal{X} - \mathbf{R}$.
11: Output shares $[\![\mathcal{X}]\!]^A$ to $\mathcal{P}_A$ and $[\![\mathcal{X}]\!]^B$ to $\mathcal{P}_B$.
```

**Figure 3: Ideal Functionality of Alioth.**

**Categorical Feature in $\mathbb{HP}$.** For the categorical features, each party knows all categories. So, $\mathcal{P}_A$ and $\mathcal{P}_B$ can construct their own indicator matrix for their own *disjoint* data samples locally. Assuming there are $W$ categorical features and the $j$-th feature is with $K_j$ categories, $\mathcal{P}_A$ computes $\mathcal{B}_U \in \{0,1\}^{H_1 \times \sum_{j=1}^{W} K_j}$ and $\mathcal{P}_B$ computes $\mathcal{B}_V \in \{0,1\}^{H_2 \times \sum_{j=1}^{W} K_j}$, where $H = H_1 + H_2$. Furthermore, the parties obtain $[\![\mathcal{B}_U]\!]$ and $[\![\mathcal{B}_V]\!]$ via *lazy sharing* as well, and merge them row-wise to form the global indicator matrix $[\![\mathcal{B}]\!] = [\![\mathcal{B}_U]\!] \cup [\![\mathcal{B}_V]\!] \in \{0,1\}^{(H_1+H_2) \times \sum_{j=1}^{W} K_j}$.

*4.1.2 Interactive $[\![\mathcal{B}]\!]$ Generation for Numerical Feature in $\mathbb{HP}$.* We show how to generate $[\![\mathcal{B}]\!]$ with known-but-secret intervals $\{[\![\mathbb{I}_k^{(j)}]\!]\}_{k=1}^{K_j}$.

**2PC Interval Test.** Let $\mathbf{u} \in \mathbf{U}$ and $\mathbf{v} \in \mathbf{V}$ denote the $\forall j$-th feature vectors from the $\mathbb{HP}$-partitioned dataset $\mathbf{U} \cup \mathbf{V}$, so that the whole feature vector is $\mathbf{u} \cup \mathbf{v}$. Assume there are $K_j$ known-but-secret bins $\{[\![\mathbb{I}_k^{(j)}]\!]\}_{k=1}^{K_j}$, where an interval $[\![\mathbb{I}_k^{(j)}]\!]$ is said to be secret-shared if its boundaries are secret-shared, i.e., $[\![\mathbb{I}_k^{(j)}]\!] = [[\![I_{k-1}^{(j)}]\!], [\![I_k^{(j)}]\!])$.

To determine whether the elements of $\mathbf{u}$ belong to $[\![\mathbb{I}_k^{(j)}]\!]$ or not, two parties compare $\mathbf{u}$ with the boundaries to get Boolean shares:

$$\langle \mathbf{c}_{u,k-1} \rangle = \Pi_{\mathsf{LE}}(\mathbf{u}, [\![I_{k-1}^{(j)}]\!]) \;\;\&\;\; \langle \mathbf{c}_{u,k} \rangle = \Pi_{\mathsf{LE}}(\mathbf{u}, [\![I_k^{(j)}]\!]), \quad (3)$$

where $\mathcal{P}_A$ provides $\mathbf{u}$. The indicator vector for $\mathbf{u}$ regarding to $[\![\mathbb{I}_k]\!]$ is computed as $\langle \mathbf{b}_{u,k} \rangle = \langle \mathbf{c}_{u,k-1} \rangle \oplus \langle \mathbf{c}_{u,k} \rangle$.

[1] $\mathcal{P}_A$ sets $[\![\mathcal{B}_U]\!]_0 = \mathcal{B}_U$ and $\mathcal{P}_B$ sets $[\![\mathcal{B}_U]\!]_1 = \mathbf{0}$. Symmetrically for $[\![\mathcal{B}_V]\!]$.

```
DDSketch for $\mathbb{R}_{>0}$
Initialization: Sketch $B^+$ with $\eta$ buckets initialized as 0, relative-error ratio $0 < \alpha < 1$, and $\gamma = (1+\alpha)/(1-\alpha)$. The number of inputs $n$.
Insert($x$):
  1: For $x \in \mathbb{R}_{>0}$, $\tau = \lceil \log_\gamma(x) \rceil$, and $B^+_\tau = B^+_\tau + 1$.
Merge($B^+, B'^{,+}$): ▷ $B'^{,+}$ is with the same $\alpha$ and $\eta$ buckets.
  1: $B^+_\tau = B^+_\tau + B'^{,+}_\tau$ for all $\tau$.
Quantile($\beta$): ▷ Percentile $0 \le \beta \le 1$.
  1: Let $i_0 = \min(\{j : B^+_j > 0\})$, $c = B^+_{i_0}$, $i = i_0$.
  2: while $c < \lfloor \beta n \rfloor$ do
  3:     $i = \min(\{j : B^+_j > 0 \wedge j > i\})$,
  4:     $c = c + B^+_i$.
  5: end while
  6: Outputs $2\gamma^i/(\gamma+1)$.
```

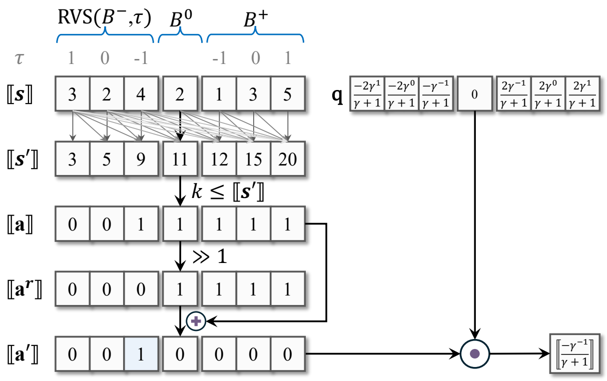
**Figure 4: Operations of DDSketch for $\mathbb{R}_{>0}$ from [36].**

Moreover, two parties can compute $\{\langle \mathbf{b}_{u,k} \rangle\}_{k=1}^{K_j}$ for intervals to generate $K_j$ indicator vectors for the $j$-th feature, and concatenate them column-wise to get the indicator matrix $\left\langle \mathbf{B}_u^{(j)} \right\rangle \in \{0,1\}^{H_1 \times K_j}$. Furthermore, all indicator matrices across all features can be concatenated in a column-wise manner to get global $\langle \mathcal{B}_U \rangle \in \{0,1\}^{H_1 \times \sum_{j=1}^{W} K_j}$ and two parties invoke secure Boolean-to-Arithmetic ($\Pi_{\mathsf{B2A}}$) conversion to get arithmetic sharing $[\![\mathcal{B}_U]\!] = \Pi_{\mathsf{B2A}}(\langle \mathcal{B}_U \rangle)$. Symmetrically, two parties can compute $[\![\mathcal{B}_V]\!] \in \{0,1\}^{H_2 \times \sum_j^W K_j}$ where $\mathcal{P}_B$ provides $\mathbf{V}$. The global indicator matrix can be concatenated row-wise as $[\![\mathcal{B}]\!] = [\![\mathcal{B}_U]\!] \cup [\![\mathcal{B}_V]\!] \in \{0,1\}^{(H_1+H_2) \times \sum_{j=1}^{W} K_j}$.

*4.1.3 Fast Approximated $[\![\mathbb{I}_k^{(j)}]\!]$ with Fully Mergeable Sketch [36].* In equal-frequency discretization as above, the interval boundaries are quantiles, and should be computed privately. Since the quantiles of different features are independent, we focus on one feature below.

Let $\mathbf{u}$ and $\mathbf{v}$ denote one arbitrary column as before. A straightforward approach is to use *secure merge*: $\mathcal{P}_A$ locally sorts $\mathbf{u}$ and $\mathcal{P}_B$ locally sorts $\mathbf{v}$, after which the two parties securely merge the sorted vectors to obtain $[\![\mathbf{q}]\!] = \mathsf{sorted}([\![\mathbf{u}]\!] \cup [\![\mathbf{v}]\!])$. The $\beta$-th quantile (percentile $0 \le \beta \le 1$) can then be retrieved as $[\![\mathbf{q}_k]\!]$, where $k = \lceil \beta \cdot (H_1 + H_2) \rceil$. However, this method requires $O(n)$ secure comparisons in $O(\log_2 \log_2 n \cdot \log_2 \ell)$ rounds, where $n = \max(H_1, H_2)$ [5, 9]. We propose a fast approximate quantile computation method based on DDSketch [36], which only requires one batch of $O(\log_2 \max(\mathbf{u} \cup \mathbf{v}) - \log_2 \min(\mathbf{u} \cup \mathbf{v}))$ comparisons in $O(\log_2 \ell)$ rounds, independent of $H_1$ and $H_2$.

**Revisit DDSektch [36].** DDSketch is a fully-mergeable quantile sketch with relative-error guarantees. It works by dividing $\mathbb{R}_{>0}$ into $K$ buckets, and each bucket $B_i^+$ (indexed by $i \in \mathbb{Z}$) counts the number of values $x$ that fall between $\gamma^{i-1} < x \le \gamma^i$, where $\gamma = (1+\alpha)/(1-\alpha)$, and $0 < \alpha < 1$ is the relative-accurate ratio. DDSketch is defined by three basic operations: Insert, Merge, and Quantile, as shown in Figure 4. Given $0 \le \beta \le 1$, Quantile($\beta$) returns an $\alpha$-accurate $\beta$-quantile $\widetilde{x}_q = \frac{2\gamma^i}{\gamma+1}$, such that $|x_\beta - \widetilde{x}_\beta| \le \alpha x_q$, where $x_\beta$ is the faithful $\beta$-quantile [36, Proposition 3]. DDSketch can support all of $\mathbb{R}$ by maintaining a second sketch $B^-$ for negative $\mathbb{R}_{<0}$ and



RVS($B^-$,$\tau$)
$B^0$
$B^+$
$\tau$
1 0 -1 -1 0 1
$[\![\mathbf{s}]\!]$
3 2 4 2 1 3 5
$\mathbf{q}$
$\frac{-2\gamma^1}{\gamma+1}$ $\frac{-2\gamma^0}{\gamma+1}$ $\frac{-\gamma^{-1}}{\gamma+1}$ 0 $\frac{2\gamma^{-1}}{\gamma+1}$ $\frac{2\gamma^0}{\gamma+1}$ $\frac{2\gamma^1}{\gamma+1}$
$[\![\mathbf{s'}]\!]$
3 5 9 11 12 15 20
$k \le [\![\mathbf{s'}]\!]$
$[\![\mathbf{a}]\!]$
0 0 1 1 1 1 1
$\gg 1$
$[\![\mathbf{a^r}]\!]$
0 0 0 1 1 1 1
$[\![\mathbf{a'}]\!]$
0 0 1 0 0 0 0
$[\![\frac{-\gamma^{-1}}{\gamma+1}]\!]$

**Figure 5: A toy example for $k = 9$. $\tau$ indicates the public logical indices, q is computed from $\tau$.**

a special bucket $B^0$ for 0. The indices for the negative sketch are computed using absolute values as $\tau = \lceil \log_\gamma |x| \rceil$.

**Implementation Details.** Masson *et al.* also proposed the *bucket collapsing* technique to prevent unbounded growth of buckets. However, as analyzed by Masson *et al.* [36, Theorem 9, Section 3], the DDSketch size bound is $O\left(\frac{\log_2 \max - \log_2 \min}{\log_2 \gamma}\right)$, which grows very slowly with the input range. In practice, the sketch size is typically set to a constant to cover a wide range of values. For example, with $\alpha = 0.01$, a sketch of size $\eta = 1000$ can handle values in $[2^{-10}, 2^{10}]$.

Considering *bucket collapsing* is MPC-unfriendly (it involves many sequential comparisons), and we can reasonably estimate the range of numerical features of real-world datasets, we set the sketch size as a public constant, *e.g.*, 1000, for better efficiency.

**Design of Protocol $\Pi_{\mathsf{AppQua}}$.** With the above considerations, we propose our fast 2PC approximated quantile protocol as follows:

(1) Each party inserts its local inputs ($\mathbf{u}$ or $\mathbf{v}$) into local sketch buckets using the Insert instruction (c.f. Figure 4), then concatenates the negative sketch (in reverse order with logical indices $\tau$), the zero bucket, and the positive sketch into a single vector ($\mathbf{s}_u$ or $\mathbf{s}_v$). [2] An example is illustrated as $[\![\mathbf{s}]\!]$ in Figure 5.
(2) Since DDSketch is fully mergeable by summation, $\mathbf{s}_u$ and $\mathbf{s}_v$ naturally serve as additive shares of the merged sketch $[\![\mathbf{s}]\!]$.

Following the Quantile($\beta$) procedure in Figure 4 naïvely within 2PC is costly, as it necessitates a linear scan over $[\![\mathbf{s}]\!]$, which involves secure comparison and extraction in $O(|\mathbf{s}| \log_2 \ell)$ rounds, to ensure oblivious access. To address this, we design an MPC-friendly variant that requires only a *single* batch comparison plus local operations in $O(\log_2 \ell)$ rounds, which is independent of $|\mathbf{s}|$. First, since the bucket logical index $\tau$ is public, we precompute a quantile vector $\mathbf{q}$ as Eq (4), and an example is given in Figure 5.

$$\mathbf{q}_\tau = \begin{cases} \frac{2\gamma^\tau}{\gamma+1}, & \tau = \lceil \log_\gamma(x) \rceil \in B^+, \\ 0, & x = 0, \\ -\frac{2\gamma^\tau}{\gamma+1}, & \tau = \lceil \log_\gamma(|x|) \rceil \in B^-. \end{cases} \tag{4}$$

Next, two parties compute the prefix sum $[\![\mathbf{s}'_i]\!] = \sum_{j=1}^{i} [\![\mathbf{s}_j]\!]$. We then compare target $k = \lfloor \beta \cdot (H_1 + H_2) \rfloor$ with $[\![\mathbf{s}']\!]$ to obtain $[\![\mathbf{a}]\!] =$

[2] Note that each bucket is with a public logical index $\tau$ that is computed as $\tau = \lceil \log_\gamma(x) \rceil$ (which can be negative). This is different from the position index $i$ (which starts from 1) in the vector. We distinguish these two kinds of indices in this work.

```
Inputs: $\mathcal{P}_A$ inputs $\mathbf{u} \in \mathbb{R}^{H_1}$ and $\mathcal{P}_B$ inputs $\mathbf{v} \in \mathbb{R}^{H_2}$. Percentiles $0 \le \beta_1 < \beta_2 < \cdots < \beta_n \le 1$. Relative-error ratio $0 < \alpha < 1$, $\gamma = (1+\alpha)/(1-\alpha)$, buckets number $\eta$.
Outputs: $\{[\![q_i]\!] \approx [\![\text{sorted}(\mathbf{u} \cup \mathbf{v})_{\lfloor \beta_i \cdot (M_1+M_2) \rfloor}]\!]\}_{i=1}^{n}$.
 1: Let $k_i = \lfloor \beta_i \cdot (H_1 + H_2) \rfloor$ for $i \in [1, n]$.
 2: $\mathcal{P}_A$ initializes sketch buckets $\mathsf{B}^{A,-} = \{0\}^{\eta}$, $\mathsf{B}^{A,0} = 0$, and $\mathsf{B}^{A,+} = \{0\}^{\eta}$ for $\mathbb{R}_{<0}$, 0, and $\mathbb{R}_{>0}$, where $\mathsf{B}^{A,-}$ and $\mathsf{B}^{A,+}$ are both with $\eta$ buckets, $\mathsf{B}^{A,0}$ has one bucket.
 3: $\mathcal{P}_B$ sets $\mathsf{B}^{B,-} = \{0\}^{\eta}$, $\mathsf{B}^{B,0} = 0$, and $\mathsf{B}^{B,+} = \{0\}^{\eta}$ similarly.
 4: for all $u \in \mathbf{u}$ do
 5:     $\mathcal{P}_A$ computes $\tau = \lceil \log_{\gamma}(u) \rceil$ and $\mathsf{B}^{A,+}_{\tau} = \mathsf{B}^{A,+}_{\tau} + 1$ if $u > 0$; $\mathsf{B}^{A,0} = \mathsf{B}^{A,0} + 1$ if $u = 0$; and $\tau = \lceil \log_{\gamma}(|u|) \rceil$ and $\mathsf{B}^{A,-}_{\tau} = \mathsf{B}^{A,-}_{\tau} + 1$ if $u < 0$.
 6: end for
 7: $\mathcal{P}_A$ reverses $\mathsf{B}^{A,-}$, sets $[\![\mathbf{s}]\!]^A = \mathsf{RVS}(\mathsf{B}^{A,-}, \tau) \mid \mathsf{B}^{A,0} \mid \mathsf{B}^{A,+}$.
 8: Similarly, $\mathcal{P}_B$ inserts $\forall v \in \mathbf{v}$ into $\mathsf{B}^{B,-}, \mathsf{B}^{B,0}$, and $\mathsf{B}^{B,+}$, and gets $[\![\mathbf{s}]\!]^B = \mathsf{RVS}(\mathsf{B}^{B,-}, \tau) \mid \mathsf{B}^{B,0} \mid \mathsf{B}^{B,+}$.
 9: Two parties compute prefix sum as $[\![\mathbf{s}'_i]\!] = \sum_{j=1}^{i} [\![\mathbf{s}_j]\!]$ for position index $i \in [1, 2\eta + 1]$.
10: Set public vector $\mathbf{q}$ of size-$(2\eta + 1)$.
11: for all position index $i \in [1, 2\eta + 1]$ do
12:     Let the logical index of $\mathbf{s}_i$ be $\tau$. Set $\mathbf{q}_i = -\frac{2\gamma^{\tau}}{\gamma+1}$ if $i \le \eta$; $\mathbf{q}_i = 0$ if $i = \eta + 1$; and $\mathbf{q}_i = \frac{2\gamma^{\tau}}{\gamma+1}$ otherwise.
13: end for
14: for all $i \in [1, n]$, in parallel do
15:     $\mathcal{P}_A$ and $\mathcal{P}_B$ jointly compare $[\![\mathbf{a}_{(i)}]\!] = \Pi_{\mathsf{LE}}(k_i, [\![\mathbf{s}']\!])$, logical-right shift $[\![\mathbf{a}_{(i)}]\!]$ to obtain $[\![\mathbf{a}^r_{(i)}]\!] = [\![\mathbf{a}_{(i)}]\!] \gg 1$, and get one-hot vector $[\![\mathbf{a}'_{(i)}]\!] = [\![\mathbf{a}_{(i)}]\!] \oplus [\![\mathbf{a}^r_{(i)}]\!]$.
16: end for
17: return $[\![\mathbf{M}]\!] \cdot \mathbf{q}^{\top}$, where $[\![\mathbf{M}]\!] = [[\![\mathbf{a}'_{(1)}]\!], \ldots, [\![\mathbf{a}'_{(n)}]\!]]^{\top}$.
```

**Figure 6: Two-party approximate quantile protocol $\Pi_{\mathsf{AppQua}}$.**

$\Pi_{\mathsf{LE}}(k, [\![\mathbf{s}']\!])$. A logical right shift of $[\![\mathbf{a}]\!]$ by one position gives $[\![\mathbf{a}^r]\!] = [\![\mathbf{a}]\!] \gg 1$, and their logical XOR yields $[\![\mathbf{a}']\!] = [\![\mathbf{a}]\!] \oplus [\![\mathbf{a}^r]\!]$, such that $[\![\mathbf{a}']\!]$ is a one-hot vector with the 1 indicating the bucket containing the $\beta$-quantile. The $\beta$-quantile is then computed as $[\![q_{\beta}]\!] = [\![\mathbf{a}']\!] \cdot \mathbf{q}$. Since $\mathbf{q}$ is public, this step requires no communication.

For computing $n$ quantiles in parallel, we arrange their one-hot vectors into a matrix $[\![\mathbf{M}]\!] \in \{0,1\}^{n \times |\mathbf{s}|}$ in increasing order along with $\beta_i$, and compute $[\![\mathbf{q}']\!] = [\![\mathbf{M}]\!] \cdot \mathbf{q}^{\top}$, where $\mathbf{q}'_i$ is the $\beta_i$-th quantile. The secret intervals $[\![I_i]\!]$ are then given by $([\![\mathbf{q}'_{i-1}]\!], [\![\mathbf{q}'_i]\!]]$. Finally, since quantiles for different numerical features are independent, they can be computed fully in parallel. The detailed protocol $\Pi_{\mathsf{AppQua}}$ is formulated in Figure 6.

## 4.2 Two-Party Count & WoE

After generating $[\![\mathcal{B}]\!]$, we count the additive secret shares of required statistics and then compute WoE securely.

### 4.2.1 *Statistics Count.*

We propose efficient 2PC statistics counting methods as follows:

**Case-I.** In the $\mathbb{VP}$ setting, for dataset $\mathbf{V}$, $\mathcal{P}_B$ can locally compute its WoE and subsequent transformation as $\mathcal{P}_B$ has all labels. Therefore, we focus on the procedures towards $\mathbf{U}$: $\mathcal{P}_A$ holds the indicator matrix $\mathcal{B}_U$ and $\mathcal{P}_B$ has label vector $\mathbf{y}$. Given the indicator vector $\mathbf{b}$ of one category (or interval) of one feature, counting the positive samples is equivalent to computing the inner product of $\mathbf{b}$ and $\mathbf{y}$. In vectorization, two parties perform the following steps:

- Two parities invoke matrix-vector multiplication $[\![\mathbf{z}^+]\!] = \Pi_{\mathsf{MV}}(\mathcal{B}_U^{\top}, \mathbf{y})$, where $\mathcal{P}_A$ inputs $\mathcal{B}_U^{\top}$ and $\mathcal{P}_B$ inputs $\mathbf{y}$. $\{\mathbf{z}^+_{K_{j-1}+1}, \ldots, \mathbf{z}^+_{K_{j-1}+K_j}\}$ represents the number of positive samples (*a.k.a.*, $\mathbf{y}_i = 1$) for all categories (or intervals) of the $j$-th feature.
- $\mathcal{P}_A$ sums up $\mathcal{B}$ column-wise, and two parties count negative samples $[\![\mathbf{z}^-]\!] = \sum_{i=1}^{H} \mathcal{B}_i - [\![\mathbf{z}^+]\!]$. $\mathcal{P}_B$ can locally compute $\xi = (\sum_i^H 1\{\mathbf{y}_i = 0\}) / (\sum_i^H 1\{\mathbf{y}_i = 1\})$ for subsequent computation.

**Case-II.** For the $\mathbb{HP}$ setting, $\mathcal{P}_A$ and $\mathcal{P}_B$ can compute the additive shares of required counts as follows:

- Two parties compute $[\![\mathbf{z}^+]\!] = \Pi_{\mathsf{MV}}([\![\mathcal{B}_U^{\top}]\!], \mathbf{y}_U) + \Pi_{\mathsf{MV}}([\![\mathcal{B}_V^{\top}]\!], \mathbf{y}_V)$, where $\mathcal{P}_A$ provides $\mathbf{y}_U$ for the former and $\mathcal{P}_B$ provides $\mathbf{y}_V$ for the latter. And $[\![\mathbf{z}^-]\!]$ is obtained via subtraction similar to **Case I**.
- Two parties count the number of positive samples and negative samples as $[\![\sum_i^H 1\{\mathbf{y}_i = 1\}]\!]$ and $[\![\sum_i^H 1\{\mathbf{y}_i = 0\}]\!]$ locally, and jointly compute $[\![\xi]\!] = \Pi_{\mathsf{Div}}([\![\sum_i 1\{\mathbf{y}_i = 0\}]\!], [\![\sum_i 1\{\mathbf{y}_i = 1\}]\!])$.

### 4.2.2 *Local Counts for Categorical Feature in $\mathbb{HP}$.*

As discussed in § 4.1.1, in the $\mathbb{HP}$ setting, each party holds its local indicator matrix and label for categorical features. Assuming the $j$-th feature is a categorical ones, we get $[\![\mathbf{z}^{(j),+}]\!]$ without communication: $\mathcal{P}_A$ computes $[\![\mathbf{z}^{(j),+}]\!]^A = \mathbf{B}_U^{(j),\top} \cdot \mathbf{y}_U$, and $\mathcal{P}_B$ computes $[\![\mathbf{z}^{(j),+}]\!]^B = \mathbf{B}_V^{(j),\top} \cdot \mathbf{y}_V$. $[\![\mathbf{z}^{(j),-}]\!]$ is subsequently obtained via subtraction. The resulting $\{[\![\mathbf{z}^{(j),+}]\!], [\![\mathbf{z}^{(j),-}]\!]\}_j$ are then concatenated with the statistics of numerical features to get the final $[\![\mathbf{z}^+]\!]$ and $[\![\mathbf{z}^-]\!]$ across all features.

### 4.2.3 *Vectorized Secure WoE.*

With the above statistics, including $[\![\mathbf{z}^+]\!]$, $[\![\mathbf{z}^-]\!]$, and $[\![\xi]\!]$, we can compute the WoE in vectorization by invoking 2PC division, multiplication, and logarithm in element-wise: $[\![\mathbf{w}]\!] = \Pi_{\mathsf{Log}}(\Pi_{\mathsf{Mul}}(\Pi_{\mathsf{Div}}([\![\mathbf{z}^+]\!], [\![\mathbf{z}^-]\!]), [\![\xi]\!]))$, with $\mathbf{w} \in \mathbb{Z}_L^{\sum_{j=1}^{W} K_j}$. And the WoE values of the $j$-th feature can be extracted from $[\![\mathbf{w}]\!]$ as $[\![\mathbf{w}^{(j)}]\!] = \{[\![\mathbf{w}_{K_{j-1}+1}]\!], \ldots, [\![\mathbf{w}_{K_{j-1}+K_j}]\!]\}$ (with $K_0 = 0$).

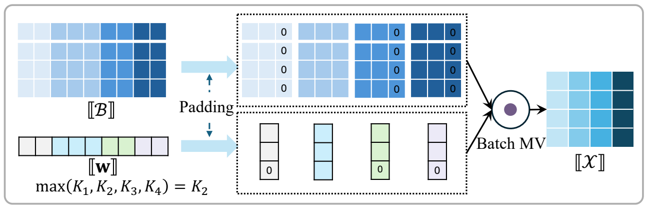


**Figure 7: A toy example of padding and HE-based batch matrix-vector multiplication.** $(K_1, K_2, K_3, K_4) = (2, 3, 2, 2)$**.**

## 4.3 Two-Party Batch Transformation

Recall that for the $j$-th feature with $K_j$ categories (intervals), we generate an indicator matrix $\mathbf{B}^{(j)} \in \{0,1\}^{H \times K_j}$, which is either held by $\mathcal{P}_A$ in $\mathbb{VP}$ or secret-shared as $[\![\mathbf{B}^{(j)}]\!]$ in $\mathbb{HP}$. Because the categories are mutually exclusive, each row of $\mathbf{B}^{(j)}$ corresponds to a one-hot vector; that is, for every row $i \in [1, H]$, there exists exactly one $k$ such that $\mathbf{B}_i^{(j),k} = 1$, while $\mathbf{B}_i^{(j),k'} = 0$ for $\forall k' \neq k$.

**Procedure for $\mathbb{VP}$ Setting $\Pi^{\mathbb{VP}}_{\text{ALIOTH}}$.**

**Inputs:** $\mathcal{P}_A$ inputs feature matrix $\mathbf{U} \in \mathbb{R}^{H\times W}$, $\mathcal{P}_B$ inputs label vector $\mathbf{y} \in \{0,1\}^H$. $K_j$ categories or intervals of the $j$-th feature.

**Outputs:** $\mathcal{P}_A$ and $\mathcal{P}_B$ outputs the shares $[\![\mathcal{X}]\!]^A$ and $[\![\mathcal{X}]\!]^B$ respectively, such that $[\![\mathcal{X}]\!]^A + [\![\mathcal{X}]\!]^B = \mathcal{X} = \mathsf{WoE}(\mathbf{U}, \mathbf{y})$.

```
1: for $j \in [1, W]$, $\mathcal{P}_A$ locally do
2:   if $\mathsf{Type}(\mathbf{U}^j) = \mathsf{Numerical}$ then
3:     $\mathcal{P}_A$ computes the $K_j$ non-overlapping equal-frequency intervals $\{B^j_k = (b^j_{k-1}, b^j_k]\}_{k=1}^{K_j}$, where $b^j_0 < b^j_1 < \cdots < b^j_{K_j}$.
4:   end if
5:   $\mathcal{P}_A$ generates indicator vector $\mathbf{b}^{(j,k)} = \mathbf{1}\{\mathbf{U}^j = f^j_k\}$ (resp. $\mathbf{b}^{(j,k)} = \mathbf{1}\{\mathbf{U}^j \in B^j_k\}$) for $f^j_k$ (resp. $B^j_k$), the $k$-th category (resp. interval) of the $j$-th feature for $k \in [1, K_j]$.
6:   $\mathcal{P}_A$ concatenates all indicator vectors column-wise as a sub-matrix $\mathbf{B}^{(j)} = [\mathbf{b}^{(j,1)} \mid \mathbf{b}^{(j,2)} \mid \cdots \mid \mathbf{b}^{(j,K_j)}]$.
7: end for
8: $\mathcal{P}_A$ concatenates all indicator sub-matrices in column-wise as one big matrix $\mathcal{B} = [\mathbf{B}^{(1)} \mid \mathbf{B}^{(2)} \mid \cdots \mid \mathbf{B}^{(W)}]$.
9: Two parties compute $[\![\mathbf{z}^+]\!] = \Pi_{\mathsf{MV}}(\mathcal{B}^\top, \mathbf{y})$, where $\mathcal{P}_A$ inputs $\mathcal{B}^\top$ and $\mathcal{P}_B$ inputs $\mathbf{y}$.
10: $\mathcal{P}_A$ sums up $\mathcal{B}$ column-wise and two parties compute $[\![\mathbf{z}^-]\!] = \sum_i^H \mathcal{B}_i - [\![\mathbf{z}^+]\!]$, $\mathcal{P}_B$ computes $\xi = (\sum_i^H 1\{\mathbf{y}_i = 0\})/(\sum_i^H 1\{\mathbf{y}_i = 1\})$.
11: Two parties sequently compute $[\![\mathbf{e}]\!] = \Pi_{\mathtt{Div}}([\![\mathbf{z}^+]\!], [\![\mathbf{z}^-]\!])$, $[\![\mathbf{f}]\!] = \Pi_{\mathtt{Mul}}([\![\mathbf{e}]\!], \xi)$, and $[\![\mathbf{w}]\!] = \Pi_{\mathtt{log}}([\![\mathbf{f}]\!])$ in element-wise.
12: Let $K_m = \max(K_1, K_2, \ldots, K_W)$, parties extract $[\![\mathbf{w}^{(j)}]\!] = \{[\![\mathbf{w}_{K_{j-1}+1}]\!], \ldots, [\![\mathbf{w}_{K_{j-1}+K_j}]\!]\}$ from $[\![\mathbf{w}]\!]$, zero-pad $\mathbf{B}^{(j)}$ to size $H \times K_m$ and $[\![\mathbf{w}^{(j)}]\!]$ to $K_m$, and compute $\{[\![\mathcal{X}^j]\!] = \Pi_{\mathsf{MV}}(\mathbf{B}^{(j)}, [\![\mathbf{w}^{(j)}]\!])\}_j$ for $j \in [1, W]$ using batched matrix-vector multiplication.
13: return Two parties output secret-shared $[\![\mathcal{X}]\!]$, where $\mathcal{X} = [\mathcal{X}^1 \mid \mathcal{X}^2 \mid \cdots \mid \mathcal{X}^W]$.
```

**Procedure for $\mathbb{HP}$ Setting $\Pi^{\mathbb{HP}}_{\text{ALIOTH}}$.**

**Inputs:** $\mathcal{P}_A$ inputs feature matrix $\mathbf{U} \in \mathbb{R}^{H_1\times W}$ and label $\mathbf{y}_U \in \{0,1\}^{H_1}$, $\mathcal{P}_B$ inputs feature matrix $\mathbf{V} \in \mathbb{R}^{H_2\times W}$ and label vector $\mathbf{y}_V \in \{0,1\}^{H_2}$. $K_j$ categories or intervals of the $j$-th feature.

**Outputs:** $\mathcal{P}_A$ and $\mathcal{P}_B$ outputs the shares $[\![\mathcal{X}]\!]^A$ and $[\![\mathcal{X}]\!]^B$ respectively, such that $[\![\mathcal{X}]\!]^A + [\![\mathcal{X}]\!]^B = \mathcal{X} = \mathsf{WoE}(\mathbf{U} \cup \mathbf{V}, \mathbf{y}_U \cup \mathbf{y}_V)$.

```
1: for $j \in [1, W]$, in parallel do
2:   if $\mathsf{Type}(\mathbf{U}^j) = \mathsf{Numerical}$ then
3:     $\mathcal{P}_A$ and $\mathcal{P}_B$ jointly invoke $\Pi_{\mathsf{AppQua}}$ to get the quantiles $[\![\mathbf{q}]\!] = [[\![q_1]\!], [\![q_2]\!], \ldots, [\![q_{K_j}]\!]]$. ▷ $([\![q_i]\!], [\![q_{i+1}]\!]]$ define Interval $[\![I_i]\!]$.
4:     for all $[\![q_k]\!] \in \{[\![q_1]\!], [\![q_2]\!], \ldots, [\![q_{K_j}]\!]\}$, in parallel do
5:       Two parties compare $[\![\mathbf{c}^j_{U,k}]\!] = \Pi_{\mathsf{LT}}(\mathbf{U}^j, [\![q_k]\!])$, where $\mathcal{P}_A$ provides $\mathbf{U}^j$, so indicator vector is $[\![\mathbf{b}^j_{U,k}]\!] = [\![\mathbf{c}^j_{U,k-1}]\!] \oplus [\![\mathbf{c}^j_{U,k}]\!]$.
6:       Asymmetrically, two parties compute indicator vector $[\![\mathbf{b}^j_{V,k}]\!]$.
7:     end for
8:     Get indicator matrices $[\![\mathbf{B}^{(j)}_U]\!]$ (resp. $[\![\mathbf{B}^{(j)}_V]\!]$) by concatenating $\{[\![\mathbf{b}^j_{U,k}]\!]\}_{k=1}^{K_j}$ (resp. $\{[\![\mathbf{b}^j_{V,k}]\!]\}_{k=1}^{K_j}$) column-wise.
9:   else if $\mathsf{Type}(\mathbf{U}^j) = \mathsf{Category}$ then
10:    $\mathcal{P}_A$ (resp. $\mathcal{P}_B$) locally generate $\mathbf{B}^{(j)}_U$ (resp. $\mathbf{B}^{(j)}_V$).
11:  end if
12: end for
13: Two parties get indicator matrices for $\mathbf{U}$ (resp. $\mathbf{V}$) as $[\![\mathcal{B}_U]\!]$ (resp. $[\![\mathcal{B}_V]\!]$), and global $[\![\mathcal{B}]\!] = [\![\mathcal{B}_U]\!] \cup [\![\mathcal{B}_V]\!]$ row-wise.
14: $\mathcal{P}_A$ and $\mathcal{P}_B$ jointly compute $[\![\mathbf{z}^+]\!] = \Pi_{\mathsf{MV}}([\![\mathcal{B}_U^\top]\!], \mathbf{y}_U) + \Pi_{\mathsf{MV}}([\![\mathcal{B}_V^\top]\!], \mathbf{y}_V)$, where $\mathcal{P}_A$ provides $\mathbf{y}_U$ and $\mathcal{P}_B$ provides $\mathbf{y}_V$. And $[\![\mathbf{z}^-]\!] = \sum_{i=1}^H [\![\mathcal{B}_i]\!] - [\![\mathbf{z}^+]\!]$. ▷ We use *local counts optimization for categorical features*, and omit here for brevity.
15: $\mathcal{P}_A$ and $\mathcal{P}_B$ sequently compute $[\![\mathbf{e}]\!] = \Pi_{\mathtt{Div}}([\![\mathbf{z}^+]\!], [\![\mathbf{z}^-]\!])$, $[\![\mathbf{f}]\!] = \Pi_{\mathtt{Mul}}([\![\mathbf{e}]\!], [\![\xi]\!])$, and $[\![\mathbf{w}]\!] = \Pi_{\mathtt{log}}([\![\mathbf{f}]\!])$ in element-wise, where scalar $[\![\xi]\!] = \Pi_{\mathtt{Div}}([\![\sum_i^H 1\{\mathbf{y}_i = 0\}]\!], [\![\sum_i^H 1\{\mathbf{y}_i = 1\}]\!])$.
16: Extract $[\![\mathbf{w}^{(j)}]\!]$, pad $[\![\mathbf{B}^{(j)}]\!]$ and $[\![\mathbf{w}^{(j)}]\!]$ as step 18 of $\Pi^{\mathbb{VP}}_{\text{ALIOTH}}$ and compute $\{[\![\mathcal{X}^j]\!] = \Pi_{\mathsf{MV}}([\![\mathbf{B}^{(j)}]\!], [\![\mathbf{w}^{(j)}]\!])\}_j$ for $j \in [1, W]$.
17: return Two parties output secret-shared $[\![\mathcal{X}]\!]$, where $\mathcal{X} = [\mathcal{X}^1 \mid \mathcal{X}^2 \mid \cdots \mid \mathcal{X}^N]$.
```

**Figure 8: The full two-party framework $\Pi_{\text{ALIOTH}}$.**

In the $\mathbb{VP}$ setting, the $j$-th column of $[\![\mathcal{X}]\!]$ is computed as

$$\begin{aligned} [\![\mathcal{X}^j]\!] &= \Pi_{\mathtt{Mul}}(\mathbf{B}^{(j),1}, [\![\mathbf{w}^{(j)}_1]\!]) + \Pi_{\mathtt{Mul}}(\mathbf{B}^{(j),2}, [\![\mathbf{w}^{(j)}_2]\!]) \\ &\quad + \cdots + \Pi_{\mathtt{Mul}}(\mathbf{B}^{(j),K_j}, [\![\mathbf{w}^{(j)}_{K_j}]\!]) \\ &= \Pi_{\mathsf{MV}}(\mathbf{B}^{(j)}, [\![\mathbf{w}^{(j),\top}]\!]) \end{aligned}, \tag{5}$$

where $\mathcal{P}_A$ provides $\mathbf{B}^{(j)}$ and $\mathbf{B}^{(j),k}$ is its $k$-th column.

$[\![\mathcal{X}^j]\!]_{j=1}^N$ are independent and therefore can be computed in parallel using batched matrix–vector multiplication [33]. However, since different features can be associated with different $K_j$ (*a.k.a.*, categories or intervals). This mismatch in dimensions prevents direct utilization of batched matrix–vector multiplication.

To overcome this, we propose a padding technique before batching. As illustrated in Figure 7, we first compute $K_m = \max(\{K_j\}_{j=1}^W)$

and then apply zero-padding as follows: i) append $(K_m - K_j)$ zero columns to $\mathbf{B}^{(j)}$, and ii) append $(K_m - K_j)$ zero entries to $\mathbf{w}^{(j),\top}$. Afterwards, all matrices and vectors have uniform dimensions. Similarly, in $\mathbb{HP}$, $\{[\![\mathcal{X}^j]\!] = \Pi_{\mathsf{MV}}([\![\mathbf{B}^{(j)}]\!], [\![\mathbf{w}^{(j)}]\!])\}_j$ can be padded and accelerated using batching matrix-vector multiplication as well.

Although the padding approach introduces "useless" zeros, the batched matrix-vector multiplication still yields nearly a 7%–30% speedup over the naïve non-batched method, even when $K_m$ is significantly larger than most $K_j$ (see Appendix 6.3.2). Put it all together, we present $\Pi_{\text{Alioth}}$ in Figure 8.

Remark 2 (Generality). *Some of our proposed protocols, e.g., 2PC approximated quantiles and batch transformation, are of independent interest. And the building blocks of Alioth are modular, making it easy to substitute any component, e.g., replacing the equal-frequency discretization with equal-width ones.*

## 5 SECURITY AND COMPLEXITY ANALYSIS

**Security Analysis.** Theorem 1 captures the security of Alioth.

Theorem 1 (Security). *In the hybrid model, Alioth securely realizes functionality $\mathcal{F}_{\text{Alioth}}$ against a semi-honest static polynomial probabilistic adversary $\mathcal{A}$, who corrupts no more than one party.*

Proof. Alioth is constructed directly from well-established cryptographic primitives in a sequential model. Apart from the necessary interactions within these primitives, all remaining operations are performed locally. Specifically:

- $[\![\mathcal{B}]\!]$ **Generation.** As shown in § 4.1.1, $[\![\mathcal{B}]\!]$ over $\mathbb{VP}$ and categorical features in $\mathbb{HP}$ can be generated non-interactively. For $[\![\mathcal{B}]\!]$ over numerical features in $\mathbb{HP}$ (§ 4.1.2), $\mathcal{P}_0$ and $\mathcal{P}_A$ engage in communication only during the 2PC Interval Test and the $\Pi_{\mathsf{AppQua}}$ protocol: i) The Interval Test relies on $\Pi_{\mathsf{LE}}$ and $\Pi_{\mathsf{B2A}}$. ii) In $\Pi_{\mathsf{AppQua}}$ (Figure 6), steps 1–14, 18, and 19 are purely local computations, while steps 15–17 involve $\Pi_{\mathsf{LE}}$ and local operations.
- **Count & WoE.** For secure count statistics, only $\Pi_{\mathsf{MV}}$ and local arithmetic are required for $\mathbb{VP}$ setting. Under $\mathbb{HP}$, we use $\Pi_{\mathsf{MV}}$ and $\Pi_{\mathtt{Div}}$ in a black-box manner, alongside local arithmetic operations. For 2PC WoE, the computation invokes $\Pi_{\mathtt{Div}}$, $\Pi_{\mathsf{Mul}}$, and $\Pi_{\mathsf{Log}}$ sequentially in arithmetic black-box model.
- **Batch Trans.** The padding procedure is local, and batched matrix–vector multiplication is securely inherited from [33].

Consequently, the security of Alioth follows naturally from the security of well-established 2PC primitives in the hybrid model. □

**Complexity Analysis.** Assume the dataset is of shape $H \times W$, with $W_c$ categorical features and $W_n$ numerical ones ($W_c + W_n = W$). For the $j$-th feature, it has $K_j$ categories or intervals. In $\mathbb{VP}$, $\mathcal{P}_A$ holds $W_1$ features, $\mathcal{P}_B$ holds $W_2$ features and labels, where $W = W_1 + W_2$; In $\mathbb{HP}$, $\mathcal{P}_A$ holds $H_1$ samples, $\mathcal{P}_B$ holds $H_2$ samples, where $H = H_1 + H_2$. The values are encoded in ring $\mathbb{Z}_{2^\ell}$. The RLWE-based $\mathsf{HE}$ is defined on the set of integer polynomials $\mathbb{A}_{N,q} = \mathbb{Z}_q[X]/(X^N + 1)$ with 2-power number $N$ and $q > 0$.

- $[\![\mathcal{B}]\!]$ **Generation.** For the **VP** setting and categorical features of **VP** setting, each party can compute the indicator vectors for each feature locally, which is free in 2PC. For the numerical features under **VP** setting, let $K_n = \sum_{j=1}^{W_n} K_j$ for brevity, protocol $\Pi_{\mathsf{AppQua}}$ requires $O(|\mathbf{s}|K_n)$ invocations of $\Pi_{\mathsf{LT}}$, and 2PC interval test requires $O(HK_n)$ invocations of $\Pi_{\mathsf{LT}}$ and $\Pi_{\mathsf{B2A}}$. This requires a total of $O((|\mathbf{s}| + H)K_n)\ell)$ bits in $O(2\log_2 \ell)$ rounds.
- **Count & WoE.** In terms of the Count phase, for the **VP** setting, we conduct $\Pi_{\mathsf{MV}}$ over a private matrix of size $(\sum_{j=1}^{W_1} K_j) \times H$ and a private vector of size $H$, let $K' = \sum_{j=1}^{W_1} K_j$, this requires $O(HK'/N)$ homomorphic multiplications and additions, $O(K'/\sqrt{N})$ automorphisms, and sending $O((H + K')/N)$ RLWE ciphertexts. For the **HP** setting, let $K = \sum_{j=1}^{W} K_j$ two parties requires $O((H_1 + H_2)K/N)$ homomorphic multiplications and additions, $O(2K/\sqrt{N})$ automorphisms, and sending $O((H_1+H_2+2K)/N)$ RLWE ciphertexts.
- **Batch Trans.** Let $K_m$ denote the maximum of $\{K_j\}_j$, we batching compute $W$ matrix-vector multiplication of size $H \times K_m$ and $K_m$, this requires $O(HWK_m/N)$ homomorphic multiplications and additions, $O(HW/\sqrt{N})$ automorphisms, and sending $O(W \cdot \lceil H/N \rceil)$ ciphertexts, which can be packed as $O(\lceil HW/N \rceil)$ ciphertexts when $H < N/2$.

## 6 EXPERIMENTAL EVALUATION

We provide Alioth implementation setup and study the performance by answering the following questions.

- **Q1:** How about the end-to-end evaluation of Alioth when processing categorical and numerical features under both $\mathbb{VP}$ and $\mathbb{HP}$ settings? (§ 6.2)
- **Q2:** What are the overheads of our proposed building blocks and protocols? (§ 6.3)
- **Q3:** Can Alioth support efficient ML applications, including feature selection and logistic regression training? (§ 6.4)

### 6.1 Experimental Setup

**Testbed.** Experiments are run on a machine with Intel(R) Xeon(R) Silver 4314@2.40 GHz CPU and 500GB RAM. The Operating System Ubuntu 22.04.4 LTS is Linux kernel 5.4.0-172-generic. A local-area network (LAN, RTT: 1ms, 1Gbps) and a wide-area network (WAN, RTT: 50ms, 160Mbps) is simulated by Linux tc command.

**Code Base.** We implement Alioth on SecretFlow-SPU [34] with 2PC backend [23, 33]. We leverage Ferret implemented in the YACL [4] for VOLE-style OT, and SEAL library [3] with Intel HEXL VX512 CPU acceleration [7] for RLWE-based $\mathsf{HE}$, using parameters recommended by [33]. All of them provide at least $\lambda = 128$ bits of security. For arithmetic, we select the ring $\mathbb{Z}_{2^{64}}$ for additive secret sharing and set $f = 18$ fractional bits for fixed-point encoding.

**Datasets, Models, and Metrics** We generate synthetic datasets of diverse sizes for end-to-end evaluation and building blocks profiling. For the practical applications study, we evaluate Alioth on two financial risk datasets: German-Credit-Data [20] and Home-Credit-Default-Risk [40]. We conduct IV-based feature selection and LR training based on our WoE-transformed datasets, report time and communication, and the predictive performance of LR in terms of AUC and weighted F1 score [22, 41].

### 6.2 End-to-End Evaluation

We evaluate the end-to-end overhead of Alioth on synthetic data with different samples $H \in \{1{,}000, 10{,}000, 100{,}000\}$, features $W \in \{10, 50, 100\}$, and categories (or intervals) $K \in \{5, 10, 20\}$. We report

**Table 3: Time and communication of Alioth under different $H$, $W$, and $K$ for categorical and numerical features under $\mathbb{VP}$ and $\mathbb{HP}$ settings in LAN and WAN. Time is in seconds and communication is in MB. Time in WAN is marked in gray for clarity.**

| Partition | $H \downarrow$ | | $W = 10$ | | | $W = 50$ | | | $W = 100$ | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $K \rightarrow$ | 5 | 10 | 20 | 5 | 10 | 20 | 5 | 10 | 20 |
| (a) $\mathbb{VP}$ | 1,000 | LAN | 2.09 | 2.09 | 2.66 | 3.23 | 3.89 | 4.73 | 4.70 | 5.60 | 8.15 |
| | | WAN | 13.32 | 13.54 | 14.93 | 21.03 | 23.21 | 27.86 | 29.98 | 34.15 | 41.97 |
| | | Comm | 32.07 | 32.44 | 33.46 | 68.29 | 69.15 | 76.69 | 113.16 | 114.88 | 129.05 |
| | 10,000 | LAN | 2.30 | 2.36 | 3.06 | 4.35 | 5.19 | 6.68 | 5.95 | 8.31 | 11.60 |
| | | WAN | 13.91 | 14.41 | 16.06 | 23.71 | 26.84 | 32.84 | 34.63 | 40.98 | 51.88 |
| | | Comm | 37.81 | 37.92 | 39.33 | 93.08 | 96.85 | 104.38 | 162.20 | 169.74 | 183.91 |
| | 100,000 | LAN | 3.79 | 5.15 | 7.24 | 11.65 | 19.47 | 29.20 | 20.91 | 37.46 | 51.57 |
| | | WAN | 18.60 | 21.20 | 25.74 | 47.89 | 61.05 | 82.92 | 83.19 | 111.32 | 153.86 |
| | | Comm | 47.52 | 47.57 | 48.10 | 181.56 | 183.17 | 186.39 | 349.26 | 352.49 | 360.24 |
| (b) $\mathbb{HP}$-Numerical | 1,000 | LAN | 6.29 | 6.76 | 7.02 | 7.21 | 8.37 | 8.78 | 9.02 | 9.67 | 13.53 |
| | | WAN | 33.04 | 33.87 | 35.51 | 45.31 | 46.81 | 47.99 | 54.93 | 59.04 | 67.92 |
| | | Comm | 58.51 | 63.45 | 74.30 | 110.53 | 135.62 | 191.61 | 175.10 | 225.27 | 341.33 |
| | 10,000 | LAN | 6.98 | 8.17 | 9.64 | 12.37 | 16.59 | 29.77 | 18.28 | 29.96 | 54.81 |
| | | WAN | 36.22 | 37.40 | 43.40 | 55.05 | 70.15 | 100.99 | 81.83 | 114.97 | 178.78 |
| | | Comm | 99.63 | 147.33 | 242.97 | 315.97 | 559.39 | 1046.01 | 584.92 | 1072.01 | 2054.06 |
| | 100,000 | LAN | 16.34 | 27.37 | 51.20 | 62.48 | 118.12 | 248.87 | 118.84 | 236.55 | 433.74 |
| | | WAN | 59.98 | 89.71 | 150.59 | 187.03 | 352.91 | 654.91 | 345.92 | 648.42 | 1032.68 |
| | | Comm | 516.06 | 994.60 | 1959.80 | 2386.86 | 4800.73 | 9628.74 | 4728.00 | 9555.84 | 18309.98 |
| (c) $\mathbb{HP}$-Categorical | 1,000 | LAN | 2.10 | 2.14 | 2.67 | 3.55 | 3.91 | 4.84 | 4.69 | 5.44 | 7.98 |
| | | WAN | 16.37 | 16.42 | 17.77 | 24.14 | 25.81 | 29.31 | 32.48 | 36.96 | 44.26 |
| | | Comm | 31.48 | 31.84 | 32.87 | 68.58 | 69.44 | 76.99 | 113.46 | 115.18 | 129.39 |
| | 10,000 | LAN | 2.27 | 2.35 | 2.97 | 4.28 | 5.07 | 6.22 | 6.11 | 7.71 | 10.73 |
| | | WAN | 16.83 | 16.99 | 18.39 | 26.46 | 28.75 | 33.17 | 36.50 | 41.24 | 49.78 |
| | | Comm | 36.70 | 36.82 | 38.21 | 92.83 | 96.61 | 104.16 | 161.97 | 169.52 | 183.73 |
| | 100,000 | LAN | 3.71 | 4.64 | 6.18 | 11.42 | 17.01 | 21.76 | 20.03 | 31.60 | 45.60 |
| | | WAN | 20.70 | 21.62 | 23.88 | 44.58 | 51.54 | 60.95 | 74.79 | 88.34 | 104.34 |
| | | Comm | 90.06 | 90.18 | 91.57 | 359.67 | 363.44 | 370.99 | 695.63 | 703.18 | 717.39 |

the costs directly on numerical features for $\mathbb{VP}$ in Table 3 as the 2PC procedures of both kinds of features are identical. For $\mathbb{HP}$, we report the overhead for numerical and categorical separately.

- Obviously, the running time and communication of all cases increase linearly with $H$, $W$, $K$. This trend is expected since Alioth's scalability is primarily determined by these parameters in our design. Besides, we find that in $\mathbb{VP}$ and $\mathbb{HP}$-Categorical settings, the costs increase much more slowly with $K$ than with $H$ and $W$. And $\mathbb{HP}$-Numerical shows significantly higher time and communication costs (approximately 2–3×) than the other two cases. This is because $K$ mainly determines the costs of $[\![\mathcal{B}]\!]$ generation, which is free in these two cases.
- Furthermore, the running time comparison between LAN and WAN reveals the expected latency amplification in WAN: for the same $(H, W, K)$ in the same partition and feature types, we require roughly 3–7× running time in WAN, emphasizing that the communication overhead dominates total time in WAN.

Overall, the results indicate that Alioth exhibits practical scalability with respect to both the dataset size and feature categories. Notably, under the most demanding $\mathbb{HP}$-Numerical setting, we complete the WoE transformation for a dataset comprising 100,000 samples and 100 features, each containing up to 20 categories, within $\approx$ 7 minutes in the LAN and 20 minutes in the WAN, communicating 17 GB messages. Since this transformation is a one-time operation and the transformed datasets can be reused across various downstream ML or data processing tasks, the incurred overhead is acceptable in many practical applications.

## 6.3 Profiling Building Blocks and Protocols

*6.3.1 Overhead of Building Blocks.* In Figure 9, we profile the overhead of building blocks with ($H$ = 10,000, $W$ = 100, $K$ = 20) under three settings.

- $\mathbb{VP}$ and $\mathbb{HP}$-Categorical incur *zero* cost for $[\![\mathcal{B}]\!]$ generation, as they rely solely on local plaintext computation through *zero-sharing*. In contrast, $\mathbb{HP}$-Numerical requires substantial time and communication overhead to generate $[\![\mathcal{B}]\!]$, since it must execute the 2PC $\Pi_{\mathsf{AppQua}}$ and Interval Test protocols, accounting for 70% of the total communication and 90% of the total runtime.

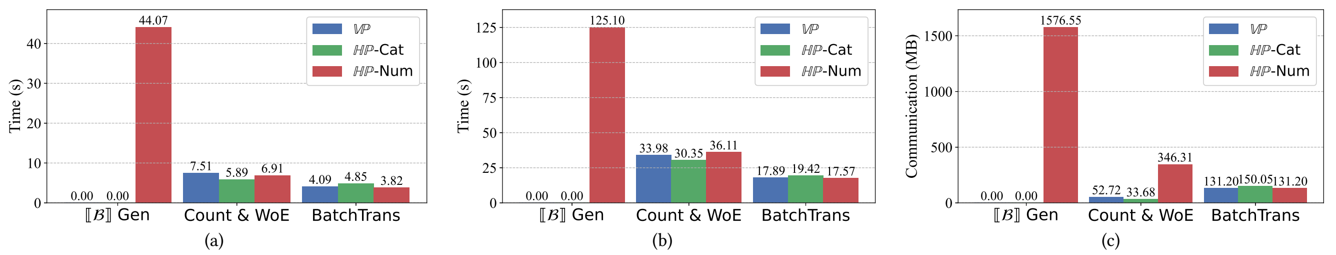


**Figure 9: Profiling the overhead of $[\![\mathcal{B}]\!]$ generation, Count & WoE, and Batch Transformation with $H = 10,000$, $W = 100$, and $K = 20$ in $\mathbb{VP}$, $\mathbb{HP}$-Categorical, and $\mathbb{HP}$-Numerical. Figure 9(a) is for time in LAN, 9(b) is for time in WAN, and 9(c) is for communication.**

**Table 4: Benchmarking $\Pi_{\mathsf{AppQua}}$ with $\beta = 50\%$, $\alpha = 0.01$. Time is in seconds and communication is in MB.**

| $\eta$ | Range | LAN | WAN | Comm. |
|---|---|---|---|---|
| 100 | $[2^{-1}, 2]$ | 1.06 | 3.56 | 1.42 |
| 1,000 | $[2^{-10}, 2^{10}]$ | 1.09 | 3.91 | 1.58 |
| 10,000 | $[2^{-100}, 2^{100}]$ | 2.90 | 11.62 | 15.77 |

- Secondly, the Count & WoE phase incurs higher latency than the Batch Transformation phase, particularly in WAN. This is because Count & WoE involves 2PC protocols for computing nonlinear functions such as comparison, division, and logarithm. Across all cases, the $\mathbb{HP}$-Numerical setting introduces the highest Count & WoE overhead, whereas the $\mathbb{HP}$-Categorical setting incurs the lowest. This stems from the statistics count for categorical features in $\mathbb{HP}$ is free of communication (§ 4.2).

The profiling results highlight that the overhead bottleneck is different across data partitions and feature types. Concretely, the $[\![\mathcal{B}]\!]$ generation is the primary performance bottleneck in $\mathbb{HP}$ for numerical features. In contrast, for $\mathbb{VP}$ and $\mathbb{HP}$-Categorical cases, the Count & WoE requires the highest running time.

#### 6.3.2 *Benchmark $\Pi_{\mathsf{AppQua}}$ and Batch Trans.*

Moreover, we benchmark the efficiency of our proposed basic $\Pi_{\mathsf{AppQua}}$ and batch transformation, which might be of independent interest.

**Performance of $\Pi_{\mathsf{AppQua}}$.** Since the performance of $\Pi_{\mathsf{AppQua}}$ is mainly dominated by the number of buckets $\eta$, we select different $\eta$, estimate range, and evaluate overhead for computing median (*a.k.a.*, $\beta = 50\%$) with $\alpha = 0.01$. And we select $H = 1{,}000$ random numbers from the range as inputs. As shown in Table 4, our $\Pi_{\mathsf{AppQua}}$ is very efficient and it can compute the median of range $[2^{-100}, 2^{100}]$ in around 10 seconds in WAN. In experimental applications, $K = 1{,}000$ is enough for precision requirements.

Note that our $\Pi_{\mathsf{AppQua}}$ differs from recent secure merge protocols [5, 9], which require $O(H)$ time and communication. In contrast, our $\Pi_{\mathsf{AppQua}}$ is more efficient, as its complexity depends only on $\eta$, which usually holds $\eta \ll H$ in practical applications.

**Performance of Batch Transformation.** We compare our batch transformation approach with a naïve baseline that transforms each feature individually in Table 5. In our setup, one feature has $K_m \in \{5, 10, 20\}$ categories, while the remaining nine features each have $\bar{K}$ categories. This configuration is designed to evaluate our method under its least favorable conditions.

Our approach outperforms the naïve method in both running time and communication, which aligns well with our analytical findings. Concretely, we reduce the running time by approximately 7%–30%, and we achieve better communication complexity as well.

**Table 5: Benchmarking batch transformation. Time is in seconds and communication is in MB.**

| $(H, W)$ | $(\bar{K}, K_m)$ | Protocol | LAN | WAN | Comm. |
|---|---|---|---|---|---|
| (10,000, 10) | (5, 5) | Ours | 0.77 | 4.31 | 32.21 |
| | | Naïve | 1.06 | 4.60 | 35.61 |
| | (5, 10) | Ours | 0.84 | 4.37 | 32.21 |
| | | Naïve | 1.20 | 4.74 | 35.61 |
| | (5, 20) | Ours | 0.91 | 4.43 | 32.21 |
| | | Naïve | 1.29 | 4.78 | 35.61 |
| (100,000, 100) | (5, 5) | Ours | 20.68 | 56.70 | 683.72 |
| | | Naïve | 23.95 | 67.34 | 720.10 |
| | (5, 10) | Ours | 26.45 | 64.69 | 683.72 |
| | | Naïve | 30.95 | 74.00 | 720.10 |
| | (5, 20) | Ours | 31.82 | 69.73 | 683.72 |
| | | Naïve | 37.08 | 80.06 | 720.10 |

**Table 6: Datasets metadata and the Alioth-based overhead. $(H, C, N)$ represents # samples, categorical, and numerical features. Time is in seconds and communication is in MB.**

| Dataset | $(H, C, N)$ | $\mathbb{VP}$ | | | $\mathbb{HP}$ | | |
|---|---|---|---|---|---|---|---|
| | | LAN | WAN | Comm. | LAN | WAN | Comm. |
| GCD [20] | (800, 13, 7) | 2.20 | 14.60 | 40.58 | 8.45 | 49.24 | 84.91 |
| HCDR [40] | (307,511, 51, 69) | 102.05 | 298.48 | 2351.36 | 530.45 | 1509.80 | 21101.40 |

## 6.4 Application: Feature Selection & LR Train

We demonstrate the applications of Alioth on two tasks: IV-based feature selection and LR training on datasets: German-Credit-Data (GCD) [20] and Home-Credit-Default-Risk (HCDR) [40]. For categorical feature, we set $K$ as the number of its categories. For floating feature, we empirically set $K = 5$ for GCD[3] and $K = 10$ for HCDR[4]. For the integral feature: i) if the number of unique values is $\leq K$, we convert it as a categorical feature, ii) otherwise, we treat it as floating ones.

The metadata of the datasets and their Alioth-based transformation overhead are presented as Table 6. Note that we add fill_value $= 1e^{-4}$ to 0 in Equation 2. It is easy to see that our Alioth is efficient in practical datasets. Concretely, it can complete the transformation of the small-scale dataset GCD within one minute in all cases. Even for dataset HCDR with more than 3 millions samples and 100 features, we can finish the transformation

[3]https://www.kaggle.com/code/mpwolke/creditability-scorecardpy/notebook
[4]https://www.kaggle.com/competitions/home-credit-default-risk/code

**Table 7: Overhead of of IV-based feature selection. Time is in seconds and communication is in MB.**

| Dataset | LAN | WAN | Comm. |
|---|---|---|---|
| GCD [20] | 0.35 | 7.68 | 4.10 |
| HCDR [40] | 12.42 | 26.48 | 162.08 |

**Table 8: Overhead, AUC, and weighted F1 score of 2PC training LR on datasets with/without WoE.**

| Dataset | Framework | LAN | WAN | Comm | AUC ↑ | F1 ↑ |
|---|---|---|---|---|---|---|
| GCD [20] | w./o. WoE | 36.65 | 607.06 | 537.35 | 0.77 | 0.71 |
| | Ours (w./ WoE) | 36.14 | 604.85 | 534.22 | 0.80 | 0.74 |
| HCDR [40] | w./o. WoE | 273.29 | 1284.93 | 8188.05 | 0.51 | 0.41 |
| | Ours (w./ WoE) | 232.43 | 1242.68 | 8096.93 | 0.74 | 0.88 |

in half hour in WAN. Next, we focus here on assessing the task-level performance (excluding the costs of Alioth's overhead) and effectiveness on the secret-shared WoE-transformed datasets.

**IV-based Feature Selection.** Information value (IV) is often used for feature selection. Assuming the feature is with $K$ WoE values $\{w_1, w_2, \ldots, w_K\}$, and $(p_k^+, p_k^-)$ represent the proportion of the positive and negative samples for $w_k$, its IV is computed as $\mathsf{IV} = \sum_{k=1}^{k}(p_k^+ - p_k^-) \cdot w_k$. More details can be referred to [17]. After Alioth-based WoE transformation, the datasets are secret-shared among two parties, so the computation is independent of original data partitions, *a.k.a.*, $\mathbb{VP}$ or $\mathbb{HP}$. Experiments show that we select the same top-5 features as those of plaintext WoE, so Table 7 focuses on the running time and communication. Concretely, we can complete IV-based feature selection securely within *half a minute* across all cases, which is efficient for practical applications.

**LR Training.** We securely train LR on GCD and HCDR by 10 iterations for financial risk assessment, with and without WoE produced by Alioth. Other hyperparameters, initialization, *etc.*, are identical. Specifically, we leverage SecretFlow-SPU [34] with 2PC backend [23, 33]. Without WoE, we apply one-hot encoding for categorical features and keep the original numerical features. For each dataset, 70% of the samples are for training and 30% for testing. We use AUC and F1 score [22, 41] to evaluate model performance, two of the most widely used metrics for binary classification.

Table 8 shows our Alioth-based transformation not only reduces secure training overhead but also improves model performance. In particular, on the large-scale HCDR dataset, Alioth achieves lower training time and communication cost and higher AUC and F1 with the same training iterations. This is because WoE reduces distribution shift between parties and enhances feature stability, which not only increases the AUC from 0.507 to 0.735 and F1 from 0.41 to 0.88, but also accelerates the model convergence. These results indicate that achieving comparable LR performance on datasets without WoE transformation would require substantially higher 2PC training overhead than ours.

In a nutshell, Alioth provides an efficient solution for privacy-preserving feature processing in the ML-pipeline. Although Alioth introduces additional overhead, the transformed datasets can support multiple downstream tasks efficiently, and enable improved secure training efficiency and LR model performance.

## 7 RELATED WORK

Secure multiparty computation (MPC) can enable distrusted parties to compute a function while keeping each input privately [13, 47, 55, 56]. In the line of works [6, 23, 25, 31, 37, 39, 46], many secure machine learning frameworks have been proposed in two-party computation (2PC), by utilizing mixed 2PC technologies, *i.e.*, Homomorphic Encryption and Oblivious Transfer, and use secret sharing to connect them. [21, 29, 33, 43, 53, 58] have achieved secure evaluations of Transformer-based large language models. In addition, there are multi-party works have constructed secure machine learning approaches beyond the 2PC setting, including 3PC [10, 26, 38, 49], 4PC [8, 11, 14, 27], and multi-party settings [12, 30, 57].

Apart from secure machine learning training and inference, some works [16, 28, 32, 50] have proposed MPC-based feature selection to address the privacy concerns. Concretely, [28] designed an efficient feature scoring protocol based on Gini impurity using 3PC and showed effectiveness over real-world datasets. [32] introduced a Gini impurity-based lightweight feature selection system for federated learning, where multiple clients interact with a single server and datasets are horizontally distributed among the clients. Differently, [16] proposed low complexity bounds on mutual information for efficient privacy-preserving discrete feature selection using MPC. [50] employed the approximated fixed-point representation to reduce the bitwidth of the mutual information-based solution, thereby reducing communication overhead. Recent work [48] proposed batch private information retrieval (PIR) and PIR-to-share conversion based on polynomial operations, thereby achieving efficient two-party secure unbalanced data alignment for vertical privacy-preserving ML. However, these works mainly focused on feature selection or data alignment. None provides an efficient solution for privately encoding the categorical and numerical features of raw data to ML-effective numerical representations.

We propose a two-party Weight-of-Evidence–based solution that securely transforms raw categorical and numerical features into ML-effective numerical values, thereby supporting efficient privacy-preserving feature selection and logistic regression training.

## 8 CONCLUSION

We propose a two-party framework Alioth that facilitates transforming raw data features into ML-effective numerical representations based on Weight-of-Evidence. By leveraging data partitions, we propose fast partition-aware 2PC protocols to protect data privacy. We integrate Alioth into the privacy-preserving ML pipeline. For future work, we aim to extend Alioth to the multi-party setting, hardware (*i.e.*, GPU) acceleration, and process missing values or low-frequency categories with privacy preservation.

## REFERENCES

[1] 1996. *The Health Insurance Portability and Accountability Act of 1996 (HIPAA).* https://www.hhs.gov/hipaa/index.html

[2] 2016. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (GDPR).* https://gdpr-info.eu/

[3] 2023. Microsoft SEAL (release 4.1). https://github.com/Microsoft/SEAL.

[4] 2023, Sep.. YACL (Yet Another Common crypto Library). https://github.com/secretflow/yacl.

[5] Mark Blunk, Paul Bunn, Samuel Dittmer, Steve Lu, and Rafail Ostrovsky. 2022. Secure merge in linear time and O (log log N) rounds. *Cryptology ePrint Archive.*

[6] Fabian Boemer, Rosario Cammarota, Daniel Demmler, Thomas Schneider, and Hossein Yalame. 2020. MP2ML: a mixed-protocol machine learning framework for private inference. In *ARES International Conference on Availability, Reliability and Security*. 14:1–14:10.

[7] Fabian Boemer, Sejun Kim, Gelila Seifu, Fillipe DM de Souza, Vinodh Gopal, et al. 2021. Intel HEXL (release 1.2). https://github.com/intel/hexl.

[8] Megha Byali, Harsh Chaudhari, Arpita Patra, and Ajith Suresh. 2020. FLASH: Fast and Robust Framework for Privacy-preserving Machine Learning. *Proc. Priv. Enhancing Technol.* 2020, 2 (2020), 459–480.

[9] Suvradip Chakraborty, Stanislav Peceny, Srinivasan Raghuraman, and Peter Rindal. 2024. Logstar: Efficient Linear* Time Secure Merge. Cryptology ePrint Archive, Paper 2024/159. https://eprint.iacr.org/2024/159

[10] Harsh Chaudhari, Ashish Choudhury, Arpita Patra, and Ajith Suresh. 2019. Astra: High throughput 3pc over rings with application to secure prediction. In *Proceedings of the 2019 ACM SIGSAC Conference on Cloud Computing Security Workshop*. 81–92.

[11] Harsh Chaudhari, Rahul Rachuri, and Ajith Suresh. 2019. Trident: Efficient 4pc framework for privacy preserving machine learning. *arXiv preprint arXiv:1912.02631* (2019).

[12] Ronald Cramer, Ivan Damgård, Daniel Escudero, Peter Scholl, and Chaoping Xing. 2018. SPD$\mathbb{Z}_{2}^{k}$: efficient MPC mod $2^k$ for dishonest majority. In *Annual International Cryptology Conference*. Springer, 769–798.

[13] Ronald Cramer, Ivan Damgård, and Ueli Maurer. 2000. General secure multi-party computation from any linear secret-sharing scheme. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 316–334.

[14] Anders Dalskov, Daniel Escudero, and Marcel Keller. 2021. Fantastic four: Honest-majority four-party secure computation with malicious security. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

[15] Sagar S De, Satchidananda Dehuri, et al. 2012. Machine Learning for Social Network Analysis: A Systematic Literature Review. *IUP Journal of Information Technology* 8, 4 (2012).

[16] David Eklund, Alfonso Iacovazzi, Han Wang, Apostolos Pyrgelis, and Shahid Raza. 2024. BMI: Bounded Mutual Information for Efficient Privacy-Preserving Feature Selection. In *European Symposium on Research in Computer Security*. Springer, 353–373.

[17] Soledad Galli. 2020. *Python feature engineering cookbook*. Vol. 1. Packt Publishing Birmingham.

[18] Soledad Galli. 2021. Feature-engine: A Python package for feature engineering for machine learning. *Journal of Open Source Software* 6, 65 (2021), 3642.

[19] Iftach Haitner, Yuval Ishai, Eyal Kushilevitz, Yehuda Lindell, and Erez Petrank. 2011. Black-box constructions of protocols for secure computation. *SIAM J. Comput.* 40, 2 (2011), 225–266.

[20] Hans Hofmann. 1994. Statlog (German Credit Data). UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5NC77.

[21] Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Wen-jie Lu, Cheng Hong, and Kui Ren. 2023. Ciphergpt: Secure two-party gpt inference. *Cryptology ePrint Archive*.

[22] Jin Huang and Charles X Ling. 2005. Using AUC and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering* 17, 3 (2005), 299–310.

[23] Zhicong Huang, Wen-jie Lu, Cheng Hong, and Jiansheng Ding. 2022. Cheetah: Lean and Fast Secure Two-Party Deep Neural Network Inference. In *USENIX Security*. 809–826.

[24] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. 2003. Extending Oblivious Transfers Efficiently. In *CRYPTO*. 145–161.

[25] Chiraag Juvekar, Vinod Vaikuntanathan, and Anantha Chandrakasan. 2018. {GAZELLE}: A low latency framework for secure neural network inference. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*. 1651–1669.

[26] Nishat Koti, Mahak Pancholi, Arpita Patra, and Ajith Suresh. 2021. {SWIFT}: Super-fast and Robust Privacy-Preserving Machine Learning. In *30th {USENIX} Security Symposium ({USENIX} Security 21)*.

[27] Nishat Koti, Arpita Patra, Rahul Rachuri, and Ajith Suresh. 2021. Tetrad: Actively Secure 4PC for Secure Training and Inference. *arXiv preprint arXiv:2106.02850*.

[28] Xiling Li, Rafael Dowsley, and Martine De Cock. 2021. Privacy-preserving feature selection with secure multiparty computation. In *International Conference on Machine Learning*. PMLR, 6326–6336.

[29] Zhengyi Li, Kang Yang, Jin Tan, Wen-jie Lu, Haoqi Wu, Xiao Wang, Yu Yu, Derun Zhao, Yancheng Zheng, Minyi Guo, et al. 2024. Nimbus: Secure and efficient two-party inference for transformers. *Advances in Neural Information Processing Systems* 37 (2024), 21572–21600.

[30] Fengrun Liu, Xiang Xie, and Yu Yu. 2024. Scalable Multi-Party Computation Protocols for Machine Learning in Honest-Majority Setting. In *USENIX Security*.

[31] Jian Liu, Mika Juuti, Yao Lu, and N. Asokan. 2017. Oblivious Neural Network Predictions via MiniONN Transformations. In *CCS*. 619–631.

[32] Xiaoyuan Liu, Hongwei Li, Guowen Xu, Xilin Zhang, Tianwei Zhang, and Jianying Zhou. 2024. Secure and lightweight feature selection for horizontal federated learning. *IEEE Transactions on Information Forensics and Security* (2024).

[33] Wen-jie Lu, Zhicong Huang, Zhen Gu, Jingyu Li, Jian Liu, Kui Ren, Cheng Hong, Tao Wei, and Wenguang Chen. 2023. BumbleBee: Secure Two-party Inference Framework for Large Transformers. In *NDSS*.

[34] Junming Ma, Yancheng Zheng, Jun Feng, Derun Zhao, Haoqi Wu, Jin Tan, Chaofan Yu, Benyu Zhang, and Lei Wang. 2023. SecretFlow-SPU: A Performant and User-Friendly Framework for Privacy-Preserving Machine Learning. In *USENIX ATC*. 17–33.

[35] Akib Mashrur, Wei Luo, Nayyar A Zaidi, and Antonio Robles-Kelly. 2020. Machine learning for financial risk management: a survey. *Ieee Access* 8 (2020).

[36] Charles Masson, Jee E Rim, and Homin K Lee. 2019. Ddsketch: A fast and fully-mergeable quantile sketch with relative-error guarantees. *arXiv preprint arXiv:1908.10693* (2019).

[37] Pratyush Mishra, Ryan Lehmkuhl, Akshayaram Srinivasan, Wenting Zheng, and Raluca Ada Popa. 2020. Delphi: A Cryptographic Inference Service for Neural Networks. In *USENIX Security*. 2505–2522.

[38] Payman Mohassel and Peter Rindal. 2018. ABY3: A mixed protocol framework for machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 35–52.

[39] Payman Mohassel and Yupeng Zhang. 2017. SecureML: A System for Scalable Privacy-Preserving Machine Learning. In *IEEE S&P*. 19–38.

[40] Anna Montoya, inversion, KirillOdintsov, and Martin Kotek. 2018. Home Credit Default Risk. https://kaggle.com/competitions/home-credit-default-risk. Kaggle.

[41] Sarang Narkhede. 2018. Understanding auc-roc curve. *Towards data science* 26, 1 (2018), 220–227.

[42] Nebojsa Nikolic, Nevenka Zarkic-Joksimovic, Djordje Stojanovski, and Iva Joksimovic. 2013. The application of brute force logistic regression to corporate credit scoring models: Evidence from Serbian financial statements. *Expert systems with applications* 40, 15 (2013), 5932–5944.

[43] Qi Pang, Jinhao Zhu, Helen Möllering, Wenting Zheng, and Thomas Schneider. 2023. BOLT: Privacy-Preserving, Accurate and Efficient Inference for Transformers. *In IEEE S&P* (2023), 1893.

[44] Christos Polykretis and Christos Chalkias. 2018. Comparison and evaluation of landslide susceptibility maps obtained from weight of evidence, logistic regression, and artificial neural network models. *Natural hazards* 93, 1 (2018), 249–274.

[45] Amir Masoud Rahmani, Efat Yousefpoor, Mohammad Sadegh Yousefpoor, Zahid Mehmood, Amir Haider, Mehdi Hosseinzadeh, and Rizwan Ali Naqvi. 2021. Machine learning (ML) in medicine: review, applications, and challenges. *Mathematics* 9, 22 (2021), 2970.

[46] Deevashwer Rathee, Mayank Rathee, Nishant Kumar, Nishanth Chandran, Divya Gupta, Aseem Rastogi, and Rahul Sharma. 2020. CrypTFlow2: Practical 2-Party Secure Inference. In *CCS*. 325–342.

[47] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.

[48] Lushan Song, Qizhi Zhang, Yu Lin, Haoyu Niu, Daode Zhang, Zheng Qu, Weili Han, Jue Hong, Quanwei Cai, and Ye Wu. 2025. Suda: An Efficient and Secure Unbalanced Data Alignment Framework for Vertical Privacy-Preserving Machine Learning. In *USENIX Security*.

[49] Sameer Wagh, Divya Gupta, and Nishanth Chandran. 2019. SecureNN: 3-Party Secure Computation for Neural Network Training. *Proc. Priv. Enhancing Technol.* 3 (2019), 26–49.

[50] Luyao Wang, Hao Guo, Weibin Wu, and Lu Zhou. 2025. Efficient and privacy-preserving feature selection based on multiparty computation. *IEEE Transactions on Information Forensics and Security* (2025).

[51] Zhouxia Wang, Tianshui Chen, Jimmy Ren, Weihao Yu, Hui Cheng, and Liang Lin. 2018. Deep reasoning with knowledge graph for social relationship understanding. *arXiv preprint arXiv:1807.00504* (2018).

[52] Douglas L Weed. 2005. Weight of evidence: a review of concept and methods. *Risk Analysis: An International Journal* 25, 6 (2005), 1545–1557.

[53] Tianshi Xu, Wen-jie Lu, Jiangrui Yu, Yi Chen, Chenqi Lin, Runsheng Wang, and Meng Li. 2025. Breaking the Layer Barrier: Remodeling Private Transformer Inference with Hybrid {CKKS} and {MPC}. In *34th USENIX Security Symposium (USENIX Security 25)*. 2653–2672.

[54] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. 2020. Ferret: Fast Extension for Correlated OT with Small Communication. In *CCS*. 1607–1626.

[55] Andrew C Yao. 1982. Protocols for secure computations. In *23rd annual symposium on foundations of computer science (sfcs 1982)*. IEEE, 160–164.

[56] Andrew Chi-Chih Yao. 1986. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*. IEEE, 162–167.

[57] Boshi Yuan, Shixuan Yang, Yongxiang Zhang, Ning Ding, Dawu Gu, and Shi-Feng Sun. 2024. {MD-ML}: Super Fast {Privacy-Preserving} Machine Learning for Malicious Security with a Dishonest Majority. In *33rd USENIX Security Symposium (USENIX Security 24)*. 2227–2244.

[58] Jiawen Zhang, Xinpeng Yang, Lipeng He, Kejia Chen, Wen-jie Lu, Yinghao Wang, Xiaoyang Hou, Jian Liu, Kui Ren, and Xiaohu Yang. 2025. Secure Transformer Inference Made Non-interactive. In *NDSS*.