Response to Reviewer Comments

We appreciate the reviewers for their valuable comments and address them accordingly. We have grouped comments related to similar topics together. To make it easier to identify the major changes in the main paper, we have highlighted them in blue.

- **Differentiate DP definitions, trust model, and DP mechanism: FDP is a DP mechanism, not DP definition (R1D1). FDP acronym has been used for DP definitions (R2D2). Better clarify the difference between GDP and LDP in Section 2.2 (R1D3). Privacy requirements section of Section 2.3 should be renamed to trust model (R1D4). Add another figure in Section 2.3 to illustrate the trust model (R1D5). In Section 3.4, rename each baseline as DP mechanism (R1D6). Since FDP also satisfies GDP, in Section 3.4, state the given example of GDP as a naive implementation of GDP (R1D7).**
  In Section 1, Section 2.2, Section 2.3, Section 3.4, Section 5.1, and throughout the rest of the text, we have distinguished between DP definitions, trust models, and DP mechanisms. We have clarified that GDP and LDP are different DP definitions in Section 2.2 and emphasized that LDP can function under a weaker trust model where the aggregator is not trusted (**R1D3**). We have described the four different types of trust models in Section 2.3 (**R1D4**): *Saibot*'s trust model with Figure 3 for illustration (**R1D5**), global model where the 2^{nd}-level aggregator is trusted, local model where no aggregator is trusted, and the shuffle model where shufflers (could be 1^{st}- or 2^{nd}-level) are trusted. We have renamed FDP to Factorized Privacy Mechanism (FPM) (**R1D2**) and clarified that it is a DP mechanism, not a DP definition in Section 1 (**R1D1**). For other mechanisms, we have renamed them to Aggregate Privacy Mechanism (for global model), Per-tuple Privacy Mechanism (for local model), and Shuffling (for shuffle model) to differentiate them from DP definitions and trust models in Section 3.4 and Section 5.1 (**R1D6**). We have clarified in Section 3.4 that the Aggregate Privacy Mechanism is a naive implementation of GDP, and in Section 1 and Section 2.3, we have clarified that FPM satisfies GDP, where the randomized algorithm is applied by the 1^{st}-level aggregators (i.e., providers, requesters) instead of by the 2^{nd}-level ones (**R1D7**).
- **To achieve privacy benefits, the final model training must be differentially private. (R1D8)**
  We have clarified in Section 1 and Section 3.5 that *Saibot* focuses on data search using differentially private proxy models (linear regression). Although the private proxy models can be directly returned, requesters often seek more complex models for model training. To accomplish this and meet the privacy requirements, *Saibot* can be integrated within a larger differentially private federated ML system. In this setup, *Saibot* first identifies augmentations, and then federated ML systems like [61, 64, 66, 72] utilize these augmentations to train more advanced models, such as deep neural networks through differentially private gradient descent.
- **Discuss the support for categorical features (R2D1).**
  Saibot employs linear regression as the proxy model, which primarily supports numerical features. However, as we clarify in Section 3.3, categorical attributes can be handled through preprocessing. In Section 3.5, we explain that low cardinality categorical features can be preprocessed using one-hot encoding, converting them into numerical features for *Saibot*. On the other hand, high cardinality categorical features pose a challenge when one-hot encoded, resulting in high-dimensional feature vectors [18, 48]. This is exactly the problem *Saibot* can address through augmentation. By joining with augmentations on high cardinality categorical features, requesters can transform these features into meaningful, lower-dimensional numerical features from the augmentations. Therefore, we recommend requesters use high cardinality categorical features as join keys.
- **Discuss the integration of *Saibot* with existing data augmentation method like ARDA (R2D2).**
  ARDA primarily uses random forests as the proxy model, and we discuss the challenges of applying DP to factorized random forests in Section 6. Unlike linear regression, which has a closed-form solution with a single semi-ring aggregate as sufficient statistics, tree-based models like random forests require additional specialized aggregates. For instance, we need to compute aggregates with additional selections based on node predicates for each leaf node. These aggregates are not reusable and are expensive to privatize.

  To improve data search utility to a level comparable to non-private methods and address these challenges, we plan to explore the following in future work: (1) alternative trust model designs where requesters trust the search platform to reduce noises, and (2) use monomial semi-ring aggregates to create differentially private synopses for synthetic data generation. The generated synthetic data can then be used to train arbitrarily complex models as post-processing.
- **Clarifications for Algorithm 1, specifically lines 2 and 7 (R2D3), and explain if it involves anything more complex beyond the standard Gaussian mechanism (R3D5)**
  We have clarified in Section 3.3 that FPM straightforwardly applies the Gaussian mechanism to factorized monomials without additional complications (**R3D5**). We have clarified that lines 2 and 7 show the optimized factorized monomial sensitivity based on the parity of the statistics order (**R2D3**). The sensitivity of even-order monomials can be reduced by a factor of $\sqrt{2}$. Furthermore, if there is only one feature, the sensitivity of even-order monomials can be further decreased by an additional factor of $\sqrt{2}$ (see the proof of Theorem 5).
- **Applying DP alone does not permit sharing HIPAA data in Example 1 (R3D1).**
  We have refocused Example 1 on Fitbit [51], a commercial wearable company that is not a covered health entity and is not subject to HIPAA law. Nevertheless, Fitbit gathers, maintains, and shares user-sensitive health data to enhance recommendation models and adheres to privacy protection for sensitive health information through its internal policies. As a result, Fitbit could employ *Saibot* to enhance its recommendation models while preserving privacy in a differentially private manner, upon obtaining consent from individuals.
- **Clarify the relationship of Section 3.2 and Section 3.3 with DP (R3D2).**
  We have clarified that Section 3.2 focuses on the semi-ring design of monomials to support ML over join and union operations without DP. This is the first work to bridge the connection between semi-rings in factorized ML for training over joins and unions, with privatized

sufficient statistics in the differentially private ML literature. We are also the first to explicitly extend semi-ring from the gram matrix to higher order monomials (that support e.g., generalized linear models). Then, in Section 3.3, we introduce the DP mechanism (FPM) to privatize these monomials for differentially private data search.

We apologize for the earlier confusion due to our use of the term FDP, which sounded like a new DP definition that requires more explanation on privacy guarantees. In actuality, FDP is a DP mechanism that satisfies GDP, but operates under a different trust model where only the direct 1^{st}-level aggregator (requester or provider) is trusted (not the 2^{nd}-level data platform as the centralized curator) (Section 2.3). As a result, FDP is applied by the direct 1^{st}-level aggregator not the 2^{nd}-level one (Section 3.3). To avoid confusion, we have renamed FDP to Factorized Privacy Mechanism (FPM) and distinguished DP mechanisms from DP definitions through the text (please see our responses to R1D1-R1D7 above). We have also articulated the significance of FPM design in Section 3.3: The primary algorithmic challenge FPM addresses is designing sufficient statistics that are composable (through semi-ring operators) and reusable (as post-processing without additional privacy cost) to support ML across various join and union augmentations for task-based data search.

- **Clarify the advantage of FPM over naive mechanism for GDP (R3D3).**
  Your understanding is correct that the advantage of FPM over APM (the naive mechanism for GDP) is that FPM only privatizes the aggregated monomials for each dataset and reuses them for join/union through semi-ring operators, while the APM has to privatize monomials for all join/union combinations. Additionally, FPM doesn't require a trusted 2^{nd}-level aggregator to join/union datasets, whereas APM does. We have emphasized these benefits in Section 3.4.
- **Better to call the work as a new algorithm (R3D4).**
  While *Saibot* as a platform comprises various crucial components for differentially private data search, We agree that our core contribution is FPM, which is introduced in the context of data search as it is our main motivating use case. In Section 1, we have emphasized that the main contribution of this work is FPM.
- **Provide more examples to facilitate understanding Section 3.2 (R3D6)**
  To illustrate the statistics based on k-order monomial in Definition 2, we have provided Example 3, which computes statistics for 1, 2, 3-order monomials using an example relation. To provide the intuition of how monomial semi-ring (Section 3.2) works, we've refined Example 2 and highlighted that this is a case of training linear regression using a 2-order monomial semi-ring. For comparison, we include a naive linear regression training example (Figure 5) to highlight the benefits of monomial semi-ring. While naive linear regression training computes aggregates after the materialization of costly joins and unions, factorized linear regression uses a semi-ring structure to push down aggregations (Figure 4). This results in identical aggregates and models but removes the large materialization overhead. We stress at the end of Section 3.2 that aggregated monomials (e.g., $\gamma(R_1), \gamma(R_2), \gamma(R_3)$ in Example 2) are ideal intermediate representations for DP due to their reusability. The FPM in Section 3.3 exploits this to ensure differential private data search while maintaining high utility.

Saibot: A Differentially Private Data Search Platform

Zezhou Huang
zh2408@columbia.edu
Columbia University

Jiaxiang Liu
jl6235@columbia.edu
Columbia University

Daniel Gbenga Alabi
alabid@cs.columbia.edu
Columbia University

Raul Castro Fernandez
raulcf@uchicago.edu
University of Chicago

Eugene Wu
ewu@cs.columbia.edu
DSI, Columbia University

ABSTRACT

Recent data search platforms use ML task-based utility measures rather than metadata-based keywords, to search large dataset corpora. Requesters submit a training dataset, and these platforms search for *augmentations*—join or union-compatible datasets—that, when used to augment the requester's dataset, most improve model (e.g., linear regression) performance. Although effective, providers that manage personally identifiable data demand differential privacy (DP) guarantees before granting these platforms data access. Unfortunately, making data search differentially private is nontrivial, as a single search can involve training and evaluating datasets hundreds or thousands of times, quickly depleting privacy budgets.

We present *Saibot*, a differentially private data search platform that employs Factorized Privacy Mechanism (FPM), a novel DP mechanism, to calculate sufficient semi-ring statistics for ML over different combinations of datasets. These statistics are privatized once, and can be freely reused for the search. This allows *Saibot* to scale to arbitrary numbers of datasets and requests, while minimizing the amount that DP noise affects search results. We optimize the sensitivity of FPM for common augmentation operations, and analyze its properties with respect to linear regression. Specifically, we develop an unbiased estimator for many-to-many joins, prove its bounds, and develop an optimization to redistribute DP noise to minimize the impact on the model. Our evaluation on a real-world dataset corpus of 329 datasets demonstrates that *Saibot* can return augmentations that achieve model accuracy within 50–90% of non-private search, while the leading alternative DP mechanisms (TPM, APM, shuffling) are several orders of magnitude worse.

1 INTRODUCTION

Augmenting training data with additional samples or features can significantly enhance ML performance [55]. However, sourcing such data in large corpora—public portals [10, 11], or enterprise data warehouses—is a complex task. To address this, a new form of data search platform [20, 37, 46, 49, 56] is emerging, wherein a requester submits a search request comprising training and testing datasets for augmentation. The platform then finds provider datasets that augment the training dataset in a way that improves *utility* (e.g., ML performance). This involves using a data discovery tool [17, 30] to locate a set of union- or join-compatible tables (*augmentations*), augmenting the training set with each candidate, and then retraining and evaluating the model to assess its *utility*. The augmentations are subsequently ranked by *utility*. Platforms largely differ in the discovery tool procedure, the models they support, and how they accelerate model retraining and evaluation. Recent works [19, 37, 56] suggest that using linear regression as a model proxy provides a good balance of search quality and runtime.

Unfortunately, privacy is a major barrier to sharing for many potential data providers and requesters with sensitive data (e.g., personally identifiable information (PII), and protected health information (PHI)). In these cases, providers are legally obligated to prevent personal data leakage [6, 7, 9]. Rather than prohibit access outright, differential privacy (DP) [23] supports data analysis on sensitive data while bounding the degree of privacy loss based on the budget ϵ set by the data provider. Each query on the dataset adds noise to the results, inversely proportional to the budget consumed; when $\epsilon = 0$, the dataset becomes inaccessible.

Ideally, a differentially private data search platform would let providers and requesters set privacy budgets for their datasets, and enforce these budgets as new datasets and requests arrive. Moreover, since the platform is often a third-party service that may not be trusted by data providers (and the individuals they collect data from), it should not have access to raw data. Unfortunately, integrating DP with data search platforms is non-trivial. To illustrate, Figure 1 shows where existing mechanisms would add noise in a two-level data-sharing architecture that matches many real-world settings. In this architecture, individuals (e.g., patients) generate sensitive data aggregated by providers/requesters (e.g., hospitals), and the search platform further aggregates their datasets.

Global DP (GDP) is a DP definition widely used by private DBMSes [39, 43, 67], where the employed mechanisms add noise after executing, e.g., a query over private data by a trusted central DBMS. However, when applied to data search, previous GDP mechanisms need to "split the budget" across every candidate augmentation on every request. The budget ends up being so small that the noise drowns any signal in the data. Further, their trust model requires the search platform, acting as the central aggregator, to be trusted, which is challenging since it is a third-party service. To address this, mechanisms for Local DP (LDP) (e.g., randomized response [21, 28]) eliminate the need for a trusted data curator by privatizing individual tuples. Nevertheless, the noise required for these mechanisms can be quite large, potentially compromising data utility [66]. Shuffling [27, 29] is a mechanism for an intermediate trust model that, instead of relying on a trusted central aggregator, requires trust in a shuffler. After privatizing tuples (using mechanisms for LDP), the shuffler shuffles the primary keys of tuples during aggregation to disassociate them from individuals; this "amplifies privacy" by allowing each tuple to have less noise applied. Variation Shuffle-1 shuffles at the provider/requester level but requires considerable noise for small datasets; Shuffle-2 shuffles within the search platform but needs to trust the platform. An alternative to shuffling,

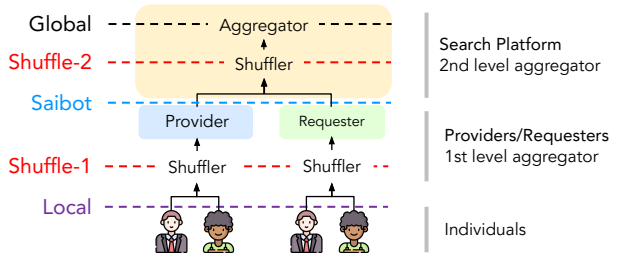


Figure 1: Summary of DP-mechanisms under various trust models in a standard data sharing architecture: Providers/users collect data from individuals, and the search platform aggregates data from providers/users. At the extremes, mechanisms for local models introduce noise to individual tuples, whereas naive mechanisms for global models add noise to query results through a trusted 2^{nd}-level aggregator. Mechanisms for the shuffle model introduce a shuffler at a point of aggregation (either at the 1^{st}-level (`Shuffle-1`) or the 2^{nd}-level (`Shuffle-2`)). In contrast, *Saibot* adds noise to sufficient statistics computed by providers/users.

widely used by federated ML [59, 61, 66, 72], is to let providers/requesters iteratively compute and privatize model gradients locally, and let an untrusted aggregator compute the final model. However, these gradients are specific to a single augmentation's model, so the budget is still split across all candidate augmentations.

Is it possible for a DP search platform to return search results of comparable quality to non-private search, *and* for the platform to scale to many datasets and requests? We are motivated by the recent data search platform Kitana [37], which uses semi-ring aggregation to quickly evaluate a candidate augmentation's *utility* on a linear regression model without materializing the augmented table and fully retraining it. These semi-rings can be computed for each dataset offline, and Kitana only needs these semi-rings to evaluate a candidate augmentation in $\approx$1–5ms, independent of the dataset size. *Our main observation is that these precomputed semi-rings also serve as ideal intermediates for DP, as they help directly estimate model parameters, can be combined over joins and unions, and can be freely reused once made private.*

This paper presents *Saibot*, a differentially private data search platform for tabular datasets that scales to unlimited datasets and requests, returns results comparable to non-private search, and doesn't need to be trusted. Data providers upload their privatized datasets to the platform. When a requester submits a privatized training dataset, the platform searches for the best combinations of privatized datasets which, when augmented with the requester's dataset, most improve the accuracy of a linear regression model. For the trust model, *Saibot* assumes that the 1^{st}-level aggregators are trusted (unlike the local model) but the 2^{nd}-level aggregators (i.e., search platform) are not (unlike global model). In practice, regulations [6–9] mandate that the 1^{st}-level aggregators (e.g., healthcare providers, schools) securely store individual data. Once *Saibot* identifies predictive augmentations using differentially private proxy models (linear regression), it can directly return the private proxy models, although they may not be complex enough for some requesters. To address this, *Saibot* can be integrated within a larger differentially private federated ML system [61, 64, 66, 72] to train more advanced models, like deep neural networks through differentially private gradient descent, on the identified augmentations.

Our key innovation is a new DP mechanism called *Factorized Privacy Mechanism* (FPM), where each requester or provider computes and privatizes sufficient statistics on their own datasets based on their privacy requirements. These sufficient statistics provide high utility, can be freely reused for ML over different augmentations, and only require the search platform to store privatized datasets. FPM satisfies GDP, but the randomized algorithm is applied by the 1^{st}-level aggregators rather than the 2^{nd}-level ones. Note that FPM has broader applications, not only for the data search but also for more general differentially private factorized learning.

The main algorithmic challenge FPM solves is to design privatized sufficient statistics for ML that are composable to support various join and union augmentations. Previous works have applied DP to sufficient statistics for privatized linear regression [65] and GLM [38, 44], but these sufficient statistics can be used for only a single dataset. Our key insight is to design these sufficient statistics as a semi-ring [32], which includes addition and multiplication operators for union and join. Although sufficient semi-ring statistics have been utilized for ML [57, 58] over joins, we are the first to explore their application in a DP setting. The results of our real-world experiments indicate that FPM is capable of identifying augmentations that achieve an average $r2$ score of ~50–90% compared to non-private searches. Additionally, FPM can support a large data corpus and unlimited requests. In contrast, the other baseline mechanisms achieve $r2$ scores <0.02.

To summarize, our contributions are as follows:

- We propose FPM, a novel DP mechanism that privatizes reusable and composable monomials for join/aggregation augmentations. We integrate FPM into *Saibot* to achieve scalability for large volumes of datasets and search requests with high utility.
- We optimize FPM based on the parity of the statistics order. For the special case of tables containing a single feature, we reduce the expected error by a further factor of $\sqrt{2}$.
- We provide a deep analysis of FPM to linear regression models. Specifically, we study the statistical bias introduced in many-to-many joins, and design an unbiased estimator to address this. We further study its bounds on errors over the model parameters.
- We design an optimization that carefully redistributes noise across sufficient statistics to improve linear regression accuracy.
- We thoroughly evaluate FPM across a real-world data corpus with >300 datasets. Our results show that FPM can accurately identify augmentations that achieve $r2$ scores close to (~50–90%) those of a non-private search. We further use ablation studies to validate our theoretical analyses and study the sensitivity.

Note: The paper is self-contained. References to appendices can be disregarded or located in the technical report [12].

2 PRIVATE TASK-BASED DATA SEARCH

In this section, we formalize the problem of task-based private data search. We start with an introduction of the non-private problem and current solutions. We then provide the primer of differential privacy, and present the differentially private data search problem.

2.1 Non-Private Task-based Data Search

We provide the background of previous task-based data search problem [20, 46, 49, 56], which is non-private, and previous solutions.

Data Model. We follow the standard relational data model. Relations are denoted as R, attributes as A, and domains as $dom(A)$. R's schema is represented by $S_R = [A_1, \cdots, A_n]$, with tuples labeled as t and attribute values as $t[A]$. For clarity, the schema is included in square brackets following the relation in examples $R[A_1, \cdots, A_n]$. The domain of a relation is the Cartesian product of attribute domains: $dom(R) = dom(A_1) \times \cdots \times dom(A_n)$. We consider each dataset as a relational table and use these terms interchangeably.

Machine Learning. A ML task M, like linear or logistic regression, aims to fit a good model based on feature-target attribute pairs. A training dataset R_{train} comprises features $X \subset S_R$ and a target attribute $y \in S_R$. The task M has a training function $M.Train(\cdot)$ that inputs R_{train} and outputs a model m that optimally predicts y from X, even for unseen X, y pairs. To assess m, M uses a function $M.Evaluate(\cdot)$ which inputs m and a testing dataset R_{test}, and outputs the model's performance on R_{test}, typically measured by accuracy, which is to be maximized.

Task-based Data Search. Given a data corpus with datasets from different providers, requesters send a request with datasets to augment and a task (e.g., ML). Task-based data search aims to identify a set of augmentable (join/union) datasets that maximize task utility.

To formalize this, let $\mathcal{R} = \{R_1, R_2, ...\}$ be a data corpus with a set of relations, with each from some provider. Requester sends a request with training and testing dataset (R_{train}, R_{test}), and chooses a model M. Requester's goal is to train model M on R_{train} and maximize its performance on R_{test}, which we call the task's *utility*.

To improve the *utility*, the requester aims to find a set of provider datasets in $\mathcal{R}$ that can be used to augment their data and enhance model performance. The function $Discover(R, augType)$ is used to find datasets in the data corpus $\mathcal{R}$ that can be joined or unioned with R, given $augType \in \{\bowtie, \cup\}$. The requester wants to try different combinations of subsets of these datasets to augment[1] and find the combination that maximizes *utility*.

Putting everything together, the problem can be formulated as:

Problem 1 (Task-Based Data Search.). *For request (R_{train}, R_{test}, M), find the set of datasets $\mathbf{R}^*_\cup, \mathbf{R}^*_\bowtie \subseteq \mathcal{R}$ from data corpus such that*

$$\begin{aligned} \mathbf{R}^*_\cup, \mathbf{R}^*_\bowtie = \quad & \underset{\mathbf{R}_\cup, \mathbf{R}_\bowtie}{\operatorname{argmax}} M.Evaluate(m, R_{testAug}) \\ s.t. \quad & \mathbf{R}_\cup \subseteq Discover(R, \cup), \mathbf{R}_\bowtie \subseteq Discover(R, \bowtie), \\ & R_{trainAug} = (R_{train} \cup_{R_1 \in \mathbf{R}_\cup} R_1) \bowtie_{R_2 \in \mathbf{R}_\bowtie} R_2 \\ & R_{testAug} = R_{test} \bowtie_{R \in \mathbf{R}_\bowtie} R \\ & m = M.Train(R_{trainAug}) \end{aligned}$$

Solutions. Current task-based data search platforms [20, 46, 49, 56] follow the architecture illustrated in black in Figure 2. Offline, when providers upload raw datasets to *Data storage*, the platform computes minhashes for data discovery [17, 30], and sketches to accelerate retraining [20, 37, 49, 56]. Online, the platform solves Problem 1 for each request (R_{train}, R_{test}, M). First, *data discovery* [17, 30] uses the minhashes or sketches to return a set of candidate datasets. *Data search* then identifies a subset that maximizes task utility. The brute-force search evaluates all possible combinations and can be expensive due to retraining costs and the large set of combinations, so approaches use various heuristics and greedy algorithms [20, 46, 56].

Our work primarily builds on Kitana [37], which follows the architecture in Figure 2 and uses specialized sketches for factorized ML. factorized ML trains models over joins without materializing them, which speeds up model retraining and evaluation after any candidate augmentation. This allows Kitana to execute task-based searches much faster, while maintaining competitive task utility. Our insight is that these sketches boost performance and act as the ideal sufficient statistics for DP, as detailed in Section 3.2.

[1]For simplicity, we consider datasets that can be directly joined or unioned with requester R_{train}. The search space could be further expanded by, e.g., 1^{st} joining provider datasets; our solution can be easily adapted to this larger search space.

2.2 Differential Privacy Primer

Before delving into our solution to differentially private dataset search, we first introduce differential privacy (DP). We focus on the Gaussian mechanism, a common, straightforward technique offering comparable performance and guarantee with other baselines (e.g., it offers the same approximate DP by shuffling [27]). In practice, our solution can also support pure DP by Laplace mechanism (Section 5.2), where shuffling falls short.

Differential Privacy. DP [23] is a technique used to protect reconstruction, membership, and inference attacks [25] by bounding the information leakage from individual records. DP guarantees that the probability that an algorithm will produce the same output on two datasets that differ by only one record is bounded. Formally:

Definition 1 ($(\epsilon, \delta) - DP$). *Let f be a randomized algorithm that takes a relation R as input. f is $(\epsilon, \delta) - DP$ if, for all relations R_1, R_2 that differ by adding or removing a row, and for every set S of outputs from f, the following holds: $Pr[f(R_1) \in S] \leq e^\epsilon Pr[f(R_2) \in S] + \delta$, where ϵ and δ are non-negative real numbers (called privacy budget). ϵ controls the level of privacy, and δ controls the level of approximation. For the special case when $\delta = 0$, $(\epsilon, 0) - DP$ is also called pure DP.*

DP definitions can be global (GDP) or local (LDP) depending on inputs: GDP applies to randomized algorithms that process an entire relation (as an aggregator) described above. In contrast, LDP guarantees the differential privacy of algorithms on individual tuples (or relations with a cardinality of 1) before transmitting tuples to any aggregator. As a result, LDP can function under a weaker trust model, where no aggregator is trusted. However, this often leads to increased noise levels and reduced data utility [70].

There are three important theorems of DP:

Theorem 1 (Robustness to Post-Processing). *Let f be a randomized algorithm that provides $(\epsilon, \delta) - DP$. Let g be an arbitrary function. Then, the composition $g \circ f$ provides $(\epsilon, \delta) - DP$.*

Theorem 2 (Sequential Composition). *Let $f_1, \ldots, f_n$ be a sequence of independent algorithms that provide $(\epsilon_1, \delta_1), \ldots, (\epsilon_n, \delta_n) - DP$, respectively. Then, the algorithm that applies each of them in sequence, i.e., $f_n \circ f_{n-1} \cdots \circ f_1$, is $(\sum_{i=1}^n \epsilon_i, \sum_{i=1}^n \delta_i) - DP$.*

Theorem 3 (Parallel Composition). *Let $dom_1, \ldots, dom_n$ be n disjoint subsets of $dom(R)$. Let $f_1, \ldots, f_n$ be a set of independent algorithms that provide $(\epsilon_1, \delta_1), \ldots, (\epsilon_n, \delta_n) - DP$ and take relations from $dom_1, \ldots, dom_n$ as input, respectively. Then, the algorithm that applies them on disjoint subsets of R is $(\max_{i=1}^n \epsilon_i, \max_{i=1}^n \delta_i) - DP$.*

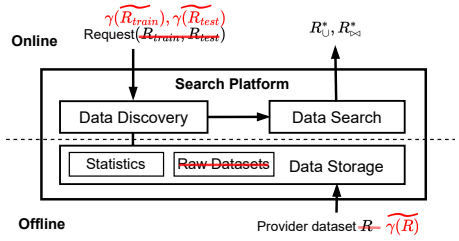


Figure 2: *Saibot* architecture. Previous data search platforms (black) store raw datasets in data storage, use data discovery to identify augmentable datasets, and search the datasets for task improvement. To ensure privacy, *Saibot* additionally applies FPM (red) to compute sufficient semi-ring statistics, that are aggregated (γ) and privatized ($\sim$) before being sent to the search platform. These statistics can support join and union queries to train and evaluate ML as post-processing.

To ensure $(\epsilon, \delta) - DP$ when Q queries need to be executed, the privacy budget (ϵ, δ) can be split among the queries using sequential composition, such as allocating $(\epsilon/Q, \delta/Q)$ for each query. This work employs (basic) sequential composition for simplicity, but it could be further optimized by advanced composition [24].

Gaussian Mechanism. The Gaussian mechanism [22] adds noise to a query function to satisfy (ϵ, δ)-differential privacy. Formally:

Theorem 4 (Gaussian Mechanism.). *Given $\epsilon, \delta \in (0, 1]$, let query q be a function that takes R as input and outputs a vector of real numbers. The Gaussian mechanism independently adds random noise to each output to satisfy (ϵ, δ)-differential privacy: $q'(R) = q(R) + \mathcal{N}(0, \sigma^2)$, where $\mathcal{N}(0, \sigma^2)$ denotes a Gaussian distribution with mean 0 and standard deviation $\sigma = \sqrt{2\ln(1.25/\delta)}\Delta_q/\epsilon$. Δ_q is the l_2-sensitivity of q defined as: for all possible neighbouring relations R_1, R_2, Δ_q is the maximum ℓ_2 distance of q outputs $\|q(R_1) - q(R_2)\|_2$.*

Different definitions exist for neighbouring relations (and can be extended to multi-relations). We adopt bounded DP [15], where neighbouring relations R_1, R_2 have identical row numbers, but one row's data differ; our system can be readily adapted for other definitions (e.g., unbounded DP where row numbers differ).

2.3 Private Task-based Data Search

We first lay out the privacy requirements based on the criteria (Section 1) and motivated by real-world use cases. Then, we define the differentially private data search problem, and discuss the challenges and the intuition for solutions.

Trust Model. We adopt a standard two-level aggregator setting illustrated in Figure 3: the 1^{st}-level aggregators are providers/requesters (e.g., hospitals, schools), and the 2^{nd}-level aggregator is the search platform. Individuals share data with their direct 1^{st}-level aggregator, who is trusted (e.g., a hospital collects data from patients and stores them securely). However, they don't trust other non-direct 1^{st}-level aggregators or the 2^{nd}-level aggregators (e.g., patients don't trust other hospitals and the search platform).

Our trust model sits between the global model (by GDP) and local model (by LDP): Previous global model [39, 43, 67] assumes that the central data curator (2^{nd}-level aggregator) is trusted. On the contrary, the local model assumes no trusted aggregators. In contrast to the shuffle model [27, 29] which requires a trusted shuffler at either 1^{st}-(Shuffle-1, similar to ours) or 2^{nd}-level (Shuffle-2, similar to the global model), we don't rely on any trusted shuffler. In practice, we believe our trust model fits the structure of many organizations, where individuals solely trust their immediate data aggregator (like a hospital or service provider), but do not trust any other aggregators. Further, regulatory requirements [6, 8] place privacy protection requirements on the 1^{st}-level aggregator.

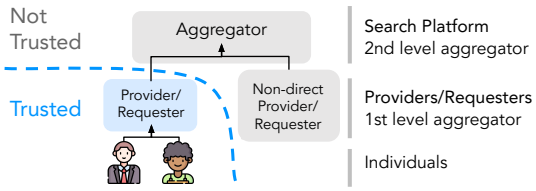


Figure 3: Illustration of *Saibot* trust model, where individuals only trust the direct 1^{st}-level aggregator, and not any others.

Privacy Requirement. Providers and requesters hope to disclose datasets to the malicious search platform for augmentation. Each provider or requester sets a DP budget (ϵ, δ) for each of their datasets, which is independent of other datasets and the search platform. As per previous works [50, 68], we assume that each individual contributes to exactly one row of one dataset. In line with prior studies [39, 43, 67], we assume that the schemas and the domains of join keys (as group-by attributes) are public.

The **differentially private task-based data search** problem is then defined as Problem 1, adhering to the above trust model and satisfying the privacy requirements.

Example 1. *Fitbit [51], a mobile health app, gathers health data from individuals and is trusted by individuals to handle sensitive information responsibly. To enhance the accuracy of its ML recommender, Fitbit plans to share data with a search platform (as requesters) but also wants to protect sensitive health data. Upon obtaining consent from individuals, Fitbit employs DP to privatize each dataset and uses Saibot to search for valuable augmentations.*

Private task-based data search is particularly challenging because, even for a single request, it requires model retraining over a combinatorially large space of augmented datasets created by joining and unioning candidate datasets. How to avoid exhausting the requester's and the providers' privacy budgets? How can massive datasets and requests be scaled without degrading search quality? Is there a one-time differentially private, yet universally useful intermediate representation [16, 34]?

We draw inspiration from Kitana [37] which uses factorized linear regression to expedite data search. Kitana computes the gram matrix semi-ring (Section 3.1) for each dataset, allowing fast join/union with a candidate dataset and evaluation of the linear regression accuracy. While semi-rings were initially used for performance, they also make an ideal intermediate representation for DP. Thus, in the next section, we design FPM to privatize sufficient semi-ring statistics to support private ML over joins and unions.

3 FACTORIZED PRIVACY MECHANISM

In this section, we introduce Factorized Privacy Mechanism (FPM), which privatizes sufficient semi-ring statistics. We start with the factorized ML background, extend it to monomial semi-ring, present our main mechanism algorithms, and analyze its errors.

3.1 Factorized Machine Learning Primer

We start with the fundamental concepts of annotated relations and aggregation pushdown, then introduce factorized ML [13, 52].

Annotated Relations. The annotated relational model [32] maps $t \in R$ to a commutative semi-ring $(D, +, \times, 0, 1)$, where D is a set, $+$ and $\times$ are commutative binary operators closed over D, and $0/1$ are zero/unit elements. An annotation for $t \in R$ is denoted as $R(t)$. Semi-ring annotation expresses various aggregations. For example, the natural numbers semi-ring expresses count aggregations.

Semi-ring Aggregation Query. Semi-ring aggregation queries can now be reformulated using annotated relations by translating group-by, union, and join operations into addition (+) and multiplication ($\times$) operations over the semi-ring annotations, respectively.

$$(\gamma_{\mathbf{A}} R)(t) = \sum \{R(t_1) |\ t_1 \in R, t = \pi_{\mathbf{A}}(t_1)\}$$
$$(R_1 \cup R_2)(t) = R_1(t) + R_2(t)$$
$$(R_1 \bowtie R_2)(t) = R_1(\pi_{S_{R_1}}(t)) \times R_2(\pi_{S_{R_2}}(t))$$

(1) The annotation for group-by $\gamma_{\mathbf{A}} R$ is the sum of the annotations within the group. (2) The annotation for union $R_1 \cup R_2$ is the sum of annotations in R_1 and R_2. (3) The annotation for join $R_1 \bowtie R_2$ is the product of annotations from contributing tuples in R_1 and R_2.

Aggregation Pushdown. The optimization of factorized ML [13, 58] involves the distribution of aggregations γ (additions) through joins $\bowtie$ (multiplications). For example, consider the query $\gamma_D(R_1[A,B] \bowtie R_2[B,C] \bowtie R_3[C,D])$. Rather than applying γ on the join (which is $O(n^3)$ where n is relation size), γ can be performed on R before $\bowtie$ with S, and this process can be repeated two more times (in $O(n)$):

$$\gamma_D(\gamma_C(\gamma_B(R_1[A,B]) \bowtie R_2[B,C]) \bowtie R_3[C,D])$$

The associativity of additions can be similarly exploited for union:

$$\gamma_A(R_1[A,B] \cup R_2[A,B]) = \gamma_A(R_1[A,B]) \cup \gamma_A(R_2[A,B])$$

Factorized Linear Regression. The fundamental optimization of factorized ML is aggregation pushdown, but different semi-rings are used for different models. We use linear regression as an example.

We start with an overview of linear regression and its sufficient statistics. Given the training data $\mathbf{X} \in \mathbb{R}^{n \times m}$, and the target variable $\mathbf{y} \in \mathbb{R}^{n \times 1}$, the goal is to find parameters $\theta \in \mathbb{R}^{m \times 1}$ that minimize the square loss $\theta^* = argmin_\theta \|\mathbf{y} - \mathbf{X}\theta\|^2$, yielding a closed-form solution $\theta^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$. Including the target variable as a special feature and appending it to $\mathbf{X}$ for $\mathbf{X}' = [\mathbf{X} \mid \mathbf{y}]$, we find that $\mathbf{X}'^T\mathbf{X}' \in \mathbb{R}^{m' \times m'}$, where $m' = m + 1$, is the core sufficient statistics to compute, where each cell represents the sum of products between feature pairs.

We can compute $\mathbf{X}'^T\mathbf{X}'$ over the join $R_\bowtie = R_1 \bowtie \ldots \bowtie R_k$ by the covariance matrix semi-ring [58]. For m' features, the semi-ring is defined as a triple $(c, \mathbf{s}, \mathbf{Q}) \in (\mathbb{Z}, \mathbb{R}^{m'}, \mathbb{R}^{m' \times m'})$, which contains the count, sums, and sums of pairwise products respectively. The zero and one elements are $\mathbf{0} = (0, \mathbf{0}^m, \mathbf{0}^{m' \times m'})$ and $\mathbf{1} = (1, \mathbf{0}^{m'}, \mathbf{0}^{m' \times m'})$. $+$ and $\times$ between two annotations a and b are defined as:

$$a + b = (c_a + c_b, \mathbf{s}_a + \mathbf{s}_b, \mathbf{Q}_a + \mathbf{Q}_b)$$
$$a \times b = (c_a c_b, c_b \mathbf{s}_a + c_a \mathbf{s}_b, c_b \mathbf{Q}_a + c_a \mathbf{Q}_b + \mathbf{s}_a \mathbf{s}_b^T + \mathbf{s}_b \mathbf{s}_a^T)$$

Then, computing $\mathbf{X}'^T\mathbf{X}'$ is reduced to executing $\gamma(R_1 \bowtie \ldots \bowtie R_k)$, where aggregation can be pushed down as discussed before.

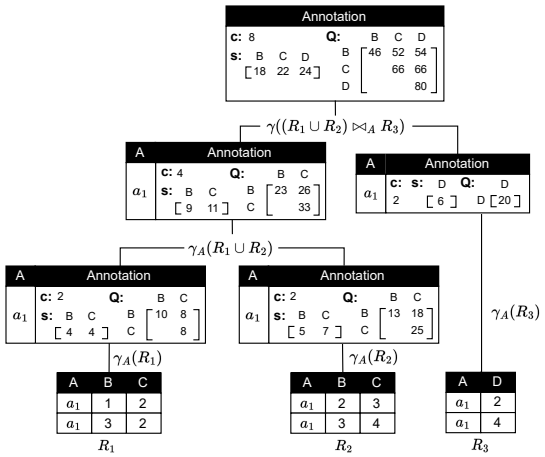


Figure 4: Optimized query plan of $\gamma((R_1 \cup R_2) \bowtie_A R_3)$ for factorized ML. Aggregations are pushed down before joins.

$(R_1 \cup R_2) \bowtie_A R_3$

A	B	C	D
a_1	1	2	2
a_1	3	2	2
a_1	2	3	2
a_1	3	4	2
a_1	1	2	4
a_1	3	2	4
a_1	2	3	4
a_1	3	4	4

Aggregated Monomials

$\sum 1 = 8$
$\sum B = 18$
$\sum C = 22$
$\sum D = 24$
$\sum B^2 = 46$
$\sum BC = 52$
$\sum C^2 = 66$
$\sum BD = 54$
$\sum CD = 66$
$\sum D^2 = 80$
$\sum B^3 = 126$
$\sum B^2C = 138$
$\sum B^2D = 136$
$\sum BC^2 = 180$
$\sum BCD = 156$
$\sum BD^2 = 164$
$\sum C^3 = 288$
$\sum C^2D = 220$
$\sum CD^2 = 198$
...

Statistics

1-order
$E[B] = 2.25$
$E[C] = 2.75$
$E[D] = 3$

2-order
$E[B^2] = 5.75$
$E[BC] = 6.5$
$E[C^2] = 8.25$
$E[BD] = 6.75$
$E[CD] = 8.25$
$E[D^2] = 10$

3-order
$E[B^3] = 15.75$
$E[B^2C] = 17.24$
$E[B^2D] = 17$
$E[BC^2] = 22.5$
$E[BCD] = 19.5$
$E[BD^2] = 20.5$
$E[C^3] = 36$
$E[C^2D] = 27.5$
$E[CD^2] = 24.75$
...

Figure 5: Aggregated monomials and statistics.

Example 2. *Consider R_1, R_2, R_3 in Figure 4. We aim to train linear regression on $(R_1 \cup R_2) \bowtie_A R_3$ using D as the feature and C as the target variable. The naive solution is to first materialize the union and join results (Figure 5) and then compute $\mathbf{X}'^T\mathbf{X}'$. Using factorized linear regression, we can optimize the query plan (Figure 4) by pushing down aggregations: $\gamma((\gamma_A(R_1) \cup \gamma_A(R_2)) \bowtie_A \gamma_A(R_3))$. This approach yields the same result as the naive solution, but avoids the costly materialization. We use the aggregates to fit the linear regression:*

$$\theta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} = \begin{bmatrix} \sum D^2 & \sum D \\ \sum D & \sum 1 \end{bmatrix}^{-1} \begin{bmatrix} \sum CD \\ \sum C \end{bmatrix} = \begin{bmatrix} 80 & 24 \\ 24 & 8 \end{bmatrix}^{-1} \begin{bmatrix} 66 \\ 22 \end{bmatrix}$$

After obtaining the model parameters θ, the model performance can also be evaluated. For square loss, $\sum(y - x^T\theta)^2 = \sum(y^2 - 2\theta^T xy + \theta^T x x^T \theta) = \mathbf{y}^T\mathbf{y} - 2\theta^T\mathbf{X}^T\mathbf{y} + \theta^T\mathbf{X}^T\mathbf{X}\theta$. The final aggregation result provides the necessary statistics to compute this expression.

3.2 Monomial Semi-ring

This section introduces sufficient statistics as vectors of monoids and extends it with semi-ring operations + and $\times$. This helps bridge ideas from two communities—semi-rings from the factorized ML literature that train models over joins and unions, but primarily focused on non-private linear regression, and privatized sufficient statistics from the ML literature [38, 44, 65] that approximate generalized linear models, but do not support joins and unions. We are the first to explicitly extend semi-ring from gram matrix (linear regression) to higher order monomial (generalized linear models). This section focuses on the semi-ring design of monomials to support join and union operations *without* DP. In the next section, we introduce FPM, a mechanism to privatize these monomials for DP.

We first define the k-order monomial [38] in sufficient statistics:

Definition 2 (k-order Monomial). *Given n random variables $f_1, f_2, ..., f_n$, the k-order monomials are random variables of monomials of the form $p = f_1^{k_1} f_2^{k_2} ... f_n^{k_n}$, where $k_1, k_2 ..., k_n$ are n non-negative integers such that $\sum_{i=1}^{n} k_i = k$.*

The core statistics to compute for ML are the expected value of each monomial $E[p]$. For example, 1-order monomials estimate means, 1, 2-order monomials estimate covariance (core sufficient statistics for linear regression), and 1, 2, 3-order monomials estimate skewness. Moreover, a generalized linear model can be approximated by high-order monomials using Taylor series expansions [38].

Example 3. *Consider the relation in Figure 5 (left) and random variables B, C, D. The 1-order monomials are B, C, D, the 2-order monomials are $B^2, BC, C^2, BD, CD, D^2$, and the 3-order monomials are $B^3, B^2C, B^2D, BC^2, BCD, ...$. The statistics (right) are the expected monomials when the relation is the population, and can be derived from the aggregated monomials (middle). The 1,2-order statistics are the sufficient statistics for linear regression training (Example 2).*

Instead of computing statistics over join and union through costly materialization and subsequent aggregation, factorized linear regression utilizes semi-ring operators for + and × to push down the aggregation of 1,2-order statistics. We extend this concept by defining operators for a k-order monomial semi-ring, thus generalizing factorized linear regression (2-order monomial semi-ring).

Definition 3 (k-order monomial semi-ring). *Given m features $f_1, f_2, ..., f_m$, the k-order monomial semi-ring has domain of a vector with size $\frac{1-m^k}{1-m}$ for $m \geq 2$, and $1+k$ for $m=1$. The domain breaks into $k+1$ subvectors $[s_0, s_1, ..., s_k]$ where s_i is a vector of size m^i. Then, given two semi-ring element $a = [s_0^a, s_1^a, ..., s_k^a]$ and $b = [s_0^b, s_1^b, ..., s_k^b]$, let:*

$$a + b = [s_0^a + s_0^b, s_1^a + s_1^b, ..., s_k^a + s_k^b]$$

$$a \times b = [s_0^a \otimes s_0^b, \sum_{i=0}^{1} s_i^a \otimes s_{1-i}^b, ..., \sum_{i=0}^{k} s_i^a \otimes s_{k-i}^b]$$

where $\otimes : R^p \times R^q \rightarrow R^{pq}$ is the tensor product defined as: for $\mathbf{a} = [a_1, a_2, ..., a_p]$ and $\mathbf{b} = [b_1, b_2, ..., b_q]$, tensor product computes the pairwise product $\mathbf{a} \otimes \mathbf{b} = [a_1 b_1, a_1 b_2, ..., a_p b_1, a_p b_2, ..., a_p b_q]$.

The zero element is a vector of all zeroes, and the one element is a vector with non-zero $s_0 = [1]$, but the rest as all zeroes.

Intuitively, each subvector s_k^a holds the k-order monomials with a size m^k, as there are m^k possible permutations with repetition. In order to compute statistics using a k-order monomial semi-ring, we annotate R by assigning to each tuple t its monomials (non-existing features are considered to be all zeros). Note that, while this vector representation provides a straightforward way to define semi-rings for arbitrary orders, it is inherently inefficient and can be optimized by the dictionary representation discussed next.

Dictionary Representation. Vector representation has redundancies (e.g., $f_1 f_2 = f_2 f_1$) and sparsity (nonexistent features are zeros). Dictionary representations [42] help reduce redundancy: monomials serve as keys to deduplicate, and monomials with zeros are not materialized. We next provide an example of semi-ring operations using the dictionary representation for join-aggregation:

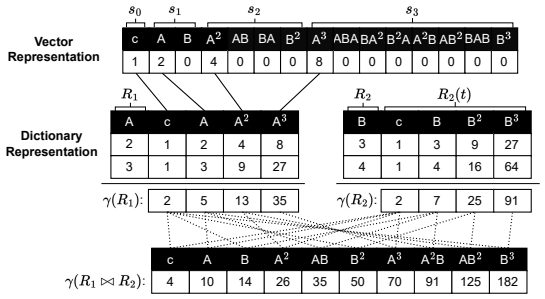


Figure 6: Aggregation of 3-order monomial semi-ring over join: $\gamma(R_1[A] \bowtie R_2[B])$. Each row is one tuple, and we show the vector representation for the first tuple in R_1. Dictionary representation removes redundancy and sparsity. Dotted lines map the contributing components to aggregated results.

Example 4. *Consider two relations of a single feature $R_1[A] = [2, 3]$ and $R_2[B] = [3, 4]$, and the aggregation query $\gamma(R_1 \bowtie R_2)$ for 3-order monomial semi-ring. Figure 6 illustrates the annotated relations and the query processing. To start, the aggregations are pushed down by summing each monomial. Next, the monomials are combined according to the multiplication operator for join.*

Assuming the join result, $R_{\bowtie}$, as the population, we can use the aggregated monomials to compute statistics (i.e., the expected monomials). Let $s=\gamma(R_{\bowtie})$ be the aggregated monomial semi-ring. Then, for monomial p: $E[p] = s[p]/s[c]$, where 4 is the count (0-order monomial). For example, in Figure 6, $E[AB]=s[AB]/s[c]=35/4=8.75$. The aggregated monomials comprise count and sum aggregations over the base tables, which can be efficiently computed by requesters/providers using SQL queries. Further, they serve as an ideal intermediate for DP due to their reusability, as discussed next.

3.3 FPM Mechanism

In this section, we present the Factorized Privacy Mechanism (FPM) which applies the Gaussian mechanism to the aggregated monomials discussed in the previous section to support differentially private data search (Section 2.3) while maintaining high utility. The primary algorithmic challenge FPM addresses is designing sufficient statistics that are composable (through semi-ring operators) and reusable (as post-processing without additional privacy cost) to support ML across various join and union augmentations.

We make the following simplifications: (1). Features consist only of numerical attributes, and join keys consist only of categorical attributes. Section 3.5 describes preprocessing to support categorical features. (2). Group-by operator γ has been extended to annotate group-by keys without tuples with zero elements (group-by attribute domains are assumed public in Section 2.3). (3). Datasets are preprocessed so that the ℓ_2 norm of the features in each tuple is bounded by a constant value B, following previous works [26, 65].

Algorithms. The FPM mechanism, detailed in Algorithm 1, is applied locally by either the requester or provider before dataset upload to the search platform. It uses as inputs: (1) the relation R to be privatized (2) the join key A[2], which = *null* if R is only for union,

[2] A could be composite. To support multiple join keys, the DP budget can be split among different key combinations. Additionally, optimization techniques can be applied to take advantage of the correlations between join keys [54].

(3) the order of monomials, k, based on the model to support, and (4) the DP budget (ϵ, δ) for R. FPM computes locally aggregated monomials $\gamma(R)$ and applies the Gaussian mechanism to these, with sensitivity optimized based on order parity and feature count (line 2, 7): For even-order monomials, sensitivity is reduced by $\sqrt{2}$, and if there's only one feature, sensitivity is reduced by another $\sqrt{2}$.

Theorem 5. *FPM is* $(\epsilon, \delta) - DP$.

Proof Sketch. FPM applies the Gaussian mechanism [22] to the aggregated monomials for $(\epsilon, \delta) - DP$. Therefore, we only need to show the correctness of Δ. We present simple cases illustrating proof concepts for the union and join of 1 feature (with lower Δ), and the union of 2 features with 1/2-order monomial semiring. These cases are meant to illustrate the key intuitions; full proofs and generalizations are available in Appendix A due to space limits.

• *(1 feature, Union, any order)* For union, count (0-order monomial) remains unchanged as we consider bounded DP, where the neighbour relation has one tuple modified (instead of removed/added). Let the modified feature value be $a \to a'$ where both a and a' have a domain of $[-B, B]$. Then, for the i-th monomial, the squared difference is $(a^i - a'^i)^2$. When i is odd, $a^i \in [-B^i, B^i]$, and $(a^i - a'^i)^2 \le (2B^i)^2$. When i is even, $a^i \in [0, B^i]$, and $(a^i - a'^i)^2 \le (B^i)^2$.

•*(1 feature, Join, any order)* For join, the query also groups results by join key A. This can be considered as a histogram [69], where each bin is a join key, and the value is the k-order monomial semiring. The neighbouring relation has two cases: the modified tuple has changed the join key or not. If the join key doesn't change, this is the same as the union case. If the join key changes, there are two bins with a maximum square difference of $\sum_{i=0}^{k} B^{2i}$ (note that, unlike the union, the counts change). Thus, the sensitivity is bounded by $\sqrt{2\sum_{i=0}^{k} B^{2i}}$. Finally, we take the maximum.

•*(2 features, Union, 1-order)* Let the modified feature value be $(a, b) \to (a', b')$ where both $a^2 + b^2$ and $a'^2 + b'^2$ are $\le B^2$. Then, consider the 1-order monomials $(a, b), (a', b')$. The squared difference is:

$$(a - a')^2 + (b - b')^2 \le (2a^2 + 2a'^2) + (2b^2 + 2b'^2) = 4B^2$$

The sensitivities for higher odd orders are similar.

• *(2 features, Union, 2-order)* For even-orders, we can obtain a tighter bound. Consider the 2-order monomials $(a^2, ab, b^2), (a'^2, a'b', b'^2)$. The squared difference is:

$$\begin{aligned}
&(a^2 - a'^2)^2 + (ab - a'b')^2 + (b^2 - b'^2)^2 \\
\le &(a^2 - a'^2)^2 + 2(ab - a'b')^2 + (b^2 - b'^2)^2 \\
= &(a^4 - 2a^2a'^2 + a'^4) + (2a^2b^2 - 4aba'b' + 2a'^2b'^2) + (b^4 - 2b^2b'^2 + b'^4) \\
= &(a^2 + b^2)^2 + (a'^2 + b'^2)^2 - 2(aa' + bb')^2 \le B^4 + B^4 - 0 = 2B^4
\end{aligned}$$

For higher even orders, we can similarly amplify the monomials by the binomial coefficients (second line) to find a non-negative red term for even-order monomials, resulting in a tighter bound. Extending to joins follows a similar approach as the single feature case, where we consider group-by queries as histograms. □

3.4 Comparison with Other Mechanisms

We next analyze the error of FPM in estimating the statistics s (expected values of monomials). Generally, the expected errors of s are correlated with the error of the target model parameter β and accuracy; we will study the confidence bound for linear regression parameter in the next section, where the s error is the key factor.

Algorithm 1: FPM mechanism

inputs : Relation R, Join Key A, Order k, DP budget (ϵ, δ)
output : Privatized Aggregated Relation $\tilde{R}$

1 **if** A = *null (Union Only)* **then**
2 $\quad \Delta = \sqrt{\sum_{i=1}^{k} (\text{if i odd: 4, elif \#fea=1: 1, else: 2}) \cdot B^{2i}}$;
3 $\quad \sigma, \tilde{R} = \sqrt{2\ln(1.25/\delta)}\Delta/\epsilon, \gamma(R)$;
4 $\quad$ // add i.i.d. noises to each $1-k$ order monomial s;
5 $\quad \tilde{R} = \{s : \tilde{R}[s] + e{\sim}\mathcal{N}(0, \sigma^2)$ for $1-k$ monomial $s\}$;
6 **else**
7 $\quad \Delta = max(\sqrt{\sum_{i=1}^{k} (\text{if i odd: 4, elif \#fea=1: 1, else: 2}) \cdot B^{2i}}, \sqrt{2\sum_{i=0}^{k} B^{2i}})$;
8 $\quad \sigma, \tilde{R} = \sqrt{2\ln(1.25/\delta)}\Delta/\epsilon, \gamma_A(R)$;
9 $\quad$ **foreach** $a \in dom(A)$ **do**
10 $\quad\quad$ // add i.i.d. noises to each $0-k$ order monomial s;
11 $\quad\quad \tilde{R}(a) = \{s : \tilde{R}(a)[s] + e{\sim}\mathcal{N}(0, \sigma^2)$ for $0-k$ monomial $s\}$;
12 **return** $\tilde{R}$;

Setting. We consider a data corpus with size n_{corp} (defined as the number of provider datasets) and has received n_{req} requests. To simplify the analysis, we assume that: (1) the search only uses union operations (and we will discuss the extension to join). (2) each dataset has one feature, n tuples, and a DP budget of (ϵ, δ). The search platform evaluates all possible augmentations, each corresponding to a unique combination of provider datasets.

Metrics. The goal is to evaluate, for each augmentation, the expected ℓ_2 error of the privatized set of monomials $\tilde{s}$: $E[\|s - \tilde{s}\|_2]$.

Mechanisms. We compare FPM with standard DP mechanisms used in various existing trust models:

- For *Saibot*'s trust model (Section 2.3), FPM (Algorithm 1) privatizes local aggregates independently for each dataset, and combines the aggregates with factorized ML.
- For the local model, the Per-tuple Privacy Mechanism (TPM)[3] applies Algorithm 1 to privatize each tuple [70].
- For the global model, the Aggregate Privacy Mechanism (APM)[4] first computes the union result $R_\cup$ after augmentation, and then applies Algorithm 1 to $\gamma(R_\cup)$ [65]. To ensure $(\epsilon, \delta) - DP$ for all $n_{req}(2^{n_{corp}-1} - 1)$ augmentations, the DP budget has to be split.
- For the shuffle model, shuffling [27] privatizes each tuple, similar to TPM, but applies Laplace mechanism with the amplified privacy budget. These tuples are shuffled either at the 1^{st}- (SF−1) or 2^{nd}-level (SF−2); akin to APM, SF−2 requires budget splits.

Proposition 6. *For the estimation of each augmentation (assuming that the number of augmented datasets and the order of s are small constants),* FPM/SF−1 *has expected* ℓ_2 *error of* $\tilde{O}(\Delta/n\epsilon)$*, while* TPM *has an error of* $\tilde{O}(\Delta/\sqrt{n}\epsilon)$ *and* APM/SF−2 *has an error of* $\tilde{O}(n_{req}2^{n_{corp}}\Delta/n\epsilon)$*, where* $\tilde{O}(\cdot)$ *hides at most a logarithmic term.*

The proof is in Appendix B.

[3] An alternative is to apply Gaussian mechanism to raw tuples and then compute monomial semi-ring; this, however, results in an even larger error.
[4] There are other alternatives like perturbing objectives and gradients; however they are similarly limited by the combinatorially large number of models to train.

Remark. Proposition 6 highlights prior mechanisms' limitations: APM/SF−2 are competitive only for small corpora and quickly exhaust budget for larger requests/corpus sizes due to budget split for all possible augmentations, and require trust in centralized aggregators/shufflers. TPM adds excessive noise to each tuple, requiring quadratically more tuples to achieve the same level of error as FPM. Although SF−1 can theoretically match FPM's complexity with privacy amplification, it's significant only for large numbers of tuples. For instance, given ϵ=1 and δ=10^{-6}, ϵ is amplified when n reaches ~650 [27, 29]. However, small n needs amplification most, where SF−1 provides much larger errors than FPM (Section 5.2).

Extending the analysis to joins involves considering group-by errors based on domain size and multiplication of privatized monomials. Comparisons remain similar: TPM needs a quadratically larger data size, while APM may outperform FPM only for small corpora and requests but exhausts budget for larger corpus sizes.

3.5 Differentially Private Data Search Platform

In this section, we discuss *Saibot*, a data search platform that integrates FPM to ensure differential privacy.

Provider. The architecture of the *Saibot*, which uses FPM for DP, is illustrated in Figure 2. For each dataset R data provider owns, the supported operation ($\bowtie$ /$\cup$[5] or $\cup$-only) is decided. If join is supported, the join key A must also be specified. FPM is then applied locally to R to privatize the sufficient statistics $\widetilde{\gamma(R)}$, which are then uploaded to *Saibot*. As *Saibot* is not trusted, *data storage* only stores privatized statistics, but not raw data. All operations over $\widetilde{\gamma(R)}$ are post-processing without additional DP costs.

Requester. The requester has model type M and R_{train}, and wants to improve accuracy on R_{test}. The requester computes and submits to *Saibot* the privatized sufficient statistics $\widetilde{\gamma(R_{train})}$ and $\widetilde{\gamma(R_{test})}$. *Data discovery* returns a set of joinable or unionable relations R from *data storage*. Then, *Data search* applies greedy algorithm (following Kitana [37]): in each iteration, it evaluates each candidate and adds the one that most improves the model accuracy. *Saibot* is agnostic to the search algorithm, and others [20, 62] can also be used.

Data Discovery. Previous data discovery systems [17, 30] leverage MinHash sketches, column type and data distribution statistics; *Saibot* supports all of them. Specifically, for categorical attributes, we utilize minhash sketches, computed from public domains, to measure set similarity. For numerical attributes, we rely on public schemas for column names and types. Additionally, we construct (approximated) data distribution statistics such as count, mean, standard deviation, and correlation from the privatized 2-order monomial semi-rings, without additional DP costs.

Preprocessing. Before applying FPM, requesters and providers can locally preprocess datasets to enhance utility and robustness. For instance, datasets may have categorical features not directly supported by the proxy model (linear regression). For low cardinality categories, standard one-hot encoding can be applied, treating the encoded features as numerical for privatization by FPM. However, high cardinality categorical features yield high-dimensional vectors when one-hot encoded, which is problematic and typically requires specialized techniques [18, 48]. This is precisely the problem *Saibot* can address through augmentation. By joining with augmentations, high cardinality categories in R_{train}, like location, can be encoded into meaningful lower dimensional numerical features, like population and economic indicator, from augmented relations. Hence, we suggest using high cardinality categorical features as join keys.

Saibot also applies two steps to boost DP robustness. First, it removes outliers (>1.5 std from the mean), which typically improves model performance and reduces the tuple ℓ_2 norms, enhancing DP noise robustness [45]. Second, *all DP mechanisms* (including ours) degrade with increasing dimensionality due to the increased tuple ℓ_2 norms. Thus, *Saibot* applies dimensionality reduction [47] to retain the top K principal components (K=1 works best in our experiments), and rescales tuples to bound max ℓ_2 norm $\leq B$. This lowers the noise scale, improves utility, and achieves a lower sensitivity Δ with #fea= 1 (Algorithm 1). These steps are applied to all datasets and DP baselines in our real-world experiments (Section 5.1).

Supporting Varied Privacy Needs. A unique benefit of *Saibot*'s design is that it can adapt to different privacy needs. In cases where pure DP (δ=0) is required, FPM can be modified to apply Laplace mechanisms [23]. In situations where individuals don't trust providers or requesters, FPM can be reduced to LPM to privatize individual tuples. Conversely, shuffling only guarantees approximate DP and GPM always requires a trusted centralized aggregator.

ML training after data search. After *Saibot* finds predictive augmentations using a differentially private proxy model (linear regression), the model could be directly returned to requesters. However, requesters may need more complex model M, and the training shall also satisfy DP. To achieve this, *Saibot* can be integrated within a larger differentially private federated ML system [61, 64, 66, 72], where *Saibot* first locates augmentations, and then the ML systems use the augmented dataset to train sophisticated models, such as deep neural networks, via differentially private gradient descent.

Scope. While *Saibot* can employ FPM to support a wide range of models [57] and approximate GLM [44], this paper focuses on linear regression [58] because it's widely used and is adopted by previous data search [19, 37, 56]. Next, we dive deep into linear regression to analyze the task utility and propose further optimizations.

4 DIVING DEEP INTO LINEAR REGRESSION

This section examines the ML task *utility* FPM provides and suggests optimizations for linear regression. We start with the assumption of linear regression on many-to-many join (as opposed to one-to-one [35, 64]), which is challenging due to unexpected duplication and independence. We then propose an unbiased estimator. Next, we explore the confidence bounds for the linear regression parameters and propose optimizations to tighten the bound further.

4.1 Linear Regression on Many-to-Many Join

Linear regression assumes a noisy linear relationship between the features and target variable: $\mathbf{y} = \mathbf{X}\beta + \mathbf{e}$, where $\mathbf{e}$ is the error term. This is consistent with our assumption so far if $R_{\bowtie} = R_1 \bowtie \ldots R_k$ is the population, and let us use the monomial semi-ring to compute the expected s. However, when many-to-many joins are involved, $R_{\bowtie}$ often doesn't represent the population as joins generate Cartesian products for each matching key. This leads to (1) duplicated tuples (the same $\mathbf{y}$ values are repeated) and (2)

[5] Any dataset supports join also supports union by aggregating out the join key.

unexpected *independence* between features from different relations with the same join key, leading to biased estimation.

To the best of our knowledge, linear regression over many-to-many joins has been understudied. The closest work is multi-view learning [31, 33], which pre-aggregates (e.g., averaging) features. However, this introduces errors for long join paths due to Simpson paradox [53] (e.g., average of average is not average). In contrast, we propose an unbiased estimator based on the assumptions from vertical federated ML that each party holds a projection; this complements prior factorized ML work [13, 52], which studied the computational complexity of many-to-many joins.

Our analysis focuses on an easy-to-explain case inspired by vertical federated ML [35, 64], where we want to train linear regression over relation R. However, R is not directly observable, and each party can only access a projection $\pi(R)$. Multiple $\pi(R)$ may have many-to-many relationships on the common attribute (join key) instead of the one-to-one relationships studied by federated ML. The objective is to train linear regression on R collectively.

Unbiased Estimator. Given R of cardinality n, suppose there are two parties holding different projections $\pi_{F_1}(R)$ and $\pi_{F_2}(R)$, and the goal is to compute the 2-order monomial semi-ring $\gamma(\pi_{F_1 \cup F_2}(R))$. However, factorized ML is trained on $R_{\bowtie} = \pi_{F_1,J}(R) \bowtie_J \pi_{F_2,J}(R)$ with join key $J = F_1 \cap F_2$; $R_{\bowtie}$ is likely to differ from $\pi_{F_1 \cup F_2}(R)$ (unless J is primary key), resulting in bias. To address this, we propose an unbiased estimator for s based on $s' = \gamma(R_{\bowtie})$.

Proposition 7 (Unbiased Estimator of s over R). *We make the simplifying assumption that J is uniformly distributed (if $d = |dom(J)|$, each $j \in J$ appears n/d times in R) and is not correlated with any other attribute. Let $s' = \gamma(R_{\bowtie})$. Then,*

$$\hat{s} = \begin{cases} f_1 f_2 = \frac{1-n}{1-d}\frac{s'[f_1 f_2]}{s'[c]} + \frac{n-d}{1-d}\frac{s'[f_1]}{s'[c]}\frac{s'[f_2]}{s'[c]} \\ \qquad\qquad \text{for } f_1 \in F_1 - J, f_2 \in F_2 - J \\ p = s'[p]/s'[c] \quad \text{for any other monomial } p \end{cases}$$

$\hat{s}$ is an unbiased estimator of monomial semi-ring $s = \gamma(\pi_{F_1 \cup F_2}(R))$.

The proof is in Appendix C. We assume vertical partitions of R, but real-world datasets may also be horizontally partitioned; the estimators could be refined for these cases. Our analysis studies the base case, and the unbiased estimator can be recursively applied for multiple joins and unions. Note that the estimators are post-processing steps without compromising DP.

4.2 Simple Linear Regression Analysis

Building on the assumption in the previous section, this section studies the confidence bound of factorized linear regression. Compared to [65], our analysis focuses on simple linear regression with one feature, under less stringent assumptions; this scenario is sufficient to show FPM's advantages over other mechanisms, and motivates optimization. We first consider a single relation case, then extend to union and join. We'll begin with defining the confidence bound, which will be used to evaluate the utility of private estimators.

Definition 4 (Confidence Bound). *Given parameter θ, the $(1-p)$ confidence bound $C^{\tilde{\theta}}_{\hat{\theta}}(p)$ for an private estimator $\tilde{\theta}$ is:*

$$C^{\tilde{\theta}}_{\hat{\theta}}(p) = \inf\ \{b : \mathbb{P}[|\tilde{\theta} - \hat{\theta}| \le b] \ge 1 - p\}$$

where $\hat{\theta}$ is the non-private estimator.

We consider relation $R[x, y]$ with one feature x, target variable y, and cardinality n. We want to train $y = \beta_x \cdot x + \beta_0$, and focus on the parameter β_x; β_x has an optimal non-privitized estimator $\hat{\beta}_x = \frac{\widehat{E[xy]} - \widehat{E[x]}\widehat{E[y]}}{\widehat{E[x^2]} - \widehat{E[x]}^2} = \widehat{\sigma^2_{xy}}/\widehat{\sigma^2_x}$, where $\widehat{\sigma^2_{xy}}$ and $\widehat{\sigma^2_x}$ are polynomials that can be derived from aggregeted 2-order monomials $\gamma(R)$. We apply FPM to compute the privatized 2-order $\gamma(R)$ and study the confidence bound of the privatized estimator $\tilde{\beta}_x$. Note that more familiar error definitions like mean-squared-error can be upper bounded, roughly, by the square of the confidence bound.

Theorem 4.1 (Confidence Bound of $\tilde{\beta}_x$). *For every p where $\tau_1 < 1$ holds, the $(1-p)$ confidence bound for $\tilde{\beta}_x$ is:*

$$C^{\tilde{\beta}_x}_{\hat{\beta}_x}(p) \le \tau_2 + \frac{\tau_1}{1-\tau_1}\left(\hat{\beta}_x + \tau_2\right)$$

where $\tilde{\beta}_x$ ($\hat{\beta}_x$) is the private (non-private) estimate of β_x. Let B_1 and B_2 be the $(1-p)$ confidence bounds for $\widehat{\sigma^2_x}$ and $\widehat{\sigma^2_{xy}}$ respectively. Then $\tau_1 = B_1/\widehat{\sigma^2_x}$ and $\tau_2 = B_2/\widehat{\sigma^2_x}$ are both $O\left(\frac{B^4 \ln(1/\delta)\ln(1/p)}{\epsilon^2 n \widehat{\sigma^2_x}}\right)$. The probability is taken over the randomness of FPM.

The proof and extension to multi-features can be found in Appendix D. Theorem 4.1 demonstrates that the private estimator $\tilde{\beta}_x$ is asymptotically close to the non-private $\hat{\beta}_x$. The key factors in reducing the discrepancy are τ_1, τ_2. APM and SF−2 have combinatorially large τ_1, τ_2 due to the budget splits. TPM requires quadratically more data than FPM to achieve the same level of τ_1, τ_2.

Extension to Factorized ML. The full procedures to extend the confidence bounds for factorized ML are in Appendix D. For the union of k datasets: $R = R_1 \cup R_2 ... \cup R_k$, τ_1 and τ_2 are reduced by a factor of $\sqrt{k}$, while the rest remain unchanged. For the join of two datasets $R[x, y, J] = R_1[x, J] \bowtie R_2[y, J]$, where $|dom(J)| = d$, there is additional noise to the count which could cause distortion if the privatized count is close to or less than 0. To address this, an additional assumption that noises to count is $o(n/d)$ is needed [65], resulting τ_1 and τ_2 to increase by a factor of $O(\sqrt{d})$ and $O(n/\sqrt{d})$.

Algorithm 2: FPM−OPT algorithm for Join

inputs : Relation R, Join Key A, Order k, DP budget (ϵ, δ)
output : Privatized Annotated Relation $\tilde{R}$

```
foreach i ∈ {0, ..., k} do
    Δ, ε', δ' = (if i odd: 2, else:√2) · B^i, ε/(k+1), δ/(k+1);
    σ, R̃ = √(2 ln(1.25/δ')) Δ/ε', γ_A(R);
    foreach a ∈ dom(A) do
        // add i.i.d. noises to each i order monomial s;
        R̃(a)={s : R̃(a)[s] + e~N(0, σ^2) for i monomial s};
return R̃;
```

4.3 Optimization: Better Noise Allocation

In Section 4.2, we analyzed the linear regression confidence bounds. We propose to adjust noise allocation to improve the bounds further.

First, previous work (e.g., [15]) has shown that β_x is usually the parameter of interest instead of β_0 for linear regression over the union. In this case, we suggest each provider adding noises directly to $\sigma^2_x, \sigma^2_{xy}$, rather than monomials x^2, xy, x, y. This reduces τ_1 and τ_2 by a factor of $O(B^2\sqrt{\ln(1/\delta)\ln(1/p)}/\epsilon)$ (Appendix E).

Second, optimizing joins is more difficult as we add noise locally to monomials to circumvent combinatorially large DP costs. However, we can reduce τ_1, τ_2 by $O(B^2)$ through smart budget allocation (Appendix E). Our insight is that lower-order monomials are multiplied by more monomials than higher-order ones. For example, in Figure 6, 0-order monomials are multiplied by 0, 1, 2, 3-order ones, while 3-order monomials only multiply with 0-order ones. Hence, we shall decrease the noise to lower-order ones. FPM−OPT in Algorithm 2 achieves this by dividing the DP budget across orders; lower order monomials have lower sensitivity and thus fewer noises.

5 EVALUATION

We evaluate FPM on NYC Open Data [11] corpus of 329 datasets for an end-to-end dataset search. We then use ablation studies via synthetic datasets to validate our theoretical analyses.

5.1 Real-world Experiments

Data and Workload. We construct a large data corpus of 329 NYC Open Data [11] datasets. Since prior DP mechanisms need to know the number of requests up front, we create a workload of 5 requests using the following random datasets:

- **Regents** [4] contains 2014-17 regents exams data.
- **ELA** [1] contains 2013-18 Early Learning Assessment (ELA) data.
- **Gender** [3] contains 2013-16 ELA data by grades and gender.
- **Grad** [5] contains 2016-17 graduation outcomes.
- **Math** [2] contains 2013-18 Math grades.

For each request, we look for a single dataset to join/union with the requested dataset. We turn off data discovery so every dataset in the platform is considered. By default, each dataset has DP budget ($\epsilon = 1, \delta = 10^{-6}$). We report the final $r2$ score evaluated non-privately. For reliability, we run each request 10 times.

Baselines. We consider different DP mechanisms. **Non−P** doesn't use DP and provides $r2$ upper bound. **FPM** applies Algorithm 2 to each dataset. **APM** (Aggregate Privacy Mechanism), following Wang [65], applies Algorithm 2 to the augmented dataset to privatize the aggregated sufficient statistics (and requires a trusted search platform). We use attribute max-frequence from Flex [39] to bound join sensitivity. Note that APM requires budget splits across all augmentations. **TPM** (Per-tuple Privacy Mechanism) applies Algorithm 2 to each tuple and uses half the ϵ to perturb the join key with generalized random response [40]. **SF** is similar to TPM, but applies the Laplace mechanism to each tuple with an amplified budget then shuffles [27, 29]. Since SF doesn't support joins (by 2^{nd}-level aggregator), we only shuffle each dataset locally by 1^{st}-level aggregators. In each case, we use a failure mechanism that reports $r2 = -\infty$ if the privatized $\widetilde{\mathbf{X}^T\mathbf{X}}$ is not positive definite [15].

Results. Figure 7 shows the non-private $r2$ of 10 runs of private data search for the 5 requests. FPM dominates the DP alternatives and is ∼50−90% of the non-DP case. FPM's performance depends on dataset cardinality: the *Gender* dataset contains on average ∼40 tuples per join key (compared to >100 tuples per join key in other datasets) and is more vulnerable to noise.

We next vary the number of datasets by sampling $n_{corp} \in \{10, 50, 100, 300\}$ datasets and rerunning each baseline over the smaller corpus. Figure 8 reports the median $r2$. For a small corpus ($n_{corp} = 10$), APM outperforms FPM because it imputes noise to the aggregated statistics across join key values and there are fewer budget splits, while FPM has to add noise to the individual statistics for each join key. TPM and SF have low $r2$ due to high noise.

Finally, we vary the number of requests ($n_{req} \in \{1, 10, 50, 100\}$) by sending the same request n_{req} times, and report median $r2$. Figure 9 shows that each baseline is almost invariant to n_{req}, and FPM dominates. In theory, APM is worse for more requests but is already poor due to the large dataset corpus.

5.2 Synthetic Dataset Experiments

We next validate our theoretical analysis of linear regression using synthetic data, and conduct ablation tests to study the impact of various parameters (number of tuples n, DP budget ϵ, δ, corpus size n_{corp}, number of requests n_{req} and join key domain size d).

5.2.1 Setup. We generate datasets by first creating a symmetric positive-definite matrix (`make_spd_matrix` in *sklearn*) as the covariance $\mathbf{X}'^T\mathbf{X}'$. We then sample from a multivariate normal distribution with this covariance to create a relation. To ensure the ℓ_2 norms of tuples $\leq B$=5, we resample for any tuples that exceed this limit.

By default, for union, we generate relations with $n = 1000$ tuples and 3 numerical attributes $[y, x_1, x_2]$. For join, we generate relations with $n = 10000$ tuples and include a categorical join key J uniformly distributed with a domain size of $d = 100$. We construct two vertical partitions with projections $[y, x_1, J]$ and $[x_2, J]$, respectively. We start with $n_{corp} = 2$ datasets, $n_{req} = 1$ request.

We will report the ℓ_2 distance to the non-private sufficient statistics (s error) and regression parameter (β error) as metrics. Each experiment will be repeated 100 times, and we will present the medians (dots), as well as the $25th$ and $75th$ percentiles (error bars).

5.2.2 DP for Union. Baselines include APM, TPM (same as in Section 5.1), SF−1, which shuffles tuples locally, SF−2, which shuffles the unioned dataset, and FPM using Algorithm 1 rather than Algorithm 2 (which is for join).

First, we vary $n \in \{10, 100, 500, 1K, 10K\}$. Figure 10a and Figure 10b report s and β errors. Since there are n_{corp}=2 datasets, APM and FPM perform similarly. In contrast, TPM requires quadratically more data to achieve the same s errors, consistent with our analysis in Section 3.4. SF's amplification is not significant for small n, when it's needed most, and both variants have high s errors. β error eventually converges to 0 for all baselines, but FPM does so at a comparable rate to APM (n=500 vs. $10K$ for the others).

Figure 10c shows that β error naturally correlates with s error, and higher s error increases the chance of failure (β error =∞). The remaining results will focus on β error, as it is of interest.

Next, we vary the DP budget ϵ=0.1 or δ=0 (pure DP). The results are shown in Figure 10d and Figure 10e, respectively. For ϵ=0.1, the plot shifts right due to a smaller budget. In the case of pure DP with δ=0, FPM, APM and TPM can adapt to it by applying Laplace mechanism, achieving similar performance. In contrast, SF−1 and SF−2 fail as only approximate DP is supported.

Figure 10f and Figure 10g vary the number of datasets and requests $n_{corp}, n_{req} \in \{1, 5, 10, 50, 100\}$, respectively. FPM's β error is flat. TPM, SF−1 and SF−2 frequently fail due to high noise, while APM only performed well when $n_{corp} \leq 5$ or $n_{req} \leq 10$. **APM is hence unsuitable for large data corpora.**

Figure 10h reports linear regression optimization benefit in Section 4.3. For a two-attribute dataset $R[y, x]$, while FPM adds noise to monomials (x, y, x^2, y^2, xy), FPM−OPT adds noise to polynomials

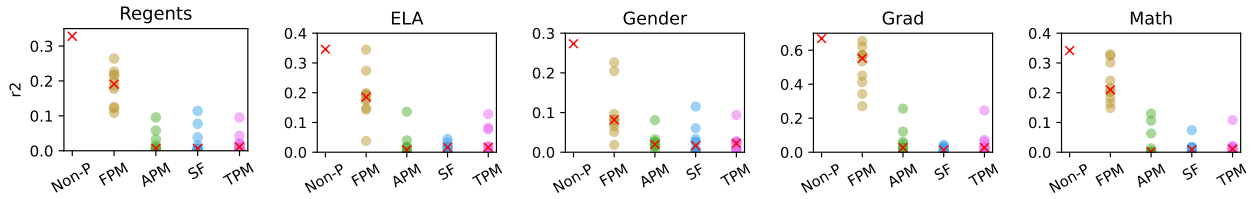


Figure 7: Utility (non-privatized $r2$) of the returned combinations of datasets searched by different baselines over 10 runs, with the median indicated by a red cross. FPM exhibits significantly better utility than the other baselines.

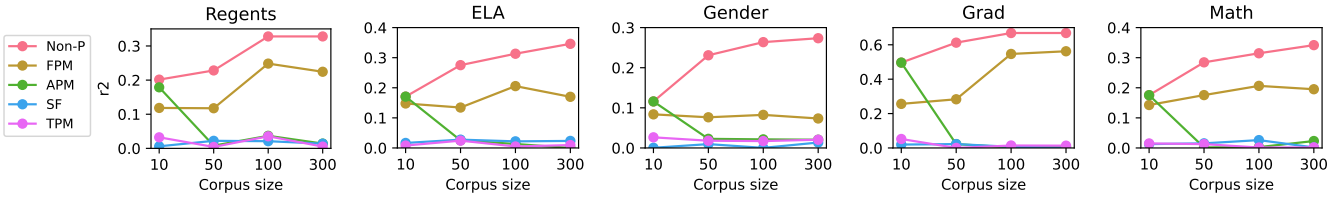


Figure 8: Utility (non-privatized $r2$) of the returned combinations of datasets searched by different baselines when varying the corpus size n_{corp}. FPM is scalable for large corpus, while APM only performs well for small n_{corp}.

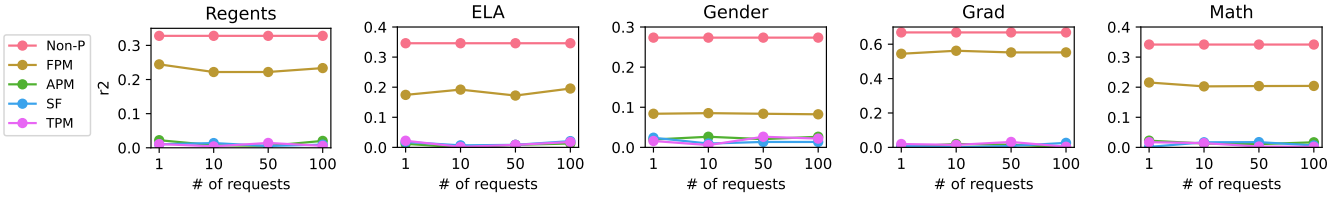
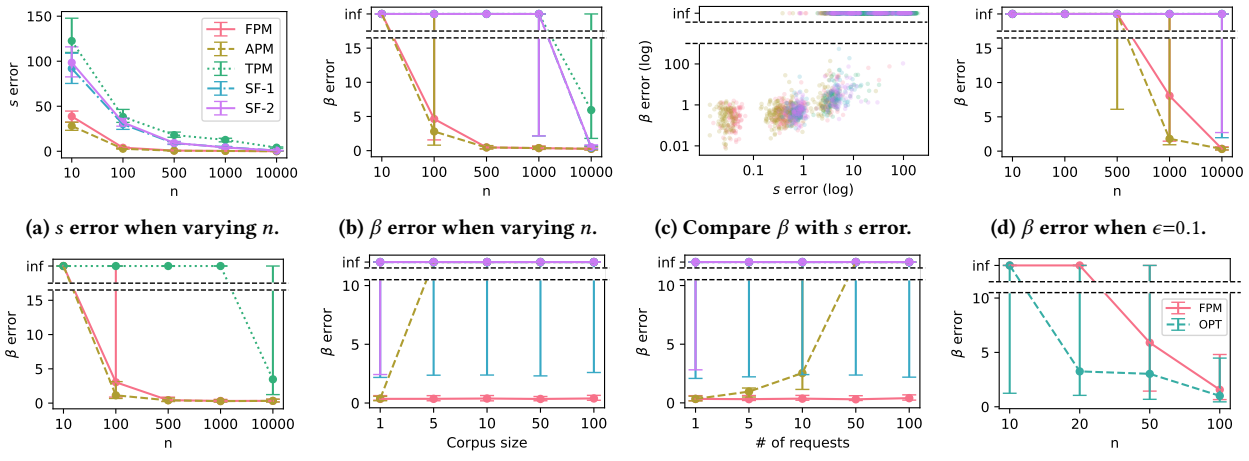


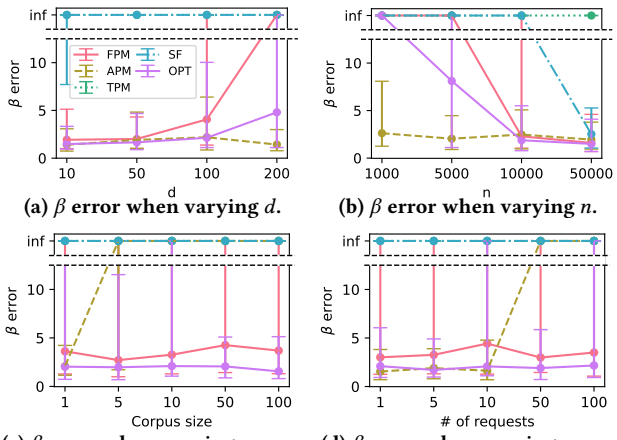
Figure 9: Utility (non-privatized $r2$) of the returned combinations of datasets searched by different baselines when varying the number of requests n_{req}. FPM consistently performs well, while the others either suffer from large noises (TPM, SF) or budget splits (APM).



(a) s error when varying n. **(b) β error when varying n.** **(c) Compare β with s error.** **(d) β error when ϵ=0.1.**

(e) β error when δ=0. **(f) β error when varying n_{corp}.** **(g) β error when varying n_{req}.** **(h) β error with optimized FPM.**

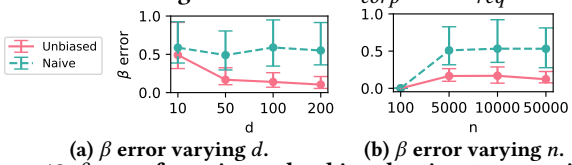
Figure 10: Ablation tests for unions: (a). (b). All baselines show reduced s and β errors as n increases, with FPM and APM exhibiting the least error; (c). Larger s error results in higher β errors and a greater risk of failure; (d). For $\epsilon = 0.1$, the β error plot shifts to the right; (e). For pure DP with $\delta = 0$, FPM can adjust to it and offer comparable utility, while SF fails; (f). (g). FPM is scalable for large corpora with large numbers of requests, while APM deteriorates significantly when $n_{corp} > 5$ or $n_{req} > 10$; (h). For simple linear regression when only β_x is of interest, FPM–OPT reduces the β errors and failure risk, particularly for small n.



(a) β error when varying d. (b) β error when varying n.

(c) β error when varying n_{corp}. (d) β error when varying n_{req}.

Figure 11: Ablation tests for joins: (a,b). FPM, FPM−OPT and APM provide low error when varying d and n, while TPM and SF are dominated by high noises (c,d). FPM, FPM−OPT show scalability for large repositories with numerous requests, whereas APM has high errors when n_{corp}>5 or n_{req}>10.



(a) β error varying d. (b) β error varying n.

Figure 12: β error for naive and unbiased estimators over join. The unbiased estimator dominates the naive one.

$(\sigma_x^2, \sigma_{xy}^2)$ because we only care about β_x. We find that FPM−OPT reduces the β error and failure likelihood, especially for n<100.

5.2.3 DP for Join. We evaluate different DP mechanisms over the join. Baselines include FPM−OPT, which uses a smart allocation strategy to reduce the noise of lower order statistics, as discussed in Section 4.3. SF−2 doesn't support joins, so it is not reported.

We use $n_{corp} = 2$ datasets: we fix cardinality $n = 10K$ but vary join key domain size $d \in \{10, 50, 100, 200\}$, then fix $d = 100$ but vary $n \in \{100, 5K, 10K, 50K\}$. The results are shown in Figure 11a and Figure 11b, respectively. FPM, FPM−OPT and APM have low β error, while TPM and SF have high failure rates. FPM−OPT outperforms FPM due to better noise allocation. APM outperforms FPM and FPM−OPT at large d or small n because APM adds noise directly to the aggregated statistics across join keys, resulting in a smaller amount of noise. In contrast, FPM adds noise for each join key value. However, for large n, FPM and FPM−OPT outperform APM because it has high sensitivity due to high join fanouts [39].

Figure 11c and Figure 11d respectively vary the number of datasets and requests: $n_{corp}, n_{req} \in \{1, 5, 10, 50, 100\}$. Both TPM and SF have high failure rates, and FPM−OPT outperforms FPM. FPM and FPM−OPT scale to arbitrary numbers of datasets and requests, while APM is restricted to $n_{corp} \leq 5$ or $n_{req} \leq 10$.

5.2.4 Join Unbiased estimator. Here, we compare the β error of the unbiased estimator proposed in Section 4.1 to the naive estimator over many-to-many joins. We first fix the number of tuples $n = 10K$ but vary join key domain size $d \in \{10, 50, 100, 200\}$, then fix $d = 100$ but vary $n \in \{100, 5K, 10K, 50K\}$, and report the results in Figure 12a and Figure 12b respectively. As d increases, the errors of the unbiased estimator converge to 0, while the biased estimator diverges as it fails to account for many-to-many join. When $n = 100$, the naive estimator achieves similar performance, as each join key has only one tuple (so one-to-one join without bias). However, increasing n introduces duplications and independence (for many-to-many join). The unbiased estimator reduces the noise and performs better than the naive estimator.

6 RELATED WORKS

Dataset search. Traditional data discovery focuses on augmentable (i.e., joinable or unionable) datasets [17, 30], whereas recent dataset search platforms [20, 46, 49, 56] are based on data augmentation *for ML tasks*. However, none addresses privacy concerns.

Differential Privacy for Databases. Differentially private databases can query over multiple tables [39, 43, 67]. They apply DP mechanisms to query results over joins and unions. Notably, join poses a significant DP challenge due to the exponential sensitivity growth along the join path. FPM may offer a solution by decomposing join query into smaller, bounded-sensitivity statistics.

Federated ML. These methods let each untrusted party compute and privatize their local gradients for horizontal [59, 61, 66, 72] or vertical [35, 64] federated ML, which are then combined to train the final model. However, the gradient is specific to training a single model. In contrast, data search repeatedly trains new models to evaluate candidate augmentations, requiring budget splits.

Differentially Private Sufficient Statistics. Previous works use sufficient statistics [38] for generalized linear models and apply perturbations [44] to guarantee DP. For linear regression, sufficient statistics perturbation, particularly with regularization, outperforms other GDP mechanisms including objective perturbation and noisy SGD [14, 65]. However, they only consider ML on a single dataset.

Factorized ML. Factorized ML decomposes ML models into semi-ring queries, designs algebraic operators to combine them, and achieves asymptotically lower time complexity. They support models like ridge regression [58], random forests [36], SVM [41], and factorization machine [57]. None are differentially private. We are the first to apply DP to factorized linear regression. Future work aims to extend *Saibot* to other proxy models like random forests used in prior data search [20]. The challenge lies in the lack of a closed-form solution in random forests, requiring iterative computation of semi-ring aggregates based on tree splits, which are not reusable and costly to privatize. To improve search utility, we plan to explore (1) alternative trust models where requesters trust the platform to lessen noise, and (2) differentially private synopses [60, 71] based on monomial semi-ring to generate synthetic data, allowing the training of arbitrarily complex models as post-processing.

7 CONCLUSIONS

Saibot is a differentially private data search platform that searches large corpora to find datasets to improve ML performance via augmentation. *Saibot* employs FPM, a novel mechanism that privatizes sufficient semi-ring statistics, which can be reused without incurring additional DP cost. In a deep study of linear regression, we propose an unbiased estimator for many-to-many joins, prove parameter bounds under augmentations, and propose an optimization to allocate DP budget better. On a >300 dataset corpus, FPM achieves an $r2$ score (~50−90%) of non-private search, while other mechanisms (TPM, APM, SF) report negligible $r2$ scores <0.02.

REFERENCES

[1] [n. d.]. 2013 - 2018 School ELA REsults. https://data.cityofnewyork.us/Education/2013-2018-School-ELA-REsults/qkpp-pbi8.

[2] [n. d.]. 2013 -2018 School Math Results. https://data.cityofnewyork.us/Education/2013-2018-School-Math-Results/m27t-ht3h.

[3] [n. d.]. 2013-16 School ELA Data Files By Grade - Gender. https://data.cityofnewyork.us/Education/2013-16-School-ELA-Data-Files-By-Grade-Gender/436j-ja87.

[4] [n. d.]. 2014-15 To 2016-17 School- Level NYC Regents Report For All Variables. https://data.cityofnewyork.us/Education/2014-15-To-2016-17-School-Level-NYC-Regents-Report/csps-2ne9/.

[5] [n. d.]. 2016-2017 Graduation Outcomes School. https://data.cityofnewyork.us/Education/2016-2017-Graduation-Outcomes-School/nb39-jx2v.

[6] [n. d.]. California Consumer Privacy Act. https://oag.ca.gov/privacy/ccpa.

[7] [n. d.]. The Family Educational Rights and Privacy Act (FERPA). https://studentprivacy.ed.gov/.

[8] [n. d.]. Health Insurance Portability and Accountability Act of 1996 (HIPAA). https://www.cdc.gov/phlp/publications/topic/hipaa.html.

[9] 2018. 2018 reform of EU data protection rules. https://ec.europa.eu/commission/sites/beta-political/files/data-protection-factsheet-changes_en.pdf.

[10] 2022. CMS Data. https://data.cms.gov/.

[11] 2022. NYC Open Data. https://opendata.cityofnewyork.us/.

[12] 2023. (Technical Report) Saibot: A Differentially Private Data Search Platform. https://anonymous.4open.science/r/Saibot-B387/tech/sailbot_tech.pdf.

[13] Mahmoud Abo Khamis, Hung Q Ngo, and Atri Rudra. 2016. FAQ: questions asked frequently. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*. 13–28.

[14] Daniel Alabi and Salil Vadhan. 2022. Hypothesis Testing for Differentially Private Linear Regression. In *Advances in Neural Information Processing Systems*, Vol. 35. 14196–14209. https://proceedings.neurips.cc/paper_files/paper/2022/file/5bc3356e0fa1753fff7e8d6628e71b22-Paper-Conference.pdf

[15] Daniel Gbenga Alabi. 2022. *The Algorithmic Foundations of Private Computational Social Science*. Ph. D. Dissertation. Harvard University.

[16] Avrim Blum, Katrina Ligett, and Aaron Roth. 2013. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)* 60, 2 (2013), 1–25.

[17] Sonia Castelo, Rémi Rampin, Aécio Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. 2021. Auctus: a dataset search engine for data discovery and augmentation. *Proceedings of the VLDB Endowment* 14, 12 (2021), 2791–2794.

[18] Patricio Cerda and Gaël Varoquaux. 2020. Encoding high-cardinality string categorical variables. *IEEE Transactions on Knowledge and Data Engineering* 34, 3 (2020), 1164–1176.

[19] Xiaojun Chen, Guowen Yuan, Feiping Nie, and Joshua Zhexue Huang. 2017. Semi-supervised Feature Selection via Rescaled Linear Regression.. In *IJCAI*, Vol. 2017. 1525–1531.

[20] Nadiia Chepurko, Ryan Marcus, Emanuel Zgraggen, Raul Castro Fernandez, Tim Kraska, and David Karger. 2020. ARDA: automatic relational data augmentation for machine learning. *arXiv preprint arXiv:2003.09758* (2020).

[21] Bolin Ding, Janardhan Kulkarni, and Sergey Yekhanin. 2017. Collecting telemetry data privately. *Advances in Neural Information Processing Systems* 30 (2017).

[22] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 486–503.

[23] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. Springer, 265–284.

[24] Cynthia Dwork, Guy N Rothblum, and Salil Vadhan. 2010. Boosting and differential privacy. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. IEEE, 51–60.

[25] Cynthia Dwork, Adam Smith, Thomas Steinke, and Jonathan Ullman. 2017. Exposed! a survey of attacks on private data. *Annual Review of Statistics and Its Application* 4 (2017), 61–84.

[26] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. 2014. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 11–20.

[27] Úlfar Erlingsson, Vitaly Feldman, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Abhradeep Thakurta. 2019. Amplification by shuffling: From local to central differential privacy via anonymity. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2468–2479.

[28] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 1054–1067.

[29] Vitaly Feldman, Audra McMillan, and Kunal Talwar. 2022. Hiding among the clones: A simple and nearly optimal analysis of privacy amplification by shuffling. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE, 954–964.

[30] Raul Castro Fernandez, Ziawasch Abedjan, Famien Koko, Gina Yuan, Samuel Madden, and Michael Stonebraker. 2018. Aurum: A data discovery system. In *2018 IEEE 34th International Conference on Data Engineering (ICDE)*. IEEE, 1001–1012.

[31] Richard Frank, Flavia Moser, and Martin Ester. 2007. A method for multi-relational classification using single and multi-feature aggregation functions. In *Knowledge Discovery in Databases: PKDD 2007: 11th European Conference on Principles and Practice of Knowledge Discovery in Databases, Warsaw, Poland, September 17-21, 2007. Proceedings 11*. Springer, 430–437.

[32] Todd J Green, Grigoris Karvounarakis, and Val Tannen. 2007. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 31–40.

[33] Hongyu Guo and Herna L Viktor. 2008. Multirelational classification: a multiple view approach. *Knowledge and Information Systems* 17 (2008), 287–312.

[34] Moritz Hardt and Guy N Rothblum. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st annual symposium on foundations of computer science*. IEEE, 61–70.

[35] Stephen Hardy, Wilko Henecka, Hamish Ivey-Law, Richard Nock, Giorgio Patrini, Guillaume Smith, and Brian Thorne. 2017. Private federated learning on vertically partitioned data via entity resolution and additively homomorphic encryption. *arXiv preprint arXiv:1711.10677* (2017).

[36] Zezhou Huang, Rathijit Sen, Jiaxiang Liu, and Eugene Wu. 2022. (Technical Report) JoinBoost: In-Database Tree-Models In Action. https://anonymous.4open.science/r/JoinBoost-FBC4/technical/JoinBoost_tech.pdf.

[37] Zezhou Huang, Pranav Subramaniam, Raul Castro Fernandez, and Eugene Wu. 2023. Kitana: Efficient Data Augmentation Search for AutoML. arXiv:2305.10419 [cs.DB]

[38] Jonathan Huggins, Ryan P Adams, and Tamara Broderick. 2017. PASS-GLM: polynomial approximate sufficient statistics for scalable Bayesian GLM inference. *Advances in Neural Information Processing Systems* 30 (2017).

[39] Noah Johnson, Joseph P Near, and Dawn Song. 2018. Towards practical differential privacy for SQL queries. *Proceedings of the VLDB Endowment* 11, 5 (2018), 526–539.

[40] Peter Kairouz, Keith Bonawitz, and Daniel Ramage. 2016. Discrete distribution estimation under local privacy. In *International Conference on Machine Learning*. PMLR, 2436–2444.

[41] Mahmoud Abo Khamis, Ryan R Curtin, Benjamin Moseley, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2020. Functional Aggregate Queries with Additive Inequalities. *ACM Transactions on Database Systems (TODS)* 45, 4 (2020), 1–41.

[42] Mahmoud Abo Khamis, Hung Q Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. 2018. AC/DC: in-database learning thunderstruck. In *Proceedings of the second workshop on data management for end-to-end machine learning*. 1–10.

[43] Ios Kotsogiannis, Yuchao Tao, Xi He, Maryam Fanaeepour, Ashwin Machanavajjhala, Michael Hay, and Gerome Miklau. 2019. Privatesql: a differentially private sql query engine. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1371–1384.

[44] Tejas Kulkarni, Joonas Jälkö, Antti Koskela, Samuel Kaski, and Antti Honkela. 2021. Differentially private bayesian inference for generalized linear models. In *International Conference on Machine Learning*. PMLR, 5838–5849.

[45] Jaewoo Lee and Chris Clifton. 2011. How much is enough? choosing ε for differential privacy. In *Information Security: 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings 14*. Springer, 325–340.

[46] Yifan Li, Xiaohui Yu, and Nick Koudas. 2021. Data acquisition for improving machine learning models. *Proceedings of the VLDB Endowment* 14, 10 (2021), 1832–1844.

[47] Andrzej Maćkiewicz and Waldemar Ratajczak. 1993. Principal components analysis (PCA). *Computers & Geosciences* 19, 3 (1993), 303–342.

[48] Julie Moeyersoms and David Martens. 2015. Including high-cardinality attributes in predictive models: A case study in churn prediction in the energy sector. *Decision support systems* 72 (2015), 72–81.

[49] Fatemeh Nargesian, Abolfazl Asudeh, and HV Jagadish. 2022. Responsible Data Integration: Next-generation Challenges. In *Proceedings of the 2022 International Conference on Management of Data*. 2458–2464.

[50] Joseph P Near, Xi He, et al. 2021. Differential Privacy for Databases. *Foundations and Trends® in Databases* 11, 2 (2021), 109–225.

[51] Ted North. 2019. Google, Fitbit, and the Sale of Our Private Health Data. https://www.fitbit.com/global/us/home.

[52] Dan Olteanu and Jakub Závodnỳ. 2015. Size bounds for factorised representations of query results. *ACM Transactions on Database Systems (TODS)* 40, 1 (2015), 1–44.

[53] Judea Pearl. 2022. Comment: understanding Simpson's paradox. In *Probabilistic and Causal Inference: The Works of Judea Pearl*. 399–412.

[54] Wahbeh Qardaji, Weining Yang, and Ninghui Li. 2014. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. 1435–1446.

[55] Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI. In *proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[56] Aécio Santos, Aline Bessa, Christopher Musco, and Juliana Freire. 2022. A sketch-based index for correlated dataset search. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2928–2941.

[57] Maximilian Schleich, Dan Olteanu, Mahmoud Abo Khamis, Hung Q Ngo, and XuanLong Nguyen. 2019. A layered aggregate engine for analytics workloads. In *Proceedings of the 2019 International Conference on Management of Data*. 1642–1659.

[58] Maximilian Schleich, Dan Olteanu, and Radu Ciucanu. 2016. Learning linear regression models over factorized joins. In *Proceedings of the 2016 International Conference on Management of Data*. 3–18.

[59] Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1310–1321.

[60] Uthaipon Tao Tantipongpipat, Chris Waites, Digvijay Boob, Amaresh Ankit Siva, and Rachel Cummings. 2021. Differentially private synthetic mixed-type data generation for unsupervised learning. *Intelligent Decision Technologies* 15, 4 (2021), 779–807.

[61] Stacey Truex, Ling Liu, Ka-Ho Chow, Mehmet Emre Gursoy, and Wenqi Wei. 2020. LDP-Fed: Federated learning with local differential privacy. In *Proceedings of the Third ACM International Workshop on Edge Systems, Analytics and Networking*. 61–66.

[62] Haleh Vafaie, Ibrahim F Imam, et al. 1994. Feature selection methods: genetic algorithms vs. greedy-like search. In *Proceedings of the international conference on fuzzy and intelligent control systems*, Vol. 51. 28.

[63] Roman Vershynin. 2018. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press.

[64] Chang Wang, Jian Liang, Mingkai Huang, Bing Bai, Kun Bai, and Hao Li. 2020. Hybrid differentially private federated learning on vertically partitioned data. *arXiv preprint arXiv:2009.02763* (2020).

[65] Yu-Xiang Wang. 2018. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. *arXiv preprint arXiv:1803.02596* (2018).

[66] Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. 2020. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3454–3469.

[67] Royce J Wilson, Celia Yuxin Zhang, William Lam, Damien Desfontaines, Daniel Simmons-Marengo, and Bryant Gipson. 2019. Differentially private SQL with bounded user contribution. *arXiv preprint arXiv:1909.01917* (2019).

[68] Genqiang Wu, Xianyao Xia, and Yeping He. 2017. Achieving Dalenius' Goal of Data Privacy with Practical Assumptions. *arXiv preprint arXiv:1703.07474* (2017).

[69] Jia Xu, Zhenjie Zhang, Xiaokui Xiao, Yin Yang, Ge Yu, and Marianne Winslett. 2013. Differentially private histogram publication. *The VLDB journal* 22 (2013), 797–822.

[70] Mengmeng Yang, Lingjuan Lyu, Jun Zhao, Tianqing Zhu, and Kwok-Yan Lam. 2020. Local differential privacy and its applications: A comprehensive survey. *arXiv preprint arXiv:2008.03686* (2020).

[71] Jinsung Yoon, James Jordon, and Mihaela van der Schaar. 2019. PATE-GAN: Generating Synthetic Data with Differential Privacy Guarantees. In *International Conference on Learning Representations*. https://openreview.net/forum?id=S1zk9iRqF7

[72] Yang Zhao, Jun Zhao, Mengmeng Yang, Teng Wang, Ning Wang, Lingjuan Lyu, Dusit Niyato, and Kwok-Yan Lam. 2020. Local differential privacy-based federated learning for internet of things. *IEEE Internet of Things Journal* 8, 11 (2020), 8836–8853.

A FPM SENSITIVITY

For union, query $q_u : \mathcal{D}^n \longrightarrow \mathcal{S}$ where datasets in $\mathcal{D}^n$ contains m features and $\mathcal{S} = \{v \mid v \in \mathbb{R}^{m^i},\ i \in \{0, \ldots, k\}\}$. q_u returns a set of vectors $s \in \mathcal{S}$ containing the sum of k-order monomial semiring across all tuples. For analysis convenience, we will overload the notation a bit and treat s as a single vector by concatenating $\{v\}_{v \in s}$. Let $t_1, t_2 \in \mathcal{D}^n$ and t_1^i, t_2^i be vectors of i-order monomial with respect to $t_1[f_1], \ldots, t_1[f_m]$ and $t_2[f_1], \ldots, t_2[f_m]$. Let $\sigma(k)$ denote the set of series $[k_1, \ldots, k_m]$ such that $k_i \in \mathbb{N}$ and $\sum k_i = k$. The squared distance between t_1^k and t_2^k can be computed as

$$\sum_{\substack{[k_1,\ldots,k_m] \\ \in\sigma(k)}} (\prod_{i=1}^{m} t_1[f_i]^{k_i} - \prod_{i=1}^{m} t_2[f_i]^{k_i})^2$$

$$\leq \sum_{\substack{[k_1,\ldots,k_m] \\ \in\sigma(k)}} \binom{k}{k_1, \ldots, k_m} (\prod_{i=1}^{m} t_1[f_i]^{k_i} - \prod_{i=1}^{m} t_2[f_i]^{k_i})^2$$

By multinomial theorem, we may rewrite the last equation as

$$(\sum_{i=1}^{m} t_1[f_i]^2)^k + (\sum_{i=1}^{m} t_2[f_i]^2)^k - 2(\sum_{i=1}^{m} t_1[f_i] t_2[f_i])^k$$

$$\leq 2B^{2k} - 2(\sum_{i=1}^{m} t_1[f_i] t_2[f_i])^k$$

That is, when k is even, the latter term is strictly positive. Hence $\|t_1^k - t_2^k\|^2 \leq 2B^{2k}$. Let $D_1, D_2 \in \mathcal{D}^n$ be two neighbouring datasets differ in one tuple, t_1 and t_2. The sensitivity of $q_u(\cdot)$ can be computed as

$$\Delta_{q_u} = \max \|q_u(D_1) - q_u(D_2)\| \leq \sqrt{\sum_{i=1}^{k} 1_{2\mathbb{Z}+1}(i) 4B^{2i} + 1_{2\mathbb{Z}}(i) 2B^{2i}}$$

For join, we inherit the notations from the union case and let ℓ be the number of join keys. $q_j : \mathcal{D}^n \longrightarrow \mathcal{S}$ returns a set of vectors $s \in \mathcal{S}$ where s concatenates vectors returned by q_u on each partition of tuples for each join key. Consider two cases: (1) t_1 and t_2 have the same join key (2) t_1 and t_2 have different join keys. In the former case,

$$\Delta_{q_j} = \max \|q_j(D_1) - q_j(D_2)\|_2 \leq \sqrt{\sum_{i=1}^{k} 1_{2\mathbb{Z}+1}(i) 4B^{2i} + 1_{2\mathbb{Z}}(i) 2B^{2i}}$$

In the latter case,

$$\Delta_{q_j} = \max \|q_j(D_1) - q_j(D_2)\|_2 = \max \sqrt{\sum_{i}^{k} \|t_1^i\|^2 + \sum_{i}^{k} \|t_2^i\|^2}$$

$$\leq \sqrt{\max \sum_{i}^{k} \|t_1^i\|^2 + \max \sum_{i}^{k} \|t_2^i\|^2} \leq \sqrt{2 \sum_{i=0}^{k} B^{2i}}$$

Table 1: Notation

Notation	Description
R_i	relations of providers/requesters.
n	size of each relation.
$J, Dom(J), d$	join key, domain of join key, domain size.
B	the ℓ_2 distance upper bound of each tuple in each relation.
$\widehat{\sigma_x^2}, \widehat{\sigma_{xy}^2}$	the empirical estimation of the variance and covariance.
$\widetilde{\sigma_x^2}, \widetilde{\sigma_{xy}^2}$	the privatized empirical estimation of the variance and covariance.
B_1, B_2	$1-p$ confidence bound on $\lvert\widehat{\sigma_x^2} - \widetilde{\sigma_x^2}\rvert$ and $\lvert\widehat{\sigma_{xy}^2} - \widetilde{\sigma_{xy}^2}\rvert$.

Hence, $\Delta_{q_j} = \max(\sqrt{\sum_{i=1}^{k} 1_{2\mathbb{Z}+1}(i) 4B^{2i} + 1_{2\mathbb{Z}}(i) 2B^{2i}}, \sqrt{2\sum_{i=0}^{k} B^{2i}})$

B ERROR ANALYSIS

For a single data provider, with relation R where $|R| = n$ and i be any integer from 1 to k. LDP computes $[f, f^2, \ldots, f^k]$ for each tuple and adds noise to each of them. By similar analysis to that of FPM, LDP's sensitivity is the same as Δ in FPM for both union and join. Hence, for each tuple, t, from R, $t[\widetilde{f^i}] \sim t[f^i] + \mathcal{N}(0, (2\ln(1.25/\delta)\Delta^2/\epsilon^2)$. The empirical expectation of f^i can be computed as

$$\widetilde{f^i_{\text{LDP}}} = \frac{1}{n}\left(\sum_{t \in R} t[f^i]\right) + e_i \quad e_i \sim \mathcal{N}(0, 2\ln(1.25/\delta)\Delta^2/n\epsilon^2)$$

Putting everything together, and by the assumption that k is a small constant, we have

$$E[\|s'_{\text{LDP}} - \hat{s}\|] = E\left[\sqrt{\sum_{i=1}^{k}\left(\widetilde{f^i_{\text{LDP}}} - \frac{1}{n}\left(\sum_{t\in R} t[f^i]\right)\right)^2}\right]$$

$$\leq \sqrt{E\left[\sum_{i=1}^{k}\left(\widetilde{f^i_{\text{LDP}}} - \frac{1}{n}\left(\sum_{t\in R} t[f^i]\right)\right)^2\right]}$$

$$= \sqrt{\sum_{i=1}^{k} E[e_i^2]} = O(\Delta/\sqrt{n}\epsilon)$$

For FPM, the only difference is that

$$\widetilde{f^i_{\text{FPM}}} \sim \frac{1}{n}\left(\sum_{t \in R} t[f^i]\right) + e_i \quad e_i \sim \mathcal{N}(0, 2\ln(1.25/\delta)\Delta^2/n^2\epsilon^2)$$

Following the same line of derivation,

$$E[\|\widetilde{s_{\text{FPM}}} - \hat{s}\|] = O(\Delta/n\epsilon)$$

However, GDP needs to account for any possible combination of a single buyer and a subset of sellers, where each party's privacy needs to be protected. Specifically, each buyer appears in $2^{n_{corp}} - 1$ combinations, since each buyer requires at least one seller. On the other hand, for a fixed buyer, each seller is involved in $2^{n_{corp}-1}$ combinations. Hence, each seller will appear in $n_{req}2^{n_{corp}-1}$ combinations in total. Because each seller and buyer have privacy budget

(ϵ, δ), in order to provide privacy guarantees for each party in any combination, the amount of privacy budget spent on perturbing pre-normalized s is $\epsilon' = \min(\epsilon/(2^{n_{corp}} - 1), \epsilon/n_{req}2^{n_{corp}-1})$ and $\delta' = \min(\delta/(2^{n_{corp}} - 1), \delta/n_{req}2^{n_{corp}-1})$

$$\widetilde{f^i_{\texttt{GDP}}} = \frac{1}{n}\left(\sum_{t\in R} t[f^i]\right) + e_i \quad e_i \sim \mathcal{N}(0, 2\ln(1.25/\delta')\Delta^2/n^2\epsilon'^2)$$

Based on the same line of analysis above

$$E[\|\widetilde{s_{\texttt{GDP}}} - \hat{s}\|] = O(n_{req}2^{n_{corp}-1}\Delta/n\epsilon)$$

Now we consider SF−1, based on [27], it suffice to guarantee $\epsilon/\sqrt{n}$-DP for local responses to achieve (ϵ, δ)-DP from the central's perspective, where each tuple t in R satisfies $t[\widetilde{f^i}] \sim t[f^i] + \text{Lap}(\Delta/\sqrt{n}\epsilon)$. Then we have

$$\widetilde{f^i_{\texttt{SF-1}}} = \frac{1}{n}\sum_{t\in R}(t[f^i] + e_t) \quad e_t \sim \text{Lap}(\Delta/\sqrt{n}\epsilon)$$

Then, we have

$$E[\|\widetilde{\texttt{SF-1}} - \hat{s}\|] = E\left[\sqrt{\sum_{i=1}^{k}\left(\widetilde{f^i_{\texttt{SF-1}}} - \frac{1}{n}\left(\sum_{t\in R} t[f^i]\right)\right)^2}\right] \leq \sqrt{\sum_{i=1}^{k} E\left[\left(\frac{1}{n}\sum_{t\in R} e_t\right)^2\right]}$$

Since $E\left[\left(\frac{1}{n}\sum_{t\in R} e_t\right)\right] = 0$, it follows that

$$E\left[\left(\frac{1}{n}\sum_{t\in R} e_t\right)^2\right] = Var\left(\frac{1}{n}\sum_{t\in R} e_t\right) = \frac{\Delta^2}{n^2\epsilon^2}$$

Substituting back to the equation, and based on assumption that k is small, we have

$$E[\|\widetilde{s_{\texttt{SF-1}}} - \hat{s}\|] = O(\Delta/n\epsilon)$$

For SDP-2, just like GDP, it also needs to account for all possible combination of a single buyer and any subsets of sellers in the centralized shuffler. However, the differences are that SF−2 allows each combination's privacy guarantee to be amplified by an amount of $O(\sqrt{n})$, and that SF−2 draw random noises from Laplace distribution instead of Gaussian distribution. That is,

$$\widetilde{f^i_{\texttt{SF-2}}} = \frac{1}{n}\sum_{t\in R}(t[f^i] + e_t) \quad e_t \sim \text{Lap}(\Delta/\sqrt{n}\epsilon')$$

Hence, the expected utility can be computed following the same line of derivation of SF−1. That is

$$E[\|\widetilde{s_{\texttt{SF-2}}} - \hat{s}\|] = O(\Delta/n\epsilon') = O(n_{req}2^{n_{corp}-1}\Delta/n\epsilon)$$

C UNBIASED PROOF

We make the simplifying assumption that J is uniformly distributed: if $d = |dom(J)|$, then each $j \in J$ appears n/d times in R. Moreover, the projection operator π will not remove duplicates in R so $|\pi_{J,f_1}(R)| = |\pi_{J,f_2}(R)| = n$.

Proposition 8 (Expected s over $R_\bowtie$). *Assume that $R_\bowtie$ is the population. For any other 1,2-order monomial p,*

$$E[p] = s[p]/s[c]$$

where c is the count (0-order monomial). Then $E[p]$ is the expected s over $R_\bowtie$.

Proposition 9 (Unbiased Estimator of s over R).

$$\widehat{E[p]} = \begin{cases} f_1 f_2 = \frac{1-n}{1-d}\frac{s[f_1f_2]}{s[c]} + \frac{n-d}{1-d}\frac{s[f_1]}{s[c]}\frac{s[f_2]}{s[c]} \\ \qquad\qquad \text{for } f_1 \in F_1, f_2 \in F_2 \\ p = s[p]/s[c] \quad \text{for any other monomial } p \end{cases}$$

$\hat{s}$ is an unbiased estimator of s.

Proof. We demonstrate that, for any 1,2-order monomial where features are from the same relation, $E[s[p]/s[c]] = E[p]$.

$$\begin{aligned} s[c] &= n \cdot n/d \\ E[s[f]/s[c]] &= E[(\sum_{t\in R} t[f] \cdot n/d)/(n \cdot n/d)] = \sum_{t\in R} E[t[f]]/n = E[f] \\ E[s[f_1f_2]/s[c]] &= E[(\sum_{t\in R} t[f_1] \cdot t[f_2] \cdot n/d)/(n \cdot n/d)] \\ &= \sum_{t\in R} E[t[f_1] \cdot t[f_2]]/n = E[f_1f_2] \end{aligned}$$

The first equality is because for each join key, the cartesian product is computed, leading to duplication of tuples with the same join key in both tables by n/d times. The count is also increased by n/d, thus resulting in the equality $s[p]/s[c] = E[p]$.

However, this equality does not hold for the f_1f_2, where f_1 and f_2 are from different relations. In this case, f_1 from R_1 is paired with all f_2 from R_2 with the same join key, but the information about which f_2 is paired with f_1 in original R is lost. Nonetheless, we can still estimate $E[f_1f_2]$ by exploiting the covariance across groups.

We first analyze $E[f_1f_2]$ for a single join key value j. We use notation s^j to denote the monomial semi-ring for the join key value k. Consider random variable of the average:

$$\begin{aligned} s^j_1 &= s^j[f_1]/s^j[c] = \left(\sum_{t\in\sigma_j(R)} t[f_1] \cdot n/d\right)/(n/d)^2 = \sum_{t\in\sigma_j(R)} \frac{t[f_1]}{n/d} \\ s^j_2 &= s^j[f_2]/s^j[c] = \left(\sum_{t\in\sigma_j(R)} t[f_2] \cdot n/d\right)/(n/d)^2 = \sum_{t\in\sigma_j(R)} \frac{t[f_2]}{n/d} \end{aligned}$$

s^j_1 and s^j_2 can be understood as the mean of f_1 and f_2 from the sample $\sigma_j(R)$. It is obvious that $E[s^j_1] = E[f_1]$ and $E[s^j_2] = E[f_2]$.

From the definition of covariance, we have:

$$\begin{aligned} E[s^j_1 s^j_2] &= cov(s^j_1, s^j_2) + E[s^j_1]E[s^j_2] \\ &= cov\left(\sum_{t\in\sigma_j(R)} \frac{t[f_1]}{n/d}, \sum_{t\in\sigma_j(R)} \frac{t[f_2]}{n/d}\right) + E[f_1]E[f_2] \end{aligned}$$

We next compute the cov:

$$cov\left(\sum_{t\in\sigma_j(R)}\frac{t[f_1]}{n/d},\sum_{t\in\sigma_j(R)}\frac{t[f_2]}{n/d}\right)=\frac{d^2}{n^2}\sum_{\substack{t_1\in\sigma_j(R)\\t_2\in\sigma_j(R)}}cov(t_1[f_1],t_2[f_2])$$
$$=\frac{d^2}{n^2}\sum_{t\in\sigma_j(R)}cov(t[f_1],t[f_2])$$
$$=\frac{d}{n}cov(f_1,f_2)$$

The first line is by the property of covariance and the second line is by the independence between tuples. Therefore,

$$E[s_1^j s_2^j]=\frac{d}{n}cov(f_1,f_2)+E[f_1]E[f_2]$$

Next, consider the random variables across join keys:

$$s_1=s[f_1]/s[c]=\sum_{t\in R}t[f_1]/n$$
$$s_2=s[f_2]/s[c]=\sum_{t\in R}t[f_2]/n$$
$$s_{1,2}=s[f_1f_2]/s[c]=\sum_{j\in dom(J)}s_1^j\cdot s_2^j/d$$

where s_1 and s_2 are the average across join keys. $s_{1,2}$ is the average products across join keys. It is obvious that $E[s_1]=E[f_1], E[s_2]=E[f_2]$. We next study $E[s_1s_2]$ and $E[s_{1,2}]$:

$$E[s_1s_2]=cov(s_1,s_2)+E[s_1^j]E[s_2^j]$$
$$=cov\left(\sum_{t\in R}t[f_1]/n,\sum_{t\in R}t[f_2]/n\right)+E[f_1]E[f_2]$$

Similar as before,

$$cov\left(\sum_{t\in R}t[f_1]/n,\sum_{t\in R}t[f_2]/n\right)=\frac{1}{n^2}\sum_{\substack{t_1\in R\\t_2\in R}}cov(t_1[f_1],t_2[f_2])$$
$$=\frac{1}{n^2}\sum_{t\in R}cov(t[f_1],t[f_2])=\frac{1}{n}cov(f_1,f_2)$$

Therefore:

$$E[s_1s_2]=\frac{1}{n}cov(f_1,f_2)+E[f_1]E[f_2]$$

Finally,

$$E[s_{1,2}]=\sum_{j\in dom(J)}E[s_1^j\cdot s_2^j]/d=\frac{d}{n}cov(f_1,f_2)+E[f_1]E[f_2]$$

Putting everything together, we show that $\frac{1-n}{1-d}s_{1,2}+\frac{n-d}{1-d}s_1\cdot s_2$ is an unbiased estimator of $E[f_1f_2]$:

$$E[\frac{1-n}{1-d}s_{1,2}+\frac{n-d}{1-d}s_1\cdot s_2]=\frac{1-n}{1-d}E[s_{1,2}]+\frac{n-d}{1-d}E[s_1s_2]$$
$$=\frac{1-n}{1-d}(\frac{d}{n}cov(f_1,f_2)+E[f_1]E[f_2])+$$
$$\frac{n-d}{1-d}(\frac{1}{n}cov(f_1,f_2)+E[f_1]E[f_2])$$
$$=cov(f_1,f_2)+E[f_1]E[f_2]=E[f_1f_2]$$

The first line is by the linearity of expectation, and the last line is by the definition of covariance.

□

D CONFIDENCE BOUND OF LINEAR REGRESSION

Let $\sigma=\sqrt{2\ln(1.25/\delta)}\Delta/\epsilon$ where $\Delta=O(B^2)$. We are interested in $n,B\to\infty$ and $\epsilon,\delta,p\to 0$ in our analysis. Hence $\sigma=O\left(\frac{B^2\sqrt{\ln(1/\delta)}}{\epsilon}\right)$. The privatized empirical expectation of the moments are defined as $\widetilde{E[X]}=\widehat{E[X]}+e_1, \widetilde{E[X^2]}=\widehat{E[X^2]}+e_2, \widetilde{E[XY]}=\widehat{E[XY]}+e_3$ and $\widetilde{E[Y]}=\widehat{E[Y]}+e_4$. Then, we have $e_1,e_2,e_3,e_4\sim\mathcal{N}(0,\sigma^2/n^2)$

Lemma D.1 (High-probability bound on $\widetilde{\sigma_x}$). *Given $\widehat{\sigma_x^2}=\widehat{E[X^2]}-\widehat{E[X]}^2$ and $\widetilde{\sigma_x^2}=\widetilde{E[X^2]}-\widetilde{E[X]}^2$, with probability at least $1-p$, $|\widehat{\sigma_x^2}-\widetilde{\sigma_x^2}|=O(B_1)$ where*

$$B_1=\frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}$$

Proof. By assumption that each tuple's ℓ_2 norm is bounded by B, each feature must also be bounded by B. Based on Gaussian tail bound, with probability at least $1-p/4$, $|e_i|\le\sigma\sqrt{2\ln(8/p)}/n$.

$$|\widehat{\sigma_x^2}-\widetilde{\sigma_x^2}|=|e_1-2e_2\sum x/n-e_2^2|$$
$$\le|e_1|+|2e_2\sum x/n|+|e_2^2|$$
$$\le\frac{\sigma\sqrt{2\ln(8/p)}}{n}\left(1+2B+\frac{\sigma\sqrt{2\ln(8/p)}}{n}\right)$$
$$=O\left(\frac{B^2\sqrt{\ln(1/\delta)\ln(1/p)}}{\epsilon n}+\frac{B^2\ln(1/\delta)\ln(1/p)}{\epsilon^2n^2}\right)$$
$$=O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2n}\right)$$

□

Similarly, D.1 can be used to derive the high probability bound on $\widetilde{\sigma_{xy}^2}$, that is

$$|\widehat{\sigma_{xy}^2}-\widetilde{\sigma_{xy}^2}|\le B_2=O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2n}\right)$$

Since the condition to satisfy both bounds coincide, with probability at least $1-p$, $|\widehat{\sigma_x^2}-\widetilde{\sigma_x^2}|\le B_1$ and $|\widehat{\sigma_{xy}^2}-\widetilde{\sigma_{xy}^2}|\le B_2$.

Let

$$\tau_1=B_1/\widehat{\sigma_x^2}=O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2n\widehat{\sigma_x^2}}\right)=\tau_2$$

When $\tau_1,\tau_2<1$ and with probability at least $1-p$, we may prove 4.1 as

Proof.

$$
\begin{aligned}
|\hat{\beta}_x - \tilde{\beta}_x| &= \left|\frac{\widehat{\sigma^2_{xy}}}{\widehat{\sigma^2_x}} - \frac{\widetilde{\sigma^2_{xy}}}{\widetilde{\sigma^2_x}}\right| = \left|\frac{\widehat{\sigma^2_{xy}}}{\widehat{\sigma^2_x}} - \frac{\widetilde{\sigma^2_{xy}}}{\widehat{\sigma^2_x}} + \frac{\widetilde{\sigma^2_{xy}}}{\widehat{\sigma^2_x}} - \frac{\widetilde{\sigma^2_{xy}}}{\widetilde{\sigma^2_x}}\right| \\
&\leq \frac{|\widetilde{\sigma^2_{xy}} - \widehat{\sigma^2_{xy}}|}{\widehat{\sigma^2_x}} + \widetilde{\sigma^2_{xy}} \cdot \left|\widetilde{\sigma^2_x}^{-1} - \widehat{\sigma^2_x}^{-1}\right| \\
&= \frac{|\widetilde{\sigma^2_{xy}} - \widehat{\sigma^2_{xy}}|}{\widehat{\sigma^2_x}} + \widetilde{\sigma^2_{xy}} \cdot \frac{|\widetilde{\sigma^2_x} - \widehat{\sigma^2_x}|}{\widetilde{\sigma^2_x}\widehat{\sigma^2_x}} \\
&\leq \tau_2 + (\widehat{\sigma^2_{xy}} + \tau_2 \cdot \widehat{\sigma^2_x}) \frac{\tau_1 \widehat{\sigma^2_x}}{(1-\tau_1)(\widehat{\sigma^2_x})^2} \\
&= \tau_2 + \frac{\tau_1}{1-\tau_1}\left(\frac{\widehat{\sigma^2_{xy}}}{\widehat{\sigma^2_x}} + \tau_2\right) = \tau_2 + \frac{\tau_1}{1-\tau_1}\left(\hat{\beta}_x + \tau_2\right)
\end{aligned}
$$

□

Extension to Factorized ML. The confidence bounds can be extended for factorized ML. The difference boils down to B_1, and the rest are the same. For union, let $R = R_1 \cup R_2 ... \cup R_k$ where $|R_i| = n$ and $k \to \infty$. Then, for $e_i \sim \mathcal{N}(0, \sigma^2)$,

$$
\widetilde{E[x^2]} = \frac{\sum_i^k (\sum x^2 + e_i)}{kn} \sim \widehat{E[X^2]} + \mathcal{N}(0, \sigma^2/kn^2)
$$

Therefore, with probability at least $1 - p/4$, $|\widetilde{E[X^2]} - \widehat{E[X^2]}| \leq \sigma\sqrt{2\ln(8/p)}/\sqrt{k}n = O\left(\frac{B^2\sqrt{\ln(1/\delta)\ln(1/p)}}{\epsilon\sqrt{k}n}\right)$ (same for all other 3 moments $E[X], E[Y], E[XY]$), by minor changes in D.1, yielding new bounds on τ_1 and τ_2 as

$$
\begin{aligned}
\tau_1 = B_1/\widehat{\sigma_x^2} &= O\left(\frac{B^2\sqrt{\ln(1/\delta)\ln(1/p)}}{\epsilon\sqrt{k}n\widehat{\sigma^2_x}} + \frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2 kn^2\widehat{\sigma^2_x}}\right) \\
&= O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\epsilon^2\sqrt{k}n\widehat{\sigma^2_x}}\right) = \tau_2
\end{aligned}
$$

For join, consider $R[x, y, J] = R_1[x, J] \bowtie R_2[y, J]$ and $d = |dom(J)|$ where $d \to n$. In contrast to union, there is additional noise added to the zero-th moment of each join key. i.e. the count of tuples within each join key. To avoid the scenario where this number is non-positive, an additional assumption is required [65] that the noise is bounded by $o(n/d)$. Note that in the unbiased estimation, the privatized $s[c]$ is computed as

$$
\begin{aligned}
\widetilde{s[c]} &= \sum_{i\in J}(n/d + o(n/d))(n/d + o(n/d)) \\
&= \sum_{i\in J}(n/d + o(n/d))^2 \\
&= d(n/d + o(n/d))^2
\end{aligned}
$$

Then, for $e_{i,1}, e_{i,2}, e_{i,3} \sim \mathcal{N}(0, \sigma^2)$ defined as the Gaussian noise added to $\sum_{t\in R_1.i} x^2, \sum_{t\in R_1.i} x, \sum_{t\in R_2.i} y$ for each join key $i \in J$, with probability at least $1-p/4$, $\sum_{i\in J} e_{i,j} \sim \mathcal{N}(0, d\sigma^2)$ and $\sum_{i\in J} e_{i,j} \leq \sigma\sqrt{2d\ln(8/p)} = O\left(\frac{B^2\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon}\right)$ for $j = \{1, 2, 3\}$

$$
\begin{aligned}
\widehat{E[X^2]} &= \frac{\sum_{i\in J}(\sum_{t\in R_1.i} x^2)n/d}{n^2/d} = \frac{\sum x^2}{n} \\
\widetilde{E[X^2]} &= \frac{\sum_{i\in J}\left((\sum_{t\in R_1.i} x^2) + e_{i,1}\right)(n/d + o(n/d))}{\sum_{j\in J}(n/d + o(n/d))(n/d + o(n/d))}
\end{aligned}
$$

By expanding $\widetilde{E[X^2]}$, we have

$$
\begin{aligned}
\widetilde{E[X^2]} &= \frac{(n/d)\sum x^2 + (n/d)\cdot\sum_{i\in J} e_{i,1} + o(n/d)\sum x^2 + o(n/d)\sum_{i\in J} e_{i,1}}{n^2/d + 2n\cdot o(n/d) + d\cdot o(n^2/d^2)} \\
&= \frac{(\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n)(1 + o(n/d)(d/n))}{1 + 2(d/n)\cdot o(n/d) + (d^2/n^2)\cdot o(n^2/d^2)} \\
&= \frac{\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n + o(1)(\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n)}{1 + o(1)} \\
&= (1 + o(1))\left(\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n + o(1)(\frac{\sum x^2}{n} + (\sum_{i\in J} e_{i,1})/n)\right)
\end{aligned}
$$

Hence

$$
\begin{aligned}
|\widetilde{E[X^2]} - \widehat{E[X^2]}| &= O\left(\frac{\sum_{i\in J} e_{i,1}}{n}\right) \\
&= O\left(\frac{B^2\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon n}\right)
\end{aligned}
$$

Similarly, and based on $d \to n$

$$
\begin{aligned}
|\widetilde{E[X]}^2 - \widehat{E[X]}^2| &= O\left(2\left(\frac{\sum x}{n}\right)\left(\frac{\sum_{i\in J} e_{i,1}}{n}\right) + \left(\frac{\sum_{i\in J} e_{i,1}}{n}\right)^2\right) \\
&= O\left(\frac{B^3\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon n} + \frac{B^4 d\ln(1/\delta)\ln(1/p)}{\epsilon^2 n^2}\right) \\
&= O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}\right)
\end{aligned}
$$

By triangle inequality, we have

$$
\begin{aligned}
|\widetilde{\sigma^2_x} - \widehat{\sigma^2_x}| &\leq |\widetilde{E[X^2]} - \widehat{E[X^2]}| + |\widetilde{E[X]}^2 - \widehat{E[X]}^2| \\
&= O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}\right)
\end{aligned}
$$

and

$$
\tau_1 = O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma^2_x}}\right)
$$

For $\widetilde{E[XY]}$ where $X \in R_1$ and $Y \in R_2$, the privatized n in the unbiased estimation is computed as

$$
d\sqrt{\frac{\widetilde{s[c]}}{d}} = n + o(n) = O(n)
$$

Thus, the privatized and non-privatized estimation of $E[XY]$ can be computed as

$$\widetilde{E[XY]} = \frac{1-n}{1-d}\frac{\sum_{i\in J}\left(\sum_{t\in R_1.i} x\right)\left(\sum_{t\in R_2.i} y\right)}{n^2/d} + \frac{n-d}{1-d}\cdot\frac{\sum_{i\in J}\left(\sum_{t\in R_1.i} x\right)}{n^2/d}\cdot\frac{\sum_{i\in J}\left(\sum_{t\in R_2.i} y\right)}{n^2/d}$$

$$\widehat{E[XY]} = \frac{1-(n+o(n))}{1-d}\frac{\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)+e_{i,2}\right)\left(\left(\sum_{t\in R_2.i} y\right)+e_{i,3}\right)}{\sum_{j\in J}(n/d+o(n/d))(n/d+o(n/d))} + \frac{n+o(n)-d}{1-d}\frac{\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)+e_{i,2}\right)\sum_{i\in J}\left(\left(\sum_{t\in R_2.i} y\right)+e_{i,3}\right)}{\left(\sum_{j\in J}(n/d+o(n/d))(n/d+o(n/d))\right)^2}$$

$$= \frac{d(1-(n+o(n)))}{n^2(1-d)}\frac{\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)+e_{i,2}\right)\left(\left(\sum_{t\in R_2.i} y\right)+e_{i,3}\right)}{1+o(1)} + \frac{d^2(n+o(n)-d)}{n^4(1-d)}\frac{\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)+e_{i,2}\right)\sum_{i\in J}\left(\left(\sum_{t\in R_2.i} y\right)+e_{i,3}\right)}{(1+o(1))^2}$$

Based on the same flow of logic as $|\widetilde{E[X^2]}-\widehat{E[X^2]}|$, we would like to bound $\sum_{i\in J}\left(\left(\sum_{t\in R_1.i} x\right)e_{i,3}+\left(\sum_{t\in R_2.i} y\right)e_{i,2}+e_{i,2}e_{i,3}\right)$. Note that

$$\sum_{i\in J}\left(\sum_{t\in R_2.i} y\right)e_{i,2} \le \frac{nB}{d}\sum_{i\in J} e_{i,2} = O\left(\frac{nB^3\sqrt{\ln(1/\delta)\ln(1/p)}}{\sqrt{d}\epsilon}\right)$$

$$\sum_{i\in J} e_{i,2}e_{i,3} \le \sigma\sqrt{\ln(8/p)}\sum_{i\in J} e_{i,2} = O\left(\frac{B^4\sqrt{d}\ln(1/p)\ln(1/\delta)}{\epsilon^2}\right)$$

Hence the first two terms are bounded by $O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\sqrt{d}\epsilon^2}\right)$. For the last term, we may also bound as

$$\left(\sum_{i\in J}\sum_{t\in R_1.i} x\right)\sum_{i\in J} e_{i,3} = O\left(\frac{nB^3\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon}\right)$$

$$\sum_{i\in J} e_{i,3}\sum_{i\in J} e_{i,2} = O\left(\frac{B^4 d\ln(1/\delta)\ln(1/p)}{\epsilon^2}\right)$$

So the last term is $O\left(\frac{nB^3 d\sqrt{d\ln(1/\delta)\ln(1/p)}}{\epsilon n^3} + \frac{B^4 d^2\ln(1/\delta)\ln(1/p)}{\epsilon^2 n^3}\right)$, which can be combined as $O\left(\frac{B^4 d^2\ln(1/\delta)\ln(1/p)}{\epsilon^2 n^3}\right)$. Therefore

$$|\widetilde{E[XY]}-\widehat{E[XY]}| = O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\sqrt{d}\epsilon^2}\right)$$

Based on the similar analysis as $|\widetilde{E[X]}^2-\widehat{E[X]}^2|$, we have $|\widetilde{E[X]}\widetilde{E[Y]}-\widehat{E[X]}\widehat{E[Y]}| = O\left(\frac{B^4\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n}\right)$. This yields

$$|\widehat{\sigma^2_{xy}}-\widetilde{\sigma^2_{xy}}| = O\left(\frac{B^4\ln(1/\delta)\ln(1/p)}{\sqrt{d}\epsilon^2}\right)$$

With an extra assumption that X and Y are 0-centered and each tuple within R_1 and R_2 is independent and the join key is uncorrelated with X and Y. By the Chernoff-Hoeffding's inequality, with probability at least $1-p/4$, we have

$$\left|\sum_{t\in R_1.i} x\right|, \left|\sum_{t\in R_1.i} y\right| \le B\sqrt{2\ln(16d/p)n/d} \quad \forall i\in J$$

This yields

$$\sum_{i\in J}\left(\sum_{t\in R_2.i} y\right)e_{i,2} \le B\sqrt{2\ln(16d/p)n/d}\sum_{i\in J} e_{i,2} = O\left(\frac{B^3\sqrt{n\ln(d/p)\ln(1/\delta)\ln(1/p)}}{\epsilon}\right)$$

Giving a bound that scale with the size of the relation

$$|\widetilde{E[XY]}-\widehat{E[XY]}| = O\left(\frac{B^4\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n}}\right)$$

Putting everything together, with probability at least $1-p$, we have

$$\tau_2 = O\left(\frac{B^4\ln(1/p)\ln(1/\delta)\sqrt{d\ln(d/p)}}{\epsilon^2\sqrt{n}\widehat{\sigma^2_x}}\right)$$

Extension to multi-features. The extension of our analysis to multi-dimensional features involves two modifications. Firstly, the bounds B_1 and B_2 are determined by matrix norm bounds through random matrix theory [63] instead of the absolute value of single random variable . Secondly, the bound of the inverse of $\widetilde{\sigma^2_x}$ is required, where $\widetilde{\sigma^2_x}$ was scalar but now is a matrix; the inverse of $\widetilde{\sigma^2_x}$ may become unboundedly large if its minimum eigenvalue is close to 0. To address this, Wang [65] makes an additional assumption that the noises to $\widetilde{\sigma^2_x}$ has a minimum eigenvalue λ_{min} of $o(|\widehat{\sigma^2_x}|)$.

E ALLOCATION OF NOISES

We analyze the implication of dynamic allocation of privacy budget for moments on linear regression confidence bound appendix D. For union, it is possible to impute noise directly to $(\widehat{\sigma^2_x})_i$, $(\widehat{\sigma^2_{xy}})_i$, empirical variance, and covariance for each dataset R_i. Each of $(\widehat{\sigma^2_x})_i$, $(\widehat{\sigma^2_{xy}})_i$ has sensitivity $\Delta' = O(B^2/n)$. Thus, let $\sigma' = \sqrt{2\ln(1.25/\delta)}\Delta'/\epsilon$ and

$$\widetilde{\sigma^2_x} \sim \frac{\sum_i^k\left((\widehat{\sigma^2_x})_i + \mathcal{N}(0,\sigma'^2)\right)}{k}$$

Applying gaussian tail bound and the independency assumption yields $|\widetilde{\sigma^2_x}-\widehat{\sigma^2_x}| = O(B^2\sqrt{\ln(1/\delta)\ln(1/p)}/\epsilon\sqrt{k}n)$, and $|\widetilde{\sigma^2_{xy}}-\widehat{\sigma^2_{xy}}| = O(B^2\sqrt{\ln(1/\delta)\ln(1/p)}/\epsilon\sqrt{k}n)$. This reduces the bound on τ_1 and τ_2 by a factor of $O(B^2\sqrt{\ln(1/\delta)\ln(1/p)}/\epsilon)$. Based on appendix A, consider the query $q_{j,i} : \mathcal{D}^n \longrightarrow \mathcal{S}^i$ where $\mathcal{S}^i = \{v \mid v \in \mathbb{R}^i\}$. $q_{j,i}$ returns a vector $s_i \in \mathcal{S}^i$ containing the sum of the i-order monomials across each join key.

$$\Delta_{q_{j,i}} = \sqrt{1_{2\mathbb{Z}+1}(i)4B^{2i} + 1_{2\mathbb{Z}}(i)2B^{2i}}$$

For linear regression, it is feasible to decomposite q_j into 3 sequential queries, $q_{j,0}$, $q_{j,1}$ and $q_{j,2}$, each with privacy budget $(\epsilon/3, \delta/3)$. Inheriting notations from appendix D, $\Delta = \Delta_{q_j}, O(B^2\Delta_{q_{j,0}}) = O(B\Delta_{q_{j,1}}) = O(\Delta_{q_{j,2}}) = O(B^2)$, note that although there is less privacy budget on releasing the count of tuples within each join key, the sensitivity is also reduced by a magnitude of B^2, i.e. from Δ to $\Delta_{q_{j0}}$. Hence, it is reasonable to assume that the noise on this number is small, and bounded by $o(n/d)$. The main implication is that $e_{i,2}, e_{i,3} = \mathcal{N}(0, 2\ln(1.25/(\delta/3))\Delta^2_{q_{j,1}}/(\epsilon/3)^2)$, and

$e_{i,1} = \mathcal{N}(0, 2\ln(1.25/(\delta/3))\Delta^2_{q_{j,2}}/(\epsilon/3)^2)$, and no more change to the analysis is required. Following the computations in appendix D, we have

$$\tau_1 = O(\frac{B^2\sqrt{d}\ln(1/\delta)\ln(1/p)}{\epsilon^2 n\widehat{\sigma_x^2}}), \tau_2 = O(\frac{B^2\ln(1/\delta)\ln(1/p)}{\sqrt{d}\widehat{\sigma_x^2}})$$