Federated Naive Bayes under Differential Privacy

Thomas Marchioro[1][a], Lodovico Giaretta[2][b], Evangelos Markatos[1][c] and Šarūnas Girdzijauskas[2][d]
[1]*Institute of Computer Science, Foundation for Research and Technology Hellas, Heraklion, Greece*
[2]*Division of Software and Computer Systems, KTH Royal Institute of Technology, Stockholm, Sweden*

Keywords: Federated Learning, Naive Bayes, Differential Privacy.

Abstract: Growing privacy concerns regarding personal data disclosure are contrasting with the constant need of such information for data-driven applications. To address this issue, the combination of federated learning and differential privacy is now well-established in the domain of machine learning. These techniques allow to train deep neural networks without collecting the data and while preventing information leakage. However, there are many scenarios where simpler and more robust machine learning models are preferable. In this paper, we present a federated and differentially-private version of the Naive Bayes algorithm for classification. Our results show that, without data collection, the same performance of a centralized solution can be achieved on any dataset with only a slight increase in the privacy budget. Furthermore, if certain conditions are met, our federated solution can outperform a centralized approach.

1 INTRODUCTION

The emergence of the Internet of Things and the pervasive use of Internet-enabled technologies have facilitated the process of gathering and collecting data, from a myriad of sources and with ever-increasing granularity. This in turn has led to the development and deployment of a vast array of data-driven technologies ranging from smart home products to medical wearable devices (Shafagh et al., 2017).

However, this has sometimes come at the detriment of personal privacy. Many organizations collect sensitive personal information about their users or customers, and thanks to the high granularity and broad scope of many data gathering processes, tracking individual users across data collections is dangerously feasible (Marchioro. et al., 2021). Due in part to high-profile data misuse incidents and to the ever-increasing threat of data exfiltration attacks (Ullah et al., 2018), many users and regulators have grown aware of the importance of ensuring a certain degree of personal privacy in the handling of user data.

In the context of machine learning, this has led to the growing popularity of Federated Learning (McMahan et al., 2017), a paradigm in which a group of data owners can train a model on their private data, without the need to disclose it or gather it into a central storage. In each iteration of the process, every participant computes local updates of the model parameters, which are sent to a central aggregator and merged.

While Federated Learning removes the need to *directly* share private data, it may still leak sensitive information to third parties. The most obvious leak is given by the fact that the central aggregator, by comparing the previous model version and the update provided by a participant, may reconstruct the training data used by that participant (Zhu et al., 2019). However, even assuming that the aggregator is trusted, research has shown that it is sometimes possible to reconstruct data used during training just by inspecting or querying the resulting trained model (Salem et al., 2018).

A robust approach to prevent these leaks is the use of Differential Privacy (Ji et al., 2014), a family of techniques that consists of injecting into the computation process noise sampled from specifically-crafted distributions. Thanks to its strong mathematical guarantees, Differential Privacy has become a de-facto standard in privacy-preserving machine learning.

Most work combining Federated Learning and Differential Privacy has focused on deep neural networks, trained using stochastic gradient descent techniques. However, not all applications require this

[a] https://orcid.org/0000-0003-3353-102X
[b] https://orcid.org/0000-0002-0223-8907
[c] https://orcid.org/0000-0003-3563-7733
[d] https://orcid.org/0000-0003-4516-7317

Marchioro, T., Giaretta, L., Markatos, E. and Girdzijauskas, Š.
Federated Naive Bayes under Differential Privacy.
DOI: 10.5220/0011275300003283
In *Proceedings of the 19th International Conference on Security and Cryptography (SECRYPT 2022)*, pages 170-180
ISBN: 978-989-758-590-6; ISSN: 2184-7711
Copyright © 2022 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

level of model complexity and capacity. In many cases, extremely simple but very robust models, such as Naive Bayes classifiers, are preferable (Zhang, 2004).

Therefore, in this paper we develop and evaluate the first (to our knowledge) implementation of a differentially-private Naive Bayes classifier in a federated setting for horizontally-partitioned data. Contrary to typical federated learning, federated Naive Bayes is trained with a single communication step, without requiring multiple iterations, thus being fast and efficient.

Our results show that the federated approach can reach the same accuracy as a centralized, differentially-private implementation, but requires a slightly higher privacy budget (i.e. slightly higher chance of information leakage). Furthermore, we show that, when the local sensitivity in each federated data partition is much lower than the sensitivity of the whole dataset, our approach outperforms a centralized one. Thus, overall, switching to a federated solution to avoid central data collection incurs minimal or no cost in terms of accuracy or privacy budget.

The contributions of this work are as follows:

- We provide an algorithmic implementation for Federated Naive Bayes under Differential Privacy;
- We extensively evaluate how the accuracy of the model varies under different choices of privacy budget and data distribution;
- We discuss several potential extensions to our algorithm, including online parameter updates, byzantine resilience and full decentralization.

2 BACKGROUND

In this section we introduce the core concept of the original Naive Bayes algorithm, along with the definition and some useful properties of differential privacy, which will serve as a background to understand the rest of the paper. The symbols used through the discussion are summarized in Table 1.

Naive Bayes. The Naive Bayes algorithm is used to train a model for sample classification, meaning that it leverages a training dataset $(X,Y) \in \mathcal{X}^n \times \mathcal{Y}^n$ to learn a function $\hat{g} : \mathcal{X} \to \mathcal{Y}$ that maps an example $x \in \mathcal{X}$ to a class $y \in \mathcal{Y}$. The classification rule is based on the maximum a posteriori probability criterion (MAP), which chooses the class maximizing the posterior probability $p_{Y|X}(y|x)$. This is computed according to the Bayes rule (hence the name):

Table 1: Notation used in the paper.

ε	Privacy budget
n	Size of the training set
n_y	Number of training examples in class y
m_{fvy}	Number of training examples in class y with categorical feature f equal to v
μ_{fy}	Mean value of feature f for class y
ς_{fy}	Second moment of feature f for class y
σ^2_{fy}	Variance of feature f for class y
$\square^{(i)}$	Value $\square$ computed on the i-th partition
$\square[k]$	k-th sample of $\square$
N	Number of dataset partitions
C	Number of classes
F_{cat}	Number of categorical features
F_{num}	Number of numerical features
$\mathcal{F}_{\text{cat}}$	Set of categorical features
$\mathcal{F}_{\text{num}}$	Set of numerical features

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p_{Y|X}(y|x) = \arg\max_{y \in \mathcal{Y}} p_Y(x) p_{X|Y}(x|y) \quad (1)$$

where the term $p_X(x)$ at the denominator is neglected as independent from y. A simplifying assumption made in Naive Bayes is that the features $f = 1, \ldots, F$ are independent, leading to

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p_Y(y) \prod_{f=1}^{F} p_{X_f|Y}(x_f|y) \quad (2)$$

The prior class probabilities p_Y and the conditional probabilities $p_{X_f|Y}$ for categorical features are estimated from the training data as ratios of counts, i.e.,

$$p_Y(y) = \frac{n_y}{\sum_{y' \in \mathcal{Y}} n_{y'}}, \quad p_{X_f|Y}(v|y) = \frac{m_{fvy}}{n_y}. \quad (3)$$

For numerical features, the standard method to estimate the conditional distribution is to assume $p_{X_f|y}$ to be normal with density

$$p_{X_f|y}(x_f|y) = \frac{1}{\sqrt{2\pi\sigma^2_{fy}}} \exp\left(-\frac{(x_f - \mu_{fy})^2}{2\sigma^2_{fy}}\right) \quad (4)$$

where

$$\mu_{fy} = \frac{\sum_{k:y[k]=y} x_f[k]}{n_y}, \sigma^2_{fy} = \frac{\sum_{k:y[k]=y} (x_f[k] - \mu_{fy})^2}{n_y}. \quad (5)$$

This method of estimating the conditional distribution is know as Gaussian Naive Bayes.

Differential Privacy. The main idea of differential privacy is to perturb aggregate queries in order to prevent information leakage. The formal definition is that a randomized mechanism $q_\varepsilon : \mathcal{D}^n \to \mathcal{Q}$ provides

ε-differential privacy (ε-DP) if for any pair of adjacent datasets D, D' (i.e., differing by one single entry) it holds

$$\Pr[q_\varepsilon(D) \in Q] \leq e^\varepsilon \Pr[q_\varepsilon(D') \in Q], \forall Q \subseteq Q, \quad (6)$$

meaning that the output distribution of q_ε should not differ "too much" by changing a single entry. The maximum variation allowed in the distribution is determined by the parameter ε, which takes the name of *privacy budget*. Intuitively, to a lower ε corresponds a higher level of privacy. Two useful properties that are used throughout the discussion are the following (McSherry, 2009):

- **P1:** If m independent $\frac{\varepsilon}{m}$-DP queries are computed on a same dataset D, then any function of these queries provides ε-DP.
- **P2:** If ε-DP queries are computed on disjoint subsets of a dataset $D^{(1)}, \ldots, D^{(N)}, D^{(i)} \cap D^{(j)} = \emptyset, i \neq j$, then any function of these queries provides ε-DP.

For numerical queries such as count or average, a well-established method to provide ε-DP is the Laplace mechanism L_ε. The Laplace mechanism L_ε for a query q computed on a dataset D consists in simply adding Laplace noise to the query with scale inversely proportional to ε:

$$L_\varepsilon(D, q) = q(D) + \xi, \quad \xi \sim \text{Lap}(0, \frac{\Delta_q}{\varepsilon}). \quad (7)$$

The value Δ_q is the maximum variation of the query between adjacent datasets, also called the *sensitivity* of the query. In this paper, we make use of two definitions of sensitivity:

- The global sensitivity of a query q

$$\Delta_q = \max_{D,D'} |q(D) - q(D')| \quad (8)$$

  which is defined for any possible pair of adjacent datasets D, D'. This implies that the global sensitivity is always the same, regardless of the dataset on which the query is computed.

- The local sensitivity of a query q for a dataset D

$$\Delta_q(D) = \max_{D'} |q(D) - q(D')| \quad (9)$$

  which is instead defined for datasets that differ by one row from D[1]. The local sensitivity depends on the dataset on which the query is computed.

[1]In practice, to compute the local sensitivity, we consider the maximum variation obtained by removing one row from the dataset.

The Laplace mechanism provides ε-DP using both definitions of sensitivity. The main difference between the two is that global sensitivity requires to apply always the same amount of noise. For example, counting queries have a global sensitivity of 1, meaning that the required amount of noise is $\text{Lap}(0, \frac{1}{\varepsilon})$. However, other queries such as sum and mean over the samples of a dataset depend on the dataset itself, so it is not possible to compute the global sensitivity. In such cases, we settle for using local sensitivity, which leads to different amounts of noise depending on the dataset.

Sensitivity of the Queries. As we discuss in later sections, our algorithm makes use of two types of queries: "count" and "sum". It is easy to compute the global sensitivity of a "count" query, since if you remove a row from a dataset, the count can change by at most 1. For sums, instead, we compute the local sensitivity, as per the above discussion. Supposing the data we are trying to sum is $v^{(1)}, \ldots, v^{(N)}$, the maximum variation in the sum is given by $\max_{i=1,\ldots,N} |v^{(i)}|$.

3 RELATED WORK

Differentially Private Machine Learning. Differential privacy has become a de facto standard to compute statistics on sensitive data (Dwork, 2008). Among its many applications, privacy-preserving machine learning is probably the most prominent and studied (Ji et al., 2014). Differentially private SGD is now a well-established training method (Abadi et al., 2016) for neural networks, and outside the domain of deep learning many studies have also explored classical models such as support vector machines, decision trees, and Naive Bayes (Lopuhaä-Zwakenberg et al., 2021). The original algorithm for a differentially private Naive Bayes, which we extend in this paper, was proposed in (Vaidya et al., 2013).

Naive Bayes on Partitioned Data. Albeit simple in its design, Naive Bayes has proven to be an effective machine learning algorithm for classification, which is virtually immune to overfitting (Rish et al., 2001; Zhang, 2004). Because of its efficacy and popularity, a number of efforts have been put into making it usable also on personal data. In (Kantarcıoglu et al., 2003), the authors designed an algorithm to train Naive Bayes on horizontally partitioned data (i.e., each node possesses different samples), while (Vaidya and Clifton, 2004) proposed solution for vertically partitioned data (i.e., each node possesses different features from the same samples). Both solu-

tions, however, rely on cryptographic methods that protect the data during the training procedure, but do not provide guarantees that information is not leaked from the resulting model. Another method to protect horizontal data partitions during training relies on semi-trusted mixers (Yi and Zhang, 2009). An algorithm that combines homomorphic encryption and differential privacy was proposed in (Li et al., 2018). However, the algorithm estimates conditional probabilities of numerical features using histograms, rather than with the standard Gaussian Naive Bayes approach. Furthermore, the paper does not report how the accuracy is affected by the privacy budget and distribution of the data among the nodes.

Federated Learning and Differential Privacy. Federated Learning was proposed in (McMahan et al., 2017) as general framework for training deep models using gradient-based optimization techniques on private data that are massively distributed in horizontal partitions. It consists of an iterative process in which, at each step, the data-owning devices compute the gradient of the model on a batch of local data, and forward it to a central aggregator. This will in turn average the gradients from all devices, update the model weights accordingly, and then communicate the new weights to each device to initiate the next iteration.
While Federated Learning ensures that the training data never leaves the device of its owning entity, it does not prevent data leakage from either the trained model (Salem et al., 2018) or from the gradients that are shared at each step (Zhu et al., 2019). Therefore, (Wei et al., 2020) incorporates differential privacy in each shared gradient to protect the individual data points in each private partition. On the other hand, (Geyer et al., 2017) employs differential privacy within the central aggregator, thus protecting not the identity of an individual data point, but rather of an entire data-owning entity participating in the protocol.
Most studies of Federated Learning present two key differences compared to our work. First, Federated Learning focuses on iterative gradient-based optimization of model weights, while Naive Bayes models are trained in a single step from simple aggregate statistics of the dataset. Second, most works focus on *massively-decentralized* Federated Learning, in which the number of nodes is very large and each contains little data, typically from a single user (e.g. next word prediction applications from the typing history stored on smartphones). On the other hand, few works, such as (Geyer et al., 2017) and (Bernal et al., 2021), focus on the scenario where a relatively small number of entities, each owning datapoints from multiple individuals, cooperate to train a model.
In (Islam et al., 2022), the authors studied how differential privacy affects the accuracy of Naive Bayes, in a federated setting where data are vertically partitioned. The contribution of our work can be considered complementary, since we consider a federated setting where data are horizontally partitioned.

4 FEDERATED SETTING

In this work, we consider a federated setting, in which the dataset is composed of N partitions $D^{(1)}, \ldots, D^{(N)}$, each held by a different entity, i.e., an organization or an individual. Each entity stores its partition on its own appliances and is unwilling to share these data due to privacy concerns. Data points within a partition may, for example, represent individual users of a service provided by the organization owning that partition and may therefore carry sensitive personal information. The entities wish to take part in the collaborative training of a joint predictive model, namely a Naive Bayes model, as long as the following conditions are satisfied:

1. each data partition $D^{(i)}$ never leaves the appliances of the owning entity, and
2. it is not possible, from the trained model, to glean sensitive information about individual data points within each partition that is not already inferrable from the rest of the data.

The first condition is met by having each entity computing aggregated queries locally on its data partition, while the second condition is met by applying the ε-DP Laplace mechanism to the queries before disclosing them. These queries are collected by a central aggregator and used to estimate the parameters of the Naive Bayes model. The participating entities are also referred as of an overlay network that is used to communicate with the central aggregator, and they are denoted by their data partition (i.e., the i-th node is denoted by $D^{(i)}$).

5 ALGORITHM DESIGN

In order to train a federated Naive Bayes model, the central aggregator should be able to collect information that allows to compute the prior and conditional probabilities for each feature and class. Indeed, all the query results must be disclosed after applying the Laplace mechanism in order to guarantee differential privacy. The parameter ε' of the Laplace mechanism is decided according to the privacy budget ε and to the

number of queries asked to each node[2].
Mirroring eq. (3), prior probabilities $p_Y(1),\dots,p_Y(C)$ can be simply estimated by collecting from each node $D^{(i)}$ the number of samples per each class $n_1^{(i)},\dots,n_C^{(i)}$ and by computing

$$p_Y(y) = \frac{\sum_{i=1}^{N} n_y^{(i)}}{\sum_{i=1}^{N}\sum_{y'=1}^{C} n_y^{(i)}}. \tag{10}$$

Similarly, conditional probabilities for categorical features are estimated according to

$$p_{X|Y}(v|y) = \frac{\sum_{i=1}^{N} m_{fvy}^{(i)}}{\sum_{i=1}^{N} n_y^{(i)}} \tag{11}$$

where $m_{fvy}^{(i)}$ is the number of samples with feature $f \in \mathcal{F}_{\text{cat}}$ equal to v for node $D^{(i)}$. For numerical features, Gaussian Naive Bayes requires to compute the parameters μ_{fy}, σ_{fy}^2 of the assumed normal distribution of each feature for each class. The mean μ_{fy} can be derived by querying the sample sum

$$S_{fy}^{(i)} = \sum_{k:y^i[k]=y} x_f^{(i)}[k] \tag{12}$$

for feature $f \in \mathcal{F}_{\text{num}}$ and class y from each node $D^{(i)}$ as

$$\mu_{fy} = \frac{\sum_{i=1}^{N} S_{fy}^{(i)}}{\sum_{i=1}^{N} n_y^{(i)}}. \tag{13}$$

For the variance σ_{fy}^2, a straightforward solution would be to calculate it by having each node computing the sum of squared deviations from the sample average μ_{fy}, i.e.,

$$\sum_{k:y_k^{(i)}[k]=y} (x_f^{(i)}[k] - \mu_{fy})^2. \tag{14}$$

However, this solution would require two exchanges between each node and the central aggregator, since μ_{fy} would need to be sent back to the nodes to compute the squared deviations. A solution that enables the central aggregator to compute means and variances with a single exchange leverages the relation between second moment, mean, and variance, i.e., $\sigma^2 = \varsigma - \mu^2$. Therefore, in our algorithm the nodes are asked to compute the sum of the squared samples

$$Q_{fy}^{(i)} = \sum_{k:y^i[k]=y} \left(x_f^{(i)}[k]\right)^2 \tag{15}$$

and the variance is obtained by the centralized aggregator as

$$\sigma_{fy}^2 = \frac{\sum_{i=1}^{N} Q_{fy}^{(i)}}{\sum_{i=1}^{N} n_y^{(i)}} - \mu_{fy}^2. \tag{16}$$

[2]In principle, each node may have a different privacy budget, but herein we assume there is a common value of ε for all the nodes.

Achieving ε-differential Privacy. In order to guarantee ε-DP to each node $D^{(i)}$, all queries must be protected with the Laplace mechanism $L_{\varepsilon'}$. The parameter ε' is determined according to properties (P1) and (P2) described in section 2. Since queries on different classes are computed on disjoint subsets of $D^{(i)}$ (a sample cannot belong to multiple classes at the same time), from a privacy perspective they count as a single query. On the other hand, queries on different features of $D^{(i)}$ are counted separately, since they involve the same entries. Therefore each node overall is asked:

- 1 class counting query, i.e, $n_y^{(i)}$;
- F_{cat} category counting queries, i.e., $m_{fvy}^{(i)}, f \in \mathcal{F}_{\text{cat}}$;
- $2F_{\text{num}}$ sum queries on numerical features, i.e., $S_{fy}^{(i)}, Q_{fy}^{(i)}, f \in \mathcal{F}_{\text{num}}$.

Therefore, in order to guarantee a privacy budget of ε, the parameter ε' of the Laplace mechanism to be applied to each query is given by

$$\varepsilon' = \frac{\varepsilon}{1 + F_{\text{cat}} + 2F_{\text{num}}}, \tag{17}$$

in congruence with the centralized ε-DP Naive Bayes (Vaidya et al., 2013). However, in our case the mechanism must be applied locally at each node, and therefore the overall noise applied to the parameters will not follow a Laplace distribution, as will be shown in section 6.3.

Training Procedure. As mentioned in the discussion above, the training procedure consists of a single exchange for each node with the central aggregator. Each node i computes locally the counting and sum queries on its own partition according to Algorithm 1. The query results $\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)}$ are sent to the central aggregator, which collects them across all nodes and computes the overall parameters of the model according to Algorithm 2.
The inference of a sample's class is done according to the computed parameters as in the original Naive Bayes (see equation 2).

6 EVALUATION

6.1 Datasets

We test our approach on 6 standard datasets, all taken from the UCI repository[3]. The statistics of each datasets are listed in table 2. For those datasets that

[3]https://archive.ics.uci.edu/ml/datasets.php

Algorithm 1: Local queries computed by i-th node.

Require: Data partition $D^{(i)} = (X^{(i)}, Y^{(i)})$, privacy budget ε

for $y = 1, \dots, C$ **do**
  $\varepsilon' \leftarrow \frac{\varepsilon}{1 + F_{\text{cat}} + 2F_{\text{num}}}$
  $\tilde{n}_y^{(i)} \leftarrow L_{\varepsilon'}(|\{k : y^{(i)}[k] = y\}|)$
  for $f \in \mathcal{F}_{\text{cat}}$ **do**
    $\tilde{m}_{fvy}^{(i)} \leftarrow L_{\varepsilon'}(|\{k : x_f^{(i)}[k] = v \wedge y^{(i)}[k] = y\}|)$
  end for
  for $f \in \mathcal{F}_{\text{num}}$ **do**
    $\tilde{S}_{fy}^{(i)} \leftarrow L_{\varepsilon'}(\sum_{k:y^{(i)}[k]=y} x_f^{(i)}[k])$
    $\tilde{Q}_{fy}^{(i)} \leftarrow L_{\varepsilon'}(\sum_{k:y^{(i)}[k]=y} (x_f^{(i)}[k])^2)$
  end for
end for
return $\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)}$

Algorithm 2: Centralized aggregation.

Require: $\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)}$ for $i = 1, \dots, N$

for $y = 1, \dots, C$ **do**
  $\tilde{p}_Y(y) \leftarrow \frac{\sum_{i=1}^N \tilde{n}_y^{(i)}}{\sum_{i=1}^N \sum_{y'=1}^{N_c} \tilde{n}_y^{(i)}}$
  for $f \in \mathcal{F}_{\text{cat}}$ **do**
    $\tilde{p}_{X_f|Y}(v|y) \leftarrow \frac{\sum_{i=1}^N \tilde{m}_{fvy}^{(i)}}{\sum_{i=1}^N \tilde{n}_y^{(i)}}$
  end for
  for $f \in \mathcal{F}_{\text{num}}$ **do**
    $\tilde{\mu}_{fy} \leftarrow \frac{\sum_{i=1}^N \tilde{S}_{fy}^{(i)}}{\sum_{i=1}^N \tilde{n}_y^{(i)}}$
    $\tilde{\sigma}_{fy}^2 \leftarrow \frac{\sum_{i=1}^N \tilde{Q}_{fy}^{(i)}}{\sum_{i=1}^N \tilde{n}_y^{(i)}} - \tilde{\mu}_{fy,\ell}^2$
  end for
end for
return $\tilde{p}_Y, \tilde{p}_{X|Y}, \tilde{\mu}, \tilde{\sigma}^2$

do not provide a standardadized train/test split, we perform a 9:1 split with a random seed that is kept fixed across the entire work.

6.2 Setup

We implement three variants of Naive Bayes:

1. a "vanilla" Naive Bayes based on eqs. (2) to (5) that has access to the entire centralized dataset
2. a differentially-private centralized Naive Bayes based on (Vaidya et al., 2013)
3. our differentially-private federated Naive Bayes based on the algorithm in section 5.

We test each variant on all datasets through a Monte Carlo analysis, using logarithmically-spaced values of ε between 10^{-2} and 10^{1} for the differentially-private variants. For the federated approach, we test with number of data partitions $N \in [1, 10, 100, 1000]$ whenever possible, skipping the higher values of N on small datasets, based on the condition that each partition should include at least 2 data points.

We repeat each experiment 1000 times to account for the randomness in both the data partitioning across federated entities and the sampling of Laplace noise. Unless otherwise stated, we report the mean of those 1000 trials.

All the code and data used in this work are available on GitHub[4].

6.3 Results

Basic Behaviour. Figure 1 plots the accuracy of the differentially-private Naive Bayes models as a function of the privacy budget ε, including both the centralized approach, and the federated approach with different choices of N. The score obtained by a non-differentially-private model is shown as an upper bound of attainable performance.

On most datasets, the centralized model presents a typical S-shaped curve, being no better than a random guess for very small values of ε, but quickly growing towards the baseline accuracy as the privacy budget increases. The federated model tends to follow the same curve, but with a "delay" proportional to the number of partitions. This behaviour is most clear in the Mushroom and Congressional Voting datasets, although in the former $\varepsilon = 10^{-2}$ is not sufficient to fully appreciate the left-hand tail. Due to its large sample count, Skin Segmentation only shows the right-hand tail of the S-shape, with the federated models dipping below the maximum performance only for very small values of ε. SPECT Heart, on the other hand, shows a long left-hand tail and peaks at an accuracy only just above that of a random guess, likely due to the small size of the training set and the non-independence of its features.

Impact of Local Sensitivity. Two datasets, however, show a different behaviour. In Adult, the expected pecking order is present on the right hand side of the spectrum, but a reversed order can be seen for small values of ε, where federated Naive Bayes is more accurate than the centralized variant and further

[4] https://github.com/thomasmarchioro3/FederatedNaiveBayesDP

Table 2: Datasets used in the evaluation.

Dataset	Samples	Labels	F_{num}	F_{cat}	Predefined train/test split
Accelerometer	153,000	3	3	0	no
Adult	48,842	2	6	8	yes (2:1)
Congressional Voting	435	2	0	16	no
Mushroom	8,124	2	0	22	no
Skin Segmentation	245,057	2	3	0	no
SPECT Heart	267	2	0	22	yes (3:7)

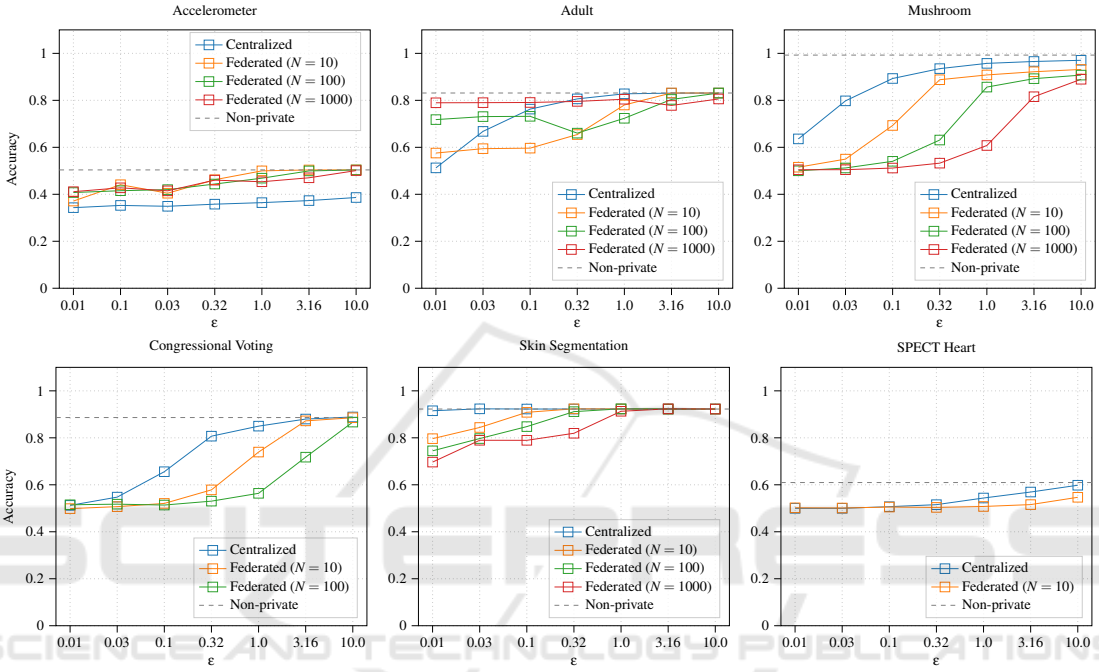

Figure 1: Comparison of centralized and federated ε-DP Naive Bayes for different privacy budgets.

improves as the number of participants N increases. On Accelerometer, the federated variant is always better than the centralized one, with N having a negligible impact.

This is likely caused by a decrease in the sensitivity of the queries in each local partition. With a centralized approach, the variance of the laplacian noise is proportional to the sensitivity of the whole dataset. In addition, a single additive noise contribution is sampled for each parameter of the model. On the other hand, in a federated setting, the noise added by each participant has variance proportional to the sensitivity of its local partition. This is never bigger than that of the full dataset and may decrease dramatically as the partition gets smaller. Furthermore, the final estimate of each model parameter is obtained by summing the contributions – and thus the noises – of each partition. The sum of many low-variance noise contributions is likely closer to zero compared to a single contribution with high variance.

On Adult and Accelerometer, this drop in local sensitivity is sufficient to reduce the generally higher noise level of the federated model below that of the centralized model, thus leading to better results. For other datasets, such as Skin, the sensitivity does not decrease much with more partitions. We can see this difference in fig. 2, which compares the sensitivity of the sum query $\tilde{S}^{(i)}$ for all numerical features on Adult and Skin.

On Adult, the ε range of our experiments perfectly captures the threshold at which this effect becomes significant. On Accelerometer, the ε range only captures the very left-hand side of the behaviour. This can be deduced by the fact that the centralized approach is far from the accuracy ceiling and only barely starting to improve.

Susceptibility to the Privacy Budget. These results highlight how the threshold at which the differential privacy mechanism starts severely affecting accuracy is highly dependent on the dataset and should be evaluated domain by domain. They also show how

switching to a federated setting, and thus avoiding the data collection phase, only requires a small increase to the privacy budget in order to achieve the same accuracy, with the possibility of even surpassing a centralized solution in certain limited scenarios.

Noise Distribution. Figure 3 compares the error distribution of $\tilde{\mu}$, as computed with algorithm 2, on Adult, for different number of partitions N. While each noise contribution added to the individual $\tilde{n}^{(i)}$ and $\tilde{S}^{(i)}$ parameters follows a Laplace distribution, their combination by the central aggregator (eq. (13)) approaches a normal distribution as the number of individual contributions increases.

7 POSSIBLE EXTENSIONS AND FUTURE WORK

The differentially-private algorithm for federated Naive Bayes introduced in section 5 is simple at its core, leaving room for possible extensions and adaptations to new use cases. In this section, we describe some of such extensions that can be added in order to increase the efficiency and security of the training procedure. Furthermore, we discuss an open problem, i.e., Byzantine resilience, which we leave to be addressed by future work.

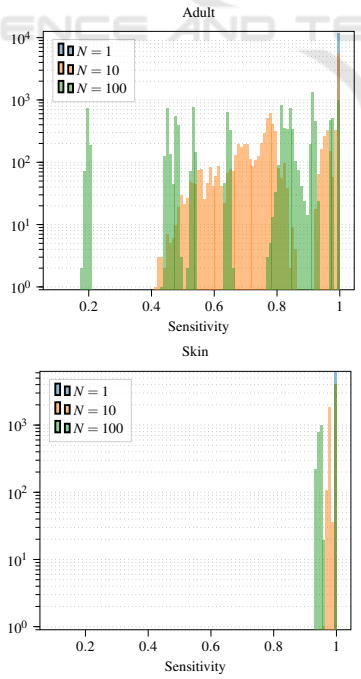

Figure 2: Distribution of the sensitivity for the sum query $\tilde{S}^{(i)}$, for different choices of N, on Adult and Skin.

Online Updates. Algorithm 2 takes as input the query results from all nodes. In reality, we expect the training procedure to be executed in an asynchronous fashion, with each participant sending its results at a different time and the aggregator performing subsequent updates to the Naive Bayes parameters. This also allows new nodes to join the overlay network and collaborate in training the model. One straightforward way to make this possible is to store the responses of each node, and recompute the model parameters every time a new set of query responses is sent by a node. However, this is inefficient and requires the central aggregator to store unnecessary information.

An alternative consists in storing the following aggregated information:

- the total number $\tilde{\mathrm{v}}_y$ of samples for each class;
- the total counters of categorical features $\tilde{M}_{fvy}$ for each class;
- the values of mean and variance of all the numerical features $\tilde{\mu}_{fy}, \tilde{\sigma}^2_{fy}$ for each class.

Notice that the parameters $\tilde{p}_Y(y)$ and $\tilde{p}_{X_f|Y}(v|y)$ can be computed from $\tilde{\mathrm{v}}_y$ and $\tilde{M}_{fvy}$ as

$$\tilde{p}_Y(y) = \frac{\tilde{\mathrm{v}}_y}{\sum_{y'=1}^{C} \tilde{\mathrm{v}}_{y'}}, \quad \tilde{p}_{X|Y}(v|y) = \frac{\tilde{M}_{fvy}}{\tilde{\mathrm{v}}_y}, \tag{18}$$

so the stored variables completely characterize the model. The complete procedure for updating such parameters is described in Algorithm 3, and the formulas employed for each of them are derived in the appendix.

Algorithm 3: Online update.

Require: current parameters $\tilde{\mathrm{v}}^{(t)}, \tilde{M}^{(t)}, \tilde{\mu}^{(t)}, [\tilde{\sigma}^2]^{(t)}$, update $\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)}$ from $D^{(i)}$

for $y = 1, \ldots, C$ **do**
  $\tilde{\mathrm{v}}_y^{(t+1)} \leftarrow \tilde{\mathrm{v}}_y^{(t)} + \tilde{n}_y^{(i)}$
  for $f \in \mathcal{F}_{\text{cat}}$ **do**
    $\tilde{M}_{fvy}^{(t+1)} \leftarrow \tilde{M}_{fvy}^{(t)} + \tilde{m}_{fvy}^{(i)}$
  end for
  for $f \in \mathcal{F}_{\text{num}}$ **do**
    $\tilde{\mu}_{fy}^{(t+1)} \leftarrow \frac{\tilde{\mathrm{v}}_y^{(t)}}{\tilde{\mathrm{v}}_y^{(t+1)}} \tilde{\mu}_{fy}^{(t)} + \frac{\tilde{S}_{fy}^{(i)}}{\tilde{\mathrm{v}}_y^{(t+1)}}$
    $\tilde{\xi}_{fy}^{(t+1)} \leftarrow \frac{\tilde{\mathrm{v}}_y^{(t)}}{\tilde{\mathrm{v}}_y^{(t+1)}} ([\tilde{\sigma}^2_{fy}]^{(t)} + [\tilde{\mu}_{fy}^{(t)}]^2) + \frac{\tilde{Q}_{fy}^{(i)}}{\tilde{\mathrm{v}}_y^{(t+1)}}$
    $[\tilde{\sigma}^2_{fy}]^{(t+1)} \leftarrow \tilde{\xi}_{fy}^{(t+1)} - [\tilde{\mu}_{fy}^{(t+1)}]^2$
  end for
end for
return $\tilde{\mathrm{v}}^{(t+1)}, \tilde{M}^{(t+1)}, \tilde{\mu}^{(t+1)}, [\tilde{\sigma}^2]^{(t+1)}$

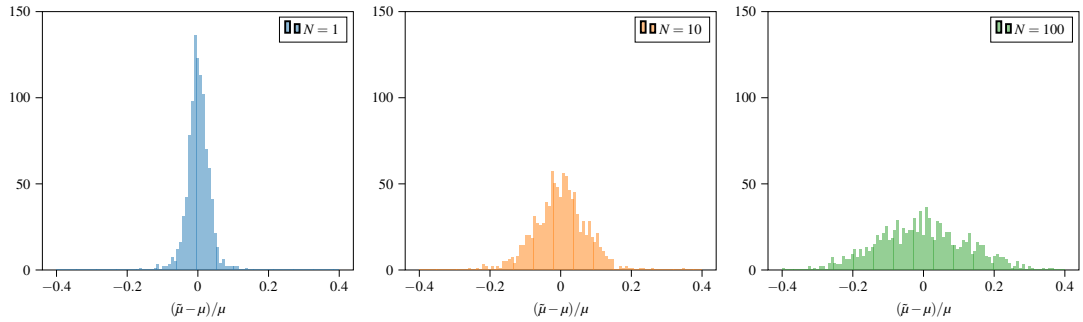

Figure 3: Distribution of the relative error between the noisy estimate of the mean $\tilde{\mu}$ and the true value μ, averaged across all the features and classes. As N increases, the shape looks less like a Laplace distribution and more like a normal one.

Fully-decentralized Naive Bayes. While our main implementation of differentially-private federated Naive Bayes achieves its goal of protecting data privacy, it still presents certain limitations that are related to the use of a central aggregator. First, since this aggregator is uniquely responsible for merging the contributions of individual entities, it must be trusted to do so in a fair and correct manner. Second, the aggregator may represents a bottleneck in terms of computation and/or network communication in the process.

One promising alternative is the use of Gossip Learning techniques (Ormándi et al., 2013), where the central aggregator is replaced by peer-to-peer model exchanges. Therein, participants update their local state based on the information received from a peer, and later broadcast their own information to another random participant. These peer-to-peer communications eventually lead to the convergence of the local state of all participants. While these techniques have not been studied to the same extent as Federated Learning, recent work has shown that they can compete with it in terms of performance (Hegedűs et al., 2021) and that they can be quite robust to different data distributions and network conditions (Giaretta and Girdzijauskas, 2019).

To apply Gossip Learning in the context of Naive Bayes, each entity still computes the statistics of its partition according to algorithm 1. Then, the gossip-based aggregation scheme detailed in (Jelasity et al., 2005) is used which, upon reaching convergence, provides every entity with the sums of each statistic across all partitions. Each entity is then able to perform algorithm 2 locally and independently, thus obtaining the full Naive Bayes model without any need for central orchestration.

Byzantine Resilience. One important research topic in the area of Federated Learning is *byzantine resilience*, i.e., the ability to converge to a high-quality model even when some of the entities involved are malicious and may take arbitrary actions to prevent this outcome. We refer the reader to (Lyu et al., 2020) and (Awan et al., 2021) for a thorough introduction to the topic.

This issue is most significant in a *massively-distributed*, *crowd-sourced* setting, where participation in the protocol is open to any entity and the total number of participants can reach millions. However, it can also affect smaller, closed settings: it might not be possible to establish trust relationships between the entities in the system, or there may be a concern that some otherwise honest entities may at some point be compromised by an external malicious actor.

One well-studied approach to achieve byzantine resilience consists of median-based aggregation rules, such as Krum (Blanchard et al., 2017) and Bulyan (Guerraoui et al., 2018). However, there are challenges that need to be addressed when applying these techniques to Naive Bayes. First, the query results $\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)}$ are not suitable for median-based aggregation, as their magnitude depends on the amount of the data (overall and per class) within each partition. Instead, local Naive Bayes parameters $\tilde{p}_Y^{(i)}(y), \tilde{p}_{X_f|Y}^{(i)}(v|y), \tilde{\mu}^{(i)}, \left[\tilde{\sigma}^2\right]^{(i)}$ need to be input to the median-based aggregation rule. This works well when the partitions are large and the data IID distributed among them. However, it may not be suitable when certain classes or feature values are underrepresented in certain partitions, causing the local parameters to be very inaccurate if not undefined. Thus, one key strength of the sum- and count-based aggregation approach is lost.

Additionally, these byzantine resilience approaches, like most others, are designed for iterative training processes, in which new updates from each entity are computed and aggregated at each time step, thus slowly tuning the model parameters over time. In a single-step process, as is the training of Naive Bayes models, such approaches may fail to capture sufficient information.

8 CONCLUSION

We introduced the first implementation of a federated Naive Bayes classifier under differential privacy. The relative performance of our approach to its centralized counterpart varies depending on the characteristics of the dataset. According to our Monte Carlo analysis, in most cases, a small privacy penalty must be paid to achieve the same accuracy level. However, when the local sensitivity in each federated data partition is much lower than the sensitivity of the whole dataset, we show that the federated Laplace mechanism better estimates the true parameters of the model.

Moreover, we demostrated that our algorithm can be easily extended to incorporate additional features such as online updates and full decentralization, while more research is needed to achieve byzantine resilience.

ACKNOWLEDGEMENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 813162: RAIS – Real-time Analytics for the Internet of Sports. The content of this paper reflects the views only of their author (s). The European Commission/ Research Executive Agency are not responsible for any use that may be made of the information it contains.

REFERENCES

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Awan, S., Luo, B., and Li, F. (2021). Contra: Defending against poisoning attacks in federated learning. In *European Symposium on Research in Computer Security*, pages 455–475. Springer.

Bernal, D. G., Giaretta, L., Girdzijauskas, S., and Sahlgren, M. (2021). Federated word2vec: Leveraging federated learning to encourage collaborative representation learning. *arXiv preprint arXiv:2105.00831*.

Blanchard, P., El Mhamdi, E. M., Guerraoui, R., and Stainer, J. (2017). Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30.

Dwork, C. (2008). Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer.

Geyer, R. C., Klein, T., and Nabi, M. (2017). Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*.

Giaretta, L. and Girdzijauskas, Š. (2019). Gossip learning: Off the beaten path. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 1117–1124. IEEE.

Guerraoui, R., Rouault, S., et al. (2018). The hidden vulnerability of distributed learning in byzantium. In *International Conference on Machine Learning*, pages 3521–3530. PMLR.

Hegedűs, I., Danner, G., and Jelasity, M. (2021). Decentralized learning works: An empirical comparison of gossip learning and federated learning. *Journal of Parallel and Distributed Computing*, 148:109–124.

Islam, T. U., Ghasemi, R., and Mohammed, N. (2022). Privacy-preserving federated learning model for healthcare data. In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0281–0287. IEEE.

Jelasity, M., Montresor, A., and Babaoglu, O. (2005). Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst.*, 23(3):219–252.

Ji, Z., Lipton, Z. C., and Elkan, C. (2014). Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*.

Kantarcıoglu, M., Vaidya, J., and Clifton, C. (2003). Privacy preserving naive bayes classifier for horizontally partitioned data. In *IEEE ICDM workshop on privacy preserving data mining*, pages 3–9.

Li, T., Li, J., Liu, Z., Li, P., and Jia, C. (2018). Differentially private naive bayes learning over multiple data sources. *Information Sciences*, 444:89–104.

Lopuhaä-Zwakenberg, M., Alishahi, M., Kivits, J., Klarenbeek, J., van der Velde, G.-J., and Zannone, N. (2021). Comparing classifiers' performance under differential privacy. In *International Conference on Security and Cryptography (SECRYPT)*.

Lyu, L., Yu, H., Ma, X., Sun, L., Zhao, J., Yang, Q., and Yu, P. S. (2020). Privacy and robustness in federated learning: Attacks and defenses. *arXiv preprint arXiv:2012.06337*.

Marchioro., T., Kazlouski., A., and Markatos., E. (2021). User identification from time series of fitness data. In *Proceedings of the 18th International Conference on Security and Cryptography - SECRYPT,*, pages 806–811. INSTICC, SciTePress.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. In Singh, A. and Zhu, J., editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR.

McSherry, F. D. (2009). Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30.

Ormándi, R., Hegedűs, I., and Jelasity, M. (2013). Gossip learning with linear models on fully distributed data. *Concurrency and Computation: Practice and Experience*, 25(4):556–571.

Rish, I. et al. (2001). An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46.

Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., and Backes, M. (2018). Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*.

Shafagh, H., Burkhalter, L., Hithnawi, A., and Duquennoy, S. (2017). Towards blockchain-based auditable storage and sharing of iot data. In *Proceedings of the 2017 on cloud computing security workshop*, pages 45–50.

Ullah, F., Edwards, M., Ramdhany, R., Chitchyan, R., Babar, M. A., and Rashid, A. (2018). Data exfiltration: A review of external attack vectors and countermeasures. *Journal of Network and Computer Applications*, 101:18–54.

Vaidya, J. and Clifton, C. (2004). Privacy preserving naive bayes classifier for vertically partitioned data. In *Proceedings of the 2004 SIAM international conference on data mining*, pages 522–526. SIAM.

Vaidya, J., Shafiq, B., Basu, A., and Hong, Y. (2013). Differentially private naive bayes classification. In *2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 1, pages 571–576. IEEE.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., and Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469.

Yi, X. and Zhang, Y. (2009). Privacy-preserving naive bayes classification on distributed data via semi-trusted mixers. *Information systems*, 34(3):371–380.

Zhang, H. (2004). The optimality of naive bayes. *Aa*, 1(2):3.

Zhu, L., Liu, Z., and Han, S. (2019). Deep leakage from gradients. *Advances in Neural Information Processing Systems*, 32.

APPENDIX

Deriving the Formulas for Online Updates

In order to derive the the online updates of our model, we consider a time-varying model where all the parameters $\nu^{(t)}, M^{(t)}, \mu^{(t)}, \sigma^{(t)}$ are defined at the t-th exchange between the central aggregator and one node of the network. According to this definition, denoting with τ_i the time at which iteration with node $D^{(i)}$ happens, the parameters of the model at time t should satisfy

$$\nu_y^{(t)} = \sum_{i:\tau_i<t} n_y^{(i)}, \qquad M_{fvy}^{(t)} = \sum_{i:\tau_i<t} m_{fvy}^{(i)}, \tag{19}$$

$$\mu_{fy}^{(t)} = \frac{\sum_{i:\tau_i<t} S_{fy}^{(i)}}{\sum_{i:\tau_i<t} n_y^{(i)}}, \quad [\sigma_{fy}^2]^{(t)} = \frac{\sum_{i:\tau_i<t} Q_{fy}^{(i)}}{\sum_{i:\tau_i<t} n_y^{(i)}} - \mu_{fy}^{(t)}. \tag{20}$$

Counting parameters such as $\nu^{(t)}$ and $M^{(t)}$ are simply updated by adding the new counts $n^{(i)}, m^{(i)}$ of the i-th node which is sending updates at time $t+1$

$$\nu_y^{(t+1)} = \nu_y^{(t)} + n_y^{(i)}, \quad M_{fvy}^{(t+1)} = M_{fvy}^{(t)} + m_{fvy}^{(t)}. \tag{21}$$

The mean $\mu_{fy}^{(t)}$, instead, can be updated by observing that

$$\mu_{fy}^{(t+1)} = \frac{\sum_{i':\tau_{i'}<t+1} S_{fy}^{(i')}}{\sum_{i':\tau_{i'}<t+1} n_y^{(i')}} \tag{22}$$

$$= \frac{\sum_{i':\tau_{i'}<t} n_y^{(i')}}{\sum_{i':\tau_{i'}<t+1} n_y^{(i')}} \frac{\sum_{i':\tau_{i'}<t} S_{fy}^{(i')}}{\sum_{i':\tau_{i'}<t} n_y^{(i')}} + \frac{S_{fy}^{(i)}}{\sum_{i':\tau_{i'}<t+1} n_y^{(i')}} \tag{23}$$

$$= \frac{\nu_y^{(t)}}{\nu_y^{(t+1)}} \mu_{fy}^{(t)} + \frac{S_{fy}^{(i)}}{\nu_y^{(t+1)}}, \tag{24}$$

where $S^{(i)}$ is the sample sum submitted by node $D^{(i)}$ at time t. Analogously, the estimated second moment at time $t+1$ is obtained as

$$\varsigma_{fy}^{(t+1)} = \frac{\nu_y^{(t)}}{\nu_y^{(t+1)}} \varsigma_{fy}^{(t)} + \frac{Q_{fy}^{(i)}}{\nu_y^{(t+1)}}. \tag{25}$$

The variance can be obtained by replacing $\varsigma_{fy}^{(t)} = [\sigma_{fy}^2]^{(t)} + [\mu_{fy}^{(t)}]^2$, which leads to

$$[\sigma_{fy}^2]^{(t+1)} = \frac{\nu_y^{(t)}}{\nu_y^{(t+1)}} ([\sigma_{fy}^2]^{(t)} + [\mu_{fy}^{(t)}]^2) + \frac{Q_{fy}^{(i)}}{\nu_y^{(t+1)}} - [\mu_{fy}^{(t+1)}]^2. \tag{26}$$