Towards a Realistic Decentralized Naive Bayes with Differential Privacy*

Lodovico Giaretta[1][0000−0002−0223−8907], Thomas Marchioro[2][0000−0003−3353−102X], Evangelos Markatos[2][0000−0003−3563−7733], and Sarunas Girdzijauskas[1][0000−0003−4516−7317]

[1] KTH Royal Institute of Technology,
Isafjordsgatan 22, 16440 Kista, Sweden
{lodovico,sarunasg}@kth.se
[2] Foundation for Research and Technology Hellas,
Nikolaou Plastira 100, 70013 Heraklion, Greece
{marchiorot,markatos}@ics.forth.gr

Abstract. This is an extended version of our work in [16]. In this paper, we introduce two novel algorithms to collaboratively train Naive Bayes models across multiple private data sources: Federated Naive Bayes and Gossip Naive Bayes. Instead of directly providing access to their data, the data owners compute local updates that are then aggregated to build a global model. In order to also prevent indirect privacy leaks from the updates or from the final model, our algorithms protect the exchanged information with differential privacy. We experimentally evaluate our proposed approaches, examining different scenarios and focusing on potential real-world issues, such as different data owner offering different amounts of data or requesting different levels of privacy. Our results show that both Federated and Gossip Naive Bayes achieve similar accuracy to a "vanilla" Naive Bayes while maintaining reasonable privacy guarantees, while being extremely robust to heterogeneous data owners.

Keywords: Federated learning · Gossip Learning · Differential privacy · Naive Bayes.

1 Introduction

Machine learning has reached unprecedented popularity in recent years, its successes enabling previously unthinkable digital services. However, training machine learning models typically requires the collection, storage and processing of large amounts of personal and potentially sensitive data, causing privacy and scalability issues.

* 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.

Thence, a variety of approaches have been developed to train machine learning models on decentralized, private data. Among these, the most popular is federated learning [17], which allows multiple data owners to cooperatively train a model without having to transfer or disclose their data. The training procedure under federated learning requires the data owners to iteratively compute local model updates and share them with a central aggregator. The aggregator, in turn, combines such updates to produce a global model, containing information from all data sources.

One limitation of federated learning is its requirement for a central, trusted entity to perform the aggregation step. Different approaches have been proposed to overcome this issue, the most notable being gossip learning [19], which instead employs peer-to-peer communication protocols to perform the aggregation.

However, federated and gossip learning alone cannot shield from all possible privacy leaks [27, 21]. Thus, they are often combined with differential privacy [11, 24, 7], a family of techniques to inject properly calibrated noise into the outputs shared by the data owners. When properly implemented, differential privacy provides strong mathematical guarantees that no private information can be inferred from the values that are shared.

While substantial efforts have been devolved to study differentially-private federated learning on deep models, in [16] we argued that, in many practical applications, simpler yet robust models like Naive Bayes classifiers are preferable. Therefore, we proceeded to provide the first (to our knowledge) implementation and evaluation of Federated Naive Bayes with differential privacy, showing that in most cases it can achieve nearly the same performance as a traditional, non-data-private counterpart.

In this extended version of [16], we provide additional algorithms and experimental results, with the focus on enabling certain realistic scenarios that were not discussed in the original work.

First, we experimentally evaluate cases where different data owners own substantially different amounts of data points or require different degrees of privacy. We provide insights into how these metrics affect the amount of noise injected by the data owners, and we show that Federated Naive Bayes is extremely robust to variable size of the data partitions and privacy budget distribution.

Additionally, we propose and evaluate Gossip Naive Bayes, a counterpart to Federated Naive Bayes that uses iterative gossip-based communications, replacing the central aggregator of the latter. Our results show that Gossip Naive Bayes converges to the same accuracy as Federated Naive Bayes in just a few iterations of the protocol, providing a high degree of scalability and privacy in fully-decentralized scenarios.

2 Background, notation, and terminology

2.1 Naive Bayes

Naive Bayes is well established as a simple-yet-effective machine learning algorithm for classification. Naive Bayes classifies data points according to bayesian

Symbol	Description
$\mathbf{x}$	Random variable
x	Observation of $\mathbf{x}$
$x^{(i)}$	Samples owned by i-th node
x_j	j-th sample
x_ϕ	Feature ϕ of x
$y^{(i)}$	Labels owned by i-th node
n_y	Count of data points in class y
$m_{x_\phi y}$	Count of data points with categorical feature ϕ equal to x_ϕ
$\mu_{\phi y}$	Average of numerical feature ϕ for class y
$\mu_{\phi y}$	Standard deviation of numerical feature ϕ for class y
$\mathcal{X}$	Set of possible values of x
$\mathcal{F}$	Set of features
$\mathcal{F}_{\text{cat}}$	Set of categorical features
$\mathcal{F}_{\text{num}}$	Set of numerical features
N_{nodes}	Number of nodes
$N^{(i)}_{\text{samples}}$	Number of data points owned by the i-th node

Table 1. Notation used in the paper.

inference, following the maximum a posteriori probability (MAP) criterion:

$$\begin{aligned}\hat{y}(x) &= \arg\max_{y\in\mathcal{Y}} \Pr[\mathbf{y}=y|\mathbf{x}=x] \\ &= \arg\max_{y\in\mathcal{Y}} \Pr[\mathbf{y}=y]\Pr[\mathbf{x}=x|\mathbf{y}=y].\end{aligned} \tag{1}$$

An underlying assumption of this algorithm is that the features $\mathbf{x}_1,\ldots,\mathbf{x}_{\Phi}$ are independent when conditioned with respect to the class variable Y. The MAP criterion under such assumption becomes:

$$\hat{y}(x) = \arg\max_{y\in\mathcal{Y}} \Pr[\mathbf{y}=y]\prod_{\phi\in\mathcal{F}}\Pr[\mathbf{x}_\phi=x_\phi|\mathbf{y}=y]. \tag{2}$$

Equation (2) is the criterion adopted by Naive Bayes to predict the most likely class $\hat{y}\in\mathcal{Y}$ to which a sample $x\in\mathcal{X}$ belongs. However, the a priori probability distribution of the classes $\Pr[\mathbf{y}=y]$ and the conditional likelihood of each feature $\Pr[\mathbf{x}_\phi=x_\phi|\mathbf{y}=y]$ are unknown. "Training" a Naive Bayes classifier consists of estimating such probabilities from the training dataset $\mathcal{D}_{\text{train}}$, which is a collection of example/class pairs $(x_j,y_j)\in\mathcal{X}\times\mathcal{Y}$. The prior probabilities of the classes are estimated as the normalized frequency for each class, i.e.,

$$p_y = \frac{n_y}{\sum_{y'\in\mathcal{Y}} n_{y'}},\ \text{with } n_y = \sum_{j=1}^{|\mathcal{D}_{\text{train}}|}\chi\{y_j=y\}, \tag{3}$$

where χ denotes the indicator function. Simply put, n_y is the number of training examples that belong to class y. The way conditional likelihoods are estimated depends on the nature of the features.

– **Categorical features**: For categorical features, the most common approach is again to compute the normalized frequency, this time normalized with respect to the categories:

$$p_{x_\phi|y} = \frac{m_{x_\phi y}}{\sum_{x'_\phi \in \mathcal{X}_\phi} m_{x'_\phi y}}, \text{ with } m_{x_\phi y} = \sum_{j=1}^{|\mathcal{D}_{\text{train}}|} \chi\{x_{j\phi} = x_\phi \wedge y_j = y\}. \quad (4)$$

– **Numerical features**: For numerical features, the conditional likelihood $\Pr[\mathbf{x}_\phi = x_\phi | \mathbf{y} = y]$ is proportional to a probability density function (PDF) $\rho_{\mathbf{x}_\phi|\mathbf{y}}$ in eq. (2). A PDF cannot be estimated by counting occurrences, since the number of possible values is infinite. Instead, the most common approach is to assume that the underlying PDF follows a certain parametric distribution. The usual choice is a normal distribution, in which case the algorithm takes the name of *Gaussian Naive Bayes*. A normal distribution is characterized by its mean $\mu_{\phi y}$ and variance $\sigma^2_{\phi y}$, which are estimated as follows:

$$\mu_{\phi y} = \frac{1}{n_y} \sum_{j:y_j=y} x_{j\phi}, \; \sigma^2_{\phi y} = \frac{1}{n_y - 1} \sum_{j:y_j=y} (x_{j\phi} - \mu_{\phi y})^2 . \quad (5)$$

The estimate of the probability density for x_ϕ, conditioned w.r.t. class y, is computed as

$$\rho_{\mathbf{x}_\phi|\mathbf{y}}(x_\phi|y) = \frac{1}{\sqrt{2\pi\sigma^2_{\phi y}}} \exp\left(-\frac{(x_\phi - \mu_{\phi y})^2}{2\sigma^2_{\phi y}}\right). \quad (6)$$

Practical computation and numerical stability Computing the posterior probabilities according to eq. (2) is problematic in terms of numerical stability, as the products tends to quickly vanish for a large number of features. For this reason, we calculate the posterior log-probabilities, which preserve the $\arg\max$ value. Log-probabilities allow to rewrite products as sums:

$$\hat{y}(x) = \arg\max_{y \in \mathcal{Y}} \log\left(p_y \prod_{\phi \in \mathcal{F}} p_{x_\phi|y}\right) \quad (7)$$

$$= \arg\max_{y \in \mathcal{Y}} \log p_y + \sum_{\phi \in \mathcal{F}} \log p_{x_\phi|y} \quad (8)$$

The prior log-probabilities can be simplified as

$$\log p_y = \log\left(\frac{n_y}{\sum_{y' \in \mathcal{Y}} n_{y'}}\right) = \log n_y - \log\left(\sum_{y' \in \mathcal{Y}} n_{y'}\right). \quad (9)$$

The normalization term $\log\left(\sum_{y' \in \mathcal{Y}} n_{y'}\right)$ in eq. (9) is common to all the classes, and can thus be neglected. Likewise, for categorical features, the conditional

log-likelihoods become

$$\log p_{x_\phi|y} = \log\left(\frac{m_{x_\phi y}}{\sum_{x'_\phi \in \mathcal{X}_\phi} m_{x'_\phi y}}\right) = \log m_{x_\phi y} - \log n_y. \tag{10}$$

For numerical features, the conditional log-likelihoods are proportional to

$$\log p_{x_\phi|y} \propto -\log \sigma_{\phi y} - \frac{(x_\phi - \mu_{\phi y})^2}{2\sigma^2_{\phi y}}. \tag{11}$$

The normalization terms can be neglected when computing the arg max, since they are common to all classes.

Algorithm 1 Naive Bayes prediction

Require: Sample x', prior counts n, conditional counts m, means μ and standard deviations σ
  for all classes $y = 1, \ldots, |\mathcal{Y}|$ **do**
    $\text{score}_y(x') \leftarrow \log(n_y)$ ▷ Initialize the scores with the log of the prior counts
    for all categorical features $\phi \in \mathcal{F}_{\text{cat}}$ **do**
      $\text{score}_y(x') \leftarrow \text{score}_y(x') + \log(m_{x'_\phi y})$
    end for
    for all numerical features $\phi \in \mathcal{F}_{\text{num}}$ **do**
      $\text{score}_y(x') \leftarrow \text{score}_y(x') - \log(\sigma_{\phi y}) + \frac{(x'_\phi - \mu_{\phi y})^2}{2\sigma^2_{\phi y}}$
    end for
  end for
  $\bar{y}' \leftarrow \text{ArgMax}_y \text{score}(x')$
  return predicted class $\bar{y}'$

2.2 Differential privacy

Introduced by Dwork et al. [6], differential privacy is an umbrella term covering different techniques to protect the output of an aggregation algorithm (henceforth, aggregated query) against membership inference. Differentially private mechanisms perturb the aggregated queries with noise, in a way that makes it hard to deduce whether a certain data point was included in the aggregation. The level of protection provided by differential privacy is determined by a parameter ε, called *privacy budget*. A lower privacy budget implies a higher level of privacy, but also requires a higher amount of noise. Formally, a randomized query $\tilde{\mathbf{q}}$ satisfies ε-differential privacy if for all adjacent[3] datasets $\mathcal{D}, \mathcal{D}'$, it holds

$$\Pr[\tilde{\mathbf{q}}(\mathcal{D}) \in \mathcal{O}] \leq \varepsilon \Pr[\tilde{\mathbf{q}}(\mathcal{D}') \in \mathcal{O}], \ \forall \mathcal{O} \subseteq \mathcal{Q}. \tag{12}$$

[3] Two datasets are adjacent if they differ by exactly one data point.

Equation (12) means that the probability for the query output falling in any subset $\mathcal{O}$ of the output space $\mathcal{Q}$ should not change "too much" if one data point is replaced.

Laplace mechanism. In our algorithm, we use the Laplace mechanism to enforce differential privacy on aggregated queries. This mechanism perturbs the query with additive noise that follows a Laplace distribution:

$$L_\varepsilon(q(\mathcal{D})) = q(\mathcal{D}) + \Xi,\ \Xi \sim \mathrm{Lap}\left(0, \frac{\Delta_q}{\varepsilon}\right). \tag{13}$$

As apparent from eq. (13), the scale of the noise is proportional to Δ_r and inversely proportional to the privacy budget ε. The value Δ_q is called *sensitivity* of the query, and can be defined in two ways:

- global sensitivity: $\Delta_q = \max_{\mathcal{D},\mathcal{D}'} |q(\mathcal{D}) - q(\mathcal{D}')|$;
- local sensitivity: $\Delta_q(\mathcal{D}) = \max_{\mathcal{D}':\mathcal{D}'=\mathcal{D}\setminus\{x_j\},x_j\in\mathcal{D}} |q(\mathcal{D}) - q(\mathcal{D}')|$.

Both notions of sensitivity guarantee that differential privacy is satisfied. However, when local sensitivity is used, the level of noise to be applied depends on the dataset $\mathcal{D}$. With global sensitivity, the noise is determined just by the query and the privacy budget.

Within the scope of our work, the Laplace mechanism is applied to two queries: Sum queries, and Count queries. A Count query simply counts how many elements satisfy a certain condition. When replacing an element in the dataset, the count can change by at most 1, meaning that the global sensitivity is $\Delta_q = 1$ for Count queries. Sum queries, instead, require to add all elements satisfying the condition. Unless the values are in a limited range, the maximum change in a Sum query cannot be determined in advance. Therefore, this must be estimated as a local sensitivity.

Properties In the design of Federated and Gossip Naive Bayes, we make use of two well-known properties of differential privacy:

- *Sequential composition*: If ℓ independent random queries $\tilde{\mathbf{q}}_1(\mathcal{D}), \ldots, \tilde{\mathbf{q}}_\ell(\mathcal{D})$ are computed on the same data $\mathcal{D}$ under $\frac{\varepsilon}{\ell}$-differential privacy, then any function of them $g(\tilde{\mathbf{q}}_1(\mathcal{D}), \ldots, \tilde{\mathbf{q}}_\ell(\mathcal{D}))$ satisfies ε-differential privacy.
- *Parallel composition*: If ℓ independent random queries $\tilde{\mathbf{q}}_1(\mathcal{D}^{(1)}), \ldots, \tilde{\mathbf{q}}_\ell(\mathcal{D}^{(\ell)})$ are computed on disjoint subsets of $\mathcal{D}$ under ε-differential privacy, then any function of them $g(\tilde{\mathbf{q}}_1(\mathcal{D}^{(1)}), \ldots, \tilde{\mathbf{q}}_\ell(\mathcal{D}^{(\ell)}))$ satisfies ε-differential privacy.

2.3 Federated and gossip learning

Several techniques have been proposed to perform distributed machine learning on private, horizontally-partitioned data sources. The most common and well-studied approach is federated learning [17]. Being typically deployed with deep learning models, federated learning consists of an iterative approach. In each

iteration, the devices holding the private data compute a gradient based on a batch of local data. These local gradients are then sent to a central entity, which aggregates them into a global gradient, modifies the model weights accordingly, and distributes the updated model to all participants for the next iteration.

The need for a central aggregator may limit the scalability of federated learning and introduce robustness and trust issues [1]. Gossip learning [19, 8] is a less-studied approach that overcomes these issues through decentralization and gossip communication protocols [13]. Data-owning devices directly share their locally-updated models with each other in a peer-to-peer fashion. More specifically, at regular intervals, each device will produce a new model by merging the two most recent models received from peers and then performing a local training step. This new model is then forwarded to a randomly-chosen peer.

Both federated and gossip learning can collaboratively train a model without explicit data sharing among participants. However, neither of them is designed to prevent implicit data leaks. These may occur during training, through the models or gradients shared in each iteration [27] or afterwards, through the final model produced by the process [21]. Therefore, previous works have integrated differential privacy with federated learning to protect from these leaks. Differential privacy can be applied on each local gradient before sharing [24], to ensure the privacy of individual data points, or within the central aggregator [7], thus hiding the identity of whole data owners participating in the protocol.

Both federated and gossip learning have been studied mostly in the context of deep learning models, where the iterative gradient-based approaches described above are necessary. Naive Bayes classifiers, on the other hand, are built based on simple statistics that can be computed by a single pass over the dataset. Furthermore, the focus of both approaches is typically on *massively-distributed* scenarios, with large number of devices each holding few data points. Fewer works [7, 3] have considered situations where a relatively small number of data brokers each contribute many data points from different individuals.

3 Algorithms

In this section, we describe our design for Federated and Gossip Naive Bayes. The setting is similar for both algorithms, and can be summarized as follows. Each data owner (henceforth, node) $i \in \{1, \ldots, N_{\text{nodes}}\}$ has a local dataset $\mathcal{D}^{(i)} = (x^{(i)}, y^{(i)})$ of multiple labeled data points. Nodes are unwilling to share their data in plain text, but are willing to participate in the training process of a Naive Bayes model by sharing the required dataset statistics, as long as the information they disclose does not leak information about individual data points. In other words, the information disclosed by each node should satisfy the following property:

> For all labeled data points $(x_j^{(i)}, y_j^{(i)})$ in a data partition $\mathcal{D}^{(i)}$, it should not be possible to infer whether $(x_j^{(i)}, y_j^{(i)})$ is present in $\mathcal{D}^{(i)}$ from the disclosed information.

This guarantees resilience against membership inference and can be achieved by protecting disclosed information with differential privacy.

In both Federated and Gossip Naive Bayes, the training process starts with all nodes independently computing their differentially-private model statistics, which we refer to as local updates. Then, these updates are aggregated to estimate the complete dataset statistics required to construct a Naive Bayes classifier, as introduced in section 2.1. However, the way this aggregation is performed differs in the two approaches. Federated Naive Bayes relies on a central aggregator, which handles the collection and merging of the updates. In Gossip Naive Bayes, instead, the updates are exchanges between nodes in a peer-to-peer fashion.

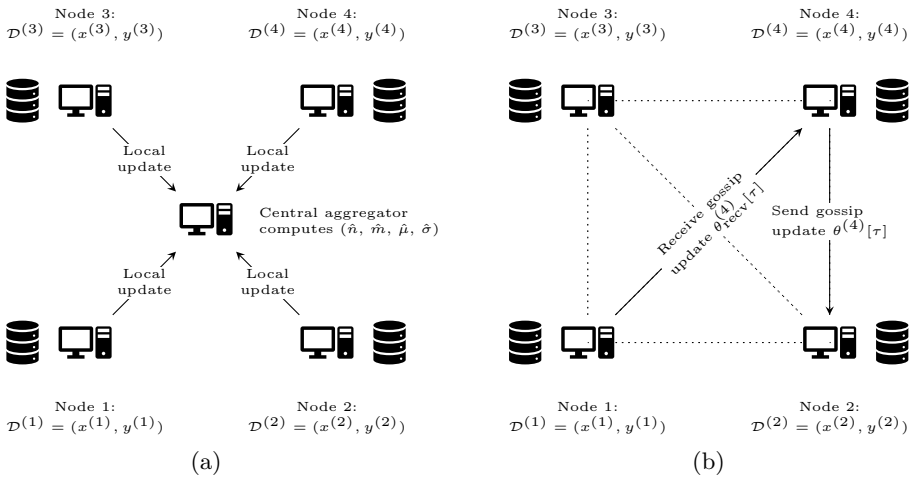

Fig. 1. Settings for Federated Naive Bayes (a) and Gossip Naive Bayes (b).

3.1 Local Update Computation

The local updates are essentially a collection of aggregated parameters that should allow to compute the final model, protected under differential privacy. More specifically, each update is a noisy version of the tuple ($n^{(i)}$, $m^{(i)}$, $S^{(i)}$, $Q^{(i)}$), where these parameters are defined as follows:

- $n^{(i)}$ is a vector of prior counts $n_y^{(i)}$ for each class $y = 1, \ldots, |\mathcal{Y}|$, computed according to eq. (3);
- $m^{(i)}$ contains the counts $m^{(i)}_{x_\phi y}$ of each category x_ϕ for each categorical feature $\phi \in \mathcal{F}_{\text{cat}}$ for each class, computed according to eq. (4);
- $S^{(i)}$ is a matrix of sums

$$S^{(i)}_{\phi y} = \sum_{j : y_j^{(i)} = y} x^{(i)}_{j\phi} \tag{14}$$

computed for each numerical feature $\phi \in \mathcal{F}_{\text{num}}$ and for each class y;

– $Q^{(i)}$ is a matrix of sums of squares

$$Q^{(i)}_{\phi y} = \sum_{j : y^{(i)}_j = y} \left(x^{(i)}_{j\phi} \right)^2 \tag{15}$$

computed for each numerical feature $\phi \in \mathcal{F}_{\text{num}}$ and for each class y.

All the parameters are perturbed with noise in order to achieve differential privacy. The privacy budget must be distributed equally between the different queries, following the sequential and parallel composition properties of differential privacy (as described in section 2.2). According to parallel composition, the same query executed on different classes counts as a single query from the standpoint of privacy budget distribution. This is due to the fact that such queries are executed on disjoint subsets of the local dataset $\mathcal{D}^{(i)}$. On the other hand, privacy budget must be equally distributed between multiple queries that affect the same class. There is 1 such query to compute $n^{(i)}$, $|\mathcal{F}_{\text{cat}}|$ queries to compute $m^{(i)}$ for all categorical features, and $2|\mathcal{F}_{\text{num}}|$ queries to compute $S^{(i)}$ and $Q^{(i)}$. Hence, each parameter $(n^{(i)}, m^{(i)}, S^{(i)}, Q^{(i)})$ is perturbed with the Laplace mechanism with privacy budget

$$\varepsilon' = \frac{\varepsilon}{1 + |\mathcal{F}_{\text{cat}}| + 2|\mathcal{F}_{\text{num}}|}, \tag{16}$$

yielding $(\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)})$. The values $\tilde{n}^{(i)}_y$ and $\tilde{m}^{(i)}_{x_\phi y}$ are computed via Count queries, thus the sensitivity is $\Delta_q = 1$. The values $S^{(i)}_{\phi y}$ and $Q^{(i)}_{\phi y}$, instead, are computed via Sum queries, meaning that their sensitivity is estimated from the local data, and varies between different features and classes: it is $\Delta_q = \max |x^{(i)}_{j\phi}|$ for $S^{(i)}_{\phi y}$, and $\Delta_q = \max |x^{(i)}_{j\phi}|^2$ for $Q^{(i)}_{\phi y}$. The randomization of each query via the Laplace mechanism is done by sampling independent values from a Laplace distribution with mean 0 and scale Δ_q/ε, and adding such values to each original parameter. The overall procedure is described in algorithm 2.

3.2 Federated Aggregation

In the case of Federated Naive Bayes, once the local updates $(\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)})$ are computed by all nodes $i = 1, \ldots, N_{\text{nodes}}$, these are collected by the central aggregator and merged into the final model.

A Naive Bayes model requires the overall prior counts n, the category counts m, and the parameters μ and σ to define a normal distribution for numerical features. Prior and category counts are trivially estimated as

$$\hat{n} = \sum_{i=1}^{N_{\text{nodes}}} \tilde{n}^{(i)}, \; \hat{m} = \sum_{i=1}^{N_{\text{nodes}}} \tilde{m}^{(i)}. \tag{17}$$

Since $\hat{n}$ and $\hat{m}$ are collections of scalars, when we say that we "sum" them we imply that we perform an element-wise sum across the nodes dimension. For

Algorithm 2 Local update computed by the i-th node

Require: Local data $\mathcal{D}^{(i)} = (x^{(i)}, y^{(i)})$, privacy budget ε

$\varepsilon' \leftarrow \frac{\varepsilon}{1 + |\mathcal{F}_{\text{cat}}| + 2|\mathcal{F}_{\text{num}}|}$ ▷ Allocate the privacy budget among the queries

for all classes $y = 1, \ldots, |\mathcal{Y}|$ **do**

$n_y^{(i)} \leftarrow \textsc{Count}_j(\{j : y_j^{(i)} = y\})$ ▷ Count samples from class y

for all categorical features $\phi \in \mathcal{F}_{\text{cat}}$ **do**

for all categories $x_\phi = 1, \ldots, |\mathcal{X}_\phi|$ **do**

$m_{x_\phi y}^{(i)} \leftarrow \textsc{Count}_j(\{j : x_{j\phi}^{(i)} = x_\phi \wedge y_j^{(i)} = y\})$ ▷ Count samples from class y with feature ϕ equal to x_ϕ

$\xi \leftarrow \textsc{RandomSample}(\text{Lap}\left(0, \frac{1}{\varepsilon'}\right))$ ▷ Sample Laplace noise

$\tilde{m}_{x_\phi y}^{(i)} \leftarrow m_{x_\phi y}^{(i)} + \xi$ ▷ Add the noise to the query

end for

end for

for all numerical features $\phi \in \mathcal{F}_{\text{num}}$ **do**

$S_{\phi y}^{(i)} \leftarrow \textsc{Sum}_j(\{x_{j\phi}^{(i)} : y_j^{(i)} = y\})$

$\xi \leftarrow \textsc{RandomSample}(\text{Lap}\left(0, \frac{\max |x_{j\phi}|}{\varepsilon'}\right))$

$\tilde{S}_{\phi y}^{(i)} \leftarrow S_{\phi y}^{(i)} + \xi$

$Q_{\phi y}^{(i)} \leftarrow \textsc{Sum}_j(\{(x_{j\phi}^{(i)})^2 : y_j^{(i)} = y\})$

$\xi \leftarrow \textsc{RandomSample}(\text{Lap}\left(0, \frac{\max (x_{j\phi}^{(i)})^2}{\varepsilon'}\right))$

$\tilde{Q}_{\phi y}^{(i)} \leftarrow Q_{\phi y}^{(i)} + \xi$

end for

end for

return $(\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)})$

instance, this means that the collection $\hat{m}$ contains the counts for all categories of all features for every class, which are obtained by summing those of each node. Regarding the numerical features, the central aggregator first estimates the overall sums and sums of squares:

$$\hat{S} = \sum_{i=1}^{N_{\text{nodes}}} \tilde{S}^{(i)}, \ \hat{Q} = \sum_{i=1}^{N_{\text{nodes}}} \tilde{Q}^{(i)}. \tag{18}$$

These are leveraged to estimate the means and standard deviations of the features. For each feature and class, the mean value of the feature within the class can be estimated simply dividing the sum by the count, using the definition of sample average

$$\hat{\mu}_{\phi y} = \frac{\hat{S}_{\phi y}}{\hat{n}_y}. \tag{19}$$

Typically standard deviation is estimated by computing the root mean square difference between samples $x_{j\phi}$ from class y and $\mu_{\phi y}$, as in eq. (5). However, this would require an additional exchange with the nodes, as they would need to first receive the aggregated $\hat{\mu}_{\phi y}$ to be able to compute the sum of $(x_{j\phi} - \hat{\mu}_{\phi y})^2$ terms. Instead of doing so, we observe that the variance of a random variable $\mathbf{z}$ can be expressed as the difference between the second moment and the squared mean, i.e. $\text{Var}(\mathbf{z}) = \mathbb{E}[\mathbf{z}^2] - (\mathbb{E}[\mathbf{z}])^2$. The second moment of each feature can be estimated as

$$\hat{\varsigma}_{\phi y} = \frac{\hat{Q}_{\phi y}}{\hat{n}_y}, \tag{20}$$

and thus the variance is approximated as

$$\hat{\sigma}^2_{\phi y} = \hat{\varsigma}_{\phi y} - \hat{\mu}^2_{\phi y} = \frac{\hat{Q}_{\phi y}}{\hat{n}_y} - \hat{\mu}^2_{\phi y}. \tag{21}$$

Indeed, this is not the most accurate approximation, and contrarily to eq. (5), it can yield negative values. When that happens, we replace negative values with a small positive value, namely 10^{-6}, before taking the square root to obtain the estimated standard deviation. We do the same with prior and categorical counts, as in some instances the noise can turn small positive counts into negative values, especially for categorical ones. The central aggregation is summarized by algorithm 3.

Online updates In many practical cases, the exchanges between different nodes and the central aggregator will happen asynchronously. Furthermore, new nodes may join the collaborative training after the model has already been computed. In such cases, the simplest solution is to keep the collection of submitted updates $(\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)})$ by each users $i = 1, \ldots, N_{\text{nodes}}$, and just recompute the parameters adding the new updates. However, in the event that the complete collection of updates is not available, it is still possible to update the model with

Algorithm 3 Central aggregation

Require: Collection of local updates $\{(\tilde{n}^{(i)}, \tilde{m}^{(i)}, \tilde{S}^{(i)}, \tilde{Q}^{(i)}), i = 1, \ldots, N_{\text{nodes}}\}$
  $\hat{n} \leftarrow \textsc{Sum}_i(\{\tilde{n}^{(i)}, i = 1, \ldots, N_{\text{nodes}}\})$
  $\hat{m} \leftarrow \textsc{Sum}_i(\{\tilde{m}^{(i)}, i = 1, \ldots, N_{\text{nodes}}\})$
  $\hat{S} \leftarrow \textsc{Sum}_i(\{\tilde{S}^{(i)}, i = 1, \ldots, N_{\text{nodes}}\})$
  $\hat{Q} \leftarrow \textsc{Sum}_i(\{\tilde{Q}^{(i)}, i = 1, \ldots, N_{\text{nodes}}\})$
  for all classes $y = 1, \ldots, |\mathcal{Y}|$ **do**
    for all numerical features $\phi \in \mathcal{F}_{\text{num}}$ **do**
      $\hat{\mu}_{\phi y} \leftarrow \dfrac{\hat{S}_{\phi y}}{\hat{n_y}}$
      $\hat{\sigma}_{\phi y} \leftarrow \sqrt{\dfrac{\hat{Q}_{\phi y}}{\hat{n_y}} - \hat{\mu}^2_{\phi y}}$
    end for

  end for
  return $(\hat{n}, \hat{m}, \hat{\mu}, \hat{\sigma})$

a new update $(\tilde{n}^{(N_{\text{nodes}}+1)}, \tilde{m}^{(N_{\text{nodes}}+1)}, \tilde{S}^{(N_{\text{nodes}}+1)}, \tilde{Q}^{(N_{\text{nodes}}+1)})$. The parameters $\hat{n}$ and $\hat{m}$ can be updated as

$$\hat{n}[\tau+1] = \hat{n}[\tau] + \tilde{n}^{(N_{\text{nodes}}+1)}, \hat{m}[\tau+1] = \hat{m}[\tau] + \tilde{m}^{(N_{\text{nodes}}+1)} \tag{22}$$

where the τ temporal index is simply used to distinguish between the new and old parameters. The new mean values $\hat{\mu}_{\phi y}[\tau+1]$ are a weighted average computed between the old means $\hat{\mu}_{\phi y}[\tau]$ and the received sums $\tilde{S}^{(N_{\text{nodes}}+1)}_{\phi y}$

$$\hat{\mu}_{\phi y}[\tau+1] = \frac{\hat{n}[\tau]}{\hat{n}[\tau+1]}\hat{\mu}_{\phi y}[\tau] + \frac{1}{\hat{n}[\tau+1]}\tilde{S}^{(N_{\text{nodes}}+1)}_{\phi y}. \tag{23}$$

The same holds for the second moments and the newly received sums of squares, for which it holds

$$\hat{\varsigma}_{\phi y}[\tau+1] = \frac{\hat{n}[\tau]}{\hat{n}[\tau+1]}\hat{\varsigma}_{\phi y}[\tau] + \frac{1}{\hat{n}[\tau+1]}\tilde{Q}^{(N_{\text{nodes}}+1)}_{\phi y}. \tag{24}$$

If also the previous value of $\hat{\varsigma}_{\phi y}[\tau]$ was not kept, it can be retrieved from the standard deviation as $\hat{\varsigma}_{\phi y}[\tau] = (\hat{\sigma}_{\phi y}[\tau])^2 + (\hat{\mu}_{\phi y}[\tau])^2$.

3.3 Gossip Naive Bayes

While in Federated Naive Bayes the central aggregator can trivially compute the sum ($\hat{n}$, $\hat{m}$, $\hat{S}$, $\hat{Q}$) of the local updates ($\tilde{n}^{(i)}$, $\tilde{m}^{(i)}$, $\tilde{S}^{(i)}$, $\tilde{Q}^{(i)}$), in Gossip Naive Bayes this is not possible. Instead, each node maintains a local estimates ($\hat{n}^{(i)}[\tau]$, $\hat{m}^{(i)}[\tau]$, $\hat{S}^{(i)}[\tau]$, $\hat{Q}^{(i)}[\tau]$) of the global statistics, which is updated and shared with peers in each iteration τ of the gossiping process.

To do this, each node i keeps track of the last and next-to-last estimates received from other peers, notated as ($\hat{n}^{(i)}_{\text{recv}}[\tau]$, $\hat{m}^{(i)}_{\text{recv}}[\tau]$, $\hat{S}^{(i)}_{\text{recv}}[\tau]$, $\hat{Q}^{(i)}_{\text{recv}}[\tau]$) and

$(\hat{n}^{(i)}_{\text{prev}}[\tau],\ \hat{m}^{(i)}_{\text{prev}}[\tau],\ \hat{S}^{(i)}_{\text{prev}}[\tau],\ \hat{Q}^{(i)}_{\text{prev}}[\tau])$ respectively. At regular intervals, every node performs one gossiping iteration by computing each component $\hat{\theta}^{(i)}[\tau]$ in the local estimate $(\hat{n}^{(i)}[\tau],\ \hat{m}^{(i)}[\tau],\ \hat{S}^{(i)}[\tau],\ \hat{Q}^{(i)}[\tau])$, based on the corresponding local update $\tilde{\theta}^{(i)}$ and the corresponding values in the latest two estimates received from peers, as

$$\hat{\theta}^{(i)}[\tau] = \hat{\theta}^{(i)}_{\text{prev}}[\tau] + \hat{\theta}^{(i)}_{\text{recv}}[\tau] + \tilde{\theta}^{(i)} \tag{25}$$

The node then sends its updated estimate to a randomly-chosen peer in the network.

To gain an intuition of this update rule, it is useful to look at the system not from the perspective of the nodes, but rather of the estimates themselves. These can be seen as acting like random walks over the network of nodes. Disregarding the first right-hand side term, eq. (25) shows that, at each step of its random walk, the estimate adds to its own values the local updates of the currently-visited node. After T steps, the estimate would have visited each node on average T/N_{nodes} times, and thus with a sufficiently large T its values would converge to the correct $(\hat{n},\ \hat{m},\ \hat{S},\ \hat{Q})$, multiplied by a factor of T/N_{nodes}. This factor would then disappear when using the estimate to build a Naive Bayes classifier as explained in section 2.1.

With such behaviour, $O(N_{\text{nodes}})$ gossiping steps would be necessary to ensure that the latest estimate at each node has converged to the correct values, which would not be scalable. Thus, the first term in eq. (25) is introduced, which brings the requirement down to $O(\log N_{\text{nodes}})$. Intuitively, it allows each estimate to not only visit the current node and learn about its local update, but also "meet" the estimate that last visited the current node and learn of all the local updates that that estimate had previously witnessed. As that previous estimate will have typically visited the node no more than a few iterations priors, this amounts to an almost doubling of the number of local updates contributing to the estimate.

Algorithm 4 Gossip aggregation at the i-th node

Require: Local update $(\hat{n}^{(i)}[\tau],\ \hat{m}^{(i)}[\tau],\ \hat{S}^{(i)}[\tau],\ \hat{Q}^{(i)}[\tau])$, last two estimates received from peers $(t^{(i)}_{\text{recv}}[\tau],\ \hat{n}^{(i)}_{\text{recv}}[\tau],\ \hat{m}^{(i)}_{\text{recv}}[\tau],\ \hat{S}^{(i)}_{\text{recv}}[\tau],\ \hat{Q}^{(i)}_{\text{recv}}[\tau])$ and $(t^{(i)}_{\text{prev}}[\tau],\ \hat{n}^{(i)}_{\text{prev}}[\tau],\ \hat{m}^{(i)}_{\text{prev}}[\tau],\ \hat{S}^{(i)}_{\text{prev}}[\tau],\ \hat{Q}^{(i)}_{\text{prev}}[\tau])$

$t^{(i)}[\tau] \leftarrow t^{(i)}_{\text{recv}}[\tau] + 1$
$t^{(i)}_{\text{tot}}[\tau] \leftarrow t^{(i)}_{\text{prev}}[\tau] + t^{(i)}_{\text{recv}}[\tau] + 1$
$\hat{n}^{(i)}[\tau] \leftarrow \frac{t^{(i)}_{\text{prev}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{n}^{(i)}_{\text{prev}}[\tau] + \frac{t^{(i)}_{\text{recv}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{n}^{(i)}_{\text{recv}}[\tau] + \frac{1}{t^{(i)}_{\text{tot}}[\tau]}\tilde{n}^{(i)}$
$\hat{m}^{(i)}[\tau] \leftarrow \frac{t^{(i)}_{\text{prev}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{m}^{(i)}_{\text{prev}}[\tau] + \frac{t^{(i)}_{\text{recv}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{m}^{(i)}_{\text{recv}}[\tau] + \frac{1}{t^{(i)}_{\text{tot}}[\tau]}\tilde{m}^{(i)}$
$\hat{S}^{(i)}[\tau] \leftarrow \frac{t^{(i)}_{\text{prev}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{S}^{(i)}_{\text{prev}}[\tau] + \frac{t^{(i)}_{\text{recv}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{S}^{(i)}_{\text{recv}}[\tau] + \frac{1}{t^{(i)}_{\text{tot}}[\tau]}\tilde{S}^{(i)}$
$\hat{Q}^{(i)}[\tau] \leftarrow \frac{t^{(i)}_{\text{prev}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{Q}^{(i)}_{\text{prev}}[\tau] + \frac{t^{(i)}_{\text{recv}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{Q}^{(i)}_{\text{recv}}[\tau] + \frac{1}{t^{(i)}_{\text{tot}}[\tau]}\tilde{Q}^{(i)}$
return new local estimate $(t^{(i)}[\tau],\ \hat{n}^{(i)}[\tau],\ \hat{m}^{(i)}[\tau],\ \hat{S}^{(i)}[\tau],\ \hat{Q}^{(i)}[\tau])$

This fast aggregation of many local updates also brings an issue: the magnitude of the components in the estimate can quickly grow, leading to a loss of floating point precision after just a few gossiping steps. Thus, we include with each estimate a step counter t, which is increased with every node visited by the estimate, and which is used as a normalization factor to maintain all scalars within a constant range. Thus, eq. (25) becomes

$$\hat{\theta}^{(i)}[\tau] = \frac{t^{(i)}_{\text{prev}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{\theta}^{(i)}_{\text{prev}}[\tau] + \frac{t^{(i)}_{\text{recv}}[\tau]}{t^{(i)}_{\text{tot}}[\tau]}\hat{\theta}^{(i)}_{\text{recv}}[\tau] + \frac{1}{t^{(i)}_{\text{tot}}[\tau]}\tilde{\theta}^{(i)} \tag{26}$$

$$\text{where } t^{(i)}_{\text{tot}}[\tau] = t^{(i)}_{\text{prev}}[\tau] + t^{(i)}_{\text{recv}}[\tau] + 1 \tag{27}$$

Algorithm 4 shows the pseudocode for this aggregation step, while algorithm 5 presents a high-level view of the full gossiping protocol performed by each node.

Algorithm 5 Overall gossip protocol at the i-th node

Require: local dataset $\mathcal{D}^{(i)}$, gossiping delay Δ

$\tilde{\theta}^{(i)} \leftarrow$ ComputeLocalUpdate($\mathcal{D}^{(i)}$) ▷ ($\tilde{n}^{(i)}$, $\tilde{m}^{(i)}$, $\tilde{S}^{(i)}$, $\tilde{Q}^{(i)}$)
$\hat{\theta}_{\text{prev}} \leftarrow 0$ ▷ ($t^{(i)}_{\text{prev}}[\tau]$, $\hat{n}^{(i)}_{\text{prev}}[\tau]$, $\hat{m}^{(i)}_{\text{prev}}[\tau]$, $\hat{S}^{(i)}_{\text{prev}}[\tau]$, $\hat{Q}^{(i)}_{\text{prev}}[\tau]$)
$\hat{\theta}_{\text{recv}} \leftarrow 0$ ▷ ($t^{(i)}_{\text{recv}}[\tau]$, $\hat{n}^{(i)}_{\text{recv}}[\tau]$, $\hat{m}^{(i)}_{\text{recv}}[\tau]$, $\hat{S}^{(i)}_{\text{recv}}[\tau]$, $\hat{Q}^{(i)}_{\text{recv}}[\tau]$)
loop
  Wait(Δ)
  $\hat{\theta} \leftarrow$ Aggregate($\hat{\theta}_{\text{prev}}, \hat{\theta}_{\text{recv}}, \tilde{\theta}^{(i)}$) ▷ Algorithm 4
  $p \leftarrow$ RandomPeer
  Send($p, \hat{\theta}$)
end loop

function OnReceive($\hat{\theta}'$)
  $\hat{\theta}_{\text{prev}} \leftarrow \hat{\theta}_{\text{recv}}$
  $\hat{\theta}_{\text{recv}} \leftarrow \hat{\theta}'$
end function

At any point during the gossip protocol, each node can use its latest local estimate ($\hat{n}^{(i)}[\tau]$, $\hat{m}^{(i)}[\tau]$, $\hat{S}^{(i)}[\tau]$, $\hat{Q}^{(i)}[\tau]$) to build a local Naive Bayes classifier as shown in section 2.1. Once a sufficient number of gossiping steps have been performed, and the estimates have reached convergence, the local classifiers of every node will be almost identical to each other and to the one that would have been built by a federated approach.

4 Experiments

We thoroughly evaluate the performance of Federated and Gossip Nave Bayes on six popular benchmark datasets, all from the UCI repository[4]. The majority

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

of the datasets comprises either only numerical or only categorical features, with the exception of Adults, which has both types. The details of each dataset are summarized in table 2. For all datasets that do not provide a default train/test split, we perform a 90/10 split with random seed 42, for reproducibility. All the accuracy results reported in the rest of the section are computed on the test data, which are kept untouched until the evaluation phase.

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)

Table 2. Datasets used in the evaluation. Table from [16].

4.1 Federated Naive Bayes: Homogeneous Setting

In the first set of experiments we compare the accuracy of Federated Naive Bayes against two baselines: a centralized ε-differentially-private Naive Bayes implementation by Vaidya et al. [23] and a "vanilla" non-private implementation based on section 2.1.

We perform the evaluation on different $(N_{\text{nodes}}, \varepsilon)$ pairs, testing our algorithm for $N_{\text{nodes}} = 1, 10, 100, 1000$ and varying ε from 10^{-2} to 10^{1}. For each pair, we run a Monte Carlo experiment with 1000 trials and average the results. This allows to estimate the accuracy for an average execution of Federate Naive Bayes, accounting for both the variation introduced by differential privacy and by the data distribution. Our results are displayed in fig. 2.

The plots follow the expected behavior of a differentially-private machine learning algorithm. When ε is low, the noise introduced by the Laplace mechanism completely hinders the prediction. However, for increasing value of ε, Federated Naive Bayes quickly approaches the accuracy value of the original centralized Naive Bayes. The value of ε at which this happens depends on the specific dataset: in Skin Segmentation a value of 1 already yields maximum accuracy in all cases, while SPECT Heart represents the opposite extreme.

It is worth noting that Federated Naive Bayes is not always worse than its centralized counterpart. It provides better accuracy for small values of ε on Adult, and for all values of ε on Accelerometer. The reason behind this counter-intuitive behavior is that Federated Naive Bayes perturbs the numerical parameters ($\tilde{S}^{(i)}$, $\tilde{Q}^{(i)}$) based on the local sensitivity. When the training data are partitioned across multiple nodes, the local sensitivity at each node may decrease, reducing the scale of Laplace noise. This hypothesis is confirmed by

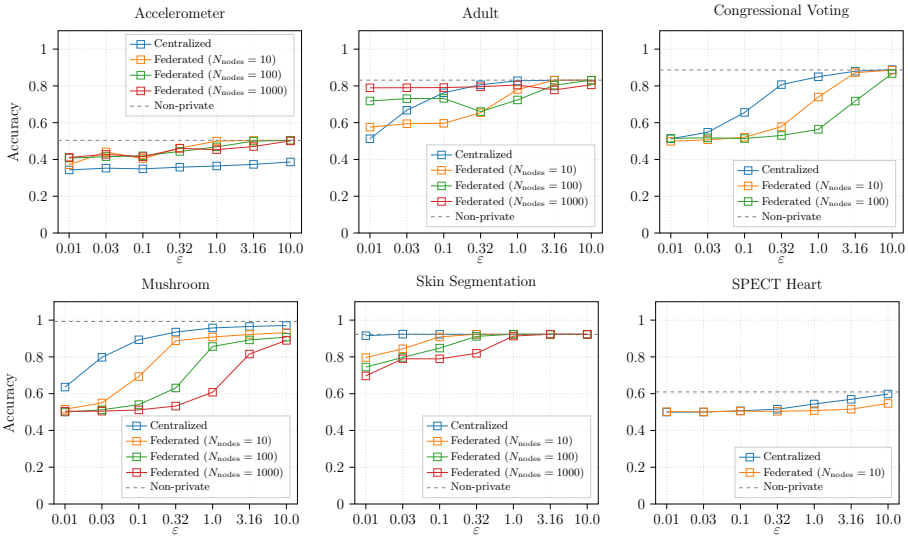

Fig. 2. Accuracy of Federated Naive Bayes for different values of ε and N_{nodes} vs centralized and non-private baselines. Image from [16].

fig. 3, which shows the local sensitivity of numerical features for $N = 10$ and $N = 100$, normalized with respect to the $N = 1$ case.

Note that the presence of numerical features does not guarantee this behavior. The distribution of the numerical features also matters, as it influences the chance of each node to have lower local sensitivity. This can be seen by looking at the Skin Segmentation dataset in figs. 2 and 3: despite having only numerical features, its local sensitivity does not decrease significantly as the number of nodes increases, and thus Federated Naive Bayes does not gain an advantage on its centralized counterpart.

For datasets of categorical features, on the other hand, the noise is applied with a global sensitivity of 1. In such cases, perturbing multiple partitions of the data only increases the overall amount of noise. Hence, the accuracy decreases for a larger number of nodes.

Overall, while the performance of Federated Naive Bayes depends on the characteristics of the dataset, in most cases it achieves similar accuracy to its centralized counterparts with the same or only slightly higher privacy budget.

4.2 Federated Naive Bayes: Heterogeneous Setting

In the experiments above, we assume that all nodes possess similar amounts of samples $N_{\text{samples}}^{(i)} \approx N_{\text{train}}/N_{\text{nodes}}$ and adopt the same privacy budget $\varepsilon^{(i)}$ for all nodes $i = 1, \ldots, N_{\text{nodes}}$. However, in realistic scenarios, nodes have a different number of data points and may have different privacy requirements. Therefore, we further evaluate Federated Naive Bayes in a more heterogeneous setting,

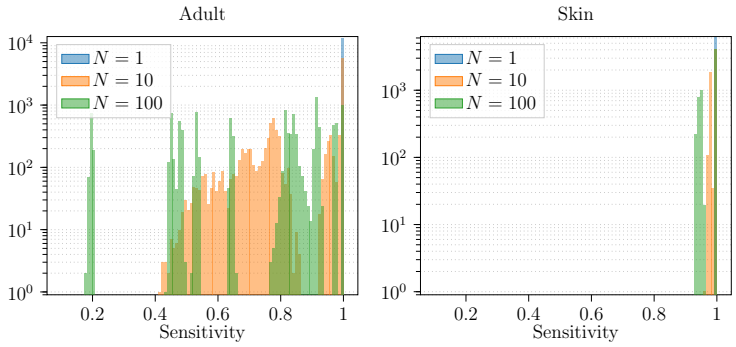

Fig. 3. Sensitivity distribution of $\tilde{S}^{(i)}$ for different numbers of nodes. Image from [16].

where we assign $N^{(i)}_{\text{samples}}$ and $\varepsilon^{(i)}$ according to some probability distribution. We start by running the same Monte Carlo experiment as above while distributing the training data according to the two following distributions:

- a uniform distribution $\mathcal{U}([1, 10])$;
- an exponential distribution, Exp(2);

meaning that each node samples a value from such distributions and gets assigned a number of data points proportional to the outcome.

Figure 4 shows the mean and standard deviation of the accuracy achieved on the Adult dataset with different node size distributions. The results suggest that Federated Naive Bayes perform equally well when different nodes have significantly different numbers of local samples.

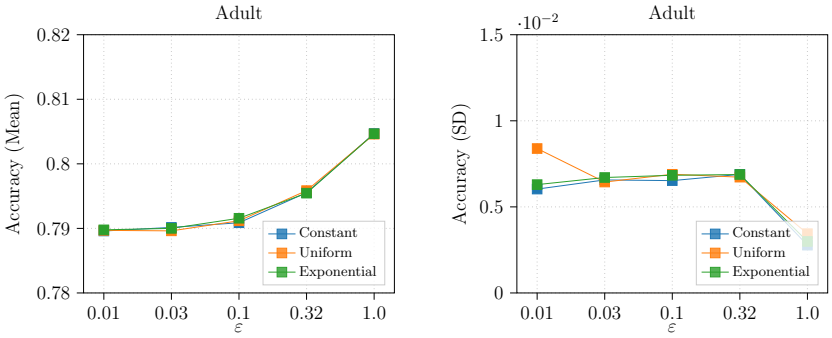

Fig. 4. Mean and standard deviation achieved by Federated Naive Bayes with $N_{\text{nodes}} = 1000$ and different distributions of $N^{(i)}_{\text{samples}}$, on the Adult dataset. Similar results are achieved on other datasets. Note that the y axis on the left plot does not start from 0.

Besides assessing the impact that variable $N^{(i)}_{\text{samples}}$ and $\varepsilon^{(i)}$ have on the results, one may wonder whether it makes sense to include all the updates in the final model. For example, if one node has a small number of data points, or a low privacy budget, one may think that it is not worth including its update in the model, as the extra noise more than compensates for the additional data points.

In order to determine whether this is the case, we run the following experiment. In a Federated Naive Bayes setting, we randomly distribute the entire training set across a number of nodes, such as the number of samples $N^{(i)}_{\text{samples}}$ at each node i follow a log-uniform distribution between 2 and $\log N_{train}/5$. We then assign a privacy budget $\varepsilon^{(i)}$ to each node, again according to a log-uniform distribution between 10^{-2} and 10^{1}. The reason behind this seemingly odd choice is that a log-uniform distribution implies that the node sizes are uniformly distributed in terms of "magnitude". In other words, a node has roughly the same likelihood of being assigned 10, 100, or 1000 data points. Likewise, a log-uniform privacy budget implies that the privacy requirements of the nodes are uniformly distributed, since the definition of differential privacy involves taking the exponential of the privacy budget.

After assigning data points and privacy budgets to all nodes, we compute the local updates normally, and measure the overall relative error as

$$\sum_{y} \frac{|\tilde{n}^{(i)}_{y} - n^{(i)}_{y}|}{n^{(i)}_{y}} + \sum_{y,x_\phi} \frac{|\tilde{m}^{(i)}_{x_\phi y} - m^{(i)}_{x_\phi y}|}{m^{(i)}_{x_\phi y}} + \sum_{y,\phi} \frac{|\tilde{S}^{(i)}_{\phi y} - S^{(i)}_{\phi y}|}{|S^{(i)}_{\phi y}|} + \sum_{y,\phi} \frac{|\tilde{Q}^{(i)}_{\phi y} - Q^{(i)}_{\phi y}|}{Q^{(i)}_{\phi y}} \quad (28)$$

These error values serve to quantify the amount of noise introduced by each node. We use each of them as a threshold, and exclude all the nodes with error above such threshold from the collaborative training.

Our results on two datasets are displayed in fig. 5, with the other datasets showing similar behaviours. The plot on the left simply show the distribution of $N^{(i)}_{\text{samples}}$ and $\varepsilon^{(i)}$ among the nodes. In addition, the nodes have a different color depending on their error value, with the more noisy nodes being assigned a brighter color. Unsurprisingly, the noise depends mostly on the value of ε, especially when the features are mostly numerical (such as in the Skin Segmentation dataset). However, N_{samples} also has a significant effect on the amount of noise, particularly in the presence of categorical features, as in the Adult dataset, as the sensitivity of COUNT queries is fixed and thus the corresponding relative noise is smaller.

For the central and right plots in fig. 5, we consider how the accuracy of a Federated Naive Bayes classifier would change if all nodes with noise level above a certain threshold were discarded from the aggregation. Based on the results, this kind of cutoff would not change the *typical* accuracy of the model. However choosing a very high or very low threshold can significantly influence the run-to-run variance. When the threshold is very high (i.e. virtually all contributions are accepted), this confirms the intuition that some nodes may be contributing more noise than information. On the other hand, when the threshold is very low, too

much data is lost and high accuracy can only be achieved if the remaining nodes happen to represent very well the overall population. This is made obvious in the right side plots, which highlight how the worst accuracy results are obtained when only a single node is below the threshold.

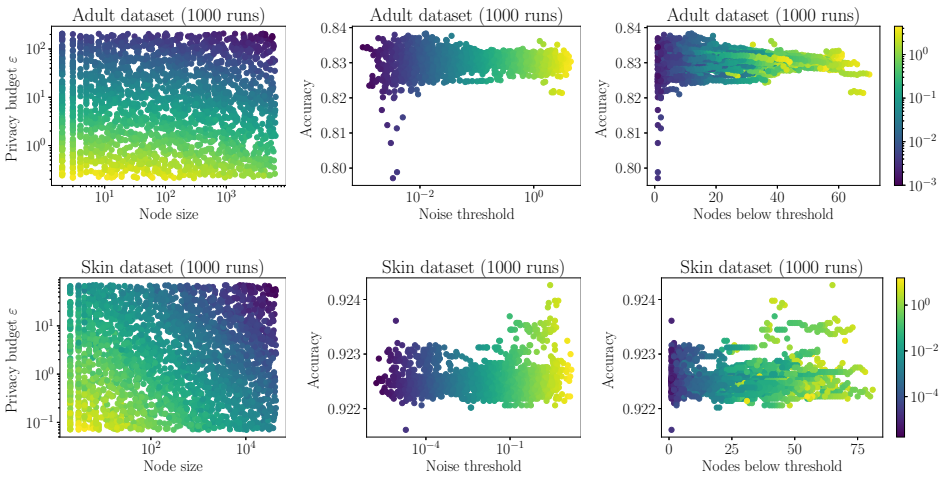

Fig. 5. Impact of noise thresholding on the accuracy of Federated Naive Bayes models.

4.3 Gossip Naive Bayes

As described in section 3.3, nodes in Gossip Naive Bayes compute the same differentially-private local updates as in Federated Naive Bayes, with the only change being the aggregation of these updates happens in a peer-to-peer fashion, rather than via a central aggregator. Thus, in our experiments we investigate whether this gossip-based aggregation can converge to the same results as a federated aggregation, and how many steps this convergence takes.

More specifically, we follow the same setup as in section 4.1 and apply both federated and gossip aggregation on the exact same runs, thus allowing a direct comparison not influenced by randomized dataset partitioning and noise generation.

Figure 6 shows how the accuracy of local Gossip Naive Bayes models evolves as more gossiping iterations are performed, on the Adult dataset with $N_{\text{nodes}} = 1000$ and $\varepsilon = 10^{\frac{1}{2}}$. Similar results were achieved on all datasets. In the first few iterations, as expected, the accuracy is very low, as the local estimates have only "seen" few local updates. Furthermore, the inter-quartile range of the accuracies is relatively wide. However, after just around 30 iterations, a vast majority of the nodes have reached the accuracy of the corresponding Federated Naive Bayes

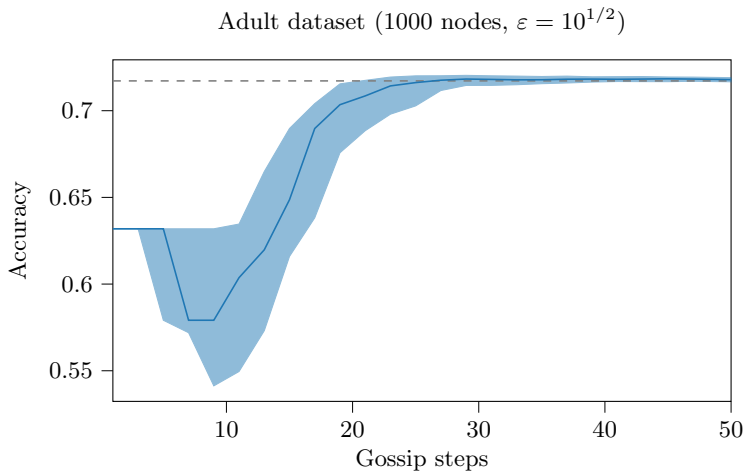

Fig. 6. Accuracy of Gossip Naive Bayes over multiple gossiping steps. The blue line is the median accuracy across 1000 participating nodes, the shaded area represents the inter-quartile range. The dotted line is the accuracy achieved by Federated Naive Bayes on the same set of local updates.

model, and this convergence further improves over additional iterations, with variance across different nodes reducing.

Figure 7 provides a more granular view, enabling a qualitative analysis of how individual Naive Bayes parameters converge over time. Prior probabilities, which on unbalanced datasets are typically the most influential parameters in the prediction, very quickly converge to the correct values. Conditional probabilities of categorical features and means of numerical features converge slightly slower and can show slight deviations from a federated aggregation after convergence. Finally, the standard deviations of numerical features significantly deviates from the expected values after convergence. These different converge speeds and deviations after convergence may be due to error propagation across the different statistics. Prior probabilities are computed directly from $\hat{n}$. Means (resp. conditional probabilities) depend from both $\hat{S}$ (resp. $\hat{m}$) and $\hat{n}$. Standard deviations depend on $\hat{Q}$, $\hat{n}$ and, importantly, the *squares* of the means. Yet, fig. 6 shows that these deviations do not affect the overall accuracy of the model, thus proving the robustness of our differentially-private Naive Bayes scheme.

5 Related work

Naive Bayes has shown consistent results in multiple applications [20, 26], drawing the attention of both researchers and practitioners. Therefore, it is not surprising that prior works explored the possibility of training Naive Bayes models across multiple data sources. Such works considered both the case of vertically

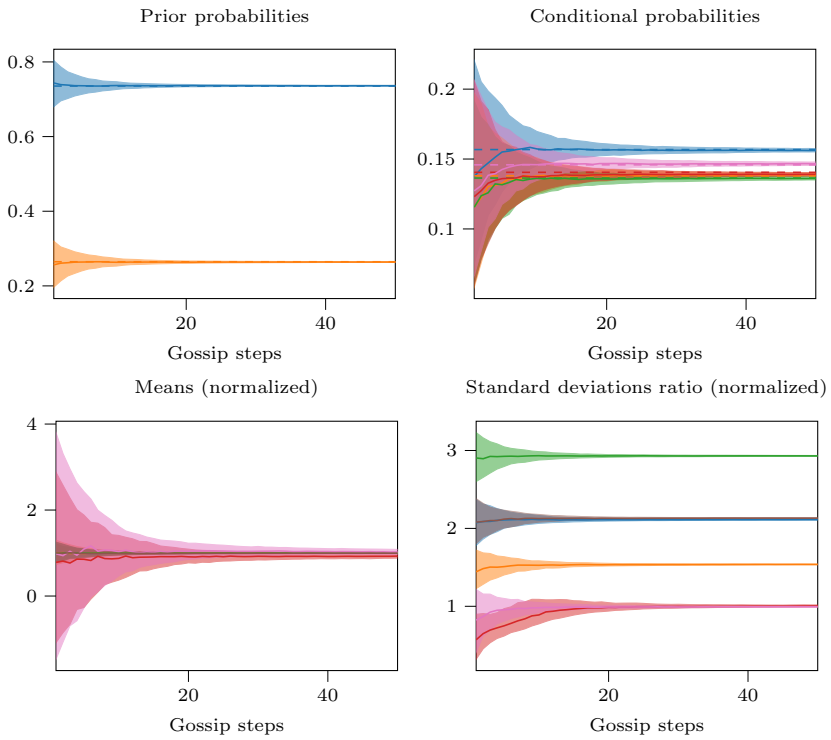

Fig. 7. Distribution of different Naive Bayes parameters across nodes over multiple steps of Gossip Naive Bayes, on the Adult dataset with $N_{\text{nodes}} = 1000$ and $\varepsilon = 10^{\frac{1}{2}}$. Lines and shaded areas of different colors represent median and inter-quartile range of each individual parameter.

partitioned data [22, 10], where each data source owns different features, and horizontally partitioned data [12, 14], where data sources own different data points comprising the same features. Our work belongs to the latter line of research. Prior designs of distributed Naive Bayes algorithms over horizontally distributed data mainly focused on protecting local updates via cryptographic methods, such as homomorphic encryption [12]. However, while homomorphic encryption is effective in hiding local updates, it does not prevent privacy leaks from the final model.

A centralized implementation of Naive Bayes under differential privacy, which served as an inspiration for this work, was introduced by Vaidya et al. [23]. A work by Li et al. [14] proposed the combined use of homomorphic encryption and differential privacy, but the utility-privacy tradeoff was not evaluated in the paper. Furthermore, the algorithm accounted only for categorical features. Another work by [25] relies on semitrusted mixers, which is based on a form of secure multiparty computation.

The works related to vertically partitioned data, instead, may be considered complementary to our work. A recent paper by Islam et al. [10] evaluated a federated version of Naive Bayes where the data are vertically partitioned. In such case, the Naive Bayes parameters for each feature can be computed locally by the node owning such feature. The final model is the collection of such parameters.

6 Future work

Non i.i.d. data distribution In this work we evaluated Federated and Gossip Naive Bayes for a varying node size and privacy budget. However, the distribution of the data among the nodes is kept i.i.d. in all our experiments. In order to further generalize our results, future work should explore different data distributions. One case may consist in considering class imbalances between the nodes. Another possibility is to investigate sample biases within the nodes. These may be simulated by clustering data points according to some similarity metric, and assigning each cluster to a node.

Feature selection A relevant component of Naive Bayes models is feature selection. Discarding less important features is essential to train a robust model with good generalization capability [5]. This is especially true when differential privacy is applied and needs to be divided between the features, and thus having irrelevant features only contributes to add more noise. A naive idea may be to have each node ranking features locally, and to combine local rankings to select most relevant features. Local ranking can be performed with several well-established feature ranking techniques [18].

Byzantine resilience The shift towards decentralized and private data sources leads to the emergence of a new challenge: byzantine data owners. Instead of following the correct protocol, these disseminate maliciously-generated model updates, typically with the goal of preventing model convergence, inserting specific biases in the final model, or causing the training process to leak specific

private data. Thus, byzantine-resilient federated learning [15, 2] has become an important research area. One of the most common approaches to achieve byzantine resilience is to use median-based aggregation functions [4, 9] to filter out contributions that diverge excessively from the majority of the nodes. However, Federated Naive Bayes relies on dataset statistics that are not directly amenable to this approach, as their magnitudes depend on data partition sizes. One option could be for the nodes to compute and share statistics that are independent of partition sizes (e.g. prior probabilities instead of prior counts), but this would significantly increase noise and numerical errors for under-represented classes. Thus, how to effectively achieve byzantine resilience for Federated and Gossip Naive Bayes is an open challenge.

7 Conclusions

This paper introduced two algorithms to collaboratively train a Naive Bayes model across multiple partitions, and under differential privacy guarantees: Federated and Gossip Naive Bayes. Both algorithms have been thoroughly evaluated, exploring the privacy-utility tradeoff for different distribution of node size and of privacy budget. Our results suggest that models trained with Federated or Gossip Naive Bayes offer comparable accuracy to their centralized counterpart, requiring only a slightly higher privacy budget depending on the dataset. Furthermore, both approaches are robust to varying distributions of the nodes' size and privacy budgets, thus making the suitable for real-world scenarios with heterogeneous data owners.

References

1. Alkathiri, A.A., Giaretta, L., Girdzijauskas, S., Sahlgren, M.: Decentralized word2vec using gossip learning. In: 23rd Nordic Conference on Computational Linguistics (NoDaLiDa 2021) (2021)
2. Awan, S., Luo, B., Li, F.: Contra: Defending against poisoning attacks in federated learning. In: European Symposium on Research in Computer Security. pp. 455–475. Springer (2021)
3. Bernal, D.G., Giaretta, L., Girdzijauskas, S., Sahlgren, M.: Federated word2vec: Leveraging federated learning to encourage collaborative representation learning. arXiv preprint arXiv:2105.00831 (2021)
4. Blanchard, P., El Mhamdi, E.M., Guerraoui, R., Stainer, J.: Machine learning with adversaries: Byzantine tolerant gradient descent. Advances in Neural Information Processing Systems **30** (2017)
5. Chen, J., Huang, H., Tian, S., Qu, Y.: Feature selection for text classification with naïve bayes. Expert Systems with Applications **36**(3), 5432–5435 (2009)
6. Dwork, C., Roth, A., et al.: The algorithmic foundations of differential privacy. Found. Trends Theor. Comput. Sci. **9**(3-4), 211–407 (2014)
7. Geyer, R.C., Klein, T., Nabi, M.: Differentially private federated learning: A client level perspective. arXiv preprint arXiv:1712.07557 (2017)

8. Giaretta, L., Girdzijauskas, v.: Gossip learning: Off the beaten path. In: 2019 IEEE International Conference on Big Data (Big Data). pp. 1117–1124 (2019). https://doi.org/10.1109/BigData47090.2019.9006216
9. Guerraoui, R., Rouault, S., et al.: The hidden vulnerability of distributed learning in byzantium. In: International Conference on Machine Learning. pp. 3521–3530. PMLR (2018)
10. Islam, T.U., Ghasemi, R., Mohammed, N.: Privacy-preserving federated learning model for healthcare data. In: 2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). pp. 0281–0287. IEEE (2022)
11. Ji, Z., Lipton, Z.C., Elkan, C.: Differential privacy and machine learning: a survey and review. arXiv preprint arXiv:1412.7584 (2014)
12. Kantarcıoglu, M., Vaidya, J., Clifton, C.: Privacy preserving naive bayes classifier for horizontally partitioned data. In: IEEE ICDM workshop on privacy preserving data mining. pp. 3–9 (2003)
13. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: 44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings. pp. 482–491. IEEE (2003)
14. Li, T., Li, J., Liu, Z., Li, P., Jia, C.: Differentially private naive bayes learning over multiple data sources. Information Sciences **444**, 89–104 (2018)
15. Lyu, L., Yu, H., Ma, X., Sun, L., Zhao, J., Yang, Q., Yu, P.S.: Privacy and robustness in federated learning: Attacks and defenses. arXiv preprint arXiv:2012.06337 (2020)
16. Marchioro, T., Giaretta, L., Markatos, E., Girdzijauskas, Š.: Federated naive bayes under differential privacy. In: 19th International Conference on Security and Cryptography (SECRYPT), JUL 11-13, 2022, Lisbon, Portugal. pp. 170–180. Scitepress (2022)
17. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.y.: Communication-Efficient Learning of Deep Networks from Decentralized Data. In: Singh, A., Zhu, J. (eds.) Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 54, pp. 1273–1282. PMLR (20–22 Apr 2017), https://proceedings.mlr.press/v54/mcmahan17a.html
18. Novakovic, J.: The impact of feature selection on the accuracy of naïve bayes classifier. In: 18th Telecommunications forum TELFOR. vol. 2, pp. 1113–1116 (2010)
19. Ormándi, R., Hegedűs, I., Jelasity, M.: Gossip learning with linear models on fully distributed data. Concurrency and Computation: Practice and Experience **25**(4), 556–571 (2013)
20. Rish, I., et al.: An empirical study of the naive bayes classifier. In: IJCAI 2001 workshop on empirical methods in artificial intelligence. vol. 3, pp. 41–46 (2001)
21. Salem, A., Zhang, Y., Humbert, M., Berrang, P., Fritz, M., Backes, M.: Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. arXiv preprint arXiv:1806.01246 (2018)
22. Vaidya, J., Clifton, C.: Privacy preserving naive bayes classifier for vertically partitioned data. In: Proceedings of the 2004 SIAM international conference on data mining. pp. 522–526. SIAM (2004)
23. Vaidya, J., Shafiq, B., Basu, A., Hong, Y.: Differentially private naive bayes classification. In: 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT). vol. 1, pp. 571–576. IEEE (2013)

24. Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q.S., Poor, H.V.: Federated learning with differential privacy: Algorithms and performance analysis. IEEE Transactions on Information Forensics and Security **15**, 3454–3469 (2020). https://doi.org/10.1109/TIFS.2020.2988575
25. Yi, X., Zhang, Y.: Privacy-preserving naive bayes classification on distributed data via semi-trusted mixers. Information systems **34**(3), 371–380 (2009)
26. Zhang, H.: The optimality of naive bayes. Aa **1**(2), 3 (2004)
27. Zhu, L., Liu, Z., Han, S.: Deep leakage from gradients. Advances in Neural Information Processing Systems **32** (2019)