Information Sciences 669 (2024) 120609

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fed-mRMR: A lossless federated feature selection method

Jorge Hermo [a,*], Verónica Bolón-Canedo [b], Susana Ladra [b]

[a] Universidade da Coruña, Department of Computer Science, A Coruña, 15071, A Coruña, Spain
[b] Universidade da Coruña, CITIC, A Coruña, 15071, A Coruña, Spain

ARTICLE INFO

Keywords:
Machine learning
Federated learning
Feature selection
Privacy preservation
Edge computing
Non-IID data

ABSTRACT

Feature selection has become a mandatory task in data mining, due to the overwhelming amount of features in Big Data problems. To handle this high-dimensional data and avoid the well-known curse of dimensionality, we need to pre-select an optimal subset of features to reduce redundant computations. Federated learning is a machine learning technique based on training an algorithm over many decentralized edge devices holding local rather than global data on a centralized server. Application of this technique is extending to fields such as self-driving cars, medicine and health, and Industry 4.0, where data privacy is compulsory. Feature selection through federated learning is a complicated task since suboptimal features calculated by feature selection methods may be different in heterogeneous datasets from different nodes. In this paper, we propose a lossless federated version of the classic minimum redundancy maximum relevance (mRMR) feature selection algorithm, called federated mRMR (fed-mRMR), which, without losing any effectiveness of the original mRMR method, is applicable to federated learning approaches and capable of dealing with data that are not independent and identically distributed (non-IID data).
Implementation can be found at: https://github.com/jorgehermo9/fed-mrmr

1. Introduction

The exponential growth in data volumes in recent years in fields such as social networks [1], bioinformatics [2], and physics [3]has led to the rise of Big Data. Although it might seem that more data are better for our models, this is not always true. A phenomenon in high-dimensional data is the curse of dimensionality, which, as Bellman [4] stated, causes model behavior to deteriorate as the dimensionality of latent space of our data increases. This effect can be mitigated, however, by applying dimensionality reduction or feature selection techniques.

Feature selection [5] is an essential data mining [6] and machine learning step in building a feasible model. It is well known that real datasets contain a large quantity of redundant [7] and irrelevant features, so it is good practice to remove those features before training the learning models. Feature selection is one of the most popular methods for reducing dimensionality, since it tries to build an optimal set of features (sometimes resulting suboptimal) that retains the main characteristics of the data while improving prediction accuracy and the computational speed of trained models [8]. Classic feature selection methods require the whole dataset in order to function properly, but this has not been an issue for traditional machine learning approaches, with all data stored on a centralized server. However, the rise of edge computing and privacy preservation awareness have led to a new machine learning

* Corresponding author.
E-mail addresses: jorge.hermo.gonzalez@udc.es (J. Hermo), veronica.bolon@udc.es (V. Bolón-Canedo), susana.ladra@udc.es (S. Ladra).

https://doi.org/10.1016/j.ins.2024.120609
Received 3 September 2023; Received in revised form 12 March 2024; Accepted 8 April 2024

Available online 10 April 2024

approach called federated learning [9], whereby local data are stored in each decentralized edge device, instead of being sent to a centralized server. Traditional feature selection methods, therefore, are not well suited to federated learning, since local device data are not usually independent and identically distributed (non-IID data) [10]. Just as traditional machine learning methods have needed to evolve to comply with data privacy standards, so too do feature selection methods need to evolve. To the best of our knowledge, apart from some works on privacy awareness [11,12], little effort has been invested in developing feature selection methods suitable for federated learning.

A classic feature selection algorithm is the minimum redundancy maximum relevance (mRMR) [13], based on mutual information, which is widely used when dealing with discrete features (a related method, minimum redundancy, performs well with microarray gene expression data [14]). As computational complexity is quadratic with the number of features, previous studies focused on speeding up mRMR computation times [15,16] using high-performance computing (HPC) techniques, but failed to take into account privacy preservation.

As mRMR is based on computing mutual information among features (and between features and classes), it does not require use of the whole raw dataset. We propose a new federated feature selection method that extracts certain statistics from the dataset and then applies the mRMR algorithm to rank and select relevant features. Our proposed method preserves privacy and achieves lossless federated feature selection, in that it keeps the same feature ranking as the original mRMR method.

Note that, although previous studies have been conducted of lossless federated learning [17], as far as we are aware, ours is the first proposal regarding lossless federated feature selection.

The remainder of this paper is organized as follows. Section 2 provides some background on feature selection, the mRMR algorithm, federated learning, and occurrence counting using bitmaps. Section 3 reviews the state of the art in federated feature selection. Section 4 describes our proposed modification to the mRMR algorithm. Section 5 shows experimentation benchmark results of our proposal. Finally, section 6 contains our conclusions.

2. Background

The following sub-sections discuss feature selection, the mRMR algorithm, federated learning, and occurrence counting, as essential background to our proposed feature selection method (fed-mRMR).

2.1. Feature selection

Feature selection is a widely used preprocessing technique that consists of selecting a subset of features applying an optimality criterion. While it is not guaranteed that this method finds the optimal subset of features, it at least finds a sub-optimal set that maximizes the desired criterion.

This technique attempts to separate features into three categories: relevant, redundant, and irrelevant. Only relevant features are ideally needed to solve a problem, as redundant features share information with relevant features and so are superfluous, and irrelevant features typically contribute nothing to our problem. This classification, however, is not usually binary, but fuzzy, so partitioning is not straightforward, given that there is a certain degree of relevance and redundancy for each feature, whose values may in turn vary in the presence of other features.

Selecting a subset of features allows us to reduce the dimensionality of the problem, thus mitigating the effects of the curse of dimensionality. Furthermore, it simplifies the learning models, making them easier to interpret for researchers and users, and also improves training speed and generalization capacity.

Feature selection methods can be divided into three different types; filter, wrapper, and embedded methods.

Filter methods [18], which are used prior to training the learning model, calculate a series of dataset metrics for each feature that are used together with the optimality criterion to select the subset that optimizes that criterion. The advantage of filter methods is that they are versatile, as they are independent of the learning model and can therefore be used as a preliminary step for any machine learning model. Some examples of these methods are RELIEF, which uses the concept of nearest neighbors to compute feature metrics, correlation-based feature selection (CFS), which ranks subsets of features using a correlation-based heuristic evaluation function, and mRMR [13]. We mainly focus on mRMR, which uses mutual information to obtain measures of feature relevance and redundancy and ultimately produces an ordered ranking of features.

Wrapper methods [19] use predictions made by the model to select the features. The model, trained successively with different subsets of features, finally selects the subset that produces the best results. The drawback of wrapper methods is that they are computationally expensive and can lead to overfitting.

Embedded methods follow a similar approach to wrapper methods, but use feature selection within the training process, rather than at the end of the training process. Since there is no separation between the training and feature selection processes, computation time over wrapper methods is improved, making embedded methods more feasible as a feature selection approach. LASSO [20] is a typical example of an embedded method.

In this work, we focused on adapting the mRMR algorithm for federated learning in a way that preserves privacy and retains the same ranking of features.

2.2. Minimum redundancy maximum relevance

The mRMR algorithm, first proposed by Peng et al. [13], is a widely used filter method for feature selection that uses mutual information to calculate measures of relevance and redundancy between the different features and the class label. Developed initially

for application to microarray gene expression data, for which it obtained very good results, nowadays it is used in telecommunications [21], medicine [22], network anomaly detection [23]. While we focus on application of this method to discrete data distributions, it can also be applied to continuous data distributions, as explained in the original proposal of this method.

The objective of the mRMR algorithm is to rank the whole set of features according to importance. To do this, it evaluates the relevance of a feature to the target and penalizes redundancy in the presence of other features. The goal is to determine maximum dependency between a set of features X and class c, using mutual information (I). Mutual information between a pair of discrete random variables (features) is defined in Eq. (1), which uses marginal probabilities $p(a)$ and $p(b)$ and joint probability $p(a,b)$.

$$I(A;B) = \sum_{b\in B}\sum_{a\in A} p(a,b)\log\left(\frac{p(a,b)}{p(a)p(b)}\right) \tag{1}$$

Mutual information has computational complexity $\mathcal{O}(|A|\times|B|)$, where A and B are the sets of distinct values for the features, but only in the case where we know in advance the probabilities that appear in the equation.

If we do not know the marginal probabilities in advance, we can estimate them from the samples we have in the data, as shown in Eq. (2), where n_a is the number of samples with the value a in the desired feature, and n is the total number of samples.

$$p(a) \approx \frac{n_a}{n} \tag{2}$$

We can also estimate the joint probability in a similar way, as per Eq. (3), where a and b are values of the desired features and $n_{a,b}$ is the number of samples that have both values a and b in those features.

$$p(a,b) \approx \frac{n_{a,b}}{n} \tag{3}$$

Since calculating n_a means that we have to run through all the samples sequentially, the computational complexity of estimating the probability of occurrence of a certain value in a particular feature is $\mathcal{O}(n)$.

If we have to compute the unknown probabilities used in Eq. (1), the computational complexity of calculating the mutual information between two features A and B is $\mathcal{O}(|A|\times|B|\times n)$. $p(a,b)$ has to be computed each time, although the estimates of $p(a)$ and $p(b)$ can be reused in further calculations.

Since the maximum dependency criterion is difficult to implement in high-dimensional spaces, an alternative used in the original proposal is the criterion of maximal relevance (max-relevance), defined in Eq. (4), where X is a set of features.

$$max\ D(X,c),\ D = \frac{1}{|X|}\sum_{x_i\in X} I(x_i;c) \tag{4}$$

The selected features may have a high level of redundancy, so applied as a penalty with the aim of selecting mutually exclusive features is the minimum redundancy criterion, defined in Eq. (5), where X is a set of features.

$$min\ R(X),\ R = \frac{1}{|X|^2}\sum_{x_i,x_j\in X} I(x_i;x_j) \tag{5}$$

Both criteria D and R can then be combined as mRMR, defined as Φ in Eq. (6)

$$max\ \Phi(D,R),\ \Phi = D - R \tag{6}$$

In practice, a greedy approach can be used with this criterion. Suppose we have selected a set of $m-1$ features, S_{m-1}, then the objective is to select the mth feature from the set $\{X \setminus S_{m-1}\}$ i.e., to select the feature that, when added to the currently selected feature set, maximizes the criterion Φ. In Eq. (7) we define the condition that this feature has to fulfill.

$$\begin{aligned} &max_{x_j\in X\setminus S_{m-1}}\left[D(S_{m-1}\cup\{x_j\}) - R(S_{m-1}\cup\{x_j\})\right] \equiv \\ &max_{x_j\in X\setminus S_{m-1}}\left[I(x_j;c) - \frac{1}{|S_{m-1}|}\sum_{x_i\in S_{m-1}} I(x_i;x_j)\right] \end{aligned} \tag{7}$$

As the original mRMR algorithm proposal has many redundant calculations,[1] we use a modified version called fast-mRMR, proposed by Ramírez-Gallego et al. [15] and described in Algorithm 1.

The fast-mRMR algorithm starts by computing relevance values for all input features, storing them in a vector (*relevancesVector*), and selecting the best features in terms of relevance, marked as the last selected feature. Then, using the mRMR criterion, the remaining features are selected in an iterative process (lines 12–25) until the desired number of features are selected (numFeaturesWanted). In this iterative process, the algorithm computes the mutual information between the last selected feature and the non-selected features. The best feature according to mRMR is added to the final set and marked as the last feature selected. One of the optimizations used in this algorithm is the *accumulatedRedundancy* vector. Since computing the mutual information between every pair of features

[1] Original proposal's source code can be found at. http://home.penglab.com/proj/mRMR/#c++

Algorithm 1 fast-mRMR.

```
INPUT: candidates, numFeaturesWanted
OUTPUT: selectedFeatures
selectedFeatures = ();
for each feature f in candidates do
    relevancesVector[f] = mutualInfo(f,class);
    accumulatexRedundancy[f] = 0;
end for
selected = getMaxRelevance(relevancesVector);
lastFeatureSelected = selected;
selectedFeatures.add(selected);
candidates.remove(selected);
while selectedFeatures.size() < numFeaturesWanted do
    max_mrmr = 0;
    for each feature fc in candidates do
        relevance = relevancesVector[fc];
        accumulatedRedundancy[fc] += mutualInfo(fc, lastFeatureSelected);
        redundancy = accumulatedRedundancy[fc]/selectedFeatures.size();
        mrmr= relevance - redundancy;
        if mrmr > max_mrmr then
            lastFeatureSelected = fc;
            max_mrmr = mrmr;
        end if
    end for
    selectedFeatures.add(lastFeatureSelected);
    candidates.remove(lastFeatureSelected);
end while
```

is costly, redundancy is accumulated in each iteration, and only computed is the mutual information between the set of non-selected features and the last selected feature.

This version of the algorithm uses the number of features to be selected to limit the number of comparisons between features. The greedy search used in fast-mRMR does not affect the final result (feature ranking), but does reduce the original complexity, as the iterative process (linear order) is limited to a small number of iterations (the number of features selected).

2.3. Federated learning

Federated learning (FL) [24] is a machine learning approach where a model is trained across multiple decentralized devices or servers holding local data samples. In a typical FL setting, the training process involves sending the model to local data sources, where it is trained and updated. The local updates are then aggregated to update the global model. This process is iterated so as to improve the model with each round. The key advantage of FL is that, by allowing for large-scale data to be used without the need to first pool the data in a central location, it maintains data confidentiality and adheres to privacy regulations.

The FL paradigm can be broadly categorized into two FL types: cross-silo and cross-device. Cross-silo FL involves a relatively small number of participants (organizations or institutions) with relatively large datasets. In this setting, each participant (silo) has a substantial quantity of data and FL takes place between these silos. This approach is common in scenarios where data sharing is restricted due to privacy and regulatory reasons, e.g., in the healthcare and finance sectors. Cross-device FL involves a large number of devices, often in the magnitude of millions (typically personal devices like smartphones or IoT devices), each with small datasets. While cross-device FL is more challenging due to device heterogeneity and network connection unreliability, it offers the potential to learn from a highly diverse and distributed dataset, which can lead to more robust and generalized models.

The two main FL approaches, horizontal division and vertical division, depend on how samples are divided. This fundamental difference in data alignment shapes how algorithms are designed and implemented in each FL scenario. Horizontal FL is used when different participants have datasets with the same feature space but different samples, e.g., two hospitals in different regions may have patient data in the same format but with different patient groups. Vertical FL applies when participants have different feature spaces but share the same sample space, e.g., a bank and a retail store may have different types of data (financial and purchasing habits) for the same customers.

Feature selection is a complex task in FL since most of the classic techniques cannot be applied to FL. Also, little research has been conducted on feature selection FL algorithms, since FL is mostly used for deep learning. While several algorithms for federated optimization are available, such as FedAvg [25] and SCAFFOLD [26], they all focus on adapting optimization algorithms to a federated environment. However, feature selection algorithms do not always rely on optimizing a loss function; this is the case with mRMR, so its adaptation to an FL environment requires more than a federated optimization algorithm.

2.4. Occurrence counting using bitmaps

As seen in Eq. (2), we need some way to count the number of occurrences of a given value in a certain feature. This counting is exemplified using the data in Table 1, which describes instances in terms of color and size as features and a class to which they belong.

Table 1
Example dataset.

Example dataset		
Color	Size	Class
blue	big	b
red	big	a
red	small	b
blue	small	a
blue	big	b

To count the number of instances with a certain color, the task is simple; go through the instances sequentially and sum the occurrences. However, to count the number of instances with a certain color and size, we need to take into account both features.

With the mRMR algorithm, the joint probability $p(a, b)$ (Eq. (1)) is estimated for each pair of features. For example, in the data in Table 1, to calculate the mutual information between the color and size features, we need to calculate the following joint probabilities: $p(blue, big)$, $p(blue, small)$, $p(red, big)$ and $p(red, small)$.

Thus, given two features A and B, the number of joint probabilities to calculate is $|A| \times |B|$, and for m features, the number of possible pairs of features is m^2. Algorithm 1 allows us to avoid redundant computations by considering only $\binom{m}{2}$ pairs of possible features, and, given that $\binom{m}{2} = \frac{m(m-1)}{2} \leq m^2$, we ultimately only have to estimate probabilities for a smaller number of pairs.

However, the calculations for the probability estimates grow quadratically with respect to the number of features, so naive implementation may not be efficient. To solve this problem, we can use feature bitmaps. A bitmap for a particular value v of a feature f is a vector of Boolean values in which 1 is the value at index i if instance i of the dataset has value v for feature f, and 0 otherwise. We denote the bitmap for the v value of feature f as $\vec{b}_v$.

For example, for Table 1, the bitmap for the value *blue* of the feature *color* is $\vec{b}_{blue} = (1\ 0\ 0\ 1\ 1)$, and the bitmap for the value *big* of the feature *size* is $\vec{b}_{big} = (1\ 1\ 0\ 0\ 1)$.

We can now calculate the number of instances that have a certain value in one or two features using the scalar product of two vectors. Thus, using the data in Table 1, the number of instances with the value *blue* in the feature *color* is calculated as $n_{blue} = \vec{b}_{blue} \cdot \vec{b}_{blue} = 3$, and the number of instances with the value *blue* in the feature *color* and the value *big* in the feature *size* is calculated as $n_{blue,big} = \vec{b}_{blue} \cdot \vec{b}_{big} = 2$.

3. Related work

The state of the art in federated feature selection is rapidly evolving, characterized as it is by innovative approaches that seek to optimize model performance while addressing the inherent challenges of FL, such as data privacy, communication efficiency, and data source heterogeneity. As mentioned above, vertical and horizontal FL approaches exist, both enabling efficient and secure feature selection

An example of vertical FL (VFL) is the federated stochastic dual-gate-based feature selection (FedSDG-FS) [27], which integrates feature selection with model training in VFL settings. This approach, designed with a focus on security and efficiency, uses, for instance, partial homomorphic encryption (PHE) and a randomized noise mechanism to ensure that important features are selected and the global model is optimized without compromising data privacy. Fundamentally underscored is the importance of balancing feature selection accuracy with privacy and communication efficiency needs in federated settings.

Another proposal for VFL feature selection is the LESS-VFL framework [28], which emphasizes communication efficiency. LESS-VFL is structured around pre-training, embedded component selection, and feature selection stages. It uses pre-trained model parameters and leverages proximal stochastic gradient descent (P-SGD) for implementation. Addressing the challenge of the high communication costs in federated settings, it optimizes the selection of significant features while minimizing the need for data exchange. LESS-VFL also considers privacy aspects by using information shared during VFL training, showcasing the importance of integrating privacy-preserving mechanisms in the feature selection process.

As an approach to horizontal FL (HFL), Hu et al. [29] introduced a framework with a credible third participant aimed at integrating optimal feature subsets from multiple participants while respecting privacy concerns. That study fills a research gap in feature selection in a privacy protection scenario, using evolutionary computation and swarm intelligence principles. The proposed algorithm, based on particle swarm optimization (PSO), incorporates two new operators: a feature assembly strategy with multi-participant cooperation, and a swarm initialization strategy. Classification accuracy is ensured in conjunction with privacy protection in feature selection from the datasets of different participants.

In Zhang et al. [30], for HFL in IoT networks, an unsupervised federated feature selection method called FSHFL eliminates non-essential features using a method for feature relevance outlier detection, enhanced by an improved one-class support vector machine. A feature relevance hierarchical clustering (FRHC) algorithm is also proposed for effective feature selection in HFL settings. The approach notably improves HFL system performance, as evidenced by enhanced global model accuracy and reduced training time. FSHFL is also significantly less costly in energy consumption terms compared to traditional FL methods.

$$\mathrm{B} = \begin{array}{c} \begin{matrix} \vec{b}_{blue} & \vec{b}_{red} & \vec{b}_{big} & \vec{b}_{small} & \vec{b}_{a} & \vec{b}_{b} \end{matrix} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

Fig. 1. Example of the block matrix associated with the bitmap example from Section 2.4.

$$\mathrm{M} = B^T \cdot B = \begin{array}{c} \\ blue \\ red \\ big \\ small \\ a \\ b \end{array} \begin{array}{c} \begin{matrix} blue & red & big & small & a & b \end{matrix} \\ \begin{bmatrix} 3 & 0 & 2 & 1 & 1 & 2 \\ 0 & 2 & 1 & 1 & 1 & 1 \\ 2 & 1 & 3 & 0 & 1 & 2 \\ 1 & 1 & 0 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 & 0 \\ 2 & 1 & 2 & 1 & 0 & 3 \end{bmatrix} \end{array}$$

Fig. 2. Example of occurrences matrix computed from the multiplication of B and its transpose, using the example depicted in Fig. 1.

4. Federated mRMR

Our proposal consists of adapting the mRMR algorithm (as described in Algorithm 1) to FL settings. Unlike a simple federated optimization algorithm, our proposal entails a modification in the way the probabilities in the mRMR formula are calculated, resulting in a lossless federated feature selection algorithm. Note that our approach is cross-silo (see Section 2.3), as we assume that all devices are available and no information is lost.

4.1. Occurrences matrix

As can be seen in Eq. (1), the only information used from the dataset is the probability of occurrence of a given value in a given feature. Furthermore, this probability can be estimated empirically using the number of occurrences of that value in the instances of the dataset (Eqs. (2) and (3)).

We can take advantage of the fact that we only need to know the number of occurrences and do not need all the data. Since we also need the number of occurrences between pairs of values over pairs of particular features ($n_{a,b}$), we have to consider how to efficiently store the number of occurrences.

Our proposal is to use an occurrences matrix, a symmetric matrix where the number of occurrences of a pair of values of a pair of features is stored in each matrix cell. Let F be the set of features of a given dataset D, with V as the set of all possible values for each feature, $V = \{v \in A \mid A \in F\}$. Moreover, let f be a bijective function that maps an element of the index set $I = \{1, \ldots, |V|\}$ to each element of the set V, $f : V \to I$.

The occurrences matrix M associated with the dataset D is a symmetric matrix of size $|V| \times |V|$, where each element M_{ij} corresponds to the number of simultaneous occurrences of the values $f^{-1}(i)$ and $f^{-1}(j)$ in the instances of the dataset D. To calculate the number of simultaneous occurrences of two values in the instances of the dataset, we can multiply their associated bitmaps (see Section 2.4). The matrix is defined as:

$$M = (m_{ij}) \in \mathbb{N}^{|V| \times |V|}, \; m_{ij} = \vec{b}_{f^{-1}(i)} \cdot \vec{b}_{f^{-1}(j)} \tag{8}$$

Since each element of this matrix is calculated through the multiplication of two vectors, we can describe it as a multiplication of two matrices. B is a block matrix in which each column consists of a bitmap, as defined in Eq. (9). We can see an example of this matrix B in Fig. 1, associated with the bitmap example shown in Section 2.4.

$$B = \left[\vec{b}_{f^{-1}(1)}, \vec{b}_{f^{-1}(2)}, \ldots, \vec{b}_{f^{-1}(n)} \right] \tag{9}$$

We can define the occurrences matrix M as shown in Eq. (10). Fig. 2 shows this matrix for the example from Section 2.4.

$$M = B^T \cdot B \tag{10}$$

Defining the occurrences matrix in terms of matrix multiplication has the advantage that matrix multiplication is a highly parallelizable operation [31–33].

Matrix B presents a sparsity value of $S = 1 - \frac{|F|}{|V|}$ (ratio of values equal to 0 over the total number of values). Therefore, depending on whether or not this value is high, we could consider using a sparse matrix to represent B and performing a smaller number of operations when multiplying two sparse matrices [34].

Since the occurrences matrix has a space complexity of $\mathcal{O}(|V|^2)$ and a minimum sparsity value of $S = \frac{\sum_{A \in F} |A|^2 - |A|}{|V|^2}$, compression techniques could possibly be applied to improve the method.

As well as the occurrences matrix, we also need to store the number of instances n of the dataset D. We can either store n previously, or we can calculate it from the occurrences matrix, since given any feature $A \in F$, $n = \sum_{a \in A} M_{f(a)f(a)}$.

4.2. Federated approach: merging occurrences matrices

So far, we have assumed that we have just a single dataset, but this assumption does not apply to FL. Let us assume, instead, that we have two federated nodes with datasets D_1 and D_2, and that each dataset has associated occurrence matrices M_1 and M_2, respectively. To be able to apply the mRMR algorithm, we need to have the information in those matrices in a single node.

The occurrences matrices of the partial datasets can be combined using a merge operation. The result is a new occurrence matrix associated with the combined partial datasets, but that does not share all the information of the original datasets.

Given the merge operation defined in Eq. (11), $M = merge(M_1, M_2)$ is the matrix associated with the dataset D, the combination of the partial datasets D_1 and D_2. Let $F = F_1 \cup F_2$, $V = V_1 \cup V_2$ and let f be the value-index mapping function associated with dataset D. Moreover, let $g_{M'} : V' \times V' \to \mathbb{N}$ be a function that returns the occurrences of two values in the dataset D', with occurrence matrix M', values set V', and value-index mapping function f', if both values are present in the dataset (using the occurrences matrix), and returns 0 if either of those values is not present in the dataset D', defined in Eq. (12). The merge operation pseudocode is shown in Algorithm 2.

$$\begin{aligned} merge(M_1,\ M_2) &= (m_{ij}) \in \mathbb{N}^{|V|\times|V|}\ , \\ m_{ij} &= g_{D_1}(a,b)\ +\ g_{D_2}(a,b) \\ where\ a &= f^{-1}(i),\ b = f^{-1}(j) \end{aligned} \tag{11}$$

$$g_{D'}(a,b) = \begin{cases} M'_{f'(a)f'(b)} & if\ a \in V' \wedge b \in V' \\ 0 & otherwise \end{cases} \tag{12}$$

Algorithm 2 Merge.

```
INPUT: M1, V1, f1, M2, V2, f2
OUTPUT: M, V, f
V = union(V1, V2)
f = combineFunctions(f1, f2)
M = newMatrix(|V|, |V|)
for each a in V do
    for each b in V do
        M[f(a), f(b)] = g(M1, f1, a, b) + g(M2, f2, a, b)
    end for
end for
return M, V, f
```

The number of instances n in dataset D (and, therefore, in the associated occurrences matrix) is the sum of the instances of datasets D_1 and D_2, $n = n_1 + n_2$.

Fig. 3 shows an example of the merger between the two occurrences matrices related to the dataset shown in Section 2.4. Note that the two matrices to be merged do not have to have the same set of feature values V; it can be seen in the example that matrix M_1 lacks the value *small* for the feature *size*, yet the merger can still take place.

In applying the merge operation to the two matrices of occurrences, since the sum of naturals and the union of sets are both associative, the merging of the two occurrences matrices is also associative. This allows us to generalize the binary operation and so to perform the merging of N occurrences matrices.

Algorithm 3 Federated merge.

```
OUTPUT: M
partialMatrixList = gatherMatricesFromAllNodes()
finalMatrix = reduce(partialMatrixList,merge)
return finalMatrix
```

Algorithm 3 shows the pseudocode to obtain an occurrences matrix M of a dataset D from partial occurrences matrices (M_1, M_2, ...) associated with the corresponding sub-datasets (D_1, D_2, ...) distributed across several federated nodes. All the partial occurrences matrices (M_1, M_2, ...) are collected from the federated nodes via the available node communication protocol (e.g., simple file download via an SFTP command, HTTP request, etc). Next applied is a reduce operation (sometimes called a fold or fold-left operation in functional paradigms) with the partial matrix list and the merge binary operation as arguments. The outcome is the final occurrences matrix M associated with the union of the partial datasets D, but with no sharing of the raw sub-datasets.

Data privacy is preserved when $min_{A \in F}\ |A|\ > 1$ and $|F| > 2$ is fulfilled at each node, making it impossible to reconstruct the original dataset from the associated occurrence matrix.

$$
\mathrm{M} = M_1 \oplus M_2 =
\begin{array}{c} \\ blue \\ red \\ big \\ a \\ b \end{array}
\begin{array}{c}
\begin{array}{ccccc} blue & red & big & a & b \end{array} \\
\begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 2 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \end{bmatrix}
\end{array}
$$

$$
\oplus
\begin{array}{c} \\ blue \\ red \\ big \\ small \\ a \\ b \end{array}
\begin{array}{c}
\begin{array}{cccccc} blue & red & big & small & a & b \end{array} \\
\begin{bmatrix} 2 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 2 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 2 \end{bmatrix}
\end{array}
$$

$$
=
\begin{array}{c} \\ blue \\ red \\ big \\ small \\ a \\ b \end{array}
\begin{array}{c}
\begin{array}{cccccc} blue & red & big & small & a & b \end{array} \\
\begin{bmatrix} 3 & 0 & 2 & 1 & 1 & 2 \\ 0 & 2 & 1 & 1 & 1 & 1 \\ 2 & 1 & 3 & 0 & 1 & 2 \\ 1 & 1 & 0 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 2 & 0 \\ 2 & 1 & 2 & 1 & 0 & 3 \end{bmatrix}
\end{array}
$$

Fig. 3. Example of occurrences matrix computed from the merger of two matrices M_1 and M_2, associated with datasets D_1 and D_2. The instances of D_1 and D_2 are the first two and last three rows, respectively, of the example dataset depicted in Table 1. The resulting matrix M is associated with the dataset D, which corresponds to the whole dataset described in Table 1. The $\oplus$ binary operator denotes the merge operation between two occurrences matrices.

Since we only need to share the computed occurrences matrices from each sub-dataset and not the raw private data, we collect in a single node all the information on the occurrences of values in all the local datasets of the distributed nodes, with no sharing of all the local data.

Finally, note that, since the calculation result of the occurrences matrix is independent of the data distribution, our fed-mRMR method, like the original mRMR method, is independent of the data distribution and works well with all possible distributions, including non-IID data.

4.3. fed-mRMR

Our algorithm, which has the same structure as Algorithm 1, is intended to be executed on a single node. We assume that the occurrence matrices of the different distributed nodes have been precomputed and merged into a single matrix that is available to the node that executes the algorithm.

The difference with our approach is in the calculation of the mutual information between two features, as defined in Eq. (1), which uses the probabilities $p(a)$ and $p(a,b)$. Mentioned earlier was that the computational complexity of mutual information, if the probabilities are to be estimated, would be $\mathcal{O}(|A| \times |B| \times n)$. In our proposed method, since we use the occurrences matrix to estimate those probabilities before the mutual information is calculated, the computational complexity of mutual information with our method is $\mathcal{O}(|A| \times |B|)$.

Given an occurrences matrix M and its associated function f to map values to indices, we define the new marginal probabilities estimates as per Eq. (13) and the new joint probability estimates as per Eq. (14), both with a computational complexity of $\mathcal{O}(1)$.

$$p(a) \approx \frac{M_{f(a)f(a)}}{n} \tag{13}$$

$$p(a,b) \approx \frac{M_{f(a)f(b)}}{n} \tag{14}$$

Once the final occurrences matrix M is in a single node, and we have the joint probability estimations $p(a,b)$ for each pair of feature values $a, b \in V$, it is trivial to apply the mRMR algorithm to the desired set of features, as described in Section 2.2.

Finally, since the marginal probability computations proposed in Eq. (13) and Eq. (14) are equivalent to those in Eq. (2) and Eq. (3), the feature selection ranking output of the mRMR remains the same, meaning that lossless federated feature selection by our method is as in the original mRMR method.

5. Experimental evaluation

In this section, we describe the experimental framework and report the experimental results of our implementation of the fed-mRMR algorithm.

Table 2
Dataset descriptions.

Datasets	# of features	# of instances
Lung	326	73
Colon	2,001	62
Lymphoma	4,027	96
Letter-recognition	16	20,000
Connect-4	42	67,557
MNIST	784	42,000
Agaricus	22	8124
Airline	19	103,904
Creditcard	30	284,807
Spambase	57	4,601

5.1. Experimental framework

In the experiments, we used a computer with the following characteristics: AMD Ryzen 5 3500U processor, 2×4GB @2667Mhz DDR4 RAM, and 256 GB NVMe SSD. Measurements were made using *hyperfine*,[2] a software benchmarking tool.

The main characteristics of the datasets used in this work are described in Table 2. To perform the experiments with different kinds of data, we used the microarray datasets[3] *Colon*, *Lung* and *Lymphoma*, which have also been used in other related works [15], and the datasets *Letter-recognition*,[4] *Connect-4*,[5] *Agaricus*,[6] *Spambase*[7] [35], *MNIST*,[8] *Airline*,[9] and *Creditcard*.[10] Large-scale low-dimensional datasets were chosen in order to test how our method works in different data conditions. Since they contain numerical values instead of categorical values, 5-bin quantile discretization was performed on the *MNIST*, *Airline*, and *Spambase* datasets, and 10-bin K-means discretization on the *Creditcard* dataset.

Our proposed algorithm, fed-mRMR, is not evaluated in terms of the ranking of features it returns. This is because our algorithm is lossless, which means that it always selects the same set of features as the original algorithm (proposed by Peng et al. [13]). Instead, fed-mRMR is evaluated with respect to computation time in a single node and in a federated environment simulation with more than one node. Computation times were calculated using means and their standard deviation from 10 repeated executions for every experiment on each dataset.

Section 5.2 shows the results for the calculation of the occurrences matrix and feature selection over the aforementioned datasets using fed-mRMR, and Section 5.4 presents performance results for fed-mRMR and the original mRMR algorithm [13] for a single computational node.

Section 5.5 shows the results for our algorithm in a simulated federated environment as the number of federated nodes increases. For the simulation of federated feature selection, we use the *MNIST* dataset as having a large number of both instances and features. Since our merge operation is independent of the data distribution, for simplicity sake, we use dataset partitions where the instances of each are cyclically distributed among n different datasets, thus simulating a scenario where we have n nodes and an occurrences matrix in each node associated with a partition in the original dataset. The merger of these n occurrences matrices and the feature selection using the resulting occurrences matrix is evaluated in terms of computation time.

5.2. Occurrences matrix and feature selection evaluation

We first evaluate the calculation times for the occurrences matrix and feature selection as the number of features we want to select increases. The results for *Colon* and *Lymphoma* are shown in Figs. 4 and 5, respectively.

The results indicate that, as expected, the computation time for the occurrences matrix remains constant irrespective of the number of features to be selected for a given dataset, while the feature selection time increases as the number of features to be selected increases, as indicated by analysis of the Algorithm 1.

Table 3 shows time results for the calculation of the occurrences matrix and selection of features for all 10 datasets. As expected, the most time-consuming datasets are *Colon* and *Lymphoma*, with the most features; this is because the complexity of our algorithm is quadratic with respect to the number of features and linear with respect to the number of samples. Note also the high computation time for the occurrences matrix compared to feature selection for *MNIST*, due to the fact that this dataset has a large number of both features and samples, which makes the computation of its occurrences matrix costly.

[2] https://github.com/sharkdp/hyperfine.
[3] These datasets can be downloaded from Peng's webpage. http://home.penglab.com/proj/mRMR/#data
[4] https://archive.ics.uci.edu/ml/datasets/Letter+Recognition.
[5] http://archive.ics.uci.edu/ml/datasets/connect-4.
[6] http://archive.ics.uci.edu/dataset/73/mushroom.
[7] https://archive.ics.uci.edu/dataset/94/spambase.
[8] https://www.kaggle.com/datasets/oddrationale/mnist-in-csv.
[9] https://www.kaggle.com/datasets/teejmahal20/airline-passenger-satisfaction.
[10] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

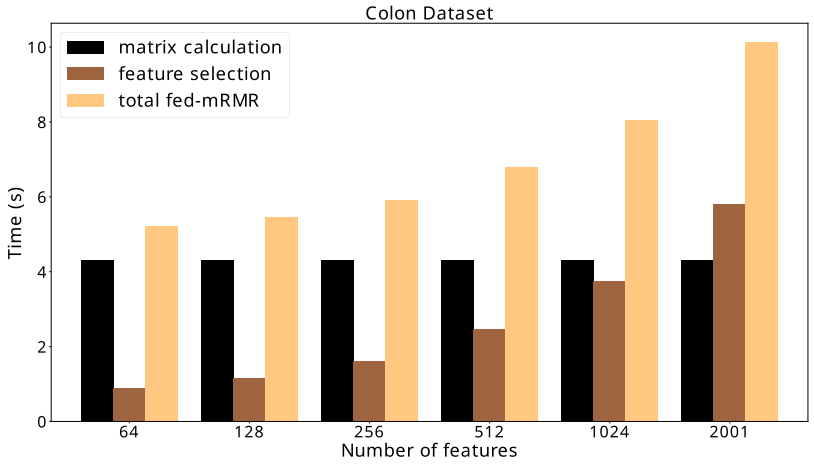

Fig. 4. Occurrences matrix calculation time and feature selection time in seconds for the *Colon* dataset as the number of features to be selected increases.

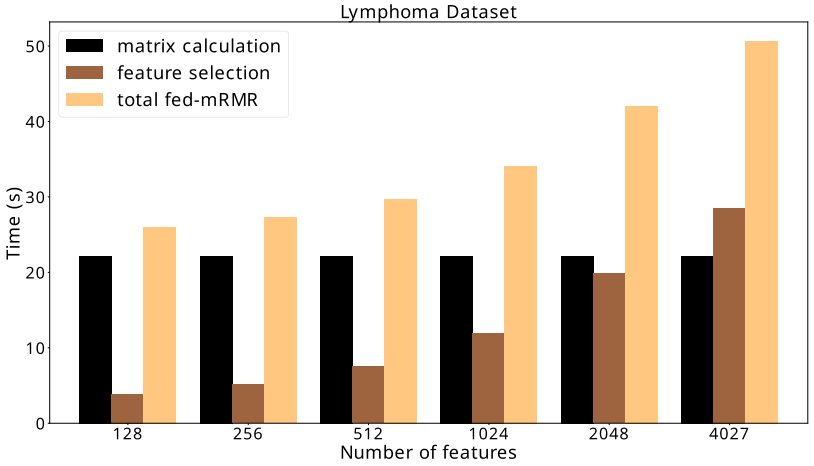

Fig. 5. Occurrences matrix calculation time and feature selection time in seconds for the *Lymphoma* dataset as the number of features to be selected increases.

Table 3
fed-mRMR mean calculation times in seconds (with standard deviations) for occurrences matrix calculation and feature selection for all features in 10 datasets.

Dataset	matrix	feature selection	fed-mRMR
Lung	0.19±0.01	0.18±0.04	0.37±0.04
Colon	4.31±0.07	5.82±0.21	10.13±0.22
Lymph.	22.16±0.61	28.50±1.06	50.66±1.22
Letter-recog.	0.47±0.05	0.07±0.01	0.54±0.05
Connect-4	1.85±0.06	0.07±0.01	1.92±0.06
MNIST	181.40±2.93	0.31±0.01	181.71±2.93
Agaricus	0.19±0.02	0.14±0.01	0.33±0.02
Airline	2.22±0.03	0.11±0.01	2.33±0.03
Creditcard	10.23±0.08	0.10±0.01	10.34±0.08
Spambase	0.26±0.01	0.15±0.01	0.41±0.01

5.3. Memory requirements

The bitmap matrix B and the occurrences matrix M potentially have a large memory usage impact. The spatial complexity of the bitmap matrix is $\mathcal{O}(N \times |V|)$, where N is the number of instances and $|V|$ is the number of values in a dataset, while the spatial complexity of the occurrences matrix is $\mathcal{O}(|V|^2)$. In this section, we analyze the memory requirements of those matrices for each of the 10 evaluated datasets. Table 4 shows, for each dataset, memory usage for those two matrices in a single node environment and also the size of the dataset file in disk (in the original text format).

Table 4
Size in the main memory for the original dataset file, the bitmap matrix B, and the occurrences matrix M.

Dataset	dataset size	matrix B	matrix M
Lung	55 KB	190.3 KB	7.7 MB
Colon	296 KB	992.4 KB	287.9 MB
Lymph.	892 KB	3 MB	1.1 GB
Letter-recog.	708 KB	2.7 MB	636 KB
Connect-4	5.5 MB	23 MB	133.1 KB
MNIST	63 MB	263.7 MB	14.3 MB
Agaricus	366 KB	1.5 MB	113.3 KB
Airline	4.0 MB	16.6 MB	78.4 KB
Creditcard	17 MB	70.6 MB	184.8 KB
Spambase	521 KB	2.1 MB	658.9 KB

Table 5
Number of features to be selected for each dataset.

Datasets	# of features to select
Lung	300
Colon	300
Lymphoma	300
Letter-recognition	16
Connect-4	42
MNIST	32
Agaricus	22
Airline	19
Creditcard	30
Spambase	57

It is important to emphasize that those two matrices do not always have to exist in memory at the same time; rather, it depends on how we perform the matrix multiplication of B and on its transpose. Once we have computed the matrix M we can safely discard the matrix B since it is no longer necessary in the algorithm.

Finally, note that we used 64-bit integers to represent each cell in both matrices. However, this value could be greatly reduced if instead we used a compact data structures approach for the bitmap matrix, since each matrix cell could be represented with just one bit using a more complex schema. This, however, is left as future work.

Matrix B serves a role solely in intermediate computations. The focal point of our algorithm, and the information to be transmitted between nodes in a federated environment, is the occurrences matrix M. From the results, we can see that the memory requirements of matrix M grow significantly for datasets with few instances but many features, e.g., the microarray datasets, and especially the *Lymphoma* dataset. In contrast, for datasets with an enormous number of instances, such as the *Creditcard* and *Airline* datasets, matrix M requires much less memory than the original dataset (almost an order of magnitude less in some cases). This is very beneficial in transfers of the occurrences matrix M to other nodes via the network, and thus demonstrates that our method is very feasible under appropriate conditions.

5.4. Comparison: fed-mRMR versus mRMR

To ensure a fair comparison of computation times, the fed-mRMR computation times for the occurrences matrix and feature selection were both taken into account. As the original mRMR is not optimized to deal with large datasets, the experiments limit the number of features to be selected to 300. Table 5 depicts the fixed number of features to be selected for each dataset; for datasets with fewer than 300 features, all the features were selected in the benchmarks. Table 6 shows computation times compared for our fed-mRMR algorithm and the original mRMR algorithm for all the datasets.

Performance is improved by fed-mMRM in 9 of the 10 datasets. Performance is the same for the *Letter-recognition* dataset, maybe because the small number of features in this dataset does not allow the original algorithm's calculations to increase too much. Since our proposal also allows occurrences matrices to be used for future feature selections, the performance improvement would be even greater if we wanted to perform several rankings of features from the same dataset.

For *MNIST* as a case study, Fig. 6 shows time results as the number of features to be selected increases. Our fed-mMRM method scales much better than the original proposal, as it allows selection of all the features in *MNIST*, unlike what happens with the original mMRM. If only a small number of features need to be selected, our proposal may not be worthwhile, since calculating the occurrences matrix would be relatively.

Table 6
Matrix calculation, feature selection, and total times (in seconds) for fed-mRMR versus mRMR.

Dataset	fed-mRMR	mRMR
Lung	0.39±0.02	20.03±0.22
Colon	6.39±0.27	40.73±0.21
Lymph.	28.54±0.56	56.53±2.43
Letter-recog.	0.54±0.12	0.49±0.09
Connect-4	1.92±0.15	37.65±2.11
MNIST	181.54±4.97	540.35±8.05
Agaricus	0.30±0.01	0.54±0.02
Airline	2.29±0.03	6.91±0.09
Creditcard	10.44±0.42	57.45±1.67
Spambase	0.39±0.01	4.27±0.06

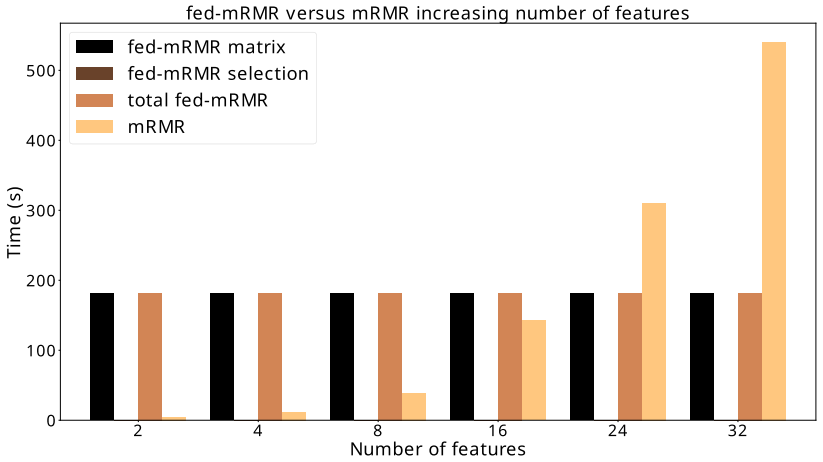

Fig. 6. fed-mRMR versus mRMR (in seconds) for the *MNIST* dataset as the number of features to be selected increases.

5.5. Federated feature selection evaluation

We now evaluate what would happen in a real federated environment. Using the *MNIST* dataset, and increasing the number of federated nodes while ranking all of the dataset features, we evaluate the behavior of the occurrences matrix calculation, merge operation, and feature selection.

For this experiment, we have assumed that the merger of all occurrences matrices is performed in a single node that has access to the occurrences matrices of all the other nodes. To evaluate the computation time of the occurrences matrix, we consider the times spent on computing the occurrences matrix in each node, but, since these can be computed in parallel, we report the maximum times.

The results are depicted in Fig. 7, which shows that the feature selection time remains constant regardless of the number of federated nodes, while the merger time for the occurrences matrices increases linearly as the number of nodes increases. Moreover, and as expected, as the number of federated nodes increases, the maximum computation times for the occurrences matrices decreases.

Note that the total time taken by the fed-mRMR algorithm decreases until the number of nodes is 16, after which it starts to increase. Since this may be because increasing the number of federated nodes does not always mean that the total computation time decreases, it may be that there is an optimal number of nodes in terms of computation time, located in the interval $(8, 64)$. In addition, due to the merger overhead for matrices, a distributed approach may be slower than a single-node approach for a certain number of nodes, as can be deduced from Fig. 7 when the number of nodes is 1024.

Note that the ranking obtained is the same, irrespective of the number of nodes used: since the merge operation is lossless, our federated algorithm is also lossless.

Regarding the communications overhead for the occurrences matrices between federated nodes in this specific simulation, the size of the occurrences matrices in each node is constant in all the experiments with different numbers of nodes – 14.3 MB in the case of the *MNIST* dataset. Matrix size is independent of the number of nodes, due to the way the data is distributed. The only data to be communicated from each node to the centralized node is that of the occurrences matrix; hence, the communications time overhead is proportional to the size of the M matrix. Furthermore, since this depends on the communication protocol used, we only depict the size of those matrices, leaving open the way the nodes communicate that data.

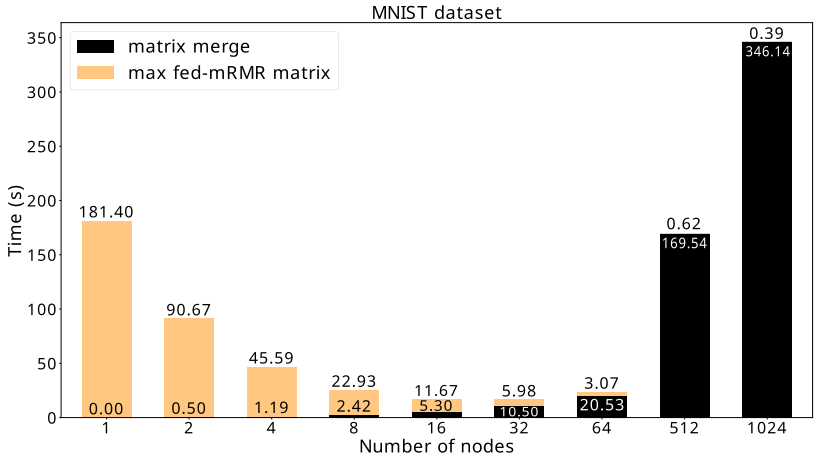

Fig. 7. Occurrences matrix merger and feature selection times (in seconds) for the *MNIST* dataset as the number of federated nodes increases. Feature selection time is constant, at 0.31 seconds, for each number of nodes value.

6. Conclusion

We have proposed a modification of the mRMR algorithm, called fed-mRMR, that allows us to perform lossless feature selection in federated environments, while achieving the same ranking as the original mRMR algorithm. Our experiments show that fed-mRMR produces an improvement in computation times with respect to the original mRMR algorithm in almost all cases, if implemented in a single computation node. This improvement is even grater when the implementation is in a distributed node environment.

To facilitate use of our algorithm and the reproducibility of our experimental results, we have made an implementation of our algorithm available in the Rust language.

A limitation of the fed-mRMR algorithm is that it presents problems with datasets with a very large number of features. This is because of the quadratic spatial complexity of the occurrences matrix, which requires a large amount of space in the main memory and a high computation time.

Left to future work is the use of a compact data structure, as reducing the space and computation time requirements of the occurrences matrix by compressing the data structure could potentially take advantage of the fact that it is possible to know the sparsity value and the approximate sparsity pattern of the occurrences matrix and the bitmap matrix in advance. Also left to future work is a possible improvement in privacy preservation using communication protocols such as Secure Aggregation [36] to merge (aggregate) the occurrences matrices between the different nodes. Additionally, we only considered a cross-silo FL approach, but it would be interesting extend this work to cross-device FL.

Finally, since there is a relationship between mutual information and the entropy of two random distributions ($I(X;Y) = H(Y) - H(Y|X)$), and since we use the occurrences matrix to compute mutual information, we believe that an approximation similar to ours could be used to perform FL using an occurrences matrix with models that use data entropy, e.g. classification and regression trees [37].

CRediT authorship contribution statement

Jorge Hermo: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Verónica Bolón-Canedo:** Writing – review & editing, Validation, Supervision, Resources, Project administration. **Susana Ladra:** Writing – review & editing, Validation, Supervision, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work was supported by CITIC, a Research Center accredited by the Galician University System, funded by the Xunta de Galicia Consellería de Cultura, Educación e Universidade, supported 80% by ERDF/FEDER funds (Operational Programme

Galicia 2014-2020) and 20% by the Xunta de Galicia Secretaría Xeral de Universidades (Grant ED431G 2019/01). It was also partially funded by the Xunta de Galicia and ERDF/FEDER (Grants ED431C 2021/53; ED431C 2022/44), the Spanish Ministerio de Ciencia e Innovación (MCIN/AEI/10.13039/501100011033) and NextGenerationEU/PRTR (Grants PDC2021-121239-C31; PID2019-105221RB-C41; PID2019-109238GB-C22; PID2022-141027NB-C2; TED2021-130599A-I00; TSI-100925-2023-1). A department collaboration grant awarded by Universidade de A Coruña also supported this research.

References

[1] S. Bazzaz Abkenar, M. Haghi Kashani, E. Mahdipour, S.M. Jameii, Big data analytics meets social media: a systematic review of techniques, open issues, and future directions, Telemat. Inform. 57 (2021) 101517, https://doi.org/10.1016/j.tele.2020.101517, https://www.sciencedirect.com/science/article/pii/S0736585320301763.
[2] C.S. Greene, J. Tan, M. Ung, J.H. Moore, C. Cheng, Big data bioinformatics, J. Cell. Physiol. 229 (2014) 1896–1900, https://doi.org/10.1002/jcp.24662.
[3] T. Niemi, J.K. Nurminen, J.-M. Liukkonen, A.-P. Hameri, Towards green big data at cern, Future Gener. Comput. Syst. 81 (2018) 103–113, https://doi.org/10.1016/j.future.2017.11.001, https://www.sciencedirect.com/science/article/pii/S0167739X17313651.
[4] R. Bellman, Dynamic Programming, Dover Publications, 1957.
[5] V. Bolón-Canedo, A. Alonso-Betanzos, L. Morán-Fernández, B. Cancela, Feature Selection: From the Past to the Future, 2022, pp. 11–34, https://doi.org/10.1007/978-3-030-93052-3_2.
[6] P. Ziemba, M. Piwowarski, Feature selection methods in data mining techniques, Res. Pap. Wrocław Univ. Econ. 206 (2011) 213–223.
[7] J. Maillo, I. Triguero, F. Herrera, Redundancy and complexity metrics for big data classification: towards smart data, IEEE Access 8 (2020) 87918–87928, https://doi.org/10.1109/ACCESS.2020.2991800.
[8] E. Hato, Impact of feature selection for data classification using naive bayes classifier, J. Phys. Conf. Ser. 1879 (2021) 022088, https://doi.org/10.1088/1742-6596/1879/2/022088.
[9] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R.G.L. D'Oliveira, H. Eichner, S.E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P.B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S.U. Stich, Z. Sun, A.T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F.X. Yu, H. Yu, S. Zhao, Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (2021) 1–210, https://doi.org/10.1561/2200000083.
[10] H. Zhu, J. Xu, S. Liu, Y. Jin, Federated learning on non-iid data: a survey, Neurocomputing 465 (2021) 371–390, https://doi.org/10.1016/j.neucom.2021.07.098, https://www.sciencedirect.com/science/article/pii/S0925231221013254.
[11] M. Sheikhalishahi, F. Martinelli, Privacy-utility feature selection as a privacy mechanism in collaborative data classification, in: 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), 2017, pp. 244–249.
[12] M. Banerjee, S. Chakravarty, Privacy preserving feature selection for distributed data using virtual dimension, in: Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11, Association for Computing Machinery, New York, NY, USA, 2011, pp. 2281–2284.
[13] H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy, IEEE Trans. Pattern Anal. Mach. Intell. 27 (2005) 1226–1238, https://doi.org/10.1109/TPAMI.2005.159.
[14] C. Ding, H. Peng, Minimum redundancy feature selection from microarray gene expression data, in: Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003, 2003, pp. 523–528.
[15] S. Ramírez-Gallego, I. Lastra, D. Martinez, V. Bolón-Canedo, J. Benítez, F. Herrera, A. Alonso-Betanzos, Fast-mrmr: fast minimum redundancy maximum relevance algorithm for high-dimensional big data: fast-mrmr algorithm for big data, Int. J. Intell. Syst. 32 (2016), https://doi.org/10.1002/int.21833.
[16] J. González-Domínguez, V. Bolón-Canedo, B. Freire, J. Touriño, Parallel feature selection for distributed-memory clusters, Inf. Sci. 496 (2019) 399–409, https://doi.org/10.1016/j.ins.2019.01.050, https://www.sciencedirect.com/science/article/pii/S0020025519300635.
[17] K. Cheng, T. Fan, Y. Jin, Y. Liu, T. Chen, D. Papadopoulos, Q. Yang, Secureboost: a lossless federated learning framework, IEEE Intell. Syst. 36 (2021) 87–98, https://doi.org/10.1109/MIS.2021.3082561.
[18] N. Sánchez-Maroño, A. Alonso-Betanzos, M. Tombilla-Sanromán, Filter methods for feature selection – a comparative study, in: H. Yin, P. Tino, E. Corchado, W. Byrne, X. Yao (Eds.), Intelligent Data Engineering and Automated Learning - IDEAL 2007, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 178–187.
[19] N. El Aboudi, L. Benhlima, Review on wrapper feature selection approaches, in: 2016 International Conference on Engineering & MIS (ICEMIS), 2016, pp. 1–5.
[20] V. Fonti, E. Belitser, Feature selection using lasso, VU Amsterdam Res. Pap. Bus. Anal. 30 (2017) 1–25.
[21] A. Idris, A. Khan, Y.S. Lee, Intelligent churn prediction in telecom: employing mrmr feature selection and rotboost based ensemble classification, Appl. Intell. 39 (2013) 659–672, https://doi.org/10.1007/s10489-013-0440-x.
[22] S. Bashir, Z.S. Khan, F. Hassan Khan, A. Anjum, K. Bashir, Improving heart disease prediction using feature selection approaches, in: 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 2019, pp. 619–623.
[23] M.H. Bhuyan, D.K. Bhattacharyya, J.K. Kalita, Network anomaly detection: methods, systems and tools, IEEE Commun. Surv. Tutor. 16 (2014) 303–336, https://doi.org/10.1109/SURV.2013.052213.00046.
[24] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (2020) 50–60, https://doi.org/10.1109/MSP.2020.2975749.
[25] H.B. McMahan, E. Moore, D. Ramage, B.A. y Arcas, Federated learning of deep networks using model averaging, preprint, arXiv:1602.05629, 2 (2016) 2.
[26] S.P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, A.T. Suresh, Scaffold: stochastic controlled averaging for federated learning, in: International Conference on Machine Learning, PMLR, 2020, pp. 5132–5143.
[27] A. Li, H. Peng, L. Zhang, J. Huang, Q. Guo, H. Yu, Y. Liu, Fedsdg-fs: efficient and secure feature selection for vertical federated learning, preprint, arXiv:2302.10417, 2023.
[28] T. Castiglia, Y. Zhou, S. Wang, S. Kadhe, N. Baracaldo, S. Patterson, Less-vfl: communication-efficient feature selection for vertical federated learning, arXiv preprint, arXiv:2305.02219, 2023.
[29] Y. Hu, Y. Zhang, X. Gao, D. Gong, X. Song, Y. Guo, J. Wang, A federated feature selection algorithm based on particle swarm optimization under privacy protection, Knowl.-Based Syst. 260 (2023) 110122.
[30] X. Zhang, A. Mavromatics, A. Vafeas, R. Nejabati, D. Simeonidou, Federated feature selection for horizontal federated learning in iot networks, IEEE Int. Things J. (2023).
[31] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, in: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC '87, Association for Computing Machinery, New York, NY, USA, 1987, pp. 1–6.
[32] K. Goto, R.A.v.d. Geijn, Anatomy of high-performance matrix multiplication, ACM Trans. Math. Softw. 34 (2008), https://doi.org/10.1145/1356052.1356053.
[33] J. Choi, D.W. Walker, J.J. Dongarra, Pumma: parallel universal matrix multiplication algorithms on distributed memory concurrent computers, Concurr. Comput., Pract. Exp. 6 (1994) 543–570, https://doi.org/10.1002/cpe.4330060702, https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4330060702.
[34] R. Yuster, U. Zwick, Fast sparse matrix multiplication, ACM Trans. Algorithms 1 (2005) 2–13, https://doi.org/10.1145/1077464.1077466.
[35] D. Dua, C. Graff, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2017.

[36] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H.B. McMahan, S. Patel, D. Ramage, A. Segal, K. Seth, Practical secure aggregation for privacy-preserving machine learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS '17, Association for Computing Machinery, New York, NY, USA, 2017, pp. 1175–1191.
[37] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Routledge, 2017.