Towards associative classification in distributed environments

Rosana Veroneze[1] (✉), Xavier Lessage[2], and Jean Vanderdonckt[1]

[1] UCLouvain, Louvain-la-Neuve, Belgium
{rosana.veroneze,jean.vanderdonckt}@uclouvain.be
[2] CETIC, Charleroi, Belgium xavier.lessage@cetic.be

Abstract. We study associative classification (AC) in a fully trusted federated environment with horizontally partitioned data. We propose two distributed AC frameworks that enable multiple sites to collaboratively train a global classifier while keeping raw data local. Both methods preserve the conventional AC pipeline at the local level (rule generation, ranking, and coverage-based pruning) while differing in how global rule metrics are computed. The first approach approximates global metrics using a communication-efficient consolidation strategy inspired by the DAC algorithm. The second computes exact global rule metrics via additional communication rounds, yielding classifiers that accurately approximate centralized models. To improve efficiency and reduce redundancy, the frameworks incorporate closed frequent itemset mining during rule generation. We evaluate the proposed methods on several datasets and compare them with DAC, logistic regression, and MLP. Experimental results show that our refinements to the DAC framework substantially improve its predictive performance. Moreover, the exact-metric variant consistently yields accurate and stable classifiers across varying degrees of data distribution while maintaining relatively low communication costs. Overall, our findings suggest that high-quality associative classifiers can be constructed in distributed settings with minimal communication, offering a transparent and competitive alternative to black-box models.

Keywords: Associative classification · Distributed learning · Horizontal data partitioning · Communication-efficient mining.

1 Introduction

Classification is a fundamental problem in machine learning. Over the years, a wide range of classification algorithms have been proposed, from interpretable to highly opaque approaches. The rising demand for Explainable Artificial Intelligence (XAI) has renewed interest in interpretability.

Among interpretable models, associative classifiers offer a compelling approach that combines association rule mining (ARM) with classification [4,9]. Unlike heuristic rule-learning approaches, ARM techniques provide formal guarantees of the completeness and optimality of the discovered set of rules, enabling a systematic exploration of all relevant attribute–value relationships during training. As a result, associative classifiers have demonstrated competitive

performance compared to other established methods in various domains [4,15]. Another key advantage of associative classification (AC) lies in the modularity of the resulting model: it can be inspected, debugged, incrementally updated, or manually refined without having to reconstruct the entire model (it contrasts with many other interpretable approaches that require structural redesign when modifications are needed) [15]. Ultimately, AC aligns with a growing consensus that high-stakes domains should prioritize inherently interpretable models over post-hoc explanations for black-box systems [13].

Although AC has been extensively studied in centralized settings [4], studies specifically focused on AC in physically distributed environments, such as data residing in different hospitals or banks, remain unexplored. In principle, one could leverage existing techniques for distributed or federated pattern mining [18] to extract rules across data silos and then proceed with a conventional AC pipeline at the central server. However, this approach faces several fundamental challenges. First, pattern mining can generate an enormous number of candidate rules, leading to a prohibitive communication cost among sites. Moreover, only a small subset of these rules typically contributes to the final classifier, making such exhaustive distributed rule mining inefficient. Second, even a complete and correct global rule set may be insufficient to build an associative classifier. Many AC algorithms require knowledge of which individual data instances are covered by each rule [4]. Such instance-level coverage information would require several communication rounds, being prohibitively expensive. Finally, from a privacy perspective, transmitting only compact rule sets to the server can be significantly less intrusive than sharing the entire set of mined rules. These considerations highlight that distributed AC requires dedicated methods that go beyond distributed rule mining alone.

One existing approach that can be leveraged for such environments is the DAC algorithm [15]. Originally designed for large-scale dataset scalability, DAC partitions data to train local AC models, which are then merged into a single global model. This architecture enables it to operate across physically distributed sites. Our preliminary experiments revealed that although DAC underperforms centralized baselines – such as Multi-Layer Perceptron (MLP), Logistic Regression (LR), and traditional AC models – its model consolidation strategy demonstrates notable stability given its simplicity. Specifically, under moderate, independent and identically distributed (IID) horizontal data partitioning, DAC's performance remains fairly consistent.

This resilience leads to our **core hypothesis**: high-quality global associative classifiers can be constructed if each local model contributes a carefully curated subset of its most informative rules. To test this, we replace DAC's heuristic for building local classifiers with a conventional, well-established AC pipeline [4]: rule enumeration, rule ranking, and coverage-based pruning. We then propose two DAC-inspired algorithmic frameworks: the first refines how local classifiers are built and how the classifier's default class is assigned while approximating global metrics; the second adds communication rounds to compute exact rule metrics. Thus, we summarize our contributions as follows:

- To our knowledge, we are the first to address AC specifically for physically distributed environments rather than just for computational scalability.
- We propose communication-efficient frameworks for distributed AC.
- Our frameworks incorporate closed frequent itemset mining (CFIM) within the AC pipeline to reduce rule redundancy and improve local mining efficiency.
- We conduct an extensive experimental evaluation across multiple benchmark datasets, comparing the proposed methods with DAC, MLP, and LR on predictive performance, robustness to varying data distributions, communication cost, and model complexity.

The remainder of this paper is organized as follows. Section 2 reviews the necessary background topics. Section 3 discusses related works. Section 4 presents the proposed distributed AC frameworks. Section 5 describes the experimental setup, and Section 6 reports and discusses the experimental results. Finally, Section 7 concludes the paper and outlines possible directions for future research.

2 Background

2.1 Pattern mining

Let $I = \{x_1, x_2, \ldots, x_m\}$ be a finite set of elements referred to as *items*, and let $\Gamma = \{t_1, t_2, \ldots, t_n\}$ be a finite set of *transaction identifiers* (or *tids*). A subset $X \subseteq I$ is called an *itemset*, and a subset $T \subseteq \Gamma$ is called a *tidset*. Let $\mathbf{D} \subseteq \Gamma \times I$ be a binary relation representing a dataset, where a pair $(t, x) \in \mathbf{D}$ indicates that transaction t contains item x. We define a function $\mathbf{t} : 2^I \to 2^\Gamma$ that maps an itemset to the set of transactions in which all items occur: $\mathbf{t}(X, \mathbf{D}) = \{t \in \Gamma \mid \forall x \in X,\ (t, x) \in \mathbf{D}\}$. The support of an itemset $X \subseteq I$ in the dataset $\mathbf{D}$ is defined as: $\sup(X, \mathbf{D}) = |\mathbf{t}(X, \mathbf{D})|$. An itemset $X \subseteq I$ is named *frequent* if $\sup(X, \mathbf{D}) \geq minsup$, where $minsup$ is a user-defined minimum support threshold. The set of all frequent itemsets is given by: $\mathcal{F} = \{X \subseteq I \mid \sup(X, \mathbf{D}) \geq minsup\}$.

The search space of frequent itemsets is typically extremely large, exhibiting exponential growth with respect to the number of items [19]. Thus, instead of enumerating all frequent itemsets, we can focus on computing condensed representations of the result set. A commonly lossless condensed representation is the set of *closed itemsets*. The set of all closed frequent itemsets is given by: $\mathcal{C} = \{X \in \mathcal{F} \mid \nexists Y \in \mathcal{F}$ such that $X \subset Y$ and $sup(X, \mathbf{D}) = sup(Y, \mathbf{D})\}$.

An *association rule* (AR) is an expression of the form $X \to Y$, where both X and Y are itemsets (i.e, $X, Y \subseteq I$), and $X \cap Y = \emptyset$. The itemset X is referred to as the antecedent (condition) of the rule, and Y as the consequent. The support of an AR $X \to Y$ is given by $\sup(X \to Y, \mathbf{D}) = \sup(X \cup Y, \mathbf{D})$ and its confidence is given by $\text{conf}(X \to Y, \mathbf{D}) = \sup(X \cup Y, \mathbf{D}) / \sup(X, \mathbf{D})$. An AR $X \to Y$ is considered *strong* if its confidence meets or exceeds a user-defined threshold $minconf \in [0, 1]$, i.e., $\text{conf}(X \to Y, \mathbf{D}) \geq minconf$.

It is often useful to mine ARs with a fixed target attribute y. Rules of the form $X \to \{y = c\}$ are referred to as *class association rules* (CARs). For simplicity, we

denote such rules as $X \rightarrow c$. An important observation is that itemsets covering the same transactions form an *equivalence class*. Consequently, all CARs within a class share identical statistical properties.

2.2 Associative classification

Associative classification (AC) is a data mining approach that integrates ARM with classification tasks [4,9]. AC algorithms typically follow a step-wise procedure. First, candidate rules that satisfy user-defined constraints, such as minimum support and confidence thresholds, are generated using traditional pattern mining algorithms. Next, the candidate rules are ranked, and only a subset is selected to compose the classifier.

Rule ranking is essential in the construction of AC models. Its purpose is to establish the precedence among mined rules based on multiple quality indicators, thereby guiding the subsequent pruning stage. The ranking strategy can be tailored to specific objectives, but it typically includes confidence, support, and the length of the rules [4].

Rule pruning is usually guided by heuristic strategies. The most widely used heuristic is based on dataset coverage [4]. In this greedy approach, rules are iteratively selected to cover the training dataset until every instance is matched by at least δ rules [8,9]. The seminal CBA algorithm [9] adopts $\delta = 1$. Moreover, to obtain an even more compact classifier (and potentially improve classifier generalization), CBA introduces an additional pruning step: it identifies the first selected rule with the fewest errors as a cutoff rule; all rules after this cutoff are discarded from the final classifier.

For **predicting** the class label of an unseen instance, AC models typically use two strategies: an ordered rule set (also called **rule list** or **decision list**), where rules are evaluated sequentially and the first matching rule determines the prediction; or an unordered rule set (also simply called **rule set**), where all matching rules vote to produce the final prediction.

2.3 The DAC algorithm [15]

DAC is designed for scalability on large datasets using distributed platforms such as Apache Spark. Its baseline procedure is as follows: (*i*) the training dataset is split into K partitions, each generated by sampling from the original dataset; (*ii*) within each partition, a rule extraction phase is executed, generating a local model consisting in a set of CARs; (*iii*) the generated K local models are consolidated into one single final model.

DAC uses a scalable heuristic for rule extraction, named CAP-growth. The authors compare CAP-growth to the dataset-coverage-based pruning heuristic used in conventional AC, arguing that both aim to produce concise, non-redundant models. The local models generated by CAP-growth are consolidated into a single global model. The merging procedure is straightforward: the final model is essentially the union of the local models. If a rule is unique (i.e., it appears in only one client), its locally computed metrics are directly retained.

When identical rules (i.e., rules with the same antecedent and consequent) appear across multiple local models, DAC aggregates them by approximating their global support, confidence, and χ^2 statistic using the maximum of the corresponding local values, thus providing an upper-bound estimation.

DAC uses a *rule set* for prediction. The matching rules are grouped by predicted class label, and a score is computed for each class. Each score is determined by the maximum confidence value in its group. If no matching rules exist for a given class label, its score is estimated under a naive independence assumption. The final class prediction corresponds to the label with the highest normalized score. If no rules match an instance, the classifier **defaults** to the **majority class** (based on training data priors).

3 Related work

There exist works addressing pattern mining in physically distributed environments [5,18]. These typically focus on efficient, privacy-aware pattern extraction in such settings. However, pattern mining is only one stage in building associative classifiers. The additional steps required for AC introduce further challenges that distributed pattern mining alone cannot address.

We are not aware of any other studies specifically devoted to AC that involve multiple independent organizations collaboratively training a unified classifier without centralizing their data. However, several AC approaches exist that are primarily designed for large-scale data processing within distributed computing frameworks such as Apache Hadoop, Spark, and Flink [3,12,14,15,16]. These methods typically partition the data within a single administrative domain for computational efficiency. Among them, DAC [15] and its predecessor, BAC [16], can operate in physically distributed environments. BAC builds local models using a variant of the L3 algorithm [2] and performs prediction through an ensemble strategy, where the K local classifiers vote on the final label. In contrast, DAC consolidates the local models into a single global classifier.

4 Distributed associative classification algorithms

Our study assumes a fully trusted federated setting with horizontally partitioned data, and focuses on collaborative training of AC models across multiple organizations that keep their raw data local and share only model information.

We propose two distributed AC algorithms inspired by DAC. However, unlike DAC, which builds local classifiers heuristically, both of our methods preserve the classical AC pipeline at each site (rule enumeration, rule ranking, and coverage-based pruning). This choice maintains consistency with standard centralized AC methods and allows us to assess the quality of the distributed approach without conflating it with heuristic local model construction.

4.1 Local classifiers

The **rule generation** stage relies on CFIM. Although most AC approaches are built on top of standard FIM algorithms [4], this choice is often inefficient. For instance, in coverage-based pruning with $\delta = 1$, only a single rule from each *equivalence class* is ultimately retained (nevertheless, FIM generates all of them). CFIM directly addresses this issue by mining only one representative per equivalence class. This avoids redundancy and eliminates the arbitrary post-hoc selection among equivalent candidates, resulting in a more principled and efficient rule-mining process. CFIM is also computationally attractive because closed itemsets can be mined with polynomial delay [6]. Moreover, while traditional AC often favors shorter rules (derived from *minimal generators* of an equivalence class) [4], it has been observed that classifiers based on closed itemsets can achieve comparable predictive performance [7].

Although any CFIM algorithm can be employed, it is convenient to select one that can be easily adapted to operate with a distinct minimum support threshold per class label. This flexibility is especially important in imbalanced classification tasks. In this work, we adopt RIn-Close_CVCP [17], a CFIM algorithm rooted in the In-Close family of polynomial-delay algorithms [1,6], capable of mining closed itemsets directly from a discrete data matrix, eliminating the need to transform the dataset into a transaction-based representation.

We tailor RIn-Close_CVCP for CAR extraction. First, we modify the algorithm to handle multiple minimum support thresholds, enabling class-dependent constraints. Second, whenever a rule achieves 100% confidence, the algorithm halts its further specialization. This prevents the generation of descendant rules that would maintain the same confidence while exhibiting lower support. Third, whereas the original In-Close implementation enqueues newly generated patterns for subsequent closure, our implementation optimizes by substituting the queue with a stack. This reduces the maximum number of patterns simultaneously awaiting closure, thereby lowering peak memory consumption and improving scalability. Fourth, we also enforce minimum confidence and minimum χ^2 constraints to filter candidate rules, ensuring that only those with sufficient predictive strength remain. Lastly, to avoid contradictory rules (identical conditions pointing to distinct class labels), for any fixed itemset in the antecedent, the rule's consequent is the class label that yields the greatest confidence.

The **rule ranking** is user-defined. As a default option, we adopt the most common ranking scheme [4]: rules are ordered primarily by decreasing confidence, then by decreasing support, and finally by increasing rule length.

We adopt dataset **coverage-based pruning** to build the local classifiers, as it remains the standard and most widely used pruning strategy in AC [4]. This choice also supports our objective of obtaining compact local models (unlike more conservative pruning approaches, such as lazy pruning [2], which often retain a large fraction of the mined rules).

Algorithm 1 Distributed Associative Classification with Approximated Metrics

1: // **Step 1: Local classifier construction**
2: **for** each client $k = 1, \ldots, K$ **do**
3: Build a local associative classifier
4: Send to server: rules; local metrics; and counts of uncovered instances per class
5: **end for**
6: // **Step 2: Global classifier construction**
7: Server merges identical rules; approximates global metrics; sets classifier's default class via aggregated uncovered counts
8: Optional: Server ranks rules

4.2 Global classifier

Approximated global metrics Algorithm 1 describes the proposed distributed AC framework with approximation of the global rule metrics. As DAC, the method follows a two-phase procedure: (i) independent local model construction at each client, and (ii) centralized aggregation at the server.

After constructing its local associative classifier, each client transmits the following information to the server: the mined rules; the local quality metrics computed for each rule (e.g., support, confidence, χ^2); and the number of training instances per class label that remain uncovered after local pruning. Importantly, the specific metrics that must be transmitted depend on the design of the final global classifier. Only the measures required for global ranking or prediction need to be communicated, allowing flexibility in the system design.

At the server side, the local classifiers are aggregated as in DAC. In addition, the counts of uncovered training instances per class label are used to determine the classifier's default class. The server aggregates these counts by summing, for each class label, the values received from all clients. The default class is then defined as the class with the highest total number of uncovered instances. This ensures consistency with the locally adopted coverage-based pruning strategy.

The **prediction mechanism** of the final classifier is user-defined. If the classifier operates as a *rule list*, the server must rank the aggregated rules. In this case, clients must have transmitted all metrics required for the chosen ranking criteria. If the classifier operates as a *rule set*, the server must instead receive the metric(s) required by the selected voting mechanism.

Exact global metrics Algorithm 2 extends the distributed framework by computing exact global rule metrics, at the cost of additional communication rounds. Unlike the approximate variant, this approach reconstructs the true global support distribution of each rule before final pruning and classifier construction.

In the initial round, each client transmits only the antecedents of its locally selected rules. The consequent and locally computed metrics are not required at this stage, since global metrics will be computed exactly in subsequent steps.

Upon receiving the antecedents from all clients, the server merges them into a unified candidate set. Identical antecedents are consolidated and retained only

Algorithm 2 Distributed Associative Classification with Exact Metrics

// **Step 1: Local classifier construction**
for each client $k = 1, \ldots, K$ **do**
    Build a local associative classifier
    Send (only) the rules' antecedents to the server
end for
// **Step 2: Rule consolidation**
Server merges rules with identical antecedents; assigns an identifier (ID) to each rule's antecedent; and broadcasts the rules' antecedent and their IDs to all clients
// **Step 3: Exact support counting**
for each client $k = 1, \ldots, K$ **do**
    Compute class-wise support counts for each rule's antecedent
    Send the counts to the server (using IDs only)
end for
// **Step 4: Rule finalization**
for each rule's antecedent **do**
    Server aggregates global class counts; assigns rule's consequent; and compute rule's metrics
end for
Server prunes rules below user-defined thresholds
Server broadcasts all rules' antecedents to clients
// **Step 5: Count of uncovered training instances**
for each client $k = 1, \ldots, K$ **do**
    Count uncovered training instances per class label; send them to the server
end for
// **Step 6: Set classifier's default class label**
Server sets classifier's default class via aggregated uncovered counts
Optional: Server ranks rules

once, ensuring that each unique condition is evaluated globally. The server then assigns a unique identifier (ID) to each antecedent and broadcasts the list of antecedents along with their corresponding IDs to all clients.

Each client computes class-wise support counts over its local dataset for each rule's antecedent. These counts are transmitted back to the server using only the rule IDs, thereby minimizing communication overhead.

The server sums class-wise counts across clients to obtain the exact global distribution for each rule's antecedent. It then assigns the majority class as the consequent and computes the global rule metrics. For example, in a binary classification task, suppose an antecedent covers 100 positive and 50 negative instances globally. The rule consequent is assigned to the positive class. The support of the antecedent is therefore 150, the support of the rule is 100, and the confidence is 100/150. Rules that do not satisfy user-defined constraints are pruned. After the final rules are determined, the server broadcasts the rules' antecedent once more to the clients in order to obtain the number of uncovered training instances per class label. Clients compute these counts locally and send them to the server. Next, the classifier's default class is determined as in Algorithm 1.

The **prediction mechanism** of the final classifier is also user-defined. Because exact global metrics are available, any ranking or voting strategy can rely on statistically accurate measures consistent with a centralized learning setting.

4.3 Theoretical analysis and limitations

Algorithm 2 guarantees that the final classifier evaluates its rules using exact, statistically accurate global metrics. However, we emphasize that this exactness applies to the metrics of the aggregated candidate set derived from locally retained rules. Our proposal does not guarantee producing a final classifier identical to one learned in a centralized setting. The conditions under which strict equivalence holds (and the reasons it often diverges) are rooted in the properties of distributed CFIM [10].

First, by the fundamental property of distributed FIM, any globally frequent itemset must be frequent in at least one local partition (assuming the same relative support threshold across clients and globally) [10]. While this ensures that no globally frequent pattern is entirely missing from the global search space, our approach relies on CFIM for computational efficiency and reducing redundancy. In a distributed setting, a global closed frequent itemset may fail to be generated if, for example, it is frequent but not closed in one partition, and closed but infrequent in another. Moreover, while every local closed itemset is a global closed itemset, the reverse is not strictly true [10]. To exemplify, let us consider a simple unsupervised scenario with $K = 2$ partitions. Suppose the dataset of site $k = 1$, $\mathbf{D}_1$, contains only one transaction, which is equal to $\{A, B, C, E\}$. So, $\{A, B, C, E\}$ is a closed itemset in $\mathbf{D}_1$. Suppose $\mathbf{D}_2$ contains only one transaction, which is equal to $\{A, B, C, D\}$. So, $\{A, B, C, D\}$ is a closed itemset in $\mathbf{D}_2$. Observe that $\{A, B, C\}$ is a closed itemset in the global dataset $\mathbf{D} = \mathbf{D}_1 \cup \mathbf{D}_2$, but it is not closed in either of the two partitions.

Thus, the global classifier **may** have unnecessarily more specialized rules than the centralized counterpart. Furthermore, the divergence between the distributed and centralized learned classifiers is sensitive to the minimum support threshold. Rules with global support close to the user-defined threshold are most prone to being missed in the distributed scenario, as slight variations in local data distributions can make them locally infrequent across all sites. When these globally valid but borderline rules are missed locally, the local coverage-based pruning compensates by selecting alternative rules to cover the dataset. Ultimately, while the exact-metric variant does not perfectly mirror a centralized model, it provides an accurate approximation that avoids the prohibitive communication costs and memory overhead of trying to replicate the centralized AC pipeline in a distributed setting.

5 Experimental setup

5.1 Datasets

In this study, we restrict our experiments to datasets composed exclusively of categorical attributes. Although associative classifiers can be constructed from

Table 1. Datasets.

ID	Name	#Inst.	#Feat.	#Items	Class Distribution
26	Connect-4	67,557	42	126	9%, 25%, 66%
76	Nursery	12,960	8	27	0%, 3%, 31%, 33%, 33%
73	Mushroom	8,124	22	116	48%, 52%
22	Chess	3,196	35	71	48%, 52%
69	Molecular Biology	3,190	60	287	24%, 24%, 52%
19	Car Evaluation	1,728	6	21	4%, 4%, 22%, 70%
101	Tic-Tac-Toe	958	9	27	35%, 65%
936	National Poll	714	14	49	18%, 30%, 52%
105	Congressional Voting	435	16	32	39%, 61%
70	MONK's Problems	432	6	17	50%, 50%
14	Breast Cancer	286	9	41	30%, 70%

general tabular datasets by applying discretization techniques to transform continuous attributes into categorical ones, the design and rigorous evaluation of such discretization procedures fall outside the scope of this work. In particular, discretization in our setting would need to explicitly account for the distributed nature of the data, which introduces additional methodological and communication challenges. To ensure that our conclusions accurately reflect the behavior of the proposed distributed AC algorithms, rather than artifacts introduced by preprocessing choices, we deliberately avoid incorporating discretization.

Table 1 summarizes the datasets used in our experiments. All datasets are publicly available[3]. The collection includes both binary and multi-class classification tasks, encompassing datasets with balanced and imbalanced class distributions. This diversity allows us to assess the robustness and generalization ability of the evaluated algorithms across different problem characteristics. We employed one-hot encoding for all datasets and baseline algorithms considered in our experiments.

5.2 Models and evaluation

Our proposed methods are evaluated against three baseline models: Multilayer Perceptron (MLP), Logistic Regression (LR), and DAC.

Each experiment was repeated **20 times** with distinct random seeds. Each run proceeds as follows: the dataset is first randomly shuffled and then split into 80% training and 20% testing partitions. Each model is trained on the training partition. For distributed settings (i.e., when the number of sites is greater than one), the training data is split IID across sites, such that all sites receive disjoint but equally sized subsets of the training data. All clients participate in every communication round. The evaluation is performed using the single global hold-out test set, identical across all settings. This ensures a fair comparison between centralized and distributed settings, since both configurations are trained and

[3] `https://archive.ics.uci.edu/`

tested on the same data. All reported results in Section 6 (*Experimental results and Discussion*) correspond to the mean values computed across the 20 runs.

The primary evaluation metrics are accuracy, balanced accuracy, macro-averaged F1-score, and Matthews correlation coefficient (MCC). For associative classifiers, we additionally report the number of rules and the average rule length. Lastly, we also report the communication cost for distributed settings.

Communication cost computation For LR and MLP, we assume a standard protocol where clients transmit full model parameters (weights and biases) each round, with the server broadcasting the aggregated model back. For all models, each parameter, *item*, integer, or metric is represented by a 32-bit value.

5.3 Implementation details

RIn-Close_CVCP implementation is available at GitHub. The experiments with LR and MLP models were implemented using the *scikit-learn* library, and the distributed settings were managed with the *Flower* framework. The AC algorithms were implemented from scratch; our code is available at GitHub.

For LR, the default parameters from *scikit-learn* were used, except that the maximum number of iterations was increased to 1000 to ensure convergence. For MLP, hyperparameters were selected via grid search with 5-fold cross-validation, considering two activation functions (ReLU and tanh) and several hidden-layer configurations with one or two layers and sizes proportional to the number of input features. The selected configuration was then retrained on the full training set. In the distributed setting, both LR and MLP used one local epoch per communication round. The number of communication rounds was chosen so that the distributed training performed approximately the same total number of passes over the data as the corresponding centralized training.

DAC does not support the use of class-dependent minimum support thresholds. Thus, we used a low threshold of 0.02% to avoid penalizing minority classes with fewer instances. This choice is feasible thanks to the efficient heuristic employed by DAC during rule generation. The minimum confidence threshold was set to 51%, and the significance level for the χ^2 test was fixed at $\alpha = 0.05$.

Regarding our proposals, minimum support thresholds were derived using the theoretical bounds proposed in [11], which compute a class-dependent minimum support ensuring that a rule can potentially satisfy the χ^2 significance test. To avoid impractically small values, we enforced strict minimum thresholds. Thus, the threshold for each class label was set to the maximum of: the χ^2-derived bound, a relative support of 1%, and an absolute support of 2 instances. The minimum confidence was also set to 51%. For three datasets (*Molecular Biology*, *Chess*, and *Connect-4*), support and confidence thresholds were adjusted to mitigate *pattern explosion* [19] (always respecting the χ^2-derived lower bound and minimum confidence $\geq 51\%$). For the coverage-based rule selection, we evaluated $\delta \in \{1, 2, 3, 4\}$. When $\delta = 1$, we also considered the additional pruning strategy introduced in CBA (denoted $\delta = 1^*$), as described in Section 2.2. Larger values of δ were not considered because our objective is to produce compact associative

classifiers. To enable a fair and direct comparison with DAC, we used our default rule ranking scheme and a *rule-list* prediction mechanism. This setting is intentionally close to DAC, whose prediction is based on maximum confidence.

Hardware and computing environment Experiments were conducted on the CÉCI academic HPC infrastructure[4], a consortium of HPC clusters operated by the universities of the Wallonia-Brussels Federation and managed through Slurm. Jobs were executed on available CÉCI clusters using a consistent software environment, with each job requesting a single CPU core and 32 GB of RAM.

6 Experimental results and Discussion

In this section, we evaluate our proposed algorithms and analyze their behavior under different experimental conditions. The experiments were designed to evaluate predictive performance, communication efficiency, model complexity, and robustness to increasing data distribution. Specifically, our evaluation seeks to address the following Research Questions (RQs): **RQ1**: How do the distributed AC algorithms compare with DAC and other baseline models in terms of predictive performance? **RQ2**: How does the number of sites affect predictive performance? In particular, does the performance of the evaluated models remain stable for a given dataset as the number of sites increases, or is it strongly dependent on the degree of data distribution? **RQ3**: What is the trade-off between predictive performance and communication cost for the evaluated models, particularly among the associative classifiers? **RQ4**: How does increasing the number of sites affect the structural complexity of associative classifiers?

Due to space constraints, we report figures for the MCC metric only; trends remained consistent across all other metrics. For interested readers, comprehensive results for all metrics and datasets are provided at our GitHub repository.

6.1 Predictive performance and stability (RQs 1 and 2)

Fig. 1 reports the predictive performance of the proposed and baseline models. For clarity, the *National Poll* dataset is excluded from the comparative analysis that follows, as the near-zero performance of all models precludes meaningful comparisons.

Overall, the modifications introduced in Algorithm 1 relative to DAC led to noticeable improvements in predictive performance. In 5 of 10 datasets, DAC achieved MCC values close to zero for the number of clients equal to 1 ($K = 1$). We attribute this performance mainly to its local model construction procedure, which was designed for extremely large datasets. In the datasets considered here, this strategy often failed to produce informative rules for all class labels. With the proposed refinements to the DAC framework, the single-site solutions ($K = 1$) became consistently stronger. At the same time, the approximate consolidation

[4] `https://www.ceci-hpc.be/clusters/`

strategy exposed a limitation that is less visible in DAC itself. Because the local models produced by our pipeline are stronger and more informative, losses due to approximate global metric estimation become easier to observe as the number of sites increases.

In contrast, the exact consolidation strategy in Algorithm 2 usually improved both predictive performance and stability across different levels of partitioning. The only exception was the *Car Evaluation* dataset, for which approximate consolidation produced better results than exact consolidation. In this dataset, several single-condition rules predicting the majority class exhibit high confidence and support. Two such rules consistently appear among the top-2 ranked rules in the classifiers produced by Algorithms 1 and 2: $\{safety = low\} \rightarrow \{label = unacc\}$ and $\{persons = 2\} \rightarrow \{label = unacc\}$, both achieving 100% confidence and relative support often above 30%. These rules are highly discriminative and contribute positively to the classification performance. However, other single-condition rules predicting *unacc* are less discriminative. In Algorithm 2, with exact computation of the global rule metrics and the server-side pruning step, some of these generic *unacc* rules appear in high positions of the rule list, being frequently activated during prediction and leading to additional misclassifications. On the other hand, under Algorithm 1, optimistic confidence estimates assigned to several other rules kept these generic *unacc* rules much lower in the ranking, causing them to be triggered infrequently and reducing their influence on the final predictions. This behavior suggests that, for the *Car Evaluation* dataset, the approximate metrics induced a rule ordering that is more favorable to the rule-list classifier, even though they are less accurate estimates of the true rule quality.

MLP and Algorithm 2 consistently delivered the strongest performance, with the superior model varying depending on the dataset and the number of sites involved. Their stability was also dataset-dependent. In some datasets, performance remained nearly constant as K increased, whereas in others it deteriorated gradually. For example, our approach remained stable on *MONK's Problems* dataset, while MLP showed a noticeable degradation. Conversely, on *Connect-4* dataset, MLP remained stable while our method showed a decline. These results indicate that the effect of horizontal partitioning is not uniform across datasets, and that robustness to increasing distribution depends on the interaction between the learning algorithm and the underlying data characteristics.

In summary, our results indicated that: the approximate consolidation approach (Algorithm 1) substantially improves upon DAC; and the exact consolidation approach (Algorithm 2) outperforms the approximate variant. Among all evaluated models, exact consolidation and MLP generally yielded the best results. Although the model performance across different numbers of clients depends on both the model and the dataset, our exact consolidation approach tended to provide the most stable results (since the degradation observed for MLP, when it occurred, was generally steeper).

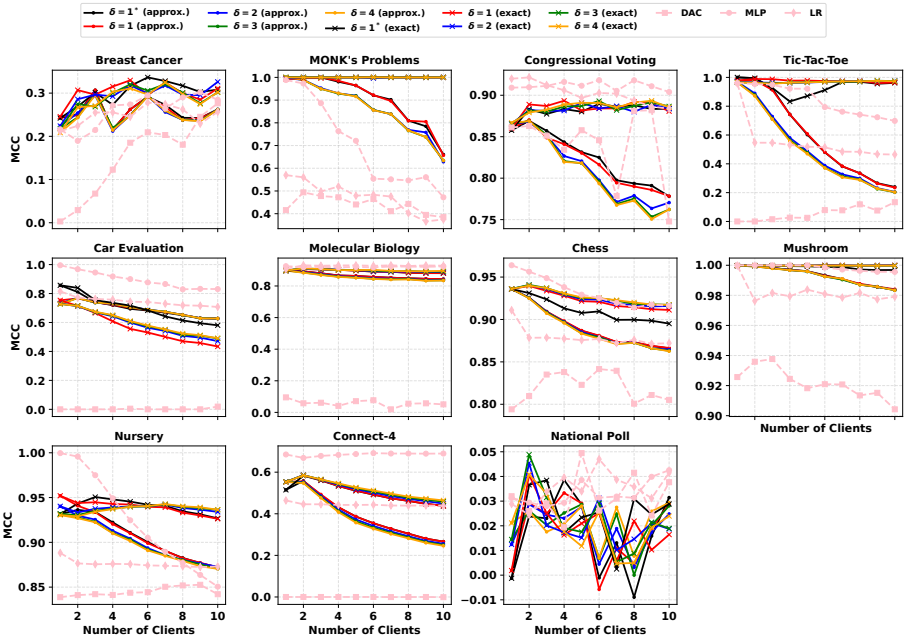

Fig. 1. MCC versus number of clients.

6.2 Communication cost (RQ3)

By our experimental design, LR and MLP incur communication costs that grow linearly with the number of sites. MLP was by far the most communication-intensive model, while LR is also often more expensive than the associative classifiers. LR exhibited communication costs of comparable magnitude only on *Tic-Tac-Toe*, *Chess*, and *Connect-4* datasets.

To illustrate the trade-off between predictive performance and communication cost, we focus on the setting with $K = 10$ clients, as this configuration typically exhibited the most pronounced differences between approximate and exact consolidation. For readability, Fig. 2 displays only the proposed methods. Although DAC often achieved communication costs that were lower than or comparable to those of the proposed methods, its lower predictive performance would compress the region of interest in the figure and hinder the visualization of differences among the proposed approaches.

As expected, the communication cost increased with the value of δ because larger coverage thresholds retain more rules in the classifier. Exact consolidation was also consistently more expensive than approximate consolidation due to the additional communication rounds. Even so, exact consolidation with $\delta < 2$ often yielded a noticeable gain in MCC while maintaining a communication cost comparable to approximate consolidation with $\delta \geq 2$. In addition, increasing δ beyond 2 in the exact variant rarely produced substantial predictive gains. In

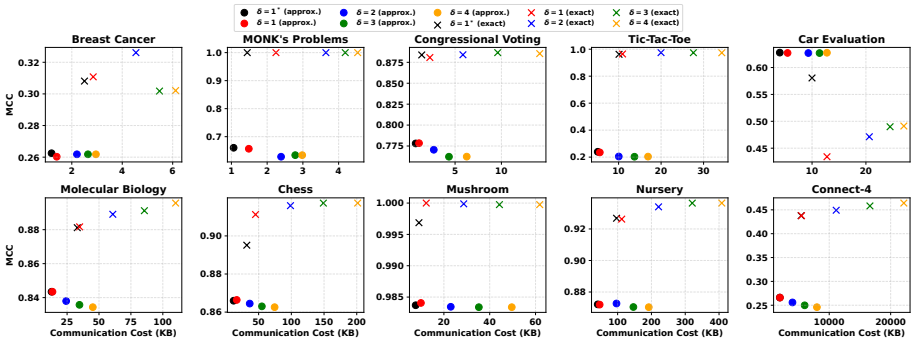

Fig. 2. MCC versus communication cost (number of clients = 10).

practical terms, the communication cost of the associative classifiers remained in the KB range for all datasets, except for the *Connect-4* dataset, where it was around 20 MB in the worst case.

These results suggest that exact consolidation usually offers the most useful practical trade-off among the distributed AC variants. Although DAC occasionally appeared on the Pareto front because of its low communication burden, its predictive performance was often too weak to make it an attractive choice in practice. Likewise, in the three datasets where LR had a communication cost comparable to the associative classifiers, its solutions were dominated. With the exception of *Car Evaluation*, where approximate consolidation was superior, Algorithm 2 provided the best balance between effectiveness and communication efficiency.

6.3 Model complexity (RQ4)

Fig. 3 shows the number of rules in the learned associative classifiers. As expected, the number of rules increased with δ. Exact consolidation produced the same number or fewer rules than approximate consolidation (because pruning was applied after computing exact global metrics). The effect of the number of sites was dataset-specific. For instance, the number of rules decreased with K for *Breast Cancer*; it increased approximately linearly for *Molecular Biology*; and it increased only up to $K = 7$ and then stabilized for *Nursery*.

The average rule length, shown in Fig. 4, was much less sensitive to the choice between approximate and exact consolidation. In general, average rule length increased slightly with the value of δ; it was often stable, or even decreased slightly, as the number of sites increased. This is an important result because it suggests that the use of closed itemsets does not lead to excessively long rules. In fact, average rule length exceeded 5 items only in a few cases: below 14 for *Chess*, below 8 for *Mushroom*, and around 30 for *Connect-4*. DAC tended to produce shorter rules, but this compactness came at the cost of substantially poorer predictive performance.

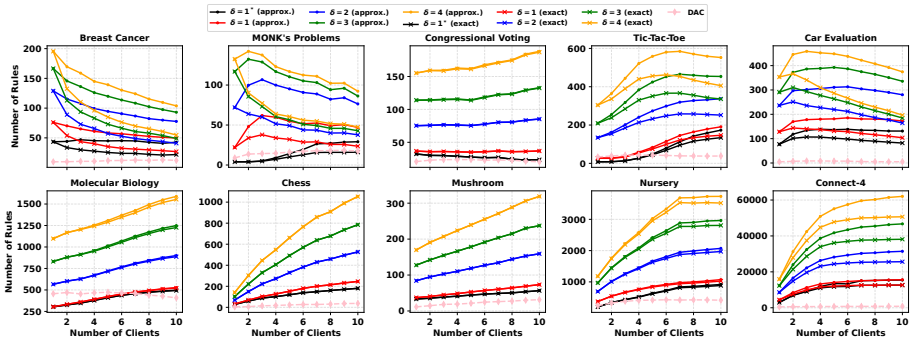

Fig. 3. Number of Rules versus number of clients.

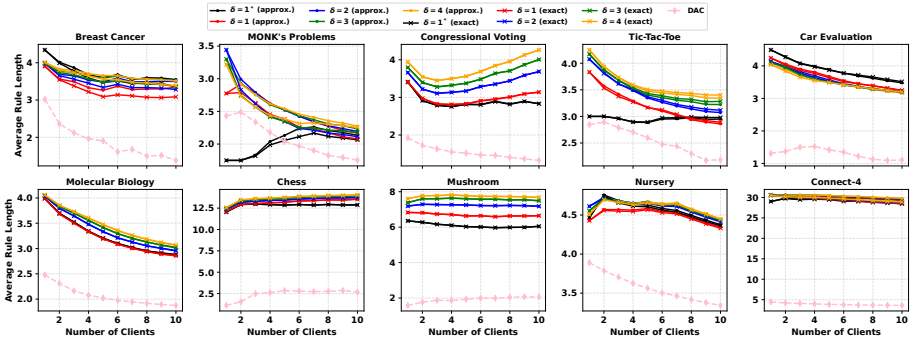

Fig. 4. Average rule length versus number of clients.

Taken together, the results presented in Figs. 3 and 4 indicate that classifier complexity is influenced more by design choices in the AC pipeline (e.g., the value of δ) than by the number of sites itself. In other words, increasing the degree of data distribution does not inherently force the learned associative classifiers to become more complex.

7 Conclusion

In this work, we studied the problem of AC in physically distributed environments with horizontally partitioned data. While associative classifiers have been extensively investigated in centralized settings, their application in collaborative scenarios involving multiple independent data owners remains largely unexplored. We addressed this gap by proposing two distributed AC frameworks that allow clients to collaboratively train a global classifier while keeping their raw data local. Both approaches preserve the classical AC pipeline at the local level, including rule generation, ranking, and coverage-based pruning, ensuring compatibility with well-established centralized methodologies. The first proposed

framework approximates global rule metrics using a communication-efficient consolidation strategy inspired by DAC, whereas the second method computes exact global rule metrics through additional communication rounds.

Experimental results on multiple datasets indicated that the proposed refinements to the DAC framework substantially improve predictive performance. Moreover, the proposed exact-metric framework consistently provided competitive classifiers across different levels of data partitioning, often achieving performance comparable to strong baseline models such as MLP. At the same time, the communication cost of the associative classifiers remained relatively low, typically in the KB range. Thus, our finding suggests that compact local rule sets can effectively capture the information necessary for building accurate global models in distributed environments, supporting the central hypothesis of this work.

Several directions remain open for future work. First, the proposed framework could be extended to more challenging federated learning settings, including non-IID data distributions and partial client participation. Second, incorporating privacy-enhancing mechanisms would further strengthen applicability in sensitive domains. Third, the design of distributed AC systems deserves further investigation, including how different choices in the AC pipeline (e.g., rule ranking, or prediction mechanisms) affect predictive performance and how these choices interact with dataset characteristics and the degree of data distribution. Finally, the approximate consolidation strategy explored in this work could be improved to provide more robust estimations of global rule metrics while preserving communication efficiency.

Acknowledgments. We thank the financial support from the AIDE project funded by the Belgian SPF BOSA under the program "Financing of projects for the development of artificial intelligence in Belgium" with reference number 06.40.32.33.00.10. Computational resources have been provided by the *Consortium des Équipements de Calcul Intensif* (CÉCI), funded by the *Fonds de la Recherche Scientifique de Belgique* (F.R.S.-FNRS) under Grant No. 2.5020.11 and by the Walloon Region.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Use of Generative AI Generative AI was used for language editing. The authors are responsible for the scientific content of the paper.

References

1. Andrews, S.: A new method for inheriting canonicity test failures in close-by-one type algorithms. In: CLA 2018 : The 14th International Conference on Concept Lattices and Their Applications, proceedings, vol. 2123, pp. 255–266. Springer (June 2018)
2. Baralis, E., Chiusano, S., Garza, P.: A lazy approach to associative classification. IEEE Transactions on Knowledge and Data Engineering **20**(2), 156–171 (2008)

3. Bechini, A., Marcelloni, F., Segatori, A.: A mapreduce solution for associative classification of big data. Information Sciences **332**, 33–55 (2016)
4. Geng, X., Yang, Z., Jiao, L., Zhou, Z.J., Ma, Z.: Association rule-based classification: A comprehensive review of methodologies and applications. Expert Systems with Applications p. 127454 (2025)
5. Khedr, A.M., Al Aghbari, Z., Al Ali, A., Eljamil, M.: An efficient association rule mining from distributed medical databases for predicting heart diseases. IEEE Access **9**, 15320–15333 (2021)
6. Konecny, J., Krajča, P.: Systematic categorization and evaluation of cbo-based algorithms in fca. Information Sciences **575**, 265–288 (2021)
7. Kulaif, A., Silva, J., Veroneze, R., Von Zuben, F.J.: Closed itemsets and minimal generators: A comparison of their effects on associative classifiers. Preprint, Research Square (2026). `https://doi.org/10.21203/rs.3.rs-8695223/v1`
8. Li, W., Han, J., Pei, J.: Cmar: Accurate and efficient classification based on multiple class-association rules. In: Proceedings 2001 IEEE international conference on data mining. pp. 369–376. IEEE (2001)
9. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings of the fourth international conference on knowledge discovery and data mining (1998)
10. Lucchese, C., Orlando, S., Perego, R., Silvestri, C.: Mining frequent closed itemsets from distributed repositories. In: Knowledge and Data Management in GRIDs. pp. 221–234. Springer (2007)
11. Nijssen, S., Kok, J.N.: Multi-class correlated pattern mining. In: International Workshop on Knowledge Discovery in Inductive Databases. pp. 165–187. Springer (2005)
12. Padillo, F., Luna, J.M., Ventura, S.: Evaluating associative classification algorithms for big data. Big Data Analytics **4**, 1–27 (2019)
13. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature machine intelligence **1**(5), 206–215 (2019)
14. Segatori, A., Bechini, A., Ducange, P., Marcelloni, F.: A distributed fuzzy associative classifier for big data. IEEE transactions on cybernetics **48**(9), 2656–2669 (2017)
15. Venturini, L., Baralis, E., Garza, P.: Scaling associative classification for very large datasets. Journal of Big Data **4**, 1–24 (2017)
16. Venturini, L., Garza, P., Apiletti, D.: Bac: a bagged associative classifier for big data frameworks. In: East European Conference on Advances in Databases and Information Systems. pp. 137–146. Springer (2016)
17. Veroneze, R., Banerjee, A., Von Zuben, F.J.: Enumerating all maximal biclusters in numerical datasets. Information Sciences **379**, 288–309 (2017)
18. Waseem, M., Abidin, S.: A comparative study on association rule mining in distributed data mining. In: 2024 IEEE International Conference on Computing, Power and Communication Technologies (IC2PCT). vol. 5, pp. 251–257. IEEE (2024)
19. Zaki, M.J., Meira, W.: Data mining and analysis: fundamental concepts and algorithms, 2nd Edition. Cambridge University Press (2020)