Collaborative Data Mining on a BDI Multi-Agent System over Vertically Partitioned Data

Jorge Melgoza-Gutiérrez Alejandro Guerra-Hernández Nicandro Cruz-Ramírez
Universidad Veracruzana,
Departamento de Inteligencia Artificial,
Sebastián Camacho No 5, Xalapa, Ver., México 91000
jorge_melgoza@yahoo.com, {aguerra, ncruz}@uv.mx

***Abstract*—This paper presents a collaborative learning protocol, modeled and implemented under the Agents & Artifacts paradigm, to deal with heterogeneous training data, i.e., vertical partitions of examples. The artifacts encapsulate Weka [7] objects and methods that implement the learning algorithm to induce decision trees, based on a modified version of J48; while the agents manage the flow of the learning process. The proposed protocol is tested with well known training sets of the UCI [1] repository, comparing the obtained accuracy against the centralized, usual implementation of J48. Our collaborative learning protocol achieves equivalent accuracy to that obtained with J48, without losing privacy.**

I. Introduction

Traditional data mining algorithms were designed assuming the information is stored in a single source [6], that means all the learning process must be done in a centralized site. This view has changed with the widespread use of networks, where data is collected from different sources and is not necessarily stored in a single place; because of high storage requirements or limited bandwidth; and also because of privacy concerns do not allow sharing the information.

The traditional centralized approach is not an option to do a learning process under those conditions, in the other hand, Distributed Data Mining (DDM) can address this issues by performing data analysis and mining process, in a distributed manner that pays attention to these resource constraints [5]. Multi-Agent System (MAS) are, in essence, also distributed so these could work with DDM almost naturally.

DDM faces two kind of problems in a distributed relational database system, homogeneous data and heterogeneous data. Homogeneous data, also known as horizontal partitioning, means every part of the distributed system knows the scheme, but no one has all the samples. In the other hand, heterogeneous data or vertically partitioned data, means, in terms of instances and attributes, each site has different attributes (features) of the same samples (instances), one example would be the distinct areas in a hospital: weight, age, blood type, etc, could be gathered by one unit; but data from oncology tests in a different unit, but due the internal policies you are not allow to share information between units, maybe you want to do some learning process and get a model about cancer disease, but you do not want to tell who is sick because of privacy concerns.

Combining DDM and MAS could be a feasible solution for those kind of problems. This paper proposes a collaborative learning protocol based on the Agents & Artifacts paradigm, implemented with Jason [2] and CArtAgO [9], [10]. This work will focus on the problem of vertically partitioned data and at the same time, the privacy preserving concern will be taken into account, while performing data mining to build decision trees, but in no case the instances will be shared directly between agents. This work also provides an implementation for the experimental setting.

Another approach for DDM modeled and implemented under the Agents & Artifacts paradigm, also using the agent oriented programming language Jason [2] and CArtAgO [9], [10] JaCA-DDM is presented in [12], this is a solution for horizontally partitioned data, the work described in this paper will be an extension to handle vertically partitioned data.

An agent-based approach for tree induction over vertically partitioned data sets and its application to the computer network intrusion detection area, is proposed in [11], it uses the original ID3 algorithm developed by Quinlan [8] and is designed for two agents interacting through a mediator to build the decision tree. Their goal was to reduce the inter-agent communication bandwidth by finding ways to reduce the amount of information necessary for agent collaboration, in order to reduce the need of collecting the data from distributed hosts by applying a distributed data analysis on those distributed hosts.

Another proposal for vertically partitioned data [3], focused the work on the efficient construction of decision trees over heterogeneously distributed data, by doing a random projection-based dot product estimation and message sharing strategy. According to their experimental results this technique reduces the communication by a factor of five while still retaining 80% of the original accuracy.

More focused on privacy preserving, in [4] a generalized privacy-preserving variant of the ID3 [8] algorithm for vertically partitioned data distributed over two or more parties is presented. In this approach even the meta-data is concealed, the schema (attributes and their possible values) are protected from disclosure and the class is only know by one participant. The algorithm has been implemented on top of Weka [7] with modifications to allow one instance to be stored in different parties, also the internal process to classify an instance is modified to let it move from parties according to the tree, so even after the model is builded the attributes values are not

share.

For this work the concern will be to achieve accuracy levels highly competitive against the standard centralized approach, using the J48 algorithm, an open source Java implementation of the C4.5 decision tree algorithm, but the instances could not be share between members. In a future work more privacy preserving techniques could be tested. The collaborative learning protocol proposed here will work with vertically partitioned data where no one share the same attribute or attributes and the class is well known for all the parties, in this case agents.

This paper is organized as follows. Section 2 details the collaborative learning protocol. Section 3 explains particular details of the implementation. Section 4 presents the experimental settings. Section 5 shows the results obtained. Finally, Section 6 closes with the discussion and future work.

II. Learning Protocol

Decision tree building algorithms like J48, in order to build the tree, need to perform this steps (speaking generally): evaluate if there's a valid split and, if is the case, evaluate the attributes and choose the best. This valid split criterion could be for example enough instances to keep splitting the data. This lead us into two main choices: split the data and add another level to the tree; or make a leaf node. In a vertically partitioned data environment, these steps can be separated in two types: those you can do it just with your own data and those where you need all the data.

Agents can evaluate his own attributes but can't chose the best by their own, need to share some data to be able to compare all attributes. In the learning protocol proposed here, the agents will perform locally the attributes's evaluations and then, using a trusted third party (the *classifier artifact*), choose the best attribute to split.

A. Agents

Agents play two different roles in the learning process, the *learner* and the *worker*. All agents know the plans to perform each role, but to keep the order in the process, just one agent can be the *learner*.

1) Learner: The *agent learner* controls the learning process step by step, he decides when to ask for data to the workers, but this doesn't mean he can access all the data, his role is more like a guide. The *learner* evaluates the stop criterions of the algorithm and builds the leaf nodes when it's necessary. He also performs workers tasks when everybody does.

2) Worker: All agents do a worker's job. This kind of agents do the local evaluations with his local data, *information-gain* measures for example. When the best attribute is chosen the *worker* with that attribute is the one that builds the new node.

Following will be described the implemented artifacts to pass to the interaction through the learning protocol between the agents, and them with the artifacts, in order to make it more clear to understand.

B. Artifacts

An artifact is a first order abstraction used for modeling computational environments. Each artifact represents an entity of the environment, offering services (operations) and data structures (observable properties) to the agents, in order for them to improve their activities. Artifacts are conceived as function-oriented computational devices, agents may exploit this functionalities in order to fulfill their individual and social activities.

For the learning process proposed here, there are two main artifacts: *classifier* and *evaluator*, each agent has an *evaluator artifact* but there is only one *classifier artifact*.

1) Classifier Artifact: This artifact encapsulates the tree structure and provides operations for agent coordination and attribute selection, gets the attributes's evaluations and new nodes to attach to the tree.

1) Observable property, *nodeStatus*, reveals the current node stage.
   a) 0 Means it's a new node so the agents will perform a learning cycle.
   b) 1 The current node has a child to be processed.
   c) 2 Leaf node or each child has been already processed.
   d) 3 When the backtrack reaches the root so the tree is complete.
2) Operations
   a) backtrack. Moves to current node's father, update *nodeStatus* and send a *backtrack* signal to all agents.
   b) checkValidM. Check the enough-valid-models flag.
   c) findBest. Chose the best attribute and check if useful split was found.
   d) getValues. For social strategy, allows the agent to put the attributes's evaluations inside the object.
   e) processAvInfG. Calculate average *info-gain*.
   f) sendRequest. Send *requestatt* signal to the winner agent.
   g) setReady. For coordination between agents. Sends a signal when all agents notify that they are ready.

2) Evaluator Artifact: This artifact encapsulates Weka [7] objects and methods to evaluate the attributes and build the nodes. It controls the stack of data partitions (add and drop). Also sends the evaluations and nodes to the *classifier artifact*. This artifact has no observable properties due the privacy preserving concerns.

1) Operations
   a) checkMultival. Check if at least one attribute is nominal and have a lot of values.
   b) drop. Remove last partition from the stack.
   c) evalAttributes. Evaluate all local attributes from the last partition in the stack.
   d) evalSplit. Check if all Instances belong to the same class or if not enough Instances to do the split.

e) getTrainData. Create the stack and set the first partition.
f) sendAttibute/sendNosplit. Send a new node according to the attribute selection results. If there is a valid split it builds the node with the winner attribute, if not Nosplit builds a leaf node.
g) sendMultival. Update multival-flag according to the results of all participants.
h) sendValues. Send attributes's evaluations to the *classifier artifact*.
i) setTraintmp. Creates new partitions and adds these to the end of the stack.

C. Learning Process

Figure1 shows the interaction between agents and artifacts through the learning process.

Here is a more detailed explanation of this steps:

1) The agent *learner*, using the *evaluator artifact* verifies the stopping criterion. All instances of the current partition belong to the same class, or there is not enough instances to perform a valid partition (2 minimum according to J48), either one of this is true it means is a leaf node.
2) Next a flag is validate for all agents according to J48, it becomes false if at least one attribute is nominal and has many values. The result is shared by the agents through the *classifier artifact*.
3) Then the agent *learner*, asks everyone to evaluate their attributes using the methods encapsulated in their own *evaluator artifact*, the results are send to the *classifier artifact*. Each agent sends a list made by the agent's name, an local identifier for his attributes (not the attribute name, in this case an integer) and the values, then tells he is ready trough the *classifier artifact*. [1]
4) When every agent notifies he is ready, a signal is emitted and the agent *learner* proceed to calculate the *total valid models* and *average information gain* values. If there is no valid models, the stopping criterion of the algorithm is reach and thus a leaf node was found.
5) After the calculations the agent *learner* uses the *classifier artifact* to establish which is the best attribute, then a signal is sent to the corresponding agent telling him the attribute's identifier.
6) The agent with the best attribute uses his *evaluator artifact* to create the corresponding node and send it to the *classifier artifact*, finishing this cycle with a signal to everyone.

Each time a leaf node is builded a signal is emitted so all agents update their partitions for the next cycle.

[1]Two solutions were proposed to test the protocol: one called *social* approach, where the *workers* pass directly to the *learner* the results of the attributes's evaluations by the use of speech acts, and the *learner* put those in the *classifier artifact* to do the attribute selection; in the other hand in the called *improved social* approach, no direct communication of the attributes's evaluations takes place, instead, the data is sent through artifact connections so no one share directly the information.

III. Implementation Details

The agent *learner* controls the learning process workflow and at certain stages requests information to other agents, such as for example, the *information gain*, so is not possible to run the procedure recursively because each agent does the calculations locally.

As is known J48, been an extension of the original ID3 algorithm, build the tree by Depth-first search. The recursion allows each new call to get the corresponding data partition that will be processed. In order to do the same, the collaborative learning strategy proposed implements a stack, where the partitions will be managed.

Each agent initializes his stack with his entire local database. Each cycle, all agents will take the last element from his stack to perform the attribute evaluation. If the agent *learner* that coordinates the process, determines that it has been found a valid partition, all agents will be notified with a list of the indexes corresponding to the instances belonging to the new partition or partitions made (it can be a list of lists), according to the selected attribute.

Next each agent will send to the end of his stacks the new partitions made using the last partition of the stack, according to the lists received. If the agent *learner* determines that there is no valid partition, it means is a leaf node, so the agents will be notified to drop the last element from the stack like doing a *backtrack*. The *backtrack* process is also call after all children from a node are processed.

Weka [7] was made for centralized processing, so another modification was made. Internally when the tree is builded, to identify the corresponding attribute for each node, only an integer is stored in the structure, because is expected there will be only one scheme and the attributes will not change position. But when this is moved to the distributed environment, every party could have an attribute identified by the same number 1, 2, etc.

To avoid this miss reference to attributes, the internal process was modified so also the name of the attribute is internally stored by the tree, no agent can see it because is only for inside processing when classifying an instance, but that way, when the classify process is called each node refers to the proper attribute.

IV. Experimental Design

Table I contains the databases used for the experiments reported, all get from the UCI [1] repository.

Stratified cross validation with 10 folds and 10 repetitions was applied. For each database, experiments were done with 2, 4, 7, 10, 15 and 20 agents, do the number of attributes only when there is enough for at least one for each agent.

JaCA-DDM [12] provides a tool to setup the experiments, it creates random stratified data partitions and distributes them among the agents, but is designed for horizontally partitioned data. This work of loading the data set and make all the partitions for the experiment is done by the *oracle artifact*. It loads an ARFF file and creates random partitions according to the number of agents that will take place in the learning process.

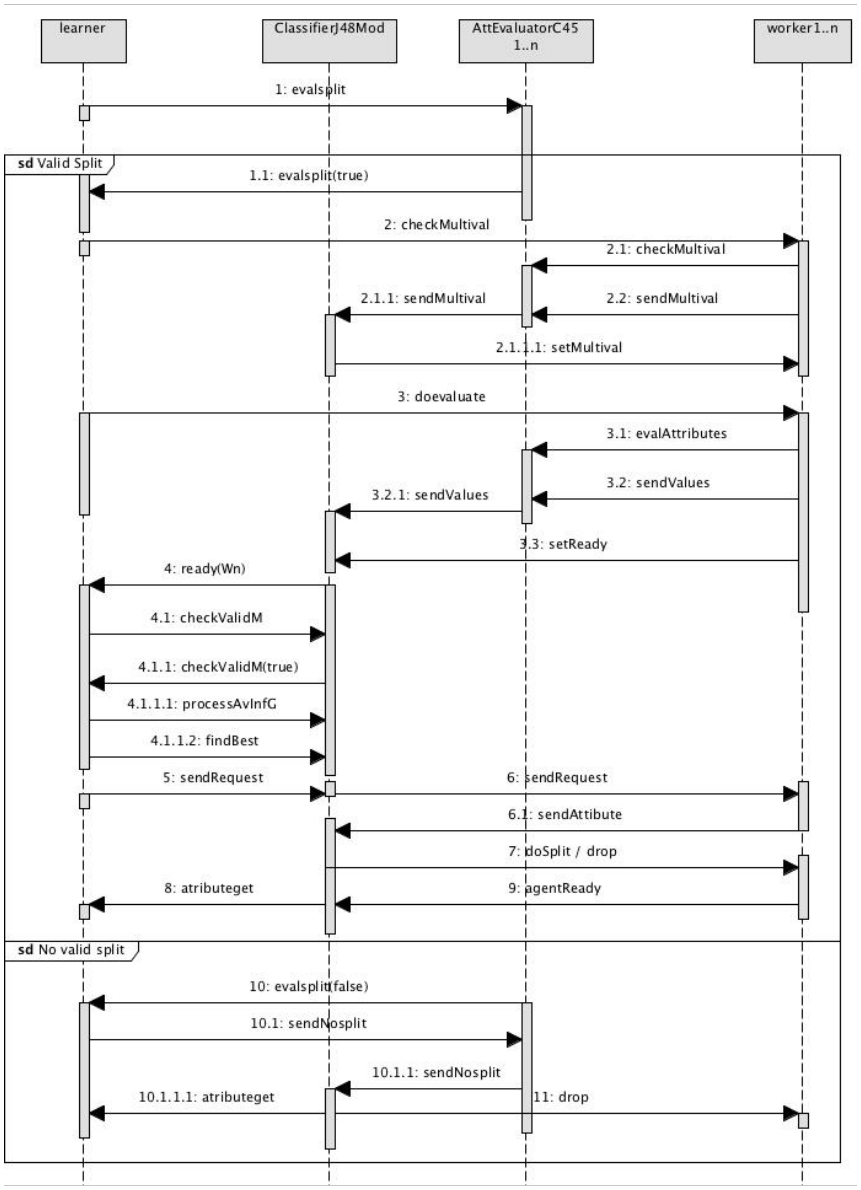

Fig. 1. Learning process interaction

TABLE I. DATA SETS

Data Set	Instances	Attributes	Classes
australian	690	14	2
breast	683	9	2
car	1728	6	4
german	1000	20	2
heart	270	13	2
iris	150	4	3
letter	20000	17	26
mushroom	8124	22	2
waveform	5000	40	3

TABLE II. ACCURACY RESULTS

Data Set	#A	Centralized	Social	Improved Social
australian	2	83.6377	83.6667	83.7101
australian	4	83.6377	83.5797	83.6377
australian	7	83.6232	83.5652	83.7681
australian	10	83.7246	83.6812	83.6667
breast	2	95.6215	95.7089	95.6503
breast	4	96.0934	96.0786	96.0782
breast	7	96.0735	96.0586	96.1172
car	2	93.4785	93.3685	93.3743
car	4	93.7211	93.6343	93.6575
german	2	68.2600	68.1400	68.1900
german	4	68.8100	68.6100	68.6700
german	7	68.4300	68.4700	68.5500
german	10	69.1600	69.1300	69.1500
german	15	68.6300	68.6100	68.6200
german	20	68.8000	68.9000	68.7000
heart	2	77.1111	77.1111	77.0741
heart	4	77.4444	77.5185	77.6667
heart	7	77.2963	77.5926	77.4074
heart	10	77.3333	76.9259	77.3333
iris	2	93.8000	93.7333	93.6000
iris	4	94.8667	94.8000	94.8000
letter	2	88.0065	88.0295	88.0210
letter	4	88.0775	88.0835	88.0575
letter	7	88.1155	88.1200	88.1025
letter	10	87.9380	87.9295	87.9520
letter	15	87.9825	87.9825	88.0105
mushroom	2	100.0000	100.0000	100.0000
mushroom	4	100.0000	100.0000	100.0000
mushroom	7	100.0000	100.0000	100.0000
mushroom	10	100.0000	100.0000	100.0000
mushroom	15	100.0000	100.0000	100.0000
mushroom	20	100.0000	100.0000	100.0000
waveform	2	75.2400	75.3220	75.3720
waveform	4	75.2040	75.1500	75.3240
waveform	7	75.5400	75.6520	75.5960
waveform	10	75.3740	75.4080	75.4460
waveform	15	75.3460	75.5020	75.5160
waveform	20	75.5900	75.4300	75.4940

This artifact was modified so it could also create vertically partitioned data, to do this it makes a shuffled list of attributes (list of integers) and using a roulette gives one by one the attributes to the *n* partitions for the *n* agents, this list is saved so all the strategies, the centralized and social ones, get the same scheme each cycle, for the cross validation method each fold also gets the same partitions, just the instances for the train and test partitions do change.

V. RESULTS

The Table II reports accuracy measures obtained for each different model. It shows the mean for every data-set and number of agents combination. The tailed paired T test with 0.05 degrees of significance was also calculated to verify if there are significant differences between the strategies. 0 means there is not significant difference; -1 means that the first strategy paired won; and 1 means that the first strategy lost. Obtained data from the tailed paired T test is not show in the table because all comparisons gave 0 as result.

TABLE III. TIME MEASURING RESULTS

Data Set	#A	Centralized	Social	Improved Social	SvsIS
australian	2	8.15	6,848.25	5,248.68	0
australian	4	7.75	6,880.79	5,483.51	0
australian	7	7.85	8,881.66	9,328.11	0
australian	10	7.89	7,339.87	7,490.95	0
breast	2	3.85	2,251.65	2,186.84	0
breast	4	4.08	3,018.82	2,595.51	0
breast	7	4.55	4,704.22	4,558.54	0
car	2	6.26	6,608.78	5,434.17	0
car	4	6.43	8,611.05	7,747.26	0
german	2	12.75	12,843.72	7,140.47	1
german	4	13.02	11,278.33	9,318.01	0
german	7	13.98	32,178.03	32,221.80	0
german	10	14.15	24,781.82	25,154.03	0
german	15	15.27	59,475.05	57,788.06	0
german	20	15.61	124,261.13	128,091.83	0
heart	2	3.99	3,080.62	1,702.95	1
heart	4	4.41	4,148.98	3,676.86	0
heart	7	4.59	5,269.74	5,508.55	0
heart	10	4.66	4,188.10	3,797.46	0
iris	2	3.48	775.19	698.34	0
iris	4	3.79	711.56	771.53	0
letter	2	3,386.61	97,067.45	80,622.19	1
letter	4	3,454.92	151,577.55	122,904.79	1
letter	7	3,441.29	196,337.16	178,184.22	1
letter	10	3,373.14	181,797.80	170,953.10	1
letter	15	3,454.74	398,810.68	352,934.27	1
mushroom	2	30.07	1,302.95	1,124.56	0
mushroom	4	31.82	1,021.48	957.93	0
mushroom	7	32.85	1,838.37	1,877.59	0
mushroom	10	34.29	2,229.76	2,245.37	0
mushroom	15	36.21	3,703.65	2,040.21	1
mushroom	20	36.86	7,814.06	5,365.93	1
waveform	2	808.59	20,076.27	15,279.44	1
waveform	4	817.30	23,973.18	18,600.12	1
waveform	7	821.73	33,587.63	31,784.22	0
waveform	10	825.37	26,397.31	21,177.41	1
waveform	15	828.80	33,652.16	22,976.08	1
waveform	20	833.82	105,877.91	95,614.76	1

The time needed to build the model was also measured. In this stage the main focus was to achieve competitive accuracy levels, the time is relevant just to compare the performance between the two social strategies but not against the centralized approach, because there is no transmission cost do the concurrent processing, this times are show in Table III, the scale is in milliseconds and the last column represents the tailed paired T test results for the social vs improved social comparison. The results comparing the centralized approach against the two social strategies gave -1.

The centralizing process takes less time to build the model, but this is expected because each cycle of the learning algorithm, the centralized approach only consumes time to evaluate the attributes and choose the best among them, but the social approach also needs time for a reasoning cycle of each agent, in the better case only the longer reasoning cycle from one of the agents (completely parallel or concurrent) but the worst case could be a sum of them (not concurrent). The results show mostly no significant differences, in the time measured, between the two social strategies, and when there is, it favors the improved social approach.

According to the time measures comparing the two social strategies, the improved social strategy proves to be a feasible solution, it conceals better the information to the learning parties than the social strategy and also take less time to build the model for some data sets.

VI. Discussion and Future Work

The results shown in Table II prove there is no significant differences on accuracy between the model obtained by the traditional centralized approach and the strategy proposed in this paper, so the collaborative learning protocol achieves equivalent accuracy to that obtained with J48, without losing privacy.

The agents & artifacts paradigm, proved to be a very helpful tool, the particular methods for attribute evaluation are enclosed inside the artifact, so the learning protocol is not linked to an specific metric or method to do this evaluation.

The strategy proposed has proven not only feasible but also, without sharing the data directly, able to build models highly competitive against the centralized traditional approach while working with vertically partitioned data. In [12] an approach for DDM was proposed to deal with horizontally partitioned data, here this was enhanced so it can also work for vertically partitioned data, another improve so it can work over data partitioned both horizontally and vertically will be interesting in a future work.

Even when the measured times not involve real transmission cost, the improved social approach is slightly faster than the social approach, so it shows that for data communication, the link between artifacts is more efficient than the communication between agents and its better in terms of security do the fact it only allows the agents to see certain predefined information not all the data in the artifact.

An agent-based approach was made in [11] for tree induction but only for two agents, in this paper a similar work was done but achieving to make it work with n number of agents, it was tested with more agents than attributes with no side effects, the agents with no attributes just do not take part in the learning process.

The work done in [3] is more focused on efficiency on communication when constructing the tree, even when they get less accuracy levels than the traditional centralized approach and here the collaborative strategy proposed achieve same accuracy levels, direct comparison its not fair, in this work the cost of communication was not measured because the main focus was to achieve the accuracy levels and at the same time do not share the information between agents, the efficiency of communication can be taken on account for future work but it was not considered at this stage of the investigation.

The strategies proposed here assume the class is well known for each party, in [4] the privacy concerns are taken more deeply and even that is not shared, both works use not the same learning algorithm, one implements ID3 and the other J48, but the techniques used for privacy preserving appear interesting to take into account for future improvements of this work. Conceal the attributes and class values, as well as security protocols for the data communication between agents could be tested with J48, this algorithm implies more communication to build the tree than ID3, so J48 needs to share more.

This lead us to some interesting questions like, how many attributes are needed to be able to build a competitive model?, in this case the gain-ratio and info-gain from all attributes is communicated, what is the result if each party just sends the best local attribute?. For future work this could be tested, also the use of some other statistical ratios to determine the best attribute, and combining more than one metric.

In this work all the experiments were done in a concurrent environment, so even when the time for each learning process was measured, it is not relevant data because the transmission costs were near to zero, it will be interesting to do more test in a real distributed environment, to see the influence of traffic and other network related problems on the learning process times and performance.

Also design another social protocol where the scheme is builded by the participant agents could be done in a future work, will be interesting to test different decision support strategies.

References

[1] Bache, K., Lichman, M.: UCI machine learning repository (2014)

[2] Bordini, R.H., Hbner, J.F., Wooldridge, M.: Programming multi-agent systems in Agent Speak using Jason, vol. 8. Wiley-Interscience (2007)

[3] C. Giannella, K. Liu, T. Olsen, and H. Kargupta. Communication Efficient Construction of Decision Trees Over Heterogeneously Distributed Data. In Proceedings of the Fourth IEEE International Conference on Data Mining, pages 6774, (2004)

[4] Jaideep Vaidya, Chris Clifton, Murat Kantarcioglu, and A. Scott Patterson. 2008. Privacy-preserving decision trees over vertically partitioned data. ACM Trans. Knowl. Discov. Data 2, 3, Article 14, 27 pages.(2008)

[5] Josenildo C. da Silva, Chris Giannella, Ruchita Bhargava, Hillol Kargupta, and Matthias Klusch. 2005. Distributed data mining and agents. Eng. Appl. Artif. Intell. 18, 7 , 791-807, (2005)

[6] Li Zeng et al, "Distributed data mining: a survey", Springer US, Volume 13, Issue 4, pp 403-409.(2012)

[7] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten; The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1. (2009)

[8] Quinlan, J. R., and Rivest, R.L., Inferring Decision Trees Using the Minimum Description Length Principle, Information and Computation, Vol. 80, no. 3, (1989)

[9] Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multi-Agent Systems 23(2), 158192 (2011)

[10] Ricci, A., Viroli, M., Omicini, A.: Construenda est CArtAgO: Toward an infrastructure for artifacts in MAS. Cybernetics and Systems 2, 569574 (2006)

[11] S. Baik, J. W. Bala, A decision tree algorithm for distributed data mining: Towards network intrusion detection, in: ICCSA, (2004)

[12] Xavier Limn; Alejandro Guerra Hernndez; Nicandro Cruz Ramrez; Francisco Grimaldo, F. Castro, A. Gelbukh, M.G. Mendoza (Eds.): MICAI 2013, Part II, LNAI 8266, pp. 338349, 2013. Springer-Verlag Berlin Heidelberg (2013)