SSL-VFC4.5: An approach to adapt the Very Fast C4.5 classification algorithm to deal with semi-supervised learning

Carlos Eduardo Nass[1], **Agustín Alejandro Ortíz Díaz**[1], **Fabiano Baldo**[1]

[1]Departamento de Ciência da Computação
Programa de Pós-Graduação em Computação Aplicada
Universidade do Estado de Santa Catarina (UDESC)
Joinville – SC – Brazil

cadunass@hotmail.com, agaldior@gmail.com, fabiano.baldo@udesc.br

Abstract. *The growing popularity of audio and video streaming, industry 4.0 and IoT (Internet of Things) technologies contribute to the fast augment of the generation of various types of data. Therefore, to analyze these data for decision-making, supervised machine learning techniques need to be fast while keeping a suitable predicting performance even in many real-life scenarios where labeled data are expensive and hard to be gotten. To overcome this problem, this work proposes an adaptation to the Very Fast C4.5 (VFC4.5) algorithm implementing on it a semi-supervised impurity metric presented in the literature. The results pointed out that this adaptation can slightly increase the accuracy of the VFC4.5 when the datasets have the presence of a very few amount of labeled instances, but it increases the training time, especially when the number of labeled instances in the datasets increase.*

1. Introduction

The amount of data generated and shared daily is increasing fast in the last years. This phenomenon is pushed up by emerging technologies like audio and video streaming, industry 4.0 and IoT (Internet of Things). According to IDC, more than 59 zettabytes (ZB) of data were created, captured, copied, and consumed in the world only in 2020 [Reinsel et al. 2021]. This means that efficient analytics solutions need to be developed to handle the data generated in this scenario [Ip et al. 2018]. Therefore, the Machine Learning techniques need to be even faster to process this huge amount of data while keeping a suitable performance.

Among the available supervised learning techniques, decision trees are preferred in many applications because they provide an easy to visualize and understand model and they are one of the most powerful methods in machine learning [Bifet et al. 2017]. However, in many supervised learning applications, labeled instances are difficult, expensive, or time-consuming to be obtained because they require empirical research or experienced human annotators [Tanha et al. 2014]. Therefore, this circumstance would be an inhibitor factor to apply decision trees.

Nevertheless, this shortcoming can be mitigated by enabling the decision tree induction algorithms to support semi-supervised learning. The goal of semi-supervised learning is to combine the information of the unlabeled examples with the explicit classification of the labeled examples for improving the classification performance [Chapelle et al. 2006]. Therefore, unlike a traditional decision tree, the semi-supervised

decision tree considers not only the class labels but also the distributions of input attributes, which reflect and incorporate the structural or topological properties of the training dataset [Kim 2016].

Decision trees have another limitation concerning fast classification related to their high computational cost, mainly in datasets with continuous attributes. To find the optimal cut point of the continuous attributes, they need to sort the n examples for the m attributes, which leads to a complexity of $O(m \times n \log n)$ and then calculates the split information gain for each k cut point which leads to a complexity of $O(m \times n \times k)$ [Cherfi et al. 2018]. Therefore, traditional decision tree algorithms are not prepared to deal with fast and semi-supervised model induction.

This work adapts the Very Fast C4.5 (VFC4.5) algorithm to enable it to deal with semi-supervised learning called SSL-VFC4.5. The VFC4.5 was chosen because it adapted the powerful and well-known C4.5 algorithm speeding up its models induction by lowering the computation time complexity of finding the best cut point of continuous attributes to $O(2 \times m \times n)$ [Cherfi et al. 2018]. Besides that, its ability to handle semi-supervised learning is achieved by replacing its entropy-based supervised induction metric to the impurity semi-supervised induction metric proposed by [Levatić et al. 2017].

The rest of the paper is organized as follows: Section 2 presents a overview of fast and semi-supervised learning decision trees. Section 3 describes the proposed adaptation in VFC4.5. Section 4 shows the experiments and results. Finally, 5 highlights the conclusion and future work.

2. Related Works

There are several works in the literature concerning semi-supervised classification using decision trees. Most of them utilize one or more supervised base learners. They iteratively train models with the original labeled data as well as with the predicted pseudo-labels of unlabeled instances from earlier iterations of the learners using approaches such as seft-training, co-training or boosting, like those presented by [Leistner et al. 2009], [Song et al. 2011], [Settouti et al. 2013], [Santos and Canuto 2014] and [Tanha et al. 2017]. However, standard decision trees can not be effective in semi-supervised learning using self-training because they do not produce reliable prediction probability estimation [Van Engelen and Hoos 2020]. Besides that, ensemble methods like boosting and Random Forest are not fast due to the induction of several models.

Methods based on the clustering-then-label approach, such as [Chen and Wang 2011], [Tanha et al. 2014] and [Kim 2016], apply an unsupervised clustering to guide the classification process, not considering the structural characteristics of data for partitioning [Van Engelen and Hoos 2020]. Besides that, other are intrinsically semi-supervised methods, which use labeled and unlabeled samples directly into the induction metric, like those presented in [Levatić et al. 2017], [Levatić et al. 2018] and [Ortiz-Díaz et al. 2020]. They have the advantage of do not rely on any intermediate steps or supervised base learners [Van Engelen and Hoos 2020].

Regarding fast learning, the Very Fast Decision Tree (VFDT) evaluates the data only once and update the model, discarding that data afterwards. This allows the model

increases continuously [Domingos and Hulten 2000]. Also, the Very Fast C4.5 (VFC4.5) improves the way of C4.5 finds the threshold of a continuous attribute by reducing the number of candidate cut points [Cherfi et al. 2018]. However, their shortcoming is that both do not support the semi-supervised learning. On the other hand, the Clustering Feature Decision Tree (CFDT) integrates a scalable method of micro-clustering within the VFDT algorithm as classifiers in tree leaves to improve classification accuracy and reinforce the any-time property [Xu et al. 2011].

We propose a fast semi-supervised decision tree induction algorithm using the VFC4.5 as a base classifier and substituting its original split metric (entropy) for the semi-supervised impurity metric proposed by [Levatić et al. 2017].

3. Proposal: SSL-VFC4.5 algorithm

This work proposes an adaptation for the Very Fast C4.5 called SSL-VFC4.5, which enables it to perform semi-supervised classification. This new proposal takes the VFC4.5 algorithm [Cherfi et al. 2018] as base classifier and introduces on it the semi-supervised impurity-based metric proposed by [Levatić et al. 2017] to calculate the gain of information when splitting a node. Therefore, the gain of information is calculated by two fundamental components: (1) the impurity for labeled instances and (2) the impurities for unlabeled instances.

The first component calculates the impurity based on the Normalized Entropy, using the original metric of the VFC4.5 algorithm. This calculation is done using the Equation 1.

$$Impurity_l(E_l, Y) = \frac{Entropy(E_l, Y_l)}{Entropy(E_l^{full}, Y_l^{full})} \tag{1}$$

Where, E_l and Y_l represent the set of instances and labels of the analyzed labeled data in a given node of the tree, and E_l^{full} and Y^{full} represent the set of instances and labels of all labeled data used in the construction of the tree.

The second component, which comprehends the treatment of unlabeled data, uses the Normalized Gini to calculate the impurity for attributes with nominal values and the Normalized Variance for attributes with numeric values, using Equation 2 and Equation 3, respectively.

$$Impurity_u(E, X_i) = \frac{Gini(E, X_i)}{Gini(E^{full}, X_i^{full})}, if X_i\_is\_nominal \tag{2}$$

$$Impurity_u(E, X_i) = \frac{Variance(E, X_i)}{Variance(E^{full}, X_i^{full})}, if X_i\_is\_numerical \tag{3}$$

Where E represents the set of instances in a given tree node (whether labeled or not) and X_i the set of values from its i^{th} attribute being evaluated, E^{full} represents the set of all instances of the tree (whether labeled or not) and X_i the set of all values from its i^{th} attribute being evaluated.

Finally, these two components are combined in Equation 4 to calculate the impurity of a node. The $Impurity_l$ performs the supervised part of the training, using only the

instances with a label in Y to calculate the impurity metric with Equation 1. In contrast, the $Impurity_u$ performs the unsupervised part of the training, carrying both labeled and unlabeled instances, where the values of the $X_i's$ attributes are used for calculating the impurity metric, with Equation 2 for nominal $X_i's$ and Equation 3 for numerical $X_i's$. This formula presents a w parameter used to define the impact of unsupervised learning on the induction process. It can vary from 0 to 1, comprehending completely unsupervised and fully supervised learning, respectively.

$$ImpuritySSL(E) = w * Impurity_l(E_l, Y) + \frac{(1-w)}{|X|} * \sum_{i=1}^{|X|} Impurity_u(E, X_i) \quad (4)$$

3.1. Implementation

The proposed SSL-VFC4.5 algorithm was implemented over the J48, which is an implementation of the C4.5 algorithm available in Weka [Hall et al. 2009]. Algorithm 1 shows its main structure. In line 1 an empty list called $Analysis\_pending\_nodes\_list$ is initialized. This list stores all the nodes of the tree that will be analyzed. In line 3, the root node N is added as the first to be analyzed. This first node comprehends all the instances of the analyzed dataset (S). Starting from the root node, the top-down classification tree induction process is performed by means of successive nodes division. Inside the loop (lines 5 to 11), while there are still nodes to be analyzed, the algorithm tries to split them (lines 7 and 8) using the best attribute selected by Algorithm 2 (line 6) and includes these new nodes in the pending nodes list to be analyzed further (line 9), otherwise it marks the node as a leaf (line 11).

Algorithm 2 presents how the best split attribute and cut-off point are selected. For each attribute (line 1), if the attribute is nominal (line 2), it splits the instances of the analyzed node according to their nominal values (line 3). However, if the attribute is numerical (line 4), it splits the instances of the analyzed node according to their mean and median values. After that, it verifies which of the two values produces the highest information gain and selects it (lines 7 to 10). Finally, it substitutes the previously found optimal split attribute and cut-off point to the new one, if it provides higher information gain.

To implement the SSL-VFC4.5, the main modifications in the original version of the C4.5 algorithm were:

1. *Modification of the metric used to calculate the impurity of the node*. The impurity metric showed in Equation 4 replaced the original entropy metric of the C4.5 algorithm. This metric introduces the analysis of unlabeled instances to determine the best division attribute of each node (line 6 of Algorithm 1). Therefore, it assesses the characteristics of the attributes in both labeled and unlabeled instances to determine if there are gains in dividing the analyzed node.
2. *Substitution of the way to select the cut-off value for numeric attributes.* In the proposed SSL-VFC4.5, the original way of calculating the cut-off points was replaced by the calculation of the mean and median values as suggested in VFC4.5 [Cherfi et al. 2018]. Algorithm 2 shows how the mean and median values are cal-

culated and the one that provides the greatest gain of information is selected as the cut-off point.

Algorithm 1: Semi-Supervised Learning Very Fast C4.5

Input:
S: Set of labeled or unlabeled examples.
$w[0 \ldots 1]$: Weight parameter for supervised and semi-supervised learning.
$attList$: List of attribute.

Output: $SSL\_VFC4.5\_Tree$: A model decision tree for semi-supervised learning induced according to the S training instances data set.

```
1  pendingNodesList ← Ø
2  Create SSL_VFC4.5_Tree a tree with a single node N(the root) with all
     instances
3  pendingNodesList ← pendingNodesList ∪ N
4  while pendingNodesList <> Ø do
5      Extract nextNode from pendingNodesList
6      splittingCriterion ← selectBestSplitAttribute(nextNode, attList)
7      if splittingCriterion = true then
8          newNodes ← split(nextNode, splittingCriterion)
9          pendingNodesList ← pendingNodesList ∪ newNodes
10     else
11         Mark nextNode as a leaf node labeled with majority class
12 return SSL_VFC4.5_Tree
```

Algorithm 2: Select the best split attribute for a node

Input: S_{mn} : Set of m instances with n attributes that belong to the analyzed node (nextNode); $attList$: List of attribute.

Output: splittingCriterion(a_{opt}: the optimal attribute, cp_{opt}: the best cut-off point);

```
1  foreach att_i ∈ attList do
2      if att_i is nominal then
3          S_cp ← Split S by the nominal values of the att_i
4      else
           /* If the attribute is numerical                    */
5          S_cp1 ← Split S by the mean(S_1i,...,mi) of the att_i
6          S_cp2 ← Split S by the median(S_1i,...,mi) of the att_i
7          if infoGain(S_cp1, att_i) ≥ infoGain(S_cp2, att_i) then
8              cp ← cp_1;
9          else
10             cp ← cp_2;
11     if infoGain(S_cp, att_i) ≥ infoGain(S_cp_opt, a_opt) then
12         a_opt ← att_i
13         cp_opt ← cp
14 return (a_opt and cp_opt as splittingCriterion)
```

4. Results assessment

We evaluate the performance of the SSL-VFC4.5 concerning its predictive accuracy and time spend to build the models comparing it with the following algorithms:

- **C4.5 algorithm:** Supervised algorithm proposed by Quinlan [1993] and available in Weka as the J48 implementation.

- **VFC4.5 algorithm:** Supervised algorithm based on C4.5, proposed by Cherfi et al. [2018], which speeds up the process of identifying the cut-off points of numerical attributes. It was implemented over the J48 algorithm available in Weka.
- **SSL-VFC4.5 algorithm:** Semi-supervised algorithm based on the VFC4.5 algorithm proposed in this work. As same as VFC4.5, it was also implemented over the J48 algorithm available in Weka.

These algorithms are compared using different datasets mainly obtained from the UCI repository [Lichman 2013]. Table 1 describes the characteristics of the seven used datasets: name, number of instances, number of discrete attributes, number of numeric attributes, number of different classes and if there are missing values for any attribute.

Table 1. Used datasets.

Dataset Name	Instances	Discrete	Numeric	Classes	Missing data
Abalone	4177	0	8	3	No
Adult	32561	9	5	2	Yes
Bank	45211	10	6	2	Yes
Banknote	1372	0	4	2	No
Biodegradation	1055	0	40	2	No
Eyestate	14980	0	14	2	No
Madelon	2000	0	500	2	No
Mushroom	8124	22	0	2	No

All the experiments were performed over Weka [Hall et al. 2009] framework for data mining. Weka provides a collection of evaluation tools and a great variety of known algorithms. The algorithms were evaluated using the ten-fold cross-validation method.

4.1. Methodology

To assess the behavior of each algorithm to deal with the presence of unlabeled instances, the class of most of the instances was removed during the training stage. The values of the rest of the attributes remained unchanged. Therefore, only a small part of the data remained classified. Therefore, the supervised assessed algorithms were trained only with the small portion of remained labeled instances, in contrast to the semi-supervised one, which used the unlabeled instances, too. However, in the testing stage, the original class of each instance was compared with the class predicted by the model to evaluate its accuracy.

To simulate different amount of labeled data, seven different datasets were created from each original ones. The difference between them is that they have subsets of 25, 50, 100, 200, 350, 500, and 1000 classified instances, respectively. The rest of the instances of each dataset had their class removed. The choice of the instances that maintained their class was made through a uniform random distribution. This approach was applied to evaluate the robustness of the algorithms to build models regardless of the number of labeled instances.

The experiments were carried out using the same standardazed configuration options for all the algorithms. Some of the configurations are: the value of 0.25 as confidence factor and the value of 0 as minimum limit to obtain information. Besides that, all the algorithms used the same pruning procedure presented in the original J48.

The proposed semi-supervised learning algorithm uses a parameter *w* that is adjusted in the training phase. Therefore, to achieve the best possible result for each test, it is necessary to find the best value of *w*. With this objective, 11 evaluations were carried out with different values of *w*, starting from 0.0 (not supervised), increasing by 0.1 in each execution, until reaching the value of 1.0 (totally supervised).

All algorithms were evaluated using two criteria. The first one was the accuracy analysis, which is the percentage of correctly classified instances. The second one was the runtime analysis, which is the total time spent to build the trees. The time consumed to find the best value of *w* was not considered. To carry out the analysis efficiently, *scripts* were developed in Python programming language to automate the execution of the tests through the command line interface provided by Weka.

4.2. Analysis of the results

This section shows the assessment results of the algorithms in each dataset. Section 4.2.1 presents the accuracy analysis while Section 4.2.2 presents the runtime analysis.

4.2.1. Accuracy analysis

Figure 1 presents the accuracy achieved by the algorithms in the presence of unlabeled instances. The proposed SSL-VFC4.5 algorithm achieved a slightly better average of accuracy, around 1%, for (a) Abalon, (c) Bank, and (h) Mushroom datasets. In contrast, it had slightly worst average accuracy, also near to 1%, in (b) Adult, (d) Banknote, (e) Biodegradation, (f) Eyestate, and (g) Madelon datasets. These results indeed do not represent a statistical significance difference when it was applied the Nemenyi statistical test.

However, when analyzing in detail each assessed dataset, it is possible to see that the proposed SSL-VFC4.5 has presented slightly better accuracy with few amount of labeled data, i.e. 25 and 50 labeled instances. This can be observed in the (b) Adult, (c) Bank, (g) Madelon, and (h) Mushroom datasets. On the other hand, it slightly decreased its accuracy when the number of labeled instances got increased. This can be seen in the (e) Biodegradation, (f) Eyestate and (g) Madelon.

Besides that, Figure 1 also shows that it obtained the best result in the (c) Bank dataset, with a stable behavior for every subset of labeled data and with an increment of accuracy with 1000 labeled instances. This behavior was similar in the other datasets with more discrete attributes than numerical ones, indicating that the proposed SSL-VFC4.5 provides better splits for discrete attributes. In contrast, it has presented its worst accuracy in the (g) Madelon dataset, especially when the number of labeled instances got increased. The (g) Madelon dataset has 500 numerical attributes, which represents a massive calculation of numerical cut-off points. Therefore, this indicates that the proposed SSL-VFC4.5 did not perform appropriately with a considerable amount of numerical attributes, caused by the variance calculation. However, this problem of degradation of accuracy did not happen when the dataset has less than a hundred numerical attributes, as seen in the (e) Biodegradation and the (f) Eyestate datasets. Therefore, this indicates that the proposed SSL-VFC4.5 has an equivalent accuracy performance compared with the others when the datasets do not present a huge amount of features.

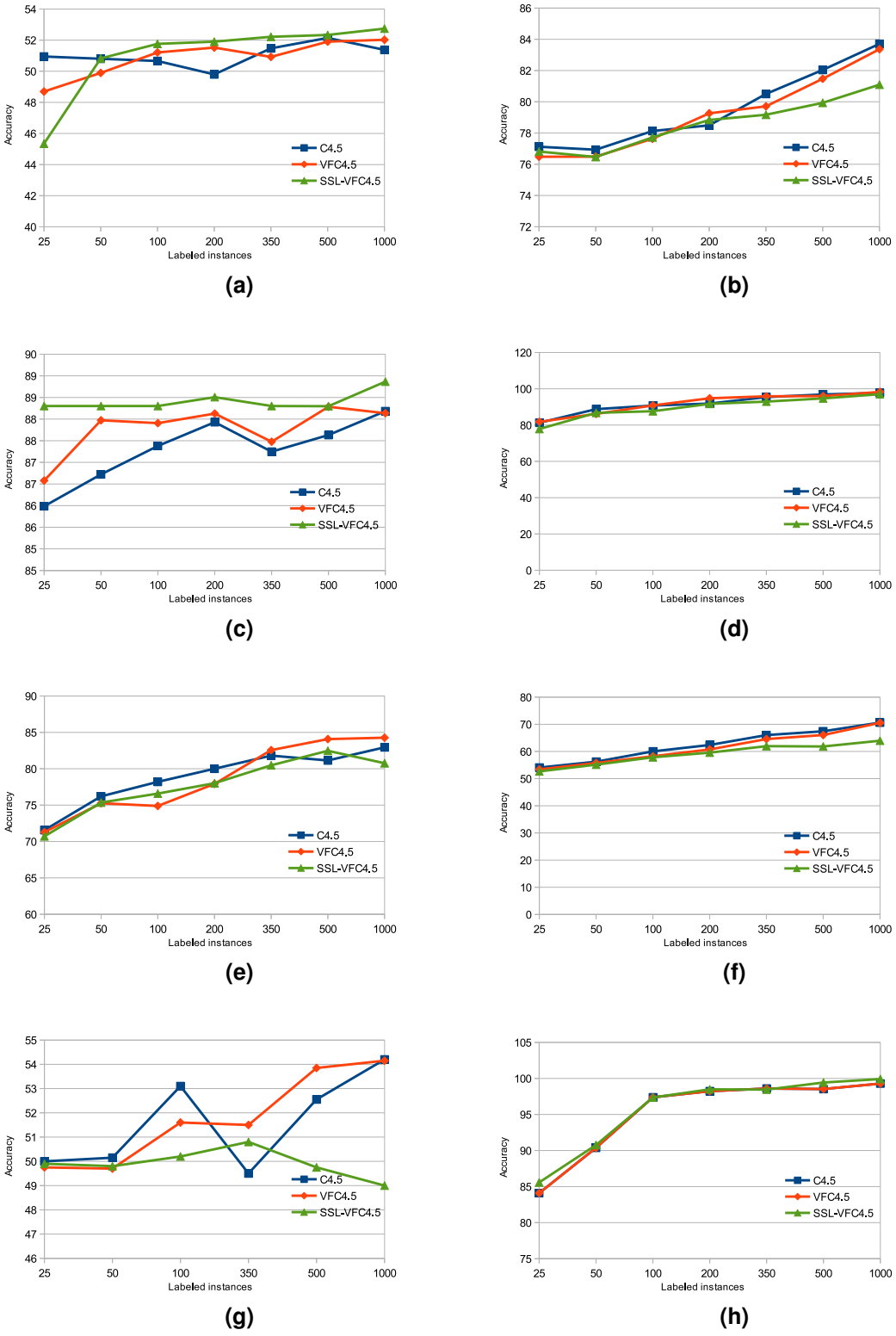

Figure 1. Results of the accuracy of the algorithms on the assessed datasets. (a) Abalone, (b) Adult, (c) Bank, (d) Banknote, (e) Biodegradation, (f) Eyestate, (g) Madelon, (h) Mushroom.

4.2.2. Runtime analysis

The second assessed criterion was the time spent to build the models. The results are presented in Figure 2 where the algorithms are compared according to the runtime (in seconds) to build the models in relation to the number of labeled instances processed. For the semi-supervised algorithm, the best value of *w* was selected in the assessments.

The supervised learning algorithms are faster than the proposed SSL-VFC4.5 algorithm, especially when the number of labeled instances increase. One reason for this result is that the SSL-VFC4.5 algorithm deals with all the instances (labeled or unlabeled), while the supervised algorithms only analyze the labeled ones. Besides that, the impurity metric applied in the SSL-VFC4.5 (see Equation 4) performs a nested-loop over all the attributes of the instances to assess the unsupervised part of the impurity metric. Therefore, this is a time-consuming task with $O(n^2)$ time complexity, which considerably increases the runtime of building the models.

When analyzing in detail each assessed dataset, it can be seen that the VFC4.5 algorithm did not present less execution time than the C4.5 algorithm as expected. Besides that, in some experiments, the VFC4.5 presented execution time equal to the SSL-VFC4.5, as observed in (d) Banknote with 50 labeled instances and (f) Eyestate with 50 labeled instances too. Also, it can be seen that the SSL-VFC4.5 had a quite worst execution time on the (e) Biodegradation, (g) Madelon and (h) Mushroom, when the number of labeled instances increase. However, even for this datasets the SSL-VFC4.5 presents a similar execution time for 25 and 50 labeled instances.

In summary, the SSL-VFC4.5 algorithm has a similar and compatible execution time compared to the other algorithms for few labeled instances, i.e., 25 to 100 instances, but in some cases, especially in datasets with many attributes, it gets worse execution time for datasets with more than 100 labeled instances.

4.3. Discussions about the experiments

Based on the results obtained in the assessments, it can be noted that regarding the average accuracy values the algorithms did not present a significant difference. For example, in datasets such as (d) Banknote and (h) Mushroom they achieved almost identical results. However, when the datasets have lower number of labeled instances, the SSL-VFC4.5 achieved better accuracy than the VFC4.5. Although, it is valid to highlight that for (f) Eyestate and (g) Madelon datasets, the SSL-VFC4.5 did not have suitable accuracy when the number of labeled instances increase. Nevertheless, in the (g) Madelon dataset, all the algorithms showed an irregular behavior depending on the number of labeled instances.

In terms of runtimes, the SSL-VFC4.5 algorithm achieved higher values than the supervised algorithms, especially in datasets with many attributes and instances. The main factor that increases the runtime of the semi-supervised algorithm is the need to calculate the impurity for all the attributes in the unsupervised part of the Equation 4, which add a $O(n^2)$ time complexity to the processing. Besides that, it needs to process all the unlabeled instances while the supervised algorithms ignore them.

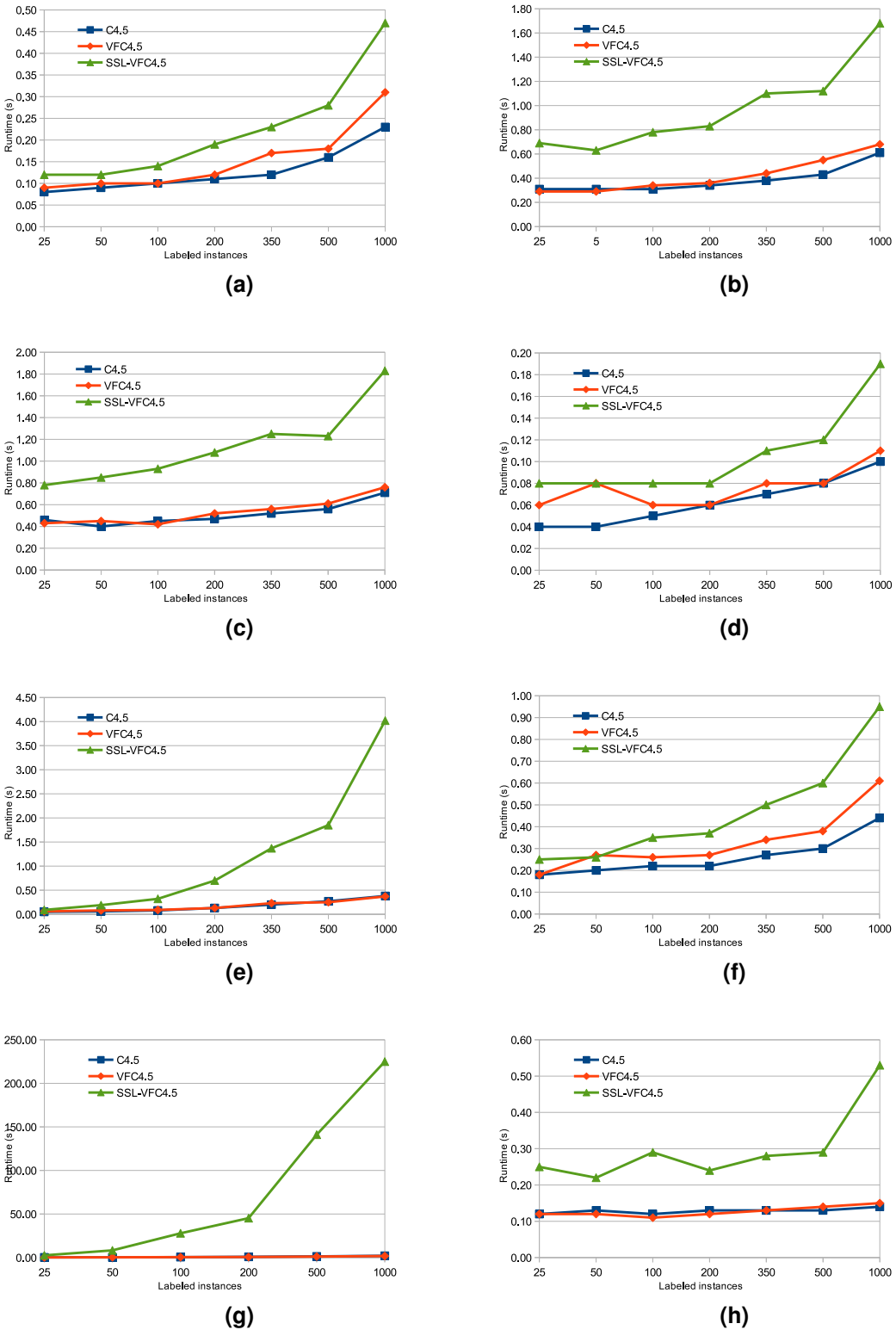

Figure 2. Results of the runtime of the algorithms on the assessed datasets. (a) Abalone, (b) Adult, (c) Bank, (d) Banknote, (e) Biodegradation, (f) Eyestate, (g) Madelon, (h) Mushroom.

5. Conclusion and future work

This paper presents an adaptation to the VFC4.5 algorithm to deal with semi-supervised learning, called SSL-VFC4.5. Therefore, this work proposed an algorithm that could perform suitably classifications in scenarios with few labeled instances, extracting relevant information from instances that are not labeled.

The original supervised entropy-based split metric used by VFC4.5 was substituted to the semi-supervised impurity-based metric proposed by [Levatić et al. 2017]. The proposed SSL-VFC4.5 was implemented over the version of the C4.5 available in Weka, named J48. Therefore, we first implemented the very fast extension of C4.5 over the J48, turning it a VFC4.5, and after we implemented the semi-supervised impurity-based metric over the already adapted VFC4.5.

Throughout the performed assessments, it was possible to test the proposed SSL-VFC4.5 in different scenarios and compare it the C4.5 and VFC4.5. As result, it was shown that the SSL-VFC4.5 achieved better accuracy than VFC4.5 in more the half of the test datasets with the presence of 25 and 50 labeled instances. However, no statistically significant differences among the achieved accuracies.

Nevertheless, considering the runtime assessment, it could be observed that the SSL-VFC4.5 can induce models as fast as the other algorithms with few labeled instances (25 and 50 instances). However, when the number of labeled instances increases, its execution time increases faster than the other assessed algorithms. Therefore, compared with C4.5 and VFC4.5, we can infer that the SSL-VFC4.5 algorithm can induce slightly more reliable models than VFC4.5, with suitable runtime, when the datasets have very few labeled instances.

Future work aims to implement a feature selection strategy to first select the more relevant attributes of the training dataset before performing the semi-supervised learning using the proposed SSL-VFC4.5. The feature selection aims to reduce the number of analyzed attributes in the unsupervised part of the impurity-based metric and thus reduce the time to build the models in the SSL-VFC4.5 algorithm.

References

Bifet, A., Zhang, J., Fan, W., He, C., Zhang, J., Qian, J., Holmes, G., and Pfahringer, B. (2017). Extremely fast decision tree mining for evolving data streams. In *Proceedings of the 23rd ACM SIGKDD*, New York. Association for Computing Machinery.

Chapelle, O., Scholkopf, B., and Zien, A. (2006). Semi-supervised learning. *MIT Press.*

Chen, K. and Wang, S. (2011). Semi-supervised learning via regularized boosting working on multiple semi-supervised assumptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):129–143.

Cherfi, A., Nouira, K., and Ferchichi, A. (2018). Very fast c4.5 decision tree algorithm. *Applied Artificial Intelligence*, 32(2):119 – 137.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. (2009). The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18.

Ip, R. H., Ang, L.-M., Seng, K. P., Broster, J., and Pratley, J. (2018). Big data and machine learning for crop protection. *Computers and Electronics in Agriculture*, 151:376–383.

Kim, K. (2016). A hybrid classification algorithm by subspace partitioning through semi-supervised decision tree. *Pattern Recognition*, 60:157 – 163.

Leistner, C., Saffari, A., Santner, J., and Bischof, H. (2009). Semi-supervised random forests. In *Proceedings of IEEE International Conference on Computer Vision*, pages 506–513. IEEE.

Levatić, J., Ceci, M., Kocev, D., and Džeroski, S. (2017). Semi-supervised classification trees. *Journal of Intelligent Information Systems*, 49(3):461–486.

Levatić, J., Kocev, D., Ceci, M., and Džeroski, S. (2018). Semi-supervised trees for multi-target regression. *Information Sciences*, 450:109 – 127.

Lichman, M. (2013). Uci machine learning repository. *<http://archive.ics.uci.edu/ml>*.

Ortiz-Díaz, A. A., Bayer, F. R., and Baldo, F. (2020). Ssl-c4. 5: Implementation of a classification algorithm for semi-supervised learning based on c4. 5. In *Brazilian Conference on Intelligent Systems*, pages 513–525. Springer.

Quinlan, R. (1993). C4.5: Programs for machine learning. *Morgan Kaufmann Publishers*.

Reinsel, D., Rydning, J., and Gantz, J. F. (2021). Worldwide global datasphere forecast, 2021–2025: The world keeps creating more data — now, what do we do with it all? *<https://www.idc.com/getdoc.jsp?containerId=US46410421>*.

Santos, A. and Canuto, A. (2014). Applying semi-supervised learning in hierarchical multi-label classification. *Expert Systems with Applications*, 41(14):6075 – 6085.

Settouti, N., El Habib Daho, M., Amine Lazouni, M. E., and Chikh, M. A. (2013). Random forest in semi-supervised learning (co-forest). In *2013 8th International Workshop on Systems, Signal Processing and their Applications (WoSSPA)*, pages 326–329.

Song, E., Huang, D., Ma, G., and Hung, C.-C. (2011). Semi-supervised multi-class adaboost by exploiting unlabeled data. *Expert Systems with Applications*, 38(6):6720 – 6726.

Tanha, J., van Someren, M., and Afsarmanesh, H. (2014). Boosting for multiclass semi-supervised learning. *Pattern Recognition Letters*, 37:63–77. Partially Supervised Learning for Pattern Recognition.

Tanha, J., van Someren, M., and Afsarmanesh, H. (2017). Semi-supervised self-training for decision tree classifiers. *International Journal of Machine Learning and Cybernetics*, 8(1):355–370.

Van Engelen, J. E. and Hoos, H. H. (2020). A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440.

Xu, W.-h., Qin, Z., and Chang, Y. (2011). Clustering feature decision trees for semi-supervised classification from high-speed data streams. *Journal of Zhejiang University SCIENCE C*, 12(8):615.