

Benchmark of Rule-based Classifiers in the News Recommendation Task

Tomáš Kliegr^{1,3} and Jaroslav Kuchar^{2,3}

¹ Multimedia and Vision Research Group
Queen Mary, University of London
t.kliegr@qmul.ac.uk
United Kingdom

² Web Engineering Group, Faculty of Information Technology,
Czech Technical University in Prague
Czech Republic

jaroslav.kuchar@fit.cvut.cz
³ Dep. of Information and Knowledge Engineering,
Faculty of Informatics and Statistics, University of Economics Prague
Czech Republic

Abstract. In this paper, we present experiments evaluating Association Rule Classification algorithms on on-line and off-line recommender tasks of the CLEF NewsReel 2014 Challenge. The second focus of the experimental evaluation is to investigate possible performance optimizations of the Classification Based on Associations algorithm. Our findings indicate that pruning steps in CBA reduce the number of association rules substantially while not affecting accuracy. Using only part of the data employed for the rule learning phase in the pruning phase may also reduce training time while not affecting accuracy significantly.

Keywords: recommender, association rules, rule learning, decision trees

1 Introduction

The large amount of content to choose from causes the *Information Overload* problem for visitors of news websites. Based on the analysis of past usage patterns, recommender systems can make a personalized list of preselected content, alleviating the users of the effort entailed in the process of choosing the content they should consume next and limiting the number of choices they need to make.

In this paper, we present experiments evaluating Association Rule Classification (ARC) algorithms on on-line and off-line recommender task of the CLEF NewsReel 2014 Challenge (further only Challenge). This research aims to investigate the execution time and accuracy of ARC algorithms on datasets with many target class values. For the on-line task with 100 millisecond response limitation, we received the best results with an association rule-based recommender, securing a 3rd place in the contest.

Obtaining promising results with a simple association rule learning approach deployed within our InBeat.eu open source recommender in the on-line task,

we hypothesize that Association Rule Classification (ARC) algorithms can yield improved results over direct application of association rules. Our benchmark also involves related symbolic machine learning algorithms – standard rule induction (FOIL) and decision tree induction (ID3).

The second focus of the experimental evaluation is to investigate possible performance optimizations of the Classification Based on Associations (CBA) ARC algorithm – through removal of its individual pruning steps or through the use of lower amount of data for pruning. There are practical problems with real time processing that are not encountered when there is “unlimited time” to provide the recommendation.

This paper is organized as follows. Section 2 presents the `InBeat.eu` recommender system in the on-line task. Section 3 briefly introduces the CBA ARC algorithm and presents the results on the off-line task. Finally, Section 4 summarizes the results and outlines future work.

2 On-line task: Setup and Results

This section gives a short introduction of the CLEF-NEWSREEL: News Recommendation Evaluation Lab⁴, which aimed at evaluating recommender systems on the task of recommending news articles on real websites. A major constraint imposed by the Challenge was a limitation on response time. Recommendations had to be provided in real-time (within 100ms). The main evaluation metric was the total number of successful recommendations, rather than the prediction accuracy (clickthrough rate).

Inputs: The main inputs are the users’ interactions and news item descriptions.

- $interaction(type, userId, itemId, context)$
where $type = \{impression|click\}$ and $context$ describes the features of the user (e.g. browser version, geolocation, etc.) and special features related to items and their presentation (e.g. keywords, position).
- $item(itemId, domain, description)$
where $domain$ is the identifier of items from the same group (e.g. news portal) and $description$ provides more detailed information about items (e.g. title, text, time of last update).

Outputs: Set of recommended items for the specific user who is reading the item within a given context.

- $(userId, itemId, context) \rightarrow \{item_x, item_y, \dots\}$

2.1 Algorithms

As the baseline, we used two simple algorithms *top interacted* and *most recent*, which we found as very effective for the given domain in the News Recommender

⁴ <http://www.clef-newsreel.org/>

Challenge'13 (our submission obtained a runner-up award).⁵ The main focus of our evaluation were association rules.

Top Interacted This algorithm is based on the daily popularity of news items. To avoid excessive effect of high short-time popularity of one item the interactions are aggregated on a daily basis. This approach addresses the evolution of popularity over time and decreases the influence of short-time peaks.

Most Recent The recency of an article plays an important role in the news domain. Our baseline recency-based algorithm uses a simple heuristic based on the newest news item within the same group as the group of the item the user is reading at the time of the request. The results is ordered list of items sorted by creation time.

Association Rules For each $interaction(type, userId, itemId, context)$ stored in our database, we prepared one entry in the training dataset as described in Table 1. Interactions are described only by the contextual features that are provided by the platform (e.g. Location, Browser, ...) and by an identifier of the item the user interacted with.

Table 1. Two instances from the CLEF#26875 offline dataset.

browser	isp	os	context			zip	class item
			geo	weekday	lang		
312613	281	431229	19051	26887	49021	62015	127563250
457399	45	952253	18851	26887	48985	65537	45360072

The training dataset was used to learn association rules. The contextual features could appear only in the rule body (antecedent) and the identifier of the item only on the right side of rule (consequent). We used the APRIORI algorithm [1] available within the *arules* package of R [6]. Example of a rule:

$$\text{isp} = \text{"281"} \wedge \text{os} = \text{"431229"} \rightarrow \text{item} = \text{"1124541"}$$

Additional mining setup is as follows. We used latest five thousand interactions as training dataset from our database. The APRIORI algorithm is run with minimum support of 0.1% (five interactions) and minimum confidence of 2%.

All discovered rules are imported into our simple rule engine. The engine finds all rules that match the contextual features of a recommendation request. The consequent of each matching rule represents a recommended item. The output is a list of unique item identifiers from the right side of the matching rules.

⁵ <https://sites.google.com/site/newsrec2013/challenge>

2.2 Performance

In this section, we present the performance of our InBeat recommender in the Challenge. The metric used in the Challenge to select the winning recommender systems was the *cumulative number of clicks* (number of successful recommendations) over the three different evaluation periods. The additional metrics provided by the organizers include *number of impressions* and *click-through rate*.

Sum of the number of impressions with the number of clicks can be interpreted as the ability of the systems to process large number of interaction observing the response time limitation.

Table 2. Leaderboard with cumulative number of clicks and average click-through rate per team in the Challenge - last evaluation period (2014-05-25 – 2014-05-31). Source: <http://orp.plista.com>

team	requests	clicks ↓	CTR
labor	285533	5614	1.97%
abc	206330	3653	1.77%
inbeat	268611	3451	1.28%
insight	508851	2012	0.4%
ba214	158593	1828	1.15%
uned	370510	1215	0.33%
riemannzeta	99920	1156	1.16%
plista GmbH	9112	137	1.5%

Table 2 presents the results for the last evaluation period. The table is sorted by the cumulative number of clicks. InBeat team is on the third position (total clicks) and on the fourth position with respect to the click through rate (CTR).

The CTR reported in Table 2 is the average for all algorithms. We also report the numbers for the individual algorithms:

- *Top Interacted*: 1.4% CTR,
- *Most Recent*: 0.8% CTR,
- *Association Rules*: 1.5% CTR.

The best CTR was obtained with a margin of 0.1% by *Association Rules*, which we explain by the fact that this algorithm takes into account both popularity (as reflected in the support score) and contextual features (the condition expressed by the antecedent of the rule). *Most Recent* is influenced only by temporal aspects and *Top Interacted* takes into account only the popularity.

3 Off-line task: Setup and Results

The objective of our experimental evaluation is to investigate the performance of Association Rule Classification (ARC) algorithms on the recommender problem cast as a standard classification task, and to compare the results with related mainstream classification algorithms.

3.1 Data and task

We used the data published within the off-line task of CLEF-NEWSREEL'14. The entire dataset consisted of 84 million records collected across multiple news portals [8]. We selected the website with the smallest amount of data (26,875 records) denoting the resulting dataset as CLEF#26875.

The dataset consists of instances described by a fixed number of attributes. In our evaluation we process the data with standard machine learning algorithms that require data in tabular form.

The task is to predict the class label (item viewed). The CLEF#26875 off-line dataset has 1,704 distinct items (target class values). This is an unusually high number in comparison with other datasets typically used for evaluation of machine learning algorithms, such as the most frequently cited datasets from the UCI repository.⁶ This distributional characteristic has an impact both on execution time and accuracy of the evaluated algorithms. The second notable feature of the dataset is that all its attributes are nominal. This is a favourable property for ARC algorithms in general, since they typically require that numerical attributes are discretized prior mining. The discretization algorithm and its parameters may have substantial impact on both accuracy and execution time.

The problem is cast as a standard machine learning classification task, where each row corresponds to a separate training instance. We also provide comparison with related mainstream machine learning algorithms that create rule or tree-based models (decision trees are convertible to rules).

3.2 Algorithms

The main focus of our evaluation is the Classification Based on Associations (CBA) ARC algorithm [10] and its two candidate successors – CMAR [9] and CPAR [16]. We compare the results with related symbolic machine learning algorithms, namely rule induction (FOIL, CPAR) and decision tree algorithms (ID3, CHAID).

The primary difference between ARC algorithms and rule induction is that the former class of algorithms first generates all association rules in the training data, and then performs pruning, while the rule learning algorithms add rules to the model one-by-one. The CPAR algorithm has some features of both ARC and rule induction algorithm, we list it under rule induction.

Association Rule Classifiers In 1998, Liu et al. introduced CBA, the first association rule classifier according to [15]. The first step in CBA is association rule learning with a modified APRIORI algorithm. The learning is constrained to produce rules that have an item corresponding to a class label value in the consequent.

In the second step, the resulting rules are subject to several pruning algorithms:

⁶ <https://archive.ics.uci.edu/ml/datasets.html>

1. Pessimistic pruning (optional). This pruning method attempts to simplify discovered rules by removing individual conditions from the rule antecedent. The rule is pruned if the pessimistic error rate [14] of the original rule is higher than that of the pruned rule.
2. Data coverage pruning⁷. This method removes rules preserving the following two conditions: i) each training case is covered by the rule with the highest precedence over other rules covering the case and ii) every rule in the classifier correctly classifies at least one training case.
3. Default rule pruning⁸. Rules pruned with data coverage pruning are ordered and all rules after the first rule with the lowest total error are replaced by a rule with empty antecedent predicting the majority class in the remaining data.

The gist of the CBA algorithm are the latter two pruning methods. The final ordered rule set is used as the classifier. Rules are sorted according to confidence, support and antecedent length. CBA performs single rule classification: for a given unlabeled instance, the first highest ranked rule whose antecedent matches the instance is selected, and its consequent is used to label the instance.

The CMAR algorithm is based on similar principles as CBA, but uses the newer FP-Growth [7] algorithm for association rule generation. In addition to data coverage pruning, CMAR performs also pruning based on chi-square test. The rule is pruned if the correlation between the rule's body and the rule's head is not statistically significant. The data coverage pruning in CMAR is slightly different from CBA as it requires at least δ rules to cover an instance before the instance is removed from training data (in CBA, $\delta = 1$).

In our benchmarks, we used the LUCS-KDD implementations of the ARC algorithms available from <http://cgi.csc.liv.ac.uk/~frans/KDD/Software/>. According to the implementations' author the software matches the description in the original papers introducing the respective algorithms, apart from that in the first rule generation step, the Apriori-TFP algorithm [2] is used instead of the modified APRIORI algorithm (CBA) or FP-GROWTH (CMAR).

It should be also noted that the LUCS-KDD implementation of CBA does not include pessimistic pruning. In evaluations on 20 UCI datasets reported in [10] CBA with pessimistic pruning had exactly the same accuracy as CBA without pessimistic pruning, but order of magnitude smaller number of rules in the classifier.

For part of the experiments with CBA, we used our own implementation of CBA. While this is not as efficient as the LUCS-KDD implementation, this allows us to test the effect of the individual pruning stages in CBA on accuracy and rule count of the resulting classifier. For rule generation phase, our implementation uses the APRIORI algorithm from the arules package followed by a filtering step which retains only rules that have one of the class labels in

⁷ We adopt the name for this method from [15].

⁸ This pruning type is omitted from the review [15], but we are of the opinion that "default rule pruning" could be perceived as a separate step from data coverage pruning.

the consequent. For the rule generation phase we implemented the version M1 of CBA [10]. The most simplified form of the classifier has a learning phase roughly corresponding to the execution of the APRIORI algorithm.

Rule learning (baseline) As a second set of baseline algorithms, we selected the First-Order Induction Learner (FOIL) [13] and the Classification based on Predictive Association Rules (CPAR) algorithm. It was shown that FOIL is prone to overfitting the training data as the size of the theory learned by FOIL can grow with the number of training examples [4]. For this reason, we tried to include Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [3] algorithm, which effectively addresses the overfitting problem [5]. We did not include RIPPER, because on the CLEF#26875 data the RapidMiner 5 implementation⁹ of the algorithm did not finish within a 12 hour time limit.

Finally, CPAR was designed to combine advantages of rule learning algorithms with association rule classifiers. The algorithm tests more rules than traditional rule-based classifiers which is claimed to ensure it does not miss important rules.

We used again the LUCS-KDD implementation of FOIL and CPAR.

Decision trees Decision tree induction algorithms produce models that to an extent resemble those produced by ARC algorithms. Each path from the root of the tree to the leaf in a decision tree corresponds to a classification rule.

Out of the multiple proposed decision tree algorithms, we included those implemented in the RapidMiner 5 open source data mining suite: ID3, RapidMiner's "DECISION TREE" and CHAID.

ID3 [12] is a frequently used baseline decision tree algorithm. Since all input attributes in CLEF#26875 are nominal, the algorithm can be used directly on input data without any preprocessing.

The RapidMiner's DECISION TREE operator was found to be the most accurate decision tree classifier in [11], which evaluated decision tree learning algorithms in three common data mining suites: SPSS-Clementine, RapidMiner and Weka. This implementation supports prepruning and postpruning methods.

The RapidMiner's CHAID implementation uses the chi-square test as a goodness criterion, otherwise it is the same as DECISION TREE.

3.3 Experimental evaluation

The algorithms described in the previous subsections were executed with parameters set according to Table 3.

The support and confidence parameters of CBA and CMAR had to be changed from the default values (of 20% and 80% respectively), since otherwise no rules were generated (no class item in the data had at least 20% support). The maximum number of frequent sets for CBA and CMAR was increased

⁹ <http://sourceforge.net/projects/rapidminer/>

Table 3. Algorithm parameters used in the off-line evaluation.

method	parameters
CBA	support = 2 records (0.008%), confidence = 2.0%, max size of antecedent = 6, max number of CARS = 80000, max number of frequent sets = 1,000,000
CMAR	support = 2 records (0.008%), confidence = 2.0%, max size of antecedent = 6, min cover (δ) = 1
CPAR	<i>default values</i> : K value = 5, min. best gain = 0.7, total weight factor = 0.05, decay factor = 1/3, gain similarity ratio = 0.99
DECISION TREE, CHAID	<i>default values</i> : criterion = gain ratio (Decision Tree), Chi-square test (CHAID), minimal size for split = 4, minimal leaf size = 2, minimal gain = 0.1, maximum depth = 20, confidence = 0.25, no prepruning, postpruning enabled
ID3	<i>default values</i> : criterion = gain ratio, minimal size for split = 4, minimal leaf size = 2, minimal gain = 0.1
FOIL	max number of attributes per rule = 6

to 1,000,000 since for support threshold lower than approximately 0.01%, the default limit of 500,000 prevented further improvements of the classifier. For DECISIONTREE, we initially obtained very low accuracy of 2%. This was caused by the prepruning step, which is enabled in RapidMiner by default. The resulting tree was composed of only one leaf class, which is the most frequent class label in the training data. The (post)pruning feature had a small but positive impact on accuracy and model size, therefore we left it enabled. For CPAR the default parameters produced acceptable results. Additional parameter tuning could have improved the performance of the algorithm.

The data were preprocessed to the form shown at Table 1 and randomly split to a training dataset (90%) and test dataset (10%). The experiments were run on Intel core i5 3320M CPU@2.6 GHz with 16 GB of RAM.

Table 4. Model benchmark on CLEF#26875 dataset (single 90/10 split). Model size refers to the number of rules for rule models and number of leaves for decision trees. Time is measured in seconds.

algorithm	time		accuracy	model size
	train	test		
DECISIONTREE	273	4	23.0	13496
ID3	290	4	22.8	13579
CHAID	284	3	25.4	13224
FOIL	815	1.5	24.7	18047
CPAR	87	1.23	4.6	18907
CBA	279	0.25	21.2	3681
CMAR	205	1.781	16.9	22516

The results depicted in Table 4 indicate that the overall best accuracy was obtained by the CHAID decision tree algorithm. CBA obtained accuracy close to the decision tree classifiers, however, with smaller training times and - for the on-line setting most significantly - shorter testing times. There are several factors contributing to the fast testing: a) the fact that CBA performs single rule classification, b) small number of rules in the classifier (compared to models created by other algorithms). The difference in test times between decision trees and the rule learning algorithms might be to a large extent caused by implementation-specific issues. Our impression is that additional optimization for the evaluation of the decision tree models could lead to substantially shorter test times.

Trading speed for accuracy Speed of training can be important in on-line recommender setting. Fast training also typically entails simpler models that are faster to apply. The accuracy/execution time balance can be controlled by the minimum leaf size and/or maximum depth parameters for decision trees and by the minimum support parameter for ARC classifiers.

Table 5. Effect of support threshold - CBA (ten-fold shuffled cross-validation). Time is measured in seconds.

metric	0.10%	0.09%	0.08%	0.07%	0.06%	0.05%	0.04%	0.03%	0.02%	0.01%
accuracy	6.68	6.88	7.07	7.64	8.1	8.65	9.48	10.4	13.47	17.55
train time	1.8	2.3	3	4.56	5.6	8.7	14.6	30.5	172	477
test time	0.02	0.03	0.03	0.04	0.03	0.04	0.05	0.05	0.1	0.19
rule count	148	178	193	228	270	317	452	576	1100	2303

Table 6. Effect of support threshold - CMAR (ten-fold shuffled cross-validation). Time is measured in seconds.

metric	0.10%	0.09%	0.08%	0.07%	0.06%	0.05%	0.04%	0.03%	0.02%	0.01%
accuracy	4.82	5.12	5.28	5.78	6.12	6.59	7.48	8	10.23	13.84
train time	0.744	0.89	1	1.39	1.75	2.13	3.83	6.5	36.34	178.92
test time	0.11	0.115	0.14	0.144	0.18	0.2042	0.32	0.46	1.05	2.26
rule count	834	999	1177	1557	1863	2251	3581	5116	11450	20561

Tables 5 and 6 show the impact of varying the support threshold on the accuracy and execution time of the CBA and CMAR classifiers. To obtain more reliable estimates especially at higher support thresholds, we performed ten-fold cross-validation. Table 7 shows the impact of minimum leaf size on the ID3 results.

The comparison between ID3 and CBA at 13% accuracy level shows that ID3 has much shorter training time (8.58s vs 172s), but it also produces more

Table 7. Effect of minimum leaf size - ID3 (ten-fold shuffled cross-validation, *based on one 90/10 split). Time is measured in seconds.

metric	100	90	80	70	60	50	40	30	20	10
accuracy	13.67	13.89	14.1	14.4	14.7	14.9	15.3	16.2	17	18.7
train time	8.58	8.58	9.04	9.57	10.57	12.17	13.93	18.09	25.4	80.66
test time	2.36	1.41	1.36	1.35	1.34	1.43	1.29	1.3	1.28	3.9
number of leaves*	3278	3362	3427	3522	3708	3959	4167	4817	5596	7389

complex models (3278 leaf nodes vs 1100 rules for CBA). The more compact model size contributes to fast test times for CBA.

Optimizing CBA In the field of decision tree induction, one of the mainstream pruning techniques is reduced error pruning, which uses different sets of data for learning the classifier and for pruning. Our experiments with CBA on CLEF#26875 showed that dividing available training data into a training set and a holdout set for pruning (validation data) does not have a positive effect on classifier accuracy. We tried multiple ratios of training set/holdout set size without obtaining a notable increase in accuracy.

An interesting finding follows from results presented in Table 8: if only part of the data used for the rule learning phase (i.e. APRIORI in CBA) is used for the pruning phase (i.e. data coverage and default pruning in CBA), the impact on accuracy is small. The training time can be reduced substantially as smaller amount of data is processed.

Table 8. Effect of pruning data set size. 100% of training data were used for rule generation, only x% used for pruning. For this experiment, we used our implementation of CBA M1.

metric	1%	2%	5%	10%	20%	30%	50%	75%
rule count	38	48	78	96	125	138	151	166
accuracy [%]	4.5	5.6	6.6	6.8	7.1	7	6.9	6.9

Table 9. Impact of pruning steps in CBA. Minimum support set to 0.1% and minimum confidence set to 2%.

algorithm	accuracy	rules
no pruning, direct use of association rules	6.4	1735
data coverage pruning	6.9	497
data coverage, default rule pruning	7	175

The results of the experiments with omission of individual pruning steps from CBA (Table 9) indicate that both data coverage pruning and default rule pruning not only reduce the size of the rule set, but also slightly improve the accuracy of the model. Interestingly, the absolute difference in accuracy between direct use of association rules (as in the on-line challenge) and CBA is very small. However, the order of magnitude decrease in the number of rules in the classifier justifies the use of CBA in on-line setting which puts emphasis on fast prediction times.

4 Conclusion and Future Work

This paper presented evaluation of multiple Association Rule Classification (ARC) algorithms in the CLEF NewsReel'14 challenge. The on-line track of the challenge required the competing systems to balance the architecture and technologies with the complexity of the involved algorithms. The practical experience that we obtained with our `InBeat.eu` recommender system underpin the choice of association rules as a fast on-line recommender algorithm. The experiments performed on the off-line dataset indicate that the CBA association rule classifier can further improve the results in terms of accuracy and especially speed, as it significantly reduces the size of the rule set. The best accuracy in our benchmark on the off-line dataset was obtained by the CHAID decision tree induction algorithm.

We further investigated the options for optimizing the pruning workflow in the CBA algorithm. The results indicate that the primary effect of the CBA pruning is the reduction of the number of rules in the model and that the impact on classifier accuracy is small. However, the potential saving in training time resulting from omission of these pruning steps might be offset by the increase of prediction time due to increased model size. Experiments showed that a viable direction of training time optimization might be using only part of the available training data for pruning. Further decrease in the number of rules could be attained by applying pessimistic pruning, an optional step in CBA, which was not covered in our evaluation.

Our benchmark on the off-line dataset was methodologically limited with respect to the typical setting for evaluation of recommender algorithms a) by ignoring the temporal dimension associated with the instances in the dataset and b) by providing results in terms of accuracy. Since recommender systems are frequently used as rankers other evaluation metric than accuracy could be more suitable. Future work could thus aim at addressing these limitations.

Acknowledgement. The authors would like to wish the anonymous reviewers for their helpful feedback. The participation in the CLEF recommender challenge was supported by the EC project FP7-287911 LinkedTV. The experimental evaluation of the CBA method was performed within grant IGA 20/2013. Tomáš Kliegr benefited in writing this paper from “long term institutional support for research activities” of the Faculty of Informatics and Statistics, UEP.

References

1. R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. *SIGMOD Rec.*, 22(2):207–216, Jun. 1993.
2. F. Coenen, P. Leng, and S. Ahmed. Data structure for association rule mining: T-trees and p-trees. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):774–778, June 2004.
3. W. W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning (ML'95)*, pp. 115–123. Morgan Kaufmann, Lake Tahoe, CA, 1995.
4. J. Fürnkranz. Fossil: A robust relational learner. In *Proceedings of the European Conference on Machine Learning on Machine Learning (ECML-94)*, pp. 122–137. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1994.
5. J. Fürnkranz, D. Gamberger, and N. Lavrač. *Foundations of Rule Learning*. Springer-Verlag, 2012.
6. M. Hahsler, B. Grün, and K. Hornik. arules - a computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, 14(15):1–25, 9 2005.
7. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, May 2000.
8. B. Kille, F. Hopfgartner, T. Brodt, and T. Heintz. The plista dataset. In *Proceedings of the International Workshop and Challenge on News Recommender Systems, NRS'13*, p. 14–22. ACM, 10 2013.
9. W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In N. Cercone, T. Y. Lin, and X. Wu, (eds.) *The 2001 IEEE International Conference on Data Mining (ICDM'01)*, pp. 369–376. IEEE Computer Society, 2001.
10. B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the 4th international conference on Knowledge Discovery and Data mining (KDD'98)*, pp. 80–86. AAAI Press, August 1998.
11. I. Moghimipour and M. Ebrahimpour. Comparing decision tree method over three data mining software. *International Journal of Statistics and Probability*, 3(3), 2014.
12. J. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
13. J. R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, Sep. 1990.
14. J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
15. K. Vanhoof and B. Depaire. Structure of association rule classifiers: a review. In *International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pp. 9–12. Nov 2010.
16. X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *Proceedings of the SIAM International Conference on Data Mining*, pp. 369–376. SIAM Press, San Francisco, 2003.