StuCoSReC

Proceedings
of the 2018
5th Student
Computer
Science
Research
Conference

StuCoSReC

Proceedings of the 2018 5th Student Computer Science Research Conference

Edited by

Iztok Fister jr. and Andrej Brodnik

Reviewers and Programme Committee

Klemen Berkovič • University of Maribor, Slovenia
Lucija Brezočnik • University of Maribor, Slovenia
Patricio Bulić • University of Ljubljana, Slovenia
Zoran Bosnić • University of Ljubljana, Slovenia
Borko Bošković • University of Maribor, Slovenia
Janez Brest • University of Maribor, Slovenia
Andrej Brodnik • University of Primorska and University
of Ljubljana, Slovenia
Haruna Chiroma • Gombe State University, Nigeria
Mojca Ciglarič • University of Ljubljana, Slovenia
Jani Dugonik • University of Maribor, Slovenia
Iztok Fister • University of Maribor, Slovenia
Iztok Fister Jr. • University of Maribor, Slovenia
Matjaž Gams • Jozef Stefan Institute, Slovenia
Tao Gao • North China Electric Power University, China
Andres Iglesias • Universidad de Cantabria, Spain
Branko Kavšek • University of Primorska, Slovenia
Miklos Kresz • University of Szeged, Hungary
Niko Lukač • University of Maribor, Slovenia
Uroš Mlakar • University of Maribor, Slovenia
Marjan Mernik • University of Maribor, Slovenia
Eneko Osaba • University of Deusto, Spain
Vili Podgorelec • University of Maribor, Slovenia
Peter Rogelj • University of Primorska, Slovenia
Xin-She Yang • Middlesex University, United Kingdom
Damjan Vavpotič • University of Ljubljana, Slovenia
Grega Vrbančič • University of Maribor, Slovenia
Aleš Zamuda • University of Maribor, Slovenia
Nikolaj Zimic • University of Ljubljana, Slovenia
Borut Žalik • University of Maribor, Slovenia

Published by

University of Primorska Press
Titov trg 4, SI-6000 Koper

Editor-in-Chief

Jonatan Vinkler

Managing Editor

Alen Ježovnik

Koper, 2018

ISBN 978-961-7055-26-9 (pdf)

www.hippocampus.si/ISBN/978-961-7055-26-9.pdf

ISBN 978-961-7055-27-6 (html)

www.hippocampus.si/ISBN/978-961-7055-27-6/index.html

DOI: <https://doi.org/10.26493/978-961-7055-26-9>

© University of Primorska Press

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in
univerzitetni knjižnici v Ljubljani

COBISS.SI-ID=296769024

ISBN 978-961-7055-26-9 (pdf)

ISBN 978-961-7055-27-6 (html)

Preface

It is no longer necessary to emphasize that computer science is omnipresent today. Terms as digitalization, Industry 4.0 etc. are frequently heard in the mass media and are becoming a part of our everyday life. The rapid advances in computer science require a large investment in research and development. In particular, it is important to educate young scientists, which is one of the objectives of the StuCoSReC conference.

In front of you is the proceedings of the fifth StuCoSReC conference. The StuCoSReC is intended for masters & PhD students to show their research achievements. Socializing is also important and the conference is an excellent opportunity for students to get to know each other. The conference connects students of computer science and goes beyond the invisible limits of faculties. The uniqueness of StuCoSReC conference is that it is organized each year by different higher education institution. The University of Ljubljana - Faculty of Computer and Information Science is proud to host the conference this year.

Twelve papers addressed this conference, covering several topics of the computer science. All the papers were reviewed by two international reviewers and accepted for the oral presentation.

ref. prof. dr. Nikolaj Zimic

Preface

II

Papers

<i>Regulacija plovca v zračnem tunelu (zračna levitacija) na osnovi mehke logike</i> • Primož Bencak, Dominik Sedonja, Andrej Picej, Alen Kolman, Matjaž Bogša	5–13
<i>Razpoznavanje literature v dokumentih in določanje njihovih sklicev v besedilu</i> • Matej Moravec, David Letonja, Jan Tovornik, Borko Boškovič	15–19
<i>Klasifikacija samomorilskih pisem</i> • Dejan Rupnik, Denis Ekart, Gregor Kovačevič, Dejan Orter, Alen Verk, Borko Boškovič	21–25
<i>Načrtovanje poprocesorja za pretvorbo NC/ISO G-kode v translacije za 3/5 osnega robota ACMA ABB XR701</i> • Gregor Germadnik, Timi Karner, Klemen Berkovič, Iztok Fister	27–33
<i>Using constrained exhaustive search vs. greedy heuristic search for classification rule learning</i> • Jamolbek Mattiev, Branko Kavšek	35–38
<i>Real-time visualization of 3D digital Earth</i> • Aljaž Jeromel	39–42
<i>Towards an enhanced inertial sensor-based step length estimation model</i> • Melanija Vežočnik, Matjaž Branko Jurič	43–46
<i>Breast Histopathology Image Clustering using Cuckoo Search Algorithm</i> • Krishna Gopal Dhal, Iztok Fister Jr., Arunita Das, Swarnajit Ray, Sanjoy Das	47–54
<i>Time series classification with Bag-Of-Words approach post-quantum cryptography</i> • Domen Kavran	55–59
<i>The problem of quantum computers in cryptography and post-quantum cryptography</i> • Dino Vlahek	61–64
<i>Named Entity Recognition and Classification using Artificial Neural Network</i> • Luka Bašek, Borko Boškovič	65–70
<i>Context-dependent chaining of heterogeneous data</i> • Rok Gomišček, Tomaž Curk	71

Using constrained exhaustive search vs. greedy heuristic search for classification rule learning

Jamolbek Mattiev
PhD student, teaching assistant
University of Primorska
Slovenia
(+386)-70366331
jamolbek.mattiev@famnit.upr.si

Branko Kavšek
Assistant Professor
University of Primorska, Jožef Stefan Institute
Slovenia
(+386)-56117654
branko.kavsek@upr.si

ABSTRACT

Existing classification rule learning algorithms use mainly greedy heuristic search to find regularities (e.g., a decision tree or a set of rules) in data for classification. In recent years, extensive research was done in the machine learning community on learning rules using exhaustive search – association rule mining. The objective there is to find all rules in data that satisfy the user-specified minimum support and minimum confidence constraints. Although the whole set of rules may not be used directly for accurate classification, effective and efficient classifiers have been built using these, so called, classification association rules.

In this paper we compare a “classical” classification rule learning algorithm that uses greedy heuristic search to produce the final classifier (a set of decision rules) with a class association rule learner that uses constrained exhaustive search to find classification rules on a “real life” dataset.

The results show that the “classical” classification rule learning algorithm generates a more compact classifier whose individual rules are somehow “less accurate”. On the other hand, the class association rule learner produces individual classification rules that are “highly accurate” but the overall classification accuracy of the classifier remains yet to be checked.

CCS Concepts

- Computing methodologies → Machine learning → Machine learning approaches → Rule learning
- Computing methodologies → Machine learning → Cross-validation
- Computing methodologies → Machine learning → Learning paradigms → Supervised learning → Supervised learning by classification

Keywords

Data mining, machine learning, classification rules, association rules, search, heuristics.

1. INTRODUCTION

Classification rule mining and association rule mining are two important data mining techniques. Classification rule mining aims at discovering a small set of rules in the database to form an accurate classifier. Association rule mining finds all rules in the database that satisfy some minimum support and minimum confidence constraints [2]. For association rule mining, the target of mining is not predetermined, while for classification rule mining there is one and only one pre-determined target, i.e., the class or dependent attribute.

Association rule mining is the discovery of what are commonly called association rules. Association rule mining, one of the most important and well researched techniques of data mining, was first

introduced in [1]. It studies the frequency of items occurring together in transactional databases, and based on a threshold called support, identifies the frequent itemsets. Another threshold, confidence, which is the conditional probability that an item appears in a transaction when another item appears, is used to pinpoint association rules.

In classification [4], by the help of the analysis of training data we develop a model which is then used to predict the class of objects whose class label is not known. The model is trained so that it can distinguish different classes in the data. The training data is having data objects whose class label is known in advance. Classification analysis is the organization of data in given classes. Also known as supervised classification, the classification uses given class labels to order the objects in the data collection. Classification approaches normally use a training set where all objects are already associated with known class labels.

There are many classification approaches such as statistical [8], divide-and-conquer [6] and covering [3] approaches. Based on these approaches numerous algorithms have been derived such as PART [5], Naive Bayes [8], See5 [12], C4.5 [11], Prism [3] and RIPPER [6]. However, traditional classification techniques often produce a small subset of rules, and therefore usually tend to miss detailed rules that might play an important role.

The aim of our research presented in this paper is to compare a “classical” state-of-the-art classification rule learning algorithm that uses greedy heuristic search (PART [5]) with a class association rule learner that uses constrained exhaustive search (a modified version of the APRIORI algorithm [2]).

2. PRELIMINARY CONCEPTS

Let D be a dataset with n attributes $\{A_1, A_2, \dots, A_n\}$ and $|D|$ records (objects) where each record has an object identifier (OID). Let $C = \{C_1, C_2, \dots, C_k\}$ be a list of class labels. A specific value of an attribute A_i and class C is denoted by lower-case letters a_{im} and c_j respectively.

Definition 1. An item is described as an attribute and a specific value for that attribute, denoted by $\langle(A_i, a_{im})\rangle$, e.g. $\langle(A_1, a_{11})\rangle$, $\langle(A_1, a_{12})\rangle$, $\langle(A_2, a_{21})\rangle$, etc.

Definition 2. An itemset is a set of items, e.g., $\langle(A_1, a_{11})\rangle, \langle(A_2, a_{21})\rangle, \langle(A_1, a_{11})\rangle, \langle(A_3, a_{31})\rangle$, etc.

Definition 3. A *Constraint_Itemsets* is a user-defined set of itemsets in which each itemset is said to be a constrained itemset, e.g., $Constraint_Itemsets = \{\langle(A_1, a_{11})\rangle, \langle(A_2, a_{21})\rangle, \langle(A_3, a_{31})\rangle\}$.

Definition 4. A class association rule R has the form $itemset \rightarrow c_j$ where $c_j \in C$ is a class label.

Definition 5. A rule R satisfies *Constraint_Itemsets* if its antecedent (itemset) contains at least one itemset in *Constraint_Itemsets*.

Definition 6. The actual occurrence $ActOcc(R)$ of a rule R in D is the number of records of D that match R 's antecedent.

Definition 7. The support of rule R , denoted by $Supp(R)$, is the number of records of D that match R 's antecedent and are labeled with R 's class.

Definition 8. The confidence of rule R , denoted by $Conf(R)$, is defined by equation (1) as follows:

$$Conf(R) = \frac{Supp(R)}{ActOcc(R)} \quad (1)$$

3. PROBLEM STATEMENT

Consider a relational data set D with n attributes. A record of D is a set of attribute-value pairs, denoted by T . A pattern is a subset of a record. We say, a pattern is a k -pattern if it contains k attribute-value pairs. All the records in D are categorized by a set of classes C .

It is required to generate the complete set of Class Association Rules by Apriori and PART algorithms. Another focus will be on comparing the advantages and disadvantages of using these two algorithms.

3.1 Apriori Algorithm

Association rule generation [14] is usually split up into two separate steps:

1. First, minimum support is applied to find all frequent itemsets in a database.
2. Second, these frequent itemsets and the minimum confidence constraint are used to form rules.

While the second step is straight forward, the first step needs more attention. Finding all frequent itemsets in a database is time consuming since it involves searching all possible itemsets (item combinations). The set of possible itemsets is the power set over I (the set of all items) and has size $2^n - 1$ (excluding the empty set which is not a valid itemset). Although the size of the powerset grows exponentially in the number of items n in I , efficient search is possible using the downward-closure property of support (also called anti-monotonicity) which guarantees that for a frequent itemset, all its subsets are also frequent and thus for an infrequent itemset, all its supersets must also be infrequent. Exploiting this property, efficient algorithms (e.g., Apriori [2]) can find all frequent itemsets.

Figure 1 gives the sketch of the Apriori algorithm. Apriori uses a "bottom up" approach, breadth-first search and a tree structure to count candidate item sets efficiently. The first pass of the algorithm simply counts item occurrences to determine the large 1-itemsets. A subsequent pass, say pass k , consists of two phases. First, the large itemsets L_{k-1} found in the $(k-1)$ -th pass are used to generate the candidate itemsets C_k . Next the database is scanned and the support of candidates in C_k is counted. For fast counting, we need to efficiently determine the candidates in C_k that are

contained in a given transaction t . All the missing details of the Apriori algorithm can be found in [1] and [2].

```

1)  $L_1 = \{\text{Large 1-itemsets}\};$ 
2) for ( $k=2$ ;  $L_{k-1} \neq \emptyset$ ;  $k++$ ) do begin
3)    $C_k = \text{apriori-gen}(L_{k-1});$  // New candidates
4)   forall transaction  $t \in D$  do begin
5)      $C_t = \text{subset}(C_k, t);$  // Candidates contained in  $t$ 
6)     forall candidates  $c \in C_t$  do
7)        $c.\text{count}++;$ 
8)   end
9)    $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$ 
10) end
11)  $\text{Answer} = \bigcup_k L_k;$ 

```

Figure 1: The Apriori algorithm

3.2 PART Algorithm

The PART rule learning algorithm [5] combines the two approaches C4.5 [11] and RIPPER [6] in an attempt to avoid their respective problems. Unlike both C4.5 and RIPPER it does not need to perform global optimization to produce accurate rule sets, and this added simplicity is its main advantage. It adopts the separate and conquer strategy in that it builds a rule, removes the instances it covers, and continues creating rules recursively for the remaining instances until none are left. It differs from the standard approach in the way that each rule is created. In essence, to make a single rule a pruned decision tree is built for the current set of instances, the leaf with the largest coverage is made into a rule, and the tree is discarded.

The prospect of repeatedly building decision trees only to discard most of them is not as bizarre as it first seems. Using a pruned tree to obtain a rule instead of building it incrementally by adding conjunctions one at a time avoids the over-pruning problem of the basic separate and conquer rule learner. Using the separate and conquer methodology in conjunction with decision trees adds flexibility and speed. It is indeed wasteful to build a full decision tree just to obtain a single rule, but the process can be accelerated significantly without sacrificing the above advantages.

The key idea is to build a "partial" decision tree instead of a fully explored one. A partial decision tree is an ordinary decision tree that contains branches to undefined subtrees. To generate such a tree, we integrate the construction and pruning operations in order to find a "stable" subtree that can be simplified no further. Once this subtree has been found tree-building ceases and a single rule is read off.

The tree-building algorithm is summarized as follows: it splits a set of examples recursively into a partial tree. The first step chooses a test and divides the examples into subsets accordingly. The PART implementation makes this choice in exactly the same way as C4.5. Then the subsets are expanded in order of their average entropy, starting with the smallest (the reason for this is that subsequent subsets will most likely not end up being expanded, and the subset with low average entropy is more likely to result in a small subtree and therefore produce a more general rule). This continues recursively until a subset is expanded into a leaf, and then continues further by backtracking.

But as soon as an internal node appears which has all its children expanded into leaves, pruning begins: the algorithm checks whether that node is better replaced by a single leaf. This is just the standard “subtree replacement” operation of decision-tree pruning, and the PART implementation makes the decision in exactly the same way as C4.5. If replacement is performed the algorithm backtracks in the standard way, exploring siblings of the newly-replaced node. However, if during backtracking a node is encountered whose every child is not a leaf (and this will happen as soon as a potential subtree replacement is not performed) then the remaining subsets are left unexplored and the corresponding subtrees are left undefined. Due to the recursive structure of the algorithm this event automatically terminates tree generation. Additional details about the PART algorithm can be found in [5].

4. EXPERIMENTAL EVALUATION AND RESULTS

In this paper, we used a real-life dataset [13] called “Laptop prices” to illustrate the class association and classification rules learning process. This dataset consists of 1.304 examples which have 11 “independent” attributes each and one class attribute. Three numeric attributes (Inches, Weight, and Price_euros) are discretized into nominal attributes (as illustrated in Table 1).

Table 1. Discretization of numeric attributes

Inches		Weight		Price_euros	
<i>Bins</i>	<i>Name</i>	<i>Bins</i>	<i>Name</i>	<i>Bins</i>	<i>Name</i>
[10.1 – 14]	Small	[0.69 – 1.75]	Light	[174 – 800]	Low
(14 – 16]	Standard	(1.75 – 2.21]	Medium	(800 – 1300]	Mid-range
(16 – 18.4]	Gaming	(2.21 – 4.7]	Heavy	(1300 – 6099]	High

In Table 2, the best class association rules which have at least 90% confidence are shown according to the Apriori algorithm. Table 3 shows the classification rules which are found by the PART algorithm (Conf. and Cov. In these tables stand for confidence or accuracy and coverage, respectively).

Results show individual class association rules are more accurate (confident) and have a higher coverage compared to individual classification rules. On the other hand, classification rules cover the search space “completely” and do not overlap while class association rules may (heavily) overlap and are not guaranteed to cover all the examples.

Table 2. Class Association Rules found by the Apriori algorithm

Class association rules	Conf.	Cov.
{ Inches=Standard, Cpu=Intel Core i3 } ==> { Price_euros=Low }	100%	114
{ ProductTypeName=Notebook, Inches=Standard, Cpu=Intel Core i3 } ==> { Price_euros=Low }	100%	110
{ Inches=Standard, Cpu=Intel Core i3, Ram=4GB } ==> { Price_euros=Low }	100%	89
{ ProductTypeName=Notebook, Inches=Standard, Cpu=Intel Core i3, Ram=4GB } ==> { Price_euros=Low }	100%	85
{ Inches=Standard, Cpu=Intel Core i3, OpSys=Windows 10 } ==> { Price_euros=Low }	100%	84
{ Inches=Standard, Cpu=Intel Core i3, Gpu=Intel } ==> { Price_euros=Low }	100%	82
{ Inches=Standard, Cpu=Intel Core i3, Memory=HDD } ==> { Price_euros=Low }	100%	76
{ ScreenResolution=1366x768, Cpu=Intel Celeron, Gpu=Intel } ==> { Price_euros=Low }	100%	67
{ Inches=Standard, Cpu=Intel Core i3, Ram=4GB, Weight=Medium } ==> { Price_euros=Low }	100%	67
{ ProductTypeName=Gaming, ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia } ==> { Price_euros=High }	94%	72
{ ProductTypeName=Gaming, ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia, OpSys=Windows 10 } ==> { Price_euros=High }	94%	72
{ Inches=Standard, Ram=4GB, Memory=HDD, Weight=Medium } ==> { Price_euros=Low }	94%	143
{ ProductTypeName=Notebook, Ram=4GB, Memory=HDD, Weight=Medium } ==> { Price_euros=Low }	94%	142
{ ProductTypeName=Gaming, ScreenResolution=Full HD, Cpu=Intel Core i7, Ram=16GB, Memory=SSD + HDD } ==> { Price_euros=High }	94%	71
{ ProductTypeName=Gaming, ScreenResolution=Full HD, Cpu=Intel Core i7, Ram=16GB, Memory=SSD + HDD, OpSys=Windows 10 } ==> { Price_euros=High }	94%	71
{ ScreenResolution=1366x768, Ram=4GB, Weight=Medium } ==> { Price_euros=Low }	93%	120
{ ProductTypeName=Notebook, ScreenResolution=1366x768, Ram=4GB, Weight=Medium } ==> { Price_euros=Low }	93%	112
{ Inches=Standard, ScreenResolution=1366x768, Ram=4GB, Weight=Medium } ==> { Price_euros=Low }	93%	120
{ ProductTypeName=Notebook, Inches=Standard, ScreenResolution=1366x768, Ram=4GB, Weight=Medium } ==> { Price_euros=Low }	93%	120
{ ProductTypeName=Gaming, ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD } ==> { Price_euros=High }	93%	75
{ ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia } ==> { Price_euros=High }	93%	75
{ ProductTypeName=Gaming, ScreenResolution=Full HD, Ram=16GB, Memory=SSD + HDD, OpSys=Windows 10 } ==> { Price_euros=High }	93%	75
{ ScreenResolution=Full HD, Cpu=Intel Core i7, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia } ==> { Price_euros=High }	93%	74
{ ProductTypeName=Gaming, Ram=16GB, Memory=SSD + HDD, Gpu=Nvidia, OpSys=Windows 10, Weight=Heavy } ==> { Price_euros=High }	93%	72

Table 3. Classification Rules found by the PART algorithm

Classification rules	Conf.	Cov.
{ Ram = 16GB, Memory = SSD, ScreenResolution = Full HD } ==> { Price_euros= High }	80%	36
{ Ram = 16GB, Memory = SSD } ==> { Price_euros= High }	94%	49
{ Ram = 16GB, Inches = Gaming } ==> { Price_euros= High }	94%	49
{ Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Core i3, ScreenResolution = 1366x768 } ==> { Price_euros=Low }	98%	52
{ Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Core i5, Memory = HDD } ==> {Price_euros=Low}	74%	48
{ Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Celeron } ==> {Price_euros=Low}	100%	48
{ Cpu = Intel Core i3, Memory = HDD } ==> { Price_euros=Low }	98%	47
{ Ram = 16GB } ==> { Price_euros= High }	70%	35
{ Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Core i5 } ==> { Price_euros= Low }	49%	17
{ Ram = 4GB, ProductTypeName = Notebook, Cpu = Intel Other } ==> { Price_euros= Low }	100%	30
{ Ram = 4GB, ProductTypeName = Notebook } ==> { Price_euros= Low }	97%	58
{ Cpu = Intel Core i7, ProductTypeName = Ultrabook, Inches = Standard } ==> { Price_euros= High }	70%	31
{ Cpu = Intel Core i7, ProductTypeName = Gaming, Inches = Standard } ==> { Price_euros= Mid-Range }	61%	33
{ Cpu = Intel Core i7, Inches = Standard, Memory = SSD } ==> { Price_euros= Mid-Range }	51%	33
{ Cpu = Intel Core i7, Inches = Small } ==> { Price_euros=High }	71%	41
{ Cpu = Intel Celeron } ==> { Price_euros= Low }	100%	40
{ ProductTypeName = Gaming, Cpu = Intel Core i7 } ==> { Price_euros= High }	76%	29
{ Memory = SSD + HDD } ==> { Price_euros= Mid-Range }	58%	42
{ Cpu = Intel Core i7 } ==> { Price_euros= Mid-Range }	49%	23
{ Cpu = Intel Core i5, Weight = Light, Company = Dell } ==> { Price_euros= Mid-Range }	49%	18
{ Cpu = Intel Core i5, Weight = Light, ProductTypeName = Ultrabook } ==> {Price_euros= Mid-Range }	50%	21
{ Cpu = Intel Core i5, Memory = SSD, Weight = Light } ==> {Price_euros= Mid-Range }	49%	30
{ Cpu = Intel Core i5, Gpu = Intel } ==> {Price_euros= Mid-Range }	48%	38

5. CONCLUSIONS AND FUTURE WORK

Overall, it can be observed from the experimental results (Apriori algorithm) that the features of the cheaper laptops (rules in Table 2 with the right-hand side *Price_euros=Low*) were inches: Standard or Small, CPU: Intel Core i3 or Celeron, RAM: 4GB, Weight: Light or Medium, Memory: HDD, GPU: Intel, Operative System: Windows 10 and Screen Resolution: 1366x768.

The expensive laptops (rules in Table 2 with the right-hand side *Price_euros=High*) were mostly Gaming laptops with SSD or SSD+HDD memory, 16GB RAM, Intel Core i7 CPU, Nvidia GPU, Heavy weight and Full HD Screen Resolution.

The results on the chosen dataset show that the Apriori algorithm generates more accurate individual classification rules with higher coverage/accuracy than the PART algorithm. However, class association rules may be highly overlapping which can affect the overall accuracy of the classifier.

In future work we shall check the overall accuracy and coverage of “complete” classifiers generated by class association rules algorithm, broaden our comparison to more (diverse) datasets and compare also the learning times of both algorithms.

6. REFERENCES

- [1] Agrawal, R., Amielinski, T., and Swami, A. (1993). “Mining association rule between sets of items in large databases”. In Proceeding of the ACM SIGMOD International Conference on Management of Data, pp. 207-216.
- [2] Agrawal, R., and Srikant, R. (1994). “Fast algorithms for mining association rules”. Research Report CA 95120, IBM Almaden Research Center, San Jose, California.
- [3] Cendrowska, J. (1987). “PRISM: An algorithm for inducing modular rules”. International Journal of Man Machine Studies. volume 27, no. 4, pp. 349-370.
- [4] Duda, R., and Hart, P. (1973). “Pattern Classification and Scene Analysis”. John Wiley & Sons.
- [5] Frank, E., and Witten, I. (1998). “Generating accurate rule sets without global optimization”. Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann, San Francisco, pp. 144-151.
- [6] Fürnkranz, J. (1996). “Separate-and-conquer rule learning”. Technical Report TR-96-25, Austrian Research Institute for Artificial Intelligence, Vienna.
- [7] Fürnkranz, J., and Widmer, G. (1994). “Incremental reduced error pruning”. Proceedings of the 11th Annual Conference on Machine Learning, Morgan Kaufmann, pp. 70-77.
- [8] John, G.H., and Langley, P. (1995). “Estimating Continuous Distributions in Bayesian Classifiers”. Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Mateo. pp. 338-345.
- [9] Liu, B., Hsu, W., and Ma, Y. (1998). “Integrating Classification and Association Rule Mining”. Singapore.
- [10] Patel, R., Vala, J., and Patel, K. (2014). “Comparative Study on Associative Classification techniques”.
- [11] Quinlan, J. R. (1993). “C4.5: Programs for Machine Learning”. San Mateo, Morgan Kaufmann.
- [12] Quinlan, J. R. (2008). “Data mining tools see5 and c5”.
- [13] <https://www.kaggle.com/ionaskel/laptop-prices/home> (accessed on 17.8.2018).
- [14] <http://software.ucv.ro/~cmihaescu/ro/teaching/AIR/docs/Lab8-Apriori> (accessed on 17.8.2018).

University of Primorska Press
www.hippocampus.si
ISBN 978-961-7055-26-9