KEM-DT: A Knowledge Engineering Methodology to Produce an Integrated Rules Set using Decision Tree Classifiers

Maqbool Ali
Dept. of Computer Science & Engineering, Kyung Hee University
Suwon, South Korea
maqbool.ali@oslab.khu.ac.kr

Sungyoung Lee*
Dept. of Computer Science & Engineering, Kyung Hee University
Suwon, South Korea
sylee@oslab.khu.ac.kr

Byeong Ho Kang
School of Engineering & ICT, University of Tasmania
Hobart, Australia
byeong.kang@utas.edu.au

ABSTRACT

In artificial intelligence, knowledge engineering is one of the key research areas in which knowledge-based systems are developed to solve the real-world problems and helps in decision making. For constructing a rule-based knowledge base, normally single decision tree classifier is used to produce *If-Then* rules (i.e. production rules). In the health-care domain, these machine generated rules are normally not well accepted by domain experts due to knowledge credibility issues. Keeping in view these facts, this paper proposes a knowledge engineering methodology called KEM-DT, which generates classification models of multiple decision trees, transforms them into production rules sets, and lastly, after rules verification and validation from an expert, integrates them to construct an integrated as well as a credible rule-based knowledge base. Finally, in order to realize the KEM-DT methodology, a *Data-Driven Knowledge Acquisition Tool* (DDKAT) is developed.

CCS CONCEPTS

• **Information systems → Information systems applications**; *Decision support systems*; *Data mining*; Expert systems; Data analytics;

KEYWORDS

Knowledge Engineering, Decision Tree, Classification Model, Model Translation, Production Rule

ACM Reference Format:
Maqbool Ali, Sungyoung Lee, and Byeong Ho Kang. 2018. KEM-DT: A Knowledge Engineering Methodology to Produce an Integrated Rules Set using Decision Tree Classifiers. In *Proceedings of The 12th International Conference on Ubiquitous Information Management and Communication (IMCOM '18).* ACM, New York, NY, USA, 5 pages. https://doi.org/https://doi.org/10.1145/3164541.3164640

*The corresponding author

IMCOM '18, January 5–7, 2018, Langkawi, Malaysia

ACM ISBN 978-1-4503-6385-3/18/01…$15.00
https://doi.org/https://doi.org/10.1145/3164541.3164640

1 INTRODUCTION

Knowledge engineering is one of the key research areas that build knowledge bases for solving the real-world problems and helps in decision making. During knowledge engineering process, most of the knowledge engineers and health-care experts are interested in knowledge representation [7] techniques such as decision trees, production rules, and decision graphs. These techniques help for discovering hidden knowledge as well as support for understanding the structure of knowledge. Among these techniques, production rules are most widely used for having features of (i) compactness, (ii) easy to understand, (iii) predictive nature, and (iv) credible for supporting decisions of recommendation [4, 5, 12]. Similarly, these rules are also more accurate than the decision tree from which they were generated [6, 12]. Due to a rapid increase in data size, it is difficult for an expert to extract production rules with naked eyes and to construct a huge knowledge base by utilizing personal expertise [16]. For constructing a knowledge base, knowledge is extracted from data and normally decision trees classifiers are used for this task [3]. However, the understanding of these decision trees can be difficult for a human expert [12].

In order to build more convenient and automatic knowledge-based system as well as to provide a compact, easy to understand, and predictive decision-support systems, there is a need to convert decision trees into production rules. This conversion task is considered as difficult task [15] and very limited number of systems exist who converts decision tree into rules [6, 13]. Most of the solutions are based on *WekatextToXml* application [9, 13], in which only J48 classifier-based decision trees are converted into inter-operable XML files. There is no evidence found to extract rule set from CART decision trees [9, 11]. Moreover, existing knowledge engineering methodologies are without considering multiple decision trees as well as without knowledge credibility as machine generated decision trees or production rules are not well accepted by health-care experts. Keeping in view these facts, this paper proposes a knowledge engineering methodology (KEM-DT) to construct a credible knowledge base. The KEM-DT methodology first builds multiple decision trees models from a single dataset and then, after performing text preprocessing transforms them into multiple production rules sets. After rules extraction, this methodology integrates all production rules sets to store them into an integrated rule-based knowledge base. Before storing into a knowledge base, an expert verifies and validates each new production rule for constructing a credible knowledge base. This study expands our previous work as mentioned in [1].

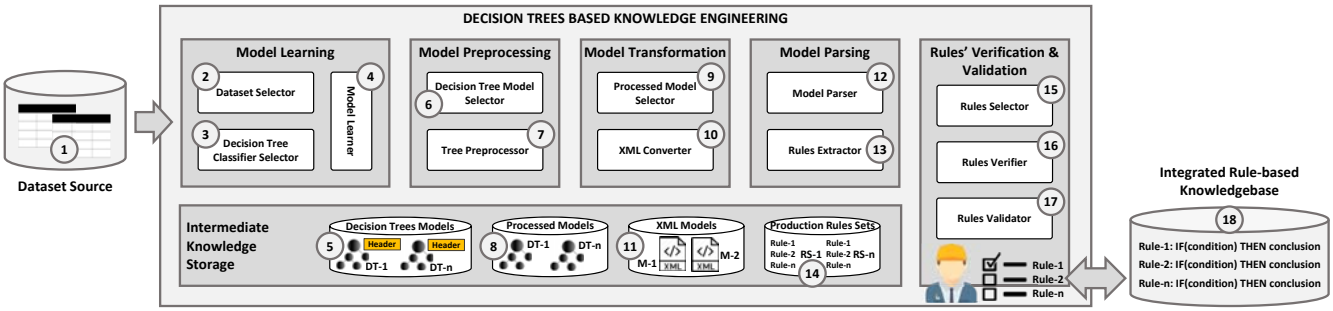

Figure 1: The proposed framework for KEM-DT methodology

For the realization of the KEM-DT methodology, we have designed and developed an easy-to-use *Data-Driven Knowledge Acquisition Tool* (DDKAT) to provide a data mining service addressed to expert and non-expert data miners. The DDKAT is a web-based application that acquires the health as well as wellness knowledge using *data-driven* approach and then shares the acquired knowledge in the form of production rules. This tool is designed for UCLab[1] project called *Mining Minds* (MMs); however, it can be utilized by other platforms as well. The MMs[2] provides benefits to users in the form of personalized life quality improving services.

The motivation behind the KEM-DT methodology is to construct a credible knowledge base using multiple decision trees. To achieve this goal, this study is undertaken with the following objectives: (1) To provide an automatic methodology for extracting production rules from multiple decision trees that was considered a difficult and time consuming task, (2) To generate acceptable production rules for health-care experts, and (3) To boost the development of machine generated knowledge-based systems.

The key contribution of this paper is to introduce an automatic methodology for extracting production rules from multiple decision trees to construct a credible knowledge base.

In order to explain the process of the KEM-DT methodology, we have simulated the DDKAT through a case study of a life-log dataset.

2 MATERIAL AND METHODS

To construct a credible knowledge base using multiple decision trees, this section describes the architecture of the KEM-DT methodology, characteristics of the selected dataset used for case study purposes, and the detailed procedure of the KEM-DT methodology.

2.1 Proposed system architecture

The functional architecture of the KEM-DT methodology is shown in Figure 1, which consists of six modules, namely *Model Learning*, *Model Preprocessing*, *Model Transformation*, *Model Parsing*, *Intermediate Knowledge Storage*, and *Rules' Verification & Validation*. There is only one user - domain expert interact with the verification & validation module. In Figure 1, the labels (1 to 18) represents the flow of the KEM-DT methodology.

[1]Ubiquitous Computing Lab., Kyung Hee University, Yongin http://uclab.khu.ac.kr
[2]http://www.miningminds.re.kr/english/

Table 1: MMDs characteristics

Parameter	Value
Title	Mining Minds Users Profile and Life-log Database
No. of Instances	408
No. of Attributes	15
Attributes' Names	Gender, Age, MaritalStatus, Height, Weight, BMI, RiskFactor, Disability, SituationCategory, Situation, HighLevelContext, Location, Recommendation (class attribute)
Attribute Type	nominal and numeric valued
Missing Attribute Values:	Yes
Class Distribution	Class values are activities names. Class Value Number of instances Sitting 140 Stretching 170 Walking 98

2.2 Case study dataset

DDKAT is simulated using the *Mining Minds Users Profile and Life-log Dataset* (MMDs). This dataset is prepared using capabilities of the *Mining Minds* framework, which consists of eleven real-world users' data. The goal for preparing MMDs is to predict generic activity based on parameters such as *gender*, *age*, *body mass index*, *risk factor*, *context location*, etc. The total number of instances of MMDs are 408. Moreover, this dataset has multi-class distributions and consists of nominal as well as numeric values. The characteristics of MMDs are illustrated in Table 1.

2.3 KEM-DT methodology

This section briefly describes the procedure for constructing a credible knowledge base from single dataset using multiple decision trees, which is represented in Figure 2.

In KEM-DT, five key steps are involved in constructing an integrated rule-based knowledge base after extracting production rules

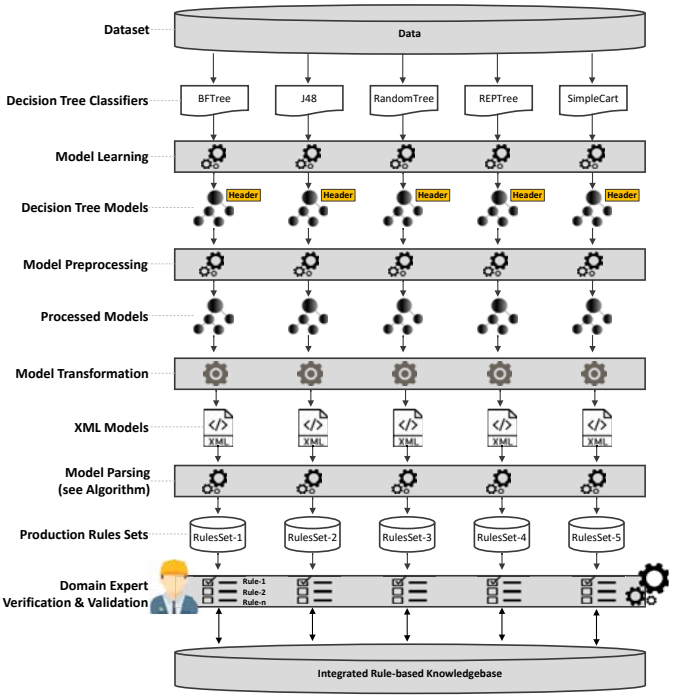

Figure 2: Workflow of the KEM-DT methodology

from a dataset as shown in Figure 2. Following is the description of each step involved in the KEM-DT methodology.

2.3.1 Step-1: Model learning. For model learning, dataset and machine learning classifier are required [2, 10]. For loading dataset, the *Data Selector* component selects dataset from an external *Dataset Source*. Similarly, for choosing classifier, the *Decision Tree Classifier Selector* component selects one of the classifier among five decision tree classifiers, namely *BFTree*, *J48*, *RandomTree*, *REPTree*, and *SimpleCart*. After getting dataset and classifier, the *Model Learner* component applies the 10-fold cross-validation technique [14] and builds the decision tree model. A partial view of decision tree model is shown in Figure 3, which includes the header information. This model is stored in the *Decision Trees Models* knowledge base.

2.3.2 Step-2: Model preprocessing. The models normally contain extra information such as term REPTree, size of the tree etc. in their headers as shown in Figure 3. To remove the header information, model preprocessing is required. For performing model preprocessing tasks, the *Decision Tree Model Selector* component first selects the model from the *Decision Trees Models* knowledge base and then the *Tree Preprocessor* component applies text trimming, text splitting, and special characters' replacements techniques. After model preprocessing, the model is stored in the *Processed Models* knowledge base.

2.3.3 Step-3: Model transformation. For extracting production rules from decision trees in more simple, interoperable, and standardize manners, it is necessary to transform the processed model into XML format. For XML conversion, the *Processed Model Selector* component first selects the model from the *Processed Models*

```
REPTree
==========

SituationCategory = None : Sitting (26/6) [11/1]
SituationCategory = Sitting
|   RiskFactor = Back Pain : Stretching (51/0) [31/0]
|   RiskFactor = Normal
|   |   Height < 168 : Stretching (28/0) [10/0]
|   |   Height >= 168
|   |   |   Age < 32.5 : Stretching (17/7) [6/1]
|   |   |   Age >= 32.5 : Walking (15/0) [13/0]
|   RiskFactor = Hypertension : Stretching (22/0) [10/0]
|   RiskFactor = None : Walking (15/0) [7/0]
SituationCategory = Standing : Sitting (72/0) [37/0]
SituationCategory = LyingDown : Walking (25/0) [11/0]

Size of the tree : 13
```

Figure 3: A view of decision tree model along-with header information

knowledge base and then the *XML Converter* component performs operators configuration, model indentation, and file-conversion tasks [9]. After model transformation, model is stored in the *XML Models* knowledge base. A partial view of XML model is shown in Figure 4.

```
<?xml version="1.0" encoding="UTF-8"?>
<DecisionTree type="treemodelFile">
        <Test attribute="SituationCategory" operator="= None" value="">
                <Output decision="Sitting " info=""/>
        </Test>
        <Test attribute="SituationCategory" operator="=" value="Sitting">
                <Test attribute="RiskFactor" operator="= Back Pain" value="">
                        <Output decision="Stretching " info=""/>
                </Test>
                .....
                .....
                <Test attribute="RiskFactor" operator="= None" value="">
                        <Output decision="Walking " info=""/>
                </Test>
        </Test>
        <Test attribute="SituationCategory" operator="= Standing" value="">
                <Output decision="Sitting " info=""/>
        </Test>
</DecisionTree>
```

Figure 4: Partial view of XML model

2.3.4 Step-4: Model parsing. For reading an XML model and extracting the inside information to construct the production rules, a model parsing is required. For performing these tasks, the *Model Parser* component first loads the model from the *XML Models* knowledge base and then parses the model using the *Document Object Model* (DOM) parser. The DOM parser converts the XML file into object tree and provides help to access the contents randomly. To construct production rules, the *Rules Extractor* component invokes model parsing algorithm, which is shown in Figure 5.

Figure 5 outlines the algorithm to extract the rules set by looking inside nodes' information such as relationships among root, child, sibling, and parent nodes. This algorithm adapts tree traversing

Algorithm 1 Model Parsing

Input: Decision Tree Model *DTM*
Globals: Root Node *RN*, Current Node *CN*, Leaf Node *LN*, Path Node *PN*, Rule ID *RID*, Rule Condition *RCd*, Rule Conditions List *RCdL*, Rule Conclusion *RCc*, Rules List *RL*
Output: Rules Set *RS*

```
RN ← RootNode(DTM)
CN ← FirstChild(RN)
for each Node {N_1, ..., N_n} in DTM
    if (ChildNode(CN) = true)
        CN ← FirstChild(CN)
    else
        if (CN = LN)
            RID ← RID + 1
            RCc ← NodeValue(CN)
            PN ← CN
            for each PathNode {PN_1, ..., PN_n} in each path until get RN
                RCd ← NodeValue(PN)
                RCdL ← RCd
                PN ← ParentNode(PN)
            RL ← RID ∪ RCdL ∪ RCc
        CN ← ParentNode(CN)
        CN ← NextSibling(CN)
RS ← RL
```

Figure 5: Algorithm for model parsing

mechanism to extract all conditions along-with conclusion from each path of the object tree. Each condition or conclusion consists of attribute(s), operator(s), and their value(s). The rules set extracted from this algorithm is stored into the *Production Rules Sets* knowledge base. A view of production rules sets obtained using REPTree and J48 decision trees are shown in Figure 6. A production rule is a form of knowledge representation, which is inherently subjective and can not be considered quantitatively [7].

2.3.5 Step-5: Rules verification & validation. The last step of this study is the credibility of the production rules set. For achieving this target, production rules are shown to domain expert through an expert interface, where expert first selects the rules from the *Production Rules Sets* knowledge base using the *Rules Selector* component and then by utilizing his/her expertise [8], verifies each generated rule using the *Rules Verifier* component and stores them into the *Integrated Rule-based Knowledge base*. Finally, before storing a rule, the *Rules Validator* component validates each verified rule with existing production rules that are stored into the *Integrated Rule-based Knowledge base*. In case of rule matching, it guides the expert about rule duplication to avoid repetition. Table 2 illustrates the total number of production rules generated and the number of matched rules against each decision tree classifier. For example, in Figure 6, total 9 production rules are generated using J48 classifier, where 3 rules (i.e. 1, 8 & 9) are similar to the REPTree classifier generated rules. In that case, only J48 classifier generated rules will be stored first into the *Integrated Rule-based Knowledge base* after an expert verification, while REPTree classifier generated rules will be considered as duplicate rules after validation process and will not be stored into a knowledge base. As a summary, we extracted total 100 production rules from the *Mining Minds Users Profile and Life-log Dataset* (MMDs) using five decision tree classifiers, where 5 rules are duplicated and 95 rules are verified by an expert.

Table 2: Rules analysis for an expert verification

Classifier	No. of Production Rules	No. of Matched Rules
BFTree	11	2 with SimpleCart
J48	9	3 with REPTree
RandomTree	66	0
REPTree	9	3 with J48
SimpleCart	5	2 with BFTree

3 CONCLUSIONS

As the final goal of every knowledge-based system is to solve the real-world problems and helps in decision making; therefore, credible and huge knowledge can play an important role in the healthcare domain. To achieve this goal, we propose a knowledge engineering methodology (KEM-DT) that constructs an integrated rule-based knowledge base. For constructing a credible knowledge base, this study extracts production rules from multiple decision trees. Lastly, in order to realize the KEM-DT methodology, a *Data-Driven Knowledge Acquisition Tool* (DDKAT) is developed.

4 ACKNOWLEDGEMENTS

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-0-01629) supervised by the IITP (Institute for Information & communications Technology Promotion). This work was supported by the Industrial Core Technology Development Program (10049079, Develop of mining core technology exploiting personal big data) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea). This work was supported by the Korea Research Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2016K1A3A7A03951968).

REFERENCES

[1] M. Ali, M. Hussain, S. Lee, and B. H. Kang. Sakem: A semi-automatic knowledge engineering methodology for building rule-based knowledgebase. In *International Symposium on Perception, Action, and Cognitive Systems (PACS2016)*, pages 63–64, 2016.
[2] R. Ali, S. Lee, and T. C. Chung. Accurate multi-criteria decision making methodology for recommending machine learning algorithm. *Expert Systems with Applications*, 71:257–278, 2017.
[3] J. L. Ardoint, P. Bonnard, and H. Citeau. Composite production rules, Jan. 27 2015. US Patent 8,943,003.
[4] N. Caetano, P. Cortez, and R. M. Laureano. Using data mining for prediction of hospital length of stay: An application of the crisp-dm methodology. In *International Conference on Enterprise Information Systems*, pages 149–166. Springer, 2014.
[5] X. Guo and Y. Li. Research on knowledge representation in expert system. In *International Conference on Education Technology, Management and Humanities Science (ETMHS 2015)*, pages 873–876, 2015.
[6] G. Holmes, M. Hall, and E. Prank. Generating rule sets from model trees. *Advanced Topics in Artificial Intelligence*, pages 1–12, 1999.
[7] M. Humphrey, S. J. Cunningham, and I. H. Witten. Knowledge visualization techniques for machine learning. *Intelligent Data Analysis*, 2(1-4):333–347, 1998.
[8] M. Hussain, M. Afzal, T. Ali, R. Ali, W. A. Khan, A. Jamshed, S. Lee, B. H. Kang, and K. Latif. Data-driven knowledge acquisition, validation, and transformation into hl7 arden syntax. *Artificial intelligence in medicine*, 2015.
[9] S. Luc. Wekatexttoxml: Convert weka decision trees into interoperable xml files, 2012. http://www.lucsorel.com/media/downloads/WekatextToXml.jar, 2012. Accessed: 2017-08-19.
[10] J. Mesarić and D. Šebalj. Decision trees for predicting the academic success of students. *Croatian Operational Research Review*, 7(2):367–388, 2016.

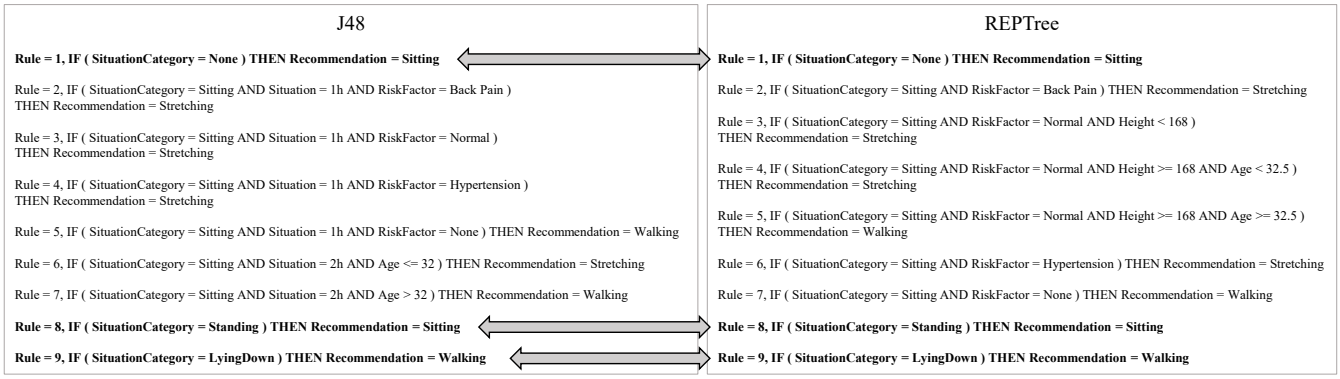

Figure 6: Production rules sets generated from J48 and REPTree decision tree classifiers

[11] M. Omar and S.-L. Syed-Abdullah. Finding the effectiveness of software team members using decision tree. In *Pattern Analysis, Intelligent Security and the Internet of Things*, pages 107–115. Springer, 2015.

[12] J. R. Quinlan. Generating production rules from decision trees. In *ijcai*, volume 87, pages 304–307, 1987.

[13] D. Trevisani and L. Cecchi. Micromanagement basado en formaciones de grupo implementado con scripting dinámico—-micromanagement group formations based on dynamic scripting implemented. In *XX Argentine Congress of Computer Science (Buenos Aires)*, 2014.

[14] G. Williams. Cross validation, data mining, desktop survival guide, 2010. https://www.togaware.com/datamining/survivor/Cross_Validation.html, 2010. Accessed: 2017-08-19.

[15] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[16] M. Zorrilla and D. García-Saiz. A service oriented architecture to provide data mining services for non-expert data miners. *Decision Support Systems*, 55(1):399–411, 2013.