From Machine Learning to Knowledge-Based Decision Support—A Predictive-Model-Markup-Language-to-Arden-Syntax Transformer for Decision Trees

Julia ZECKL[a], Matthias WASTIAN[b], Dominik BRUNMEIR[b], Andrea RAPPELSBERGER[c], Sergei B. ARSENIEV[d], Klaus-Peter ADLASSNIG[a,c]

[a]Medexter Healthcare GmbH, Borschkegasse 7/5, 1090 Vienna, Austria

[b]dwh GmbH, Neustiftgasse 57–59, 1070 Vienna, Austria

[c]Section for Artificial Intelligence and Decision Support, Medical University of Vienna, Spitalgasse 23, 1090 Vienna, Austria

[d]Clinical and Research Institute for Emergency Pediatric Surgery and Trauma 119180, Moscow, Bolshaya Polyanka 22, Russian Federation

Abstract Arden Syntax is an HL7 International standard for the representation and execution of clinical knowledge in knowledge-based clinical decision support (CDS) systems. The predictive model markup language (PMML) specifies a file format for the representation and exchange of statistical and data mining models. To use those machine-learned models in Arden-Syntax-based CDS systems, the PMML files have to be transformed into an Arden Syntax representation. A PMML-to-Arden-Syntax transformer was created to process PMML structures and generate the Arden Syntax code. It employs the extensible stylesheet language transformation to create Arden Syntax medical logic modules (MLMs) out of PMML files. The transformer may create multiple MLM files from a single PMML. Currently the transformer is able to transform decisions tree models only. Its transforming capabilities may be extended to additional models available in PMML format in the future. This approach generated a new way of creating MLMs based on machine learning results, in addition to the traditional method of knowledge design with clinical experts.

Introduction

Medical decisions are usually based on a combination of clinical knowledge and experience, results of medical research, and personal judgement [1, 2]. The increasing numbers of solved clinical cases will facilitate medical decisions and probably also render them more reliable. Clinical decision-making is becoming increasingly complex because of the consistently growing quantity of data that needs to be taken into account. Moreover, time slots for making decisions are becoming tighter. Thus, clinical decision-making can and will be assisted by clinical decision support (CDS) systems [3-6].

Arden Syntax [7-9] is a medical knowledge representation and processing standard for CDS system building offered by Health Level Seven International [10]. It defines the way clinical and scientific knowledge can be represented, computerized, and processed. Arden Syntax was first approved by the American Society for Testing and Materials in 1992 [8]. Several extensions have followed since. The current version (v2.10) was released in November 2014 [11].

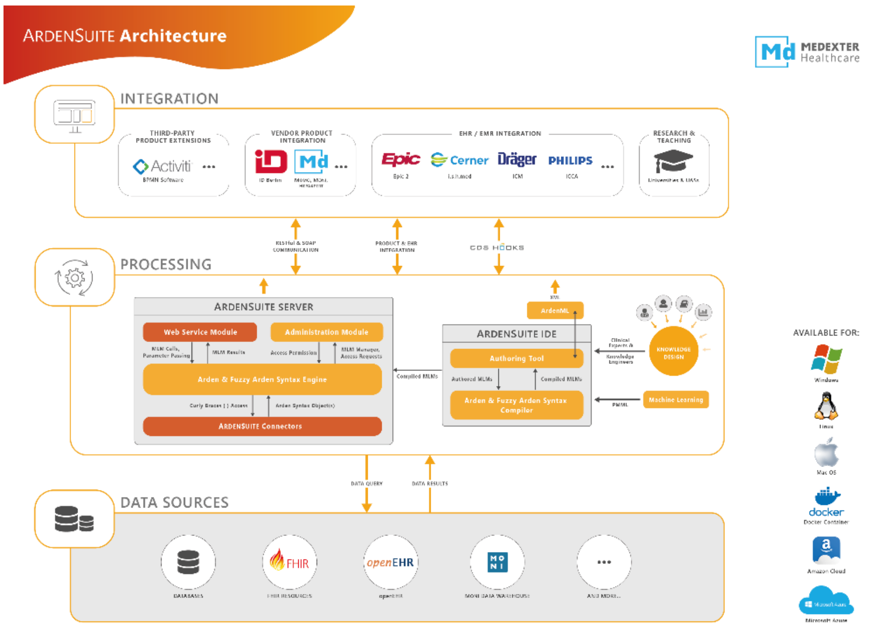


Fig. 1. The ArdenSuite's two main components IDE and server, as well as its interoperability components: the SOAP and REST webservice APIs, the database, FHIR, and OpenEHR connectors, the ArdenML and PMML transformers, the CDS hooks API, and the Activiti workflow extension.

In Arden Syntax, decision support and logic are contained and separated from each other in so-called medical logic modules (MLMs). Each of these MLMs contains enough clinical knowledge for at least one medical decision. MLMs can receive input data via a direct argument, with its call from other MLMs, or via so-called read/curly brace expressions. The latter may be defined as expressions to access external data sources such as an SQL database or an FHIR resource. MLMs are split into four sections: maintenance, library, knowledge, and resources. The maintenance and library sections contain information about the MLM itself and some medical background data. The knowledge section contains the medical logic, and the resources section comprises localized messages in different languages if applicable.

The ArdenSuite software [12, 13] is a commercial CDS authoring and processing platform based on Arden Syntax. It was developed by Medexter Healthcare, Vienna, Austria. As seen in the middle part of Figure 1 (processing), it consists of two main components: the ArdenSuite integrated development environment (IDE) to create, compile, and test MLMs; and the ArdenSuite server that grants external client applications access to uploaded MLMs via a REST or SOAP interface.

The top portion of Figure 1 (integration) shows external software that may be integrated with or into the ArdenSuite. The lower part of Figure 1 (data sources) shows the different types of data sources the ArdenSuite is able to communicate with.

The predictive model markup language (PMML) [14] is an XML-based specification for the representation of statistical and data mining models. It was developed and is continuously supported by the Data Mining Group [15]. Its latest version (v4.3) was released in August 2016 [16]. The terms "predictive analytic models" and "data mining models" refer to mathematical models that use statistics, clustering, and similar approaches to learn patterns from given data with a known outcome. The learned models use the acquired knowledge to predict patterns in new data. The basic purpose of PMML is to create a vendor-independent method of exchanging models between applications. This approach is adapted by over 20 vendors, such as IBM, KNIME, Phyton, LigaData, R, Java, and JavaScript [15]. As shown in Figure 2, PMML files may also include data pre-processing and post-processing methods and parameters, in addition to the models themselves.

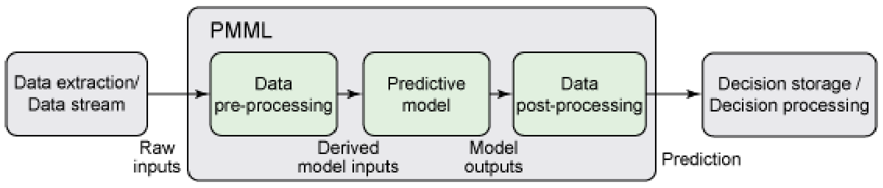


Fig. 2. PMML content (Source: [14])

The following models are currently supported by the PMML specification [16]:

- Association rules
- Baseline models
- Bayesian network
- Cluster models
- Gaussian process
- General regression
- K-nearest neighbors
- Naïve Bayes

- Neural network
- Regression
- Rule set
- Scorecard
- Sequences
- Text models
- Time series
- Trees
- Vector machine

A single PMML file can contain several models of the same or different type; a PMML file may even contain no model at all. In the latter case, it is used to carry metadata alone. Each model in a PMML file may have the attribute “isScorable”, which indicates that the model is valid for processing. If this attribute is set to “false”, the model is meant to be purely informative and was not meant to be applied to any data.

Decision trees [17] have been developed for statistical and machine learning purposes, and are successfully applied to various medical tasks [18]. They are extensively used for the classification of data. A decision tree has a “tree-like” structure with a single root node at the top. Each node may have any number of child nodes, which in turn may have any number of further child nodes. Nodes without child nodes are known as leaf nodes. Each node defines a condition used to select a child node to be checked next. The leaf nodes define the outcome of the tree evaluation. The first regression tree algorithm was published over fifty years ago [19]. Classification and regression trees (CART) is an algorithm first described by Breiman et al. [20] in 1984. It introduced some novel improvements, including the ability to control the size of the tree being generated.

The goal of this project was to create a PMML-to-Arden-Syntax transformer that is able to produce well formatted Arden Syntax MLMs out of PMML files. A bridge between the Arden Syntax standard to develop knowledge-based decision systems and machine learning results that autonomously learn from data has thus been created. The PMML-to-Arden-Syntax transformer has been implemented as an add-on of the ArdenSuite IDE.

The completed PMML-to-Arden-Syntax transformer will be useful for development teams that wish to integrate their machine learning results into the established ArdenSuite CDS platform environment of a hospital. The results of machine learning can be directly converted into Arden Syntax MLMs and processed within this environment.

The present PMML-to-Arden-Syntax transformer generates Arden Syntax MLMs from decision tree models only.

Methods

The PMML version 4.3 and the Arden Syntax version 2.9 were used to transform PMML files into Arden Syntax MLMs [21].

First, a Java project was created with the Eclipse Oxygen 3a release. This program gets a PMML file as well as an output folder location as input. The program starts by validating the PMML file against an XML Schema Definition (XSD) file to ensure the validity of the file against the PMML specification. This XSD file can be downloaded free of cost from the PMML website [16].

Once the validity of the input file has been confirmed, the transformation into Arden Syntax takes place. This is done by using the extensible stylesheet language transformation (XSLT) method. The rules for the transformation are written into XSL files. In XSLT, the transformation rules are split into templates. Each template matches a specific XML tag and defines how it should be transformed. Using the XPath syntax, data from other XML tags can be transformed.

The main template matches the "PMML" tag which contains the entire PMML file. This template creates the basic structure of the MLM file and calls other templates to fill the file with the necessary data. Some of the basic XSLT elements for the creation of the MLM are: "text", "value-of", "call-template", "apply-templates", "if", "for-each" and "choose". The "text" element writes an arbitrary text into the result file. With a "value-of" element, the content or an attribute of an XML element is written into the result file. "Call-template" and "apply-templates" are used to call other templates. The "call-template" executes a specific template, while "apply-templates" iterates through all XML tags given to the method and executes a matching template. The "if" and "for-each" elements have the common functionalities of these methods in other programming languages while the "choose" element correlates to an "if-elseif-else" block. In addition to the named elements, a number of XSLT elements were employed to create a well formatted MLM out of the PMML file.

We used three PMML files for the purpose of testing. The first was taken from the examples given in the PMML specification [16]. It defines a tree to evaluate the suitability of weather conditions for playing golf. The second MLM uses the "iris" data set (also Fisher's Iris data set) that is publicly available and often used for demonstrating machine learning algorithms (see UCI Machine Learning Repository [22]). The third tree uses the breast cancer data set (also from the UCI Machine Learning Repository [22]), which includes 286 cases described by ten attributes.

The data were imported in R and the rpart package was used to train a decision tree for the breast cancer data set using the CART algorithm as well as for the iris data set. The tree was generated by the R script in Figure 3 that uses the patient's age, the size of the tumor, as well as other features to predict whether there will be any recurrence of events.

The three PMML files and unit tests were used for testing. The PMML files were fed as input into the PMML-to-Arden-Syntax transformer. The resulting MLM files were copied into the ArdenSuite IDE and then compiled. In the subsequent tests, known input data were applied and the outcome was compared with the known results of the test cases in order to guarantee reproducibility.

```
library(pmml)
library(rpart)

train <- read.csv(file="[...]/test-data/breast-cancer.data",
          header=TRUE, sep=",")

fit <- rpart(class ~ age + tumor.size + inv.nodes +
          deg.malig + irradiat,
             data = train,
          method = "class")

saveXML(pmml(fit), "Breast-Cancer_tree.pmml")
```

Fig. 3. R script for the creation of a test PMML.

Results

The completed PMML-to-Arden-Syntax transformer consists of two Java classes, three XSL files, and eight prewritten MLMs. The two Java classes are the transformer itself and a separate class for the XSD check. The interface in the form of the method headers of the public methods is shown in Figure 4.

```
static public PMMLtoArdenTransformer getInstance()

public void convertToArdenFile(String pmmlPath, String resultPath) throws
            IllegalArgumentException, SAXException, IOException,
            TransformerException, NoTreeModelException

public void convertToArdenFile(File pmmlPointer, File resultPointer) throws
            SAXException, IOException, TransformerException,
            NoTreeModelException, IllegalArgumentException
```

Fig. 4. Transformer interface/public methods [23, p. 82].

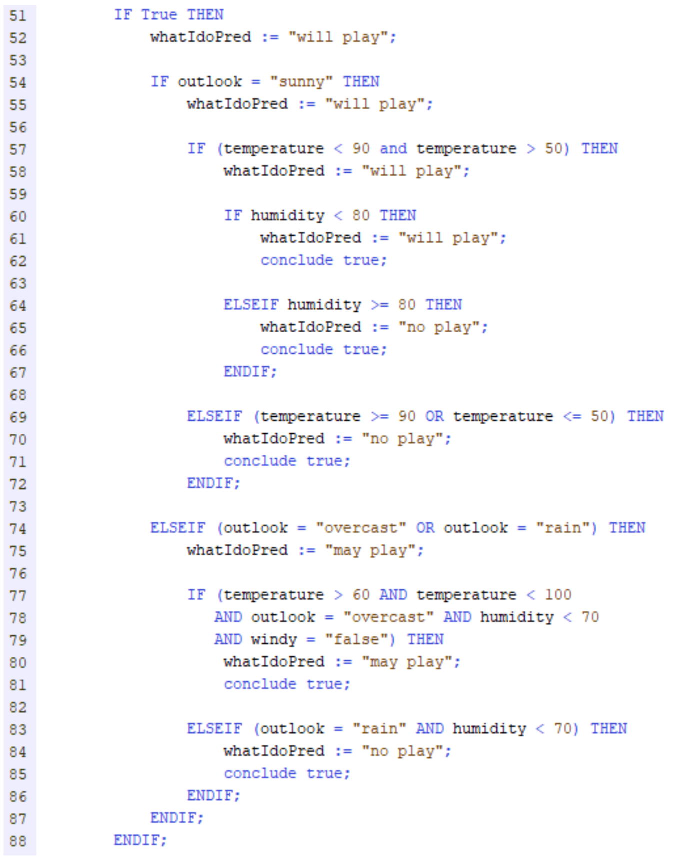
The “getInstance” method is the first that has to be called, because it creates and returns an instance of the transformer. The two overloaded methods “convert-ToArdenFile” take a PMML file and a result folder in the form of either a string containing the local path or a java.io.File. These methods start the entire transformation process by calling the necessary private methods.

The XSLT rules were separated into three XSL files since one transformation can only create one new file. The first XSL file has 2875 lines of XSLT rules and defines the main transformation of a PMML decision tree into an Arden Syntax MLM. The second XSL file consists of 903 lines of XSLT rules. Its purpose is to create additional MLMs for the custom-designed functions. These functions consist of the build-in functions of the PMML specification and are used for pre- and post-processing the data. These are written into separate MLMs to create a more concise structure. The third XSL consists of 1069 lines of XSLT rules and is needed in the event of multiple models within one PMML file. This file creates a control MLM that checks the conditions given for each model to be executed. All models that meet their conditions are called and the results collected and combined to receive an output. In the event of multiple models, the first XSL file is used to transform each model individually and write each model into a separate MLM file.

Only the first XSL file is always used for the transformation. The other two files are used if needed. The check as to whether one of the other XSL files is needed for the transformation is part of the Java program. The PMML file is searched for specific XML tags or a specific content of an XML tag that serves as an indicator for the need of one of the additional transformations.

The eight prewritten MLMs are created in the same folder as the transformed MLM, if they are needed to execute the logic of the PMML. The need for these MLMs is checked the same way as the need for additional XSL files. Seven of those are needed for some pre- and post-processing methods, while one is used for the execution of the model itself. The latter takes a list of Boolean values as input and applies the “exclusive or” (XOR) operation on them. The PMML specification defines the XOR operation to only return true if an odd number of predicates evaluate to true and all others to false. The other seven pre-written MLMs for pre- and post-processing are: “DateDaysSinceYear” “DateSecondsSinceYear”, “LevenshteinDistance”, “normalPDF”, “piecewiseLinearInterpolation”, “replaceString” and “TextIndex”. The two MLMs “DateDaysSinceYear” and “DateSecondsSinceYear” are similar. They calculate the number of days/seconds that passed since the 1st January 00:00 of the year given as an input. Dates prior to that year are shown as negative numbers. The “TextIndex” MLM determines the frequency of a term in a text. An optional input parameter for this MLM is the maximum Levenshtein distance for a term to be still accepted for counting. The Levenshtein distance between two terms may be calculated with the MLM “LevenshteinDistance”. This distance is the number of character additions, omissions, or replacements needed to change one term into another. For normalizing a sequence of points, the MLM “piecewiseLinearInterpolation” is needed. The MLM “replaceString” searches for all occurrences of a string in a text and replaces them with another string. The “normalPDF” MLM calculates the normal distribution.

All eight prewritten MLMs were needed because the implementation of these methods in Arden Syntax involves a complex code. Therefore, instead of many "text" tags with a complete prewritten code in the main MLM, the prewritten MLMs are included in the data slot and called at the appropriate line in the knowledge slot.



```
51          IF True THEN
52              whatIdoPred := "will play";
53
54              IF outlook = "sunny" THEN
55                  whatIdoPred := "will play";
56
57                  IF (temperature < 90 and temperature > 50) THEN
58                      whatIdoPred := "will play";
59
60                      IF humidity < 80 THEN
61                          whatIdoPred := "will play";
62                          conclude true;
63
64                      ELSEIF humidity >= 80 THEN
65                          whatIdoPred := "no play";
66                          conclude true;
67                      ENDIF;
68
69                  ELSEIF (temperature >= 90 OR temperature <= 50) THEN
70                      whatIdoPred := "no play";
71                      conclude true;
72                  ENDIF;
73
74              ELSEIF (outlook = "overcast" OR outlook = "rain") THEN
75                  whatIdoPred := "may play";
76
77                  IF (temperature > 60 AND temperature < 100
78                     AND outlook = "overcast" AND humidity < 70
79                     AND windy = "false") THEN
80                      whatIdoPred := "may play";
81                      conclude true;
82
83                  ELSEIF (outlook = "rain" AND humidity < 70) THEN
84                      whatIdoPred := "no play";
85                      conclude true;
86                  ENDIF;
87              ENDIF;
88          ENDIF;
```

Fig. 5. Representation of a decision tree in Arden Syntax. Snippet of an MLM generated by the PMML-to-Arden-Syntax transformer. Example taken from [23, p. 49].

Figure 5 shows a part of an MLM that was created by the PMML-to-Arden-Syntax transformer from the PMML given as an example by the PMML specification. Each node in the tree is represented by an "if-block" (e.g., line 54 in Figure 5). Child nodes are contained within a parent "if-block" (e.g., line 57 in Figure 5), while sibling nodes are connected via an "elseif" statement (e.g., line 64 in Figure 5). Leaf nodes contain a "conclude true" statement after assigning a result to the appropriate variable (e.g., lines 60 until 62 in Figure 5). The node represented in lines 77, 78, and 79 of Figure 5 was originally in one line but was split into three for better readability.

A separate Java project that implements the PMML-to-Arden-Syntax transformer project was created in order to test the transformer. This program creates an instance of the transformer and subsequently uses the "convertToArdenFile"-

method that takes the local path as a string to start the transformation. In addition, 88 unit tests were written to test different input parameters and a range of PMML to Arden conversions according to the PMML and Arden specifications.

The transformation of all three test PMML files and all unit tests were executed without any errors. To compile the three main resulting MLMs, in all three cases the date slot of the MLM had to be changed. The example given in the PMML specification did not include a date at all; therefore the date had to be added to the resulting MLM. The file created by the R script contained a date and time, but in a different format. The date was written in the “YYYY-MM-DD hh:mm:ss” format, whereas the Arden Syntax requires the ISO 8601 format “YYYY-MM-DDThh:mm:ss”. After adding and editing the date, all MLMs could be compiled without any error. The last part of the test, involving the entry of test inputs with known results into the compiled MLMs, was passed without any errors.

Discussion

The tree model is currently the only model that can be transformed with the PMML-to-Arden-Syntax transformer. Its transforming capabilities may be expanded to other models in the future. Extending the interoperability of the PMML-to-Arden-Syntax transformer to other models will take less time and effort than the first implementation of the transformer. Many of the XSLT rules can be reused for new models because they constitute PMML structures that may be present in all models and do not change from one model to another. For example, pre- and post-processing is identical in all models. In addition, most of the functions may be used in all of the models.

Although the PMML specification does include the possibility of no model being contained in a PMML file, the PMML-to-Arden-Syntax transformer will return an error if there is no (tree) model present.

The PMML-to-Arden-Syntax transformer was created as a Java project that may be implemented as a library into another program. There is currently no graphical user interface because it was developed for being implemented as a subsequent add-on for the ArdenSuite IDE. This work has now been completed.

The decision tree model was translated as precisely as possible from the PMML specification into Arden Syntax code. Although Arden Syntax supports many different operators, not all functions defined in the PMML specification could be translated. For example, the functions normalCDF, stdNormalPDF, and erf, which may be used for the pre- and post-processing data, could not be represented in Arden Syntax. These functions use formulas that need an integral operation to be calculated. Since Arden Syntax does not define an integral function, these functions cannot be executed. Another example is the replace function which is used to replace each occurrence of a term in a text with another term. This functionality was implemented in Arden Syntax, but according to PMML the function should also understand regular expressions as input. This was not implemented into the transformer because Arden Syntax does not support regular expressions. Therefore, the replace function was only translated partially into Arden Syntax.

If one of the functions that could not be translated is used in a PMML file, a text file will be created in the same folder as the generated MLM. It contains information about the missing function and why it could not be implemented. Then the Arden Syntax developer can decide how to get around the problem. They could, for example, find a way to avoid needing this exact function or replace this function with a similar one that can be realized in Arden Syntax.

Conclusions

A decision tree model saved in PMML can be translated into an Arden Syntax MLM, which can subsequently be compiled with the ArdenSuite IDE. Thus, a new way to generate MLMs was created. The traditional method of knowledge design involves the combined efforts of clinical experts and knowledge engineers to create MLMs. However, in some cases a medical expert is only available for a short period of time because of time or financial restrictions; these make it impossible to use the traditional method of knowledge design. By using the machine learning approach, very complex and deep decision tree models can be built for large bodies of data with many features—models that would take a long time to build manually. The validation of such a model by a medical expert may take less time and involve less effort than the traditional approach.

With the PMML-to-Arden-Syntax transformer, machine learning results can be implemented into Arden Syntax directly. As seen in the example of breast cancer, given the availability of the appropriate data, an algorithm can create a new way of classifying medical data. This results in the creation of new knowledge rather than the digitization of a medical expert's knowledge. It does not fully replace the traditional approach of creating MLMs using the knowledge design approach, but is a very useful add-on.

References

[1] D. Hausmann, C. Zulian, E. Battegay, L. Zimmerli, Tracing the decision-making process of physicians with a Decision Process Matrix, *BMC Medical Informatics and Decision Making* (2016) **16**:133, doi: 10.1186/s12911-016-0369-1.

[2] D. Sílvério Rodrigues, P.F. Sousa, N. Basílio, A. Antunes, M. da Luz Antunes, M.I. Santos, B. Heleno, Primary care physicians' decision-making processes in the context of multimorbidity: protocol of a systematic review and thematic synthesis of qualitative research, *BMJ Open* (2019) **9**:e023832, doi: 10.1136/bmjopen-2018-023832.

[3] P.E. Beeler, D. W. Bates, B.L. Hug, Clinical decision support systems, *Swiss Medical Weekly* (2014) 144:w14073, doi: 10.4414/smw.2014.14073.

[4] R.A. Jenders, Advances in Clinical Decision Support: Highlights of Practice and the Literature 2015-2016, *IMIA Yearbook of Medical Informatics* (2017) **26**(1), 125-132.
[5] T.J. Bright, A. Wong, R. Dhurjati, E. Bristow, L. Bastian, R.R. Coeytaux, G. Samsa, V. Hasselblad, J.W. Williams, M.D. Musty, L. Wing, A.S. Kendrick, G.D. Sanders, D. Lobach, Effect of Clinical Decision-Support Systems, *Annals of Internal Medicine* (2012) **157**(1), 29-43.
[6] E.H. Shortliffe, M.J. Sepúlveda, Clinical Decision Support in the Era of Artificial Intelligence, *JAMA* (2018) **320**(21), 2199-2200.
[7] M. Samwald, K. Fehre, J. de Bruin, K.-P. Adlassnig, The Arden Syntax standard for clinical decision support: Experiences and directions, *Journal of Biomedical Informatics* (2012) **45**(4), 711-718.
[8] K.-P. Adlassnig, P. Haug, R.A. Jenders, Arden Syntax: Then, now, and in the future, *Artificial Intelligence in Medicine* (2018) **92**, 1-6.
[9] R.A. Jenders, K.-P. Adlassnig, K. Fehre, P. Haug, Evolution of the Arden Syntax: Key Technical Issues from the Standards Development Organization Perspective, *Artificial Intelligence in Medicine* (2018) **92**, 10-14.
[10] Health Level Seven International [Internet]. Ann Arbor: Health Level Seven International; c2018 [cited 2018 Aug 1]. Available from: http://www.hl7.org/index.cfm.
[11] Health Level Seven International. Health Level Seven Arden Syntax for Medical Logic Systems, Version 2.10, November 2014 [Internet]. Ann Arbor: Health Level Seven International; 2014 [cited 2019 June 20]. Available from: http://www.hl7.org/implement/standards/product_brief.cfm?product_id=372.
[12] Medexter Healthcare GmbH [Internet]. Vienna: Medexter Healthcare; c2018 [cited 2018 Aug 1]. ArdenSuite. Available from: https://www.medexter.com/products-and-services/ardensuite.
[13] K.-P. Adlassnig, K. Fehre, Service-Oriented Fuzzy-Arden-Syntax-Based Clinical Decision Support, *Indian Journal of Medical Informatics* (2014) **8**(2), 75-79.
[14] A. Guazelli, What is PMML? [Internet]. 2010 [cited 2018 Aug 6]. Available from: http://www.ibm.com/developerworks/library/ba-ind-PMML1/index.html.
[15] Data Mining Group. Data Mining Group [Internet]. 2018 [cited 2018 Aug 6]. Available from: http://dmg.org/.
[16] Data Mining Group. Data Mining Group - PMML version 4.3 [Internet]. [cited 2018 Aug 1]. Available from: http://dmg.org/pmml/pmml-v4-3.html.
[17] J. Fürnkranz, Decision Tree. In: Sammut C, Webb GI, editors. *Encyclopedia of Machine Learning and Data Mining* [Internet]. Boston, MA: Springer US; 2017 [cited 2018 Aug 2]. p. 330–5. Available from: http://link.springer.com/10.1007/978-1-4899-7687-1_66.
[18] V. Podgorelec, P. Kokol, B. Stiglic, I. Rozman, Decision trees: an overview and their use in medicine, *Journal of Medical Systems* (2002) **26**(5), 445-463.
[19] W.-L. Loh, Fifty Years of Classification and Regression Trees, *International Statistical Review* (2014) **82**(3), 329-348.
[20] L. Breiman, J. H. Friedman, R.A. Olshen, C. J. Stone, *Classification and regression trees*. Boca Raton: Chapman & Hall/CRC, 1984.
[21] Health Level Seven International. The Arden Syntax for Medical Logic Systems, Version 2.9, March 2013 [Internet]. Ann Arbor: Health Level Seven International; 2013 [cited 2019 June 20]. Available from: https://www.hl7.org/implement/standards/product_brief.cfm?product_id=290.
[22] D. Dua, E. Karra Taniskidou, UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, 2017.
[23] M. Spineth, *Interoperability with a clinical decision support rule engine*, September 2018 [master thesis]. Vienna: University of Applied Sciences Technikum Wien.