Semantic Composition of Data Analytical Processes

Peter Bednár, Juliana Ivančáková, Martin Sarnovský

Technical University of Kosice, Faculty of Electrical Engineering and Informatics, Department of Cybernetics and Artificial Intelligence
Letna 9, 042 00 Kosice, Slovakia
peter.bednar@tuke.sk, juliana.ivancakova@tuke.sk, matrin.sarnovsky@tuke.sk

Abstract: This paper presents the semantic framework for the description and automatic composition of the data analytical processes. The framework specifies how to describe goals, input data, outputs and various data operators for data pre-processing and modelling that can be applied to achieve the goals. The main contribution of this paper is the formal language for the specification of the preconditions, postconditions, inputs and outputs of the data operators. The formal description of the operators with the logical expressions allows automatic composition of operators into the complex workflows achieving the specified goals of the data analysis. The evaluation of the semantic framework was performed on the two real-world use cases from the medical domain, where the automatically generated workflow was compared with the implementation manually programmed by the data scientist.

Keywords: data science; data mining; semantic technologies; ontology

1 Introduction

Analysis of data and application of machine learning and artificial intelligence methods become very important approach successfully applied to the research or business problems in the various domains. However, the application of these methods is not always straightforward and requires extensive knowledge exchange between data scientists and domain experts. Additionally, the implementation of these applications is rather complex, with many constraints coming from the definition of the goal, properties of input data and constraints of the algorithms applied to generate the results. The result is that implementation of the data analytical approach can be time and resource-consuming. In this paper, we propose the semantic framework for automatization of the data-analytical processes based on the application of ontologies and logical inference. The proposed framework allows to automatically compose data-analytical workflow based on the semantic description of the goals.

The paper is organized as follows. The first chapter describes the current state-of-the-art and our motivation. In the following Chapter 3, we introduce our semantic model for data-analytical processes, followed by the main contribution of this paper: semantic description of the data analytical processes for the automatic composition of workflows and additional types of data operators covering various data and model visualization techniques. Last Chapter 4 then presents the evaluation of the proposed approach on the real application cases from the medical domain.

2 Semantic Description of Data Analytical Processes

A few semantics have been proposed to describe data-analytical process models formalized as ontologies [1-2]. One of the main proposals is the OntoDM ontology [3], which consists of 3 modules. The first module deals with the specification of input and output data and is based on the ISO standard for describing data types (atomic and composite). The main module characterizes the concepts that describe the data, the data analysis tasks (such as classification, clustering, etc.), the data mining algorithms, and the analysis outputs in the form of general data mining models. The third module uses concepts from the first two modules to formalize the different phases of the overall process according to the CRISP-DM methodology [4].

The other proposed ontologies extend the definition of some concepts introduced in OntoDM. DMOP ontology [5] deals mainly with the detailed description of the data mining algorithms, including their internal principle, e.g., a description of the numerical optimization method used, the error function, or the regularization of learning. DMOP covers 3 phases of CRISP-DM. DMWF ontology [6] defines data operators, which are applicable in data pre-processing, modelling, and evaluations. Operators are described similarly to services by definition of their inputs, outputs, assumptions, and effects. These input and output conditions are defined as logical expressions in the SWRL language as far as possible use of automatic derivation in the composition of workflows at data analysis.

One of the most general proposals that extend OntoDM is ontology Exposé [7], which extends the CRISP-DM phase of the data analysis process to the level of experiments to ensure, e.g., reusability of procedures in data analysis and reproducibility of results. In addition to conceptualization, it also provides a language for describing experiments based on XML [8], which allows you to publish and share a description of experiments on the web in a machine-readable format.

Panov et al. describe OntoDT, ontology to represent knowledge about data types. This ontology defines basic entities such as datatypes and their properties, specifications, characterizing operations, and datatype taxonomy [9].

In [10] Tianxing et al. presents a meta-mining ontology, which is used for building a domain-oriented ontology. The main goal of creating INPUT ontology is to understand data and business goals better and use it as an input interface for user queries.

The conceptualization of these semantic models is based on the description interfaces of existing software tools used for data analysis, such as R environment or sci-kit-learn library. Other relevant technologies can also include formats for exchanging data-analytical models when deploying them, such as XML standard PMML, PFA format based on JSON notation or currently the most supported ONNX format focused mainly on the exchange of models of deep learning.

Data Science Ontology (DSO) is a data science knowledge base focusing on computer programming. DSO is a way of organizing and classifying the concepts and entities within the field of data science. It helps define the relationships between different aspects of data science work and provides a framework for understanding and communicating about the field. One important aspect of data science ontology is the classification of data types and sources. This includes things like structured data, unstructured data, and semi-structured data, as well as data sources such as databases, APIs, and text files. Another important aspect is the classification of data analysis and modelling techniques. This can include things like statistical methods, machine learning algorithms [11], data visualization techniques, and natural language processing. The concepts for this ontology are gleaned from statistics, machine learning [29], and software engineering for data science. In addition to concepts, ontology also provides semantic annotations for data science. The annotations map the types and functions of the libraries to the universal concepts of ontology [12]. Data science ontology also includes the different roles and responsibilities within a data science team. This can include roles such as data engineer, data analyst, data scientist, and machine learning engineer, as well as the specific tasks and responsibilities associated with each role. Data science ontology can also include the models, frameworks and methodologies that are used in the field. These can include things like CRISP-DM for data mining, SEMMA for data mining and statistics, and the OSEMN framework for data science.

Data visualization and models should not be forgotten in the preprocessing and modelling framework; there is a VISO ontology [13] for describing such concepts, which formally models concepts and facts specific to visualizations. Visualization ontology is an important area of study, as it provides a structured and consistent way of understanding and discussing the field of data visualization. By understanding the different types of visualizations, design elements, and ways of representing data, we can create more effective and engaging visualizations that can help to communicate complex data in a more understandable way. The advantages of this ontology, also achieved thanks to well-established semantics standards such as RDFs [14] and OWL [14-16], are technical interoperability, support for a common understanding between interdisciplinary

parties in the visualization process, and the ability to derive new knowledge from existing facts. Viso is characterized by being composed of 7 modules: Graphic - formalizes concepts related to graphical relations and representations; Data - defines data variables and structures; Facts - formalizes constraints, rankings and defaults; Activity - deals with the human aspect within the visualization; System - this module covers HW and SW; User - characterizes user extensions and Domain - describes the domain specifications.

3 Semantic Framework for Automatization of Data-Analytical Processes

In our previous work [17-18], we have defined the semantic framework for the description of the data analytical processes, which is divided into the following modules:

- Domain Concepts - concepts for the description of entities and known relationships in the domain under investigation used for data analysis methods.
- Data Items and Performance Indicators - concepts for the description of key performance indicators formalising business and research requirements and goals of data analysis and concepts to describe input and output data attributes and data sets.
- Algorithms and Data Mining Models - concepts for describing methods and data mining algorithms and their settings, and concepts for describing data mining models (main outputs of data analysis).

The first module is mainly designed from the point of view of a domain expert, using domain concepts to formalise the description of a given domain. The second module contains concepts that are shared between the domain expert and the data analyst and is used to formally describe the goals of the analysis and the data. The last module was mainly proposed for the semantic documentation of the existing data-analytical processes to achieve interoperability and reproducibility of the processes. The main contribution of this paper is in the extension of the framework, with the concepts which will allow the automatic composition of the workflows and automation of the data-analytical processes. The following subchapters will describe the proposed modules in detail.

3.1 Domain Concepts

Concepts for domain formalisation are specified using the SKOS metamodel [19], which allows the specification of title, narrative description and definition of

concepts localised in several natural languages. Concepts can be arranged hierarchically in the form of a thesaurus/taxonomy by the relations skos#broader/skos#narrower. It is also possible to define polyhierarchical schemes. In addition to the hierarchical arrangement, terms can also be linked associatively by the relation skos#related.

A defined common dictionary of domain terms can also be used as a classification scheme for organising different types of documents that enter into data analysis as domain documentation, created by data analysts to document the analysis process itself, the data and the results achieved. The document types themselves can be specified as a classification taxonomy in SKOS.

In addition to the narrative description, in some domains, it is also appropriate to explain existing concepts using various diagrams, schemes and other types of graphical notations (e.g., using BPMN diagrams for modelling business processes or process diagrams for visualising production processes in the field of Industry 4.0, etc.). In this case, the individual graphical elements (e.g., an activity block in a BPMN diagram) are described as separate SKOS concepts that are linked to a given element to allow bi-directional navigation between the graphical notation and a set of semantic concepts. However, the proposed formalism does not define how these links are represented either in a semantic representation or inserted directly into the graphical notation format.

3.2 Data Elements and Performance Indicators

Figure 1 illustrates the basic concepts that represent the input and output data. Data elements are specified on two levels: logical and physical. Logical data elements are utilized to describe every input data attribute that the domain expert recognizes as relevant for resolving the given task of data analysis or for describing all output data produced during the analysis process, including prediction of the data-analytical models or values of domain and technical performance indicators.

The logical representation defines the metadata for each data attribute, which includes its name and definition in natural language, logical data type (for example, whether it is nominal, ordinal, numeric data, scalar, vector quantity, spatially-arranged data, time series, etc.), the commonly used physical unit of measurement, and the role of the data element in the data analysis process (such as whether it is input, output, or input-output data).

Logical data attributes can be connected by interdependencies, which are represented by the Dependency class. The Dependency class defines a relationship between one dependent data element and one or more independent elements. The dependency can be described in text or specified mathematically using a known physical or economic model. Dependencies can be further specified by

basic type, for example, expressing whether the value of a dependent element is derived from an independent element through transformation or is an aggregation of multiple independent elements, etc.

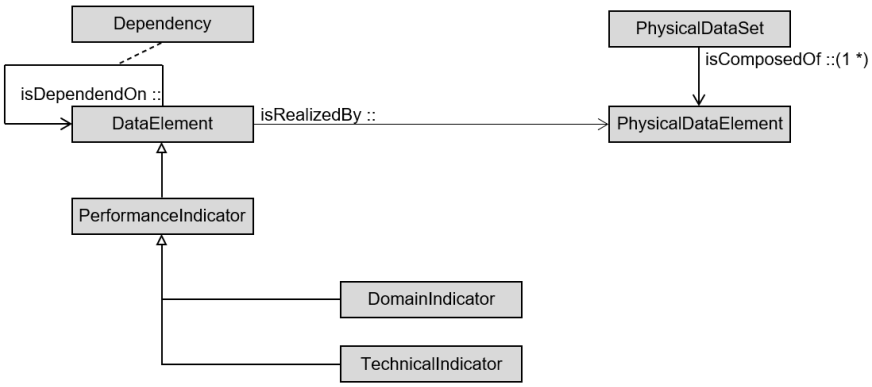

Figure 1
Data elements and performance indicators

The physical location of the data during analysis is represented by the physical data elements, and one logical attribute can have multiple physical realizations. Physical data elements are assigned physical data formats and types. The physical type is based on the ISO standard for describing data formats in software systems and can be atomic (real/integer, Boolean value, string) or composite (record with data fields, list, or unordered set). The specific location of the data is defined using IRI, which represents a unique identifier in a software environment for data analysis or a URL for placement on the web. Multiple physical data elements can be combined into a single data element set referenced through IRI.

The results of the data analysis are quantitatively described by the measurable performance indicators, which are defined as the special type of data elements. Similarly, to the description of input data, the performance indicators are divided into two subclasses: domain indicators and technical indicators. Domain indicators are commonly introduced by the domain expert for the evaluation of the results from the business perspective. They include indicators which specify, for example, financial costs/savings, energy or resource consumption, environmental impact, etc. The technical indicators include statistics expressing the performance of the data-mining model, such as accuracy, specificity, sensitivity, etc., estimated on the test or validation data set or using the cross-validation. In addition to performance metrics, technical indicators also cover the various metrics expressing the complexity or interpretability of data-mining models (e.g., the number of numerical parameters, the number of classifications or association rules, the number of clusters, etc.). Technical indicators are formally mapped by the data analyst to domain indicators by specifying the dependencies represented by the Dependency class, which allows to retrospectively determine how the

technical quality of data-mining models quantitatively affects the required quality of business goals.

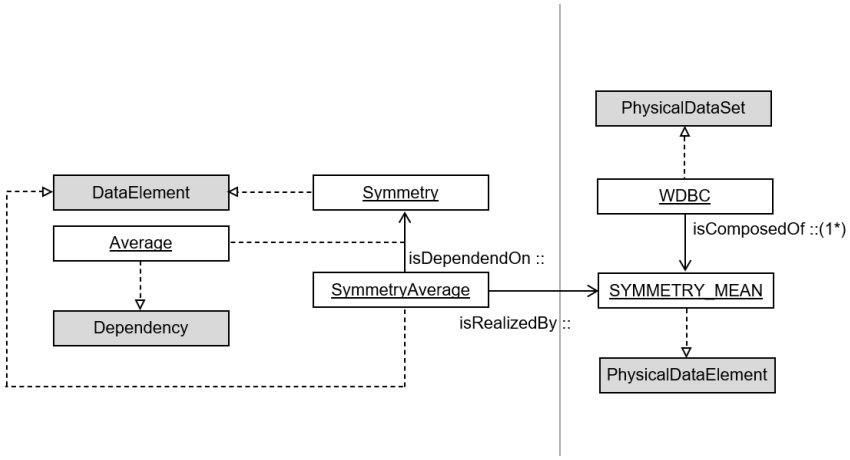

Figure 2
Example of the physical and logical data elements

Figure 2 presents an example which demonstrates the relations between logical and physical data elements. The physical dataset is represented with the WDBC instance, which points to the data stored in the comma-separated value file on the disk. The file consists of multiple columns, which are represented as instances of the Physical Data Element class. The example column is represented with the SYMMETRY_MEAN instance, which corresponds to the logical data element represented by the SymmetryAvrg instance. Another example of the logical element is Symmetry instance from which the SymmetryAvrg element is derived by arithmetic averaging. The dependency between the two logical elements is formally represented by the Average instance of type Dependency. In this case, the Average instance can specify directly by a structured formula for the computation of the arithmetic average over the source element.

3.3 Algorithms and Data-Analytical Models

The use of the algorithm in the analysis is determined by the definition of the data mining task as specified by the data analyst (Figure 3). The task is defined by a set of constraints (Constraint class) that use logical expressions to outline the desired properties of the resulting data mining model. These constraints cover the characteristics of the input data (such as data type attributes and the presence of missing or extreme values), the type of desired model (such as classification, regression, clustering, association rules, anomaly detection, etc.), and the quality and interpretability of the model by limiting technical performance indicators.

These constraints are further divided into hard constraints, which must be fully met in the solution of the task, and soft constraints, which should be taken into consideration in the solution but may not necessarily be met unconditionally.

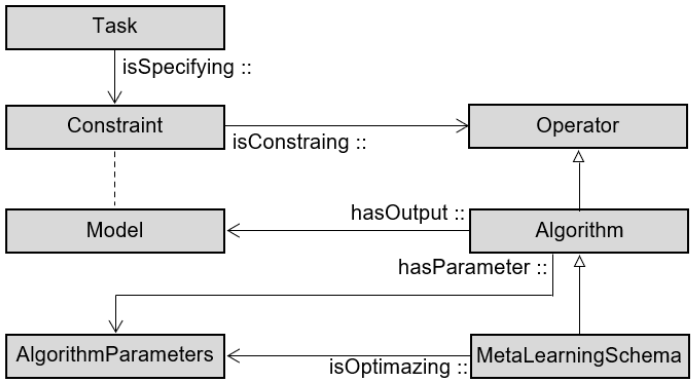

Figure 3
Data mining tasks, algorithms, and models

The main concept which covers all operations over the data is the Operator class. The machine learning algorithms are represented with the Algorithm concept, which is the special type of the operator with the data input and output in the form of the data mining model. The Model itself is the Operator, which can be applied to the input data for scoring. The output of the Model operator are data elements with predicted values and with the optional additional metadata such as confidence scores, identifiers of the classification rules, or weights of the contributing input values. In addition to the output model, an algorithm can have a set of (hyper) parameters that need to be set before it can be executed. Each parameter has a defined data type and a predetermined value. A special type of algorithm is the MetaLearningSchema class [20], which internally optimises parameter settings for a given training set and basic learning algorithm or selects the best algorithm from a group of algorithms for a given training data.

Figure 4 presents an example of the formalisation of the predictive machine learning algorithm and model. The RandomForest algorithm is the instance of the Algorithm concept, which is constrained for the classification task. The classification task specifies the constraint for the output predicted value (it must be of ordinal or nominal type). The algorithm is the data operator, which can be applied to input data represented as the physical dataset and which outputs the data mining model represented as the RFModel instance. An algorithm can have multiple parameters (settings which must be specified by the data scientist or automatically optimised), which are represented as the instances of type AlgorithmParameter. The figure shows a parameter for the number of trees included in the random forest model. Parameters can be semi-automatically optimised using the meta-learning schema. In the presented example, the number

of trees parameter is optimised by GridSearch meta-algorithm, which iteratively learns and evaluates model using the given algorithm (e.g., RandomForest) and selects the optimal value of the selected parameter.

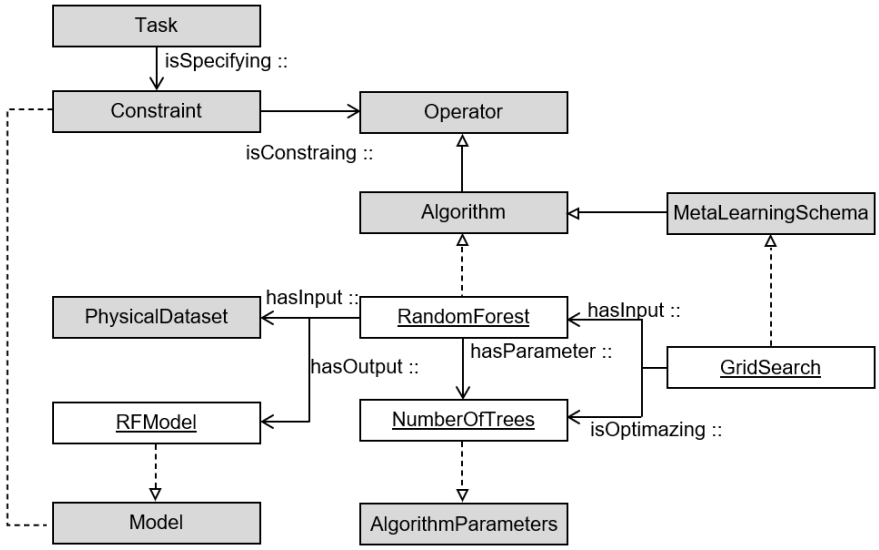

Figure 4
Example of the algorithms and data-analytical models

3.4 Process Model for Data Analytical Workflows

The proposed process model can be used to automatically generate workflows for data analysis tasks and also to formally describe existing data analysis scripts, ensuring their replicability and reusability. It is designed similarly to a process model for choreography and orchestration of web services, with the state represented by instances assigned to the shared variables. The process consists of nodes (Figure 5) that represent individual operations (Operator class) for data preprocessing, modelling, and evaluation, or control blocks such as branches, cycles, parallel execution, and synchronization (ControlNode class). The nodes are connected in a workflow by edges represented by the GuardedTransition class, which represents conditional transitions between nodes. Operators are described functionally, with inputs, outputs, assumptions, and effects defined as logical expressions. The flow of the process is determined by backward chaining of the effects and assumptions, taking the desired target effects from the task specification of the data mining. Some restrictions may be imposed by the algorithms used in the workflow, such as the ability to only work with certain types of attributes or to handle missing values.

An important part of our semantic framework is the formalism used for the description of the logical expressions in the specification of the operators inputs, outputs, preconditions and postconditions. The formalism is divided into variants with gradually increased expressiveness, which allows for choosing a trade-off between complexity and expressiveness and simplify the implementation of the automatic planning. The full specification was based on the Web Service Modelling Language (WSML) formalism [21], which combines constructs from descriptive logic and logic programming. WSML expressions consist of logical variables, functional symbols, logical operators (and, or, classical negation and negation as a failure) and quantifiers (existential and universal). Our current proposal substantially reduced the expressiveness of the WSML expressions in order to even further simplify formalism and implementation of the automatic method for the process composition.

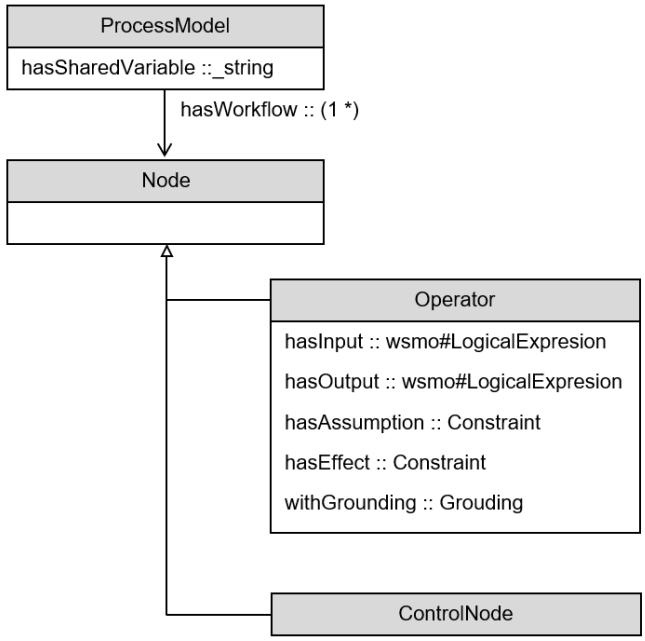

Figure 5
Process model for data analytical workflows

The current specification corresponds to the WSML light variant with the following constraints:

- All logical variables in the expressions are universally quantified.
- If α, β and γ are terms (identifiers, data values or variable symbols): α **subConceptOf** γ, α **memberOf** γ and $\alpha[\beta$ **hasValue** $\gamma]$ are atomic formulas where *sub-concept of* predicate specifies that the type α is the

sub-type of the γ, *member of* predicate constrains the type of the term α and *has value* predicate defines that the term α has value γ for the property β.

- Atomic formulas can be further combined with the logical operators **and**, **or** and negation as a failure (denoted as **not**).

Examples

The following example describes inputs, outputs, preconditions and postconditions for the operator, which replaces missing values of all numerical attributes in the input dataset.

Inputs:

?x **memberOf** PhysicalDataset

Preconditions:

?x[hasDataElement **hasValue** ?y] **and** ?y **memberOf** NumericalAttribute

Outputs:

?x **memberOf** PhysicalDataset

Postconditions:

not hasMissingValues(?y)

The operator defines the physical dataset as an input with the precondition that all data elements (columns in the input physical dataset) have a numerical type (note that all variables are implicitly universally quantified). The output of the operator is the same dataset with the postcondition that all attributes are without missing values (tested with the hasMissingValues predicate). All variables within the definition are shared between the inputs/precondition and outputs/postconditions, so for example, it is not necessary to bind variable ?y in the postconditions since it was already constrained in the preconditions.

The following example defines the operator for the logistic regression machine learning algorithm.

Inputs:

?x **memberOf** PhysicalDataset

Preconditions:

?x[hasDataElement **hasValue** ?y] **and**

?y **memberOf** NumericalAttribute **and**

not hasMissingValues(?y)

Outputs:

?z **memberOf** LogisticRegressionModel

The input to the operator is the physical dataset with numerical attributes without the missing values, and the output is the logistic regression machine learning model represented with the type LogisticRegressionModel (which is subsequently sub-concept of ClassificationModel, etc.).

Finally, the process of creating an executable workflow in a specific software environment involves anchoring the operators with the Grounding class. Grounding is a customizable text template that generates code for the operator's function in the environment when filled with variables. Operators can have multiple anchors, each designed for a different programming language (e.g. R or Python) or version of the software library used. The task specification can also include constraints on the anchoring, ensuring that only operators compatible with the given environment are used in the workflow.

3.5 Visualization Concepts

These concepts describe visualization in data mining processes. The key concepts are Algorithm, which is a type of operator that depends on input variables, and VisualizationMethod, which is an operator that takes data or a model as an input and outputs a visualization in the form of a graph.

Figure 6 illustrates the concepts of data visualization. As an example, we have selected the visualization of the PDP method for explaining machine learning models. In the beginning, it is important to define the type of visualization that is required. In principle, any visualization requires data as input. Either the data can be visualized as part of the data understanding or preprocessing phase, or the output of the visualization can be in the form of graphical representations of the models used for data mining. The concept of Algorithm is treated as a class operator, which is represented as a generic operation that transforms an input of a specific type into the desired output. The output of algorithms is then represented by the Model class. This concept is used together with data as input for the PDP (Partial Dependence Plot) [22], a graphical tool that helps understand the relationship between the input and the predicted variable. The outputs of the PDP operator are used to compute PDPStats, defined as TechnicalIndicator, which is a technical indicator specified based on domain indicators. The values obtained from PDPStats are inputs to the PDPVisualMethod. The final desired output is a PDP graph, which is a subclass of the VisualizationMethod operator and is specified in this case as a Graph.

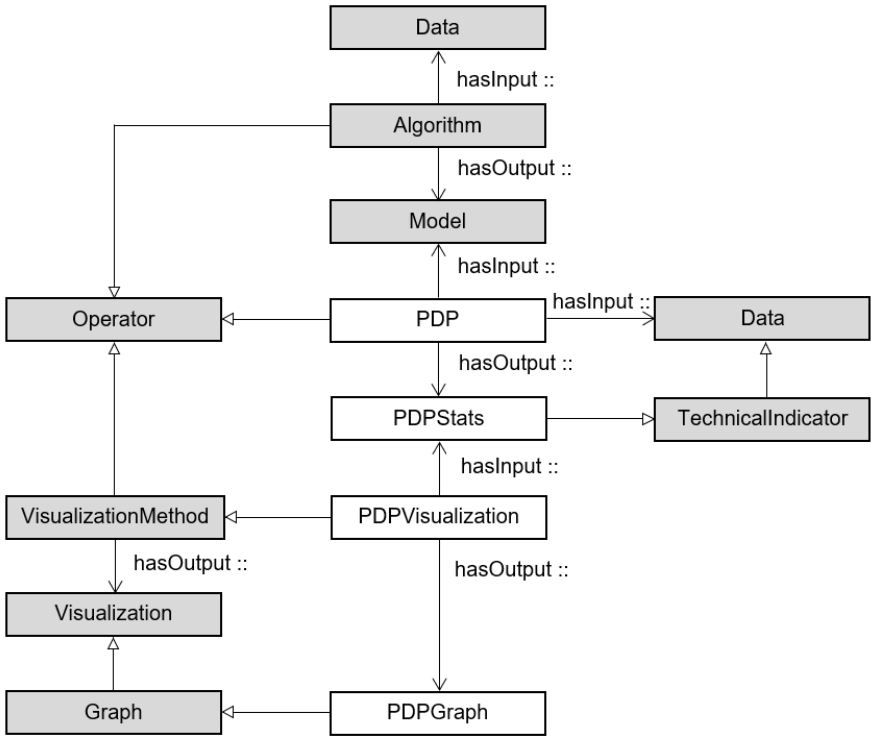

Figure 6
Example of the visualization concepts

4 Experiments

The proposed semantic model is intended for the formal description of the data analytical processes, ensuring their reproducibility and interoperability and for the automation of the analytical processes. To evaluate the proposed approach, we have applied the semantic model to two real-world case studies from the medical domain [23]. In the evaluation, we have at first manually annotated all scripts used for the pre-processing, modelling and evaluation of the machine learning models with the concepts from the proposed semantic model; and, second, compared code manually created by the data scientists with the code automatically generated from the knowledge graph.

For the comparison of the code, we have defined mainly metrics based on the code coverage [24], namely the number of lines of the code (including control statements for branching and cycling), the number of exact operator matches and

the number of operators with the partially matched parameters. We have defined the semantic constraints for the final data mining model to exactly match the results of the expert's code without the additional automatic optimization using the meta-learning schema. In addition to evaluating the coverage of the code, the accuracy of the learned models was also tested, and there were no significant differences in the quality of the learned models between the generated and the original code.

5 Evaluation

The first task was a binary classification for the diagnosis of breast cancer. The features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image, such as radius, texture, perimeter, area, etc. All features were real-valued numerical attributes without the missing values. Some attributes were derived from the source attribute but aggregation functions (e.g., overall average symmetry computed from the symmetry of each annotated cell, etc.). Together the dataset [25] contains 32 attributes. The code manually created by the data scientists [26] was in the R language and covered pre-processing, modelling, and evaluation for the six data mining algorithms for the classification ranging from decision trees, random forest, k-nearest neighbours, naïve Bayes classifier (with the normal distribution of the attribute probabilities) and support vector machines with the linear and polynomial kernels. The models were evaluated using the standard metric for the overall accuracy and precision/recall for the positive class.

For the evaluation, we have defined the semantic constraints for the final data mining model to exactly match the results of the expert's code without the additional automatic optimization using the meta-learning schema. The comparison of the code coverage between the experts' code and automatically generated code is summarized in Table 1.

Metric	**Expert code**	**Generated code**	**Coverage %**
Number of code lines	328	-	-
Number of code lines with operators	126		
Number of algorithms	6	6	100
Number of visualization methods	6	4	67
Number of evaluation metrics	1	1	1
Number of variables	98	80	82
Number of operator arguments	26	21	81
Number of functions	2	-	-
Number of branchings	8		

The scripts for this case were developed as the Shiny application, where a large part of the code is related to the programming of the user interface, which is not relevant to the data analytical task. From the remaining code, 126 lines contain the operations over data and models. The automatically generated code covers all 6 evaluated algorithms (decision tree, random forest, Naïve Bayes, k-nearest neighbour, neural network and SVM). Some parameters (i.e. number of decision trees in the random forest and the number of neighbours for kNN) were optimised in the original code with the simple cycle. This part of the optimisation was replaced in the semantic model with the equivalent grid-search operator. The models were evaluated with the same set of technical KPIs for binary classification (accuracy and contingency table) [27]. The coverage of the visualisation operators was 67%, where two visualisation methods were not covered by our current model. Non-covered visualisation methods show dependency between the selected technical KPI (e.g. precision) and one of the algorithm parameters (e.g. number of trees in the random forest algorithm). The coverage of variables also includes all variable aliases in the initial code (i.e. when the same data value is assigned to the two variables with different names). Overall, the semantic model has a much lover and consistent set of unique variables. All branchings in the original code were covered since they were related to the selection of the model. Additionally, the original code contains the definition of two helper functions. Both were used as data preprocessing operators, and both were replaced in the generated code with the equivalent functions from the standard R packages.

The second use case was also from the medical domain for the diagnosis of Acute lymphoblastic leukaemia (ALL). The dataset [28] was preprocessed from the set of images with a convolutional neural network, and the extracted features were reduced with the ANOVA method and with the importance of weighting based on the random forest tree algorithm. The final set of features includes 584 numerical attributes without the missing values. The code manually created by the data scientists was implemented in Python and covers preprocessing, modelling and evaluations.

Metric	**Expert code**	**Generated code**	**Coverage %**
Number of code lines	211	-	-
Number of code lines with operators	88		
Number of algorithms	4	4	100
Number of visualization methods	4(2)	4	100
Number of evaluation metrics	5	5	100
Number of variables	75		
Number of operator arguments	20		
Number of functions	5	-	-
Number of branchings	3		

In this case, the generated code covers all classification algorithms (random forest, support vector machine, naïve Bayes and k-nearest neighbours). Code also covers all five technical key performance indicators (accuracy, precision, recall, F1 score and confusion matrix). The visualisation methods cover mainly the visualisation of the confusion matrix with the heatmap. Additionally, scripts contain two visualisations of input image data before and after cropping, but these visualisations serve just for the visual checking for data scientists and are not relevant to the automatically generated code. An interesting case which was not covered by our current semantic framework is the usage of machine learning models for feature extraction. The framework just specifies that each predictive machine learning model can be used as a data operator for scoring (i.e. computation of the output prediction data element from the input data). However, in the presented use case, the internal state of the model (convolutional deep learning neural network) is used to extract input data features. Annotated extension, which was not covered in our current framework, is the case where the output of the predictive machine learning model is used for the feature importance weighting for feature selection. The latter case was a straightforward extension, but the former case requires better specification of the internal structure of the models (especially for deep learning models), which will be the goal of future work.

Conclusions

In this paper, we have presented the application of semantic technologies for automatization of the data-analytical processes. We have demonstrated that it is possible to semantically describe the goals of the data mining tasks and data analysis and automatically orchestrate the data mining workflows to find a solution to the goals. Additionally, the proposed semantic description can be used to formally document existing data analytical scripts for their reproducibility. In our experiments, we have demonstrated good coverage of the proposed semantic framework on the various real-case scenarios of data analysis in the medical domain. During the experiments, we have also identified some corner cases which were not initially covered in the framework, namely, visualization of dependencies between technical KPIs (performance metrics) and algorithm's parameters, usage of the internal state of the predictive models as the feature extraction method and usage of the output of the predictive models for feature selection.

Our experiments also showed that besides the overall quality of the final model (i.e., optimal business KPIs), an important part of the data scientist's work is also data understanding and post-analysis of results for explainability of the model. This is also the motivation for our future research, where we are planning to extend our semantic model to cover also constraints for better data understanding and explainability of the machine learning algorithms.

Acknowledgement

This work was supported by the grant APVV-16-0213, APVV-20-0232 and VEGA 1/0685/21.

References

[1] N. Guarino and P. Giaretta. “Formal ontology, conceptual analysis and knowledge representation”. International Journal of Human - Computer Studies 43, 625-640, 1995

[2] P. Panov. “A modular ontology of data mining”. Doctoral dissertation. Jožef Stefan International Postgraduate School, Ljubljana. 2012

[3] P. Panov, S. Dzeroski and L. N. Soldatova. “OntoDM: An Ontology of Data Mining”. In: 2008 IEEE International Conference on Data Mining Workshops. 2008. pp. 752-760

[4] M. Muchova, J. Paralic and M. Nemcik. “Using Predictive Data Mining Models for Data Analysis in a Logistics Company”. Information systems architecture and technology, PT I. Springer International Publishing AG, Gewerbestrasse 11, CHAM, Switzerland. pp. 161-170, 2018, doi: 10.1007/978-3-319-67220-5_15

[5] M. Hilario, P. Nguyen, H. Do, A. Woznica and A. Kalousis. “Ontology-Based Meta-Mining of Knowledge Discovery Workflows”. In: Meta-Learning in Computational Intelligence. 2011

[6] J. Kietz, F. Serban, A. Bernstein and S. Fischer. “Towards Cooperative Planning of Data Mining Workflows”. In: Proceedings of the Third Generation Data Mining Workshop at the 2009 European Conference on Machine Learning. 2009

[7] J. Vanschoren and L. Soldatova. “Exposé: An ontology for data mining experiments”. In: International workshop on third generation data mining: Towards service-oriented knowledge discovery. 2010. pp. 31-46

[8] S. Decker, S. Melnik, F. Van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann and I. Horrocks. “The Semantic Web: the roles of XML and RDF”. IEEE Internet Computing 4, 2000. pp. 63-73

[9] P. Panov, L. N. Soldatova, and S. Džeroski. “Generic ontology of datatypes”, Information Sciences. 2016. pp. 900-920, doi: https://doi.org/10.1016/j.ins.2015.08.006

[10] M. Tianxing, N. Zhukova, A. Vodyaho, and T. Aung. “A Meta-Mining Ontology Framework for Data Processing”. International Journal of Embedded and Real-Time Communication Systems. 2021. pp. 37-56, doi: 10.4018/IJERTCS.2021040103

[11] J. Palša, J. Hurtuk, M. Chovanec and E. Chovancová. “Using Machine Learning Algorithms to Detect Malware by Applying Static and Dynamic Analysis Methods”. Acta Polytechnica Hungarica. 2022, pp. 177-196

[12] E. Patterson, I. Baldini, A. Mojsilovic and K. Varshney. “What is the Data Science Ontology?” [online] [cit. 31.01.2023] doi: <https://www.datascienceontology.org/help>

[13] J. Polowinski and M. Voigt. “VISO: a shared, formal knowledge base as a foundation for semi-automatic infovis systems”. In: CHI ’13 Extended Abstracts on Human Factors in Computing Systems, Paris France. 2013, pp. 1791-1796, doi: 10.1145/2468356.2468677

[14] G. Antoniou and F. Van Harmelen. “A semantic Web primer”. 2nd ed. Cambridge, Mass: MIT Press (Cooperative information systems) 2008, pp. 264

[15] Ontologies and Semantic Web: Working with Ontologies, [online] [cit. 2023-01-13] doi: <https://www.obitko.com/tutorials/ontologies-semantic-web/working-with-ontologies.html>

[16] A. Gomez-Perez and O. Corcho. “Ontology languages for the Semantic Web”. IEEE Intelligent Systems 17. 2002, pp. 54-60

[17] P. Bednár, J. Ivančáková and M. Sarnovský. “Semantic automatiozation of the data-analytical processes”. In: International Symposium on Applied Computational Intelligence and Informatics. 2022

[18] M. Sarnovsky, P. Bednar, and M. Smatana. “Cross-Sectorial Semantic Model for Support of Data Analytics in Process Industries”. Processes 7, No. 5: 281, 2019, doi: https://doi.org/10.3390/pr7050281

[19] A. Miles and S. Bechhofer. SKOS Simple Knowledge Organization System Reference. W3C Recommendation. World Wide Web Consortium. 2009, doi: https://www.w3.org/TR/skos-reference/

[20] M. Sarnovsky and J. Marcinko. “Adaptive Bagging Methods for Classification of Data Streams with Concept Drift”. Acta Polytechnica Hungarica. 2021. pp. 47-63, doi: 10.12700/APH.18.3.2021.3.3

[21] J. Bruijn, H. Lausen, A. Polleres, and D. Fensel. “The web service modelling language WSML: An overview”. 2006, pp. 604, doi: 10.1007/11762256_43

[22] Ch.Molnar, “Interpretable Machine Learning”, A Guide for Making Black Box Models Explainable, 2023, [online] [cit. 2023-03-02] doi: <https://christophm.github.io/interpretable-ml-book/>

[23] P. Šatala, P. Butka, A. Samaiev and P. Levická. “Cueing of Parkinson’s Disease Patients by Standard Smart Devices and Deep Learning Approach”. Acta Polytechnica Hungarica. 2023, pp. 165-184

[24] S.Pittet. “What is code coverage?” [online] [cit. 2023-01-29] doi: <https://www.atlassian.com/continuous-delivery/software-testing/code-coverage>

[25] Breast Cancer Wisconsin (Diagnostic) Data Set. [online] [cit. 2023-02-25] doi: <https://www.kaggle.com/uciml/breast-cancer-wisconsin-data>

[26] J. Ivančáková, F. Babič and P. Butka. “Comparison of different machine learning methods on Wisconsin dataset”. 2018 IEEE 16th World Symposium on Applied Machine Intelligence and Informatics. Kosice and Herlany, Slovakia, 2018, pp. 173-178, doi: 10.1109/SAMI.2018.8324834

[27] P. P. Bakucz and J. Z. Szabó. “Determining the Embedded Key Performance Indicator (KPI), based on a Fuzzy FxLMS Algorithm”. Acta Polytechnica Hungarica. 2021, pp. 127-139, doi: 10.12700/APH.18.9.2021.9.8

[28] A. Gupta and R. Gupta. ALL Challenge dataset of ISBI 2019 The Cancer Imaging Archive. Data set [online] [cit. 2023-02-25] doi: <https://doi.org/10.7937/tcia.2019.dc64i46r>

[29] V. Diaz and G. Rodríguez. “Machine Learning for Detection of Cognitive Impairment”. Acta Polytechnica Hungarica. pp. 195-213, 2022, doi: 10.12700/APH.19.5.2022.5.10