Ontology-Based Data Mining Workflow Construction

Man Tianxing[1(✉)], Sergey Lebedev[3], Alexander Vodyaho[3], Nataly Zhukova[2,3], and Yulia A. Shichkina[3]

[1] ITMO University, St. Petersburg, Russia
[2] St. Petersburg Federal Research Center of the Russian Academy of Sciences, St. Petersburg, Russia
[3] St. Petersburg State Electrotechnical University "LETI", St. Petersburg, Russia

Abstract. Currently, Data mining is applied in various domains. Many data science researchers are confused on which algorithms are suitable for the context. Hundreds of the operators/algorithms are combined within complex data mining workflows. A data mining assistant can significantly improve the efficiency of workflow construction. In our previous work, we constructed data mining ontologies based on "semantic meta mining" to support the selection of algorithms and operators for solving data mining tasks. But the use of such ontologies is still unfriendly. Strict query syntax still plagues many users. This paper proposes an interactive interface based on the reasoning mechanism to help users generate queries and build suitable data mining workflows.

Keywords: Meta-learning · Data mining · Semantic meta mining · Ontology

1 Introduction

Data mining (DM) technology allows extracting knowledge from real data in various domains. However, the raw data in real life is always complicated and messy. Researchers don't know which method can be used in context and often resort to trial and error. The task to construct the DM workflow for data processing in various domains is a complex one due to the complexity of data and the diversity of analysis methods.

To solve the DM algorithm selection task, some DM assistants were developed to support the DM workflow generation. As one type of such technique, semantic meta-mining (SMM) [1] is proposed for DM optimization through algorithm/model selection. It provides a suitable description framework to clarify the complex relationships between tasks, data, and algorithms at different stages in the DM process. SMM mines DM metadata through using knowledge bases that contain expertise on DM domain. This expertise should be stored in a machine-interpretable format, so it can be automatically extracted and applied for solving new tasks.

Most existing SMM systems support solving general DM problems. Panov et al. proposed a data mining ontology OntoDM, which includes formal definitions of basic DM entities [2]. Then, Panov et al. developed a separate ontology module, named OntoDT,

© Springer Nature Switzerland AG 2021
O. Gervasi et al. (Eds.): ICCSA 2021, LNCS 12956, pp. 417–431, 2021.
https://doi.org/10.1007/978-3-030-87010-2_31

for representing the knowledge about data types [3]. Hilario et al. present the data mining optimization ontology (DMOP), which provides a unified conceptual framework for analyzing data mining tasks, algorithms, models, datasets, workflows, and performance metrics, as well as their relationships [4]. There are several other data mining ontologies currently existing, such as the Knowledge Discovery (KD) Ontology [5], the OntoDTA ontology [6], the KDDONTO Ontology [7], which are based on similar ideas. These ontologies present the description of DM knowledge in a general situation, rather than in a specific domain.

In our previous work, we proposed a meta mining framework for data processing [8]. A DM core ontology was constructed by integrating the existing DM ontologies. The users can search the suitable algorithms according to the task requirements and the data characteristics and execute them in the corresponding applications.

Three DM ontologies are constructed to describe the knowledge about DM workflow:

- The DM core ontology describes the general DM workflow, including the algorithms, process, characteristics, goals, etc.
- The DM process ontology supports the management of DM processes, which assumes defining steps of the processes and selecting algorithms taking into account the input and the output data.
- The DM dataset characterization ontology describe the data characteristics and the corresponding definitions.

The application of the ontologies is based on building queries using OWL syntax. However, in actual use, the framework often loses its effect due to strict query syntax. Researchers who are non-computer professionals do not know how to construct the corresponding queries according to their needs.

In this article, we propose a rule-based interactive interface to support the ontology-based DM workflow building. We use DL query to search for answers in the DM ontologies [9]. The interactive interface is developed based on Drools engine [10]. It converts the user's request into the corresponding DL queries and uses the query results to plan DM workflow.

2 Related Work

2.1 CRISP-DM

To build DM workflows effectively and correctly, it is important to have a structured framework to follow. Today some frameworks are more frequently used than others. CRISP-DM [11] is a DM framework that defines six different phases: business understanding, data understanding, data preprocessing, modeling, evaluation, and deployment. KDD [12] is a DM framework that contains detailed descriptions of CRISP-DM phases. In KDD, more detailed phases are provided: developing and understanding of the application, creating a target dataset, data cleaning and preprocessing, data transformation, choosing a suitable data mining task, choosing a suitable data mining algorithm, employing data mining algorithm, interpreting mined patterns, and using discovered knowledge. Unlike CRISP-DM and KDD, SEMMA [13] focuses mostly on data management. The model of data mining consists of five phases: sample, explore, modify, model, and access. It doesn't consider the problem of understanding and the deployment of the generated

models. At present, all the DM process frameworks are used manually to support the DM tasks solving. The premise of automatic workflow generation is to query for the suitable process from a DM knowledge base. By now the CRISP-DM framework is the most suitable framework for building DM processes, as it is more complete and has a more concise description compared to the others.

2.2 Meta Mining Framework for DM Workflow Construction

In our previous work [8], we proposed a meta mining framework for DM workflow construction based on CRISP-DM, which is shown in Fig. 1.

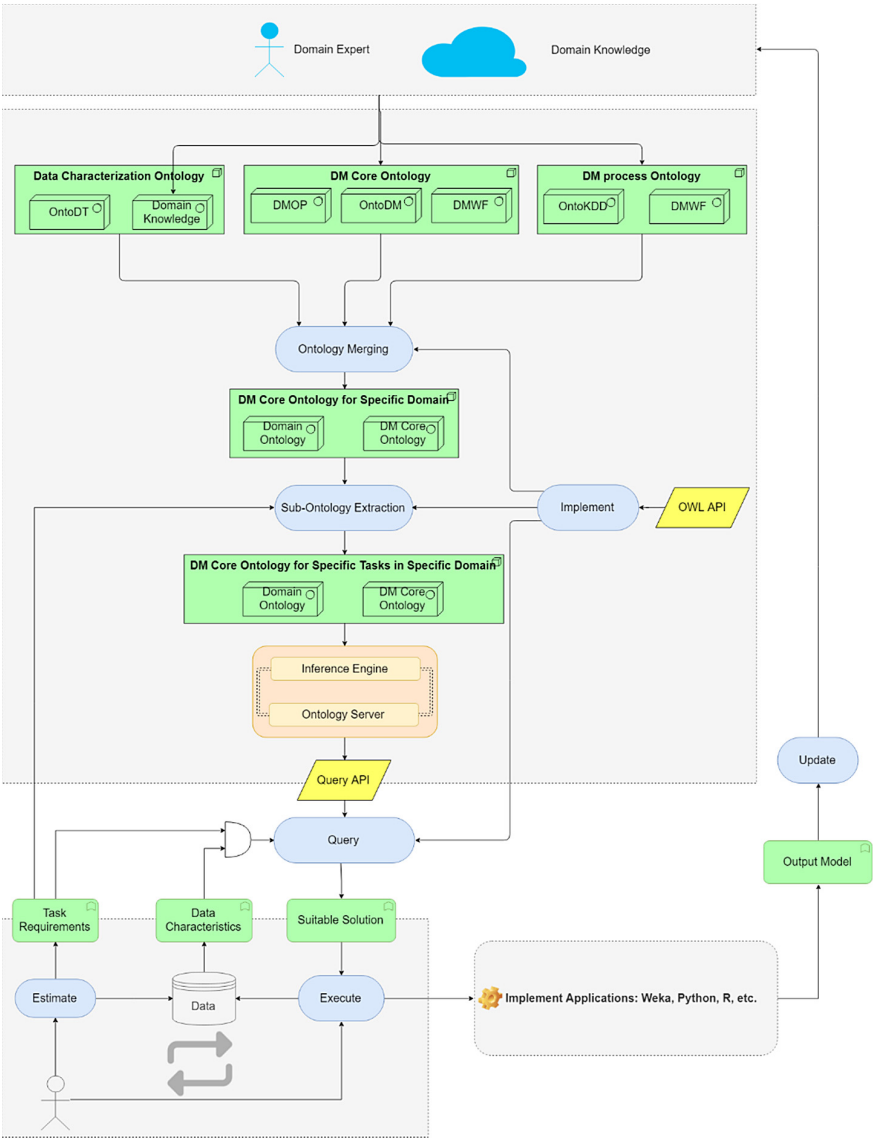


Fig. 1. The meta mining framework for DM workflow construction.

The data characterization ontology and DM core ontology describe the datasets and DM domain knowledge. The DM process ontology describes the hierarchy of DM processes, at the top of which are abstract processes that are detailed at lower levels. In the proposed framework, the ontologies are merged and extracted via an ontology merging method and sub-ontology extraction method [14]. The inference engine and ontology server provide the query API for the algorithm selection server. The users can search the suitable algorithms according to the task requirements and the data characteristics and execute them in the corresponding applications. The result can be used to update the domain knowledge.

The query server requires construction of queries using description logic (DL). The DL queries have to be formatted according to strict Manchester OWL syntax [15]. Most users of the framework are non-computer professional researchers and they cannot construct queries manually. This makes the framework hard to be applied in practice. In this article, we focus on the inference engine and ontology server of the proposed framework. We propose a rule-based solution to support building queries for the query server. The proposed interactive interface allows to covert the users' requests to DL queries automatically.

2.3 DM Ontologies

The premise of the meta mining framework is the construction of the DM ontologies. We proposed three DM ontologies for the DM workflow construction: DM core ontology, Data characterization ontology, and DM process ontology.

DM Core Ontology

We built a DM Core ontology to describe the general DM knowledge, including the algorithms, process, characteristics, goals, etc. The DM core ontology is constructed by integrating some existing DM ontologies (OntoDM [2], DMOP [4], and DMWF [16]).

In the DM core ontology, dataset characteristics and task requirements are defined as the conditions for algorithm selection. Suitable algorithms, output models, and processes are selected based on these conditions. Description of the algorithms contains information that allows defining their parameters and executing them [14]. The main classes of the DM core ontology are presented in Fig. 2.

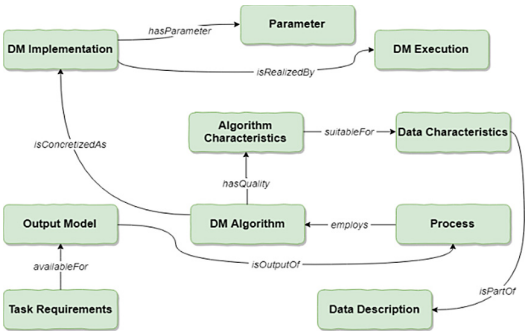


Fig. 2. The main classes in DM core ontology.

DM Data Characterization Ontology

We proposed a DM data characterization ontology to characterize the DM datasets with three categories of the characteristics:

– standard measures
– statistics measures
– information-theoretic measures

The ontology is built by integrating the existing DM ontologies (OntoDT and IAO [23]) and representing the abstract description of the dataset characteristics, which is shown in Fig. 3.

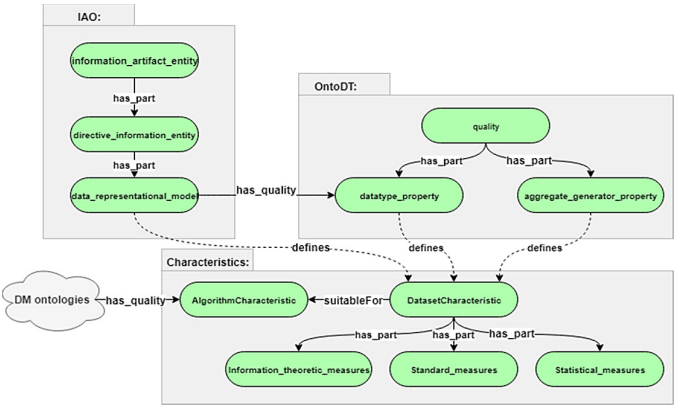


Fig. 3. The Data Characterization ontology.

The notions of domain data characteristics are described both in DM Core ontology and data characterization ontology as ontology classes. The definitions of the notions in the form of the axioms are given only in the data characterization ontology. when dealing with specific domain tasks these two ontologies are merged.

DM Process Ontology

The proposed DM process ontology has a hierarchical structure, which is shown in Fig. 4. Four layers are defined:

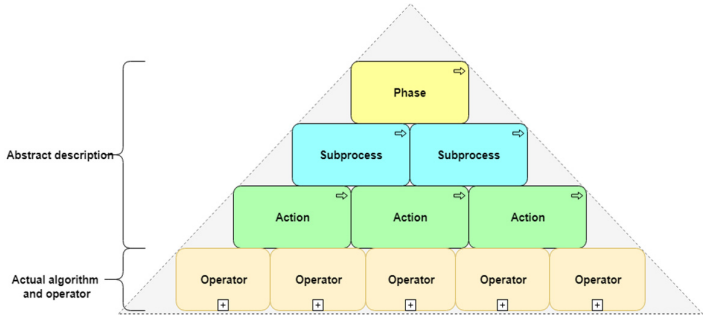


Fig. 4. The hierarchical structure of the DM process ontology.

– Phase
– Subprocess
– Action
– Operator

The first three layers describe the abstract processes and the last one presents the concrete implemented operators/algorithms [17].

The proposed ontology is constructed with a hierarchical structure and integrating OntoKDD [18], which provides the entities of DM process, ad DMWF [5], which provides the concrete operators, input and output objects.

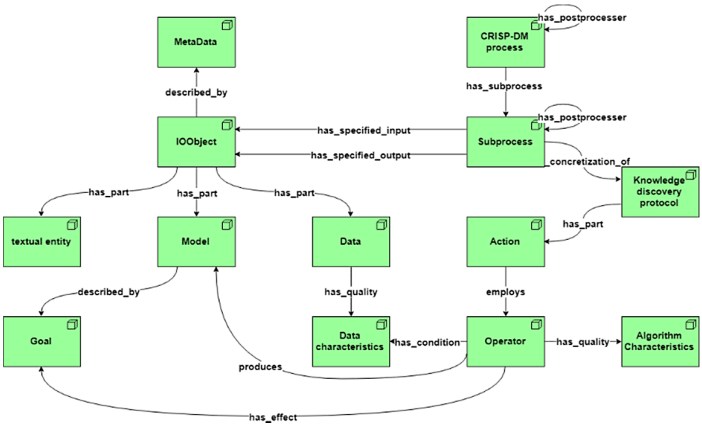


Fig. 5. The hierarchical DM process ontology

The DM hierarchical process ontology is presented in Fig. 5.

- At "Phases" layer, the general phases of DM process according to CRISP-DM framework are defined.
- At "Subprocesses" layer, the subprocesses are described as the decompotion of the "Phases." And the input and output of the process are also presented at this layer.
- At "Actions" layer, the specific activities in DM process are defined.
- At "Operators" layer, the concrete implemented operators/algorithms are described. The order of the actions and operators is defined by the restriction of their input and output.

3 Rule-Based Interactive Interface

The proposed DM ontologies provide the framework based on knowledge for support of DM workflows. Users can construct queries to ontologies using DL for building workflows. But the majority of the users of these ontologies are non-computer researchers. It is hard to construct DL queries for them. Thus, we propose a rule-based interactive interface that supports query generation. DL query is applied to the reasoned ontologies. The proposed rule-based interface is built on the base of Drools.

3.1 DL Query

For the application of the ontologies, many querying tools were developed for querying on a classified ontology [25] easily. In protégé [19], a DL query tab was developed as a plugin. DL query uses Manchester syntax, which is a user-friendly language.

The DL queries are expressed using constructs ("and," "some," "exactly" and so on). The subclasses, which match the conditions, are obtained. An example of DL query is as followings:

***Q**: Who is Matthew?*
***DL query**: hasGivenName value "Matthew"*
or
Person and hasGivenName value "Matthew"

Due to the ease of understanding of DL query, we choose it as the way to query the ontology. But in our previous work, we found that even though a DL query is simple to understand when it has been written, it is tricky to write the one.. This still makes some users give up ontologies.

3.2 Drools

A production system is based on using if-else rules: it executes those rules whose conditions are matched by some facts. So when the task is changed, the rules need to be modified, which can be difficult. In addition, it may cause errors due to neglecting the semantic relationship of context when modifying the rules, so the risk of modification is very high. So there is a need for a rule management system.

Drools, which is JBoss rules, is an open-source rule engine written in Java language. It is a hybrid reasoning system - it supports forward as well as backward reasoning [link on the official documentation]. Rules are stored in the production memory, while facts are maintained in the working memory. The production memory remains unchanged during an analysis session, i.e. no rules are added or removed, or changed. The contents of the working memory can change. Facts may be modified, removed or added, by executing rules or received from external sources. After a change in the working memory, the inference engine is triggered and it determines which rules become "true" for the given facts. If there are multiple selected rules, their execution order will be managed via the Agenda, using a conflict resolution strategy [21].

With the advantage of the Rete algorithm [20], the rules engine Drools is very convenient to manage the rules, it can also meet the demand of higher frequency of rule change. Drools is also more efficient than traditional logical inference for the application scenario with huge rule variables. In our case, as the DM ontologies are extended with new algorithms, the DL queries should be updated correspondingly. Thus, we choose Drools as the rule engine to develop the interactive interface to generate the needed DL queries for users.

3.3 Framework of the Rule-Based Interactive Interface

In this article, we propose a rule-based interactive interface developed using Drools. It receives the users' request and generates the needed DL queries for querying DM ontologies. The framework is shown in Fig. 6.

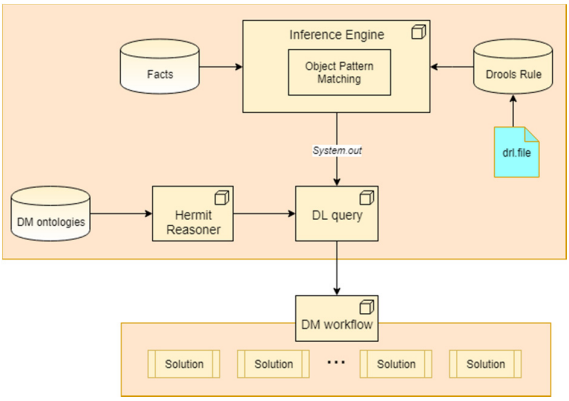


Fig. 6. The framework of the rule-based interactive interface.

The core of the Drools suite is an advanced Inference Engine that uses an improved Rete algorithm for object pattern matching. Rules are stored in the production memory, while facts are maintained in the working memory. The production memory remains unchanged during an analysis session, i.e. no rules are added or removed, or changed. The contents of the working memory can change. Facts may be modified, removed or added, by executing rules or received from external sources. After a change in the working memory, the inference engine is triggered and it determines which rules become "true" for the given facts. If there are multiple selected rules, their execution order will be managed via the Agenda, using a conflict resolution strategy.

In our case, the users' requests define the input facts and the needed DL queries are output. The queries are generated using preset rules. The output DL queries are formatted as the rules with two parts: 1. The fixed contents is presented in the form "String"; 2. The variables are presented as the name of the property of the class. An example is given in Fig. 7:

```
rule "query for actions which are suitable for the specific data characteristics"
    when
        $a : Action(state == true)
        $dc : DataCharacteristics()
    then
        System.out.println("query for actions which are suitable for "
                           + $dc.getName()
                           + " with the following query:"
                           + " Action and suitableFor "
                           + $dc.getName());
end
```

Fig. 7. An example of the Drools rules

The proposed DM ontologies are served as part of the knowledge base. After inferenced by the reasoner (we employ the reasoner "Hermit" [24]), DL queries on these ontologies are generated according to the users' requests. By triggering a preset Drools rule "starter()", the query for basic phases of DM workflow is generated. Users can query for suitable algorithms/operators/solutions according to the task requirements and the data characteristics and execute them in the corresponding operating environment at each phase of data processing. The corresponding DM workflow is generated as the sequence of the suitable algorithms/operators/solutions.

4 Rule-Based DM Workflow Construction

With the help of the Drools, the interactive interface supports the DM workflow construction.

The implementation of the DM workflow construction is based on defining specific queries on each layer of the DM process ontology. The queries are generated based on the designed Drools rules.

The functions of the layers are defined as following:

- "Phase" layer is used to define the general sequence of the DM process. It follows the CRISP-DM framework and defines six phases. The sequence of the phases, which is shown in Fig. 8, is specified by the property "hasPostprocess".

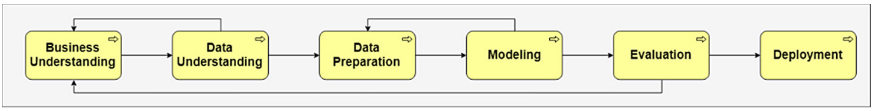


Fig. 8. The general sequence of the DM workflow defined in the DM process ontology

- "Subprocess" layer is used to decide whether the process is necessary for solvingspecific tasks. Each subprocess has specific input and output which are described in the class "IOObject", for example a precondition of a subprocess "handling_missing_attribute_values" is "has_specific_input some attributeWithMissingValues". "attributeWithMissingValues" is a type of dataset which is described in "IOObject". First, data properties (such as number of samples, number of attributes, number of missing values for each attribute, etc.) are queried. Then, the corresponding data characteristics/defects are obtained from the data characterization ontology according to the data properties.
- "Action" layer is used to decide which actions defined at the 'subprocess' layer are suitable for the data with characteristics/defects. The relations between the entities action and data characteristics are defined by the property "suitableFor".
- "Operator" layer is similar to the "Action" layer. But it decides which algorithms are suitable for the data with the defined characteristics/defects. The former determines

the abstract conceptual process, and the latter determines the specific algorithm or operator, for example: process "Imputation" and algorithm "KNN_Imputation_ED (K Nearest Neighbor algorithm with Euclidean distance)".

Algorithm Pseudocode of the DM workflow planning

```
//Notion
The query module is the process: users input the requests in Drools
and get DL query, then run the DL query on the DM ontologies. The
results are a list of ontology entities or an entity.
//init
Map<entity, entity> workflow = new HashMap();
//Task starts
List<entity> phase = query.starter();
//Get the subprocesses for each phase
For (entity p : phase){
    List<entity> subprocess = query.getSubprocess(p);
//Get the considered data property and corresponding data
characteristics for each subprocess
    For (entity sp : subprocess){
        List<entity> dataProperty = query.getDataproperty(sp);
        For (entity dp : dataProperty){
            List<entity> dataCharacteristics =
query.getDataCharacteristics(dp);
            For (entity dc : dataCharacteristics){
                While (query.availableFor(dc, sp)){
                    Entity Action = query.getAction(dc);
                    Entity Operater = query.GetAction(dc);
                    Workflow.add((Action, Operator));
                }
            }
        }
    }
}
Return workflow;
```

Fig. 9. The pseudocode of the DM workflow construction process.

The pseudocode of the DM workflow construction process is presented in Fig. 9. The query module supports the following process: users input the requests in Drools program and get DL query, then run the DL query on the DM ontologies. The query results are a list of ontology entities or an entity. An example of such process is presented in Fig. 10.

At the beginning, the starter() function is called, the basic phases of the DM workflow are returned. Each phase has a list of the subprocesses. By querying data characterization ontology with the data basic properties, the data characteristics are obtained as the base for algorithm selection. If the subprocesses are available for the defined data characteristics, the suitable actions and operators/algorithms are obtained by querying on the ontology and added to the workflow. They are stored in the Map data structure with two



Fig. 10. An example of generating the DL query for defining suitable actions.

variables: key - process and value - operator. The former is the abstract description of the process, and the latter is the concrete employed algorithm or operator.

5 Case Study

As a case study, we consider the Labeled Faces in the Wild (LFW) people dataset [22], which is a database of face photographs designed for studying the problem of unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set.

We have extracted the data of the persons that have at least 70 photos and used them as our dataset and built a data mining workflow using the proposed framework. The extracted dataset has 1288 images and 1850 attributes (50*37 pixels) for 7 persons. We have considered the task to train the classification model for face recognition.

In the beginning, we start up by inputting a task name and triggering the Drools engine, the general sequence of the DM workflow which is shown in Fig. 8 has been returned. Then it has become possible to query on each phase. Following the guide of the interactive interface, we have obtained knowledge about the required input and the queries that should be performed.

The processes at some phases are subjectively determined, such as “Determine Business Objectives”, “Verify Data Quality”, etc. However, we need to consider the data characteristics to decide the suitable algorithms. In our case, according to the rules for phase “Data Preparation”, we have considered the following data properties: number of samples and number of attributes. The corresponding data characteristics are queried based on the data properties. Since the number of attributes that equals 1850 is more than the number of samples that equals 1288, we have got the data characteristic “HighDimensionality”.

As the Fig. 11 shows, we have input the data characteristics (1), the query for suitable DM process has returned "Action and suitableFor some HighDimensionality" (2). And the generated query has been performed on the DM ontology, the queried result is "DimensionalityReduction" (3) and with the similar process, the employed algorithm PCA (Principal Component Analysis) is obtained.

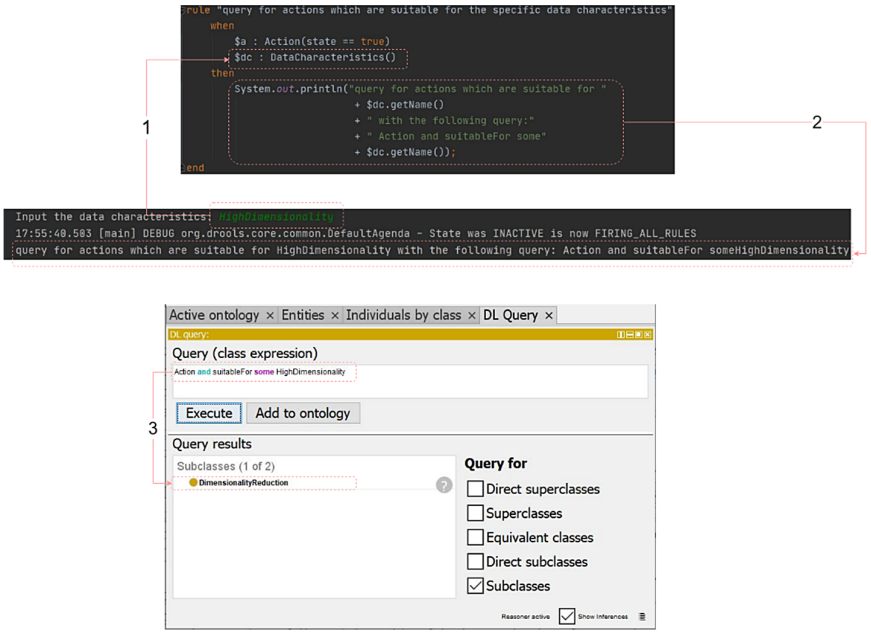


Fig. 11. The process of query for the suitable action for characteristics "HighDimensionality"

The similar procedure has been performed at the "modeling" phase. We have obtained the suitable classification algorithm for "HighDimensionality" that is SVC (Support Vector Classification algorithm).

Thus, we have followed the guide of the interactive interface step by step. The corresponding DM workflow for the LFW dataset that is shown in Fig. 12 has been built. The subprocesses and the actions are the abstract description of the DM workflow, and the algorithms (SVC and PCA) are the concrete implementation of the DM workflow.

In order to verify the effectiveness of the obtained workflow, we have constructed workflows using alternative algorithms and compared the results. The result of comparison is presented in Fig. 13. Three metrics are considered: precision, recall, and F1-score. Other workflows include: SVC, MLP (MutiLayer Perception), MLP + PCA, NB (Naive Bayes), NB + PCA, DT (Decision Tree), DT + PCA. The workflow SVC + PCA that was constructed using Drools has the best performance: 0.87, 0.86, 0.85. Usage of PCA in combination with SVC allowed reduce the dimension of the dataset attributes, and thus archive higher performance.

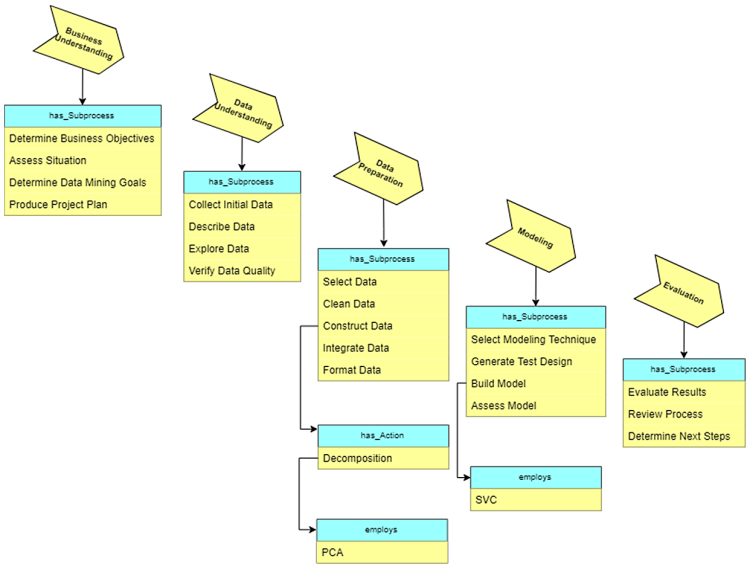
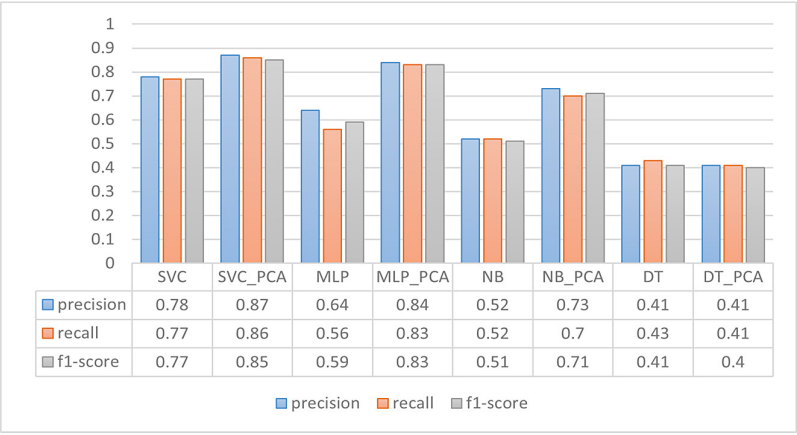


Fig. 12. The workflow for LFW dataset processing



	SVC	SVC_PCA	MLP	MLP_PCA	NB	NB_PCA	DT	DT_PCA
precision	0.78	0.87	0.64	0.84	0.52	0.73	0.41	0.41
recall	0.77	0.86	0.56	0.83	0.52	0.7	0.43	0.41
f1-score	0.77	0.85	0.59	0.83	0.51	0.71	0.41	0.4

Fig. 13. The comparison results of obtained workflow with alternative workflows

6 Conclusions

Addressing data mining workflow construction, we have built an ontology-based meta mining framework. In previous work, we found that the strict syntax makes users confused on using the framework. In this article, we propose a rule-based interactive interface based on Drools, which converts the users' requests to DL queries automatically.

The DM workflow construction is based on the DM process ontology that has hierarchical structure. It assumes that the workflows are constructed at different layers. At

"Phase" layer, the basic sequence of the DM phases is specified. At "Subprocess" layer, the available processes are determined. Then, at "Action" and "Operator" layer, the suitable actions and corresponding algorithms/operators are stored as the DM workflow for the specific tasks solving.

A use case is presented in this article, we construct a DM workflow for the LFW dataset. The PCA algorithm is selected at the "Data preparation" phase and the SVC algorithm is selected as the modeling algorithm. Compared with other workflows, the constructed workflow archives higher performance.

Acknowledgments. «The paper was prepared in Saint- Petersburg Electrotechnical University (LETI), and is supported by the Agreement № 075–11-2019–053 dated 20.11.2019 (Ministry of Science and Higher Education of the Russian Federation, in accordance with the Decree of the Government of the Russian Federation of April 9, 2010 No. 218), project «Creation of a domestic high-tech production of vehicle security systems based on a control mechanism and intelligent sensors, including millimeter radars in the 76–77 GHz range»

References

1. Hilario, M., et al.: Ontology-based meta-mining of knowledge discovery workflows. In: Meta-Learning in Computational Intelligence, pp. 273–315. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20980-2_9
2. Panov, P., Džeroski, S., Soldatova, L.: OntoDM: an ontology of data mining. In: 2008 IEEE International Conference on Data Mining Workshops. IEEE (2008)
3. Panov, P., Soldatova, L.N., Džeroski, S.: Generic ontology of datatypes. Inf. Sci. **329**, 900–920 (2016)
4. Keet, C.M., et al.: The data mining optimization ontology. J. Web Semant. **32**, 43–53 (2015)
5. Žáková, M., et al.: Automating knowledge discovery workflow composition through ontology-based planning. IEEE Trans. Autom. Sci. Eng. **8**(2), 253–264 (2010)
6. Benali, K., Rahal, S.A.: OntoDTA: ontology-guided decision tree assistance. J. Inf. Knowl. Manag. **16**(03), 1750031 (2017)
7. Diamantini, C., Potena, D., Storti, E.: Kddonto: an ontology for discovery and composition of KDD algorithms. In: Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery (SoKD'09), pp. 13–24 (2009)
8. Tianxing, M., et al.: A meta-mining ontology framework for data processing. Int. J. Embedded Real-Time Commun. Syst. (IJERTCS) **12**(2), 37–56 (2021)
9. Pan, J.Z., Thomas, E., Zhao, Y.: Completeness guaranteed approximations for OWL-DL query answering. Description Logics **477** (2009)
10. Proctor, M.: Drools: a rule engine for complex event processing. In: Schürr, A., Varró, D., Varró, G. (eds.) AGTIVE 2011. LNCS, vol. 7233, pp. 2–2. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34176-2_2
11. Wirth, R., Hipp, J.: CRISP-DM: towards a standard process model for data mining. In: Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, vol. 1. Springer, London (2000)
12. Brachman, R.J., Anand, T.: The process of knowledge discovery in databases: a first sketch. In: KDD Workshop, vol. 3 (1994)
13. Shafique, U., Qaiser, H.: A comparative study of data mining process models (KDD, CRISP-DM and SEMMA). Int. J. Innov. Sci. Res. **12**(1), 217–222 (2014)

14. Tianxing, M., Stankova, E., Vodyaho, A., Zhukova, N., Shichkina, Y.: Domain-Oriented Multilevel Ontology for Adaptive Data Processing. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12249, pp. 634–649. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58799-4_46
15. Horridge, M., et al.: The Manchester OWL Syntax. OWLed, vol. 216 (2006)
16. Doukas, C., Chatziioannou, A., Maglogiannis, I.: Intelligent planning of biomedical image mining workflows. In: Proceedings of the 10th IEEE International Conference on Information Technology and Applications in Biomedicine. IEEE (2010)
17. Tianxing, M., et al.: A hierarchical data mining process ontology. In: 2021 28th Conference of Open Innovations Association (FRUCT). IEEE (2021)
18. Panov, P., Soldatova, L., Džeroski, S.: OntoDM-KDD: ontology for representing the knowledge discovery process. In: Fürnkranz, J., Hüllermeier, E., Higuchi, T. (eds.) DS 2013. LNCS (LNAI), vol. 8140, pp. 126–140. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40897-7_9
19. Noy, N.F., et al.: Protégé-2000: an open-source ontology-development and knowledge-acquisition environment. In: AMIA... Annual Symposium proceedings. AMIA Symposium, vol. 2003. American Medical Informatics Association (2003)
20. Liu, D., Gu, T., Xue, J.-P.: Rule engine based on improvement rete algorithm. In: The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding. IEEE (2010)
21. Yang, P., et al.: An intelligent tumors coding method based on drools. J. New Media **2**(3), 111 (2020)
22. Huang, G.B., et al.: Labeled faces in the wild: a database for studying face recognition in unconstrained environments. In: Workshop on Faces in'Real-Life'Images: Detection, Alignment, and Recognition (2008)
23. Information Artifact Ontology (IAO) web page. http://www.obofoundry.org/ontology/iao.html
24. Glimm, B., et al.: HermiT: an OWL 2 reasoner. J. Autom. Reasoning **53**(3), 245–269 (2014)
25. DL Query tab. https://protegewiki.stanford.edu/wiki/DLQueryTab