Data Science and Engineering (2024) 9:409–430
https://doi.org/10.1007/s41019-024-00262-x

RESEARCH PAPER

Towards ML Models' Recommendations

Lara Kallab[1] · Elio Mansour[2] · Richard Chbeir[3]

Received: 18 February 2024 / Revised: 5 August 2024 / Accepted: 3 September 2024 / Published online: 28 October 2024

Abstract
Artificial Intelligence encompasses a range of technologies that replicate human-like cognitive abilities through computer systems, enabling the execution of tasks associated with intelligent beings. A prominent way to achieve this is machine learning (ML), which optimizes system performance by employing learning algorithms to create models based on data and its inherent patterns. Today, a multitude of ML models exist having diverse characteristics, including the algorithm type, training dataset, and resultant performance. Such diversity complicates the selection of an appropriate model for a specific use case, answering user demands. This paper presents an approach for ML models retrieval based on the matching between user inputs and ML models criteria, all described in a semantic ML ontology named SML model (Semantic Machine Learning model), which facilitates the process of ML models selection. Our approach is based on similarities measures that we tested and experimented to score the ML models and retrieve the ones matching, at best, user inputs.

Keywords Machine learning model · Supervised learning · Ontology · User input · Similarities criteria · ML models and user inputs alignment · ML models retrieval

1 Introduction

In contemporary times, artificial intelligence (AI) has emerged as a turning point across diverse domains, including social, commercial, and industrial ones, such as speech recognition, medical diagnosis, autonomous vehicles, and building automation [1]. Essentially, AI represents a computer system crafted to emulate human intelligence, leveraging data from myriad sources and systems to make decisions and acquire knowledge from the outcomes. Machine learning (ML) acts as an instantiation of AI, allowing computers to learn from data without explicit programming [2]. Its primary focus lies in constructing models capable of learning from historical data, discerning meaningful relationships and patterns within the data [3], and autonomously making logical decisions with minimal or no human intervention. ML automates the creation of analytical models by utilizing diverse forms of numerical information, including numbers, words, images, and more.

In the world of Machine learning (ML), there exists a plethora of models available for users to adopt and reuse (particularly for non-experts), reducing the need to create new models for each task. In fact, the data necessary to create the models are, often, not available, nor the machines/processors that are used to train the models as they require a lot of performance and calculation time, hence the importance of reusing and adapting existing ML models to users needs. A learning model comes with its own set of specifications and applications, including distinct algorithm types (e.g., Linear Regression or Bayes Classifier [4]), the training dataset utilized, the application domain (e.g., finance, travel, and transportation), and the model performance. This diversity adds complexity to the task of selecting an apt model that meets user demands, especially for non-expert users with limited or no ML knowledge. Selecting the appropriate model for a specific use case, in alignment with user needs, holds paramount importance. The more closely a machine learning model matches a given case and fulfills user requirements, the more adeptly it can identify data

✉ Lara Kallab
larakallab@gmail.com

Elio Mansour
emansourfr@gmail.com

Richard Chbeir
richard.chbeir@univ-pau.fr

[1] Huntington Beach, CA 92646, USA

[2] Brussels, Belgium

[3] University Pau & Pays Adour, 64600 Anglet, Pyrénées-Atlantiques, France

features or patterns, and meet user demands. This translates into improved decision-making, offering more accurate analyses and forecasts. For instance, using a regression model trained on winter season data in France to predict information relevant to a summer season (as it can be requested by the user) is likely to yield poor results and dissatisfy the user. This discrepancy arises because the model's learning is based on a different data set pattern (in terms of season) that does not correspond to the user request. Hence, it becomes imperative to properly describe ML models and represent their characteristics, along with the specifications of user needs (or inputs), in a semantic format. This facilitates the understanding of how and where each model can be optimally used or adopted, taking into account user requests. Such an approach enables the comparison, evaluation, and retrieval of the most suitable model(s) for a specific application scenario, effectively addressing user requirements.

In the literature, numerous models, approaches, and reviews represent the characteristics, applicability, and performance of machine learning models. However, these works exhibit several limitations. Notably, a significant portion of them, including references [5–9], inadequately details the datasets used for model training and testing. Moreover, a majority of the works, specifically [7–10], lacks in considering models application domain and operational performance. Additionally, none of these sources comprehensively explores aspects such as models usability, their contexts (e.g., temporal and spatial contexts), and various levels of models metadata (e.g., ML model metadata, algorithm metadata, dataset metadata). Considering these essential ML criteria, and matching them with user demands, is crucial for enhancing the selection of ML models that satisfy, at best, user needs. After recognizing the limitations in existing representations of machine learning models, which are necessary for the understating of their functioning and correct use, their applications, their evaluation and comparison, we present in this paper a framework for ML models representation and retrieval, as depicted in Fig. 2. The framework is based on an ontology-based model, named "SML", for Semantic Machine Learning description. SML describes machine learning models characteristics through a vocabulary understandable by both humans and machines, facilitating the comprehension of their behavior and use for a given context. SML also describes user input specifications, which can be compared to ML models characteristics to identify the best models matching user needs. In this context, the presented framework includes a formally defined user request (part of which is embedded in SML), and proposes similarity calculations between the ML models characteristics and user inputs to retrieve the most suitable ML models aligning with users needs. The similarity measures are based mainly on four criteria: (1) Feature, (2) Feature Value Type, (3) Temporal Context, and (4) Spatial Context. As an ontology model [11], SML gives the same meaning to the specified ML model characteristics and user specifications. It eases the storage, integration, and sharing of ML knowledge and user inputs across diverse organizations and platforms, fostering both syntactic and semantic interoperability.

The rest of the sections are organised as follows. In Sect. 2, a scenario is presented to highlight the motivation behind the usability and applicability of our work. Section 3 provides a review of existing related works, emphasizing the added value of our solution. Detailed specifications of our proposed semantic machine learning model ontology, along with the defined similarities measures between ML characteristics and user inputs, are outlined in Sect. 4. Tests and experiments proving the efficiency and the performance of our approach are giving in Sect. 5. Lastly, Sect. 6 offers a summary of the work and delves into potential directions for future research.

2 Motivating Scenario

In order to show the motivation behind our proposal, let us consider the depicted Smart City scenario in Fig. 1. The environment is extensively covered by a Wireless Sensor Network (WSN), gathering diverse data (e.g., CO_2 emissions, lighting conditions, noise levels, energy consumption, temperature) from the city. A team of experts has been assigned to monitor, analyze, and forecast elements within the city, aiming to transform it into a smart, proactive, safe, and healthy habitat for its residents. These experts, each with unique skills, focus on forecasting and analyzing data within their respective domains. Fig. 1 provides some examples: (i) environmental experts predict noise, air, and water pollution levels for a healthier city; (ii) road safety experts forecast traffic congestion, risky conditions, and road deterioration to prevent accidents; (iii) weather experts predict rising temperatures and extreme conditions, proactively disseminating crucial information; and (iv) energy experts analyze and predict energy consumption and production for a greener, eco-friendly city.

In this collaborative environment, team members frequently collaborate on interdisciplinary projects. More importantly, they all require the generation, training, testing, and deployment of prediction models that utilize the collected data to provide necessary forecasts. In this dynamic and collaborative setting, the potential for a substantial number of machine learning models to be quickly developed is evident.

To sustain this collaborative workspace, prevent isolated analysis, and establish a decision-making process rooted in collective intelligence and shared insights, the team requires a system capable of storing and retrieving ML models for each new application use case. This approach

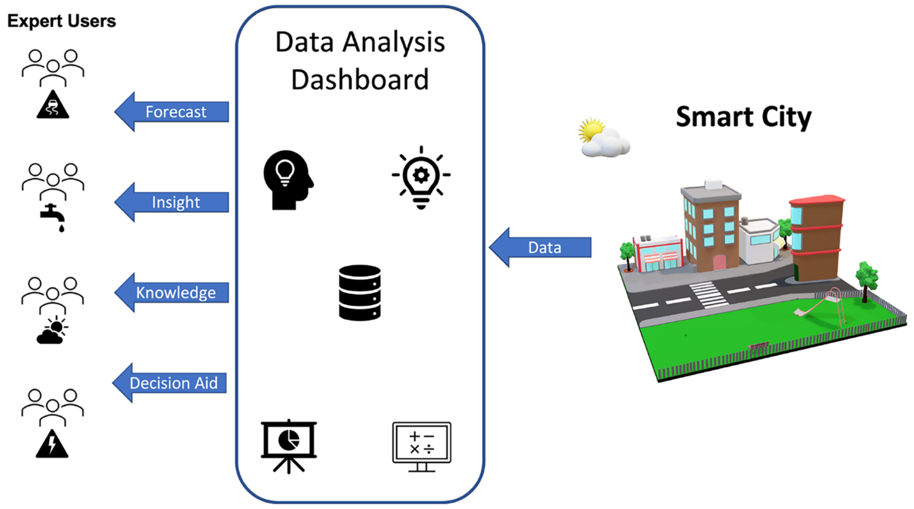


Fig. 1 Smart City Use Case

promotes model reusability rather than creating slightly different models for every prediction, ensuring experiment reproducibility in the context of open science. The system suggests and retrieves a fitting model (if it already exists), that meets experts needs, allowing them to generate a new model only when necessary. This will significantly prove useful, considering the growing number of ML models generated over time, and allows users to respond to their needs more quickly because they do not need to generate models, which are often very expensive. Furthermore, such system enhances users' understanding of existing models, improves result explainability, and fosters more productive collaboration within the team.

In order to be able to efficiently select ML models that meet users demands, several challenges have to be addressed related to:

1. *Model Representation* this entails the challenges related to the unified description of the models, as well as their metadata, technical aspects (i.e., algorithmic specifications), used data sets (i.e., training, testing features/data specifications), the application domains in which the models are eventually deployed, and the evaluation metrics/scores. More specifically: *Challenge 1.a*: How to extensively represent machine learning models and their descriptive metadata to facilitate ML models search, versioning, and retrieval? *Challenge 1.b*: How to cover technical aspects and map models to the algorithms that generated them, while categorizing technical specifications for efficient search and retrieval? *Challenge 1.c*: How to encompass the intricacies of training and testing datasets, capturing contextual information (spatial, temporal, etc.) to compare model similarity from a data perspective and understand the usage context? *Challenge 1.d*: How to include the application domains where the models are deployed, allowing higher-level clustering and categorization of ML models based on their field of application? *Challenge 1.e*: How to incorporate model evaluation metrics and scores in the representation for ranking and presenting models tailored to user needs?
2. *User Input Representation* this entails the challenges related to the description of user needs, more specifically: *Challenge 2.a*: Through what form (i.e., query expression model) users' inputs can be defined, allowing users to express easily what they aim for at different aspects, e.g., the application domain, the desired outcome (data analysis, forecasting, classification, etc.), and specific contexts (e.g.,weather forecasts for a specified period or location)? *Challenge 2.b*: How to include user needs with ML models representation, in a way facilitating their matching analysis with the models different characteristics? *Challenge 2.c*: What are the main users' preferences to be considered in users' input, e.g., to define priorities for the entities used to filter the necessary ML models (based on their application domains, their type of algorithm, etc.), and to express the desired type of results (e.g., retrieve the set of the ML models having the top-k scores, or the ones having a max global score, etc.)?
3. *Model Retrieval and Recommendation* this entails a different set of challenges related to the alignment between user needs and ML models characteristics, for models retrieval, models recommendation and models optimization. For this part, we focus, in this work, on the challenges related to the retrieval of ML models aligned with users inputs[1]: *Challenge 3.a*: What are the similarities criteria that can be used to study the matching between ML models specifications and users inputs? *Challenge*

[1] The other sub-challenges related to the ML models recommendation and optimization scopes, will be considered in a future dedicated work.

3.b: How to compute a global similarity matching score to retrieve the most suitable ML models answering user needs/preferences based on the identified similarities criteria?

While existing works primarily focus on data set similarity or certain performance metrics when suggesting ML models, our aim, in this paper, is to extend these solutions by considering a more comprehensive set of ML concepts (e.g., application domains, usage scenarios/contexts, technical algorithmic aspects) that could impact ML model retrieval and recommendations. We also consider user input representation, and propose similarity criteria to measure the matching between the ML models characteristics and user needs, in order to retrieve the most suitable ML models aligning with users demands. However, before delving into our proposal, we will review some works related to Machine Learning models representation, and evaluate them based on the challenges and requirements identified in our motivating scenario.

3 State of the Art

In this section, we investigate multiple models, approaches, and reviews focused on providing comprehensive knowledge about machine learning (ML) techniques and algorithms. This includes details about their categories, advantages, and other relevant aspects. The objective is to describe the performance and applicability of these ML methodologies. For this aim, we conducted a comparative analysis based on distinct criteria, grouped into two main categories:

1. *ML Representation Criteria* This encompasses criteria used to represent ML models, their building/generation process, behavior, performance, and some useful metadata descriptors:
    - *Criterion 1.A. Algorithm Representation* It assesses the ability to describe and link the ML models to the algorithms that generated them. This facilitates the inference of their usability and technical limitations.
    - *Criterion 1.B. Data Representation* This evaluates how well ML datasets used for training and testing, are represented, including related characteristics such as their features, their values, and some statistical descriptors.
    - *Criterion 1.C. Performance Representation* It measures the ability to incorporate accuracy and performance metrics/descriptions for each ML model. This provides insights into the quality of results, and allows for comparisons between ML models.
    - *Criterion 1.D. Metadata Representation* This examines the capability of including ML models meta descriptors, enhancing the ML modeling process with various high-level information/features (e.g., model metadata, algorithm metadata, and dataset metadata).
2. *ML Usability and Compatibility Criteria* These criteria are focused on describing the application domain and the context of each ML model:
    - *Criterion 2.A. Application Domain Representation* This refers to the ability to represent various application domains via a keyword-based representation, and link them to the necessary ML models (e.g., associating a temperature prediction model with the environmental monitoring application domain).
    - *Criterion 2.B. Usability Representation* It assesses the capacity of specifying various ML model contexts within each application domain. This knowledge helps determine where each ML model is best used for achieving accurate results (e.g., adopting a prediction model that is trained on winter data to predict summer-related data outputs will negatively impact result quality).

3.1 Ontology-Based Models for Describing Machine Learning

MLOnto [6], which stands for Machine Learning Ontology, is an model designed to capture knowledge related to the field of Machine Learning. It comprises seven primary classes: 'Applications', 'Algorithms', 'Dependencies', 'Frameworks', 'Dictionary', 'MLTypes', and 'Involved'. While encompassing various ML types such as AutoML, Semi-supervised Machine Learning, Supervised Learning, Reinforcement Learning, and Unsupervised Learning, the model's representation is limited. It falls short in addressing several critical criteria, e.g., representing models data sets (training and testing data sets), model performance, and model usability.

In [12], an approach based on an ontology model is introduced to instill accountability in Machine Learning systems. The methodology unfolds three key phases: (1) the creation and deployment of predictive models to ensure availability, (2) the annotation of relevant information extracted from the models and forecasts using ontological terms, and (3) the storage of data annotations with provisions for utilizing them for accountability purposes. The second phase comprises two areas. In the first area, the forecasts generated by predictive models are represented using three ontology models: the AffectedBy ontology (https://iesnaola.github.io/AffectedBy), the Execution-Executor-Procedure (EEP) (https://iesnaola.github.io/EEP), and the Result Context (RC) (https://iesnaola.github.io/RC). In the second area, the predictive procedures employed to generate forecasts are modeled using the ML-Schema ontology [13]. Despite that these ontology models address various facets of Machine Learning, encompass model performance and represent model training datasets, they exhibit limitations in considering specific criteria. These include the model context (beyond temporal and spatial aspects) with its associated constraints when necessary, as well as the application domain of the model.

In [5], the authors introduce OnML, an ontology-based approach for Interpretable Machine Learning (IML). This methodology employs interpretable models, ontologies, and information extraction techniques to generate semantic explanations. The process involves the identification and inclusion of ontology-based tuples in a sampling strategy, wherein semantic relationships between terms, words, and ideas are sampled and incorporated into the training of the interpretable model, without employing each of them individually. Additionally, to streamline the search space for semantic explanations, the authors propose an anchor learning method. The primary focus of this work lies in leveraging ontology models for the semantic explanation of predicted ML results, without explicitly representing or describing ML datasets, their context, behavior, etc. Nonetheless, by relying on some ontology models, the approach provides insights into the application domains of the utilized MLs, along with their usability.

3.2 Context-Based Approaches for Describing Machine Learning

The research presented in [10] outlines an approach that employs contextual information for training ML models. The primary concept involves training ML models to optimize a specific scoring function tailored to each operational context. In experimental comparisons, the results of the context-aware approach, derived from specialized models trained for individual contexts, were contrasted with the utilization of a general model trained across all contexts. The outcomes indicate that the proposed approach mitigates bias towards a strategy employing a universally trained model, albeit with a relatively minor difference in error. Therefore, a thorough evaluation is necessary to determine the strategy that better aligns with specific application requirements. Nevertheless, the context-aware approach merits consideration, particularly in relation to the criticality of application resource needs, such as connectivity and memory. In comparison to our approach, the ML model contexts in the proposed method are manually defined and utilized without being represented, along with other aspects like ML datasets and ML application domains, in a machine-understandable form. This representation allows for the correct and automatic utilization of ML models in appropriate contexts.

3.3 Machine Learning Description Based on Reviews and Surveys

In [9], a comprehensive review is presented to establish definitions and a foundational understanding of various machine learning (ML) categories, including Unsupervised, Supervised, and Reinforcement Learning. The paper delves into the methodologies employed in the design of supervised ML studies, and introduces the bias-variance trade-off as a critical theoretical foundation in supervised machine learning. While the work provides an overview and description of common supervised ML algorithms such as Logistic Regression, Linear Regression, and Naive Bayes, it falls short of representing them, including aspects like ML datasets and ML application domains, in a comprehensive machine-understandable model, which is essential for ensuring their accurate usage in specific scenarios.

In [8], a survey is presented to assess the strengths and weaknesses of various machine learning (ML) algorithms, including Logic basic algorithms (e.g., Learning Set of Rules and Decision Trees), Instance-based Learning, Statistical learning algorithms (e.g., Bayesian Networks), Deep Learning, and Support Vector Machines. While the survey outlines the utility of each ML algorithm, the descriptions are tailored for users with a certain level of expertise who understand how and where these ML methods are best applied. In contrast, our work extends this further, by providing detailed descriptions of ML models, encompassing aspects such as the ML datasets used, the corresponding ML application domains, and more. This information is presented in an intelligible machine-readable format, facilitating the use of ML models across different contexts.

A study is given in [7] to provide an overview of ML categories, including Unsupervised, Semi-supervised, and Reinforcement learning. It outlines three distinct ways in which ML models are applied in enterprises: Classification, Clustering, and Prediction. Additionally, the work introduces a process model for selecting ML algorithms based on factors such as data type, desired accuracy, and intended interpretability. While the study contributes to understand the landscape of ML techniques and their relevance in enterprise applications, including the trade-off between interpretability and accuracy, it overlooks several important aspects. These include essential considerations for determining the most suitable ML model for specific cases, such as describing ML datasets, their related contexts, etc. Furthermore, the study fails to provide ML model descriptions in a comprehensible machine format, thereby limiting users' expertise and knowledge.

Table 1 Evaluation of prevailing machine learning (ML) description models and approaches in relation to the specified criteria

	1. ML representation criteria				2. ML usability & compatibility	
	Algorithm	Data	Performance	Metadata	Application domain	Usability
MLOnto, 2020 [6]	+	–	–	Limited	+	–
ML-Schema, 2021 [12]	+	+	+	Limited	–	Limited
OnML Approach, 2022 [5]	–	–	–	–	Limited	Limited
Context-aware ML-based Approach, 2018 [10]	–	–	–	–	–	Limited
Review, 2019 [9]	Limited*	–	Limited*	–	Limited*	Limited
Survey, 2015 [8]	Limited*	–	Limited*	–	Limited*	Limited
Study, 2020 [7]	Limited*	–	Limited*	–	Limited*	Limited

In Table 1, we present a comparative analysis of the machine learning description models, approaches, and reviews discussed earlier, with respect to the criteria outlined at the beginning of this section. We use the symbol "+" to signify positive coverage of a criterion, "-" to denote the absence of coverage, the term "Limited" to indicate partial coverage, and the term"Limited*" to denote partial coverage without an implemented or proposed model. Our comparative analysis show that, to our knowledge, there is no existing solution/approach able to describe ML models characteristics, with respect to the specified criteria.

4 ML Models Retrieval Approach

In this section, we present our approach for ML models retrieval, which is based on three main parts: (1) Machine Learning models representation, through the definition of a semantic model ontology that describes the characteristics of ML models, (2) User inputs representation, through a formally defined user request, and (3) the definition of the similarities measures that are used to match between ML models specifications and users needs.

Our proposal for ML models representation and retrieval is presented in a framework illustrated in Fig. 2. The framework is composed of eight steps. In the first step, existing ML models are described using SML, a semantic ML model ontology that defines the characteristics of ML models based on a unified vocabulary (see Sect. 4.1). Once the models are described, we obtain, in the second step, ML model instances, which refer to concrete model entities exemplified by the concepts and properties defined within the SML ontology. In the third step, some elements of the user request (presented formally in Eq. 1) are exploited by a Generative AI solution [14]. Specifically, the user can provide a set of keywords and a given file to specify his needs (e.g., the domain of application required for the ML models to be suggested, the desired type of ML model algorithm, etc.). In the fourth step, these user request elements are passed to the Generative AI solution, whose main objective is to complete the user given keywords with other relevant words based on the analysis of the user provided file. The set of keywords enriched by the Generative AI solution is then used in the fifth step to filter ML model instances and retrieve the ones matching user needs. This helps in reducing the search space and resources (time, memory consumption, etc.) required to apply similarity measures between user demands and ML model specifications. Once the ML models are filtered, they are transferred to the Similarities Measures module (in the sixth step), along with other elements of the user request, i.e., the user provided file and user preferences (in the seventh step). Finally, in the eighth step, a matching between some user input elements and the filtered ML models is applied by the Similarities Measures module, based on multi-criteria similarity measures (i.e., Feature, Feature Value Type, Temporal Context, and Spatial Context), to retrieve the ML models that meet user needs.

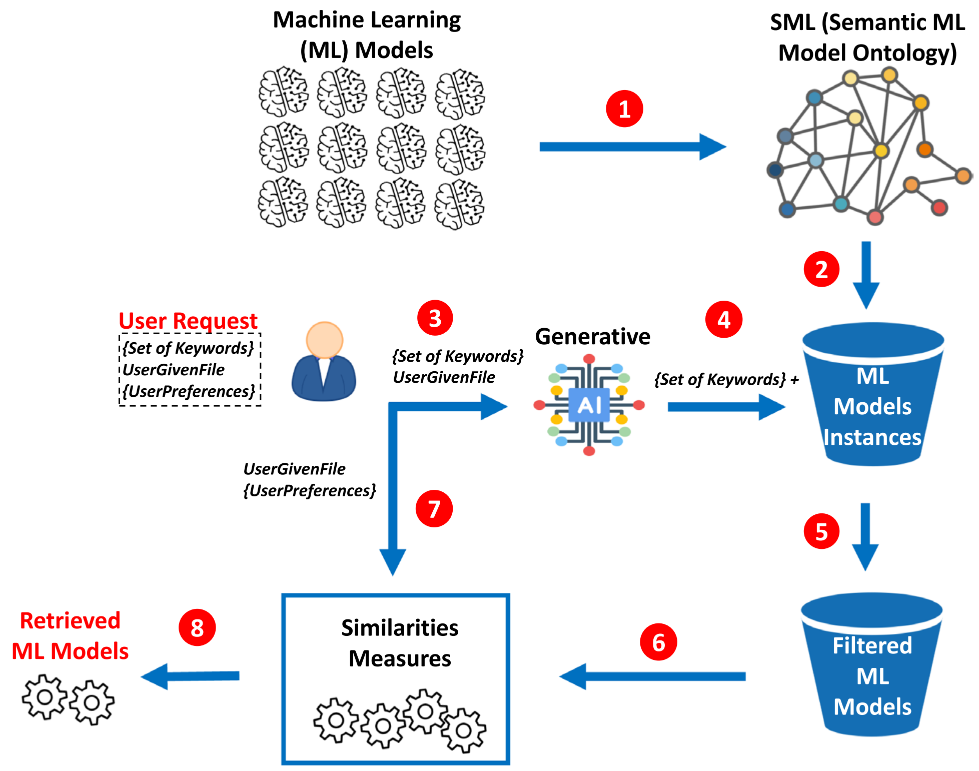
4.1 SML: Semantic Machine Learning Model Ontology

In this part, we introduce our ontology-based model called "SML" (Semantic Machine Learning), defined to describe and to store the characteristics of ML models. This is crucial for enhancing the understanding of ML models and facilitating their selection in specific contexts. SML, which is designed using entities and relations between these entities (see Fig. 3), relies on a vocabulary that ensures a common description of ML models, applicable across various environments and platforms. It is worth noting that attributes of each entity are omitted in Fig. 3 for clarity.

4.1.1 SML Model Representation and Application

As depicted in Fig. 4, our SML representation is crafted to discern patterns or behaviors within collected data. This is accomplished by leveraging previous or historical data known as the training data set (*SML:TrainingDataSet*). The

Fig. 2 Framework of our proposal for ML models representation and retrieval



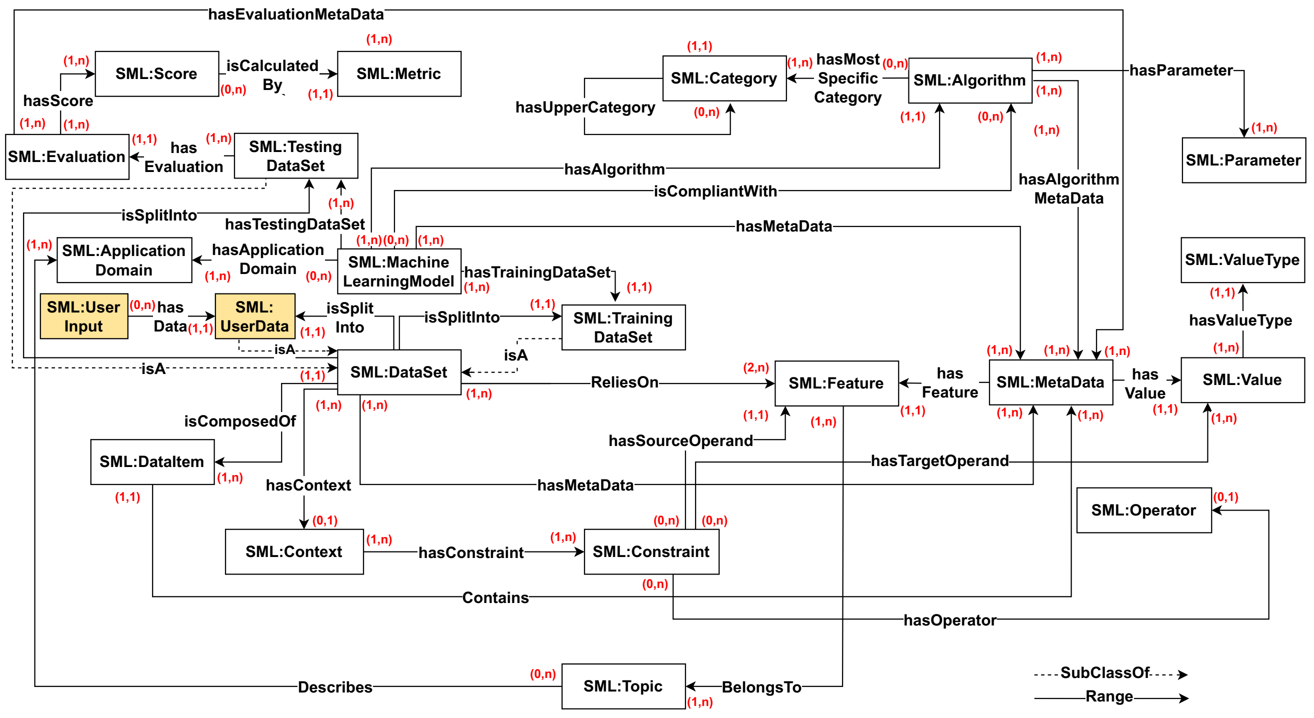


Fig. 3 Overview of the proposed Semantic Machine Learning (SML) model ontology

training data set, which inherits from the *SML:DataSet* entity, is utilized during the learning phase to tailor (train) the model for predicting or classifying values known in the training set but unknown in other (future) data. Each model has its own metadata (*SML:MetaData*) that give some information about the created model, such as Model Creation

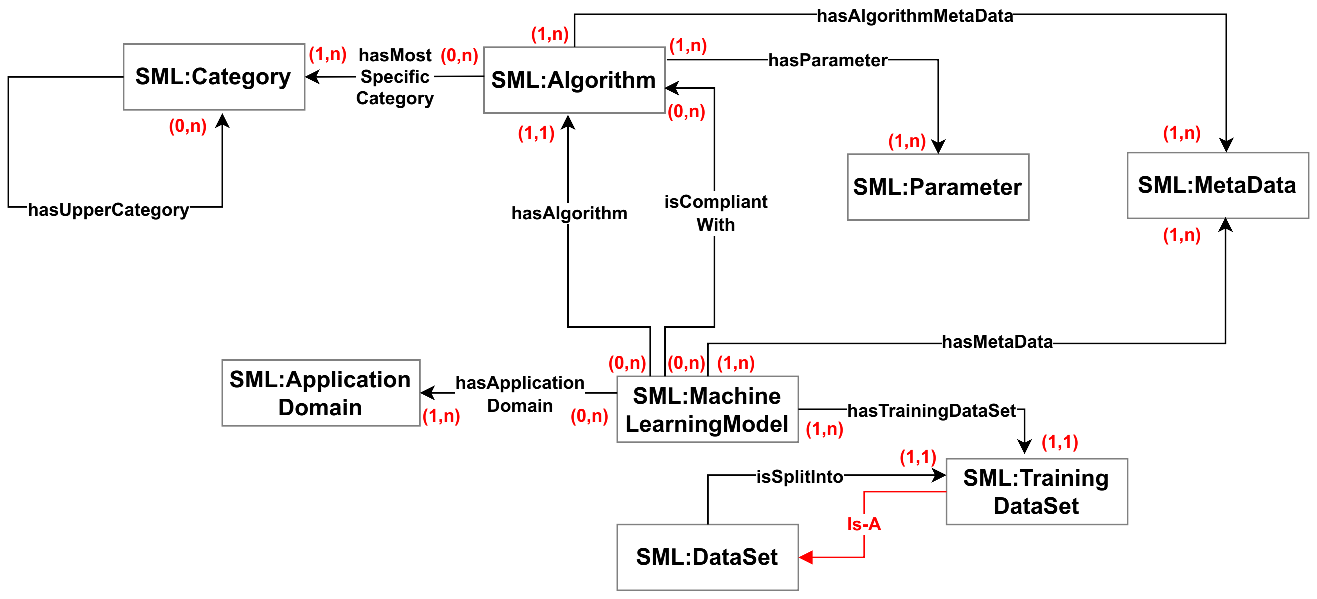


Fig. 4 ML model representation and application

Date and Model Developer. Furthermore, every model is specifically applicable in defined application domains (*SML:ApplicationDomain*), such as smart buildings, healthcare, transportation, and more.

The learning models use distinct algorithms (*SML:Algorithm*), such as Decision Trees, Support Vector Machines, Naïve Bayes, etc. Each algorithm is associated with metadata (e.g., Algorithm Creation Date, Algorithm Description) and specific parameters represented as key-value pairs. Categorization of algorithms is defined through a category entity (*SML:Category*), encompassing types like Classification and Regression, potentially including subcategories linked by the “hasUpperCategory” relation. Certain models that are related to specific algorithms (e.g., Linear Regression), may demonstrate compliance with other algorithms (e.g., Lasso Regression) through the “isCompliantWith” relation. This alignment can be discerned through some calculations and analyses conducted on the training data set of the models, taking into account their respective contexts (detailed in the subsequent subsection).

4.1.2 ML Data Set Modeling and Context

As illustrated in Fig. 5, a dataset (*SML:DataSet*), can be partitioned into several categories, including: (1) a training dataset (*SML:TrainingDataSet*), used for training the learning model, and (2) a testing dataset (*SML:TestingDataSet*), employed to assess and evaluate the model post-training (refer to Fig. 6). A dataset comprises some data items (*SML:DataItem*), each associated to metadata (*SML:MetaData*). The metadata for each data item includes a feature (*SML:Feature*), such as the Creation Date, the Description, Humidity, Location, and a corresponding value (*SML:Value*) linked to a value type (*SML:ValueType*). Concepts have been defined for each Feature, Value, and Value Type, enabling our solution to define specific constraints (as elaborated below) on certain feature values. This is necessary in many cases for accurately describing the context of ML models.

A dataset, associated to some metadata, has at least two features, each possessing attributes such as Name, Type (e.g., Categorical, Textual, Numerical), Range, and a Boolean value indicating if it is an independent feature. An independent feature serves as the cause, and its value is unaffected by other variables in the study. Conversely, a dependent feature is the effect, as its value is contingent on changes in the independent feature. Features are related to topics (*SML:Topic*), which are used for the description of the application domain (*SML:ApplicationDomain*) of a learning model. A dataset encompasses a context (*SML:Context*) that has some constraints (*SML:Constraint*). Each constraint comprises a source operand (i.e., *SML:Feature*), a target operand (i.e., *SML:Value*), and an operator (*SML:Operator*). For example, a spatio-temporal context could be defined by the “Season” feature with the value “Winter” and the “Location” feature with “Paris” as its value. This context informs that, for instance, the training dataset of a specific ML model is associated with Paris during Winter. Contexts also facilitate the usage of certain datasets for other ML models, depending on the degree of matching or closeness of their respective contexts.

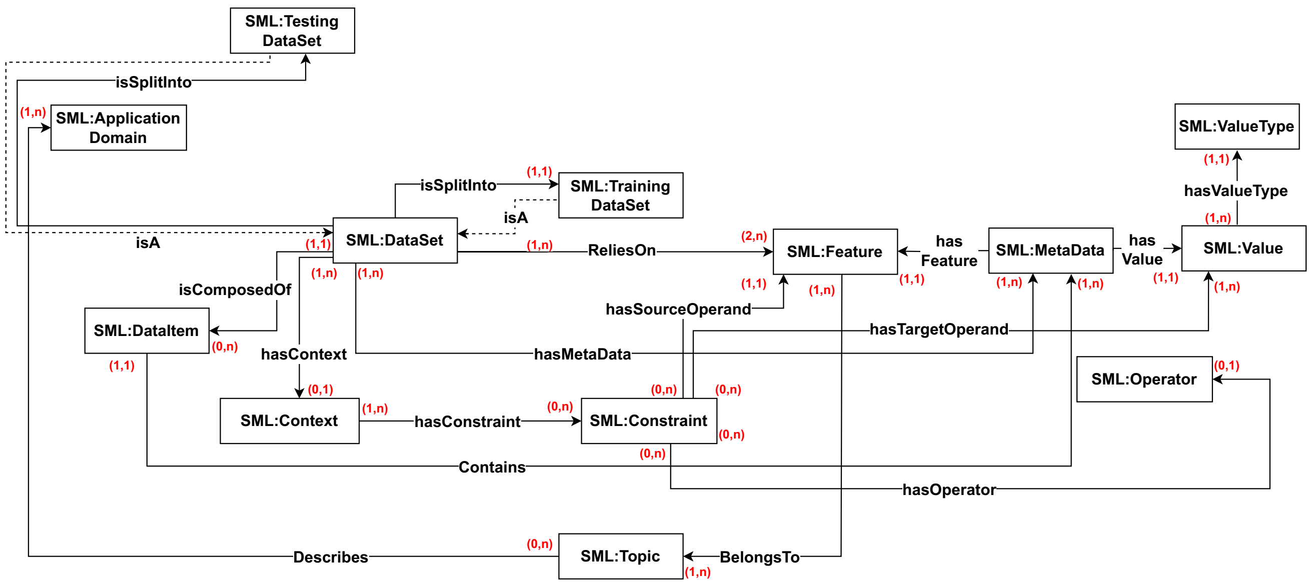


Fig. 5 ML data set modeling and context

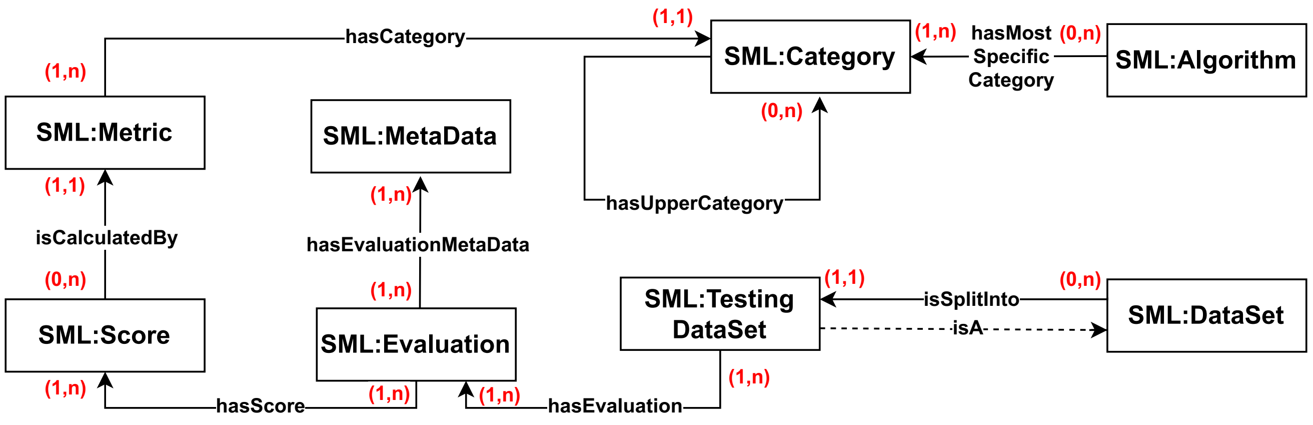


Fig. 6 ML model evaluation

4.1.3 ML Model Evaluation

Upon constructing the machine learning model with the training dataset, a crucial step involves testing its performance using a distinct set of data known as the testing dataset (*SML:TestingDataSet*). This dataset is employed to assess the performance of the ML model's training, with potential adjustments or optimizations to enhance results. As depicted in Fig. 6, a testing dataset entails an evaluation (*SML:Evaluation*). Each evaluation is associated to some metadata (*SML:hasEvaluationMetaData*) and to a computed score (*SML:Score*), derived from specific metrics (*SML:Metric*), such as MSE (Mean Squared Error) and MAPE (Mean Absolute Percentage Error) [15]. The metrics are categorized into categories (*SML:Category*) based on the algorithm employed in constructing the ML model.

4.1.4 SML-Based ML Model Instantiation Example

Let us consider a brief example of an instantiated ML model described using the concepts and properties defined in the SML ontology. As illustrated in Fig. 7, an instance of the class *SML:MachineLearningModel* is created and named *Machine Learning Model 1*. *Machine Learning Model 1* is based on a *Linear Regression* algorithm, which falls under the *Regression* category. This category refers to ML models instances used to predict a continuous target variable based on one or more predictor variables. The *Linear Regression* algorithm has a *Description* instance of the class *SML:MetaData*, which is associated with the *SML:Algorithm* concept. The description states that the linear regression algorithm is a supervised learning algorithm used for predicting a continuous target variable. Additionally, the *Linear*

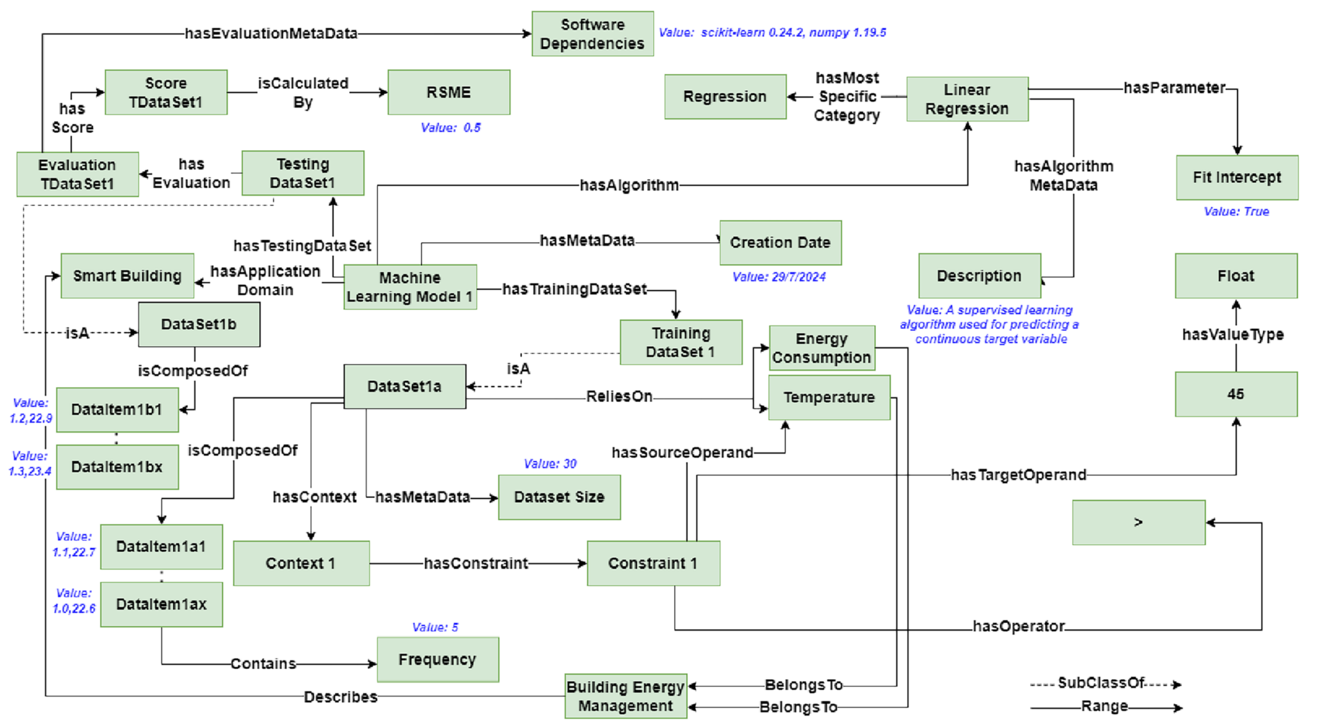


Fig. 7 An example of an instanced ML model using SML ontology concepts and properties

Regression algorithm instance has a *Fit Intercept* parameter set to ”True”, meaning that the model is configured to find the best-fit line that does not necessarily pass through the origin (0,0). This provides greater flexibility and potentially leads to a more accurate representation of the relationship between the variables.

The *Machine Learning Model 1*, which has the *Smart Building* instance as its application domain, includes metadata named *Creation Date* with the value “*29/07/2024*”. It comprises: (1) a Training Dataset, *Training DataSet 1*, that is of type *SML:DataSet* (*DataSet1a*) having a metadata instance, *Dataset Size*, with a value of “*30*” MB, and (2) a Testing Dataset, *Testing DataSet 1*, which is also of type *SML:DataSet* (*DataSet2b*). The *Training DataSet 1* instance utilizes two features: *Energy Consumption* and *Temperature*, both belonging to the same topic instance: *Building Energy Management*, describing the *Smart Building* application domain. Additionally, *Training DataSet 1* is composed of several data item instances, such as *DataItem1a1* and *DataItem1ax*, each containing a metadata instance named *Frequency* with a value of “5”.

The *Training DataSet 1* has a context, *Context 1*. This context holds a source operand (i.e., *Temperature* feature instance), a target operand (i.e., “*45*” value instance) with a value type (i.e., *Float*), and an operator (i.e., “>”) that refers to the “greater than” operator instance. The context indicates that all temperature values included within the *Training DataSet 1* are greater than “*45*” Celsius degrees. In addition to providing insights into the data sets, contexts facilitate the use of certain data sets for other ML models, based on the degree of similarity or closeness between their respective contexts.

As per the *Evaluation DataSet 1*, it has a score instance, *Score TDataSet1*, that was calculated using the RSME metric (Root Mean Square Deviation). The value of RSME, which designates the differences between the observed values and predicted ones, is equal to “0.5”, showing good prediction accuracy of the ML model. Also, the *Evaluation DataSet 1* has some metadata instance, i.e., *Software Dependencies*, inherited from the *SML:MetaData* class. The *Software Dependencies* has the value of “textitscikit-learn 0.24.2, numpy 1.19.5”.

4.1.5 User Input Representation

In order to allow and facilitate the matching between user needs and ML models, we represent user input by using two main concepts: *“SML:UserInput”* and *“SML:UserData”*, which are related together through *“SML:hasData”* property. In our work, we considered that the user data is a data set (*SML:DataSet*) that has the same representation of any ML training or testing data set, each, having an application domain, data-items, features, contexts, etc. This eases the study of the similarities measures between user data

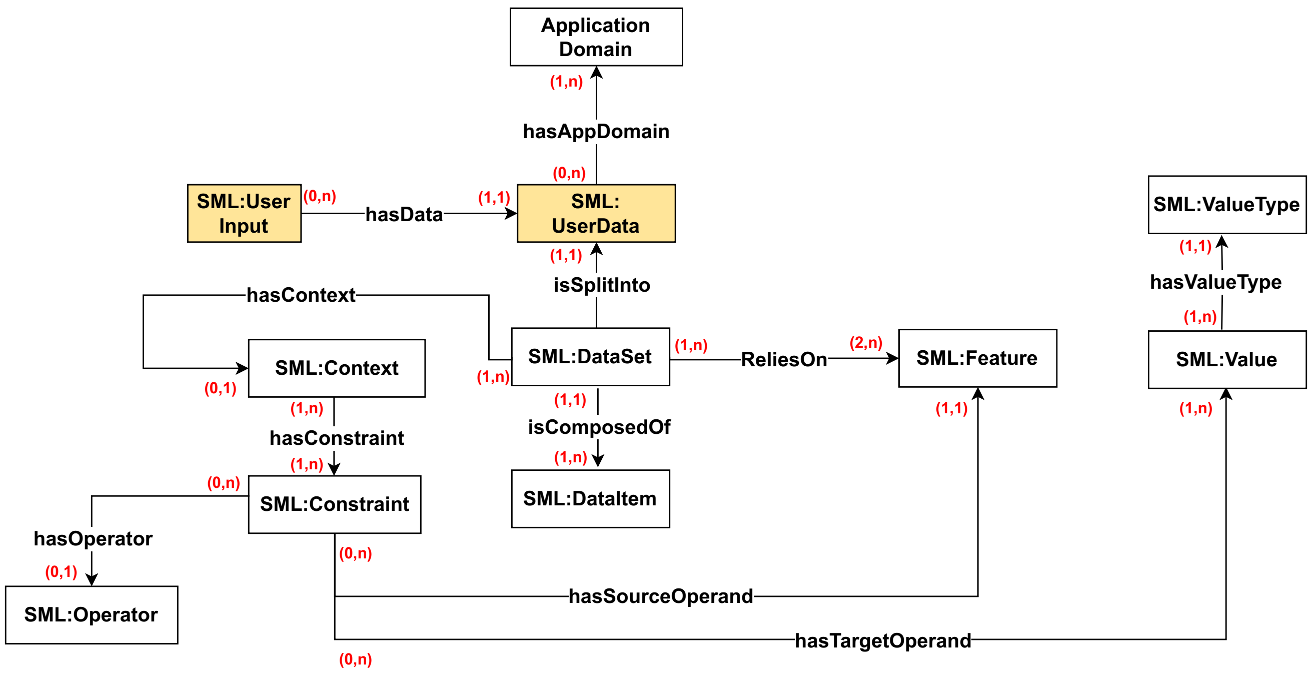


Fig. 8 SML user input representation

and ML data characteristics, and makes it more efficient, as both matching entities derive from the same concept (*SML:DataSet*) (Fig. 8).

4.2 User Request Definition

Our approach, outlined in this paper, suggests employing several similarity measures between the characteristics of ML models and user inputs, to retrieve the most suitable ML models that align with users requests. In our work, we formally define a user request as follows:

$$User\,Request = \{Keywords\}, UserGivenFile^{*}, \{User\,Preferences\} \tag{1}$$

where,

- {Keywords}: refers to a bag of words that are, optionally, given by the user. These words can be tagged by the user for pre-structuring purposes, according to different ML models characteristics (or entities): "Application Domain", "Category", "Algorithm" and "MetaData". These pre-structured words are used to filter the models that correspond more to user needs. This helps in reducing the search space and similarities measures, in terms of time and memory consumption (see the following sections), between ML models and user demands. It is to be noted that, in our approach, the set of keywords can be completed by other words (not specified by the user), through the help of the Generative AI (GAI) [14], applied on the user data file (see below). The GAI involves the creation of digital content, in our case a set of words, based on AI models that extract and understand information provided by human (in this case the user data file).
- UserGivenFile*: it is a required variable that can include one of the following:

  1. Either a data file, holding some data items with features and values. When provided, it is compared to existing ML models characteristics (mainly ML training datasets) to identify the ones that can act at best on the given data.
  2. Or a ML model file, covering the behavior of a developed model. In this case, the file is compared to existing ML models to identify the ones that can perform better for instance (with some recommended optimizations that may be applicable), or the ones that are similar (in terms of context for example), etc.

In this work, we will focus on the alignment between the specifications of given user data files and ML models characteristics.[2] A user data file is exploited by the GAI to retrieve some useful words used for ML models filtering (as mentioned above), and is compared to the training data sets of existing ML models, according to the

[2] The user files having the type of ML models files will be treated in another dedicated paper.

following similarities criteria (see sections below): (1) Feature, (2) Value Type, (3) Temporal Context, and (4) Spatial Context, to retrieve the most convenient models matching user needs. The user data file is represented by the "SML:UserData" entity in the SML model. Since "SML:UserData" has the same structure of the ML models training data set (SML:TrainingDatSet), as both are sub-classes of "SML:DataSet" entity, the matching between their characteristics will be straightforward based on their related entities.

- {User Preferences} = {Filtering Order}, top-k, top-max, $\{\alpha, \beta, \gamma, \delta\}$, where:
  - {Filtering Order}, involves the preferences that the user can, optionally, give (through a specified order) to the entities used to apply the filtering process on the ML models based on the set of keywords. For example, the user may prefer to start the filtering of the ML models according to their application domain (SML:ApplicationDomain) at first, then according to their categories (SML:Category). In this case, an order of '1' will be given to the application domain, and an order of '2' will be set to the categories. Thus, the {Filtering Order} = {1234}, with '3' and '4' refer to the ordering related to the rest of the entities "SML:Algorithm" and "SML:MetaData", respectively (if existed).
  - top-k, is an integer type variable that can be specified by the user, whenever he wants to retrieve the ML models having the 'top-k' scores.
  - top-max, is a boolean type variable that can be specified by the user, whenever he requires the best ML model(s) having the 'top-max' global score.
  - $\{\alpha, \beta, \gamma, \delta\}$, are weights that can be given by the user (between 0 and 1), respectively to: the 'SyntacticScore', the 'SemanticScore', the 'DataFreqScore', and the 'TemporalCoverageScore' (see Eqs. 4 and 5). α and β are weights that the user may use to specify the type of similarity score he wants to emphasize more while matching the features between the existing ML models and those in his given file (UserGivenFile). α is a weight value attributed to the 'SyntacticScore', and β is a weight value assigned to the 'SemanticScore', such that $\alpha + \beta = 1$. Both, the 'SyntacticScore' and the 'SemanticScore', are used in the calculations of the 'FeatureSimScore' (see Eq. 4). γ and δ are wights that the user can assign to determine the type of similarity score he requires to highlight more when computing the temporal context similarity score 'TempSimScore' (see Eq. 5). γ is a weight value given to the 'DataFreqScore', and δ is a weight value given to the 'TemporalCoverageScore', such that $\gamma + \delta = 1$.

When receiving a user request, our solution begins the process of ML models filtering based on the set of keywords, which is enriched by the GAI applied on the given user data file. Then, the similarity score calculations start between the user data file and the filtered ML model training data sets. The similarities measures rely on a multidimensional representation, unique in the literature, that is based on four criteria: (1) Feature, (2) Feature Value Type, (3) Temporal Context, and (4) Spatial Context. A global score is given to each training data set related to the filtered models, based on their matching with the user request (including his preferences), and the best ones, having the highest scores and answering user needs, are finally retrieved.

4.3 Similarities Measures for ML and User Needs Alignment

Before presenting the computations of the different similarities measures, we propose, in this work, the formulas: A and B (see Eqs. 2 and 3), that we used to normalize the obtained scores values (when it is necessary) between 0 and 1. In cases where the higher the result is, the most similar the entities are, Formula A is applied (e.g., When adopting Jaccard, algorithm, which is a string-based matching algorithm [16]). Whereas, in cases where the higher the result is, the most dissimilar the entities are, Formula B is applied (e.g., When adopting Levenshtein algorithm, another string-based matching algorithm [17]). While there are various normalization functions available in the literature [18], the decision to use formulas: A and B, is driven by the fact that they offer a straightforward approach to normalization, making the calculations easier to interpret and understand. Moreover, the chosen normalization functions align well with the specific requirements and constraints of our approach, by ensuring that the scores are scaled in a manner that preserves the relative differences between them, which is crucial for the accurate comparison and combination of scores in our context.

$$Formula\,A = \frac{x}{x+1} \tag{2}$$

where,

$$\lim_{x\to 0} f(x) = 0, \text{ and } \lim_{x\to\infty} f(x) = 1$$

$$Formula\ B = \frac{1}{1+x} \tag{3}$$

where,

$$\lim_{x\to 0} f(x) = 1, \text{ and } \lim_{x\to\infty} f(x) = 0$$

Table 2 Examples of feature similarity scores calculations

ML model training dataset	Feature used in the training dataset	Feature extracted from user data file	SyntacticScore (Levenshtein)	SemanticScore (Path)	FeatureSimScore
A	Temperature	Temp	0.125	0.1	0.1125
B	Humidity	Temp	0.125	0.083	0.104
C	CO_2	Temp	0.2	0.013	0.1065

4.3.1 Feature-Based Criteria

The first matching measure applied in our proposal is done between the features of ML models training data set (SML:TrainingDataSet), and the features of the user data file (SML:UserData). The similarity between the identified features is realized according to two aspects: (1) Syntactic, and (2) Semantic. For each of these aspects, a score is given, and thus, we, formally, define the feature similarity score as follows:

$$FeatureSimScore = \frac{\sum_{u=1}^{U}\sum_{v=1}^{V}(\alpha.SyntacticScore_{u,v} + \beta.SemanticScore_{u,v})}{U+V} \quad (4)$$

where,

- α and β are weights given to 'SyntacticScore' and 'SemanticScore', respectively, and with: $\alpha + \beta = 1$.
- 'SyntacticScore', is a normalized score computed using a string-based matching algorithm, e.g., Levenshtein, Cosine, and Jaccard [17], between each of the features extracted from the user data file (SML:UserData), and the features forming the ML models training datasets (SML:TrainingDataset).
- 'SemanticScore', is a normalized score computed using a knowledge-based or topological method, e.g., Path, Wup, and Lin [19], between each of the features extracted from the user data file, and the features forming the ML models training datasets. The semantic similarity between features helps in assessing the proximity of their meanings rather than relying solely on their lexical resemblance. And thus, it increases the understanding of the sense of words in different contexts, independently from their lexicographical similarity.
- $U \in \mathbb{N}^*$, and $V \in \mathbb{N}^*$, refer to the number of features presented, respectively, in the user data file and in a ML model training data set.

In the definition of the 'FeatureSimScore' equation (Eq. 4), we employed the weighted-average aggregation method [20]. This well-known method is particularly suitable for decision making scenarios where different criteria have varying degrees of importance. While other aggregation methods, such as the arithmetic mean or median, could be employed, they do not account for the varying importance of each criterion. For instance, the arithmetic mean treats all criteria equally, which is not suitable given the differing relevance of each criterion in our approach. On the other hand, methods like the OWA (Ordered Weighted Averaging) operator [21] may not offer significant advantages in our specific context, specifically as weights in OWA are assigned to the ordered position of the values rather than the values themselves.

In order to illustrate the feature similarity score calculations, we give some examples in Table 2. In the presented table, there are three ML model training datasets, having one feature, respectively: 'Temperature', 'Humidity', 'CO_2'. By comparing each of these features with the extracted feature from user data file 'Temp', and based on the normalized syntactic/semantic scores, the highest feature similarity score is the one between 'Temperature' and 'Temp', thus, the ML Model Training dataset A is the most convenient to the user input.

4.3.2 Feature Value Type-Based Criteria

The second matching measure, *FeatureValTypeScore*, is related to the value type of the previously matched features between ML models training data sets and user data file (see Sect. 4.3.1). In fact, and whenever the 'FeatureSimScore' is $\geqslant 0.5$, based on Eq. 4, the feature value-type score calculations are launched. As such, if the matched two features (any ML model training dataset feature and any user data file feature) have the same data type, e.g., Integer, String, or Boolean, 'FeatureValTypeScore' is incremented to 1. In the end, to normalize the obtained score, the latter is divided by the sum of the number of times there were two matching features (i.e., 'FeatureSimScore' is $\geqslant 0.5$).

4.3.3 Temporal Context-Based Criteria

The temporal context similarity score, *TempSimScore*, is calculated based on the similar observations (or frequency), and the intersected time covered by the ML models training data, and the user given data (user data file). Formally, it is defined as follows:

$$TempSimScore = (\gamma.DataFreqScore) + (\delta.TemporalCoverageScore) \quad (5)$$

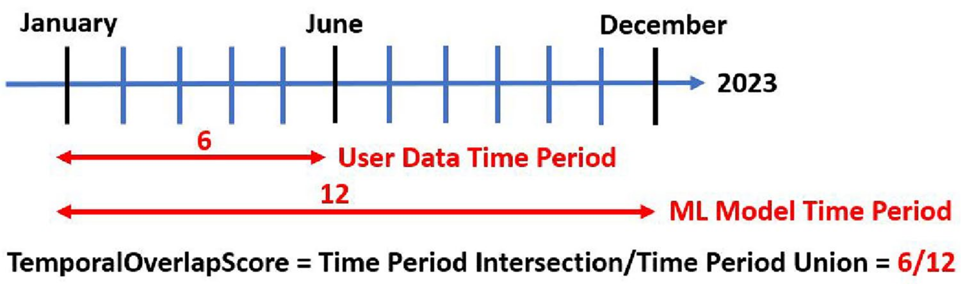

January
June
December
2023
6
User Data Time Period
12
ML Model Time Period
TemporalOverlapScore = Time Period Intersection/Time Period Union = 6/12

Fig. 9 An example of the 'TemporalOverlapScore' calculation

where,

- γ and δ are weights given to 'DataFreqScore' and 'TemporalCoverageScore', respectively, and with: $\gamma + \delta = 1$.
- 'DataFreqScore', which is computed only if the ML model training data and the user data are regular, refers to the difference of the number of observations per second. Such score is normalized using Formula B (see Eq. 3), because the lower the score is, the more similar two entities are. As illustrated in Tables 3 and 4, we can see that ML model A is more similar, in terms of the observations per second, to user data.
- 'TemporalCoverageScore': refers to the covered time periods between ML model training data, and user given data (user data file). It is defined as: TemporalCoverageScore = TemporalOverlapScore $\times$ TemporalDistanceScore, such that:
  - 'TemporalOverlapScore' $= \frac{TimePeriodIntersection}{TimePeriodUnion}$, is the time period intersection between ML model training data and user data, divided by the union of the periods (see Fig. 9).
  - 'TemporalDistanceScore' $= MLModelTimePeriod - UserDataTimePeriod$, is the normalized difference between the time period related to the ML model training data and the time period covered by user data. For example, in Fig. 9, the 'TemporalDistanceScore' is equal to 0 (before normalization), as there is an overlap between both periods. After normalization, 'TemporalDistanceScore' is equal to 1 (based on Formula B presented in Eq. 3). It is to be noted, that whenever the ML model training data period covers different years, we consider that both periods are in the same year to avoid having a 'TemporalDistanceScore' equal to 0. An example of such case is given in Fig. 10.

To illustrate the overall value of the temporal similarity score, TempSimScore, we give few examples in Table 5, where we consider that $\alpha = 0.5$ and $\beta = 0.5$.

4.3.4 Spatial Context-Based Criteria

The spatial context similarity score, *SpatialSimScore*, is based on the intersection between the location covered by the ML model training data, and the location covered by the user given data. Formally, we define the spatial similarity score as follows:

$$SpatialSimScore = \frac{LocGranIntersection}{LocGranUnion} \tag{6}$$

where,

- 'LocGranIntersection', refers to the common locations types covered by the ML model training data, and the user given data (user data file), starting from a continent granularity till the smallest location granularity, which is based on the latitude and the longitude.
- 'LocGranUnion', is the set of all the possible location types that two or more set of data can share. In our work, we define it as: 'LocGranUnion' = {L} = {Continent, Country, City, Street, Building, Floor, Room, Location}.

We illustrate the spatial similarity score calculations in different examples given in Table 6.

4.3.5 Global Similarity Score

After computing the different similarities scores between user data file and ML models training data sets (obtained after ML models filtering process as explained in Eq. 1), we present, formally, the global similarity score, *GlobalSimScore*, assigned to a ML model training data, based on the previously calculated measures:

$$GlobalSimScore = \frac{SumScores}{ScoresNumber} \tag{7}$$

where,

- *'SumScores', is the sum of all the matching similarities measures based on four criteria: (1) Feature (FeatureSimScore), (2) Feature Value Type (FeatureValTypeS-*

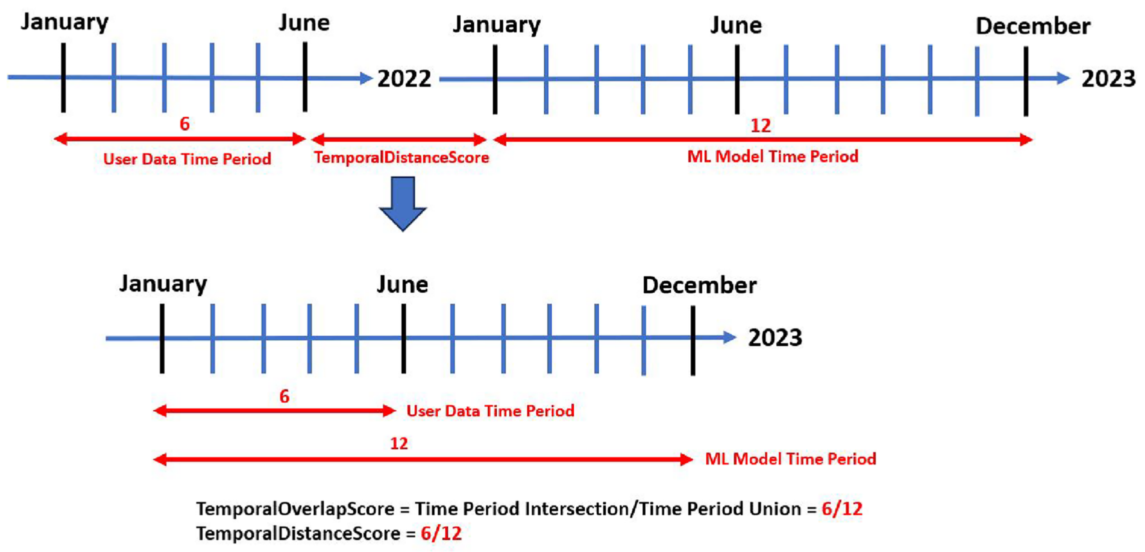


Fig. 10 An example of the 'TemporalDistanceScore' calculation

Table 3 Example of a 'DataFeqScore' between user data and ML model A

User data file observation/Sec	ML model A observation/Sec	DataFreqScore (before normalization)	DataFreqScore (after normalization)
9	15	15 – 9 = 6	1/(1+6) = 0.143

Table 4 Example of a 'DataFeqScore' between user data and ML model B

User data file observation/Sec	ML model B observation/Sec	DataFreqScore (before normalization)	DataFreqScore (after normalization)
9	28	28 - 9 = 19	1/(1+19) = 0.05

core), (3) Temporal Context (TempSimScore), and (4) Spatial Context (SpatialSimScore).

- *'ScoresNumber', refers to the number of the matching scores used in 'SumScores' for normalisation between 0 and 1. In this work, 'ScoresNumber' is equal to '4'.*

The 'GlobalSimScore' formula is calculated using the average operator, which refers to a method of aggregating multiple scores into a single composite score by taking their average [22]. The resulting score represents the overall similarity matching value between an existing ML model and a user request. In our current work, we used this aggregative approach instead of a non-aggregative one [23], which does not consolidate multiple scores into a single composite measure, to provide a straightforward summary matching measure and allow for quick comparison between different ML models. Nevertheless, as non-aggregative methods maintain the distinctiveness of each score, providing a more detailed view of evaluation, it will also be interesting to consider them in future work. The choice of the scoring method used to compute the matching between ML models and user requests can be defined by the user based on his preferences.

Table 5 Examples of temporal similarity scores calculations

DataFreqScore	TemporalOverlapScore	TemporalDistanceScore	TemporalCoverageScore	TempSimScore
0.143	6/12 (0.5)	1	0.5	0.32
0.05	6/12 (0.5)	6/12 (0.5)	0.25	0.15

Based on the user preferences included in user request (see Eq. 1), if the user assigns a number in the 'top-k' variable, our solution will return the ML models having the top-k highest global similarity scores (GlobalSimScore). In case the user assigns 'true' in the 'top-max' variable, the ML model(s) having the max global similarity score will be retrieved.

5 Experimental Evaluation

In this section, we present the experimental process employed for assessing: (1) the defined SML model ontology (defined in the following file: http://tinyurl.com/yxhu6bt5), and (2) the applicability of the proposed similarities measures between user inputs and ML models specifications.

Table 6 Examples of spatial similarity scores calculations

User data location	ML dataset location	LocGranIntersection	LocGranUnion	Spatial-Sim-Score
France	United States	{ }	{L}	0/8
France	Italy	{Continent}	{L}	1/8
Paris (France)	La Défense (France)	{Continent, Country}	{L}	2/8
1st Arrondissement (France)	5th Arrondissement (France)	{Continent, Country, City}	{L}	3/8
Areva (La Défense)	Total (La Défense)	{Continent, Country, City, Street}	{L}	4/8
AXA Investment (Majunga Tower, La Défense)	Deloitte (Majunga Tower, La Défense)	{Continent, Country, City, Street, Building}	{L}	5/8
Meeting Room 1 (23rd Floor, Deloitte)	Meeting Room 2 (23rd Floor, Deloitte)	{Continent, Country, City, Street, Building, Floor}	{L}	6/8
Meeting Room 1	Meeting Room 1	{Continent, Country, City, Street, Building, Floor, Room}	{L}	7/8
Lat1, Long1 (Meeting Room 1)	Lat1, Long1 (Meeting Room 1)	{Continent, Country, City, Street, Building, Floor, Room, Location}	{L}	8/8

5.1 SML Evaluation

The evaluation of SML ontology is based on two distinct parts:

- *Efficiency Evaluation:* This involves determining whether the concepts and properties (data and objects properties) defined in the SML ontology can effectively address the model representation challenges outlined in Sect. 2, and the criteria outlined in Sect. 3.
- *Performance Evaluation:* This includes examining the response time of the SML ontology through the application of various simple and complex queries on simulated instances of ML models. These instances are created based on different configurations, such as increasing the number of ML models, the number of ML models data items, and the number of the models features utilized in the training datasets.

5.1.1 SML Efficiency Evaluation

In this part, we establish the most beneficial queries (refer to Table 7) that can be applied on the SML ontology, aiming to address the model representation challenges outlined in Sect. 2. Additionally, we evaluate the effectiveness of these queries in fulfilling the criteria outlined in Sect. 3. The list of queries used, presented in SPARQL (a standard query language for retrieving and manipulating data stored in Resource Description Framework (RDF) format), can be accessed through the following link: http://tinyurl.com/2z2fyjw3.

As depicted in Table 7, diverse queries are available for interrogating the SML ontology. For instance, Query *Q1*, which seeks the algorithm of a specified ML model, addresses challenge 1.b, by mapping models to the algorithms that generated them. The representation of a ML model algorithm is a criterion related to the ML representation, hence the link between Query *Q1* and the ML representation criteria.

Another example is Query *Q7*, which necessitates the description of the training dataset context for a given ML model. This query tackles challenge 1.c, encompassing the context of data sets (training or testing data sets) that is important for constructing and evaluating ML models. Moreover, Query *Q7* aligns with ML usability and compatibility criteria by specifying the ML context.

5.1.2 SML Performance Evaluation

In this part, we considered five different scenarios to assess the performance of the SML ontology in terms of response time. This evaluation involved applying various queries to the SML in simulated scenarios created using the "Protégé" tool (https://protege.stanford.edu/). All details about ML data sets, ML metadata, etc., were filled using this tool, and the scenarios were diversified based on: (1) the number of ML model instances, (2) the number of data items in the models training data sets, (3) the number of features used in the models training data sets, (4) the number of metrics utilized to compute the score of the models testing data sets, and (5) the number of metadata associated to the models.

We present the query response time (in milliseconds) in the experiments, computed according to the results average of 10 sequential executions for each query. The tests were conducted using "Stardog" (https://www.stardog.com/), a platform for enterprise knowledge graph, operating on a

Table 7 List of useful queries addressing the specified challenges and fulfilling the necessary criteria

	ML Representation Criteria	ML Usability and Compatibility Criteria
Challenge 1.a	Q5- Retrieve the metadata of a given ML model, with those related to its algorithm, its training data set, its testing data set, and to the evaluations applied to the testing data	
Challenge 1.b	Q1 - Retrieve the algorithm of a given ML model	
Challenge 1.c	Q2 - Describe the training data set of a given ML model Q3 - Describe the testing data set of a given ML model	Q7 - Describe the training data context of a given ML model
Challenge 1.d		Q6 - Find the application domain of each ML model, and give a clear description of this domain
Challenge 1.e	Q4 - Retrieve the performance of a given ML model (i.e., the scores and the metrics used to calculate the evaluation applied to the testing data)	

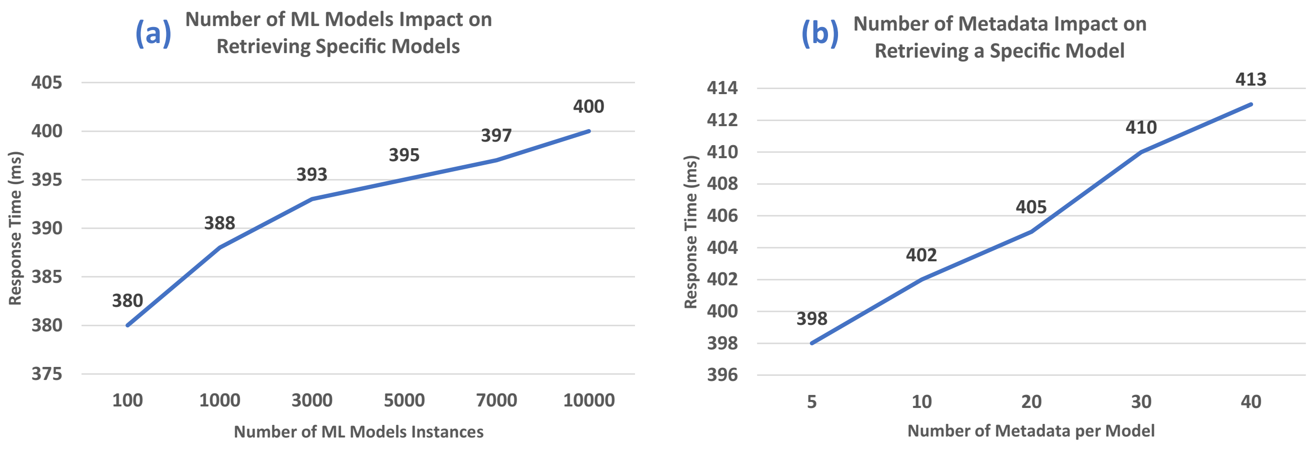


Fig. 11 Impact of the number of ML models instances and the number of ML models metadata

Windows 10 Professional machine with an Intel i7-8665U CPU @ 1.90GHz 2.11GHz processor and 1 GB RAM.

Impact of ML Models Instances and their Metadata. In the initial scenario (refer to Fig. 11-(a)), we explored the influence of varying the number of instances of ML models while requiring the models sharing a specific algorithm (i.e., Linear Regression). In this scenario, we kept the number of algorithms constant at 50. Each ML model, ranging from 100 to 10,000 models, was associated with a single algorithm, as defined in the SML ontology. The corresponding query response time was measured.

According to the resulting graph curve, the query runtime exhibits nearly linear growth with the increasing number of ML model instances. Notably, the time evolution is very noticeable between the initial two tests, where the number of ML models increased from 100 to 1000 (a difference of 900 models), in contrast to subsequent tests where the increase in the number of ML models remained more consistent (a difference of 2000/3000 models).

In the second scenario (refer to Fig. 11b), we examined the effect of changing the number of metadata associated with each instance of an ML model, particularly when requesting the metadata set linked to a specific model. In this scenario, we held the number of ML models constant at 500, altered the number of the models data items (ranging from 5 to 40), and measured the respective query response time. The resulting curve illustrates that the query execution time progresses linearly with the augmented number of metadata defined for each ML model.

Impact of Data Items and Features used in ML Training Data Sets. In Fig. 12a, we examined the impact of altering the number of data items present in the training data sets of ML models, while requesting the set of data items for a specific model. During the tests, we constrained the number of ML models to 100, varied the number of the data items used in the models training data sets (ranging from 100 to 1000), and subsequently measured the query response time. The depicted graph illustrates that as more metadata are specified for each ML model's training data set, the query runtime exhibits linear growth.

In Fig. 12b, we explored the consequences of increasing the number of features employed in the training data sets of

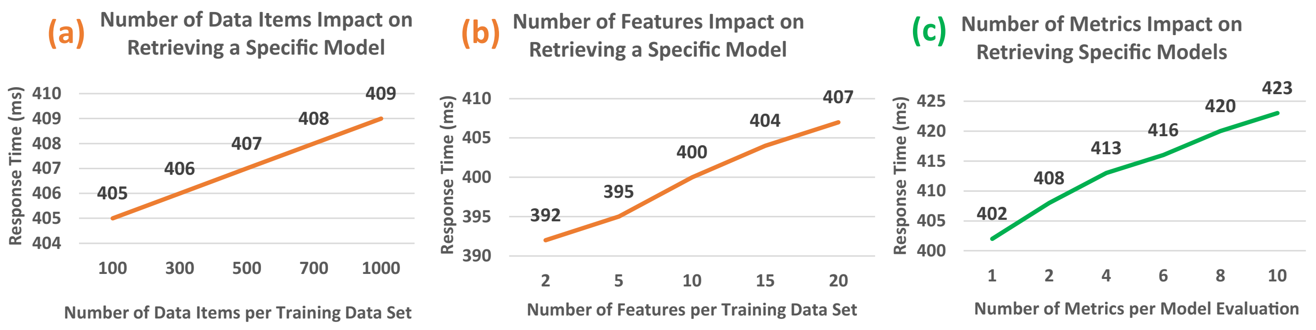


Fig. 12 Impact of the training data set data items, the number of the training data set features, and the number of ML evaluation metrics

ML models, while requesting the set of features used for a specific model. For each test, we limited the number of ML models to a maximum of 1000, assigned different numbers of features (ranging from 2 to 20) to each ML model's training data set, and subsequently recorded the corresponding query response time. The resulting graph illustrates that the runtime progresses linearly with the increased number of features utilized in each ML model's training data set.

Impact of ML Evaluation Metrics. In the final scenario, we examined the impact of the number of metrics used into the evaluation score of the testing data set of ML models (refer to Fig. 12c). In the tests, where we requested the top 3 ML models with the highest evaluation scores, we kept the number of ML models constant at 1000 and varied the number of the score metrics from 1 to 10 (e.g., MAPE and MSE [15]). Subsequently, we recorded the corresponding query response time. The depicted graph reveals that as the number of metrics used in the score (for evaluating ML models' testing data sets) increases, the runtime progresses linearly.

Discussion. The outcome of the experimental scenarios reveals encouraging and positive linear trends, indicating that the query execution response time increases proportionally with the expanding number of instances of ML models, models metadata, models data items, models features used in the models training data, and the metrics employed to compute the score related to models testing data sets. This shows a consistent relationship with a uniform growth between the various variables considered and the query execution time. Notably, the findings highlight certain scenarios where the growth is more substantial, specifically, in the graphs where we increased the number of ML model instances (see Fig. 11a), resulting in a time jump of 20 ms, and in the case where we increased the number of score metrics for ML model evaluation (see Fig. 12c), resulting in a time jump of 21 ms. In these scenarios, the impact is more significant than in others, where the time jumps are 15 ms, 4 ms, and 15 ms, respectively, in Figs. 11b and 12a, b.

In Fig. 11a, the notable time jump is attributed to the large number of ML model instances used (10000), while in Fig. 12c, the substantial jump can be attributed to the increased number of concepts targeted in the query (*SML:MachineLearningModel*, *SML:TestingDataSet*, *SML:Evaluation*, *SML:Score*, and *SML:Metric*). Additionally, it is observed that the increased number of metadata and the number of features used in ML model training data sets exhibit similar resulting curves with a time jump of 15 ms, despite differing variables: from 5 to 40 for the number of metadata, compared to 2 to 20 for the features number. This can be explained by the very close response times (405 and 407 ms) when both variables are equal to 20.

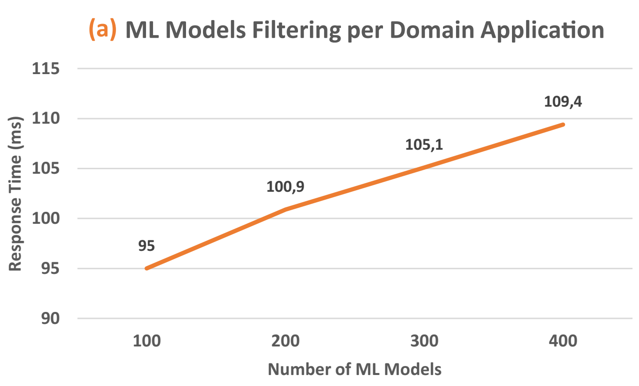
Regarding the increased number of data items in the training data sets of ML models, it has the least impact on the query response time, with a time jump of 4 ms.

5.2 ML Models and User Input Matching Evaluation

In this section, we test the performance of our solution when matching ML models specifications with user inputs. For this aim, we considered the following user request example[3] (refer to its formal definition in Eq. 1):

User Request = {Energy Efficiency,,Regression,}, UserDataFile.csv, {{1234},,true, {0.5,0.5,0.5,0.5}}, where: (1) "Energy Efficiency" and "Linear Regression" are values related, respectively, to "SML:ApplicationDomain" and "SML:Algorithm" entities, and used to filter the corresponding ML models, (2) UserDataFile.csv, is a file that contains time-series temperature data (i.e., "Date" and "Temperature" values), and (3) {{1234},,true, {0.5,0.5,0.5,0.5}} are user preferences given to the filtering process ordering, the resulted retrieved ML models (i.e., the ones having the highest score with top-max=true), and the weights used in the similarities measures.

[3] It is an example of a user request that helps in evaluating the ML models filtering process, as well as the matching similarities measures between user needs, including user given file, and ML models.



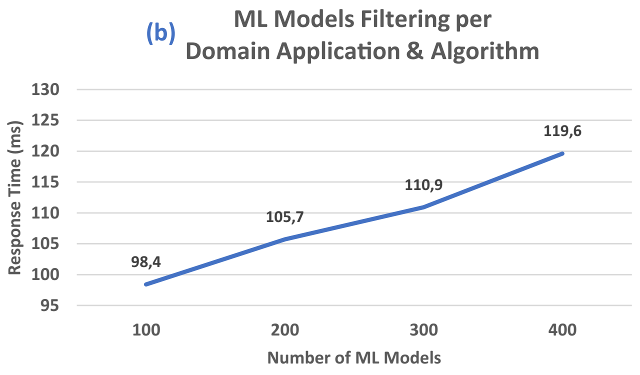


Fig. 13 Filtering process according to ML domain application and algorithm

5.2.1 Filtering Process Evaluation

The matching between user request and ML models specifications starts by reducing the search space of ML models before applying the necessary similarities measures. To do so, a filtering process of ML models is done according to the given set of words in user request[4] (in our example: "Energy Efficiency" and "Linear Regression", which are related, respectively, to the ML application domains and algorithms). In the experiments, in which we used the data presented in: http://tinyurl.com/ymjach45, we applied two different scenarios regarding the filtering process. In the first scenario, we varied the number of ML models between 100 and 400, and applied the filtering process based on "Energy Efficiency" value, that is related to the ML application domains (see Fig. 13a). In the second scenario, we used the same number of ML models (between 100 and 400), and executed the ML filtering process based "Energy Efficiency" and "Linear Regression", related, respectively, to ML application domains and algorithms (see Fig. 13b). In both scenarios, we retrieved the query response time (in milliseconds), computed according to the results average of 10 sequential executions for each filtering query. As it is shown in both curves, the response time of the filtering process evolves with the number of ML models. The increase in the response time is a bit more noticeable in the second scenario, compared to the first one (with a 6 ms of difference in average), since the filtering process is based on more than one criteria (i.e., application domain and algorithm) instead of only one (i.e., application domain).

[4] In the current tests, we did not rely on the Generative AI to enrich the set of keywords used for ML models filtering. This will be held in other dedicated work.

5.2.2 Similarities Measures Evaluation

Once the filtering process is done, the similarities measures between the user given file and ML specifications are launched, with respect to the user preferences. In our tests, based on the data presented in http://tinyurl.com/s23cyka5, which include the filtered ML models according to models application domains and algorithms (see Fig. 13b), we assumed the following:

- The feature (s), e.g., "Temperature", with their values data types that are included in the filtered ML models training data sets and the user given data set (in the user given file) are already obtained, using SPARQL queries applied on the SML model ontology.
- The 'DataFreqScore' and the 'TemporalOverlapScore' between the filtered ML models training data sets and the user given data set are, also, already retrieved.
- The filtered ML models training data sets and the user given data set correspond to the same zone, with 'SpatialSimScore' = 5/8.

Based on the previous assumptions, we show, in Table 8, some examples of the data used to conduct our similarities measures experiments. The similarities calculations have been realized between the filtered ML models (varied between 12 and 50 models) obtained from the filtering process in Fig. 13b, and the user given data file (i.e., UserDataFile.csv in this case). We considered that each of the filtered ML model training data set and the user given data set, has one single feature (apart from the 'Date' column). The features were compared together on the syntactic level (using the Levenshtein algorithm), as well as on the semantic level (using the Wup methodology), to calculate the 'FeatureSimScore'. Using the retrieved value types of the features included in the ML model training data sets and user data set (refer to Table 8), the 'FeatureValTypeScore' was

Table 8 Examples of data used to compute similarities measures between user input and ML specifications

ML Id	ML Feature	ML Feature Value Type	User Data File Feature	User Data File Feature Value Type	Data-FreqScore	Temporal-Overlap-Score
1	Temperature	String	Temperature	Float	6	0,50
2	Temperature	Float	Temperature	Float	6	0,50
3	CO2	Float	Temperature	Float	5	0,67
4	Energy	Float	Temperature	Float	8	0,42
5	Occupants	Integer	Temperature	Float	7	0,50
6	Temp	Float	Temperature	Float	6	0,75
7	CO2	Float	Temperature	Float	7	0,33
8	Temp	String	Temperature	Float	10	0,58
9	Energy	Float	Temperature	Float	9	0,50
10	Energy	String	Temperature	Float	9	0,67
11	Temperature	Float	Temperature	Float	5	0,33
12	CO2	Float	Temperature	Float	12	0,58

Table 9 Similarity measures values between the filtered ML models and user request, before retrieving the model with the highest GlobalSimScore

ML Id	FeatureSimScore	FeatureVal-TypeScore	TempSimScore	SpatialSimScore	GlobalSimScore
1	1	0	0,321	0,625	0,486
2	1	1	0,321	0,625	0,736
3	0,184	0	0,416	0,625	0,306
4	0,341	0	0,263	0,625	0,307
5	0,135	0	0,312	0,625	0,268
6	0,562	1	0,446	0,625	0,658
7	0,184	0	0,229	0,625	0,259
8	0,562	0	0,336	0,625	0,381
9	0,341	0	0,3	0,625	0,316
10	0,341	0	0,383	0,625	0,337
11	1	1	0,249	0,625	0,718
12	0,184	0	0,329	0,625	0,284

calculated. As per the 'TempSimScore' value, it has been computed according to the 'DataFreqScore' values (that have been normalized) and the 'TemporalOverlapScore' values. All of these scores ('FeatureSimScore', 'FeatureVal-TypeScore', and 'TempSimScore'), along with the 'Spatial-SimScore', have been used to compute the 'GlobalSim-Score' based on the Eq. 7. In Table 9, we show the different similarities scores applied between the filtered ML models listed in Table 8, and user request (defined in the beginning of Sect. 5.2). As presented in the table, we highlight in green the ML model (i.e., ML ID 2) having the maximum 'GlobalSimScore' value that is '0,736', corresponding to the best ML model matching user input.

In order to test the impact of the number of filtered ML models on the calculation of the similarity measures, we conducted experiments to observe the response time (in ms) to obtain the similarity scores when having different numbers of filtered ML models. The results, shown in Fig. 14, were obtained using Jupyter Notebook[5] with Python as the programming language. The experiments results reveal a positive linear relationship between the time taken for the similarity calculations and the number of filtered ML models. This can be attributed to the increased number of similarity measures that need to be computed with more filtered ML models.

6 Conclusion

This paper introduces a Semantic Machine Learning Model ontology called: SML, aimed at describing the characteristics and operational details of Machine Learning (ML) models. The descriptions encompass various elements such as ML models used algorithms, ML models metadata,

[5] https://jupyter.org/.

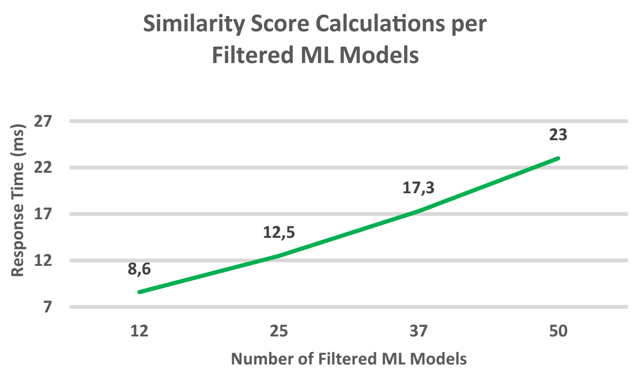


Fig. 14 Similarity score calculations applied on the filtered ML models

ML models training and testing datasets, and ML models evaluation metrics. SML facilitates the dissemination of ML knowledge across diverse platforms and environments, thereby enhancing the understanding of ML models and aiding their selection in different scenarios. Following the implementation of SML, we conducted evaluations to assess its efficiency and performance across various scenarios. Our experimental findings demonstrate promising and encouraging results.

Based on SML model, we also propose, in this work, an approach for retrieving ML models by leveraging the alignment between user inputs, which we formally defined, and ML models characteristics. Our approach relies on similarity measures related to four criteria: (1) Feature, (2) Feature Value Type, (3) Temporal Context, and (4) Spatial context. We have examined and experimented the proposed similarity computations to assess and identify ML models that closely align with user inputs.

As part of our ongoing assessment of SML ontology, our objectives include verifying its consistency to ensure that the defined concepts and properties do not introduce any structural inconsistencies. This verification process involves employing various reasoners. Additionally, we aim to assess its clarity by examining whether the names or labels of the concepts and properties are understandable to both experts and non-experts. Furthermore, we plan to integrate the SML ontology into real-world environments or projects to further evaluate its practical utility. Moreover, and based on the proposed similarities measures that we defined to study the matching between user inputs and ML models characteristics, we will focus on developing a dedicated recommendation engine that recommends the most appropriate ML model(s) for specific contexts and diverse scenarios, while giving some optimizations applicable for models adjustments (related to the existing models in SML or to the ML models that can be given by the user in his request), etc. And finally, we seek to study the usage of the Generative AI (on ML models metadata for instance) to enrich the set of keywords that are used for ML models filtering.

Acknowledgements Not applicable.

Author Contributions All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Lara Kallab. The first draft of the manuscript was written by Lara Kallab, and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding Not applicable.

Data Availability Enquiries about data and materials availability should be directed to the authors.

Declarations

Conflict of interest All authors declare that there is no conflict of interest regarding the publication of this article.

References

1. Hassani H, Silva ES, Unger S, Taj Mazinani M, Mac Feely S (2020) Artificial intelligence (ai) or intelligence augmentation (ia): what is the future? Ai 1(2), 8
2. Ngiam KY, Khor W (2019) Big data and machine learning algorithms for health-care delivery. Lancet Oncol 20(5):262–273
3. Janiesch C, Zschech P, Heinrich K (2021) Machine learning and deep learning. Electron. Markets 31(3):685–695
4. Lindholm, A., Wahlström, N., Lindsten, F., Schön, T.B.: Supervised machine learning. Department of Information Technology, Uppsala University: Uppsala, Sweden, 112 (2019)
5. Ayranci P, Lai P, Phan N, Hu H, Kolinowski A, Newman D, Dou D (2022) Onml: an ontology-based approach for interpretable machine learning. J Combin Optim 1–24
6. Braga J, Dias JL, Regateiro F (2020) A machine learning ontology
7. Lee I, Shin YJ (2020) Machine learning for enterprises: applications, algorithm selection, and challenges. Bus Horizons 63(2):157–170
8. Muhammad I, Yan Z (2015) Supervised machine learning approaches: a survey. ICTACT J Soft Comput 5(3)
9. Rashidi HH, Tran NK, Betts EV, Howell LP, Green R (2019) Artificial intelligence and machine learning in pathology: the present landscape of supervised methods. Academic Pathol 6:2374289519873088
10. Nascimento N, Alencar P, Lucena C, Cowan D (2018) A context-aware machine learning-based approach. In: Proceedings of the 28th annual international conference on computer science and software engineering, pp. 40–47

11. Mishra S, Jain S (2020) Ontologies as a semantic model in IoT. IJCA 42(3):233–243
12. Esnaola-Gonzalez I (2021) An ontology-based approach for making machine learning systems accountable. Semantic Web J
13. Publio GC, Esteves D, Ławrynowicz A, Panov P, Soldatova L, Soru T, Vanschoren J, Zafar H (2018) Ml-schema: exposing the semantics of machine learning with schemas and ontologies. arXiv preprint arXiv:1807.05351
14. Cao Y, Li S, Liu Y, Yan Z, Dai Y, Yu PS, Sun L (2023) A comprehensive survey of ai-generated content (aigc): a history of generative ai from gan to chatgpt. arXiv preprint arXiv:2303.04226
15. Chicco D, Warrens MJ, Jurman G (2021) The coefficient of determination r-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation. PeerJ Comput Sci 7:623
16. Rinartha K, Suryasa W, Kartika LGS (2018) Comparative analysis of string similarity on dynamic query suggestions. In: 2018 electrical power, electronics, communications, controls and informatics seminar (EECCIS), pp 399–404. IEEE
17. Ochelska-Mierzejewska J (2018) The evaluation of text string matching algorithms as an aid to image search. J Appl Comput Sci 26(1):33–62
18. Patro S, Sahu KK (2015) Normalization: a preprocessing stage. arXiv preprint arXiv:1503.06462
19. Zhu G, Iglesias CA (2016) Computing semantic similarity of concepts in knowledge graphs. IEEE Trans Knowl Data Eng 29(1):72–85
20. Vafaei N, Ribeiro RA, Camarinha-Matos LM (2018) Selection of normalization technique for weighted average multi-criteria decision making. In: Doctoral conference on computing, electrical and industrial systems, pp 43–52. Springer
21. Csiszar O (2021) Ordered weighted averaging operators: a short review. IEEE Syst Man Cyber Mag 7(2):4–12
22. Sahoo SK, Goswami SS (2023) A comprehensive review of multiple criteria decision-making (MCDM) methods: advancements, applications, and future directions. Dec Making Adv 1(1):25–48
23. Zlaugotne B, Zihare L, Balode L, Kalnbalkite A, Khabdullin A, Blumberga D (2020) Multi-criteria decision analysis methods comparison. Rigas Tehniskas Universitates Zinatniskie Raksti 24(1):454–471