InterpretME : A Tool for Interpretations of Machine Learning Models Over Knowledge Graphs

Yashrajsinh Chudasama [a,b,*], Disha Purohit [a,b], Philipp D. Rohde [a,b,c], Julian Gercke [b] and Maria-Esther Vidal [a,b,c]

[a] *TIB Leibniz Information Centre for Science and Technology, Hannover, Germany*
E-mails: yashrajsinh.chudasama@tib.eu, disha.purohit@tib.eu, philipp.rohde@tib.eu, maria.vidal@tib.eu
[b] *Leibniz University, Hannover, Germany*
E-mail: julian.gercke@stud.uni-hannover.de
[c] *L3S Research Center Germany, Hannover, Germany*

Abstract. In recent years, knowledge graphs have been considered as pyramids of interconnected data enriched with semantics for complex decision-making. The potential of knowledge graphs and the demand for interpretability of machine learning models in diverse domains (e.g., healthcare) have gained more attention. The lack of model transparency impacts negatively the understanding and, in consequence, interpretability of the predictions made by a model. Data-driven models should be empowered with the knowledge required to trace down their decisions, and the transformations made to the input data to increase model transparency. In this paper, we propose InterpretME, a tool for fine-grained representations, in a knowledge graph, of the main characteristics of trained machine learning models. They include data- (e.g., features' definition and SHACL validation) and model-based characteristics (e.g., relevant features, and interpretations of prediction probabilities and model decisions). InterpretME allows for the definition of a model's features over knowledge graphs; SHACL states domain integrity constraints. InterpretME traces the steps of data collection, curation, integration, and prediction, and documents the collected metadata in a knowledge graph. InterpretME is publicly available as a tool; it includes a pipeline for enhancing the interpretability of machine learning models, and a knowledge graph and an ontology to describe the main characteristics of trained machine learning models.

Keywords: Interpretability, Knowledge Graphs, Machine Learning Models, SHACL, Ontologies

1. Introduction

Interpretability is the degree to which humans can understand the decisions made by computational frameworks. Specifically, in Artificial Intelligence (AI), the higher the interpretability of predictive models, the easier for humans to understand why a model makes certain decisions. The recent advancements and complexity of machine learning methods have demonstrated their success in forecasting complex problems (e.g., disease diagnosis or progression [1, 2]). However, they are often opaque and usually do not provide interpretation for their predictions, and the interpretability of the outcomes is not always achieved. Interpretable predictive models have rapidly become a relevant problem [3]. Nevertheless, although various tools aim to interpret the algorithmic decisions of machine learning models [4, 5], they are incapable of capturing the knowledge required to translate a model's insights into

[*] All the authors have equally contributed to this work.

the application domain. On the contrary, knowledge graphs encode data and knowledge; they, together with domain ontologies (e.g., ML Schema [6]), represent building blocks for increasing the understanding of the behavior and effects of a predictive model. **Our Tool:** we propose an analytical tool, named **InterpretME**, for tracing and explaining the predictive models built over data collected from KGs. InterpretME implements a set of validating integrity constraints that provide a meaningful description of a target entity of a prediction model, and its main properties. The current version of InterpretME is customized for supervised machine learning models (e.g., decision trees) and interpretable tools (e.g., LIME [5]). InterpretME is publicly available as a resource in GitHub[1], Zenodo[2], and in PyPI[3]. We empirically evaluate InterpretME in real-world knowledge graphs regarding execution time, interpretations quality, and traceability of target entities. The observed results reveal the key role of Semantic Web technologies in interpretable predictive models.

The rest of the paper is structured as follows: Section 2 describes the main concepts and motivates our work. Section 3 defines the InterpretME architecture, while Section 4 reports the results of our experimental studies. Section 5 presents the main characteristics of InterpretME as a resource. Section 6 discusses the state of the art, and our conclusions and future work are outlined in Section 7.

2. Preliminaries and Motivation

2.1. Main Concepts

2.1.1. Predictive Modeling Frameworks.

Predictive Modeling comprises methods to forecast future outcomes based on data encoding what has happened in the past (e.g., historical data) and what is currently happening (e.g., current data). Predictive modeling resorts to a variety of models and algorithms (e.g., random forest, gradient boosted model, or decision trees) to solve predictive problems (e.g., classification or outlier detection) [7]. Despite the large spectrum of mature models and well-defined problems, predictive modeling is a complex task that requires user expertise in the domain context and modeling. De Bie et al. [3] conceptualizes predictive modeling in four stages, as depicted in Figure 2.

a) *Data Engineering* comprises the tasks of acquiring, organizing, curating, and preparing data for the prediction modeling. **b)** *Data Exploration* includes the understanding of an application domain, and the interpretation of missing values, integrity constraints, and data structures. **c)** *Model Building* involves the selection of the predictive model (e.g., machine or deep learning), target function, hyperparameters, and relevant features. **d)** *Exploitation* covers the transformation of the predictions into the decisions on the application domain.

Predictive modeling complexity and the mandatory requisite of knowledgeable users motivate automated techniques for facilitating the implementation of the stages in Figure 2. De Bie et al. [3] identify three forms of automation: **i)** *Mechanization* applied in well-known tasks that can be algorithmically performed. **ii)** *Composition* conducted on a sequence of tasks implemented as workflows specified in high-level programming languages. **iii)** *Assistance* implemented to enhance interpretability and user efficiency by presenting visualizations, prediction scores, and explanatory model decisions.

Current developments in automation have mainly impacted the stages of *Model Building* and *Exploitation*. In particular, Automated Machine Learning (AutoML) systems (e.g., AutoML[4] and AutoWeka [8]) successfully implement mechanization, and can automatize the processes of model and feature selection, and the optimization of the hyperparameters and target functions. Furthermore, interactive tools like RapidMiner[5], the Local Interpretable Model-agnostic Explanations (LIME) framework [5], and SHapely Additive exPlanations (SHAP) [4] provide assistance by reporting explanations of a model's results. Specifically, LIME relies on local surrogate models to explain the prediction of each entity of a test dataset. It computes, for the tested entities, the feature contribution for each

[1] https://github.com/SDM-TIB/InterpretME
[2] https://doi.org/10.5281/zenodo.7025752
[3] https://pypi.org/project/InterpretME/
[4] https://www.automl.org/
[5] https://rapidminer.com/

target class and the prediction probability. Similarly, SHAP explains individual entity predictions and calculates shapley values from coalitional game theory, indicating how to distribute the prediction fairly among the attribute values of a tested entity.

2.1.2. Knowledge Graph Frameworks.

The Semantic Web [9] aims at humans and machines working cooperatively in data exchange. Technologies capable of encoding semantics have been defined to achieve this goal. One of these technologies is the Resource Description Framework (RDF) [10]; the W3C standard for publishing and exchanging data over the web. An RDF graph $\mathcal{G} = \langle V_{\mathcal{G}}, E_{\mathcal{G}} \rangle$ is a directed graph with labeled edges [11]. The nodes represent subjects and objects of RDF triples, while predicates correspond to the labels of edges between nodes. We will use the terms RDF graph and knowledge graph [12] interchangeably.

SPARQL [13] is the W3C recommendation language to query RDF graphs. SPARQL 1.1 [14] provides the SERVICE clause and allows for specifying a federated query over various RDF graphs accessible via SPARQL endpoints. However, usually, SPARQL endpoints prohibit using these clauses. Federated query engines offer the possibility to answer federated queries without the need to know from where parts of a query will be answered [15].

The Shapes Constraint Language (SHACL) [16] is the W3C recommendation language to define integrity constraints over RDF data. SHACL integrity constraints are expressed in RDF and modeled as a network of shapes, called *shape schema*. A shape consists of **i)** a definition of the target; usually an RDF class or set of nodes, and **ii)** a set of constraints that are imposed over the instances of the target. Constraints can also link shapes to each other, hence, a shape network. An RDF graph's entity that matches the target definition of a shape, satisfies the shape if it validates all the constraints of the shape; the problem is, in general, intractable [17]. However, algorithms have been proposed and implemented to validate tractable SHACL fragments [18, 19] efficiently.

2.2. Motivating Example

The motivation of our work originates from the lack of automated assistance despite great potential of integrating knowledge graphs with predictive modeling frameworks. Tracing and explaining the predictive models built over data collected from the KGs in order to provide assistance is the main goal of our tool **InterpretME**. Even though the state of the art successfully has developed automated machine learning systems, through mechanization and composition, pipelines for predictive modeling are unable to generate human- and machine-readable decisions to assist users and enhance their efficiency. Figure 1 depicts a predictive modeling pipeline, where an automated machine learning system (e.g., AutoML) is utilized for model and feature selection, and hyperparameters' optimization. Moreover, interpretable tools (e.g., LIME and SHAP) are providing interpretation results. Figure 1 illustrates the pipeline **B**; an input dataset **A** is collected from an **RDF knowledge graph** that integrates data about lung cancer patients. This dataset includes features describing the main characteristics of a lung cancer patient, i.e., patient identifier (a.k.a. EHR), Gender, Age, cancer stage (a.k.a. CancerStage), smoking habits (a.k.a. SmokingHabit), and lung cancer Biomarkers. Additionally, the dataset comprises information about the cancer types of the relatives of the lung cancer patients; this information is maintained in the features CancerType and FamilialDegree. The predictive task is a binary classification to predict if a patient will be positive for the biomarker ALK or by any other biomarker. AutoML performs model selection and hyperparameter optimization **C**; based on AutoML recommendations, the random forests and decision tree models are selected to implement the classification problem. Further, the interpretable surrogate tools LIME and SHAP are utilized to provide local interpretations of each patient in the test dataset. SHAP yields the relevant features which contributes to the model outcomes **D**. **E** depicts an exemplar entity where LIME determines a prediction probability of 0.7 to belong to the target class ALK (i.e., Class 0), otherwise, 0.3 target class Others (i.e., Class 1). LIME also identifies the top-10 relevant features for the target entity and assigns weights. These outcomes allow for understanding the quality of the implemented pipeline. Nevertheless, when they are reported to oncologists, many questions may still arise **F**: **Q1)** Who is this patient interpreted by LIME? **Q2)** How does the Stages feature contribute to the classification of this patient in the target class ALK? **Q3)** Which other features are relevant for this classification? **Q4)** Does this patient satisfy the domain integrity constraints? **Q5)** What are the main characteristics of this patient? Although, our user would have been able to interpret the results produced by LIME or SHAP, he/she would need to trace back these results to the original

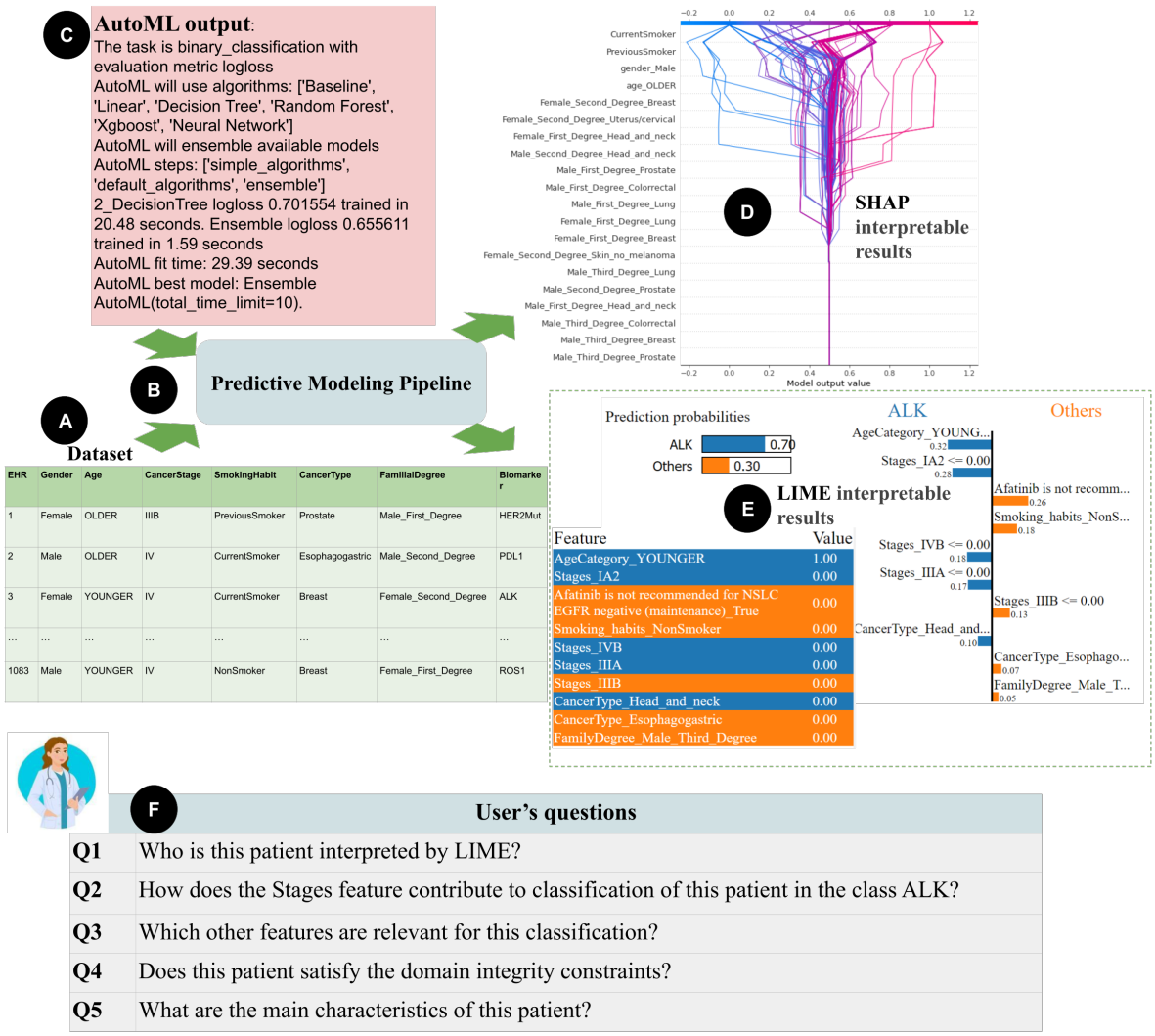

Fig. 1. Motivating Example

data properties to know if, for example, the reported patient violates the domain constraints or not. InterpretME combines local explanations with knowledge graphs and SHACL constraints to ensure traceability.

3. InterpretME: An Interpretable Pipeline for Predictive Modeling over Knowledge Graphs

3.1. Tracing Metadata in Machine Learning Pipelines

We aim to collect metadata during the different stages of a predictive model pipeline. Figure 2 shows, in a pictorial view, the characteristics of the data collected towards the improvement of automation *assistance*. In the data engineering stage, InterpretME captures metadata from the input KG (e.g., features' definition, endpoint, and target classes) and records the SHACL constraints, used for data validation. During the model building, suggestions from AutoML systems can be considered. InterpretME traces the optimized hyperparameters and estimated features' relevancy, and records the model performance metric outcomes (e.g., precision) for a particular run. Moreover, SHACL validation reports are stored. Tracing the metadata collected from input KGs will help to provide explanations about the predictions made by the predictive models.

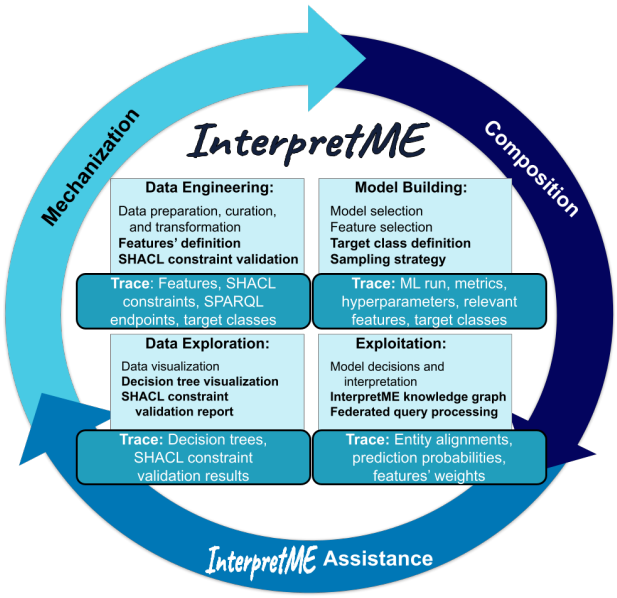

Fig. 2. Tracing Metadata in Machine Learning Pipelines

InterpretME also exploits decision trees and visualizes the validation report to enhance exploration. Lastly, for exploitation, InterpretME aligns target entities with entities in KGs, where the model's features are defined. This semantic context allows for **i)** tracing back the main properties of target entities; **ii)** understanding the interpretability models' results (e.g., LIME prediction probabilities); and **iii)** checking SHACL validation reports. InterpretME identifies uniquely all the collected metadata.

3.2. The InterpretME Architecture

Figure 3 depicts the InterpretME architecture. InterpretME is a tool for fine-grained representations of the main characteristics of trained predictive modeling framework. The architecture of InterpretME deals with training the predictive models and collecting information generated as output of predictive models (i.e., model accuracy, list of important features, prediction probabilities, and classified classes for each instance), and provides assistance (Figure 2) to the user by tracing metadata to generate instances of the InterpretME KG and executes federated queries on top of the InterpretME KG and the input KG to trace back, and answers all the questions in Figure 1. InterpretME takes a JSON file input (i.e., endpoints of KGs, features' definition, target definition, SHACL constraints, sampling strategy, and class definition) from the user[6]; a SPARQL query is generated based on the feature definition given by the user and the query is used to retrieve the application domain data from the input KGs. The structure of the query enables entities in the input KGs can be aligned to the identifiers of the instances in the ML models' datasets. These alignments facilitate the SHACL validation and federated query processing over the InterpretME KG and the input KGs. **1.** InterpretME evaluates the SHACL constraints over the nodes of the input KGs and generates a validation report per constraint and target entity. These results indicate whether an entity validates or invalidates the constraints defined by the user. These validation reports states the validity of data used by the predictive models[7]. Therefore, SHACL constraints are a crucial part of the InterpretME KG, to identify if a particular entity is violating the integrity constraints, where *True* represents the particular entity is valid, inversely represented by *False*.

2. Data preparation includes transforming data collected from the input KGs into a form that can be used in the training of a predictive model. Many machine learning models cannot handle categorical values directly,

[6] https://github.com/SDM-TIB/InterpretME/blob/v1.1.1/example/example.json
[7] https://github.com/SDM-TIB/InterpretME/blob/v1.1.1/InterpretME/pipeline.py#L105-L137

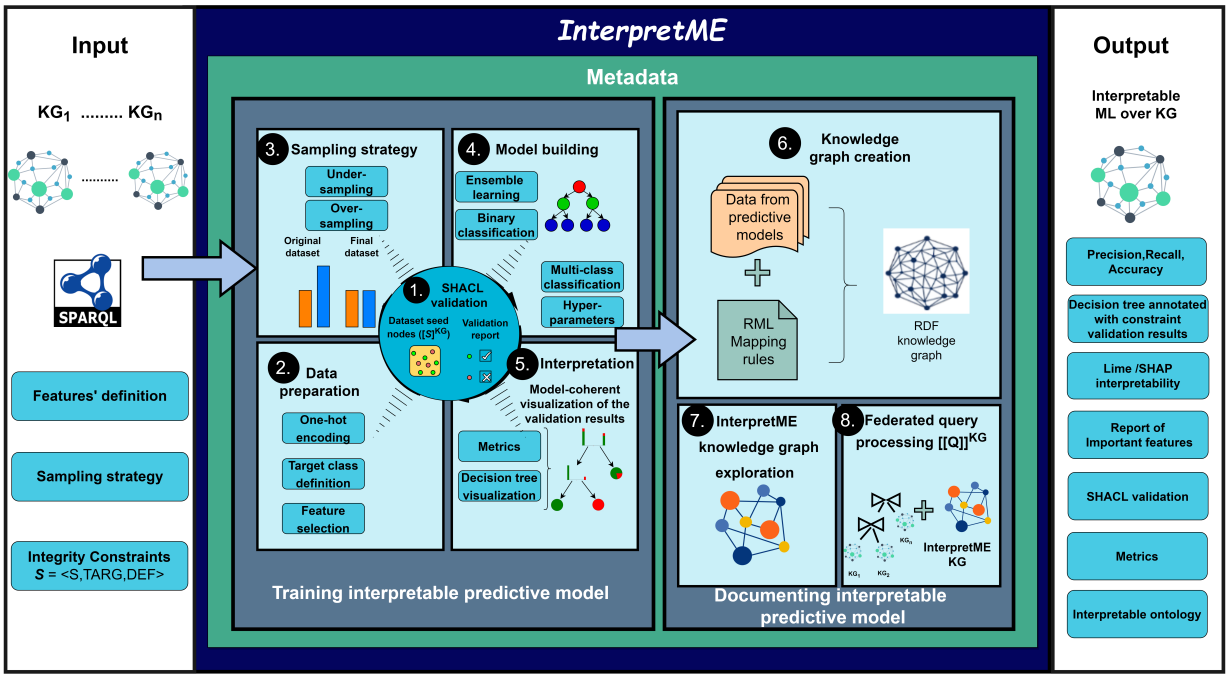

Fig. 3. The InterpretME Architecture

InterpretME creates one hot-encoding technique using Python library (e.g., sklearn.preprocessing) to transform the model's categorical features into binary features usable for predictive models[8]; it makes training data more expressive, and can be easily re-scaled. Also, the target class can be defined in InterpretME. The Lung Cancer KG have many categorical features, e.g., Age, Gender, Biomarker, CancerType, and FamilyDegree.

3. Sampling strategies (e.g., under-sampling or over-sampling) are also received to reduce data imbalance so that the machine learning algorithms can perform better. Many machine learning algorithms, like decision trees, random forests, and neural networks resort to class distribution in the training dataset to compute the probability of instances in each class when the model will be used to make predictions.

4. Model building can be done based on automated systems' selections (e.g., AutoML/OpenML) or based on a user's preferences. Automated tools can be used to generate optimized hyperparameters for predictive models. Here, the automated model can also perform stratified shuffle split cross-validation with random forest, and identify the relevant features; they are used to train a decision tree classifier to predict and visualize the outcomes[9].

5. Interpretation is utilized to enhance the understanding of the trained predictive model. Also, a visualization of the decision tree can be obtained with associated constraints. This module provides more insights of a validated/invalidated feature, and interpretable surrogate tools like SHAP [4] and LIME [5] (as in Figure 1) yield the most relevant features and local explanations list. They reflect the contribution of each feature to the prediction.

6. Knowledge graph creation is a module of InterpretME, where mapping rules are defined with the RDF Mapping Language (RML) [20] using the data collected from the predictive models. RML triple map; it is composed of *rml:logicalSource*, *rr:subjectMap*, *rr:predicateObjectMap*. The mapping rule component **(1)** states the logical source using the term *rml:logicalSource*. **(2)** The *rr:subjectMap* element defines the class of subject. **(3)** The term *rr:predicateObjectMap* establishes how the predicate *intr:hasDefinition* will be populated with the collected data.

7. SPARQL queries can be executed to explore InterpretME results of a particular target entity. The InterpretME KG provides clarification and eases the interpretation of the model's prediction of a particular entity aligned with the SHACL validation results.

[8] https://github.com/SDM-TIB/InterpretME/blob/v1.1.1/InterpretME/preprocessing_data.py#L57-L115
[9] https://github.com/SDM-TIB/InterpretME/blob/v1.1.1/InterpretME/classification.py

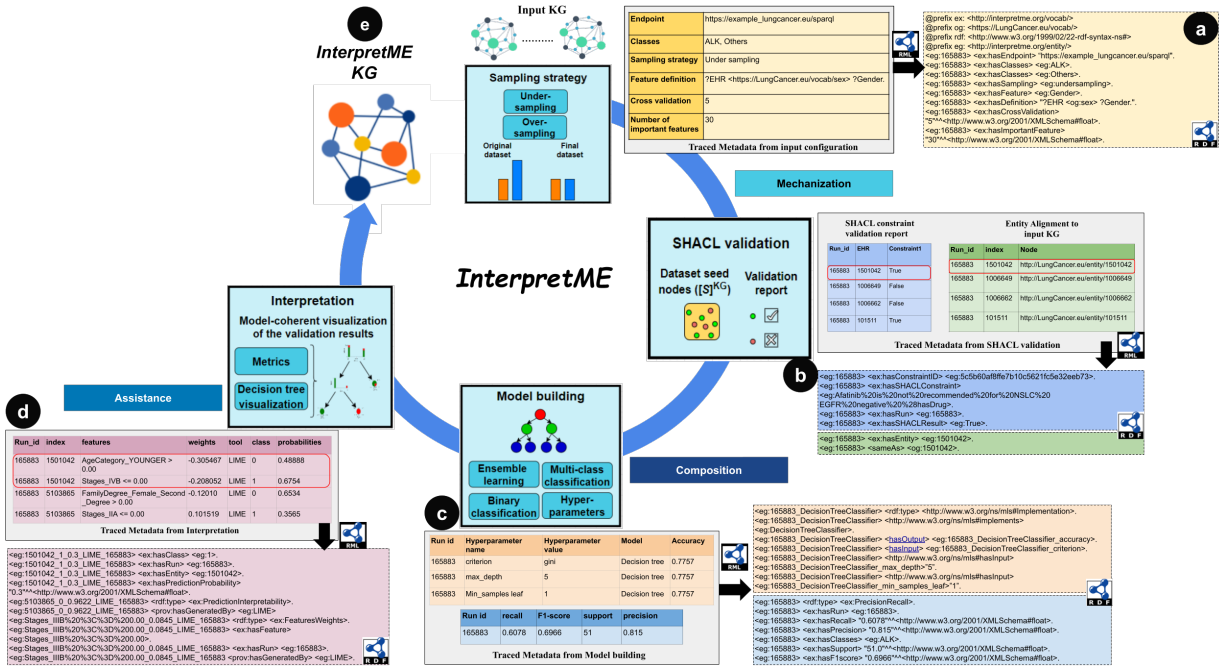

Fig. 4. Running example

8. Federated query processing is utilized to query both the input KGs and the InterpretME KG. Mostly interpretable tools results are related with the target entity in the dataset, which is hard to interpret and understand the characteristics of the target entity. Thus, a user can trace back via query federation to interpret the results for a specific target entity with input characteristics aligned with SHACL validation results. Exemplar queries are shown in InterpretME GitHub repository[10].

3.3. Running example

To exemplify different components of the InterpretME architecture, we have used the following running example (Figure 4). The prerequisites to run an example of the **French Royalty KG** with InterpretME is available[11].

a (Lung Cancer KG). This KG integrates the features' and class target definitions about lung cancer patients; their constraints are defined in terms of SHACL. Features' definitions are classified into independent and dependent variables; they are used later in the predictive modeling pipeline and are defined as following:

```
{
  "Endpoint": "https://example_lungcancer/sparql",
  "Index_var": "EHR",
  "Independent_variable": {
    "Gender": "?EHR <https://LungCancer.eu/vocab/sex> ?Gender."
  },
  "Dependent_variable": {
    "Biomarker": "?EHR <https://LungCancer.eu/vocab/hasBio> ?Biomarker."
  }
}
```

[10] https://github.com/SDM-TIB/InterpretME/tree/v1.1.1/example/queries/template
[11] https://github.com/SDM-TIB/InterpretME/v1.1.1/example/InterpretME_french_royalty.ipynb

where **EHR** is electronic health record of a patient, the first part states the feature name (i.e., *Gender*), and the later part (i.e., *<https://LungCancer.eu/vocab/sex>*) describes the feature in the KG. Figure 4 shows the traced input configuration, i.e., features' definitions, sampling strategies, class definitions, number of important features, SPARQL endpoint that are represented in the form of *RDF triples*, later used for the **InterpetME KG**. In the current running example Figure 4, target class (Biomarker) is imbalanced (i.e., *patient with target class ALK is 872 instances while patient with Others as target class is 432*). Therefore, the under-sampling technique[12] is selected for this use case to tackle the problem of imbalance.

b Considering our running example in Figure 4, the SHACL shapes represent integrity constraints. A SHACL schema defines integrity constraints, used by InterpretME for validation results and for uncovering their impact on a model's decisions. Here, the *"Constraint": "Afatinib is not recommended for NSCLC EGFR negative (hasDrug)"* are applied over the extracted data from input KG. The above constraint states a medical protocol that Afatinib is a drug which is not recommended for a patient having EGFR negative biomarker. Figure 4 depicts the traced metadata about SHACL validation, i.e., *EHR: 1501042 with EGFR positive satisfying the constraint, while EHR: 1006649 with EGFR negative violates the constraint*, an entity alignment is performed to trace original entity of input KG with SHACL validation results and predictive modeling pipeline.

c The preprocessed and sampled data is then fed to the automated tools for model building (e.g., ensemble learning) and optimized hyperparameters' selection for the predictive task. Here, the automated model can also perform stratified shuffle split cross-validation with models like *Random forest, AdaBoost classifier, Gradient boost classifier*, and identify the relevant features; they are used to train a decision tree classifier to predict and visualize the outcomes[13]. InterpretME stores the metadata about evaluation of the model for a particular run.

d The trained predictive model in **c** can be visualized by means of *Decision trees*; they can be visualized with SHACL constraints to observe which subtree validates or invalidates the protocol[14]. To understand the predictive model's outcomes, InterpretME resorts to interpretable tools, e.g., LIME, to provide local explanation of each entity. LIME provides prediction probability of the target class *ALK (i.e., Class 0)* as 0.65 and target class *Others (i.e., Class 1)* as 0.35. The relevant features list is often used to illustrate which features may cause a change in prediction of the trained model. LIME results are also represented in the InterpretME KG.

e Traced metadata can be semantified using RML mappings. RML mapping rules specify the role of transforming collected metadata into RDF triples for the InterpretME KG[15]. The unified schema of InterpretME represents the meaning of data in the InterpretME KG[16]. The following RML syntax is used to define one of the mapping file:

```
<InputEndpoint>
  rml:logicalSource [ rml:source "interpretme/files/endpoint.csv";
                      rml:referenceFormulation ql:CSV; ];
  rr:subjectMap [
    rr:template "http://interpretme.org/entity/{run_id}_{endpoint}";
    rr:class intr:Endpoint ];
  rr:predicateObjectMap [
    rr:predicate intr:hasEndpoint;
    rr:objectMap [ rml:reference "endpoint" ] ] .
```

The SDM-RDFizer [21] semantically enriches and integrates the metadata into the InterpretME KG. The generated RDF data is uploaded to an instance of *Virtuoso*. Entities in the InterpretME KG and the input KGs are aligned to ensure traceability. The InterpretME federated query engine allows for tracing back the ML models' results, i.e., main properties of target entities, LIME prediction probabilities, and SHACL validation reports. The short tutorial is available to demonstrate the use of InterpretME in the example[17] of the GitHub repository.

[12] https://github.com/SDM-TIB/InterpretME/blob/v1.1.1/InterpretME/sampling_strategy.py
[13] https://github.com/SDM-TIB/InterpretME/blob/v1.1.1/InterpretME/classification.py
[14] https://github.com/SDM-TIB/InterpretME/tree/v1.1.1/images
[15] https://github.com/SDM-TIB/InterpretME/tree/v1.1.1/InterpretME/mappings
[16] https://github.com/tibonto/InterpretME
[17] https://github.com/SDM-TIB/InterpretME/blob/v1.1.1/example/InterpretME_french_royalty.ipynb

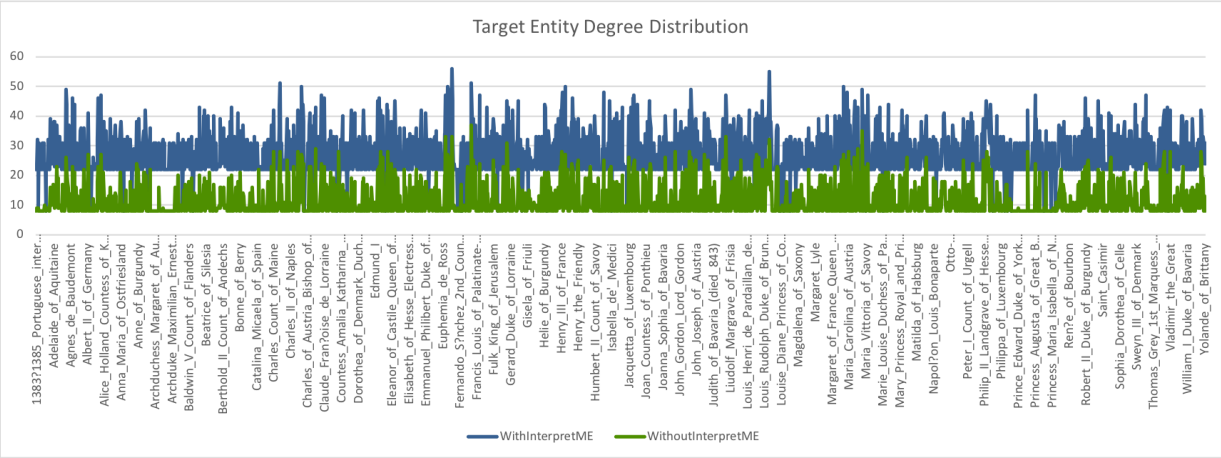

Fig. 5. Degree distributions of the entity targets in the French Royalty KG. The distribution in blue depicts the degrees of the target entities semantically enhanced with InterpretME, while green shows their original degree distribution.

4. Empirical Evaluation

We evaluate InterpretME with the goal of answering the following research questions: **RQ1)** What is the impact of integrating predictive modeling frameworks with KGs to enhance interpretability? **RQ2)** Can InterpretME trace decisions made by predictive models? **RQ3)** To which extent does the InterpretME KG satisfy standard quality criteria? **RQ4)** How much is the observed overhead at each stage of InterpretME? The experimental settings are as follows:

Benchmarks. Table 1 presents our KGs in numbers. **The French Royalty KG [22]** is fully curated; for each person in the KG, we added the class `dbo:Person` as well as the different number of children, predecessors, and further triple related counts. Further, we extended the KG with the rule-based derived `dbo:hasSpouse` relationships from [22]. Forecasting whether a member of the French royalty has a spouse is our predictive task. In the French Royalty KG, a SHACL constraint explains a `dbo:hasSpouse` link for each logical rule proposed in [22]. We added ten SHACL constraints. Each of the SHACL constraints validates a person, if it can be explained with the original rule that the person has a spouse.

The Lung Cancer KG [23]. We use a private KG from the biomedical domain, which integrates lung cancer patients mentioned in Section 2. This KG includes information describing the main characteristics of a lung cancer patient.The prediction task is a binary classification to predict the biomarker of the patient, which can be 'ALK' or 'others'. The SHACL constraints used are the medical protocols that recommend when treatments should be prescribed according to a patient's biomarker values; we defined four different SHACL constraints stating that EGFR negative patients should not take Afatinib or Gefitinib.

Engines. InterpretME resorts to Trav-SHACL [19] to validate SHACL constraints. It is accessed through a wrapper to minimize the execution cost by restricting the SHACL validation to the nodes in a KG required for verification.

Metrics. *Execution time:* Elapsed time spent to execute a prediction pipeline and create the KG; it is measured using the Python library `time`. Experiments are executed five times and an average is reported; they are run on a machine equipped with an Intel® Core(TM)i7- 10850H at 2.71 GHz and 16 GiB RAM.

4.1. Degree Distribution of the InterpretME KG

In graph theory and complex networks, an entity's degree (or the number of neighbors ignoring edges' direction) is defined as the number of relations with other entities. InterpretME traces the entities of the target classes (e.g., HasSpouse) defined as independent variables via SPARQL queries over the input KGs. In the InterpretME KG, these entities are described in terms of metadata collected by InterpretME components (e.g., hasInterpretedFeature, hasFeatureWeight, hasEntityClassProbability, hasPredictionProbability, hasSHACLResult, hasSHACLConstraint).

Figure 5 depicts the degree distribution results of French Royalty KG; the average number of neighbors is *WithInterpretME*: 26.99 (with a standard deviation of 6.94) and 11.39 (with a standard deviation of 5, 06) *WithoutInterpretME*. InterpretME increases the number of RDF triples that describe a target entity; a user can query these triples to explore the predictive model decisions. The InterpretME KG properties are retrieved. These values of degree distribution quantify the information gain for each target entity in terms of all what InterpretME traces. The execution of queries 1 and 4 [18] on the InterpretME KG retrieve the values of these properties for interpretME:Louis_XIV. As a result of tracing the machine learning pipeline, the number of properties of interpretME:Louis_XIV (i.e., outdegree) goes from 32 in the French Royalty KG to 55, and thus entity degree is increased. Based on these outcomes, we can answer the **RQ1)**, and InterpretME enhances the interpretability of a target entity.

Table 1
Statistics about Input KG and InterpretME KG

Knowledge Graph		#triples	#entities	#predicates	#objects	#triples /#entities
French Royalty	Input KG	31,599	3,439	133	4,390	9.18
	InterpretME KG	275,806	36,384	152	44,109	7.58
Lung Cancer	Input KG	77,466,343	21,420,170	357	15,053,863	2.30
	InterpretME KG	62,403	8,881	152	11,546	7.03

4.2. Traceability

We evaluate InterpretME in terms of the traceability of a target entity. Table 2 reports on the average number of answers to the type of questions presented in our motivating example (Section 2). InterpretME efficiently traces the target entity and provides the user with additional information about the prediction probability of the entity. Also, it helps users to uncover relevant features of an entity that contribute to the prediction, with assigned weight distribution for the top-10 features. The federated query engine, DeTrusty [24], evaluates SPARQL queries to retrieve data from the original KG, the InterpretME KG, or both. Instances in the InterpretME KG are linked to the entity in the original KG via `owl:sameAs`. The KGs are accessible via SPARQL endpoints. In Table 2, the questions presented in Section 2 are expressed as SPARQL queries over the two KGs. For the French Royalty KG, InterpretME models LIME interpretations in terms of 20 (on average), and the contribution of a given feature is documented with 2 RDF triples on average for binary classification task. Moreover, target entities satisfy the 10 domain integrity constraints, and the average number of main characteristics of the target entities in the French Royalty KG is 12.24. The average number of answers in the Lung Cancer KG is higher because a lung cancer patient is described in more detail than a person in the French Royalty KG. Based on these outcomes, we can positively answer **RQ2)** because InterpretME can translate back to the original KGs, the decisions made by the predictive models.

Table 2
Average Number of Answers per Target Entity to Questions from Figure 1

ID	Question	Avg Answers in	
		Lung Cancer KG	French Royalty KG
Q1	Which is the target entity interpreted by LIME?	20.57	10.00
Q2	How does *feature* contribute to the classification of this entity?	2.05	2.00
Q3	Which other features are relevant for this classification?	9.77	4.38
Q4	Does this target entity satisfy the domain integrity constraints?	4.00	10.00
Q5	What are the main characteristics of the target entity?	46.28	12.24

[18] https://github.com/SDM-TIB/InterpretME/blob/v1.1.1/example/queries/french_royalty

Table 3
Evaluation Metrics

Metric	Value	Explanation	Formula (in [25])
Accuracy			Section 2.2.1
Synt. validity of RDF doc	1	RDF files are valid	$m_{synRDF}(g)$
Synt. validity of literals	1	Literals have a label that defines a range of possible values	$m_{synLit}(g)$
Semant. validity of triples	1	Literals are collected from input or can be measured	$m_{semTriple}(g)$
Trustworthiness			Section 2.2.2
KG level	0.25	Automated data curation from structured data sources	$m_{graph}(hg)$
Statement level	0.5	Provenance is traced	$m_{fact}(g)$
Unknown/empty values	0	Empty values are not indicated	$m_{NoVal}(g)$
Relevancy			Section 2.3.1
Ranking of statements	0	Ranking is not useful in this context	$m_{Ranking}(g)$
Ease of Understanding			Section 2.4.1
Description of resources	1	Resources have label and comment	m_{Descr}
Labels in multiple lang	0	Labels are only available in English	$m_{Lang}(g)$
RDF serialization	1	Serialization in Turtle	$m_{uSer}(hg)$
Self-describing URIs	0.5	Partial use of self-describing URIs	$m_{uURI}(g)$
Interlinking			Section 2.5.3
Interlinking via owl:sameAs	1	Use of owl:sameAs for linking the InterpretME KG to the input KG	$m_{Inst}(g)$

Table 4
Average execution time (secs.) for KGs per pipeline stage

Components	French Royalty KG	Lung Cancer KG
SHACL Validation	1.64	26.74
Data Curation	0.33	0.46
Model Training	0.01	0.01
Decision Trees	3.25	4.43
Constraint Visualization	0.01	0.01
LIME	16.48	8.07
Semantification	44.55	15.87

4.3. The Quality of the InterpretME Knowledge Graph in Numbers

This section answers **RQ3)** and reports the results of the evaluation of the InterpretME KG in terms of the quality metrics proposed by Färber et al. [25]; four data quality categories are considered: **i)** Intrinsic category is independent of the use case context; **ii)** Contextual category depends on the application context of a data consumer; **iii)** Representational category quantities the form how information is available; and **iv)** Accessibility category determines how data can be accessed. Table 3 depicts all the metrics by category from Färber et al. [25] to check the quality of the InterpretME KG. They help the user to better estimate the pros and cons of InterpretME. Färber et al. [25] propose extended quality parameters and category (i.e, trustworthiness and interlinking). The scores for quality metrics (Table 3) are assigned manually, following scores' definition by Färber et al. [25]. Since, the InterpretME KG covers a new domain of structured data, the eloquence of several suggested criteria is still relatively low. However, we are optimistic that these values will be increased once InterpretME starts to be used in real-world applications.

4.4. Execution Time

This section describes Table 4 and aims at answering **RQ4)**, i.e., the overhead caused by each of the components of InterpretME. The analysis of the execution times measures the impact of different parameters on the performance

of the InterpretME steps. We have calculated the average execution time based on the two use cases, French Royalty and Lung Cancer KGs by taking the average of five runs per use case. The French Royalty KG comprises RDF triples 31,599, while the Lung Cancer KG comprises 77,466,343 RDF triples (Table 1), i.e., the Lung Cancer KG is 3.8 orders of magnitude larger than the French Royalty KG. The average execution time of the first component, i.e., the SHACL validation, is 1.64 and 26.74 seconds, respectively. The second component – data curation – where data is extracted and preprocessed to be given to the machine learning model, takes on average 0.33 and 0.46 seconds in the two use cases, respectively. The next component is training the model; the average execution time is 0.01 seconds for the French Royalty KG and the Lung Cancer KG. Further, the decision tree component, where the decision trees for the model are generated, takes on average 3.25 and 4.43 seconds, respectively. Using an interpretable tool (e.g., LIME), generating the interpretable results requires 16.48 and 8.07 seconds in the use cases, respectively. Lastly, the *semantic enrichment* of the traced metadata takes 44.55 and 15.87 seconds for the French Royalty KG and the Lung Cancer KG, respectively. These results are consistent with the fact [19] that the KG size and the number of SHACL constraints impacts on the SHACL validation. As a result, the ML pipeline execution time is also impacted, as well as the generation of the InterpretME KG instances.

5. InterpretME as a Tool

The InterpretME is publicly available as a Python library on PyPI[19]. While the core of the pipeline is entirely new, InterpretME reuses available tools from the community. InterpretME *pipeline()* receives a JSON file as a configuration input from the user to extract all the features' definition, SPARQL endpoints, SHACL constraints, target classes, and sampling strategies. Features are defined in the form of independent and dependent variables, provided to the InterpretME pipeline to perform the prediction tasks. For the SHACL validation, InterpretME relies on Trav-SHACL [19] since it is capable of validating the shape schema against a SPARQL endpoint and scales better compared to other approaches with the same capability. InterpretME has provided a platform for data curation and sampling strategy. Model building of InterpretME can provide binary and multi-class classification with ensemble learning techniques. In model building, cross-validation and hyperparameters are obtained for a particular predictive model. Currently, InterpretME uses LIME [5] to create model interpretations. The Mapping Language (RML) [20] defines the process of integrating the traced metadata into the InterpretME KG; it is semantified using SDM-RDFizer [21] (version 4.5.5 [26]), an efficient RML-compliant engine for KG creation. The RDF data is uploaded into an instance of Virtuoso 7.20.3233 representing the InterpretME KG. The following *pipeline()* command executes the whole pipeline; including extracting data and metadata from the input KGs, validating SHACL constraints, preprocessing the data, running predictive models, semantifying the results, and populating the InterpretME KG:

```
from InterpretME import pipeline
results = pipeline(path_config='./example.json', lime_results='./LIME',
    server_url='endpoint of InterpretME KG', username='username to upload to InterpretME KG',
    password='password to upload to InterpretME KG')
```

For defining the InterpretME ontology, we extend the ML Schema [27]. ML Schema is considered a core vocabulary to deal with machine learning algorithms; it can be used to address different machine learning algorithms, implementation, model evaluation, and the input and output considered by the algorithms. The ontology represents relationships between machine learning algorithms and their executions. ML Schema also deals with interoperability issues and allows reproducibility to link machine learning results into linked data. We are reusing 12 concepts in the mappings from ML Schema and have others from InterpretME. Following the FAIR principles [28], the InterpretME ontology is publicly available on an instance of the collaborative ontology development VoCoL[20]; it enables the publication, management, and exploration of the ontology.

[19] https://pypi.org/project/InterpretME/
[20] http://ontology.tib.eu/InterpretME/

```
from InterpretME.federated_query_engine import configuration, federated
interpretme_endpoint = 'SPARQL endpoint of InterpretME KG'
input_endpoint = 'SPARQL endpoint of input KG'
config = configuration(interpretme_endpoint, input_endpoint)
query_answer = federated(input_query, config)
```

The data from the original KGs, as well as the InterpretME KG, can be queried using the federated query engine DeTrusty (version 0.6.1 [24]). DeTrusty is based on MULDER [29], i.e., it uses semantic source descriptions during decomposition and planning. Rohde [30] describes the vision of incorporating the SHACL validation result into SPARQL query answers by executing the validation during query processing. While we are not providing an engine fulfilling this vision, since the SHACL validation result is used in different parts, the user can query the original KGs with the InterpretME KG to achieve a similar outcome as described by Rohde [30]. InterpretME comprises the following resources: **i)** the InterpretME pipeline enhancing interpretability of machine learning models; **ii)** the InterpretME KG describing the main characteristics of trained machine learning models; and **iii)** the InterpretME ontology specifying the vocabulary for describing the main characteristics of trained predictive models.

InterpretME is utilized on top of the RDF KGs of the following projects: **1)** CLARIFY[21] to define machine learning models to predict biomarkers of a lung cancer patient based on demographic features (e.g., age and gender), smoking habits, and relatives with cancer; **2)** ImProVIT[22] to develop models to predict the impact of the immune system and demographic features into the response of Hepatitis B and Influenza vaccines; and **3)** P4-LUCAT[23] to implement predictive models to forecast the relapse after a surgery or the disease progression in advance stages.

6. Related Work

Artificial intelligence and machine learning have become global in many domains. The necessity of automated machine learning frameworks with assistance has gained popularity in research field, in every sector. De Bie et al. [3] discuss the challenges to achieve automated machine learning and highlight that existing tools contribute to mechanization and composition automatization, lacking support in assistance. InterpretME also aims at bridging this gap, and resorts to Semantic Web technologies to enhance users' assistance. Lundberg et al. [4] propose an interpretation framework called SHAP based on coalition game theory (Shapely values). SHAP provides feature contribution for each individual instance, global explanations, and feature importance. Ribeiro et al. [5] present LIME, an approach for local surrogate models, which are used to explain individual predictions of a pipeline. Local surrogate models are trained to approximate the predictions of models locally, instead of training a surrogate model globally. As a result, LIME generates human-friendly explanations for target entities. However, these explanations are not machine-readable and cannot be translated into the domain application. InterpretME overcomes these limitations, and provides fine-grained representations of local interpretations which are linked to the target entities in the domain application KGs.

Semantic Web technologies like ontologies are used to improve the accuracy of predictive models. Kulmanov et al. [31] study the role of ontologies in semantic similarity and machine learning, and present ontology embeddings as background knowledge. Further, ontologies can constrain the output of a machine learning model, i.e., making the model consistent with the axioms of the ontology. Min et al. [32] improve the performance of predictive models by using ontological adjustments, i.e., using the hierarchy of an ontology to move samples of rare classes into the next broader concept. On average, Min et al. [32] reduce the area under the receiver operating curve (AROC) by 9.0% in predicting the effectiveness of antidepressants in patients with rare conditions. Haug et al. [33] propose the combination of a large enterprise data warehouse with the medical knowledge from a disease-oriented ontology. This combination allows for automating the generation of computable diagnostic models, which aim at supporting researchers in generating and evaluating tools for real-time clinical diagnosis. Similarly, InterpretME resorts to Semantic Web technologies to enhance not only accuracy but interpretability as well.

[21] EU H2020 Funded project https://www.clarify2020.eu/
[22] German Funded project https://www.tib.eu/en/research-development/project-overview/project-summary/improvit
[23] EraMed project https://p4-lucat.eu/

7. Conclusions and Future Work

InterpretME empowers predictive models with metadata traced along a predictive modeling pipeline. The InterpretME KG reuses concepts defined in the ML Schema and resorts to a state-of-the-art SHACL engine for efficient constraint validation. Empirically, we have observed that InterpretME empowers the description of a predictive model's insights using factual statements and links to the application domain KGs. As a result, InterpretME broadens the portfolio of Semantic Web tools to enhance domain understating by bridging the gap between data meaning and predictive modeling [34]. The current version of InterpretME considers random forest, decision trees, and the LIME interpretable model. Nevertheless, we plan to integrate other models and tools. Furthermore, connecting InterpretME with automated machine learning systems and causal knowledge graphs [35] is part of our future work.

References

[1] M.A. Myszczynska, P.N. Ojamies, A.M.B. Lacoste, D. Neil, A. Saffari, R. Mead, G.M. Hautbergue, J.D. Holbrook and L. Ferraiuolo, Applications of machine learning to diagnosis and treatment of neurodegenerative diseases, *Nat Rev Neurol* **16** (2020), 440–456–.

[2] M.F. Pinto, H. Oliveira, S. Batista, L. Cruz, M. Pinto, I. Correia, P. Martins and C. Teixeira, Prediction of disease progression and outcomes in multiple sclerosis with machine learning, *Sci Rep* **10** (2020).

[3] T. De Bie, L. De Raedt, J. Hern_'andez-Orallo, H.H. Hoos, P. Smyth and C.K.I. Williams, Automating Data Science, *Communications of the ACM* **65**(3) (2022), 76–87.

[4] S.M. Lundberg and S.-I. Lee, A Unified Approach to Interpreting Model Predictions, in: *NIPS '17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, Curran Associates Inc., Red Hook, NY, USA, 2017.

[5] M.T. Ribeiro, S. Singh and C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York, NY, USA, 2016.

[6] P.I. Nakagawa, L.F. Pires, J.L.R. Moreira, L.O. Bonino da Silva Santos and F. Bukhsh, Semantic Description of Explainable Machine Learning Workflows for Improving Trust, *Applied Sciences* **11**(22) (2021). https://www.mdpi.com/2076-3417/11/22/10804.

[7] R. Prasad, L. Joseph and R.C. Deo, Modeling and Forecasting Renewable Energy Resources for Sustainable Power Generation: Basic Concepts and Predictive Model Results, in: *Translating the Paris Agreement into Action in the Pacific*, Springer, Cham, 2020.

[8] L. Kotthoff, C. Thornton, H.H. Hoos, F. Hutter and K. Leyton-Brown, Auto-WEKA 2.0: Automatic model selection and hyperparameter optimization in WEKA, *J. Mach. Learn. Res.* **18** (2017), 25:1–25:5. https://www.cs.ubc.ca/labs/algorithms/Projects/autoweka/papers/16-599.pdf.

[9] T. Berners-Lee, J. Hendler and O. Lassila, The Semantic Web: A New Form of Web Content That is Meaningful to Computers Will Unleash a Revolution of New Possibilities, 2001. http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21.

[10] O. Lassila and R.R. Swick, Resource Description Framework (RDF) Model and Syntax Specification, 1999. https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[11] M. Arenas, C. Gutierrez and J. Pérez, Foundations of RDF databases, in: *Reasoning Web. Semantic Technologies for Information Systems. Reasoning Web 2009. Lecture Notes in Computer Science*, Vol. 5689, Springer, Berlin, Heidelberg, 2008.

[12] A. Hogan, E. Blomqvist, M. Cochez, C. D'amato, G.D. Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier and et al., Knowledge Graphs, *ACM Computing Surveys* **54**(4) (2021). doi:10.1145/3447772.

[13] E. Prud'hommeaux and A. Seaborne, SPARQL Query Language for RDF, 2008. https://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[14] S. Harris and A. Seaborne, SPARQL 1.1 Query Language, 2013. https://www.w3.org/TR/2013/REC-sparql11-query-20130321/.

[15] M. Acosta, M. Vidal, T. Lampo, J. Castillo and E. Ruckhaus, ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints, in: *The Semantic Web – ISWC 2011*, 2011.

[16] H. Knublauch and D. Kontokostas, Shapes Constraint Language (SHACL), 2017. https://www.w3.org/TR/2017/REC-shacl-20170720/.

[17] J. Corman, J.L. Reutter and O. Savković, Semantics and Validation of Recursive SHACL, in: *The Semantic Web – ISWC 2018*, Springer, Cham, 2018.

[18] J. Corman, F. Florenzano, J.L. Reutter and O. Savković, Validating SHACL Constraints over a SPARQL Endpoint, in: *The Semantic Web – ISWC 2019*, Springer, Cham, 2019.

[19] M. Figuera, P.D. Rohde and M.-E. Vidal, Trav-SHACL: Efficiently Validating Networks of SHACL Constraints, in: *The Web Conference*, ACM, New York, NY, USA, 2021.

[20] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens and R. Van de Walle, RML: a generic language for integrated RDF mappings of heterogeneous data, in: *7th Workshop on Linked Data on the Web*, 2014.

[21] E. Iglesias, S. Jozashoori, D. Chaves-Fraga, D. Collarana and M.-E. Vidal, SDM-RDFizer: An RML Interpreter for the Efficient Creation of RDF Knowledge Graphs, in: *CIKM '20: Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, ACM, New York, NY, USA, 2020. doi:10.1145/3340531.3412881.

[22] N. Halliwell, F. Gandon and F. Lecue, User Scored Evaluation of Non-Unique Explanations for Relational Graph Convolutional Network Link Prediction on Knowledge Graphs, in: *K-CAP '21: Proceedings of the 11th Knowledge Capture Conference*, ACM, New York, NY, USA, 2021.

[23] M. Vidal, K.M. Endris, S. Jazashoori, A. Sakor and A. Rivas, Transforming Heterogeneous Data into Knowledge for Personalized Treatments - A Use Case, *Datenbank-Spektrum* **19**(2) (2019), 95–106.
[24] P.D. Rohde, DeTrusty v0.6.1, 2022. doi:10.5281/zenodo.6998001.
[25] M. Färber, F. Bartscherer, C. Menne and A. Rettinger, Linked Data Quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO, *Semantic Web Journal* (2017).
[26] E. Iglesias, S. Jozashoori and D. Chaves-Fraga, SDM-RDFizer v4.4.2, 2022.
[27] D. Esteves, A. Ławrynowicz, P. Panov, L. Soldatova, T. Soru and J. Vanschoren, ML Schema Core Specification, 2016. http://www.w3.org/2016/10/mls/.
[28] P. Groth and M. Dumontier, Introduction - FAIR data, systems and analysis, *Data Sci.* **3**(1) (2020), 1–2.
[29] K.M. Endris, M. Galkin, I. Lytra, M.N. Mami, M.-E. Vidal and S. Auer, Querying Interlinked Data by Bridging RDF Molecule Templates, in: *Transactions on Large-Scale Data- and Knowledge-Centered Systems XXXIX*, Vol. 11310, Springer, Berlin, Heidelberg, 2018.
[30] P.D. Rohde, SHACL Constraint Validation during SPARQL Query Processing, in: *Proceedings of the VLDB 2021 PhD Workshop*, CEUR-WS.org, Aachen, Germany, 2021. http://ceur-ws.org/Vol-2971/paper05.pdf.
[31] M. Kulmanov, F.Z. Smaili, X. Gao and R. Hoehndorf, Semantic similarity and machine learning with ontologies, *Briefings in Bioinformatics* **22**(4) (2020).
[32] H. Min, F. Alemi, C.A. Hane and V.S. Nori, Improving the Accuracy of Predictive Models for Outcomes of Antidepressants by Using an Ontological Adjustment Approach, *Applied Sciences* **12**(3) (2022).
[33] P. Haug, J. Holmen, X. Wu, K. Mynam, M. Ebert and J. Ferraro, Ontology-Based Tools to Expedite Predictive Model Construction, *AMIA Summits on Translational Science Proceedings* **2014** (2014). https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4419766/pdf/1859946.pdf.
[34] T. Gebru, J. Morgenstern, B. Vecchione, J.W. Vaughan, H.M. Wallach, H.D. III and K. Crawford, Datasheets for datasets, *Commun. ACM* **64**(12) (2021), 86–92.
[35] U. Jaimini and A.P. Sheth, CausalKG: Causal Knowledge Graph Explainability Using Interventional and Counterfactual Reasoning, *IEEE Internet Comput.* **26**(1) (2022).
[36] W. Ali, M. Saleem, B. Yao, A. Hogan and A.-C.N. Ngomo, A survey of RDF stores & SPARQL engines for querying knowledge graphs, *The VLDB Journal* **31**(3) (2022).
[37] M. Andreşel, J. Corman, M. Ortiz, J.L. Reutter, O. Savković and M. Šimkus, Stable Model Semantics for Recursive SHACL, in: *The Web Conference*, ACM, New York, NY, USA, 2020.
[38] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, DBpedia: A Nucleus for a Web of Open Data, in: *The Semantic Web – ISWC 2007*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.
[39] F. Belleau, M.-A. Nolin, N. Tourigny, P. Rigault and J. Morissette, Bio2RDF: Towards a Mashup to Build Bioinformatics Knowledge Systems, *Journal of Biomedical Informatics* **41**(5) (2008).
[40] S.R. Bader, J. Pullmann, C. Mader, S. Tramp, C. Quix, A.W. Müller, H. Akyürek, M. Böckmann, B.T. Imbusch, J. Lipp, S. Geisler and C. Lange, The International Data Spaces Information Model - An Ontology for Sovereign Exchange of Digital Content, in: *The Semantic Web – ISWC 2020*, Springer, Cham, 2020.
[41] J. Corman, F. Florenzano, J.L. Reutter and O. Savković, SHACL2SPARQL: Validating a SPARQL Endpoint against Recursive SHACL Constraints, in: *Proceedings of the ISWC 2019 Satellite Tracks*, CEUR-WS.org, Aachen, Germany, 2019. http://ceur-ws.org/Vol-2456/paper43.pdf.
[42] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer and C. Bizer, DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia, *Semantic Web* **6**(2) (2015). http://svn.aksw.org/papers/2013/SWJ_DBpedia/public.pdf.
[43] F. Maali, R. Cyganiak and V. Peristeras, A Publishing Pipeline for Linked Government Data, in: *The Semantic Web – ESWC 2012*, Springer, Berlin, Heidelberg, 2012.
[44] D.N. Nicholson and C.S. Greene, Constructing knowledge graphs and their biomedical applications, *Computational and Structural Biotechnology Journal* **18** (2020).
[45] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson and J. Taylor, Industry-Scale Knowledge Graphs: Lessons and Challenges, *Communications of the ACM* **62**(8) (2019).
[46] B. Otto, S. Lohmann, S. Auer, G. Brost, J. Cirullies, A. Eitel, T. Ernst, C. an Haas, M. Huber, C. Jung, J. J_"urjens, C. Lange, C. Mader, N. Menz, R. Nagel, H. Pettenpohl, J. Pullmann, C. Quix, J. Schon, D. Schulz, J. Sch_"utte, M. Spiekermann and S. Wenzel, Reference Architecture Model for the Industrial Data Space, Technical Report, Fraunhofer, 2017. https://www.fraunhofer.de/content/dam/zv/de/Forschungsfelder/industrial-data-space/Industrial-Data-Space_Reference-Architecture-Model-2017.pdf.
[47] H. Paulheim, Knowledge graph refinement: A survey of approaches and evaluation methods, *Semantic Web* **8**(3) (2017).
[48] J. Pérez, M. Arenas and C. Gutierrez, Semantics and Complexity of SPARQL, *ACM Transactions on Database Systems* **34**(3) (2009).
[49] M. Samwald, A. Jentzsch, C. Bouton, C. Kallesoe, E. Willighagen, J. Hajagos, M. Marshall, E. Prud'hommeaux, O. Hassanzadeh, E. Pichler and S. Stephens, Linked open drug data for pharmaceutical research and development, *Journal of Cheminformatics* **3**(1) (2011).
[50] M. Schmidt, M. Meier and G. Lausen, Foundations of SPARQL Query Optimization, in: *ICDT '10: Proceedings of the 13th International Conference on Database Theory*, ACM, New York, NY, USA, 2010.
[51] D. Vrandečić and M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Communications of the ACM* **57**(10) (2014).