Executable Knowledge Graphs for Machine Learning: A Bosch Case of Welding Monitoring

Zhuoxun Zheng[1,2], Baifan Zhou[3(✉)], Dongzhuoran Zhou[1,3], Xianda Zheng[4], Gong Cheng[5], Ahmet Soylu[2], and Evgeny Kharlamov[1,3]

[1] Bosch Center for Artificial Intelligence, Renningen, Germany
zhuoxun.zheng@de.bosch.com
[2] Department of Computer Science, Oslo Metropolitan University, Oslo, Norway
[3] SIRIUS Centre, University of Oslo, Oslo, Norway
baifanz@ifi.uio.no
[4] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
[5] School of Computer Science and Engineering, Southeast University, Nanjing, China

Abstract. Data analysis including ML are essential to extract insights from production data in modern industries. However, industrial ML is affected by: the low transparency of ML towards non-ML experts; poor and non-unified descriptions of ML practices for reviewing or comprehension; ad hoc fashion of ML solutions tailored to specific applications, which affects their re-usability. To address these challenges, we propose the concept and a system of executable Knowledge Graph (KG). It relies on semantic technologies to formally encode ML knowledge and solutions in KGs, which can be translated to executable scripts in a reusable and modularised fashion. In addition, the executable KGs also serve as common language between ML experts and non-ML experts, and facilitate their communication. We evaluated our system extensively with an impactful industrial use case at Bosch, including a user study, workshops and scalability evaluation. The evaluation demonstrates the system offers a user-friendly way for even non-ML experts to discuss, customise, and reuse ML methods.

Keywords: Knowledge graph · Machine learning · Data analytics · Industrial application · Welding monitoring

1 Introduction

Data analysis technologies play an important role in modern manufacturing industries. Examples include production monitoring, fault detection, root cause analysis, as well as robot positioning [1–3]. Among these technologies, machine learning attracts substantial yet increasing attention, for its strong modelling capability without the need of explicit programming [4] and the voluminous data that become available due to the introduction of internet of things into manufacturing [5]. Take the welding monitoring at Bosch as an example (Fig. 1a), which is an impactful automated manufacturing process that accounts for the global production of millions of cars every year. In welding monitoring, massive heterogeneous data from many sources need to be analysed for

Z. Zheng and B. Zhou—Contributed equally to this work as first authors.

U. Sattler et al. (Eds.): ISWC 2022, LNCS 13489, pp. 791–809, 2022.
https://doi.org/10.1007/978-3-031-19433-7_45

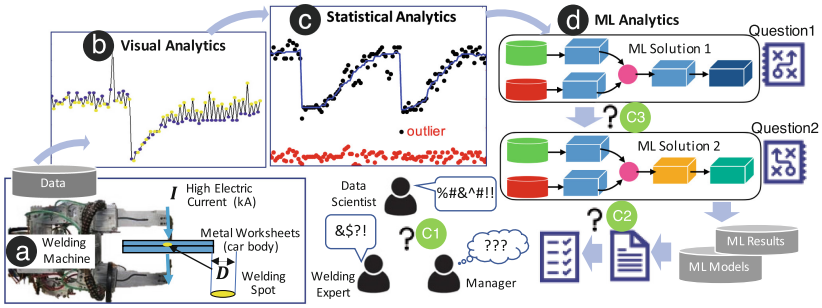

Fig. 1. Three of the activities of machine learning practice for (a) welding quality monitoring: visual analytics (b), statistical analytics (c), machine learning analytics (d). It faces three challenges: (C1) transparency of machine learning; (C2) standardised description; (C3) reusability.

various applications to solve different questions, e.g., to estimate or predict numerical quality indicators that are essential for ensuring high quality car production. Traditional quality monitoring approaches often require tearing the welded car bodies apart in random samples and measuring the diameter of the welded parts connection point, which is extremely costly. In contrast, data-driven methods like machine learning will help reduce the waste and contribute to more economical manufacturing industry [6]. Three important activities of machine learning practice at Bosch (Fig. 1b–d) include visual, statistical analytics (these two are often known as exploratory data analysis and seen as important preceding steps for machine learning analytics [7]), and machine learning analytics based on algorithms such as neural networks.

However, there exist still challenges of machine learning practice (Fig. 1) in modern industry, which often involve an interdisciplinary team of experts with distinct background. The transparency of machine learning (C1) to non-machine learning experts is usually challenging, since the latter often specialise in their domain knowledge and did not receive excessive training of machine learning that is often required to understand the sophisticated machine learning methods and interpret the machine learning results. The non-machine learning experts need to understand machine learning and trust that machine learning applied in manufacturing robots operating with high electricity can ensure product quality and personnel safety [8]. In addition, in traditional machine learning projects, the machine learning procedures, methods, scripts, and decisions are described in the technical language of machine learning, which is highly dependent on the person who writes the document. Machine learning knowledge and solutions are hardly described or documented in a standardised way (C2), causing difficulties for later review and retrospective comprehension of the projects in big companies like Bosch, which have strict regulations in reporting the details for later audit and analysis. Moreover, ML solutions are often developed in an ad hoc fashion and tailored to specific applications, which complicates its reusability (C3) for new data or questions.

To address these challenges, we propose to combine semantic technologies and machine learning, to encode machine learning solutions in knowledge graphs in a smart way, so that the knowledge graphs help in describing machine learning knowledge and solutions in a standardised and transparent way via GUI-based system and knowledge graphs visualisation. We name our approach as executable knowledge graphs, because our knowledge graphs can be translated to modularised executable machine learning

scripts that can be modified and reused for new questions. In particular, our contributions are as follows:

- We introduce the concept and a basic framework of executable knowledge graph, that represents the machine learning solutions for solving machine learning questions. The executable knowledge graphs can be translated into modularised executable scripts and are highly reusable.
- We present a use case of Bosch welding monitoring with machine learning, and derive the requirements for executable knowledge graph system.
- We propose a system of executable knowledge graphs. The system has five layers, including the layer of semantic artefacts that serve as the schemata of the knowledge graphs, the of layer semantic modules which construct the knowledge graphs in a semi-automatic fashion based on GUI, knowledge graph data layer that stores the knowledge graphs, application layer that covers visual analytics, statistical analytics and ML analytics, and the (non-knowledge graph) data layer.
- We evaluate our system of executable knowledge graphs extensively: in an user study that verifies whether our system really help in improve the transparency, reusability, etc.; and a system evaluation that verifies the scalability of our approach.

The paper is organised as follows: Sect. 2 explains the use case of Bosch Welding Monitoring, Sect. 4 introduces our framework for executable knowledge graphs, Sect. 5 describes the executable knowledge graph system, Sect. 6 demonstrates the evaluation, Sect. 3 discusses some related work, Sect. 7 presents the conclusion.

2 Use Case: Bosch Welding Monitoring

Resistance Spot Welding and Quality Monitoring. Resistance Spot Welding is a type of fully automated and impactful manufacturing process widely applied in automotive industry [9], accounting for the production of millions of cars globally every year. We illustrate RSW with Fig. 1a, in which the two electrode caps of the welding gun press two or three metal worksheets between the electrodes with force, and pass a high electric current flow through the worksheets. A huge amount of heat is generated due to resistance. The material in a small area between the electrodes will melt, and form a welding nugget connecting the worksheets, known as the welding spot. The quality of welding operations is typically quantified by quality indicators like spot diameters, as prescribed in international and German standards [10,11]. To obtain the spot diameters precisely, the common practice is to tear the welded car body apart and measure them [11], which destroys the welded cars and is extremely expensive. Now Bosch is developing machine learning-based methods to reduce the need of destroyed car bodies and thus reducing waste, aiming at more economical and sustainable manufacturing [12].

Machine Learning Development: Interdisciplinary, Documented, Reusable. Machine learning projects at Bosch involve experts of distinct backgrounds [13,14]: e.g., welding experts know the domain knowledge of the process and the questions that need to be solved, measurement experts know the data particularities like sensor setting, data scientists (typically machine learning experts) know the machine learning technology to solve the question, managers need to prioritise the activities according to available resource the strategic interest of the companies. They work together

for machine learning development yet speak different language. Their communication requires the transparency of machine learning practice (knowledge, solution, options, etc.), so that the non-machine learning experts can understand machine learning and trust that machine learning applied in heavy robots that operate with high electricity can ensure product quality and personnel safety [15,16]. In addition, Bosch has strict regulations on documenting and reporting machine learning projects for later review or audit. Thus, the process of machine learning development, and the developed machine learning solutions, knowledge, and insights need to be documented properly by the experts. Moreover, Bosch has many data sources, similar manufacturing processes. Alone the resistance spot welding has data sources of at least 4 locations and 3 customers, while Bosch has other similar welding processes like hot-staking, ultrasonic welding, etc. Thus the reusability of machine learning solutions is highly desired so that they can be transferred to similar data or machine learning questions.

Visual Analytics, Statistical Analytics, and ML Analytics. Here we discuss three important ML activities at Bosch. We refer *visual analytics* to the visualisation of data in various plots [17], e.g., line plot, scatter plot, bar plot, heat map. It helps the experts to gain an intuitive understanding of the data, detect potential interest data subset, and visualise machine learning results. We discuss *statistical analytics* as using a broad range of statistical methods for generating insights from data [18], such as calculation of mean, median, standard deviation, sliding window filter, outlier detection, etc. *machine learning analytics* is understood [19] as relying on two schools of machine learning approaches, feature engineering and deep learning, to train machine learning models and make machine learning inference, e.g., classification, regression.

Requirements for Executable Knowledge Graph System. We derive the requirements for the proposed system and the executable knowledge graphs in the system as follows:

- *R1 Transparency.* Our system should provide standardised description of machine learning knowledge and solutions and make them easier to understand for the non-machine learning experts. It is essential for big manufacturing companies like Bosch since machine learning can only be trusted when they are understood for manufacturing industries with high standards of quality and safety regulations.
- *R2 Usability.* The system should be easy to use, in three aspects [20]: effectiveness – users can use the system correctly; efficiency – users can use the system fast; satisfaction – users are satisifed with the system.
- *R3 Executability.* The executable knowledge graphs in the system should be able to be translated to scripts that are executable, namely not having bugs.
- *R4 Coverage.* The executable knowledge graphs should be able to represent most solutions of visual analytics, statistical analytics, and ML analytics.
- *R5 Reusability and Modularity.* The system and the executable knowledge graphs should support users to reuse developed solutions for similar data or questions by e.g., slightly modifying existing solutions or reusing modules of the solutions.
- *R6 Scalability.* The scripts translated from the executable knowledge graphs should not consume excessive time and thus be scalable for large-scale deployment.

3 Related Work

In recent years, researchers have begun to use graph structures and knowledge graphs to represent codes and the relationships between them in programming languages. They both treat code artefacts, which containing classes, methods, variables, as nodes, use their predefined relationships as edges [21,22], and use them to complete downstream tasks like defect prediction [23] and query-based analytics [24]. However, these approaches only consider the connections and semantic relationships between codes and insufficiently discuss the more complex graph form, knowledge graphs, which provide more expressivity, e.g., treat the information flow of data between codes as edges and define semantic constraints. Many knowledge graphs were discussed in the literature, e.g., Freebase [25], DBpedia [26]. Specialised knowledge graphs have been used in areas, e.g., e-commerce [27], procurement [28,29], and healthcare [30]. KGs are gaining popularity in the industries [31–33], but few works were dedicated into describing machine learning practice in industries.

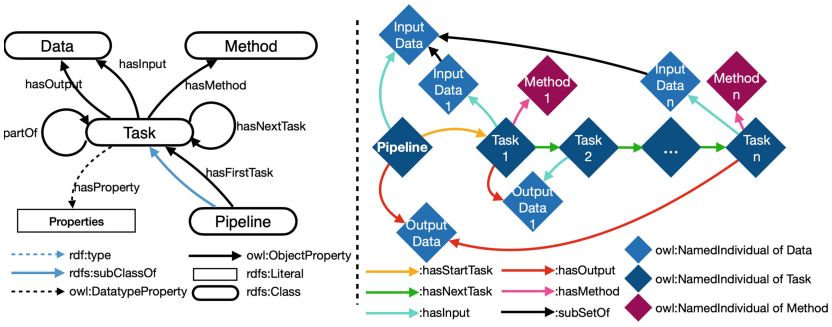

Fig. 2. Framework of executable knowledge graph

4 Executable Knowledge Graphs Framework

In this section we introduce the framework for executable knowledge graph that represents the ML solutions (pipelines) for solving machine learning questions. The framework supports the executable knowledge graph to be translated to executable scripts and modularised, thus the system based on executable knowledge graph can fulfil the requirements of *Executability* and *Reusability and Modularity*.

We first define data, methods and tasks in this framework. *Data* $\mathcal{D}$ is a set of items of information, it can be in forms such as numerals, diagrams or strings organised in different structures such as tables. A *Method* $\mathcal{F}$ is a function in form of language-dependent script. A method takes some data which fulfils certain constraints as input and can output specific data. Formally, $\mathcal{D}_{out} = \mathcal{F}(\mathcal{D}_{in})$. A *Task* $\mathcal{T}$ is the process of invoking a method by feeding it with some data that meets certain constraints, and obtaining some other data. Formally, $\mathcal{T}\langle\mathcal{D}_{in},\ \mathcal{F}\rangle = \mathcal{F}(\mathcal{D}_{in}) = \mathcal{D}_{out}$.

Some tasks have methods which are unified, while other more complex tasks can not solved by invoking a single integrated method while can be unfolded into a sequence of tasks where each task is a part of the complex one. We refer the complex tasks as

data pipelines $\mathcal{T}_p$. Formally, a pipeline $\mathcal{T}_p$ with input data $\mathcal{D}_{in}$ to get $\mathcal{D}_{out}$, expressed as $\mathcal{T}_p\langle\mathcal{D}_{in},\ \mathcal{F}\rangle = \mathcal{D}_{out}$ can be unfolded in the sequence $\{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_n\}$. Formally:

$$\mathcal{T}_1\langle\mathcal{D}_{in_1},\ \mathcal{F}_1\rangle = \mathcal{D}_{out_1}, \mathcal{D}_{in_1} \in \mathcal{D}_{in}, ...\mathcal{T}_n\langle\mathcal{D}_{in_n},\ \mathcal{F}_n\rangle = \mathcal{D}_{out_n},$$
$$\mathcal{D}_{in_n} \in \dot{\bigcup}_{i\in\{1,2...n-1\}} \mathcal{D}_{out_i} \cup \mathcal{D}_{in}, \longrightarrow \mathcal{D}_{out} \in \dot{\bigcup}_{i\in\{1,2,...,n\}} \mathcal{D}_{out_i}.$$

Based on the above definitions, we determine the framework for the executable knowledge graphs as the left part of Fig. 2, such executable knowledge graph should take the form as the right part of Fig. 2. Here we split the properties from the data $\mathcal{D}$, which strictly speaking also belong to $\mathcal{D}$, but correspond to the properties rather than objects of a $Task$. Except those $Tasks$ with their $Methods$ already been integrated in script, all other $Tasks$ can be modularised in a $Pipeline$ and be unfolded into a sequence of $Tasks$. The objectProperty *:hasFirstTask* connects the $Pipeline$ with the first task in its unfolded sequence, while *:hasNextTask* connects the task in the sequence with its following task. In this framework, as long as the $Data$ and $Properties$ of every $Task$ fulfil the constraints of the $Method$ in the $Task$, the $Task$ is executable. If every $Task$ in a $Pipeline$ is executable, the $Pipeline$ is executable. In addition, as a $Task$, the $Pipeline$ can also be a part of another $Task$, which represents the modularity of the executable knowledge graph.

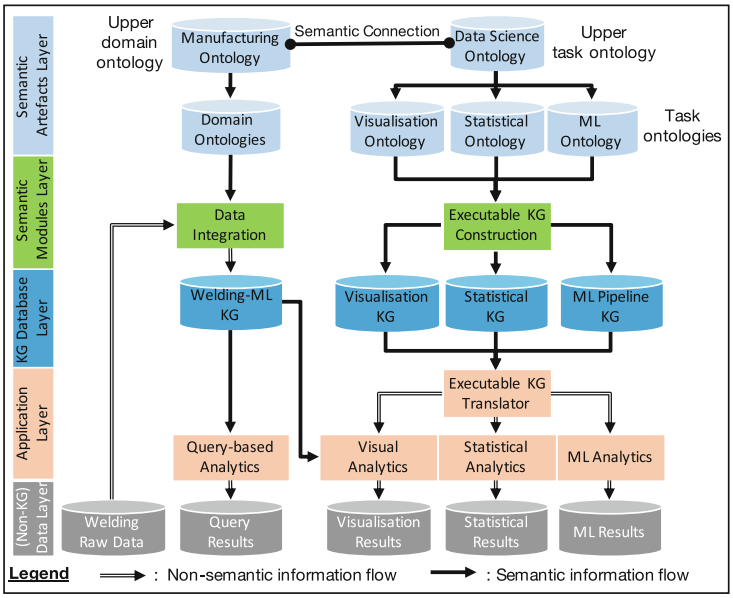

Fig. 3. An architectural overview of our knowledge graph solution

5 Our Executable Knowledge Graph Based ML System

5.1 Architectural Overview

We now give an architectural overview of our system. Our system consists of five layers (Fig. 3). These layers are (from bottom to top): (non-knowledge graph) data layer,

application layer, knowledge graph database layer, semantic modules layer, and semantic artefacts layer. From the bottom left, we start with the welding raw data collected from production lines. These data are transformed by the Data Integration module (with the help of domain ontologies) to Welding-machine learning knowledge graphs, which is a type of welding data knowledge graph with some machine learning annotation [34]. These knowledge graphs are used by four types of analytics applications in the application layer.

The domain ontologies include various welding ontologies, e.g., resistance spot welding ontology, hot-staking ontology. These ontologies are created based on the upper domain ontology [2], the manufacturing ontology. The manufacturing ontology is semantically connected with an upper task ontology, the data science ontology (O^{ds}), in a way that the datatype properties in the former one are annotated by some classes in the latter one. A series of task ontologies (Fig. 4), including the visualisation ontology (O^{visu}), the statistical ontology (O^{stats}), and ML ontology (O^{ml}), are created based on the O^{ds}. These task ontologies serve as the schemata for the Executable knowledge graph Construction module, which encodes the executable data pipelines in the executable knowledge graphs, including the visualisation knowledge graph, statistical knowledge graph, and ML pipeline knowledge graph. These executable knowledge graphs then can be translated by the Executable knowledge graph Translator module to executable scripts for three analytics applications: Visual Analytics, Statistic Analytics, and ML Analytics, which generate the corresponding results.

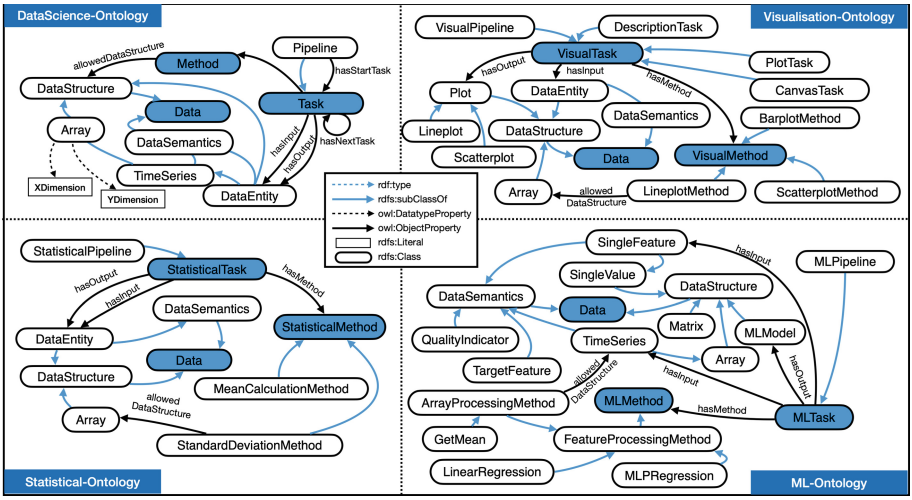

Fig. 4. Task Ontologies for the executable KG

5.2 Semantic Artefacts

We now introduce our ontologies some of which are in Fig. 4.

Upper Domain Ontology and Domain Ontologies. The upper domain ontology, the manufacturing ontology, consists of 1170 axioms containing 95 classes, 70 object properties and 122 datatype properties [9]. It is an OWL 2 ontology modelling the general

knowledge of discrete manufacturing process, which refers to a broad range of manufacturing processes whose products are easily identifiable and countable, e.g., welding spots, and differ greatly from continuous process manufacturing where the products are undifferentiated, e.g. petrol. The ontology has the manufacturing *operation* as the most important class, and has other classes to describe other concepts related to the *operation*, e.g., the operations process *resource*, produce *product*s and are performed by *machines*. The domain ontologies describe several manufacturing domains at Bosch, e.g., resistance spot welding ontology, hot-staking ontology [35]. These ontologies are created by domain experts in such a way that all classes (properties) in the domain ontologies are sub-classes (sub-properties) of that in upper domain ontologies.

Data Science Ontology. The upper task ontology is the data science ontology O^{ds} (OWL 2) created by Bosch data scientists, which formalise the general knowledge of data science activities. It contains three most important classes (Fig. 4a): *Data* that is the class of all data concepts (the existential being in data science), *Method* is the class of all algorithms and functions (the way that data move), whose allowed input, output and parameters are defined, and *Task* is the class of the scripts that invoke the functions, which has an important sub-class, *Pipeline* that consists of a series of ordered tasks (the way that the data movement is organised). The *Data* can have *DataSemantics* that describe the meaning of data and *DataStructure* that prescribes the format (in the form of datatype properties) of the data, e.g., a *TimeSeries* has the format *Array*. A *DataEntity* is the class for a concrete dataset or a feature. In addition, there exist some constraints, e.g., an *Array* must have *XDimension* greater than 1 and *YDimension* smaller than 2.

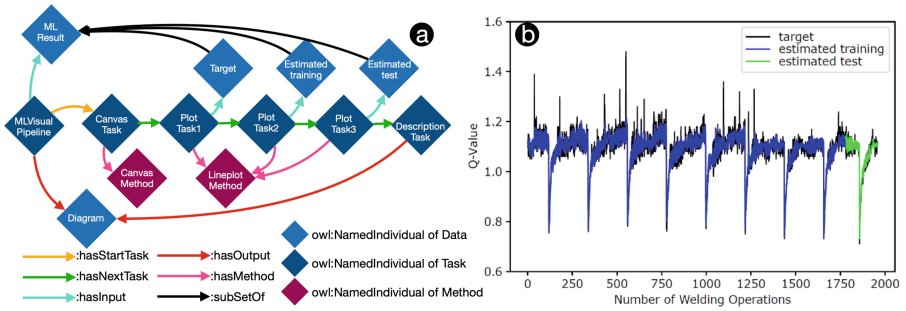

Fig. 5. The executable KG for visualisation (a) and its results (b). It needs to be created in *VisuTask1* in the user study (Sect. 6), which aims to visualise the ML learning results by plotting the q-value arrays of target (ground-truth), and estimated q-value training and test.

Visualisation, Statistical, and Machine Learning Ontologies are the three task ontologies created based on O^{ds} in such as way that all classes in the task ontologies are sub-classes of that in O^{ds}, and all properties in the task ontologies are sub-properties of that in O^{ds}. The visualisation ontology O^{visu} describes most common visual analytics methods, such as *Lineplot*, *Scatterplot*, etc., and the *DataStructure* that is allowed for the methods, e.g., *Lineplot* allows *Array*s as input. In addition, O^{visu} also prescribes the construction of a visualisation *Pipeline*, which should has the *CanvasTask* as the first task, several *PlotTask* after that, and has *DescriptionTask* as the last task. Similarly,

the statistic ontology O^{stats} (and the machine learning ontology O^{ml}, resp.) describes the most common statistical analytics (machine learning analytics, resp.) methods, their allowed *DataStructure*, and the organisation of the tasks in *Pipeline*. In addition, some rules determine explicitly the constraints between the input data of *Task*, e.g., the input *DataEntity*s of the *Concatenation* task should have the same concatenation dimension.

Executable Knowledge Graphs. Based on the task ontologies, executable knowledge graphs are constructed for visual, statistical and machine learning analytics, including *visualisation*, *statistical* and *Machine Learning Knowledge Graph*s. All such knowledge graphs are in the form of pipelines, which consist of a series of tasks. We illustrate this with example knowledge graphs in Fig. 5, Fig. 6, and Fig. 7.

5.3 Executable KG Construction

In our system the executable KGs are constructed semi-automatically in three ways: KG creation, KG modification and KG integration.

Executable Knowledge Graph Creation via GUI is common for relative easy ones such as visualisation and statistical knowledge graphs. For ML pipeline knowledge graphs, advanced users can also create knowledge graphs from the scratch, but most users would prefer to modify or integrate existing ML pipeline knowledge graphs.

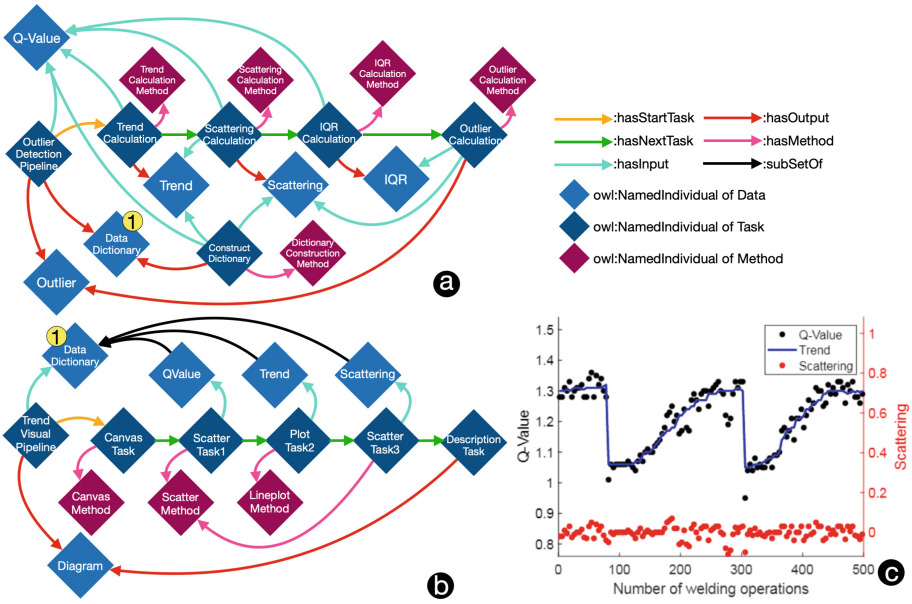

Fig. 6. The knowledge graph for (a) computing the outliers of the *Q-Value* array and (b) visualising the Q-Value array and its the trend, scattering statistical analysis. (c) The visualisation diagram of (b). (a) and (b) are used in *VisuT2* and *StatsT3* in Table 2, respectively.

Visualisation Knowledge Graph (Fig. 5a). Once the user chooses to create a visualisation knowledge graph, by default the GUI will show a *owl:NamedIndividual* with the label *VisualPipeline* (the user can change it), the *CanvasTask* and *DescriptionTask*. Next the users will need to select the input data (commonly a csv file), and several *PlotTask*s from available tasks based on O^{visu}, between the *CanvasTask* and the *DescriptionTask*. For each *PlotTask*, the input data, the method and some parameters will mandatorily be given based on O^{visu}. After that, the visualisation knowledge graph creation is finished.

Statistical Knowledge Graph (Fig. 6a). Once the user chooses to create a statistical knowledge graph, by default the GUI will show a *owl:NamedIndividual* with the label *StatisticalPipeline*, and the *DictionaryTask*, which wraps the output into a dictionary (user can opt to delete it). Next the users will need to select the input data (commonly a csv file), and *StatisticalTask*s from available tasks based on O^{stats}. For each *StatisticalTask*, the users need to select the input data, the method and some mandatory parameters, based on O^{stats}. After the user configuration, constraints verification and resolution, the statistical knowledge graph creation is finished.

Executable Knowledge Graph Modification. Another way to create an executable knowledge graph is to modify existing knowledge graphs, which is common for all three types of knowledge graphs, especially ML pipeline knowledge graphs.

Statistical Knowledge Graph (Fig. 7a–b). Once the user chooses to modify an existing knowledge graph, first the user needs to load a knowledge graph from our knowledge graph database. Now we load the statistical knowledge graph in Fig. 7a, which calculates *mean*, *standard deviation* (std.), *minimum* (min.), and *maximum* (max.) from an array. We want to modify this knowledge graph to another knowledge graph that can do *z-score normalisation*, which subtracts the mean from the array and then divides by std., $(arr - mean)/std.$. To achieve this, the user only need to delete the two statistical tasks, namely *MinimumCalculation* (Fig. 7 3) and *MaximumCalculation* (Fig. 7 4), then add another task *NormalisationCalculation* (Fig. 7 6) (which has *NormalisationMethod*, and select its input as the *MeanValue* and *StandardDeviation*. After that, the system will suggest *NormalisedData* as the output of the task *NormalisationCalculation*. The user then select *NormalisedData* as the final output of the pipeline and change the label of the pipeline. Knowledge graph modification is done.

Machine Learning Knowledge Graph. (Fig. 7c) takes *TimeSeries* and *SingleFeatures* as input data, and does *LRRegression* to predict the *Q-Value*. The users can simply change the input data, output data, and method of the pipeline, by changing the named individuals, e.g., the users can delete *TimeSeries* if they do not have the sensor curves in their data, because the sensor curves are costly to collect. The users can also change the machine learning method (from *LRRegression* to *MLP*), the output data (from *Q-Value* to *spot diameter*) and some hyper-parameters (MLT2 in Table 2).

Executable Knowledge Graph Integration. (Fig. 6), where a statistical pipeline that does outlier detection can be integrated with a visualisation pipeline to visualise the detection results. To do so, the users only need to select one output of Fig. 6a, the

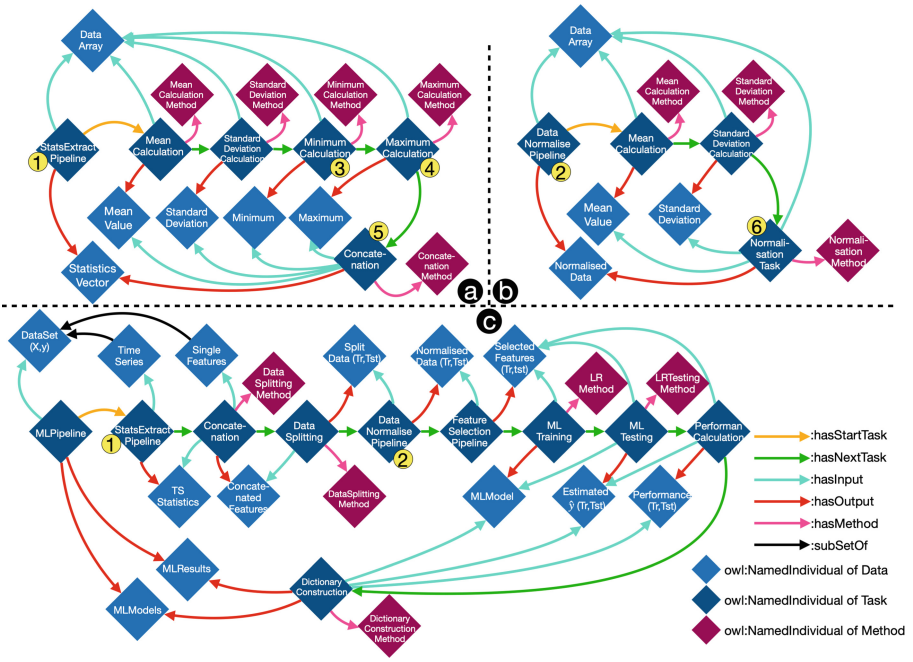

Fig. 7. (a) *StatsExtractPipeline* that extracts four statistics for a data array: mean, std., min., max. (b) *DataNormalisePipeline* that performs z-score normalisation for a data array. (c) *MLPipeline* that takes time series and single features as inputs and relies on linear regression (LRMethod). (a)(b)(c) are used in *StatsT1*, *StatsT2*, *MLT1* in Table 2, respectively.

DataDictionary, *Trend*, and *Scattering*) as the inputs of the *TrendVisualPipeline* in Fig. 6b. Another example is Fig. 7c, which is the result of reusing/integrating the *StatsExtractPipeine* (Fig. 7 1) in Fig. 7a and *DataNormalisationPipeline* in Fig. 7b.

5.4 Executable Knowledge Graph Translation

The translation of executable knowledge graphs is language-dependent. Here we use Python as the language for discussion. Each individual of *Method* is a Python function script, whose mandatory inputs/outputs and parameters are clearly defined. Each executable knowledge graph is in the form of a *Pipeline*, which consists of a series of *Tasks* of sequential or parallel structures connected with *hasNextTask*. Thus, the translation of an executable knowledge graph invokes the Python function scripts with the inputs/outputs and parameters given by *DataEntity* and datatype properties of knowledge graphs, according to the order defined by *hasNextTask*.

6 Evaluation

6.1 User Study: Transparency and Usability

Design of the User Study. We invited 28 experts in backgrounds of machine learning experts, welding experts, sensor engineers, etc. to attend the user study. For the user study, we first give a short *introduction* of our system, then the participants will perform a series of *tasks* related to visual, statistical and machine learning analytics, and finally they will answer *questionnaires* to record their subjective evaluation. For the tasks, to avoid user bias, we divided the participants into two groups, who will follow the schedule in Table 1. To contrast the situation of doing analytic tasks without and with our system, we designed the workflow as follows: Each group, including 2 machine learning experts and 12 non-machine learning experts, will first perform an analytic task without our system (T1), e.g. VisuT1 for Group A, and answer several single selection questions (SSQ) to test the understanding of the non-machine learning experts about the task (T2). We have designed 5–7 SSQs for each task, and there are 18 SSQs in total (Table 3). Then, they will do a similar task with our system (T3) and answer the SSQs (T4). Finally, they will revisit the previous task (T1) with our system. The same process repeat for the StatsT (T6–T10) and MLT (T12–T16). In addition, we design T11, T17 and T18 to test whether our system can realise the modularised reuse of executable knowledge graphs.

Tasks and Metrics. We list the tasks, their content and their knowledge graph visualisation in Table 2. For each task, the machine learning experts will explain the non-machine learning experts the tasks. In the case of "without our system" (T1, 6, 12), the experts communicate with technical language, and the non-machine learning experts will need to perform the tasks. Due to time limit, it is infeasible to do coding during the user study. The non-machine learning experts will answer whether they can finish the tasks with their programming an machine learning knowledge, and estimate the needed time for that. Thus, we will have two metrics: *complete percentage* and *time*. In addition, we compare the answers of SSQs with the correct answers and record the *correctness* (T2, 7, 13). In case of "with our system" (T3, 5, 8, 10, 11, 14, 16–18), the experts communicate using our system and the non-machine learning experts will need to perform the tasks. They do so by creating, modifying or merging knowledge graphs via a GUI. We record their actions and needed *time* for each task, and compare their action with a ground truth (we designed the task and GUI to make such comparison possible) to measure the *correctness*. Some users cannot finish the task, and thus we also recorded the *complete percentage*. In addition, we compare the answers of SSQs with the correct answers and record the *correctness* (T4, 9, 15).

Results and Discussion. We first look at results of using our system (Fig. 8a–c). It can be seen that most users have a high *complete percentage* (above 90%) using the system. When they complete the tasks, their correctness is also very high, about 80% (effectiveness, R2), even for the complex tasks T17 and T18. In addition, they usually do not need much time for each task, in average only 227.3 s, about 4 min (user efficiency, R2). Then we compare the correctness of SSQs without and with our system (Fig. 8d), which also show that the non-experts can gain better understanding of the three machine learning

Table 1. Workflow of the tasks in the user study.

#	Group A	Group B	Method
T1	VisuT1	VisuT2	Without our system
T2	VisuT1 SSQ	VisuT2 SSQ	–
T3	VisuT2	VisuT1	With our system: create KG
T4	VisuT2 SSQ	VisuT1 SSQ	–
T5	VisuT1	VisuT2	With our system: modify KG
T6	StatsT1	StatsT2	Without our system
T7	StatsT1 SSQ	StatsT2 SSQ	–
T8	StatsT2	StatsT1	With our system: create KG
T9	StatsT2 SSQ	StatsT1 SSQ	–
T10	StatsT1	StatsT2	With our system: modify KG
T11	StatsT3	StatsT3	With our system: modify KG
T12	MLT1	MLT2	Without our system
T13	MLT1 SSQ	MLT2 SSQ	–
T14	MLT2	MLT1	With our system: modify KG
T15	MLT2 SSQ	MLT1 SSQ	–
T16	MLT1	MLT2	With our system: modify KG
T17	ComplexTask1	ComplexTask1	With our system: merge KG
T18	ComplexTask2	ComplexTask2	With our system: merge KG

Table 2. Tasks and their content

Tasks	Content	KG
VisuT1	Visualise machine learning results with three line plots: target, estimated training, estimated test	Fig. 5
VisuT2	Visualise a quality indicator, its trend and scattering with scatter plots and line plots	Fig. 6b
StatsT1	Extract four statistics from a sequence: mean, std., min. and max	Fig. 7a
StatsT2	Z-score normalise a vector by substracting the mean and dividing by the standard deviation	Fig. 7b
StatsT3	Compute the trend, scattering and outliers of a sequence with median filter, etc.	Fig. 6a
MLT1	Reuse a ML pipeline for q-value estimation with linear regression	Fig. 7c
MLT2	Reuse a ML pipeline for diameter estimation with multilayer perceptron	Fig. 7c
ComplexT1	Visualise the results of StatsT1: merging/reusing the pipelines of StatsT1 and VisuT2	Fig. 7c
ComplexT2	Visualise the results of MLT1: merging/reusing the pipelines of StatsT1, StatsT2, MLT1 and VisuT1	Fig. 7c

Table 3. Examples of single selection questions (SSQ) for machine learning tasks.

Questions (Q) and Answers (A)
Q1: What are the input data we use for machine learning training? I: single features, II: sensor curves, III: quality indicator
A1: (A) I + II + III (B) I + II (C) II (D) II + III
Q2: What is the output feature we try to estimate?
A2: (A) Diameter (B) Q-value (C) Current mean (D) Process stability factor
Q3: What features will be the input of StatsExtractPipeline? I: single features, II: sensor curves, III: quality indicator
A3: (A) I + II + III (B) I + II (C) II (D) II + III

Table 4. Questionnaires (partial) and scores for subjective evaluation. The scores range from 1 (disagree), 2 (fairly disagree), 3 (neutral), 4 (fairly agree), to 5 (agree). The column *Score* is aggregated by reversing the scores of negative questions (such as Q2, 4, 6, 8, 9) and then computing the average (avg.) and standard deviation (std.) (`avg.±std.`)

#	Questions	Dimensions	Score
Q1	(For ML experts) I am confident to help non-expert to develop ML approaches based on the system	R1 Transparency	4.28 ± 0.47
	(For non-ML experts) I found it's easy to get basic understanding for ML approaches based on the system		
Q2	I felt the system hampers the communication on ML approaches		
Q3	I felt very confident using the system	R2 Usability	4.73 ± 0.39
Q4	I thought there was too much inconsistency in this system		
Q5	(For ML experts) I have confidence in the system to perform ML tasks	R3 Executability	4.60 ± 0.72
	(For non-ML experts) I am happy about the executability of this system		
Q6	I need the support of technical persons to be able to use this system		
Q7	I think the system can in general cover my need	R4 Coverage	4.24 ± 0.83
Q8	(For ML experts) I found the system didn't cover some basic ML functions that are commonly used in industry		
	(For non-ML experts) I think the system has very limited application in production		
Q9	(For ML experts) I think the ML pipelines developed in the system can only be reused in a limited range of applications	R5 Reusability	4.87 ± 0.36
	(For non-ML experts) I don't think I would try to reuse a developed pipeline when facing a new task		
Q10	I am happy that the system reduce time for reusing developed pipelines.		

activities (transparency, R1), as their SSQ correctness systematically increases when they use our system for communication. The comparison of time and task correctness without (w/o) and with (w) our system (Fig. 8e–f) shows that when users doing tasks with our systems they can save substantial time and increase the complete percentage (user efficiency, R2), and make analytics tasks that cannot be done by the non-machine learning experts now doable (usability, R2). The users also save much time when they reuse and modify (contrasting bars with m and w) existing knowledge graphs to solve the tasks.

It can be seen from the scores of questionnaires (Table 4) that the users indeed think our system improves the transparency (R1), and has good usability (R2), as these scores are all above 4. In addition, the users also satisfied with the coverage of the tasks (R4), and the reusability and modularity of the analytics pipelines (R5) brought by our system. The later two will be further discussed in the next section.

6.2 Evaluation of Executability, Coverage, and Reusability

Executability (R3). Beside the 9 executable knowledge graphs in the user study, we also programmatically generate 1372 executable knowledge graphs covering most of the tasks (Table 5) encountered by automated modification of a set of executable knowledge graph templates. As expected, all these knowledge graphs can be translated into scripts that are executable (Fig. 9). Thus, the executable knowledge graphs that follow the our framework in Sect. 4 are also evaluated as executable.

Coverage (R4). We organised extensive workshops with the machine learning and non-machine learning experts. After discussion, we categorised most tasks of visual, statistical and machine learning analytics encountered in our project in groups (see Table 5), and give the coverage percentage according to our empirical cases. Observe, for the 3 groups of visual analytics, and 5 groups of statistical, most of them can be covered (above 80%). While for the feature engineering school of machine learning analytics, we cover 80%. The feature learning school is currently not our focus of the work.

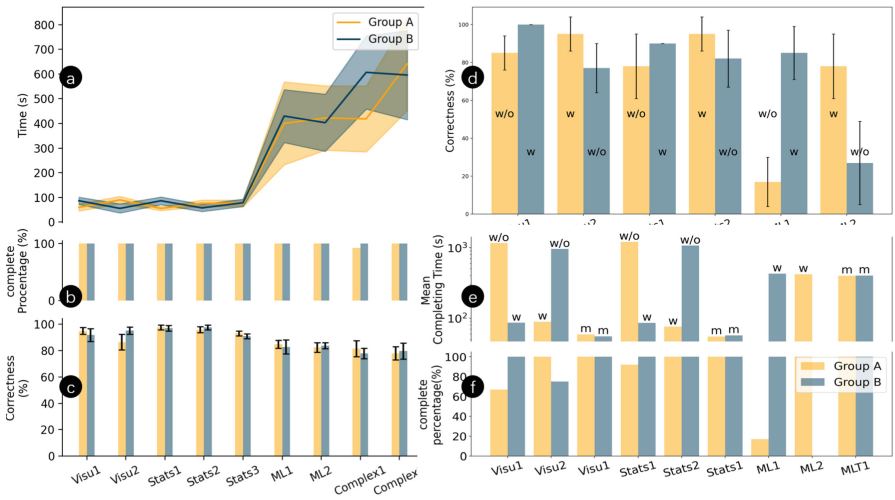

Fig. 8. The user study results in time (a), complete percentage (b) and correctness (c); comparing the SSQ correctness between without and with our system (d); comparing users doing tasks without (w/o) and with (w) our system or only modify knowledge graph (m) in time (e) and complete percentage (f)

Reusability (R5). In user study, multiple tasks demonstrate the high reusability and modularity supported by our system. In T5, 10, 14, 16 (Fig. 5a, Fig. 6a, Fig. 7a and c), users modify existing knowledge graphs by adding named individuals and changing task parameters, and thus reuse the knowledge graphs for new tasks, which is a strong evidence of reusability. In T17 and 18 (Fig. 7c), they simply merge existing knowledge graphs and form more complicated ones, this demonstrates the modularity (and thus also reusability).

Table 5. Tasks categories and coverage

Category	Sub-category	Coverage
Visual	Line plot, scatter plot, bar chart	100%
	Pie chart	85%
	Heatmap	85%
Statistic	Statistics calculation	95%
	Basic mathematical operation	100%
	Sliding window filtering	90%
	Sub-sampling	80%
	Interpolation & extrapolation	80%
ML	Feature engineering	80%
	Feature learning	0%

6.3 System Evaluation of Scalability

Apart from the aforementioned requirements, we evaluate the scalability (R6) of our system for large deployment. We tested the running time of different types of analytics pipelines for welding quality monitoring (Fig. 9). The tasks include 1372 programmatically generated analytics pipelines and thus 1372 executable knowledge graphs, including 242 knowledge graphs for visual analytics, 253 knowledge graphs for statistical analytics, 291 merged knowledge graphs that combine visual and statistical analytics, 272 merge knowledge graphs the combine statistical and ML analytics, and 314 merge knowledge graphs that combine three of them. We conducted experiments on an MacBook Pro with Apple M1 Processor, 16 GB of RAM. To have controllable scope, we tested these executable knowledge graphs on a sample welding production dataset collected from a factory in Germany. The dataset is in the form of relational tables after integration, and contains 4585 welding operation records.

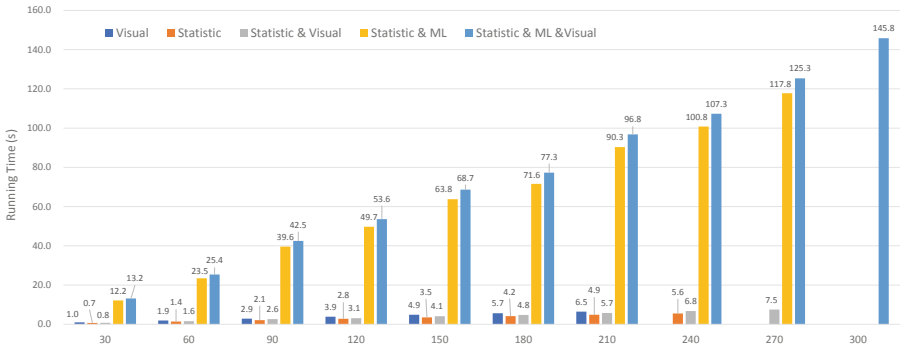

Fig. 9. System evaluation results for the Bosch welding use cases. x-Axis: number of tasks.

Results and Discussion. The running time (including knowledge graph translation and execution time) in Fig. 9 demonstrate that our system scales well since it takes limited time to translate the executable knowledge graphs to scripts and execute the scripts. Specifically, the running time grows sublinearly with respect to the number of tasks. On the most right hand side, 300 of the most challenging tasks, namely the hybrid tasks that combine statistic, machine learning modelling and visual analysis only takes less than 3 min on the given data, which is considered to have good scalability by our experts.

7 Conclusion, Lessons Learned, And Outlook

In this work we present our concept and system of executable knowledge graphs, which address the challenges of transparency, formal description, and reusability of machine learning practice, including visual, statistical, and machine learning analytics tasks. The system helps users to do the machine learning-related analytics tasks by providing a GUI and executable knowledge graphs that can be translated to executable scripts. We evaluated our approach with a user study, discussion, workshops, and system evaluation and obtained promising results. The lessons learned for us that as follows: many users are very interested in machine learning-related knowledge and solutions. They are eager to spend time to learn such knowledge and practice machine learning, but did not have a good starting point. They highly value our system and proposed many comments for improvement, especially for the GUI, and for covering the hyper-parameter tuning and feature learning. In the future, we plan to further improve our system, to host the system regularly on the Bosch environment and constantly collect more user feed-backs. We also plan to develop more theory to improve the generality of the approach.

Acknowledgements. The work was partially supported by the H2020 projects Dome 4.0 (Grant Agreement No. 953163), OntoCommons (Grant Agreement No. 958371), and DataCloud (Grant Agreement No. 101016835) and the SIRIUS Centre, Norwegian Research Council project number 237898.

References

1. Naab, C., Zheng, Z.: Application of the unscented Kalman filter in position estimation a case study on a robot for precise positioning. Robot. Auton. Syst. **147**, 103904 (2022)
2. Svetashova, Y., et al.: Ontology-enhanced machine learning: a bosch use case of welding quality monitoring. In: ISWC (2020)
3. Celik, O., Zhou, D., Li, G., Becker, P., Neumann, G.: Specializing versatile skill libraries using local mixture of experts. In: Conference on Robot Learning, PMLR, pp. 1423–1433 (2022)
4. Mahesh, B.: Machine learning algorithms-a review. Int. J. Sci. Res. (IJSR) (Internet) **9**, 381–386 (2020)
5. Kagermann, H.: Change through digitization - value creation in the age of industry 4.0. In: Management of Permanent Change (2015)
6. Zhou, B., et al.: The data value quest: a holistic semantic approach at Bosch. In: European Semantic Web Conference, pp. 287–290. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11609-4_42
7. Perer, A., Shneiderman, B.: Integrating statistics and visualization: case studies of gaining clarity during exploratory data analysis. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 265–274 (2008)

8. Zhou, B., et al.: SemML: facilitating development of ML models for condition monitoring with semantics. J. Web Semant. **71**, 100664 (2021)
9. Zhou, B., Pychynski, T., Reischl, M., Kharlamov, E., Mikut, R.: Machine learning with domain knowledge for predictive quality monitoring in resistance spot welding. J. Intell. Manuf. **33**(4), 1139–1163 (2022)
10. ISO, Resistance Welding - Procedures for Determining the Weldability Lobe for Resistance Spot, Projection and Seam Welding, Standard, International Organization for Standardization, Geneva, CH (2004)
11. DVS, Widerstandspunktschweißen von Stählen bis 3 mm Einzeldicke - Konstruktion und Berechnung, Standard, Deutscher Verband für Schweißen und verwandte Verfahren e. V., Düsseldorf, DE (2016)
12. Zhou, B., Svetashova, Y., Byeon, S., Pychynski, T., Mikut, R., Kharlamov, E.: Predicting quality of automated welding with machine learning and semantics: a Bosch case study. In: CIKM (2020)
13. Zhou, B., Svetashova, Y., Pychynski, T., Kharlamov, E.: Semantic ML for manufacturing monitoring at Bosch. In: ISWC (Demos/Industry), vol. 2721, p. 398 (2020)
14. Svetashova, Y., Zhou, B., Schmid, S., Pychynski, T., Kharlamov, E.: SemML: reusable ML for condition monitoring in discrete manufacturing. In: ISWC (Demos/Industry), vol. 2721, pp. 213–218 (2020)
15. Zhou, B.: Machine learning methods for product quality monitoring in electric resistance welding, Ph.D. thesis, Karlsruhe Institute of Technology, Germany (2021)
16. Zhou, B., Zhou, D., Chen, J., Svetashova, Y., Cheng, G., Kharlamov, E.: Scaling usability of ML analytics with knowledge graphs: exemplified with a Bosch welding case. In: The 10th International Joint Conference on Knowledge Graphs, pp. 54–63 (2021)
17. Keim, D., Andrienko, G., Fekete, J.-D., Görg, C., Kohlhammer, J., Melançon, G.: Visual analytics: definition, process, and challenges. In: Kerren, A., Stasko, J.T., Fekete, J.-D., North, C. (eds.) Information Visualization. LNCS, vol. 4950, pp. 154–175. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-70956-5_7
18. Endert, A., Han, C., Maiti, D., House, L., North, C.: Observation-level interaction with statistical models for visual analytics. In: IEEE Conference on Visual Analytics Science and Technology (VAST), pp. 121–130. IEEE (2011)
19. LaCasse, P.M., Otieno, W., Maturana, F.P.: A survey of feature set reduction approaches for predictive analytics models in the connected manufacturing enterprise. Appl. Sci. **9**(5), 843 (2019)
20. C. ISO, 9241-11.3. Part II: Guidance on Specifying and Measuring Usability, ISO 9241 Ergonomic Requirements for Office Work With Visual Display Terminals (VDTs) (1993)
21. Kartchner, D., Christensen, T., Humpherys, J., Wade, S.: Code2Vec: embedding and clustering medical diagnosis data. In: 2017 IEEE International Conference on Healthcare Informatics, ICHI 2017, Park City, UT, USA, 23–26 August 2017, pp. 386–390. IEEE Computer Society (2017)
22. Atzeni, M., Atzori, M.: CodeOntology: RDF-ization of source code. In: d'Amato, C., et al. (eds.) ISWC 2017. LNCS, vol. 10588, pp. 20–28. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68204-4_2
23. Qu, Y., et al.: node2defect: using network embedding to improve software defect prediction. In: 2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 844–849. IEEE (2018)
24. Zheng, Z., et al.: Query-based industrial analytics over knowledge graphs with ontology reshaping. In: ESWC (Posters & Demos). Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11609-4_23
25. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250 (2008)

26. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC -2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-76298-0_52
27. Li, F.-L., et al.: AliMeKG: domain knowledge graph construction and application in e-commerce. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 2581–2588 (2020)
28. Soylu, A., et al.: TheyBuyForYou platform and knowledge graph: expanding horizons in public procurement with open linked data. Semant. Web **13**(2), 265–291 (2022)
29. Roman, D., et al.: The euBusinessGraph ontology: a lightweight ontology for harmonizing basic company information. Semant. Web **13**(1), 41–68 (2022). https://doi.org/10.3233/SW-210424
30. Li, L., et al.: Real-world data medical knowledge graph: construction and applications. Artif. Intell. Med. **103**, 101817 (2020)
31. Ryen, V., Soylu, A., Roman, D.: Building semantic knowledge graphs from (semi-)structured data: a review. Future Internet **14**(5), 129 (2022). https://doi.org/10.3390/fi14050129
32. Zhou, D., et al.: Ontology reshaping for knowledge graph construction: applied on Bosch welding case. In: ISWC. Springer, Cham (2022)
33. Zhou, D., Zhou, B., Chen, J., Cheng, G., Kostylev, E.V., Kharlamov, E.: Towards ontology reshaping for KG generation with user-in-the-loop: applied to Bosch welding. In: IJCKG (2021)
34. Zhou, D., et al.: Enhancing knowledge graph generation with ontology reshaping-Bosch case. In: ESWC (Demos/Industry). Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11609-4_45
35. Yahya, M., et al.: Towards generalized welding ontology in line with ISO and knowledge graph construction. In: ESWC (Posters & Demos). Springer, Cham (2022). https://doi.org/10.1007/978-3-031-11609-4_16