Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

An extension of iStar for Machine Learning requirements by following the PRISE methodology

Jose M. Barrera [a,b], Alejandro Reina-Reina [a,b], Ana Lavalle [a,*], Alejandro Maté [a], Juan Trujillo [a]

[a] Lucentia Research Group, Department of Software and Computing Systems, University of Alicante, Carretera San Vicente del Raspeig s/n, San Vicente del Raspeig, 03690, Spain
[b] Lucentia lab, Av. Pintor Pérez Gil, 16, Alicante, 03540, Spain

ARTICLE INFO

Keywords:
Requirements engineering
Machine learning
iStar
Conceptual modeling

ABSTRACT

The rise of Artificial Intelligence (AI) and Deep Learning has led to Machine Learning (ML) becoming a common practice in academia and enterprise. However, a successful ML project requires deep domain knowledge as well as expertise in a plethora of algorithms and data processing techniques. This leads to a stronger dependency and need for communication between developers and stakeholders where numerous requirements come into play. More specifically, in addition to functional requirements such as the output of the model (e.g. classification, clustering or regression), ML projects need to pay special attention to a number of non-functional and quality aspects particular to ML. These include explainability, noise robustness or equity among others. Failure to identify and consider these aspects will lead to inadequate algorithm selection and the failure of the project. In this sense, capturing ML requirements becomes critical. Unfortunately, there is currently an absence of ML requirements modeling approaches. Therefore, in this paper we present the first i* extension for capturing ML requirements and apply it to two real-world projects. Our study covers two main objectives for ML requirements: (i) allows domain experts to specify objectives and quality aspects to be met by the ML solution, and (ii) facilitates the selection and justification of the most adequate ML approaches. Our case studies show that our work enables better ML algorithm selection, preprocessing implementation tailored to each algorithm, and aids in identifying missing data. In addition, they also demonstrate the flexibility of our study to adapt to different domains.

1. Introduction

The use of Artificial Intelligence (AI) has dramatically increased in recent years. Thanks to the latest advances and the proliferation of data science and deep learning, data-intensive problems can now be solved and better decisions can be made. One of the branches of AI is Machine Learning (ML). In this field, mathematical models learn weights to define the rules that guide the distribution function of a data set.

The use of ML models can aid to answer questions related to classification such as *"Will this person infected with COVID-19 survive with this health and demographic data?"*, regression *"Which are the estimated sales of our company with these characteristics?"* and clustering *"Which are the common patterns of people infected by COVID-19?"* among others.

Despite its widespread use and power, as [1] argues, requirements are a non-resolved challenge in ML projects. How well project requirements are translated into concepts, features and ML metrics depends entirely on the expertise of the data scientist or data analyst. To further aggravate the situation, capturing requirements in an ML project requires both technical knowledge of ML as well domain knowledge [2], making it difficult to find suitable candidates.

According to [3], requirements are paramount in ML projects. Incomplete or incorrect requirements capture can imply costly drawbacks in the project, which are commonly detected too late during implementation [4]. Consequently, there is a need to improve ML requirements capture. Given the involvement and interaction between domain and technical knowledge, the ideal ML requirements language is one that

Abbreviations: AI, Artificial Intelligence; ML, Machine Learning; i*, iStar; RE, Requirements Engineering; GORE, Goal-Oriented Requirements Engineering; GRL, Goal Requirements Language; PRISE, PRocess to support IStar Extensions; SVM, Support Vector Machine; KNN, K-nearest neighbors; DBScan, Density Based Spatial Clustering of applications with noise; OCL, Object Constraint Language; API, Application Programming Interface; IT, Information Technologies; FRs, Functional Requirements; NFRs, Non-Functional Requirements; SCADA, Supervisory Control and Data Acquisition; GMM, Gaussian Mixture Model; SPEC, Spectral Clustering; EHR, Electronic Health Record; QA, Quality Aspect; DS, Dataset; PSO, Particle Swarm Optimization

* Corresponding author.
E-mail address: alavalle@dlsi.ua.es (A. Lavalle).

https://doi.org/10.1016/j.csi.2023.103806
Received 16 January 2023; Received in revised form 13 October 2023; Accepted 28 October 2023
Available online 31 October 2023

bridges the gap between the domain expert and the ML developer. It has to be understandable enough to specify the project objectives to be met while relating these objectives to ML concepts. Among the existing requirement languages, Goal-Oriented Requirements Engineering (GORE) frameworks stand out for this task, organizing project requirements into goals that can be specified by domain experts and linking them to domain concepts that can be specialized.

One of the main and most widespread GORE frameworks is iStar (henceforth dubbed i*). i* has become the de-facto standard since it can be applied to any field thanks to its high level of abstraction. It has received numerous extensions in different areas. In 2017 Gonçalves et al. [5] made a systematic literature review of iStar extensions identifying 51 papers which directly extend iStar, 70 which extend iStar through Tropos [6] and 56 through GRL [7]. It also has been applied to largely disparate areas such as safety-critical systems [8], multi-agent systems [9], analytical visualizations [10], and risk assessment [11] among others. Consequently, due to its widespread use, a systematic methodology to create i* extensions called PRISE [12] (PRocess to support IStar Extensions) emerged. By following the steps outlined by this methodology, it is possible to create i* specializations that can capture the specific constructs and rules of the field where it is applied.

Additionally, we should highlight that, although capturing requirements is a high-level task, each requirement should be converted into technical concepts that can be implemented by ML Experts in lower-level abstraction layers. Therefore, any solution that aims to tackle ML requirements capture needs to consider not only high-level abstraction concepts, but also how these concepts will be translated into alternative ML implementations and solutions.

Therefore, in this paper, we have conducted an extensive research by applying the formal PRISE [12] methodology in order to create the first i* for capturing ML requirements. Our work not only enables capturing ML requirements, but it also includes a set of guidelines that facilitate its application. On one hand, it drives the creation of the requirements model through a questionnaire. This questionnaire guides ML experts in their interview with domain experts, ensuring that all crucial requirements are evaluated. On the other hand, we provide a set of tables that reflect how existing ML models contribute to non-functional requirements and which ML metrics can be used for each kind of goal. This facilitates the identification and selection of the most appropriate ML models and metrics according to the goals and priorities of the quality aspects. In order to evaluate our research, we have applied it in two real-world projects, the first one based on gas turbines for electricity generation and the second one based on analyzing patients affected by COVID-19. We should also point out that we have used two real world projects from absolutely different domains in order to better check the suitability of our research.

As demonstrated by our case studies, the advantages of our work are:

- It allows domain experts to specify their goals, driven by the questionnaire provided and the aid of ML experts.
- Improves the selection of the most adequate ML models and metrics according to the specified goals, non-functional requirements and the contribution tables provided.
- Aids in the identification of missing requirements and data that must be gathered for the project.
- Can be applied to ML projects in different domains.

The remainder of the paper is structured as follows: Section 2 presents the related work in the field of Machine Learning requirements and the use of the i* and its extensions to different fields. In Section 3 we have conducted an extensive research by following the main steps of PRISE in order to create a novel iStar extension for ML. Section 4 presents our guide to apply our work and facilitate the capture of requirements in ML projects. In Section 5 our iStar extension for ML is evaluated through two real case studies. Finally, Section 6 summarizes the conclusions and future works.

2. Related work

Over the last two decades, Goal-Oriented Requirements Engineering (GO-RE) has attracted attention from the research community [13]. The i* framework is a goal-based modeling language used to model requirements. More recently, the i* framework has been modified from its initial version [14] to its current version 2.0 [15].

Several extensions have been created since its creation. The range of specifications of i* covers a wide range of fields, from law to gamification [5]. Among recent extensions, we can find for instance [16], where authors proposed an extension to help non-expert users in data visualization to communicate their analytical needs and generate automatically the visualizations that best fit their requirements. In [17], an iStar extension that contributes to the Open Innovation business paradigm is presented. Other works focused on Human-Centric Characteristics [18,19] use personas and contexts to model human-centric aspects of the software in goal models. Other novel tools such as BiStar [20] combine graphical and textual i* modeling.

However, authors in [5] detected an elevated number of anomalies related to symbols in i* extensions. These anomalies included symbol redundancy (when a semantic construct is represented by multiple symbols), symbol overload (when the same symbol is used to represent multiple constructs), symbol deficit (when a semantic construct is not represented by any symbol) or symbol excess (when a symbol does not represent any semantic construct). As a result, the PRISE [12] methodology was proposed for defining i* extensions, in order to avoid the aforementioned issues.

Nevertheless, despite the extensive use of i*, to date there have been no i* approaches related to Machine Learning requirements that cover functional and non-functional requirements. In fact, no extension related to ML can be found in the i* extensions catalogue [21].

Although no i* extensions can be found, there is a small set of works related to Machine Learning and Requirements Engineering (RE). In [1], authors support ML system development by suggesting checklists that allow developers to complete the basic tasks in ML project development. However, they do not specifically address the issues of RE. In [3], a survey with experts in ML projects is presented. Authors highlight the substantial differences between ML and non-ML projects requirements. Nevertheless, no solution is provided to guide the capture of requirements in ML projects.

There are also some works that partially cover either functional or non-functional requirements. In [22] (GORE-MLOps), a specific method for linking GORE with ML is proposed. Nevertheless, they only consider functional requirements (FRs). In our approach, we also include non-functional requirements (NFRs) such as transparency, drift in production environments, fairness, scalability and others. In [23], the authors focus on the quality attributes and define the main NFRs in the ML field. Similarly, in [24] the authors focus on capturing only NFRs in ML projects whereas we also consider FRs such as temporal resolutions, reliability or validation.

Regarding works that consider requirements in general, the authors in [25] present a research roadmap for capturing and identifying requirements based on questions. However, they do not define indicators to monitor the requirements. In [26] a roadmap to follow requirements in ML productions is presented, based on their experience, but without taking formal requirements into account. In [27], the authors propose a model that covers the main gaps between requirements engineering and machine learning through solution patterns. However, the framework lacks specific constraints related to constructs from the ML field. For this reason, it requires deep domain knowledge to be applied.

Finally, it is key to note that works such as [22,23,25] have provided initial steps in ML requirements. However, they only provide the theoretical fundamentals and do not provide guidance or implementation of the solution.

As shown, to the best of our knowledge none of the approaches summarized above provide a solution that tackles the problem considering both functional, non-functional requirements as well as their associated implementation. Most of the approaches define the theoretical fundamentals without including practical and systematic derivation of the implementation that could help designers with the requirements gathering and elicitation. Compared to these approaches, our approach is based on a specific and semantically-tested language, providing a suitable metamodel that can be implemented, helping us with gathering both functional and non-functional requirements.

Moreover, our work is the first i* extension applied to Requirements Engineering in ML. This extension has been created following an authoritative framework specifically designed for this purpose (PRISE). This allows us to lower the high-level abstraction of i*, developing a more-specific, less error-prone, semantically-valid, and more constrained modeling language that is adapted to the needs of the ML field. Moreover, our approach includes a series of guidelines to follow in order to avoid skipping any important requirement that should be considered in an ML project.

Finally, as the presented model is more directed and less error-prone, ML projects can be developed with less effort and time, making ML development more efficient. Therefore, we believe that our approach complements all the previous theoretical works by providing a metamodel that allows designers to gather the main requirements of ML projects in a systematic fashion.

3. Applying PRISE to create an istar extension for ML

As previously mentioned, the i* framework has received numerous extensions and as a result, a systematic process for the creation of i* extensions called PRISE [12] has been proposed in the literature. In this section, we detail the PRISE steps that we have applied in order to create an i* extension to capture Machine Learning (ML) requirements. It is important to highlight that PRISE only provides the steps to be followed in order to create the i* extension. The actual domain knowledge and decisions on what concepts and relationships should be included must be provided by the authors of the extension created. In the case of the i* extension for ML, the authors of this paper.

PRISE is structured in 6 main steps summarized in Fig. 1. The process starts with the necessity of building an i* extension and it ends when the extension is built, in this case formalized through a metamodel (Fig. 3). It is an iterative process where new constructs can appear while steps 2, 3 and 4 are being performed. Despite being an iterative process, the analysis of external iStar models is performed individually, once the constructs of an extension are added, the next model is explored.

Every main step in PRISE has sub-processes. However, with the aim of providing only relevant information, the most important topics related to the main step are presented. By following the steps outlined by PRISE, we make sure to create a complete, consistent, and conflict-free i* extension. The following sections describe in detail the creation of our work by applying the PRISE methodology.

3.1. INPUT: Need to extend istar

In recent years, the adoption of ML solutions has experienced a significant growth. Despite its widespread use and power, requirements are a non-resolved challenge in ML projects [1]. Incomplete or incorrect requirements capture can imply costly drawbacks in the project, which are commonly detected too late during implementation [4]. Moreover, capturing requirements in an ML project requires both technical knowledge of ML as well domain knowledge [2].

Therefore, there is a need to improve the process of capturing ML requirements. Considering the involvement and interaction between domain expertise and technical knowledge, the ideal language for ML requirements should act as a bridge connecting domain experts and ML developers. This language has to be clear enough to specify the project objectives while also relating these objectives to ML concepts.

3.2. STEP I: Analyze the need of extension

The first step is to analyze the need of an i* extension. As mentioned in Section 2, Requirements Engineering (RE) in Machine Learning (ML) projects is still a challenging problem. Only a few approaches, such as [27], tackle this task by trying to cover the main gaps between RE and ML. However, the solution patterns offered have too much freedom, and some inconsistencies are likely to appear.

Moreover, according to [2], RE in ML requires both domain knowledge and the ability to translate high-level organization requirements into low-level understandable and implementable goals. This implies that talent can be expensive or hard to retain. Moreover, RE can be difficult for non-experts in ML, producing errors that would be detected in late stages during the implementation phase.

Another reason for considering crucial the need of an i* extension for ML is the incapability of capturing ML constraints. Using i* on its high-abstraction layer can produce inconsistencies. For instance, as Fig. 2 shows, although both elements are correctly typed as tasks and goals, the relationship between them is not correct. Kmeans is a classification task, while R^2 is a regression metric, which cannot be used to define classification goals. They cannot be related. Consequently, the correctness of the chosen elements relies entirely on the burden of knowledge of the ML expert.

Therefore, it is paramount to provide a semantically-valid extension of i*, where construct constraints and specific rules of ML are captured.

3.3. STEP II: Describe concepts of the istar extension

According to PRISE [12], it is recommended to search, select and reuse concepts from other works. We have reused generic concepts that are not related specifically to ML. For instance, i* elements such as Actor, Goal or Task [28] or concepts proposed in [29] as Datasource.

For the specific case of ML elements, approaches as [27] describe interesting elements related to RE and ML. However, the relationships between specific ML constructs are not ideal. Therefore, we prefer not to use directly those concepts and, instead, we will provide similar ML constructs but with other relationships between them.

In the following, we describe the new concepts that have been introduced in our proposed work of i* extension from ML requirements:

- ***Source***: This element represents a specific source from which data is obtained. It can be internal data (data that belongs to the company) or external data (provided by open data repositories or institutional open API's). It specializes the i* class *Resource*.
- ***Dataset***: Sources can be very different from each other. For instance, they can have different timestamp units and metrics. As a result, they must be unified under a single criterion, forming the Dataset element. *Dataset* will be the main input for the *Task* elements (specifically, *MLTask)*. The dataset element has two related attributes: *temporalResolution*, which is an attribute that unifies the temporal resolution of different Sources, and *refreshPeriod*, which specifies when the Dataset must be refreshed with new data for other *MLTasks* (for instance, re-training the ML model).
- ***MLGoals***: This abstract class is a specification of the i* Goal class. *MLGoals* are the abstraction that can be instantiated into three different ML goals. It must be highlighted that with the construct of *MLGoals*, it is easier to establish a boundary between goals related to different fields (for instance, ML goals and chemical goals). Therefore, RE can be approached from a multidisciplinary point of view if required by the project. The three specialized classes are related to the three main tasks in ML: classification, regression, and clustering. A *MLGoal* is the desired level of compliance that should be achieved for a specific qualitative goal.
- ***ClassificationGoal***: The specification of *MLGoals* associated with a classification goal. It has a parameter to specify the temporal resolution of the goal.

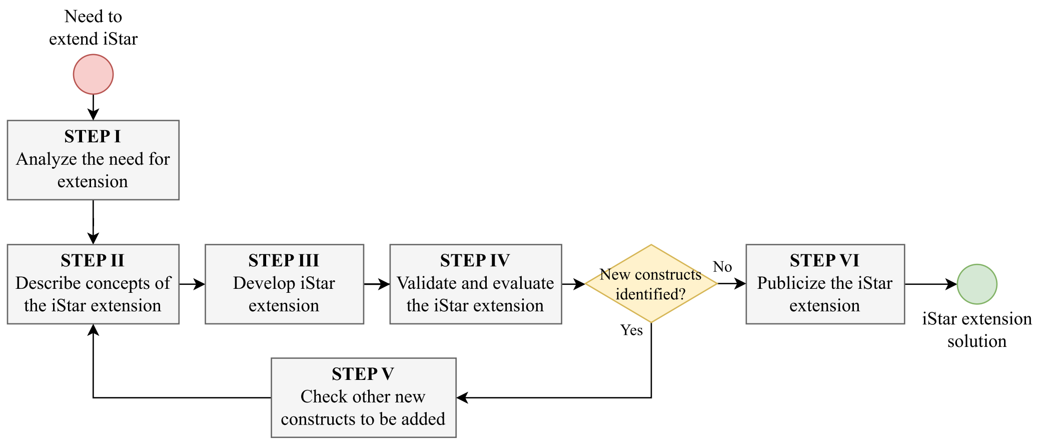
Need to extend iStar
STEP I Analyze the need for extension
STEP II Describe concepts of the iStar extension
STEP III Develop iStar extension
STEP IV Validate and evaluate the iStar extension
New constructs identified?
No
Yes
STEP V Check other new constructs to be added
STEP VI Publicize the iStar extension
iStar extension solution

Fig. 1. Main steps of PRISE.

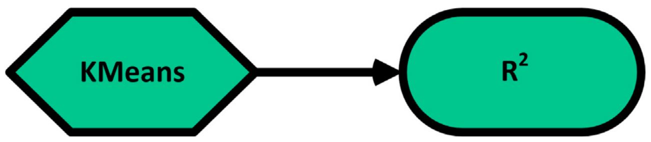

Fig. 2. Inconsistency between a task and a goal in ML, using base i*.

- ***RegressionGoal***: The specification of *MLGoals* associated with a regression goal. As in the previous case, it has a parameter to specify the temporal resolution of the goal.
- ***ClusteringGoal***: The specification of *MLGoals* associated with a clustering goal. Unlike *ClassificationGoal* and *RegressionGoal*, it has no temporal resolution attribute.
- ***MLTask***: It is a specialization of i* Task class. It can be split into three more specific classes: *ClassificationTask*, *RegressionTask* and *ClusteringTask*. Furthermore, it has common attributes belonging to all ML tasks. It has the *validationTypes* attribute, which specifies if either a train test-split must be performed (the hyper-parametrization must be realized afterwards), or a train-test-validation split should be performed instead (the hyper-parametrization is included in that process). Moreover, it has the parameters *TrainingSize* and *ValidationSize*, which specify the train and the validation sizes (for instance, 80% train size and 20% train size; or 90% train size and 10% test size). Finally, the *CVFolds* attribute specifies how many folds must be done in a cross-validation technique, with the aim of providing more robust results to the ML model [30]. Every *MLTask* will generate an *Indicator*.
- ***ClassificationTask***: The specification of *MLTask* associated with a classification task. Classification can be binary or multiclass, specified by a boolean attribute (*binaryClass*). Furthermore, simple classification models can be aggregated to form a more complex classification model, following techniques such as bagging and boosting. These techniques are included through the *aggregatedClassifier* relation.
- ***RegressionTask***: The specification of *MLTask* associated with a regression task. Similar to *ClassificationTask*, simple regression models can be aggregated to form a more complex regression model, represented through the *aggregatedRegressor* relation.
- ***ClusteringTask***: The specification of *MLTask* associated with a clustering task.
- ***Indicator***: An Indicator is the parameter that specifies the degree of compliance of a specific *MLGoal*, to be achieved by means of a *MLTask*. In an ideal use case, the *MLIndicator* would have an equal or higher level of compliance than the one set for the *MLGoal*.
- ***ClassificationIndicator***: The specialization of *Indicator* associated with a classification indicator.
- ***RegressionIndicator***: The specialization of *Indicator* associated with a regression indicator.
- ***ClusteringIndicator***: The specialization of *Indicator* associated with a clustering indicator.
- ***DataPreparation***: This is a specialization of i* Task. This element has an enumeration of additional preprocessing operations that can be performed in ML, according to project needs and the algorithm used. For instance, *Normalization* is a requirement for using a SVM algorithm, but it is not required for a *RandomForest* approach. On the other hand, an operation related to *OutliersTreatment* is mandatory if the algorithm is sensible to outliers (as KNN), while other algorithms such as DBScan are quite robust to them. These relationships between algorithms and *DataPreparation* attributes will be controlled through OCL (Object Constraint Language) constraints [31].
- ***MLQualityAspects***: This element specifies the different ML qualities that the model may be expected to fulfill. These can be: Scalability, NoiseRobustness, ConceptDriftAdaptation, Explainability, HighDimensionality, Equity, Simplicity and Transparency. However, not all ML models are able to cover all quality aspects. Therefore, in order to help with the selection of the most suitable ML models to achieve the defined *MLQualityAspects* we have proposed Table 2 (fully explained in Section 4).

We must highlight that our study is focused on capturing requirements from the ML point of view. This implies that other aspects related to IT projects but not related to ML are not included. For example, the requirements of visualizations, security in data transactions, documentation, or credibility of data are out of the scope.

3.4. STEP III: Develop iStar extension

After describing all the concepts that are included in our study, we have formalized it by creating a metamodel, implemented using the ECORE (ECLIPSE) tool. Fig. 3 shows our resulting metamodel.

One of the key points of our work is the relation between the specifications of *MLGoal—MLTask—Indicator*. Our model avoids invalid configurations: for instance, the degree of compliance of a classification task is measured with a classification metric, which is trying to achieve a classification goal. Moreover, different *MLQualityAspects* are captured from business needs. The different *MLTasks* that help to achieve *MLGoals* can be related to these quality aspects, and by using Table 2, *MLTasks* that are not suitable (due to these algorithms breaking or hurting desirable *MLQualityAspects*) can be discarded. This ensures that only valid ML algorithms will be selected. Therefore, the number of

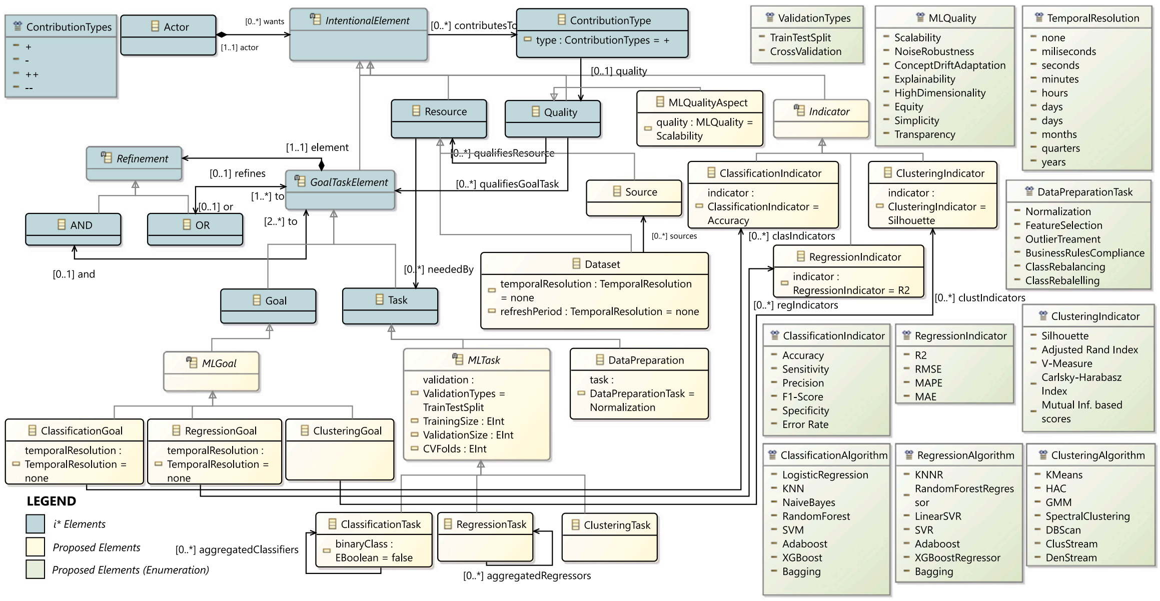

Fig. 3. Proposed extension of i* for Machine Learning projects.

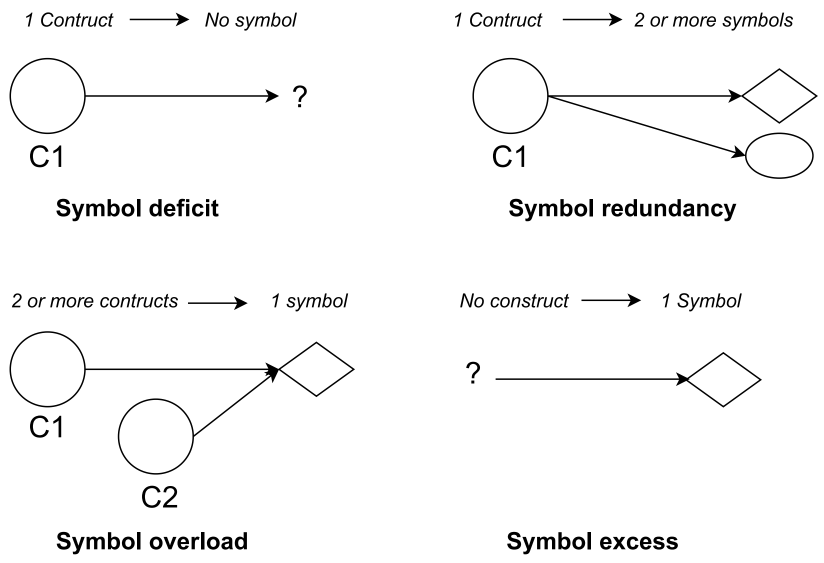

Fig. 4. Main errors in iStar extensions while encoding.

mistakes and inefficient resource consumption training less-than-ideal algorithms will be greatly reduced. Moreover, the use of enumerated classes helps to avoid mistakes, since it is ensured that only appropriate indicators for the ML task will be selected.

Finally, we have created a questionnaire in order to guide and assist in the application of our work (described in Section 4). By answering those questions, the metamodel can be built iteratively with information about the specific use case.

3.5. STEP IV: Validate and evaluate the iStar extension

Our research has been validated and evaluated in each iteration until reaching the final metamodel. We have thoroughly revised our metamodel with the aim of avoiding mistakes. The main common mistakes of an i* extension are represented in Fig. 4. Our study has focused on avoiding these errors. In our proposed metamodel it is not possible to find errors of symbol redundancy, symbol deficit, symbol overload or symbol excess. Only the specialization of the abstract classes to more specific elements could be interpreted as a symbol redundancy. However, it has been discarded as an error, since it is not possible to instantiate an abstract class, the specification is needed, and it discriminates between different ML elements. Moreover, we have also checked and corrected any problems related to completeness, consistency, and conflicts.

Furthermore, in this paper, we fully apply our study to two real case studies. In Section 5.1 we apply it to an industrial project based on gas turbines for electricity generation. In Section 5.2, we apply it to a case study based on patients affected by COVID-19.

3.6. STEP V: Check other new constructs to be added

This step has been carried out iteratively. PRISE is an iterative process where new constructs can be identified during steps II, III and IV. Thus, when new constructs are identified during an iteration, they are listed by this step to be considered in the next iteration.

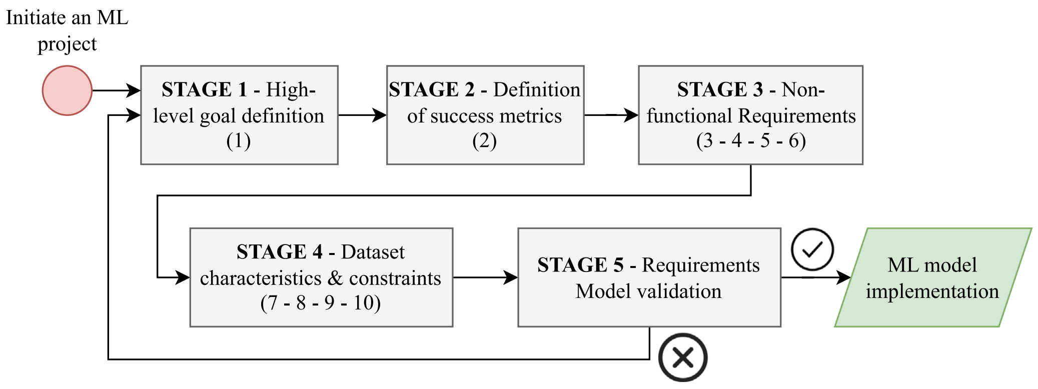
Initiate an ML project
STAGE 1 - High-level goal definition (1)
STAGE 2 - Definition of success metrics (2)
STAGE 3 - Non-functional Requirements (3 - 4 - 5 - 6)
STAGE 4 - Dataset characteristics & constraints (7 - 8 - 9 - 10)
STAGE 5 - Requirements Model validation
ML model implementation

Fig. 5. Main stages of our process for collecting Machine Learning requirements.

3.7. *STEP VI: Publicize the iStar extension*

Finally, the last step of PRISE is to publicize our new i* extension for ML. First, this extension is aimed to be published in a widely known journal in order to reach a wider public. Second, we are also working to include it in the i* extension catalogue[1] after its publication.

4. Guide to capture ML requirements by using our iStar extension

One of the most important aspects of RE in ML is to capture and transform business objectives into clear analysis objectives [32]. Moreover, those analytical objectives must be measured using suitable metrics. However, capturing project goals and defining metrics to measure whether they have been achieved or not is a challenging task.

In this work, we have formalized the process of capturing ML requirements through a complete metamodel based on i*. Moreover, with the aim of supporting the requirements capture step, we have established a guided process shown in Fig. 5. This process is comprised of a set of stages based on our requirements questionnaire tool, that is summarized at the end of the section in Table 3. For quick reference, in the following we will refer to questions within the questionnaire tool as *QX*, where *X* indicates the number of the question. The process begins when an ML project is going to be developed, it provides an initial guide about the direction that an ML project must follow. Each stage helps the ML expert to build iteratively the i* model in a guided way and to ensure that all the relevant aspects of the ML solution are covered. After having collected all the information necessary to build the i* model and validate its construction, we will obtain as output the selection of the most suitable ML model that fulfills the requirements defined during the process.

The aim of this process and the requirements questionnaire tool is to guide the ML expert in building capturing the project requirements. Since the some of the points in the questionnaire can require high technical ML knowledge, the ML expert can translate them into more understandable questions for the domain expert if necessary. Thus, by following the defined process, the metamodel can be instantiated for a specific use case, and consequently, converted into a specific i* for ML model.

In the following, we present in detail the set of stages defined by the authors together with the points associated to each stage:

4.1. *Stage 1 - High-level goal definition*

To start with, we must detect which are the main goals of the ML project. These ML goals are usually supplied by the stakeholder in natural language, where no computer science knowledge is required. Moreover, these high abstraction goals must be split into more specific goals that can be assigned to specific ML models. This stage can be tackled by with the first point from the questionnaire tool presented in Table 3:

P1. Defining the overall objective of the project from a ML point of view. By having the stakeholder answer Q1 from the questionnaire tool, we will capture the highest abstraction *Goals* of the project on a qualitative level. Moreover, the direction of the project will be established, since the approach to follow will be selected from:

- **Classification Goal:** I want to predict a discrete variable or a specific class from a predetermined set (e.g. predict within a 7-day margin if the COVID-19 patient will die).
- **Regression Goal:** I want to predict the numerical value of a continuous variable (e.g. predict properly in how many days a COVID-19 patient will die).
- **Clustering Goal:** I want to group data that have similar behaviors (e.g. detect common patterns in the patients that have died of COVID-19).

4.2. *Stage 2 - Definition of success metrics*

After specifying which are the ML goals of the project, we have to specify how to determine whether the project can be considered a success or not, which will in turn enable selecting across implementation alternatives. In order to perform trade-off analysis between ML goals and the efficacy/efficiency of the ML model, suitable metrics and thresholds for the goals established in the previous stage must be selected. In conjunction with quality aspects defined in the next step will enable ML experts and stakeholders to compare across existing alternatives. This stage can be tackled by the second point of the questionnaire tool:

P2. Define proper measures for planned goals that establish the success or failure of the project. While Point 1 establishes qualitative goals, this second point is focused on a quantitative perspective. By interviewing the stakeholder using Q2 from the questionnaire tool, a suitable indicator and a metric value will be selected for measure each of the previously defined goals.
In order to assist in the definition of the Indicators we created Table 1. This table summarizes the possible metrics that can be selected for each type of MLGoal.
Although metric selection may apparently be easy, there are many issues that can arise in real ML projects such as noise, biased datasets, creation and deletion of new features, etc, which can lead to choosing non-proper metrics and undesirable results. For example, Accuracy is a metric with very widespread use, although on many occasions it may not be suitable. If we want

[1] https://istarextensions.cin.ufpe.br/catalogue/

Table 1
Suitable Indicators for each MLGoal type.

Classification	Regression	Custering
- Accuracy	- R2	- Silhouette Coefficient
- Sensitivity	- RMBE	- Adjusted Rand Index
- Precision	- MAPE	- V-Measure
- F1-Score	- MAE	- Mutual Based Inference
- Specificity		- Carlinsky-Harabasz Index
- Error Rate		

to predict deaths in a COVID-19 project (a classification goal), and the dataset has a bigger representation of survivors than deaths, choosing an Accuracy metric may cause considerable errors when results are interpreted. Thus, if the model learns a simple rule such as *everyone will survive*, it will provide a good Accuracy metric. In this case, focusing the attention on a highly accurate prediction on the deaths group (using metrics such as precision, recall, or F1) will result in a more robust model. These types of metrics are more suitable for biased datasets, multi-class classification, etc.

Therefore, this second point will result in obtaining suitable *Indicators* (*ClassificationIndicator*, *RegressionIndicator*, *ClusteringIndicator)* that will be linked to specific *MLGoals* (*ClassificationGoal*, *RegressionGoal*, *ClusteringGoal*). Moreover, a specific value for each *Indicator* will be provided. And each *Indicator* will provide information about the degree of compliance between the actual value of the indicator vs the desired value of the indicator (*Indicator* vs *MLGoal*).

4.3. Stage 3 - Non-functional requirements

The aforementioned steps 1 and 2 will gather enough information to start dealing with the technical part of the project. However, other requirements such as quality aspects (including NFRs) that the project must fulfill are not covered yet. In fact, some recent literature [23] has highlighted the special attention NFR's require. However, ML qualities are not directly treated as NFR's (since some ML qualities can be considered functional requirements).

In our study, we present NFR's and ML qualities as different possibilities of one abstract element: *MLQuality*. While different ML qualities can be defined, there are three that we consider crucial from the requirements engineering point of view. For that reason, we have specific points in our requirements questionnaire that address each of those three ML qualities, and a fourth general one. The points related with ML qualities are the following:

P3. Rate the importance of being able to explain the overall decisions of the model as well as identifying the weights of individual dimensions in the output. Explainability is paramount in some research fields such as medicine [33,34]. Moreover, there are specific libraries designed to provide explainability to ML models, like SHAP [35]. In fact, in [36] authors show how legal requirements can be implemented into ML models, with the aim of dealing properly with explainability. Consequently, it must be established clearly if the explainability of the model is a desirable ML quality. If it is, some techniques and algorithms are more suitable to be used, while other techniques which lack transparency or are more difficult to be used to generate explanations coherent with the model logic are prone to be discarded (such as deep neural networks) [37].
The explainability of the model helps domain experts to understand which dimensions are relevant in the model, thus avoiding the black-box problem. Moreover, the explainability can be extended with the aim of providing the reasoning for a specific prediction including the effect of each dimension, as we can see in Fig. 6. Therefore, this point will aid in specifying if the *Explainability MLQuality* is required in the project and to which degree it should be fulfilled.

P4. Define whether the behavior of data can be expected to change as new data is gathered. This point is related to the change in data distribution. If the behavior of data changes, the model must be re-trained, due to the fact that new predictions will have a different distribution than old data. This change in data trend is known as concept drift [38]. Therefore, the trend must be captured to provide proper predictions. The retraining of the model usually is a time-consuming process and cannot be treated lightly. However, if data usually changes and the model must be re-trained continuously due to changes in the distribution, then traditional batch techniques cannot be used. For those cases, there are specific algorithms that deal properly with concept drift. By having the stakeholder answer Q4 from our questionnaire tool, the ML expert will be able to specify whether *ConceptDrift* as *MLQuality* is a requirement of the project.

P5. Determine whether data is expected to be biased and corrective actions should be taken. Real projects can have bias in multiple phases of the project (from social bias introduced into input data to deployment bias when the project has been finished) [39]. As a result, undesired effects such as unfairness or wrong decisions may arise during an ML project [40]. If bias is detected, it must be analyzed, and additional preprocessing steps should be performed with the aim of minimizing the aforementioned bias. If *Equity* is specified as an *MLQuality*, additional preprocessing operations should be performed.

P6. Establish the importance of the remaining quality aspects listed in the metamodel enumeration. There is a list of other possible ML quality aspects that may be desirable to take into account in the project (Table 2 lists the possible quality aspects). By posing Q6 of the questionnaire tool to the stakeholder during the interview, the ML expert will be able to identify other ML qualities that have to be considered, allowing filtering which algorithms are more suitable for fulfilling these quality aspects.

In order to assist in the selection of the most suitable *MLTask*, we present Table 2. This table relates the different ML qualities and the algorithms that fulfill them since not all ML algorithms are able to deal properly with all ML aspects. Table 2 represents the different relationships between each instance of an *MLTask* (*ClassificationTask*, *RegressionTask*, and *CusteringTask*) and *MLQualityAspects*. This relationship is established through a *ContributionType* class.

We establish a scale in Table 2 where the different symbols mean: ++ Makes; + Supports; - Harms; - - Breaks. It should be mentioned that it is possible to extend the list of ML algorithms and *MLQualityAspects* if the context needs it.

4.4. Stage 4 - Dataset characteristics & constraints

After defining the quality aspects, we focus on gathering information related to data sources. With this, we want to post which are the main tasks to develop over the data sources. Thus, those sources could be processed into a more refined dataset, which will help to achieve the aforementioned qualitative and quantitative ML goals. The following points from our questionnaire guide the identification of tasks that should be modeled in this stage:

P7. Establish the time horizon (if any) for which the solution should provide an output. By addressing this point we are specifying which data will be used for training the ML model and which data will be discarded.
For instance, we want to predict if a car will break down within the margin of 7 days with data provided by the car sensors in real time. In order to train the ML model, historical data generated

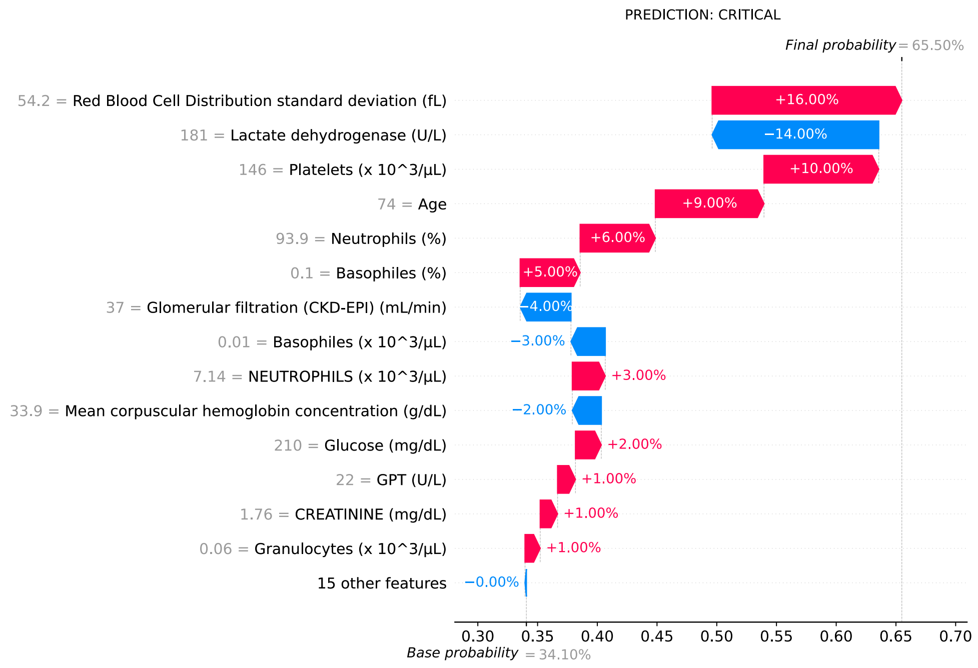

Fig. 6. Effects of every feature in a prediction. Explainability in an ML model.

Table 2
Suitable MLTasks for the different MLQualityAspects.

	Scalab.	Noise R.	C. Drift	Explain.	Dim.	Simp.
Classification algorithms						
LogisticRegression	+		−−	++	−	+
KNN	−−	−−	−−	+	−	+
NaiveBayes	++	+	−−		++	+
RandomForest	−−	+	−−		+	
SVM	−		−−	−	−−	
AdaBoost		−	−−	−	−	−
XGBoost	+	+	−−	−	−	−
Autoencoders	+	−−	−−	−−	+	++
Bagging			−−	+	+	−
Regression algorithms						
	Scalab.	Noise R.	C. Drift	Explain.	Dim.	Simp.
KNNR	++	−	−−	+	−	−
RandomForestRegressor	−−	+	−−		+	
LinearSVR	+		−−	++	−	+
SVR	−		−−	−	−−	
XGBoostRegressor	+	+	−−	−	−	−
Bagging			−−	+	+	−
Clustering algorithms						
	Scalab.	Noise R.	C. Drift	Explain.	Dim.	Simp.
Adaboost(Clustering)		−	−−	−	−	−
KMeans	+	−	−−	+		+
HAC	−	−−	−−	+	−	
GMM	+	+	−−	−	−	
SpectralClustering	−		−−		+	
DBScan	−	++	−−	−	−	−
CluStream			+		−	−
DenStream			++		−	−−

for sensors 7 days before its breakdown must be removed for every car from dataset (with the aim of avoiding the introduction of future information). Consequently, the data from training *Datasets* will be reduced. That could imply that no data (or insufficient data) is available for the qualitative goal pursued, and the answer should be re-evaluated. For instance, reducing the initial margin established.

P8. Identify available data sources for the model and whether domain knowledge can be applied over them. Much literature can be found where the inclusion of domain knowledge improves ML results [41–43]. Therefore, experts in the application field can provide valuable information with the aim of focusing an ML project. Moreover, the ML expert usually applies her knowledge in multiple different fields, where the ML expert is a newcomer. Consequently, domain knowledge is required quite often and it is usually provided by a stakeholder. However, it must be highlighted that sometimes either no domain knowledge is available or an agnostic approach has to be followed due to other constraints. Therefore, considering Q8 from the questionnaire tool will help the ML expert determine whether it will be necessary to modify the *Sources* element and therefore the *Dataset* element.

P9. Determine the granularity of data required according to the overall goal and compare it with the granularity available in the data sources. Granularity of data is paramount in the performance of an ML project. Some detailed metrics require a high frequency of input data (for instance, ten temperature values per second). Therefore, if the sensor can provide only one value every ten seconds, the goal cannot be achieved. On the other hand, if the goal requires computing an aggregated value, the aggregation operation can imply a heavy computing process and therefore, it could be unaffordable if time restrictions exist. This point will allow the parametrization of the attributes *refreshPeriod* and *temporalResolution* of *Dataset* element.

P10. Establish whether external validation will be required for the project. Sometimes, it may be required to train the model with specific data, and then test the model with data from another source (e.g. training with data from one hospital, and test it with data from another hospital). When external data is available (which implies additional *Source* and *Dataset* elements), it can provide more solid results to the specified model. However, this data is not always available. Consequently, internal validation is quite often the chosen option for testing the model.

Table 3
Questionnaire tool for ML experts to aid in ML requirements capture.

Question	Metamodel Class/es affected
Q1. From a qualitative point of view, which objectives do I have? Is a regression/clustering/classification model required?	*Goal* *MLGoal(ClassificationGoal, RegressionGoal, ClusteringGoal)*
Q2. Which metric is the proper one to measure the planned goal? Which value of that metric would consider the project as success?	*Indicators* (*ClassificationIndicator*, *RegressionIndicator*, *ClusteringIndicator*)
Q3. Is the explainability of the model or the effect of every dimension on each prediction mandatory?	*MLQuality* (*Explainability/Transparency*)
Q4. Can the behaviour of data be changed? New instances can be added, or it is static?	*MLQuality* (*ConceptDriftAdaptation*)
Q5. Is data biased? Is it necessary any additional preprocessing step with the aim of providing equity in data?	*MLQuality* (*Equity*)
Q6. Is there any other specific ML quality aspect that the ML model should have?	*MLQuality*
Q7. Should the system provide a prediction within a time limit?	*Dataset*
Q8. Which data should be considered for the model? Must an agnostic approach be followed to deal with the problem, or is domain knowledge available?	*Dataset* *Source*
Q9. Which granularity of data is suitable for achieving the goal? Is the required granularity available?	*Dataset* (*Parameters temporalResolution*) *and refreshPeriod*)
Q10. Is external validation required, or is it enough with internal validation?	*Source* *Dataset*

4.5. Stage 5 - Model validation

Finally, once the questionnaire is completed and the model is built, the domain expert and the ML expert review it in order to identify potential inconsistencies and oversights. In case the ML requirements model is validated, the ML project development will proceed. In the case of any inconsistency is detected, the questionnaire and the model will be reviewed in order to update it according to the domain expert's needs. In Table 3, we can observe the relationship between each posed question and the elements affected in the metamodel.

Once the ML Expert has passed through these stages and has collected the answer to these 10 Questions, an overview of the project will be available. Therefore, it will be possible to build the i* model by following the metamodel specified (Fig. 3) in an easy way avoiding common mistakes.

Each question will capture information about different elements of the metamodel. Questions 1 and 2 provide information about goals and metrics. Questions 3 to 6 are related to the ML Quality aspects that the ML solution must fulfill. Finally, questions from 7 to 10 are related to data and its processing before the ingestion in the ML model.

After gathering all the requirements for the ML model we have to specify which algorithms are prone to be candidate solutions and which algorithms should be discarded because they do not fulfill the project constraints. In order to accomplish this, the domain expert should evaluate the thresholds established for functional requirements (FRs) as well as the priorities for quality aspects (including NFRs). In order to facilitate the evaluation, we propose Eq. (1) to weight the suitability of each algorithm, establishing a ranking among the most promising ones:

$$S_A = \sum_{i=1}^{N} Da_i W_i \tag{1}$$

Where:

- S_A: Score of the algorithm.
- Da_i: Degree of compliance of the algorithm with Quality Aspect *i*. According to the relation between the algorithm and the quality aspect, four degrees of compliance can be identified. For algorithms that have ++ in Table 2, the degree of compliance will be considered 1. For algorithms that have + in Table 2, the degree of compliance will be considered 0.5. Conversely, negative contributions to quality aspects will be weighted with −1 (- -) and −0.5 (-) respectively.
- W_i: Weight of the Quality Aspect according to the priorities established by the project requirements.

5. Cases of study

In this section, we describe the application of our work in two real case studies with the aim of evaluating and demonstrating its applicability to different domains. The first case study is based on predicting anomalies in gas turbines for electricity production (Section 5.1). The second one is based on predicting the patients' evolution for patients affected by COVID-19 (Section 5.2).

It should be noted that in order to create the visual model, the base i* syntax has been respected. Consequently, *Task* specifications have *Task* visual syntax and *Goals* specifications have *Goals* visual syntax. However, *Indicator* cannot be inherited directly. Therefore, a widespread symbol for indicators previously used in other works such as [27] has been adopted.

5.1. Case study 1: Automatic anomalous working process detection in gas turbines

In this section, we present the application of our work to a real ML case study. First, we will provide the answers to the questions that allow us to iteratively build the model. These questions are presented in Section 4. Then, we will present the resulting requirements model.

The first case of study is based on a gas turbine used for electric energy generation. A gas turbine is the prime mover for the generator. Its input is thermal energy from burning gas and the output is mechanical power that drives the generator. Thus, this device converts the mechanical energy generated by the gas combustion into electrical energy that is supplied as a product at the terminals of the generator.

The industrial process that governs the turbine is very complex. The turbine model used in the case study incorporates more than 100 sensors, which provide information about every part of the device. The system measures 31 physical values with backup sensors and multiple measuring points distributed along the turbine and the generator. The aim of the project is to predict whether the turbine is working correctly or not, and if risks breaking down in order to avoid damaging the machinery. However, the project has the added difficulty that only data from correct operations is available, since generating real error data risks breaking down the machinery and it is unaffordable due to its cost.

Consequently, the project has two business requirements: 1) The number of phases that compose the industrial process must be detected automatically to identify how the gas turbine transits from one phase to another, and 2) for each phase, we must detect if the data tuple that is received from the sensors belongs to a normal working process or to an abnormal one.

The first step is to collect information through the requirements questionnaire in order to get a general vision of the problem and collect information to create the i* model. For simplicity, we include for each point the number and question from our requirements questionnaire along with the result for this particular case study.

5.1.1. Stage 1 - High-level goal definition

1. **From a qualitative point of view, which objectives do I have? Is a regression/clustering/classification model required?** The customer wants an ML model capable of detecting, without human intervention, if the turbine is working properly or not. And if the turbine has an anomalous working process, an alarm will be raised in the SCADA software for the domain expert.

5.1.2. Stage 2 - Definition of success metrics

2. **Which metric is the proper one to measure the planned goal? Which value of that metric would consider the project as success?** The accuracy of the ML model should be at least 0.85.

5.1.3. Stage 3 - Non-functional requirements

3. **Is the explainability of the model or the effect of every dimension on each prediction mandatory?** It is not necessary. Only an accuracy rate above the threshold is required.
4. **Can the behavior of data change as new instances are added, or is it static? If so, is continuous re-training necessary?** No, the information provided has all the different data distributions that the industrial process can follow. It is not expected to change.
5. **Is data biased? Is it necessary any additional preprocessing step with the aim of providing equity in data?** Data provided belongs only to correct tuples. Consequently, only a semi-supervised approach can be performed. Data equity is not required for this industrial process.
6. **Is there any other specific ML quality aspect that the ML model should have?** Yes, the model must be simple, and it should be scalable. The candidate algorithms must, at the very least, to have a neutral relationship with simplicity; and at most a “hurt” relationship with scalability.

5.1.4. Stage 4 - Dataset characteristics & constraints

7. **Should the system provide a prediction within a time limit?** The solution should provide information about the error at least 1 min before an incorrect working process. Moreover, the system should dampen false anomalies. A notification will be launched only when the turbine has an anomalous working process.
8. **Which data should be considered for the model? Must an agnostic approach be followed to deal with the problem, or is domain knowledge available?** All data retrieved by all the sensors is considered important. However, there are 2 redundant sensors for each measured value. Consequently, the system could be simplified by retrieving the mean for every physical value. This reduces a problem with a dimensionality of around 100 to a dimensionality of around 30.
9. **Which granularity of data is suitable for achieving the goal? Is the required granularity available?** Every sensor has its own working process for sending the data: some of them send a few signals every second and some others send the value after a few hours. Consequently, the values should have been interpolated from every signal, on a 10-sec interval. With that granularity, question 9 can be answered properly.
10. **Is external validation required, or is it enough with internal validation?** No more data is available. As a result of that, internal validation will be enough.

5.1.5. Stage 5 - Model validation

Once we have finished answering the questionnaire and built the model, the domain expert and the ML expert review it in order to identify potential inconsistencies and oversights. In this particular case, model validation arises the need to include additional datasets, as the dataset provided does not include enough information to test the classification model. We provide more details about the model and the dataset in the following subsection.

5.1.6. Resulting ML requirements model

After iterating through the questionnaire, we have obtained the corresponding ML requirements model. The model resulting from applying our approach is shown in Fig. 7:

The visual elements that have been used for representing our work have been taken from the i* language guide 2.0 [15]. Moreover, since they represent more specific lower-abstraction concepts from the ML domain, we have added a label corresponding to the type of element. Consequently, *MLQualityAspect* has the “cloud” figure with the QA specification; the different goals have the “oval” figure, with the specification of either *ClusG* or *ClassG* corresponding to a clustering goal or a classification goal respectively; the tasks notation uses the “hexagon” figure, with the notation of *ClusT* for clustering tasks, *ClassT* for classification tasks and DP for data preparation tasks; *Sources* use the “rectangle”, with two different specific classes marked as S (sources) and DS (datasets) respectively. Moreover, the same notation as in [15] has been used for contribution links, refinement links and actors (and their boundaries). Nevertheless, since the i* language guide 2.0 does not include any notation for indicators and evaluation links, we have used the common representations that appear in related works, such as [44].

As we can see in the figure, the highest-level goal is the *Automatic Anomalous Working Process Detection*. Due to data generation limitations, the data provided for training the models is related to a correct working process of the turbine only. Thus, a semi-supervised approach will be used, where one of the classes (anomalous operations) is unknown.

The main goal has been split into two different *MLGoals*. The first one is the *Automatic Detection of the number of phases* that compose the industrial process. This is considered a clustering goal (*ClusG*) since the objective is to group the data into clusters (phases) in order to identify the behavior of the process from the data available. The second one, *Correctness on each tuple* is a classification goal (*ClassG*) since it aims to determine whether each tuple is correct or anomalous given the current operations data.

Regarding the clustering goal, two quality aspects (*QA*) have been defined according to the requirements elicited from Question 6: *Scalability* and *Simplicity*. *Scalability* is a requirement since it is possible that the number of tuples sent on each second could vary in the future (this is common in an IoT environment) and the algorithm should be able to deal with larger volumes of data in a timely manner. Moreover, *Simplicity* is a desirable quality since a well-trained simple algorithm should be able to provide the suitable information and favors quicker training and evaluation.

The remaining ML quality aspects have been discarded. *NoiseRobustness* quality is not applicable since only correct data is available.

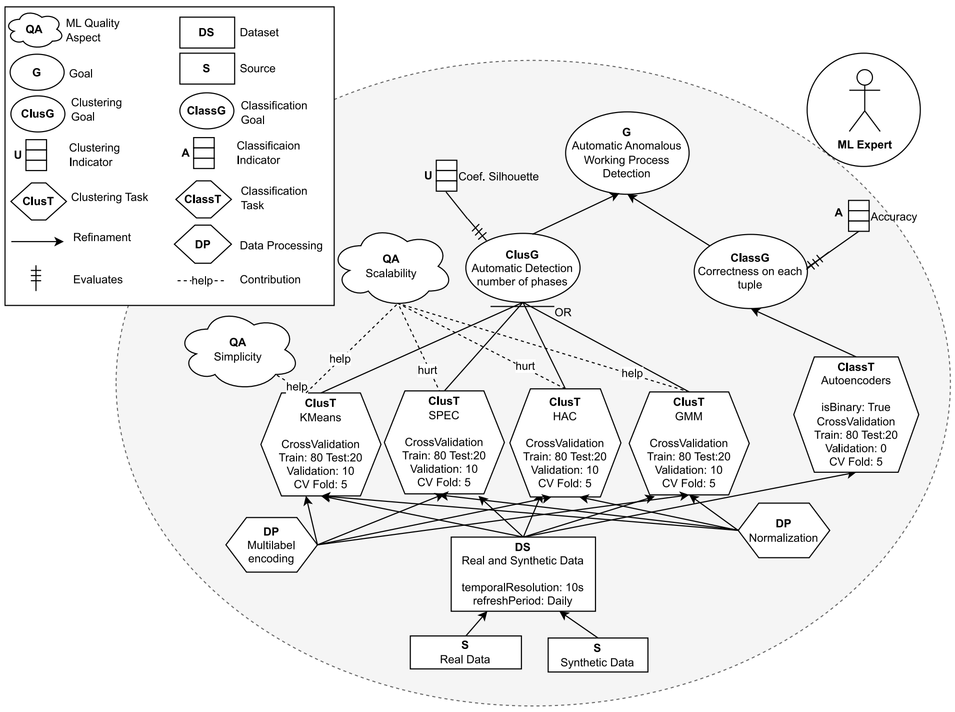
QA ML Quality Aspect
G Goal
ClusG Clustering Goal
U Clustering Indicator
ClusT Clustering Task
Refinement
Evaluates
DS Dataset
S Source
ClassG Classification Goal
A Classificaion Indicator
ClassT Classification Task
DP Data Processing
help Contribution
ML Expert
G Automatic Anomalous Working Process Detection
U Coef. Silhouette
A Accuracy
QA Scalability
QA Simplicity
ClusG Automatic Detection number of phases
ClassG Correctness on each tuple
OR
help
hurt
hurt
help
help
ClusT KMeans CrossValidation Train: 80 Test:20 Validation: 10 CV Fold: 5
ClusT SPEC CrossValidation Train: 80 Test:20 Validation: 10 CV Fold: 5
ClusT HAC CrossValidation Train: 80 Test:20 Validation: 10 CV Fold: 5
ClusT GMM CrossValidation Train: 80 Test:20 Validation: 10 CV Fold: 5
ClassT Autoencoders isBinary: True CrossValidation Train: 80 Test:20 Validation: 0 CV Fold: 5
DP Multilabel encoding
DP Normalization
DS Real and Synthetic Data temporalResolution: 10s refreshPeriod: Daily
S Real Data
S Synthetic Data

Fig. 7. Application of our ML approach to a case study 1 (industry).

Table 4
Weights of candidates algorithms for ML model in industrial application.

Algorithm	Quality Aspects				S_A
	Simplicity		Scalability		
	Da	W	Da	W	
Kmeans	0, 5	0, 5	1	0, 25	0, 5
HAC	0	0, 5	−0, 5	0, 25	−0, 125
GMM	0	0, 5	0, 5	0, 25	0, 125
Spectral Clustering	0	0, 5	−0, 5	0, 25	−0, 125

Moreover, the industrial process is fixed, and not expected to change. Consequently, supporting or detecting *Concept Drift* is unnecessary. Furthermore, the *Explainability* of the industrial process is not required in the project. In addition, *Equity* is not a problem since there are no sensitive-to-equity groups involved in the process, only the machinery. Finally, the number of sensors in the turbine is not expected to change significantly. Thus, given the current number of dimensions in the data, the *HighDimensionality* requirement is not considered. Regarding the different parameters of *MLTask* associated to clustering, the ML expert has opted for these parameters: a *TrainTestSplit*= 80–20 and a *CVFold*= 5 (related to the 80–20 partition). Finally, a *ValidationSize*= 10 has been used for hyperparameter tuning.

Since the ML quality aspects of the project are not very restrictive, being assisted by Table 2, 4 different algorithms are suitable candidates to carry out the project (they have been represented as clustering tasks, *ClusT*). These algorithms are Gaussian Mixture Model (GMM), Kmeans, Density-Based spatial clustering (DBSCAN) and Spectral Clustering (SPEC). In order to rank these algorithms, we apply Eq. (1) as shown in Table 4. For this use case, the domain expert has considered a weight of 1 for *Simplicity* and a weight of 0.5 for *Scalability*. Then, according to Table 2, the contributions to each quality aspect have been weighted, giving a final suitability score for each algorithm.

In this case, as we are very interested in the optimal clustering, the algorithm that provides the best silhouette coefficient will be used in the model, with the suitability score acting as the discriminator between algorithms that obtain similar results. Due to the fact that the algorithms are based on distances, the *Normalization* task has been linked with every clustering task. Moreover, a multilabel encoding will be performed on a variable, and consequently, this DP task will be a contribution to every clustering task.

Furthermore, for the classification goal *Correctness on each tuple*, only one classification task *ClassT* is available. As we are using an approach based on semi-supervised learning, the use of Autoencoders (a type of neural network) is suitable for this binary classification. Moreover, due to the heavy restrictions of data (we only have labeled one class), only this approach has been considered. The ML expert has opted for the same parameters for the classification task as in clustering tasks, with two exceptions: parameter *isBinary*= True, as it has been explained before (and it only belongs to classification tasks) and the *ValidationSize*= 0 due to the fact that Autoencoder uses its own method for the tuning of its elements. Thus, no specific data is needed for this part.

In addition, *Accuracy* has been chosen as the classification indicator (*ClassI*) for the aforementioned classification goal because we are using a binary classification (a tuple can be correct or not) related to the semi-supervised learning. Finally, each algorithm requires information provided by a *DataSet* element (*DS*). Data provided by the domain expert is real and belongs to a correct working process of the turbine. This data has been represented as a *Source* (*S*). Exploring the information provided by this source, we note that we are missing erroneous tuples in order to test whether our classifier identifies anomalies correctly. Therefore, we capture the need to provide an additional "erroneous" dataset. Tuples within this dataset should be classified as "erroneous" by the classification model. Due to the lack of erroneous real data, this dataset has been created synthetically.

5.2. Case study 2: Analysis of medical data for evolution prediction on patients affected by COVID-19 disease

In this section, we present the application of our work to another study. As previously, we will provide the answers to questions presented in Section 4. Then, we present the requirements model.

In this case study, we analyze retrospectively the data created in a set of hospitals between 27th February 2020 to 12th November 2020, regarding patients affected by COVID-19 disease. The Electronic Health Record (EHR) is formed by demographics, comorbidities tests, prescription medication and outpatient data. To summarize, the business requirement is related to the prediction of the evolution of patients. It must predict if a patient will die or not, according to her comorbidities. As before, for simplicity we present each stage indicating directly the question from the requirements questionnaire along with the results for this particular case study.

5.2.1. Stage 1 - High-level goal definition

1. **From a qualitative point of view, which objectives do I have? Is a regression/clustering/classification model required?** The project aims to develop a binary classification ML model that predicts if a patient will die or not, according to her comorbidities.

5.2.2. Stage 2 - Definition of success metrics

2. **Which metric is the proper one to measure the planned goal? Which value of that metric would consider the project as success?** Accuracy is not accepted as a valid metric in this project. Instead, the ML model must fulfill two metrics: precision, and sensitivity. The precision of the model should be at least 0.85. The sensitivity should be at least 0.80. Those metrics must be obtained with another hospital corpus as well.

5.2.3. Stage 3 - Non-functional requirements

3. **Is the explainability of the model or the effect of every dimension on each prediction mandatory?** It is absolutely mandatory. The model must provide what has been learned and which elements are the most relevant for the classification. It is not necessary to provide the effect of each dimension for each prediction, only the overall explainability of the model.
4. **Can the behavior of data change as new instances are added, or is it static? If so, is continuous re-training necessary?** No, the model will not be retrained. Other elements that could alter the data distribution (for instance, the vaccines) will have their own model prediction. Those ML models are not related to this project.
5. **Is data biased? Is it necessary any additional preprocessing step with the aim of providing equity in data?** Data provided has a great bias in survivors among deaths. As a result, proper metrics should be used for evaluating the model (precision and sensitivity). However, no additional preprocessing steps are required.
6. **Is there any other specific ML quality aspect that the ML model should have?** No, there are no other desirable ML quality aspects.

5.2.4. Stage 4 - Dataset characteristics & constraints

7. **Should the system provide a prediction within a time limit?** The prediction must be performed with a 7-day margin.
8. **Which data should be considered for the model? Must an agnostic approach be followed to deal with the problem, or is domain knowledge available?** An agnostic approach must not be followed. According to domain knowledge, for the death prediction model, age, sex, and two types of comorbidities (chronic comorbidities, and comorbidities arisen during COVID-19 period infection) are important.
9. **Which granularity of data is suitable for achieving the goal? Is the required granularity available?** Only the age has a numerical attribute and its minimum granularity is a year. This granularity is enough for the considered problem.
10. **Is external validation required, or is it enough with internal validation?** External validation is required with data coming from other hospitals. That data is available and provided by the stakeholders.

Table 5
Weights of candidates algorithms for ML model in e-health application.

Algorithm	Quality Aspects				S_A
	Simplicity		Explainability		
	Da	W	Da	W	
Logistic Regression	0, 5	1	1	4	4, 5
KNN	0, 5	1	0, 5	4	2, 5
NaiveBayes	0, 5	1	0	4	0, 5
RandomForest	0	1	0	4	0

5.2.5. Stage 5 - Model validation

Once we have finished answering the questionnaire and built the model, the domain expert and the ML expert review it in order to identify potential inconsistencies and oversights. In this particular case, the model validation step finds no inconsistencies and thus the model is validated. As previously, we present the resulting model in more detail in the following subsection.

5.2.6. Resulting ML requirements model

Once we have answered the set of questions to frame the problem, we proceed to build our requirements model. The model resulting from applying our approach is shown in Fig. 8.

As we can see in Fig. 8, the project is focused on a classification goal (Mortality risk prediction). Moreover, it has been imposed two different classification metrics for achieving the success on the project (*Precision* and *Sensitivity*)

Regarding the non-functional requirements, two quality aspects (*QA*) have been imposed as requirements: (i) *Simplicity*, due to the high dimensionality of the problem only simple algorithms will be used (and the quality aspect will be assigned a weight of (1), and (ii) *Explainability*, since the project is framed in the e-health field the explainability of the model is paramount, (thus resulting in the quality aspect having a weight of 4). Consequently, four algorithms have been selected as *ClassificationTasks* candidates: Logistic Regression, KNN, Naïve Bayes and Random Forest. The results of these algorithms can be seen in Table 5:

As an additional requirement, models must be trained and tested with internal data (data from other hospital) and additionally tested with external data (data from other hospital). Thus, the classification tasks are parametrized with a *CrossValidation* = 5, and *TrainTestSplit*= 80–20. Moreover, the tuning of hyperparameters will be realized with a specific external technique called PSO (Particle Swarm Optimization), with the aim of getting better Sensitivity and Precision metrics, at the cost of *Accuracy* metrics. Moreover, the external data will compose a pure test dataset, that must fulfill the specified *MLQualityAspects*.

Finally, it must be highlighted that there are two *DataProcessing* tasks that must be performed, depending on the algorithms used: *Multilabel* encoding for the codification of the diseases, and *Normalization* for the algorithms that require it. Thanks to our work, we obtain a personalized preprocessing pipeline ensuring that we cover the necessary data transformations even if these vary from one algorithm to the other.

6. Conclusions and future works

The latest advances and proliferation of Artificial Intelligence (AI) have forced us to focus on improving the development of Machine Learning (ML) systems. Succeeding in ML projects requires expertise in a wide variety of ML techniques as well as identifying project requirements and how they are translated into ML concepts. Consequently, the need to improve requirements capture in the field of ML arises in order to bridge the gap between the domain expert and the ML developer.

Therefore, in this paper, we have presented the first i* extension for capturing ML requirements. The main objective of this study is to provide a solid and coherent modeling language that captures ML requirements, including high abstraction concepts and also how these

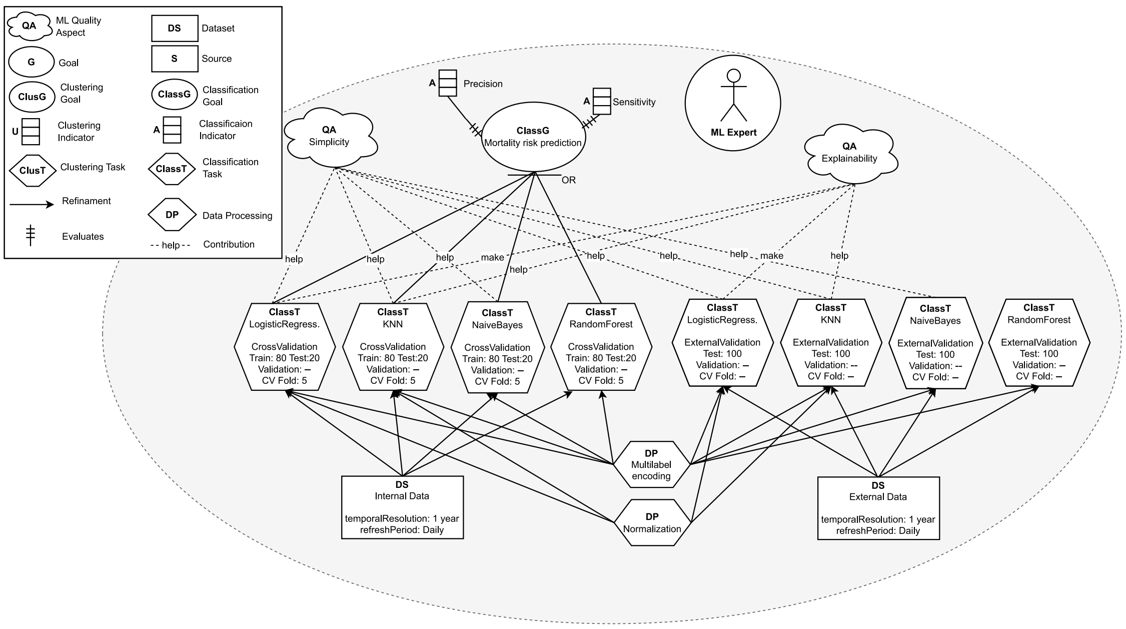
QA ML Quality Aspect
G Goal
ClusG Clustering Goal
ClassG Classification Goal
U Clustering Indicator
A Classificaion Indicator
ClusT Clustering Task
ClassT Classification Task
Refinament
DP Data Processing
Evaluates
help Contribution
DS Dataset
S Source
A Precision
A Sensitivity
ClassG Mortality risk prediction
OR
ML Expert
QA Simplicity
QA Explainability
help
make
ClassT LogisticRegress. CrossValidation Train: 80 Test:20 Validation: – CV Fold: 5
ClassT KNN CrossValidation Train: 80 Test:20 Validation: – CV Fold: 5
ClassT NaiveBayes CrossValidation Train: 80 Test:20 Validation: – CV Fold: 5
ClassT RandomForest CrossValidation Train: 80 Test:20 Validation: – CV Fold: 5
ClassT LogisticRegress. ExternalValidation Test: 100 Validation: – CV Fold: –
ClassT KNN ExternalValidation Test: 100 Validation: – CV Fold: –
ClassT NaiveBayes ExternalValidation Test: 100 Validation: – CV Fold: –
ClassT RandomForest ExternalValidation Test: 100 Validation: – CV Fold: –
DP Multilabel encoding
DP Normalization
DS Internal Data temporalResolution: 1 year refreshPeriod: Daily
DS External Data temporalResolution: 1 year refreshPeriod: Daily

Fig. 8. Application of our ML approach to a case study 2 (ehealth).

concepts are translated into explicit ML implementations. With the aim of creating a complete, consistent, and conflict-free i* extension, we have followed the steps outlined by PRISE (Process to Support iStar Extensions). The result has been the creation of a metamodel implemented in the ECORE (Eclipse) modeling tool that enables capturing ML requirements.

In order to capture the elements that compose our novel i* extension for ML requirements, we provide a guided process composed of a set of stages based on (i) a requirements questionnaire, (ii) a set of tables that facilitates the identification and selection of the most appropriate ML models and metrics according to the goals and priorities of the quality aspects and (iii) an equation to weight the suitability of each algorithm and establish a ranking among the most promising one. Each stage of our process helps the ML expert to iteratively build the model in a guided manner and ensure that all relevant aspects of the ML solution are covered.

Finally, have applied our modeling language in two real-world case studies. The first one had an industrial context, where we aimed to detect malfunctioning in electricity generation using gas turbines. The second one was related to the healthcare sector, where we tried to predict which patients risked dying according to their comorbidities before their health worsened. In both case studies, our work has significantly contributed to explore the project goals, identify key non-functional requirements, select the most adequate algorithms and analyze the available data sources with project goals in mind. Moreover, in our first case study, our work helped identifying missing data that was supplied through synthetic generation. In our second case study, it contributed to define specific preprocessing pipelines for each algorithm. Overall, the application of our work to different contexts was successful, demonstrating the flexibility of the modeling language and the overall approach.

Regarding the limitations, it is important to highlight that our study is focused on ML requirements and algorithms. More specifically, we have covered the most popular ML algorithms for classification, regression and clustering tasks. However, our tables would need to be expanded with additional algorithms and variants in order to cover all possible cases. Therefore, in the event that a team aims to implement an algorithm not included within our tables, they would need to first analyze the contributions of the algorithm to the different quality aspects.

Similarly, our approach can be partially ported as-is to other AI areas such as Deep Learning. In this case, however, both the questionnaire and tables would likely require to be expanded.

Finally, it is worth noting that we have not covered other aspects such as Reinforcement Learning or all the details involved in eXplainable AI (XAI).

Regarding future works, our first step is to carry out a controlled experiment with several users in order to better evaluate the impact of our study. We are also working on extending our work through the inclusion of new algorithms and specific elements related to deep learning. Finally, we plan to develop tools that support the framework, with the aim of facilitating requirements capture and implementation.

CRediT authorship contribution statement

Jose M. Barrera: Conceptualization, Methodology, Investigation, Software, Writing – original draft, Visualization. **Alejandro Reina-Reina:** Conceptualization, Methodology, Investigation, Software, Writing – original draft, Visualization. **Ana Lavalle:** Conceptualization, Methodology, Investigation, Writing – original draft, Writing – review & editing, Visualization. **Alejandro Maté:** Conceptualization, Methodology, Investigation, Resources, Writing – original draft, Writing – review & editing, Visualization, Supervision, Project administration, Funding acquisition. **Juan Trujillo:** Conceptualization, Methodology, Investigation, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Data availability

Data will be made available on request.

Acknowledgments

This work has been co-funded by the AETHER-UA project (PID2020-112540RB-C43), a smart data holistic approach for context-aware data analytics: smarter machine learning for business modeling and analytics, funded by the Spanish Ministry of Science and Innovation. And the BALLADEER (PROMETEO/2021/088) project, a Big Data analytical platform for the diagnosis and treatment of Attention Deficit Hyperactivity Disorder (ADHD) featuring extended reality, funded by the Conselleria de Innovación, Universidades, Ciencia y Sociedad Digital (Generalitat Valenciana). A. Reina-Reina (I-PI 13/20) hold Industrial PhD Grants co-funded by the University of Alicante and the Lucentia Lab Spin-off Company.

References

[1] E.D.S. Nascimento, I. Ahmed, E. Oliveira, M.P. Palheta, I. Steinmacher, T. Conte, Understanding development process of machine learning systems: Challenges and solutions, in: International Symposium on Empirical Software Engineering and Measurement, Vol. 2019-Septemer, IEEE Computer Society, 2019, http://dx.doi.org/10.1109/ESEM.2019.8870157.

[2] S. Fabi, T. Hagendorff, Why we need biased AI – how including cognitive and ethical machine biases can enhance AI systems, 2022, URL http://arxiv.org/abs/2203.09911.

[3] Z. Wan, X. Xia, D. Lo, G.C. Murphy, How does machine learning change software development practices? IEEE Trans. Softw. Eng. 47 (2021) 1857–1871, http://dx.doi.org/10.1109/TSE.2019.2937083.

[4] B. Berenbach, D.J. Paulish, J. Kazmeier, A. Rudorfer, Software & Systems Requirements Engineering: In Practice, McGraw-Hill, 2009, p. 321.

[5] E. Gonçalves, J. Castro, J. Araújo, T. Heineck, A systematic literature review of istar extensions, J. Syst. Softw. 137 (2018) 1–33, http://dx.doi.org/10.1016/j.jss.2017.11.023.

[6] P. Bresciani, A. Perini, P. Giorgini, F. Giunchiglia, J. Mylopoulos, Tropos: An agent-oriented software development methodology, Auton. Agents Multi-Agent Syst. 8 (3) (2004) 203–236, http://dx.doi.org/10.1023/B:AGNT.0000018806.20944.ef.

[7] S. Ghanavati, D. Amyot, A. Rifaut, Legal goal-oriented requirement language (legal GRL) for modeling regulations, in: Proceedings of the 6th International Workshop on Modeling in Software Engineering, 2014, pp. 1–6, http://dx.doi.org/10.1145/2593770.2593780.

[8] M. Ribeiro, J. Castro, J. Pimentel, Istar for safety-critical systems, 2019, URL http://www.cin.ufpe.br/~jhcp/pistar/4safety/.

[9] E. Gonçalves, J. Araujo, J. Castro, IStar4RationalAgents: Modeling requirements of multi-agent systems with rational agents, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 11788 LNCS, Springer Science and Business Media Deutschland GmbH, 2019, pp. 558–566, http://dx.doi.org/10.1007/978-3-030-33223-5_46/COVER/.

[10] A. Lavalle, A. Maté, J. Trujillo, S. Rizzi, Visualization requirements for business intelligence analytics: A goal-based, iterative framework, in: Proceedings of the IEEE International Conference on Requirements Engineering, Vol. 2019-Septe, IEEE Computer Society, 2019, pp. 109–119, http://dx.doi.org/10.1109/RE.2019.00022.

[11] Y. Asnar, P. Giorgini, J. Mylopoulos, Goal-driven risk assessment in requirements engineering, Requir. Eng. 16 (2) (2011) 101–116, http://dx.doi.org/10.1007/s00766-010-0112-x.

[12] E. Gonçalves, J. Araujo, J. Castro, PRISE: A process to support iStar extensions, J. Syst. Softw. 168 (2020) 110649, http://dx.doi.org/10.1016/j.jss.2020.110649.

[13] J. Horkoff, F.B. Aydemir, E. Cardoso, T. Li, A. Maté, E. Paja, M. Salnitri, L. Piras, J. Mylopoulos, P. Giorgini, Goal-oriented requirements engineering: an extended systematic mapping study, Requir. Eng. 24 (2) (2019) 133–160, http://dx.doi.org/10.1007/s00766-017-0280-z.

[14] E.S.K. Ỹu, Modeling Strategic Relationships for Process Reengineering, 1995, URL http://www.cs.utoronto.ca/pub/eric/DKBS-TR-94-6.pdf.

[15] F. Dalpiaz, X. Franch, J. Horkoff, iStar 2.0 language guide, 2016, arXiv:1605.07767, URL http://istarwiki.org, http://arxiv.org/abs/1605.07767.

[16] A. Lavalle, A. Maté, J. Trujillo, M.A. Teruel, S. Rizzi, A methodology to automatically translate user requirements into visualizations: Experimental validation, Inf. Softw. Technol. 136 (2021) 106592, http://dx.doi.org/10.1016/j.infsof.2021.106592.

[17] L.M. Tapia, J.P. Carvallo, iStar support to open innovation management, in: Proceedings of the 15th International iStar Workshop (iStar 2022) Co-Located with 41st International Conference on Conceptual Modeling (ER 2022), Virtual Event, Hyderabad, India, October 17, 2022, in: CEUR Workshop Proceedings, Vol. 3231, CEUR-WS.org, 2022, pp. 21–24, URL http://ceur-ws.org/Vol-3231/iStar22_paper_4.pdf.

[18] H. Singh, H. Khalajzadeh, S. Paktinat, U.M. Graetsch, J. Grundy, Modelling human-centric aspects of end-users with iStar, J. Comput. Lang. 68 (2022) 101091, http://dx.doi.org/10.1016/j.cola.2022.101091.

[19] S. Alwidian, Towards integrating human-centric characteristics into the goal-oriented requirements language, in: 2022 IEEE 30th International Requirements Engineering Conference Workshops (REW), IEEE, 2022, pp. 200–204, http://dx.doi.org/10.1109/REW56159.2022.00045.

[20] H. Xiong, Y. Wang, T. Li, BiStar: A template-based istar modeling tool combining graphical and textual modeling, in: 2022 IEEE 30th International Requirements Engineering Conference (RE), IEEE, 2022, pp. 260–261, http://dx.doi.org/10.1109/RE54965.2022.00034.

[21] E. Gonçalves, T. Heineck, J. Castro, J. Araújo, iStar extension repository, 2016, https://istarextensions.cin.ufpe.br/catalogue/, (Accessed: 20/11/2022).

[22] F. Ishikawa, Y. Matsuno, Evidence-driven requirements engineering for uncertainty of machine learning-based systems, in: Proceedings of the IEEE International Conference on Requirements Engineering, 2020-Augus, IEEE Computer Society, 2020, pp. 346–351, http://dx.doi.org/10.1109/RE48521.2020.00046.

[23] J. Horkoff, Non-functional requirements for machine learning: Challenges and new directions, in: Proceedings of the IEEE International Conference on Requirements Engineering, 2019-Septe, IEEE Computer Society, 2019, pp. 386–391, http://dx.doi.org/10.1109/RE.2019.00050.

[24] L.M. Cysneiros, J.C.S. do Prado Leite, Non-functional requirements orienting the development of socially responsible software, in: Lecture Notes in Business Information Processing, Vol. 387 LNBIP, Springer, 2020, pp. 335–342, http://dx.doi.org/10.1007/978-3-030-49418-6_23.

[25] H.-M. Heyn, E. Knauss, A.P. Muhammad, O. Eriksson, J. Linder, P. Subbiah, S.K. Pradhan, S. Tungal, Requirement engineering challenges for AI-intense systems development, 2021, URL http://arxiv.org/abs/2103.10270.

[26] E. Breck, S. Cai, E. Nielsen, M. Salib, D. Sculley, The ML test score: A rubric for ML production readiness and technical debt reduction, in: Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017, Vol. 2018-Janua, Institute of Electrical and Electronics Engineers Inc., 2017, pp. 1123–1132, http://dx.doi.org/10.1109/BigData.2017.8258038.

[27] S. Nalchigar, E. Yu, Y. Obeidi, S. Carbajales, J. Green, A. Chan, Solution patterns for machine learning, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 11483 LNCS, Springer Verlag, 2019, pp. 627–642, http://dx.doi.org/10.1007/978-3-030-21290-2_39.

[28] F. Dalpiaz, X. Franch, J. Horkoff, iStar 2.0 language guide, 2016, CoRR arXiv:1605.07767.

[29] A. Lavalle, M.A. Teruel, A. Maté, J. Trujillo, Improving sustainability of smart cities through visualization techniques for Big Data from iot devices, Sustainability (Switzerland) 12 (14) (2020) 5595, http://dx.doi.org/10.3390/su12145595.

[30] P. Refaeilzadeh, L. Tang, H. Liu, Cross-validation, in: Encyclopedia of Database Systems, Springer US, 2009, pp. 532–538, http://dx.doi.org/10.1007/978-0-387-39940-9_565.

[31] J. Cabot, M. Gogolla, Object constraint language (OCL): A definitive guide, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 7320 LNCS, 2012, pp. 58–90, http://dx.doi.org/10.1007/978-3-642-30982-3_3.

[32] J. Siebert, L. Joeckel, J. Heidrich, A. Trendowicz, K. Nakamichi, K. Ohashi, I. Namba, R. Yamamoto, M. Aoyama, Construction of a quality model for machine learning systems, Softw. Qual. J. (2021) 1–29, http://dx.doi.org/10.1007/s11219-021-09557-y.

[33] J. Amann, A. Blasimme, E. Vayena, D. Frey, V.I. Madai, Explainability for artificial intelligence in healthcare: a multidisciplinary perspective, BMC Med. Inform. Decis. Mak. 20 (1) (2020) 1–9, http://dx.doi.org/10.1186/s12911-020-01332-6.

[34] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nat. Mach. Intell. 1 (5) (2019) 206–215, http://dx.doi.org/10.1038/s42256-019-0048-x, arXiv:1811.10154.

[35] S.M. Lundberg, S.I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, Vol. 2017-Decem, 2017, pp. 4766–4775, arXiv:1705.07874, URL https://github.com/slundberg/shap.

[36] A. Bibal, M. Lognoul, A. de Streel, B. Frénay, Legal requirements on explainability in machine learning, Artif. Intell. Law 29 (2) (2021) 149–169, http://dx.doi.org/10.1007/s10506-020-09270-4.

[37] W. Samek, K.R. Müller, Towards explainable artificial intelligence, in: Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Vol. 11700 LNCS, Springer, Cham, 2019, pp. 5–22, http://dx.doi.org/10.1007/978-3-030-28954-6_1, arXiv:1909.12072.

[38] J. Zenisek, F. Holzinger, M. Affenzeller, Machine learning based concept drift detection for predictive maintenance, Comput. Ind. Eng. 137 (2019) 106031, http://dx.doi.org/10.1016/j.cie.2019.106031.

[39] T. Fahse, V. Huber, B. van Giffen, Managing Bias in Machine Learning Projects, Springer, Cham, 2021, pp. 94–109, http://dx.doi.org/10.1007/978-3-030-86797-3_7.

[40] F. Schrag, Review of Weapons of math destruction: How big data increases inequality and threatens democracy, Educ. Rev., Reseñas Educativas 24 (2017) http://dx.doi.org/10.14507/er.v24.2197.

[41] D. Berrar, P. Lopes, W. Dubitzky, Incorporating domain knowledge in machine learning for soccer outcome prediction, Mach. Learn. 108 (1) (2019) 97–126, http://dx.doi.org/10.1007/s10994-018-5747-8.

[42] T. Yu, S. Simoff, T. Jan, VQSVM: A case study for incorporating prior domain knowledge into inductive machine learning, Neurocomputing 73 (13–15) (2010) 2614–2623, http://dx.doi.org/10.1016/j.neucom.2010.05.007.

[43] C. Deng, X. Ji, C. Rainey, J. Zhang, W. Lu, Integrating machine learning with human knowledge, IScience 23 (11) (2020) 101656, http://dx.doi.org/10.1016/j.isci.2020.101656.

[44] S. Nalchigar, E. Yu, Designing business analytics solutions: A model-driven approach, Bus. Inf. Syst. Eng. 62 (1) (2020) 61–75, http://dx.doi.org/10.1007/s12599-018-0555-z.