Neurosymbolic Artificial Intelligence -1 (2024) 1–13
DOI 10.3233/NAI-240767
IOS Press
CORRECTED PROOF


# Machine learning with requirements: A manifesto

Eleonora Giunchiglia [a,*], Fergus Imrie [b], Mihaela van der Schaar [c,d] and Thomas Lukasiewicz [e,f]
[a] *Imperial-X, Department of Electrical and Electronic Engineering, Imperial College, United Kingdom*
*E-mail: e.giunchiglia@imperial.ac.uk*
[b] *Department of Electrical and Computer Engineering, University of California, Los Angeles, USA*
[c] *DAMTP, University of Cambridge, United Kingdom*
[d] *Alan Turing Institute, United Kingdom*
[e] *Institute of Logic and Computation, Vienna University of Technology, Austria*
[f] *Department of Computer Science, University of Oxford, United Kingdom*

**Editor:** Annette Ten Teije, Vrije Universiteit Amsterdam, The Netherlands
**Solicited reviews:** Floris van der Hengst, Vrije Universiteit Amsterdam, The Netherlands; two anonymous reviewers

Received 10 July 2023
Revised 24 June 2024
Accepted 8 July 2024

**Abstract.** In the recent years, machine learning has made great advancements that have been at the root of many breakthroughs in different application domains. However, it is still an open issue how to make them applicable to high-stakes or safety-critical application domains, as they can often be brittle and unreliable. In this paper, we argue that requirements definition and satisfaction can go a long way to make machine learning models even more fitting to the real world, especially in critical domains. To this end, we present two problems in which (i) requirements arise naturally, (ii) machine learning models are or can be fruitfully deployed, and (iii) neglecting the requirements can have dramatic consequences. Our proposed pyramid development process integrates requirements specification into every stage of the machine learning pipeline, ensuring mutual influence between requirements and subsequent phases. Additionally, we explore the pivotal role of Neuro-symbolic AI in facilitating this integration, paving the way for more reliable and robust machine learning applications in critical domains. Through this approach, we aim to bridge the gap between theoretical advancements and practical implementations, ensuring machine learning's safe and effective deployment in sensitive areas.

Keywords: Safe AI, machine learning, requirements, machine learning operations, software engineering

## 1. Introduction

In recent years, machine learning has made great advancements that have been at the root of many breakthroughs in different application domains. For example, AlphaFold [63] is a deep learning model that solved the "protein folding problem", a grand challenge in the field of biology for more than half a century, while Halicin [71] is the

*Corresponding author. E-mail: e.giunchiglia@imperial.ac.uk.

first antibiotic discovered using machine learning, which could help in the battle against bacterial resistance. Results like those above, though ground-breaking, overshadow the dangers that come with the careless use of machine learning models in critical applications. Indeed, even though such models report an astonishingly high performance in terms of accuracy (or alternatively chosen metric), they do not give any guarantee that the model will not have any unintended behaviour when used in practice. Indeed, as pointed out by Rudin in [59], there have been several notable instances of machine learning failures. For example, individuals have been wrongly denied parole [81], deep learning models have inaccurately reported that highly polluted air was safe to breathe [47], and there has been generally inefficient use of limited resources across fields such as medicine, criminal justice, and finance [76]. Such problems in the behavior of the model are often rooted in the quality of the dataset used for training the model. This is, for example, what happened in the early stages of the Covid-19 pandemic, where (as reported in [58]) a common chest scan dataset from [35] made of pediatric scans, was used as control group against Covid-19 positive scans: as a result, instead of Covid-19 detectors, adult/child classifiers were built. Unintended outcomes like the ones described above (i) are particularly dangerous in safety-critical applications, where even a single unforeseen mistake can lead to dramatic consequences, and (ii) undermine human confidence in the models themselves, thus slowing down their adoption.

In this paper, we argue that requirements specification and verification can go a long way to make machine learning models even more fitting to the real world, by reducing the risk of getting potentially dangerous unintended behaviors. We start from the simple observation that unintended behaviors correspond to the model violating some requirements which may be known even before the data collection and model development start. To support this claim, we consider two examples, one in healthcare, and the other one in autonomous driving, in which (i) some requirements are known in advance, (ii) machine learning models have very high performance in terms of accuracy (or other selected metric), and (iii) despite the positive performance, the outcome of standardly developed machine learning models often violates the known requirements with possible dramatic consequences given the criticality of the application domains. Although we consider just two examples, we believe that analogous considerations apply to most application domains given (i) the body of knowledge developed over the years in many application domains, which can be translated in corresponding requirements (see, e.g., [22,33,50,70]), (ii) the impressive results in performance obtained by machine learning models, which will continue to push for their adoption despite the possible unintended behaviors (see, e.g., [5,24]), and (iii) the fact that it is not surprising that the resulting model will violate the requirements if they are not taken into account during data collection and model development. Refining the last observation, it is not surprising that a machine learning model has an unintended behavior if, before its deployment, the data and the model itself are not somehow verified against the requirement capturing the intended behavior. We therefore claim that the requirements' definition should precede and involve the entire machine learning model development cycle, including the dataset construction. In particular, we propose a novel *pyramid* machine learning development process, which (i) highlights how the requirement definition can help improve every single step of the standard machine learning model development process, and (ii) takes into account that some requirement adjustment might be necessary because of difficulties and/or discoveries that emerge during the development process of machine learning models, especially given their data-driven nature. In line with what has been argued in the Machine Learning Operations (MLOps) field [7,25,32,36,74], the above proposal can be seen as a call to adapt and adopt the general methodologies used in software engineering for generic system development, where it is well known (i) that the later the requirement are taken into account the higher the cost is to repair the system, and also (ii) that the requirement definition is an iterative process in which requirements affect the system development and vice versa, given that during the system development some requirement may be newly discovered and/or adjusted if not canceled (see, e.g., [52,67,68]). We will ground the above facts in the specific machine learning model development pipeline, highlighting the benefits of adopting requirements in the different phases.

Our proposal represents a significant shift from the traditional *"performance-driven"* machine learning development pipeline, which solely concentrates on how to get better performance, measured in terms of accuracy or alternatively chosen metric. Indeed, we are proposing a *"requirements-driven"* machine learning development pipeline in which performance is just one of the many requirements that the model has to satisfy. This obviously might make the development process more complex, because different requirements might be contradictory with one another: a problem that is well known in software engineering [68]. For example, at a high level of abstraction, complex models will likely have a better performance at the possible price of being less explainable and/or verifiable and/or

sustainable. Analogously, it might be the case that satisfying certain fairness properties (e.g., equalized odds or demographic parity) is preferable to a higher performance. Independently from the properties that we wish for the model, it is clear that (i) the sooner the requirements are made explicit the better, and (ii) taking into account the requirements in the dataset definition and model construction will likely lead to models that will be easier to verify with respect to the stated requirements. This proposal thus aligns with the efforts in many other machine learning fields (see, e.g., fairness, robustness, explainability, sustainability, and safety), where it is argued that performance should not be the only factor to evaluate a model. However, in the aforementioned fields, there is often either no formal or unique definition of what is the desired property (see, e.g., interpretable) and, even when there is, its incorporation in the machine learning pipeline requires a customised loss/architecture that requires a lot of time and knowledge to design. Here, on the other hand, we will take a step further and we will claim that the requirements will ideally need to be formally expressed. This will be pivotal to their wide applicability in the machine learning domain. If the requirements were to be formally defined, then we could not only assign a unique meaning to each requirement, but also create automatic procedures to incorporate such requirements into the machine learning development pipeline. To this end, the field of Neuro-symbolic AI is well-positioned to lead this research. Researchers in this field possess the expertise to (i) encode background knowledge and desiderata formally, (ii) understand the inner workings of machine learning models, and (iii) automatically integrate such requirements into the topology and loss functions of various models. For example, in recent years researchers in the Neuro-symbolic AI field have developed models that are guaranteed-by-design to be compliant with a set of given requirements expressed as logical constraints (see, e.g., [3,21,22,28]): in these works, it has been shown how the requirements can be incorporated in the model, which automatically satisfies them, and have a positive impact also on performance.

Given the above, our proposal enjoys the same spirit of works such as [19,49,61], which advocate for a more structured approach to machine learning model development. In particular, Gebru et al. in [19] propose documentation guidelines for new datasets, Mitchell et al. in [49] advocate for standardized reporting of models, including training data, performance measures, and limitations, while Seedat et al. [61] propose an actionable checklist-style framework to elicit data-centric considerations at different stages of the development pipeline.

The remainder of the paper is structured as follows. First, in Section 2, we consider examples from the healthcare and autonomous driving application domains in order to show that in these domains (i) requirements arise naturally, (ii) machine learning models are or can be fruitfully deployed, and (iii) neglecting the requirements can have dramatic consequences. Secondly, in Section 3, we show how the requirements definition can be fruitfully integrated into the standard machine learning development pipeline, impacting all the phases in the pipeline. Finally, we have the conclusions and possible plans for the road ahead in Section 5.

## 2. The need for requirements

The standard machine learning pipeline naturally tends to have a sequential nature. The main steps that compose it are:

1. **Data curation**, in which the dataset for training the model is created, encompassing the following phases: (i) data collection, (ii) data pre-processing, (iii) data augmentation, and (iv) data quality evaluation;
2. **Model Creation**, in which the model is designed and built;
3. **Model Training**, in which the created model is trained and in which the right hyperparameters are chosen (often through testing over a validation dataset);
4. **Model Testing**, in which the performance of the model is assessed (ideally using different metrics and test sets).
5. **Model Deployment**, in which the model is actually deployed in the real world.

A visual representation of the standard sequential pipeline is given in Fig. 1. Even though we do not show them in the figure, in practice loop-backs might be necessary in order to modify the dataset or, more often, the model. Still, all such possible changes happen mostly between the model creation step and the model training step, as they are usually driven by the desire to get a better performance, while the dataset is often considered a static element of the pipeline. As we can see from the depicted pipeline, requirements do not appear anywhere. This


Fig. 1. Visualization of the standard machine learning pipeline.

is even more disconcerting when we compare such development pipelines with the standard software engineering pipelines, where requirements play a central role at every step of the process – no matter how small the project is. The most surprising fact though is that the development pipelines that have been proposed to deal with big projects in big companies either do not consider requirements at all (see, e.g., [6,12]) or when they do (see, e.g., [46,72]) they only mention data-centric requirements (e.g., the training dataset should have at least a certain number of data points or the model should achieve a certain accuracy) or requirements that cannot be formally specified (e.g., the model should be robust and/or interpretable). If formal requirements are considered at all, then they are normally considered just at the very end of the pipeline, during the model testing phase, where the model gets verified and/or tested against a set of properties [54,55]. As we can imagine, acting only at the very end of the pipeline can be very costly, as one might have to re-start a project if the requirements are not satisfied. For this reason, our development pipeline will consider the requirements from the very beginning and they will effect every step of the process.

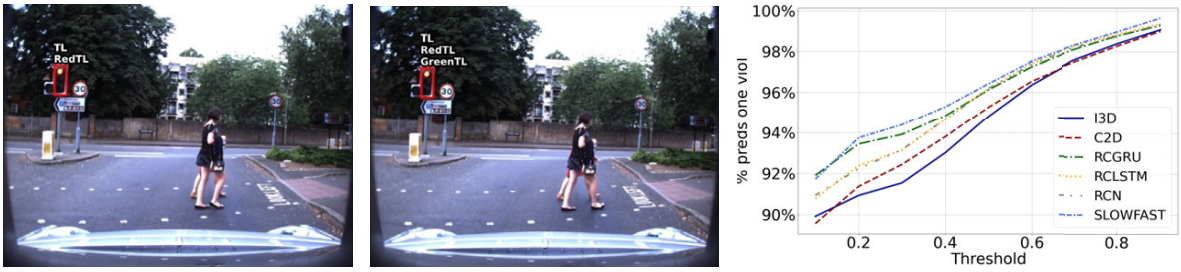
We now focus on two application domains, healthcare and autonomous driving, and we show how requirements arise naturally in both applications domains, how applying machine learning techniques has already brought and can bring tremendous advantages to the fields, and how deploying machine learning models in these fields without explicitly taking into account the requirements can lead to unexpected behaviors corresponding to violations of the requirements. Though we consider just two application domains, we believe that analogous considerations and results hold virtually in any application domain, given (i) the body of knowledge developed over the years in any application domain, which can be translated in corresponding requirements, (ii) the impressive results in performance obtained by machine learning models which will continue to push for their adoption despite the possible unintended behaviors, and (iii) the impossibility to certify/rule out the absence of undesired behaviors without explicitly spelling out the corresponding requirements and the testing/verification of the model against the requirements themselves.

### *2.1. Autonomous driving*

In recent years, the developments in machine learning, and in particular in computer vision, have fuelled the hopes of autonomous vehicles being finally in reach. However, every so often, such dreams are shattered by car crashes that, in some cases, have injured or even killed people. For example, in March 2018, an autonomous vehicle developed and tested on public roads by Uber's Advanced Technologies Group fatally injured a pedestrian who was pushing their bicycle across the street outside of a designated crossing area [51]. The most striking characteristic of this accident is the fact that the car did not make any attempt to break and/or to avoid the pedestrian [40]. Unfortunately, this is not an isolated incident, as nearly 400 car crashes involving autonomous vehicles have been reported in the United States over a period of only ten months in 2022 [9]. As reported in [9], these vehicles in order to take their decisions rely, among others, on computer vision models, which if trained to simply maximise their performance (following the standard machine learning development pipeline) might fail to abide to even the simplest requirements.

To exemplify the problem described above, we consider the ROAD-R dataset [22], which consists of (i) 22 relatively long (∼8 minutes each) videos annotated with road events, i.e., a sequence of frame-wise bounding boxes linked in time, each labelled with the agent in the bounding box, together with its action(s) and location(s), and (ii) 243 requirements expressed in propositional logic stating what is an admissible road event.[1] The possible labels associated to each bounding box are 41, and the requirements state which combinations of labels a model can output. Hence, for instance, a requirement states that a traffic light cannot be red and green at the same, while another states that a traffic light cannot move. Given ROAD-R, six state-of-the-art temporal feature learning architectures (I3D [10], C2D [79], RCGRU [29], RCLSTM [29], RCN [66] and SlowFast [16]) as part of a 3D-RetinaNet model

[1] Dataset available at: https://github.com/gurkirt/road-dataset. Requirements available at: https://github.com/EGiunchiglia/ROAD-R.




(a) (b) (c)

Fig. 2. (a) and (b) show the predictions made by the I3D model (with $\theta = 0.5$) for the same traffic light and just one frame apart. (c) shows the number of predictions that violate at least one requirement when varying $\theta$; (c) is from [22].

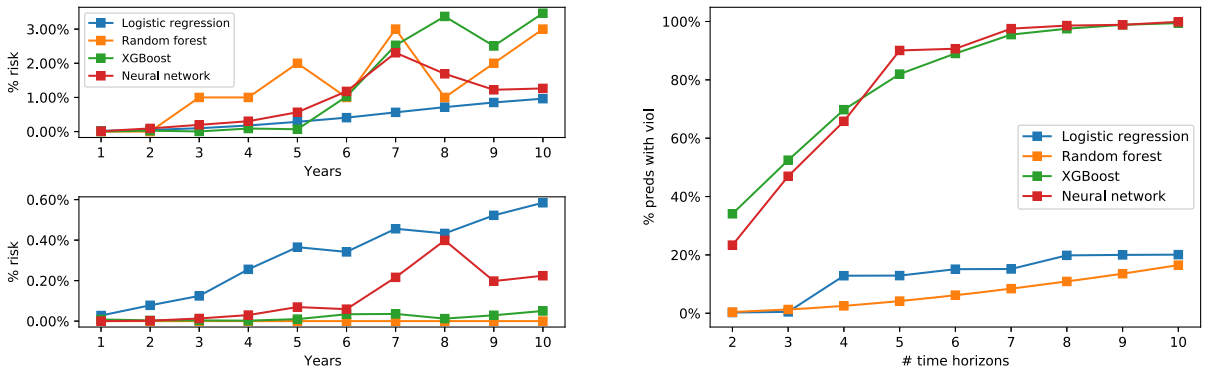
[65] (with a 2D-ConvNet backbone made of Resnet50 [26]) for event detection, have been trained. These models take as input a set of consecutive frames, and for each frame they output (i) a set of bounding boxes, and (ii) a set of labels for each bounding box. Such labels are decided in the standard way: for each of the 41 labels the model outputs a number $o \in [0, 1]$, and if $o > \theta$ then the label is returned, otherwise it is not. $\theta$ is a user-defined threshold. An example of prediction is given in Fig. 2a, where the prediction for the depicted bounding box is {Traffic Light, Red Traffic Light}. The surprising finding of the work [22] is that, as shown in Fig. 2c, no matter the chosen threshold $\theta$, at least 89% of the predictions violate at least one requirement. Even more, some of the predictions may violate requirements corresponding to common knowledge possessed by humans, making the prediction difficult to interpret and manage. For example, if we consider the prediction in Fig. 2b, done by the same model that made the prediction in Fig. 2a, for the same traffic light just one frame later, then we see that not only the prediction is wrong, but it also violates the common knowledge (in this case corresponding also to a formally stated requirement) that the traffic light cannot be red and green at the same time. Such prediction, if not further elaborated by the system controlling the vehicle could indeed have dramatic consequences. For this reason, we expect that any system controlling the vehicle will have mechanisms in place to handle the predictions that violate the requirements. However, if the requirements are incorporated into and verified by the entire system (as must be the case in this setting), it is far from clear why they are standardly neglected in the process of building the machine learning component of the system. Indeed, none of the six evaluated systems is able to handle requirements on their inputs and/or outputs. Even more, as reported in [22], the dataset used for training the model violates some of the requirements (like the fact that it is not possible for a vehicle to be both incoming and outgoing), which surely will have to be incorporated into any real application of the models.[2]

### 2.2. *Healthcare*

For our second example, we discuss healthcare. There is great hope that developments in machine learning will revolutionize medicine and transform clinical practice [75]. The range of applications in medicine is vast, from computer vision systems analyzing medical images in radiology and longitudinal monitoring of patient trajectories throughout a hospital stay, to genomic screening of future disease risk and much more. However, as with many other high-stakes and safety-critical applications of machine learning, there are a number of requirements for machine learning systems in healthcare. These include ethical considerations, such as fairness and bias; practical considerations, such as controlling false positive rate to prevent alarm fatigue [62] and ensuring appropriate resource allocation; and logical considerations, such as the example discussed below, among others.

As a concrete example, we consider clinical risk scores. Clinical risk scores estimate the likelihood of a specific outcome occurring after a certain period of time, such as a patient developing a particular disease or condition in the next ten years or experiencing an adverse event following a medical procedure. By definition, the chance of

[2]While it might be argued that it is normal to have errors in the dataset because of noise, it seems odd to train a model with known-to-be-wrong-data and then correct the model outputs when they respect the known-to-be-wrong-data.




(a) Example predictions at different time horizons.

(b) Proportion of patients with $\geqslant 1$ violation.

Fig. 3. Fig. 3a shows two examples of the predictions made by different classification models for the same patient at different time horizons. Figure 3b shows the proportion of samples for which the predictions violate the requirement for risk to be increasing for longer time horizons.

an outcome occurring within some time horizon $t$ must increase with time, thus risk scores must be monotonically increasing functions of time, i.e., the predicted $n$-year risk is greater than their $m$-year risk, for all $n > m$. However, there are many studies that use classification models to predict risk after a specific time horizon, for example, 5-year risk. Naturally, we can generalize this by predicting risk at multiple time horizons rather than just one. Often this is very useful clinically, since patient trajectories can vary greatly and patients with equivalent long-term risks might have divergent short-term risks. A straightforward way to achieve this is to train multiple classification models to predict the risk at each time point of interest. However, there is no guarantee that the predicted risk will increase for longer time horizons.

To illustrate the consequence of not incorporating the simple requirement described above into the machine learning systems, we consider the problem of predicting the risk of developing diabetes. We construct a cohort of patients from UK Biobank [73], a large-scale observational study with around 500,000 participants from 22 assessment centers across England, Wales, and Scotland enrolled between 2006 and 2010. We extracted a cohort of participants who were 40 years of age or older at enrollment with no diagnosis or history of diabetes at baseline. We considered the 18 features employed by QDiabetes [27]. We performed data imputation using HyperImpute [31] and trained classification algorithms using AutoPrognosis [30] to predict $n$-year risk of developing diabetes for $n \in \{1, \ldots, 10\}$. We averaged results over five random initializations. Two examples are provided in Fig. 3a. For the first patient (Fig. 3a, top), only the logistic regression models met the requirement that the risk of developing diabetes should be monotonically increasing over longer time horizons, with the random forest, XGBoost, and neural network models having at least one time horizon with lower predicted risk than the previous. A similar situation can be seen for the second patient (Fig. 3a, bottom), where only the random forest model satisfies the requirement. Considering the entire dataset and all ten time horizons, around 20% of patients have predictions that violate the requirement for risk to monotonically increase at least once for the logistic regression and random forest models, while for over 99% of patients XGBoost and neural networks issued predictions that violated the requirement. Note that the number of violations made by the models was not necessarily correlated with performance, with the lowest-performing model as measured by area under the receiver operating curve at the 5-year horizon exhibiting the fewest violations (random forests). Such predictions, at best, are simply inaccurate but, at worst, can erode trust in machine learning systems and have more serious consequences depending on the actions taken. There are several solutions to satisfy this particular requirement, such as specialized functional forms [13,37], multi-task learning with specialized loss functions [38], or bespoke architectures. However, the fact that even for such a specific requirement we need to study and develop specialized loss functions and architectures clearly highlights the need for automatic requirements parsing and incorporation procedures.

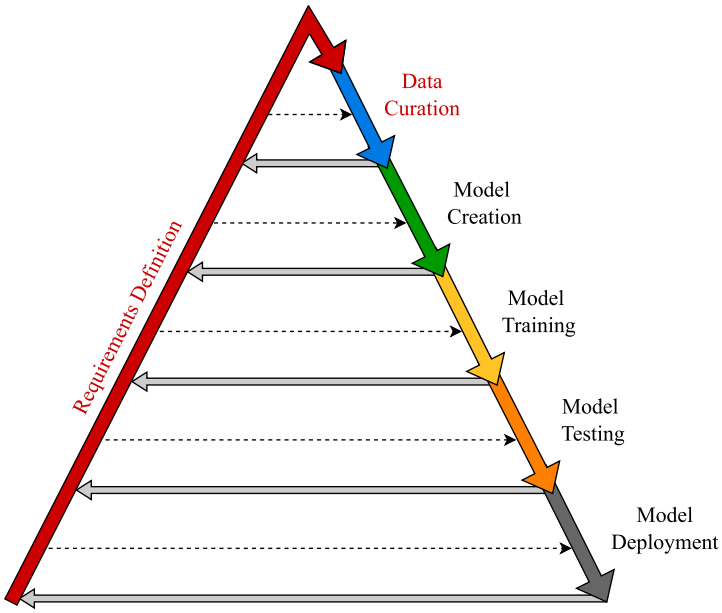



Fig. 4. Visualization of the pyramid model. The full arrows stand for the standard procedural processes, while the dotted arrows show that the requirements impact every stage of the process.

## 3. The pyramid model

In the previous section, we have shown the standard performance-driven machine learning development pipeline, in which the requirements are traditionally neglected, and, with the means of two examples, we have highlighted how such negligence can have dire consequences. This automatically calls for the inclusion of the requirements in the development process, which are at the core of any development pipeline proposed in the field of software engineering. We thus propose to adapt the general methodologies used in software engineering for generic system development, where it is well known that the later the requirements are taken into account the higher the cost is to repair the system. In order to adapt such methodologies, we need to take into account that, differently from standard software, machine learning models are learnt from data, and thus we have less control over the behaviour of the model, which entails not only that at design time it is impossible to predict the future behaviour of the model, but also, at testing time, if a bug is found, then the steps necessary to fix it are entirely different. A development pipeline for a machine learning model thus needs to take into account that:

1. data are central to the process, as the quality of the model heavily depends on the quality of the collected data,
2. the model is learnt from data, and thus it is not possible to fully know a priori its behaviour, and
3. if the model does not behave as expected, it is advisable to not only check the model but also check the data.

Further, it is unrealistic to expect that all requirements are always known a priori and that they remain immutable throughout the development process. This is already known in the general software engineering field, and it has been already specifically tested to be the case also in our field. Indeed, as highlighted by the survey conducted in [41], some of the biggest problems that machine learning practitioners face on a daily basis include the accessibility of data (which might not be known in the requirements definition phase) and the unrealistic expectations of the stakeholders of the system (which might reveal themselves during the testing phase of the model). Thus, we also need to take into account that the requirements might change at each stage of the development process.

Given the above, we propose the *pyramid machine learning development model* in Fig. 4, in which the requirement definition phase is (as expected) put at the beginning of the development process and in which the pyramid shape allows to better highlight the close relationship that must exist between the requirement definition phase and all the others in the pipeline. Ideally, the requirements should be annotated in a formal way, to allow for automatic (i)

parsing, (ii) integration in every step of the pipeline, and (iii) testing and verification. Consider the grey backward lines. These represent the fact that if something unexpected happens at any point of the pipeline (e.g., the model does have some unintended behavior during the testing phase), then it is necessary to go back and first check if the defined requirements were suitable or need to be updated, then if the collected data are representative of the phenomenon that we want to capture, and so on. Obviously, the later in the pipeline we realise that there is a bug, the more costly it is, and this is captured in Fig. 4 by the length of the path to cover as we advance in the pipeline. Consider now the dotted arrows, which illustrate the fact that the requirements can help in shaping each of the development steps, and in particular:

1. **Data Curation**: the requirements on the data obviously define the data collection phase, as they state which properties the dataset should have. This can positively impact this phase as it forces the stakeholders of the system to carefully reflect on which aspects of the total population the dataset should capture and which should be ignored. Furthermore, they can help reduce mistakes in the annotation phase (as any annotation that is not compliant with the requirements has to be corrected) and they can even accelerate it (as the annotators can be shown only those options that are compliant with the requirements, or they can be automatically corrected in real-time).
2. **Model Creation**: the requirements over the model define how the model should behave, and thus they should be taken into account at creation time. Following the same principles used for the data curation step, having requirements can positively impact this phase, because it forces the stakeholders of the system to reflect upon its expected behaviour and which outcomes must be avoided at all times. Such requirements can then be mapped in the model itself. As an example, given a set of formal requirements, some works [4,21,28] in the neuro-symbolic field were able to map the requirements directly to the topology of neural networks and guarantee that the constraints are always satisfied, while exploiting the background knowledge expressed by the requirements to even get better performance.
3. **Model Training**: the requirements over the model can again be used to define the objectives of our training. This is what it is done for example in many works in the fairness field, where the model is trained with two objectives: one to maximise performance and one to maximise the desired fairness definition (see, e.g., [1]). Furthermore, if we map the requirements to the loss function, we can even use them to alleviate the need for labelled data in semi-supervised settings (see, e.g., [8,14,83]). Indeed, given an unlabelled data point, the model can be taught to simply return an output which is compliant with the requirements.
4. **Model Testing**: as done in standard software engineering, the requirements should guide the testing phase. Each requirement should be checked to hold (eventually using formal verification techniques; see, e.g., [39, 55]) and in case it does not, then either the requirement has to be further analyzed and eventually modified or some procedure has to be put in place in order to signal that the requirement has to be properly handled outside of the machine learning model.
5. **Model Deployment**: requirements can also shape the deployment phase, especially if the stakeholders want the model to retain certain properties even in the presence of data shift and/or drift.

## 4. The role of neuro-symbolic AI

As already stated in the introduction, researchers in the Neuro-symbolic AI field find themselves in the unique position of being able to not only understand the deep impact that the violation of the requirements can have on the trust the public poses in the AI models, but also leverage the techniques developed in the traditional AI field to fill the gaps existing in the current state-of-the-art machine learning models.

As a testament of this fact, they have already been at the forefront of this line of research, with researchers developing methods which are guaranteed to be compliant with a set of requirements over the output space expressed as a logic program [42,43], in propositional logic [4,23], as linear inequalities [18,70] and even as quantifier free linear arithmetic formula over the rationals [28]. If we think about it, this was already an incredible leap forward, as up to that point specialised methods were being developed for different types of requirements. As an example, we can think of the field of *hierarchical multi-label classification*, where the labels are organised in a hierarchy

and every prediction had to be compliant with such hierarchy. Many different models (see, e.g., [60,64,77]) were developed to deal with these very simple constraints. If the requirement were to be changed even in the slightest, then those methods would no longer be applicable. On the contrary, thanks to Neuro-symbolic AI we can now build models compliant by-design with any set of requirements which can be expressed with a pre-defined syntax (e.g., propositional logic).

We expect that in the future the expressivity of the requirements studied will increase and will thus encompass more scenarios. For example, requirements expressing the relations existing over multiple data points might be expressed in first order logic (FOL) or requirements over the behaviour of an auto-regressive model might be expressed in linear temporal logic over finite traces ($\text{LTL}_f$) [20]. Then, thanks to Neuro-symbolic AI, we will not have to study each requirement singularly, but we will be able to develop novel models able to deal with entire classes of requirements altogether. Note, that a step in this direction has already been made, with Neuro-symbolic AI models able to incorporate requirements expressed in FOL [8] and to deal with auto-regressive models [2]. Despite the fact that these models cannot guarantee the satisfaction of the constraints, they denote the pro-activeness of Neuro-symbolic AI researchers to tackle more and more complex problems. Further, Neuro-symbolic AI could even be used to accelerate the phase of requirements elicitation, by using automatic methods able to discover constraints directly from the data (see, e.g., [53,56]) thus making the integration of the requirements in any pipeline even easier.

Finally, Neuro-symbolic AI has already shown very promising results in dealing with even more complex cases than the standard fully-supervised ones. For example, many works have been developed to be able to exploit (and sometimes impose) the requirements in semi-supervised settings (see, e.g., [8]), in partial label settings (see, e.g., [17]) and sometimes even in fully unsupervised settings (see, e.g., [69]). Naturally, these scenarios pose numerous challenges, including the exploitation of reasoning shortcuts by Neuro-symbolic AI models [45]; however, there is already research underway to mitigate this issue [44,78].

## 5. Summary and outlook

In this paper, we have shown that even though applying machine learning to a given problem can bring great benefits, applying it without specifying and taking into account the requirements of the problem can lead to unexpected behaviors and, possibly, dramatic consequences. We have thus proposed a new machine learning development pipeline in which the requirements definition phase is explicitly incorporated at the beginning of the process, causing a deep connection between the requirements definition and all the other phases in the development process, in which (as standard in software engineering) the former may affect the latter and vice versa. From this perspective, this paper can be seen as a call to adopt the general methodologies used in software engineering for generic system development to the specific field of machine learning, highlighting the risks of not doing so. We believe that in many cases the benefits of explicitly defining the requirements and taking them into account in the other phases outweigh their cost. It is also clear that the requirement definition and exploitation is unavoidable when machine learning models are used as stand-alone systems in domains with strict requirements to satisfy (e.g., in safety critical domains) or when used as part of larger systems with a given set of stringent requirements (e.g., memory usage; see, e.g., [57]). This view is supported by the recent trends in machine learning in which performance is just one of the requirements to satisfy. In particular, work in the fields of interpretability (see, e.g., [11,15],Imrie2023multiple), fairness (see, e.g., [48,82]) and robustness (see, e.g., [34,80]) all privilege certain characteristics of the model over performance, and all these works roots their motivation in the necessity to take into account requirements emerging from the respective application domain. However, how to define and fully exploit the domain knowledge expressed by requirements in the development process is still largely an open question, as many technical challenges need to be overcome. To this end, the Neuro-symbolic AI field can be seen as a bridge between standard machine learning and software engineering, as researchers in the field have already started developing techniques to create models that are built not only by the data, but that take explicitly as input also other types of knowledge, like formally stated requirements and use them either to design the model (see, e.g., [21,28]), or to incorporate the requirements in the loss function (see, e.g., [8,14]).

Despite the huge amount of work that can be categorized under the umbrella of "requirement-driven" machine learning, this is the first paper which explicitly advocates for a general model development methodology in which

the requirement definition is a first class citizen like the other phases in the pipeline. We believe that this corresponds to a more structured approach to machine learning model development, which – as a general principle – has already been advocated in [19,49,61].

## References


[1] A. Agarwal, A. Beygelzimer, M. Dudík, J. Langford and H.M. Wallach, A reductions approach to fair classification, in: *Proc. of ICML, PMLR*, 2018, pp. 60–69.

[2] K. Ahmed, K. Chang and G.V. den Broeck, A pseudo-semantic loss for autoregressive models with logical constraints, in: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, New Orleans, LA, USA, December 10–16, 2023, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt and S. Levine, eds, 2023.

[3] K. Ahmed, S. Teso, K. Chang, G.V. den Broeck and A. Vergari, Semantic probabilistic layers for neuro-symbolic learning, in: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, New Orleans, LA, USA, November 28–December 9, 2022, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, eds, 2022, http://papers.nips.cc/paper_files/paper/2022/hash/c182ec594f38926b7fcb827635b9a8f4-Abstract-Conference.html.

[4] K. Ahmed, S. Teso, K. Chang, G.V. den Broeck and A. Vergari, Semantic probabilistic layers for neuro-symbolic learning, in: *Proc. of NeurIPS*, 2022.

[5] R. Ambadipudi, How Machine Learning Will Transform Your Industry, 2023, https://www.forbes.com/sites/forbestechcouncil/2023/02/27/how-machine-learning-will-transform-your-industry/.

[6] S. Amershi, A. Begel, C. Bird, R. DeLine, H.C. Gall, E. Kamar, N. Nagappan, B. Nushi and T. Zimmermann, Software engineering for machine learning: A case study, in: *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP)*, Montreal, QC, Canada, May 25–31, 2019, H. Sharp and M. Whalen, eds, IEEE/ACM, 2019, pp. 291–300.

[7] R. Ashmore, R. Calinescu and C. Paterson, Assuring the machine learning lifecycle: Desiderata, methods, and challenges, *ACM Comput. Surv.* **54**(5) (2022), 111:1–111:39. doi:10.1145/3453444.

[8] S. Badreddine, A. d'Avila Garcez, L. Serafini and M. Spranger, Logic Tensor Networks, *Artif. Intell.* **303** (2022). doi:10.1016/j.artint.2021.103649.

[9] N.E. Boudette, C. Metz and J. Ewing, *Tesla Autopilot and Other Driver-Assist Systems Linked to Hundreds of Crashes, the New York Times*, 2022, https://www.nytimes.com/2022/06/15/business/self-driving-car-nhtsa-crash-data.html.

[10] J. Carreira and A. Zisserman, Quo vadis, action recognition? A new model and the kinetics dataset, in: *Proc. of CVPR*, 2017.

[11] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin and J. Su, This looks like that: Deep learning for interpretable image recognition, in: *Proc. of NeurIPS*, H.M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E.B. Fox and R. Garnett, eds, 2019, pp. 8928–8939.

[12] I.B.M. Corporation, Agile Software Unified Process (ASUM), 2016, https://public.dhe.ibm.com/software/data/sw-library/services/ASUM.pdf.

[13] D.R. Cox, Regression models and life-tables, *Journal of the Royal Statistical Society: Series B (Methodological)* **34**(2) (1972), 187–202. doi:10.1111/j.2517-6161.1972.tb00899.x.

[14] M. Diligenti, M. Gori, M. Maggini and L. Rigutini, Bridging logic and kernel machines, *Mach. Learn.* **86** (2012). doi:10.1007/s10994-011-5243-x.

[15] F. Doshi-Velez and B. Kim, *Towards a Rigorous Science of Interpretable Machine Learning*, 2017.

[16] C. Feichtenhofer, H. Fan, J. Malik and K. He, SlowFast networks for video recognition, in: *Proc. of ICCV*, 2019.

[17] J. Feldstein, M. Jurcius and E. Tsamoura, Parallel neurosymbolic integration with Concordia, in: *International Conference on Machine Learning, ICML 2023*, 23–29 July 2023, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato and J. Scarlett, eds, Proceedings of Machine Learning Research, Vol. 202, PMLR, Honolulu, Hawaii, USA, 2023, pp. 9870–9885, https://proceedings.mlr.press/v202/feldstein23a.html.

[18] A.M. Ferber, B. Wilder, B. Dilkina and M. Tambe, MIPaaL: Mixed integer program as a layer, in: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, the Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, the Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, New York, NY, USA, February 7–12, 2020, AAAI Press, 2020, pp. 1504–1511.

[19] T. Gebru, J. Morgenstern, B. Vecchione, J.W. Vaughan, H.M. Wallach, H. Daumé III and K. Crawford, Datasheets for Datasets, 2018, CoRR, http://arxiv.org/abs/1803.09010 arXiv:1803.09010.

[20] G.D. Giacomo and M.Y. Vardi, Linear temporal logic and linear dynamic logic on finite traces, in: *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI/AAAI*, Beijing, China, August 3–9, 2013, F. Rossi, ed., 2013, pp. 854–860, http://www.aaai.org/ocs/index.php/IJCAI/IJCAI13/paper/view/6997.

[21] E. Giunchiglia and T. Lukasiewicz, Multi-Label Classification Neural Networks with Hard Logical Constraints, *JAIR* **72** (2021).

[22] E. Giunchiglia, M.C. Stoian, S. Khan, F. Cuzzolin and T. Lukasiewicz, ROAD-R: the Autonomous Driving Dataset for Learning with Requirements, *Machine Learning Journal* (2023).

[23] E. Giunchiglia, A. Tatomir, M.C. Stoian and T. Lukasiewicz, CCN+: A neuro-symbolic framework for deep learning with requirements, *International Journal of Approximate Reasoning* **171** (2024), 109124. doi:10.1016/j.ijar.2024.109124.

[24] F. Global, *The Rise of AI in Manufacturing: Streamlining Processes Through Machine Learning*, 2023, https://fintech.global/2023/06/02/the-rise-of-ai-in-manufacturing-streamlining-processes-through-machine-learning/.

[25] M. Haakman, L. Cruz, H. Huijgens and A. van Deursen, AI lifecycle models need to be revised, *Empir. Softw. Eng.* **26**(5) (2021), 95. doi:10.1007/s10664-021-09993-1.
[26] K. He, X. Zhang, S. Ren and J. Sun, Deep residual learning for image recognition, in: *Proc. of CVPR*, 2016.
[27] J. Hippisley-Cox and C. Coupland, Development and validation of QDiabetes-2018 risk prediction algorithm to estimate future risk of type 2 diabetes: cohort study, *BMJ* **359** (2017). doi:10.1136/bmj.j5019.
[28] N. Hoernle, R. Karampatsis, V. Belle and K. Gal, MultiplexNet: Towards fully satisfied logical constraints in neural networks, in: *Proc. of AAAI*, 2022.
[29] Y. Hua, Z. Zhao, Z. Liu, X. Chen, R. Li and H. Zhang, Traffic prediction based on random connectivity in deep learning with long short-term memory, in: *Proc. of VTC-Fall*, 2018.
[30] F. Imrie, B. Cebere, E.F. McKinney and M. van der Schaar, AutoPrognosis 2.0: Democratizing diagnostic and prognostic modeling in healthcare with automated machine learning, *PLOS Digit. Health* **2**(6) (2023), e0000276. doi:10.1371/journal.pdig.0000276.
[31] D. Jarrett, B.C. Cebere, T. Liu, A. Curth and M. van der Schaar, HyperImpute: Generalized iterative imputation with automatic model selection, in: *Proceedings of Machine Learning Research*, Vol. 162, PMLR, 2022, pp. 9916–9937.
[32] M.M. John, H.H. Olsson and J. Bosch, Towards MLOps: A framework and maturity model, in: *Proc. of SEAA*, IEEE, 2021, pp. 1–8.
[33] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang and L. Yang, Physics-informed machine learning, *Nature Reviews Physics* **3**(6) (2021), 422–440. doi:10.1038/s42254-021-00314-5.
[34] G. Katz, C.W. Barrett, D.L. Dill, K. Julian and M.J. Kochenderfer, Towards proving the adversarial robustness of deep neural networks, in: *Proceedings First Workshop on Formal Verification of Autonomous Vehicles, FVAV@iFM 2017*, Turin, Italy, 19th September 2017, L. Bulwahn, M. Kamali and S. Linker, eds, EPTCS, Vol. 257, 2017, pp. 19–26. doi:10.4204/EPTCS.257.3.
[35] D. Kermany, M. Goldbaum, W. Cai, C. Valentim, H.-Y. Liang, S. Baxter, A. McKeown, G. Yang, X. Wu, F. Yan, J. Dong, M. Prasadha, J. Pei, M. Ting, J. Zhu, C. Li, S. Hewett, J. Dong, I. Ziyar and K. Zhang, Identifying medical diagnoses and treatable diseases by image-based deep learning, *Cell* **172** (2018), 1122–1131.e9. doi:10.1016/j.cell.2018.02.010.
[36] D. Kreuzberger, N. Kühl and S. Hirschl, Machine learning operations (MLOps): Overview, definition, and architecture, *IEEE Access* **11** (2023), 31866–31879. doi:10.1109/ACCESS.2023.3262138.
[37] C. Lee, W. Zame, J. Yoon and M. Van Der Schaar, DeepHit: A deep learning approach to survival analysis with competing risks, in: *Proc. of AAAI*, Vol. 32, 2018.
[38] Y. Li, J. Wang, J. Ye and C.K. Reddy, A multi-task learning formulation for survival analysis, in: *Proc. of ACM SIGKDD*, 2016, pp. 1715–1724.
[39] A. Lomuscio and L. Maganti, An approach to reachability analysis for feed-forward ReLU neural networks, 2017, CoRR, http://arxiv.org/abs/1706.07351 arXiv:1706.07351.
[40] C. Macrae, Learning from the Failure of Autonomous and Intelligent Systems: Accidents, Safety, and Sociotechnical Sources of Risk, *Risk analysis: an official publication of the Society for Risk Analysis* **42** (2022).
[41] S. Mäkinen, H. Skogström, E. Laaksonen and T. Mikkonen, Who needs MLOps: What data scientists seek to accomplish and how can MLOps help? in: *1st IEEE/ACM Workshop on AI Engineering – Software Engineering for AI*, IEEE, 2021, pp. 109–112.
[42] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester and L.D. Raedt, DeepProbLog: Neural Probabilistic Logic Programming, 2018, CoRR, http://arxiv.org/abs/1805.10872 arXiv:1805.10872.
[43] R. Manhaeve, S. Dumancic, A. Kimmig, T. Demeester and L.D. Raedt, Neural probabilistic logic programming in DeepProbLog, *Artif. Intell.* **298** (2021), 103504. doi:10.1016/j.artint.2021.103504.
[44] E. Marconato, S. Bortolotti, E. van Krieken, A. Vergari, A. Passerini and S. Teso, BEARS Make Neuro-Symbolic Models Aware of their Reasoning Shortcuts, 2024, CoRR arXiv:2402.12240.
[45] E. Marconato, S. Teso, A. Vergari and A. Passerini, Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts, in: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, New Orleans, LA, USA, December 10–16, 2023, 2023.
[46] F. Martínez-Plumed, L.C. Ochando, C. Ferri, J. Hernández-Orallo, M. Kull, N. Lachiche, M.J. Ramírez-Quintana and P.A. Flach, CRISP-DM twenty years later: From data mining processes to data science trajectories, *IEEE Trans. Knowl. Data Eng.* **33**(8) (2021), 3048–3061. doi:10.1109/TKDE.2019.2962680.
[47] M. McGough, How bad is Sacramento's air, exactly? Google results appear at odds with reality, some say, *Sacramento Bee* (2018), https://www.sacbee.com/news/california/fires/article216227775.html.
[48] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman and A. Galstyan, A survey on bias and fairness in machine learning, *ACM Comput. Surv.* **54**(6) (2022), 115:1–115:35. doi:10.1145/3457607.
[49] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I.D. Raji and T. Gebru, Model cards for model reporting, in: *Proc. of FAT**, D. Boyd and J.H. Morgenstern, eds, ACM, 2019, pp. 220–229.
[50] F.K. Nakano, M. Lietaert and C. Vens, Machine learning for discovering missing or wrong protein function annotations – a comparison using updated benchmark datasets, *BMC Bioinform.* **20**(1) (2019), 485:1–485:32.
[51] NTSB, Preliminary report: Highway HWY18MH010, 2018, Washington, DC: National Transportation Safety Board.
[52] B. Nuseibeh and S.M. Easterbrook, Requirements engineering: A roadmap, in: *Proc. of ICSE*, A. Finkelstein, ed., ACM, 2000, pp. 35–46.
[53] A. Paulus, M. Rolínek, V. Musil, B. Amos and G. Martius, CombOptNet: Fit the right NP-hard problem by learning integer programming constraints, in: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, Virtual Event*, 18–24 July 2021, M. Meila and T. Zhang, eds, Proceedings of Machine Learning Research, Vol. 139, PMLR, 2021, pp. 8443–8453, http://proceedings.mlr.press/v139/paulus21a.html.

[54] K. Pei, Y. Cao, J. Yang and S. Jana, DeepXplore: Automated whitebox testing of deep learning systems, *Commun. ACM* **62**(11) (2019), 137–145. doi:10.1145/3361566.

[55] L. Pulina and A. Tacchella, An abstraction-refinement approach to verification of artificial neural networks, in: *Computer Aided Verification, 22nd International Conference, CAV 2010*, Edinburgh, UK, July 15–19, 2010, T. Touili, B. Cook and P.B. Jackson, eds, Lecture Notes in Computer Science, Vol. 6174, Springer, 2010, pp. 243–257. doi:10.1007/978-3-642-14295-6_24.

[56] L.D. Raedt, A. Passerini and S. Teso, Learning constraints from examples, in: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th Innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*, New Orleans, Louisiana, USA, February 2–7, 2018, S.A. McIlraith and K.Q. Weinberger, eds, AAAI Press, 2018, pp. 7965–7970.

[57] M. Rhu, N. Gimelshein, J. Clemons, A. Zulfiqar and S.W. Keckler, vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design, in: *Proc. of IEEE/ACM MICRO*, IEEE Computer Society, 2016, pp. 18:1–18:13.

[58] M. Roberts, D. Driggs, M. Thorpe, J. Gilbey, M. Yeung, S. Ursprung, A.I. Aviles-Rivero, C. Etmann, C. McCague, L. Beer, J.R. Weir-McCall, Z. Teng, E. Gkrania-Klotsas, A. Ruggiero, A. Korhonen, E. Jefferson, E. Ako, G. Langs, G. Gozaliasl, G. Yang, H. Prosch, J. Preller, J. Stanczuk, J. Tang, J. Hofmanninger, J. Babar, L.E. Sánchez, M. Thillai, P.M. Gonzalez, P. Teare, X. Zhu, M. Patel, C. Cafolla, H. Azadbakht, J. Jacob, J. Lowe, K. Zhang, K. Bradley, M. Wassin, M. Holzer, K. Ji, M.D. Ortet, T. Ai, N. Walton, P. Lio, S. Stranks, T. Shadbahr, W. Lin, Y. Zha, Z. Niu, J.H.F. Rudd, E. Sala, C.-B. Schönlieb and AIX-COVNET, Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans, *Nature Machine Intelligence* **3**(3) (2021), 199–217. doi:10.1038/s42256-021-00307-0.

[59] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, *Nature Machine Intelligence* **1**(5) (2019). doi:10.1038/s42256-019-0048-x.

[60] L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev and S. Dzeroski, Predicting gene function using hierarchical multi-label decision tree ensembles, *BMC Bioinform.* **11** (2010), 2. doi:10.1186/1471-2105-11-2.

[61] N. Seedat, F. Imrie and M. van der Schaar, DC-Check: A Data-Centric AI checklist to guide the development of reliable machine learning systems, 2022, CoRR arXiv:2211.05764. doi:10.48550/arXiv.2211.05764.

[62] S. Sendelbach and M. Funk, Alarm fatigue: A patient safety concern, *AACN Advanced Critical Care* **24**(4) (2013), 378–386. doi:10.4037/NCI.0b013e3182a903f9.

[63] A.W. Senior et al., Improved protein structure prediction using potentials from deep learning, *Nature* **577**(7792) (2020). ISBN 1476-4687.

[64] C.N.J. Silla and A.A. Freitas, A survey of hierarchical classification across different application domains, *Data Min. Knowl. Discov.* **22**(1–2) (2011), 31–72. doi:10.1007/s10618-010-0175-9.

[65] G. Singh, S. Akrigg, M.D. Maio, V. Fontana, R.J. Alitappeh, S. Saha, K.J. Saravi, F. Yousefi, J. Culley, T. Nicholson, J. Omokeowa, S. Khan, S. Grazioso, A. Bradley, G.D. Gironimo and F. Cuzzolin, ROAD: The ROad event Awareness Dataset for Autonomous Driving, *IEEE TPAMI* (2022).

[66] G. Singh and F. Cuzzolin, Recurrent convolutions for causal 3D-CNNs, in: *Proc. of ICCV Workshops*, 2019.

[67] I. Sommerville, *Software Engineering*, 10th edn, International Computer Science Series, Addison-Wesley, 2015.

[68] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, Wiley, 1997.

[69] R. Stewart and S. Ermon, Label-free supervision of neural networks with physics and domain knowledge, in: *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, February 4–9, 2017, S. Singh and S. Markovitch, eds, AAAI Press, 2017, pp. 2576–2582. doi:10.1609/AAAI.V31I1.10934.

[70] M.C. Stoian, S. Dyrmishi, M. Cordy, T. Lukasiewicz and E. Giunchiglia, How realistic is your synthetic data? Constraining deep generative models for tabular data, in: *Proceedings of ICLR*, 2024.

[71] J.M. Stokes et al., A Deep Learning Approach to Antibiotic Discovery, *Cell* **180**(4) (2020). doi:10.1016/j.cell.2020.01.021.

[72] S. Studer, T.B. Bui, C. Drescher, A. Hanuschkin, L. Winkler, S. Peters and K. Müller, Towards CRISP-ML(Q): A machine learning process model with quality assurance methodology, *Mach. Learn. Knowl. Extr.* **3**(2) (2021), 392–413. doi:10.3390/make3020020.

[73] C. Sudlow, J. Gallacher, N. Allen, V. Beral, P. Burton, J. Danesh, P. Downey, P. Elliott, J. Green, M. Landray, B. Liu, P. Matthews, G. Ong, J. Pell, A. Silman, A. Young, T. Sprosen, T. Peakman and R. Collins, UK biobank: An open access resource for identifying the causes of a wide range of complex diseases of middle and old age, *PLOS Medicine* **12**(3) (2015), 1–10. doi:10.1371/journal.pmed.1001779.

[74] G. Symeonidis, E. Nerantzis, A. Kazakis and G.A. Papakostas, MLOps – definitions, tools and challenges, in: *Proc. of IEEE CCWC*, IEEE, 2022, pp. 453–460.

[75] E.J. Topol, High-performance medicine: The convergence of human and artificial intelligence, *Nature Medicine* **25**(1) (2019), 44–56. doi:10.1038/s41591-018-0300-7.

[76] K.R. Varshney and H. Alemzadeh, *On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products, Big Data 5(3)*, 2016.

[77] C. Vens, J. Struyf, L. Schietgat, S. Dzeroski and H. Blockeel, Decision trees for hierarchical multi-label classification, *Mach. Learn.* **73**(2) (2008), 185–214. doi:10.1007/s10994-008-5077-3.

[78] K. Wang, E. Tsamoura and D. Roth, On learning latent models with multi-instance weak supervision, in: *Proceedings of NeurIPS*, 2023.

[79] X. Wang, R. Girshick, A. Gupta and K. He, Non-local neural networks, in: *Proc. of CVPR*, 2018.

[80] T. Weng, H. Zhang, P. Chen, J. Yi, D. Su, Y. Gao, C. Hsieh and L. Daniel, Evaluating the robustness of neural networks: An extreme value theory approach, in: *Proc. of ICLR*, OpenReview.net, 2018.

[81] R. Wexler, When a computer program keeps you in jail: How computers are harming criminal justice, *The New York Times* (2017), https://www.nytimes.com/2017/06/13/opinion/how-computers-are-harming-criminal-justice.html.

[82] M. Wick, S. Panda and J.-B. Tristan, Unlocking fairness: A trade-off revisited, in: *Proc. of NeurIPS*, Curran Associates, Vol. 32, Inc., 2019.

[83] J. Xu, Z. Zhang, T. Friedman, Y. Liang and G. Van den Broeck, A semantic loss function for deep learning with symbolic knowledge, in: *Proc. of ICML*, 2018.