Non-Functional Requirements for Machine Learning: Challenges and New Directions

Jennifer Horkoff
Chalmers and the University of Gothenburg
jennifer.horkoff@cse.gu.se

***Abstract*—Machine Learning (ML) provides approaches which use big data to enable algorithms to "learn", producing outputs which would be difficult to obtain otherwise. Despite the advances allowed by ML, much recent attention has been paid to certain qualities of ML solutions, particularly fairness and transparency, but also qualities such as privacy, security, and testability. From a requirements engineering (RE) perspective, such qualities are also known as non-functional requirements (NFRs). In RE, the meaning of certain NFRs, how to refine those NFRs, and how to use NFRs for design and runtime decision making over traditional software is relatively well established and understood. However, in a context where the solution involves ML, much of our knowledge about NFRs no longer applies. First, the types of NFRs we are concerned with undergo a shift: NFRs like fairness and transparency become prominent, whereas other NFRs such as modularity may become less relevant. The meanings and interpretations of NFRs in an ML context (e.g., maintainability, interoperability, and usability) must be rethought, including how these qualities are decomposed into sub-qualities. Trade-offs between NFRs in an ML context must be re-examined. Beyond the changing landscape of NFRs, we can ask if our known approaches to understanding, formalizing, modeling, and reasoning over NFRs at design and runtime must also be adjusted, or can be applied as-is to this new area? Given these questions, this work outlines challenges and a proposed research agenda for the exploration of NFRs for ML-based solutions.**

***Index Terms*—Non-Functional Requirements, NFRs, qualities, Machine Learning, Requirements Engineering**

I. Introduction

Machine Learning (ML) describes a computational approach which uses large amounts of data to enable algorithms to "learn", performing tasks which are difficult to achieve via standard software. This enables solving difficult problems such as recognizing images, diagnosing cancer, and estimating insurance [1]. Despite the advances allowed by ML, recently much attention has been paid to certain *qualities* of ML solutions, particularly fairness [2], but also transparency [3], security [4], privacy [5], and testability [6].

Much existing work has been devoted to understanding, decomposing, managing, formalizing and reasoning over qualities of typical non-ML software. Such qualities are often included as part of non-functional requirements (NFRs). These include well-studied NFRs such as performance, reliability, maintainability, and usability, but also security, privacy, and customer satisfaction [7], [8].

From the perspective of traditional software, the meaning of certain qualities, how to refine those qualities, and how to use such qualities for design and runtime reasoning is relatively well understood. However, when the software solution involves ML, some of our knowledge about NFRs may no longer apply. Fundamentally, the way in which we 'design', 'run', and 'maintain' ML-based solutions differs. The broad question of how SE methods and procedures can be adapted for ML-based solution development is already starting to be considered in venues such as the SEMLConf [9]. Here we focus particularly on methods for NFRs.

In particular, the nature of ML means that the meaning of many NFRs for ML solutions differs compared to regular software, and these NFRs are often not well understood (e.g., what is fairness? [10]). What does it mean for an ML-enabled system to be maintainable? Are NFRs such as compatibility and modularity still relevant? Some NFRs may have reduced importance for ML solutions compared to typical software. On the other hand, NFRs such as fairness [2] and transparency [3] have become critical from an ML perspective, whereas previous NFR work has not typically emphasized these dimensions. Further, as-yet-unexplored NFRs such as "retrainability" may also become relevant.

The complexity of NFRs has long been managed by refinement, e.g., security is typically refined to confidentiality, integrity, etc. Not only may the meaning of certain NFRs change in an ML context, but the refinements may also need to be rethought and updated. In typical NFR research, we are aware of common quality trade-offs, often called conflicting NFRs [7], e.g., security and performance. But recent work is only just beginning to explore quality trade-offs in the ML space [2]. Do known trade-offs still apply in the case of ML? Do new trade-offs exist?

In a traditional system, one can collect and implement many functional requirements (FRs). The overall function or purpose of an ML application is much more focused; e.g., recognize a face or diagnose a disease. Thus, there are far fewer FRs, and ML research has focused on the NFRs associated directly with those key FRs, e.g., accuracy of facial recognition, performance of diagnosis. Because an ML application has few FRs, one can argue that the effective satisfaction of NFRs becomes particularly critical. However, in practice, ML implementations will be integrated with more standard software as part of larger and more complex systems (e.g., in a self-driving car), which, as a whole, will have many complex FRs and NFRs.

In this paper, we consider whether traditional knowledge about NFRs and quality from a requirements perspective can apply to ML-based systems. We can view this knowledge

2332-6441/19/$31.00 ©2019 IEEE
DOI 10.1109/RE.2019.00050

from two dimensions: 1) knowledge of NFRs, i.e., what are common and important NFRs, how are they interpreted, refined, measured, and how they can conflict, and 2) methods for NFRs, i.e., catalogues of NFRs [8], modeling methods like the NFR framework [7], methods to reason over NFRs, e.g., [11], [12], to use NFRs to monitor software, e.g., [13], and drive software adaptation and evolution, e.g., [14], [15], etc. In this work, we make the argument that the first dimension (1) must be at least partially re-thought in light of the rise of ML. Many of the ideas and techniques considering NFRs for traditional software (2) may still be valid, but it is possible that techniques may need revamping in light of this new paradigm, or that new, completely novel techniques are needed.

The next section outlines the state-of-the-art, followed by an illustrative and motivating ML example. Research challenges and an agenda are outlined, followed by a discussion and consideration of future work.

II. State-of-the-Art

A. NFRs in Requirements Engineering

Requirements Engineering (RE) research has long made the argument that eliciting and considering NFRs is critical for the success of systems [7]. Such systems could be technically sound, but fail due to issues in quality. Such an argument is particularly relevant for ML solutions, whose effectiveness lies mainly in the quality of the outcomes they provide.

What is an NFR? Simply, an NFR is any quality or attribute which is non-functional. This broad definition, defining something critical in terms of what it is not, is not ideal, as has been discussed by several authors, e.g. [16], [17]. Our purpose here is not to define NFRs in a satisfactory way, but to explore their application to ML. The concept of quality has had better luck in terms of a precise definition, being covered by several prominent ontologies, e.g., DOLCE [18]. More recent work in RE uses ideas from DOLCE to treat NFRs as qualities over an entity [19], usually a functional requirement, the system, or a system component, e.g., "send mail (entity) quickly (quality)" or "the system (entity) should be secure (quality)".

Although qualities of ML solutions and NFRs for ML solutions are similar, technically one can think of NFRs as the requirements over the quality, e.g., the quality is usability of system X, while the NFR is "System X must be usable", which ideally should be defined in a measurable way, e.g., "90% of test users would rate the system as an 8/10 in terms of usability". Although there is a distinction, for simplicity, in this work we treat NFRs and qualities as synonyms.

NFR Catalogues. To facilitate a consideration of NFRs, catalogues of software qualities were created. For example, the ISO/IEC 25010 standard divides system/software product quality into eight categories, including performance efficiency, compatibility, usability, and security [8]. Each quality is further decomposed; e.g., compatibility is refined into co-existence and interoperabilty. Such catalogues provide iterative refinement of NFRs into sub-qualities, possibly sub-sub-qualities, sometimes down to measurable indicators, when possible.

Languages and Reasoning. Much work focuses on capturing NFRs in visual modeling languages, sometimes with an underlying metamodel and semantics, facilitating (semi-) automated qualitative and quantitative methods to support decision making, e.g., [7], [11], [12]. Usually, approaches allow users to use NFRs to select among possible alternative functional requirements, e.g., given FRs and NFRs, many of which are in conflict, which requirements should we implement?

Runtime, Adaptation, and Evolution. NFR approaches were extended to consider a requirements-based view of runtime system operation, where functional and quality requirements could be monitored at runtime, based on data from the running system, e.g., [13], [20]. Work in this area went further to consider requirements-based runtime adaptation, e.g., a certain quality aspect is not sufficiently satisfied at runtime, thus the system will evolve and adjust to try to gain better performance or quality, all while considering quality trade-offs [14], [15].

Linking Data to Quality. A related line of work uses an adaption of common requirements notations to link business data to organizational goals, including qualities [21], allowing for continuous goal-based business intelligence. More recent work focuses on the design of data analytic systems for business, which may include ML algorithms [22], [23]. This work focuses on finding designs which fit domain-specific analytic questions, considering aspects of quality performance for various ML options. In this case, the authors adapt existing RE languages to consider data analytics at the syntax level (no formal semantics or metamodel), and they use existing analysis procedures without modification.

B. Qualities for Machine Learning.

ML encompasses over a dozen algorithm types (e.g., Regression, Bayesian, Instance-based, Deep Learning, Neural Networks), with many more specific algorithms (e.g., Logistic Regression, Linear Regression, Naïve Bayes, Nearest Neighbor) [1]. Most work on ML topics provide examples and algorithm details, including performance results, but do not focus on a wide range of NFRs or quality aspects. We summarize a selection of current work considering NFRs for ML in the following.

Accuracy & Performance. Most ML work reports on algorithm accuracy (often precision and recall), i.e., how "correct" the output is compared to reality. Further work looks more broadly at algorithm performance (e.g., [24]), including comparisons of performance in specific contexts (e.g., [25]).

Fairness. Recent work has focused on technical solutions to make ML algorithms more fair, finding that the removal of sensitive features (e.g., race, gender) is not sufficient to ensure fair results, and considering the trade-off between fairness and other NFRs [2]. Work in this area has attempted to find mathematical or formal definitions of fairness, e.g. statistical parity, individual fairness, and has found that the accurate implementation of fairness depends more on how fairness is defined and measured than how it is implemented [26]. Empirical work has asked practitioners about their needs for fairness in ML, finding that in practice, engineers want to consider the side effects of fairness and see fairness in the context of the broader system [27].

Transparency. Although the results of ML can have significant real-world impact, it is often not clear how these results are derived, causing issues in trust and transparency. Work has begun to look at better explaining ML results [3], [28] to try to mitigate this issue.

Security & Privacy. Efforts have been made to address privacy concerns when using big (often personal) data to facilitate ML. Work in [4] introduces protocols for preserving privacy in various ML approaches, and explicitly acknowledges the trade-off in terms of algorithm speed when revising techniques for privacy. Similarly, Bonawitz et al. introduce a method to preserve privacy in ML, focusing on keeping the overhead in terms of runtime and communication low [5]. Papernot et al. recognize the increase in ML-related security and privacy threats and create a threat model for ML [29].

Testability. Work exists which considers systematically testing the outcome of ML systems (e.g., [6]). However, the majority of work focuses on the other direction, applying ML to improve software testing strategies (e.g., [30]).

Reliability. Further work has considered reliability in ML, e.g., looking at the reliability of individual ML predictions, focusing on reliability estimation [31].

Other NFRs for ML, such as sustainability or maintainability, have not seen significant attention. In most cases, one can find work applying ML techniques for prediction of the NFR, e.g., to predict maintainability [32], but not work considering the NFR as it applies to ML. Similarly, efforts like the AIRE workshop series focuses on applying AI to RE, and typically not the other direction.

From a broader perspective, there are efforts to apply SE techniques to the application of ML [9], with a focus on reliability, testing and evolution. As far as we are aware, there is no unified collection or consideration of many NFRs for ML, including a consideration of ML-specific quality trade-off data. Current work consists of only individual considerations of specific quality trade-offs, e.g., privacy vs. processing time. Similarly, we are not aware of approaches for explicitly monitoring ML implementations at runtime, or considerations of what exactly runtime monitoring may mean in this context.

III. NFRs for Machine Learning Example

As a concrete example of an ML solution and its associated qualities, consider an airport which may screen passengers against images of people of interest; here a precise match is desirable, as the cost of misidentification is relatively high, yet the processing time may be relatively slow (e.g., 20 seconds), given the time one takes to get through security. These desired quality requirements should be captured and considered through the solution lifetime.

In order to understand what ML solutions may meet our quality requirements, we can attempt a relatively straightforward application of existing frameworks for NFR modeling, such as the NFR Framework [7] or iStar [11]. We do so using layers to separate different ML concepts in Fig. 1. We decompose available algorithms by problem type (classification, regression, etc.), then by algorithm characteristics (supervised, unsupervised, etc.), and by algorithm types (regression, Bayesian, etc.). The hierarchy and classification is yet incomplete, with the dimensions of incompleteness indicated by ellipses (...). We add a simple syntax addition to capture algorithm assumptions (A) and optimizations (O), e.g., the Naïve Bayesian algorithm assumes that the probability of all relevant events are independent, and optimizations such as Laplace Smoothing can be applied to account for unseen observations. The Nearest Neighbor algorithm can apply optimizations such as data projection or weighting the distances between neighbors.

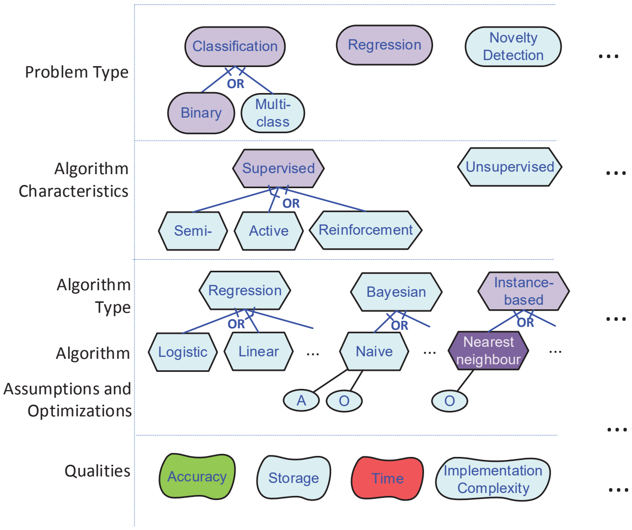

Fig. 1. Example Solution Space Representation evaluating Nearest Neighbor against Relevant Quality attributes (Simplified and Incomplete)

We can collect information on quality performance from existing sources. In some cases, such data is available generally; e.g., Nearest Neighbor without any optimizations has a very high running time, comparatively, but this decreased with a data projection optimization [1]. Other comparisons are more contextual, but often provide more specific findings; e.g., when applied to identify spam email, a variant of Nearest Neighbor has a high accuracy (~97%) [33]. We can use existing analysis procedures such as [12] to evaluate an option in the model, such as Nearest Neighbour (in dark purple), against relevant qualities (bottom row). In this model, the evaluated algorithm solution, Nearest Neighbour, may be suitable for the airport recognition case due to its high accuracy and lower speed.

Although this example helps to illustrate the use of NFR-based RE techniques for ML understanding and selection, many questions remain. Here we stick to well-known and easily measurable qualities such as accuracy and speed (Time). What if we had included fairness, transparency or security? Here we show only one level of qualities, but how do these qualities decompose into sub-qualities? How can these qualities be decomposed to measurements? We have also found some information linking the technical space to quality performance, but this information is scattered and has significant gaps. The notion of trade-offs here (e.g., accuracy vs. time) is covered only implicitly, and we are similarly lacking data to feed into models of this type. Finally, this model considers only design-time decisions, not covering runtime monitoring, or re-design or evolution in the face of change. We outline these challenges further in the next section.

IV. Challenges

Inspired by our example, and given our understanding of ML and of related work, we outline challenges related to treatment of NFRs for ML. The first four challenges relate to knowledge of NFRs (1), while the last three focus on methods (2). We acknowledge that this initial list of challenges may be incomplete.

C1. Our understanding of NFRs for ML is fragmented and incomplete, including how to define and refine NFRs in ML-specific contexts. Take fairness as an example. The general public are beginning to realize that ML algorithms may have an influence on critical decisions, and that such decisions may enforce unintended biases towards various vulnerable groups [34]. ML researchers have heard the call, and are working to find technical solutions to account for fairness (e.g., [2]). But what is fairness in these cases? How can it be effectively defined in ways understandable by ML? Does this definition change depending on the ML approach used (e.g., linear regression, neural networks). What type of fairness is needed? How does this change depending on the domain and context? We can ask similar questions for all of these qualities in an ML context, e.g., what constitutes maintainability of a ML solution? Portability? Usability? Similarly, how are these qualities refined in an ML-context, and does this refinement echo the non-ML cases?

C2. Our knowledge of how various ML algorithms, along with their optimizations and assumptions, affect relevant ML qualities, including trade-offs among qualities, is incomplete and fragmented.

The new meanings, refinements, and contexts of NFRs as applied to ML mean that our known space of NFR conflicts will have to be readjusted. Recent work has already begun to explore conflicts between specific NFRs in ML, e.g., fairness trade-offs can include performance or accuracy [2], privacy-preserving ML involves trade-offs with algorithm speed [4], but further work may find trade-offs in other, unanticipated quality dimensions such as reusability or testability. How does one balance trade-offs between these critical qualities in ML solutions? Such questions will become pressing as ML becomes more widespread.

C3. Given our new understanding of the meaning and refinements of NFRs, we must also understand how NFRs can be measured in practice for ML-based solutions.

In typical SE, work on software metrics has established a large body of possible metrics, linked to desired system quality, measurable at design or runtime, e.g., cross-references between components indirectly measure modularity, or number of runtime errors measure reliability. Some of these metrics can be applied as-is to ML solutions, while others must be reconsidered and rethought in an ML context, based on revised definitions of qualities, e.g., modifiability. Simple measurements like "average time to implement a change to the code" will no longer apply.

C4. We need to understand the effects of ML algorithms on desired qualities not only during ML solution design, but at runtime – during the lifetime of the ML solution.

In an ML context, runtime monitoring presents new challenges, understanding how to measure relevant qualities such as fairness or privacy, as ML-supported decisions are made in practice. This is particularly challenging due to the nature of ML implementations, as transparency concerning the inner workings of such algorithms is an open challenge.

C5. ML researchers and users currently lack an ML-specific way to express and specify quality requirements for ML, including targets and trade-offs, and the influence of domain context.

Although much of existing NFR-aware approaches can be reused in an ML context (see our Section III example), we anticipate that the rise of ML will reveal a variety of new concepts and challenges (e.g., training data, retraining, optimizations, networks, supervision) that will change the content and design of languages capturing NFRs for ML systems. Similarly, approaches supporting decision making using these languages will have to be adapted, not only as the underlying concepts are updated, but as the type of decisions and space of alternatives differ. E.g., the question is no longer "which requirements do I implement?", but "which algorithm type, with what characteristics, assumptions, training data, and optimizations do I use?"

C6. We need to understand how evolution, both in terms of available training data, and in terms of quality requirements and thresholds, may affect our ML solutions. How do we use our knowledge of ML quality performance to understand when to re-train? When to modify our solutions?

If we can understand how to define, refine and measure quality of ML solutions, monitoring quality over ML-based systems at runtime, we can understand when quality thresholds reach a point such that changes must be made. This can be changes in terms of available data, or changes in the requirements, particularly quality requirements or permissible quality levels. Existing approaches for software evolution for non-ML systems will need to be revisited and updated for an ML context.

C7. We need to understand how ML-based solutions integrate with typical software from a quality perspective.

Increasingly, data scientists and software engineers must work together to produce long-term, holistic, complex software solutions which include ML components. Thus, NFRs must be considered not only over ML components, with their unique quality definitions, refinements, and trade-offs, but also over a combination of ML solutions and typical software.

V. Research Directions

Given the challenges outlined in the previous section, here we outline a number of research objectives and plans which may address these challenges. Given the size of the problem, there are a wide variety of possible approaches which may address these challenges. Our suggestion is one of many possible approaches, and may be expanded as further challenges arise.

Obj1. Explore and define NFRs for ML, including possible interpretations and refinements of prominent NFRs in a variety of exemplar contexts.

Survey ML literature for NFRs. There is a need for one or more systematic literature reviews to find NFR-related definitions and refinements for ML. This should include relevant contextual information (e.g., the type of problem solved, the algorithm applied, and optimizations) which may affect the nature of the NFR decomposition and definition. For example, sustainability may have a different meaning when applying ML in a medical domain as compared to application for autonomous driving.

Ask ML experts about NFRs. One could use empirical methods like surveys or interviews to derive knowledge from ML experts or researchers, asking questions like: "what are important qualities for ML solutions? In what context? How are they refined?" Etc.

Consider Existing NFR Refinements. Taking sources such as [8] or various NFR catalogues, consider if and in what contexts such NFRs and their refinement may apply to ML. This may also involve elicitation from ML experts, asking them to confirm or deny the applicability of established knowledge to ML. Via these efforts, NFR ML would be derived bidirectionally, from ML to RE and back.

Obj2. Using the output of Obj1, create a catalogue of NFRs for ML.

Build NFR for ML Catalogue. The SLRs conducted in Obj1 will have produced much information in terms of NFRs for MLs, and this can be captured in a publicly available catalogue. The catalogue should be made available online in a format that allows feedback and comments.

Link to empirical knowledge. The empirical findings found as part of Obj1 can be captured in the catalogue, including links to sources and data concerning quality performance of particular ML approaches in certain contexts.

Crowd sourcing. In order to improve the completeness and accuracy of the catalogue, it may be possible to collect feedback via crowd sourcing, e.g., to ML-related conferences and groups. The output can be used not only for research but educational purposes.

Obj3. Collect operationalizations and measures of NFRs for ML refinements, when possible.

Focus on possible measurements of ML qualities, including potential contextual variance of measurements. In addition to searching through ML-related literature, one should explore relevant literature in software quality and quality metrics which may apply to ML NFRs.

Obj4. Develop a language and representation for expressing NFRs specifically for ML solutions and required concepts.

Develop conceptual underpinning. Based on the surveys conducted previously, researchers can produce an ontology of concepts needed for expressing NFRs over ML. This can be based on existing quality and NFR ontologies (e.g., [19]), but adapted and adjusted for ML-specific concepts and needs as based on the findings of **Obj1**. For comprehension and extension, the outcome can be captured as a visual metamodel.

Develop semantics. A formal semantics should be developed for the concepts, as appropriate. This will facilitate precise modeling and reasoning as in many existing RE approaches, and should be inspired by existing guidelines such as [35].

Develop graphical and textual syntax. Work in [22], [23] has already made progress in this subgoal. In order to be usable, language instance models should be read/writable in a variety of forms, including text, tables and graphical syntax, using principles, for example, from the Physics of Notation [36].

Obj5. Develop methods to reason over NFRs for ML at design time, making appropriate trade-offs to inform implementation decisions, and a runtime, continually monitoring achievement of critical qualities.

Develop reasoning. At system design time, we want to select ML solutions which make a good context-based trade-off between desired NFRs. Using the concepts and semantics developed thus far, and inspired by many existing approaches, one can develop procedures to allow for interactive and automated trade-off analysis, helping users to design an ML solution which adequately accounts for NFRs. When accurate and meaningful design-time data is available, approaches should favor quantitative reasoning, providing numerical information about NFR achievement. A mix of quantitative and qualitative reasoning can be supported, as in [21], for example, to deal with incomplete data.

Develop runtime reasoning. One can make use of the measurements collected, along with the reasoning procedures developed, to facilitate runtime monitoring and reasoning over ML qualities, i.e. to understand how sub- and super-qualities are satisfied as the ML solution runs.

Obj6. Create methods to deal with changing data and changing quality requirements, including triggers based on measurement thresholds, and a process for periodically re-evaluating ML quality needs. Recent work concerning software adaptation and evolution can feed into the satisfaction of this objective.

Methods for changing NFRs. When quality requirements change there must be procedures and methods in place to guide changes to the ML-based systems. This may include eliciting thresholds and deciding when and if to make changes, depending on desired and current quality values.

Methods for changing data. As the sources, quality, and content of data available for ML changes, methods are needed to guide system changes. This includes when to change data sources, or perhaps ML algorithm choices, and how often to retrain based on current data.

VI. Discussion and Conclusions

Although our agenda is preliminary, to our knowledge this is the first RE-oriented broad consideration of many types of qualities over ML solutions. We aim to bring NFRs for ML to the forefront, facilitating early consideration, definition, and trade-off analysis, raising awareness over ML performance in terms of such qualities, and understanding how this information will lead to necessary changes.

The presented agenda opens significant opportunity for future work. The quality findings and catalogue will reveal gaps in ML knowledge, particularly defining and refining quality characteristics for ML algorithms, as well as empirical data reporting on quality trade-offs. This can lead to comparative empirical work evaluating ML algorithms against different quality attributes in different contexts. Gaps in terms of ML quality achievement will also spearhead new technical efforts to improve ML techniques to better satisfy such qualities, similar to the effort of [2] for fairness and [4] for privacy.

Just as ML qualities such as fairness and transparency are currently receiving much attention, other NFRs such as sustainability and modifiability may receive similar attention. Execution of the proposed agenda paves the way for considering these and other critical NFRs in an ML context.

References

[1] A. Smola and S. Vishwanathan, *Introduction to machine learning*. Cambridge University Press, 2008.

[2] T. Kamishima, S. Akaho, and J. Sakuma, "Fairness-aware learning through regularization approach," in *2011 IEEE 11th International Conference on Data Mining Workshops*. IEEE, 2011, pp. 643–650.

[3] O. Biran and C. Cotton, "Explanation and justification in machine learning: A survey," in *IJCAI-17 Workshop on Explainable AI (XAI)*, 2017, p. 8.

[4] P. Mohassel and Y. Zhang, "Secureml: A system for scalable privacy-preserving machine learning," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 19–38.

[5] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 1175–1191.

[6] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, 2011.

[7] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, *Non-functional requirements in software engineering*. Springer Science & Business Media, 2000, vol. 5.

[8] ISO/IEC, "ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models," Tech. Rep., 2010.

[9] SEMLA, "Software engineering for machine learning applications (SEMLA) initiative," https://semla2018.soccerlab.polymtl.ca/program/, 2018, accessed: 2018-07-14.

[10] R. Binns, "Fairness in machine learning: Lessons from political philosophy," *arXiv preprint arXiv:1712.03586*, 2017.

[11] E. S. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," in *Proceedings of ISRE'97: 3rd IEEE International Symposium on Requirements Engineering*. IEEE, 1997, pp. 226–235.

[12] D. Amyot, S. Ghanavati, J. Horkoff, G. Mussbacher, L. Peyton, and E. Yu, "Evaluating goal models within the goal-oriented requirement language," *International Journal of Intelligent Systems*, vol. 25, no. 8, pp. 841–877, 2010.

[13] Y. Wang, S. A. Mcilraith, Y. Yu, and J. Mylopoulos, "Monitoring and diagnosing software requirements," *Automated Software Engineering*, vol. 16, no. 1, p. 3, 2009.

[14] V. E. S. Souza, A. Lapouchnian, K. Angelopoulos, and J. Mylopoulos, "Requirements-driven software evolution," *Computer Science-Research and Development*, vol. 28, no. 4, pp. 311–329, 2013.

[15] F. Dalpiaz, P. Giorgini, and J. Mylopoulos, "Adaptive socio-technical systems: a requirements-based approach," *Requirements engineering*, vol. 18, no. 1, pp. 1–24, 2013.

[16] M. Glinz, "On non-functional requirements," in *15th IEEE International Requirements Engineering Conference (RE 2007)*. IEEE, 2007, pp. 21–26.

[17] L. Chung and J. C. S. do Prado Leite, "On non-functional requirements in software engineering," in *Conceptual modeling: Foundations and applications*. Springer, 2009, pp. 363–379.

[18] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, "Sweetening ontologies with dolce," in *International Conference on Knowledge Engineering and Knowledge Management*. Springer, 2002, pp. 166–181.

[19] F.-L. Li, J. Horkoff, J. Mylopoulos, R. S. Guizzardi, G. Guizzardi, A. Borgida, and L. Liu, "Non-functional requirements as qualities, with a spice of ontology," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. IEEE, 2014, pp. 293–302.

[20] F. Dalpiaz, A. Borgida, J. Horkoff, and J. Mylopoulos, "Runtime goal models: Keynote," in *Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on*. IEEE, 2013, pp. 1–11.

[21] J. Horkoff, D. Barone, L. Jiang, E. Yu, D. Amyot, A. Borgida, and J. Mylopoulos, "Strategic business modeling: representation and reasoning," *Software & Systems Modeling*, vol. 13, no. 3, pp. 1015–1041, 2014.

[22] S. Nalchigar and E. Yu, "Business-driven data analytics: a conceptual modeling framework," *Data & Knowledge Engineering*, vol. 117, pp. 359–372, 2018.

[23] ——, "Designing business analytics solutions," *Business & Information Systems Engineering*, Aug 2018.

[24] M. Sokolova, N. Japkowicz, and S. Szpakowicz, "Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation," in *Australasian joint conference on artificial intelligence*. Springer, 2006, pp. 1015–1021.

[25] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 161–168.

[26] S. Corbett-Davies and S. Goel, "The measure and mismeasure of fairness: A critical review of fair machine learning," *arXiv preprint arXiv:1808.00023*, 2018.

[27] K. Holstein, J. W. Vaughan, H. D. III, M. Dudík, and H. M. Wallach, "Improving fairness in machine learning systems: What do industry practitioners need?" *CoRR*, vol. abs/1812.05239, 2018. [Online]. Available: http://arxiv.org/abs/1812.05239

[28] A. Datta, S. Sen, and Y. Zick, "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems," in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 598–617.

[29] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[30] C. Murphy, G. E. Kaiser, and L. Hu, "Properties of machine learning applications for use in metamorphic testing," 2008.

[31] Z. Bosnić and I. Kononenko, "An overview of advances in reliability estimation of individual predictions in machine learning," *Intelligent Data Analysis*, vol. 13, no. 2, pp. 385–401, 2009.

[32] R. Malhotra and A. Chug, "Software maintainability prediction using machine learning algorithms," *Software Engineering: An International Journal (SEIJ)*, vol. 2, no. 2, 2012.

[33] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. D. Spyropoulos, and P. Stamatopoulos, "Learning to filter spam e-mail: A comparison of a naïve bayesian and a memory-based approach," *arXiv preprint cs/0009009*, 2000.

[34] R. Hauser, "Can we protect AI from our biases?" https://tinyurl.com/y9lvqxc8, March 2019.

[35] I. Jureta, *The design of requirements modelling languages: how to make formalisms for problem solving in requirements engineering*. Springer, 2015.

[36] D. Moody, "The physics of notations: toward a scientific basis for constructing visual notations in software engineering," *IEEE Transactions on software engineering*, vol. 35, no. 6, pp. 756–779, 2009.