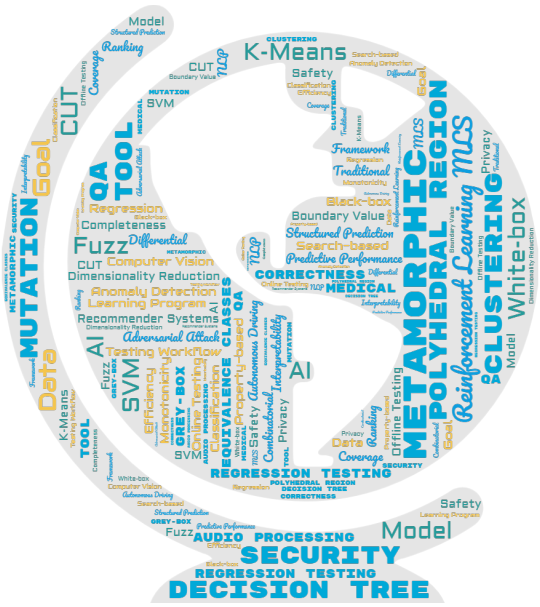
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



A Guide to Quality Assurance of Machine Learning Software

A Toolkit based on Goals, Contexts, and Testing Solutions

Master's thesis in Computer science and engineering

ERIK BOCK
ALEXANDER SIMOLA BERGSTEN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

A Guide to Quality Assurance of Machine Learning Software

A Toolkit based on Goals, Contexts, and Testing Solutions

Erik Bock
Alexander Simola Bergsten

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering

Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

A Quality Assurance Guide for Machine Learning Software
A Toolkit based on Goals, Contexts, and Testing Solutions
Erik Bock & Alexander Simola Bergsten

Supervisor: Robert Feldt, Computer Science and Engineering
Advisor: Leonard Aukea, Volvo Cars
Examiner: Regina Hebig, Computer Science and Engineering

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A word cloud resembling a globe containing the tags used in the mapping.

Typeset in LaTeX
Gothenburg, Sweden 2022

A Quality Assurance Guide for Machine Learning Software
A Toolkit based on Goals, Contexts, and Testing Solutions
Erik Bock
Alexander Simola Bergsten
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

The intersection of quality assurance (QA) and machine learning (ML) is a research area that has sparked interest in recent years. In this thesis, the focus is on traditional ML, which includes all types of ML algorithms except deep learning. To this end, we set out to investigate the context of different ML-based systems (MLS) and the QA goals that are applicable to ML. By snowballing three recent literature reviews and surveys, and analysing the collected set of papers, we created a curated list of papers that propose testing techniques for different MLSs. Each paper focuses on one or more testing goals, such as fairness or robustness, to help developers achieve these goals for their systems. In total 14 goals have been defined and incorporated as tags that classify the collected papers. With this knowledge we have created a mapping of testing solutions, that would output a paper proposing a solution given a testing goal and a context of a system as the input. This forms the foundation for our toolkit that guides developers in finding appropriate testing solutions for their particular ML project. This tool was implemented as a filterable table in Microsoft Lists. Eight employees from Volvo Cars participated in a judgement study to evaluate the effectiveness and user experience of applying the toolkit on two industrial use cases. These eight subjects, each experienced within ML, were interviewed to collect feedback about the mapping and toolkit. The evaluation shows that the current implementation of the toolkit has some limitations but is capable of helping developers find relevant papers for their use cases. By making it easier for ML developers to find QA solutions, both current and future products and services that utilise ML can become more reliable for its users.

Keywords: traditional machine learning, testing, software engineering, toolkit, mapping, quality assurance, project, thesis.

Acknowledgements

We would like to thank our supervisor Robert Feldt for his guidance and support during the project and the writing of this thesis. We would also like to thank our advisor Leonard Aukea and the team at Volvo Cars for providing us with the opportunity to work closely with the real needs in the industry, which helped bring further relevance to our thesis. Last but not least, we would like to extend our gratitude to the participants of the judgement study, since they were instrumental in evaluating the proposed toolkit.

Erik Bock & Alexander Simola Bergsten, Gothenburg, June 2022

Contents

List of Figures

List of Tables

1
Introduction

How can developers test machine learning (ML) software to assure themselves that it is ready for deployment? More and more research within ML focuses on this problem [1], as system failures caused by poorly constructed ML can be devastating for individual companies. Such failures can also be harmful to Artificial Intelligence (AI) in general as the trust of AI in the general public decreases. In this thesis, we set out to map quality attributes, which we refer to as goals, and the ML context of different systems to papers proposing testing solutions. The focus is on traditional ML, which refers to all ML algorithms that are not part of deep learning (DL). We focus on this area partly because these algorithms are preferable in certain situations, in which case it is important for companies to be able to test their ML-based system (MLS). This mapping is used as the basis for a toolkit that can guide developers in how they should test their MLS given that they know their goals with testing and the context of their systems.

The following sections explain the motivation, scope, and structure of this thesis. First, Section 1.1 gives a background about the need for quality assurance (QA) of ML software, and hence the need for this study. Second, in Section 1.2, the problem statement for this thesis project is provided. Then, Section 1.3 describes the academic and industrial motivations for the different parts of this project. This section also describes the research questions, limitations/delimitations, and significance of the study in their respective sections. In Section 1.3.1 the three research questions are presented, while Section 1.3.2 motivates the constraints on the scope. Section 1.3.3 highlights the value that the thesis contributes both to the academic field and the industry. Following that, the societal and ethical concerns of this project are addressed in Section 1.4. Finally, in Section 1.5, an outline of the other chapters in this thesis is given.

1.1 Background

Machine Learning is a tool that is used in an increasing capacity within software engineering (SE) in many business areas [2]. Some typical applications of ML are autonomous driving [3], [4] and recommender systems [5], where it is used to solve the issues that these fields have in an efficient and accurate way [6]. Just like any other software component, when an ML component is included in a software system it needs to be tested. While much research in the ML research community is focused on testing [1], [7], this is typically considered only in a basic sense, i.e., to ensure that

the predictive performance (e.g., accuracy) of the model is satisfactory. In terms of traditional software testing, this corresponds most closely to ensuring that the main use case is correctly handled. This is a specific and narrow form of testing, that is rarely considered sufficient. Typically, traditional software testing includes a much broader set of techniques, such as pairwise testing [8], metamorphic testing [9], fairness testing [10], testing for boundary and edge cases [11], stress testing [12] etc. In the traditional setting, it is also more common to test for different quality attributes of the system, which this thesis refers to as the **goals** of doing the testing. For example, a goal could be to achieve quality attributes like the established ones in ISO 25010 [13] like correctness and completeness. It could also be attributes that are particularly important for ML like robustness and fairness.

It is not yet clear what these additional testing techniques would correspond to for ML components. Some of them have been investigated and ML counterparts have been at least partly developed [14]. Others might not have been tried in an ML setting yet, with some likely needing adaptation before they are. The rest might not work at all due to the stochastic nature of ML. Similarly, for traditional software testing, there is also a broader set of overall processes for testing that have been developed, while ML is lacking a standard process or method covering a broad set of testing techniques [15], [16]. The goal of this thesis project is to investigate how to better support software practitioners in doing QA, i.e., covering multiple facets and testing requirements, of ML software components.

This project involves both theoretical investigation of the literature, to establish existing solutions, as well as work at a case company, to gather test requirements and provide a concrete environment in which to develop and evaluate our work. Within ML, our focus is on traditional ML which we define as all ML that is not DL. The reason why traditional ML is the focus is that the team at the case company, Volvo Cars, has a need for testing techniques for those types of models. Most current research focuses on either testing DL models or general testing of models, with the number of DL papers especially increasing each year [1]. There has not been as much research into traditional ML specifically [17], so this thesis is an opportunity to help provide value to that field.

A useful and popular research technique to lay out the landscape of existing knowledge is by way of "mapping knowledge domains" [18]. The end result of this is a **mapping**, i.e., a model of the associations between items from two or more groups. Shiffrin and Börner [18] claim that this type of research can assist people in finding domain-specific information when searching for it on the internet, as it gathers and structures sources of information that are otherwise scattered on the web. A mapping can also provide valuable information by itself in highlighting relationships or enabling statistics to be generated. This thesis maps the knowledge domains of ML and QA to existing testing techniques. Moreover, the method used to create this mapping is essentially a systematic mapping study, as presented by Petersen *et al.* [19]. To avoid confusion in this thesis, the categorised set of papers will be referred to as the "mapping", while the tool that is built to convey the information of the

mapping to developers is referred to as the "toolkit".

1.2 Problem

Software testing is important in the field of SE [20], as untested applications are prone to decreased usability and correctness due to bugs and unexpected behaviour. Because ML is increasingly used in components included in software systems [7], it is important to have processes and techniques for QA on these MLSs. Merely testing the predictive accuracy will rarely be enough. While there are many testing techniques for traditional software, many of them are not applicable to testing of ML software due to reasons such as the stochastic nature of an MLS [21]. However, there exist techniques that can still be adapted, which are essential to be able to provide a good method for assuring software quality of the ML components.

Volvo Cars apply ML in some of their products, such as neural networks for autonomous driving. However, the context they provide for this project is the development of ML models that are specifically not neural networks. The main reason for this is that neural networks can be difficult to interpret [22], in comparison to other types of models. In many cases, other types of models that are more interpretable are favoured despite the sometimes-improved predictive accuracy that neural networks offer. Since this thesis focuses on traditional ML to fit with this context, that means that there is less literature available compared to if the scope would include DL. This issue was overcome by not only looking at traditional ML methods but also general ML methods that can be applied to any type of ML. The important part is that the techniques proposed in the studied literature should at least be applicable to traditional ML.

1.3 Purpose

The purpose of our study is to investigate specific techniques for QA, covering multiple aspects and requirements, of ML components focusing on traditional ML. Additionally, the purpose includes developing a toolkit for these techniques. This toolkit will aid developers in identifying papers that can guide them to achieve their testing goals within their own context. The thesis project is split into two main parts.

First, the study considers which goals are addressed by testing techniques from traditional software testing, in order to create a mapping for selecting one or more testing techniques for testing an MLS. The mapping suggests different techniques based on the context (e.g., type of ML task, limitations) and the testing goal (e.g., robustness, safety, fairness) as well as an overall process for when to apply them. This part of the study provides theoretical value for the research community but also serves as a basis for supporting practitioners.

Second, the study takes real project cases from Volvo Cars that can be used as a basis for the cases used during evaluation. The completed toolkit is then evaluated

on these realistic cases, which helps determine if it can effectively help a company such as Volvo Cars in achieving its testing goals. This also provides insight into if the mapping itself, that is the foundation of the toolkit, has been constructed in an effective way. Recall that "testing" here refers not only to the act of measuring the predictive accuracy of ML models but also to a broader type of testing, for example, concerning the robustness or fairness of different ML components. The evaluation is done through a judgement study [23], where experts at Volvo Cars are able to give their feedback on the toolkit given that it is applied to some cases that have been derived from real projects. With this feedback, it is possible to remedy some of the weaknesses of the toolkit, and it provides insight into potential future work.

Together, these parts benefit both the academy and the industry. For the academic side, the thesis provides more research on testing techniques for traditional ML, and it also provides a study based on real project data from Volvo Cars. The industry benefits from this study since Volvo Cars will be able to use our toolkit and provide feedback on it, while they also receive an idea of what testing techniques they can apply to their own projects. If other companies are interested in the concept they are also free to utilise the toolkit in their own projects.

1.3.1 Research Questions

Listed below are the three research questions for this thesis.

RQ1: Based on traditional machine learning literature that has taken a quality assurance perspective on testing, which goals exist for testing the machine learning-based systems and how are they defined?

Testing has been a part of SE and QA for a long time, due to the importance of knowing that software behaves correctly and performs adequately. Since ML is becoming more and more prevalent in different types of software, it is important to know which of the traditional software testing techniques can help fulfil the same testing goals, while also investigating new QA testing techniques that have been created specifically for the ML domain. Important to note is that our strategy for identifying relevant literature focuses on studying ML literature that has been inspired by QA testing, instead of looking at pure QA literature. The reason for this is that studying pure QA literature reveals very little about how well it can be applied to the ML domain, while our strategy on the other hand builds upon all the work that has already been done in this regard.

The answer to RQ1 is presented as a list of goal definitions. These definitions have in many cases been adapted from traditional software testing into goals that are also applicable to ML models and MLSs.

RQ2: How can a generalised mapping that outputs a paper proposing a testing solution, given a testing goal to be achieved and the context of a machine learning-based system, be designed? How can a toolkit be designed to help the user utilise

the mapping?

The mapping is meant to connect papers that propose testing solutions with QA goals and contextual variables of ML projects. To "design a mapping" consists of first identifying the items in the groups that are to be related. In this case to find relevant papers, define concrete goals, specify features that constitute "context" and define values for these context variables. The second step is to associate the items, i.e., connect the papers with the goals and context variables. The created mapping takes goals and context variables as input and then outputs one or more papers about different testing techniques as a result. For example, when Volvo Cars are testing a specific model, they might have the goal of identifying cases where that model performs poorly. To extend the example, the context could be that the model is trained on image data, through online learning, and used to classify the images. All this, when put into the toolkit, would for example recommend a paper that would guide the developer in how to apply techniques such as coverage-based and search-based testing [20] for this type of system. The mapping is generalised in the sense that it can be applied across domains, and is not limited to the automotive domain of Volvo Cars.

RQ3: Given real project descriptions from Volvo Cars, does the toolkit effectively help them with identifying testing techniques that can help them achieve their goals?

The purpose of this question is to evaluate the effectiveness of the toolkit in an industrial setting. The project descriptions will be based on actual projects, to provide additional realism to the evaluation. Volvo Cars employees that work with ML will be able to use the toolkit to figure out which goal to prioritise and then based on the context of the project figure out which testing techniques it would recommend them to implement. A judgement study is conducted to evaluate this research question.

1.3.2 Limitations and Delimitations

The main limitation is that we focus mainly on ML that does not include DL. We leave potential extensions to DL techniques and settings for future work, to allow the toolkit to be applicable to all types of ML. Below, we discuss other limitations in more detail.

Limitations:
The project is done in cooperation with Volvo Cars. Working with a company allows for finding real requirements and working with real data, which provides a lot of interesting opportunities for research in a real-life context. However, working with an enterprise also means that we are limited to working within the context that is given, so some avenues of research are not possible. There is also the issue of external validity [23], since much of the research is context-dependent it was not possible to generalise all of it. To help remedy this, we considered the possibility to take the techniques discovered and the toolkit developed during the project and

apply them to a different domain. This is also the reason why the first part of the project is more general, to make sure that there are some parts that are useful even outside the context of Volvo Cars. Through our supervisor, we would have access to additional industrial cases, but due to time limitations, we did not have the time for more than one judgement study.

Delimitations:
There was a time limit on this project which meant that we needed to delimit the project to fit within the time frame. This meant that the project needed to be limited to researching and creating a mapping for the most well-known and respected methods for testing ML software. Conducting a complete literature review would take too much time away from the toolkit construction and evaluation stages, and would therefore not be feasible for this project. There was also no time for any concrete implementations of testing techniques recommended by the toolkit since these would have consumed a lot of time without bringing much value to the report. The focus is therefore instead on evaluating the toolkit on project cases based on real projects, where experts can help determine if the recommendation is sensible or not.

There are also delimitations that had to be set because of the needs of Volvo Cars described earlier since the aim of the project was to provide value both to the research field and to Volvo Cars. The latter part of the project applies the toolkit to the goals and context (other models than neural networks) of Volvo Cars. The evaluation will therefore only be done on data provided by Volvo Cars.

1.3.3 Significance

The mapping created in the first part of the project provides a general way of choosing a certain testing technique given a context and a goal. This provides academic value, where the mapping provides an overview of existing solutions and can serve as a basis for future research. On the other hand, the toolkit that is built around the mapping provides value for practitioners. For them it becomes easier to pick which techniques they should use to test their ML software components with they guidance of the toolkit.

The research conducted during this project also helps extend knowledge of testing traditional ML models, which is currently a field that has not been explored to an extent like the testing of DL systems [17]. Furthermore, since this project is a collaboration with Volvo Cars there is a real need for the research, and the development is done within a real context. This means that the thesis can be immediately useful to practitioners, especially Volvo Cars, and the hope is that the real-life context given in this project can also be useful to future research since it gives an idea of what the needs of the industry can be as well as what data the industry is most likely able to provide.

1.4 Societal and Ethical Aspects/Considerations

In this section, we discuss the societal and ethical concerns that might be relevant to this thesis.

First off is the ethical aspect of fair AI, based on bias and fairness as discussed in [24]. This thesis deals with the testing of ML software, a subset of AI software, which means that this research might have implications for the fairness of this type of software. Since AI fairness is a big concern in both the academic field and in the industry, it is very important to use techniques that try to remedy bias and other fairness issues within the software [25]. This thesis, therefore, plays an important role in helping developers identify the need for fairness testing, and giving them concrete ideas for how to do it, which will hopefully lead to increased fairness in many MLSs in the future.

Another similar ethical aspect is that of security and privacy, which has been a growing concern in the past decade [26]. One of the biggest concerns is how to design secure and robust AI that will not make disastrous mistakes or be easy to take advantage of by adversaries. Since the goal of this thesis is as previously stated to improve the quality of MLSs, this goal aligns with the current interests of large organisations such as the EU [26] that want AI to be more regulated to make sure that the quality is assured.

AI and ML in general have been argued to have a few potential negative societal impacts. One example is that their increasing use might lead to increasing unemployment rates in the future [27]. To deal with this there is a need for societal changes that can handle the fact that robots can do a job equally well as or better than a human, but until those changes happen AI will most likely have a negative societal impact on a lot of people. This drawback of AI does not mean that the development and research should stop, and once the hurdle of machines replacing human jobs is surpassed we believe that society will be better for it. The research in this paper strives to help improve the current generation of ML models by providing insight into how the quality of the models can be tested. This also means that this thesis might be helpful in mitigating these issues to some extent, by making it easier to create fair and safe models.

Another disputed concern in the AI research field is also the concept of a future technological singularity [28], where machines will reach a point where their intelligence has surpassed the collective human intelligence. According to the theory, at that point, the machines would begin rapidly improving themselves which would leave humanity behind and lead to big changes being necessary for human civilisation. However, even if this theory turns into reality, this thesis will not have played a significant role in it. This thesis focuses on improving the quality of ML-based software, and therefore only affects the intelligence of AI by preventing failures in the form of bugs or fairness concerns, for example.

1.5 Thesis Outline

This section outlines the structure of the thesis. **Chapter 2** presents all the necessary background information regarding AI and ML, as well as testing within the field of SE. The same chapter also includes all of the related work, divided into subsections based on their connection to this thesis. **Chapter 3** presents the work that was performed during the project, starting with the research phase that resulted in the mapping. After that, the creation of the toolkit is described, and the set-up for the judgement study at Volvo Cars is described. **Chapter 4** presents the results from the research phase and the evaluation. It first presents the definitions of all the goals, followed by summaries of all the paper included in the mapping. Subsequently, the toolkit is presented and the results from its evaluation at Volvo Cars are shown. **Chapter 5** discusses the results presented in the previous section and provides further insight into what the toolkit has done well and what is open for improvement. Finally, **Chapter 6** summarises the thesis and presents opportunities for future work.

2
Background

This section presents background information on ML and AI as well as the current field of testing in SE. This information is needed to understand the components of the toolkit presented in later chapters. Related work is presented in the last section of this chapter.

2.1 Machine Learning and Artificial Intelligence

Artificial intelligence as a concept has been notoriously hard to define, with several different definitions existing in the literature. Borg [29] argues for a definition that would be applicable across time, as the definition of AI has changed many times throughout history. He recognises the historical definition of AI as "the science and engineering of making intelligent machines" and proposes the new definition of AI as "software that enables automation of tasks that normally would require human intelligence" [29, p. 72]. Russell and Norvig [22] describe four different schools of AI, depending on the definition of intelligence. They also provide a definition of their own, saying that AI is "the study of agents that receive percepts from the environment and perform actions" [22, p.7].

Machine learning is a subfield of AI, which Russell and Norvig [22] define as the study of agents that improve their performance based on experience. Experience usually takes the form of data, and performance often refers to how accurately the agent behaves when receiving input it has not seen before. A more technical definition of ML, also known as statistical learning, provided by James *et al.* [21] states that it is a family of tools for analysing data and constructing predictive models. The authors categorise ML into supervised and unsupervised learning techniques, and some tools that lie in-between form the semi-supervised learning category. There is also another major category called reinforcement learning where the learning agents are not passively learning from data as in the supervised/unsupervised setting, but instead through trial and error [22]. The following paragraphs explain the fundamentals of these learning problems.

Supervised learning concerns the construction of models that can predict an output given an input, using data containing labelled examples of the phenomenon to be predicted [22]. Some typical instances of supervised learning problems are image classification and speech recognition. The training of a model is done with an ML algorithm and involves finding good values for the model's parameters. Some

common algorithms in use are Naive Bayes, Support Vector Machine, and Random Forest. For example, to make a program that can classify animal images, the ML engineer can supply a large number of labelled images to the learning algorithm, where the labels say which type of animal is depicted. This is the training step of the supervised ML workflow. The output of the training is a model, a piece of software that represents the knowledge gathered from the training dataset of labelled images. With a model, the computer can begin to predict the label of new images that were not part of the training dataset. There are several kinds of models, and the most appropriate type differs between problems.

Unsupervised learning is about analysing data without any labels, also called ground truth, to identify relationships and patterns [22]. For example, given a dataset containing coordinates of things or people, it is possible to look for clusters of things that are located close together, and perhaps these clusters distinguish themselves in some other feature of the members. In particular for clustering, there exists a wide range of algorithms, such as K-Means, DBSCAN, and Gaussian Mixture [30]. Another unsupervised problem is that of dimensionality reduction, in which Principal Component Analysis (PCA) is a technique that can be used to visualise high-dimensional data in a lower-dimensional space [21].

Reinforcement learning (RL) is the problem of learning to act optimally in an environment to achieve some goal [22]. The behaviour is learned from receiving rewards and punishments, called reinforcements, from the environment. A trade-off that must be made in all RL tasks is whether the learnt behaviour should maximise the reward in the short or long term. Some applications of RL is in teaching a computer to play video games, and in robotics to perform advanced movement and coordination. In the case of playing games, the rules of the game might be initially unknown to the machine, and the only feedback it receives regarding its performance is whether it won or lost. Varying degrees of information can be at the disposal of the machine, e.g., the computer might receive feedback at certain time intervals. There exist different algorithms within RL whose appropriateness and applicability depend on what information is available. Q-Learning is a popular method that can be applied when there is no model of the environment.

Deep learning is a class of ML algorithms that produce non-linear function approximations, called neural networks [21]. This is a type of model made up of layers of nodes that compute an output based on input from the previous layer. The term "deep" refers to the multi-layered structure of the models, essentially transforming the input data in multiple steps to produce the output. Convolutional neural networks (CNN) is a popular neural network model architecture used extensively in image classification tasks. For sequence processing, like speech and text classification, recurrent neural networks (RNN) are often preferred.

Natural language processing (NLP) is a popular field within AI and ML. It is about teaching computers to interpret natural language and communicate with their human users with words [22]. A common textbook example is the problem of classify-

ing e-mails as spam or not spam. Other NLP tasks are speech recognition, machine translation, and question answering. While many solutions within NLP involve ML, there are also many rule-based techniques that involve traditional programming. Sometimes the output of a model takes on the form of more structured objects like trees or graphs, as is the case in NLP where the model output could be a parse tree or a paragraph of text. This problem is called structured prediction [31]. Besides NLP, structured prediction can be about predicting the structure of proteins given a string of descriptive characters, or the problem of de-noising blurry or otherwise perturbed images.

An assumption of ML, in general, is that the distribution of the training data and production data is the same, but this is usually not the case after some time has passed. Online learning is a technique to maintain the performance over time of a model that is deployed in its application environment. Russell and Norvig [22] explain online learning as being a continual learning process where the machine receives input, predicts the output, and lastly receives the correct output that can be used to tweak the model parameters.

Taking a SE perspective on ML, MLS development can be seen as a new programming paradigm [32]. As a concept, a program can be regarded abstractly as a black box that takes input and produces output, like a mathematical function. An ML model could then be treated as a program since it acts like a ready-to-use piece of software that, when given an input, can compute the corresponding output. ML models can also be stored in a computer's memory or hard drive, just like an executable program. In traditional SE, developers manually write source code that is transformed into an executable program by a compiler. Providing this program and input data to a computer produces output data. In MLS development, the computer is given input and the corresponding intended output, which when passed to the learning program produces a program, or model. One could say that the position of the program and output are swapped, with respect to what developers need to provide the computer and in turn what the computer produces. However, this assumes that the learning program has been written, if not then the developer first has to create a learning program that defines the ML algorithm used to train the desired model.

2.2 Testing in Software Engineering

Testing software is an important part of SE [33], and one that has been heavily researched. Testing is seen as vital to healthy and quality software, since different types of testing techniques and testing attributes can provide assurance that the software functions correctly and that it is maintainable. It is worth noting that, as the maxim goes, testing can never prove the absence of bugs, only the existence of them.

The ISO 25010 standard [13], specifies a large number of quality attributes that together help developers determine the quality of their software. To test for different types of quality attributes, such as functional correctness or performance efficiency,

there are various techniques and metrics that can be used. However, these differ depending on the attribute at hand, since not all techniques will guarantee that all attributes are fulfilled. Fulfilling an attribute is also not a completely black and white issue, a piece of software can for example be at various levels of maintainability, not only maintainable or not maintainable. The goal is often to maximise these attributes as much as possible, but focusing on improving one attribute might mean that other ones are affected negatively. One example of this would be when the code is changed to be more efficient, this might result in the side effect of reducing the code's readability [34]. In most cases, however, there are standard practices that are meant to deal with issues such as these, with the example above having the practice of always designing readable simple code first and then only optimising the parts that are necessary. Furthermore, attributes affecting each other negatively is not usually the case since many of them benefit from what has become standard practices in SE such as keeping the dependencies to a minimum and achieving high modularity.

There is a wide array of different testing techniques that are useful in different contexts and provide different ways of testing the quality of a piece of software. These techniques are usually divided into black-box and white-box testing techniques [35]. A black-box technique does not use source code but instead bases its tests on the software specification, and therefore only cares about the input and output of the system. This also means that black-box techniques are best suited for testing attributes that are directly related to fulfilling the specification such as functional suitability. An example of a black-box technique is boundary value analysis [11], which is about looking for pairs of close-proximity inputs that differ a lot in the output, used to identify boundary values that are fault-prone. White-box techniques on the other hand do the opposite of black-box techniques, they fully utilise the source code but do not use the software specification, which means that they are better at identifying and locating faults, or bugs, in the code. An example of a white-box technique is coverage-based testing [36], also known as structural testing, for which a test adequacy criterion is defined, e.g., path coverage. The goal with this type of testing is to maximise the test adequacy criteria, so in the case of path coverage, the goal would be to make sure that every path through the program has been executed at least once. Picking stronger criteria will reveal more faults, but will also be more difficult and resource-intensive to create and run. There are also testing techniques that utilise both the source code and the specification, and these are sometimes referred to as grey-box testing techniques [35].

Testing itself is often divided into different levels [37], which groups different testing techniques together and guides developers to where in the SE pipeline they should conduct specific types of testing to achieve their testing goals. The original idea comes from the paper by Forsberg and Mooz [38], which introduced the V-Model. This model was meant to improve the waterfall model by integrating testing into the entire pipeline, and not having it as an action that was only performed at the end of a project. The model describes how developers should start planning acceptance tests while specifying the requirements for the software. Acceptance tests refer to

tests that are performed to guarantee that a product lives up to its customer/user requirements [37], so naturally, the tests must be designed based on these requirements. Next, we have tests on the system level that are meant to guarantee that the software as a whole is correct according to the specification created from the requirements. Important to note is that a piece of software can live up to the specification, and thus be verified, without living up to the user requirements, and therefore not be validated, since the specification might have missed or been unable to properly catch all the requirements. The next level of tests is the integration level, where different components are tested together to make sure that they have been properly integrated with each other. This is important since even if a component can seem to function well on its own, there might be issues once it is integrated with other components [37]. The final level is the unit testing level, where individual modules are tested. Modules in this context can be packages, classes, or even individual methods. This type of testing is important since it is the testing that is closest to the code and therefore the one that helps the most in revealing and localising faults.

An ever-present problem in SE is the *oracle problem* [39], [40], which is related to the concept of test oracles. A test oracle is a mechanism used to derive the correct output from a function, so that it can be compared to the actual output and determine the actual output's correctness. The problem is that it is often not a simple task to generate an oracle for a given test, since if there was a guaranteed way to know the correct output of a test given the input, then that method could have been used instead of the method that is being tested. Some existing solutions to remedy the oracle problem is to use either a derived, specified, or an implicit oracle [41]. For a derived oracle, the developers base the oracle on already existing software artefacts such as documentation or previous executions, as is the case in regression testing. With a specified oracle the correct results are instead based directly on what the specification states, which means that the specification needs to specify what the output should be for a given input. Implicit oracles are oracles that require no additional information from a specification or from software artefacts. Instead implicit oracles determine correctness more generally, detecting crashes being a prime example of a test that can be conducted for most software [41].

2.3 Related Work

In this section, a number of related works are presented and their relevance and overlap with this thesis are discussed. The goal is to provide some insight into what has already been done in the field, which also shows where there is still a need for research to be conducted. The first section here contains papers that discuss goals and solutions in a similar manner to our own thesis, by presenting the goals and then discussing new or existing solutions to them. The second section has papers that are purely focused on discussing ML testing techniques. The final section contains any other related work that could not fit into these two categories.

2.3.1 Papers About Goals and Solutions

To address confusion over different definitions of terms used both in SE and ML, Felderer and Ramler [42] provides an overview of quality attributes and challenges for QA of MLSs. For example, they state that the term "testing" has different meanings and associations between SE and ML, which is something that this thesis also has the goal of clarifying. The authors define categories of AI components almost identical to the ones used in our mapping, i.e., data, learning program, framework, model, and MLS. However, when characterising AI systems, they consider two more dimensions including process type (isolated vs continuous) and quality characteristics (e.g., software quality, data quality). A set of challenges for QA of AI systems are discussed, some of which we also mention here like the oracle problem and non-functional QA. Comparatively, our thesis focuses less on elaborating challenges and instead provides an overview of existing testing techniques from the literature.

Marijan *et al.* [40] has written a short paper about some of the problems in MLS testing, such as the oracle problem, the large input space and the high cost of white-box testing. For each problem, they give high-level recommendations on which attempted test techniques have been successful in circumventing the issues, and also discuss how traditional SE testing techniques have been adapted to work for ML software. Based on this, there is some overlap with RQ1 in this thesis, but the authors keep the recommendations at a fairly high level and also do not focus on traditional ML.

Fischer *et al.* [43] brings up challenges in the current ML and AI fields, such as trustability and missing/duplicate data. The challenges that the authors bring up have some overlap with what this thesis refers to as goals, but both works also contain goals/challenges that are unique to them. The authors of the paper then also present approaches that are currently being used in the field to address the challenges. Unlike this thesis, they do not present a tool that is meant to aid developers in finding relevant testing techniques for their cases but instead present some concrete ideas that the developers can utilise if they encounter any of the challenges they have mentioned.

Huang *et al.* [44] present a paper focused on the current challenges and solutions in the field of deep learning. While the paper does not propose a toolkit like the one presented in this thesis, it does go over challenges similar to those that have been defined as goals in this thesis. These are goals/challenges such as interpretability, robustness and safety. The solutions discussed are techniques such as adversarial attack testing and coverage criteria, but since there is a focus on deep learning as opposed to traditional ML the coverage criteria are focused more on neuron coverage and surprise coverage, although the MC/DC (Modified Condition/Decision Coverage) criteria from SE appears as well. This is different from the criteria proposed in some of the papers that are part of the mapping in this thesis, as those focus on traditional ML and therefore utilise criteria such as decision tree coverage instead of neuron coverage.

Braiek and Khomh [45] presents a knowledge-seeking paper that aims to discuss the current state of the art of ML testing. The paper goes over both the creation of ML-based software, the problems with testing it and the current popular approaches to solving the problems as well as identifying gaps in the current field of ML testing. While the paper could be seen as making recommendations for testing MLSs, based on the fact that it presents useful techniques for different scenarios, it does not propose any type of framework or toolkit in the same sense as this thesis. Since it is knowledge-seeking the goal is instead to pave the way for future researchers by identifying which areas are in the most need of further study.

Hamada *et al.* [46] and Fujii *et al.* [47] present summaries of the guidelines for ML QA by the QA4AI consortium, which was created as a joint project in the Japanese industry for the purpose of QA for AI/ML. The project has a strong focus on the areas of Generative Systems, Voice User Interfaces, Industrial Processes, and Autonomous driving. The guidelines provided are high-level recommendations for how to achieve a robust, fair and interpretable MLS within the different areas. While the scope of the guidelines is much broader than this paper, it does not propose any toolkit that is similar to the one presented in this paper, nor do they have a focus on traditional ML. Furthermore, the English translation of the original guidelines is currently only available in a machine-translated format, and while it still provides immense value this also means that the translation might not be as good as the original Japanese text.

2.3.2 Papers About Testing Techniques

Masuda *et al.* [48] surveyed the literature for techniques used to evaluate and improve ML applications in terms of their software quality, simultaneously looking for solutions to problems in the field of testing ML software. While fairly similar to our thesis in that they tagged a collection of relevant papers, we provide an evaluation of our toolkit on realistic industrial cases and also include more recent papers as their survey was done in 2018. In their results, they describe their findings for the seven most frequently occurring tags, among which "Deep Learning" and "Fault Localisation" were the topmost tags.

Tao *et al.* [49] review the joint field of AI software testing, which includes testing of AI software and doing testing with AI software. They explain a wide range of existing black-box techniques in the literature for testing AI software, which tests the external behaviour of a system that incorporates AI. Furthermore, they define a set of quality parameters for image recognition, which resemble the testing goals in our mapping as they refer to different attributes of an ML component. Compared to this thesis, the aim of [49] is to outline how AI software testing can be done with existing techniques and provide a case for this on robustness evaluation. Their focus is more on explaining the current difficulties of testing AI systems and providing some examples of techniques out there, not restricted to traditional ML. Our aim on the other hand is to map testing goals like robustness and fairness to existing solutions in the research literature, with a focus on traditional ML, and also to pro-

vide a tool to guide ML engineers in finding appropriate techniques for their context.

Salman Sherin [17] performed a systematic mapping of the ML testing field in 2019, where they classify relevant literature up until January that year. They investigate matters like the proportion of all existing techniques that are applicable for supervised versus unsupervised learning, and the corresponding proportion for function versus non-functional testing, but also which types of testing is being done (e.g., metamorphic, mutation and adversarial testing). Their work is most akin to this thesis out of all related works, however, their scope is on ML as a whole, without restrictions to just traditional ML. Unlike them, we classify papers not only by the type of testing but also by which quality attributes they target. The authors concluded that there was a lack of empirical evidence on the effectiveness of the techniques in general, and specifically that reinforcement learning and testing of non-functional quality are needed. Since many papers have been published on the topic of ML testing since their systematic mapping was done [1], our thesis includes many other works that had not been published at the time.

2.3.3 Other Related Works

There have been other works that aim to facilitate the testing of MLSs. In the position paper by Borg [29], the problem of QA for AI applications is discussed. Several working definitions are presented, for example, to refer to certain kinds of discrepancies between existing and required conditions in an AI system. Borg argues that creating a "testbed" for AI QA should be a top priority in the industry right now. A testbed refers to a place that facilitates technical evaluation by providing a controlled environment, such as a research lab. The author proposes the not yet realised AIQ Meta-testbed, in addition to a mapping between the EU requirements for trustworthy AI and the testing properties defined by Zhang *et al.* [1].

Several QA frameworks have been proposed that seek to provide a language for evaluating MLSs and/or defining metrics for this. Ishikawa [50] propose an ML Quality framework to aid developers in determining the quality of both their MLSs and components. Their contribution is a general template for assessing MLS or component quality, while our work is more concrete, being a mapping of specific papers that utilise different techniques. Nishi *et al.* [51] propose a testing framework for MLSs. They define five principles of QA for such systems. Furthermore, they define seven levels of ML products, e.g., the MLS in the real world or the training data. Based on this, they define four test levels, akin to the concept of "component under test" in our mapping. Some recommended testing strategies are provided like snapshot testing, which spans metamorphic testing, that can help the developer adhere to the five QA principles for each of the different levels of the ML products. However, these recommended testing strategies are general advice, they are not mapped from contextual variables of a system, which is the case for our mapping.

3
Methods

In this chapter, the research method used to map the current research field is presented. After that the methods used to create the toolkit as well as details of how it was evaluated are described. The goal is to provide some insight into why certain methods were chosen, and how they were meant to aid in answering the research questions.

3.1 Mapping of the Current Research Field

To create a good basis for the mapping, a large amount of literature in the fields of QA testing and ML was analysed. To do this within a reasonable time frame, without missing any of the most important pieces of literature, some existing extensive literature reviews, surveys and mappings were used (see [1], [7], [32]). These papers helped provide a better insight into what the current field of ML testing looks like. By using techniques like snowballing [52], even more papers were found that were relevant and that could potentially help to shape the proposed mapping. The full method used to find and filter papers can be seen in Figure 3.1, and is discussed further in the sections below.

A systematic literature review, as described by Kitchenham *et al.* [53], was not conducted for this thesis due to two main reasons. First, the time constraint on this thesis project made it unfeasible to perform a systematic literature review together with the rest of the research that had to be conducted. Second, there already existed some recent literature reviews, as mentioned in the paragraph above, that were similar or larger in scope than what was needed for this thesis. This meant that conducting another one would not provide much value to the research field. That is not to say it would be without merit to conduct a literature review, but it was decided that the time could be better allocated in the creation and evaluation of the toolkit.

Rather, this thesis is inspired by systematic mapping studies, as described by Petersen *et al.* [19]. As a process, it consists of five steps: "Definition of Research Questions", "Conduct Search", "Screening of Papers", "Keywording of Abstracts", and "Data Extraction and Mapping Process". This thesis did not adhere strictly to the guidelines proposed by these authors, but it covers the essential parts of each step. The first step was performed at the beginning of this project, and only differs slightly in the aim of the research questions compared to the examples provided by

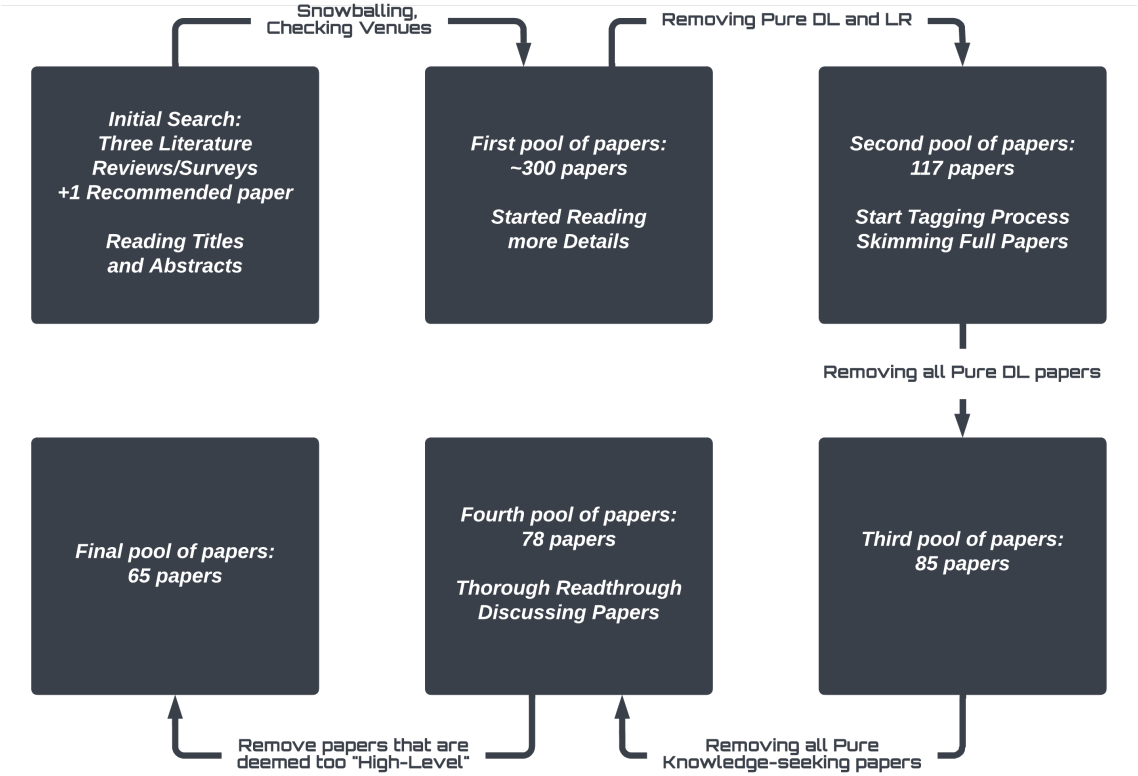


Figure 3.1: A depiction of how the papers were filtered down from the initial pool of over 300 papers (gathered from previous literature reviews/surveys) into the final 65 that were included in the mapping.

the authors. One such example question could be "Which journals cover papers on (insert topic)?". The second step has been performed in the sense that we searched for literature on scientific databases using different search terms, albeit without restricting ourselves to specific and fixed search queries. We experimented with different combinations of the search terms we deemed as important keywords. In the third step the researchers should define inclusion and exclusion criteria for the papers to be part of the study. We describe the corresponding rules we followed when determining if a paper would be included or excluded from the mapping below in Section 3.1.3, without presenting an explicit list of criteria which Petersen *et al.* [19] is a proponent of. The fourth step concerns reading and eliciting keywords from the abstracts of the papers that remain after applying the inclusion/exclusion criteria. This process corresponds closely to what is described in Section 3.1.2. There is a second part to the fourth step where the researcher combines the identified keywords and clusters them to form categories. We performed some additional steps before the fifth step of systematic mapping studies, consisting of further discarding irrelevant papers which is described in Section 3.1.3, and then reading the remaining papers more thoroughly as well as summarising them which is explained in Section 3.1.4. In the final step, the researcher sorts papers based on the aforementioned categories, generates relevant statistics, and analyses the results with a focus on presenting the frequencies of publications in the categories. This step has also been covered by our

research method.

3.1.1 Snowballing and Exploring Relevant Venues

A technique called "snowballing" was used to gather a set of relevant papers for this thesis work [52]. The four papers that were snowballed were the three literature reviews mentioned earlier ([1], [7], [32]), but also [14]. The last paper was provided by the Volvo Cars employee that has acted as an advisor on this project, as an example of paper that was more specific to the context they were looking for. Both backwards and forwards snowballing was performed during the process. Backwards snowballing means that, for a specific paper, the papers in the list of references are collected. This task could have been repeated in turn for the collected papers to gather a bigger set of potentially relevant papers, but this would require too much time in the following stages compared to the time allocated to research in the project. As mentioned, forwards snowballing was also done, which consists of finding papers that cited a specific paper. By utilising Google Scholar [54], the four subject papers were snowballed forward to find any papers that cited them and that were related to MLS testing. The end result was a list of roughly 300 papers, which had relevant titles or abstracts. In Figure 3.1 this is represented by the first pool of papers.

Using the four aforementioned papers as a starting point, their publication venues were used to find additional relevant studies. The conference that provided the most value for the research was IEEE International Conference On Artificial Intelligence Testing (AITest), in which the years 2019-2021 were used to find papers. This provided an additional handful of papers that fit well with the ones that had been collected during the snowballing.

Riccio *et al.* [7] deemed grey literature to be highly relevant for the field of ML testing since it is an emerging field that only recently started to establish itself. They discuss how an increasing number of papers on the subject are published each year. Furthermore, many new papers have not had the time to be officially published in a journal or conference, but can still provide interesting new ideas. Therefore, the decision was made to include some papers from arXiv [55], that were recently published and close to the area of research. These papers were counted among the ones mentioned in an earlier paragraph in this section since many of the papers were found during the snowballing process.

Due to the large number of papers found during the analysis, it was deemed necessary to filter the collected papers down into a smaller collection of highly relevant literature. The approximately 300 papers that were collected during the previously mentioned stages were therefore read in a bit more detail to do a first relevance filtering. The thesis focuses on traditional ML, so any papers that seemed to purely focus on deep learning could be removed immediately. This was also true for any paper that called itself a survey or literature review since these would not provide any concrete solution that could be part of the mapping. As shown in Figure 3.1, this ended up filtering away more than half of the papers. The new, second pool

contained 117 papers that were deemed relevant enough for further examination.

3.1.2 Tagging Papers

Once the initial pool of papers had been collected, the idea was for each paper to be categorised with a set of tags. But before this could be done, these tags had to be defined. Each of the tag categories will be discussed in this section. Most of the tags that were defined are related to either the Context or the Goal of the mapping, but some are also related to the recommendation like the testing technique tags. There were also some tags that were only included to help with the filtering process so that the remaining papers could be highly relevant to the subject. The following paragraphs explain each tag category and give examples of its corresponding tags, formatted in **bold** and *italics* font, respectively.

Important to note before the tags themselves are discussed, is that only concepts that appeared in the collected set of papers would receive tags defining them. For example, this means that there are some ISO 25010 quality attributes [13] that have not been made into tags, since they were simply not used within the collected literature. This was done to make sure that only relevant tags were included, which made the filtering easier to organise and later helped reduce cluttering in the mapping.

First off are the context-related tags, with each of the categories being visible in Table 3.1. One of the most important tags for filtering was **Deep Learning or Traditional ML**, which had two options as indicated by the name, *Deep Learning* and *Traditional ML*. The main purpose of these tags was to help filter away all the papers that were solely about deep learning. This was of course provided they also had no generalisability, which there also exists a tag for that will be discussed in a later paragraph.

Next, there is **Component under test (CUT)**, which has 5 different options: *Data*, *Learning Program*, *Framework*, *Model*, and *MLS*. This category was inspired by the distinction made by Zhang *et al.* [1] between the different CUTs. It refers to which part of the system it is that is being tested and is thus related to the testing context. There were five different options for this tag category. *Data* means that the training data is tested, for example by checking for bias. *Learning Program* means that testing is done on the code/algorithm that is used to create and train a model. *Framework* means that the CUT is a framework used to specify a model or implement the learning program, for example, Scikit-learn [30], PyTorch [56], or Keras [57]. *Model* means that a trained model is tested, like a decision tree or a neural network. *MLS* refers to the testing of a complete system that incorporates one or more ML components, for example, an autonomous vehicle.

Offline/Online Testing was used to make a distinction between *Offline Testing* and *Online Testing*. The definitions of these terms are based on the work of Haq *et al.* [58]. Offline testing is when a component is tested as stand-alone, while online testing refers to a component being tested as part of a system, e.g. an autonomous

Tag Category	Tags
Deep Learning or Traditional ML	Deep Learning
	Traditional ML
Component Under Test	Data
	Learning Program
	Framework
	Model
	MLS
Offline/Online	Offline Testing
	Online Testing
Type of System (Use Case)	NLP
	AD
	Computer Vision
	Medical
	Recommender Systems
	Machine Translation
Type of ML	Supervised Learning
	Semi-supervised Learning
	Unsupervised Learning
	Reinforcement Learning

Table 3.1: The tags used during the literature collection that represent the context of a given paper that proposes a testing solution.

car.

The tag **Type of System (Use Case)** was used for types of MLSs that showed up frequently enough to deserve a tag on their own. These were systems such as *Natural Language Processing* (NLP), *Autonomous Driving* (AD), *Computer Vision*, *Medical*, *Recommender Systems*, and *Machine Translation*. **Type of ML** is meant to reflect which major category of ML the paper concerns. The options that appeared in the reviewed papers were *Supervised Learning*, *Unsupervised Learning*, *Semi-Supervised Learning*, and *Reinforcement Learning*. These four tags however had many more subcategories, which are shown in Table 3.2. These tags were used when it was clear exactly which type of ML task the paper was targeting, such as *Classification* for **Supervised Learning** or *Clustering* for **Unsupervised Learning**. Do note however that *Semi-Supervised Learning* and *Reinforcement Learning* did not have any subcategory tags.

For the **Goal** category, goals in the literature like *Interpretability*, *Correctness*, *Robustness* etc. were identified, as can be seen in Table 3.3. During the research phase, these tags were added as needed, since whenever a new paper was read it might have contained a goal that was not already represented by a tag. Interesting to note is that many papers targeted several goals at once, which means that they also got tagged with multiple goal tags.

Tag Category	Tags
Supervised Learning	Classification
	Regression
	Structured Prediction
	Ranking
Unsupervised Learning	Clustering
	Dimensionality Reduction
Classification	Discriminant Analysis
	SVM
	Nearest Neighbour
	Naive Bayes
	Logistic Regression
	Random Forest
	Decision Tree
	Neural Network
	Gradient-Boosted Tree
	Multilayer Perceptron
Regression	Perceptron
	Linear Regression
	Neural Network
Ranking	SVM
	MartiRank
Clustering	K-Means
	K-Means++
	Spectral Clustering
	Expectation Maximisation - Gaussian Mixture
	Gaussian Mixture
	Hierarchical Clustering - Agglomerative
	Affinity Propagation
	DB-SCAN
	Bisecting K-Means
Dimensionality Reduction	PCA
	Variational Autoencoder

Table 3.2: The tags used during the literature collection that represent the type of ML in focus in a given paper. Together with the tags in Table 3.1, the tags in this table represent the concept of "context" in the mapping.

Tag Category	Tags
Goal	Robustness
	Fairness
	Privacy
	Efficiency
	Security
	Safety
	Avoid Overfitting
	Correctness
	Completeness
	Consistency
	Monotonicity
	Interpretability
	Retraceability
	Model Relevance
	Predictive Performance
	Trustability

Table 3.3: The tags used during the literature collection that represent the testing goal of a given paper that proposes a testing solution.

Testing Type was one of the most important categories since it contains the actual techniques that were proposed and evaluated in the collected papers. The testing techniques that were applied in several papers can be seen in Table 3.4. Most of these are common testing techniques within QA, but there is also a category called *Other type of testing* that is used to tag any paper that did not have a testing technique that fits within any of the more common definitions.
The **Access Level** category specified if the proposed testing solution was black-box, data-box, or white-box, as defined in [7]. These tags reflect the varying degrees of how much of a system that the tests need access to. *Black-box* testing requires access to the input and output of the model. *Data-box* testing requires black-box access extended with access to the model's training and testing data. *White-box* testing requires data-box access extended with access to the model's learning program and parameters. This means that white-box subsumes data-box, which in turn subsumes black-box.

Type of Proposed Solution was meant to indicate which form the solution took. For example, if it could be described as a method, process, approach, strategy, technique, activity, or methodology, then the paper was tagged with *Method*. There were also tags for *Framework*, *Tool*, *Metric*, and *Concept* since they often appeared. Two more tags were added, *Test Adequacy Criteria* and *Metamorphic Relations*, which provide a bit more information than the others. The reason for their inclusion was that even though they are more specific, they still appeared frequently enough for it to be worth creating their own tags.

Tag Category	Tags
Testing Type	Adversarial Attack Testing
	Fuzz Testing
	Boundary Value Testing
	Metamorphic Testing
	Mutation Testing
	Search-based Testing
	Coverage-based Testing
	Combinatorial Testing
	Regression Testing
	Property-based Testing
	Surprise Adequacy Testing
	Differential Testing
	Data Intervention Testing
	Random Testing
	Model-based Testing
	Other type of testing
Access Level	Black-box
	Data-box
	White-box
Type of Proposed Solution	Method
	Framework
	Tool
	Metric
	Concept
	Test Adequacy Criteria
	Metamorphic Relations
Testing Workflow	Test Creation
	Test Design
	Test Execution
	Test Result Analysis
	Test Case Analysis
	Test Input Data Generation
	Test Input Data Preparation
	Test Automation
	Test Oracle Generation
	Runtime Monitoring

Table 3.4: The tags used during the literature collection that represent features of the testing solution proposed in a given paper.

The final category related to the proposed testing solution for any given paper was **Testing Workflow**. The corresponding tags refer to the part of the testing pipeline that the proposal concerns, i.e., which activity or step in the process of testing a system that the proposed solution conforms with. All the **Testing Workflow** tags can be seen in Table 3.4, and most of their purposes are clear from just their names. The one that does need some clarification is *Test Automation*, since this tag is meant to be used together with some of the other tags in the **Testing Workflow** category to also show which part of the workflow it is that is being automated.

The next two categories were created purely for filtering and were therefore only a part of the mapping and not part of the toolkit presented in Chapter 4. Both of the categories and their tags can be seen in Table 3.5. **Aim of paper** was a category that contained two options, either a paper was *Knowledge-seeking* or *Solution-seeking*, as defined by Stol and Fitzgerald [23]. For the creation of the mapping, only solution-seeking papers were relevant which meant that this tag could help filter all of the collected papers to remove knowledge-seeking ones.

Tag Category	Tags
Aim of Paper	Solution-seeking
	Knowledge-seeking
Generalisability	Potentially Generalisable Solution
	General Solution

Table 3.5: A few utility tags that were used during the filtering stage.

The **Generalisability** category was meant to sort papers into three groups. The first group would be *General Solution* representing testing solutions that could be applied to any type of ML model, although limitations like "only for classification" were allowed. The second group was *Potentially Generalisable Solution*, and it represented solutions that seemed possible to modify into general solutions, even if they were not described as such by the paper. Being black-box was one typical trait and sign of a potentially generalisable testing technique since these techniques rely solely on input and output. The final group, which was represented by the lack of either of the previous two tags, was meant for all papers that had techniques that were not generalisable. This was often the case with papers that used a white-box technique that was directly dependent on the implementation of a model. This separation was supposed to help filter away papers exclusively related to deep learning, the process of which is described next.

3.1.3 Filtering Papers

After all the papers had been tagged it became easier to do another round of relevance filtering. The most impactful filtering was the removal of all papers tagged with *Deep Learning*, since the mapping is focused on Traditional ML. However, it

was important to keep any *Deep Learning* papers that were deemed generalisable to all types of ML. These were for example papers that were shown to work for DL, but that also did not depend on any attributes that are unique to DL like the layers and weights in deep neural networks. So the result was that this filtering was done such that all papers tagged with *Traditional ML* OR (*Deep Learning* AND (*General Solution* OR *Potentially Generalisable Solution*)) were kept. After this filter had been applied there were 85 papers left, as can be seen in Figure 3.1 by looking at the third pool of papers.

The next filtering was done on the **Aim of paper** category, where the decision was made to only keep the papers tagged with *Solution-seeking*. This was done because the toolkit is meant to suggest a testing technique to a developer in order to help them achieve their testing goal in a given context. Knowledge-seeking papers would not provide anything concrete that could be used in this scenario. The knowledge-seeking papers were not removed completely however, and were instead used to support and motivate decisions throughout this thesis. When this filter had been applied there remained 78 papers, as can be seen in Figure 3.1 by looking at the fourth pool of papers.

3.1.4 Summarising papers

After the filtering, the remaining papers were read again in more detail, so that they could be summarised. The purpose of this was to reduce the time spent rereading papers when details had to be discussed, but also to be able to provide an overview of the papers included in the mapping as part of Section 4.2 of this paper. These summaries also function as the basis of the Paper Summaries Document that is presented in 4.3.2.

During the summarising process, a few papers were found to not be as relevant for the project as expected, because they had been mislabelled while the papers were previously skimmed. This was either because they did not really seem as generalisable as was previously believed, and were therefore only meant for deep learning, or because they turned out to only have high-level descriptions of their ideas and were therefore more suitable as a related work to this thesis. Thus, some more papers were removed and in the end there were 65 papers left that could be added to the mapping. This has also been represented in Figure 3.1, which shows the final pool containing the papers.

3.2 Creating the Toolkit

In this section, the process that was followed to create the toolkit is described. It begins by describing the tools used, then moves on to explain how the basis for the toolkit was designed, and then finally how items were added to it. An item in this case refers to a tuple of a goal, a context, and a paper that proposes a testing solution.

3.2.1 Early Considerations

During the design phase of the toolkit, a few alternative implementations were considered. One option was to create a decision tree consisting of context- and goal-related questions in the non-leaf nodes. The developer can then answer these questions in order to arrive at the leaf nodes. A leaf node of this tree would then represent a relevant paper for the developer's unique context and QA goals. This idea was abandoned due to the complexity of its structure and the difficulty of maintaining it. Once a tree like this is constructed, it is hard to make changes to it since changing one node could have a cascading effect through the tree such that potentially many other nodes need to change too. Another possible implementation option was to create a Wizard application [59], that would pose questions to the user and then present relevant papers depending on the answers. In the end, it was decided to use Microsoft Lists, mostly because it alleviated the need to write source code for a program. Therefore, this was instrumental in allowing for more time to be spent in the research phase which allowed for a better curated list of literature in the toolkit.

3.2.2 Tools Used

The main tool of the toolkit was created using Microsoft Lists [60]. This was partly because it was easy to work with and it provided some very useful features to build and use the toolkit while also allowing for work collaboratively online. The main tool was therefore implemented as a spreadsheet with the rows representing the different tagged papers in the mapping while the columns are the different tag categories. A specific column could for example represent the testing goal or a context variable like **ML task**. An important feature of Microsoft Lists in building the main tool was the possibility to format the columns. To allow multiple goals, for example, to be covered by the same paper, it was possible to define a set of tags for a specific column and apply several of those in the same cell. Another important feature of Microsoft Lists is that it has an easy way of filtering the items based on tags. This meant that the tags could be utilised similarly to how they were during the research phase described in Section 3.1.

LucidChart [61] is another tool that was used, which is a program designed for creating flowcharts. There are many similar tools, but LucidChart was selected because we had previous experience with it and it would therefore be the most efficient one to use.

3.2.3 Setting up the Main Tool

Before items could be added to the main tool there was a need to determine the possible values for all categories in the mapping. As explained in the section above, it was possible to assign one or more tags as the value of a category, based on a set of user-defined tags. Since most of the papers were already fully tagged in the research phase, the decision was made to continue using all the relevant tags from that phase. These tags were related to the goal, context, or recommended testing

technique in the mapping. During this process, some tags were changed slightly or removed while some new tags were also added mostly to improve the comprehensibility of the mapping. The definitions of the goals can be found in Section 4.1, and all other tags are defined in the toolkit that can be found in the project Github repository[1].

The main changes to the tags were the removal of the tags *Supervised Learning* and *Unsupervised Learning*, which were replaced by a tag category called **ML Task**. This category had tags such as *Classification* and *Regression* which by themselves imply if it is supervised or unsupervised learning, which meant that there was no need to include the other tags. Four other categories were removed as well, **Deep Learning or Traditional ML**, **Type of Proposed Solution**, **Generalisability**, and **Aim of paper**. These were not included in the toolkit because they did not provide much additional value, and would only make it more crowded in the main tool. They had already served their purpose during the filtering phase, so the decision was made to not include them in the finalised version of the toolkit. There were also some categories that had some new options added and old ones modified, such as **Goal**, **Testing Type**, and **Testing Workflow**. These changes were made after reading the papers more thoroughly, at which point it became clear that there had been misunderstandings based on the first read-through that required a different set of tags. Tags that were no longer used for any of the papers were also not included in the mapping anymore. The final change was that the category **Type of System (Use Case)** became **Limitations**, and had some additional new tags added that fit with the context given by the new name.

Besides the main tool, an additional document was created to aid users of the toolkit. It does so by providing clear definitions and references for the meaning of all the different tags in the mapping, which was titled the *Definitions Document*. Another document named the *Goal Identification Document* was created to facilitate the use of the toolkit, and it contains a list of questions that developers can ask themselves to identify their testing goal. This second document is mostly targeting developers that do not have a clear plan or problem that they need to test for. It is meant to aid them in finding the goals that will have the most impact on the quality of their systems.

3.2.4 Building Process

Once the foundation of the main tool had been created it needed to be populated with list items. This was done by reviewing each of the collected papers in the mapping one final time, by looking at both the tags and the summaries as well as rereading the papers if there were any uncertainties. This process was done in meetings were all authors of this thesis participated, since it was deemed important to have a final discussion about the papers before they were put into the main tool. All 65 papers from the filtering/summarising phase were added to the main tool, although some of them got their tags adjusted further after discussion due to initial

[1] `https://github.com/alexandersimolaIT/QA-Mapping-for-ML`

mislabelling. The final definitions used for the goal tags in the toolkit can be found in Section 4.1. These can also be found in the project repository[2] alongside the rest of the tag definitions.

When the main tool was complete, a document containing a set of tree diagrams was created. These diagrams were meant to give an overview of what the toolkit can provide, without going into too much detail which might make it more difficult to comprehend them. They were created with the previously mentioned tool LucidChart [61] by starting with a component under test (CUT) as the root node, and hence there were five diagrams in total, one for each CUT tag (i.e., *Data*, *Learning Program*, *Framework*, *Model*, and *MLS*). In each diagram, a node was added for each possible value of the **Goal** category, for example, *Group Fairness* and *Interpretability*. If a goal type was never paired with a specific CUT in any of the tagged papers, then it was not included in the tree. The next step was to connect nodes for testing techniques from the goal nodes. For each tagged paper, all of the **CUT** and **Goal** tags were marked. Any testing technique tags that were included on the same tagged paper were then added to the trees as leaf nodes to all pairwise combinations of the previously marked **CUT** and **Goal** tags. For example, an item with the goal tags *Correctness* and *Completeness*, CUT tags *Model* and *Data*, and the testing tag *Fuzz Testing* would mean that there would be a *Fuzz Testing* leaf node added under the *Correctness* and *Completeness* nodes in both the tree with the *Model* root node and *Data* root node. The resulting tree diagrams can be seen in Section 4.3.2.

3.3 Evaluating the Toolkit

This section describes how a judgement study was conducted in order to examine the effectiveness of the toolkit. It details how the realistic project cases were collected, how the interviews were conducted, and how the gathered data and feedback was organised. The timeline of the evaluation can be seen in Figure 3.2.

A judgement study is a research strategy with several use cases, the one applied in this thesis being to collect data from participants about some topic under investigation [23]. The main focus is on the participants' responses to the stimuli provided by the researchers, rather than to draw general conclusions about the participants themselves. In this thesis, the stimuli is a set of interview questions and the responses are the answers to these. An important characteristic of a judgement study is that it is conducted with experts on the topic, so that they can provide valid and useful feedback. For the evaluation of the toolkit proposed in this thesis, a judgement study was deemed a good choice as it was not crucial to collect a large sample size. Considering the short period of time that this evaluation study would occupy, and the fact that it could be hard to find qualified participants to interview, it was deemed unreasonable to collect a very large data set. Multiple types of judgement studies have been recognised in the literature, and the one conducted for this thesis is called the "evaluation study". It is about letting participants judge an object, in

[2] `https://github.com/alexandersimolaIT/QA-Mapping-for-ML`

this case the toolkit, so that the researchers can evaluate it [23].

Before the decision was made to do a judgement study, other evaluation methods were considered too. One such consideration was to conduct an experimental simulation, which is a research strategy that seeks to replicate real conditions to a limited degree [23]. In such a study, one could have focused more on the tags and seek to find patterns and trends in the chosen tags in order to see correlations between attributes of the participants and the choice of tags. However, it became evident that there would not be enough participants to draw any statistical conclusions. This was one of the main motivations as to why a judgement study was a better fit, since it could still benefit greatly from even just a few participants' feedback.

Another option could have been to let Volvo Cars use the toolkit on one of their current projects, to identify some testing solution to implement for that project. The effectiveness of this implementation would then be treated as a proxy measure of the toolkit's ability to recommend useful and relevant solutions. This idea was discarded because it would be limited to a sample size of one implementation. In order to draw any reliable statistical conclusions, it would require a large enough sample of techniques implemented, which would be unfeasible for this thesis project.

3.3.1 Gathering Realistic Test Cases

To make the evaluation more realistic and comprehensive, two problem cases were collected from Volvo Cars. These two cases are descriptions of real industrial projects/problems and were used in the study. Each case was explained by an employee at the company, and was then rephrased to anonymise and generalise the cases in order to hide sensitive information and to make it less specific to Volvo Cars, while preserving the main essence of them. This process is represented by the first two steps in Figure 3.2. The final version of these case descriptions can be found in Appendix A. The reason for putting a lot of effort into gathering realistic test cases is that it provides more insight into how well the toolkit actually functions in practice compared to applying it to completely fabricated cases. It also helps avoid bias that might be introduced if the cases were created by us, since these cases could potentially be designed to better fit the toolkit compared to the real cases.

3.3.2 The Participants

A request was given to Volvo Cars to get a group of participants in the study, with the requirement that they should be experts within the ML domain. This request is represented by the third item in Figure 3.2. Volvo Cars obliged and recommended around 14 employees, of which 8 had time to partake in the study. These participants were recommended because they had a background in data science and most were also knowledgeable in the specific area of ML combined with testing. They also had a wide variety of roles, such as ML engineers, data scientists, and product owners. The added benefit of this was that they were also used to reading research

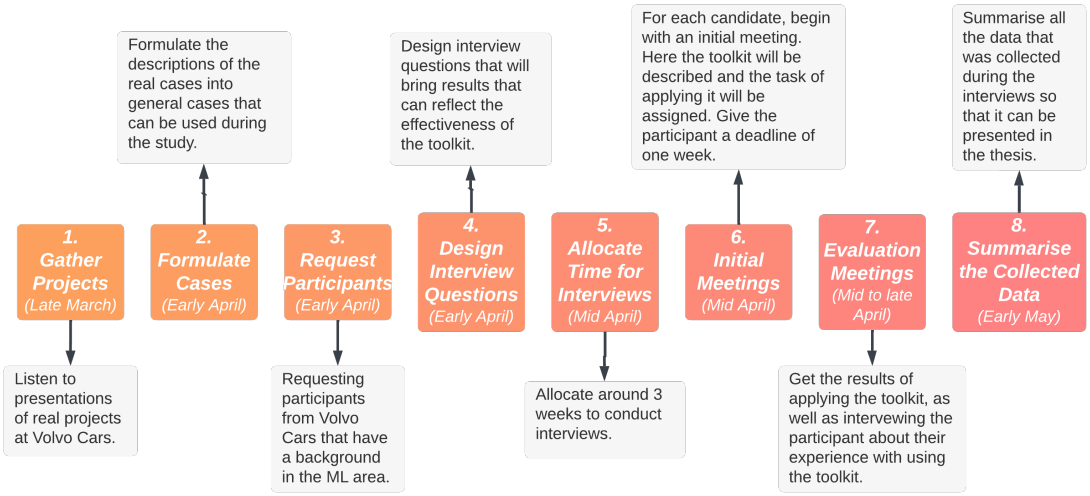


Figure 3.2: A timeline of the entire process of the judgement study, including how it was set up. The timeline is divided in eight steps.

literature, which was one of the areas where the tool could save them time. The reason for this is that it could provide them with a curated list of literature instead of them having to search it up and evaluate the quality of the literature themselves.

With this background in ML we deemed that the participants would be capable of providing valuable feedback on the toolkit. Their feedback was therefore valued highly. If several of the participants had similar input on a certain aspect, then that was an even stronger indication that it was something to keep in mind.

Another important consideration was how much time the participants had available to actually partake in the study. At first the whole experiment was thought to be possible to conduct in a single morning or afternoon (4-5 hours), but it was not possible due to the schedules of the participants. To better align with their available time they were given a week to learn and apply the toolkit, so that they could spend the time as they saw fit. The deadline also had some flexibility, as it was important that the results were not rushed. However, giving the participants more than one week to use the toolkit initially was not seen as viable. This was because a few participants admitted that they would just get started later if given more time. In the end, they always had their normal work to do.

Some participants stated that they would not have time to apply the toolkit to both cases, no matter how much time they were given. Those that could only complete one case were told to do case 1. Getting their input on one of the cases was still seen as important. The alternative would have been to get no feedback at all, which would have reduced the participants from 8 to 5. Furthermore, the cases were mostly used as a way for the participants to get to know the toolkit. This means that they were still able to give feedback on their user experience with the toolkit even with only having solved one case.

3.3.3 The Design of the Study

The purpose of the study was to determine the effectiveness of the main tool and the support documents when applied to the aforementioned use cases. The effectiveness was measured through attributes such as the relevance and quality of the recommended papers, as well as how helpful and intuitive the tool is in guiding the user to the results. With this study, the goal was not to be able to draw statistically generalisable conclusions, which would require a larger sample size than was feasible to collect during this project. Rather, the goal was to collect feedback specifically from ML professionals. To be able to gather information regarding the effectiveness, a list of questions was designed where each question corresponded to an important attribute that was to be measured. The steps described in this section are represented by steps 4-7 depicted in Figure 3.2.

Each participant first took part in an information session where they got introduced to the concept and purpose of the toolkit. They were not given a full tutorial on how to use the toolkit, since learning how to use it on their own was a part of the study. Instead, they were directed to the README Document as well as the Toolkit Guide Document. Once they had an understanding of the toolkit they were tasked to apply it to the two project cases, and a deadline of a week (with an evaluation meeting scheduled at that point). During the duration of this study they were asked not to share any information regarding the toolkit and their results with any of their colleagues, to prevent any validity issues the could arise from such discussions.

The following information was collected from each participant:

- Time spent on learning how to use the toolkit
- Time spent on solving Case 1/2
- Tags and motivations for Case 1/2
- The resulting paper(s) that the toolkit recommended for Case 1/2

Additionally, they were told to take notes of their experience with the toolkit, to aid them in remembering information that would be important during the interview section of the evaluation meeting.

The evaluation meeting was split into three main parts, one for each case and then a separate interview about the user experience and other feedback. During the parts for case 1 and case 2, information was collected in accordance with the list above, so that it could be used to compare the results of the participants. Comparing the results of the different participants could provide some interesting insight into if most developers would argue for the same types of goals and context given the same project description.

To measure the difficulty of utilising the toolkit, the time spent learning the toolkit and applying it to each case, were recorded. This is used to help evaluate if it takes long to understand the toolkit, as well as a sort of check of how long the participant actually spent with the toolkit (less time might mean less accurate results).

Collecting the duration is meant to help mitigate some validity issues that might occur if certain test subjects could not spend as much time as they needed to fully understand the toolkit. The duration measurement then gives some insight into how the usage of the toolkit is affected by more constrained time limits.

Then there was also the final result in the form of a recommended paper, that was selected by the participant to be the most fitting for a case. This paper is used to create some discussion with the participant regarding the sensibility of the suggestion as well as how difficult implementation would be in general with these types of suggestions.

The final part of the evaluation meeting was an interview where questions regarding the user experience with the toolkit were asked. Since all the participants had previous experiences with software engineering tools, the interviews are conducted to gauge their opinions of the toolkit and its intuitiveness as well as performance. The goal was to figure out if they felt that the tool was easy to get into and learn, while also providing results that they deemed to be reasonable.

3.3.4 Organising Collected Information

Besides taking notes of what each test subject said during the interviews, it was also recorded and transcribed to make it possible to verify the information in the notes. The information from each interview was then entered into a form, created using Google Forms [62], to store the information using a common format. Saving the information like this also made it possible to get a better overview of the data from all the interviews through the visualisations that Google Forms provides. This was also how the whole interviews were summarised, as described in step 8 in Figure 3.2.

Numeric data, such as ratings, were also put into spreadsheets since this was an efficient way of storing as well as visualising the data. Most of the graphs and diagrams in Chapter 4 were created from the data stored in these sheets.

4
Results

This chapter is divided into four major sections. The first section presents the goal definitions and is meant to be the answer to RQ1. After that comes the section that presents summaries of all the papers that were part of the final pool of papers in the mapping and therefore part of the toolkit. This builds into the third section, where the toolkit specified in RQ2 is presented. The final section showcases the results of the study that took place at Volvo Cars to evaluate the toolkit, which are used as a basis to motivate the answer to RQ3.

4.1 Goal Definitions

This section presents the finalised definitions of all the QA goals covered by the mapping. The definitions for all of the other tags can be found in the Definitions Document on the project Github[1]. Important to note is that the Goal tags in the actual toolkit are slightly different from the definitions below since the tags in the toolkit were written to be more easily digestible while the tags in this thesis had no such limitations.

Adversarial Robustness
Adversarial robustness, as discussed by Chen [63], refers to the ability of a model or MLS to withstand adversarial attacks. The system should be impervious to certain inputs that have been tailored to exploit its weaknesses. Testing for adversarial robustness involves identifying these weaknesses. Some examples of adversarial input could be outliers or invalid input. Goel *et al.* [64] identify four categories of existing testing tools for checking robustness. Our definition of adversarial robustness closely matches the robustness checked for by tools in their "Attack" category. Note that there are several other definitions of adversarial robustness, e.g., Katz *et al.* [65] define it partially as local and global adversarial robustness. Those definitions merge the definitions of robustness in this thesis, as small perturbations are regarded as particularly challenging for DNNs to handle, hence noise can be an adversary test case for such models.

Noise Robustness
This refers to the ability of a model or MLS to handle noisy input. Given a test input whose output is known, a perturbation of the input should yield the same output. For example, an image classified as a pedestrian should still be classified

[1] `https://github.com/alexandersimolaIT/QA-Mapping-for-ML`

as such even if the picture is altered slightly. This definition was taken from Zhang *et al.* [1], however, they refer to it purely as "robustness". We distinguish it from adversarial robustness by calling it noise robustness, since noise is often manifested as small perturbations to "clean" input. Based on the categories of testing tools from Goel *et al.* [64], the "Transformations" category best fits our definition of noise robustness. With this category, they give the example of checking that the predicted meaning of a sentence by a text classifier is invariant to the transformation of changing a word to a synonym.

Group Fairness
Based on Zhang *et al.* [1], a model/MLS exhibits group fairness if two groups that only differ significantly by some protected attribute[1] have the same probability of a decision outcome. Group Fairness differs from the other two types of fairness in the sense that it is focused on making sure that large groups that only differ by protected attributes are treated similarly, and therefore has no impact on the individual level. This means that there can still exist Individual and Counterfactual Fairness even when Group Fairness has been achieved.

Individual Fairness
The definition by Dwork *et al.* [66] states that a model/MLS exhibits individual fairness if two individuals that only differ significantly by some protected attribute have the same probability of a decision outcome from the model/MLS. Important to note for distinguishing the difference between Individual Fairness and Counterfactual Fairness is that Individual Fairness is based on calculating how similar two individuals are when protected attributes are not taken into account. If they are similar enough, then the model should give them the same output or else it has an unwanted bias based on a protected attribute.

Counterfactual Fairness
Kusner *et al.* [67] define a model as exhibiting counterfactual fairness for a specific individual if, given the output of that individual, the model yields the same output if the value of a protected attribute is switched. For example, let gender be a protected attribute of a person. Then, for a model/MLS to be counterfactually fair, it should not return different decisions for a person if, theoretically, they switch gender from female to male while fixing all the other features.

Interpretability
Based on Zhang *et al.* [1], interpretability is the degree to which a model/MLS can be understood. They define the concepts of "transparency" and "post hoc explanations" as different aspects of interpretability. Transparency is the understandability of a model's internal mechanism. The concept of post hoc explanations refers to explanations of how an input lead to a certain output or decision outcome from the model/MLS. Interpretability encapsulates both of these concepts.

[1] A *protected attribute* refers to a trait that should be protected from discrimination, for example gender, income, ethnicity.

Safety
Based on the ISO 25010 definition [13], an MLS exhibits safety if the probability of a system failure is low. A decrease in the risk of failure, which can lead to loss of resources or life, means an increase in the safety of the system.

Privacy
Refers to a more traditional kind of privacy than, for example, differential privacy [68]. Here, privacy is a guarantee that the identity of individual people can not be identified in the data, or by the model/MLS. If this is not the case, then the data should be stored securely.

Security
Based on ISO 25010 [13], security refers to the enforcement of appropriate access levels for the users of the system, in terms of what data is available to a given user. To clarify, this is meant to prevent unnecessary weaknesses in a system that adversaries could take advantage of.

Correctness The term correctness is used in a wide array of ways in the literature, but the decision was made to focus on two distinct correctness definitions. The first is based on the definition of functional correctness in ISO 25010 [13]. This definition states that for a given task the system should be within some bound of precision, which for a model could be that the developers are trying to guarantee that the accuracy is at least higher than some minimum value for a specific task. The other definition used is that correctness is the absence of bugs. Any paper dealing with finding and correcting bugs is therefore by this definition trying to improve the correctness of the implementation.

Completeness
Based on functional completeness from ISO 25010 [13], completeness is the degree to which a component fulfils its set of requirements. For example, if the component under test is a model, and its requirements state a few classes of input that it should be able to handle sufficiently well, then the completeness of the model increases the more input classes it can handle. If the CUT is the training data, and the requirements state that a set of scenarios should be represented in the data, then the completeness of the data increases as more and more scenarios are represented in the data.

Monotonicity
The definition of monotonicity in this paper is based on the definition by Sharma and Wehrheim [69]. Essentially, having monotonicity means that variables that are directly correlated in the input and the output should always have this correlation when a prediction is made. If an input has a positive correlation to the output then an increase in the input value should always lead to an increase in the predicted output, given that no other input values change as well. Testing a model's monotonicity is useful when the developers know that two or more variables have a correlation, since it helps guarantee that the model is aware of these relationships

and can make more accurate predictions.

Predictive Performance
This term was used to summarise all the metrics related directly to a model's predictive performance, such as accuracy, recall, precision, F1-score, specificity etc. Predictive performance was not a focus in this thesis, but it was useful to still have defined since some of the literature focused on Predictive Performance as well as quality goals at the same time.

Efficiency
In this thesis, the definition of efficiency focuses more on making test data efficient rather than making entire models more efficient, even though the first has an impact on the second. The reason for this definition is that it was the way efficiency was mostly used in the literature [70]. Making test data more efficient can be things like removing redundant features, finding smaller test cases that can trigger the same failures, or fixing faults in the data.

4.2 Research Findings on Machine Learning Testing

In this section, an overview of all of the 65 collected papers is presented. This summarises different concepts and groups similar papers together to explain how their methods were used to achieve their goals. The main split between the sections below is which **Component Under Test (CUT)** it is that the papers are mainly focusing on. The distribution among the different CUTs can be seen in Figure 4.1. The full summary of each paper in the mapping is available in the Paper Summaries Document as a part of the full toolkit on GitHub[2].

4.2.1 Data

In this section, all the papers that focus on how to test the Data CUT are presented. These papers conduct testing directly on the data for reasons such as being able to filter out data that is faulty or that contains a bias. If used, this data would lead to the model learning incorrect behaviours, which would reduce model quality. In Figure 4.2 the distribution of goals among the papers with *Data* as the CUT can be seen. The figure shows that there is no goal that is dominating the papers, but *Correctness* is in the lead followed by *Noise Robustness*, *Interpretability*, *Completeness*, *Predictive Performance*, and *Efficiency*.

For papers focused on *Correctness*, this often comes down to removing data bugs or finding other faults in the data. The BoostClean paper by Krishnan *et al.* [71] introduces a tool for this exact purpose that allows a developer to automatically identify common data faults, so that they can instead focus their efforts on fixing

[2] `https://github.com/alexandersimolaIT/QA-Mapping-for-ML`

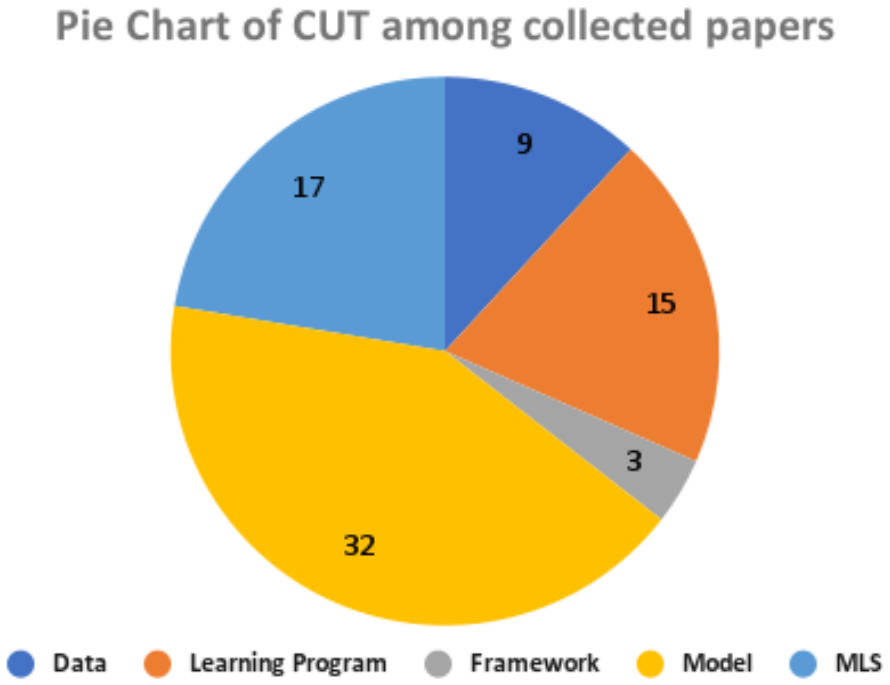


Figure 4.1: Distribution of Component Under Test (CUT) among the different papers in the mapping. Some papers are part of several CUTs, so the total sum of the slices (76) is larger than the total number of papers (65).

domain-specific bugs. The Data Linter tool by Hynes *et al.* [70] fulfils a similar purpose, by giving the developers recommendations for how they can transform their data features to make it easier for models to learn from them. Breck *et al.* [72] on the other hand propose a tactic similar to unit testing, but instead of testing the code, they are testing the data. This idea is meant to be integrated into a pipeline so that when new data arrives it will be validated against a specified schema to help determine its correctness. Similarly, Schelter *et al.* [73] propose an API which developers can use to declare unit tests for the data that can then be assessed by different quality metrics to determine the quality of the data. They focus on the continuous evaluation of these quality metrics, so they provide metrics that can be computed incrementally with access to a small subset of the complete data. Breck *et al.* [74] present a list of different tests for the data, for example, to verify feature expectations in the data by encoding them in a schema.

One aspect of ML testing that is important is that the model should be able to maintain its quality and performance even on out-of-sample datasets. However, if the test datasets are too similar to the training dataset the developers might get the impression that the models are more robust than they actually are. Cabitza *et al.* [75] try to remedy this problem by providing recommendations for how to conduct external validation, based on a study of many different external validation methods. Lanus *et al.* [76] attempt to aid the developers by creating metrics for data similarity so that two datasets can be compared before any testing is done. This means that a

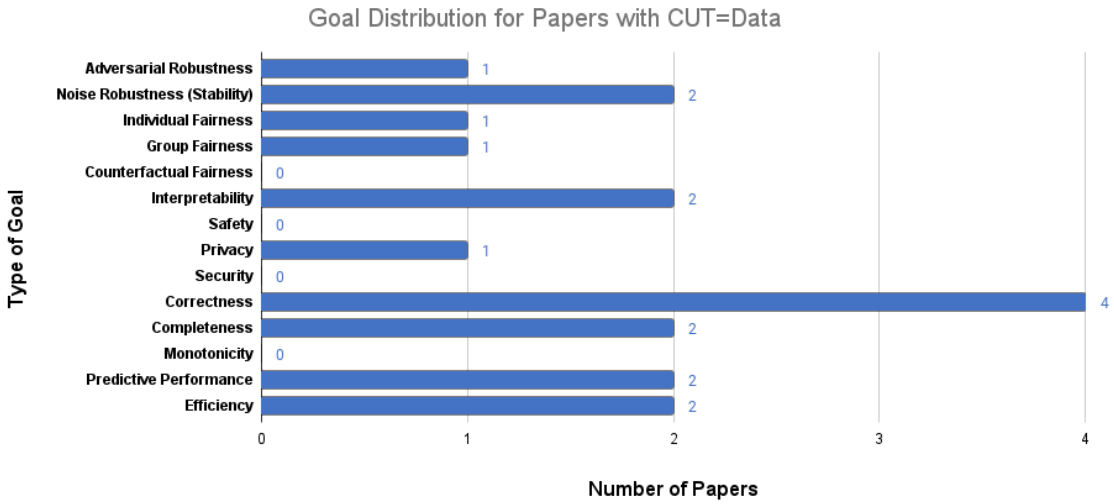


Figure 4.2: Distribution of Goals between the papers that had *Data* as the Component Under Test (CUT). Papers could have several goals at once, so the total number of values for the goals (18) is larger than the number of papers (9).

training dataset can be compared to the testing dataset to make sure that they are not too similar, since if they were, the external validation would not provide much value.

There are also some other approaches for validating the data. Udeshi *et al.* [77] present a tool that is meant to help the developer improve the fairness of their datasets by finding discriminatory inputs that can then be added to the dataset. If a model is then retrained with this dataset it will then hopefully have less bias than if it was trained in the original dataset. Ackerman *et al.* [78] on the other hand, focus on what they refer to as *data drift* which is the gradual change of the data over time. This can lead to a model's performance worsening over time, so the authors propose a new concept called *auditors* that is meant to help identify the data drift so that it can be handled before it is too late.

4.2.2 Learning Program

This section presents summarising information about the papers that at least partially targeted testing of the learning program. The specific subject ranges from finding bugs in the code that implements the training algorithm to making the code as efficient as possible. *Correctness* was the most popular goal to test for, as can be seen in Figure 4.3, although it should be noted that many of the papers have a focus on locating and fixing bugs, and not the functional correctness defined by ISO 25010 [13].

Quite a few of the collected papers are dealing with the oracle problem that is ever-present in the area of software engineering, as discussed in Section 2.2. The most popular approach to remedy the oracle problem among the papers with the

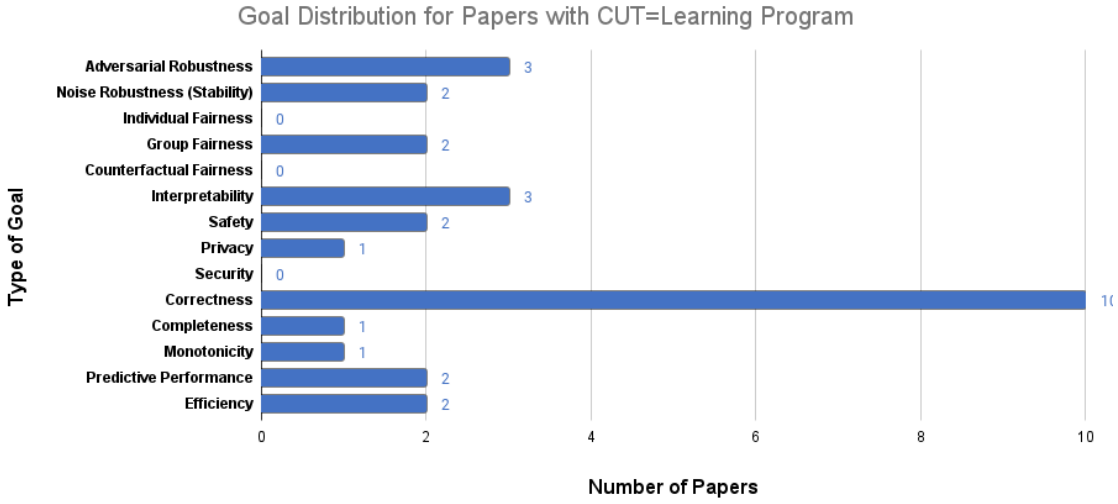


Figure 4.3: Distribution of Goals between the papers that had *Learning Program* as the Component Under Test (CUT). Papers could have several goals at once, so the total number of values for the goals (27) is larger than the number of papers (15).

Learning Program CUT was metamorphic testing [79], [80], [81], [82], [83], but other approaches for dealing with the problem do exist as well [84]. For metamorphic testing of the ML programs the authors first define metamorphic relationships (MR), which state how a change to the input, by applying a function, will change the output. The papers propose different MRs that can affect their ML algorithms such as renaming classes [83] or removing labels [79]. Some also apply mutation testing to help validate their MRs [81], [82].

Srisakaokul *et al.* [84] on the other hand introduce the concept of "multiple implementation" testing to tackle the oracle problem. In this approach, they utilise different learning programs and frameworks as pseudo-oracles to each other, which is done by letting each of the combinations make a prediction. The most popular choice among the combinations is then deemed correct. Imagine it as a majority vote, and any program that failed to make the most popular prediction likely has a fault in either the learning program or the framework. This approach had a much lower false-positive rate than metamorphic testing according to the authors.

In the collected papers there was a wide variety of other ideas for how to test the learning program, and these ideas are discussed in the following paragraphs. The first idea is presented by Breck *et al.* [72] and comes in the form of verifying input data against a schema and fuzz testing to help identify hidden assumptions made in the learning program. Together these actions are meant to be able to deal with bad data having been fed to the ML pipeline according to the authors. Herbold and Haar [85] on the other hand propose smoke testing of ML programs, in which the code is tested through inputs that are meant to try to crash it. Any input that does not result in a failure passes, but if the program does fail then a weakness has been

found. The smoke tests rely on the concepts of combinatorial testing, boundary value analysis, and equivalence class analysis. Another framework meant to help with debugging programs is the Storm framework by Dutta *et al.* [86]. When a program fails, Storm is able to generate a smaller version of the same arguments and program that can trigger the same failure. This can help developers both optimise and debug their programs.

There are also papers that propose statistical hypothesis testing methods for testing learning programs. Ramanathan *et al.* [87] utilise the testing to determine if a test case is able to tell the difference between a faulty and correct implementation of an ML algorithm. For example, detecting a bit-flip in the learning program for K-means Clustering. Similarly, Yin *et al.* [88] use statistical hypothesis testing for a wide array of clustering implementations to measure the variation of its output, which can help determine the correctness of the implementation.

Schaul *et al.* [89] propose a framework for unit testing to help test the optimisation algorithm in the learning program. According to the authors, it is meant to be run alongside other types of testing, since the technique helps with validating robustness but not accuracy overall. Some examples of what the unit tests are designed to test are metrics such as curvature scales and various noise conditions.

Saranti *et al.* [90] propose using property-based testing for algorithms that require a large amount of data like expectation-maximisation. The reason is that property-based testing is excellent for generating large amounts of data in a resource-efficient way, as long as the developers know the properties that the algorithm should have.

A generic approach to testing the implementation of an ML learning program is proposed by Ahmed *et al.* [91]. It is similar to equivalence class testing in that it identifies a partition, called the polyhedral region, of the input space that should produce the same output. The authors evaluate the technique on the lasso regression algorithm, stating that it manages to kill 98% of mutants introduced. They claim that testing by polyhedral region is not specific to the lasso algorithm, but many others where the model is sparse (with few features).

Breck *et al.* [74] propose a wide array of assertion tests across the entire ML pipeline that can then be calculated into an ML Test Score. For the learning program specifically, they want the developers to assert that the training should be reproducible and to assure the correctness of the learning program, both from how it uses the framework and the algorithmic correctness itself.

4.2.3 Framework

The *Framework* CUT was the least popular research field among the collected papers, with only three out of the 65 papers being about the subject. A reason for this could be that testing the framework is the type of testing least connected to ML testing out of all the CUTs, since it is about testing the framework that a developer

uses to create their *Learning Program*. As can be seen in Figure 4.4, the only goal that was a target for the framework was *Correctness* with no other goal having been used even once. Also worth noting is that all papers that had *Framework* as a CUT also had *Learning Program* as a CUT since there was no paper that only focused on the framework. Srisakaokul *et al.* [84], Yin *et al.* [88], and Xie *et al.* [81] all test the *Learning Program*, but while doing so also compares the same learning program implemented using different frameworks. This was meant to reveal faults in the frameworks themselves, since if the same learning program results in a worse model when using a different framework there might be some issues with it.

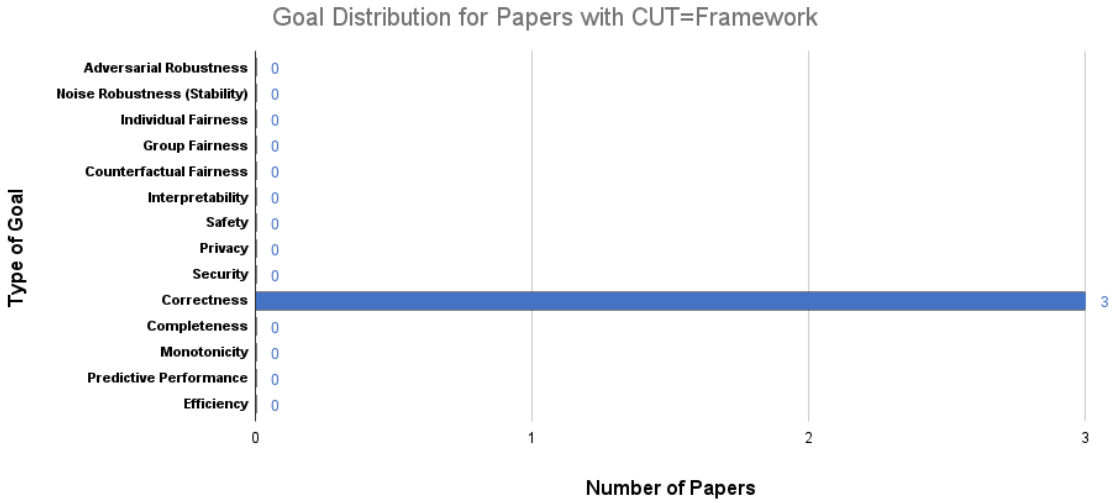


Figure 4.4: Distribution of Goals between the papers that had *Framework* as the Component Under Test (CUT).

4.2.4 Model

The *Model* CUT was by far the most used in the literature gathered for the mapping, with a wide array of testing techniques being applied directly to the models. robustness, both noise and adversarial, and fairness, both individual and group, as defined in Section 4.1 were the most popular goals to test for, with robustness being slightly more common as can be seen in Figure 4.5.

Robustness:
A robust model is a model that can handle either data that is meant to exploit its weaknesses, when talking about adversarial robustness, or that can perform almost just as well on out-of-sample data as on the training data, when talking about noise robustness. A common approach of testing for adversarial robustness in the literature collected for the mapping is to use adversarial attack testing [92]–[94]. Bartolo *et al.* [92] propose using another model that can take a small dataset of known adversarial input, and with this generate a much larger sample of adversarial inputs that can be used to test the robustness of the model. Yang and Xie [93] on the other hand have an idea for how to test models that are meant to be able to detect

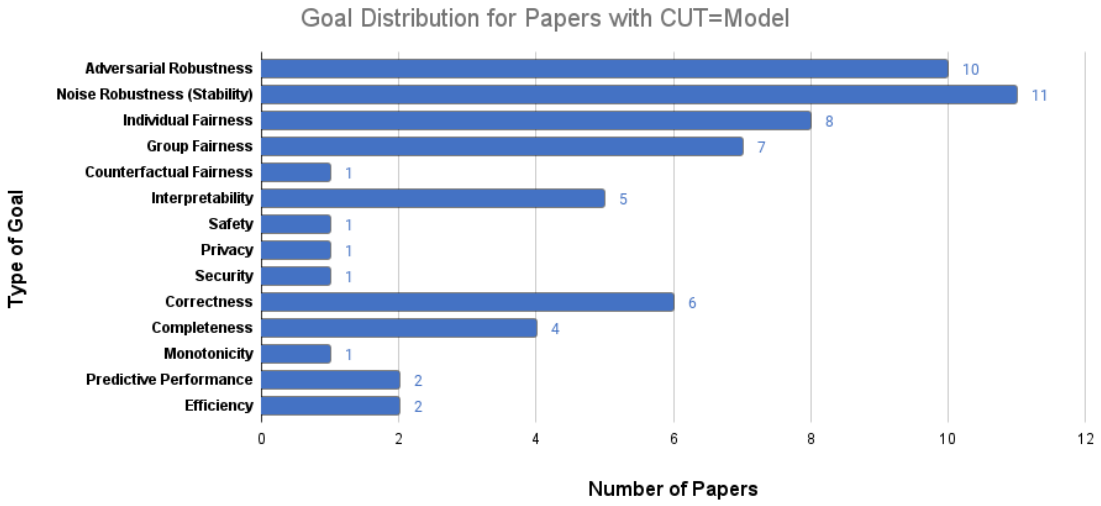


Figure 4.5: Distribution of Goals between the papers that had *Model* as the Component Under Test (CUT). Papers could have several goals at once, so the total number of values for the goals (60) is larger than the number of papers (32).

malware with the help of adversarial metamorphic testing. The idea is that the generated test input data should replicate the natural evolution of malware to see if the model manages to still detect it, or else there might be issues with the model's robustness. There also exists a toolkit, that was created by Soklaski *et al.* [94], to test the adversarial robustness of a model. This tool comes in the form of two testing libraries that can help developers design tests for both robustness and explainability.

There are other approaches than adversarial testing to try to achieve adversarial robustness. Wang *et al.* [95] utilise a technique that allows them to extract tokens from a model that can help them find out if the model is using spurious relations in its predictions. Spurious relations are relationships between variables that might hold for the training data, but not for the entire input space, meaning that the model fails to achieve high robustness. However, knowing these relationships allows the developers to create tests specifically targeting them and also to train the behaviour away.

In the collected literature it was quite common to test the robustness of a model by checking how well it performed on out-of-sample data, which in this thesis is referred to as noise robustness. Often this out-of-sample data is generated by performing perturbations on already existing data, to try to replicate real-world perturbations. The reason for this is that it is more time and resource-efficient to generate a large chunk of data from existing data rather than gathering entirely new data, mostly because of the time it takes to label newly gathered data. One case where perturbations are inserted into data to generate data that can reveal robustness issues is presented by Benedick *et al.* [96]. They present two perturbations, namely swapping data values or removing data points, that can be applied to the data which show promising results in managing to mimic real-world perturbations. Elazar *et*

al. [97] present a method to improve the consistency, meaning noise robustness, of pre-trained language models through the use of paraphrasing. The type of noise this paper is concerned with is when a word in a sentence is replaced with a synonym since this should not change the prediction of the model. The authors of the paper recommend a new step in the pretraining processes where the focus lies on identifying relationships between words through paraphrased sentences, which is meant to improve the noise robustness of the models. Liu *et al.* [98] provide a toolkit that is meant to aid developers in the generation of perturbed test data for a wide array of different models, with everything from classification models to language understanding models.

With the definitions of robustness presented in this thesis, there is some room for overlap between the terms, for example when the out-of-sample data is similar to the training data and the model is weak to small changes to the input, which is the case in the paper by Ribeiro *et al.* [14] and Ramanathan *et al.* [87]. For [14], the focus lies on allowing the developer to define properties that the model should have, which then allows for generating test data that can reveal the model having issues with both types of robustness as well as counterfactual fairness. On the other hand, [87] uses statistics to identify test cases where the model would be correct, but where even a slight change to the input could lead to the model being incorrect. There are also times where both types of robustness are taken into account, but separately, like in the paper by Goel *et al.* [64]. They introduce Robustness Gym which is a toolkit that has functionality such as identifying weak subpopulations of data, applying transformations to input and determining effect, adversarial attack testing, and generic/targeted datasets to help evaluate models.

In the mapped papers, sometimes there is an overlap when testing for robustness and fairness, like in some previously mentioned papers [14], [64]. Another paper where this is the case is presented by Aggarwal *et al.* [99], which proposes a framework that is able to generate data that can assess a model's robustness and fairness. The authors have tried to support many different input data types, such as tabular, textual, and time-series data.

Dealing with Bias:
There are also quite a few papers among the collected papers that aim to address the problems with bias in models. However there are different variants of fairness that need to be achieved to remove biases, like individual, group, and counterfactual fairness as can be seen in Section 4.1. Soremekun *et al.* [100] present a testing approach called ASTREA that focuses on both individual and group fairness by allowing developers to generate a large amount of test data with context-free grammars that can reveal fairness violations. The generated data will contain pair of sentences where the only difference is a protected attribute, which means that unless the model predicts the same for both sentences it must have a bias. Prabhakaran *et al.* [101] deal with the same idea of fairness and focuses on NLP models with perturbations such as swapping the names of named entities and seeing how it affects the model's predictions.

Udeshi *et al.* [77] and Aggarwal *et al.* [102] present papers that have a focus on individual fairness. Aequitas [77], is a tool that uses a search-based technique to find an initial set of discriminatory inputs, and then their hypothesis is that there should be other discriminatory inputs close to these initial values. Given that the model under test is robust, the authors achieve promising results. Aggarwal *et al.* [102] also propose a test input data generation method, but this time based on symbolic execution and local explainability. Chakraborty *et al.* [103] on the other hand chose to focus on group fairness in their paper, with their method relying on semi-supervised learning to be able to label a large set of data while making sure to balance all the labels in a fair way. The goal is then to use this data to train a model to achieve higher fairness than what it would have gotten without the balancing.

The Oracle Problem:
The oracle problem is an ever-present issue for software systems, as discussed in Section 2.2. Therefore it is no surprise that many of the collected papers had approaches for trying to mitigate the problem, with metamorphic testing being one of the most popular for the model CUT [64], [80], [97]–[101], [104]–[108]. Most of these papers propose different types of MR, such as inverting the input should also invert the output [107]. Important to note is that not all of these examples will hold for each type of model and ML task, so reading the complete summaries is recommended to see the full use cases. Other examples are that if the order of the input data to a clustering model changes, then the output should still remain the same [106] and if a small noise is added to a signal then an acoustic classifier should still be able to make the same prediction [108]. Other than the predefined MRs, many papers also offer a way for the developers themselves to create or derive MRs from their models [80], [104], [106]. Murphy *et al.* [104] specifically also deal with the issues of metamorphic testing having many false positives for real values, by allowing any values that are "close enough" to the correct value to also be deemed correct.

SynEva is another approach for dealing with the oracle problem presented by Qin *et al.* [109]. The goal of SynEva is to synthesise a mirror-program to the program under test so that it can be used as a pseudo-oracle. So if the mirror program behaves differently from the main program, then that would mean that there is most likely something wrong with the prediction. The mirror program is synthesised from a training dataset and a knowledge program, with the knowledge program being based on the knowledge that has been extracted from a learning program on training scenarios. Sharma and Wehrheim [110] on the other hand were inspired by metamorphic testing and had an idea where they generate a white-box approximation of the black-box model under test, which makes it possible to verify properties and especially to identify fairness violations. Anything learned from the white-box model can then be applied to the black-box model and the process can start again. The authors call this technique verification based testing. In another paper by the same authors, they utilise this type of testing again [69], but this time with the goal of assuring monotonicity.

Data Drift:
Data drift was another area that was a concern in some of the collected papers since a model performance might be affected negatively as the data drifts over time. Data drift is when the data gradually changes over time, meaning that the training data that the model used to learn how to predict becomes less and less relevant [78]. Saha *et al.* [111] propose a framework that makes it possible to synthesise input data that is meant to be able to test model fairness and robustness, as well as how it reacts to data drift. The framework comes with base settings and constraints, but it is possible for a developer using it to override the initial ones to better fit their own purpose. Ackerman *et al.* [78] instead propose a new concept called *auditors* that is meant to directly estimate data drift which is done by looking at the predictions and the model's confidence in those predictions. The authors present a concrete auditor for classifiers, but also a framework for how to create custom auditors.

Interpretability:
There were also a few papers discussing the interpretability of the models. The first of these is a paper by Ribeiro *et al.* [112], which proposes a tool that is meant to help with the interpretability of classification models. The goal is to help a developer see why a model made a certain prediction, by allowing them to see which parts of the input had the biggest impact on the output. Similarly, Ma *et al.* [113] propose Lamp which is a provenance computation system that uses automatic differentiation to calculate the derivative between the input and output. This allows them to see which input played the largest role in determining the output. The authors believe that it will be useful in both optimising the models as well as helping with interpretability and therefore debugging.

Other Ideas:
In the following paragraphs, some of the other ideas for testing models from the papers in the mapping are presented. Barash *et al.* [114] propose using combinatorial testing between ML solutions and business requirements. The goal is to reveal areas where an enterprise might not live up to its quality requirements, and therefore allow the developers to address these issues. The authors state that their results from case studies indicate that the technique is a good way of gaining insight into the quality of their ML solutions.

Santos *et al.* [115] present two new types of test criteria in their paper. The first criterion is called decision tree coverage, which says that all possible root-to-leaf paths should be traversed in a decision tree trained on the same data as the model under test. The second criterion is referred to as Boundary Value Analysis and it states that the test cases should cover valid boundary values of decision nodes. The overall goal of these criteria according to the authors is to facilitate the task of assessing the effectiveness of tests for MLSs, and consequently to help testers generate good tests that could highlight weaknesses of a model so that they can be addressed.

Breck *et al.* [74] present a wide variety of tests that can be applied to the entire ML pipeline, but specifically for the model CUT they propose 7 assertions that

should be made for every model. These are assertions such as that all the hyper-parameters should have been tuned, knowing the impact of the model going stale, and affirming that the model has a better behaviour than a simple baseline model. All these assertions are meant to come together to affirm the model's quality, and the authors propose an ML test score where the model and the other parts of the pipeline are rated based on how many of these assertions are true.

4.2.5 Machine Learning-Based System

This section outlines the different techniques and solutions that target *MLS* as the CUT during ML testing. The distribution of testing goals in the respective papers can be seen in Figure 4.6. *Correctness* is the most popular goal among the papers, followed by *Group Fairness* and *Safety*.

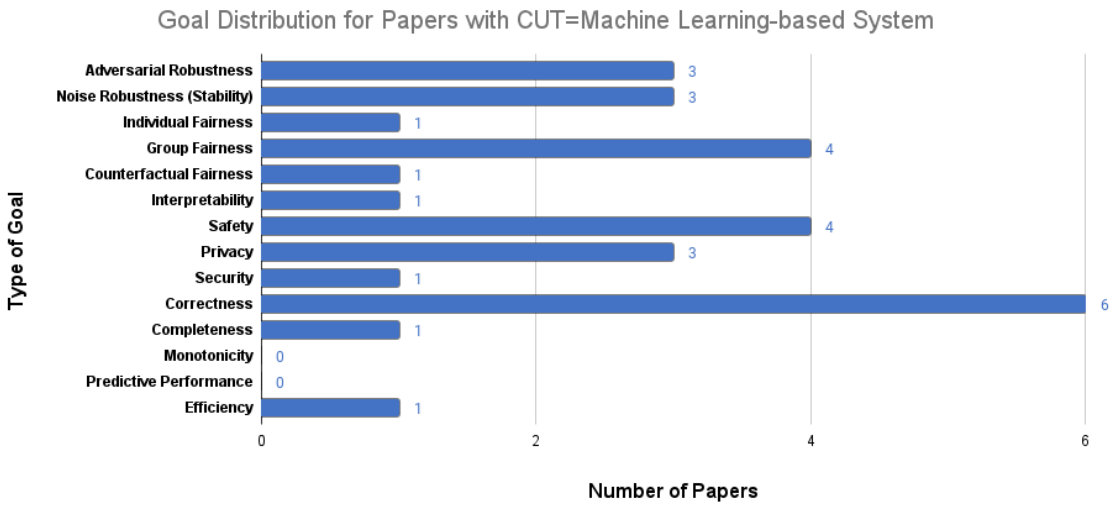


Figure 4.6: Distribution of Goals between the papers that had *MLS* as the Component Under Test (CUT). Papers could have several goals at once, so the total number of values for the goals (29) is larger than the number of papers (17).

Correctness:
Techniques for verifying the correctness of MLSs include [116]–[120]. Zhu *et al.* [119] propose explorative and combinatorial strategies for deriving new test cases from existing test data and applying these to testing facial recognition software. Bozic *et al.* [117] test chatbots using AI planning where the tester first writes a specification of how the chatbot should react to different conditions, and then tests are generated as sequences of user questions. The expected reactions are derived from a planner program that has been given the aforementioned specification. Asyrofi *et al.* [118] propose using differential testing for verifying automatic speech recognition (ASR) software behaviour. They use the behaviour of other ASR implementations as a pseudo-oracle and automatically generate test cases in an efficient manner by keeping the test suite small and the probability of exposing system flaws high. Regression testing is the focus of Zhu *et al.* [116] who propose a methodology of automatically

authoring regression tests. Their proposal is implemented for a spelling checker system, but the authors claim that it is generalisable to many kinds of MLSs. One particular issue with regression test suites for ML software is the degradation of test cases as the oracle gradually turns obsolete. This is solved by periodically updating the test suite with production data from the system under test (SUT).

Robustness:
There are several works that propose techniques for assessing the robustness of MLSs [121]–[126]. Starting with adversarial robustness, Guichard *et al.* [121] use paraphrasing to test the robustness of conversational agents, such as Amazon Alexa or Google Home. This involves switching some words to synonyms in the input to the SUT. Abeysirigoonawardena *et al.* [122] generate test data in the shape of adversarial scenarios for use in autonomous driving simulators. Specifically, they design tests to challenge the system's ability to handle troublesome movement patterns of pedestrians. Regarding the noise robustness of MLSs, metamorphic testing is employed in all relevant papers. Sun *et al.* [123] utilise metamorphic testing to find and repair inconsistent output of machine translation software. They formulate MRs, one of which is the invariance to synonyms, and generate suitable test cases that are ensured to be grammatically correct. Another work by Dwarakanath *et al.* [124] describes a way of finding implementation bugs in image classifiers, one of which is SVM-based, by metamorphically transforming features in the training/testing data. They check the quality of the metamorphic tests by doing mutation testing by injecting bugs into the SUT. Guo *et al.* [125] apply metamorphic testing to plant identification applications by defining MRs based on input features that should not affect output, such as the background of a plant image. Finally, the testing of text localisation systems was investigated by Yan *et al.* [126]. Such systems receive inputs as images, and then scan the image for text and put bounding boxes around the discovered text. The authors propose six MRs that mimic real-world scenarios, to check the noise robustness of such systems.

Fairness:
Regarding the fairness of MLSs, three papers were found [127]–[129] and tagged with some of the relevant categories, i.e., group, individual, or counterfactual fairness. Albarghouthi and Vinitsky [127] introduce a new concept called fairness-aware programming, a method of programming decision-based systems that needs to take fairness into account. They propose letting the developers themselves state in the code what the fairness expectations of the systems are, and provide a general specification language for making this possible. During runtime, the system's decisions are validated against these programmed fairness checks and reports are generated when fairness is not adhered to. FairTest is a tool by Tramer *et al.* [128] that helps developers in identifying fairness issues. It implements the Unwarranted Associations framework, which defines fairness violations as certain associations between input and output. The tool seems to be model-agnostic, since it's a black-box approach, and it provides error profiling and discovery/testing of unwarranted associations. The third work here is Themis by Angell *et al.* [129], which is a technique for automatically generating fairness tests, for both causal and group discrimination

exhibited by software systems. Themis works by defining equivalence classes on the input space of the SUT, and then several optimisations are utilised to minimise the test suite size.

Safety:
Some works also facilitate testing the safety of an MLS [120], [130]–[132]. In the domain of autonomous driving, Neves *et al.* [130] developed a genetic algorithm that can be used to generate test data. Their test generation strategy aims to reach a certain degree of coverage on the system, which is helpful in assuring safety since these systems rely on correctness and completeness to be safe. The authors propose ways to generate new offspring for such an algorithm. While this paper is focused on the domain of autonomous vehicles, the approach of genetic algorithms to generate test cases might be generalisable to other applications. Metamorphic testing has been applied to test automated protein function prediction tools [120]. The authors identify metamorphic relations used to generate new test cases for such tools based on existing test inputs. These tests help to check the correctness and hence safety of such systems since they are used in the medical domain where correctness is vital. In the vision paper by Aniculaesei *et al.* [131], the authors focus on the concept of dependability, which they break down into security, privacy, and safety. To achieve these dependability goals the authors propose to use "dependability cages", derived from the concept of safety cages by Heckemann *et al.* [132]. Basically, a dependability cage is a set of tests on the input and output of a component, that are continuously checked and upon failure will make the system take appropriate safety measures [132]. These cages can be created from existing software artefacts. These cages are designed to be able to deal with both external and internal problems for the software.

Other goals:
Lastly, a paper mentioned throughout the other CUTs is the work by Breck *et al.* [74]. The list of tests provided in the paper partially applies to MLS testing. They propose tests for monitoring an MLS, e.g., to verify that the developers are notified when dependencies of the models have been updated, and to monitor the age of a deployed model so it does not get "stale". A model is referred to as stale when its performance has degraded over time and can not be relied upon anymore. This can be avoided by re-training the model regularly.

4.3 The Toolkit

In this section, the finalised toolkit is presented. The main tool is represented by a list of papers that can be filtered based on a company's goals and context, after which it can suggest appropriate testing techniques. Other than the main tool, there are also several documents provided, all of which are described in Section 4.3.2 below. The full toolkit can be found on GitHub[3].

[3] `https://github.com/alexandersimolaIT/QA-Mapping-for-ML`

4.3.1 The Main Tool

In this section, the implementation of the main tool is shown. As explained in Section 3.2.2, the framework used to build the tool was Microsoft Lists [60], which provided the functionality necessary to make this tool work. A screenshot of the end result can be found in Figure 4.7 and although it is only a small representation of the main tool, it does give an idea of how it looks and functions. Each of the columns has a list of possible tags, which are the same ones as those defined in the Definitions Document that can be found in the GitHub repo. The column names are the same as the section names in the document, and the definitions within those sections reflect the tags that can be selected in that column in the main tool. For most of the columns, the only requirement for the tags is that there is at least one, so that all relevant tags can be used. The only exceptions are the columns **Testing Workflow Main** where there has to be exactly one tag and **Testing Workflow Other** where having zero tags is allowed. The reason for this is that **Testing Workflow Main** is meant to represent where in the testing workflow the main part of the testing takes place, while **Testing Workflow Other** is meant to give additional information in case there are any other parts of the workflow that play a role in the recommended testing technique.

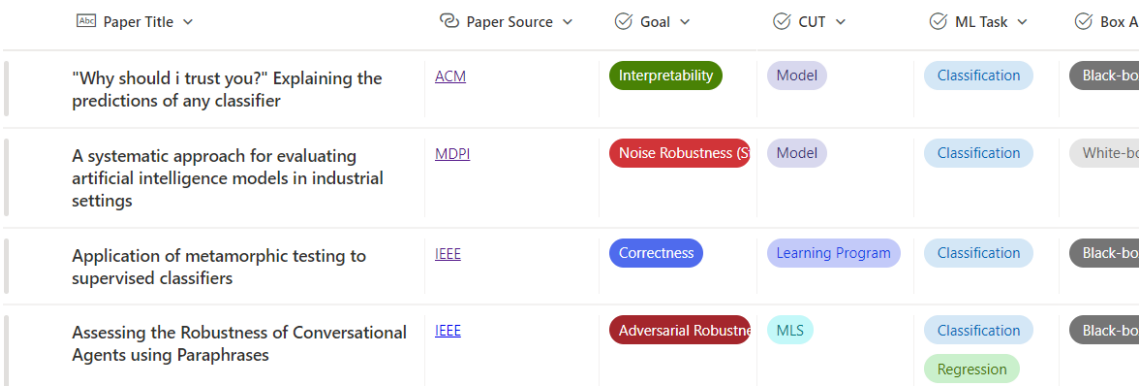


Figure 4.7: A screenshot of the main tool implementation in Microsoft Lists. The columns are the different features that represent the goal, context, and used testing technique. The rows are the list items that are concrete testing technique recommendations. Note that the user interface does continue further to the right, but had to be cut off to increase visibility in this figure.

For each column in the main tool, the user can specify which tags are relevant for their case by applying filters. The window in which this is done can be seen in Figure 4.8. A set of tags used to constrain a search query in the tool is called a "filter". Using these filters essentially corresponds to providing the input for the mapping function, since the goals and context variables that are filtered on results in recommended papers that propose different types of testing techniques.

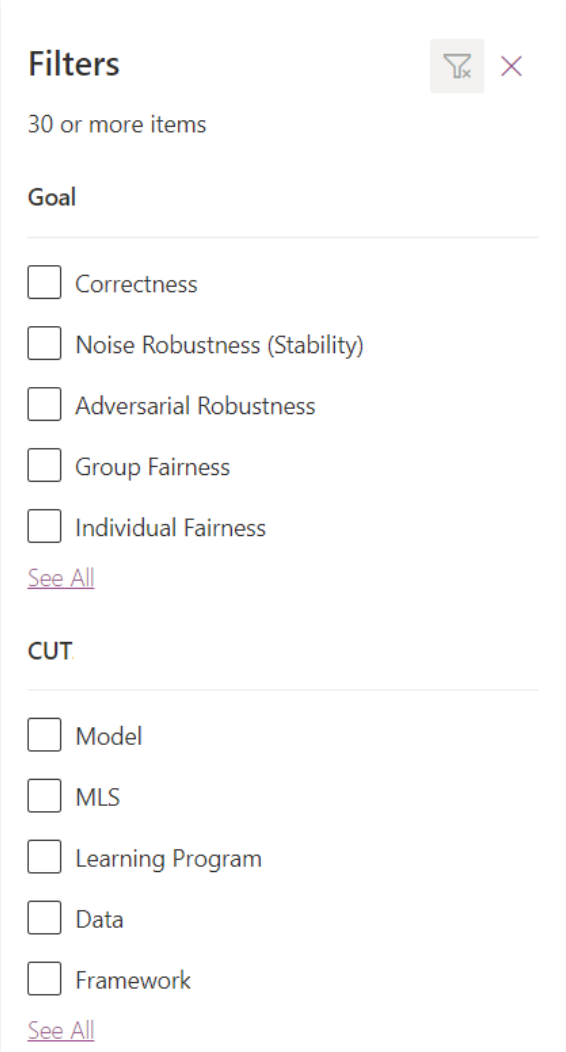


Figure 4.8: The filtering interface containing some of the possible filtering options that can be utilised. It is possible to scroll further down to see more categories, or to press view all to see more options inside a specific category.

4.3.2 The Support Documents

In this section, all the additional documents that have been created are presented. These documents are meant to aid the developers in different ways when it comes to using the main tool. All of them are described in the different paragraphs below.

Overview Tree Document:
To provide a first look at the mapping, five decision trees were created, each representing a different CUT. One of these trees can be seen in Figure 4.9, while the rest can be found on GitHub using the link mentioned at the section's beginning. The tree in Figure 4.9 has the green root node as *Data*, which is meant to show that this tree represents the possible recommendations given that the CUT is the model data. The rest of the trees in the Overview Tree Document are structured in the same way but instead have *Learning Program*, *Framework*, *Model*, or *MLS* as the green

root node. The root node then has outgoing edges to the red nodes that represent different goals that a developer can have with their testing, with each of these goals having been defined in Section 4.1. The goals that can be seen in the tree are only the goals that during the research phase were found to be relevant for that CUT, based on the papers that targeted that component and goal. The goals then have outgoing edges to different testing techniques that, based on papers collected during the research phase, are meant to help achieve the goal for the CUT.

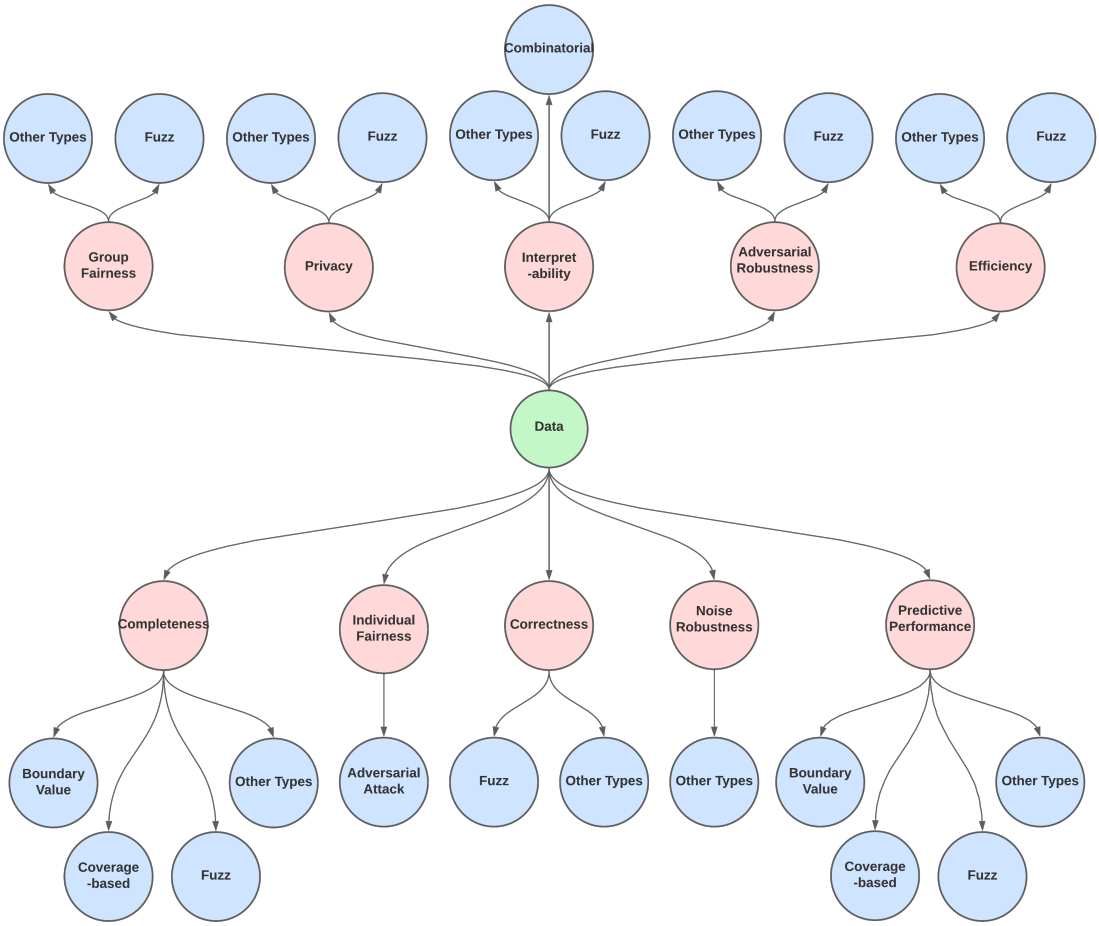


Figure 4.9: Overview of testing techniques (blue nodes) that are recommended for goals (red nodes) given that the component under test is the data (green root node)

The purpose of this overview is to allow developers to get an initial idea of what the toolkit can recommend, without having to be fully informed on its details. For example, knowing all possible feature combinations they can tweak for the goal and context.

Definitions Document:
This document contains definitions for all the tags used in the toolkit. It was created to deal with the problem that there exist multiple definitions for the same term in the ML testing research literature. The document is structured such that each column in the main tool has its own section that contains definitions for all the

tags in that column. A screenshot of the document has been included in Figure 4.10.

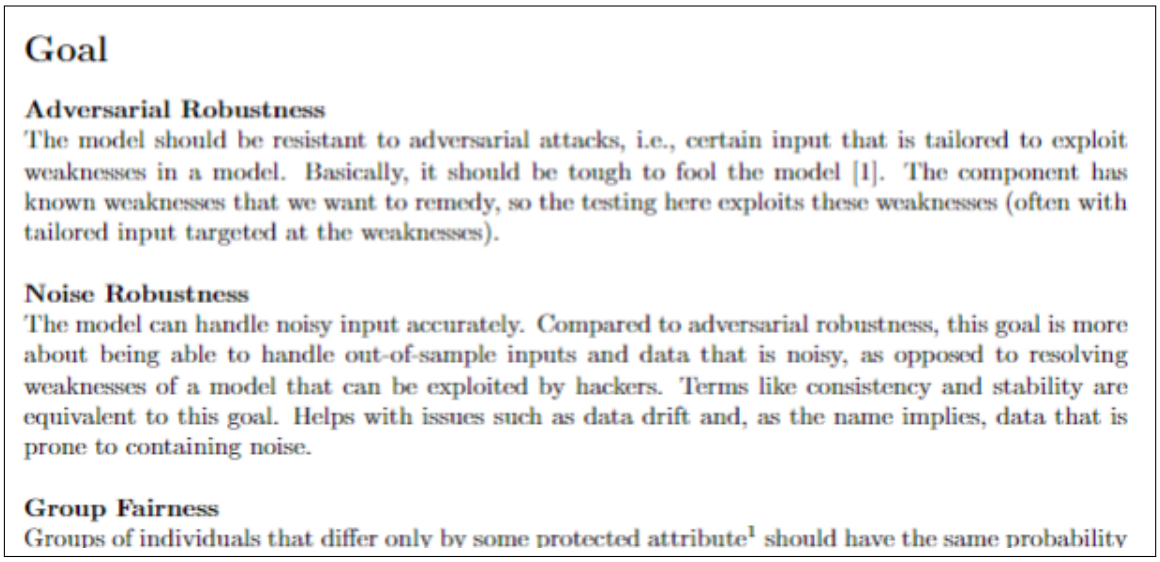


Figure 4.10: A sample of the content in the Definitions Document. Parts of the Goal category describing the different types of robustness is shown.

Goal Identification Document:
The Goal Identification Documents can be seen as a tool that is meant to aid developers in choosing the correct goals for their projects. The document contains *yes/no* questions regarding different aspects of the model, like its data and purpose. An example of these questions can be seen in Figure 4.11. If the answer to a question is *yes* then there are goal recommendations that can be useful in that scenario. This document was created since it is necessary to know what goal to target when using the main tool, which means that it can be useful to offer some assistance to developers that have not already decided what goals they are aiming for in their project.

Paper Summaries Document:
The Paper Summaries Document contains full summaries of all the papers included in the toolkit, which are similar to abstracts but more meant to explain how a paper is unique from all the other papers in the toolkit. Examples of these summaries can be seen in Figure 4.12. It is meant to be useful when the main tool has been used since it can help give a quick idea of what the recommended techniques are about without the developers having to commit to reading the entire paper immediately.

Toolkit Guide Document and README Document:
These documents were created late in the design process of the toolkit to help the developers understand how it should be utilised. The README Document, explains the purpose of each support document in the toolkit. The Toolkit Guide Document provides a guideline for how the toolkit should be applied and when certain documents can be the most useful in the process of applying the toolkit.

- **Does the ML system use data on customers as an input?**
  - **Individual, Group, Counterfactual Fairness:** If the system deals with customer data as an input, then it is important to make sure that the system does not discriminate against customers based on protected attributes such as gender or ethnicity.
  - **Privacy:** If the system deals with customer data it is important to think about privacy concerns since it can otherwise lead to issues with for example EU laws like GDPR.
- **Do you want to have a better understanding of *why* the ML system makes a certain prediction?**
  - **Interpretability:** If you want for example to be able to understand why a model classified an image as a dog then it could be a good idea to focus on interpretability, since this will give you more insight into why the model chose to classify the way it did.

Figure 4.11: A sample of the content in the Goal Identification Document. Two different questions are shown and their recommended goal concerns are described.

This paper contains summaries of all the papers in the mapping. The purpose is to provide insight into the papers similarly to an abstract, so that the user of the mapping can get an overview of the proposed solution in the paper before reading it completely. Every paper gets one section, and the title of the paper is written in bold at the beginning of the section. The sections are ordered alphabetically with regards to the paper title.

A systematic approach for evaluating artificial intelligence models in industrial settings: A systematic approach to test the robustness of a trained model is presented in [1]. It injects perturbations in a test data set, aiming to expose model vulnerabilities in the form of input that the model handles insufficiently well. Two kinds of perturbations are defined in the paper: "swapping", i.e., altering the sequence of consecutive data points; and "dropping", i.e., removing data points. These are applicable to time-series data and mimics the data degradation that can occur in a realistic setting. The authors assess the method on 20 datasets for 7 state-of-the-art algorithms. They conclude that the perturbations are effective at challenging the algorithms, to different degrees, based on the results that the model accuracy is lower on the perturbed than on the clean test data. Furthermore, the specific perturbations that should be used to evaluate model robustness in any circumstance depends on the industrial context and domain. The authors also investigate if the robustness can be predicted before applying the systematic approach described above. They trained a set of decision trees on metadata about each dataset, the parameters of the perturbation used, and a label. The ground truth label corresponds to how much the algorithm was affected by the perturbations, i.e., a measure of the robustness. It was found that this was not a reliable method to predict robustness, so the authors recommend the original approach of generating perturbations instead.

Application of metamorphic testing to supervised classifiers: Xie et al. [2] presents more MRs for classifiers, while also showing real world results from a case study the authors have con-

Figure 4.12: A sample of the content in the Paper Summaries Document. The introductory text as well as an example of a full summary is shown.

4.4 Evaluation of the Toolkit with Volvo Cars

This section presents the results from evaluating the toolkit at Volvo Cars. These results are based on the actions described in Section 3.3. The cases that were used for the study can be found in Appendix A.

A total of eight interviews were conducted during the evaluation phase. These interviews were all conducted with employees at Volvo Cars who worked within some area of ML, including product owners, data scientists, ML engineers, and a technical specialist and architect. It is important to note that some questions were not answered by all interviewees, partly because they might have not used or looked at a certain document in the toolkit during the study. Furthermore, regarding the question about the relevance of recommended papers, some interviewees did not look at the specific papers and could therefore not answer the question.

The tags chosen for the different cases were collected from each test subject. This information turned out to not be as relevant to measuring the effectiveness of the toolkit as the interviews about the usage of the toolkit, so these results have been put in Appendix B. This is further discussed in Section 5.3.1. One part of these results that is worth mentioning is that not all subjects decided to use the same method for selecting tags. The Toolkit Guide Document recommended gathering a large set of tags to get as many recommendations as possible, and then remove tags and narrow down the search until the number of recommended papers feel manageable. Six out of the eight subjects used this method, but the remaining two came up with the alternative strategy of applying several smaller filters and do several searches.

4.4.1 General Toolkit Evaluation Results

The paragraphs below follow the structure of a leading question (in *italics*) and then a summary of the interviewees' answers to that question. Many of the questions were answered with a rating (between 0 and 10) together with a comment that motivates the rating. These questions have been marked with "(Rating and Comment)", and also have a figure that shows the distribution of ratings among the interviewees.

How long did it take to learn how to use the toolkit? (Approximately):
As can be seen in Figure 4.13, most of the interviewees spent less than 1 hour on learning how to use the toolkit before they felt comfortable enough to try to solve the cases. There were also two outliers that spent two and three hours, and motivated this by saying that they wanted to thoroughly go through all the tools in the toolkit to be able to properly assess its functionality.

How intuitive was it to use the toolkit? (Rating and Comment):
The general consensus was that the individual parts of the toolkit toolkit were quite easy to understand and make use of, as can be seen by the ratings on the first (topmost) bar in Figure 4.14. The downside was that the user would have to open

Figure 4.13: A dot plot showing how much time each interviewee, represented by a blue circle, spent on learning how to use the toolkit.

a handful of different documents and then navigate back and forth between them which could be a bit disorienting and tedious. Some of the interviewees proposed solutions to make the toolkit easier to use. The first of these were to merge all the documents, except the main tool, into a singular manual that would contain hyperlinks to facilitate navigation. The other idea was to go even further and create an entirely new app or website that would make it easier to present the information from the documents to the users as well as provide more tools to help the user navigate.

Was it clear how to apply a tag? (Rating and Comment):
This question was clarified during the interviews to be specifically about the mechanics of applying the tags in the main tool, meaning how intuitive it was to find the filtering button and actually apply tags. The ratings given for this question are presented by the second bar in Figure 4.14. Most interviewees stated that they found this task trivial, but a few mentioned that they had had some trouble locating the filtering options. They believe that the user interface of Microsoft Lists could have been more clear. Another participant mentioned that they would like the option of adding tags to the filter by clicking on the tags shown on the list items in the main tool. For example, clicking a *Noise Robustness* tag would in that case add *Noise Robustness* to the tags currently being filtered on and thus update which papers are shown.

Was it clear why and how filters should be applied? (Rating and Comment):
This question was clarified during the interviews to be about how it was to use tags in a more conceptual sense, i.e., how easy it was to understand why and how they were being used. The ratings for this question are represented by the third bar in Figure 4.14. Generally, the opinions were that it was quite clear, but most interviewees had some comments on aspects that could use some work to become better. One interviewee for example mentioned that it was not clear to them that "not selecting any of the tags of that column at all" was equivalent to "selecting all the tags for that column", as it yields the same output from the main tool. Two

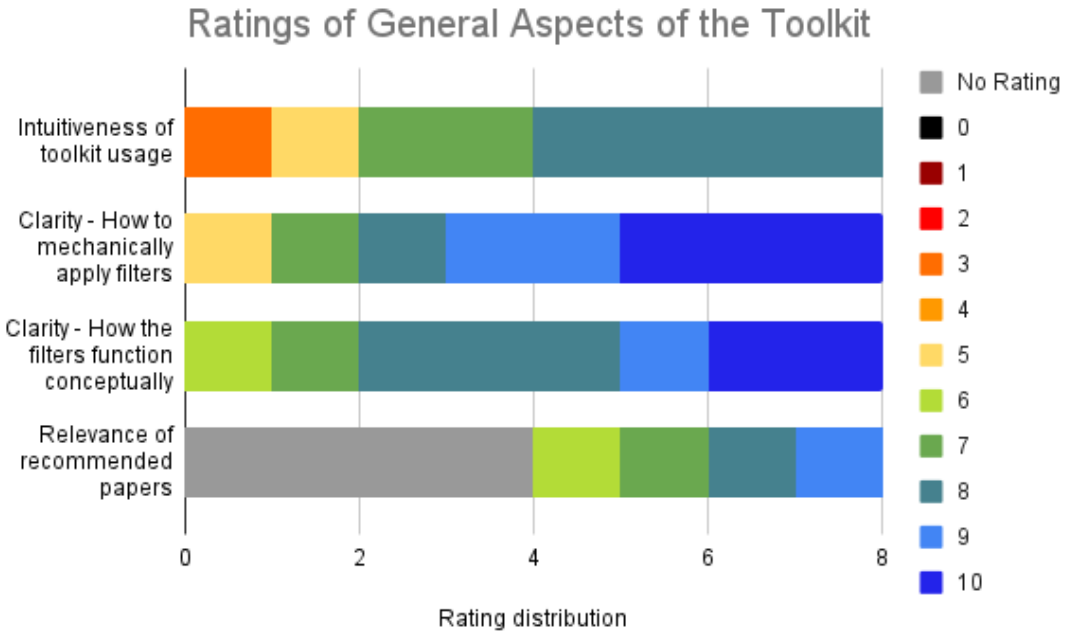


Figure 4.14: The ratings given by the interviewees in regards to general aspects of the toolkit.

interviewees remarked that it was a bit tedious to locate the definitions when they did not understand a tag since they had to open a separate document and then navigate to the right place there. They would have liked to see the definitions when hovering the mouse over the tags, which was not possible in Microsoft Lists. Another interviewee proposed the idea of utilising a question-based system to set the tags instead of doing it manually. This question-based system would ask questions to the developer and then automatically add tags to the filter based on the developer's answers. They meant that this would be less error-prone, and also help guide the users in the right direction by giving them recommended tags.

Regarding the column categories in the main tool, do you feel that there were some missing categories that you would have wanted to be able to filter on? (Yes/No and Comment):
The answers to this question are represented by the first bar in Figure 4.15, which shows that a majority of the interviewees had ideas for other categories. For the interviewees that answered yes there was a follow-up question that asked them to clarify which categories they thought were missing, which resulted in a wide variety of ideas. Two interviewees discussed business aspects, and how it would be nice to have these represented in the toolkit as well. There were also two interviewees that mentioned wanting to see testing methods such as A/B testing, that they thought were missing from the testing types. Another interviewee mentioned categories for model deployment and model implementation, the latter with tags such as *Ensemble Methods* and *Federated Learning*. One interviewee wanted some "quality of life" categories that would show if a paper was available to the public or locked behind

an institutional access/a paywall as well as if code examples were available.

Finally, some interviewees requested more granular tags in some categories, mostly focused on the **Goal** and **ML Task** categories. For the **Goal** category the tags *Correctness* and *Predictive Performance* could be broken down further in the same way as fairness and robustness have been. For **ML Task** the interviewee would have liked to see granular versions as well as the wider variants, so there would for example both exist a *Classification* tag and a "*Classification: NLP*" tag. The interviewee expressed that they did not like the **Limitations** category, so this change would allow for that category to be merged with other categories.

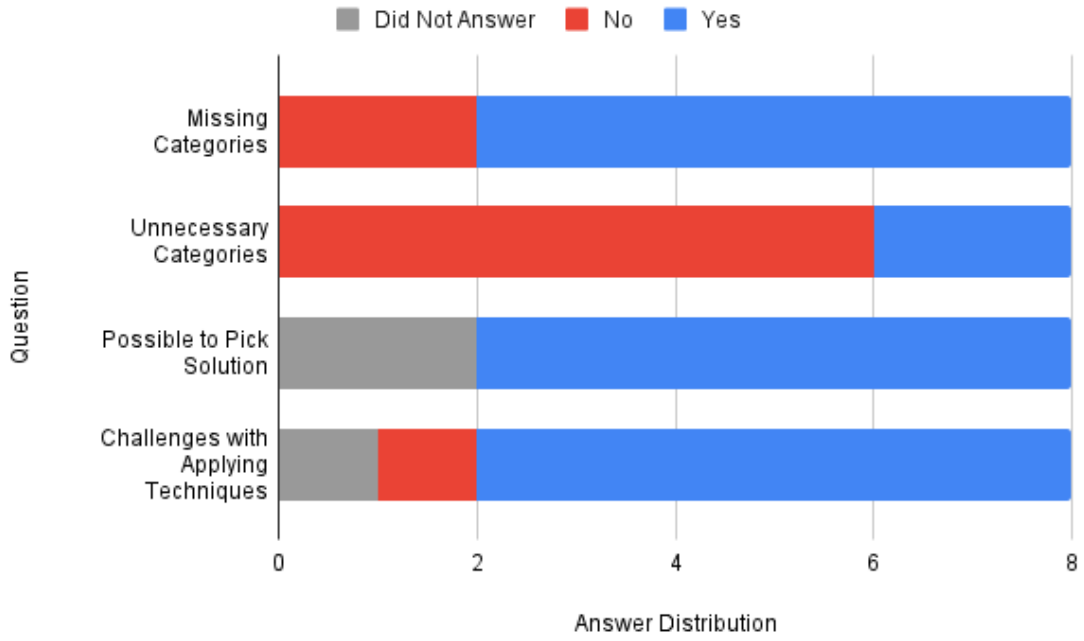


Figure 4.15: The distributions of answers between "yes"/"no"/"did not answer" to the corresponding questions.

Regarding the column categories in the main tool, are there categories that you feel were unnecessary? If yes, then which? (Yes/No and Comment):
The answers to the yes/no-part of this question can be seen on the second bar in Figure 4.15. Most of the interviewees believed that the categories that were currently in the toolkit seemed useful, even if they had not utilised all of them for the cases they had been asked to test the toolkit on. There were comments from two interviewees however, which were related to some tags and the testing type category. The first comment was about the tags *(Empty)*, *None*, and *N/A*, as their purpose was not easy to understand. The second comment was about how the testing type category was confusing as it was not intended to be filtered on, but since it was part of the filterable table it was still possible to do so.

Given the recommended papers, based on your first impression, how relevant did the papers seem to be to your case? (Rating and Comment):
Half of the interviewees felt like they could not give a fair rating to this question without studying the recommended papers more, and therefore elected to not give one. The ratings that were given by the rest of the interviewees can be seen on the bottom bar in Figure 4.14. The consensus was that, once a filter had been applied, the toolkit often seemed to output at least some papers that could be relevant to the case. However, some interviewees noted that it still required some manual work to select a paper once some had been recommended, since it could still be quite difficult to determine its relevance just based on a title and summary. One interviewee stated that the tagging seemed to be accurate, so that helped them in their decision-making.

Given the recommended papers, did you feel like it was possible to pick a solution from the presented ones? (Yes/No and Comment):
The six interviewees that chose to answer this question were in agreement that there always seemed to be at least one paper to pick from when given a list of recommended ones from the toolkit. This can be seen on the third bar in Figure 4.15. However, some interviewees did point out that this was just a first impression and that seemingly relevant papers could after closer inspection turn out not to be so. Most interviewees based their assumptions of relevance on the titles of the papers, and then when they had a few they moved on to the paper summaries and sometimes the abstracts of the papers. The general consensus was that they had to read the entire paper to be completely sure if the paper was relevant or not, but that they had at least gotten a good idea of what the papers were about beforehand.

Do you think there are some challenges in applying a solution from a recommended paper? (Yes/No and Comment):
As can be seen in Figure 4.15, the majority of the interviewees state that there will be issues involved with implementing proposed solutions from the recommended papers. The consensus is that most research papers are difficult to directly apply to the industrial setting, but that it in the end always depends on the paper itself. A paper that provides concrete examples or even code will almost always be easier to implement than one that does not. One interviewee also mentioned that the goal of looking at research papers is not always to directly find a concrete solution, but instead, it is more focused on getting ideas for what an implementation could look like. Another downside with using research papers, according to an interviewee, is that they often make assumptions that will not always be the same as the context of the actual project the interviewee is working on. This means that even if the paper looks promising, it might turn out to not be applicable to that use case at all.

4.4.2 Ratings of Toolkit Components

This section presents how the interviewees rated the different components of the toolkit. The tools are discussed separately, and which tool it is that is currently being presented is shown by the bolded headers. Each tool has received both have a numbered rating as well as some comments on how the interviewees motivated their choices.

Microsoft Lists Implementation

What did you think of the implementation of the tool itself, i.e., how well did Microsoft Lists help in identifying relevant testing solutions/techniques based on the context and goal of the testing? (Rating and Comment):
As can be seen in the first bar in Figure 4.16, the tool implementation in Microsoft Lists got a wide variety of ratings. The main reasoning for this was that, while they thought the concept for the tool was good and deserving of a high rating, they also believed that Microsoft Lists was not the best choice of user interface to represent it. In general, the interviewees said that they could see themselves using the tool instead of going to Google Scholar like they would usually do, although two of them said that this would only be the case if the tool was no longer using Microsoft Lists. Three major downsides of Microsoft Lists were brought up during the interviews. First, it does not have a user-friendly interface. Second, it could be made clearer where the filter button is located. Third, it is not extendable in a way that allows for a lot of new functionality to be added. Two interviewees pushed that they saw the tags and the categorising of papers as the main contribution of the tool, even if the Microsoft Lists implementation left a lot to be desired.

README Document

How helpful did you find the README Document? (Rating and Comment):
In Figure 4.16, the second bar shows the ratings that the interviewees gave for the README Document that is part of the toolkit. The ratings for this document are fairly high, since the interviewees thought that the document had a clear purpose (showing where information is located) and that it managed to successfully pass on the information that it should. The one complaint most interviewees had concerned the file format of the document, which was an MD file due to the fact it was located in the thesis project's GitHub repository. This made the file hard to comprehend since some of the interviewees did not have a program that could properly open MD files. Their recommendation was therefore to use the PDF format for the README file as well since that will make it easier to read.

Was the README Document a good entryway into the toolkit? I.e. did it provide a good overview of what the other documents were capable of?: Most interviewees believed that the README Document was a good entryway into the toolkit, since it provided the information necessary to know how to progress in using the toolkit. One interviewee brought up some criticism against the README Document for not containing any explanation for why a developer should be using the toolkit. They ar-

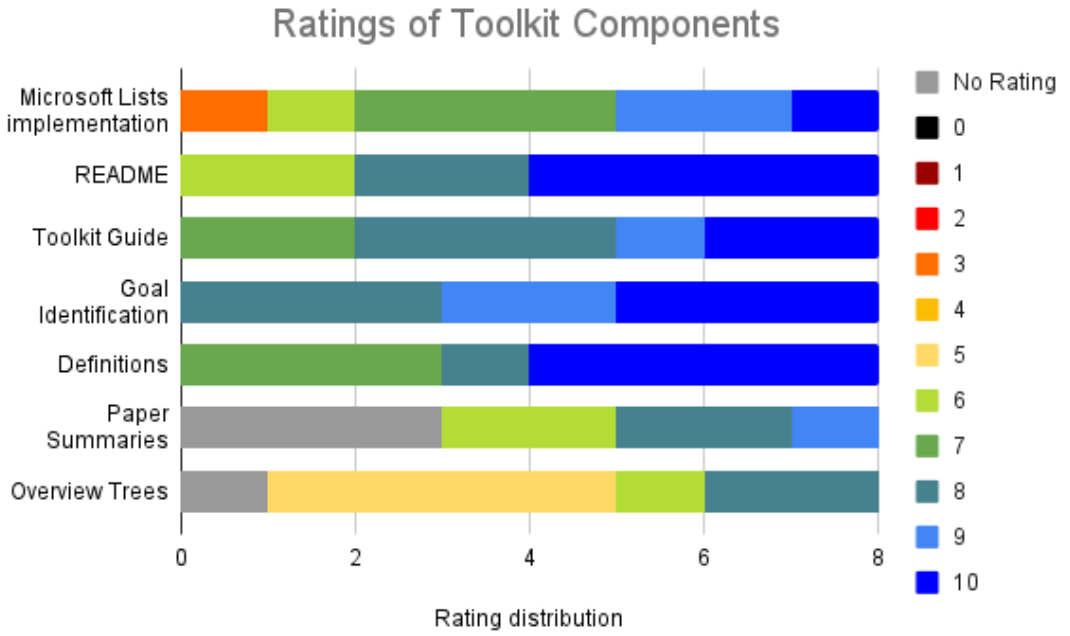


Figure 4.16: The ratings given by the interviewees to the different tools that are part of the toolkit.

gued that this could be good to know since that would help developers understand if the toolkit is properly aligned with their own goals. Some other interviewees mentioned that the README Document kept explanations fairly short and simple, which was positive since they did not want to get overwhelmed with information immediately. Getting a good overview first allowed them to then dive into the other documents for a further understanding as they saw fit.

Toolkit Guide Document

How helpful did you find the Toolkit Guide Document? (Rating and Comment):
The third bar in Figure 4.16 shows the ratings that were given to the Toolkit Guide Document. It was rated fairly high by all the interviewees, as they found it to be a useful resource when using the toolkit. Around half of the interviewees based their entire usage of the toolkit on what the toolkit guide recommended, while the rest used it when they were unsure how to progress with using the toolkit. One interviewee commented that the Toolkit Guide Document had some overlap with the README Document, and that they found the guide to be less useful than the README Document overall.

Difficult to follow the guide? Did the guide lack information about some components of the toolkit? Strengths? Weaknesses?:
All interviewees except one said that they had no issues following the toolkit guide. The issue that one interviewee had was regarding how the overview trees were described inside the toolkit guide, since they thought that their purpose was not clear

enough for them to be useful. Seeing this as a weakness of the document, they recommended that the overview trees could be introduced in a more structured way, but had no exact recommendations for how to do so. None of the interviewees thought that the toolkit was lacking information regarding any subject, except that just like the README it did not contain much information about why the developer should be using the toolkit. The interviewee said that by solving this it would allow them to more easily determine if their goals align with the purpose of the toolkit.

Goal Identification Document

How helpful did you find the Goal Identification Document? (Rating and Comment):
The Goal Identification Document is the highest rated toolkit component, as can be seen on the fourth bar in Figure 4.16. The consensus was that this document was the most useful part of the toolkit, since it helped the interviewees identify which goals they thought could be relevant for the cases they had been assigned. Two interviewees said that this document should be further expanded to include more questions, including more general ones as well as examples from the real world of what can happen if certain goals are not tested for. One interviewee even said that the Goal Identification Document should be made into its own complete tool, since it already provided as much value as the main tool itself and could be even more valuable if allowed to expand further. Several of the interviewees also appreciated that the questions in the document were simple yes/no questions, which made the document easier to use and less dependent on technical ML knowledge. However, one interviewee expressed annoyance at having to jump back and forth between the Definitions Document and the Goal Identification Document each time they encountered a new type of goal that they did not know the meaning of.

Any other example cases and goals you would have liked to see?:
This question was included to allow the interviewees to propose questions that could be added to the Goal Identification Document, since all interviewees had experience with ML projects and could therefore provide valuable ideas to the document. The interviewees brought up possible questions like: "Is the MLSs are used to influence customers?", "Does the system contain several models making predictions based on each other?", "Is the data being collected from edge devices?", "Does the model need to be retrained often?", and "Is the dataset being used unbalanced?".

Definitions Document

How helpful did you find the Definitions Document? (Rating and Comment):
The ratings for the Definitions Document are presented in the fifth bar in Figure 4.16. The document got high ratings from most of the interviewees, since they thought the document provided a good resource for the definitions of the tags and categories in the toolkit. One interviewee emphasised that the document provided a lot of value because it provided what they called "common grounds" for all the definitions. To clarify, this meant that different people might initially have different ideas about what these terms mean, but this document resolves the confusion

by clearly stating the definition of the terms used in the toolkit. One interviewee did express criticism for the definitions themselves, saying that sometimes the first sentence started with a reference to where the definition came from instead of the definition of the term itself. They stated that this might be the correct order to do it in an academic text, but with the tool you want to know what the definition means immediately even if it is good to also get a reference.

Were the definitions clear enough for you to know what they represent?:
The general consensus among the interviewees was that most definitions were understandable and provided good insight into what the toolkit referred to with the different terms. However, some concerns were brought up regarding a few of the definitions. One interviewee thought some definitions were too high-level, and would have liked to see some more examples included in the document. Another interviewee thought that the definitions based on the ISO 25010 standard [13] could have been more clear, since they felt that it was currently necessary to read the standard to completely understand the definition. There were also some comments on specific tag definitions, with one interviewee mentioning that the Offline/Online tags could have better names to not be confused with for example being online and offline on the internet. Another interviewee expressed that they would prefer the toolkit to write out "Component Under Test" fully instead of "CUT", since this abbreviation was not made clear in the toolkit. Finally, one interviewee mentioned that the **Box Access** category had a disconnect between the definitions and the main tool, since if *White-box* subsumes the other types then it should automatically filter on *Data-box* and *Black-box* when *White-box* is selected. This is currently not being done, which made the interviewee a bit confused about whether *White-box* actually subsumes anything or not.

Paper Summaries

How helpful did you find the Paper Summaries Document? (Rating and Comment):
As indicated on the sixth bar in Figure 4.16, three interviewees did not give a rating to this document because they did not look at it, which in turn meant that we got fewer answers to the questions about this document. The helpfulness of the document was still rated between 6 and 9 in aiding the user to find relevant papers for the two use cases. Among the interviewees that did use the Paper Summaries Document, none of them read it through in its entirety due to the large amount of text. Instead, they read the paragraphs that corresponded to papers that were left after a filter had been applied in the main tool. Some of those interviewees thought that the summaries were not detailed enough for a developer to go and implement the technique from the paper. The summaries would require more concrete information about the proposed techniques in order to be realisable, according to one interviewee who would have liked the summaries to contain code snippets and examples. The same interviewee would ideally also like the toolkit to recommend concrete techniques rather than papers about techniques, as they would prefer not to read any papers at all. Still, several interviewees deemed the summaries as helpful to read to avoid being misled by a paper title before reading the actual paper.

Were the summaries that you read easy to understand? Did the summaries bring clarity as to what a paper could offer (if you read any)? Other strengths and weaknesses of the document?:
Among those who used this document, all agreed with the statement that the summaries were easy to understand, and likewise for the statement that they brought clarity as to what a paper could offer. According to one person, the document offered concise summaries of what to expect from a paper, and that it helps a lot in further filtering out papers after having applied tags. Regarding weaknesses of the document, a commonly expressed concern was the amount of time required to manually write summaries for all new papers added to the toolkit, which simply is not scalable. Also, the ordering of the papers is currently alphabetical, which one interviewee thought to be suboptimal compared to clustering the papers based on their content.

Overview Trees

How helpful did you find the Overview Trees Document? (Rating and Comment):
The ratings that the interviewees gave the Overview Trees can be seen on the bottom bar in Figure 4.16. Among the different tools, the Overview Trees were the lowest rated, since many of the interviewees did not believe that they got any assistance from using them. Several of the interviewees stated that they barely used the document at all, and one even specified that they believe their final recommendations for each case would be the same with or without using the trees. When asked, half of the interviewees also said that the trees were not intuitive, and therefore they had difficulties understanding both their purpose and meaning. One interviewee recommended that a description should be added to the Overview Tree Document to make it more clear how they are meant to be used. Two interviewees thought that the current presentation of the trees was a major weakness, one of them wanting the trees to be interactive instead of static, and the other participant thought it would be better to not have them represented by graphs at all.

Would you have liked any other category represented in the tree?:
The interviewees thought that this was a difficult question to answer since to them the purpose of the trees was unclear. One interviewee however, did express that they would prefer having the **ML Task** category represented in the tree instead of the **CUT** category. Some other interviewees thought the current choice of nodes was good, but that the presentation could have been better.

4.4.3 Interviewees' Ideas for Improvements

In this section, the interviewees' ideas for improving the toolkit are presented. The paragraphs below are based on the interviewees' answers to the final question of the interview, which was about the changes and improvements they would recommend for the toolkit to make it better.

Several of the interviewees discussed different ways of improving the presentation of the toolkit. One interviewee would have liked to see the different documents combined into a manual, so that it would be easier to navigate between them with hyperlinks and a table of contents. Another person recommended implementing it as a web app, since that would allow for the information included in the toolkit to be integrated in a better way. For example, the definitions of tags could be shown when hovering over a tag with the mouse, and when a paper was clicked it could show the paper summary. There was also an interviewee that stated that the biggest downside with the toolkit currently was how much you had to go back and forth between documents, and that any change that reduced this would be good. Two other interviewees mentioned making the overview trees and the Goal Identification Document more interactive.

Another idea that was brought up by several interviewees was to make the tool more of an online service. This would allow the users to comment on and like/dislike papers for other users to see, which would help determine which papers had valuable solutions and which were not as useful. One interviewee recommended taking the concept one step further, by basing the usefulness of a paper on collected metrics such as number of clicks, like/dislike ratio, and how long people spent reading it.

Some of the interviewees believed that the toolkit was currently lacking a bit in the number of papers available as recommendations, but that it would not be feasible for one person to continue adding them manually. Instead, it would be better to create a model that can fairly accurately tag given papers, and in that way expand the number of papers available. Another idea was to add an interface for updating the list of papers, so that the developers using the tool could add additional papers that they find useful themselves.

Regarding the content of the papers, there were also some recommendations by the interviewees. One interviewee would have liked to see other types of sources recommended rather than just scientific papers, since they believe that blog posts, guides, and lectures could also provide a lot of value. According to the interviewee, allowing these types of recommendations would help boost the value of the entire toolkit by quite a bit, since these resources can often be easier to digest and get ideas from. Another interviewee enjoyed the focus on traditional ML in the toolkit, but would in the future like to see the toolkit being extended to deep learning as well. There could then of course be a filter on if we are looking for deep learning or traditional ML papers, but given the user the flexibility could provide a lot of value.

There were also some smaller comments made by the interviewees. One participant stated that they would have liked for the filter button in Microsoft Lists to be more apparent, since it is currently a bit hard to spot. Another interviewee wanted to be able to press the tags on the list items themselves to directly filter on them. Finally, there was one interviewee that thought some categories could be more granular, since some tags like *Correctness* and *Classification* could probably be broken down into smaller similar groups.

5
Discussion

This chapter discusses the findings and work presented in the Results. The discussion is focused around relating the research questions to the findings as well as the created toolkit. Additionally, the contributions of this thesis to the field of ML testing are elaborated on. Suggestions for future work and descriptions of validity threats conclude this chapter.

5.1 Machine Learning Quality Assurance Goals

Section 4.1 presented the findings of which goals were relevant for ML software from a QA perspective. These goals come both from traditional software engineering, such as correctness and completeness that are based on the ISO 25010 standard [13], and more ML-specific QA goals, such as interpretability and fairness. During the research phase, it became apparent that the definitions of these goals were used differently in many papers. The robustness goal was one of the main culprits of this, with the term being used in a wide variety of ways in the literature often without a clear definition. The purpose of Section 4.1 was therefore to present definitions of all these goals to give the readers of this thesis and the users of the toolkit a full understanding of what the different goal terms are referring to. We also hope that these definitions will be helpful even when not using the toolkit, to help facilitate discussion among developers with regard to SE testing in ML.

The other purpose of the goal definitions is to provide an overview of the goals that are most frequently discussed within QA for ML, since this was the point of the first research question. The type of testing done when looking at software from a QA perspective differs from the testing usually done for MLSs, which focuses solely on their predictive performance through metrics such as accuracy and recall. QA-based testing, which is promoted by our toolkit, instead focuses on what can be thought of as the quality of the software as a whole, with aspects such as maintainability and safety getting more of a spotlight. These aspects are just as important for ML as they are for traditional software, but due to the difficulties with achieving high quality for stochastic software there currently exists no universally accepted standard for ML like the ISO 25010 standard for traditional software. With the goal definitions, we hope to have summarised and expanded the current research field into QA attributes for MLSs to help future researchers and practitioners create this standard.

5.2 The Toolkit

Section 4.3 presented the toolkit that was built around the generalised mapping mentioned in RQ2. Accompanying the main tool is a set of support documents, and together they form the toolkit. This section discusses several aspects of the toolkit components.

First, we will discuss the main tool of the toolkit. It was implemented in Microsoft Lists since we believed that it was more efficient for this project to build upon an already existing tool than to create one from scratch. The reason for this was that we could then spend more time focusing on conducting the research into papers that should be part of the mapping and therefore the toolkit. This also left more time on the evaluation of the toolkit. The choice to use Microsoft Lists can also be argued to have led to the creation of some of the support documents, since weaknesses of the implementation had to be remedied by these additional documents. For example, the Definitions Document had to be created since there was no way of displaying this information in the main tool in a clear and understandable way. The same can be said for many of the other documents. The final implementation of the main tool had all the functionality that we had planned for it to have. We did note that if we ever want to make some major updates and add new features it would probably be better to start thinking about building something new from the ground up, since Microsoft Lists has limitations to how extensible it is. So to summarise, the current implementation works well for the basic purpose of providing recommendations, but in the future, it will most likely need to be replaced with a more extensible application.

The Definitions Document was designed such that the definitions were relatively succinct and not too formal, which can be seen when comparing the definitions of the goals presented in this thesis compared to the corresponding parts of the Definitions Document. The motivation was to make the document easier to use in practice for developers. For the first iteration of the mapping, we decided that some definitions should be made more specific than how most literature uses the terms. Fairness is one of the goals which was divided into three more specific types, namely group fairness, individual fairness, and counterfactual fairness. The reason was to make it clear what type of fairness a given paper addressed. Robustness is another goal that was also divided, for the same reason as fairness, specifically into adversarial robustness and noise robustness.

The Paper Summaries Document contains the summaries of all the papers in the toolkit. The point of these summaries was to provide the developers who are using the toolkit with short summaries focused on the techniques and the uniqueness of each paper. This means that if they get several papers recommended by the toolkit, they can utilise these summaries to get an indication of whether a paper is relevant or not. It can also help developers prioritise which papers to read first, and hopefully save time spent on reading papers overall. We believe that reading the abstracts of the papers themselves is still useful to get even more insight into what a paper is about, but that our own summaries will provide more value when it comes to getting

a basic understanding of the testing technique applied in the paper. As discussed in 3.1.4, this document was actually first created for ourselves in the research phase to help remind us what the different papers were about. This was incredibly useful for us during the tagging process of the papers, which made us realise that this could be useful to share with the developers that are meant to use the toolkit as well.

The Goal Identification Document was recognised as important during the design process, since we realised that to be able to use the main tool the developers must have an idea of what goals they are testing for. If they do not, then the main tool will lose most of its utility. The document is based on common scenarios that can occur when dealing with ML models. We speculate that this document has a lot of potential to improve and become more valuable with the help of industry experts that can provide questions and answers that will give insight into actual scenarios from the industry. After looking at the results of this document it also becomes clear that the concept of "goal identification" could be expanded to include other categories from the mapping. This would mean that the developers can get further guidance in selecting other tags than goals as well. We believe this could make it even easier to use the toolkit, as long as the questions in the document are well thought out.

The Overview Tree diagrams were created when the main tool had been finished, since it became apparent that there was no easy way to get an overview of the input and output space of the mapping. These tree diagrams were therefore meant to act like overviews of the content for different **CUTs**, so that a developer could quickly get an idea of what goals the toolkit could help them fulfil. **CUT** was deemed one of the most important context variables, and thus, we chose to make it the main variable and to not combine it with any other context-related variables (like **Box Access** or **ML Task**). Having too many variables in the tree would lead to the same issues that the main tool had during the design process when it was implemented as a decision tree, as explained in Section 3.2.3. Choosing **CUT** over the other variables was because of the fact that **CUT** has a lot of importance when it comes to the actual implementation of the testing techniques, since it decides where in the pipeline they should be. It could also have been possible to use **ML Task**, but in the end, we believed that **CUT** provided a better overview than **ML Task**, which would have been dominated by the hypothetical *Classification* and *General* trees due to the large share of papers with these two tags.

The Toolkit Guide Document and README Document were added late in the process of creating the toolkit. They were created as the toolkit grew larger with more support documents, since it became more difficult to understand how to use it without getting a good overview of the available content as well as an example use case. We hope that these documents will be enough to get over this hurdle, so that the toolkit can remain straightforward to use. These two documents themselves were designed to be as clear and simple to understand as possible, since if they were too complex then they could possibly just make the developer even more confused.

Another part of the toolkit worth discussing is the final curated list of papers. We believe that the process used to derive the papers from the initial pool described in Chapter 3 resulted in a good list of relevant papers that could be useful to developers. The papers also cover a fairly wide array of areas, as shown in Section 4.2, which hopefully means that the toolkit will be useful for many different types of project cases and not limited to a certain industry or scope. However, we do note that this curated list will get outdated as new papers are released, and we therefore discuss potential solutions to this problem among the future works in Section 5.4.

To summarise what has been said, one implication of this toolkit is that it provides an accessible way for researchers to view the content of the mapping that forms the foundation of the toolkit, and in that way allow them to take part in extending and enhancing it. The other implication is that the tool itself will provide value in the industry, where ML engineers can utilise the toolkit to help them find solutions for MLS testing.

5.3 Effectiveness of the Toolkit

To be able to give an answer to RQ3, we determined that the way to measure the effectiveness of the toolkit was to look at the relevance and quality of the recommended papers, as well as how helpful and intuitive the toolkit was in achieving the desired results of finding these papers. The sections below discuss these different aspects of effectiveness and draw some conclusions from them.

5.3.1 Application on Two Realistic Cases

The chosen tags and papers from the application of the toolkit on the two project cases by the test subjects were mentioned only briefly in Section 4.4, with the full results being available in Appendix B. We believed that these results were not very relevant compared to the feedback from the interviews, when it came to determining the effectiveness of the toolkit. This was mostly because of the small sample size, which allowed the interviews to be longer and more valuable, but it also meant that there was not enough data on the chosen tags and papers to make any larger conclusions. A pattern might have become apparent with a larger sample, which for example would allow us to measure consistency by looking at similar choices. We do however believe that the feedback provided by the interviewees is more valuable to the future implementations of the toolkit, since they could provide more concrete and direct ideas on what was in need of improvement.

One interesting aspect of these results is that two participants in the study independently came up with the same new strategy of filtering the papers. We recommended, in the Toolkit Guide Document, that they should select a large number of tags and then narrow down the search by iteratively removing them, which we thought was the most efficient way of ending up with a few papers relevant to the case. However, these two participants both decided to instead create several small sets of tags, choosing at most two or three, and then filtering on them like more

specific searches. This of course meant that not every search gave many or even some results (if the combination didn't exist), but since they made several searches they found relevant papers just like the participants that used the recommended method. The insight we draw from this is that the method we had come up with and thought was the most straightforward, might actually not be the only good way of utilising the toolkit.

5.3.2 Relevance and Quality of papers

Before discussing the relevance and quality of the papers, we will describe what these terms are referring to. Relevance refers to how well the paper fits a given use case, which in this case is a project description, with respect to contextual variables. A paper's relevance therefore varies between projects depending on the goals and context. However, the quality of a paper does not change. One perspective of the quality of a paper is to look at how difficult it would be to apply the technique(s) it proposes. This means that papers that go into more detail and provide examples of code or equations could be seen as higher quality from a developer's point of view. This idea is based on what was said by the developers during the interviews about the difficulties in applying papers. Another aspect of quality is how well written a paper is, but we argue that as long as the paper is written decently this type of quality does not matter much to the developers. This means that a developer and a researcher value the quality of a paper a bit differently, which is something we did not consider before performing the study.

The general opinion of the interviewees that tested the toolkit was that it could be difficult to determine the relevance and quality of the papers included in the toolkit without reading them in their entirety. Half of the interviewees elected to not rate the relevance at all. Most of the interviewees also had pressed schedules at work, so it was not feasible for them to read multiple of the included papers. Despite this, the majority of interviewees stated that at least one of the recommended papers could be useful for the use case they were given, as shown by their ability to pick a paper from the recommendations described in Section 4.4.1 and indicated by Figure 4.15.

The ability to pick a paper is mostly based on the paper's relevance, but the quality of the paper will help the developer determine if they should stick with the paper or look for other alternatives. With the help of the interviewees, we got feedback on the relevance of the papers. Although the cases were based on real projects from Volvo Cars, and therefore not written in such a way that it would be straightforward to apply the toolkit to them, it was still possible for the interviewees to find papers they thought were suitable. We argue that this shows that the toolkit has a promising start when it comes to the relevance of the papers, and that it could be even further improved as more papers are added.

We were not able to measure the quality of the papers during these interviews in a similar way to the relevance, mostly due to interviewees not having the time

to go through many papers in their entirety. However, they did provide valuable insight in the form of the challenges in applying a paper, which helped us define the attribute of quality for the papers. With this definition in mind, we could say that the papers in the toolkit come in a wide array of different qualities. Some of them showcase everything in detail, like [69], while others provide less concrete, more high-level descriptions and therefore less valuable for a developer. If more papers are being added to the toolkit, it would be valuable to take this new definition of quality and apply it in the filtering process. This could help raise the quality of all the papers in the toolkit, and therefore provide more value to developers.

5.3.3 Intuitiveness and Helpfulness of Tools

During the interviews, each interviewee was asked to directly rate the intuitiveness of the toolkit, and the helpfulness of the different supplementary tools. This section focuses on discussing the results presented in Section 4.4.2.

The rating of each toolkit component given by the interviewees can be seen in Figure 4.16, which shows that the Goal Identification Document was the most well-liked with the Toolkit Guide Document and Definitions Document following closely behind. These documents were generally seen as very helpful by the interviewees, which was understandable since they contained some explanations and definitions that are almost a necessity to use the main tool. The highest-rated toolkit component, the Goal Identification Document, aids developers in identifying which goals could be important for their projects, which simplifies the usage of the main tool since this can be one of the most difficult tasks when using it. As mentioned in Section 4.4.2, several of the interviewees mentioned that they wanted to see this document grow into its own complete tool, because they believed it could be useful even when not using the rest of the toolkit. This shows that the tool brings a lot of value to the toolkit, and that the developers found it valuable. It also shows the need for future work focused on the Goal Identification Document, either as a continued part of the toolkit or even as a stand-alone tool.

The Toolkit Guide Document was also found very helpful by the interviewees, but this time it was more in sense of it fulfilling its purpose in an efficient and understandable way. The goal of this document is simply to demonstrate how the toolkit can be used, and based on the feedback and rating it received, there are only minor adjustments necessary to make it live up to this role fully. However, due to the nature of the document, it will have to be updated in tandem with the rest of the toolkit, since it would otherwise become outdated and lose some of its value.

The Definitions Document was appreciated due to the fact that it gave definitions for all the terms used in the main tool. Without the contribution of these definitions, the terms would have been ambiguous and therefore open to interpretation. That would mean that the relevance of the papers would become much harder to evaluate. The reason for this is that if a developer thought a term meant a certain thing, but it means something else in this toolkit, then the recommendations

will contain results based on a term that the developer was not interested in. The feedback and ratings that were given to the document show that it is successful in remedying this problem even if there are some improvements that can still be made to the document itself.

The README Document got between fairly good and great ratings, but not quite as high as the previously mentioned ones. Based on the feedback we believe that this is due to two reasons. The first is that the document, while being a good resource, is not essential to using the toolkit. It provides a nice first overview of what the documents contain, but this information can also be seen in the Toolkit Guide Document. The README Document therefore does not add anything new to the toolkit, other than to specify what the other tools can do. This is valuable as an entryway into the toolkit according to the interviewees, which is in line with what the goal of the document was when it was created.

The second reason for why the README Document got lower ratings than some of the more well-liked documents was because of its file format. It was sent to the interviewees in the MD format, which several of them had trouble opening due to no native support on their systems. The reason for the file being an MD file instead of a PDF was that the toolkit is originally located in our GitHub repository[1]. Github uses MD files to display information about a repository in a nicely formatted way, however, when opened as a plain text file, this information is not presented in the same way. During the study, the participants were handed zip files with the contents of the repository, which meant that they could not view the MD file as intended. This is a fault on our part, as we should have taken this into consideration and provided the README Document as a PDF alongside the other documents.

The main tool implementation in Microsoft Lists got one low rating, but was otherwise rated fairly good. As explained in Section 4.4.2, the interviewees felt that the ratings depended on the context. They said they give it a high rating compared to what they were using currently, which was mostly Google Scholar. However, they would also rate it lower if they were comparing it to what they imagined was the best way of implementing it. This is understandable since we also believe that the implementation of the tool could have been much better if done through the implementation of a web application where we would have more control of the features. The reasons for choosing to go with Microsoft Lists were motivated briefly in Section 3.2, but it essentially comes down to time constraints and choosing to focus on the research stages over creating our own tool. The prospect of creating this new version of the toolkit is discussed as future work in Section 5.4.

The Overview Trees Document was rated fairly low compared to the other documents. Many of the interviewees motivated their rating by stating that they did not really understand the trees or their purpose. This motivation becomes quite clear when we look at the Overview Tree Document and see that there are no explanations or clarifications anywhere. One interviewee mentioned that they wanted to

[1] `https://github.com/alexandersimolaIT/QA-Mapping-for-ML`

see a small introductory section in the document which would give some necessary insight into the trees and their usage. We agree with this idea, and also note that there is some basic information in the Toolkit Guide Document that pertains to using the Overview Trees Document. This information was however also criticised for not being clear enough, so the overall lower ratings for this document are understandable. A first step to improving them would be to add the description, but an interesting future work would be to see if there would be better ways of presenting the information that we are trying to express with these trees.

The Paper Summaries Document was not rated by three of the interviewees, since they believed they had not looked at it enough. The interviewees that did look at it believed that the summaries were easy to understand. The goal of the document was to write summaries that differed from the abstracts. Our idea was that these summaries should focus more on the testing technique in the paper and how it differs from those in other papers with similar goals. This could be achieved because we had the benefit through our research of knowing about several papers in the same area that discussed similar techniques, and could therefore focus on explaining the basic concept and the differences between them. This together with the abstract of the paper itself was meant to give enough insight for the user to know if the paper was worth looking into. Based on the feedback and ratings it seems like the document managed to live up to this purpose, at least for the papers that are currently included in the toolkit. In Section 5.4 we discuss the issues that this document might face when more papers are added to the toolkit, and how we propose of dealing with these problems.

Based on the discussions thus far, the toolkit seems to have met our expectations for it, for the most part. The implication is therefore that the toolkit is effective enough, with respect to its purpose, to be useful at a real company, but that there are of course still parts that could be further improved. The Overview Trees Document is the only document that would require a major overhaul to become useful in the way it is intended. The other tools are fully functioning as they are, but of course, also have a lot of avenues for improvements especially when it comes to the Microsoft Lists implementation. Some of these improvements are discussed in the section below.

5.4 Future Work

In this section, some potential future work is presented. Most of the ideas are based on the feedback gathered from evaluating the toolkit and the discussions it generated.

The Concept of Quality:
Based on the discussion above about different views on the quality of research papers between developers and researchers, it would be valuable to make a clear definition of what quality means from the perspective of a developer. Section 5.3.2 lays the groundwork for the idea, but it needs to be further investigated to determine which

attributes in a paper are the most important for a developer. The goal would be for these changes to make the list into one that is directly curated for the developers with their expectations and needs in focus. With the new concept of quality, this might also mean that the toolkit is not limited solely to peer-reviewed papers. Lecture notes, blog posts, and other types of ML testing resources could also be included, as long as they live up to these new quality standards.

Making a Web Application:
Based on interviewee opinions, it was evident that the toolkit would be much more valuable in the form of a web application, or a similar format. Although the main tool is already web-based, the idea would be to incorporate the support documents into it as well. This would allow us to integrate several of the current toolkit components in more efficient ways, and solve some of the current issues with navigation between the documents. Future work would therefore be to build an application by potentially starting with the current implementation in Microsoft Lists as a frame of reference for the main tool. The definitions from the Definitions Document could be added as tooltips on the tags, the summaries from the Paper Summaries Document could be shown when clicking a paper, and the Goal Identification Document could be integrated so that you would answer questions that would automatically select tags for you. These are just a few potential improvements that we could imagine from this early work, much more will probably become apparent as the web application is developed.

The Future of the Goal Identification Document:
The potential of the Goal Identification Document was made clear from the evaluation of the toolkit. Two interviewees expressed their desire to see either a stand-alone version or an expanded version that is still a part of the toolkit. In either case, the first improvement would be to add more questions, since the current one is still limited to our experiences and knowledge. We would recommend reaching out to industry experts to get insight into real cases from which questions could be derived. An improvement for a stand-alone version would be to move from the document-based form to a more interactive application form, to allow for even more features to be added in the future. For the toolkit version, this could instead be to have the document more integrated with the rest of the toolkit, as described in the paragraph above.

Improving the Overview Trees:
The Overview Trees Document was meant to offer an overview of the types of testing techniques that the toolkit can output given that you know what component you are testing and for what goal. Based on the evaluation, the overview trees did not manage to live up to this purpose, since they were difficult to understand and hard to utilise. Future work would therefore be to take the feedback from this thesis, and see if there is a potential use for these overview trees through specific improvements. These improvements could be as simple as adding better descriptions or changing the nodes, but also larger in scope such as making the entire tree interactive. This would allow for much larger trees, easy navigation from branch to

branch, and the possibility of adding more information to each node. On the other hand, there could also be other ways to better present the same information that we have not thought about, so researching such possible venues could also be of interest.

The Process of Adding New Papers:
Possibly one of the most important future works is how to update the papers in the mapping and hence the toolkit. Some contributions, such as the definitions in the Definitions Document, will potentially be useful for a long time, but the list of papers in the main tool itself needs to be updated, or else it will quickly become outdated. The manual approach would be to repeat the research process described early in Chapter 3 each year, add recently released papers, and to check if any old ones have become outdated. The process could also be partially automated, by creating an ML model that can help tag papers and require as little human input as possible. The question then would be how much that would affect the quality of the curated list.

Another approach to updating the list could be to make the entire toolkit more community-based, allowing users to submit papers and also vote and comment on their quality. The toolkit would first have to be implemented as a web application since in this case, it can no longer be an offline tool. Allowing users to vote on the quality of the papers would make papers that many developers find useful be highly rated, hence the quality of the papers would be anchored in reality rather than on some arbitrary definition of "quality". The downside to this idea is that it needs a large user base to truly become useful, and since it will be an online service it will also still be needing moderation to ensure that the comments are kept civil.

Extension into Deep Learning:
Currently, this toolkit focuses on testing techniques for traditional ML as well as general ML. One interviewee from the evaluation that worked with deep learning systems would have liked to see the toolkit get extended to also work with deep learning, since they felt that it could make it more useful for them. With this thesis and tool we wanted to bring more value to the field of traditional ML that was lacking in research compared to deep learning, and therefore focused on making sure that we were putting traditional ML in the spotlight. We do however concur that there could be a benefit in also supporting deep learning, since this would make the tool helpful to more ML developers.

5.5 Threats to Validity

When conducting research it is always important to keep the validity of said research in mind. In this section, we primarily discuss the aspects of internal and external validity that have played a role in this project. The two stages of the project where validity was especially considered were the selection of papers as well as the interviews conducted to evaluate the toolkit.

5.5.1 Research Paper Selection

During the phase of the thesis project where we collected papers to include in the mapping, a couple of validity issues were recognised. For this phase, the largest concern was selection bias, since the toolkit is reliant on a list of papers that is curated by us the authors. We attempted to remedy any validity issues with the initial pool of papers by selecting three independent literature reviews and surveys, to hopefully get an unbiased first selection of papers. It was then also important for us to clearly state the rules with which we filtered away papers, as can be seen in Section 3.1.3, so that the selection process will be reproducible in the future. However, there are some issues with these rules, since different researchers might look at the *Potentially Generalisable Solution* tag definition with an array of different levels of harshness, which would lead to slightly different papers being filtered by the rules. We believe that this could be an issue with most similar studies, like literature reviews, and therefore hope that our definitions are clear enough for the results to be approximately the same would this study be replicated.

5.5.2 Evaluation Interviews

The evaluation interviews were meant to bring insight into the effectiveness of the toolkit. Any validity issues that these interviews could have would make this insight less valuable and less reliable. The validity issues that were therefore considered are discussed below.

From an external validity perspective, it was important for us to make sure that all of the interviewees did not have the same ML background. We wanted to make sure that the sample of developers had a wide variety of different experiences with different types of ML models and projects, so that the feedback they provided would not be biased towards what would be best for a certain type of ML model. All interviewees are from the same company, working in the automotive industry, which poses a validity concern, but we do not believe that this will create a bias towards ML models that are usually used within that domain. The reason for this is that several of the interviewees had previous experience with non-automotive industry ML, and the cases for the study were also designed to not depend on any specific attributes of that industry. The hope is that the evaluation was generalised in such a way that it is easily applicable to other contexts while still yielding similar results.

When it comes to internal validity it was a bit more difficult to identify and prevent unobserved variables from affecting the study before the interviews. One factor that we imagined would play a big role was the amount of time that the participant had to work with the toolkit during the study. Most of the participants had fairly busy schedules at work, and if they were not given enough time to take part in the study, this could negatively affect the accuracy of their feedback. If they dedicated a minimal amount of time to learning how to use the toolkit, then the results they got from applying it might not be accurate nor representative of the toolkit's effectiveness. We tried to remedy this issue by setting a deadline around a week after the initial meeting, to allow the participants to allocate time to learning and applying the

toolkit as best they could. We recognised that this could introduce other variables that we had not anticipated, but we believed the issue of "not enough time" to be large enough for it to have been a large concern if not taken care of.

6
Conclusion

The purpose of this thesis was to identify the goals that are currently being used for testing in SE that can be applied to the ML domain. A list of goal definitions was created from studying the literature, with the goal of creating a "common ground" for many of the definitions that differed from paper to paper. These goals could then be used as the basis for categorising papers, which led to the creation of the mapping. This mapping was then the foundation of the toolkit that helps a developer find out which testing techniques they should be utilising. The created toolkit contains the main tool in the form of a filterable table, representing the mapping, and several other documents that are meant to aid and guide the developer in applying it to the fullest. These are documents like the Definitions Document which provides definitions of the tags used in the main tool and the Paper Summaries Document which provides summaries for all the recommended papers. Then, there is also the Goal Identification Document that guides the developer to the goals that they should be testing their MLS for, and the Overview Trees Document that provides an overview of all the goals and testing techniques that are available in the toolkit. Finally, there is the Toolkit Guide Document which gives an example of how the toolkit can be applied to a real case, and the README Document which provides insight into what each of the other documents contains.

To evaluate the toolkit, eight employees from Volvo Cars participated in an study where they applied the toolkit to two cases based on real industrial projects. The study concluded with an interview, where the toolkit garnered generally favourable reviews in most aspects. It was considered a good replacement for the traditional way of identifying which goals to test for, which was more of an ad hoc process, and gave the developers easy access to a curated list of literature. Several of the interviewees said that without the toolkit they would have utilised Google Scholar, so they found that the toolkit helped speed up the process. Some negative feedback was also received, mostly targeting the current implementation of the main tool in Microsoft Lists, and the Overview Trees Document. Most interviewees would have liked to see the main tool implemented as its own application, and that the purpose of the Overview Trees Document should be clarified through more descriptions or a full rework. All the feedback provided excellent avenues for future work.

In the future, the toolkit could both be extended and improved. As discussed in Section 5.4, it could be extended to contain more than just research papers and to include resources for techniques applicable to deep learning. The toolkit could be transformed into a unified web application, instead of the current more modular ver-

sion containing the limited implementation in Microsoft Lists and its set of support documents. Out of these toolkit components, the Goal Identification Document has shown much potential to be useful even as a stand-alone tool. For the toolkit to remain useful in the long term, it needs to be updated with new resources on QA for ML, for example by automating the tagging process or community-sourcing the toolkit.

The hope with this thesis is that it will provide value both to researchers and developers. For the researchers, we have presented goals and their definitions, as well as how to identify when they are applicable. This has the potential to be extended and therefore help categorise future research. The developers in the industry, on the other hand, are presented with a tool that, based on the feedback from the evaluation, is capable of helping them with the process of selecting which testing techniques from SE they can apply to their own ML projects.

Bibliography

[1] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, 2020.

[2] R. Feldt, F. G. de Oliveira Neto, and R. Torkar, "Ways of applying artificial intelligence in software engineering," in *2018 IEEE/ACM 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, IEEE, 2018, pp. 35–41.

[3] L. Garcia Cuenca, J. Sanchez-Soriano, E. Puertas, J. Fernandez Andres, and N. Aliane, "Machine learning techniques for undertaking roundabouts in autonomous driving," *Sensors*, vol. 19, no. 10, p. 2386, 2019.

[4] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[5] I. Portugal, P. Alencar, and D. Cowan, "The use of machine learning algorithms in recommender systems: A systematic review," *Expert Systems with Applications*, vol. 97, pp. 205–227, 2018.

[6] I. Goodfellow, Y. Bengio, and A. Courville, "Machine learning basics," *Deep learning*, vol. 1, no. 7, pp. 98–164, 2016.

[7] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: A systematic mapping," *Empirical Software Engineering*, vol. 25, no. 6, pp. 5193–5254, 2020.

[8] J. Czerwonka, "Pairwise testing in real world," in *Proc. 24th Pacific Northwest Software Quality Conference, 2006*, 2006.

[9] T. Y. Chen, F.-C. Kuo, H. Liu, *et al.*, "Metamorphic testing: A review of challenges and opportunities," *ACM Comput. Surv.*, vol. 51, no. 1, 2018, ISSN: 0360-0300. DOI: 10.1145/3143561. [Online]. Available: https://doi.org/10.1145/3143561.

[10] S. Galhotra, Y. Brun, and A. Meliou, "Fairness testing: Testing software for discrimination," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2017, Paderborn, Germany: Association for Computing Machinery, 2017, 498–510, ISBN: 9781450351058. DOI: 10.1145/3106237.3106277. [Online]. Available: https://doi.org/10.1145/3106237.3106277.

[11] F. Dobslaw, F. G. de Oliveira Neto, and R. Feldt, "Boundary value exploration for software analysis," in *2020 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, IEEE, 2020, pp. 346–353.

[12] M. Čihák, "Introduction to applied stress testing," 2007.
[13] I. E. C. I. O. for Standardization., *Systems and software engineering : systems and software quality requirements and evaluation (SQuaRE) : system and software quality models.* ISO : IEC, 2011.
[14] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, "Beyond accuracy: Behavioral testing of nlp models with checklist," *arXiv preprint arXiv:2005.04118*, 2020.
[15] "Construction of a quality model for machine learning systems," *Software Quality Journal*, Jun. 2021, ISSN: 0963-9314. DOI: 10.1007/s11219-021-09557-y.
[16] A. Lavin, C. M. Gilligan-Lee, A. Visnjic, *et al.*, "Technology readiness levels for machine learning systems," *arXiv*, Jan. 2021. [Online]. Available: http://arxiv.org/abs/2101.03989.
[17] M. Z. I. Salman Sherin Muhammad Uzair khan, "A systematic mapping study on testing of machine learning programs," *arXiv preprint arXiv:1907.09427*, 2019.
[18] R. M. Shiffrin and K. Börner, *Mapping knowledge domains*, 2004.
[19] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering (EASE) 12*, 2008, pp. 1–10.
[20] S. Cha, R. N. Taylor, and K. Kang, *Handbook of software engineering*. Springer, 2019.
[21] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning with applications in R*. Springer, 2021.
[22] S. J. Russell and P. Norvig, *Artificial Intelligence: A modern approach*, 4th ed. Pearson Education, 2021.
[23] K.-J. Stol and B. Fitzgerald, "The abc of software engineering research," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 3, pp. 1–51, 2018.
[24] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM Comput. Surv.*, vol. 54, no. 6, 2021, ISSN: 0360-0300. DOI: 10.1145/3457607. [Online]. Available: https://doi.org/10.1145/3457607.
[25] S. Feuerriegel, M. Dolata, and G. Schwabe, "Fair ai," *Business & information systems engineering*, vol. 62, no. 4, pp. 379–384, 2020.
[26] R. Hamon, H. Junklewitz, and I. Sanchez, "Robustness and explainability of artificial intelligence," *Publications Office of the European Union*, 2020.
[27] M. Au-Yong-Oliveira, D. Canastro, J. Oliveira, J. Tomás, S. Amorim, and F. Moreira, "The role of ai and automation on the future of jobs and the opportunity to change society," in *World Conference on Information Systems and Technologies*, Springer, 2019, pp. 348–357.
[28] M. Shanahan, *The technological singularity*. MIT press, 2015.
[29] M. Borg, "The aiq meta-testbed: Pragmatically bridging academic ai testing and industrial q needs," Stefan, M. Daniel, W. Manuel, B. J. W. Dietmar, and Biffl, Eds., Springer International Publishing, 2021, pp. 66–77, ISBN: 978-3-030-65854-0.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[31] T. Joachims, T. Hofmann, Y. Yue, and C.-N. Yu, "Predicting structured objects with support vector machines," *Commun. ACM*, vol. 52, no. 11, 97–104, 2009, ISSN: 0001-0782. DOI: 10.1145/1592761.1592783. [Online]. Available: https://doi.org/10.1145/1592761.1592783.

[32] G. Giray, "A software engineering perspective on engineering machine learning systems: State of the art and challenges," *Journal of Systems and Software*, vol. 180, p. 111 031, 2021.

[33] B. Agarwal, S. Tayal, and M. Gupta, *Software engineering and testing*. Jones & Bartlett Learning, 2010.

[34] D. Boswell and T. Foucher, *The Art of Readable Code: Simple and Practical Techniques for Writing Better Code*. " O'Reilly Media, Inc.", 2011.

[35] M. G. Limaye, *Software testing*. Tata McGraw-Hill Education, 2009.

[36] W. Masri and F. A. Zaraket, "Coverage-based software testing: Beyond basic test requirements," in *Advances in Computers*, vol. 103, Elsevier, 2016, pp. 79–142.

[37] P. Oladimeji, M. Roggenbach, and H. Schlingloff, "Levels of testing," *Advance Topics in Computer Science*, 2007.

[38] K. Forsberg and H. Mooz, "The relationship of system engineering to the project cycle," in *INCOSE international symposium*, Wiley Online Library, vol. 1, 1991, pp. 57–65.

[39] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The oracle problem in software testing: A survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015. DOI: 10.1109/TSE.2014.2372785.

[40] D. Marijan, A. Gotlieb, and M. K. Ahuja, "Challenges of testing machine learning based systems," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, IEEE, 2019, pp. 101–102.

[41] M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "A comprehensive survey of trends in oracles for software testing," *University of Sheffield, Department of Computer Science, Tech. Rep. CS-13-01*, 2013.

[42] M. Felderer and R. Ramler, "Quality assurance for ai-based systems: Overview and challenges (introduction to interactive session)," in *Software Quality: Future Perspectives on Software Engineering Quality*, D. Winkler, S. Biffl, D. Mendez, M. Wimmer, and J. Bergsmann, Eds., Cham: Springer International Publishing, 2021, pp. 33–42, ISBN: 978-3-030-65854-0.

[43] L. Fischer, L. Ehrlinger, V. Geist, *et al.*, "Ai system engineering—key challenges and lessons learned," *Machine Learning and Knowledge Extraction*, vol. 3, no. 1, pp. 56–83, 2020.

[44] X. Huang, D. Kroening, W. Ruan, *et al.*, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100 270, 2020.

[45] H. B. Braiek and F. Khomh, "On testing machine learning programs," *Journal of Systems and Software*, vol. 164, p. 110 542, 2020.

[46] K. Hamada, F. Ishikawa, S. Masuda, *et al.*, "Guidelines for quality assurance of machine learning-based artificial intelligence.," in *SEKE*, 2020, pp. 335–341.

[47] G. Fujii, K. Hamada, F. Ishikawa, *et al.*, "Guidelines for quality assurance of machine learning-based artificial intelligence," *International Journal of Software Engineering and Knowledge Engineering*, vol. 30, no. 11n12, pp. 1589–1606, 2020.

[48] S. Masuda, K. Ono, T. Yasue, and N. Hosokawa, "A survey of software quality for machine learning applications," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2018, pp. 279–284. DOI: 10.1109/ICSTW.2018.00061.

[49] C. Tao, J. Gao, and T. Wang, "Testing and quality validation for ai software–perspectives, issues, and practices," *IEEE Access*, vol. 7, pp. 120 164–120 175, 2019. DOI: 10.1109/ACCESS.2019.2937107.

[50] F. Ishikawa, "Concepts in quality assessment for machine learning-from test data to arguments," in *International Conference on Conceptual Modeling*, Springer, 2018, pp. 536–544.

[51] Y. Nishi, S. Masuda, H. Ogawa, and K. Uetsuki, "A test architecture for machine learning product," in *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, IEEE, 2018, pp. 273–278.

[52] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, 2014, pp. 1–10.

[53] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – a systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009, Special Section - Most Cited Articles in 2002 and Regular Research Papers, ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2008.09.009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584908001390.

[54] *Google scholar*, 2004. [Online]. Available: https://scholar.google.com/ (visited on 04/12/2022).

[55] *Arxiv*, 1991. [Online]. Available: https://arxiv.org/ (visited on 02/05/2022).

[56] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[57] F. Chollet *et al.* "Keras." (2015), [Online]. Available: https://github.com/fchollet/keras (visited on 03/05/2022).

[58] F. U. Haq, D. Shin, S. Nejati, and L. C. Briand, "Comparing offline and online testing of deep neural networks: An autonomous car case study," in *2020 IEEE 13th International Conference on Software Testing, Validation*

and Verification (ICST), 2020, pp. 85–95. DOI: 10.1109/ICST46399.2020.00019.

[59] J. Tidwell, *Designing interfaces: Patterns for effective interaction design.* " O'Reilly Media, Inc.", 2010.

[60] *Microsoft lists*, 2020. [Online]. Available: https://www.microsoft.com/sv-se/microsoft-365/microsoft-lists (visited on 03/02/2022).

[61] *Intelligent diagramming*, 2008. [Online]. Available: https://www.lucidchart.com/pages/ (visited on 03/02/2022).

[62] *Google forms*, 2008. [Online]. Available: https://docs.google.com/forms/ (visited on 03/29/2022).

[63] P.-Y. Chen, *Securing ai systems with adversarial robustness*, 2021. [Online]. Available: https://research.ibm.com/blog/securing-ai-workflows-with-adversarial-robustness.

[64] K. Goel, N. Rajani, J. Vig, *et al.*, "Robustness gym: Unifying the nlp evaluation landscape," *arXiv preprint arXiv:2101.04840*, 2021.

[65] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: An efficient smt solver for verifying deep neural networks," in *Computer Aided Verification*, R. Majumdar and V. Kunčak, Eds., Cham: Springer International Publishing, 2017, pp. 97–117, ISBN: 978-3-319-63387-9.

[66] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel, "Fairness through awareness," Association for Computing Machinery, 2012, pp. 214–226, ISBN: 9781450311151. DOI: 10.1145/2090236.2090255. [Online]. Available: https://doi.org/10.1145/2090236.2090255.

[67] M. J. Kusner, J. Loftus, C. Russell, and R. Silva, "Counterfactual fairness," I Guyon, U. V. Luxburg, S Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/a486cd07e4ac3d270571622f4f316ec5-Paper.pdf.

[68] C. Dwork, "Differential privacy," in *Automata, Languages and Programming*, M. Bugliesi, B. Preneel, V. Sassone, and I. Wegener, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12, ISBN: 978-3-540-35908-1.

[69] A. Sharma and H. Wehrheim, "Higher income, larger loan? monotonicity testing of machine learning models," in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2020, pp. 200–210.

[70] N. Hynes, D Sculley, and M. Terry, "The data linter: Lightweight, automated sanity checking for ml data sets," *NIPS MLSys Workshop*, vol. (Vol. 1), 2017. [Online]. Available: https://github.com/brain-research/data-linter.

[71] S. Krishnan, M. J. Franklin, K. Goldberg, and E. Wu, "Boostclean: Automated error detection and repair for machine learning," *arXiv preprint arXiv:1711.01299*, 2017.

[72] E. Breck, N. Polyzotis, S. Roy, S. E. Whang, and M. Zinkevich, "Data validation for machine learning," 2019, pp. 334–347. [Online]. Available: https://proceedings.mlsys.org/paper/2019/file/5878a7ab84fb43402106c575658472fa-Paper.pdf.

[73] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, and A. Grafberger, "Automating large-scale data quality verification," vol. 11, Asso-

ciation for Computing Machinery, 2018, pp. 1781–1794. DOI: 10.14778/3229863.3229867.

[74] E. Breck, S. Cai, E. Nielsen, M. Salib, and D Sculley, "The ml test score: A rubric for ml production readiness and technical debt reduction," in *2017 IEEE International Conference on Big Data (Big Data)*, IEEE, 2017, pp. 1123–1132.

[75] F. Cabitza, A. Campagner, F. Soares, *et al.*, "The importance of being external. methodological insights for the external validation of machine learning models in medicine," *Computer Methods and Programs in Biomedicine*, vol. 208, p. 106 288, 2021.

[76] E. Lanus, L. J. Freeman, D. R. Kuhn, and R. N. Kacker, "Combinatorial testing metrics for machine learning," in *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, IEEE, 2021, pp. 81–84.

[77] S. Udeshi, P. Arora, and S. Chattopadhyay, "Automated directed fairness testing," Association for Computing Machinery, Inc, Sep. 2018, pp. 98–108, ISBN: 9781450359375. DOI: 10.1145/3238147.3238165.

[78] S. Ackerman, O. Raz, M. Zalmanovici, and A. Zlotnick, "Automatically detecting data drift in machine learning classifiers," *ArXiv*, 2021.

[79] X. Xie, J. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Application of metamorphic testing to supervised classifiers," in *2009 Ninth International Conference on Quality Software*, IEEE, 2009, pp. 135–144.

[80] S. Nakajima and H. N. Bui, "Dataset coverage for testing machine learning computer programs," in *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, IEEE, 2016, pp. 297–304.

[81] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," vol. 84, Apr. 2011, pp. 544–558. DOI: 10.1016/j.jss.2010.11.920.

[82] Y. Yang, Z. Li, H. Wang, C. Xu, and X. Ma, "Towards effective metamorphic testing by algorithm stability for linear classification programs," *Journal of Systems and Software*, vol. 180, p. 111 012, 2021.

[83] A. Sharma and H. Wehrheim, "Testing machine learning algorithms for balanced data usage," Institute of Electrical and Electronics Engineers Inc., Apr. 2019, pp. 125–135, ISBN: 9781728117355. DOI: 10.1109/ICST.2019.00022.

[84] S. Srisakaokul, Z. Wu, A. Astorga, O. Alebiosu, and T. Xie, "Multiple-implementation testing of supervised learning software," in *Workshops at the thirty-second AAAI conference on artificial intelligence*, 2018.

[85] S. Herbold and T. Haar, "Smoke testing for machine learning: Simple tests to discover severe bugs," *Empirical Software Engineering*, vol. 27, 2 Mar. 2022, ISSN: 1382-3256. DOI: 10.1007/s10664-021-10073-7.

[86] S. Dutta, W. Zhang, Z. Huang, and S. Misailovic, "Storm: Program reduction for testing and debugging probabilistic programming systems," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 729–739.

[87] A. Ramanathan, L. L. Pullum, F. Hussain, D. Chakrabarty, and S. K. Jha, "Integrating symbolic and statistical methods for testing intelligent systems: Applications to machine learning and computer vision," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2016, pp. 786–791.

[88] X. Yin, V. Musco, I. Neamtiu, and U. Roshan, "Statistically rigorous testing of clustering implementations," Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 91–98, ISBN: 9781728104928. DOI: 10.1109/AITest.2019.000-1.

[89] T. Schaul, I. Antonoglou, and D. Silver, "Unit tests for stochastic optimization," *arXiv preprint arXiv:1312.6055*, 2013.

[90] A. Saranti, B. Taraghi, M. Ebner, and A. Holzinger, "Property-based testing for parameter learning of probabilistic graphical models," in *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, Springer, 2020, pp. 499–515.

[91] M. S. Ahmed, F. Ishikawa, and M. Sugiyama, "Testing machine learning code using polyhedral region," Association for Computing Machinery, Inc, Nov. 2020, pp. 1533–1536, ISBN: 9781450370431. DOI: 10.1145/3368089.3417043.

[92] M. Bartolo, T. Thrush, R. Jia, S. Riedel, P. Stenetorp, and D. Kiela, "Improving question answering model robustness with synthetic adversarial data generation," *arXiv preprint arXiv:2104.08678*, 2021.

[93] W. Yang and T. Xie, "Telemade: A testing framework for learning-based malware detection systems," *The Workshops of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. [Online]. Available: www.aaai.org.

[94] R. Soklaski, J. Goodwin, O. Brown, M. Yee, and J. Matterer, "Tools and practices for responsible ai engineering," *arXiv preprint arXiv:2201.05647*, 2022.

[95] T. Wang, D. Yang, and X. Wang, "Identifying and mitigating spurious correlations for improving robustness in nlp models," *arXiv preprint arXiv:2110.07736*, 2021.

[96] P. L. Benedick, J. Robert, and Y. L. Traon, "A systematic approach for evaluating artificial intelligence models in industrial settings," *Sensors*, vol. 21, 18 Sep. 2021, ISSN: 14248220. DOI: 10.3390/s21186195.

[97] Y. Elazar, N. Kassner, S. Ravfogel, *et al.*, "Measuring and improving consistency in pretrained language models," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 1012–1031, 2021.

[98] J. Liu, R. Takanobu, J. Wen, *et al.*, "Robustness testing of language understanding in task-oriented dialog," *arXiv preprint arXiv:2012.15262*, 2020.

[99] A. Aggarwal, S. Shaikh, S. Hans, S. Haldar, R. Ananthanarayanan, and D. Saha, "Testing framework for black-box ai models," IEEE Computer Society, May 2021, pp. 81–84, ISBN: 9781665412193. DOI: 10.1109/ICSE-Companion52605.2021.00041.

[100] E. Soremekun, S. S. Udeshi, and S. Chattopadhyay, "Astraea: Grammar-based fairness testing," *IEEE Transactions on Software Engineering*, 2022.

[101] V. Prabhakaran, B. Hutchinson, and M. Mitchell, "Perturbation sensitivity analysis to detect unintended model biases," *arXiv*, Oct. 2019. [Online]. Available: http://arxiv.org/abs/1910.04210.

[102] A. Aggarwal, P. Lohia, S. Nagar, K. Dey, and D. Saha, "Black box fairness testing of machine learning models," Association for Computing Machinery, Inc, Aug. 2019, pp. 625–635, ISBN: 9781450355728. DOI: 10.1145/3338906.3338937.

[103] J. Chakraborty, H. Tu, S. Majumder, and T. Menzies, "Can we achieve fairness using semi-supervised learning?" *arXiv preprint arXiv:2111.02038*, 2021.

[104] C. Murphy, K. Shen, and G. Kaiser, "Automatic system testing of programs without test oracles," in *Proceedings of the eighteenth international symposium on Software testing and analysis*, 2009, pp. 189–200.

[105] S. Nakajima, "Generalized oracle for testing machine learning computer programs," in *International Conference on Software Engineering and Formal Methods*, Springer, 2017, pp. 174–179.

[106] X. Xie, Z. Zhang, T. Y. Chen, Y. Liu, P.-L. Poon, and B. Xu, "Mettle: A metamorphic testing approach to assessing and validating unsupervised machine learning systems," *IEEE Transactions on Reliability*, vol. 69, no. 4, pp. 1293–1322, 2020.

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

[108] D. Moreira, A. P. Furtado, and S. Nogueira, "Testing acoustic scene classifiers using metamorphic relations," in *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*, IEEE, 2020, pp. 47–54.

[109] Y. Qin, H. Wang, C. Xu, X. Ma, and J. Lu, "Syneva: Evaluating ml programs by mirror program synthesis," in *2018 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, IEEE, 2018, pp. 171–182.

[110] A. Sharma and H. Wehrheim, "Automatic fairness testing of machine learning models," in *IFIP International Conference on Testing Software and Systems*, Springer, 2020, pp. 255–271.

[111] D. Saha, A. Aggarwal, and S. Hans, "Data synthesis for testing black-box machine learning models," in *5th Joint International Conference on Data Science & Management of Data (9th ACM IKDD CODS and 27th COMAD)*, 2022, pp. 110–114.

[112] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?" explaining the predictions of any classifier," vol. 13-17-August-2016, Association for Computing Machinery, Aug. 2016, pp. 1135–1144, ISBN: 9781450342322. DOI: 10.1145/2939672.2939778.

[113] S. Ma, Y. Aafer, Z. Xu, *et al.*, "Lamp: Data provenance for graph based machine learning algorithms through derivative computation," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 786–797.

[114] G. Barash, E. Farchi, I. Jayaraman, O. Raz, R. Tzoref-Brill, and M. Zalmanovici, "Bridging the gap between ml solutions and their business requirements using feature interactions," in *Proceedings of the 2019 27th ACM Joint*

Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 1048–1058.

[115] S. Santos, B. Silveira, V. Durelli, R. Durelli, S. Souza, and M. Delamaro, "On using decision tree coverage criteria for testing machine learning models," Association for Computing Machinery, Sep. 2021, pp. 1–9, ISBN: 9781450385039. DOI: 10.1145/3482909.3482911.

[116] J. Zhu, T. Long, and A. Memon, "Automatically authoring regression tests for machine-learning based systems," IEEE Computer Society, May 2021, pp. 374–383, ISBN: 9780738146690. DOI: 10.1109/ICSE-SEIP52600.2021.00049.

[117] J. Bozic, O. A. Tazl, and F. Wotawa, "Chatbot testing using ai planning," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, IEEE, 2019, pp. 37–44.

[118] M. H. Asyrofi, F. Thung, D. Lo, and L. Jiang, "Crossasr: Efficient differential testing of automatic speech recognition via text-to-speech," in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, IEEE, 2020, pp. 640–650.

[119] H. Zhu, D. Liu, I. Bayley, R. Harrison, and F. Cuzzolin, "Datamorphic testing: A method for testing intelligent applications," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, 2019, pp. 149–156. DOI: 10.1109/AITest.2019.00018.

[120] M. P. Shahri, M. Srinivasan, G. Reynolds, D. Bimczok, I. Kahanda, and U. Kanewala, "Metamorphic testing for quality assurance of protein function prediction tools," Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 140–148, ISBN: 9781728104928. DOI: 10.1109/AITest.2019.00017.

[121] J. Guichard, E. Ruane, R. Smith, D. Bean, and A. Ventresque, "Assessing the robustness of conversational agents using paraphrases," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, IEEE, 2019, pp. 55–62.

[122] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8271–8277.

[123] Z. Sun, J. M. Zhang, M. Harman, M. Papadakis, and L. Zhang, "Automatic testing and improvement of machine translation," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 974–985.

[124] A. Dwarakanath, M. Ahuja, S. Sikand, *et al.*, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2018, pp. 118–128.

[125] H. Guo, C. Tao, and Z. Huang, "Metamorphic testing for plant identification mobile applications based on test contexts," in *International Conference on Mobile Computing, Applications, and Services*, Springer, 2020, pp. 209–223.

[126] R. Yan, S. Wang, Y. Yan, H. Gao, and J. Yan, "Stability evaluation for text localization systems via metamorphic testing," *Journal of Systems and Software*, vol. 181, p. 111 040, 2021.

[127] A. Albarghouthi and S. Vinitsky, "Fairness-aware programming," in *Proceedings of the Conference on Fairness, Accountability, and Transparency*, 2019, pp. 211–219.

[128] F. Tramer, V. Atlidakis, R. Geambasu, *et al.*, "Fairtest: Discovering unwarranted associations in data-driven applications," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, 2017, pp. 401–416.

[129] R. Angell, B. Johnson, Y. Brun, and A. Meliou, "Themis: Automatically testing software for discrimination," Association for Computing Machinery, Inc, Oct. 2018, pp. 871–875, ISBN: 9781450355735. DOI: 10.1145/3236024.3264590.

[130] V. D. O. Neves, M. E. Delamaro, and P. C. Masiero, "Combination and mutation strategies to support test data generation in the context of autonomous vehicles," vol. 8, Inderscience Publishers, 2016, pp. 464–482. DOI: 10.1504/IJES.2016.080388.

[131] A. Aniculaesei, J. Grieser, A. Rausch, K. Rehfeldt, and T. Warnecke, "Towards a holistic software systems engineering approach for dependable autonomous systems," in *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, 2018, pp. 23–30.

[132] K. Heckemann, M. Gesell, T. Pfister, K. Berns, K. Schneider, and M. Trapp, "Safe automotive software," in *Knowledge-Based and Intelligent Information and Engineering Systems*, A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, and L. C. Jain, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 167–176, ISBN: 978-3-642-23866-6.

A

Project Cases for Study

Below are the two cases that were used during the evaluation to determine how well the designed toolkit was functioning, the results of which are presented in Section 4.4. These cases were not meant to be phrased in a flawless way or be a good match for the toolkit, but are instead based on real projects at Volvo Cars. This was done to bring realism into the cases, which would potentially reveal concepts that the toolkit had not been designed to handle.

A.1 Case 1

This case is about a system that is tasked to decide which potential customers are going to be targeted by an ad campaign. The company gathers data from different countries, and wishes to predict who is likely to buy a new product within the coming six months, based on data. These kinds of predictions will be done regularly between ad campaigns. Rather than sending ads to everyone in their customer base, they want to find a subset that is most likely to be convinced to make a purchase after receiving ads.

The solution is a churn model (but not applied to a subscription service like it usually is) adapted to the problem above. The model rates a customer between 0 and 1; 1 being very likely to buy a new product, and 0 being not likely. The model's predictions are used to group customers into N segments of equal size. This is done by predicting, for each person, the likelihood that the person will purchase a product within the coming six months. Then all of these potential buyers are sorted based on their score (likelihood) and assigned into N segments.

The goal is not to predict exactly who is going to buy a new product, but instead to roughly order the customers along the probability line in order of how likely they are to buy a new product. The model has a lot of false positives, meaning customers that are rated highly but are not going to buy a new product. The high rate of false positives is not really a big concern since, as mentioned, the goal is not to accurately identify which customers are actually going to buy a new product, but just their potential in doing so.

Once these segments have been identified, the company sends advertisements to the segment with the highest scoring potential buyers (segment H). As a kind of sanity check for the model, the company also sends advertisements to the lowest scoring

group (segment L), in order to verify that the difference in "lift" is logical/reasonable given the predicted scores. Lift here refers to the percent increase/decrease in the number of purchases from potential buyers who got targeted by an ad campaign versus a control group. If the model performs well, then it is expected that the lift of segment H should be noticeably greater than the lift of segment L.
The current model is a redeployment of a model from another country. It has not yet been through a full iteration with data from this country, meaning that it's still in an early phase. After training the model the first time, an iteration can be summarised as:

1. Prediction
2. Targeted marketing at predicted "most likely to buy" segment (and "least likely to buy" as a sanity check)
3. Evaluation of marketing effectiveness on segments
4. Retraining of model
5. Back to step 1.

An iteration takes around 6 months, mostly to measure the effect of the targeted marketing. The model is trained through batches, and retraining only takes one hour. There is data drift, so retraining on fresh data is meant to help deal with this.

The model uses a variety of data in its predictions. It has customer data, which includes personal data, historical purchase data, and if the customer has been in contact with the company recently. Most of the personal data is not used in predictions, but some of it is used to clean the data before it is inserted into the model (for example removing employees of the company to not create a bias). Data that is included is how long the customers have owned their current product. There are some privacy concerns, but this is currently mostly being handled in the data ingestion stage. The model does also not actively use sensitive data (protected attributes), so there are currently no checks related to detecting discriminatory bias. The developers expect some data drift as well, but if the prediction distributions change drastically there might be some issues with the model. For example, the predicted number of "likely customers" drops from 10000 to 0.

The development team currently performs some different types of testing. Integration testing is being done but it takes a lot of time to run the test suite, which takes time away from other kinds of tests. The team is also using unit tests for the code behind the model, but they are not testing the features of the model yet.

Logistic regression is used in the model. This type of model is relatively easy to interpret, since the output can be seen as the likelihood of success. It is transparent in how the end output is calculated, and is not too complex unlike random forest or deep neural networks. This property of the model is helpful since some customers are interested in why they get targeted by an ad campaign, so this means that the company can offer an explanation that is not too difficult to understand.

A.2 Case 2

In this project, the goal is to predict the value of an unreliable measurement. This value represents the current state of a product that degrades over time. The value is therefore between 100 and 0 where 100 is the optimal state and anything below shows that some degradation has happened. The reason for measuring this value is that there are warranty concerns from the company's side, since if the product degrades too quickly they might have to compensate the customer or replace the product.

The main issue is that the real measurement is incredibly noisy and can therefore be seen as quite unreliable. The product is very complex and there are many causes as to why the measurement behaves poorly, including aspects of the product's environment. To address the poor reliability of the measurement, the company has therefore created a regression model that is meant to estimate the measurement based on features of the product. The model accounts for a subset of the things that could cause the bad measurements, because it is too complex and not feasible to account for all causes. If the model's prediction and the real measurement differ by a lot, then this means that the real measurement should be retaken to verify that it is correct. In the case of remeasuring, a more robust measurement is taken that needs more computing time, so the company wants the model to be tuned such that at most 5% of them need to be remeasured. Also, the company wants to avoid basing decisions on information about the country of the product to avoid bias that can affect the accuracy of the predictions due to spurious correlations.

The company has set up an ML pipeline for this problem. In the first stages of the pipeline, there are checks for faulty data. The data is generated from multiple sources, although the same kind of source. Currently, the company mostly runs tests on the source code of the pipeline, but there are very few tests on the models overall. They would like more tests all throughout the pipeline, but their main priority right now is the stability of the model, meaning that it should not be too affected by outliers in the data. They have access to both the specification and source code of all components, so there is no limitation on the tests in that regard.

Retraining the model is important, mostly due to data drift. They have conducted experiments that indicate that retraining too often leads to worse results, but that retraining is still essential. It's also necessary to train different models for different subpopulations of data that represent different releases of products. These products differ enough that a single model would not be enough to accurately predict their degradation purely from the features. To make training more efficient the developers have thought about adding testing to the data, which would allow them to make assertions on what data should be used or not for training. There are also concerns about not making the data biased, so instead of removing bad data it could also just be labelled as such.

B

Results for Project Cases

In this appendix, the results are presented from the task of applying tags to each case. This is meant as a supplement to the results presented in Section 4.4.

The time spent on solving the cases can be seen in Figure B.1 and Figure B.2. For Case 1, the times range from 10 to 120 minutes, and the majority of subjects spent under one hour. For Case 2, it ranges from 10 to 90 minutes.

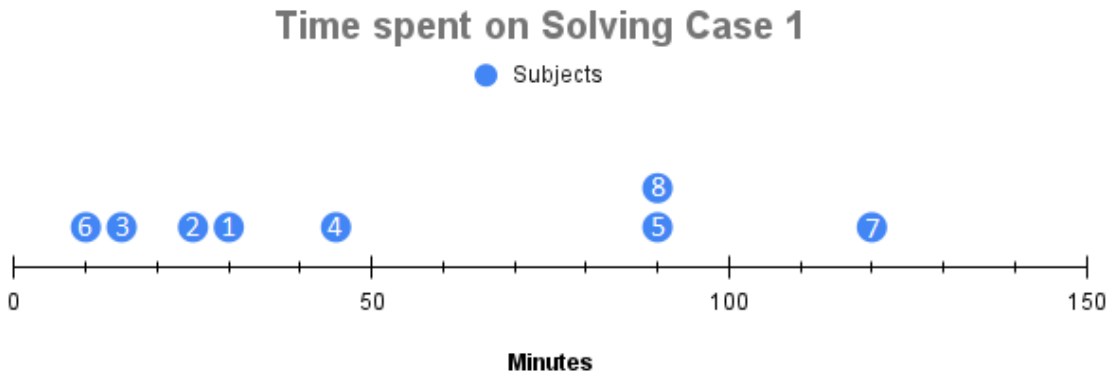


Figure B.1: The time spent by each subject on applying the toolkit to Case 1. The numbers indicate which subject a dot refers to.

All of the eight subjects worked on applying tags to Case 1, the results of which can be seen in the respective columns in Table B.1 and Table B.2. A grey cell in the tables represents that the subject for that column selected the tag for that specific row, whereas a white cell indicates that a specific tag was not selected by that subject. For example, it can be seen that subject 2 selected the tag *Noise Robustness* to be included in their search filter, while subject 3 did not select this tag. The tag category **Limitations** was excluded from the table since no subject used any of its tags when filtering, partly because they either did not understand how to apply this category or because the tags were deemed irrelevant for the case.

Five out of eight subjects worked on applying tags to Case 2, since the remaining three (subjects 3, 4, 5) could not allocate enough time to take part in the study because of time constraints from work. For this reason, only subjects 1, 2, 6, 7 and 8 are shown in Table B.3. Like for Case 1, a grey box means that the tag

		Subjects					
Tag Categories	**Tags**	**1**	**2**	**3**	**4**	**5**	**6**
Goal	Adversarial Robustness						
	Noise Robustness (Stability)		■		■		
	Individual Fairness	■		■			
	Group Fairness	■		■			
	Counterfactual Fairness	■					
	Interpretability	■	■	■			■
	Safety						
	Privacy	■		■			■
	Security	■		■			■
	Correctness	■	■				■
	Completeness			■	■		
	Monotonicity	■		■			
	Predictive Performance						■
	Efficiency	■					
Component Under Test	Data	■	■	■		■	
	Learning Program	■		■			
	Framework						
	Model	■	■				■
	MLS	■	■	■			
ML Task	General	■	■	■	■	■	
	Classification		■			■	■
	Regression	■	■	■	■		
	Clustering	■					
	Dimensionality Reduction						
	Reinforcement Learning						
	Structured Prediction						
	Ranking	■					
	Anomaly Detection						
Access Level	Black-box	■	■	■			
	Data-box	■	■	■			
	White-box	■		■		■	
Offline/Online	Online		■	■			
	Offline	■		■			

Table B.1: The selected tags by subjects 1-6 for case 1. Each row represents a tag from the main tool, and each column represents a subject from the study. If a tag was selected by a subject, given the case description, then the corresponding cell has been coloured grey.

		Subjects														
Tag Categories	**Tags**	**7**					**8**									
Goal	Adversarial Robustness															
	Noise Robustness (Stability)															
	Individual Fairness									■						
	Group Fairness										■					
	Counterfactual Fairness															
	Interpretability	■														
	Safety															
	Privacy				■											
	Security															
	Correctness															
	Completeness		■											■		
	Monotonicity															
	Predictive Performance						■	■	■							
	Efficiency					■										
Component Under Test	Data		■											■		
	Learning Program														■	
	Framework															
	Model			■												
	MLS															
ML Task	General													■		
	Classification								■	■	■					
	Regression						■									
	Clustering							■								
	Dimensionality Reduction															
	Reinforcement Learning															
	Structured Prediction															
	Ranking	■														
	Anomaly Detection															
Access Level	Black-box															
	Data-box															
	White-box															■
Offline/Online	Online															
	Offline						■	■	■	■	■	■				

Table B.2: The selected tags by subjects 7 and 8 for case 1. Each row represents a tag from the main tool, and each column represents a subject from the study. If a tag was selected by a subject, given the case description, then the corresponding cell has been coloured grey.

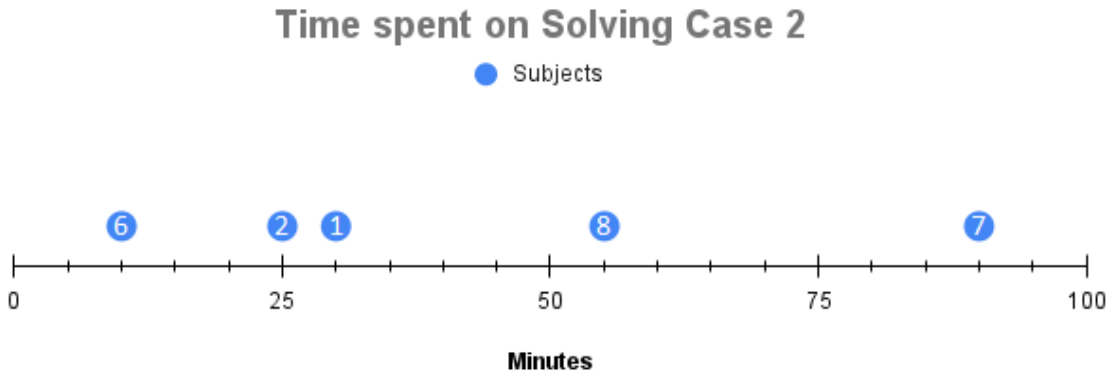


Figure B.2: The time spent by each subject on applying the toolkit to Case 2. The numbers indicate which subject a dot refers to.

was selected while a white box means it was not. Subjects 7 and 8 continued using the alternative approach of applying several smaller filters separately as opposed to using just one filter. The interviewees did not provide much reasoning for their tags for this case, other than that it was based on what they had identified within the case description.

In both cases, subjects 1-6 answered with a single set of tags, which was the method recommended in the Toolkit Guide Document. Subjects 7 and 8 on the other hand did not select just one set of tags, but instead several smaller sets applied separately. Due to this, there are multiple columns for these two subjects in Table B.1. For each filter applied, the mapping presented them with a different set of papers.

During the interviews the subjects got the opportunity to comment on their choices of tags. Subject 1 based their choices on the Goal Identification Document. Subject 2 originally wanted to only select the *Regression* tag, but this yielded to few papers so *Classification* was added as they argued that regression could perhaps be formulated as a classification problem. As for the General tag, most of the interviewees did not select it at first, but when asked why they changed their minds. During the interview with subject 6 this follow-up question was missed, which might be the reason for why the General tag is not selected. Subject 5 wanted to select *Data* together with the other tags they selected, but this yielded only two papers that did not feel relevant which made them decide not to include the *Data* tag in their final selection. Most of these reasonings were also used for Case 2.

Once the filters were applied and the list of papers reduced, some subjects managed to find a few potentially useful papers for Case 1. Subjects 2, 4, and 8 found at least two papers each, while subjects 5 and 7 found exactly two and three papers, respectively. Subject 3 and 6 did not look at the papers because of the limited amount of time they could dedicate to the study. Subject 1 explained that it would not be possible for them alone to determine which papers would be useful, and hence chose no papers, because it felt necessary to discuss with the product owner in an

		Subjects											
Tag Categories	**Tags**	**1**	**2**	**6**	**7**				**8**				
Goal	Adversarial Robustness												
	Noise Robustness (Stability)												
	Individual Fairness												
	Group Fairness				■					■			
	Counterfactual Fairness												
	Interpretability	■											
	Safety												
	Privacy										■		
	Security	■										■	
	Correctness	■	■			■							■
	Completeness	■		■									
	Monotonicity	■											
	Predictive Performance	■				■							
	Efficiency	■	■										
Component Under Test	Data	■	■	■			■						
	Learning Program	■											
	Framework												
	Model	■	■	■				■					
	MLS	■	■										
ML Task	General	■	■		■				■	■	■	■	■
	Classification												
	Regression	■	■	■					■	■	■	■	■
	Clustering												
	Dimensionality Reduction												
	Reinforcement Learning												
	Structured Prediction												
	Ranking												
	Anomaly Detection	■											
Access Level	Black-box	■	■										
	Data-box	■	■										
	White-box	■	■										
Offline/Online	Online	■	■						■	■			
	Offline		■										

Table B.3: The tags selected by all subjects for case 2 in the study.

iterative way.

The subjects were also asked whether they found sensible choices of papers for Case 2 once they had applied their filters. Subject 1 did not look at the papers, for the same reason as in Case 1, while subject 6 did not specify any papers. Subject 2 said there were at least two interesting papers. Subject 7 and 8 found four and three papers, respectively.

Subject 7 made three different searches for Case 1. When asked about the tags they wanted to select, it first seemed like the same approach that subjects 1-6 used, as they went over each tag they thought were relevant. Some tags, once added to the search filter, they said resulted in zero papers remaining in the mapping, so the subject chose to exclude them. The specific tags this concerned was *Privacy* and *Efficiency*, separately, when applied together with other selected tags. It became apparent that subject 7 had made several smaller filtered searches in order to find some relevant papers.