Model driven engineering for machine learning components: A systematic literature review

Hira Naveed, Chetan Arora, Hourieh Khalajzadeh, John Grundy, Omar Haggag

Monash University, Clayton, VIC, Australia
Deakin University, Burwood, VIC, Australia

A R T I C L E   I N F O

Dataset link: MDE4ML SLR data (Original data)

Keywords:
Model driven engineering
Software engineering
Artificial intelligence
Machine learning
Systematic literature review

A B S T R A C T

Context: Machine Learning (ML) has become widely adopted as a component in many modern software applications. Due to the large volumes of data available, organizations want to increasingly leverage their data to extract meaningful insights and enhance business profitability. ML components enable predictive capabilities, anomaly detection, recommendation, accurate image and text processing, and informed decision-making. However, developing systems with ML components is not trivial; it requires time, effort, knowledge, and expertise in ML, data processing, and software engineering. There have been several studies on the use of model-driven engineering (MDE) techniques to address these challenges when developing traditional software and cyber–physical systems. Recently, there has been a growing interest in applying MDE for systems with ML components.

Objective: The goal of this study is to further explore the promising intersection of MDE with ML (MDE4ML) through a systematic literature review (SLR). Through this SLR, we wanted to analyze existing studies, including their motivations, MDE solutions, evaluation techniques, key benefits and limitations.

Method: Our SLR is conducted following the well-established guidelines by Kitchenham. We started by devising a protocol and systematically searching seven databases, which resulted in 3934 papers. After iterative filtering, we selected 46 highly relevant primary studies for data extraction, synthesis, and reporting.

Results: We analyzed selected studies with respect to several areas of interest and identified the following: (1) the key motivations behind using MDE4ML; (2) a variety of MDE solutions applied, such as modeling languages, model transformations, tool support, targeted ML aspects, contributions and more; (3) the evaluation techniques and metrics used; and (4) the limitations and directions for future work. We also discuss the gaps in existing literature and provide recommendations for future research.

Conclusion: This SLR highlights current trends, gaps and future research directions in the field of MDE4ML, benefiting both researchers and practitioners.

1. Introduction

The ability of Machine Learning (ML) to autonomously learn data patterns and predict outcomes has tremendous potential to solve complex problems [1]. The proliferation of data and advances in hardware processing capabilities have contributed to the rapid growth and adoption of ML in recent years. ML components have now becoming integral to software systems with application domains including healthcare [2], finance [3], transport [4], entertainment [5], and many more. However, the development, integration, and maintenance processes of traditional software components and ML components differ significantly [6]. Traditional software components are deterministic and developed through a deductive development process by explicitly coding the system’s required behavior. In contrast, ML component development follows an inductive process by exploring data and recognizing patterns [7]. ML components are dynamic and it is extremely difficult to specify their ‘well-defined’ behavior [6,8]. Unlike software engineers, ML engineers need to perform exploratory steps to identify and curate datasets, select relevant features, select the most suitable ML model, tune hyperparameters, monitor the ML component, and re-train...
The main contributions of this SLR are as follows:

1. Identification, analysis, data extraction, and synthesis of 46 primary studies highly relevant to MDE4ML;
2. Insights into current trends in MDE for ML components, e.g., most MDE approaches focus on supervised learning;
3. Key limitations in existing studies on MDE for ML components, e.g., scalability – one of the most important concerns in ML development – is seldom considered in MDE4ML; and
4. Key future research directions and recommendations for further studies on MDE for ML.

The rest of the paper is organized as follows: Section 2 provides an overview of the background and related work on MDE4ML. Section 3 presents details of our research methodology. Section 4 reports our findings from the selected primary studies. Section 5 addresses the threats to validity. Section 6 discusses the interesting results of our SLR and recommendations for future research, and Section 7 concludes the paper.

2. Background and related work

2.1. Model-driven engineering

Model-driven Engineering (MDE) is a software development methodology that relies on models as the primary artifacts that drive the development process [10, 22]. This differs from traditional software development processes such as waterfall and agile, where the focus is on development phases like requirements engineering, design, and implementation, and models are only used as auxiliary artifacts to support these activities and serve as documentation [22]. The focus of MDE is on the continual refinement and transformation of models, beginning with computation-independent models (CIMs), to platform-independent models (PIMs) and then platform-specific models (PSMs) [12]. Finally, these models are transformed into code, documentation, configurations, and tests for the software system.

MDE relies on two key aspects: abstraction and automation [23]. Models are abstractions of complex entities; they hide unwanted information so modelers can easily focus on areas of interest [12, 24]. In MDE, models are automatically transformed into artifacts such as code, documentation, and other models to achieve various goals such as merging, translation, refinement, refactoring, or alignment [12]. These transformations help reduce developers’ manual effort and production time by generating executable artifacts — leading to improved software quality, reduced complexity, and decreased development time and effort [25]. There are two types of transformations in MDE: (1) Model-to-Text (M2T) transformations, for a given input model a M2T transformation produces a textual artifact such as code or documentation as output; and Model-to-model (M2M) transformations, for a given input model an M2M transformation produces a different kind of model, for example translating a model from one language to another [12].

A model is created in a modeling language, conforming to a metamodel that defines the concepts and properties of that language, also known as the abstract syntax [12]. The notation used to represent the model – textual or graphical – is known as the concrete syntax [12]. There are two types of modeling languages: general-purpose languages (GPL) and domain-specific languages (DSL). GPLs are intended for modeling generic concepts applicable to multiple domains; some examples include the Unified Modeling Language (UML) [26], Petri-nets [27] and finite state machines [28]. On the other hand, a DSL has modeling concepts tailored to a specific domain or context, like SysML for embedded systems, HTML for web page development, and SQL for database queries [12].
While exploring the literature, one encounters terms similar to MDE: examples include model-driven architecture (MDA), model-driven development (MDD), and model-based engineering (MBE). MDA is an architectural standard [29] developed by the Object Management Group (OMG) [30] for MDD. MDD refers to automatically generating artifacts from models, whereas MDE has a broader scope and includes analysis, validation [31], interoperability of artifacts and reverse engineering [12]. MBE is a lighter version of MDE, where models are not necessarily the central focus of the engineering process; however, they provide critical support [12]. This SLR primarily focuses on MDE.

### 2.2. Machine learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that enables machines to learn patterns from data without being explicitly programmed [32]. ML algorithms are fed with existing data to train them and produce an ML model. This trained ML model then has the capability to infer, i.e., predict outcomes for new data inputs or also commonly known as ML model inference [33]. For example, an ML model trained on stock prices for a company till September 2023 can predict stock prices in the following months. ML is preferable when solving problems that would require very complex and difficult-to-maintain traditional algorithms [34]. Since ML algorithms can learn autonomously, they reduce complexity and facilitate easier maintenance [34]. This ability of ML to minimize complexity, learn from changing data, and make future predictions is immensely valuable for businesses [35]. According to a recent survey [36], organizations report that applying ML increases employee efficiency by 20%, innovation by 17%, and lowers costs by 16% – leading to increased adoption of ML in practical settings [36].

ML can further be divided into three broad categories: supervised learning, unsupervised learning, and reinforcement learning. The most suitable ML approach depends on the specific problem and data. Supervised learning is when an ML algorithm is trained on a labeled dataset that has labels to define the meaning of data [33]. For example, a dataset with images labeled as “cat” or “not cat” images. Supervised learning algorithms learn to make classifications or predictions by learning patterns and relationships in labeled data [33,35]. Once the algorithm is trained, the performance is evaluated on unseen data. Some popular supervised learning algorithms include linear regression, decision trees, naïve Bayes classifier, support vector machines (SVM), random forest, and artificial neural networks (ANNs) [35]. Supervised model applications include fraud detection and recommender systems [33].

Unsupervised learning is when an ML algorithm is trained on an unlabeled dataset with few or no labels to define the meaning of data [33,35]. Unsupervised learning algorithms attempt to understand hidden patterns in data and group similar data together creating a classification of the data [33]. Unsupervised learning works without any guidance, hence it is most suitable for large volumes of data when classifications are unknown and data cannot be labeled [33]. Evaluating the performance of such algorithms can be challenging due to the lack of ground truth. Some popular unsupervised techniques include clustering, k-means, principal component analysis, and association rules [35]. Applications of unsupervised models include customer segmentation and clustering user reviews [33].

Reinforcement learning is when an ML algorithm receives feedback on actions to guide the behavior toward an optimal outcome [33,35]. Reinforcement learning algorithms are not trained with datasets; instead, they learn from trial and error in a simulated environment or a real-world environment [33]. Desired behaviors are rewarded and reinforcement learning algorithms attempt to maximize rewards through successful decisions [33,35]. These algorithms are most suitable when sequential decision-making is required, interaction with an environment is possible and feedback is available. Some popular reinforcement learning algorithms are Q-learning, temporal difference learning, hierarchical reinforcement learning, and policy gradient [35]. Applications of reinforcement learning include robotics, self-driving cars, and game playing [35].

### 2.3. Model-driven engineering for machine learning (MDE4ML)

Developing and managing systems with ML models and components is challenging. Some aspects of this complexity are immature requirements specification [6,37], constantly evolving data [38], lack of ML domain knowledge [39], integration with traditional software [40], responsible use of ML [39], and deployment and maintenance of ML models [41].

These complexities introduce several challenges. For example, Nils Baumann et al. [38] describe how challenging it is to handle changing datasets; ML engineers have to manually merge new and old datasets and re-train the entire ML model; Benjamin Jahi et al. [42] point out how challenging it is to describe the dataset and neural network requirements to satisfy customer expectations; Benjamin Benni et al. [43] state how the development of a correct ML pipeline is a highly demanding task, data scientists must have knowledge and experience to go through numerous data pre-processing and ML models to select the best one; and Kaan Koseler et al. [44] mention the difficulties developers face when attempting to use ML techniques with big data, developers need to acquire knowledge of the problem space, domain and ML concepts. There is a need for solutions to efficiently and effectively address these challenges [45].

A synergy between MDE and ML development exists, where software models are leveraged to drive the development and management of ML components [39,41]. This should not be confused with AI or ML for MDE (AI4MDE), where intelligent agents and recommenders support users in modeling and related activities [31,46]. The application of MDE for ML-based systems (MDE4ML) offers many potential benefits to developers, such as reduced complexity [15,41], development effort, and time [39,47]. Domain experts, software engineers and ML novices can also take advantage of ML through the abstraction and automation of MDE [15,48]. Additionally, MDE can also improve the quality of the ML-based system through easier maintainability, scalability, reusability, and interoperability [12].

### 2.4. Key MDE4ML related work

MDE4ML has received growing attention from researchers in recent years. We found six relevant secondary studies comprising SLRs, scoping reviews, and surveys. In their SLR [45], the authors identify 15 primary studies on MDE for AI and analyze them with respect to MDE practices for the development of AI systems and the stages of AI development aligned with CRoss Industry Standard Process for Data Mining (CRISP-DM) [49] methodology. However, this study only considers a small subset of studies and performs a shallow analysis with no details about goals, end-users, types of models, implemented tools, and evaluation. A second SLR [50] reviews 24 papers on MDE for ML in the context of Big data analytics. This study has a narrower scope compared to ours and provides only a brief overview of the models, approaches, tools, and frameworks in the studies. In a third SLR [51], 31 studies on no/low code platforms for ML applications are reviewed. This study is limited to no/low code approaches and therefore misses out on many other MDE for ML studies. A scoping review is presented in [52] on MDE for ML in IoT applications. The study examines 68 studies in depth; however, the review focuses more on MDE for IoT applications and only four of the selected studies apply ML techniques. A preliminary survey on DSLs for ML in Big data is presented in [53], with an extended version in [54]. These surveys do not follow a systematic review process, include studies only for big data, and briefly highlight the DSLs and frameworks in the studies. From the analysis of existing literature, we found that the available secondary studies consist of limited subsets of papers on MDE for ML, lack analysis of key areas like goals, end-users, ML aspects, MDE approach details, evaluation methods, and limitations, and often do not follow a systematic and rigorous review process. Therefore, we aim to address these gaps in this SLR.
3. Research methodology

We conducted a Systematic Literature Review (SLR) on Model-driven Engineering (MDE) approaches for systems with machine learning (ML) components (MDE4ML). This review aims to analyze existing primary studies and synthesize significant findings to guide future research and practice. This literature review has been conducted in conformance with the systematic literature review guidelines for SE presented by Kitchenham et al. [21].

Fig. 2 provides an overview of our review process. We divide the work into three stages: planning, conducting, and reporting. During the planning stage, we identified the need for this SLR, formulated research questions (RQs), and defined the SLR protocol. In the conducting stage, all authors collaborated to formulate search strings and select databases. The first author searched all selected databases and removed duplicate studies to create an initial pool of papers. Over multiple iterations, the first author filtered studies based on predefined criteria. Cross-validation was used to check searches and selection and to resolve ambiguities with the other authors. Forward and backward snowballing was used to identify other highly relevant primary studies. This resulted in 46 primary studies for analysis. Data extraction and synthesis activities for the final 46 papers, listed in Appendix A, were performed by the first author under the close supervision of the other authors. In the final stage reporting, we documented all the significant findings and analyzed threats to the validity of the SLR.

3.1. Research questions

We developed four research questions (RQs) to identify the motivations, solutions, evaluation techniques, and limitations of MDE4ML.

RQ1. What is the motivation behind applying MDE approaches to systems with ML components? – RQ1 explores the key motivations and goals for applying MDE for ML-based systems in our selected primary studies. We further looked into other relevant aspects, such as the ML techniques, application domain, end users, and outcomes.

RQ2. Which MDE approaches and tools are presented in the literature for systems with ML components? – RQ2 examines the MDE solutions presented in the literature for ML-based systems. We identified and classified the modeling languages applied, transformations and automation levels supported, and the artifacts generated. We also explore the MDE tools available and the underlying meta-tools and frameworks. From the ML perspective, we looked at the ML phases, training data, and ML libraries considered in the primary studies.

RQ3. How are existing studies on MDE4ML evaluated? – RQ3 investigates the evaluation techniques, metrics, datasets, and settings applied in the selected primary studies.

RQ4. What are the limitations and future work of existing studies on MDE4ML? – RQ4 identifies the limitations of current studies and key future research challenges of MDE4ML. In this context, we further examine the studies in terms of limitations in their approach, evaluation, and solution quality.

3.2. Study selection

Our database search results contain a number of irrelevant studies. Therefore, we developed detailed selection criteria as part of our SLR protocol. These criteria are divided into inclusion and exclusion criteria (shown in Table 1) that we used to filter primary studies. For this review, we only consider primary English-language studies that focus on MDE for ML-based systems. The studies must be academic and have their full text available online. We excluded any irrelevant papers or did not have enough information to extract, such as vision papers, posters, magazine articles, keynotes, opinion papers, and experience papers. This SLR focuses on MDE4ML approaches, so we did not include any AI4MDE papers. These criteria were applied to all papers to select the most relevant ones, and discussions between all authors resolved any ambiguities in the study filtering process.

3.3. Search strategy

Fig. 3 provides an overview of our search process, with all the steps detailed below.

3.3.1. Search string formulation

For automated search in online databases, a search string was defined using key terms and alternative terms. We realize that model-based software engineering and low code/no code are not the same as MDE; however, we included them in the search string since they are sometimes used interchangeably in the literature. This string was tested on seven online databases, IEEE Xplore, ACM Digital Library, Springer, Wiley, Scopus, Web of Science, and Science Direct, and refined over several iterations to get the most relevant results. Each time the search was executed, the first author randomly selected a sample of 5–7 papers from the search results and skimmed through them to ensure relevance and further refine the string. We combine search string keywords in the same category using the OR operator and keywords in different categories using the AND operator.
categories using the AND operator. For example, all ML keywords are joined by the OR operator, whereas the AND operator joins the ML and MDE keywords. The final search string below was slightly modified for some databases to get the best results.

### 3.3.2. Automated search and duplicate removal

The automated search was conducted in March 2023. The final search string was executed on the selected online database search engines to extract an initial pool of 3934 papers. The search was restricted to academic articles, including journals, conference papers, and workshop papers, with no time range specified. We used the default search for all databases except Scopus, where we performed an advanced search on titles, abstracts, and keywords. The list of resultant papers was downloaded, and a Python script was developed to remove duplicates. The script also removed conference or workshop papers from the list if a journal version was found with the same title and authors. After duplicate removal, we were left with 3570 papers for further screening.

### 3.3.3. Filtering studies

We filtered the 3570 potentially relevant papers in three iterations by applying the inclusion and exclusion criteria. The first screening based on title and abstract yielded 72 potentially relevant papers. The significant reduction in the paper count at this stage was due to the following reasons: (1) the automated search resulted in ∼100 AI4ML papers because of similar keywords (Exclusion criteria 4), (2) the performance of the Springer search engine was subpar, resulting in ∼1500 completely irrelevant papers (Exclusion criteria 1, 2, 10), (3) the automated search was conducted on the full text of the papers which resulted in ∼1800 studies that mentioned the keywords from the search string but not in the context of MDE4ML (Exclusion criteria 1 and 2). In the second screening, we skimmed through the entire paper and further narrowed it down to 55 papers. The third and final screening was done during data extraction, resulting in 32 highly relevant papers. The drop in paper count in the second and third screening was primarily because some highly relevant studies had insufficient information to extract for the SLR (Exclusion criteria 7) and others were found to be irrelevant upon detailed reading (Exclusion criteria 1 and 2).

During the filtering process, we maintained a list of all papers on Google Sheets and color-coded them on relevance, i.e., green for ‘relevant’, yellow for ‘might be relevant’, and red for ‘irrelevant’. The color codes helped authors focus and classify the papers that needed further attention.
To account for any missed papers in the automated search process, we performed a manual search following the snowballing guidelines by Wohlin [55]. The 32 selected papers and related work papers were snowballed in forward and backward directions over three iterations until no new suitable papers were found. We gathered eight (8) papers through our forward snowballing and six (6) papers through our backward snowballing.

3.4. Data extraction and synthesis

We created a Google Form with 40 questions corresponding to our four RQs to ensure all required data was extracted from the papers. This form was divided into five sections: the first section for general information and publication trends, including title, authors, publication venue, and citation count; the second section for motivations, goals, application domain, and users; the third section for MDE approaches for ML-based systems; the fourth section for evaluation techniques and application domain, and users; the third section for MDE approaches; the second section for motivations, goals, and objectives; and the last section for limitation and future challenges. The answer options for the questions in the form consisted of 23 short answer options for the questions in the form consisted of 23 short answer options, ten long answers, two checkboxes and 14 radio buttons. For quality assessment, we ran pilot tests; the first author extracted data for six papers and compared it with data extracted by the other authors for the same papers. A close match was found between both, after which the first author extracted data for all the remaining papers. During pilot tests, small updates were made to improve the Google Form, for example, adding check boxes for common answers and improving the structure of the question. From the data extraction, we gathered qualitative and quantitative data for the next stage of data synthesis. The first author performed synthesis using various graphs, figures, and tables under the guidance of the other authors.

3.5. Quality assessment

We devised a five-point scoring system to assess the quality of the selected primary studies to answer five predefined questions. The scores range from very poor (1) and inadequate (2) to moderate (3), good (4), and excellent (5). Such QA scoring for evaluating the quality of primary studies is commonplace in SLRs [14,56]. The quality assessment questions we used are based on Kitchenham's guidelines [21] and are shown below:

- QA1: Are the aims clearly stated?
- QA2: Is the solution clearly defined?
- QA3: Are the measures used clearly defined?
- QA4: Does the report have implications for practice?
- QA5: Does the report discuss how the results add to the literature?

The results of our quality assessment are shown in Appendix B. Questions QA3–QA5 are only applicable to studies that provide an evaluation and are marked as not applicable (NA) for all other studies without an explicit evaluation component. From our quality assessment, we found that 19/46 included papers were of good quality, 15/46 were of average quality, and 12/46 were poor quality. We did not exclude any papers since MDE4ML is an emerging research area and our selection of 46 studies was already limited. Furthermore, we aimed to minimize any publication bias in our research.

4. Results

This section presents the results of our SLR on MDE approaches for systems with ML components. We organize these findings based on the four previously mentioned research questions.

4.1. Publication trends

We selected 46 primary studies on MDE4ML, published over a span of 16 years, from 2008 to 2023, as shown in Fig. 4(a). Appendix A lists the full citation details of these primary studies. There are no studies included from 2009, 2010, and 2013. The number of papers published from 2011 to 2018 was low compared to 2019 and onward. There was a drop in 2020, which could have been due to the COVID-19 outbreak. However, we are not sure. The high number of studies in 2021 and 2022 show the increasing research interest in MDE for ML and we hope to see the same trend in the future. Our search process was conducted in March 2023 and could thus be the reason for the low numbers in 2023.

Most of the studies were published in conferences and journals, contributing to ~43.5% and ~37% of the total paper count, followed by workshops, making up for the remaining ~19.6%. The selected 46 studies belonged to 34 different venues; the venues with two or more published studies are shown in Table 2. Model Driven Engineering Languages and Systems (MODELS) conference, co-located workshops, and Software and Systems Modeling (SoSyM) journal have the highest number of papers in each publication type category. This is unsurprising as these are all highly reputed publication venues dedicated to MDE.

In the following sections, we describe the features of MDE solutions for ML-based systems. Fig. 5 shows a feature tree of the examined features of the primary studies. The features in the tree were derived from the data extraction categories based on the RQs for this SLR.

4.2. RQ1 - motivation for MDE4ML approaches

4.2.1. Motivation, goals, and objectives

All the analyzed studies described their goals, objectives, and motivations for applying MDE techniques to systems with ML components. We divided these goals into three high-level categories: effort reduction, increased stakeholder understanding, and quality improvement. These categories are not mutually exclusive and many studies fall under more
Effort Reduction was the most common aim mentioned in (43 out of 46 studies) MDE4ML papers, and focuses on effort reduction in development (e.g., P4, P9, P13), integration (e.g., P1, P8, P20), monitoring (e.g., P3, P6, P13) and managing systems (e.g., P3, P11, P12) with ML components. Abstraction is one way to achieve effort reduction by creating models of complex systems that hide unnecessary details to focus only on the required aspects. For example, the goal of P5 and P9 is to use models to reduce complexity when developing ML solutions for cyber-physical systems. Automation is another way to
reduce development effort, artifacts such as code, configurations, and documentation can be generated from the models without any manual effort. For example, the goal of P10 is to generate dataset description documents from models with dataset details such as structure, provenance, and social concerns. P28 aims to automatically generate code for neural networks by transforming MDE models into code. Integration of ML components into the rest of the system is not a trivial task; the objective of P11 is to ease this process through an ML artifact model. Another example is P8, which aims to simplify integration between neural network components and non-ML components to create reusuable neural networks. Monitoring refers to observing a system at runtime to ensure desired behavior. Despite being important, setting up monitoring mechanisms is a complex task. P6 aims to support ML experts through MDE by setting up a monitoring solution to detect performance drops. System management and data management are two other goals mentioned in the studies that correspond to less effort required for management. P12 identifies the challenges of managing dynamic datasets and presents a model-driven approach with the goal of automating data retraining and version management. For system management, a model-driven approach is described in P13 with the objective of managing ML-based analytics services on IoT devices.

<table>
<thead>
<tr>
<th>Goal</th>
<th>Sub-goal</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Integration</td>
<td>P1, P5, P8, P11, P20, P22</td>
</tr>
<tr>
<td></td>
<td>Monitoring</td>
<td>P3, P6, P13</td>
</tr>
<tr>
<td></td>
<td>System Management</td>
<td>P3, P13</td>
</tr>
<tr>
<td></td>
<td>Data management</td>
<td>P11, P12</td>
</tr>
</tbody>
</table>

| Quality improvement                                         | Reusability       | P8, P19, P23, P25                                                      |
|                                                            | Extensibility     | P1, P8, P25, P26                                                      |
|                                                            | Standardization   | P1, P7, P10                                                           |
|                                                            | Responsible ML    | P2, P3, P10                                                           |
|                                                            | Interoperability  | P7, P45                                                               |
|                                                            | Maintainability   | P11                                                                   |
|                                                            | Scalability       | P16                                                                   |
|                                                            | Reliability       | P16                                                                   |

| Increased stakeholder understanding                        | Support non-ML Experts | P2, P17, P24, P28, P34, P39                                           |
|                                                            | Common Language     | P14, P15, P32, P35, P36                                                 |

**Increased Stakeholder Understanding** is the least common category (11 out of 46 studies) with papers that aim to support collaboration (e.g., P35, P36) and improve system understanding in stakeholders other than ML experts (e.g., P2, P17, P34). We found six studies that support non-ML experts in developing ML-based systems; these studies use domain-specific terminology and hide complex technical details. In P39, the goal is to provide a DSL to support software engineers in specifying requirements for neural networks. This is, otherwise, a challenging task since most software engineers are not experienced in deep learning [5]. We also found five studies focused on providing a common language for easier communication and stakeholder collaboration. For example, P35 aims to facilitate multi-disciplinary teams developing data analytics and machine learning solutions using DSLs.

### 4.2.2. Machine learning techniques

Table 4 shows the different subsets of ML techniques considered in the included primary studies. Of the 46 primary studies, 31 studies (67%) explicitly focused on supervised ML, and 4 studies (9%) on reinforcement learning. None of the studies focused explicitly on unsupervised learning or clustering. The remaining 11 studies (24%) are termed as generic ML studies, i.e., (i) the studies did not explicitly specify the type of ML their solution targets (e.g., P41, P43); (ii) the studies target both supervised and unsupervised learning (e.g., P22, P32), and (iii) the studies target supervised, unsupervised and reinforcement learning (e.g., P13). For instance, P22 offers an MDE solution for IoT devices using generic ML and supports various supervised and unsupervised ML algorithms. Supervised machine learning papers are further categorized into traditional supervised learning, where studies use models such as decision trees, linear regression and naive Bayes (e.g., P19, P35, P38); neural networks, where studies use neural networks and deep learning (e.g., P29, P44, P46) and traditional and neural networks, where the studies support traditional supervised learning models and neural networks (e.g. P27, P34, P45). An example of a study exclusively for reinforcement learning is P40, which presents a DSL to model multi-agent reinforcement learning in distributed systems.

### 4.2.3. Application domain

This section highlights the eleven application domains found in the selected studies. 52% of studies mention an application domain with cyber–physical systems (CPS) and subsets of it being the most common. We found seven studies for generic CPS (P5, P8, P9, P12, P22, P23, and...
P31), two for manufacturing systems (P36 and P45), two for autonomous vehicles (P1 and P3), one for smart homes (P20), one for traffic signal control (P30), one for satellite communication system (P33) and one for network planning (P25). The second most common application domain found in the primary studies was big data analytics in five studies (P13, P17, P21, P35, and P43) and data analytics in three studies (P32, P34, and P38). We also found one study (P24) for developing social bots. The MDE solutions proposed in almost half the studies (P2, P4, P6, P7, P10, P11, P14–P16, P18, P22, P24, P32, P34, P35, P36, P38).

Other roles

Table 4

<table>
<thead>
<tr>
<th>Machine learning technique</th>
<th>Sub-type</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic machine learning</td>
<td>–</td>
<td>P9, P10, P13, P15, P21, P22, P23, P32, P33, P41, P43</td>
</tr>
<tr>
<td>Supervised machine learning</td>
<td>Traditional</td>
<td>P2, P6, P16, P17, P19, P35, P38</td>
</tr>
<tr>
<td></td>
<td>Neural Networks</td>
<td>P1, P3, P4, P7, P8, P11, P12, P14, P20, P24, P25, P26, P28, P29, P36, P39, P42, P44, P46</td>
</tr>
<tr>
<td></td>
<td>Traditional and Neural Networks</td>
<td>P27, P31, P34, P37, P45</td>
</tr>
<tr>
<td>Unsupervised machine learning</td>
<td>–</td>
<td>-</td>
</tr>
<tr>
<td>Reinforcement learning</td>
<td>–</td>
<td>P5, P18, P30, P40</td>
</tr>
</tbody>
</table>

Table 5

<table>
<thead>
<tr>
<th>User category</th>
<th>End user</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML-related roles</td>
<td>ML Engineer</td>
<td>P1, P5, P6, P7, P8, P10, P11, P12, P13, P15, P25, P37, P40, P41, P42, P43, P44, P46</td>
</tr>
<tr>
<td></td>
<td>Data Analyst/ Engineer/ Scientist</td>
<td>P2, P10, P12, P14, P16, P18, P22, P24, P32, P34, P35, P36, P38</td>
</tr>
<tr>
<td>Software &amp; Systems roles</td>
<td>Software Engineer</td>
<td>P1, P2, P3, P4, P9, P11, P13, P19, P20, P22, P23, P27, P28, P29, P30, P39</td>
</tr>
<tr>
<td></td>
<td>Systems Engineer</td>
<td>P9, P31, P33</td>
</tr>
<tr>
<td></td>
<td>Business Analyst</td>
<td>P35</td>
</tr>
<tr>
<td></td>
<td>Formal Methods Analyst</td>
<td>P14</td>
</tr>
<tr>
<td>Other roles</td>
<td>Domain Expert</td>
<td>P17, P21, P24, P25, P26, P28, P30, P32, P35, P36, P45</td>
</tr>
</tbody>
</table>

P31), two for manufacturing systems (P36 and P45), two for autonomous vehicles (P1 and P3), one for smart homes (P20), one for traffic signal control (P30), one for satellite communication system (P33) and one for network planning (P25). The second most common application domain found in the primary studies was big data analytics in five studies (P13, P17, P21, P35, and P43) and data analytics in three studies (P32, P34, and P38). We also found one study (P24) for developing social bots. The MDE solutions proposed in almost half the studies (P2, P4, P6, P7, P10, P11, P14–P16, P18, P19, P26–P29, P37, P39–P42, P44, and P46) were generic and could be applied in any domain.

4.2.4. End users

Fig. 6(c) provides an overview of the end user distribution in the primary studies. The proposed MDE solutions for ML-based systems are intended for three different categories of users, as shown in Table 5. The ML-related roles category includes ML engineer and data analyst/ engineer/scientist; the software and systems roles category includes software engineer, system engineer, business analyst, and formal methods analyst; and the last category other roles includes domain expert. Of the 46 selected studies, 13 presented approaches were for more than one user category, while the remaining 33 focused on a single end-user category. Of all the studies (including overlaps), 18 focused on MDE solutions for ML engineers, followed by solutions for software engineers in 16 studies, and 11 focused on approaches for domain experts. An example of each of these three main categories is: P1 proposes an approach for ML engineers to model deep neural network architectures for autonomous vehicle perception; P9 presents a modeling approach for software engineers to model ML components integrated on edge IoT devices for data analytics; and P17 enables domain experts to represent ML problems as models and derive code from them.

4.2.5. Contributions

We identified eleven different contributions in the analyzed MDE4ML approaches, tabulated in Table 6. We show tool-specific contributions in the Venn diagram in Fig. 6(b). The most common contribution (in 35 out of 46 studies) is a code generator that transforms model(s) into code. For example, in P23, the code generator transforms CPS domain models to code for domain classes and then weaves ML code into it. An alternate approach in P17 reads a domain-specific model for baseball analytics using binary classification and produces ML classification code. Other kinds of generators presented in studies include model generator (10 out of 46 studies) that transforms an input model(s) into an output model(s) and text generator (8 out of 46 studies) that transforms model(s) into text other than code, e.g., documentation. The second common contribution (in 30 out of 46 studies) is a domain-specific language (DSL), which includes graphical and textual modeling languages intended for a particular domain. A graphical DSL is proposed in P1 to model deep learning-based computer vision tasks in autonomous vehicles. A textual DSL is presented in P2 to mitigate biases in ML datasets, to model the dataset, training methods, bias metrics, and bias mitigation methods. Another common contribution (in 21 out of 46 studies) is an MDE framework for systems with ML components. A framework to monitor ML components in CPS is introduced in P3. The study leverages goal models to evaluate the ML components at runtime and ensure correct behavior in uncertain environments. Several other outcomes were also identified such as models (6 out of 46 studies), modeling approach (6 out of 46 studies), modeling language extension (5 out of 46 studies), ML knowledge base (4 out of 46 studies), data synthesizer (3 out of 46 studies), OCL constraints (1 out of 46 studies study), APIs (1 out of 46 studies study), and meta-modeling language (1 out of 46 studies study). However, in comparison, these remaining contributions were significantly lower in number. Most papers make multiple contributions, e.g., P25 provides a framework, DSL, model generator, and code generator.

Fig. 7 shows a bubble chart of the study goals and contributions against their relevant ML aspect. The size of the bubble depicts the frequency of studies in that category. For example, related to the requirements engineering aspect of ML components, the goal of three studies is to achieve automation. In comparison, three studies provide a code generator as their contribution. While analyzing this figure, we found that in the selected studies, the preferred ML aspects are the design and development of ML components, and training of ML components. In contrast, monitoring of ML components, and documentation were the most neglected aspects.
Table 6
Contributions of studies.

<table>
<thead>
<tr>
<th>Contribution</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tool</strong></td>
<td></td>
</tr>
<tr>
<td>Model-to-Text Transformer</td>
<td>Code Generator</td>
</tr>
<tr>
<td>Text Generator</td>
<td>P3, P6, P10, P15, P21, P26, P35, P36</td>
</tr>
<tr>
<td><strong>Model-to-Model Transformer</strong></td>
<td>Model Generator</td>
</tr>
<tr>
<td>P1, P3, P4, P18, P25, P29, P33, P40, P41, P42</td>
<td></td>
</tr>
<tr>
<td><strong>Domain-specific Language (DSL)</strong></td>
<td></td>
</tr>
<tr>
<td>Framework</td>
<td>P1, P3, P5, P7, P8, P12, P13, P18, P19, P23, P24, P25, P26, P27, P29, P31, P34, P36, P37, P38, P44</td>
</tr>
<tr>
<td><strong>Framework</strong></td>
<td></td>
</tr>
<tr>
<td>Model</td>
<td>P4, P11, P13, P20, P31, P37</td>
</tr>
<tr>
<td>Modeling approach</td>
<td></td>
</tr>
<tr>
<td>P9, P11, P14, P16, P20, P30</td>
<td></td>
</tr>
<tr>
<td>Modeling language extension</td>
<td></td>
</tr>
<tr>
<td>P5, P17, P22, P33, P45</td>
<td></td>
</tr>
<tr>
<td><strong>ML knowledge base</strong></td>
<td></td>
</tr>
<tr>
<td>P15, P16, P20, P34</td>
<td></td>
</tr>
<tr>
<td><strong>Data synthesizer</strong></td>
<td></td>
</tr>
<tr>
<td>P31, P39, P45</td>
<td></td>
</tr>
<tr>
<td><strong>OCL constraints</strong></td>
<td></td>
</tr>
<tr>
<td>P27</td>
<td></td>
</tr>
<tr>
<td><strong>API</strong></td>
<td></td>
</tr>
<tr>
<td>P40</td>
<td></td>
</tr>
<tr>
<td><strong>Meta-modeling language</strong></td>
<td></td>
</tr>
<tr>
<td>P23</td>
<td></td>
</tr>
</tbody>
</table>
4.3. RQ2 - MDE4ML approaches and tools used

4.3.1. Modeling characteristics

**Model Representation** refers to the graphical notations or textual notations used for expressing a model, also known as the concrete syntax. Fig. 8(a) shows our findings; half of the studies (23 out of 46 studies) use graphical models (P1, P3, P4, P13–P19, P21, P24–P27, P30, P32–P38), nearly half of the studies (21 out of 46 studies) use textual models (P2, P5–P12, P20, P23, P28, P31, P39–P46) and only a small number of studies (2 out of 46 studies) use both graphical and textual models (P22 and P29). An example of graphical representation is P35. It generates graphical SysML models of the architecture of autonomous systems with ML and non-ML components using restricted natural language requirements. An example of textual model representation is P41, wherein the authors build a textual domain-specific language (OptiML) to generate ML code for heterogeneous hardware platforms. P29 uses both graphical and textual representations to model artificial neural networks.

**Modeling Languages** refer to specific languages or notations for creating abstract representations of systems in graphical, textual, or combined form. As discussed in Section 2.1, modeling languages can be classified as general purpose languages (GPLs) such as UML and iStar, domain-specific languages (DSLs), or extensions of an existing language, such as UML profiles. As shown in Fig. 8(b), we found the majority of studies (34 out of 46 studies) propose a new DSL (P1, P2, P6–P8, P10–P16, P19, P21, P23, P25–P30, P32, P34–P44, and P46), a significantly fewer fraction (9 out of 46 studies) use a GPL (P3, P4, P9, P17, P18, P20, P24, P31, and P45), and only 3 out of 46 studies extend an existing language (P5, P22, and P33). A DSL for MLops is introduced in P37 to automate the ML pipeline. The authors use Kubernetes and a blockchain-based infrastructure. P6 presents a DSL for monitoring ML models and selects Kubernetes as a platform. In contrast, P3 leverages GSN and KOAS general-purpose goal models to monitor ML components. An example of a study that extends an existing modeling language is P5. The authors extend the deep learning DSL family MontiAnna to support reinforcement learning, particularly for cyber–physical systems.

4.3.2. Supported machine learning aspects

From the included primary studies we found 17 different ML aspects that were addressed by studies, 30 out of 46 studies focused on two or more ML aspects, whereas the remaining 16 studies only focused on one ML aspect. Fig. 9(a) shows the distribution of ML aspects in studies, these include the development stages of ML components and other related aspects. The development stages consist of requirements engineering, to gather, analyze and specify requirements for ML-based systems (e.g., P3, P35, P39); data preprocessing, cleaning and transforming data into a suitable format for input to the ML model (e.g., P13, P14); design and development of ML models/components, algorithm selection, feature engineering and coding for the ML component (e.g., P7, P24, P36); training, feeding data to a ML model for it to learn patterns and relationships (e.g., P22, P25, P42); evaluation, evaluating the ML model’s performance on unseen (test) data before deployment (e.g., P1, P27, P39); deployment, releasing the ML component into the production environment (e.g., P9, P35, P37); integration, integrating ML components with other software or system components (e.g., P20, P23, P31); inference, using a trained ML model to make predictions on unseen data after deployment (e.g., P3, P6, P21); monitoring, observing the ML component at runtime for correct behavior (e.g., P3, P6) and management, managing ML components after deployment through retraining, parameter adjustment, version control and maintenance (e.g., P12, P26, P31). Other related ML aspects consist of data generation, artificially creating data for training ML models (e.g., P39, P45); data storage, storing and organizing datasets (e.g. P13); data visualization, creating visual representations of data to uncover patterns, relationships, and insights (e.g., P26); documentation, recording various aspects of the ML component such as dataset details, architecture, training setup and more (e.g., P10); ML pipeline development, building the process for developing a ML component from data pre-processing to deployment (e.g., P4, P16, P32); and ML knowledge base development, creating a...
repository of information about ML models and resources (e.g., P15, P16, P34). More than half the studies (28 out of 46 studies) were targeted towards designing and developing ML models or components, while nearly half (22 out of 46 studies) were for training the ML models. A reasonable portion of studies (10 out of 46 studies) addressed deploying ML components with a primary focus on practices such as DevOps and MLOps. The least explored ML aspects were documentation, data storage, visualization with only one study, and monitoring and data generation with two studies.

An ML framework is a comprehensive platform that offers a structured foundation to build, train, and deploy ML models. In other words, they offer various tools and functionalities that cover the entire ML model development lifecycle. An ML library is a collection of functions and methods that allow developers to perform specific ML tasks, e.g., splitting datasets into training and test sets. Libraries are generally more lightweight than frameworks and focus on particular ML pipeline aspects. We examined the ML frameworks and libraries used in the studies, our results summarized in Table 7. Based on our review of the 46 studies, Tensorflow was the most used ML framework, followed by MXNet, whereas Weka was the most used ML library, followed by Scikit-learn and Numpy. See Table 7

### 4.3.3. Tool support

**Model Transformations.** MDE involves transforming models into text or different kinds of models. We refer to these as *model-to-text (M2T)* and *model-to-model (M2M)* transformations respectively, as described in Section 2. We found the largest portion of studies (35 out of 46 studies) using solely M2T transformations (P2, P5–P8, P10–P17, P19–P24, P26–P28, P30–P32, P34–P39, P43–P46). For example, P7 generates a Predictive Model Markup Language (PMML) file from a model with ML model details. A small portion of studies (4 out of 46 studies) use only M2M transformations (P18, P29, P33, and P40). For instance, P18 converts goal models into formal specification models to generate domain simulations for reinforcement learning. We also found some studies (7 out of 46 studies) that apply both M2M and M2T transformations in their MDE solutions (P1, P3, P4, P9, P25, P41, and P42). For example, P25 applies M2T transformation to convert a model with ML prediction and dataset details into deep learning code, and M2M transformation to convert models with wireless network plan details into models with component allocation and resource usage. Another example is P35 in which models with big data analytics tasks are converted into ML code and documentation. We also found that all 46 included studies use only forward engineering.
Generated Artifacts. The artifacts generated by the studies include ML model code or training code in 36 studies (P1–P3, P5, P7–P9, P11–P14, P16, P17, P19–P25, P27, P28, P30–P32, P35–P39, P41–P46), software or intermediate models in 15 studies (P1, P4, P6, P7, P9, P14, P18, P20, P25, P28, P29, P33, P40–P42), deployment configurations in 8 studies (P3, P6, P13, P16, P26, P31, P34, P37), and datasets or subsets of datasets in 4 studies (P31, P36, P39, P45). The remaining were text files in 2 studies (P10 and P35), API code in 2 studies (P4 and P8), recommendation rules or queries in 2 studies (P15 and P38), and metamodels 1 study (P26). We further examined the languages in which text and code were generated. Our findings are shown in Fig. 9(b), the highest number of studies generated artifacts in Python (15 out of 46 studies), Java (10 out of 46 studies) and C++ (4 out of 46 studies). Study P11 generates ML components in Python and C++ using artifact and reference models.

Automation Levels. Fig. 8(c) shows the automation categories supported by the transformations, fully automated and partially automated. The former category works independently while the latter requires some manual effort. Transformations in the majority of studies (38 out of 46 studies) were fully automated (P1, P2, P4–P10, P12–P18, P20, P22–P25, P27–P37, P40–P44, and P46), whereas, in a few studies (8 out of 46 studies) they were partially automated (P3, P11, P19, P21, P26, P38, P39, P45).

Tool Availability. We searched primary studies for details of developed tool development details and found that an open-source tool was provided in 17 studies (P1, P2, P4, P8, P10, P13, P14, P16, P17, P22, P23, P32, P34, P35, P39, P42, and P44), a proprietary tool was mentioned in 6 studies (P7, P26, P27, P31, P33, and P45) and no tool was mentioned in 23 studies (P3, P5, P6, P9, P11, P12, P15, P18–P21, P24, P25, P28–P30, P36–P38, P40, P41, P43, and P46).

Meta Tools and Frameworks. Tools and frameworks that facilitate the development of modeling languages, modeling tools and frameworks are known as meta tools and frameworks, for example, the Eclipse Modeling Framework (EMF). Table 8 provides an overview of the modeling frameworks, meta tools, and model transformation languages found in the selected studies. Among modeling frameworks, the most frequent was the Eclipse modeling framework (EMF) in 15 out of 46 studies. The second most frequent was the MontiAnna/MontiArc framework in 4 out of 46 studies. Considering meta tools, Sirius was the most common tool found in 10 out of 46 studies. The next most common tools were MontiAnna/MontiArc and Eclipse IDE both used in 4 studies. For model transformation languages, XTend and Epsilon Generation Language (EGL) were most common, found in 5 and 4 studies, respectively. The studies absent from the table did not mention the framework, meta-tool, or model transformation language.

### 4.4. RQ3 - MDE4ML studies evaluation

#### 4.4.1. Target area

We looked at the domain examples and evaluation context in each primary study — this can be academia, industry, or both. We classify studies as academia if examples and evaluations occur in controlled environments like labs and as industry when conducted in real-world settings. A large portion of studies (89%) were from an academic context (P1–P11, P13, P14, P16–P30, P32, P34, and P36–P46) and a small portion (9%) were from an industrial context (P12, P15, P31, P33). Interestingly, one study, P35 with domain-specific visual languages for Big Data Analytics and ML had evaluations in academic and industrial contexts.

#### 4.4.2. Evaluation methods

We analyzed the evaluation methods in studies and categorized them under the following five categories: case study, experiment, survey, criteria-based assessment; no evaluation. We borrow our categorization of empirical studies from Wohlin et al. [57]. We note that for case studies, we classify them otherwise if the primary study terms their evaluation method as a case study, albeit it does not match the SE definition [57]. We also added a new category here “criteria-based assessment” where the primary studies are compared against certain well-established guidelines/criteria in the area. We found that the majority of MDE solutions (23 out of 46 studies) were evaluated on case studies (P1–P3, P9, P11–P15, P17, P19, P20, P22, P23, P27, P29, P30, P33, P35, P36, P39, P40, and P45). Out of these, a few (4 of the 23) were industrial case studies (P12, P15, P33, and P35). The second most common evaluation method in studies (17 out of a total 46 studies) was experiments (P4, P7, P10, P16, P22, P24, P25, P28, P31, P34, P35, P37, P38, P41, P42, P44, and P46). Among these, only one was an industrial experiment study (P31), and four were ‘user studies’ (P10, P22, P24, and P35). We note that the user studies also included a post-experiment interview or questionnaire (survey) with the participants; however, the experiment was the main method of evaluation. The criteria-based assessment was done in two studies (P28 and P35). For example, P35 performed an evaluation against the ‘physics of notation’ guidelines in their study. Three primary studies (P22, P28, and P35) used multiple methods to evaluate their proposed MDE solution. For example, P22 evaluated their work with an IoT case study and a user study with four volunteers. No evaluation was found in eight studies (P5, P6, P8, P18, P21, P26, P32 and P43).

#### 4.4.3. Evaluation metrics and datasets

Evaluation metrics in studies are divided into metrics for ML and metrics for MDE, as shown in Figs. 10(a) and 10(b). The category not mentioned includes studies that perform evaluation but do not mention the metrics. We found 10 studies (P14, P15, P20, P24, P29, P30, P33, P35, P36, and P40) that do not mention any ML metrics and 18 studies (P1, P3, P7, P9, P11, P13–P15, P20, P22, P23, P27, P29, P34, P36, P39, P42, and P46) that do not mention any MDE metrics. The not applicable (N/A) category contains studies that either have no evaluation or the solution cannot be evaluated through such metrics. For instance, P10 provides a DSL to model dataset descriptions and then transform the models into HTML documents. Since this study has no relevant ML evaluation, we categorized it as N/A in Fig. 10(a). There are ten studies (P5, P6, P8, P10, P18, P21, P26, P32, and P43) for which ML metrics are not applicable and eight studies (P5, P6, P8, P18, P21, P26, P32, and P43) for which MDE metrics are not applicable.

We examined the primary studies and found ML evaluation metrics related to classification, regression, time and resource, and fairness. Among all studies, classification metrics were used the most, with the frequently occurring ones being accuracy of the ML component in 16 studies’ evaluation (P1, P2, P7, P9, P11, P13, P16, P17, P22, P23, P27, P34, P37–P39, and P44) and precision of the ML component in 7 studies’ evaluation (P1, P4, P9, P13, P22, P27, and P44). Other classification metrics included F1-score, recall, and ROC AUC. For regression, accuracy, and mean absolute error were the most common metrics. For time and resource, execution time and CPU time were the most frequently used metrics. For fairness, we found metrics like equal opportunity, equalized odds, and disparate impact.
Table 8
Modeling frameworks, meta tools and model transformation languages.

<table>
<thead>
<tr>
<th>Modeling framework</th>
<th>Studies</th>
<th>Meta tool</th>
<th>Studies</th>
<th>Model transformation language</th>
<th>Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eclipse modeling framework (EMF)</td>
<td>P1, P2, P4, P6, P7, P9, P14, P22, P27, P28, P30, P32, P33, P34, P37</td>
<td>Sirius</td>
<td>P1, P7, P9, P14, P25, P27, P30, P34, P37, P38</td>
<td>XTend</td>
<td>P9, P14, P22, P28, P39</td>
</tr>
<tr>
<td>xText</td>
<td>P9, P22, P27, P28, P39</td>
<td>Eclipse IDE</td>
<td>P22, P28, P29, P39</td>
<td>Epsilon Generation Language (EGL)</td>
<td>P2, P6, P7, P17</td>
</tr>
<tr>
<td>MontiAnna/MontiArc framework</td>
<td>P5, P8, P11, P12</td>
<td>MontiArc/MontiAnna</td>
<td>P5, P8, P11, P12</td>
<td>MontiAnna/MontiArc generators</td>
<td>P5, P8, P11, P12</td>
</tr>
<tr>
<td>PyEcore</td>
<td>P25, P38</td>
<td>Papyrus</td>
<td>P17, P33</td>
<td>Acceleo</td>
<td>P1, P27, P34</td>
</tr>
<tr>
<td>Meta object facility (MOF) framework</td>
<td>P17</td>
<td>TouchCore</td>
<td>P4</td>
<td>TouchCore</td>
<td>P4</td>
</tr>
<tr>
<td>GreyCat (extension of EMF)</td>
<td>P23</td>
<td>IntelliJ IDE</td>
<td>P23</td>
<td>Apache Velocity</td>
<td>P23</td>
</tr>
<tr>
<td>Langium</td>
<td>P10</td>
<td>Langium Workbench</td>
<td>P10</td>
<td>Langium</td>
<td>P10</td>
</tr>
<tr>
<td>SyncMeta</td>
<td>P24</td>
<td>SyncMeta</td>
<td>P24</td>
<td>ANTLR</td>
<td>P28</td>
</tr>
<tr>
<td>JastAdd</td>
<td>P20</td>
<td>Pyro</td>
<td>P32</td>
<td>JastAdd</td>
<td>P20</td>
</tr>
<tr>
<td>i* framework</td>
<td>P18</td>
<td>MetaEdit+</td>
<td>P35</td>
<td>Xpand language</td>
<td>P30</td>
</tr>
<tr>
<td>CINCO framework</td>
<td>P32</td>
<td>CINCO Workbench</td>
<td>P32</td>
<td>ENLIL</td>
<td>P3</td>
</tr>
<tr>
<td>OPC UA framework</td>
<td>P45</td>
<td>OPC UA modeler</td>
<td>P45</td>
<td>OPC UA code generator</td>
<td>P45</td>
</tr>
<tr>
<td>KM3 framework</td>
<td>P29</td>
<td>i* framework</td>
<td>P18</td>
<td>MetaEdit+</td>
<td>P35</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JastAdd</td>
<td>P20</td>
<td>Pyro</td>
<td>P32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CINCO Workbench</td>
<td>P32</td>
<td>CINCO Workbench</td>
<td>P32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPC UA modeler</td>
<td>P45</td>
<td>OPC UA modeler</td>
<td>P45</td>
</tr>
<tr>
<td></td>
<td></td>
<td>i* framework</td>
<td>P18</td>
<td>i* framework</td>
<td>P18</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JastAdd</td>
<td>P20</td>
<td>JastAdd</td>
<td>P20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CINCO Workbench</td>
<td>P32</td>
<td>CINCO Workbench</td>
<td>P32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OPC UA modeler</td>
<td>P45</td>
<td>OPC UA modeler</td>
<td>P45</td>
</tr>
</tbody>
</table>

Fig. 10. Evaluation metrics.

(a) ML Evaluation Metrics in Studies
A) Classification B) Regression C) Time and Resource D) Fairness E) No metrics

(b) MDE Evaluation Metrics
A) Quality B) Time and Resource C) Code metrics D) No metrics

evaluation metrics include recall in six studies (P1, P3, P4, P9, P22, and P27), f measure in five studies (P13, P22, P27, P34, and P38) and area under ROC curve in one study (P16). The second highest evaluation metrics were time and resource metrics, with execution time in eight studies (P13, P19, P23, P25, P41, P42, P44, and P46) and training time in six studies (P9, P12, P22, P37, P38, and P44). Other time and resource evaluation metrics such as resource usage were found in four studies (P19, P25, P42, and P44), execution latency in one study (P13) and inference time in one study (P44). In comparison, regression and fairness metrics were found in much fewer studies’ evaluations. Among evaluation metrics for regression tasks, loss was found in three studies (P1, P13, and P39), root mean square error in two studies (P31 and P45), mean absolute error in two studies (P12 and P31), mean relative error in one study (P37), prediction error in one study (P31), relative absolute error in one study (P45), and R² score in one study (P13). Among evaluation metrics for fairness tasks, mean difference and average odds difference were both found in one study (P2).

MDE metrics found in the primary studies were related to quality, time and resource, and code. Quality metrics were often used to evaluate MDE approaches, especially productivity increase when developing an
ML solution found in six studies (P25, P30, P31, P37, P38, and P41), and usability of the MDE approach found in three studies (P10, P24, and P35). Other quality-related evaluation metrics such as scalability were found in two studies (P19 and P44), learnability in one study (P35), desirability in one study (P35), completeness in one study (P33), effectiveness in one study (P30), correctness in one study (P4), expressiveness in one study (P10), usefulness in one study (P10), reduced complexity in one study (P10), generated code quality in one study (P17), and flexibility in one study (P19). The second highest was time and resource metrics, with generation time and execution time used the most in five studies (P2, P25, P37, P38, and P45) and two studies (P25 and P45), respectively. Other time and resource-related evaluation metrics found in the primary studies were execution time in one study (P2) and re-training time reduction in one study (P12). Code-related evaluation metrics were found in relatively fewer studies. Among these metrics, we found lines of code in four studies (P2, P28, P30, and P40), number of words in one study (P28), number of characters in one study (P28), and number of generated pipelines in one study (P16).

During the analysis of the evaluations described in the primary studies, 33 datasets were identified. The datasets most frequently used in the evaluations were the MNIST handwritten digits dataset (7 out of 46 studies) and the Iris flowers dataset (3 out of 46 studies) – we note that these datasets are widely used for ML classification.

4.5. RQ4 - Limitations and future work of existing MDE4ML studies

4.5.1. Limitations in the primary studies

In our analysis of the selected primary studies we found several limitations and we classified these into three high-level categories: approach, evaluation, and solution quality. Among the selected 46 studies, the following 19 studies, P3, P5, P7, P8, P11, P16, P18, P26, P27, P29, P31, P33, P37, P40-P44, and P46, did not mention any limitations.

Limitations in Approach. A key limitation found in the approach taken by several of the selected primary studies is the manual effort required to configure the generated artifacts (studies P19, P20, P22, P24, P36, and P39). Whereas considerable manual effort goes into modeling for P15 and code generator implementation for P21. Limited ML models are supported in studies P7, P22, P30, P33, P34, and P37. This may restrict the applicability of these approaches to a broader range of ML techniques. P4 has the limitation that a single error in the model can defeat the purpose of the entire solution. The approach in P21 is not generic and is only relevant to one ML framework (Infer.NET).

Limitations in Evaluation. We discovered evaluation-related limitations in several primary studies. However, only a few of them were actually discussed in the studies. P17, P28, P30 mention the absence of a user study to evaluate the approach with real-world users. This raises concerns about the practical usability of the approach, as user feedback is never obtained. Another limitation found in two studies (P39 and P40) was the absence of an industrial evaluation. This signifies an important gap regarding the application and usefulness of MDE solutions for ML-based systems in the industry. Interestingly, among the studies that do provide an evaluation, ten (10) studies (P9, P12, P14, P19, P20, P23, P24, P38, P42 and P45) describe their evaluations as being limited to simple scenarios or single case studies. Furthermore, P18 highlights the lack of an evaluation of the proposed solution.

Limitations in Quality. The quality of a proposed solution is of high importance; while analyzing the studies we found quality limitations related to scalability and accessibility. P14 and P39 propose approaches that are difficult to scale, making them less suitable for large-scale applications. The study P35 describes accessibility issues during evaluation due to the MetaEdit+ meta-tool. These issues include the lack of a web interface and the need for a tool license.

4.5.2. Future work suggested in the primary studies

While analyzing our selected primary studies, we were interested in examining the key future work and challenges described. We have classified these future works into three high-level categories related to enhancements in the approach, solution quality, and evaluation. P5, P11, P13, P26, P29, P41, and P44 do not mention any future work.

Improvement or Extension of Approach. Many kinds of improvements to the approach or extensions to add new features have been grouped under this category. The addition of new features and improvements like a recommender and new modeling concepts were proposed in many studies, such as P6, P9, P10, P12, P14–P16, P18–P20, P22, P24, P25, P28, P34–P36, P39, and P42. For example, the authors of P9 intend to add support for future collaborative training of ML models. Studies P1, P21, P27, P32, and P35 plan to support additional platforms like embedded systems and websites to make their solutions compatible with a wider range of platforms. Similarly, in P7, P8, P22, and P36 it is suggested that the code generators will support more programming languages. Currently, limited ML models are supported in P7, P22, P23, P30, P33, P34, P37, and P45; future work involves supporting a wider range of ML models to enhance system capabilities. In P3, P6, P15, and P36, the future goal involves considering more complex scenarios in the approach. Some studies like P9, P13, P14, P17, P22, and P34 aim to add training data processing and preparation as a part of their solutions. A future goal described in P16 is the development of a DSL, whereas the authors of P14 and P27 suggest the creation of a textual DSL to support the existing graphical one. P15 states tool implementation of their approach as a future work, although we note that 50% of studies do not specify if they have developed a tool for their approach.

Further Evaluation. As discussed in the primary study limitations section, the evaluations performed in many of the selected primary studies have limitations. To address this gap, thirteen studies P9, P12, P14, P17, P19, P20, P23, P24, P38–P40, P42, and P45, intend to perform additional evaluations. For example, in P23 multiple use cases for ML models were planned to be evaluated on high-power computers and P40 considered industrial evaluation an important next step. Surprisingly, from the eight studies that do not provide any sort of evaluation, only P18 states evaluation as a future task. Similarly, out of the 42 studies that do not conduct a user study, only three studies, P17, P28, and P30, report it as work to be done in the future.

Quality Enhancement. A small portion of the primary studies mention quality enhancements as a future target. The two more frequent quality improvements include integration with other languages and tools (P1, P8, P24, P35, P43) and interoperability support (P24, P32, P45). For instance, in P1 a future goal is to integrate with EAST-ADL an architectural language for automotive embedded systems. Other enhancements mentioned are optimization of generated code, in P2 and P33, optimal resource allocation in P37, support for model checking in P32, and improved scalability, reusability, and adaptability in P40, P20, and P25 respectively.
To ensure the quality of studies considered in our SLR, we only included searches, we had clearly defined inclusion and exclusion criteria. To reduce bias during data analysis and synthesis all authors had the first author proceeded with data extraction of the remaining studies. We compared the data extracted by the first author and validated by the other authors. The first round of filtering was based on the title and abstract. The second round was based on a brief reading of the paper, and the third round on a detailed reading. These measures ensure minimal selection bias in our study. After selecting the final pool of studies, a data extraction form was created, and all the authors participated in pilot tests for extracting data from these papers.

5.2. Construct validity

We attempted to reduce the threat to construct validity by searching seven relevant scientific databases and employing two search strategies (automated and manual). The selected primary studies were highly relevant to MDE for ML and our RQs. After several rounds of discussions, we refined our inclusion and exclusion criteria to ensure that our criteria support selecting the most suitable studies for this SLR. Some of the chosen studies use inconsistent terminology for ML, which is a potential threat to our study. However, all ambiguities were discussed with the second and the third authors to reach a consensus.

5.3. Conclusion validity

We aimed to minimize threats to conclusion validity through a well-planned and validated search and data extraction process. A data extraction form was created with questions based on our RQs, ensuring the selected data was relevant to the study. The first author extracted data using a data extraction form for a small subset of studies. All other authors followed the same method and extracted data for the same subset of studies. We compared the data extracted by the first author and other authors and found a close match between them, after which the first author proceeded with data extraction of the remaining studies. To reduce bias during data analysis and synthesis all authors had several rounds of discussion on how to best categorize and represent data.

5.4. External validity

To mitigate threats to external validity, we employed a systematic search process combining automatic search and manual search (snowballing) from the widely accepted guidelines in [21,55]. For both searches, we had clearly defined inclusion and exclusion criteria. To ensure the quality of studies considered in our SLR, we only included peer-reviewed academic studies, excluding grey literature, book chapters, opinion-, vision-, and comparison papers. We only included studies in the English language since it is the most widely used language for reporting research studies. While we acknowledge that this may have led to the exclusion of some potentially relevant studies, we deem the impact of this bias on our research is minimal. We did not exclude any study based on publication quality to eliminate publication bias in our study. Additionally, our search was not restricted to any time frame to capture all developments in the area of MDE for ML.

6. Discussion and research roadmap

6.1. MDE solutions for ML

6.1.1. Data for ML

ML is a data-driven technique [58], but most studies focus on other aspects instead. We found a surprisingly small number of studies related to data generation, pre-processing, storage, and visualization. Studies often assume that data has been pre-processed and is ready to use, whereas in reality, cleaning, wrangling, and transforming raw data is a tedious and time-consuming process [59]. Hence, the MDE techniques should consider the cases where real data is unavailable or insufficient and make data processing a part of the MDE solution. We also identified six studies (P3, P19, P29, P31, P41, and P43) that do not consider the training data or training process in their MDE solutions for ML.

We recommend researchers consider data as a first-class citizen when devising MDE solutions for ML-based systems. MDE can be particularly useful for generating data or simulators from models [42], creating meaningful visualizations [60], modeling data pre-processing workflows, and generating code to prepare data for training [18].

6.1.2. Solution focus

Development Aspects. One of our major findings from this study was the high volume of MDE solutions for the design, development, and training of ML components. This narrow solution focus leads to the issue of many other important development aspects for ML being neglected, such as requirements engineering and integration. Requirements engineering for ML and integration of ML components with traditional software components is particularly challenging [6,40] due to the inherent under-specification of ML [61] and the unique differences between ML components and traditional software components [40]. However, we only found five papers focusing on requirements engineering and four on integration. Furthermore, we found a lack of MDE solutions for ML pipelines [45], automated deployment, and monitoring of ML components, also known as MLOps. Despite the importance of runtime monitoring for early detection of unwanted behavior [62], we only found two papers relevant to this. Another key area that has not received enough attention is documentation [63]; documentation of datasets, ML models, training parameters, deployment configurations, and ML pipelines is extremely important for maintenance, reusability, and scalability.

We recommend that researchers broaden the focus of their studies and leverage MDE capabilities to address the challenges in requirements engineering, integration, deployment, monitoring, and documentation of ML components. MDE can be beneficial in several ways, such as DSLs to support requirements engineering for ML, automated generation of runtime monitors from models, and models to automate and ease integration.

Machine Learning Type. We identified a large fraction of studies focused on supervised and deep learning, while unsupervised and reinforcement learning were not as widely covered in the literature.
Unsupervised learning is a powerful technique to identify hidden patterns in large unlabeled datasets. To this end, we did not find any paper providing an MDE solution focusing solely on unsupervised learning. Reinforcement learning has strengths in learning from experience, sequential decision-making, and handling complex state spaces, which are highly useful in robotics, personalized learning, and gaming. Only four studies proposed MDE solutions for reinforcement learning.

We recommend researchers should explore developing MDE solutions that cater to unsupervised and reinforcement learning. Given the wealth of unlabeled data in various domains, MDE tools for unsupervised learning could pave the way for more efficient data analysis and knowledge extraction. Similarly, reinforcement learning, with its expansive applications, presents a significant potential for applying MDE.

**Model-driven Engineering Details.** We found a lack of MDE details in studies published in ML venues. These studies were focused on textual DSLs for specifying ML operations and transforming them into high-performance code for various hardware platforms like CPUs and GPUs. For instance, P41 provides a textual DSL for ML, and the model is automatically converted into optimized CUDA code for GPUs. Other similar studies include P40, P42, P43, P44, and P46. Although there are significant mathematical details about ML in these studies, the explanation of MDE aspects, such as the meta-models and transformations, is often overlooked. Additionally, we also found some studies in the MDE domain that lacked MDE details, such as studies P15, P18, P19, P24, P26, P29, P32, and P38.

We recommend authors, when specifying MDE approaches, add some level of detail with regard to the MDE steps taken in their solution. This would allow researchers and practitioners from the SE and MDE domains to better understand and apply these solutions.

### 6.1.3. Solution maturity

Most reported MDE solutions for ML in our selected primary studies are still in their early stages, e.g., based on simple cases, and do not support an end-to-end ML lifecycle. This is not surprising since active research in the area dates only to a few years back (e.g., in our 46 primary studies, the earliest paper is from 2008, but 39/46 papers are from 2018 and later). Upon examination of the literature, we found that tools are available for 23 studies. Out of these, only 17 studies provided open-source tools. Some notable ones are P2, P10, P22, and P23. Moreover, existing solutions often overlook complex scenarios and focus only on simple models, for example, P3, P6, P15, and P26. Our analysis also revealed that manual configurations are required for the artifacts generated by ten studies (P3, P14, P19, P20, P22, P24, P35, P36, P39, and P45). The lack of available tool support, automation, and consideration of complex scenarios hinders the adoption, extension and reproducibility of MDE solutions for ML.

We suggest researchers and practitioners further develop their solutions to consider the entire ML lifecycle and develop research prototypes that cater to end users. In the interest of lowering costs, fostering collaboration, and creating more opportunities for innovation, we recommend researchers to open-source their solutions, data, and tools. Open-source software hosting platforms like GitHub, Zenodo, SourceForge, and Gitlab can be leveraged. Accessible and mature solutions are also more likely to be adopted in the industry.

### 6.1.4. MDE solutions for domain experts

While there were a substantial number of studies providing MDE solutions for ML engineers and software engineers, we found a lack of solutions for domain experts. With the rise of new trends like no code and low code [64], the complexity of developing ML-based systems can be significantly reduced for domain experts. No code and low code approaches allow users (mostly domain experts) to develop and deploy applications without writing much code [64]. Additionally, ML experts, engineers, and analysts can also benefit from such low-code platforms.

We recommend researchers and practitioners develop low-code platforms for systems with ML components so that non-ML experts can benefit from the capabilities of ML. Low code platforms can significantly reduce development complexity and time to deployment.

### 6.1.5. Machine learning algorithms and terminology

We came across varied ML terminology and granularity levels used in studies to describe ML. For instance, P9 and P32 mention at a very high level that their MDE solution is for ML, but studies like P8 and P30 are more specific and specify the exact nature of ML techniques, e.g., neural networks and reinforcement learning. The inconsistent terminologies made it challenging to analyze studies and draw conclusions. This issue was further exacerbated when studies did not explicitly mention the supported ML algorithms; these include P12, P24, P25, P27, P35, P38, and P40.

We encourage the MDE community to build a consensus on the terminology used for devising ML-based solutions to facilitate understanding and comparisons with other studies. We further highlight the need for clearly specifying the details of the ML algorithms supported in the study.

### 6.1.6. Solution scalability

Scalability in MDE is a well-known challenge [15]. Little to no focus is placed on scalability in the selected studies on MDE for ML. Almost 75% of the studies (P1–P8, P10–P12, P15, P17, P18, P20–P22, P24–P29, P31, P32, P34–P36, P38, P40, P42, P43, P45, and P46) do not discuss the scalability of their solution. One possible reason behind this could be that often, the primary goal is to develop a proof of concept, leading to the development of basic MDE tools that only work for simple projects. Scalability is often an afterthought in such solutions, rendering them of little or no practical use.

We recommend researchers consider the importance of scalability in MDE solutions and report their scalability results to facilitate practical adoption and comparison.

### 6.1.7. Responsible ML

Responsible use of ML refers to applying ML to maximize benefits for the end users and society while minimizing harm. This consists of developing and managing ML components prioritizing human-centric needs such as fairness, trust, safety, explainability, privacy, and human values [65]. A survey of large-scale ML systems in industrial settings reported a lack of solutions for human-centric challenges such as privacy and safety [66]. Despite the growing awareness of responsible ML, only nine of the selected primary studies (P2, P3, P7, P10, P27, P31, P33, P35, and P45) out of the total 46 considered human-centric aspects and/or the responsible use of ML in their MDE solutions. This is concerning since neglecting these critical human-centric requirements can have serious consequences, such as ML-based systems that are not trustworthy, biased, and violate ethical principles and legal policies [65].

We suggest researchers prioritize the responsible applications of ML in their MDE solutions. As a result, the ML artifacts generated will better meet the needs of end-users and foster trust [65]. Some potential MDE approaches to achieve this include DSLs tailored for modeling human-centric requirements in ML components. These automated code generators of ML components conform to responsible ML practices and runtime monitoring for responsible ML.

### 6.2. Evaluations of MDE solutions for ML

#### 6.2.1. Real-world evaluation

Real-world evaluations in an industrial context demonstrate the usefulness of MDE solutions in the industry. Such evaluations also impact the likelihood of research being applied to the industry. Yet, only five of the selected studies (P12, P15, P31, P33, P35) perform an industrial evaluation. User studies (or surveys) are another useful...
method for evaluating the practical usability of the approach with actual end users. However, we found only four studies (P10, P22, P24, P35) with user study as an evaluation method.

We encourage researchers to evaluate their proposed MDE solutions on industrial case studies and with real-world users to get realistic results and feedback. More value can be added to user studies by evaluating with diverse groups of users representing a wide range of demographics and perspectives. We realize this may not always be possible, but it remains an ideal goal to strive in order to achieve a more comprehensive and inclusive evaluation.

6.2.2. Evaluation rigor

One of the recurring issues we observed in studies was the lack of rigor in evaluation. For instance, eight studies (P5, P6, P8, P18, P21, P26, P32, and P43) did not report on any evaluation, 22 studies performed a partial evaluation (i.e., focused on either MDE or ML but not both), 14 studies (P7, P14, P15, P20, P29, P36–P42, P45, P46) had insufficient evaluation details, no clear rationale behind the evaluation settings (e.g., the choice of evaluation metrics and parameters selection) or provided only limited discussion on the implication of results. In studies P14, P15, P20, P24, P29, P30, P33, P35, and P40, there is no evaluation of the ML aspect. A greater number of studies P1, P3, P7, P9, P11, P13, P22, P23, P27, P34, P39, P42, and P46, provide no evaluation of the MDE aspect. These limitations impact the quality of the resulting evaluation and the ability to draw meaningful and reliable conclusions from the results.

We propose researchers perform and report empirical evaluations with respect to both aspects of MDE and ML. Clearly justify their choices of evaluation methods and metrics and provide comprehensive details of the result, based on the guidelines for reporting empirical software engineering research (e.g., [67, 68]). Rigorous evaluation methods increase the credibility of research findings, making them more trustworthy for academia and industrial applications.

7. Conclusion

Software engineering for ML-based systems is in many ways more complicated and challenging for developers compared to traditional software systems [40]. In this article, we report on an SLR of MDE solutions for software systems with ML components. The goal was to explore the potential of MDE for ML-based systems and identify trends, benefits, and limitations of existing work. For the SLR, we followed the systematic process described by Kitchenham et al. [21] and selected 46 highly relevant primary studies from an initial pool of 3,496 papers. We have explored many interesting aspects of the MDE solutions and reported our findings along with gaps and potential directions for future research. Our key findings suggest that over the last five years, there has been a significant increase in the studies on MDE4ML; with the rapidly growing popularity of ML and AI, we expect this trend to continue in the future.

Our examination of selected studies suggests that: (1) there are few studies on the data required for ML; (2) proposed solutions are limited to the design, development, and training of ML components (less studies on requirements engineering, integration, ML pipelines, automated deployment, monitoring and documentation); (3) there are limited studies for unsupervised learning and reinforcement learning; (4) MDE steps are not comprehensively explained; (5) MDE solutions for ML-based systems require more maturity and better tool support; (6) there is a lack of MDE4ML solutions for domain experts; (7) studies use inconsistent terminology for describing ML; (8) there is a need for more focus on solution scalability; (9) there is a lack of responsible ML and human-centric development practices in MDE approaches for ML-based systems; (10) most evaluations lack rigor and are conducted in an academic setting.

CRediT authorship contribution statement

H. Naveed: Writing – original draft, Visualization, Software, Methodology, Investigation, Data curation, Conceptualization. Chetan Arora: Writing – review & editing, Supervision. Houriieh Khalajzadeh: Writing – review & editing, Supervision. John Grundy: Writing – review & editing, Supervision, Funding acquisition. Omar Haggag: Writing – review & editing, Supervision.

Data availability

SLR data is available at the following link MDE4MLSLRData(OriginalData).

Acknowledgments

Naveed is supported by a Faculty of IT PhD scholarship. Grundy is supported by ARC Laureate Fellowship FL19010035. This work has been partially supported by ARC Discovery Project DP200100020.

Appendix A. Selected primary studies


Table 9

<table>
<thead>
<tr>
<th>ID</th>
<th>QA1</th>
<th>QA2</th>
<th>QA3</th>
<th>QA4</th>
<th>QA5</th>
<th>P19</th>
<th>QA1</th>
<th>QA2</th>
<th>QA3</th>
<th>QA4</th>
<th>QA5</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>P24</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P2</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>P25</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>P3</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>P26</td>
<td>2</td>
<td>3</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>P4</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>P27</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>P5</td>
<td>5</td>
<td>5</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>P28</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>P6</td>
<td>5</td>
<td>3</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>P29</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>P7</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>P30</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>P8</td>
<td>5</td>
<td>5</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>P31</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>P9</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>P32</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>P10</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
<td>P33</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>P11</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>P34</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>P12</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>P35</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>P13</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>P36</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>P14</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>P37</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>P15</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>P38</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>P16</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>P39</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>P17</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>P40</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>P18</td>
<td>4</td>
<td>3</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>P41</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>P19</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>P42</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>P20</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>P43</td>
<td>2</td>
<td>2</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>P21</td>
<td>3</td>
<td>3</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>P44</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>P22</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>P45</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>P23</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>P46</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>


P43: Weimer, Markus, Tyson Condie, and Raghu Ramakrishnan. “Machine learning in ScalOps, a higher order cloud computing language”. In NIPS 2011 Workshop on parallel and large-scale machine learning (BigLearn), vol. 9, pp. 389–396. 2011


Appendix B. Quality assessment of primary studies

See Table 9.

References


