Challenges in Deploying Machine Learning: A Survey of Case Studies

ANDREI PALEYES, University of Cambridge, United Kingdom
RAOUL-GABRIEL URMA, Cambridge Spark, United Kingdom
NEIL D. LAWRENCE, University of Cambridge, United Kingdom

In recent years, machine learning has transitioned from a field of academic research interest to a field capable of solving real-world business problems. However, the deployment of machine learning models in production systems can present a number of issues and concerns. This survey reviews published reports of deploying machine learning solutions in a variety of use cases, industries, and applications and extracts practical considerations corresponding to stages of the machine learning deployment workflow. By mapping found challenges to the steps of the machine learning deployment workflow, we show that practitioners face issues at each stage of the deployment process. The goal of this article is to lay out a research agenda to explore approaches addressing these challenges.

CCS Concepts: • **Computing methodologies** → **Machine learning**

Additional Key Words and Phrases: Machine learning applications, sofware deployment

ACM Reference format:
Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. 2022. Challenges in Deploying Machine Learning: A Survey of Case Studies. *ACM Comput. Surv.* 55, 6, Article 114 (December 2022), 29 pages.
https://doi.org/10.1145/3533378

1 INTRODUCTION

Machine learning (ML) has evolved from being an area of academic research to an applied field. According to a recent global survey conducted by McKinsey, machine learning is increasingly adopted in standard business processes with nearly 25% year-over-year growth [1] and with growing interest from the general public, business leaders [2], and governments [3].

This shift comes with challenges. Just as with any other field, there are significant differences between what works in an academic setting and what is required by a real world system. Certain bottlenecks and invalidated preconceptions should always be expected in the course of that process. As more solutions are developed and deployed, practitioners report their experience in various forms, including publications and blog posts. Motivated by such reports and our personal

A. P. and N. L. are grateful for funding from a Senior AI Fellowship from the Alan Turing Institute (ATI) and UK Research & Innovation (UKRI).
Authors' addresses: A. Paleyes and N. D. Lawrence, University of Cambridge, Department of Computer Science and Technology, Cambridge, United Kingdom; emails: {ap2169, ndl21}@cam.ac.uk; R.-G. Urma, Cambridge Spark, Cambridge, United Kingdom; email: raoul@cambridgespark.com.

0360-0300/2022/12-ART114
https://doi.org/10.1145/3533378

experiences, in this study, we aim to survey the challenges in deploying machine learning in production[1] with the objective of understanding what parts of the deployment process cause the most difficulties. First, we provide an overview of the machine learning deployment workflow. Second, we review case studies to extract problems and concerns practitioners have at each particular deployment stage. Third, we discuss cross-cutting aspects that affect every stage of the deployment workflow: ethical considerations, law, end-users' trust, and security. Finally, we conclude with a brief discussion of potential solutions to these issues and further work.

Ours is not the first survey of machine learning in production. A decade ago, such surveys were already conducted, albeit with a different purpose in mind. Machine learning was mainly a research discipline, and it was uncommon to see ML solutions deployed for a business problem outside of "Big Tech" companies in the information technology industry. So, the purpose of such a survey was to show that ML can be used to solve a variety of problems and illustrate it with examples, as was done by Pěchouček and Mařík [4]. Nowadays the focus has changed: Machine learning is commonly adopted in many industries, and the question becomes not "Where is it used?," but rather "How difficult is it to use?"

One approach to assess the current state of machine learning deployment for businesses is a survey conducted among professionals. Such surveys are primarily undertaken by private companies and encompass a variety of topics. Algorithmia's report [5, 6] goes deep into the deployment timeline, with the majority of companies reporting between 8 and 90 days to deploy a single model, and 18% taking even more time. A report by IDC [7] surveyed 2,473 organizations and found that a significant portion of their attempted AI deployments fail, quoting lack of expertise, bias in data, and high costs as primary reasons. O'Reilly has conducted an interview study that focused on ML practitioners' work experience and tools they use [8]. Broader interview-based reports have also been produced by dotscience [9] and dimensional research [10].

While there are a number of business reports on the topic, the challenges of the entire machine learning deployment pipeline are not covered nearly as widely in the academic literature. There is a burgeoning field of publications focusing on specific aspects of deployed ML, such as Bhatt et al. [11], which focuses on explainable ML, or Amershi et al. [12], which discusses software engineering aspects of deploying ML. A large number of industry-specific surveys also exist, and we review some of these works in the appropriate sections below. However, the only general purpose survey we found is Baier et al. [13], which combines a literature review and interviews with industry partners. In contrast with that paper, which concentrates on the experience of the information technology sector, we aim at covering case studies from a wide variety of industries, thus increasing the breadth of our survey. We also discuss the most commonly reported challenges in the literature in much greater detail.

In our survey, we consider three main types of papers:

- Case study papers that report experience from a single ML deployment project. Such works usually go deep into discussing each challenge the authors faced and how it was overcome.
- Review papers that describe applications of ML in a particular field or industry. These reviews normally give a summary of challenges that are most commonly encountered during the deployment of the ML solutions in the reviewed field.
- "Lessons learned" papers where authors reflect on their past experiences of deploying ML in production.

To ensure that this survey focuses on the current challenges, only papers published in the past six years are considered, with only a few exceptions to this rule. We also refer to other types of

[1]By *production*, we understand the setting where a product or a service is made available for use by its intended audience.

papers where it is appropriate, e.g., practical guidance reports, interview studies, and regulations. We have not conducted any interviews ourselves as part of this study. Further work and extensions are discussed at the end of this article.

The main contribution of our article is to show that practitioners face challenges at each stage of the machine learning deployment workflow. This survey supports our goal to raise awareness in the academic community of the variety of problems that practitioners face when deploying machine learning and start a discussion on what can be done to address these problems.

2 MACHINE LEARNING DEPLOYMENT WORKFLOW

For the purposes of this work, we are using the ML deployment workflow definition suggested by Ashmore et al. [14], however, it is possible to conduct a similar review with any other ML pipeline description, such as CRISP-DM [15] or TDSP [16]. In this section, we give a brief overview of the definition we are using.

According to Ashmore et al. [14], the process of developing an ML-based solution in an industrial setting consists of four stages[2]:

- **Data management**, which focuses on preparing data that is needed to build a machine learning model;
- **Model learning**, where model selection and training happens;
- **Model verification**, the main goal of which is to ensure the model adheres to certain functional and performance requirements;
- **Model deployment**, which is about integration of the trained model into the software infrastructure that is necessary to run it. This stage also covers questions around model maintenance and updates.

Each of these stages is broken down further into smaller steps. It is important to highlight that the apparent sequence of this description is not necessarily the norm in a real-life scenario. It is perfectly normal for these stages to run in parallel to a certain degree and inform each other via feedback loops. Therefore, this or any similar breakdown should be considered not as a timeline, but rather as a useful abstraction to simplify references to concrete parts of the deployment pipeline. We have chosen this representation of the workflow, because comparing to alternatives it defines a larger number of deployment steps, which allows for finer classification of the challenges.

For the remainder of this article, we discuss common issues practitioners face at each step. We also discuss cross-cutting aspects that can affect every stage of the deployment pipeline. Where appropriate, the concrete class of ML problems is specified (such as supervised learning or reinforcement learning), although the majority of the challenges can be encountered during the deployment of many types of learning tasks. Table 1 provides a summary of the issues and concerns we discuss. By providing illustrative examples for each step of the workflow, we show how troublesome the whole deployment experience can be. Note that the order in which challenges appear in that table and in the article does not necessarily reflect their severity. The impact that a particular issue can have on a deployment project depends on a large variety of factors, such as business area, availability of resources, team experience, and more. For that reason, we do not attempt to prioritize the challenges discussed in the survey.

[2]There is a requirements formulation stage that Ashmore et al. do not consider a part of the ML workflow. Similarly, we see this as a challenge across many domains and thus out of the scope of this study. We refer interested readers to the work by Takeuchi and Yamamoto [17].

Table 1. All Considerations, Issues, and Concerns Explored in This Study

Deployment Stage	Deployment Step	Considerations, Issues, and Concerns
Data management	Data collection	Data discovery
	Data preprocessing	Data dispersion Data cleaning
	Data augmentation	Labeling of large volumes of data Access to experts Lack of high-variance data
	Data analysis	Data profiling
Model learning	Model selection	Model complexity Resource-constrained environments Interpretability of the model
	Training	Computational cost Environmental impact Privacy-aware training
	Hyper-parameter selection	Resource-heavy techniques Unknown search space Hardware-aware optimization
Model verification	Requirement encoding	Performance metrics Business-driven metrics
	Formal verification	Regulatory frameworks
	Test-based verification	Simulation-based testing Data validation routines Edge case testing
Model deployment	Integration	Operational support Reuse of code and models Software engineering anti-patterns Mixed team dynamics
	Monitoring	Feedback loops Outlier detection Custom design tooling
	Updating	Concept drift Continuous delivery
Cross-cutting aspects	Ethics	Aggravation of biases Fairness and accountability Authorship Decision-making
	Law	Country-level regulations Abiding by existing legislation Focus on technical solution only
	End-users' trust	Involvement of end-users User experience Explainability score
	Security	Data poisoning Model stealing Model inversion

Each is assigned to the stage and step of the deployment workflow where it is commonly encountered.

3 DATA MANAGEMENT

Data is an integral part of any machine learning solution. The overall effectiveness of the solution depends on the training and test data as much as on the algorithm. The process of creating quality datasets is usually the very first stage in any production ML pipeline. Unsurprisingly, practitioners face a range of issues while working with data, as reported by Polyzotis et al. [18]. Consequently, this stage consumes time and energy that is often not anticipated beforehand. In this section, we describe issues concerning four steps within data management: data collection, data preprocessing, data augmentation, and data analysis. Note that we consider storage infrastructure challenges, such as setting up databases and query engines, beyond the scope of this survey. We refer readers to the survey by Cai et al. [19] for further discussion of big data storage.

3.1 Data Collection

Data collection involves activities that aim to discover and understand what data is available, as well as how to organize convenient storage for it. The task of discovering what data exists and where it is can be a challenge by itself, especially in large production environments typically found within organizations [20]. Finding data sources and understanding their structure is a major task, which may prevent data scientists from even getting started on the actual application development. As explained by Lin and Ryaboy [21], at Twitter this situation often happened as a result of the same entity (e.g., a Twitter user) being processed by multiple services. Internally, Twitter consists of multiple services calling each other, and every service is responsible for a single operation. This approach, known in software engineering as the "single responsibility principle" [22], results in an architecture that is very flexible in terms of scalability and modification. However, the flip side of this approach is that at a large scale it is very hard to keep track of what data related to the entity is being stored by which service and in which form. Some data may only exist in a form of logs, which by their nature are not easily parsed or queried. An even worse case is the situation when data is not stored anywhere, and to build a dataset one needs to generate synthetic service API calls. Such dispersion of data creates major hurdles for data scientists, because without a clear idea of what data is available or can be obtained it is often impossible to understand what ML solutions can achieve.

3.2 Data Preprocessing

The preprocessing step normally involves a range of data cleaning activities: identification of a schema, imputation of missing values, reduction of data into an ordered and simplified form, and mapping from raw form into a more convenient format. Methods for carrying out data manipulations like this is an area of research that goes beyond the scope of this study. We encourage readers to refer to review papers on the topic, such as Abedjan et al. [23], Patil and Kulkarni [24], and Ridzuan et al. [25].

A lesser known but important problem that can also be considered an object of the preprocessing step is data dispersion. It often turns out that there can be multiple relevant separate data sources that may have different schemas, different conventions, and their own way of storing and accessing the data. Joining this information into a single dataset suitable for machine learning can be a complicated task in its own right, known as the data integration process [26]. An example of this is what developers of Firebird faced [27]. Firebird is an advisory system in the Atlanta Fire Department, that helps identify priority targets for fire inspections. As a first step towards developing Firebird data was collected from 12 datasets including the history of fire incidents, business licenses, households, and more. These datasets were combined to give a single dataset covering all relevant aspects of each property monitored by the Fire Department.

Authors particularly highlight data joining as a difficult problem. Given the fact that buildings can be identified through their geospatial location, each dataset contained spatial data specifying the building's address. Spatial information can be presented in different formats and sometimes contains minor differences such as different spellings. All this needed to be cleaned and corrected. These corrections can be highly time-consuming and proved to be so in the Firebird project.

3.3 Data Augmentation

There are multiple reasons why data might need to be augmented, and in practice one of the most problematic ones is the absence of labels. A label is a value that the ML model seeks to predict from the input data in a classic supervised learning setting. Real-world data is often unlabeled, thus labeling turns out to be a challenge in its own right. We discuss three possible factors for lack of labeled data: limited access to experts, absence of high-variance data, and sheer volume.

Label assignment is difficult in environments that tend to generate large volumes of data, such as network traffic analysis. To illustrate a scale of this volume, a single 1-GB/s Ethernet interface can deliver up to 1.5 million packets per second. Even with a huge downsampling rate, this is still a significant number, and each sampled packet needs to be traced to be labeled. This problem is described by Pacheco et al. [28], which surveys applications of machine learning to network traffic classification with tasks such as protocol identification or attack detection. There are two main ways of acquiring data in this domain, and both are complicated for labeling purposes:

- Uncontrolled, collecting real traffic. This approach requires complex tracking flows belonging to a specific application. Due to this complexity, very few works implement reliable ground truth assignment for real traffic.
- Controlled, emulating, or generating traffic. This approach is very sensitive to the choice of tooling and its ability to simulate the necessary traffic. Studies have shown that existing tools for label assignment can introduce errors into collected ML datasets of network traffic data, going as high as almost 100% for certain applications [29]. Moreover, these tools' performance degrades severely for encrypted traffic.

Access to experts can be another bottleneck for collecting high-quality labels. It is particularly true for areas where expertise mandated by the labeling process is significant, such as medical image analysis [30]. Normally, multiple experts are asked to label a set of images, and then these labels are aggregated to ensure quality. This is rarely feasible for big datasets due to experts' availability. A possible option here is to use noisy label oracles [31] or weak annotations [32], however, these approaches provide imprecise labels, which can lead to a loss in quality of the model [33]. Such losses are unacceptable in the healthcare industry, where even the smallest deviation can cause catastrophic results (this is known as the Final Percent challenge, according to Budd et al. [30]).

Lack of access to high-variance data (data that covers large parts of the feature space) can be among the main challenges one faces when deploying machine learning solutions from the lab environment to the real world. Dulac-Arnold et al. [34] explain that this is the case for **Reinforcement Learning (RL)**. RL is an area of ML that focuses on intelligent agents that learn by taking actions and receiving feedback from their environment. Agents make decisions on what action to take next to maximize the reward based on their previous experience. It is common practice in RL research to have access to separate environments for training and evaluation of an agent. However, in practice, all data comes from the real system, and the agent can no longer have a separate exploration policy—this is simply unsafe. Therefore, the data available becomes low-variance—very little of the state space is covered. While this approach ensures safety, it means that the agent is not trained to recognize an unsafe situation and make the right decision in it. A practical example

of this issue can be seen in the area of autonomous vehicle control [35]. Their simulations are often used for training, but complex interactions, such as friction, can be hard to model, and small variations in simulation may result in the agent not being transferable to the real world.

Budd et al. [30] show that interface design directly impacts the quality of applications built to collect annotations for unlabeled data. They discuss a range of projects that collected labels for medical images, all of which benefited from a well-designed user interface. The authors conclude that the end-user interface plays a large part in the overall success of the annotation applications.[3]

3.4 Data Analysis

Data needs to be analyzed to uncover potential biases or unexpected distribution shifts in it. Availability of high-quality tools is essential for conducting any kind of data analysis. One area that practitioners find particularly challenging in that regard is visualization for data profiling [36]. Data profiling refers to all activities associated with troubleshooting data quality, such as missing values, inconsistent data types, and verification of assumptions. Despite obvious relevance to the fields of databases and statistics, there are still too few tools that enable the efficient execution of these data mining tasks. The need for such tools becomes apparent considering that, according to the survey conducted by Microsoft [37], data scientists think data issues are the main reason to doubt the quality of the overall work.

4 MODEL LEARNING

Model learning is the stage of the deployment workflow that enjoys the most attention within the academic community. As an illustration of the scale of the field's growth, the number of submissions to NeurIPS, a primary conference on ML methods, has quadrupled in six years, going from 1,678 submissions in 2014 to 6,743 in 2019 [38]. Nevertheless, there are still plenty of practical considerations that affect the model learning stage. In this section, we discuss issues concerning three steps within model learning: model selection, training, and hyper-parameter selection.

4.1 Model Selection

In many practical cases the selection of a model is decided by one key characteristic of a model: complexity. Despite areas such as deep learning and reinforcement learning gaining in popularity with the research community, in practice simpler models are often chosen. Such models include shallow neural network architectures, simple approaches based on **Principal Component Analysis (PCA)**, decision trees, and random forests.

Simple models can be used as a way to prove the concept of the proposed ML solution and get the end-to-end setup in place. This approach reduces the time to get a deployed solution, allows the collection of important feedback, and also helps avoid overcomplicated designs. This was the case reported by Haldar et al. [39]. In the process of applying machine learning to AirBnB search, the team started with a complex deep learning model. The team was quickly overwhelmed by its complexity and ended up consuming development cycles. After several failed deployment attempts the neural network architecture was drastically simplified: a single hidden layer NN with 32 fully connected ReLU activations. Even such a simple model had value, as it allowed the building of a whole pipeline of deploying ML models in a production setting, while providing reasonably good performance.[4] Over time the model evolved, with a second hidden layer being added, but it still remained fairly simple, never reaching the initially intended level of complexity.

[3]For further discussion of the role user interface can play in adoption of an ML system, see Section 7.3.
[4]We discuss more benefits of setting up the automated deployment pipeline in Section 6.3.

Another advantage that less complex models can offer is their relatively modest hardware requirements. This becomes a key decision point in resource-constrained environments, as shown by Wagstaff et al. [40]. They worked on deploying ML models to a range of scientific instruments onboard the Europa Clipper spacecraft. Spacecraft design is always a tradeoff between the total weight, robustness, and the number of scientific tools onboard. Therefore, computational resources are scarce and their usage has to be as small as possible. These requirements naturally favor the models that are light on computational demands. The team behind Europa Clipper used machine learning for three anomaly detection tasks; some models took time series data as input and some models took images, and on all three occasions simple threshold or PCA-based techniques were implemented. They were specifically chosen because of their robust performance and low demand for computational power.

A further example of a resource-constrained environment is wireless cellular networks, where energy, memory consumption, and data transmission are very limited. Most advanced techniques, such as deep learning, are not considered yet for practical deployment, despite being able to handle high-dimensional mobile network data [41].

The ability to interpret the output of a model into understandable business domain terms often plays a critical role in model selection and can even outweigh performance considerations. For that reason, **decision trees (DT)** [42], which can be considered a fairly basic ML algorithm, are widely used in practice. DT presents its branching logic in a transparent way that resembles human decision process, which allows for interpretation and auditing. Hansson et al. [43] describe several cases in manufacturing that adopt DT because of their interpretability.

Banking is another industry where DT finds extensive use. As an illustrative example, it is used by Keramati et al. [44] where the primary goal of the ML application is predicting churn in an interpretable way through if-then rules. While it is easy to imagine more complicated models learning the eventual input-output relationship for this specific problem, interpretability is a key requirement here because of the need to identify the features of churners. The authors found DT to be the best model to fulfill this requirement.

Nevertheless, **deep learning (DL)** is commonly used for practical background tasks that require analysis of a large amount of previously acquired data. This notion is exemplified by the field of **unmanned aerial vehicles (UAV)** [45]. Data collected by image sensors is the most common UAV data modality being exploited by DL, because of the sensors' low cost, low weight, and low power consumption. DL offers excellent capabilities for processing and presentation of raw images acquired from sensors, but computational resource demands still remain the main blocker for deploying DL as an online processing instrument onboard UAVs.

4.2 Training

Model training is the process of feeding the chosen model with a collected dataset to learn certain patterns or representations of the data. One of the biggest concerns with the model training stage is the economic cost associated with carrying out the training procedure due to the computational resources required. This is often true in the field of **natural language processing (NLP)**, as illustrated by Sharir et al. [46]. The authors observe that while the cost of individual floating-point operations is decreasing, the overall cost of training NLP is only growing. They took one of the state-of-the-art models in the field, BERT [47], and found out that depending on the chosen model size full training procedure can cost anywhere between $50 K and $1.6 M in cloud computing resources, which is unaffordable for most research institutions and many companies. The authors observe that training dataset size, number of model parameters, and number of operations used by the training procedure are all contributing towards the overall cost. Of particular importance here

is the second factor: Novel NLP models are already using billions of parameters, and this number is expected to increase further in the near future [48].

A related concern is raised by Strubell et al. [49] regarding the impact the training of ML models has on the environment. By consuming more and more computational resources, ML model training is driving up energy consumption and greenhouse gas emissions. According to the estimates provided in the paper, one full training cycle utilizing neural architecture search emits an amount of CO_2 comparable to what four average cars emit in their whole lifetime. The authors stress how important it is for researchers to be aware of such impact of model training and argue that the community should give higher priority to computationally efficient hardware and algorithms. Similar concerns around environmental impact are voiced by Bender et al. [50].

As more businesses start using ML techniques on their users' data, concerns are being raised over the privacy of data and how well individuals' sensitive information is preserved over the course of the model training process [51]. Illustrating the gravity of this concern, Shokri et al. constructed an attack for membership inference, that is, to determine if a given input record was a part of the model's training dataset. They verified it on models trained on leading ML-as-a-service providers [52], achieving anywhere from 70% up to 94% accuracy of membership inference. Consequently, companies have to consider privacy-aware approaches, which most likely come at a cost of the model accuracy. Navigating the tradeoff between privacy and utility is considered an open challenge for practitioners dealing with sensitive data [53]. Some of the ways this tradeoff is resolved in practice are differential privacy that explicitly corrupts the data [54], homomorphic encryption that restricts the class of learning algorithm but allows for training on encrypted data [55], and federated learning that distributes training across personal devices to preserve privacy, but thereby constrains the algorithms that can be used for model fitting [56].

4.3 Hyper-parameter Selection

In addition to parameters that are learned during the training process, many ML models also require hyper-parameters. Examples of such hyper-parameters are the depth of a decision tree, the number of hidden layers in a neural network, or the number of neighbors in k-Nearest Neighbors classifier. **Hyper-parameter optimization (HPO)** is the process of choosing the optimal setting of these hyper-parameters. Most HPO techniques involve multiple training cycles of the ML model. This is computationally challenging, because in the worst case the size of the HPO task grows exponentially: Each new hyper-parameter adds a new dimension to the search space. As discussed by Yang and Shami [57], these considerations make HPO techniques very expensive and resource-heavy in practice, especially for applications of deep learning. Even approaches such as Hyperband [58] or Bayesian optimization [59], which are specifically designed to minimize the number of training cycles needed, are not yet able to deal with the high-dimensional searches that emerge when many hyper-parameters are involved. Large datasets complicate matters by leading to long training times for each search.

Many hyper-parameter tuning approaches require the user to define a complete search space, i.e., the set of possible values each of the hyper-parameters can take. Unfortunately, in practical use cases this is often impossible due to insufficient knowledge about the problem at hand. Setting the hyper-parameter optimization bounds remains one of the main obstacles preventing wider use of the state-of-the-art HPO techniques [60].

HPO often needs to take into account specific requirements imposed by the environment where the model will run. This is exemplified by Marculescu et al. [61] in the context of hardware-aware ML. To deploy models to embedded and mobile devices, one needs to be aware of energy and memory constraints imposed by such devices. This creates a need for customized hardware-aware

optimization techniques that efficiently optimize for the accuracy of the model and the hardware jointly.

5 MODEL VERIFICATION

Verification is considered an essential step in any software development cycle, as it ensures the quality of the product and reduces maintenance costs. As is the case with any software, ML models should generalize well to unseen inputs, demonstrate reasonable handling of edge cases and overall robustness, as well as satisfy all functional requirements. In this section, we discuss issues concerning three steps within model verification: requirement encoding, formal verification, and test-based verification.

5.1 Requirement Encoding

Defining requirements for a machine learning model is a crucial prerequisite of testing activities. It often turns out that an increase in model performance does not translate into a gain in business value, as Booking.com discovered after deploying 150 models into production [62]. One particular reason they highlight is a failure of proxy metrics (e.g., clicks) to convert to the desired business metric (e.g., conversion). Therefore, alongside accuracy measures, additional domain specific metrics need to be defined and measured. Depending on the application, these may be inspired by KPIs and other business-driven measures. In the case of Booking.com, such metrics included conversion, customer service tickets, or cancellations. A cross-disciplinary effort is needed to even define such metrics, as understanding from modeling, engineering, and business angles is required. Once defined, these metrics should also be used for monitoring the production environment and for quality control of model updates.

Besides, simply measuring the accuracy of the ML model is not enough to understand its performance. Performance metrics should also reflect audience priorities. For instance, Sato et al. [63] recommend validating models for bias and fairness, while in the case described by Wagstaff et al. [40], controlling for consumption of spacecraft resources is crucial.

5.2 Formal Verification

The formal verification step verifies that software functionality follows the requirements defined within the scope of the project. For ML models such verification could include mathematical proofs of correctness or numerical estimates of output error bounds, but as Ashmore et al. [14] point out, this rarely happens in practice. More often, quality standards are being formally set via extensive regulatory frameworks that define what quality means and how models can be shown to meet them.

An example of where ML solutions have to adhere to regulations is the banking industry [64]. This requirement was developed in the aftermath of the global financial crisis, as the industry realized that there was a need for heightened scrutiny towards models. As a consequence, an increased level of regulatory control is now being applied to the processes that define how the models are built, approved, and maintained. For instance, official guidelines have been published by the UK's Prudential Regulation Authority [65] and European Central Bank [66]. These guidelines require ML model risk frameworks to be in place for all business decision-making solutions, and implementation of such frameworks requires developers to have extensive tests suites to understand the behavior of their ML models. The formal verification step in that context means ensuring that the model meets all criteria set by the corresponding regulations.

Regulatory frameworks share similarities with country-wide policies for governing the use of ML-powered technology, which we discuss in greater detail in Section 7.1.

5.3 Test-based Verification

In the context of ML, test-based verification is intended for ensuring that the model generalizes well to previously unseen data. While collecting a validation dataset is usually not a problem, as it can be derived from splitting the training dataset, it may not be sufficient for production deployment.

In an ideal setting, testing is done in a real-life setting, where business-driven metrics can be observed, as we discussed in Section 5.1. Full-scale testing in a real-world environment can be challenging for a variety of safety, security, and scale reasons and is often substituted with testing in simulation. That is the case for models for autonomous vehicles control [35]. Simulations are cheaper, faster to run, and provide flexibility to create situations rarely encountered in real life. Thanks to these advantages, simulations are becoming prevalent in this field. However, it is important to remember that simulation-based testing hinges on assumptions made by simulation developers, and therefore cannot be considered a full replacement for real-world testing. Even small variations between simulation and real world can have drastic effects on the system behavior, and therefore the authors conclude that validation of the model and simulation environment alone is not enough for autonomous vehicles. This point is emphasized further by the experiences from the field of reinforcement learning [34], where use of simulations is a de facto standard for training agents.

Hackett et al. presented an instructive use case of how limited simulation-based testing can be [67]. The authors were part of a team that conducted an experiment that explored a reinforcement learning–based **cognitive engine (CE)** for running a software-defined radio unit on board of the **International Space Station (ISS)**. Preparation for the experiment included extensive ground testing in an emulated environment that informed many hyper-parameter choices and the computational setup. Nevertheless, when the software was deployed on ISS, the actual conditions of the testing environment were so harsh the team was able to test only a subset of all planned experiments. The authors observed that, despite extensive preparation, CE was unable to cope with these emergency scenarios.

In addition, the dataset itself needs to be constantly validated to ensure data errors do not creep into the pipeline and do not affect the overall quality. Data issues that go unnoticed can cause problems down the line that are difficult to troubleshoot. Breck et al. [68] argue that such issues are common in the setup where data generation is decoupled from the ML pipeline. Data issues can originate from bugs in code, feedback loops, changes in data dependencies. They can propagate and manifest themselves at different stages of the pipeline, therefore it is imperative to catch them early by including data validation routines in the ML pipeline.

6 MODEL DEPLOYMENT

Machine learning systems running in production are complex software systems that have to be maintained over time. This presents developers with another set of challenges, some of which are shared with running regular software services, and some are unique to ML.

There is a separate discipline in engineering, called DevOps, that focuses on techniques and tools required to successfully maintain and support existing production systems. Consequently, there is a necessity to apply DevOps principles to ML systems. However, even though some of the DevOps principles apply directly, there are also a number of challenges unique to productionizing machine learning. This is discussed in detail by Dang et al. [69], which uses the term AIOps[5] for DevOps tasks for ML systems. Some of the challenges mentioned include lack of high-quality telemetry data as well as no standard way to collect it, difficulty in acquiring labels that makes

[5]Readers might have also encountered term MLOps (https://ml-ops.org/).

supervised learning approaches inapplicable,[6] and lack of agreed best practices around handling of machine learning models. In this section, we discuss issues concerning three steps within model deployment: integration, monitoring, and updating.

6.1 Integration

The model integration step constitutes of two main activities: building the infrastructure to run the model and implementing the model itself in a form that can be consumed and supported. While the former is a topic that belongs almost entirely in systems engineering and therefore lies out of the scope of this work, the latter is of interest for our study, as it exposes important aspects at the intersection of ML and software engineering. In fact, many concepts that are routinely used in software engineering are now being reinvented in the ML context.

Code reuse is a common topic in software engineering, and ML can benefit from adopting the same mindset. Reuse of data and models can directly translate into savings in terms of time, effort, or infrastructure. An illustrative case is an approach Pinterest took towards learning image embeddings [70]. There are three models used in Pinterest internally that use similar embeddings, and initially they were maintained completely separately to make it possible to iterate on the models individually. However, this created engineering challenges, as every effort in working with these embeddings had to be multiplied by three. Therefore, the team decided to investigate the possibility of learning a universal set of embeddings. It turned out to be possible, and this reuse ended up simplifying their deployment pipelines as well as improving performance on individual tasks.

A broad selection of engineering problems that machine learning practitioners now face is given in Sculley et al. [71]. Many of them are known anti-patterns in engineering,[7] but are currently widespread in machine learning software. Some of these issues, such as abstraction boundary erosion and correction cascades, are caused by the fact that ML is used in cases where the software has to take an explicit dependency on external data. Others, such as glue code or pipeline jungles, stem from the general tendency in the field to develop general-purpose software packages. Yet another source of problems discussed in the paper is the configuration debt: In addition to all configurations, a regular software system may require ML systems to add a sizable number of ML-specific configuration settings that have to be set and maintained.

Researchers and software engineers often find themselves working together on the same project aiming to reach a business goal with a machine learning approach. On the surface, there seems to be a clear separation of responsibilities: Researchers produce the model while engineers build the infrastructure to run it. In reality, their areas of concern often overlap when considering the development process, model inputs, and outputs and performance metrics. Contributors in both roles often work on the same code. Thus, it is beneficial to include researchers in the whole development journey, making sure they own the product codebase along with the engineers, use the same version control, and participate in code reviews. Despite obvious onboarding and slow-start challenges, this approach was seen to bring long-term benefits in terms of speed and quality of product delivery [12].

6.2 Monitoring

Monitoring is one of the issues associated with maintaining machine learning systems, as reported by Sculley et al. [71]. While monitoring is crucial for the maintenance of any software service, the ML community is in the early stages of understanding what are the key metrics of data and

[6]Please refer to Section 3.3 for detailed discussion about data labeling.

[7]In software engineering, an anti-pattern is understood as a common response to a recurring problem that is considered ineffective or counterproductive.

models to monitor and how to trigger system alarms when they deviate from normal behavior. Monitoring of evolving input data, prediction bias, and overall performance of ML models is an open problem. Another maintenance issue highlighted by this article that is specific to data-driven decision-making is feedback loops. ML models in production can influence their own behavior over time via regular retraining. While making sure the model stays up to date, it is possible to create a feedback loop where the input to the model is being adjusted to influence its behavior. This can be done intentionally, as well as inadvertently, which is a unique challenge when running live ML systems.

Klaise et al. [72] point out the importance of outlier detection as a key instrument to flag model predictions that cannot be used in a production setting. The authors name two reasons for such predictions to occur: the inability of the models to generalize outside of the training dataset and overconfident predictions on out-of-distribution instances due to poor calibration. Deployment of the outlier detector can be a challenge in its own right, because labeled outlier data is scarce, and the detector training often becomes a semi-supervised or even an unsupervised problem.

Additional insight on monitoring of ML systems can be found in Ackermann et al. [73]. This article describes an **early intervention system (EIS)** for two police departments in the US. On the surface, their monitoring objectives seem completely standard: data integrity checks, anomaly detection, and performance metrics. One would expect to be able to use out-of-the-box tooling for these tasks. However, the authors explain that they had to build all these checks from scratch to maintain good model performance. For instance, the data integrity check meant verifying updates of a certain input table and checksums on historical records, the performance metric was defined in terms of the number of changes in top k outputs, and anomalies were tracked on rank-order correlations over time. All of these monitoring tools required considerable investigation and implementation. This experience report highlights a common problem with currently available end-to-end ML platforms: The final ML solutions are usually so sensitive to a problem's specifics that out-of-the-box tooling does not fit their needs well.

As a final remark, we note that there is an overlap between the choice of metrics for monitoring and validation. The latter topic is discussed in Section 5.1.

6.3 Updating

Once the initial deployment of the model is completed, it is often necessary to be able to update the model later on to make sure it always reflects the most recent trends in data and the environment. There are multiple techniques for adapting models to new data, including scheduled regular retraining and continual learning [74]. Nevertheless, in the production setting, model updating is also affected by practical considerations.

A particularly important problem that directly impacts the quality and frequency of model update procedure is the concept drift, also known as dataset shift [75]. Concept drift in ML is understood as changes observed in joint distribution $p(X, y)$, where X is the model input and y is the model output. Such changes can occur discretely, for example after some outside event that affects the input data distribution, or continuously, when data is gradually changing over time. Undetected, this phenomenon can have major adverse effects on model performance, as is shown by Jameel et al. [76] for classification problems or by Celik and Vanschoren [77] in the AutoML context. Concept drift can arise due to a wide variety of reasons. For example, the finance industry faced turbulent changes as the financial crisis of 2008 was unfolding, and if advanced detection techniques were employed they could have provided additional insights into the ongoing crisis, as explained by Masegosa et al. [78]. Changes in data can also be caused by an inability to avoid fluctuations in the data collection procedure, as described in the paper by Langenkämper et al. [79], which studies the effects of slight changes in marine images on deep learning models' performance.

Data shifts can have noticeable consequences even when occurring at a microscopic scale, as Zenisek et al. [80] show in their research on predictive maintenance for wear and tear of industrial machinery. Even though concept drift has been known for decades [81], these examples show that it remains a critical problem for applications of ML today. Consequently, detection of concept drift becomes a growing concern for teams that maintain ML models in production [82, 83], which makes this challenge directly related to monitoring discussed in the previous section.

On top of the question of when to retrain the model to keep it up to date, there is an infrastructural question on how to deliver the model artifact to the production environment. In software engineering such tasks are commonly solved with **continuous delivery (CD)**, which is an approach for accelerating the development cycle by building an automated pipeline for building, testing, and deploying software changes. CD for machine learning solutions is complicated because, unlike in regular software products where changes only happen in the code, ML solutions experience change along three axes: the code, the model, and the data. An attempt to formulate CD for ML as a separate discipline can be seen in Sato et al. [63]. This work describes the pieces involved and the tools that can be used at each step of building the full pipeline. A direct illustration of the benefits that a full CD pipeline can bring to the real-life ML solution can be found in Wider and Deger [84].

While updating is necessary for keeping a model up to date with recent fluctuations in the data, it may also inflict damage on users or downstream systems because of the changes in the model's behavior, even without causing obvious software errors. To study updates in teams where AI is used to support human decision-making, Bansal et al. [85] introduced the notion of compatibility of an AI update. Authors define an update as compatible only if it does not violate the user's trust characterized via model behavior. The authors proceeded to show that updating an AI model to increase accuracy, at the expense of compatibility, may degrade overall AI-Human team performance. This line of research is continued by Srivastava et al. [86], who provide a detailed empirical study of backward-compatibility issues in ML systems. They show how ML model updates may become backward-incompatible due to optimization stochasticity or noisy training datasets. These findings motivate the need for de-noising and compatibility-aware training methods as the means to ensure reliable updates of deployed ML models.

7 CROSS-CUTTING ASPECTS

In this section, we describe three additional aspects that ML projects have to consider: ethics, law, end-users' trust, security. These aspects can affect every stage of the deployment pipeline.

7.1 Ethics

Ethical considerations should always inform data collection and modeling activities.[8] As stated in the report on ethical AI produced by the Alan Turing Institute [87], "it is essential to establish a continuous chain of human responsibility across the whole AI project delivery workflow." If researchers and developers do not follow this recommendation, then complications may come up due to a variety of reasons, some of which we illustrate in this section.

Since ML models use previously seen data to make decisions, they can rely on hidden biases that already exist in data—a behavior that is hard to foresee and detect. This effect is discussed in detail by O'Neil [88] in the field of criminal justice. Models that calculate a person's criminal "risk score" are often marketed as a way to remove human bias. Nevertheless, they use seemingly neutral demographic information like a neighborhood that often ends up serving as a proxy for more

[8]Here, by "ethics," we understand moral principles and techniques that inform the responsible development and use of ML or AI solutions.

sensitive data such as race. As a result, people are disadvantaged on the basis of race or income. Machine translation is another example of an area where such hidden biases exist in data and can be exploited via an ML system. Prates et al. [89] show a strong tendency towards male defaults in popular online translation service, in particular for fields typically associated with unbalanced gender distribution, such as STEM.

Likewise, Soden et al. [90] mention the aggravation of social inequalities through the use of biased training datasets as one of the main hurdles in applying ML to **Disaster Risk Management (DRM)**. It is argued that ML causes privacy and security[9] concerns through a combination of previously distinct datasets. Reducing the role of both experts and the general public is also seen as an ethical issue by DRM professionals, because they feel it increases the probability of error or misuse. Similar worries about unintentional or malign misuse of ML decision-making systems are expressed by Muthiah et al. [91]. Their software for predicting civil unrest, called EMBERS, is designed to be used as a forecasting and communication tool, however, authors remark that it can also be potentially misused by governments, either due to a misunderstanding of its role in society or deliberately.

ML models for facial analysis often become subjects to criticism due to their unethical behavior. For example, Buolamwini and Gebru [92] analyzed popular datasets for facial analysis and discovered them to be imbalanced on the basis of skin colour. They have also evaluated three commercial classification systems and showed darker-skinned females to be the most misclassified group. Authors conclude that urgent attention towards gender and skin type is needed for businesses that want to build genuinely fair facial analysis algorithms. Such questions about fairness and accountability of ML algorithms and data are studied by the branch of machine learning known as Fairness in ML [93].

An interesting ethical aspect arises in the usage of ML in the field of creative arts, discussed by Anantrasirichai and Bull [94]. When a trained model is used to create a piece of visual art, it is not entirely clear where the authorship of this piece resides. The questions of originality therefore requires special attention. Closely related to this question is the growing concern of fake content being generated with ML, such as deepfake images and video, which can be easily used for the wrong purposes [95].

7.2 Law

As ML grows its influence on society's everyday life, it is natural to expect more regulations to govern how ML models should function and how businesses, governments, and other bodies can use them. Such legal frameworks can sometimes be used to guide decisions on ethics, although in general ethics and legal should be considered separate aspects.

Various countries have produced regulations to protect personal data rights. Typically, the more sensitive the information collected from the individual, the stronger the regulations governing its use. Examples of such regulations include the General Data Protection Regulation in the European Union [96] and ethical screening laws in a range of Asian countries [97]. One domain that deals with some of the most sensitive information is healthcare. According to Han et al. [98], many countries have strict laws in place to protect the data of patients, which makes the adoption of ML in healthcare particularly difficult. On one hand, there is no doubt that these rules are absolutely necessary to make sure people are comfortable with their data being used; on the other hand, the amount of reviews, software updates, and cycles of data collection/annotation that are required make it exceptionally hard to keep up with technical advances in ML, as Han et al. [98] explain following their experience deploying ML solutions in the healthcare sector in Japan.

[9]We discuss related cross-cutting security concerns in Section 7.4.

Legislation takes time to develop and often cannot keep up with the speed of progress in ML. This phenomenon is well known in policymaking and has been discussed in the context of ML as well as other technological advances by Marchant [99] or more recently by the World Economic Forum [100, 101]. As Malan [101] explains, by the time regulations are written they can already be out of date, resulting in a cat-and-mouse game that is wasteful on resources and causes legal framework abuses. Additionally, it is generally challenging to formulate specific and unambiguous laws for such a rapidly developing area as ML. For example, Wachter et al. show that GDPR lacks precise language as well as explicit and well-defined rights and safeguards, therefore failing to guarantee the "right to explanation" [102]. As a result of these challenges, ML applications often have to abide by the existing laws of the area where they are deployed. Minssen et al. analyze the current regulatory approaches to medical ML in the US and Europe and discuss how existing laws evaluate ML applications [103]. At the same time governments have to provide ML adoption plans. For instance, the US Food and Drug Administration released an action plan outlining steps the agency plans to take to produce a regulatory framework for medical ML solutions [104].

Companies should not be focusing solely on the technological side of their solutions, as DeepMind and Royal Free NHS Foundation Trust discovered while working on Streams, an application for automatic review of test results for serious conditions. Their initial collaboration was not specific enough on the use of patient data and on patient involvement overall, which triggered an investigation on their compliance with data protection regulations. The revised collaboration agreement was far more comprehensive and included a patient and public engagement strategy to ensure data is being used ethically [105].

7.3 End-users' Trust

ML is often met cautiously by the end-users [3, 106, 107]. On their own accord, models provide minimal explanations, which makes it difficult to persuade end-users of their utility [98]. To convince users to trust ML based solutions, the time has to be invested to build that trust. In this section, we explore ways in which that is done in practice.

If an application has a well-defined accessible audience, then getting that audience involved early in the project is an efficient way to foster their confidence in the end product. This approach is very common in medicine, because the end product is often targeted at a well-defined group of healthcare workers and/or patients. One example is the project called Sepsis Watch [108]. In this project the goal was to build a model that estimates a patient's risk of developing sepsis. It was not the first attempt at automating this prediction and, since previous attempts were considered failures, medical personnel were skeptical about the eventual success of Sepsis Watch. To overcome this skepticism, the team prioritized building trust, with strong communication channels, early engagement of stakeholders, front-line clinicians and decision-makers, and established accountability mechanisms. One of the key messages of this work is that model interpretability has limits as a trust-building tool, and other ways to achieve high credibility with the end-users should be considered. This aligns with conclusions made by "Project explAIn," which found that the relative importance of explanations of AI decisions varies by context [109]. A similar argument is made by Soden et al. [90], who explore the impact ML has on **disaster risk management (DRM)**. Due to the growing complexity of the ML solutions deployed, it is becoming harder for the public to participate and consequently to trust the ML-based DRM services, such as flooding area estimates or prediction of damage from a hurricane. As a mitigation measure, the authors recommend making the development of these solutions as transparent as possible by taking into account the voice of residents in the areas portrayed by models as "at risk" and relying on open software and data whenever possible. The importance of strong communication and engagement

with early adopters is also emphasized by Mutembesa et al. [110] as they analyzed their experience of deploying a nation-wide cassava disease surveillance system in Uganda.

While the projects described above focused on engagement and accountability, in other circumstances explainability is the key to building the trust of the target audience. This is often the case when the users have experience and an understanding of ML. Rudin [111] called on the ML community to stop using black-box models and explaining their behavior afterward, and instead design models that are inherently interpretable. Bhatt et al. [11] analyzed explainability as a feature of machine learning models deployed within enterprises and found that it is a must-have requirement for most stakeholders, including executives, ML engineers, regulators, and others. Moreover, their survey showed that explainability score is a desired model metric, along with measures of fairness and robustness. Explainability is also necessary in cases where it is demanded by the existing regulations,[10] and users will not trust decisions that are made automatically without provided explanations. Wang et al. [112] describe such a requirement in the context of credit risk scoring. They observed that the XGBoost algorithm outperforms traditional scorecard approaches, but lacks the necessary explainability component. This prompted the authors to develop a custom loan-decision explanation technique for XGBoost, subsequently deployed by QuickBooks Capital.

A poorly designed user interface can be one of the main obstacles in the adoption of any new technology. While this problem is not specific to ML, it is nevertheless worth mentioning it as a challenge ML applications face. For example, Wang et al. studied the deployment of the AI-powered medical diagnosis tool "Brilliant Doctor" in rural China and discovered that the majority of doctors could not use it productively. One of the main reasons quoted was the UX design that did not take into account particularities of the environment (screen sizes, interaction with other software in the clinic) where it was installed, often resulting in an unusable software [113]. On the contrary, investing time in specialized user interfaces with tailored user experience can pay off with fast user adoption. Developers of Firebird [27], a system that helps identify priority targets for fire inspection in the city of Atlanta (Georgia, USA), found that the best way to avoid resistance from the end-users while transitioning to an ML solution as a replacement of the previously used pen-and-paper method was to develop a user interface that presented the results of modelling in a way that the end-users (fire officers and inspectors in the fire department) found most useful and clear. Similarly, authors of EMBERS [91], a system that forecasts population-level events (such as protest) in Latin America, noticed that their users have two modes of using the system: (a) high recall: obtain most events and then filter them using other methods; (b) high precision: focus on a specific area or a specific hypothesis. To improve the user experience and thus increase their confidence in the product, the user interface was improved to easily support both modes. This case study emphasizes the importance of context-aware personalization for ML systems' interfaces, one of the key observations delivered by "Project explAIn" [109].

7.4 Security

Machine Learning opens up opportunities for new types of security attacks across the whole ML deployment workflow [114]. Specialized adversarial attacks for ML can occur on the model itself, the data used for training, and also the resulting predictions. The field of adversarial machine learning studies the effect of such attacks against ML models and how to protect against them [115, 116]. Recent work from Siva et al. found that industry practitioners are not equipped to protect, detect, and respond to attacks on their ML systems [117]. In this section, we describe the three most common attacks reported in practice that affect deployed ML models: data poisoning, model stealing, and model inversion. We focus specifically on adversarial machine learning and

[10]Effects of regulations on ML deployment are also discussed in Section 5.2

consider other related general security concerns in deploying systems such as access control and code vulnerabilities beyond the scope of our work.

In data poisoning, the goal of the adversarial attack is to deliberately corrupt the integrity of the model during the training phase to manipulate the produced results. In data-poisoning scenarios, an attacker is usually assumed to have access to the data that will ultimately be used for training, for example by sending emails to victims to subvert their spam filter [118]. Poisoning attacks are particularly relevant in situations where the machine learning model is continuously updated with new incoming training data. Jagielski et al. reported that in a medical setting using a linear model, the introduction of specific malicious samples with an 8% poisoning rate in the training set resulted in incorrect dosage for half of the patients [119].

Data poisoning can also occur as a result of a coordinated collective effort that exploits feedback loops we have discussed in Section 6.2, as happened with Microsoft's Twitter bot Tay [120]. Tay was designed to improve its understanding of the language over time but was quickly inundated with a large number of deliberately malevolent tweets. Within 16 hours of its release a troubling percentage of Tay's messages were abusive or offensive, and the bot was taken down.

Another type of adversarial attack is reverse engineering a deployed model by querying its inputs (e.g., via a public prediction API) and monitoring the outputs. The adversarial queries are crafted to maximize the extraction of information about the model to train a substitute model. This type of attack is referred to as model stealing. In a nutshell, this attack results in the loss of intellectual property, which could be a key business advantage for the defender. Tramèr et al. [121] have shown that it is possible to replicate models deployed in production from ML services offered by Google, Amazon, and Microsoft across a range of ML algorithms including logistic regression, decision trees, SVMs, and neural networks. In their work, they report the number of queries ranging from 650 to 4,013 to extract an equivalent model and in time ranging from 70 s to 2,088 s.

A related attack is that of model inversion, where the goal of the adversarial attack is to recover parts of the private training set, thereby breaking its confidentiality. Fredrikson et al. have shown that they could recover training data by exploiting models that report confidence values along with their predictions [122]. Veale et al. [123] emphasize the importance of protecting against model inversion attacks as a critical step towards compliance with data protection laws such as GDPR.

8 DISCUSSION OF POTENTIAL SOLUTIONS

This survey looked at case studies from a variety of industries: computer networks, manufacturing, space exploration, law enforcement, banking, and more. However, further growth of ML adoption can be severely hindered by poor deployment experience. To make the ML deployment scalable and accessible to every business that may benefit from it, it is important to understand the most critical pain points and to provide tools, services, and best practices that address those points. We see this survey as an initial step in this direction: By recognizing the most common challenges currently being reported, we hope to foster an active discussion within the academic community about what possible solutions might be. We classify possible research avenues for solutions into two categories, which we discuss below. We also give some concrete examples, but, since the purpose of this section is illustrative, we do not aim to provide a complete survey of ML tools and development approaches.

8.1 Tools and Services

The market for machine learning tools and services is experiencing rapid growth [124]. As a result, tools for individual deployment problems are continuously developed and released. Consequently, some of the problems we have highlighted can be solved with the right tool.

For example, this is most likely the case for operational maintenance of ML models, discussed in Sections 6.2 and 6.3. Many platforms on the market offer end-to-end experience for the user, taking care of such things as data storage, retraining, and deployment. Examples include AWS SageMaker [125], Microsoft ML [126], Uber Michelangelo [127], TensorFlow TFX [128], MLflow [129], and more. A typical ML platform would include, among other features, a data storage facility, model hosting with APIs for training and inference operations, a set of common metrics to monitor model health, and an interface to accept custom changes from the user. By offering managed infrastructure and a range of out-of-the-box implementations for common tasks, such platforms greatly reduce the operational burden associated with maintaining the ML model in production.

Quality assurance, which is the focus of Section 5, also looks to be an area where better tools can be of much assistance. Models can greatly benefit from the development of a test suite to verify their behavior, and the community actively develops tools for that purpose. Jenga [130] ensures model's robustness against errors in data, which very commonly occur in practice, as was mentioned in Section 3. CheckList methodology [131] provides a formal approach towards assessing the quality of NLP models. The Data Linter [132] inspects ML datasets to identify potential issues in the data.

As discussed in Section 3.3, obtaining labels is often a problem with real-world data. Weak supervision has emerged as a separate field of ML that looks for ways to address this challenge. Consequently, a number of weak supervision libraries are now actively used within the community and show promising results in industrial applications. Some of the most popular tools include Snorkel [133], Snuba [134], and cleanlab [135].

A growing field of AutoML [136] aims to address challenges around a model selection and hyperparameter tuning, discussed in Sections 4.1 and 4.3. There is a large variety of tools that provide general-purpose implementations of AutoML algorithms, such as Auto-keras [137], Auto-sklearn [138], or TPOT [139]. However, practical reports of applying AutoML to real-world problems indicate that practitioners need to exercise extreme caution, as AutoML methods might not be ready for decision-making in high-stakes areas yet [140, 141].

Given the potential damage an unnoticed dataset shift can cause to the quality of predictions of deployed ML models (see Section 6.3), there are many techniques to detect and mitigate its effects. A large variety of methods based on dimensionality reduction and statistical hypothesis testing is reviewed by Rabanser et al. [142] and are now implemented in software libraries (e.g., Alibi Detect [143]) and services (e.g., Azure ML[11] and AWS Sagemaker[12]), available for use in deployment's monitoring suites. The community has also made strides in developing strategies for dealing with the shift once it was detected. We refer interested readers to the works on using domain adaptation [144], meta-learning [145], and transfer learning [146, 147] as ways of addressing dataset shift.

Using specific tools for solving individual problems is a straightforward approach. However, practitioners need to be aware that by using a particular tool they introduce an additional dependency into their solution. While a single additional dependency seems manageable, their number can quickly grow and become a maintenance burden. Besides, as we mentioned above, new tools for ML are being released constantly, thus presenting practitioners with the dilemma of choosing the right tool by learning its strengths and shortcomings.

8.2 Holistic Approaches

Even though ML deployments require software development, ML projects are fundamentally different from traditional software engineering projects. The main differences arise from unique

[11] https://docs.microsoft.com/en-us/azure/machine-learning/how-to-monitor-datasets.
[12] https://docs.aws.amazon.com/sagemaker/latest/dg/clarify-model-monitor-feature-attribution-drift.html.

activities such as data discovery, dataset preparation, model training, deployment success measurement, and so on. Some of these activities cannot be defined precisely enough to have a reliable time estimate (as discussed in Section 3.1 in regard to data collection), some require a different style of project management (Section 6.1 discusses the challenges of managing mixed team engineers and scientists), and some make it difficult to measure the overall added value of the project (see Section 5.1 for the discussion of translation of ML model performance to the business value). For these reasons, ML deployment projects often do not lend themselves well to widespread approaches to software engineering management paradigms and neither to common software architectural patterns [148].

Compared to classical **software engineering (SE)**, ML introduces unique artifacts with unique characteristics: datasets and models. Unlike program source and configuration files, these artifacts are not distributed as program code and exist in the form of tabular or binary files. Regular SE tools for such common operations as source control, branching and merging, review, cannot be applied to these new artifacts "as is" [149]. Consequently, it is essential to develop documentation approaches that are most suitable for these artifacts. There is a growing body of literature that aims to adapt existing or develop new practices for handling datasets and models in a coherent and reliable way. Lawrence [150] proposes an approach toward classifying the readiness of data for ML tasks, which Royal Society DELVE applied in one of their reports in the context of COVID-19 pandemic [20]. Gebru et al. suggested "datasheets for datasets" [151] to document dataset's motivation, composition, collection process, and intended purposes. **Data Version Control (DVC)** is an open-source project that aims to create a Git-like source control experience for datasets [152]. For models, Mitchell et al. [153] proposed model cards, short documents that accompany trained models detailing their performance, intended use context, and other information that might be relevant for model's application.

Data-Oriented Architectures (DOA) [154–156] is an example of an idea that suggests rethinking how things are normally approached in software development, and by doing so promises to solve many of the issues we have discussed in this survey. Specifically, the idea behind DOA is to consider replacing micro-service architecture, widespread in current enterprise systems, with dataflow-based architectures, thus making data flowing between elements of business logic more explicit and accessible. Micro-service architectures have been successful in supporting high scalability and embracing the single responsibility principle. However, they also make dataflows hard to trace, and it is up to owners of every individual service to make sure inputs and outputs are being stored in a consistent form (these issues are also discussed in Section 3.1). DOA provides a solution to this problem by moving data to streams flowing between stateless execution nodes, thus making data available and traceable by design, therefore making simpler the tasks of data discovery, collection, and labeling. In its essence, DOA proposes to acknowledge that modern systems are often data-driven and therefore need to prioritize data in their architectural principles.

As noted above, ML projects normally do not fit well with commonly used management processes, such as Scrum or Waterfall. Therefore, it makes sense to consider processes tailored specifically for ML. One such attempt is done by Lavin et al. [157], who propose **Machine Learning Technology Readiness Levels (MLTRL)** framework. MLTRL describes a process of producing robust ML systems that takes into account key differences between ML and traditional software engineering with a specific focus on the quality of the intermediate outcome of each stage of the project. As we discussed in Section 5, verification is the area of ML deployment that suffers from a lack of standard practices, and in that context MLTRL suggests a possible way to define such standards.

A very widespread practice in software engineering is to define a set of guidelines and best practices to help developers make decisions at various stages of the development process. These

guidelines can cover a wide range of questions, from variable names to execution environment setup. For example, Zinkevich [158] compiled a collection of best practices for machine learning that are utilized in Google. While this cannot be viewed as a coherent paradigm for doing ML deployment, this document gives practical advice on a variety of important aspects that draw from the real-life experiences of engineers and researchers in the company. Among others, rules and suggestions for such important deployment topics as monitoring (discussed in Section 6.2), end-user experience (Section 7.3), and infrastructure (Section 6.1) are described.

Besides serving as a collection of advice for common problems, guidelines can also be used as a way to unify approaches towards deploying ML in a single area. The Association of German Engineers (VDI) has released a series of guidelines on various aspects of big data applications in the manufacturing industry [159]. These documents cover a wide range of subjects, including data quality, modeling, user interfaces, and more. The series aims to harmonize the available technologies used in the industry, facilitate cooperation and implementation. Such initiatives can help bridge the gap between ML solutions and regulations in a particular applied area discussed in Section 7.2 of this survey.

Holistic approaches are created with ML application in mind, and therefore they have the potential to offer significant ease of deploying ML. But it should be noted that all such approaches assume significant time investment, because they represent significant changes to current norms in project management and development. Therefore, a careful assessment of risks versus benefits should be carried out before adopting any of them.

9 FURTHER WORK

Even though the set of challenges we reviewed covers every stage of the ML deployment workflow, it is far from complete. Identifying other, especially non-technical, challenges is a natural direction of further work and could be augmented by conducting interviews with industry representatives about their experiences of deploying ML.

In this article, we reviewed reports from a variety of industries, which shows the ubiquity and variety of challenges with deploying ML in production. An interesting extension would be the comparative analysis of industries. Quantitative and qualitative analysis of most commonly reported challenges may open interesting transferability opportunities, as approaches developed in one field may be applicable in the other.

Our work includes a brief discussion of existing tools in Section 8.1. However, the community would benefit from a comprehensive review of currently available tools and services, mapped to challenges reported in our study. This new work could be combined with our survey to enable practitioners to identify the problem they are facing and choose the most appropriate tool that addresses that problem.

10 CONCLUSION

In this survey, we showed that practitioners deal with challenges at every step of the ML deployment workflow due to practical considerations of deploying ML in production. We discussed challenges that arise during the data management, model learning, model verification, and model deployment stages, as well as considerations that affect the whole deployment pipeline including ethics, end-users' trust, law, and security. We illustrated each stage with examples across different fields and industries by reviewing case studies, experience reports, and the academic literature.

We argue that it is worth the academic community's time and focus to think about these problems, rather than expect each applied field to figure out their own approaches. We believe that ML researchers can drive improvements to the ML deployment experience by exploring holistic approaches and taking into account practical considerations.

ML shares a lot of similarities with traditional computer science disciplines and consequently faces similar challenges, albeit with its own peculiarities. Therefore, ML as a field would benefit from a cross-disciplinary dialog with such fields as software engineering, human-computer interaction, systems, policymaking. Many pain points we have described in this work were already experienced by communities in these fields, and the ML community should turn to them for solutions and inspiration.

As an observation that follows from the process of collecting papers to review in this survey, we note the relative shortage of deployment experience reports in the academic literature. Valuable knowledge obtained by industry ML practitioners goes unpublished. We would like to encourage organizations to prioritize sharing such reports, as they provide valuable information for the wider community, but also as a way to self-reflect, collect feedback, and improve on their own solutions. Nevertheless, several venues focused on ML deployment already exist, and we encourage interested readers to follow them:

- International Conference Knowledge Discovery and Data Mining (ACM SIGKDD), Applied Data Science track.
- Conference on Innovative Applications of Artificial Intelligence (IAAI).
- European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD), Applied Data Science track.
- IEEE International Conference on Machine Learning and Applications (ICMLA).
- *Applied Artificial Intelligence* journal.
- *Neural Computing and Applications* journal.

We hope this survey will encourage discussions within the academic community about pragmatic approaches to deploying ML in production.

ACKNOWLEDGMENTS

We would like to thank our reviewers for thoughtful and detailed comments that helped improve the article. We would also like to thank Jessica Montgomery, Diana Robinson and Ferenc Huszar for insightful discussions.

REFERENCES

[1] Arif Cam, Michael Chui, and Bryce Hall. 2019. Global AI survey: AI proves its worth, but few scale impact. *McKinsey Analytics* (2019). https://www.mckinsey.com/featured-insights/artificial-intelligence/global-ai-survey-ai-proves-its-worth-but-few-scale-impact. Accessed from 13/06/2022.

[2] Thomas H. Davenport and Rajeev Ronanki. 2018. Artificial intelligence for the real world. *Harv. Bus. Rev.* 96, 1 (2018), 108–116.

[3] Royal Society (Great Britain). 2017. *Machine Learning: The Power and Promise of Computers that Learn by Example: an Introduction*. Royal Society.

[4] Michal Pěchouček and Vladimír Mařík. 2008. Industrial deployment of multi-agent technologies: Review and selected case studies. *Auton. Agents and Multi-agent Syst.* 17, 3 (2008), 397–431.

[5] Kyle Wiggers. 2019. Algorithmia: 50% of companies spend between 8 and 90 days deploying a single AI model. Retrieved from https://venturebeat.com/2019/12/11/algorithmia-50-of-companies-spend-upwards-of-three-months-deploying-a-single-ai-model/.

[6] Lawrence E. Hecht. 2019. Add It Up: How Long Does a Machine Learning Deployment Take? Retrieved from https://thenewstack.io/add-it-up-how-long-does-a-machine-learning-deployment-take/.

[7] Kyle Wiggers. 2019. IDC: For 1 in 4 companies, half of all AI projects fail. Retrieved from https://venturebeat.com/2019/07/08/idc-for-1-in-4-companies-half-of-all-ai-projects-fail/.

[8] Ben Lorica and Nathan Paco. 2018. *The State of Machine Learning Adoption in the Enterprise*. O'Reilly Media.

[9] 2019. The state of development and operations of AI applications. *Dotscience* Retrieved from https://dotscience.com/assets/downloads/Dotscience_Survey-Report-2019.pdf.

[10] 2019. Artificial intelligence and machine learning projects are obstructed by data issues. *Dimens. Res.* Retrieved from https://telecomreseller.com/wp-content/uploads/2019/05/EMBARGOED-UNTIL-800-AM-ET-0523-Dimensional-Research-Machine-Learning-PPT-Report-FINAL.pdf.

[11] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. 2020. Explainable machine learning in deployment. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 648–657.

[12] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP'19)*. IEEE, 291–300.

[13] Lucas Baier, Fabian Jöhren, and Stefan Seebacher. 2019. Challenges in the deployment and operation of machine learning in practice. In *Proceedings of the 27th European Conference on Information Systems*.

[14] Rob Ashmore, Radu Calinescu, and Colin Paterson. 2021. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *ACM Comput. Surv.* 54, 5 (2021), 1–39.

[15] Colin Shearer. 2000. The CRISP-DM model: The new blueprint for data mining. *J. Data Warehous.* 5, 4 (2000), 13–22.

[16] W. A. R. Roald Bradley Severtson, L. Franks, and G. Ericson. 2017. What is the team data science process? *Microsoft Azure* (2017). https://docs.microsoft.com/en-us/azure/architecture/data-science-process/overview. Accessed from 13/06/2022.

[17] Hironori Takeuchi and Shuichiro Yamamoto. 2020. Business analysis method for constructing business–AI alignment model. *Procedia Comput. Sci.* 176 (2020), 1312–1321.

[18] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. 2018. Data lifecycle challenges in production machine learning: A survey. *ACM SIGMOD Rec.* 47, 2 (2018), 17–28.

[19] Hongming Cai, Boyi Xu, Lihong Jiang, and Athanasios V. Vasilakos. 2016. IoT-based big data storage systems in cloud computing: Perspectives and challenges. *IEEE Internet Things J.* 4, 1 (2016), 75–87.

[20] The DELVE Initiative. 2020. *Data Readiness: Lessons from an Emergency*. Technical Report. Retrieved from https://rs-delve.github.io/reports/2020/11/24/data-readiness-lessons-from-an-emergency.html.

[21] Jimmy Lin and Dmitriy Ryaboy. 2013. Scaling big data mining infrastructure: The Twitter experience. *ACM SIGKDD Explor. Newslett.* 14, 2 (2013), 6–19.

[22] Robert C. Martin. 2002. The single responsibility principle. *Princ., Patterns, Pract. Agile Softw. Devel.* (2002), 149–154.

[23] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting data errors: Where are we and what needs to be done? *Proc. VLDB Endow.* 9, 12 (2016), 993–1004.

[24] Rajashree Y. Patil and R. V. Kulkarni. 2012. A review of data cleaning algorithms for data warehouse systems. *Int. J. Comput. Sci. Inf. Technol.* 3, 5 (2012), 5212–5214.

[25] Fakhitah Ridzuan and Wan Mohd Nazmee Wan Zainon. 2019. A review on data cleansing methods for big data. *Procedia Comput. Sci.* 161 (2019), 731–738.

[26] Alfredo Nazabal, Christopher K. I. Williams, Giovanni Colavizza, Camila Rangel Smith, and Angus Williams. 2020. Data engineering for data analytics: A classification of the issues, and case studies. *arXiv preprint arXiv:2004.12929* (2020).

[27] Michael Madaio, Shang-Tse Chen, Oliver L. Haimson, Wenwen Zhang, Xiang Cheng, Matthew Hinds-Aldrich, Duen Horng Chau, and Bistra Dilkina. 2016. Firebird: Predicting fire risk and prioritizing fire inspections in Atlanta. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 185–194.

[28] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar. 2019. Towards the deployment of machine learning solutions in network traffic classification: A systematic survey. *IEEE Commun. Surv. Tutor.* 21, 2 (2019), 1988–2014.

[29] Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli. 2011. Quantifying the accuracy of the ground truth associated with Internet traffic traces. *Comput. Netw.* 55, 5 (2011), 1158–1167.

[30] Samuel Budd, Emma C. Robinson, and Bernhard Kainz. 2021. A survey on active learning and human-in-the-loop deep learning for medical image analysis. *Med. Image Anal.* 71 (2021), 102062.

[31] Jun Du and Charles X. Ling. 2010. Active learning with human-like noisy oracle. In *Proceedings of the IEEE International Conference on Data Mining*. IEEE, 797–802.

[32] Julia Peyre, Josef Sivic, Ivan Laptev, and Cordelia Schmid. 2017. Weakly-supervised learning of visual relations. In *Proceedings of the IEEE International Conference on Computer Vision*. 5179–5188.

[33] Wendi Ren, Yinghao Li, Hanting Su, David Kartchner, Cassie Mitchell, and Chao Zhang. 2020. Denoising multi-source weak supervision for neural text classification. *arXiv preprint arXiv:2010.04582* (2020).

[34] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Mach. Learn.* 110, 9 (2021), 2419–2468.

[35] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. 2020. A survey of deep learning applications to autonomous vehicle control. *IEEE Trans. Intell. Transport. Syst.* 22, 2 (2020), 712–733.
[36] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. 2012. Enterprise data analysis and visualization: An interview study. *IEEE Trans. Visualiz. Comput. Graph.* 18, 12 (2012), 2917–2926.
[37] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. 2017. Data scientists in software teams: State of the art and challenges. *IEEE Trans. Softw. Eng.* 44, 11 (2017), 1024–1038.
[38] Diego Charrez. 2019. NeurIPS 2019 Stats. Retrieved from https://medium.com/@dcharrezt/neurips-2019-stats-c91346d31c8f.
[39] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C. Turnbull, Brendan M. Collins, et al. 2019. Applying deep learning to AirBnB search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. http://dx.doi.org/10.1145/3292500.3330658
[40] Kiri L. Wagstaff, Gary Doran, Ashley Davies, Saadat Anwar, Srija Chakraborty, Marissa Cameron, Ingrid Daubar, and Cynthia Phillips. 2019. Enabling onboard detection of events of scientific interest for the Europa Clipper spacecraft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'19)*. Association for Computing Machinery, New York, NY, 2191–2201. http://dx.doi.org/10.1145/3292500.3330656
[41] Ursula Challita, Henrik Ryden, and Hugo Tullberg. 2020. When machine learning meets wireless cellular networks: Deployment, challenges, and applications. *IEEE Commun. Mag.* 58, 6 (2020), 12–18.
[42] J. Ross Quinlan. 1986. Induction of decision trees. *Mach. Learn.* 1, 1 (1986), 81–106.
[43] Karl Hansson, Siril Yella, Mark Dougherty, and Hasan Fleyeh. 2016. Machine learning algorithms in heavy process manufacturing. *Amer. J. Intell. Syst.* 6, 1 (2016), 1–13.
[44] Abbas Keramati, Hajar Ghaneei, and Seyed Mohammad Mirmohammadi. 2016. Developing a prediction model for customer churn from electronic banking services using data mining. *Financ. Innov.* 2, 1 (2016), 10.
[45] Adrian Carrio, Carlos Sampedro, Alejandro Rodriguez-Ramos, and Pascual Campoy. 2017. A review of deep learning methods and applications for unmanned aerial vehicles. *J. Sensors* (2017).
[46] Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The cost of training NLP models: A concise overview. *arXiv preprint arXiv:2004.08900* (2020).
[47] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*.
[48] Nathan Benaich and Ian Hogarth. 2020. State of AI report 2020. Retrieved from www.stateof.ai.
[49] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2020. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34, 13693–13696.
[50] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*. 610–623.
[51] Nicolas Papernot, Patrick McDaniel, Arunesh Sinha, and Michael P. Wellman. 2018. SoK: Security and privacy in machine learning. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P'18)*. IEEE, 399–414.
[52] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'17)*. IEEE, 3–18.
[53] Brendan Avent, Javier González, Tom Diethe, Andrei Paleyes, and Borja Balle. 2020. Automatic discovery of privacy–Utility Pareto fronts. *Proc. Privac. Enhanc. Technol.* 2020, 4 (2020), 5–23.
[54] Cynthia Dwork. 2006. Differential privacy. In *Proceedings of the International Colloquium on Automata, Languages, and Programming*. Springer, 1–12.
[55] Craig Gentry. 2009. Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*. 169–178.
[56] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. 2015. Federated optimization: Distributed optimization beyond the datacenter. In *Proceedings of the OPT Workshop on Optimization for Machine Learning, NIPS*.
[57] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.
[58] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.* 18, 1 (2017), 6765–6816.
[59] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 2951–2959.
[60] Bobak Shahriari, Alexandre Bouchard-Cote, and Nando Freitas. 2016. Unbounded Bayesian optimization via regularization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (Proceedings of*

Machine Learning Research), Arthur Gretton and Christian C. Robert (Eds.), Vol. 51. PMLR, 1168–1176. Retrieved from http://proceedings.mlr.press/v51/shahriari16.html.

[61] Diana Marculescu, Dimitrios Stamoulis, and Ermao Cai. 2018. Hardware-aware machine learning: Modeling and optimization. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'18)*. Association for Computing Machinery, New York, NY. http://dx.doi.org/10.1145/3240765.3243479

[62] Lucas Bernardi, Themistoklis Mavridis, and Pablo Estevez. 2019. 150 successful machine learning models: 6 lessons learned at Booking.com. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1743–1751.

[63] Danilo Sato, Arif Wider, and Christoph Windheuser. 2019. Continuous Delivery for Machine Learning. Retrieved from https://martinfowler.com/articles/cd4ml.html.

[64] Ram Ananth, Seph Mard, and Peter Simon. 2019. Opening the "Black Box": The path to deployment of AI models in banking, white paper. *DataRobot* (2019). https://www.datarobot.com/resources/opening-the-black-box/. Accessed from 13/06/2022.

[65] Prudential Regulation Authority. 2018. Model risk management principles for stress testing. https://www.bankofengland.co.uk/prudential-regulation/publication/2018/model-risk-management-principles-for-stress-testing-ss. Accessed from 13/06/2022.

[66] ECB TRIM Guide. 2017. Guide for the targeted review of internal models (TRIM). *European Central Bank*. https://www.bankingsupervision.europa.eu/ecb/pub/pdf/trim_guide.en.pdf. Accessed from 13/06/2022.

[67] Timothy M. Hackett, Sven G. Bilén, Paulo Victor Rodrigues Ferreira, Alexander M. Wyglinski, Richard C. Reinhart, and Dale J. Mortensen. 2018. Implementation and on-orbit testing results of a space communications cognitive engine. *IEEE Trans. Cog. Commun. Netw.* 4, 4 (2018), 825–842.

[68] Eric Breck, Marty Zinkevich, Neoklis Polyzotis, Steven Whang, and Sudip Roy. 2019. Data validation for machine learning. In *Proceedings of the Conference on Systems and Machine Learning*. Retrieved from https://mlsys.org/Conferences/2019/doc/2019/167.pdf.

[69] Yingnong Dang, Qingwei Lin, and Peng Huang. 2019. AIOps: Real-world challenges and research innovations. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering*. IEEE, 4–5.

[70] Andrew Zhai, Hao-Yu Wu, Eric Tzeng, Dong Huk Park, and Charles Rosenberg. 2019. Learning a unified embedding for visual search at Pinterest. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2412–2420.

[71] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in machine learning systems. In *Proceedings of the Conference on Advances in Neural Information Processing Systems*. 2503–2511.

[72] Janis Klaise, Arnaud Van Looveren, Clive Cox, Giovanni Vacanti, and Alexandru Coca. 2020. Monitoring and explainability of models in production. In *Proceedings of the Workshop on Challenges in Deploying and Monitoring Machine Learning Systems*.

[73] Klaus Ackermann, Joe Walsh, Adolfo De Unánue, Hareem Naveed, Andrea Navarrete Rivera, Sun-Joo Lee, Jason Bennett, Michael Defoe, Crystal Cody, Lauren Haynes, et al. 2018. Deploying machine learning models for public policy: A framework. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 15–22.

[74] Tom Diethe, Tom Borchert, Eno Thereska, Borja Balle, and Neil Lawrence. 2018. Continual learning in practice. In *Proceedings of the Workshop on Continual Learning, NeurIPS*.

[75] Joaquin Quiñonero-Candela, Masashi Sugiyama, Neil D. Lawrence, and Anton Schwaighofer. 2009. *Dataset Shift in Machine Learning*. The MIT Press.

[76] Syed Muslim Jameel, Manzoor Hashmani, Hitham Alhussian, Mobashar Rehman, and Arif Budiman. 2020. A critical review on adverse effects of concept drift over machine learning classification models. *Int. J. Adv. Comput. Sci. Applic.* 11 (1 2020). http://dx.doi.org/10.14569/IJACSA.2020.0110127

[77] Bilge Celik and Joaquin Vanschoren. 2021. Adaptation strategies for automated machine learning on evolving data. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 9 (2021), 3067–3078.

[78] Andrés R. Masegosa, Ana M. Martínez, Darío Ramos-López, Helge Langseth, Thomas D. Nielsen, and Antonio Salmerón. 2020. Analyzing concept drift: A case study in the financial sector. *Intell. Data Anal.* 24, 3 (2020), 665–688.

[79] Daniel Langenkämper, Robin van Kevelaer, Autun Purser, and Tim W. Nattkemper. 2020. Gear-induced concept drift in marine images and its effect on deep learning classification. *Front. Marine Sci.* 7 (2020), 506.

[80] Jan Zenisek, Florian Holzinger, and Michael Affenzeller. 2019. Machine learning based concept drift detection for predictive maintenance. *Comput. Industr. Eng.* 137 (2019), 106031.

[81] Jeffrey C. Schlimmer and Richard H. Granger. 1986. Incremental learning from noisy data. *Mach. Learn.* 1, 3 (1986), 317–354.

[82] Dennis Soemers, Tim Brys, Kurt Driessens, Mark Winands, and Ann Nowé. 2018. Adapting to concept drift in credit card transaction data streams using contextual bandits and decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[83] Zijian Sun, Jian Tang, Junfei Qiao, and Chengyu Cui. 2020. Review of concept drift detection method for industrial process modeling. In *Proceedings of the 39th Chinese Control Conference (CCC'20)*. IEEE, 5754–5759.

[84] Arif Wider and Christian Deger. 2017. Getting Smart: Applying Continuous Delivery to Data Science to Drive Car Sales. Retrieved from https://www.thoughtworks.com/insights/blog/getting-smart-applying-continuous-delivery-data-science-drive-car-sales.

[85] Gagan Bansal, Besmira Nushi, Ece Kamar, Daniel S. Weld, Walter S. Lasecki, and Eric Horvitz. 2019. Updates in human-AI teams: Understanding and addressing the performance/compatibility tradeoff. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2429–2437.

[86] Megha Srivastava, Besmira Nushi, Ece Kamar, Shital Shah, and Eric Horvitz. 2020. An empirical analysis of backward compatibility in machine learning systems. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3272–3280.

[87] David Leslie. 2019. Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector. *Available at* Retrieved from: SSRN 3403301 (2019).

[88] Cathy O'Neil. 2016. *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Broadway Books.

[89] Marcelo O. R. Prates, Pedro H. Avelar, and Luís C. Lamb. 2020. Assessing gender bias in machine translation: A case study with Google Translate. *Neural Comput. Applic.* 32, 10 (2020), 6363–6381.

[90] Robert Soden, Dennis Wagenaar, Dave Luo, and Annegien Tijssen. 2019. Taking Ethics, Fairness, and Bias Seriously in Machine Learning for Disaster Risk Management. arXiv:cs.CY/1912.05538.

[91] Sathappan Muthiah, Patrick Butler, Rupinder Paul Khandpur, Parang Saraf, Nathan Self, Alla Rozovskaya, Liang Zhao, Jose Cadena, Chang-Tien Lu, Anil Vullikanti, et al. 2016. Embers at 4 years: Experiences operating an open source indicators forecasting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 205–214.

[92] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Proceedings of the Conference on Fairness, Accountability and Transparency*. PMLR, 77–91.

[93] Solon Barocas, Moritz Hardt, and Arvind Narayanan. 2017. Fairness in machine learning. *NIPS Tutor.* 1 (2017).

[94] Nantheera Anantrasirichai and David Bull. 2021. Artificial intelligence in the creative industries: A review. *Artif. Intell. Rev.* (2021), 1–68.

[95] Yisroel Mirsky and Wenke Lee. 2021. The creation and detection of deepfakes: A survey. *ACM Comput. Surv.* 54, 1 (2021), 1–41.

[96] John Mark Michael Rumbold and Barbara Pierscionek. 2017. The effect of the general data protection regulation on medical research. *J. Med. Internet Res.* 19, 2 (2017), e47.

[97] Syed Mohamed Aljunid, Samrit Srithamrongsawat, Wen Chen, Seung Jin Bae, Raoh-Fang Pwu, Shunya Ikeda, and Ling Xu. 2012. Health-care data collecting, sharing, and using in Thailand, China mainland, South Korea, Taiwan, Japan, and Malaysia. *Val. Health* 15, 1 (2012), S132–S138.

[98] Changhee Han, Leonardo Rundo, Kohei Murao, Takafumi Nemoto, and Hideki Nakayama. 2020. Bridging the gap between AI and healthcare sides: Towards developing clinically relevant AI-powered diagnosis systems. In *Proceedings of the IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 320–333.

[99] Gary E. Marchant. 2011. The growing gap between emerging technologies and the law. In *The Growing Gap between Emerging Technologies and Legal-ethical Oversight*. Springer, 19–33.

[100] D. Malan et al. 2016. Values and the fourth industrial revolution: Connecting the dots between value, values, profit and purpose. In *Proceedings of the World Economic Forum*.

[101] Daniel Malan. 2018. The law can't keep up with new tech. Here's how to close the gap. In *Proceedings of the World Economic Forum*.

[102] Sandra Wachter, Brent Mittelstadt, and Luciano Floridi. 2017. Why a right to explanation of automated decision-making does not exist in the general data protection regulation. *Int. Data Privac. Law* 7, 2 (2017), 76–99.

[103] Timo Minssen, Sara Gerke, Mateo Aboy, Nicholson Price, and Glenn Cohen. 2020. Regulatory responses to medical machine learning. *J. Law Biosci.* (2020).

[104] Keri Stephens. 2021. FDA releases artificial intelligence/machine learning action plan. *AXIS Imag. News* (2021). https://www.fda.gov/news-events/press-announcements/fda-releases-artificial-intelligencemachine-learning-action-plan. Accessed from 13/06/2022.

[105] Mustafa Suleyman and Dominic King. 2017. The Information Commissioner, the Royal Free, and what we've learned. (2017). https://www.deepmind.com/blog/the-information-commissioner-the-royal-free-and-what-weve-learned. Accessed from 13/06/2022.

[106] M.-C. Laï, M. Brian, and M.-F. Mamzer. 2020. Perceptions of artificial intelligence in healthcare: Findings from a qualitative survey study among actors in France. *J. Translat. Med.* 18, 1 (2020), 1–13.
[107] Ramprakash Ramamoorthy, P. Satya Madhuri, and Malini Christina Raj. 2019. AI from labs to production—challenges and learnings. USENIX Association. https://www.usenix.org/conference/opml19/presentation/madhuri. Accessed from 13/06/2022.
[108] Mark Sendak, Madeleine Clare Elish, Michael Gao, Joseph Futoma, William Ratliff, Marshall Nichols, Armando Bedoya, Suresh Balu, and Cara O'Brien. 2020. "The Human Body is a Black Box": Supporting clinical decision-making with deep learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT'20)*. Association for Computing Machinery, New York, NY, 99–109. http://dx.doi.org/10.1145/3351095.3372827
[109] Information Commissioner's Office. 2019. Project ExplAIn interim report. Retrieved from https://ico.org.uk/about-the-ico/research-and-reports/project-explain-interim-report/.
[110] Daniel Mutembesa, Christopher Omongo, and Ernest Mwebaze. 2018. Crowdsourcing real-time viral disease and pest information: A case of nation-wide cassava disease surveillance in a developing country. In *Proceedings of the 6th AAAI Conference on Human Computation and Crowdsourcing*.
[111] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* 1, 5 (2019), 206–215.
[112] Wei Wang, Christopher Lesner, Alexander Ran, Marko Rukonic, Jason Xue, and Eric Shiu. 2020. Using small business banking data for explainable credit risk scoring. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 13396–13401.
[113] Dakuo Wang, Liuping Wang, Zhan Zhang, Ding Wang, Haiyi Zhu, Yvonne Gao, Xiangmin Fan, and Feng Tian. 2021. "Brilliant AI Doctor" in rural clinics: Challenges in AI-powered clinical decision support system deployment. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. 1–18.
[114] Ram Shankar Siva Kumar, David O. Brien, Kendra Albert, Salomé Viljöen, and Jeffrey Snover. 2019. Failure modes in machine learning systems. *arXiv preprint arXiv:1911.11034* (2019).
[115] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recog.* 84 (2018), 317–331.
[116] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *Proceedings of the International Conference on Learning Representations*.
[117] Ram Shankar Siva Kumar, Magnus Nystrom, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. 2020. Adversarial machine learning-industry perspectives. Retrieved from *SSRN 3532474* (2020).
[118] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. Doug Tygar, and Kai Xia. 2008. Exploiting machine learning to subvert your spam filter. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*. 1–9.
[119] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. 2018. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'18)*. IEEE, 19–35.
[120] Oscar Schwartz. 2019. In 2016 Microsoft's racist chatbot revealed the dangers of online conversation. *IEEE Spectr.* 11 (2019).
[121] Florian Tramèr, Fan Zhang, Ari Juels, Michael K. Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction APIs. In *Proceedings of the 25th USENIX Security Symposium (USENIX Security'16)*. 601–618.
[122] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. 1322–1333.
[123] Michael Veale, Reuben Binns, and Lilian Edwards. 2018. Algorithms that remember: Model inversion attacks and data protection law. *Philos. Trans. Roy. Societ. A: Math., Phys. Eng. Sci.* 376, 2133 (2018), 20180083.
[124] Chip Huyen. 2020. What I learned from looking at 200 machine learning tools. Retrieved from https://huyenchip.com/2020/06/22/mlops.html.
[125] Kumar Venkateswar. 2019. Using Amazon SageMaker to operationalize machine learning. USENIX Association. https://www.usenix.org/conference/opml19/presentation/venkateswar. Accessed from 13/06/2022.
[126] AML Team. 2016. AzureML: Anatomy of a machine learning service. In *Proceedings of the Conference on Predictive APIs and Apps*. 1–13.
[127] Jeremy Hermann and Mike Del Balso. 2017. Meet Michelangelo: Uber's machine learning platform. Retrieved from https://eng.uber.com/michelangelo.
[128] Denis Baylor, Kevin Haas, Konstantinos Katsiapis, Sammy Leong, Rose Liu, Clemens Menwald, Hui Miao, Neoklis Polyzotis, Mitchell Trott, and Martin Zinkevich. 2019. Continuous training for production ML in the TensorFlow extended (TFX) platform. In *Proceedings of the USENIX Conference on Operational Machine Learning (OpML'19)*. 51–53.

[129] Matei Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, et al. 2018. Accelerating the machine learning lifecycle with MLflow. *IEEE Data Eng. Bull.* 41, 4 (2018), 39–45.

[130] Sebastian Schelter, Tammo Rukat, and Felix Biessmann. 2021. JENGA-A framework to study the impact of data errors on the predictions of machine learning models. (2021). https://www.amazon.science/publications/jenga-a-framework-to-study-the-impact-of-data-errors-on-the-predictions-of-machine-learning-models. Accessed from 13/06/2022.

[131] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Association for Computational Linguistics (ACL'20)*.

[132] Nick Hynes, D. Sculley, and Michael Terry. 2017. The Data Linter: Lightweight, automated sanity checking for ML data sets. In *Proceedings of the NIPS MLSys Workshop*.

[133] Stephen H. Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. 2019. Snorkel DryBell: A case study in deploying weak supervision at industrial scale. In *Proceedings of the International Conference on Management of Data*. 362–375.

[134] Paroma Varma and Christopher Ré. 2018. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment* 12, 3 (2018). NIH Public Access.

[135] Curtis Northcutt, Lu Jiang, and Isaac Chuang. 2021. Confident learning: Estimating uncertainty in dataset labels. *J. Artif. Intell. Res.* 70 (2021), 1373–1411.

[136] Yaliang Li, Zhen Wang, Bolin Ding, and Ce Zhang. 2021. AutoML: A perspective where industry meets academy. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4048–4049.

[137] Haifeng Jin, Qingquan Song, and Xia Hu. 2019. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1946–1956.

[138] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf.

[139] Randal S. Olson and Jason H. Moore. 2016. TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Proceedings of the Workshop on Automatic Machine Learning*. PMLR, 66–74.

[140] Ashkan Ebadi, Yvan Gauthier, Stéphane Tremblay, and Patrick Paul. 2019. How can automated machine learning help business data science teams? In *Proceedings of the 18th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 1186–1191.

[141] Livia Faes, Siegfried K. Wagner, Dun Jack Fu, Xiaoxuan Liu, Edward Korot, Joseph R. Ledsam, Trevor Back, Reena Chopra, Nikolas Pontikos, Christoph Kern, et al. 2019. Automated deep learning design for medical image classification by health-care professionals with no coding experience: A feasibility study. *Lancet Digit. Health* 1, 5 (2019), e232–e242.

[142] Stephan Rabanser, Stephan Günnemann, and Zachary Chase Lipton. 2019. Failing loudly: An empirical study of methods for detecting dataset shift. In *Proceedings of the Neural Information Processing Systems Foundation Conference*.

[143] Arnaud Van Looveren, Janis Klaise, Giovanni Vacanti, Oliver Cobb, Ashley Scillitoe, and Robert Samoilescu. 2022. Alibi Detect: Algorithms for outlier, adversarial and drift detection. (3 2022). Retrieved from https://github.com/SeldonIO/alibi-detect. Accessed from 13/06/2022.

[144] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Comput. Surv.* 46, 4 (2014), 1–37.

[145] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1126–1135.

[146] Peilin Zhao, Steven C. H. Hoi, Jialei Wang, and Bin Li. 2014. Online transfer learning. *Artif. Intell.* 216 (2014), 76–102.

[147] Ge Xie, Yu Sun, Minlong Lin, and Ke Tang. 2017. A selective transfer learning method for concept drift adaptation. In *Proceedings of the International Symposium on Neural Networks*. Springer, 353–361.

[148] A. Zujus. 2018. AI project development–how project managers should prepare. *Tratto il giorno Agosto* 31 (2018), 2019.

[149] Amine Barrak, Ellis E. Eghan, and Bram Adams. 2021. On the co-evolution of ML pipelines and source code–empirical study of DVC projects. In *Proceedings of the IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER'21)*. IEEE, 422–433.

[150] Neil D. Lawrence. 2017. Data readiness levels. *arXiv preprint arXiv:1705.02245* (2017).

[151] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. 2021. Datasheets for datasets. *Commun. ACM* 64, 12 (2021), 86–92.

[152] Ruslan Kuprieiev, Saugat Pachhai, Dmitry Petrov, Paweł Redzyński, Casper da Costa-Luis, Peter Rowlands, Alexander Schepanovski, Ivan Shcheklein, Batuhan Taskaya, Jorge Orpinel, Gao, Fábio Santos, David de la Iglesia Castro, Aman Sharma, Zhanibek, Dani Hodovic, Nikita Kodenko, Andrew Grigorev, Earl, Nabanita Dash, George Vyshnya,

maykulkarni, Max Hora, Vera, Sanidhya Mangal, Wojciech Baranowski, Clemens Wolff, and Kurian Benoy. 2022. DVC: Data Version Control - Git for Data & Models. (Feb. 2022). http://dx.doi.org/10.5281/zenodo.6195393

[153] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. 220–229.

[154] Neil Lawrence. 2019. Modern Data-oriented Programming. Retrieved from http://inverseprobability.com/talks/notes/modern-data-oriented-programming.html.

[155] Tom Borchert. 2020. Milan: An Evolution of Data-oriented Programming. Retrieved from https://tborchertblog.wordpress.com/2020/02/13/28/.

[156] Andrei Paleyes, Christian Cabrera, and Neil D. Lawrence. 2021. Towards better data discovery and collection with flow-based programming. In *Proceedings of the Data-centric AI Workshop, NeurIPS*.

[157] Alexander Lavin, Ciarán M. Gilligan-Lee, Alessya Visnjic, Siddha Ganju, Dava Newman, Sujoy Ganguly, Danny Lange, Atılım Güneş Baydin, Amit Sharma, Adam Gibson, Yarin Gal, Eric P. Xing, Chris Mattmann, and James Parr. 2021. Technology Readiness Levels for Machine Learning Systems. arXiv:cs.LG/2101.03989.

[158] Martin Zinkevich. 2017. Rules of machine learning: Best practices for ML engineering. Retrieved from https://developers.google.com/machine-learning/guides/rules-of-ml.

[159] Verein Deutscher Ingenieure (VDI). 2019. Einsatz von Big Data in produzierenden Industrien. Retrieved from https://www.vdi.de/news/detail/einsatz-von-big-data-in-produzierenden-industrien.

Received 3 March 2021; revised 8 April 2022; accepted 23 April 2022