Challenges in Deploying Machine Learning: a Survey of Case Studies

Andrei Paleyes
Department of Computer Science
University of Cambridge
ap2169@cam.ac.uk

Raoul-Gabriel Urma
Cambridge Spark
raoul@cambridgespark.com

Neil D. Lawrence
Department of Computer Science
University of Cambridge
ndl21@cam.ac.uk

Abstract

In recent years, machine learning has received increased interest both as an academic research field and as a solution for 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. Our survey shows that practitioners face challenges at each stage of the deployment. The goal of this paper is to layout a research agenda to explore approaches addressing these challenges.

1 Introduction

Machine learning (ML) has evolved from purely an area of academic research to an applied field. In fact, according to a recent global survey conducted by McKinsey, machine learning is increasingly adopted in standard business processes with nearly 25 percent 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 academic setting and what is required by a real world system. Certain bottlenecks and invalidated assumptions should always be expected in the course of that process. As more solutions are developed and deployed, practitioners sometimes report their experience in various forms, including publications and blog posts. In this study, we undertake a survey of these reports to capture the current challenges in deploying machine learning in production[1]. First, we provide an overview of the machine learning deployment workflow. Second, we review use 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, end users' trust and security. Finally, we conclude with a brief discussion of potential solutions to these issues and further work.

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 solution deployed for a business problem outside of "Big Tech" companies in the information technology industry. So

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

NeurIPS 2020 Workshop: ML Retrospectives, Surveys & Meta-Analyses (ML-RSA).

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?"

A popular 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 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] that surveyed 2,473 organizations and their experience with ML found that a significant portion of attempted 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]. A 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 literature review and interviews with industry partners. In contrast with that paper, which concentrates on experience of information technology sector, we aim at covering case studies from a wide variety of industries. 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 last five years are considered, with only few exceptions to this rule. We also refer other types of 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 paper.

To main contribution of our paper 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 to 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 pipeline description. 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:

- **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 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 referring to concrete parts of the deployment pipeline.

For the remainder of this paper we discuss common issues practitioners face at each each stage and granular constituent steps. We also discuss cross-cutting aspects that can affect every stage of the deployment pipeline. Table 1 provides a summary of the issues and concerns we discuss. We do not claim that this list is exhaustive, nonetheless by providing illustrative examples for each step of the workflow we show how troublesome the whole deployment experience is today.

3 Data Management

Data is an integral part of any machine learning solution. 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 previously reported by Polyzotis et al. [15]. 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.

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. 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 [16], at Twitter this situation often happened due to the fact that the same entity (e.g. a Twitter user) is 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 "single responsibility principle" [17], 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 impossible to keep track of what data related to the entity is being stored by which service, and in which form. Besides, 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 in order to build a dataset one needs to generate synthetic service API calls. Such a 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 realistically achieve.

3.2 Data preprocessing

The preprocessing step normally involves a range of data cleaning activities: 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. [18], Patil and Kulkarni [19], Ridzuan et al. [20]. An approach towards classifying readiness of data for ML tasks is given by Lawrence [21].

A lesser known but also 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 which 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 on its own right. An example of this is what developers of Firebird faced [22]. Firebird is an

Table 1: All considerations, issues and concerns explored in this study. Each is assigned to the stage and step of the deployment workflow where it is commonly encountered.

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
	Hyper-parameter selection	Resource-heavy techniques
		Hardware-aware optimization
Model verification	Requirement encoding	Performance metrics
		Business driven metrics
	Formal verification	Regulatory frameworks
	Test-based verification	Simulation-based 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	Country-level regulations
		Focus on technical solution only
		Aggravation of biases
		Authorship
		Decision making
	End users' trust	Involvement of end users
		User experience
		Explainability score
	Security	Data poisoning
		Model stealing
		Model inversion

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 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 building location is the best way to identify that building, each dataset contained spatial data specifying building's address. Spatial information can be presented in different formats, and sometimes contain minor differences such as different spellings. All this needed to be cleaned and corrected, and turned out to be a massive effort that consumed a lot of time.

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. 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.

Labels 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 in order to be labeled. This problem is described by Pacheco et al. [23], 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. Even in this case it has been shown that existing tools for label assignment introduce errors into collected ML datasets, going as high as almost 100% for certain applications. 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 [24]. 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 or weaker annotations, however these approaches are by their definition a trade off that provides imprecise labels, which ultimately leads to a severe loss in quality of the model. Such losses are unacceptable in healthcare industry, where even the smallest deviation can cause catastrophic results (this is known as The Final Percent challenge).

Lack of access to high-variance data can be among the main challenges one faces when deploying machine learning solution from lab environment to real world. Dulac-Arnold et al. [25] explain that this is the case for Reinforcement Leaning (RL). 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. While this approach ensures safety, it means that agent is not trained to recognize an unsafe situation and make right decision in it. A practical example of this issue is the goal specification for autonomous vehicles [26].

3.4 Data analysis

Data needs to be analyzed in order to uncover potential biases or unexpected distributions 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 [27]. 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 efficient execution of these data mining tasks. The need for such tools becomes apparent considering that, according to the survey conducted

by Microsoft [28], data scientists think about data issues as the main reason to doubt 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. All modern research in machine learning methods contributes towards better selection and variety of models and approaches that can be employed at this stage. As an illustration of the scale of the field's growth, the number of submissions to NeurIPS, primary conference on ML methods, has quadrupled in six years, going from 1678 submissions in 2014 to 6743 in 2019 [29]. Nevertheless, there is 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 often decided by one key characteristic of a model: complexity. Despite areas such as deep learning and reinforcement learning gaining increasing levels of popularity with the research community, in practice simpler models are often chosen as we explain below. Such model include shallow network architectures, simple PCA-base approaches, 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 accelerates 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. [30]. 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 production setting, while providing reasonably good performance[2]. 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.

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. [31]. They worked on deploying ML models to a range of scientific instruments onboard Europa Clipper spacecraft. Spacecraft design is always a trade-off 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 on 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 highly dimensional mobile network data [32].

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), which can be considered a fairly basic ML algorithm, are widely used in practice. For example, Hansson et al. [33] describe several cases in manufacturing that adopt DT due to its high interpretability.

Banking is yet another example of an industry where DT finds extensive use. As an illustrative example, it is used by Keramati et al. [34] where the primary goal of the ML application is to predict customer churn by understanding if-then rules. While it is easy to imagine more complicated

[2]We discuss more benefits of setting up the automated deployment pipeline in Section 6.3.

models learning the eventual input-output relationship for this specific problem, interpretability is 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 a large amount of previously acquired data. This notion is exemplified by the field of unmanned aerial vehicles (UAV) [35]. Image sensors are commonplace in UAVs due to their low cost, low weight, and low power consumption. Consequently, processing images acquired from sensors is the main way of exploiting excellent capabilities in processing and presentation of raw data that DL offers. But computational resource demands still remain the main blocker for deploying DL as an online processing instrument on board of UAVs.

4.2 Training

One of the biggest concern with model training is the economic cost associated with carrying the training stage due to the computational resources required. This is certainly true in the field of natural language processing (NLP), as illustrated by Sharir et al. [36]. 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 [37], and found out that depending on the chosen model size full training procedure can cost anywhere between $50k and $1.6m, which is unaffordable for most research institutions and even companies. The authors observe that training dataset size, number of model parameters and number of operations utilized 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 nearest future [38].

A related concern is raised by Strubell et al. [39] regarding the impact the training of ML models has on the environment. By consuming more and more computational resources, ML models training is driving up the energy consumption and greenhouse gas emissions. According to the estimates provided in the paper, one full training cycle utilizing neural architecture search emits the 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 community should give higher priority to computationally efficient hardware and algorithms.

4.3 Hyper-parameter selection

In addition to parameters that are learned during the training process, many ML models also define several hyper-parameters. Hyper-parameter optimization (HPO) is the process of choosing the optimal set of these hyper-parameters. Most HPO techniques involve multiple training cycles of the ML model. Besides, the size of HPO task grows exponentially with each new hyper-parameter, because it adds a new dimension to the search space. As discussed by Yang and Shami [40], these considerations make HPO techniques very expensive and resource-heavy in practice, especially for applications of deep learning. Even approaches like Hyperband [41] or Bayesian optimization [42], that are specifically designed to minimize the number of training cycles needed, are still not able to deal with certain problems due to the complexity of the models or the size of the datasets.

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. [43] in the context of hardware-aware ML. In order to deploy models to embedded and mobile devices, one needs to be well aware of energy and memory constraints imposed by such devices. This creates a need for customized hardware-aware optimization techniques that co-optimize for the accuracy of the model and the hardware efficiency.

5 Model Verification

The goal of the model verification stage is multifaceted, because an ML model 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 [44]. Therefore more specific metrics need to be defined and measured, such as KPIs and other business driven measures. In the case of Booking.com such metrics included conversion, customer service tickets or cancellations. Cross-disciplinary effort is needed to even define such metrics, as understanding from modeling, engineering and business angles is required. Once defined, these metrics are used for monitoring of 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. Essentially, performance metrics should reflect audience priorities. For instance Sato et al. [45] recommend validating models for bias and fairness, while in the case described by Wagstaff et al. [31] controlling for consumption of spacecraft resources is crucial.

5.2 Formal Verification

The formal verification step verifies that the model functionality follows the requirements defined within the scope of the project. 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.

An example of where ML solutions have to adhere to regulations is the banking industry [46]. 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 has been published by the UK's Prudential Regulation Authority [47] and European Central Bank [48]. These guidelines require 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 in order to understand 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, which we discuss in greater details in Section 7.1.

5.3 Test-based Verification

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

In an ideal scenario 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 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 [26]. 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 [25], where use of simulations is a de-facto standard for training agents.

In addition, the dataset itself also needs to be constantly validated to ensure data errors do not creep into the pipeline and do not affect the overall quality. Breck et al. [49] argue that one of the most common scenarios when issues in data can go unnoticed is the setup where data generation is decoupled from the ML pipeline. There could be multiple reasons for such issues to appear, including bugs in code, feedback loops, changes in data dependencies. Data errors can propagate and

manifest themselves at different stages of the pipeline, therefore it is imperative to catch them early by including data validation routines into 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 is also a number of challenges unique to productionizing machine learning. This is discussed in detail by Dang et al. [50] which uses the term AIOps 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 which makes supervised learning approaches inapplicable[3] 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 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 the approach Pinterest took towards learning image embeddings [51]. There are three models used in Pinterest internally which use similar embeddings, and initially they were maintained completely separately, in order 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 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. [52]. Most of them are considered anti-patterns in engineering, but are currently widespread in machine learning software. Some of these issues, such as abstraction boundaries erosion and correction cascades, are caused by the fact that ML is used in cases where the software has to take 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, which is caused by the fact that ML systems, besides all configurations a regular software system may require, 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 surface there seems to be a clear separation of responsibilities: researchers produce the model while engineers build 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 loop researchers into the whole development journey, making sure they own the product code base 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].

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

6.2 Monitoring

Monitoring is one of the issues associated with maintaining machine learning systems as reported by Sculley et al. [52]. The community is in the early stages of understanding what are the key metrics of data and models to monitor and how to alarm on them. Monitoring of evolving input data, prediction bias and overall performance of ML models is an open problem. Another maintenance issue highlighted by this paper 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 feedback loop where the input to the model is being adjusted to influence its behavior. This can be done intentionally, as well as happen inadvertently which is a unique challenge when running live ML systems.

Klaise et al. [53] 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 also 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. [54]. This paper 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 in order to maintain good model performance. For instance, the data integrity check meant verifying updates of a certain input table and checksums on historical records, 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 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 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 in order to make sure it always reflects the most recent trends in data and the environment. There are multiple techniques for adapting models to a new data, including scheduled regular retraining and continual learning [55]. Nevertheless in 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. 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. Undetected, this phenomenon can have major adverse effects on model performance, as is shown by Jameel et al. [56] for classification problems or by Celik and Vanschoren [57] in 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 it could have provided additional insights into the ongoing crisis, as explained by Masegosa et al. [58]. Changes in data can also be caused by inability to avoid fluctuations in the data collection procedure, as described in paper Langenkämper et al. [59] which studies the effects of slight changes in marine images capturing gear and location on deep learning models' performance. Data shifts can have noticeable consequences even when occurring at microscopic scale, as Zenisek et al. [60] show in their research on predictive maintenance for wear and tear of industrial machinery. Even though concept drift has been known for decades [61], these examples show that it remains a critical problem for applications of ML today.

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

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 axis: 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. [45]. 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 benefits that a full CD pipeline can bring to the real-life ML solution can be found in Wider and Deger [62].

7 Cross-cutting aspects

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

7.1 Ethics

Ethical considerations should always inform data collection activities. As stated in the report on ethical AI produced by the Alan Turing Institute [63], "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, complications may come up due to a variety of reasons: breach of governmental regulations, unjustifiable outcomes, aggravation of existing issues, and more [63].

Various countries have produced regulations to protect personal data rights. The more sensitive is the information collected from the individual, the severer are the regulations governing its use. And of course industry that deals with some of the most sensitive information is healthcare. According to Han et al. [64], many countries have strict laws in place to protect data of patients, which makes adoption of ML in healthcare particularly difficult. Examples of such regulations include the General Data Protection Regulation in European Union [65] and ethical screening laws in a range of Asian countries [66]. 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 amount of reviews, software updates and cycles of data collection/annotation that is required makes it exceptionally hard to keep up with technical advances in ML, as Han et al. [64] explain following their experience deploying ML solutions in healthcare sector in Japan.

Companies should not be focusing solely on the technological side of their solutions, as DeepMind and Royal Free NHS Foundation Trust have discovered while working on Streams, an application for automatic review of test results for serious conditions [67]. 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 patient and public engagement strategy in order to ensure data is being used ethically.

Since ML models use previously seen data to make decisions, they can worsen issues that already exist in data. This effect in the field of criminal justice is discussed in detail by O'Neil [68]. Models that calculate person's criminal "risk score" are often marketed as a way to remove human bias. Nevertheless, they use seemingly neutral demographic information that often ends up serving as a proxy. As a result, people are disadvantaged on the basis of race or income.

Likewise, Soden et al. [69] mention aggravation of social inequalities through use of biased training datasets as one of the main concerns in applying ML to Disaster Risk Management (DRM) field. Furthermore, it is argued that ML causes privacy and security concerns in Fragility, Conflict and Violence settings through combination of previously distinct datasets. Reducing role of both experts and general public is also seen as an ethical issue by DRM professionals. Some of these issues are studied by the branch of machine learning known as Fairness in ML [70]. We discuss related cross-cutting security concerns in Section 7.3.

An interesting ethical aspect arises in usage of ML in the field of creative arts, discussed by Anantrasirichai and Bull [71]. 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 a special attention. Closely related with this question is the growing concern of fake content being generated with ML, which can be easily used for the wrong purposes [72].

Another facet of ethical issues encountered by machine leaning is the data based decision making. As machine learning tools become utilized in critical decision making processes ethical concerns with their usage grow. Illustrative note is made by Muthiah et al. [73]. Their system of 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 misunderstanding of its role in society, or intentionally.

7.2 End users' trust

ML is often met cautiously by the end users [74], [3], [75]. On their own accord models provide minimal explanations, which makes it difficult to persuade end users in their utility [64]. In order to convince users to trust ML based solutions, 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, getting these people 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. One example is the project called Sepsis Watch [76]. In this project the goal was to build a model that estimates 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 eventual success of Sepsis Watch. To overcome this skepticism, the team prioritized building trust, by:

- Building strong communication channels;
- Sharing with stakeholders progress towards developing the goal instead of showing technical advances;
- Establishing mechanisms for public and external accountability;
- Engaging both front-line clinicians and enterprise-level decision makers at early stages of the project.

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. While it may sound controversial, in fact it aligns with conclusions made by "Project explAIn" which found that relative importance of explanations of AI decisions varies by context [77].

Similar argument is made by Soden et al. [69], who explore an impact ML has on disaster risk management (DRM). Due to growing complexity of the ML solutions deployed, it is becoming increasingly hard for 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 development of these solutions as transparent as possible, by taking into account voice of residents in the areas portrayed by models as "at risk" and relying on open software and data whenever possible.

A reasonable approach towards building trustworthy products is investing time in specialised user interfaces with tailored user experience. Developers of Firebird [22], a system that helps identify priority targets for fire inspection in the city of Atlanta, USA, found that the best way to introduce 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 dept) found most useful and clear. Similar experience is reported by Ackermann et al. [54].

Authors of EMBERS [73], 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 specific area or specific hypothesis. To improve the user experience and thus increase their confidence in the product, 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 [77].

Budd et al. [24] show that interface design directly impacts 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 well designed user interface. The authors conclude that end user interface plays a large part in overall success of the annotation applications.

Finally, the solutions based on models whose decisions can be explained are preferred when the target audience has experience and an understanding of ML. 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 of the 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.

7.3 Security

Machine Learning opens up new threat vectors across the whole ML deployment workflow as described by Kumar et al. [78]. Specialised adversarial attacks for ML can occur on the model itself, the data used for training but 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 [79, 80]. Recent work from Kumar et al. found that industry practitioners are not equipped to protect, detect and respond to attacks on their ML systems [81]. In this section, we describe the three most common attacks reported in practice which affects deployed ML models: data poisoning, model stealing and model inversion. We focus specifically on adversarial machine learning and 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 in order to manipulate the produced results. 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 a 8% poisoning rate in the training set resulted in incorrect dosage for half of the patients [82].

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 it happened with Microsoft's Twitter bot Tay [83]. 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 in order to train a substitute model. This type of attack is referred to as model stealing. In a nutshell, this attack results in loss of intellectual property which could be a key business advantage for the defender. Tramèr et al. [84] 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 4013 to extract an equivalent model and in time ranging from 70s to 2088s.

A related attack is that of model inversion where the goal of the adversarial attack is recover parts of the private training set, thereby breaking its confidentialty. Fredrikson et al. have shown that they could recover training data by exploiting models that report confidence values along their predictions [85]. Veale et al. [86] 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 growths 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.

8.1 Tools and services

The market for machine learning tools and services is experiencing rapid growth [87]. As a result, tools for individual deployment problems are continuously developed and released. For some of the problems we have highlighted, making use of the right specific tool is an entirely reasonable approach.

For example, this is most likely the case for operational maintenance of ML models. 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 [88], Microsoft AzureML [89], Uber Michelangelo [90], TensorFlow TFX [91], MLflow [92] and more. A typical ML platform would include, among other features, 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 operational burden associated with maintaining the ML model in production.

Quality assurance also looks to be an area where better tools can be of much assistance. As mentioned in Section 3, data integrity plays a big part in quality control, and is an active branch of research [18]. Models themselves can also greatly benefit from development of a test suite to verify their behavior. This is normally done by trial and error, but more formal approaches are being developed, as is the case with CheckList methodology for NLP models [93].

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 which 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 [94], Snuba [95] and cleanlab [96].

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 strength and shortcomings.

8.2 Holistic approaches

Even though ML deployments require a certain amount of software development, ML projects are fundamentally different from traditional software engineering projects, and we argue they need a different approach. The main differences arise from unique activities like data discovery, dataset preparation, model training, deployment success measurement etc. Some of these activities cannot be defined precisely enough to have a reliable time estimate, some assume huge potential risks, and some make it difficult to measure the overall added value of the project. Therefore ML deployments do not lend themselves well to widespread approaches to software engineering management paradigms, and neither to common software architectural patterns.

Data Oriented Architectures (DOA, [97], [98]) is an example of an idea that suggests rethinking how things are normally approached in software development, and by doing so promises to solve quite a few 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 streaming-based architecture, 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 data flows 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. 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. [99], who propose Technology Readiness Levels for ML (TRL4ML) framework. TRL4ML describes a process of producing robust ML systems that takes into account key differences between ML and traditional software engineering.

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. Similarly, Zinkevich [100] compiled a collection of best practices for machine learning that are utilized in Google. While this cannot be viewed as a coherent paradigm towards 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.

Holistic approaches are created with ML application in mind, and therefore they have the potential to offer a significant ease of deploying ML. But it should be noted that all such approaches assume a big 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, we believe that it is far from being 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 paper, we reviewed reports from a variety of industries, which shows the ubiquity and variety of challenges with deploying ML in production. An interesting extension can 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 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 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.

As an observation that follows from the process of collecting papers to review in this survey, we note the shortage of deployment experience reports in the academic literature. Valuable knowledge obtained by industry ML practitioners goes unpublished. We would like to encourage organisations 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.

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

Acknowledgments and Disclosure of Funding

We would like to thank our reviewers for their thoughtful comments and suggestions. We are also grateful to Jessica Montgomery, Diana Robinson and Ferenc Huszar for discussions that helped shape this work.

References

[1] Arif Cam, Michael Chui, and Bryce Hall. Global AI survey: AI proves its worth, but few scale impact. *McKinsey Analytics*, 2019.

[2] Thomas H. Davenport and Rajeev Ronanki. Artificial intelligence for the real world. *Harvard business review*, 96(1):108–116, 2018.

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

[4] Michal Pěchouček and Vladimír Mařík. Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous agents and multi-agent systems*, 17(3):397–431, 2008.

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

[6] Lawrence E. Hecht. Add it up: How long does a machine learning deployment take?, 2019. Available at `https://thenewstack.io/add-it-up-how-long-does-a-machine-learning-deployment-take/`.

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

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

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

[10] Artificial intelligence and machine learning projects are obstructed by data issues. *dimensional research*, 2019. Available at `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. Explainable machine learning in deployment, 2019.

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

[13] Lucas Baier, Fabian Jöhren, and Stefan Seebacher. Challenges in the deployment and operation of machine learning in practice. 2019.

[14] Rob Ashmore, Radu Calinescu, and Colin Paterson. Assuring the machine learning lifecycle: Desiderata, methods, and challenges. *arXiv preprint arXiv:1905.04223*, 2019.

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

[16] Jimmy Lin and Dmitriy Ryaboy. Scaling big data mining infrastructure: the Twitter experience. *Acm SIGKDD Explorations Newsletter*, 14(2):6–19, 2013.

[17] Robert C. Martin. The single responsibility principle. *The principles, patterns, and practices of Agile Software Development*, pages 149–154, 2002.

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

[19] Rajashree Y Patil and RV Kulkarni. A review of data cleaning algorithms for data warehouse systems. *International Journal of Computer Science and Information Technologies*, 3(5):5212–5214, 2012.

[20] Fakhitah Ridzuan and Wan Mohd Nazmee Wan Zainon. A review on data cleansing methods for big data. *Procedia Computer Science*, 161:731–738, 2019.

[21] Neil D Lawrence. Data readiness levels. *arXiv preprint arXiv:1705.02245*, 2017.

[22] Michael Madaio, Shang-Tse Chen, Oliver L. Haimson, Wenwen Zhang, Xiang Cheng, Matthew Hinds-Aldrich, Duen Horng Chau, and Bistra Dilkina. 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*, pages 185–194, 2016.

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

[24] Samuel Budd, Emma C. Robinson, and Bernhard Kainz. A survey on active learning and human-in-the-loop deep learning for medical image analysis, 2019.

[25] Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning, 2019.

[26] Sampo Kuutti, Richard Bowden, Yaochu Jin, Phil Barber, and Saber Fallah. A survey of deep learning applications to autonomous vehicle control, 2019.

[27] Sean Kandel, Andreas Paepcke, Joseph M. Hellerstein, and Jeffrey Heer. Enterprise data analysis and visualization: An interview study. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2917–2926, 2012.

[28] Miryung Kim, Thomas Zimmermann, Robert DeLine, and Andrew Begel. Data scientists in software teams: State of the art and challenges. *IEEE Transactions on Software Engineering*, 44(11):1024–1038, 2017.

[29] Diego Charrez. NeurIPS 2019 stats, 2019. Available at `https://medium.com/@dcharrezt/neurips-2019-stats-c91346d31c8f`.

[30] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C. Turnbull, Brendan M. Collins, and et al. Applying deep learning to AirBnB search. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Jul 2019.

[31] Kiri L. Wagstaff, Gary Doran, Ashley Davies, Saadat Anwar, Srija Chakraborty, Marissa Cameron, Ingrid Daubar, and Cynthia Phillips. 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, page 2191–2201, New York, NY, USA, 2019. Association for Computing Machinery.

[32] Ursula Challita, Henrik A. Ryden, and Hugo Tullberg. When machine learning meets wireless cellular networks: Deployment, challenges, and applications, 2019.

[33] Karl Hansson, Siril Yella, Mark Dougherty, and Hasan Fleyeh. Machine learning algorithms in heavy process manufacturing. *American Journal of Intelligent Systems*, 6(1):1–13, 2016.

[34] Abbas Keramati, Hajar Ghaneei, and Seyed Mohammad Mirmohammadi. Developing a prediction model for customer churn from electronic banking services using data mining. *Financial Innovation*, 2(1):10, 2016.

[35] Adrian Carrio, Carlos Sampedro, Alejandro Rodriguez-Ramos, and Pascual Campoy. A review of deep learning methods and applications for unmanned aerial vehicles. *Journal of Sensors*, 2017, 2017.

[36] Or Sharir, Barak Peleg, and Yoav Shoham. The cost of training NLP models: A concise overview. *arXiv preprint arXiv:2004.08900*, 2020.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[38] Nathan Benaich and Ian Hogarth. State of AI report 2020. 2020. Available at `www.stateof.ai`.

[39] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243*, 2019.

[40] Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[41] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[42] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[43] Diana Marculescu, Dimitrios Stamoulis, and Ermao Cai. Hardware-aware machine learning: Modeling and optimization. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18, New York, NY, USA, 2018. Association for Computing Machinery.

[44] Lucas Bernardi, Themistoklis Mavridis, and Pablo Estevez. 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*, pages 1743–1751, 2019.

[45] Danilo Sato, Arif Wider, and Christoph Windheuser. Continuous delivery for machine learning, 2019. Available at `https://martinfowler.com/articles/cd4ml.html`.

[46] Ram Ananth, Seph Mard, and Peter Simon. Opening the "black box": The path to deployment of AI models in banking, white paper. *DataRobot and REPLY AVANTAGE*, 2019.

[47] Prudential Regulation Authority. Model risk management principles for stress testing, 2018.

[48] ECB TRIM Guide. Guide for the targeted review of internal models (TRIM). *European Central Bank*, 2017.

[49] Eric Breck, Marty Zinkevich, Neoklis Polyzotis, Steven Whang, and Sudip Roy. Data validation for machine learning. In *Proceedings of SysML*, 2019.

[50] Yingnong Dang, Qingwei Lin, and Peng Huang. AIOps: real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*, pages 4–5. IEEE, 2019.

[51] Andrew Zhai, Hao-Yu Wu, Eric Tzeng, Dong Huk Park, and Charles Rosenberg. Learning a unified embedding for visual search at Pinterest, 2019.

[52] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems*, pages 2503–2511, 2015.

[53] Janis Klaise, Arnaud Van Looveren, Clive Cox, Giovanni Vacanti, and Alexandru Coca. Monitoring and explainability of models in production. *arXiv preprint arXiv:2007.06299*, 2020.

[54] 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. Deploying machine learning models for public policy: A framework. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 15–22, 2018.

[55] Tom Diethe, Tom Borchert, Eno Thereska, Borja Balle, and Neil Lawrence. Continual learning in practice. *arXiv preprint arXiv:1903.05202*, 2019.

[56] Syed Muslim Jameel, Manzoor Hashmani, Hitham Alhussian, Mobashar Rehman, and Arif Budiman. A critical review on adverse effects of concept drift over machine learning classification models. *International Journal of Advanced Computer Science and Applications*, 11, 01 2020.

[57] Bilge Celik and Joaquin Vanschoren. Adaptation strategies for automated machine learning on evolving data. *arXiv preprint arXiv:2006.06480*, 2020.

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

[59] Daniel Langenkämper, Robin van Kevelaer, Autun Purser, and Tim W Nattkemper. Gear-induced concept drift in marine images and its effect on deep learning classification. *Frontiers in Marine Science*, 7:506, 2020.

[60] Jan Zenisek, Florian Holzinger, and Michael Affenzeller. Machine learning based concept drift detection for predictive maintenance. *Computers & Industrial Engineering*, 137:106031, 2019.

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

[62] Arif Wider and Christian Deger. Getting smart: Applying continuous delivery to data science to drive car sales, 2017. Available at `https://www.thoughtworks.com/insights/blog/getting-smart-applying-continuous-delivery-data-science-drive-car-sales`.

[63] David Leslie. Understanding artificial intelligence ethics and safety. *arXiv preprint arXiv:1906.05684*, 2019.

[64] Changhee Han, Leonardo Rundo, Kohei Murao, Takafumi Nemoto, and Hideki Nakayama. Bridging the gap between AI and Healthcare sides: towards developing clinically relevant AI-powered diagnosis systems, 2020.

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

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

[67] Mustafa Suleyman and Dominic King. The Information Commissioner, the Royal Free, and what we've learned, 2017.

[68] Cathy O'Neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Broadway Books, 2016.

[69] Robert Soden, Dennis Wagenaar, Dave Luo, and Annegien Tijssen. Taking ethics, fairness, and bias seriously in machine learning for disaster risk management, 2019.

[70] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *NIPS Tutorial*, 1, 2017.

[71] Nantheera Anantrasirichai and David Bull. Artificial intelligence in the creative industries: A review, 2020.

[72] Yisroel Mirsky and Wenke Lee. The creation and detection of deepfakes: A survey. *arXiv preprint arXiv:2004.11138*, 2020.

[73] Sathappan Muthiah, Patrick Butler, Rupinder Paul Khandpur, Parang Saraf, Nathan Self, Alla Rozovskaya, Liang Zhao, Jose Cadena, Chang-Tien Lu, Anil Vullikanti, et al. 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*, pages 205–214, 2016.

[74] M-C Laï, M Brian, and M-F Mamzer. Perceptions of artificial intelligence in healthcare: findings from a qualitative survey study among actors in France. *Journal of Translational Medicine*, 18(1):1–13, 2020.

[75] Ramprakash Ramamoorthy, P Satya Madhuri, and Malini Christina Raj. AI from labs to production - challenges and learnings. Santa Clara, CA, May 2019. USENIX Association.

[76] Mark Sendak, Madeleine Clare Elish, Michael Gao, Joseph Futoma, William Ratliff, Marshall Nichols, Armando Bedoya, Suresh Balu, and Cara O'Brien. "The human body is a black box": Supporting clinical decision-making with deep learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20, page 99–109, New York, NY, USA, 2020. Association for Computing Machinery.

[77] Information Commissioner's Office. Project ExplAIn interim report, 2019. Available at `https://ico.org.uk/about-the-ico/research-and-reports/project-explain-interim-report/`.

[78] Ram Shankar Siva Kumar, David O Brien, Kendra Albert, Salomé Viljöen, and Jeffrey Snover. Failure modes in machine learning systems. *arXiv preprint arXiv:1911.11034*, 2019.

[79] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[80] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

[81] Ram Shankar Siva Kumar, Magnus Nystrom, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. Adversarial machine learning-industry perspectives. *Available at SSRN 3532474*, 2020.

[82] Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

[83] Oscar Schwartz. In 2016 Microsoft's racist chatbot revealed the dangers of online conversation. *IEEE Spectrum*, 11, 2019.

[84] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. Stealing machine learning models via prediction APIs. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618, 2016.

[85] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333, 2015.

[86] Michael Veale, Reuben Binns, and Lilian Edwards. Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180083, 2018.

[87] Chip Huyen. What I learned from looking at 200 machine learning tools, 2020. Available at `https://huyenchip.com/2020/06/22/mlops.html`.

[88] Kumar Venkateswar. Using Amazon SageMaker to operationalize machine learning. 2019.

[89] AML Team. AzureML: Anatomy of a machine learning service. In *Conference on Predictive APIs and Apps*, pages 1–13, 2016.

[90] Jeremy Hermann and Mike Del Balso. Meet Michelangelo: Uber's machine learning platform. *URL https://eng.uber.com/michelangelo*, 2017.

[91] Denis Baylor, Kevin Haas, Konstantinos Katsiapis, Sammy Leong, Rose Liu, Clemens Menwald, Hui Miao, Neoklis Polyzotis, Mitchell Trott, and Martin Zinkevich. Continuous training for production ML in the TensorFlow extended (TFX) platform. In *2019 {USENIX} Conference on Operational Machine Learning (OpML 19)*, pages 51–53, 2019.

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

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

[94] Stephen H Bach, Daniel Rodriguez, Yintao Liu, Chong Luo, Haidong Shao, Cassandra Xia, Souvik Sen, Alex Ratner, Braden Hancock, Houman Alborzi, et al. Snorkel DryBell: A

case study in deploying weak supervision at industrial scale. In *Proceedings of the 2019 International Conference on Management of Data*, pages 362–375, 2019.

[95] Paroma Varma and Christopher Ré. Snuba: Automating weak supervision to label training data. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 12, page 223. NIH Public Access, 2018.

[96] Curtis G. Northcutt, Lu Jiang, and Isaac L. Chuang. Confident learning: Estimating uncertainty in dataset labels, 2019.

[97] Neil Lawrence. Modern data oriented programming, 2019. Available at `http://inverseprobability.com/talks/notes/modern-data-oriented-programming.html`.

[98] Tom Borchert. Milan: An evolution of data-oriented programming, 2020. Available at `https://tborchertblog.wordpress.com/2020/02/13/28/`.

[99] Alexander Lavin and Gregory Renard. Technology readiness levels for machine learning systems. *arXiv preprint arXiv:2006.12497*, 2020.

[100] Martin Zinkevich. Rules of machine learning: Best practices for ML engineering. *URL: https://developers. google. com/machine-learning/guides/rules-of-ml*, 2017.