machine learning & knowledge extraction

Article

Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology

Stefan Studer [1,*], Thanh Binh Bui [2,*], Christian Drescher [1], Alexander Hanuschkin [1,3], Ludwig Winkler [2], Steven Peters [1] and Klaus-Robert Müller [2,4,5,6]

1 Mercedes-Benz AG, Group Research, Artificial Intelligence Research, 71059 Sindelfingen, Germany
2 Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany
3 Department of Computer Science & Engineering, Esslingen University of Applied Sciences, 73732 Esslingen, Germany
4 Google Research, Brain Team, 10117 Berlin, Germany
5 Department of Artificial Intelligence, Korea University, Seoul 136-713, Korea
6 Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany
* Correspondence: stefan.studer@daimler.com (S.S.); bui@tu-berlin.de (T.B.B.)

Abstract: Machine learning is an established and frequently used technique in industry and academia, but a standard process model to improve success and efficiency of machine learning applications is still missing. Project organizations and machine learning practitioners face manifold challenges and risks when developing machine learning applications and have a need for guidance to meet business expectations. This paper therefore proposes a process model for the development of machine learning applications, covering six phases from defining the scope to maintaining the deployed machine learning application. Business and data understanding are executed simultaneously in the first phase, as both have considerable impact on the feasibility of the project. The next phases are comprised of data preparation, modeling, evaluation, and deployment. Special focus is applied to the last phase, as a model running in changing real-time environments requires close monitoring and maintenance to reduce the risk of performance degradation over time. With each task of the process, this work proposes quality assurance methodology that is suitable to address challenges in machine learning development that are identified in the form of risks. The methodology is drawn from practical experience and scientific literature, and has proven to be general and stable. The process model expands on CRISP-DM, a data mining process model that enjoys strong industry support, but fails to address machine learning specific tasks. The presented work proposes an industry- and application-neutral process model tailored for machine learning applications with a focus on technical tasks for quality assurance.

Keywords: machine learning applications; quality assurance methodology; process model; automotive industry and academia; best practices; guidelines

check for updates

Citation: Studer, S.; Bui, T.B.; Drescher, C.; Hanuschkin, A.; Winkler, L.; Peters, S.; Müller, K.-R. Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *Mach. Learn. Knowl. Extr.* **2021**, 3, 392–413. https://doi.org/10.3390/make3020020

Academic Editor: Andreas Holzinger

Received: 2 March 2021
Accepted: 20 April 2021
Published: 22 April 2021

1. Introduction

Many industries, such as manufacturing [1,2], personal transportation [3], and healthcare [4,5], are currently undergoing a process of digital transformation, challenging established processes with machine learning driven approaches. The expanding demand is highlighted by the Gartner report [6], claiming that organizations expect to double the number of Machine Learning (ML) projects within a year.

However, 75 to 85 percent of practical ML projects currently do not match their sponsors' expectations, according to surveys of leading technology companies [7]. Reference [8] name data and software quality among others as the key challenges in the machine learning life cycle. Another reason is the lack of guidance through standards and development process models specific to ML applications. Industrial organizations, in particular, rely heavily on standards to guarantee a consistent quality of their products or services. A Japanese industry consortium (QA4AI) was founded to address those needs [9].

Due to the lack of a process model for ML applications, many project organizations rely on alternative models that are closely related to ML, such as, the Cross-Industry Standard Process model for Data Mining (CRISP-DM) [10–12]. This model is grounded on industrial data mining experience [12] and is considered most suitable for industrial projects among related process models [13]. In fact, CRISP-DM has become the de facto industry standard [14] process model for data mining, with an expanding number of applications [15], e.g., in quality diagnostics [16], marketing [17], and warranty [18].

However, two major shortcomings of CRISP-DM are identified:

First, CRISP-DM focuses on data mining and does not cover the application scenario of ML models inferring real-time decisions over a long period of time (Figure 1). The ML model must be adaptable to a changing environment or the model's performance will degrade over time, such that permanent monitoring and maintenance of the ML model is required after the deployment.

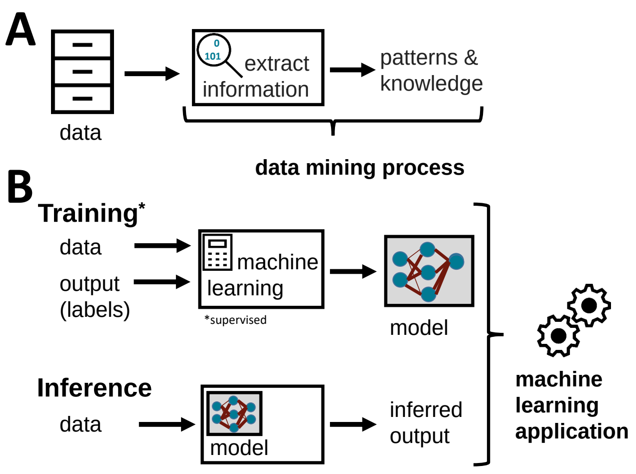

Figure 1. Difference between (**A**) data mining processes and (**B**) machine learning applications.

Secondly and more concerning, CRISP-DM lacks guidance on Quality Assurance (QA) methodology ([11]). This deficit is particularly evident in comparison to standards in the area of information technology [19], but also apparent in alternative process models for data mining [20,21]. In the context of process models for ML, quality is not only defined by the product's fitness for purpose [14], but the quality of the task executions in any phase during the development of a ML application. This ensures that errors are caught as early as possible to minimize costs in the later stages of the development process.

The paper provides two contributions addressing the mentioned shortcomings:

In particular, the first shortcoming is addressed by deriving an end-to-end process model for the development of practical ML applications that covers all relevant phases in the life-cycle of a ML application, using CRISP-DM as a basis, but enlarging the scope with relevant phases supported by literature. The relevance for a process model is motivated by standards in the field of information technology that are proven in use, but do not cover ML specifics (e.g., IEEE 1074-1997 [19]). The model follows the principles of CRISP-DM, in particular by keeping the model industry- and application-neutral, but is modified to the particular requirements of ML applications.

The second shortcoming is addressed by anchoring QA methodology in the proposed process model. The QA methodology is adopted from widespread standards for quality assurance (e.g., IEEE 730-1998 [22]), particularly building on the principle of 'risk based thinking' (DIN EN ISO 9001 [23]). The risk based process is kept generic to be industry and application-neutral and is summarized in a flow chart [24] to give a visual understanding. In this work, risk management is included early in the ML project as proposed by [25] in a preventative way (in contrast to reactive risk management that defines actions for contingencies [19]). The focus of the QA methodology is primarily on the technical tasks needed to produce evidence that every step in the development process is of sufficient quality to warrant adoption into business processes.

The necessary infrastructure is not covered in the process model to be tool, domain, and technology agnostic. Examples on possible infrastructure is given in [26,27].

The following second section describes related work and ongoing research in the development of process models for machine learning applications. In the third chapter, the tasks and QA methodology are introduced for each process phase. Finally, a conclusion and an outlook are given in the fourth chapter.

2. Related Work

CRISP-DM defines a reference framework for carrying out data mining projects and sets out activities to be performed to complete a product or service. The activities are organized in six *phases* as shown in Table 1. The successful completion of a phase initiates the execution of the subsequent activity. CRISP-DM revisits previous steps until success or completion criteria are met. It can be therefore characterized as a waterfall life cycle with backtracking [20]. During the development of applications, processes and tasks to be performed can be derived from the standardized process model. Methodology instantiates these tasks, i.e., stipulates how to do a task (or how it should be done).

For each activity, CRISP-DM defines a set of (generic) tasks that are stable and general. Hereby, tasks are described as *stable* when designed to keep the process up to date with future modeling techniques and are described as *general* when they are intended to cover many possible project scenarios. CRISP-DM has been specialized, e.g., to incorporate temporal data mining (CRISP-TDM; [28]), null-hypothesis driven confirmatory data mining (CRISP-DM0; [29]), evidence mining (CRISP-EM; [30]), data mining in the healthcare (CRISP-MED-DM; [31]), and data mining for engineering applications (DMME; [32,33]).

Complementary to CRISP-DM, process models for ML applications have been proposed ([34,35], and Table 1). [34] conducted an internal study at Microsoft on challenges of ML projects and derived a process model with nine different phases. However, their process model lacks QA methodology and does not cover the business needs. Reference [35] proposed 28 specific tests to quantify issues in the ML pipeline to reduce the technical debt [36] of ML applications. These tests estimate the production readiness of a ML application, i.e., the quality of the application. However, their tests do not completely cover all project phases, e.g., excluding the business understanding activity. Practical experiences [6,7,32–34,37,38] reveal that business understanding is a necessary first step that defines the success criteria and the feasibility for the subsequent tasks. Without first considering the business needs, the ML objectives might be defined orthogonally to business objectives and cause one to spend a great deal of effort producing the rights answers to the wrong questions.

Reference [20] were first to consider quality in the context of process models for data mining. Borrowing ideas from software development, their work suggests creating traceability, test procedures, and test data for challenging the product's fitness for purpose during the evaluation phase.

The need for software life cycle models is apparent and has been extensively studied in the last decade and harmonized in standards, e.g., IEEE 1074-1997 [19]. The focus, however, is on information technology and does not address machine learning specifics. However, core terms defined in standards [19] can be transferred to today's process models and will be used in this work: A software project is defined by 'requirements' and limited by 'constraints' when running the software in a target system. 'Risk management' is introduced as part of a 'software life cycle model' including 'technical', 'operational', and 'economic' risks. Measurable 'metrics', a key component in Six Sigma [39], are defined to trigger tasks when a metric exceeds a defined threshold. Risks can be described quantitative and qualitative following the process of Failure Mode and Effects Analysis (FMEA [40,41]). Due to the importance of QA in software projects, self-contained standards exist, e.g., IEEE

730-1998 [22]. In this context, QA is not only defined as the 'reasonable degree of confidence' for the product's quality, but rather as the 'reasonable degree of confidence that the product is in the process of acquiring required attributes' respective quality during the software development process. This definition motivates the presented approach.

This work builds on the foundations of the mentioned process models for data mining and ML, the standards for software life cycles and QA while addressing the mentioned issues in today's process models. In addition, this work will provide a curated list of references for an in-depth analysis on the specific tasks.

Table 1. Comparing different process models for data mining and machine learning projects. Deep red color highlight data and petrol blue color model related phases.

CRISP-ML(Q)	**CRISP-DM**	**Amershi et al. [34]**	**Breck et al. [35]**	
Business and Data Understanding	Business Understanding	Requirements	-	
	Data Understanding	Collection		
Data Preparation	Data Preparation	Cleaning Labeling Feature Engineering	Data	Infra-structure
Modeling	Modeling	Training	Model	
Evaluation	Evaluation	Evaluation	-	
Deployment	Deployment	Deployment	-	
Monitoring & Maintenance	-	Monitoring	Monitoring	

3. Quality Assurance in Machine Learning Projects

A process model is proposed: CRoss-Industry Standard Process model for the development of Machine Learning applications with Quality assurance methodology (CRISP-ML(Q)). The name is chosen to highlight its compatibility to CRISP-DM. It is designed for the development of machine applications, i.e., application scenarios where a ML model is deployed and maintained as part of a product or service (Figure 1). CRISP-ML(Q) is an iterative model, consequently, the step back to a prior phase or step (e.g., from the modeling phase back to data collection) is an essential part of the process model.

As a first contribution, CRISP-ML(Q) covers a *monitoring and maintenance phase* to address risks of model degradation in a changing environment. This extends the scope of the process model as compared to CRISP-DM and related process models [34,35] (Table 1). Moreover, *business and data understanding* are merged into a single phase because industry practice has taught that these two activities, which are separate in CRISP-DM, are strongly intertwined, since business objectives can be derived or changed based on available data (Table 1). A similar approach has been outlined in the W-Model [42].

As a second contribution, *quality assurance methodology* is introduced in each phase and task of the process model, giving a reasonable degree of confidence that the ML application acquires the expected quality throughout the development process [22]. The CRISP-ML(Q) approach for QA follows a generic quality assurance methodology (Figure 2) founded on 'risk based thinking' (DIN EN ISO 9001 [23]). For every CRISP-ML(Q) phase, requirements and constraints are defined supported by measurable metrics (Six Sigma [39]). Then, steps and tasks are instantiated for the specific application followed by the identification of task-specific risks. Risks are checked for feasibility, e.g., by combining severity and probability of occurrence [40]. If risks are not feasible, appropriate QA tasks are chosen to mitigate the risks.

While general risk management has diverse disciplines [19], this approach focuses on risks that affect the efficiency and success of the ML application and require technical tasks for risk mitigation. Whenever possible, measurable metrics (Six Sigma [39]) are implemented and checked for compliance with the defined objectives.

In what follows, selected tasks from CRISP-ML(Q) are described and QA methodology is proposed to determine whether these tasks were performed according to current standards from industry best practice and academic literature, which have proven to be general and stable, and are therefore suitable to mitigate the task specific risks. The selection reflects tasks and methods that are considered the most important.

It is noted that the processes and QA methods in this document are not designed for safety-critical systems. Safety-critical systems might require different or additional processes and quality measures.

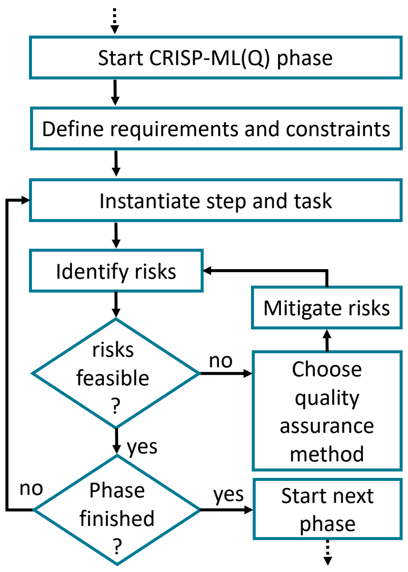

Figure 2. Illustration of the CRISP-ML(Q) approach for QA. The flow chart [24] shows the instantiation of one specific task in a development phase, and the dedicated steps to identify and mitigate risks.

3.1. Business and Data Understanding

The initial phase is concerned with tasks to define the business objectives and translate it to ML objectives, to collect and verify the data quality, and to finally assess the project feasibility.

3.1.1. Define the Scope of the ML Application

CRISP-DM names the data scientist responsible to define the scope of the project. However, in daily business, the separation of domain experts and data scientists carries the risk that the application will not satisfy the business needs. Moreover, the availability of training samples will to a large extent influence the feasibility of the data-based application [34]. It is, therefore, best practice to merge the requirements of the business unit with ML requirements while keeping in mind data related constraints in a joint step.

3.1.2. Success Criteria

The success criteria of a ML project should be measured on three different levels: the business success criteria, the ML success criteria, and the economic success criteria. According to the IEEE standard for developing software life cycle processes [19] and Six Sigma [39], the requirement *measurable* is one of the essential principles of QA methodology. In addition, each success criterion has to be defined in alignment to each other and with respect to the overall system requirements [43] to prevent contradictory objectives. Aligned with the 'Define, Measure, Analyze, Design and Verify' methodology [19], this approach handles risk management in a preventative way to limit the risk of not meeting the success criteria to a reasonable level. Nevertheless, this process model can be applied in a reactive way whenever contingencies occur and offers iterative paths to, e.g., redefine objectives in an earlier phase, stop, or postpone the project.

Business Success Criteria: Define the purpose and the success criteria of the ML application from a business point of view [10] and identify and classify risks. Dependent on the planned application, the business success criteria can include various perspectives on the project, e.g., operational, technical, and economic criteria [19]. In the sense of QA, the common denominator is to define measurable criteria and deduct ML success criteria in the next step.

ML Success Criteria: Translate the business objective into ML success criteria (Table 2) as a measure of technical risks [19]. Unfortunately, there is no such metric that performs best on all ML applications, and the choice of a metric can even privilege one model over the other [44]. As there might be different and often concurrent success criteria, approaches like multi objectives or weighting of success criteria are applicable (Section 3.3). Still, success criteria defined in an theoretical approach can face the issue of a theory-practice gap [8] that might be hard to find in the early phases. It is advised to define a minimum acceptable level of performance to meet the business goals. As an option, a Minimal Viable Product (MVP, [45]) can be defined, and the learnings from the MVP can be used for the reduction of the theory-practice gap in the next iteration of CRISP-ML(Q).

Table 2. Quality measure of machine learning models.

Performance	The Model's Performance on Unseen Data
Robustness	Ability of the ML application to maintain its level of performance under defined circumstances (ISO/IEC technical report 24029 [24])
Scalability	The model's ability to scale to high data volume in the production system.
Explainability	The model's direct or post hoc explainability.
Model Complexity	The model's capacity should suit the data complexity.
Resource Demand	The model's resource demand for deployment.

Economic Success Criteria: It is best practice in business monitoring [46] and manufacturing [47] to add an economic success criterion in the form of a Key Performance Indicator (KPI) to the project. A KPI is an economic measure for the current and future business relevance of an application and requires a precise definition ([48], ISO 22400). Adding a KPI to the machine learning application helps to objectify the business goals of the ML application and can be used for decision making [46]. Once the application is deployed, predictions of a future KPI based on past and present data is applicable [46] and, e.g., costs can be weighted by their expected probability of occurrence [49]. Deviations from the defined success criteria are an indicator for risk.

CRISP-ML(Q) example-step 'success criteria': A ML application is planned for a quality check in production and is supposed to outperform the current manual failure rate of 3%, so the business success criterion can be derived as, e.g., "failure rate less than 3%". Next, the minimal success criterion is defined as "accuracy greater 97%". A KPI can be defined as "cost savings with automated quality check per part". Those success criteria will be monitored throughout the development process and whenever a risk is not feasible (or a risk has occurred), adequate QA measures are implemented. In the current example, there might be the risk of not achieving a accuracy greater than 97%, therefore extra effort is planned for the tasks in Sections 3.1.4 and 3.1.5.

3.1.3. Feasibility

Checking the feasibility before setting up the project is considered best practice for the overall success of the ML approach [50] and can minimize the risk of premature failures due to false expectations. A feasibility test of the ML application should assess the situation and whether further development should be pursued. It is crucial that the assessment includes the availability, size, and quality of the training sample set. In practice, a major source of project delays is the lack of data availability (Section 3.1.4). A small sample size carries the risk of low performance on out-of-sample data [51]. The risk might be mitigated by, e.g., adding domain knowledge or increasing data quality. However, if the sample size is not sufficient, the ML project should be terminated or put on hold at this stage.

Applicability of ML technology: A literature search for either similar applications on a similar domain or similar methodological approaches on a different domain could assess the applicability of the ML technology. It is common to demonstrate the feasibility of a ML application with a Proof of Concept (PoC) when the ML algorithm is used for the first time in a specific domain. If a PoC already exists, setting up a software project that focuses on the deployment directly is more efficient, e.g., in case of yet another price estimation of used cars [52].

Legal constraints: It is beyond the scope of this paper to discuss legal issues, but they are essential for any business application [53,54]. Legal constraints are frequently augmented by ethical and social considerations such as fairness and trust [55–57].

Requirements on the application: The success criteria that have been defined in Section 3.1.2 must be augmented with requirements that arise from running the application in the target domain or, if not accessible, an assumed target domain [43]. The requirements include robustness, scalability, explainability, and resource demand, and are used for the development and verification in later phases (Section 3.3). The challenge during the development is to optimize the success criteria while not violating the requirements and constraints.

3.1.4. Data Collection

Costs and time are needed to collect a sufficient amount of consistent data by preparing and merging data from different sources and different formats (Section 3.2). A ML project might be delayed until the data is collected or could even be stopped if the collection of data of sufficient quality (Section 3.1.5) is not feasible.

Data version control: Collecting data is not a static task, but rather an iterative task. Modification on the data set (Section 3.2) should be documented to mitigate the risk of obtaining irreproducible or wrong results. Version control on the data is one of the essential tools to assure reproducibility and quality, as it allows errors and unfavorable modifications to be tracked during the development.

3.1.5. Data Quality Verification

The following three tasks examine whether the business and ML objectives can be achieved with the given quality of the available data. A ML project is doomed to fail if the data quality is poor. The lack of a certain data quality will trigger the previous data collection task (Section 3.1.4).

Data description: The data description forms the basis for the data quality verification. A description and an exploration of the data is performed to gain insight about the underlying data generation process. The data should be described on a meta-level and by their statistical properties. Furthermore, a technically well-funded visualization of the data should help to understand the data generating process [58]. Information about format, units, and description of the input signals is expanded by domain knowledge.

Data requirements: The data requirements can be defined either on the meta-level or directly in the data, and should state the expected conditions of the data, i.e., whether a certain sample is plausible. The requirements can be, e.g., the expected feature values (a range for continuous features or a list for discrete features), the format of the data and the maximum number of missing values. The bounds of the requirements has to be defined carefully to include all possible real world values but discard non-plausible data. Data that does not satisfy the expected conditions could be treated as anomalies and need to be evaluated manually or excluded automatically. To mitigate the risk of anchoring bias in the definition phase discussing the requirements with a domain expert is advised [35]. Documentation of the data requirements could be expressed in the form of a schema [59,60].

Data verification: The initial data, added data, but also the production data has to be checked according to the requirements (Section 3.6). In cases where the requirements are not met, the data will be discarded and stored for further manual analysis. This helps to reduce the risk of decreasing the performance of the ML application through adding low-quality data and helps to detect varying data distributions or unstable inputs. To mitigate the risk of insufficient representation of extreme cases, it is best practice to use data exploration techniques to investigate the sample distribution.

3.1.6. Review of Output Documents

The Business and Data Understanding phase delivers the scope for the development (Section 3.1.3), the success criteria (Section 3.1.2) of a ML application, and a data quality verification report (Section 3.1.5) to approve the feasibility of the project. The output documents need to be reviewed to rank the risks and define the next tasks. If certain quality criteria are not met, re-iterations of previous tasks are possible.

3.2. Data Preparation

Building on the experience from the preceding data understanding phase, data preparation serves the purpose of producing a data set for the subsequent modeling phase. However, data preparation is not a static phase and backtracking circles from later phases are necessary if, for example, the modeling phase or the deployment phase reveal erroneous data. To path the way towards ML life-cycle in a later phase, methods for data preparation that are suitable for automation as demonstrated by [8] are preferable.

3.2.1. Select Data

Feature selection: Selecting a good data representation based on the available measurements is one of the challenges to assure the quality of the ML application. It is best practice to discard under-utilized features, as they provide little to no modeling benefit, but offer possible loopholes for errors, i.e., instability of the feature during the operation of the ML application [36]. In addition, the more features are selected, the more samples are necessary. Intuitively, an exponentially increasing number of samples for an increasing number of features is required to prevent the data from becoming sparse in the feature space. This is termed as the curse of dimensionality [61,62]. Thus, it is best practice to select just necessary features. A checklist for the feature selection task is given in [63]. Note that data often forms a manifold of lower dimensions in the feature space and models have to learn this, respectively, [64]. Feature selection methods can be separated into three categories: (1) *filter methods* select features from data without considering the model, (2) *wrapper methods* use a learning model to evaluate the significance of the features, and (3) *embedded methods* combine the feature selection and the classifier construction steps. A detailed explanation and in-depth analysis on the feature selection problem are given in [65–68]. Feature selection could carry the risk of selection bias, that could be reduced when the feature selection is performed within the cross-validation of the model (Section 3.3) to account for all possible combinations [69]. Surveys on the feature selection problem are given in [70,71].

However, the selection of the features should not be relied purely on the validation and test error, but should be analyzed by a domain expert as potential biases might occur due to spurious correlation in the data. Lapuschkin et al. [72,73] showed that classifiers could exploit spurious correlations, here the copyright tag on the horse class, to obtain a remarkable test performance and, thus, fake a false sense of generalization. In such cases, explanation methods [74] could be used to highlight the significance of features (Section 3.4) and analyzed from a human's perspective.

Data selection: Discarding samples should be well documented and strictly based on objective quality criteria. However, certain samples might not satisfy the necessary quality, i.e., does not satisfy the requirements defined in Section 3.1.5 and are not plausible and, thus, should be removed from the data set.

Unbalanced Classes: In cases of unbalanced classes, where the number of samples per class is skewed, different sampling strategies could improve the results. Over-sampling of the minority class and/or under-sampling of the majority class [75–78] have been used. Over-sampling increases the importance of the minority class, but could result in overfitting on the minority class. Under-sampling by removing data points from the majority class has to be done carefully to keep the characteristics of the data and reduce the chance of introducing biases. However, removing points close to the decision boundary or multiple data points from the same cluster should be avoided. Comparing the results of different sampling techniques reduces the risk of introducing bias to the model.

3.2.2. Clean Data

Noise reduction: The gathered data often includes, besides the predictive signal, noise and unwanted signals from other sources. Signal processing filters could be used to remove the irrelevant signals from the data and improve the signal-to-noise ratio [79,80]. However, filtering the data should be documented and evaluated because of the risk that an erroneous filter could remove important parts of the signal in the data.

Data imputation: To get a complete data set, missing, NaN (Not a Number), and special values could be imputed with a model readable value. Depending on the data and ML task, the values are imputed by mean or median values, interpolated, replaced by a special value symbol [81] (as the pattern of the values could be informative), substituted by model predictions [82], matrix factorization [83] or multiple imputations [84–86], or imputed based on a convex optimization problem [87]. To reduce the risk of introducing substitution artifacts, the performance of the model should be compared between different imputation techniques.

3.2.3. Construct Data

Feature engineering: New features could be derived from existing ones based on domain knowledge. This could be, for example, the transformation of the features from the time domain into the frequency domain, discretization of continuous features into bins or augmenting the features with additional features based on the existing ones. In addition, there are several generic feature construction methods, such as clustering [88], dimensional reduction methods such as Kernel-PCA [89], or auto-encoders [90]. Nominal features and labels should be transformed into a one-hot encoding, while ordinal features and labels are transformed into numerical values. However, the engineered features should be compared against a baseline to assess the utility of the feature. Underutilized features should be removed. Models that construct the feature representation as part of the learning process, e.g., neural networks, avoid the feature engineering steps [91].

Data augmentation: Data augmentation utilizes known invariances in the data to perform a label preserving transformation to construct new data. The transformations could either be performed in the feature space [76] or input space, such as applying rotation, elastic deformation, or Gaussian noise to an image [92]. Data could also be augmented on a meta-level, such as switching the scenery from a sunny day to a rainy day. This expands the data set with additional samples and allows the model to capture those invariances.

CRISP-ML(Q) example-step 'data augmentation': A ML application is developed to optically detect the position and entity of a metal part on a conveyor and needs to be invariant to different illuminations and backgrounds. Therefore, a data set is synthetically generated, exposing the reflective part with different lighting, backgrounds, and noise levels in a digital tool [93]. The risk of introducing artifacts in the input space is eminent. To mitigate the risk, explanation methods like heat maps [72] are used for QA to verify the model's intended behavior.

3.2.4. Standardize Data

File format: Some ML tools require specific variable or input types (data syntax). Indeed, in practice, the comma separated values (CSV) format is the most generic standard

(RFC 4180). ISO 8000 recommends the use of SI units according to the International System of Quantities. Defining a fix set of standards and units, helps to avoid the risks of errors in the merging process and further in detecting erroneous data (Section 3.1.5).

Normalization: Without proper normalization, the features could be defined on different scales and might lead to strong bias to features on larger scales. In addition, normalized features lead to faster convergence rates in neural networks than without [94,95]. Note that the normalization, applied to the training set has to be applied also to the test set using the same normalization parameters.

3.3. Modeling

The choice of modeling techniques depends on the ML and the business objectives, the data and the boundary conditions of the project the ML application is contributing to. The requirements and constraints that have been defined in Section 3.1 are used as inputs to guide the model selection to a subset of appropriate models. The goal of the modeling phase is to craft one or multiple models that satisfy the given constraints and requirements.

Literature research on similar problems: It is best practice to screen the literature (e.g., publications, patents, internal reports) for a comprehensive overview on similar ML tasks, since ML has become an established tool for a wide number of applications. New models can be based on published insights, and previous results can serve as performance baselines.

Define quality measures of the model: The modeling strategy is required to have multiple objectives in mind [96]. At least six complementary properties of a model (Table 2) should be evaluated. Besides a *performance metric*, soft measures such as *robustness*, *explainability*, *scalability*, *resource demand*, and *model complexity* need to be evaluated (Table 2). In practical application, explainability [51,97,98] or robustness might be valued more than accuracy. In such cases, the measures can be weighted differently depending on the application. The models could be ranked either by summing up the weighted quality measures to a scalar or finding Pareto optimal models in a multi-objective optimization process [99]. Additionally, the model's *fairness* [55,56] or *trust* might have to be assessed and mitigated.

Model Selection: There are plenty of ML models and introductory books on classical methods [62,100], and Deep Learning [91] can be used to compare and understand their characteristics. The model selection depends on the data and has to be tailored to the problem. There is no such model that performs the best on all problem classes (*No Free Lunch Theorem for ML* [101]). It is best practice to start with models of lower capacity, which can serve as baseline, and gradually increase the capacity. Validating each step assures its benefit and avoid unnecessary complexity of the model.

Incorporate domain knowledge: In practice, a specialized model for a specific task performs better than a general model for all possible tasks. However, adapting the model to a specific problem involves the risk of incorporating false assumptions and could reduce the solution space to a non-optimal subset. Therefore, it is best practice to validate the incorporated domain knowledge in isolation against a baseline. Adding domain knowledge should always increase the quality of the model, otherwise, it should be removed to avoid false bias.

Model training: The trained model depends on the learning problem, and as such, they are tightly coupled. The learning problem contains an *objective*, *optimizer*, *regularization*, and *cross-validation* [62,91]. The *objective* of the learning problem depends on the application. Different applications value different aspects and need to be tweaked in alignment with the business success criteria. The objective is a proxy to evaluate the performance of the model. The *optimizer* defines the learning strategy and how to adapt the parameters of the model to improve the objective. *Regularization*, which can be incorporated in the objective, optimizer, and in the model itself, is needed to reduce the risk of overfitting and can help to find unique solutions. *Cross-validation* is performed for feature selection, to optimize the hyper-parameters of the model and to test its generalization property to unseen data [102]. Cross-validation [62] is based on a splitting of historical data in training, validation, and testing data, where the latter is used as a proxy for the target environment [103]. Frameworks such as Auto-ML [104,105] or Neural Architecture Search [106] enable the hyper-parameters optimization and the architecture search to be automated.

Using unlabeled data and pre-trained models: Labeling data could be very expensive and might limit the available data set size. Unlabeled data might be exploited in the training process, e.g., by performing unsupervised pre-training [107] and semi-supervised learning algorithms [108–110].Complementary *transfer learning* could be used to pre-train the network on a proxy data set (e.g., from simulations) that resembles the original data to extract common features [111].

Model Compression: Compression or pruning methods could be used to obtain a more compact model. In kernel methods, low rank approximation of the kernel matrix is an essential tool to tackle large scale learning problems [112,113]. Neural Networks use a different approach [114] by either pruning the network weights [115] or applying a compression scheme on the network weights [116].

Ensemble methods: Ensemble methods train multiple models to perform the decision based on the aggregate decisions of the individual models. The models could be of different types or multiple instantiations of one type. This results in a more fault-tolerant system as the error of one model could be absorbed by the other models. Boosting, Bagging, or Mixture of Experts are mature techniques to aggregate the decision of multiple models [117–119]. In addition, ensemble models are used to compute uncertainty estimates and can highlight areas of low confidence [120,121].

Assure Reproducibility

A major principle of scientific methods and the characteristics of robust ML applications is reproducibility. However, ML models are difficult to reproduce due to the mostly non-convex and stochastic training procedures and randomized data splits. It has been proposed to distinguish reproducibility on two different levels. First, one has to assure that the method itself is reproducible and secondly its results [122].

Method reproducibility: This task aims at reproducing the model from an extensive description or sharing of the used algorithm, data set, hyper-parameters, and run-time environment (e.g., software versions, hardware, and random seeds [123]). The algorithm should be described in detail, i.e., with (pseudo) code and on the meta-level including the assumptions.

Result reproducibility: It is best practice to validate the mean performance and assess the variance of the model on different random seeds [124,125]. Reporting only the top performance of the model [124,126] is common but dubious practice. Large performance variances indicate the sensitivity of the algorithm and question the robustness of the model.

Experimental Documentation: Keeping track of the changed model's performance and its causes by precedent model modifications allows model comprehension by addressing which modifications were beneficial and improve the overall model quality. The documentation should contain the listed properties in the *method reproducibility* task. Tool-based approaches on version control and meta-data handling while experimenting on ML models and hyper-parameters exist [127].

CRISP-ML(Q) example-step 'assure reproducibility': It is favorable to update a ML model every time a deficit in the performance is detected. There is a risk that the ML model cannot be reproduced and hence the deficit in performance cannot be resolved. To mitigated this risk, QA has to address the model reproducibility by tracking and storing relevant model information (e.g., ML source code to train the model, data sets, computation environment). Now, if the ML model requires an update, e.g., in the absence of the developing data scientist, another data scientist can first reproduce the last model and results (as code, data set, environment, etc., are reproducible) and then develop an improved model with a changed code, data set, or environment.

3.4. Evaluation

Validate performance: A risk occurs when information from a test set leak into the validation or even training set. Hence, it is best practice to hold back an additional test set, which is disjointed from the the validation and training set, stored only for a final evaluation and never shipped to any partner to be able to measure the performance metrics (blind-test). The test set should be assembled and curated with caution and ideally by a team of experts that are capable to analyze the correctness and ability to represent real cases. In general, the test set should cover the whole input distribution and consider all invariances, e.g., transformations of the input that do not change the label, in the data. Another major risk is that the test set cannot cover all possible inputs due to the large input dimensionality or rare corner cases, i.e., inputs with low probability of occurring [128–130]. Extensive testing reduces this risk [103]. It is recommended to separate the teams and the procedures collecting the training and the test data to erase dependencies and avoid methodology dependence. Additionally, it is recommended to perform a sliced performance analysis to highlight weak performance on certain classes or time slices.

CRISP-ML(Q) example-step 'validate performance': A ML application is trained to detect ten types of noises in sound data. The number of samples per class vary as some noises have been recorded less frequently. A first ML model achieves an overall accuracy on a test set of 95%. After intensive tuning of hyper-parameters, a second model achieves 96%. A sliced performance analysis is used for QA to mitigate risks in model validation. The analysis reveals, that the second model while performing better on frequent classes has lower accuracy on less frequent classes, which results in a better overall accuracy. The second model is discarded, and the model objectives (Section 3.3) are modified to include a minimal accuracy per class.

Determine robustness: A major risk occurs if ML applications are not robust (in terms of not being able to perform appropriately), if data is perturbed, e.g., noisy or wrong, or manipulated in advance to fool the system (e.g., adversarial attacks) as shown by Chang [131]. This requires methods to statistically estimate the model's local and global robustness. One approach is adding different kinds of noisy or falsified input to the data or varying the hyper-parameters to characterize the model's generalization ability and then verify the quality metrics for robustness defined in Section 3.3. This could be for example the failure rate on adversarial attacks. Formal verification approaches [43] and robustness validation methods using cross-validation techniques on historical data [103] exist. For a comprehensive overview on how to assess robustness of neural networks, including the definition of robustness goals and testing, the ISO/IEC technical report 24029 [24] is recommend.

Increase explainability for ML practitioner & end user: Explainability of a model helps to find errors and allows strategies, e.g., by enriching the data set, to improve the overall performance [132]. In practice, inherently interpretable models are not necessary inferior to complex models in case of structured input data with meaningful features [97]. To achieve explainability and gain a deeper understanding of what a model has already learned and to avoid spurious correlations [73], it is best practice to carefully observe the features which impact the model's prediction the most and check whether they are plausible from a domain expert's point of view [133–135]. Moreover, case studies have shown that explainability helps to increase trust and users' acceptance [136,137] and could guide humans in ML-assisted decisions [98]. Reference [138] lever the value of an explainable framework in a real life example on time series data. Unified frameworks to explore model explainability are available (e.g., [139,140]). With emerging regulations for ML applications, explainability can be a prerequisite for the admission of a ML application in a specific domain. An ongoing topic of research is a standard procedure to measure and compare approaches for explainability [141].

Compare results with defined success criteria: Finally, domain and ML experts must decide if the model can be deployed. Therefore, it is best practice to document the results of the evaluation phase and compare them to the business and ML success criteria defined in Section 3.1.2. If the success criteria are not met, one might backtrack to earlier phases (modeling or even data preparation) or stop the project. Identified limitations of robustness and explainability during evaluation might require an update of the risk assessment (e.g., FMEA [40,41]) and might also lead to backtracking to the modeling phase or stopping the project.

3.5. Deployment

The deployment phase of a ML model is characterized by its practical use in the designated field of application.

Define inference hardware: Choose the hardware based on the requirements defined in Section 3.1.3 or align with an existing hardware. While cloud services offer scalable computation resources, embedded system have hard constraints. ML specific options are, e.g., to optimize towards the target hardware [142] regarding CPU and GPU availability, to optimize towards the target operation system (demonstrated for Android and iOS by [143]) or to optimize the ML workload for a specific platform [144]. Monitoring and maintenance (Section 3.6) need be considered in the overall architecture.

Model evaluation under production condition: The risk persists that the production data does not resemble the training data. Previous assumptions on the training data might not hold in production and the hardware that gathered the data might differ. Therefore, it is best practice to evaluate the performance of the model under incrementally increasing production conditions by iteratively running the tasks in Section 3.4. On each incremental step, the model has to be calibrated to the deployed hardware and the test environment. This allows identifying wrong assumptions on the deployed environment and the causes of model degradation. Domain adaptation techniques can be applied [145,146] to enhance the generalization ability of the model. This step will also give a first indication whether the business and economic success criteria defined in Section 3.1.2, could be met.

CRISP-ML(Q) example-step 'model evaluation under production condition': A ML model for an optical quality check in production is under development to detect several classes of defects. Training data with images from mobile devices is available, whereas in the target environment an industrial camera is used. The use of different cameras carry the risk of a performance gap in the target environment. A production test for QA is advanced with a reduced model trained on only one defect. As the performance does not meet the model objectives defined in Section 3.3, an iteration to the data preparation phase (Section 3.2) to align images from source and target domain is planned.

Assure user acceptance and usability: Even after passing all evaluation steps, there might be the risk that the user acceptance and the usability of the model is underwhelming. The model might be incomprehensible and or does not cover corner cases. It is best practice to build a prototype and run an field test with end users [103]. Examine the acceptance, usage rate, and the user experience. A user guide and disclaimer shall be provided to the end users to explain the system's functionality and limits.

Minimize the risks of unforeseen errors: The risks of unforeseen errors and outage times could cause system shutdowns and a temporary suspension of services. This could lead to user complaints and the declining of user numbers and could reduce the revenue. A fall-back plan, that is activated in case of, e.g., erroneous model updates or detected bugs, can help to tackle the problem. Options are to roll back to a previous version, a pre-defined baseline or a rule-based system. A second option to counteract unforeseen errors is to implement software safety cages that control and limit the outputs of the ML application [147] or even learn safe regions in the state space [148].

Deployment strategy: Even though the model is evaluated rigorously during each previous step, there is the risk that errors might be undetected through the process. Before rolling out a model, it is best practice to setup an, e.g., incremental deployment strategy that includes a pipeline for models and data [96,149]. When cloud architectures are used, strategies can often be aligned on general deployment strategies for cloud software applications [150]. The impact of such erroneous deployments and the cost of fixing errors should be minimized.

3.6. Monitoring and Maintenance

With the expansion from knowledge discovery to data-driven applications to infer real-time decisions, ML models are used over a long period and have a life cycle which has to be managed. The risk of not maintaining the model is the degradation of the performance over time which leads to false predictions and could cause errors in subsequent systems. The main reason for a model to become impaired over time is rooted in the violation of the assumption that the training data and the input data for inference come from the same distribution. The causes of the violations are:

- Non-stationary data distribution: Data distributions change over time and result in a stale training set and, thus, the characteristics of the data distribution are represented incorrectly by the training data. Either a shift in the features and/or in the labels are possible. This degrades the performance of the model over time. The frequency of the changes depends on the domain. Data of the stock market are very volatile, whereas the visual properties of elephants will not change much over the next few years.
- Degradation of hardware: The hardware that the model is deployed on and the sensor hardware will age over time. Wear parts in a system will age and friction characteristics of the system might change. Sensors get noisier or fail over time. This will shift the domain of the system and has to be adapted by the model or by retraining it.
- System updates: Updates on the software or hardware of the system can cause a shift in the environment. For example, the units of a signal got changed during an update. Without notifications, the model would use this scaled input to infer false predictions.

After the underlying problem is known, the necessary methods to circumvent stale models and assure the quality can be formulated. Two sequential tasks in the maintenance phase are proposed to assure or improve the quality of the model. In the *monitor* task, the staleness of the model is evaluated and returns whether the model has to be updated or not. Afterward, the model is updated and evaluated to gauge whether the update was successful.

Monitor: Baylor et al. [96] propose to monitor all input signals and notify when an update has occurred. Therefore, statistics of the incoming data and the predicted labels can be compared to the statistics of the training data. The schema defined in Section 3.1.5 can be used to validate the correctness of the incoming data. Inputs that do not satisfy the schema can be treated as anomalies and denied by the model [96]. Libraries exist to help implement an automatic data validation system [60]. If the labels of the incoming data are known, e.g., in forecasting tasks, the performance of the model can be directly monitored and recorded. An equal approach can be applied to the outputs of the model that underlie a certain distribution if environment conditions are stable and can give an estimate on the number of actions performed when interacting with an environment [36]. The monitoring phase also includes a comparison of the current performance or a predicted future performance [46] with the defined success criteria. The future performance of the application can be predicted using ML. Based on the monitoring results, it can then be decided whether the model should be updated, e.g., if input signals change significantly, the number of anomalies reaches a certain threshold or the performance has reached a lower bound. The decision as to whether the model has to be updated should consider the costs of updating the model and the costs resulting from erroneous predictions due to stale models.

CRISP-ML(Q) example-step 'monitor': A ML based recommendation system proposes digital features as part of a software application. As the application programming interface, that provides input data to the model is frequently updated, the risk of model degradation due to changing input is eminent. Close monitoring of the input data of the ML model is implemented as a QA measure. A counter is implemented to monitor input data rejections and if a predefined threshold is exceeded a data scientist is informed automatically.

Update: In the updating step, new data is collected to re-train the model under the changed data distribution. Consider that new data has to be labeled which could be very expensive. Instead of training a completely new model from scratch, it is advised to fine-tune the existing model to new data. It might be necessary to perform some of the modeling steps in Section 3.3 to cope with the changing data distribution. Every update step has to undergo a new evaluation (Section 3.4) before it can be deployed. The performance of the updated model should be compared against the previous versions and could give insights on the time scale of model degradation. It should be noted, that ML systems might influence their own behavior during updates due to direct, e.g., by influencing its future training data selection, or indirect, e.g., via interaction through the world, feedback loops [36]. The risk of positive feedback loops causing system instability has to be addressed, e.g., by not only monitoring, but limiting the actions of the model.

In addition, as part of the deployment strategy, a module is needed that tracks the application usage and performance and handles several deployment strategies like A/B testing [96,149]. The module can, e.g., be set up in form of a microservice [151] or a directed graph [152]. To reduce the risk of serving erroneous models, an automatic or human controlled fallback to a previous model needs to be implemented. The automation of the update strategy can be boosted up to a continuous training and continuous deployment of the ML application [96] while covering the defined QA methods.

4. Discussion

While ML is a frequently tried technique to challenge established processes with the progress of digital transformation, industry practice experiences difficulties in efficient and effective ML project development, for which three reasons were identified: lack of guidance in the development process, poor data, and poor ML development execution. CRISP-ML(Q) is addressing these issues, covering all development phases that range from project idea formulation to maintaining and monitoring an existing ML application.

After extensive review and comparison of existing processes for related development projects, the proposed process model is based on CRISP-DM, which was adapted to cover the application scenario of ML models inferring real-time decisions. Hence, CRISP-ML(Q) can draw from some advantages of CRISP-DM, in particular, a close relationship to established and proven industry standards that are included in the education of data mining practitioners. In line with the principles of CRISP-DM, the tasks devised by CRISP-ML(Q) were formulated with the goals of being stable enough to support future modeling techniques and general enough to cover many possible knowledge discovery scenarios. Whilst many tasks are shared with CRISP-DM, this paper contributes technical tasks needed to produce evidence that every step in the ML development process is of sufficient quality to warrant the adoption into business processes. The proposed steps/phases are incomplete for safety-critical applications, as their requirements are dependent on the application domain. Safety-critical systems, e.g., for safety in autonomous driving [153], might require different or additional processes and quality measures.

However, as CRISP-ML(Q) is intended to be industry-, tool-, and application-neutral, the selection of quality assurance methodology was limited to rather generic tasks that cannot be a complete set for any ML application domain. The presented methods were also not chosen for their degree of novelty, but rather for having passed the test of time to become best practices in automotive industry projects and academia.

5. Conclusions

This paper puts forward CRISP-ML(Q), to path the way towards a standard cross-industry process model for the development of ML applications with QA methodology. The presented process model is tailored to cover the entire life-cycle of a ML application and is designed to be industry- and application-neutral. The model is aligned with

CRISP-DM, a process model for data mining, but modified and augmented to cover the specifics of a ML application, especially monitoring and maintaining a ML application in a changing environment.

Quality assurance (QA) methodology is integrated into every phase and task of the process model. The QA methodology is adopted from standards following the goal to guarantee quality during the process execution. Core element of the methodology is the identification and mitigation of risks. The methodology is designed to be general and stable, and therefore follows a generic approach that is shown in form of a flow chart. The presented work then focuses on the technical tasks needed in all of the six phases to provide evidence for the quality of the task execution, supported by illustrative examples.

The need for guidance in ML development is revealed in surveys of leading technology companies and by industrial consortia. While there exist standards for QA and risk management in the field of information technology, the field of ML shows a lack of a standards. CRISP-ML(Q) addresses the identified gap presenting a process model to improve efficiency and success rate of ML projects.

A direction of future research includes an evaluation of this work with other industries and applications. Practitioners are encouraged to contribute their knowledge towards the goal of establishing a true CRoss-Industry Standard Process model for the development of Machine Learning applications, that can be integrated into the curriculum of data engineering education. Defining the standard is left to future work.

Author Contributions: Conceptualization, S.S., T.B.B., C.D., A.H., L.W., S.P., and K.-R.M.; methodology, S.S., T.B.B., C.D., A.H., L.W., S.P., and K.-R.M.; writing—original draft preparation, S.S., T.B.B., C.D., A.H., L.W., and S.P.; writing—review and editing, S.S., T.B.B., C.D., A.H., L.W., S.P., and K.-R.M.; supervision, S.P. and K.-R.M.; project administration, S.St. and T.B.B.; funding acquisition, S.P. and K.-R.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the German Federal Ministry of Education and Research (BMBF) via project AIAx-Machine Learning-driven Engineering (Nr. 01IS18048). K.-R.M. acknowledges partial financial support by the BMBF under Grants 01IS14013A-E, 01IS18025A, 01IS18037A, 01GQ1115, and 01GQ0850; Deutsche Forschungsgesellschaft (DFG) under Grant Math+, EXC 2046/1, Project ID 390685689, and by the Technology Promotion (IITP) grant funded by the Korea government (No. 2017-0-00451, No. 2017-0-01779).

Acknowledgments: Special thanks to the Daimler internal AI community. We would like to thank Miriam Hägele, David Headley, Simon Letzgus, Lorenz Linhardt, Danny Panknin, and Andreas Ziehe for proofreading the manuscript and the in-depth discussions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lee, J.; Bagheri, B.; Kao, H.A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manuf. Lett.* **2015**, *3*, 18–23. [CrossRef]
2. Brettel, M.; Friederichsen, N.; Keller, M.; Rosenberg, M. How virtualization, decentralization and network building change the manufacturing landscape: An Industry 4.0 Perspective. *Int. J. Mech. Ind. Sci. Eng.* **2014**, *8*, 37–44.
3. Dikmen, M.; Burns, C.M. Autonomous driving in the real world: Experiences with Tesla autopilot and summon. In Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, Ann Arbor MI USA, 24–26 October 2016; pp. 225–228.
4. Kourou, K.; Exarchos, T.P.; Exarchos, K.P.; Karamouzis, M.V.; Fotiadis, D.I. Machine learning applications in cancer prognosis and prediction. *Comput. Struct. Biotechnol. J.* **2015**, *13*, 8–17. [CrossRef] [PubMed]
5. Esteva, A.; Kuprel, B.; Novoa, R.A.; Ko, J.; Swetter, S.M.; Blau, H.M.; Thrun, S. Dermatologist-level classification of skin cancer with deep neural networks. *Nature* **2017**, *542*, 115. [CrossRef]
6. Andrews, W.; Hare, J. *Survey Analysis: AI and ML Development Strategies, Motivators and Adoption Challenges*; Gartner: Stamford, CT, USA, 2019.
7. Nimdzi Insights. Artificial Intelligence: Localization Winners, Losers, Heroes, Spectators, and You. Available online: https://www.nimdzi.com/wp-content/uploads/2019/06/Nimdzi-AI-whitepaper.pdf (accessed on 21 April 2021).
8. Fischer, L.; Ehrlinger, L.; Geist, V.; Ramler, R.; Sobiezky, F.; Zellinger, W.; Brunner, D.; Kumar, M.; Moser, B. AI System Engineering—Key Challenges and Lessons Learned. *Mach. Learn. Knowl. Extr.* **2021**, *3*, 56–83. [CrossRef]

9. Hamada, K.; Ishikawa, F.; Masuda, S.; Matsuya, M.; Ujita, Y. Guidelines for quality assurance of machine learning-based artificial intelligence. In Proceedings of the SEKE2020: The 32nd International Conference on Software Engineering& Knowledge Engineering, virtual, USA, 9–19 July 2020; pp. 335–341.
10. Chapman, P.; Clinton, J.; Kerber, R.; Khabaza, T.; Reinartz, T.; Shearer, C.; Wirth, R. CRISP-DM 1.0 Step-by-Step Data Mining Guide. Available online: https://www.kde.cs.uni-kassel.de/wp-content/uploads/lehre/ws2012-13/kdd/files/CRISPWP-0800.pdf (accessed on 21 April 2021).
11. Wirth, R.; Hipp, J. CRISP-DM: Towards a standard process model for data mining. In Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining, Manchester, UK, 11–13 April 2000; pp. 29–39.
12. Shearer, C. The CRISP-DM Model: The New Blueprint for Data Mining. *J. Data Warehous.* **2000**, *5*, 13–22.
13. Kurgan, L.; Musilek, P. A survey of Knowledge Discovery and Data Mining process models. *Knowl. Eng. Rev.* **2006**, *21*, 1–24. [CrossRef]
14. Mariscal, G.; Marbán, O.; Fernández, C. A survey of data mining and knowledge discovery process models and methodologies. *Knowl. Eng. Rev.* **2010**, *25*, 137–166. [CrossRef]
15. Kriegel, H.P.; Borgwardt, K.M.; Kröger, P.; Pryakhin, A.; Schubert, M.; Zimek, A. Future trends in data mining. *Data Min. Knowl. Discov.* **2007**, *15*, 87–97. [CrossRef]
16. de Abajo, N.; Diez, A.B.; Lobato, V.; Cuesta, S.R. ANN Quality Diagnostic Models for Packaging Manufacturing: An Industrial Data Mining Case Study. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, 22–25 August 2004; pp. 799–804.
17. Gersten, W.; Wirth, R.; Arndt, D. Predictive modeling in automotive direct marketing: Tools, experiences and open issues. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Boston, MA, USA, 20–23 August 2000; pp. 398–406.
18. Hipp, J.; Lindner, G. Analysing Warranty Claims of Automobiles; An Application Description following the CRISP-DM Data Mining Process. In Proceedings of the Fifth International Computer Science Conference, Hong Kong, China, 13–15 December 1999; pp. 31–40.
19. IEEE. *Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes*; Technical Report; IEEE: New York, NY, USA, 1997.
20. Marbán, O.; Segovia, J.; Menasalvas, E.; Fernández-Baizán, C. Toward data mining engineering: A software engineering approach. *Inf. Syst.* **2009**, *34*, 87–107. [CrossRef]
21. SAS. *SEMMA Data Mining Methodology*; Technical Report; SAS Institute: Cary, NC, USA, 2016.
22. IEEE. *Standard for Software Quality Assurance Plans*; IEEE Std 730-1998; IEEE: New York, NY, USA, 1998; pp. 1–24. [CrossRef]
23. de Normalisation, Comite Europeen. *EN ISO 9001:2015 Quality Management Systems-Requirements*; Technical Report; ISO: Geneva, Switzerland, 2015.
24. ISO/IEC JTC 1/SC 42 Artificial intelligence; *ISO/IEC TR 24029: Artificial Intelligence (AI) — Assessment of the Robustness of Neural Networks*; Technical Report; ISO/IEC: Geneva, Switzerland, 2021.
25. Holzinger, A.; Kieseberg, P.; Weippl, E.; Tjoa, A.M. Current Advances, Trends and Challenges of Machine Learning and Knowledge Extraction: From Machine Learning to Explainable AI. In *Machine Learning and Knowledge Extraction*; Holzinger, A., Kieseberg, P., Tjoa, A.M., Weippl, E., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 1–8.
26. Hazelwood, K.; Bird, S.; Brooks, D.; Chintala, S.; Diril, U.; Dzhulgakov, D.; Fawzy, M.; Jia, B.; Jia, Y.; Kalro, A.; et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In Proceedings of the 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA), Vienna, Austria, 24–28 February 2018; pp. 620–629.
27. Breck, E.; Polyzotis, N.; Roy, S.; Whang, S.E.; Zinkevich, M. Data infrastructure for machine learning. In Proceedings of the SysML Conference, Stanford, CA, USA, 15–16 February 2018.
28. Catley, C.; Smith, K.P.; McGregor, C.; Tracy, M. Extending CRISP-DM to incorporate temporal data mining of multidimensional medical data streams: A neonatal intensive care unit case study. In Proceedings of the 22nd IEEE International Symposium on Computer-Based Medical Systems, Sao Carlos, SP, Brazil, 22–25 June 2009; pp. 1–5.
29. Heath, J.; McGregor, C. CRISP-DM0: A method to extend CRISP-DM to support null hypothesis driven confirmatory data mining. In Proceedings of the 1st Advances in Health Informatics Conference, Kitchener, Ontario, Canada, 28–30 April 2010; pp. 96–101.
30. Venter, J.; de Waal, A.; Willers, C. Specializing CRISP-DM for evidence mining. In Proceedings of the IFIP International Conference on Digital Forensics. Orlando, FL, USA, 30 January–1 February 2007; pp. 303–315.
31. Niaksu, O. CRISP Data Mining Methodology Extension for Medical Domain. *Balt. J. Mod. Comput.* **2015**, *3*, 92–109.
32. Huber, S.; Wiemer, H.; Schneider, D.; Ihlenfeldt, S. DMME: Data mining methodology for engineering applications—A holistic extension to the CRISP-DM model. In Proceedings of the 12th CIRP Conference on Intelligent Computation in Manufacturing Engineering, Gulf of Naples, Italy, 18–20 July 2018; doi:10.1016/j.procir.2019.02.106. [CrossRef]
33. Wiemer, H.; Drowatzky, L.; Ihlenfeldt, S. Data Mining Methodology for Engineering Applications (DMME)—A Holistic Extension to the CRISP-DM Model. *Appl. Sci.* **2019**, *9*, 2407, doi:10.3390/app9122407. [CrossRef]
34. Amershi, S.; Begel, A.; Bird, C.; DeLine, R.; Gall, H.; Kamar, E.; Nagappan, N.; Nushi, B.; Zimmermann, T. Software Engineering for Machine Learning: A Case Study. In Proceedings of the International Conference on Software Engineering (ICSE 2019)-Software Engineering in Practice Track, Montréal, QC, Canada, 25–31 May 2019.

35. Breck, E.; Cai, S.; Nielsen, E.; Salib, M.; Sculley, D. The ML test score: A rubric for ML production readiness and technical debt reduction. In Proceedings of the 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, USA, 11–14 December 2017; pp. 1123–1132.
36. Sculley, D.; Holt, G.; Golovin, D.; Davydov, E.; Phillips, T.; Ebner, D.; Chaudhary, V.; Young, M.; Crespo, J.F.; Dennison, D. Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems*; ACM: New York, NY, USA, 2015; pp. 2503–2511.
37. Kim, M.; Zimmermann, T.; DeLine, R.; Begel, A. Data Scientists in Software Teams: State of the Art and Challenges. *IEEE Trans. Softw. Eng.* **2018**, *44*, 1024–1038. [CrossRef]
38. d. S. Nascimento, E.; Ahmed, I.; Oliveira, E.; Palheta, M.P.; Steinmacher, I.; Conte, T. Understanding Development Process of Machine Learning Systems: Challenges and Solutions. In Proceedings of the 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), Porto de Galinhas, Brazil, 19–20 September 2019; pp. 1–6. [CrossRef]
39. Surange, V.G. Implementation of Six Sigma to reduce cost of quality: A case study of automobile sector. *J. Fail. Anal. Prev.* **2015**, *15*, 282–294. [CrossRef]
40. Yang, C.; Letourneau, S.; Zaluski, M.; Scarlett, E. APU FMEA Validation and Its Application to Fault Identification. In Proceedings of the ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, QC, Canada, 15–18 August 2010; Volume 3. [CrossRef]
41. AIAG; der Automobilindustrie (VDA), V. *FMEA Handbook-Failure Mode and Effects Analysis*; AIAG: Southfield, MI, USA, 2019.
42. Falcini, F.; Lami, G.; Mitidieri Costanza, A. Deep Learning in Automotive Software. *IEEE Softw.* **2017**, *34*, 56–63. [CrossRef]
43. Kuwajima, H.; Yasuoka, H.; Nakae, T. Open Problems in Engineering and Quality Assurance of Safety Critical Machine Learning Systems. *CoRR* **2018**, *abs/1812.03057*. Available online: https://arxiv.org/pdf/1904.00001v1.pdf (accessed on 21 April 2021).
44. Gunawardana, A.; Shani, G. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. *J. Mach. Learn. Res.* **2009**, *10*, 2935–2962.
45. Lenarduzzi, V.; Taibi, D. MVP Explained: A Systematic Mapping Study on the Definitions of Minimal Viable Product. In Proceedings of the 2016 42th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Limassol, Cyprus, 31 August–2 September 2016; pp. 112–119. [CrossRef]
46. Thakur, A.; Beck, R.; Mostaghim, S.; Großmann, D. Survey into predictive key performance indicator analysis from data mining perspective. In Proceedings of the 2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Vienna, Austria, 8–11 September 2020; Volume 1, pp. 476–483. [CrossRef]
47. Ramis Ferrer, B.; Muhammad, U.; Mohammed, W.M.; Martínez Lastra, J.L. Implementing and Visualizing ISO 22400 Key Performance Indicators for Monitoring Discrete Manufacturing Systems. *Machines* **2018**, *6*. [CrossRef]
48. Badawy, M.; El-Aziz, A.A.; Idress, A.M.; Hefny, H.; Hossam, S. A survey on exploring key performance indicators. *Future Comput. Inform. J.* **2016**, *1*, 47–52. [CrossRef]
49. Hoffmann, M.W.; Wildermuth, S.; Gitzel, R.; Boyaci, A.; Gebhardt, J.; Kaul, H.; Amihai, I.; Forg, B.; Suriyah, M.; Leibfried, T.; et al. Integration of Novel Sensors and Machine Learning for Predictive Maintenance in Medium Voltage Switchgear to Enable the Energy and Mobility Revolutions. *Sensors* **2020**, *20*. [CrossRef]
50. Watanabe, Y.; Washizaki, H.; Sakamoto, K.; Saito, D.; Honda, K.; Tsuda, N.; Fukazawa, Y.; Yoshioka, N. Preliminary Systematic Literature Review of Machine Learning System Development Process. *arXiv* **2019**, arXiv:cs.LG/1910.05528.
51. Rudin, C.; Carlson, D. The Secrets of Machine Learning: Ten Things You Wish You Had Known Earlier to be More Effective at Data Analysis. *arXiv* **2019**, arXiv:cs.LG/1906.01998.
52. Pudaruth, S. Predicting the price of used cars using machine learning techniques. *Int. J. Inf. Comput. Technol* **2014**, *4*, 753–764.
53. Reed, C.; Kennedy, E.; Silva, S. Responsibility, Autonomy and Accountability: Legal liability for machine learning. Available online: https://ssrn.com/abstract=2853462 (accessed on 21 April 2021).
54. Bibal, A.; Lognoul, M.; de Streel, A.; Frénay, B. Legal requirements on explainability in machine learning. *Artif. Intell. Law* **2020**, 1–21; doi: 10.1007/s10506-020-09270-4. [CrossRef]
55. Binns, R. Fairness in Machine Learning: Lessons from Political Philosophy. In Proceedings of the 1st Conference on Fairness, Accountability and Transparency, New York, NY, USA, 23–24 February 2018; Friedler, S.A., Wilson, C., Eds.; PMLR: New York, NY, USA, Volume 81, pp. 149–159.
56. Corbett-Davies, S.; Goel, S. The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning. *arXiv* **2018**, arXiv:cs.CY/1808.00023.
57. Barredo Arrieta, A.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. *arXiv* **2019**, arXiv:cs.AI/1910.10045.
58. McQueen, J.; Meilă, M.; VanderPlas, J.; Zhang, Z. Megaman: Scalable manifold learning in python. *J. Mach. Learn. Res.* **2016**, *17*, 5176–5180.
59. Polyzotis, N.; Roy, S.; Whang, S.E.; Zinkevich, M. Data management challenges in production machine learning. In Proceedings of the 2017 ACM International Conference on Management of Data, Chicago, IL, USA, 14–19 May 2017; pp. 1723–1726.
60. Schelter, S.; Biessmann, F.; Lange, D.; Rukat, T.; Schmidt, P.; Seufert, S.; Brunelle, P.; Taptunov, A. Unit Testing Data with Deequ. In Proceedings of the 2019 International Conference on Management of Data, Amsterdam, The Netherlands, 30 June–5 July 2019; pp. 1993–1996.

61. Keogh, E.; Mueen, A., Curse of Dimensionality. In *Encyclopedia of Machine Learning and Data Mining*; Sammut, C., Webb, G.I., Eds.; Springer US: Boston, MA, USA, 2017; pp. 314–315._192. [CrossRef]
62. Bishop, C.M. *Pattern Recognition and Machine Learning, 5th ed.*; Information Science and Statistics, Springer: Berlin/Heidelberg, Germany, 2007.
63. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
64. Braun, M.L.; Buhmann, J.M.; Müller, K.R. On relevant dimensions in kernel feature spaces. *J. Mach. Learn. Res.* **2008**, *9*, 1875–1908.
65. Hira, Z.M.; Gillies, D.F. A review of feature selection and feature extraction methods applied on microarray data. *Adv. Bioinform.* **2015**, *2015*. [CrossRef]
66. Saeys, Y.; Inza, I.n.; Larrañaga, P. A Review of Feature Selection Techniques in Bioinformatics. *Bioinformatics* **2007**, *23*, 2507–2517. [CrossRef]
67. Chandrashekar, G.; Sahin, F. A Survey on Feature Selection Methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. leceng.2013.11.024. [CrossRef]
68. Guyon, I.; Gunn, S.; Nikravesh, M.; Zadeh, L.A. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*; Springer: Berlin/Heidelberg, Germany, 2006.
69. Ambroise, C.; McLachlan, G.J. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 6562–6566. [CrossRef]
70. Cai, J.; Luo, J.; Wang, S.; Yang, S. Feature selection in machine learning: A new perspective. *Neurocomputing* **2018**, *300*, 70–79. [CrossRef]
71. Blum, A.L.; Langley, P. Selection of relevant features and examples in machine learning. *Artif. Intell.* **1997**, *97*, 245–271. [CrossRef]
72. Lapuschkin, S.; Binder, A.; Montavon, G.; Müller, K.R.; Samek, W. Analyzing classifiers: Fisher vectors and deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2912–2920.
73. Lapuschkin, S.; Wäldchen, S.; Binder, A.; Montavon, G.; Samek, W.; Müller, K.R. Unmasking Clever Hans predictors and assessing what machines really learn. *Nat. Commun.* **2019**, *10*, 1096. [CrossRef] [PubMed]
74. Samek, W.; Montavon, G.; Vedaldi, A.; Hansen, L.K.; Müller, K.R. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*; Springer Nature: Berlin/Heidelberg, Germany, 2019; Volume 11700.
75. Lawrence, S.; Burns, I.; Back, A.; Tsoi, A.C.; Giles, C.L. Neural network classification and prior class probabilities. In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 299–313.
76. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
77. Batista, G.E.A.P.A.; Prati, R.C.; Monard, M.C. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *SIGKDD Explor. Newsl.* **2004**, *6*, 20–29. [CrossRef]
78. Lemaître, G.; Nogueira, F.; Aridas, C.K. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *J. Mach. Learn. Res.* **2017**, *18*, 559–563.
79. Walker, J.S. *A Primer on Wavelets and Their Scientific Applications*; CRC Press: Boca Raton, FL, USA, 2002.
80. Lyons, R.G. *Understanding Digital Signal Processing*, 2nd ed.; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2004.
81. Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Sci. Rep.* **2018**, *8*, 6085. [CrossRef] [PubMed]
82. Biessmann, F.; Salinas, D.; Schelter, S.; Schmidt, P.; Lange, D. Deep Learning for Missing Value Imputationin Tables with Non-Numerical Data. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Turin, Italy, 22–26 October 2018; pp. 2017–2025.
83. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
84. Murray, J.S. Multiple imputation: A review of practical and theoretical findings. *Stat. Sci.* **2018**, *33*, 142–159. [CrossRef]
85. White, I.R.; Royston, P.; Wood, A.M. Multiple imputation using chained equations: Issues and guidance for practice. *Stat. Med.* **2011**, *30*, 377–399. [CrossRef]
86. Azur, M.J.; Stuart, E.A.; Frangakis, C.; Leaf, P.J. Multiple imputation by chained equations: What is it and how does it work? *Int. J. Methods Psychiatr. Res.* **2011**, *20*, 40–49. [CrossRef]
87. Bertsimas, D.; Pawlowski, C.; Zhuo, Y.D. From Predictive Methods to Missing Data Imputation: An Optimization Approach. *J. Mach. Learn. Res.* **2018**, *18*, 1–39.
88. Coates, A.; Ng, A.Y. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 561–580.
89. Schölkopf, B.; Smola, A.; Müller, K.R. Kernel principal component analysis. In *Proceedings of the International Conference on Artificial Neural Networks*; Springer-Verlag: Berlin/Heidelberg, Germany, 1997; pp. 583–588.
90. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. *Learning Internal Representations by Error Propagation*; Technical Report; California Univ San Diego La Jolla Inst for Cognitive Science: La Jolla, CA, USA.
91. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
92. Wong, S.C.; Gatt, A.; Stamatescu, V.; McDonnell, M.D. Understanding data augmentation for classification: When to warp? In Proceedings of the 2016 international conference on digital image computing: techniques and applications (DICTA), Gold Coast, Australia, 30 November– 2 December 2016; pp. 1–6.

93. Andulkar, M.; Hodapp, J.; Reichling, T.; Reichenbach, M.; Berger, U. Training CNNs from Synthetic Data for Part Handling in Industrial Environments. In Proceedings of the 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich, Germany, 20–24 August 2018; pp. 624–629.
94. LeCun, Y.A.; Bottou, L.; Orr, G.B.; Müller, K.R. Efficient backprop. In *Neural Networks: Tricks of the Trade*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 9–48.
95. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015; Volume 37, pp. 448–456
96. Baylor, D.; Breck, E.; Cheng, H.T.; Fiedel, N.; Foo, C.Y.; Haque, Z.; Haykal, S.; Ispir, M.; Jain, V.; Koc, L.; et al. TFX: A TensorFlow-based production-scale machine learning platform. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 1387–1395.
97. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef]
98. Schmidt, P.; Biessmann, F. Quantifying Interpretability and Trust in Machine Learning Systems. *arXiv* **2019**, arXiv:1901.08558.
99. Marler, R.T.; Arora, J.S. Survey of multi-objective optimization methods for engineering. *Struct. Multidiscip. Optim.* **2004**, *26*, 369–395. [CrossRef]
100. Schölkopf, B.; Smola, A.J. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*; MIT Press: Cambridge, MA, USA, 2002.
101. Wolpert, D.H. The Lack of a Priori Distinctions Between Learning Algorithms. *Neural Comput.* **1996**, *8*, 1341–1390. [CrossRef]
102. Müller, K.R.; Mika, S.; Rätsch, G.; Tsuda, K.; Schölkopf, B. An introduction to kernel-based learning algorithms. *IEEE Trans. Neural Netw.* **2001**, *12*, 181–201. [CrossRef]
103. Zhang, J.M.; Harman, M.; Ma, L.; Liu, Y. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering* **2020**, *1-1*.. [CrossRef]
104. Hutter, F.; Kotthoff, L.; Vanschoren, J. (Eds.) *Automated Machine Learning-Methods, Systems, Challenges*; The Springer Series on Challenges in Machine Learning, Springer: Berlin/Heidelberg, Germany, 2019; doi:10.1007/978-3-030-05318-5. [CrossRef]
105. Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.; Blum, M.; Hutter, F. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28*; MIT Press: Cambridge, MA, USA, 2015; pp. 2962–2970.
106. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv* **2016**, arXiv:1611.01578.
107. Erhan, D.; Bengio, Y.; Courville, A.; Manzagol, P.A.; Vincent, P.; Bengio, S. Why Does Unsupervised Pre-training Help Deep Learning? *J. Mach. Learn. Res.* **2010**, *11*, 625–660.
108. Dreher, D.; Schmidt, M.; Welch, C.; Ourza, S.; Zündorf, S.; Maucher, J.; Peters, S.; Dreizler, A.; Böhm, B.; Hanuschkin, A. Deep Feature Learning of In-Cylinder Flow Fields to Analyze CCVs in an SI-Engine. *Int. J. Engine Res.* **2020**. [CrossRef]
109. Kingma, D.P.; Mohamed, S.; Jimenez Rezende, D.; Welling, M. Semi-supervised Learning with Deep Generative Models. In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; MIT Press: Cambridge, MA, USA, 2014; pp. 3581–3589.
110. Chapelle, O.; Schlkopf, B.; Zien, A. *Semi-Supervised Learning*, 1st ed.; The MIT Press: Cambridge, MA, USA, 2010.
111. Yosinski, J.; Clune, J.; Bengio, Y.; Lipson, H. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems 27*; Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q., Eds.; MIT Press: Cambridge, MA, USA, 2014; pp. 3320–3328.
112. Williams, C.K.I.; Seeger, M. Using the Nyström Method to Speed Up Kernel Machines. In *Advances in Neural Information Processing Systems 13*; Leen, T.K., Dietterich, T.G., Tresp, V., Eds.; MIT Press: Cambridge, MA, USA, 2001; pp. 682–688.
113. Drineas, P.; Mahoney, M.W. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *J. Mach. Learn. Res.* **2005**, *6*, 2153–2175.
114. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. A Survey of Model Compression and Acceleration for Deep Neural Networks. *CoRR* **2017**, *abs/1710.09282*. Available online: https://arxiv.org/pdf/1710.09282.pdf (accessed on 21 April 2021).
115. Frankle, J.; Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv* **2018**, arXiv:1803.03635.
116. Wiedemann, S.; Kirchhoffer, H.; Matlage, S.; Haase, P.; Marbán, A.; Marinc, T.; Neumann, D.; Nguyen, T.; Osman, A.; Marpe, D.; Schwarz, H.; Wiegand, T.; Samek, W. DeepCABAC: A Universal Compression Algorithm for Deep Neural Networks. *IEEE J. Sel. Top. Signal Process.* **2020**, *14*, 700–714; doi: 10.1109/JSTSP.2020.2969554. [CrossRef]
117. Rokach, L. Ensemble-based classifiers. *Artif. Intell. Rev.* **2010**, *33*, 1–39. [CrossRef]
118. Zhou, Z.H.; Wu, J.; Tang, W. Ensembling neural networks: Many could be better than all. *Artif. Intell.* **2002**, *137*, 239–263. [CrossRef]
119. Opitz, D.; Maclin, R. Popular ensemble methods: An empirical study. *J. Artif. Intell. Res.* **1999**, *11*, 169–198. [CrossRef]
120. Lakshminarayanan, B.; Pritzel, A.; Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2017; pp. 6402–6413.
121. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1050–1059.
122. Pineau, J. The Machine Learning Reproducibility Checklist. 2019. Available online: https://www.cs.mcgill.ca/~jpineau/ReproducibilityChecklist.pdf (accessed on 21 April 2021).

123. Tatman, R.; VanderPlas, J.; Dane, S. A Practical Taxonomy of Reproducibility for Machine Learning Research. Available online: https://openreview.net/forum?id=B1eYYK5QgX (accessed on 21 April 2021).
124. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep reinforcement learning that matters. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018.
125. Sculley, D.; Snoek, J.; Wiltschko, A.; Rahimi, A. Winner's Curse? On Pace, Progress, and Empirical Rigor. Available online: https://openreview.net/forum?id=rJWF0Fywf (accessed on 21 April 2021).
126. Bouthillier, X.; Laurent, C.; Vincent, P. Unreproducible Research is Reproducible. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Volume 97, pp. 725–734.
127. Vartak, M.; Subramanyam, H.; Lee, W.E.; Viswanathan, S.; Husnoo, S.; Madden, S.; Zaharia, M. ModelDB: A System for Machine Learning Model Management. In Proceedings of the Workshop on Human-In-the-Loop Data Analytics, San Francisco, CA, USA, 26 June 2016; pp. 14:1–14:3. [CrossRef]
128. Zhou, Z.Q.; Sun, L. Metamorphic Testing of Driverless Cars. *Commun. ACM* **2019**, *62*, 61–67. [CrossRef]
129. Tian, Y.; Pei, K.; Jana, S.; Ray, B. DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars. In Proceedings of the 40th International Conference on Software Engineering, 27 May–3 June 2018. Gothenburg, Sweden; pp. 303–314. [CrossRef]
130. Pei, K.; Cao, Y.; Yang, J.; Jana, S. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, 28–31 October 2017; pp. 1–18. [CrossRef]
131. Chan-Hon-Tong, A. An Algorithm for Generating Invisible Data Poisoning Using Adversarial Noise That Breaks Image Classification Deep Learning. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 192–204. [CrossRef]
132. Chakarov, A.; Nori, A.V.; Rajamani, S.K.; Sen, S.; Vijaykeerthy, D. Debugging Machine Learning Tasks. *arXiv* **2016**, arXiv:1603.07292.
133. Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.R.; Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS ONE* **2015**, *10*, e0130140. [CrossRef] [PubMed]
134. Baehrens, D.; Schroeter, T.; Harmeling, S.; Kawanabe, M.; Hansen, K.; Müller, K.R. How to explain individual classification decisions. *J. Mach. Learn. Res.* **2010**, *11*, 1803–1831.
135. Arras, L.; Horn, F.; Montavon, G.; Müller, K.R.; Samek, W. "What is relevant in a text document?": An interpretable machine learning approach. *PLoS ONE* **2017**, *12*, e0181142. [CrossRef] [PubMed]
136. Hois, J.; Theofanou-Fuelbier, D.; Junk, A.J. How to Achieve Explainability and Transparency in Human AI Interaction. In *HCI International 2019-Posters*; Stephanidis, C., Ed.; Springer International Publishing: Cham, Switzerland, 2019; pp. 177–183.
137. Schneider, T.; Hois, J.; Rosenstein, A.; Gerlicher, A.; Theofanou-Fülbier, D.; Ghellal, S. ExplAIn Yourself! Transparency for Positive UX in Autonomous Driving. In Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems, Denver, US, USA, 6–11 May 2021; CHI '21. [CrossRef]
138. Thrun, M.C.; Ultsch, A.; Breuer, L. Explainable AI Framework for Multivariate Hydrochemical Time Series. *Mach. Learn. Knowl. Extr.* **2021**, *3*, 170–204. [CrossRef]
139. Alber, M.; Lapuschkin, S.; Seegerer, P.; Hägele, M.; Schütt, K.T.; Montavon, G.; Samek, W.; Müller, K.R.; Dähne, S.; Kindermans, P.J. iNNvestigate neural networks! *J. Mach. Learn. Res.* **2019**, *20*, 1–8.
140. Nori, H.; Jenkins, S.; Koch, P.; Caruana, R. InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv* **2019**, arXiv:cs.LG/1909.09223.
141. Burkart, N.; Huber, M.F. A Survey on the Explainability of Supervised Machine Learning. *J. Artif. Intell. Res.* **2021**, *70*, 245–317. [CrossRef]
142. Wu, C.J.; Brooks, D.; Chen, K.; Chen, D.; Choudhury, S.; Dukhan, M.; Hazelwood, K.; Isaac, E.; Jia, Y.; Jia, B.; et al. Machine learning at Facebook: Understanding inference at the edge. In Proceedings of the 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA), Washington, DC, USA, 16–20 February 2019; pp. 331–344.
143. Sehgal, A.; Kehtarnavaz, N. Guidelines and Benchmarks for Deployment of Deep Learning Models on Smartphones as Real-Time Apps. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 450–465. [CrossRef]
144. Christidis, A.; Davies, R.; Moschoyiannis, S. Serving Machine Learning Workloads in Resource Constrained Environments: A Serverless Deployment Example. In Proceedings of the 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA), Kaohsiung, Taiwan, China, 18–21 November 2019; pp. 55–63. [CrossRef]
145. Wang, M.; Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing* **2018**, *312*, 135 – 153. [CrossRef]
146. Sugiyama, M.; Krauledat, M.; Müller, K.R. Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res.* **2007**, *8*, 985–1005.
147. Heckemann, K.; Gesell, M.; Pfister, T.; Berns, K.; Schneider, K.; Trapp, M. Safe automotive software. In Proceedings of the International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, Kaiserslautern, Germany, 10–12 September 2011; pp. 167–176.
148. Berkenkamp, F.; Moriconi, R.; Schoellig, A.P.; Krause, A. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. In Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, USA, 12–14 December 2016; pp. 4661–4666.
149. Derakhshan, B.; Mahdiraji, A.R.; Rabl, T.; Markl, V. Continuous Deployment of Machine Learning Pipelines. In Proceedings of the 22nd International Conference on Extending Database Technology (EDBT), Lisbon, Portugal, 26–29 March 2019; pp. 397–408.

150. Fehling, C.; Leymann, F.; Retter, R.; Schupeck, W.; Arbitter, P. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*; Springer: Berlin/Heidelberg, Germany, 2014. [CrossRef]
151. Muthusamy, V.; Slominski, A.; Ishakian, V. Towards Enterprise-Ready AI Deployments Minimizing the Risk of Consuming AI Models in Business Applications. In Proceedings of the 2018 First International Conference on Artificial Intelligence for Industries (AI4I), Laguna Hills, CA, USA, 26–28 September 2018; pp. 108–109. [CrossRef]
152. Ghanta, S.; Subramanian, S.; Sundararaman, S.; Khermosh, L.; Sridhar, V.; Arteaga, D.; Luo, Q.; Das, D.; Talagala, N. Interpretability and Reproducability in Production Machine Learning Applications. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Miami, FL, USA, 14–17 December 2018; pp. 658–664.
153. Aptiv; Audi; Baidu; BMW; Continental; Daimler; FCA; here; infineo; intel; et al. Safety First For Automated Driving. 2019. Available online: https://www.daimler.com/dokumente/innovation/sonstiges/safety-first-for-automated-driving.pdf (accessed on 2 July 2019).