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[3], Ludwig Winkler[2], Steven Peters [1] and Klaus-Robert Müller[4]

[1] Mercedes-Benz AG, Group Research, Artificial Intelligence Research, 71059 Sindelfingen, Germany; stefan.studer@daimler.com (S.St.); christian.d.drescher@daimler.com (C.D.); steven.peters@daimler.com (S.P.)
[2] Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany; bui@tu-berlin.de (T.B.B.); winkler@tu-berlin.de (L.W.)
[3] Mercedes-Benz AG, Group Research, Artificial Intelligence Research, Sindelfingen, Germany and Esslingen University of Applied Sciences, Germany; alexander.hanuschkin@daimler.com (A.H.)
[4] Machine Learning Group, Technische Universität Berlin, 10587 Berlin, Germany and Google Research, Brain team, Berlin, Germany and Dept. of Artificial Intelligence, Korea University, Seoul 136-713, South Korea and Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany; klaus-robert.mueller@tu-berlin.de (K.-R.M.)
* Correspondence: stefan.studer@daimler.com (S.St.) and bui@tu-berlin.de (T.B.B.); equal contribution

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 have a need for guidance throughout the life cycle of a machine learning application to meet business expectations. We therefore propose a process model for the development of machine learning applications, that covers six phases from defining the scope to maintaining the deployed machine learning application. The first phase combines business and data understanding as data availability oftentimes affects the feasibility of the project. The sixth phase covers state-of-the-art approaches for monitoring and maintenance of a machine learning applications, as the risk of model degradation in a changing environment is eminent. With each task of the process, we propose quality assurance methodology that is suitable to address challenges in machine learning development that we identify in 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 lacks to address machine learning specific tasks. Our work proposes an industry and application neutral process model tailored for machine learning applications with focus on technical tasks for quality assurance.

Keywords: Machine Learning Applications; Quality Assurance Methodology; Process Model; Automotive Industry and Academia; Best Practices; Guidelines

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. *Preprints* **2021**, *1*, 0. https://doi.org/

Received: 2 March 2021
Accepted:
Published:

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]. Fischer *et al.* [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]. It is grounded on industrial data mining experience [12] and is considered most suitable for industrial projects amongst 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, we have identified two major shortcomings of CRISP-DM:

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 (see fig. 1). The ML model has to be adaptable to a changing environment or the model's performance will degrade over time, such that, a permanent monitoring and maintaining of the ML model is required after the deployment.

Second, and more worrying, CRISP-DM lacks guidance on quality assurance methodology.

This oversight 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 our definition, quality is not only defined by the product's fitness for its 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 during the development. The initial effort and cost to perform the quality assurance methodology is expected to outbalance the risk of fixing errors in a later state, that are typically more expensive due to increased project complexity [22,23]. Our process 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 and proposes quality assurance methodology that became industry best practice. Our contributions focus primarily on the technical tasks needed to produce evidence that every step in the development process is of sufficient quality to warrant the adoption into business processes.

The following second section describes the related work and ongoing research in the development of process models for machine learning applications. In the third chapter, the tasks and quality assurance 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* (see table 1). The successful completion of a phase initiates the execution of the subsequent activity. CRISP-DM includes iterations of revisiting 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 called *stable* when they are designed to keep the process model up to date with new modeling techniques to come and *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; [24]), null-hypothesis driven confirmatory data mining (CRISP-DM0; [25]), evidence mining (CRISP-EM; [26]), and data mining in the healthcare (CRISP-MED-DM; [27]).

Complementary to CRISP-DM, process models for ML applications have been proposed [28,29] (see Table 1). Amershi *et al.* [28] 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 quality assurance methodology and does not cover the business needs.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2021 doi:10.20944/preprints202103.0135.v1

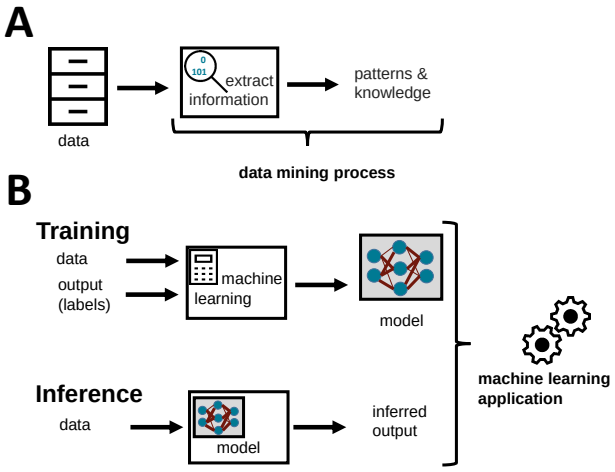

Figure 1. Difference between data mining processes and machine learning applications. A) In the data mining process information is directly extracted from data to find pattern und gain knowledge. B) A machine learning application consists of two steps. A machine learning model on data is trained and applied to perform inference on new data. Note that the model itself can be studied to gain insight within a knowledge discovery process.

Breck *et al.* [29] proposed 28 specific tests to quantify issues in the ML pipeline to reduce the technical debt [30] of ML applications. These tests estimate the production readiness of a ML application, i.e., the quality of the application in our context. However, their tests do not completely cover all project phases, e.g., excluding the business understanding activity. Practical experiences reveal that business understanding is a necessary first step that defines the success criteria and the feasibility for the subsequent tasks. Without considering the business needs, the ML objectives might be defined orthogonal to the business objectives and causes to spend a great deal of effort producing the rights answers to the wrong questions.

To our knowledge, Marbán *et al.* [20] were the 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 its purpose during the evaluation phase.

We address these issues by devising a process model for the development of practical ML applications. In addition, we will provide a curated list of references for an in-depth analysis on the specific tasks.

3. Quality Assurance in Machine Learning Projects

We propose a process model that we call **CR**oss-**I**ndustry **S**tandard **P**rocess model for the development of **M**achine **L**earning applications with **Q**uality assurance methodology (CRISP-ML(Q)) 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 (see fig. 1).

As a first contribution, *quality assurance methodology* is introduced in each phase and task of the process model (see fig. 2). The quality methodology serves to mitigate risks that affect the success and efficiency of the machine learning application. As a second 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, see Table 1. Moreover, *business and data understanding* are merged into a single phase because industry practice has taught us 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 (see Table 1). A similar approach has been outlined in the W-Model [31].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2021 doi:10.20944/preprints202103.0135.v1

<table>
<tr><th>CRISP-ML(Q)</th><th>CRISP-DM</th><th>Amershi et al. [28]</th><th colspan="2">Breck et al. [29]</th></tr>
<tr><td rowspan="2">Business & Data Understanding</td><td>Business Understanding</td><td>Requirements</td><td colspan="2">-</td></tr>
<tr><td>Data Understanding</td><td>Collection</td><td rowspan="4">Data</td><td rowspan="7">Infrastructure</td></tr>
<tr><td rowspan="3">Data Preparation</td><td rowspan="3">Data Preparation</td><td>Cleaning</td></tr>
<tr><td>Labeling</td></tr>
<tr><td>Feature Engineering</td></tr>
<tr><td>Modeling</td><td>Modeling</td><td>Training</td><td>Model</td></tr>
<tr><td>Evaluation</td><td>Evaluation</td><td>Evaluation</td><td>-</td></tr>
<tr><td>Deployment</td><td>Deployment</td><td>Deployment</td><td>-</td></tr>
<tr><td>Monitoring & Maintenance</td><td>-</td><td>Monitoring</td><td colspan="2">Monitoring</td></tr>
</table>

Table 1: Comparing different process models for DM and ML projects. Business and data understanding phases are merged in CRISP-ML(Q) and a separate maintenance phase is introduced in comparison to CRISP-DM. Amershi *et al.* [28] and Breck *et al.* [29] lack the business understanding phase. Deep red color highlight data and petrol blue color model related phases.

In what follows, we describe selected tasks from CRISP-ML(Q) and propose quality assurance methodology 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 suitable to mitigate the task specific risks. The selection reflects tasks and methods that we consider the most important.

The flow chart in fig. 2 explains the CRISP-ML(Q) approach for quality assurance. Requirements and constraints define the objectives of a generic phase, instantiate specific steps and tasks and identify risks, that can affect the efficiency and success of the ML application. If risks aren't feasible, appropriate quality assurance methods are chosen to mitigate risks in an iterative approach using guidelines and checklists. 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. Holzinger *et al.* [32] propose to include risk management early in the ML project.

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

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 finaly 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 [28]. 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.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2021 doi:10.20944/preprints202103.0135.v1

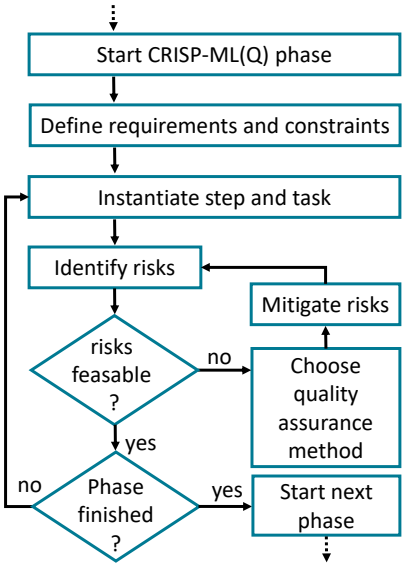

Figure 2. Illustration of the CRISP-ML(Q) approach for quality assurance. The flow chart shows the instantiation of one specific tasks in a development phase, and the dedicated steps to identify and mitigate risks.

3.1.2. Success Criteria

We propose to measure the success criteria of a ML project 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], the requirement *measurable* is one of the essential principles of quality assurance methodology. In addition, each success criterion has to be defined in alignment to each other and with respect to the overall system requirements [33] to prevent contradictory objectives.

Business Success Criteria: Define the purpose and the success criteria of the ML application from a business point of view. For example, if an ML application is planned for a quality check in production and is supposed to outperform the current manual failure rate of 3%, the business success criterion could be derived as e.g. "failure rate less than 3%".

ML Success Criteria: Translate the business objective into ML success criteria (see table 2). It is advised to define a minimum acceptable level of performance to meet the business goals (e.g. for a Minimal Viable Product (MVP)) In the mentioned example, the minimal success criterion is defined as "accuracy greater 97%", but data scientists might optimize further, for example, the true-positive rate to miss items with quality issues.

Economic Success Criteria: It is best practice to add an economic success criterion in the form of a Key Performance Indicator (KPI) to the project. A KPI is an economical measure for the relevance of the ML application. In the mentioned example, a KPI can be defined as "cost savings with automated quality check per part".

3.1.3. Feasibility

Checking the feasibility before setting up the project is considered best practice for the overall success of the ML approach [34] 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 (see section 3.1.4). A small sample size carries the risk of low performance on out-of-sample data [35]. 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: 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 [36].

Legal constraints: It is beyond the scope of this paper to discuss legal issues but they are essential for any business application [37,38]. Legal constraints are frequently augmented by ethical and social considerations like fairness and trust [39–41].

Requirements on the application: The success criteria that have been defined in section 3.1.2 have to be augmented with requirements that arise from running the application in the target domain or if not accessible an assumed target domain [33]. The requirements include robustness, scalability, explainability and resource demand and are used for the development and verification in later phases (see 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 is needed to collect a sufficient amount of consistent data by preparing and merging data from different sources and different formats (see 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 (see 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 (see 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 to track errors and unfavorable modifications 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 (see 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 [42]. 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 have 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 [29]. Documentation of the data requirements could be done in the form of a schema [43,44].

Data verification: The initial data, added data but also the production data has to be checked according to the requirements (see 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

 doi:10.20944/preprints202103.0135.v1

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 & 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 Fischer *et al.* [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 underutilized features as they provide little to none modeling benefit but offer possible loopholes for errors i.e. instability of the feature during the operation of the ML application [30]. 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 [45,46]. Thus, it is best practice to select just necessary features. A checklist for the feature selection task is given in [47]. Note that data often forms a manifold of lower dimensions in the feature space and models have to learn this respectively [48]. 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* combines the feature selection and the classifier construction steps. A detailed explanation and in-depth analysis on the feature selection problem are given in [49–52]. Feature selection could carry the risk of selection bias but could be reduced when the feature selection is performed within the cross-validation of the model (see section 3.3) to account for all possible combinations [53].

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.* [54,55] showed that classifiers could exploit spurious correlations, here the copyright tag on the horse class, to obtain a remarkable test performance and, thus, fakes a false sense of generalization. In such cases, explanation methods [56] could be used to highlight the significance of features (see 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. doesn't 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 [57–60] 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

 doi:10.20944/preprints202103.0135.v1

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 [61,62]. 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 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 [63] (as the pattern of the values could be informative), substituted by model predictions [64], matrix factorization [65] or multiple imputations [66–68] or imputed based on a convex optimization problem [69]. 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 [70], dimensional reduction methods such as Kernel-PCA [71] or auto-encoders [72]. 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 [73].

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 [58] or input space, such as applying rotation, elastic deformation or Gaussian noise to an image [74]. 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.

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 (see 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 [75,76]. 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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 March 2021 doi:10.20944/preprints202103.0135.v1

phase is to craft one or multiple models that satisfy the given constraints and requirements. An outline of the modeling phase is depicted in fig. 3.

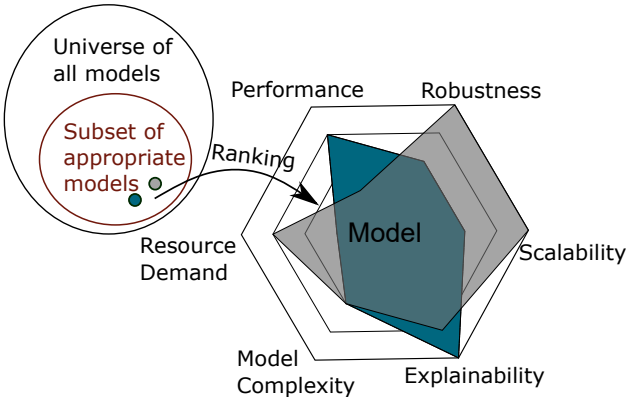

Figure 3. An outline of the modeling phase. Only a subset of models fulfill the constraints and requirements defined in section 3.1 and section 3.1.3 and have to be evaluated using quality measures (see table 2)

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 has to have multiple objectives in mind [77]. We suggest to evaluate the models on at least six complementary properties (see fig. 3). Besides a *performance metric*, soft measures such as *robustness*, *explainability*, *scalability*, *resource demand* and *model complexity* have to be evaluated (see table 2). The measures can be weighted differently depending on the application. In practical application, explainability [35,78,79] or robustness might be valued more than accuracy. Additionally, the model's *fairness* [39,40] or *trust* might have to be assessed and mitigated.

Performance	The model's performance on unseen data.
Robustness	The model's resiliency to inconsistent inputs and to failures in the execution environment.
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.

Table 2: Quality measure of machine learning models

Model Selection: There are plenty of ML models and introductory books on classical methods [46,80] and Deep Learning [73] 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* [81]). It is best practice to start with models of lower capacity,

 doi:10.20944/preprints202103.0135.v1

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 assumption 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 are tightly coupled. The learning problem contains an *objective*, *optimizer*, *regularization* and *cross-validation* [46,73]. The *objective* of the learning problem depends on the application. Different applications value different aspects and have 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 [82]. Cross-validation [46] 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 [83]. Frameworks such as Auto-ML [84,85] or Neural Architecture Search [86] enables to partly automatize the hyper-parameters optimization and the architecture search.

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 [87,88] and semi-supervised learning algorithms [89,90]. 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 [91].

Model Compression: Compression or pruning methods could be used to obtain a more compact model. In kernel methods low rank approximations of the kernel matrix is an essential tool to tackle large scale learning problems [92,93]. Neural Networks use a different approach [94] by either pruning the network weights [95] or applying a compression scheme on the network weights [96].

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 [97–99]. In addition, ensemble models are used to compute uncertainty estimates and can highlight areas of low confidence [100,101].

3.3.1. 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 [102].

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 [103]). The algorithm should be described in detail i.e. with (pseudo) code and on the meta-level including the assumptions.

 doi:10.20944/preprints202103.0135.v1

Result reproducibility: It is best practice to validate the mean performance and assess the variance of the model on different random seeds [104,105]. Reporting only the top performance of the model [104,106] 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 approach on version control and meta-data handling while experimenting on ML models and hyper-parameters exist [107].

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 disjoint from 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 occuring [108–110]. Extensive testing reduces this risk [83]. 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.

Determine robustness: A major risk occurs if ML applications are not robust to perturbed, e.g. noisy or wrong, or even designed adversarial input data as show by Chan-Hon-Tong [111]. 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. Formal verification approaches [33] and robustness validation methods using cross-validation techniques on historical data [83] exist.

The model's robustness should match the quality claims made in table 2.

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 [112]. In practice, inherently interpretable models are not necessary inferior to complex models in case of structured input data with meaningful features [78]. To achieve explainability and gain a deeper understanding of what a model has already learned and to avoid spurious correlations [55], 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 experts' point of view [113–115]. Moreover, case studies have shown that explainability helps to increase trust and users' acceptance [116] and could guide humans in ML assisted decisions [79]. Thrun *et al.* [117] lever the value of an explainable framework in a real life example on time series data. Unified frameworks to explore model explainabilty are available (e.g. [118,119]).

Compare results with defined success criteria: Finally, domain and ML experts have to 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. Failure Mode and Effects Analysis (FMEA)) and might also lead to backtracking to the modeling phase or stopping the project.

 doi:10.20944/preprints202103.0135.v1

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 [120] regarding CPU and GPU availability, to optimize towards the target operation system (demonstrated for Android and iOS by Sehgal and Kehtarnavaz [121]) or to optimize the ML workload for a specific platform [122]. Monitoring and maintenance (see section 3.6) have to 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 [123,124] to enhance the generalization ability of the model. This step will also give a first indication whether the business and economic success criteria, which was defined in section 3.1.2, could be met.

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 [83]. 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 [125] or even learn safe regions in the state space [126].

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 [77,127]. When cloud architectures are used, strategies can often be aligned on general deployment strategies for cloud software applications [128]. 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

 doi:10.20944/preprints202103.0135.v1

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 won't change much over the next 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, we can formulate the necessary methods to circumvent stale models and assure the quality. We propose two sequential tasks in the maintenance phase 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.[77] proposes 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. Complementary, 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 [77]. Libraries exist to help implementing an automatic data validation system [44]. 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 [30]. The monitoring phase also includes a comparison of the performance with the defined success criteria. Based on the monitoring results, it can be decided upon 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 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.

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 [30]. 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 [77,127]. The module can e.g. be set up in form of a microservice [23] or a directed graph [129]. 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 [77] while covering the defined quality assurance methods.

4. Discussion

We have introduced CRISP-ML(Q), a process model for ML applications with quality assurance methodology, that helps organizations to increase efficiency and the success

rate in their ML projects. It guides ML practitioners through the entire ML development life-cycle, stepping into the phases and tasks of the iterative process including maintenance and monitoring. Whenever tasks specific risks can be identified, we provide quality-oriented methods to mitigate those risks. All methods provided are considered best practices in ML projects in industry and academia.

Our survey is indicative of the existence of specialist literature, but its contributions are not covered in ML textbooks and are not part of the academic curriculum. Hence, novices to industry practice often lack a profound state-of-the-art knowledge to mitigate risks and ensure project success. Stressing quality assurance methodology is particularly important because many ML practitioners focus solely on improving the predictive performance.

5. Conclusions

An important future step on the basis of our and related work is the standardization of a process model. This would contribute to more successful ML projects and thus would have a major impact on the ML community [14].

Note that the process and quality measures in this work are not designed for safety-relevant systems. Their study and and the discussion of legal constrains are left to future work.

We encourage industry from automotive and other domains to implement CRISP-ML(Q) in their machine learning applications and contribute their knowledge to establish a CRoss-Industry Standard Process model for the development of machine learning applications with Quality assurance methodology in the future.

Author Contributions: Conceptualization, S.St., T.B.B., C.D., A.H., L.W., S.P. and K.-R.M.; methodology, S.St., T.B.B., C.D., A.H., L.W., S.P. and K.-R.M.; writing—original draft preparation, S.St., T.B.B., C.D., A.H., L.W. and S.P.; writing—review and editing, S.St., 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 funded by the German Federal Ministry of Education and Research (BMBF) for funding the 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 internal Daimler AI community. We would like to thank Miriam Hägele, Lorenz Linhardt, Simon Letzgus, Danny Panknin and Andreas Ziehe for proofreading the manuscript and the in-depth discussions.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the writing of the manuscript, or in the decision to publish the results.

References

1. Lee, J.; Bagheri, B.; Kao, H.A. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manufacturing letters* **2015**, *3*, 18–23.
2. Brettel, M.; Friederichsen, N.; Keller, M.; Rosenberg, M. How virtualization, decentralization and network building change the manufacturing landscape: An Industry 4.0 Perspective. *International journal of mechanical, industrial science and engineering* **2014**, *8*, 37–44.
3. Dikmen, M.; Burns, C.M. Autonomous driving in the real world: Experiences with Tesla autopilot and summon. Proceedings of the 8th international conference on automotive user interfaces and interactive vehicular applications. ACM, 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. *Computational and structural biotechnology journal* **2015**, *13*, 8–17.
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.
6. Andrews, W.; Hare, J. Survey Analysis: AI and ML Development Strategies, Motivators and Adoption Challenges, 2019.

7. Nimdzi Insights. Artificial Intelligence: Localization Winners, Losers, Heroes, Spectators, and You. Technical report, Pactera EDGE, 2019.
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. *Machine Learning and Knowledge Extraction* **2021**, *3*, 56–83. doi:10.3390/make3010004.
9. Hamada, K.; Ishikawa, F.; Masuda, S.; Matsuya, M.; Ujita, Y. Guidelines for quality assurance of machine learning-based artificial intelligence. SEKE2020: the 32nd International Conference on Software Engineering & Knowledge Engineering, 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. Technical report, The CRISP-DM consortium, 2000.
11. Wirth, R.; Hipp, J. CRISP-DM: Towards a standard process model for data mining. Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining, 2000, pp. 29–39.
12. Shearer, C. The CRISP-DM Model: The New Blueprint for Data Mining. *Journal of Data Warehousing* **2000**.
13. Kurgan, L.; Musilek, P. A survey of Knowledge Discovery and Data Mining process models. *The Knowledge Engineering Review* **2006**, *21*, 1–24.
14. Mariscal, G.; Marbán, O.; Fernández, C. A survey of data mining and knowledge discovery process models and methodologies. *Knowledge Eng. Review* **2010**, *25*, 137–166. doi:10.1017/S0269888910000032.
15. Kriegel, H.P.; Borgwardt, K.M.; Kröger, P.; Pryakhin, A.; Schubert, M.; Zimek, A. Future trends in data mining. *Data Mining and Knowledge Discovery* **2007**, *15*, 87–97.
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. Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004, pp. 799–804.
17. Gersten, W.; Wirth, R.; Arndt, D. Predictive modeling in automotive direct marketing: tools, experiences and open issues. Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 398–406.
18. Hipp, J.; Lindner, G. Analysing Warranty Claims of Automobiles; An Application Description following the CRISP-DM Data Mining Process. Proceedings of the Fifth International Computer Science Conference, 1999, pp. 31–40.
19. IEEE. Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes. Technical report, IEEE, 1997.
20. Marbán, O.; Segovia, J.; Menasalvas, E.; Fernández-Baizán, C. Toward data mining engineering: A software engineering approach. *Information Systems* **2009**, *34*, 87–107.
21. SAS. SEMMA Data Mining Methodology. Technical report, SAS Institute, 2016.
22. Surange, V.G. Implementation of Six Sigma to reduce cost of quality: a case study of automobile sector. *Journal of Failure Analysis and Prevention* **2015**, *15*, 282–294.
23. Muthusamy, V.; Slominski, A.; Ishakian, V. Towards Enterprise-Ready AI Deployments Minimizing the Risk of Consuming AI Models in Business Applications. 2018 First International Conference on Artificial Intelligence for Industries (AI4I), 2018, pp. 108–109. doi:10.1109/AI4I.2018.8665685.
24. 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. *22nd IEEE International Symposium on Computer-Based Medical Systems* **2009**, pp. 1–5.
25. Heath, J.; McGregor, C. CRISP-DM0: A method to extend CRISP-DM to support null hypothesis driven confirmatory data mining. 1st Advances in Health Informatics Conference, 2010, pp. 96–101.
26. Venter, J.; de Waal, A.; Willers, C. Specializing CRISP-DM for evidence mining. IFIP International Conference on Digital Forensics. Springer, 2007, pp. 303–315.
27. Niaksu, O. CRISP Data Mining Methodology Extension for Medical Domain. *Baltic Journal of Modern Computing* **2015**, *3*, 92–109.
28. 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. International Conference on Software Engineering (ICSE 2019) - Software Engineering in Practice track, 2019.
29. Breck, E.; Cai, S.; Nielsen, E.; Salib, M.; Sculley, D. The ML test score: A rubric for ML production readiness and technical debt reduction. 2017 IEEE International Conference on Big Data (Big Data). IEEE, 2017, pp. 1123–1132.
30. 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. Advances in neural information processing systems, 2015, pp. 2503–2511.
31. Falcini, F.; Lami, G.; Mitidieri Costanza, A. Deep Learning in Automotive Software. *IEEE Software* **2017**, *34*, 56–63. doi:10.1109/MS.2017.79.
32. 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. Machine Learning and Knowledge Extraction; Holzinger, A.; Kieseberg, P.; Tjoa, A.M.; Weippl, E., Eds.; Springer International Publishing: Cham, 2018; pp. 1–8.
33. Kuwajima, H.; Yasuoka, H.; Nakae, T. Open Problems in Engineering and Quality Assurance of Safety Critical Machine Learning Systems. *CoRR* **2018**, *abs/1812.03057*, [1812.03057].

34. 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, 2019, [arXiv:cs.LG/1910.05528].
35. Rudin, C.; Carlson, D. The Secrets of Machine Learning: Ten Things You Wish You Had Known Earlier to be More Effective at Data Analysis, 2019, [arXiv:cs.LG/1906.01998].
36. Pudaruth, S. Predicting the price of used cars using machine learning techniques. *Int. J. Inf. Comput. Technol* **2014**, *4*, 753–764.
37. Reed, C.; Kennedy, E.; Silva, S. Responsibility, Autonomy and Accountability: legal liability for machine learning. *Queen Mary School of Law Legal Studies Research Paper* **2016**.
38. Bibal, A.; Lognoul, M.; de Streel, A.; Frénay, B. Legal requirements on explainability in machine learning. *Artificial Intelligence and Law* **2020**, pp. 1–21.
39. Binns, R. Fairness in Machine Learning: Lessons from Political Philosophy. Proceedings of the 1st Conference on Fairness, Accountability and Transparency; Friedler, S.A.; Wilson, C., Eds.; PMLR: New York, NY, USA, 2018; Vol. 81, *Proceedings of Machine Learning Research*, pp. 149–159.
40. Corbett-Davies, S.; Goel, S. The Measure and Mismeasure of Fairness: A Critical Review of Fair Machine Learning, 2018, [arXiv:cs.CY/1808.00023].
41. Barredo Arrieta, A.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; Chatila, R.; Herrera, F. Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI **2019**. [arXiv:cs.AI/1910.10045].
42. McQueen, J.; Meilă, M.; VanderPlas, J.; Zhang, Z. Megaman: scalable manifold learning in python. *The Journal of Machine Learning Research* **2016**, *17*, 5176–5180.
43. Polyzotis, N.; Roy, S.; Whang, S.E.; Zinkevich, M. Data management challenges in production machine learning. Proceedings of the 2017 ACM International Conference on Management of Data. ACM, 2017, pp. 1723–1726.
44. Schelter, S.; Biessmann, F.; Lange, D.; Rukat, T.; Schmidt, P.; Seufert, S.; Brunelle, P.; Taptunov, A. Unit Testing Data with Deequ. Proceedings of the 2019 International Conference on Management of Data. ACM, 2019, pp. 1993–1996.
45. 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, 2017; pp. 314–315. doi:10.1007/978-1-4899-7687-1_192.
46. Bishop, C.M. *Pattern recognition and machine learning, 5th Edition*; Information science and statistics, Springer, 2007.
47. Guyon, I.; Elisseeff, A. An Introduction to Variable and Feature Selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
48. Braun, M.L.; Buhmann, J.M.; Müller, K.R. On relevant dimensions in kernel feature spaces. *Journal of Machine Learning Research* **2008**, *9*, 1875–1908.
49. Hira, Z.M.; Gillies, D.F. A review of feature selection and feature extraction methods applied on microarray data. *Advances in bioinformatics* **2015**, *2015*.
50. Saeys, Y.; Inza, I.n.; Larrañaga, P. A Review of Feature Selection Techniques in Bioinformatics. *Bioinformatics* **2007**, *23*, 2507–2517. doi:10.1093/bioinformatics/btm344.
51. Chandrashekar, G.; Sahin, F. A Survey on Feature Selection Methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. doi:10.1016/j.compeleceng.2013.11.024.
52. Guyon, I.; Gunn, S.; Nikravesh, M.; Zadeh, L.A. *Feature Extraction: Foundations and Applications (Studies in Fuzziness and Soft Computing)*; Springer-Verlag: Berlin, Heidelberg, 2006.
53. Ambroise, C.; McLachlan, G.J. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the national academy of sciences* **2002**, *99*, 6562–6566.
54. Lapuschkin, S.; Binder, A.; Montavon, G.; Müller, K.R.; Samek, W. Analyzing classifiers: Fisher vectors and deep neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2912–2920.
55. Lapuschkin, S.; Wäldchen, S.; Binder, A.; Montavon, G.; Samek, W.; Müller, K.R. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature communications* **2019**, *10*, 1096.
56. Samek, W.; Montavon, G.; Vedaldi, A.; Hansen, L.K.; Müller, K.R. *Explainable AI: interpreting, explaining and visualizing deep learning*; Vol. 11700, Springer Nature, 2019.
57. 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, 1998; pp. 299–313.
58. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* **2002**, *16*, 321–357.
59. 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. doi:10.1145/1007730.1007735.
60. 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.
61. Walker, J.S. *A primer on wavelets and their scientific applications*; CRC press, 2002.
62. Lyons, R.G. *Understanding Digital Signal Processing (2Nd Edition)*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 2004.
63. Che, Z.; Purushotham, S.; Cho, K.; Sontag, D.; Liu, Y. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* **2018**, *8*, 6085.

64. Biessmann, F.; Salinas, D.; Schelter, S.; Schmidt, P.; Lange, D. Deep Learning for Missing Value Imputationin Tables with Non-Numerical Data. Proceedings of the 27th ACM International Conference on Information and Knowledge Management. ACM, 2018, pp. 2017–2025.
65. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer* **2009**, *42*, 30–37. doi:10.1109/MC.2009.263.
66. Murray, J.S.; others. Multiple imputation: a review of practical and theoretical findings. *Statistical Science* **2018**, *33*, 142–159.
67. White, I.R.; Royston, P.; Wood, A.M. Multiple imputation using chained equations: issues and guidance for practice. *Statistics in medicine* **2011**, *30*, 377–399.
68. Azur, M.J.; Stuart, E.A.; Frangakis, C.; Leaf, P.J. Multiple imputation by chained equations: what is it and how does it work? *International journal of methods in psychiatric research* **2011**, *20*, 40–49.
69. Bertsimas, D.; Pawlowski, C.; Zhuo, Y.D. From Predictive Methods to Missing Data Imputation: An Optimization Approach. *Journal of Machine Learning Research* **2018**, *18*, 1–39.
70. Coates, A.; Ng, A.Y. Learning feature representations with k-means. In *Neural networks: Tricks of the trade*; Springer, 2012; pp. 561–580.
71. Schölkopf, B.; Smola, A.; Müller, K.R. Kernel principal component analysis. International conference on artificial neural networks. Springer, 1997, pp. 583–588.
72. 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, 1985.
73. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016.
74. Wong, S.C.; Gatt, A.; Stamatescu, V.; McDonnell, M.D. Understanding data augmentation for classification: when to warp? 2016 international conference on digital image computing: techniques and applications (DICTA). IEEE, 2016, pp. 1–6.
75. LeCun, Y.A.; Bottou, L.; Orr, G.B.; Müller, K.R. Efficient backprop. In *Neural networks: Tricks of the trade*; Springer, 2012; pp. 9–48.
76. Ioffe, S.; Szegedy, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR* **2015**, *abs/1502.03167*, [1502.03167].
77. Baylor, D.; Breck, E.; Cheng, H.T.; Fiedel, N.; Foo, C.Y.; Haque, Z.; Haykal, S.; Ispir, M.; Jain, V.; Koc, L.; others. TFX: A TensorFlow-based production-scale machine learning platform. Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 1387–1395.
78. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* **2019**, *1*.
79. Schmidt, P.; Biessmann, F. Quantifying Interpretability and Trust in Machine Learning Systems. *arXiv preprint arXiv:1901.08558* **2019**.
80. Schölkopf, B.; Smola, A.J. *Learning with kernels: support vector machines, regularization, optimization, and beyond*; MIT press, 2002.
81. Wolpert, D.H. The Lack of a Priori Distinctions Between Learning Algorithms. *Neural Comput.* **1996**, *8*, 1341–1390. doi:10.1162/neco.1996.8.7.1341.
82. Müller, K.R.; Mika, S.; Rätsch, G.; Tsuda, K.; Schölkopf, B. An introduction to kernel-based learning algorithms. *IEEE transactions on neural networks* **2001**, *12*, 181–201.
83. Zhang, J.M.; Harman, M.; Ma, L.; Liu, Y. Machine Learning Testing: Survey, Landscapes and Horizons. *CoRR* **2019**, *abs/1906.10742*, [1906.10742].
84. Hutter, F.; Kotthoff, L.; Vanschoren, J., Eds. *Automated Machine Learning - Methods, Systems, Challenges*; The Springer Series on Challenges in Machine Learning, Springer, 2019. doi:10.1007/978-3-030-05318-5.
85. 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*; Cortes, C.; Lawrence, N.D.; Lee, D.D.; Sugiyama, M.; Garnett, R., Eds.; Curran Associates, Inc., 2015; pp. 2962–2970.
86. Zoph, B.; Le, Q.V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* **2016**.
87. 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.
88. 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. *International Journal of Engine Research* **2020**. doi:10.1177/1468087420974148.
89. 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.; Curran Associates, Inc., 2014; pp. 3581–3589.
90. Chapelle, O.; Schlkopf, B.; Zien, A. *Semi-Supervised Learning*, 1st ed.; The MIT Press, 2010.
91. 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.; Curran Associates, Inc., 2014; pp. 3320–3328.

92. 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, 2001; pp. 682–688.
93. 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.
94. Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. A Survey of Model Compression and Acceleration for Deep Neural Networks. *CoRR* **2017**, *abs/1710.09282*, [1710.09282].
95. Frankle, J.; Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635* **2018**.
96. 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. *CoRR* **2019**, *abs/1907.11900*, [1907.11900].
97. Rokach, L. Ensemble-based classifiers. *Artificial Intelligence Review* **2010**, *33*, 1–39.
98. Zhou, Z.H.; Wu, J.; Tang, W. Ensembling neural networks: many could be better than all. *Artificial intelligence* **2002**, *137*, 239–263.
99. Opitz, D.; Maclin, R. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research* **1999**, *11*, 169–198.
100. Lakshminarayanan, B.; Pritzel, A.; Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in Neural Information Processing Systems, 2017, pp. 6402–6413.
101. Gal, Y.; Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. international conference on machine learning, 2016, pp. 1050–1059.
102. Pineau, J. The Machine Learning Reproducibility Checklist, 2019. Accessed: 2019-06-11.
103. Tatman, R.; VanderPlas, J.; Dane, S. A Practical Taxonomy of Reproducibility for Machine Learning Research **2018**.
104. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep reinforcement learning that matters. Thirty-Second AAAI Conference on Artificial Intelligence, 2018.
105. Sculley, D.; Snoek, J.; Wiltschko, A.; Rahimi, A. Winner's Curse? On Pace, Progress, and Empirical Rigor, 2018.
106. Bouthillier, X.; Laurent, C.; Vincent, P. Unreproducible Research is Reproducible. Proceedings of the 36th International Conference on Machine Learning; Chaudhuri, K.; Salakhutdinov, R., Eds.; PMLR: Long Beach, California, USA, 2019; Vol. 97, *Proceedings of Machine Learning Research*, pp. 725–734.
107. Vartak, M.; Subramanyam, H.; Lee, W.E.; Viswanathan, S.; Husnoo, S.; Madden, S.; Zaharia, M. ModelDB: A System for Machine Learning Model Management. Proceedings of the Workshop on Human-In-the-Loop Data Analytics; ACM: New York, NY, USA, 2016; HILDA '16, pp. 14:1–14:3. doi:10.1145/2939502.2939516.
108. Zhou, Z.Q.; Sun, L. Metamorphic Testing of Driverless Cars. *Commun. ACM* **2019**, *62*, 61–67. doi:10.1145/3241979.
109. Tian, Y.; Pei, K.; Jana, S.; Ray, B. DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars. Proceedings of the 40th International Conference on Software Engineering; ACM: New York, NY, USA, 2018; ICSE '18, pp. 303–314. doi:10.1145/3180155.3180220.
110. Pei, K.; Cao, Y.; Yang, J.; Jana, S. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. Proceedings of the 26th Symposium on Operating Systems Principles; ACM: New York, NY, USA, 2017; SOSP '17, pp. 1–18. doi:10.1145/3132747.3132785.
111. Chan-Hon-Tong, A. An Algorithm for Generating Invisible Data Poisoning Using Adversarial Noise That Breaks Image Classification Deep Learning. *Machine Learning and Knowledge Extraction* **2019**, *1*, 192–204. doi:10.3390/make1010011.
112. Chakarov, A.; Nori, A.V.; Rajamani, S.K.; Sen, S.; Vijaykeerthy, D. Debugging Machine Learning Tasks. *CoRR* **2016**, *abs/1603.07292*.
113. 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.
114. Baehrens, D.; Schroeter, T.; Harmeling, S.; Kawanabe, M.; Hansen, K.; Müller, K.R. How to explain individual classification decisions. *The Journal of Machine Learning Research* **2010**, *11*, 1803–1831.
115. 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.
116. Hois, J.; Theofanou-Fuelbier, D.; Junk, A.J. How to Achieve Explainability and Transparency in Human AI Interaction. HCI International 2019 - Posters; Stephanidis, C., Ed.; Springer International Publishing: Cham, 2019; pp. 177–183.
117. Thrun, M.C.; Ultsch, A.; Breuer, L. Explainable AI Framework for Multivariate Hydrochemical Time Series. *Machine Learning and Knowledge Extraction* **2021**, *3*, 170–204. doi:10.3390/make3010009.
118. 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! *Journal of Machine Learning Research* **2019**, *20*, 1–8.
119. Nori, H.; Jenkins, S.; Koch, P.; Caruana, R. InterpretML: A Unified Framework for Machine Learning Interpretability, 2019, [arXiv:cs.LG/1909.09223].
120. Wu, C.J.; Brooks, D.; Chen, K.; Chen, D.; Choudhury, S.; Dukhan, M.; Hazelwood, K.; Isaac, E.; Jia, Y.; Jia, B.; others. Machine learning at Facebook: Understanding inference at the edge. 2019 IEEE International Symposium on High Performance Computer Architecture (HPCA). IEEE, 2019, pp. 331–344.

121. Sehgal, A.; Kehtarnavaz, N. Guidelines and Benchmarks for Deployment of Deep Learning Models on Smartphones as Real-Time Apps. *Machine Learning and Knowledge Extraction* **2019**, *1*, 450–465. doi:10.3390/make1010027.
122. Christidis, A.; Davies, R.; Moschoyiannis, S. Serving Machine Learning Workloads in Resource Constrained Environments: a Serverless Deployment Example. 2019 IEEE 12th Conference on Service-Oriented Computing and Applications (SOCA), 2019, pp. 55–63. doi:10.1109/SOCA.2019.00016.
123. Wang, M.; Deng, W. Deep visual domain adaptation: A survey. *Neurocomputing* **2018**, *312*, 135 – 153.
124. Sugiyama, M.; Krauledat, M.; Müller, K.R. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research* **2007**, *8*, 985–1005.
125. Heckemann, K.; Gesell, M.; Pfister, T.; Berns, K.; Schneider, K.; Trapp, M. Safe automotive software. International Conference on Knowledge-Based and Intelligent Information and Engineering Systems. Springer, 2011, pp. 167–176.
126. Berkenkamp, F.; Moriconi, R.; Schoellig, A.P.; Krause, A. Safe learning of regions of attraction for uncertain, nonlinear systems with gaussian processes. 2016 IEEE 55th Conference on Decision and Control (CDC). IEEE, 2016, pp. 4661–4666.
127. Derakhshan, B.; Mahdiraji, A.R.; Rabl, T.; Markl, V. Continuous Deployment of Machine Learning Pipelines. EDBT, 2019, pp. 397–408.
128. Fehling, C.; Leymann, F.; Retter, R.; Schupeck, W.; Arbitter, P. *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*; Springer, 2014. doi:10.1007/978-3-7091-1568-8.
129. 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. 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2018, pp. 658–664.