# An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions

Iris Figalist[1(✉)], Christoph Elsner[1], Jan Bosch[2], and Helena Holmström Olsson[3]

[1] Corporate Technology, Siemens AG, 81739 Munich, Germany
{iris.figalist,christoph.elsner}@siemens.com

[2] Department of Computer Science and Engineering, Chalmers University of Technology, Hörselgången 11, 412 96 Göteborg, Sweden
jan.bosch@chalmers.se

[3] Department of Computer Science and Media Technology, Malmö University, Nordenskiöldsgatan, 211 19 Malmö, Sweden
helena.holmstrom.olsson@mau.se

**Abstract.** Nowadays, machine learning (ML) is an integral component in a wide range of areas, including software analytics (SA) and business intelligence (BI). As a result, the interest in custom ML-based software analytics and business intelligence solutions is rising. In practice, however, such solutions often get stuck in a prototypical stage because setting up an infrastructure for deployment and maintenance is considered complex and time-consuming. For this reason, we aim at structuring the entire process and making it more transparent by deriving an end-to-end framework from existing literature for building and deploying ML-based software analytics and business intelligence solutions. The framework is structured in three iterative cycles representing different stages in a model's lifecycle: prototyping, deployment, update. As a result, the framework specifically supports the transitions between these stages while also covering all important activities from data collection to retraining deployed ML models. To validate the applicability of the framework in practice, we compare it to and apply it in a real-world ML-based SA/BI solution.

**Keywords:** Machine learning · Software analytics · Business intelligence

## 1 Introduction

A vast amount of data is produced by software-intensive systems every day. As a result, software providers often try to gain insights from data using software analytics [23] or business intelligence [24] (SA/BI) tools. As existing tools are typically quite generic and can often not provide the desired depth of product-specific and stakeholder-targeted information, there is often a need for customized software analytics or business intelligence (SA/BI) solutions that leverage the full potential of modern machine learning (ML) techniques.

However, as such solutions are used as internal systems for monitoring or decision-making, these are often not perceived as something of direct customer value by managers. This results in a lack of priority, time and, resources assigned to setup and maintain ML-based SA/BI solutions [15]. In addition to that, the effort of going beyond a prototypical analysis and deploying it to and maintaining it in production is perceived as extremely high [15,30]. Paired with a lack of expertise in this domain, which is often the case if the actual product is not related to ML [6], custom ML-based SA/BI solutions are rarely deployed in production [15]. Nevertheless, this is considered crucial in order to continuously gain valuable insights and use it for actual decision making [21].

To address this, we conduct a literature review of important domains related to ML, specifically data management and processing, model building, and model deployment. The results are then used to derive a framework for building end-to-end ML-based SA/BI solutions consisting of three iterative cycles: a prototyping cycle, a deployment cycle, and an update cycle. To validate the applicability of the framework in practice, we compare it to and apply it in a real-world, customized ML-based SA/BI solutions.

The contribution of this paper is an end-to-end approach that covers all steps from data collection to retraining deployed ML models while at the same time taking the different conceptual stages into consideration (prototypical, deployment, and update). By specifically addressing the transition between these stages, our framework supports practitioners in advancing their prototypical analysis to a deployed and continuously retrained ML model.

The remainder of this paper is structured as follows: First, we outline the background of our study in Sect. 2. In Sect. 3 we provide an overview of the research method and the study design. The results of the literature review are presented in Sect. 4, before introducing the framework in Sect. 5. The framework validation is outlined in Sect. 6, followed by a conclusion in Sect. 7.

## 2 Background

The term software analytics (SA) describes analytics performed on software data to generate valuable insights for various stakeholders, from managers to software engineers, that ultimately support their decision making [6,23]. Related to this, the field of business intelligence (BI), sometimes also referred to as business analytics, applies data mining techniques to operational data in order to derive high-quality information for managerial decision making [24].

With its increase in popularity, artificial intelligence soon became an integral part of SA and BI solutions [7,11]. As a result, many companies aim at getting the most out of their data by running ML-based analyses on it. While some knowledge can be extracted using out-of-the-box tools, more in-depth analyses often require custom ML solutions.

In many cases, these custom solutions start out as a prototypical analysis or a proof of concept [15,30]. However, in order to make actual use of the results, they need to be provided in a continuous manner by deploying the model to production and retraining the model on a regular basis [21]. Precisely this is the point at which custom ML-based SA/BI solutions often get stuck. In a previous study [15], we identified a vicious circle that frequently prevents an end-to-end implementation of such analyses. One of the key issues is that an ineffective prototypical analysis can often not prove the value that it could deliver in production, leading to a lack of priority, time, and resources assigned to the topic [15].

Moreover, in the context of SA and BI there is often a lack of expertise in data engineering, data analytics, and in building an infrastructure for both [6,15]. For this reason, the framework presented in our study aims at compensating this to some extent by providing a structured approach for the transition between prototypical analysis and productively usable analysis.

## 3 Research Method and Study Design

As an end-to-end development of ML-based SA/BI solutions requires broad knowledge that is distributed across several, well-researched domains, we selected a deductive research approach for our study. Deductive approaches rely on existing theories for building hypotheses which are then confirmed or rejected using real-world observations [28]. The overall research process is outlined in Fig. 1.

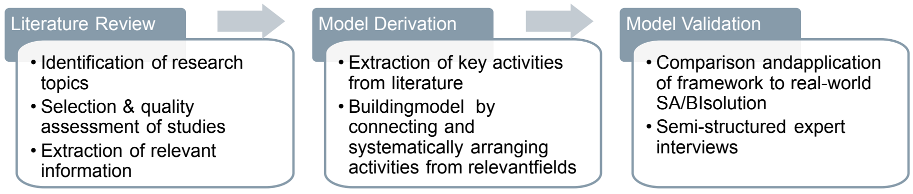

Literature Review
• Identification of research topics
• Selection & quality assessment of studies
• Extraction of relevant information
Model Derivation
• Extraction of key activities from literature
• Buildingmodel by connecting and systematically arranging activities from relevantfields
Model Validation
• Comparison andapplication of framework to real-world SA/BIsolution
• Semi-structured expert interviews

**Fig. 1.** Research process

As a first step, we conducted a literature review [18] which serves as the foundation for our study. Based on the requirements of our framework, we identified three overarching categories that comprise the results of our review: data management and processing, model building, and model deployment. To achieve our research goal, we queried common scientific libraries (IEEXplore, ACM Digital Library, ScienceDirect, Springer Link) using search terms related to the respective categories: data (quality, cleaning, preprocessing, transformation, management, continuous extraction) and machine learning model (training, evaluation, deployment [pipeline], management, serving).

As inclusion criteria we defined 1) research papers that outline approaches and/or challenges in data management and processing, model building, or model deployment; and 2) case studies and experience reports describing concrete actions and processes for at least one of the categories. We excluded non-scientific contributions (e.g. posters or presentations/talks) and studies that were not written in English.

Next, we extracted all mentioned activities and challenges out of each selected paper and accumulated the results to common activities and challenges based on the frequency of occurrences. In order to derive a framework for productively applying ML in SA/BI solutions, we merged and systematically arranged the key activities of the investigated domains.

To validate the applicability of the framework in practice, we first compare it to the current state of a real-world ML-based SA/BI solution being developed for an industrial platform provider. In a second step, we utilize the framework to strategically plan and direct the upcoming activities. To achieve this, we collaborated with two software architects and two product managers of the platform. The product managers are the future user of the system and, therefore, provided us with a specific use case while the software architects supported us in building the ML-based SA/BI solution. The platform itself is based on Amazon Web Services[1] (AWS). For this reason, we utilize existing AWS services for implementing and deploying our solution. In order to get a comprehensive picture of all the activities, we interviewed the stakeholders in several recap sessions to gain a detailed understanding of individual steps that we could not directly be involved in due to company processes.

## 4 Literature Review

### 4.1 Data Management and Processing

The most important prerequisite for training accurate ML models is providing high-quality training data [26,29]. At the same time, assembling high-quality data sets, and engineering and selecting appropriate features based on it, is very time-consuming and requires a vast amount of effort and resources [14].

As a result, we investigate the common activities (see Table 1) in data management and data processing required for a successful application in machine learning systems as well as the challenges (see Table 2) that come with these activities. The identified activities can be grouped into six overarching categories: 1) Data preparation; 2) data cleaning; 3) data validation; 4) data evaluation; 5) data serving; and 6) extract, transform, and load (ETL) tasks.

During the *data preparation*, raw input data is examined for suitable features before being transformed (e.g. aggregations of one or more raw input data fields) into training data [4,5,14,21,26,27]. Next, the data is *cleaned* by filtering out uncorrelated data [10,26], specifying quality rules, detecting errors, inconsistencies and anomalies [4,8,19], and fixing these errors [8,19,26,36].

To guarantee a successful preparation and cleaning of the data, each batch of data needs to be *validated* based on its properties [4,5,26,27,29,36] and potential

[1] https://aws.amazon.com/.

dependencies [26], deviations [5,26], or impact of features on model accuracy or performance [14,26] need to be identified.

Once a model is trained, the goal of *data evaluation* is to evaluate the choice and encoding of the data based on the results produced by a model trained on the data, for instance by performing sanity checks [14,26]. After a suitable solution was found, the newly emerging input data needs to be transformed to so-called *serving data* which is processible by the model [4,26]. This usually involves the same transformation steps as required for the training data. After the serving data was successfully processed by the model, it is channeled back as training data for future iterations [26].

In order to execute the aforementioned steps in an iterative and continuous manner, automated ETL tasks need to be set up. This involves the extraction of data from its source, transporting it to a processing pipeline, transforming it to target values, and finally making it accessible to and loadable by respective machine learning models [13,34,35].

Table 1 provides a detailed overview of common activities in data management and data processing grouped into six categories.

**Table 1.** Common activities in data management and processing for machine learning

| Activity | Publications |
|---|---|
| *Data preparation* | |
| Identification of features and their properties based on raw data | [14,21,26] |
| Transformation of input data to training data | [4,5,14,26,27] |
| *Data cleaning* | |
| Investigating and understanding effect of cleaning data on model accuracy & filtering out uncorrelated data | [10,26] |
| Ensure data quality, specification of (quality) rules & actions for rules | [4,8,19] |
| Detection of data errors | [8,19] |
| Definition of data fixes & execution of error repairs | [8,19,26,36] |
| *Data validation* | |
| Triggering validation pipeline for each batch of data | [5,29,36] |
| Generation of descriptive statistics of data, checking data properties based on specified schema/patterns & identification of errors or anomalies in training data | [4,5,26,27,29,36] |
| Identification of features with significant impact on model accuracy | [14,26] |
| Identification of dependencies to other data sources or infrastructure | [26] |
| Comparison of training and serving data to identify potential deviations | [5,26] |
| *Data evaluation* | |
| Performing sanity checks on data | [26] |
| Evaluation of choice and encoding of data based on model results | [14,26] |

(*continued*)

**Table 1.** (*continued*)

| Activity | Publications |
|---|---|
| *Data serving* | |
| Transformation of serving input data to serving data processible by model | [4,26] |
| Channeling serving data back as training data | [26] |
| *Extract, transform, load (ETL)* | |
| Extraction of data from sources | [13,34,35] |
| Transportation of data to processing pipeline (e.g. for data cleaning or filtering) | [13,34,35] |
| Transformation of source data to target values | [13,34,35] |
| Loading of cleaned & transformed data | [13,34,35] |

As a natural consequence, these activities also entail a couple of challenges which are presented in Table 2 and mostly related to 1) data understanding; 2) data preparation; 3) data cleaning; and 4) data validation.

**Table 2.** Common challenges in data management and processing

| Category | Challenge |
|---|---|
| DU | Set expectations of data; How to know something (e.g. a distribution) is "right"? [26] |
| DU | Analysis of features in conjunction [26] |
| DU | Understanding if data reflects reality [26] |
| DU | Identification of sources of data errors [26] |
| DC | Dealing with data inconsistency, missing features, unit changes,... [26], [29], [36] |
| DC & DV | Dealing with dynamic data environments (constantly changing constraints) [10], [29], [36] |
| DV | Formulation of understandable and actionable alerts [4], [26] |
| DP | Engineering set of features most predictive of the label [26] |
| DP | Unused data due to data overload / too much data to be processed [14] |
| DP | Feature experiments (e.g. different combinations of input features to examine their predictive value) affect multiple stakeholders (e.g. software or site reliability engineers responsible for pipeline) [26] |
| DP & DC | Merging data from multiple sources & deal with unstructured data [8], [10], [13], [16], [29] |
| DP, DC & DV | Achieving scalability of data processing and error detection in distributed settings [8], [10], [19] |

Data understanding = DU, data cleaning = DC, data validation = DV, data preparation = DP

## 4.2 Model Building

In the model building phase, one or multiple models are prepared, built, and evaluated based on the previously generated input features. This is typically an iterative process that involves running an analysis, evaluating the results, and adapting or optimizing parameters and input features until an adequate solution is found [14,22,33]. Table 3 outlines common activities that are part of this process.

**Table 3.** Common activities in model preparation, building, and evaluation

| Activity | Publications |
|---|---|
| *Model preparation* | |
| Selection of appropriate analysis/model type | [14,17,21,27] |
| Selection of input features | [4,12,14,21] |
| *Model building* | |
| Splitting input data into training and test set | [14,21] |
| Model training on training data | [4,14,21,25,27,31,33] |
| Application of model to test data | [4,14,21,31,33] |
| *Model evaluation* | |
| Quality evaluation based on test results (e.g. accuracy, precision, recall, F1-score) | [4,14,21,25,27,31,33] |
| Decision: accept or rework model (e.g. by adapting input features or model parameters) | [3,14,21] |

Initially and based on the respective problem to solve, appropriate analysis techniques and model types need to be selected as part of the *model preparation* [14,17,21,27]. For instance, if the input data is labeled and the goal is to classify data according to these labels, a supervised ML technique (e.g. logistic regression, support vector machines etc.) can help to achieve this. On the other side, if the requirement is to group unlabeled objects by their similarity, an unsupervised approach (e.g. k-means clustering, hierarchical clustering etc.) is the better choice.

Oftentimes, it is not advisable to use all available features as input features for the selected model as this can create noise and cause a decrease in model accuracy [14]. This results in the need for feature selection techniques that aim at identifying the most relevant input features for a given model [4,12,14,21].

Once the input data is filtered according to the determined feature relevance, a training and a test data set need to be created as part of the *model building* phase [14,21]. In a next step, the training set is used to train a model that was selected to solve a specific problem [4,14,21,25,27,31,33]. Before being able to validate the quality of the model, it is applied on the test set to investigate how it performs on previously unseen data [4,14,21,31,33].

Consequently, the results of this step can be used for the *model evaluation*. There are several metrics that support practitioners in assessing the quality of their models, for instance by calculating the accuracy, precision, recall, or F1-score [4,14,21,25,27,31,33]. Based on the evaluation results, the model can either be accepted as a reasonable solution or it needs to be reworked, for example by adapting the parameters or input features that are used for training the model [3,14,21].

This iterative process is often accompanied by several challenges. Table 4 summarizes a few of these challenges that we feel like are most important for our study. The presented challenges are categorized in *model preparation*, *model building*, and *model evaluation*.

**Table 4.** Common challenges in model preparation, building, and evaluation

| **Category** | **Challenge** |
|---|---|
| MP | Selecting appropriate model types for a specific problem [1], [14], [20] |
| MP | Dealing with too many (irrelevant) input features [14] |
| MP | Coordination and communication of involved stakeholders (e.g. ML specialists, software engineers,...) [1], [2], [20] |
| MB | Avoidance of overfitting [14] |
| MB | Debugging of ML models [1], [2], [32] |
| ME | Defining quality specifications (e.g. "when is the prediction quality good enough?", "is the model save to serve?" [4] |

Model preparation = MP, model building = MB, model evaluation = ME

## 4.3 Model Deployment and Serving

In order to fully leverage the benefits of ML to gain valuable insights, it is crucial to go beyond prototypical analyses by deploying models in production where they are actually used [21]. It has even been observed that "organizations that make the most of machine learning are those that have in place an infrastructure that makes experimenting with many different learners, data sources, and learning problems easy and efficient" [14].

For that reason, we summarize the common activities in model deployment and model serving in Table 5.

Deployment infrastructures for ML models often consist of multiple *components* each responsible for a specific task and executable as an automated

**Table 5.** Common activities in model deployment and model serving

| Activity | Publications |
|---|---|
| *Components* | |
| Component for model validation (before serving & often coupled with data validation) | [4, 9, 25] |
| Component for continuous model evaluation & monitoring (performance, quality,...) | [4, 9, 25, 27] |
| Component or serving solution to deploy model in production | [4, 9, 21] |
| Component for monitoring pipelines (checkpoint after each pipeline) | [27, 31] |
| *Setups* | |
| Setup model lifecycle management (to keep overview of deployed models) | [9, 33] |
| Setup workflow manager for job coordination | [9, 21] |
| *Process* | |
| Loading new model before unloading old model | [25] |
| Validation of model and serving infrastructure (incl. reliability checks) before pushing to production environment | [4, 9, 25] |
| Continuous application of model to (new) serving data | [9, 21, 27] |
| Continuous model evaluation / monitoring | [4, 9, 25, 27] |
| Rollback in case of errors | [25, 27] |
| Periodically update models | [9, 21] |

workflow coordinated by a workflow manager. Besides the component that handles the actual deployment of models in production [4,9,21], it is advisable to have additional components for validating the model before deployment [4,9,25], for continuously evaluating and monitoring the model after being deployed in production [4,9,25,27], and for monitoring if all pipelines are up and running as expected [27,31].

In addition to the components, a few *setups* are required for automating the deployments while keeping an overview of the deployed models. For one, a model lifecycle management should be set up that allows the comparison and monitoring of models over time and provides information on the currently deployed models [9,33]. For another, the jobs required to deploy a model can be coordinated an executed using a workflow manager [9,21]. After triggering the workflow, the model is updated in an automated manner and in case of errors a predefined rollback plan is executed.

In general, the *process* of model deployment and model serving requires the following steps which are typically encapsulated in respective components: First, a new model is loaded for deployment before unloading the old model [25]. In a next step, the model as well as the serving infrastructure are validated (e.g. reliability checks) before pushing the new model to the production environment [4,9,25].

Once the model is deployed to production, it can be used and continuously applied to newly emerging serving data [9,21,27]. In order to guarantee that the model works as expected, a continuous evaluation and monitoring of the model is required [4,9,25,27]. In case the model does not behave as expected, a rollback plan is executed and typically the current model is replaced by a previous well-working version of the model [25,27]. Following this process, models can be periodically updated and deployed to production [9,21].

Analogously to data management and model building, model deployment and model serving also entails several challenges. Four of the key challenges are presented in Table 6. The challenges are categorized into *infrastructure* and *model*-specific topics.

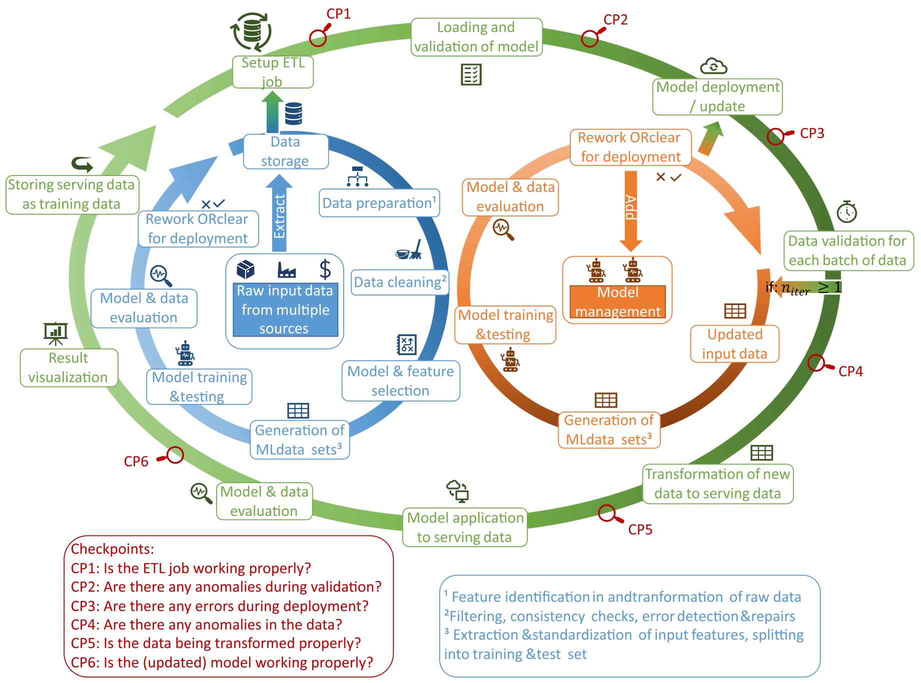
**Table 6.** Common challenges in model deployment and model serving

| **Category** | **Challenge** |
|---|---|
| I | Integration of third-party packages or tools [21], [30] |
| I | Brittle pipelines / "pipeline jungle" [21], [30] |
| M | Managing and monitoring multiple models [9], [30], [33] |
| M | Dealing with expected and unexpected variations during model evaluation [4], [30] |

Infrastructure = I, model = M

## 5 Framework Derivation

Based on the insights gained from the literature review, we derive a framework for supporting an end-to-end development and deployment of ML models in the context of software analytics and business intelligence (see Fig. 2).


CP1
Loading and validation of model
CP2
Setup ETL job
Model deployment / update
CP3
Data storage
Rework ORclear for deployment
Storing serving data as training data
Data preparation¹
Model & data evaluation
Extract
Rework ORclear for deployment
Add
Data validation for each batch of data
Raw input data from multiple sources
Data cleaning²
if: $n_{iter} \geq 1$
Model & data evaluation
Model training &testing
Model management
Updated input data
Result visualization
Model & feature selection
Model training &testing
Generation of MLdata sets³
Generation of MLdata sets³
CP4
Transformation of new data to serving data
CP6
Model & data evaluation
Model application to serving data
CP5

Checkpoints:
CP1: Is the ETL job working properly?
CP2: Are there any anomalies during validation?
CP3: Are there any errors during deployment?
CP4: Are there any anomalies in the data?
CP5: Is the data being transformed properly?
CP6: Is the (updated) model working properly?

¹ Feature identification in andtranformation of raw data
²Filtering, consistency checks, error detection&repairs
³ Extraction &standardization of input features, splitting into training &test set

**Fig. 2.** Framework for productively applying machine learning (Color figure online)

While the literature review examines the topics *data management and processing*, *model building*, and *model deployment and serving* individually, in reality a separation of the three is not that trivial. In fact, for building end-to-end solutions the fields are very much interrelated as the activities depend on each other and sometimes even overlap.

Oftentimes, ML projects start out as a *prototypical* analysis due to a limited amount of time and resources [15,30]. In order to use and actually benefit from the ML model, it needs to be *deployed* to a production environment which can be time and cost-intensive but nonetheless crucial [21,30]. To avoid the deployed models from being outdated, it is important to provide a functionality for dynamically deploying new models or iteratively *retraining and updating* existing models [9,21].

As a result, we identify three iterative cycles which are passed through during an end-to-end development of ML solutions and, therefore, serve as the main dimensions in our framework: 1) Prototyping cycle (blue), 2) deployment cycle (green), and 3) update cycle (orange).

## 5.1 Prototyping Cycle

In software analytics and business intelligence, relevant input data typically emerges from multiple sources that need to be extracted, set in relation, and stored in a common data storage [15]. As part of the data preparation, potential input features can be identified and extracted based on a snapshot of raw data [14,21,26]. To ensure a sufficient data quality, a couple of data cleaning activities need to be performed (e.g. filtering, consistency checks, or other error detection techniques and repairs) [4,8,19,26,36].

Depending on the overarching goal of the analysis, appropriate ML models have to be selected that are suitable to achieve a specific task [14,17,21,27]. Based on the selected model, it is recommended to apply feature selection techniques to the input data set to identify a subset of the most relevant input features [4,12,14,21]. This subset is then extracted from the overall data set and the values of each input features are standardized. Before training the model, the respective subset should be split into a training and a test data set (usually 70%/30% or 80%/20%) [14,21].

Training the model using training data and testing it on test data, allows an examination of how well the model performs on previously unseen data [4], [14,21,31,33]. Based on the test results, the choice of model parameters and input features should be evaluated [4,14,21,25,27,31,33]. If the evaluation indicates a decent quality of the model (e.g. based on accuracy, precision, recall, and F1-score), it can be cleared for deployment. Otherwise, the cycle is run through again and the model is reworked until its quality reaches a desired level.

## 5.2 Deployment Cycle

Taking a ML model to production involves much more than only model deployment. For one, a ETL job needs to be set up that continuously extracts, transforms, and loads the latest input and serving data [13,34,35]. Before a new model is deployed, it is loaded and validated to avoid errors of faulty behavior in the production environment [4,9,25].

For each batch of new data a data validation pipeline is triggered that checks the data for anomalies [5,29,36]. After one or more iterations, the update cycle can be entered at this point in case the model needs to be retrained which is evaluated during the model evaluation later on.

Analogous to the training data, the new input data is transformed to serving data [4,26]. This involves data cleaning, feature extraction and standardization. In a next step, the model can be applied to the new and preprocessed data [9,21,27].

Since both data and model behavior evolves over time, it is crucial to continuously evaluate the model performance and its input data [4,9,25,27]. If the model does not perform as expected, a retraining of the model is triggered for the next iteration [9,21].

In addition to this, the results of the analysis need to be visualized. By explaining the results as intuitively as possible, users of SA/BI solution will be able to understand and interpret the results and turn it into actionable insights [15]. In the last step before the cycle is repeated from the beginning, the recently processed serving data is channeled back as training data which will be included in upcoming retrainings of the model [26].

Since these steps are typically automated in one or multiple pipelines, it is important to implement several checkpoints along the way, that continuously monitor and check whether each task is working properly [27,31]. In case of errors or anomalies, an alert should be sent to the respective stakeholder.

### 5.3 Update Cycle

As a natural consequence of constantly evolving data, the model's accuracy can start to decrease some time after being deployed [21]. As soon as this is detected during the model evaluation in the deployment cycle, a retraining of the model will be triggered after the upcoming data validation.

An updated data set is created that consists of the initial training data as well as the new serving data that was channeled back as training data [26]. Analogous to the initial training only the relevant input features are extracted, standardized and split into a training and a test set. The model is retrained based on the new training set and tested on the test set respectively [4,14,21,31,33].

Based on the results, the model's input features and parameters needs to be evaluated before deciding whether to improve the model's quality in an additional iteration or whether to clear it for deployment and add both the current and the new model to the model management [4,14,21,25,27,31,33]. The latter enables a clear overview of all models and allows an easy rollback in case of erroneous behavior in production.

## 6 Framework Validation

In order to validate the applicability of our framework in practice, we use a real-world ML-based SA/BI solution that is currently being developed for an industrial platform provider to 1) compare the activities of the framework to the actual activities executed in practice; and 2) to strategically plan and direct upcoming activities to finalize the end-to-end implementation.

### 6.1 Current Status

At the beginning of our collaboration, the product managers were interested in running customized analyses on their customers' usage data. Specifically, whenever a customer's action triggers a request to the platform or one of its applications, it is tracked in the platform and app usage logs. The platform itself is based on AWS. Therefore, we decided to setup the custom ML-based SA/BI solution using the existing AWS infrastructure and services.

Currently, the platform and app usage logs produce 100 GB of data every day. For this reason, the data is aggregated and stored in a compressed format (26 GB per day) in AWS S3 buckets[2]. The log data is available for the past 1.5 years and, in addition to that, we also have access to the sales data that keeps track of which customer purchased what kind of licenses.

[2] https://aws.amazon.com/s3/.

**Prototyping Cycle.** As the future users of the system, the product managers were interested in analyzing customer churn for the applications hosted on the platform as a first use case.

In the beginning, we focus on one specific application to build a first prototype. Therefore, we identify potential input features based on the information that is available in the logs (e.g. user id, http status code, relative URL path) and pre-filter the data by the selected application. We setup a script that extracts and aggregates the input features on customer level ($n = 174$) while enriching and labeling it with the sales data (binary label for churn/non-churn).

Next, we applied several different supervised ML models (support vector machines, decision trees, logistic regression, neural network) to the data set in an iterative manner. We apply principal component analysis to the standardized input data in order to identify the most relevant subset of features, before splitting the data set into a training and a test set.

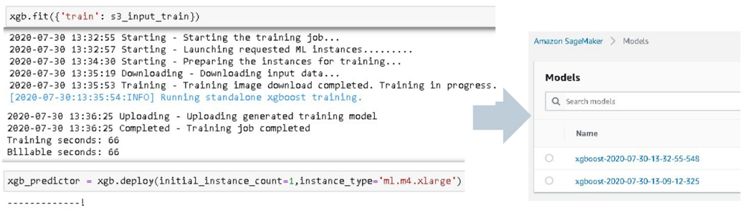
Based on this, the models were trained on the training data and tested using the test set. It took us several iterations and experiments with different models, model parameters and input features before ending up with the final model that is being deployed in the upcoming step.

**Deployment Cycle.** At the current state of our ML-based SA/BI solution, we have not yet completed the deployment cycle. Before being able to deploy the model, we had to come up with an efficient, reliable and robust solution for handling the enormous amounts of data (100 GB per day). It was a complex task to get an overview of the data, to define which fields to keep for long-term storage, and finally to specify the format to store it in.

After coming up with a concept for this, one of the software architects setup pipelines that continuously extract, transform, compress and load the latest log data into a S3 bucket using the specified format. In addition to that, he also setup data validation pipelines that check each batch of new data for anomalies and inconsistencies. It is important to continuously monitor all pipelines. During one of the interviews, the software architect explains that "we have to make sure the pipelines are not failing for whatever reasons and if they're failing we're notified and can restart them". Moreover, they need to ensure that "the pipeline elements that are doing the preprocessing are always up and triggered at appropriate times". The software architect also notes that it "requires a lot of engineering effort to keep the pipeline running in a correct manner."

After the data pipelines are set up, we load and deploy our ML model using Amazon SageMaker[3]. The SageMaker modules for Python offer out-of-the-box functionalities for deploying ML models to an AWS instance that are accessible via an API (see Fig. 3).

[3] https://aws.amazon.com/sagemaker/.



```
xgb.fit({'train': s3_input_train})

2020-07-30 13:32:55 Starting - Starting the training job...
2020-07-30 13:32:57 Starting - Launching requested ML instances.........
2020-07-30 13:34:30 Starting - Preparing the instances for training...
2020-07-30 13:35:19 Downloading - Downloading input data...
2020-07-30 13:35:53 Training - Training image download completed. Training in progress.
[2020-07-30:13:35:54:INFO] Running standalone xgboost training.

2020-07-30 13:36:25 Uploading - Uploading generated training model
2020-07-30 13:36:25 Completed - Training job completed
Training seconds: 66
Billable seconds: 66

xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')

-------------!
```

Amazon SageMaker > Models
Models
Search models
Name
xgboost-2020-07-30-13-32-55-548
xgboost-2020-07-30-13-09-12-325

**Fig. 3.** Model deployment in Amazon SageMaker

**Findings - current status**: The activities identified in literature are consistent with the activities we had to perform for successfully implementing our prototype, setting up continuous data extraction, processing and validation pipelines, and deploying our model; it is important to accept that prototyping is an iterative process; continuous checkpoints after each automated task are crucial

## 6.2 Planning and Evolution

In order to use the deployed model to make actual predictions, we now plan and execute the remaining steps following the presented framework.

**Deployment Cycle.** In the upcoming step, the model is applied to new serving data. This constitutes a bit of a challenge as up until now all extracted data is stored in the same S3 bucket. As a result, we now need to create an additional S3 bucket for storing the serving data. In order to preprocess the newly emerged data to serving data, we can reuse the script created during prototyping for transforming and extracting the input features out of the raw data.

In order to continuously evaluate the model in production, we setup Amazon SageMaker's Model Monitor that provides summary statistics, detects concepts drifts and indicates when a model needs to be retrained. In order to perform potential retrainings on the newest data available, we transfer the data from the serving S3 bucket to the training S3 bucket once it was processed by the model. Lastly, we plan to visualize the results in Amazon QuickSight[4].

**Update Cycle.** For continuous updates of models, AWS offers the Step Functions Data Science SDK[5] for Amazon Sagemaker to automate model retraining and deployment. An ETL job is setup to extract and preprocess the latest data. Following this, a new model is trained and evaluated. If the model accuracy is above a certain threshold (e.g. 90%), a new endpoint is created for deployment and the model is added to the model management.

[4] https://aws.amazon.com/quicksight/.
[5] https://docs.aws.amazon.com/step-functions/latest/dg/concepts-python-sdk.html.

**Findings - planning and evolution**: the framework supported us in keeping an overview of remaining tasks; by following the cycles and activities in the framework the definition of the roadmap and next steps was efficient and easy; we were able to quickly identify errors or missing components in our original approach (storage of training and serving data)

## 7 Conclusion

Gaining customized insights on product or usage behavior can be a valuable asset for many stakeholders involved in software-intensive businesses which results in a need for ML-based SA/BI solutions. Building and, more importantly, deploying and maintaining such solutions is, however, time-consuming, complex and burdensome as it requires knowledge from several different domains.

For this reason, we scanned existing literature on data management and processing, model building, and model deployment to derive a framework that comprises all key activities from data collection to retraining of deployed models. In addition to that, our framework is structured in three iterative cycles: a prototyping cycle, deployment cycle, and an update cycle. These cycles resemble stages in the lifecycle of a ML model and by outlining the transitions between stages, our framework specifically guides the journey from a prototypical analysis to a productively running ML model.

The results of the validation indicate that the activities of the framework are consistent with the activities performed in practice. Moreover, the framework is a practical tool to keep an overview of all required steps and to efficiently define and plan upcoming activities. Moreover, we observed that the separation of activities across the conceptual phases creates the perception that the overall, potentially overwhelming process now consists of several smaller ones that are easier to handle.

One limitations of our study is the development state of our ML-based SA/BI solution. As we are still in the process of implementing parts of the deployment and update cycle, we are only partially able to compare the framework's activities to the activities executed in practice. Further research could, therefore, be dedicated to a long-term validation of the framework based on already established SA/BI solutions and to identifying remaining challenges and needs for more in-depth guidance by practitioners to adapt the framework to their needs.

## References

1. Amershi, S., et al.: Software engineering for machine learning: a case study. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), pp. 291–300. IEEE (2019)
2. Arpteg, A., Brinne, B., Crnkovic-Friis, L., Bosch, J.: Software engineering challenges of deep learning. In: 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 50–59. IEEE (2018)

3. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach. Learn. **36**(1–2), 105–139 (1999). https://doi.org/10.1023/A:1007515423169
4. Baylor, D., 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, pp. 1387–1395 (2017)
5. Breck, E., Polyzotis, N., Roy, S., Whang, S.E., Zinkevich, M.: Data validation for machine learning. In: Conference on Systems and Machine Learning (2019)
6. Buse, R.P., Zimmermann, T.: Information needs for software development analytics. In: 34th International Conference on Software Engineering, pp. 987–996. IEEE (2012)
7. Chen, H., Chiang, R.H., Storey, V.C.: Business intelligence and analytics: from big data to big impact. MIS Q. **36**, 1165–1188 (2012)
8. Chu, X., Ilyas, I.F., Krishnan, S., Wang, J.: Data cleaning: overview and emerging challenges. In: Proceedings of the 2016 International Conference on Management of Data, pp. 2201–2206 (2016)
9. Crankshaw, D., et al.: The missing piece in complex analytics: low latency, scalable model management and serving with velox (2015)
10. Cuzzocrea, A., Song, I.Y., Davis, K.C.: Analytics over large-scale multidimensional data: the big data revolution! In: Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP, pp. 101–104 (2011)
11. Dam, H.K., Tran, T., Ghose, A.: Explainable software analytics. In: Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results, pp. 53–56 (2018)
12. Dash, M., Liu, H.: Feature selection for classification. Intell. Data Anal. **1**(3), 131–156 (1997)
13. Dayal, U., Castellanos, M., Simitsis, A., Wilkinson, K.: Data integration flows for business intelligence. In: Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, pp. 1–11 (2009)
14. Domingos, P.: A few useful things to know about machine learning. Commun. ACM **55**(10), 78–87 (2012)
15. Figalist, I., Elsner, C., Bosch, J., Olsson, H.H.: Breaking the vicious circle: Why AI for software analytics and business intelligence does not take off in practice. In: 46th Euromicro Conference on Software Engineering and Advanced Applications. IEEE (2020)
16. Hernández, M.A., Stolfo, S.J.: Real-world data is dirty: data cleansing and the merge/purge problem. Data Min. Knowl. Disc. **2**(1), 9–37 (1998). https://doi.org/10.1023/A:1009761603038
17. Jordan, M.I., Mitchell, T.M.: Machine learning: trends, perspectives, and prospects. Science **349**(6245), 255–260 (2015)
18. Keele, S.: Guidelines for performing systematic literature reviews in software engineering. Technical report, Version 2.3 EBSE Technical Report (2007)
19. Khayyat, Z., et al.: BigDansing: a system for big data cleansing. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 1215–1230 (2015)
20. Kim, M., Zimmermann, T., DeLine, R., Begel, A.: Data scientists in software teams: State of the art and challenges. IEEE Trans. Softw. Eng. **44**(11), 1024–1038 (2017)
21. Lin, J., Kolcz, A.: Large-scale machine learning at twitter. In: Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data, pp. 793–804 (2012)

22. Lwakatare, L.E., Raj, A., Crnkovic, I., Bosch, J., Olsson, H.H.: Large-scale machine learning systems in real-world industrial settings a review of challenges and solutions. Inf. Softw. Technol. **127**, 106368 (2020)
23. Menzies, T., Zimmermann, T.: Software analytics: so what? IEEE Softw. **30**(4), 31–37 (2013)
24. Negash, S., Gray, P.: Business Intelligence. In: Handbook on Decision Support Systems 2. International Handbooks Information System. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-48716-6_9
25. Olston, C., et al.: Tensorflow-serving: flexible, high-performance ml serving. In: Workshop on ML Systems at NIPS (2017)
26. Polyzotis, N., Roy, S., Whang, S.E., Zinkevich, M.: Data lifecycle challenges in production machine learning: a survey. ACM SIGMOD Rec. **47**(2), 17–28 (2018)
27. Rajaram, S., Mishra, K., O'mara, M.: Finite state automata that enables continuous delivery of machine learning models, US Patent App. 16/229,020, April 2020
28. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case study research in software engineering: guidelines and examples. Wiley, Hoboken (2012)
29. Schelter, S., Lange, D., Schmidt, P., Celikel, M., Biessmann, F., Grafberger, A.: Automating large-scale data quality verification. Proc. VLDB Endow. **11**(12), 1781–1794 (2018)
30. Sculley, D.: Hidden technical debt in machine learning systems. In: Advances in neural information processing systems, pp. 2503–2511 (2015)
31. Sparks, E.R., Venkataraman, S., Kaftan, T., Franklin, M.J., Recht, B.: KeystoneML: Optimizing pipelines for large-scale advanced analytics. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 535–546. IEEE (2017)
32. Tata, S., et al.: Quick access: building a smart experience for google drive. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1643–1651 (2017)
33. Vartak, M., et al.: ModelDB: a system for machine learning model management. In: Proceedings of the Workshop on Human-In-the-Loop Data Analytics (2016)
34. Vassiliadis, P.: A survey of extract-transform-load technology. Int. J. Data Warehous. Min. (IJDWM) **5**(3), 1–27 (2009)
35. Vassiliadis, P., Simitsis, A.: Extraction, transformation, and loading. Encycl. Database Syst. **10**, 1–10 (2009)
36. Volkovs, M., Chiang, F., Szlichta, J., Miller, R.J.: Continuous data cleaning. In: 30th International Conference on Data Engineering, pp. 244–255. IEEE (2014)