Industrial Machine Learning for Enterprises

Deliverable D2.1

Baseline methods and techniques for data collection, processing, and valorisation

Project title:	IML4E
Project number:	20219
Call identifier:	ITEA AI 2020
Challenge:	Safety & Security

Work package:	WP2
Deliverable number:	D2.1
Nature of deliverable:	Report
Dissemination level:	PU
Internal version number:	1.0
Contractual delivery date:	2022-01-31
Actual delivery date:	2022-01-31
Responsible partner:	Software AG

Contributors

Editor(s)	Mohamed Abdelaal (Software AG)
Contributor(s)	Mohamed Abdelaal (Software AG), Timo Sinisalmi (Basware), Luca Szegletes (BUTE), Jürgen Großmann (Fraunhofer), Dorian Knoblauch (Fraunhofer), Abhishek Shrestha (Fraunhofer), Heikki Ihasalo (Granlund), Jukka Remes (Silo AI), Harry Souris (Silo AI), Kimmo Sääskilahti (Silo AI)
Quality assuror(s)	Dorian Knoblauch (Fraunhofer), Luca Szegletes (BUTE)

Version history

Version	Date	Description
1.0	22-01-31	Version for publication

Abstract

The overall objective of IML4E WP2 is to develop tools and techniques for data collection, processing, and valorisation to efficiently provide and manage data sets used for learning and training in different phases of a system life cycle. This deliverable introduces the industrial and scientific state of the art relevant to this objective. The structure of this deliverable is defined considering the main research tasks of WP2, which are to develop support for i) data preparation automation, ii) data management and version control, and iii) continuous data quality assurance. Additionally, we present the baseline for the WP2 research activities which represents a set of tools and techniques that may serve as promising starting points for further development. To identify the baseline, we exploited the conclusions drawn from the state of the art together with our pre-defined relevant research questions that are addressed by the IML4E project.

Keywords

Data preparation, data valorisation, anomaly detection, data version control, data quality assurance, ML certification, state of the art

Executive Summary

The overall objective of IML4E WP2 is to develop techniques and tools for automated data preparation, data version control, and risk-based validation. Such techniques and tools will be used alongside the automated model selection and model reuse tools, developed in WP3, to achieve fully automated machine learning (ML) pipelines. The purpose of this document is twofold. First, we give an overview of the state of the art that is relevant for the WP2 research objectives. Second, we introduce the IML4E WP2 baseline which is the existing tools and techniques that serve as a promising starting point for the research tasks.

The description of the state of the art and the baseline are organized according to the three main research tasks of WP2, which are to develop tools and techniques to support:

- data preparation automation;
- data version control, traceability, and documentation of data evolution; and
- a risk-based validation approach including a catalogue of formalized quality attributes dedicated to precisely validating the collected data as well as the developed ML-based software.

The discussion of the state of the art and the identification of the WP2 baseline are guided by the IML4E research questions that are relevant for this work package. These research questions are the following:

- How to enable automated processing with integrated quality assurance of data in the data preparation pipeline?
- How to improve the modularity and reuse of development and data artefacts, datasets and metadata that may serve the training of models in different application contexts, throughout the development process?
- How to enable flexible data versioning and traceability of development and data artefacts (data sets, models, parameters, test results) during data preparation, training, and operations?
- What are good methods and tools for continuous testing and verification of ML artefacts during development, reuse, and deployment?

Considering the challenges of developing fully automated data preparation pipelines that exploit existing artefacts, these are timely research questions. As shown in this deliverable, there exists no adequate support for the design of automated data preparation as well as continuous testing and verification tools. Most of the existing tools and methods suffer either from being limited to a certain data model, being not designed to deal with real-world big data, or overlooking important features such as data valorisation. While using the existing tools and techniques for data preparation and version control will provide some basic support, WP2 will develop novel techniques for data preparation, version control, and continuous verification to broadly support the automated modelling phase.

Table of contents

1 Introduction

One of the most time-consuming and tedious tasks in ML projects is typically data preparation. Relieving this burden requires the careful design of automated data preparation tools and techniques which can analyse the data, identify fixes, screen out fields, derive new attributes when necessary, and improve the predictive performance through intelligent screening techniques. In this case, automated data preparation enables to make the collected data ready for model training quickly and easily, without requiring prior knowledge of the statistical concepts involved. Moreover, automated data preparation improves the robustness of automated modeling processes. However, the task of designing a fully automated data preparation pipeline remains challenging due to diversity of data being processed and the ultimate need to involve domain knowledge.

Another challenge in most ML projects is how to efficiently manage and share the available data and artefacts. An efficient data management system has to help individuals, businesses, and associated things enhance the use of their data within the limits of strategy and regulation so that they can make decisions and take actions that make the most of the benefit to the companies. To achieve these objectives, organisations have to properly define strategies for dealing with continuously increasing data. Moreover, data security best practices have to be considered for smooth and secure data collection, storage and retrieval. Additionally, strategies for dealing with varying versions of the available artefacts, which should be tracked and managed, need to be incorporated in any efficient data management system. The latter issue enables organisations and companies to easily share and reproduce their models and data.

The objective of IML4E WP2 is to design efficient and effective techniques and tools for data collection, processing, and valorisation. This main objective includes designing tools and techniques for fully automated data cleaning, transformation, and valorisation. Such tools and techniques are planned to consider different data models collected from different sources at a large scale. To ensure reproducibility of models and results, WP2 also involves designing tools and techniques for efficient data version control and managing of the available data. Moreover, WP2 entails developing techniques, tools, and methods to systematically document and model the uncertainties and risks in ML models due to data quality issues and means to continuously test data quality over the whole ML-lifecycle.

This deliverable gives an overview of the state of the art within automated data preparation, data management, and continuous data quality assurance. Moreover, the deliverable provides the baseline for the R&D work of IML4E WP2. The baseline is the existing techniques and tools that are promising and may serve as a basis for the upcoming project work within WP2. The description of the state of the art and the identification of the baseline are guided by the relevant research questions that need to be addressed when tackling the WP2 research challenges, including:

- How to enable automated processing with integrated quality assurance of data in the data preparation pipeline?
- How to improve the modularity and reuse of development and data artefacts, datasets and metadata that may serve the training of models in different application contexts, throughout the development process?
- How to enable flexible data versioning and traceability of development and data artefacts (data sets, models, parameters, test results) during data preparation, training, and operations?
- What are good methods and tools for continuous testing and verification of ML artefacts during development, reuse, and deployment?

The remainder of this document is structured according to the main research tasks of WP2. Accordingly, we give in Section 2 an overview of the state of the art within data profiling; data validation; data cleaning for structured, unstructured, and hybrid data; and data valorisation (related to task T2.1 in the IML4E FPP). In Section 3, we provide the state of the art within data storage solutions, data processing and ML pipelines, and data version control (related to task T2.2). In Section 4, we introduce the state of the art within data quality-based uncertainties and risks in ML, and data quality for ML certification (related to task T2.3). Based on the state of the art and the IML4E research questions, we introduce the WP2 baseline in Section 5. Finally, we draw a conclusion to summarize the main ideas and research directions in Section 6.

2 Data Preparation Automation

In this section, we provide an overview of data preparation techniques and the state of the art for data cleaning, data profiling, data validation, and data valorisation. Data profiling represents a crucial step to extract a set of useful metadata that can broadly assist in the subsequent data preprocessing steps. For instance, several data validators leverage the extracted metadata to precisely detect the data discrepancies, before employing suitable data cleaning algorithms to replace or delete the noisy data samples. In this context, data valorisation stands as an important optimization technique to reduce the amount of irrelevant data samples, which in turn reduces the overall computational and space complexities.

The section is structured as follows. In Section 2.1, we elaborate on the different techniques for profiling the data for the sake of extracting useful metadata. Section 2.2 presents the most prominent data validation methods. In Section 2.3, we introduce the various data cleaning and manipulation techniques for structured, semi-structured, and unstructured data. Finally, Section 2.4 introduces the different approaches for data valorisation.

2.1 Data Profiling

In general, data profiling aids in understanding the data structure and interrelationships as well as finding a suitable dataset to satisfy a specific modeling requirement. It can be used in several processing steps of the data lifecycle. In a data quality process, data profiling can be applied for recognizing dependencies, redundancies, and data rules. There exists a broad range of data profiling tools, methods, and techniques. In a survey of data profiling, Abedjan et al. (Abedjan, et al., 2015) found 30 simple and advanced methods. Figure 1 demonstrates the various data profiling methods and their primary use cases. Among the simple methods are, for example, the estimation of the number of null values and finding the minimum and maximum values. More advanced methods include, for instance, the correlation and cluster analysis. There are details specific to big data (Liu & Zhang, 2020), but even middle size data can also gain a lot of value from data profiling.

	Database management	Data integration	Data cleansing	Database reverse engineering	Data exploration	Data analytics	Scientific data management
Single-column							
Cardinalities	✓				✓	✓	
Patterns and data types		✓	✓	✓			
Value distributions	✓		✓		✓	✓	
Domain classification		✓	✓	✓			✓
Multi-column							
Correlations					✓	✓	✓
Association rules						✓	✓
Clustering		✓			✓	✓	✓
Outliers			✓				✓
Summaries and sketches			✓		✓	✓	
Dependencies							
Unique column combinations	✓		✓	✓			
Inclusion dependencies	✓	✓		✓	✓		✓
Conditional inclusion dependencies		✓			✓	✓	✓
Functional dependencies	✓		✓	✓	✓		
Conditional functional dependencies			✓		✓	✓	

Figure 1. Data profiling tasks and their primary use-cases (Abedjan, et al., 2015)

A common example of data profiling is estimating the data correlation. Correlation analysis shows whether data are changing the same way, i.e., if they are inter-related. However, correlation does not mean causation, i.e., two values can be correlated, but they are not causing each other. For instance, the number of ice creams sold, and the number of people drown in the river can be correlated as both happen mostly in the summer, but they are not causing each other. In general, correlation is defined in terms of a value, referred to as the correlation

coefficient. To define such a coefficient, there exist in the literature several methods. However, the most used method is the Pearson correlation coefficient (Wikipedia, 2021). Equation 1 expresses the Pearson correlation coefficient in terms of the covariance matrix and the standard deviation of the random variables X, Y. Furthermore, the covariance can be defined in terms of the mean and expectation, as given in Equation 2, where μ_X , μ_Y represents the means of the random variables X, Y.

$$\boldsymbol{\rho_{X,Y}} = \frac{\boldsymbol{cov(X,Y)}}{\boldsymbol{\sigma_X \sigma_Y}} \quad (1)$$

$$cov(X,Y) = \mathrm{E}[(X - \mu_X)(Y - \mu_Y)] \quad (2)$$

2.2 Data Validation

Industrial ML projects de facto demands the adoption of data validation processes and tools. Data validation is not a unique challenge, there are many solutions outside the field of ML (Lwakatare, et al., 2020). However, the context of ML presents new problems that need to be solved, because ML software requires high-quality input data to accurately train a model. Missing or incorrect information compromises the output of a ML model and can result in serious performance degradation or even in unpredictable behaviour. Moreover, data bias may severely lead to discriminative ML models against minorities or women. Such data biases can occur due to wrong data gathering and lack of data validation. Accordingly, data validation takes place early in the machine learning pipeline, usually between data ingestion and data preprocessing because the cost of data error increases exponentially. In general, data errors can be defined as the data points different from a given ground truth, they come in many forms e.g., outliers, duplicates, rule, or pattern violations (Abedjan, et al., 2016). Data validation is the process of checking the quality of the input data. Research and industrial practice suggest the integration of data validation tools into the development process of ML projects instead of manually checking the quality of the data (Breck, et al., 2019).

Among the most prominent data validation tools stands Deequ which is a library built on top of Apache Spark for defining "unit tests for data" to measure the data quality in large datasets (Schelter, et al., 2019). It was developed by Amazon Research, and it is used internally at Amazon. Deequ also provides a declarative API, it allows developers to explicitly state their expectations about the data. In their original paper, Shelter et al. proposed a unit test-like approach to verify the data (Schelter, et al., 2018). In this work, certain constraints need to be defined and later these constraints will be translated into computable metrics. Similarly, Tensorflow data validation (TFDV) is an open-source scalable data analysis and validation system for ML that was developed at Google (Breck, et al., 2019). TFDV runs on top of Apache Beam where it has been designed to be highly scalable and to work well with TensorFlow. Moreover, TFDV can analyse training and serving data to compute descriptive statistics, infer a schema, and detect data anomalies (Tensorflow, 2021). Finally, Great Expectations is another open-source data validation and documentation framework (Great Expectations, 2021). In this platform, data scientists can specify the so-called *expectations*, a set of declarative statements about the data. The Great Expectations library contains built-in Expectation types to enable a wide range of Expectations, e.g., missing values, standard deviation, table row counts.

2.3 Data Cleaning

Recently, several AutoML frameworks and commercial platforms have been introduced to help users with no prior knowledge about analytics and predictive models to use them in their enterprises and organisations. Software AG Zementis (Guazzelli, et al., 2009) is an example of such platforms which enable the users to quickly deploy, execute and manage predictive models that score data in real-time or batch mode. However, these AutoML platforms broadly lack appropriate data cleaning methods for automatically dealing with contaminated datasets. Data cleaning plays an important role in the field of data management as well as analytics and machine learning. In general, data cleaning denotes the task of identifying and correcting erroneous samples in the dataset that may have a negative impact on the downstream applications. Below, we elaborate on the different data cleaning methods and tools introduced to tackle quality problems in structured, semi-structured, and unstructured data.

2.3.1. Methods for Structured Data

In general, structured data is data that adhere to a pre-defined data model and is therefore straightforward to analyse. Examples of structured data comprise weather information, stock records, sensor readings, etc. In fact, real-world data inevitably contain incomplete, inconsistent, or missing values. In this case, the corrupted data may hinder the process or provide inaccurate results, which may lead to improper business decisions. Consequently, data scientists, in typical data science projects, leverage different statistical analysis and data visualization techniques to explore the data for the sake of identifying the appropriate data cleaning operations. However, the process of exploring the data and finding the erroneous samples is usually a cumbersome task which wastes most of the development time. Moreover, data experts are needed to extract and apply business rules and statistical measures to find the erroneous samples. To overcome these challenges, several data cleaning methods have been introduced to automatically tackle the data quality problems. Since IML4E mainly focuses on ML pipelines, we classify the state of the art data cleaning methods according to their relation to ML models. Specifically, we divide the data cleaning methods for structed data into three categories, including (1) un-supervised data cleaners, (2) semi-supervised data cleaners, and (3) integrated cleaning and learning solutions.

The first category comprises rule-based and statistical methods which target specific error types, e.g., duplicates, outliers, missing values. For instance, NADEEF (Dallachiesa, et al., 2013) treats data quality rules holistically via providing an interface for implementing denial constraints and other user-defined functions. KATARA (Chu, et al., 2015) is a knowledge base and crowd-driven error detection method. Specifically, it aligns a dataset with available knowledge bases to identify erroneous samples and to suggest top-k possible repairs for incorrect data. For detecting outliers, several clustering approaches, such as KMeans, DBSCAN, and Hierarchical clustering, have been proposed (Tepić, et al., 2020). The core idea behind this approach is to group the data points into clusters, and a point is considered as an outlier if it is too far from the various clusters. Similarly, dBoost (Mariet, et al., 2016) integrates several of the most widely applied outlier detection algorithms, including histograms, Gaussian, and multivariate Gaussian mixtures. Another relevant work is HoloClean (Rekatsinas, et al., 2017) which combines qualitative and quantitative repair signals, i.e., denial constraints, in a statistical model that enables repairing the erroneous samples. The major drawback of such methods is overlooking the fact that real-world datasets typically have heterogenous error profiles. As a result, their recall becomes low due to missing other error types.

The second category comprises the methods which formulate the error detection task as a classification problem. The core idea behind these methods is to extract a set of features that enables a classifier to differentiate between clean and contaminated samples. For example, RAHA (Mahdavi, et al., 2019) leverages a set of unsupervised data cleaners to generate the feature vector. Similarly, metadata-driven error detection (Visengeriyeva & Abedjan, 2018) extracts a set of metadata to drive the training process. To further improve the detection performance, several methods, such as ED2 (Neutatz, et al., 2019), Picket (Liu, et al., 2021), and HoloDetect (Heidari, et al., 2019), models properties correspond to attribute-level, tuple-level, and dataset level features that describe the distribution governing a dataset. To avoid the need to manually provide labels, each of these methods adopts a different strategy. For instance, RAHA clusters the samples by similarity and acquires labels on a per-cluster basis, before propagating the acquired labels in each cluster. ED2 exploits active learning to acquire labels for clean/erroneous samples that the model is uncertain about. In HoloDetect, the labelling budget is reduced via augmenting the dirty samples. Specifically, HoloDetect learns the observed error patterns, before applying the patterns to generate synthetic erroneous samples. Finally, Picket strives to entirely avoid users' intervention where it does not lean on any external labels. Instead, it employs self-supervision to learn an error detection model that can be applied during training or testing.

The third category includes the data cleaning methods designed specifically for ML models. For instance, BoostClean (Krishnan, et al., 2017) treats data cleaning as a boosting problem. BoostClean uses knowledge of the labels to adaptively select from a set of repair actions to maximize prediction accuracy. Through a combination of boosting and feature selection, a good series of cleaning operations can be generated to improve the performance of the predictive model. Similarly, AlphaClean (Krishnan & Wu, 2019) transforms data cleaning into a hyperparameter optimization problem. Specifically, AlphaClean composes several pipelined cleaning operations which need to be extracted from a predefined search space. Another relevant data cleaning method is ActiveClean (Krishnan, et al., 2016) which is principally used for models with convex loss functions. ActiveClean initially trains a model on a dirty training set, where such a model is to be iteratively updated until reaching a

global minimum. Afterward, ActiveClean deals with cleaning as a stochastic gradient descent problem. In each step, the system samples and asks the user to clean records that are expected to shift the model along the steepest gradient. A similar work is CPClean (Karlaš, et al., 2020) which incrementally cleans a training set until it is certain that no more repairs can possibly change the model predictions.

For repairing the detected erroneous data samples, several approaches have been introduced. For instance, missing values represents a major problem for many industrial cases. Therefore, they have to be either deleted or imputed. The most common imputation methods are: (1) filling the missing values with a constant, e.g., dummy text, median value, average of neighbouring samples, or most frequent values; (2) filling the missing values with predicted values based on data in the entire dataset; and (3) replacing the missing values based on neighbouring values, for example if it is time-series data. For the second and third approaches, there exist many algorithms developed to determine how exactly to fill the missing values. These algorithms can be as simple as regression models or as advanced as neural networks. To find holistic repairs which deal with different error types, the BARAN system (Mahdavi & Abedjan, 2020) is introduced where a semi-supervised technique is implemented. Specifically, BARAN trains incrementally updatable models which leverage the value, the vicinity, and the domain contexts of data errors to propose correction candidates. To further increase the training data, BARAN exploits external sources, such as Wikipedia page revision history.

2.3.2. Methods for Unstructured Data

In this section, we introduce the state-of-the-art methods and techniques for cleaning unstructured data, such as images, videos, signals, and texts. In general, unstructured data has no predefined data model, where they cannot be stored in a traditional database, and they are also difficult to search. Due to their diversity of unstructured data, their preparation processes usually require data science expertise. For instance, videos consist of a set of images, thus the preprocessing steps are extremely similar to those of images or signals. Conversely, preprocessing large text typically requires completely different strategies. Below, we introduce the most prominent methods for dealing with images, signals, and videos; before elaborating on the methods developed for textual data.

In traditional image processing, filters are mainly applied to the input image to suppress certain frequencies and enhance essential features (Bishop, 2006). On the one hand, traditional filters are meticulously feature engineered (e.g., using Fourier transform and low-pass/high-pass filtering). On the other hand, recent filters are automatically computed using modern deep learning approaches. Images are usually processed by a convolutional neural network (CNN) architecture which proposes certain constraints towards the input data (LeCun & Bengio, 1995). As a prerequisite, the spatial dimensions of the input images are required to be standardized. Therefore, the input images are resized to a unified dimension. Specifically, the standardization process translates to a reduction in the dimension with the aim to reduce the computation time. However, exaggeration in reducing the size of the images usually leads to incoherent behaviour (e.g., in reinforcement learning the agent needs to be visible on the input screen). Aside from standardization, normalization is another important process, where it is almost indispensable to stabilize the CNN models and avoid overfitting. To further reduce the CNN complexity and the overall runtime, the colour of the images is converted into grayscale, unless the case where the colour encodes key information (e.g., lane detection).

Aside from data standardization, data labelling, augmentations, segmentation, and annotation are essential steps in image processing. Data augmentation is introduced to artificially increase the diversity of the training data by applying perturbed versions of the original data. To this end, realistic but random transformations are added to the training data. These transformations are usually scaling, rotations and affine transformation. Whereas data labelling is a necessary step in supervised learning to train the ML models. In data segmentation, the training data are categorized. For this purpose, the data needs to be annotated using a suitable tool, e.g., LabelMe (Russell, et al., 2007). Specifically, metadata of the bounding sets are manually added to the input data. It is worthwhile mentioning that popular CNN architectures require different annotation formats.

For signal processing, the preparation pipeline typically relies on the application domain and the ML architecture. In particular, various preparation steps, such as basic resampling, interpolation, and noise reduction, can be added to the processing pipeline. For instance, the input data may pass through traditional filters, such as FIR, high pass, low pass, to eliminate certain frequencies. Finally, textual data are processed using a set of Natural Language Processing (NLP) algorithms which require a unique preprocessing methodology (Vajjala, et al., 2020).

Such a methodology mainly consists of four steps: First, the whole input text is converted into lowercase. Second, input data is cleaned by removing noise, e.g., hashtags, single characters, links. After cleaning, the textual data may contain the so-called stop words, a set of commonly used words in a language that carries no useful information, such as “me”, “his”, “the”, and “is”. Accordingly, the third step revolves around tokenizing the input text to remove the stop words. Finally, the morphological affixes are removed by adding word normalization, e.g., stemming or lemmatization.

2.3.3. Methods for Hybrid Data

In this section, we describe the data extraction and cleaning tool for hybrid data referred to as SmartPDF AI. Figure 2 depicts the architecture of the SmartPDF tool which has been developed by Basware to extract invoice information using a set of powerful AI algorithms. The training set, utilized as an input to such algorithms, consists of hybrid image-relational data pairs, where the image represents a 2D raster image of an invoice and the relational data contains the extracted invoice values. Since the training sets are populated manually, it usually contains human errors - both instance-level and population-level errors. In this case, the training sets can be hardly cleaned using a production-grade model because it is expensive to re-train such a model. Therefore, we lean currently on a lightweight algorithm, called *Mosquito,* to clean the collected data. The cleaning strategy begins with training the Mosquito model on a validation dataset whose ground truth is available. Afterward, Mosquito re-extracts the data from the training invoices and compares the extracted values against the ground truth. Finally, the mismatched values (i.e., anomalies) are given low sample weights for the final production-grade model training. This approach boosts the accuracy of the final model by circa 30%.

In fact, the current Mosquito algorithm, adopted in SmartPDF, is entirely deterministic where no ML models are involved. The so-called "learning" process of Mosquito consists of the following steps. At the outset, the training set is clustered using n-gram-like signatures of the invoices. The clustering leverages a signature-based invoice similarity measure, and it uses the hierarchical clustering technique. Inside every cluster, the labels of the fields, so-called anchors, are estimated. Afterward, the anchors are employed for data re-extraction. The process of anchor estimation is itself affected by possible errors in the ground truth. However, the anchor ambiguity originating from ground truth errors is typically resolved through majority voting. In this case, the selected anchor candidate is the one that makes less mistakes.

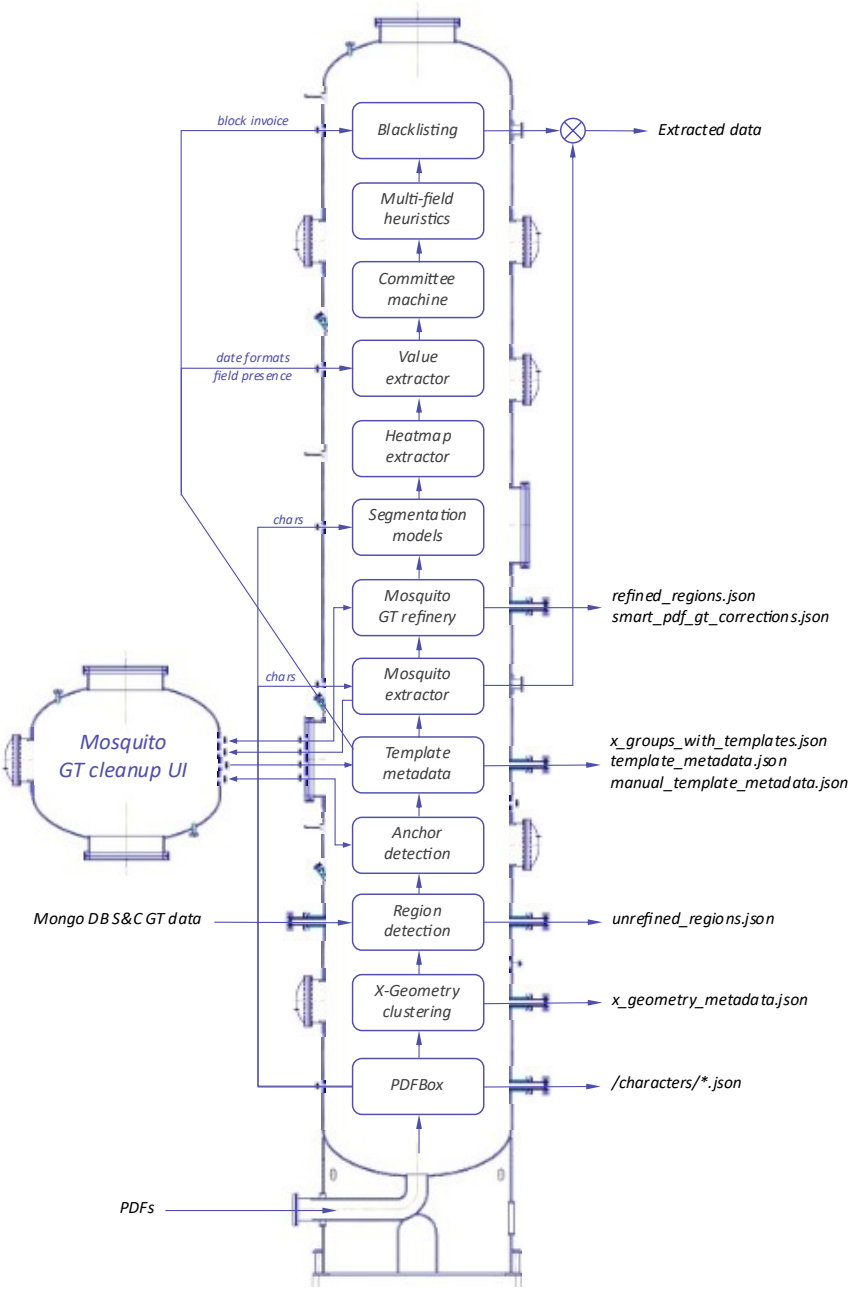

Figure 2. SmartPDF AI "Mosquito" data extractor

2.4. Data Valorisation

In this section, we report on the most prominent methods and tools in the realm of data valorisation. In general, data valorisation is a discipline which mainly deals with assessing the value of the data collected, stored, and analysed by enterprises and organizations. In fact, large-scale datasets, comprising thousands/millions of samples, has nowadays become the norm while designing modern ML models in several domains, e.g., natural language processing, computer vision, and speech recognition. Nevertheless, managing and storing data at such a scale turns into a challenging task. Furthermore, training advanced predictive models, e.g., deep neural networks (DNN), on large-scale data typically requires special arrangements as well as long execution times. To mitigate such burden, data valorisation has been introduced as an effective strategy to properly identify the most representative training samples which promote the efficiency and accuracy of the predictive models. For instance, Toneva et al. (Toneva, et al., 2018) carried out several experiments on different datasets and DNN models. The experiments revealed that not all training samples contribute equally to the modelling task.

Specifically, similar or sometimes even higher test performance can be achieved through removing a significant portion of the training data, i.e., low-quality, or noisy data may be harmful.

In fact, data valuation has been tackled from different perspectives. For instance, several data valuation methods rely on coreset construction (Feldman, et al., 2020). The core idea behind these methods is to consider only the centroid of each data cluster for downstream applications. However, these methods suffer from heavily leaning on heuristics, e.g., picking cluster centres that does not guarantee any optimal solution for the downstream task. Another line of research goes into the direction of data distillation (Zhao, et al., 2020; Wang, et al., 2018). These methods tend to learn a small set of informative images from large training data. Specifically, these methods model the network parameters as a function of the synthetic training data and learn them by minimizing the training loss over the original training data with respect to synthetic data. Despite generating small summaries effectively, the optimization process of data distillation is computationally intensive owing to the large number of parameters. Furthermore, data distillation methods do not explicitly estimate the values of the original data samples.

Recently, several permutation-based data valuation methods have been proposed to provide a value estimate for each training sample. A classical data valuation method is the so-called leave-one-out (LOO). For a dataset, the LOO method computes the value of a training sample as the performance difference of a predictive model before and after omitting this sample. Apparently, the LOO method is straightforward and simple to implement. However, the computational complexity represents a major concern where it scales linearly with the number of training samples. As a result, the computational cost becomes broadly high for large-scale datasets and complex predictive models. Moreover, the LOO method underestimates the value of samples when replicas for such samples exist in the dataset. Along a similar line, Koh et al. (Koh & Liang, 2017) exploit the influence functions from robust statistics to identify the data value for the sake of explaining the behaviour of predictive models. They trace a model's prediction through the learning algorithm and back to its training data, thus identifying the training samples most responsible for a given prediction. The original influence functions require expensive second derivative calculations and assume model differentiability and convexity. To mitigate the computational complexity, the authors approximate the influence functions with the help of second-order optimization techniques. However, such approximations typically lead to performance limitations while inheriting the major limitations of the LOO method.

Several other methods strive to efficiently estimate the data value through exploiting relevant technologies, such as game theory and reinforcement learning (Yoon, et al., 2020; Ghorbani & Zou, 2019; Ghorbani, et al., 2020). For instance, Yoon et al. (Yoon, et al., 2020) introduce a meta-learning-based data valorisation method that employs reinforcement learning to simultaneously optimize a data value estimator and a target task predictor model. Specifically, the authors propose to determine a reward via assessing the performance of a predictive model on a small validation set. Afterward, they use the reward as a reinforcement signal to learn the value of each data sample. In this case, such a value is harnessed to decide whether to engage this data sample while training the predictive model. Despite being effective in estimating the data value, this method suffers from several drawbacks, including the need for a validation set and the valuation function depends on the dataset and the target predictive model.

Another relevant work is Data Shapley (Ghorbani & Zou, 2019) which is a game-theoretic data valuation method which formulates the data valuation problem as a game. The Shapley value is a method for assigning payouts to players depending on their contribution to the total payout. Specifically, the game denotes the prediction task for a single instance of the dataset. Accordingly, each feature represents a player where all players collaborate to receive the gain. In this case, the gain is determined as the actual prediction for this instance minus the average prediction for all instances. A major drawback of Data Shapley lies in overlooking the statistical aspects of the data. As a result, it does not provide a mechanism to reason about points outside the dataset. To overcome this limitation, the authors introduce in (Ghorbani, et al., 2020) a distributional Shapley method which defines the value of a datum in the context of an underlying data distribution. However, the distributional Shapley method still leans on a known task, algorithm, and performance metric. Moreover, the computational complexity is exponential with the number of samples since it is required to compute the gain over all possible combinations.

3. Data Management and Version Control

In this section, we give an overview of data management in machine learning applications. We first describe different solutions for data storage and exchanging data between systems. We move on to discuss data processing, focusing especially on machine learning pipelines. Moreover, we cover version control for MLOps, focusing on the versioning of both data and other ML-relevant assets. Before delving into the data management tools, it is necessary to briefly discuss common data formats. Data formats represent forms for encoding data for storage in a computer file. Whenever data is sent over the network or written to a file, it needs to be encoded as a sequence of bytes according to the data format. The program reading the data then needs to decode the bytes to read it into memory. Textual data formats are human-readable and therefore very convenient as data exchange formats. Example textual formats include JSON, XML, and CSV. JSON is a prevalent format, thanks in part to its built-in support in web applications (Kleppmann, 2017). XML is a verbose format, but is still very popular due to its simplicity, generality, and cross-platform usability. Both JSON and XML have their own optional schema languages that can be used to enforce a document schema. CSV works well together with spreadsheets but does not have any schema and can be ambiguous as CSV parsers may differ subtly in how they are implemented. The main drawbacks of textual data formats are their inefficiency and the lack of built-in support for schema evolution.

Binary encoding is typically more efficient as encoding protocol than textual representation. Popular binary encoding libraries include Apache Thrift, Apache Avro, and Protocol Buffers. These libraries come with their own schema language and a code generation tool for reading and writing data, with type safety. Coupled with the schema, binary encoding can allow for a very compact representation of data. The encoding can, for example, omit all field names from the binary representation. Strict schema is useful for enforcing data integrity, as documentation, and for schema evolution. As an alternative to CSV, Apache Parquet built on Apache Thrift is a popular choice for storing big row-based data. Parquet's columnar, compressed format is optimized for efficient storage and retrieval (Kleppmann, 2017). Finally, many programming languages support their own language-specific encoding formats. For example, Python offers the pickle module for serializing and deserializing Python objects. Such serialization can be very convenient for transient data but is not recommended as a general transport mechanism due to its inherent security issues. Moreover, language-specific encoding formats such as Python's marshal are not guaranteed to be portable across language versions (Kleppmann, 2017).

3.1. Data Storage Solutions

In this section, we give a brief overview of the common data storage solutions, including general-purpose file storages, databases, data warehouses, and data lakes. In general, large-scale file storages can be used for storing massive amounts of both structured and unstructured data. There exist two common approaches, including object storage services and distributed file systems. Examples of object storage services include Amazon S3, Azure Blob Storage, Google Cloud Storage, and MinIO. In an object storage service, objects are typically created and retrieved through an HTTP API. Each object is identified by a unique key. A standard example of a distributed filesystem is Hadoop Distributed File System (HDFS). It runs on commodity hardware, is highly resilient against hardware failure, and supports performing computations directly on the machines holding the data. As an example of the level of scale achievable with HDFS, LinkedIn runs a 10 000-node Hadoop cluster storing 500 PB of data (Shvachko, et al., 2021).

Databases are commonly used for storing structured data. In SQL databases, data is stored into tables containing rows, with the approach based on the relational model developed in the 1970s. The prevalence of the SQL query language, efficient joins across tables and good support for ACID transactions make SQL databases the go-to choice for applications requiring low latency reads and writes targeting only a small subset of rows. While SQL databases require a predefined data schema, modern SQL databases also support storing and querying data stored in more flexible formats, such as JSON documents. Popular open-source SQL databases include PostgreSQL and MySQL (Kleppmann, 2017). In the late 2000s, NoSQL databases emerged as alternatives to SQL databases to offer greater horizontal scalability and freedom in the data model. NoSQL databases can be very

roughly divided into four categories: document stores, key-value databases, wide column stores, and graph databases. Thanks to the horizontal scalability, NoSQL databases make good choices for storing large volumes of data powering machine learning applications. Popular open-source alternatives include Apache HBase (built on top of HDFS), Apache Cassandra, and MongoDB (Schaefer, 2021).

The databases described above typically use indexes for low-latency access and are well suited for online transaction processing (OLTP) serving applications in real-time. Data warehouses have emerged as powerful alternatives for online analytic processing (OLAP) common in business reporting and analytics. In data warehouses, data structure and schema are defined in advance to allow efficient SQL queries spanning billions of rows. Data is stored in a column-oriented fashion to optimize bulk queries. Popular open-source alternatives for data warehousing include Apache Hive, Apache Impala and Presto. Many such tools use a distributed filesystem such as HDFS or a general-purpose object storage such as Amazon S3 as the underlying data storage. Cloud vendors offer data warehousing as a service with products such as Amazon Redshift, Google BigQuery and Azure Synapse Analytics (Kleppmann, 2017). While data warehouses are used to house structured data collected from, for example, transactional systems, data lakes can be used also for storing big non-relational datasets collected from diverse sources such as IoT and mobile devices, websites, or social media. Typically, data is stored to a data lake in raw form, so it is not expected to adhere to a predefined schema or be curated beforehand (Kleppmann, 2017). The same raw data can be used for multiple different purposes across the organization. Example implementations of a data lake could include, for example, Amazon S3 or HDFS for large-scale storage, an analytic query engine such as Presto for queries and a distributed data processing engine such as Apache Spark for data processing.

Data catalogs act as an organized inventory of data assets in the organization (Wells, 2020). Data catalog can include, for example, metadata of all data housed in a semi-structured data lake. With the metadata, the data can be efficiently cleaned, analysed, queried, and processed. Commercially available data catalogs include AWS Glue, Azure Data Catalog, and Google Data Catalog. Apache Hudi can be used for managing data in the data lake at the record level, allowing data ingestion in real-time and addressing data privacy use cases requiring record-level updates and deletes. Tools such as Hudi for managing data are commonly referred to as data Lakehouses (Armbrust, et al., 2021). Other open-source examples include Apache Iceberg and Delta Lake. Finally, feature stores are a novel type of storage specific to machine learning. They act as centralized locations for storing, retrieving, and documenting feature datasets used in building machine learning models. Feature stores enable the reuse of features across projects and teams, better governance of features computed from sensitive data, and eliminating training-serving skew. Feature stores are typically optimized for serving features for both model training requiring lots of historical data and online inference requiring up-to-date feature values with low latency. This balance is achieved by a combination of storage technologies including, for example, a data warehouse for historical features and a key-value store for online features. Examples of open-source feature store implementations include Feast and Hopsworks Feature Store. Drawbacks of feature stores are the increased complexity of the serving infrastructure and the strong coupling between the feature store and the trained models (Lakshmanan, et al., 2020).

3.2. Data Flow between Systems

Applications need to exchange data to, for example, notify each other of a change in state. For example, when a user signs up to an application, their details should be persisted, and a welcome message sent to their email address. How do applications exchange data? A simple way to exchange data is to use a data store such as a database as the intermediary. The data written to a database by one application can be read by another (or the same) application later. One challenge in using the data storage as a transport mechanism is that the internal data model used by one application can be tightly coupled to the application logic in another application, making it difficult to evolve the schemas independently (Kleppmann, 2017). Another way to exchange data is through application programming interfaces (APIs). Example implementations of such interfaces can leverage architectural styles such as representational state transfer (REST), data query and manipulation languages such as GraphQL, or a remote procedure call framework such as gRPC. Common client-server protocols used for data

exchange include HTTP, HTTP/2 and WebSockets. HTTP/2 and websockets are well suited for bidirectional data streaming in real-time (Kleppmann, 2017).

Applications can also exchange data through message brokers. Clients publish messages to the message broker and the consumers interested in messages subscribe to the broker to receive messages. Message brokers can be used as simple job queues or to implement large-scale event-driven architectures. Decoupling applications using message brokers often results in systems that are very reliable, scalable, and maintainable. The drawback is the increased complexity. Popular message brokers include AMQP-based message queues such as ActiveMQ and RabbitMQ and even streaming platforms such as Apache Kafka (Kleppmann, 2017). One emerging technology combining databases and message brokers is change data capture (CDC). CDC system monitors and captures the changes in data so that other software can respond to those changes (Hattemer, 2021). For example, the open-source CDC platform Debezium watches for row-level changes in an SQL database such as Postgres and stores all events into Apache Kafka. Any application interested in such changes can subscribe to Kafka to receive notifications.

3.3. Data Processing and ML Pipelines

Data processing systems can be roughly divided into three categories: online systems, batch processing systems and stream processing systems. With the rise of mini-batch processing, the boundary between batch and stream processing systems has become blurrier, and it is often more appropriate to distinguish between systems processing bounded and unbounded datasets instead (Lakshmanan, et al., 2020). Online systems such as HTTP/gRPC APIs wait for a client request to start processing data. After receiving a request, the online system tries to respond with a result as quickly as possible. Batch processing systems ingest a large amount of input data at a time and are often scheduled to run periodically. Stream processing systems, also called near-real-time systems, are designed to have lower latency than batch systems and process events from a broker such as Apache Kafka as fast as possible. Examples of distributed processing engines capable of batch and stream processing include Apache Spark and Apache Flink. Apache Beam provides a unified workflow definition language that can be executed on multiple processing engines.

Batch processing jobs are typically composed of multiple steps, making a pipeline. One well-known type of a batch processing pipeline is the extract-transform-load (ETL) pipeline. Its purpose is to move data from one system to another. With ETL pipelines, it is possible to use the same source data to derive multiple datasets that generate value in different business contexts. Data ingestion is an important type of data processing in machine learning. In the data ingestion pipeline, raw data is ingested into the ML system either from external systems or from the inference part of the stack, to be used for training ML models. Data ingestion can be implemented as a batch or streaming process. The pipeline typically contains steps such as data cleaning and validation to catch invalid data as early as possible. The pipeline may also contain steps for data transformation and for adding data to version control. Data may also be transformed into features and ingested into a feature store.

ML pipelines process data with the purpose of using those data by machine learning algorithms both in the training – learning phase - and in the prediction - inference phase. ML pipelines consist of common steps and the management and orchestration of those steps can make the machine learning model generation more efficient, reproducible, automated, and well-versioned (Xin, et al., 2021). In the context of reproducibility, the demands for proper versioning are high. Therefore, well-defined ML pipelines provide solutions to those demands by introducing model and data versioning steps and by introducing deployment pipelines that enable versioning on the predictions. Figure 3 illustrates the usage of model registry and feature stores that help with versioning. From a high-level view, common steps in the ML pipelines include DataOps activities such as data extraction, validation, and data preparation steps in addition to model training, evaluation, and validation steps. Beyond those steps, and when we want to operate our model to production the introduction of the model registration, model deployment and monitoring steps are needed. There are a lot of challenges on orchestrating those steps on the ML pipelines and when is the right time to re-trigger the pipeline execution for the creation of new models. New technologies and solutions are introduced to address those challenges. Technologies such as model registries and feature stores are introduced for those purposes (Orr, et al., 2021).

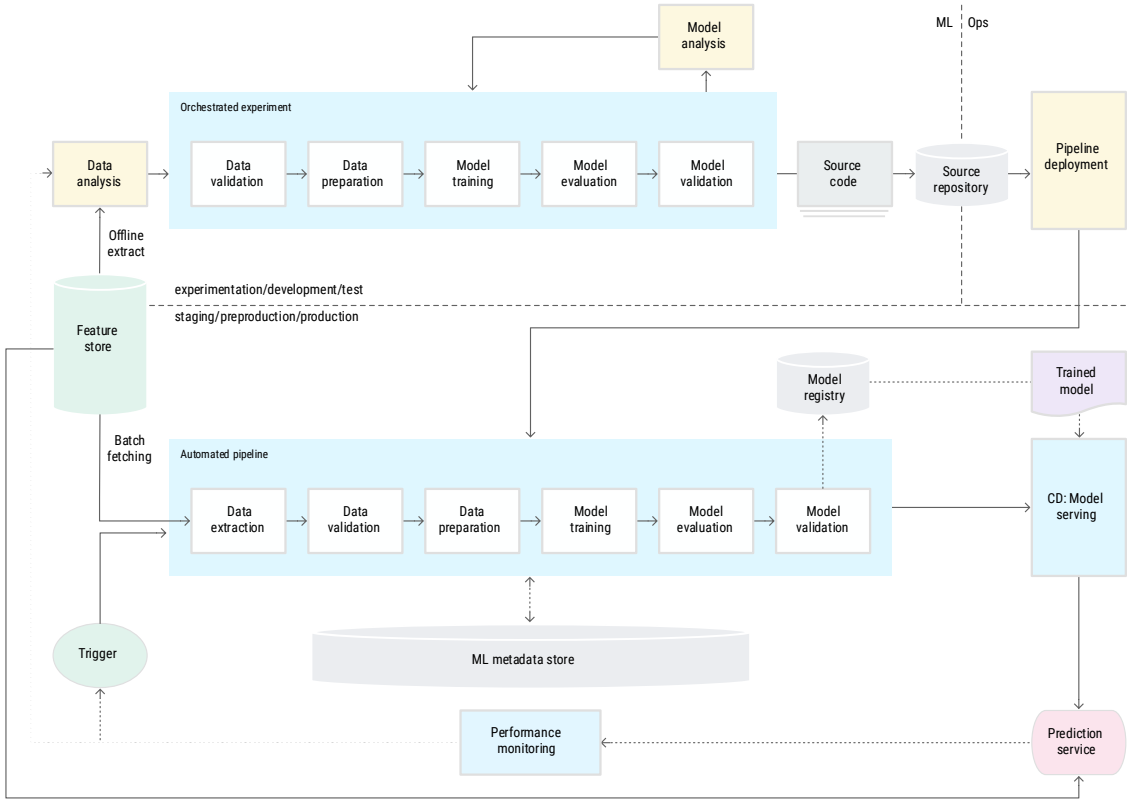

Figure 3. Google MLOps reference architecture (Google LLC, 2020)

Focusing further on the challenges that ML pipelines are facing with regards to data management and governance, enterprises are investing in structured ways when onboarding data to their data lakes and the data storage layers. ML practitioners and data scientists are always eager to know what kind of data is stored in the data layer, they want to know what their business context is, and if they can trust the data in terms of their quality. In addition, ML/AI practitioners want to be in a position to share and reuse features they have generated from those data. Thus, ML/AI practitioners can avoid repeating the same tasks and can reuse the computational resources for same operations. Therefore, companies are investing in processes and tools that increase their data discoverability, re-usability, and quality capabilities. Those processes and tools are integrated into their ETL pipelines, but they are also integrated into the ML pipelines especially when introducing technologies such as features stores. In addition to introducing new tools and processes for better data governance, enterprises are also introducing new roles, such as data stewards and data owners that govern the processes around data management and ensure that the right metadata and quality standards are met for their data. Having those tools and processes in place enterprises are moving to a more data-centric approach for machine learning where effort is placed on creating features and datasets that can be shared and trusted among ML/AI practitioners in an organization. Those reusable data assets are a prerequisite for successful ML pipelines, and they are an accelerator in the process of building ML/AI products, and data products in general.

Aside from reproducibility, proper data splitting usually plays an important role in most ML projects. When training machine learning models, the available data is typically split into three sets: training set, validation set, and test set (Géron, 2017). The training set is fed to the model during the training process. Machine learning models such as neural networks typically have such a large capacity that they overfit the training set. Therefore, the model's predictive performance on the training set cannot be used to estimate the performance on unseen data. The validation set is not shown to the model when training. It is used for, for example, deciding when to stop the training loop or selecting the model and hyperparameters that perform best on the validation set. Instead of splitting the data into fixed training and validation sets, it is common to use cross-validation. Cross-

validation uses multiple splits of the data to train the model and measure its performance. Once the model and hyperparameters performing the optimum have been found, the model can be re-trained with all available data, excluding the test set (Géron, 2017).

The test set is used to estimate the generalization error: the model performance on unseen data. Degrading performance on the test set when re-training the model can signal errors such as concept drift or data drift. To ensure that the test set properly reflects unseen data, it is important to avoid snooping into the test set at any stage of the model development (Géron, 2017). Splitting the data into training, validation and test sets should be repeatable and reproducible. One common solution is to pick a well-distributed column and a deterministic hash function to split the data. If every row is independent and each row has a unique identifier (such as the row number), one can calculate the modulo hash of the identifier of each row to determine which set it should go to. If the rows are not independent, one needs to identify the columns that capture the correlation relationship between the rows and use those as inputs to the hash function. Finally, one needs to ensure that the target labels are well distributed between the splits. If, for example, some target labels are underrepresented in the dataset, one should stratify the dataset based on the label and split each stratification evenly (Lakshmanan, et al., 2020). However, it is also important to mention that there are several cases when stratified sampling is not advised, because it would distort the real statistical properties of the label distributions. In these cases, stratified sampling may result in poorer performance. e.g., in tasks like outlier detection, in those cases the weighting of the objective function can be a better approach.

3.4. Data Privacy

Many datasets are about people: their behaviour, health, interests, and identity. When personal information is stored or processed, great care must be taken to respect their privacy: their freedom to choose which things to reveal to whom, what to make public, and what to keep secret (Kleppmann, 2017). GDPR (General Data Protection Regulation) regulates the storage and processing of personal data in all organizations targeting or collecting data related to people in the EU. In the context of GDPR, personal data is any information that relates to an individual who can be directly or indirectly identified. GDPR includes several protection and accountability principles for processing personal data. These are lawfulness, fairness and transparency, purpose limitation, data minimization, accuracy, storage lifetime limitation, integrity and confidentiality, and accountability.

According to these principles, users' consent to processing their personal data must be freely given, specific, informed, and unambiguous. Data must be processed only for the legitimate purposes known to the data subject. Only much data is necessary should be collected. Personal data must be accurate and up-to-date and only stored as long as necessary for the specified purpose. Data must be stored and processed in a secure manner, using, e.g., encryption for data at rest and in transit. Finally, the data controller must be able to demonstrate the compliance of all these principles. In addition, GDPR gives data subjects the right of access to information concerning automated decision-making using their personal data. Depending on the type of decision-making, the subjects may have the right of access to meaningful information about the logic involved and the significance of consequences of such processing to them. Machine learning systems used for automated decision-making should be designed with this in mind and therefore be explainable enough and ready for audit purposes.

Data processing techniques such as pseudonymization and anonymization can be used to protect personal data. Pseudonymisation means the processing of personal data in such a manner that the personal data can no longer be attributed to a specific person without the use of additional information. Pseudonymized data are still personal data and subject to data protection regulation. Anonymisation means the processing of personal data so that it is impossible to identify individuals from them. Anonymised data does not constitute personal data (ENISA, 2021). However, reaching full anonymity in the data can be hard as increasing number of data features may also describe individual persons more personally and make their identities easier to be determined. In this sense, treating data related to persons as pseudonymized, instead of anonymized, is the safer strategy from a data privacy regulation compliance point-of-view.

3.5. Version Control in ML Systems

Version control is a standard best practice when developing software code, updating data schemas, or building software artifacts such as container images. Versioning helps understand and monitor software solutions running in production and meet compliance requirements. With a good versioning methodology, an enterprise can identify and reproduce issues that occurred in the production even in the past. In addition, enterprises can identify the quality assurance steps they performed before releasing software, increasing the trust and the compliance of a solution. Versioning in ML systems is complicated by the fact that ML systems are fundamentally different from traditional software systems (Kumeno, 2019). Two sources of complexity are (1) the black-box nature of ML models, requiring us to pay very careful attention to understanding the model behaviour, and (2) the fact that ML models are not only dependent on the source code that was used to train them but also on the data used in the training phase. Below, we give a brief overview of data version control and then extend the discussion to other artifacts relevant for ML.

Data versioning means having a unique reference to an immutable collection of data. Because datasets used in machine learning are typically very large, one cannot directly use the same techniques as in software version control. A simple approach for data versioning is to duplicate the dataset for every new version. For example, you could have a cloud bucket with one folder for each version, each folder containing all data included in the version. This approach works well as a starting point but wastes storage space and does not scale well for big datasets. Also, the workflow for managing data versions can be quite inconvenient without sophisticated tooling. A slightly more scalable approach is to duplicate only references to data objects such as images across versions. The list of references included in each version can be stored in a database. This kind of metadata index is cheaper to maintain than duplicating raw data. However, neither of the above approaches considers the incremental manner of how data is typically added or deleted. The data collected this week is likely added to the dataset created last week. Luckily, standard software version control systems such as Git are built with exactly such a workflow in mind. Instead of duplicating files or references from one version to another, they keep track of the difference between versions.

Version control systems such as Git are not, however, designed to directly include large files or datasets. Git LFS (Large-File Storage) is an open-source extension of Git that replaces large files with text pointers and stores the file contents on a remote server. Git LFS solves the problem of versioning large files but does not come with any tooling for managing data versions. DVC (Data Version Control) is a Python library specifically created for data versioning on top of Git. It offers a command-line tool with Git-like commands for adding, updating, and removing data. Data files can be stored to a user-configurable remote, analogous to git remote, such as a shared file system, cloud bucket or SSH server. Data version in DVC is a Git revision such as commit tag or hash that can be logged during the training run. DVC also offers tooling for creating lightweight data manipulation pipelines.

Git-based tools such as DVC only track changes in file digest hashes instead of tracking row-level changes to data. Data Lakehouses introduced in Section 3.1 bring transactional operations to data lakes and can be used for row-level data versioning. Tools such as Apache Hudi, Apache Iceberg and Delta Lake allow time-travel queries that return, for example, changes to data since a given point in time. These tools record changes as new appended rows in the metadata table, updating the version column. This approach of keeping track of updates is known as Slowly Changing Dimension (SDC) Type 2. These technologies are used for creating feature stores over data lakes. For example, Hopsworks Feature store supports Apache Hudi as storage format for feature groups. Many of the platforms built for full-fledged data science can also cover data versioning needs. Such platforms include tools available as open-source and commercially, such as Neptune, Pachyderm, LakeFS, and MLFlow.

Having discussed about data versioning and moving towards describing the versioning control of ML systems, we need to connect the data version used for creating a model with the created model version and with the source code used for training and operating that model. However, we also need to establish a connection with many other assets important for, e.g., explainability and auditability, including:

(a) Quality assurance code and validation reports for data

(b) Featuring engineering code

(c) Hyperparameters selected in hyperparameter tuning

(d) Evaluation metrics as well as the validation and test datasets used

(e) Explainability and bias reports

(f) Quality assurance steps performed for the training and inference source code

(g) The hardware, operating system and library versions used for training and hosting the model

One of the popular approaches used by practitioners to meet the versioning requirements in the ML solutions is to combine the following open-source tools: (1) Git for source code versioning, (2) MLFlow for versioning models, training artifacts and reports, and (3) DVC for data versioning. Finally, we should mention that versioning in ML systems is not only related to the model building and training phase. When operating ML models in production it is important for transparency purposes (European Union, Ethics Guidelines for trustworthy AI) to be able to track model predictions, to explain them and to know under which context they have been made. Thus, it is important to bind and store model predictions with the corresponding input data and with the model version that we have deployed and created during the training phase. In this way, we can trace model predictions in our ML solutions and analyse them for monitoring and compliance purposes. For achieving the above, our model deployment pipelines play a key role. Our deployment pipelines and software must persist all the necessary ML metadata (for example model and data versions) together with the model predictions to enable full traceability and transparency.

4. Continuous Data Quality Assurance

4.1. Quality Attributes and Typical Faults

ML models learn patterns in data and make decisions based on what they learned during training. Such dependency on data clearly indicates that the quality of data is integral to the system's ability to perform. In (A4Q, 2019), A4Q provides an example of how data quality issues can affect a model's behaviour. The authors consider the Titanic dataset (Kaggle, 2012) which includes gender as one of the features. A model trained on this data learned that majority of survivors were female and developed a bias towards this feature. While it had impressive accuracy on the training data, when the model was applied to the data from another passenger ship - The Lusitania - which sank three years later, the model did not have good accuracy. This example shows how a high level of bias in data can lead models to pick up features that are not sufficiently representative of the real world. Given the importance of data quality in ML, how do we define it? What are the metrics that can be used to access the quality of data? This section addresses these questions by providing a brief survey on how the state-of-the-art treat data quality.

In general, data quality can be defined simply as "fitness for use" (Gudivada, et al., 2017); however, ML models can be highly specific to a particular domain with a large variety of possible learning algorithms. Thus, the interpretation of quality along with the metrics and techniques must be adjusted accordingly. In (Gudivada, et al., 2017), the authors define data quality as a function of multiple dimensions like accuracy, currency, and consistency. Expressing data quality in terms of dimensions allows to define quality in terms of specific domains. Accessing data quality then involves quantifying the individual dimensions. The authors list a set of dimensions in the context of ML which are listed below:

- Duplication – Presence of duplicates on training data can be detrimental to model accuracy. In (Schofield, et al., 2017), the authors find that algorithms like LSA (Deerwester, et al., 1990) which look for patterns of repetitions in data perform poorly when documents are repeated multiple times in a corpus. The extent to which duplicates affect the performance and the rate of duplication that can be considered safe can vary for different learning algorithms (Schofield, et al., 2017). In (Kołcz, et al., 2003), the authors argue that the presence of duplicates can affect the quality of learning of the classifiers. For instance, for Bayesian classifiers in which the loss computed during training depends on the number of repetitions of a sample, high rates of duplicates would mean that the classifier will try to reduce the cost giving more importance to the region of duplicates. This can cause classifiers to overfit.
- Outliers – In a set of data, an outlier can be defined as an observation (or subsets of observations) which appears to be inconsistent with the remainder of that set of data (Barnett & Lewis, 1994). Outliers can originate from human errors like mislabelling of data or due to sampling errors (Laurikkala, et al., 2000); however, they can also be legitimate data points (Gudivada, et al., 2017; Liu, et al., 2002). Outliers can influence how well a model fits to the data (Laurikkala, et al., 2000).
- Dimensionality – ML algorithms that utilize Lp-Norm between the data points to perform regression or classification tasks do not work well when the input vector is high dimensional (Gudivada, et al., 2017). Similarly, higher dimensionality of data in neural networks can result in models being more sensitive to small noises in the dataset (Goodfellow, et al., 2015). Large input gradients due to high dimensionality of input space facilitate adversarial attacks which leverage gradient amplitude of a network to craft adversarial examples, such as the Fast Gradient Method (FGM) as proposed in (Goodfellow, et al., 2015). Tiny imperceptible changes can thus lead to more successful attacks (Demontis, et al., 2019).
- Accuracy – Accuracy measures the degree of correctness of data. While it is obvious that inaccurate data results in a non-reliable model, the extent to which a model is affected by data accuracy issues can depend on the learning algorithm itself (Sessions & Valtorta, 2006).
- Bias – Sample bias occurs when the training data is not sufficiently representative of the input space. Racial or gender bias in the training data can be picked up by ML models during training (A4Q, 2019).

- Consistency – Consistency defines the extent to which the data is presented in the same format (Wang & Strong, 1996). In ML, relevant variables and features are often extracted from raw data. In this case, consistent patterns should be established to identify the relevant features (Gudivada, et al., 2017).
- Data currency – A model might become irrelevant as the correlation between input and output degrades overtime (A4Q, 2019). For instance, as fashion trends change or when certain products are not available in the market anymore, models trained on old training data may not give relevant predictions.
- Completeness – Completeness defines the extent to which data are of sufficient breadth, depth, and scope for the task at hand (Wang & Strong, 1996). In other words, the extent to which data is missing in a dataset. Both missing features and missing data points can impact the performance of a model in terms of model accuracy (Hakim & Darari, 2021).
- Size of the dataset – The size of data required for model building often depends on the problem domain (Gudivada, et al., 2017). For instance, to model a linear relationship, a small set of data is sufficient as simple linear models can be tasked to solve the problem. However, for more complex problems like self-driving cars or obstacle detection, complex models are required along with a large volume of data. Training a complex model on a limited dataset can lead to the model being overfitted.

In the list above, the dimensions like accuracy, consistency and currency are generic and are often associated with operational data. This signifies that, albeit insufficient, traditional data quality metrics are still applicable to ML. There is a distinction of data quality dimensions into hard dimensions and soft dimensions (Ehrlinger et. al., 2019) where the first includes dimensions like accuracy, completeness, or data currency. Hard dimensions are characterized by the capability of being objectify measured, while soft dimensions which includes for instance bias are obtained with subjective means.

In addition to the dimensions listed above, class imbalance is another common problem in datasets. When building classification models, an ideal assumption is that the training data is distributed equally among each class. However, more often than not, data distribution is not only skewed, but also it is against the class that is of more importance (Krawczyk, 2016). This results in the classifier not being accurate when identifying the minority class. Perhaps, the most common examples used in this regard are from the medical diagnosis field. Training a simple binary classifier that is to separate benign tumours from the malign ones may be problematic when most of the data in the training set fall under the benign category. The classifier does not get enough opportunity to learn features from data that makes a tumour malign and thus will be biased towards the "benign" class. In (Krawczyk, 2016), the author provides a list of recent real-life applications with data imbalance. Class imbalance issues are not only limited to small datasets but are prevalent in multi-class classification with large datasets as well (Krawczyk, 2016; Johnson & Khoshgoftaar, 2019). Skewed distributions are not a new problem though, as such, there are methods that can be employed to deal with imbalanced data, for instance, by modifying the data itself (i.e., data-level methods), or the learning algorithm (i.e., algorithm-level methods) or both (i.e., hybrid) (Krawczyk, 2016).

Learning conflicts among the data points in a dataset is yet another concern. Learning conflicts occur when two input points that are similar to each other have different target values (Ledesma, et al., 2018). The target value of input refers to the corresponding output from the model for which the loss is zero. In (Ledesma, et al., 2018), the authors demonstrate that learning conflicts can affect the performance of a model by training a small neural network on a dataset with and without learning conflicts. The authors show that removing learning conflicts results in a drastic decrease in root- mean-square-error (RMSE) of the classifier. Further experiments in a more real-world setting involved training a network on a dataset for a refrigeration system. The network, like in the previous experiment, showed a considerable drop in performance when learning conflicts were present in the data. Furthermore, data inherently constitute of "non-robust" features (Ilyas, et al., 2019) which result in models being vulnerable to adversarial examples.

Adversarial examples are out-of-distribution data points that are created by adding slight perturbations to the benign samples of data. These examples look exactly like their benign counterpart to human oracles but are misclassified when fed as input to a classifier. These "adversarial examples" are usually studied in terms of model

properties like linearity (Goodfellow, et al., 2015) and complexity (Demontis, et al., 2019), among others. However, recent studies show that the adversarial vulnerability of networks might be related to data itself (Ilyas, et al., 2019). In (Ilyas, et al., 2019), the authors argue that the training data constitute of features that are incomprehensible to humans but are highly expressive to machines. Figure 4 shows one of the examples from the ImageNet dataset (Deng, et al., 2009). As it can be seen in the figure, although the features do not resemble to a Frog to humans, it is expressive to networks. Since these features are highly predictive, networks learn these features during training and become highly reliant on them during inference. These features are called non-robust because they are highly brittle in nature and any small changes in them can cause networks to misclassify the image. Thus, the brittle and the imperceptible nature makes them easily exploitable by adversaries to craft adversarial examples. The "non-robust" features are not isolated to certain datasets but are inherently present in all data distributions (Ilyas, et al., 2019).

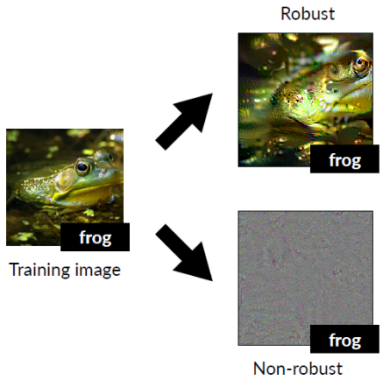

Figure 4. Disentangled robust and non-robust features from an image. The robust features are much more recognizable to humans while the non-robust are incomprehensible (Ilyas, et al., 2019)

4.2. Data Quality-Based Uncertainties and Risks in ML

ML-based systems are inherently probabilistic in nature, in that, there is always a degree of uncertainty attached to the output decisions (A4Q, 2019). Since ML/AI-based systems are increasingly being used in safety-critical domains like medical diagnosis, self-driving cars, and aircraft among others, precise sources of uncertainties for a given situation should be considered during model development (Kläs & Vollmer, 2018). Mapping uncertainty to different sources enables developers to effectively deal with such uncertainties. Various literatures characterize uncertainty in terms of different sources like the model's execution environment, system goals and the nature of the model itself (Kläs & Vollmer, 2018). In this section, we focus on the data quality as one of such sources. In (Kläs & Vollmer, 2018), the authors present a three-layered model of classification of uncertainty based on the model building process. As shown in Figure 5, the authors identify data quality, model fit and scope compliance as the major sources of uncertainty.

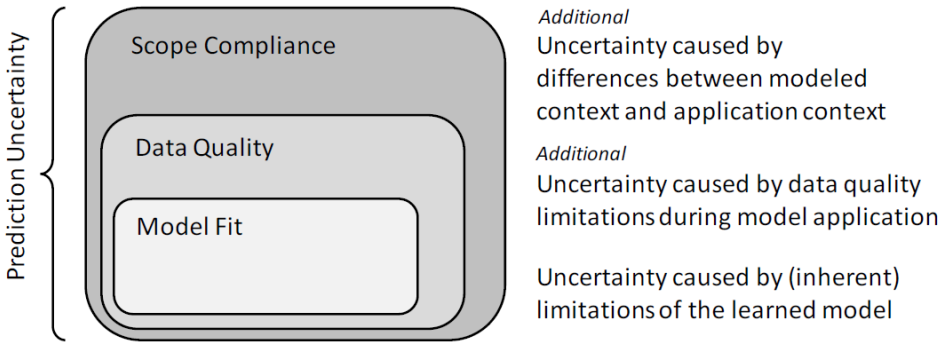

Figure 5. Onion layer model of uncertainty (Kläs & Vollmer, 2018)

The model building process constitutes of collecting raw data based on the scope of the model application followed by the data cleaning process which prepares data for training, validation, and testing the model. Based on this process of learning, hyperparameters of the learning algorithm (accounted by uncertainty caused by model fit); use of the model outside of its intended scope or application area (accounted by uncertainty due to scope compliance); and quality of the data the model is currently being applied on (accounted by data quality) are identified as major sources of uncertainty (Kläs & Vollmer, 2018). Figure 4 also indicates that given the model is applied under correct context, uncertainty due to data quality can be defined as the delta between the uncertainty observed at the model's outcome and the uncertainty that can be explained due to model fit.

Uncertainty estimates are expressed in terms of confidence and probability intervals in the case of numerical outcomes and in terms of confidence scores in the case of categorical outcomes. Models like decision trees incorporate uncertainty estimates based on the data used to build the model on the model outcome in the form of probability values (Kläs & Vollmer, 2018). Further, the uncertainty estimates derived from the data used to build the model (as opposed to the ones derived based on previously unseen test data) are prone to overfitting which may lead to inaccurate values when applied to new unseen data. To handle this, evaluation strategies should not only consider evaluating model outcomes but also the correctness of the uncertainty estimates (Kläs & Vollmer, 2018). One of the challenges in determining the impact of data quality on the model outcome is that the corner case inputs that are not available in test or training datasets can have a large impact on model output and can cause models to behave in an unexpected way. One of the ways to test a model's robustness against these data points is using Generative-Adversarial-Networks (GANs) (Goodfellow, et al., 2015) to augment the benign samples from test or training data is by adding specific perturbations or varying weather or lighting conditions and provide them as input to the model (Kläs & Vollmer, 2018).

4.3. Data Quality for ML Certification

Certification is the formal acknowledgment of certain characteristics of an entity. The goal of certification is to establish trust, while the means of establishing this trust generally may vary its common in the IT domain that an assessment of the existents of certain characteristics via a 3rd party is the foundation of any certification. Rules and Requirements for those assessments are laid out by a certification body, which also grants a certification based on the results of the assessment. Via such a certification scheme trust gets established. Therefore, to certify the existence of the above-mentioned characteristics of data for ML its necessary to perform an assessment as well as lay down rules.

Regarding the evaluation and certify data quality ISO released the 25000 series standard for the evaluation and management of IT system quality. In particular, standards such as ISO/IEC 25012 (ISO/IEC, 2008) and ISO/IEC 25024 (ISO/IEC, 2015) address data quality. ISO/IEC 25012 defines a set of characteristics for data quality, namely: Accuracy, Completeness, Consistency, Currentness and Credibility. ISO/ICE 25024 provides means to evaluate certain features of the data via assigning so-called quality properties to the quality characteristics introduces in ISO/IEC 25012. For example, accuracy can be evaluated via the properties: syntactic accuracy, semantic accuracy, and accuracy range. For the quality assessment suitable quality characteristics and requirements must be defined, so that the desired data quality requirements are met. Regarding ML, more quality characteristics have been defined, like fairness and correctness. In IML4E, we address the state-of-the-art set of characteristics for data quality for ML as well as extending on defining necessary quality properties.

The ISO/ICE 25000 series do not specify how the quality properties are being measured, this lies in the judgment and experience of the assessing party. In (Gualo, et al., 2021, p. 176), the authors present a data quality evaluation process that is carried out in activities from quality requirement establishment to executing the quality evaluation and through which the measures for the selected data quality properties get obtained. As pointed out in their work the actual specification of the data quality as well as the measurements for their assessment depends on the actual system as well as the business case. In the context of IML4E we are extending the approach from the ISO/ICE 25000 toward a standardised set of measurements based on the IML4E MLOps Framework. We are also thriving for an automated assessment of the quality which will lead to a continuous certification.

Even for a single quality characteristic like accuracy literature shows several different definitions. This variety continues when it comes to metrics. To drive the automated assessment of data quality the utilization of tools is necessary. In (Ehrlinger et. al., 2019), the authors provide a survey of data quality measurement and monitoring tools, where they evaluated 13 tools including the used metrics for data quality measurement and their capabilities for continuous data quality monitoring (Ehrlinger et. al., 2019). To be able to continuously assess the data quality based on rules, DaQL 2.0 has been introduced, a tool that once rules are implemented for complex data objects, it allows a continuous data quality measurement (Ehrlinger L. et. al., 2019).

5. IML4E Baseline

In this section we recapitulate by describing the baseline for the upcoming IML4E R&D activities of WP2. The baseline consists of the state-of-the-art methods and tools that serve as a good starting point for addressing the research questions relevant for this work package. The section is structured according to the state-of-the-art sections of this deliverable, which also corresponds to the three main research tasks of WP2.

5.1. Baseline for Data Preparation Automation

One of the research questions that IML4E addresses is how to automate the data preparation processes efficiently and effectively. To properly answer this question, we started with identifying the weaknesses of existing methods and tools. As discussed in Section 2.1, the already-existing data profiling methods are typically used to analyse data for a single ML model. During the project, the goal is to develop a solution for examining the suitability of data for a large amount of ML models (i.e., hundreds to thousands of models). The solution aims to ensure and automate that the data in these several different models is suitable for the purpose. Here, energy management of buildings is used as a use case, as each building has its own ML model. In our contribution, we will focus on unstructured data, such as videos and images, where our aim is to reduce the computation time of the cleaning process and to automate data preparation on site. Moreover, an automated data validation step will be added to the cleaning process. The data validation step should be computed almost in real time. In Section 2.2, different data validation tools were introduced, the main goal here is to add an automated data validation step to the incoming video data stream that can be computed close to real time. The different tools will be analyzed, and expectations about the data will be defined.

In IML4E, Basware plans to carry out extensive research to advance its current cleaning algorithms for hybrid data extracted from invoices or documents. As aforementioned in Section 2.3.3, current Mosquito algorithms are vulnerable to errors in the ground truth, e.g., there might not be enough training data, empty values, and too many human mistakes cause AI model to learn wrong. In this context, self-supervised learning seems to be a promising way to recognize and understand patterns without carefully labelling the data. Suitable models identified for invoice extracting domain are as follows.

- 2D-SCFG: considering the part-whole hierarchy of the invoice as Stochastic 2D Context-Free Grammar (2D-SCFG), e.g., DeepCPCFG. The grammar will be learned in a self-supervised manner. The learned part-whole hierarchy of the invoice will be used to assign the unique embedding vectors to the invoice areas, which can later be mapped to the ground truth labels with the current Mosquito voting technique.
- GLOM: learning the part-whole hierarchy through using the Geoffrey Hinton's idea of GLOM. In this approach, the model learns the part-whole hierarchy of the invoice using an attention-based autoregressive NN model.

In both models, self-supervised training is used to learn the structural representation of the invoice. The self-supervised learning is not affected by the ground truth errors, so it is not necessary to perform any cleaning to train the model. The labelling of the discovered invoice areas and the final anomaly detection is done with existing deterministic algorithms. For structured data, we find that the already-existing data cleaning methods mostly overlook context information and historical knowledge which can help in making the right cleaning decisions throughout the life cycle of ML projects. Moreover, existing methods and tools designed for ML models mostly target specific predictive models or optimizers, such as gradient descent. We envision that existing methods and tools can be further extended to make proper cleaning decisions under different scenarios. To this end, we plan to exploit relevant concepts and technologies, such as transfer learning, meta learning, and reinforcement learning. Additionally, existing methods and tools treat all data samples equally, which might be incorrect in several real-world applications. For instance, data scraped from the Internet typically contains samples that might be irrelevant to the target applications. Accordingly, data cleaning methods and tools can safely overlook these samples while curating the data.

Based on this analysis, we plan to address shortcomings by deeply investigating the interplay between data valuation and data cleaning. In addition to improving the predictive performance, data valuation also enables several other use cases. Among these use cases is to suggest better practices for data collection, e.g., what kinds of additional data would the model benefit the most from. It is worth mentioning that combining different

features typically creates new insights which might change the value of data samples. Figure 6 delineates a data science workflow with our envisioned automated data preparation. As it can be seen in the figure, several assets, such as historical artifacts, context information, and cleaning signals, are to be adopted to guide the data preparation process.

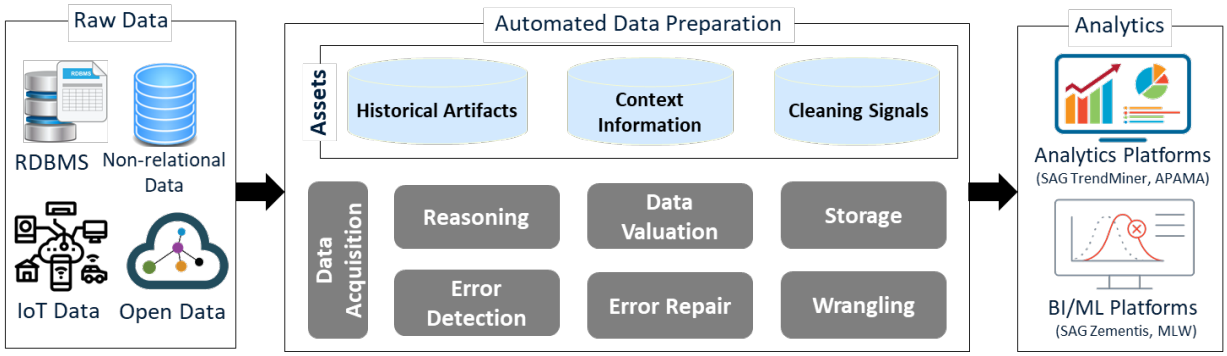

Figure 6. Automated data preparation within data science workflows

5.2. Baseline for Data Management and Version Control

We presented the state-of-the-art tools and methods for data management and version control in Section 3. We discussed databases and data warehouses as storage solutions for structured data and distributed filesystems and object storage services for any data. For data version control, we discussed row-level, transactional approaches based on data Lakehouses and feature stores as well as file-tracking approaches similar to those used in software version control. One of the open research questions is how to best implement these different approaches for large and rapidly changing data assets including structured, unstructured, and semi-structured data. Can we propose a unified data version control solution for data coming from multiple master data sources? We also discussed how the data version control needs to be combined with the version control of other assets specific to ML systems. There are no established best practices for this kind of holistic version control. How to best combine the existing tools for such ML system version control? We identified some potential software packages to be used as a basis.

5.3. Baseline for Continuous Data Quality Assurance

The assurance that the data quality meets the requirements cannot be obtained via one-time activity. Ensuring sufficient data quality over the lifetime of the ML-System requires a continuous assessment that the data meets the required characteristics. In Section 4, we discuss what quality attributes according to the current state of the art are considered important for machine learning tasks. Additionally, we describe typical flaws in datasets like class imbalance, learning conflicts, label noise, and non-robust features. In summary, we can say that the current state of the art presents a variety of characteristics for data which differ depending on the particular case. As we are thriving for a continuous assessment of quality in IML4E it is important to rely on measurements that are computable. Hence, we will work on implementing measurements for the so-called hard characteristics.

The issue of missing corner cases is described in the section Data Quality-Based Uncertainties and Risks in ML as uncertainties that require a trained model for its estimation as well as the correctness of the uncertainty. In IML4E, we address the complete MLOps pipeline. Therefore, we utilize the available models to detect uncertainties in the data set. In Section 4.3, we describe the state of the art in data quality certification. To our knowledge, current approaches do not specify how the data quality properties are measured for the assessment. On the side of the automated measurement of data quality, there exist few tools and methods emerging. In IML4E, we want to close the gap between high level requirements coming from high level frameworks and the measurements of dedicated tools. Those tools require a definition on the rules for the measurement, we are planning of automating this rules definition as much as possible. Executing measurements based on schemas like ISO 25012 will result in a continuous certification of the data quality.

6. Summary

WP2 aims to analyse requirements that drive the data collection, processing and management infrastructure and implement the data layer according to these requirements. It will introduce data management and version control management process along with a system supporting it. A continuous data quality assurance process will ensure qualitative and quantitative sufficiency of the data to be used by analytics and machine learning (ML).

In this deliverable, we have given an overview of the most relevant stat of the art, structured according to the three main research tasks of WP2. Specifically, task T3.1 deals with developing mechanisms (algorithms, components, workflows) for handling large-scale, multi-faceted, real-time, and noisy data collection. The state of the art relevant to T3.1 is elaborated in Section 2. Whereas task T3.2 revolves around defining and establishing a data management procedure following a stringent process which is accompanying the data life cycle throughout the project. The state of the art relevant to task T3.2 is provided in Section 3. Finally, task T3.3 considers establishing a continuous data quality process and ensure that data quality and quantity fulfil the requirements of its intended use in operation, planning and training throughout all data sources that will be used within the ML project. Additionally, task T3.3 deals with the uncertainties and risks in ML owing to data quality issues. The state of the art relevant to task T3.3 is presented in Section 4.

As a summary, we briefly conclude in the following on each of the relevant IML4E research questions listed in the introduction:

- How to enable automated processing with integrated quality assurance of data in the data preparation pipeline? As discussed in Section 2, there exist several techniques for data profiling and validation, and data cleansing. However, these techniques and tools assume that the data is already prepared and resides in a single location. However, most real-world data sets are messy and originate from several sources, and probably without semantics that can help data scientists to make use of the data. In this context, WP2 plans to overcome these challenges via exploiting the already-existing artefacts to design fully automated data preparation pipelines which can deal with different data models at various scales.
- How to improve the modularity, privacy, and reuse of development and data artefacts, datasets and metadata that may serve the training of models in different application contexts, throughout the development process? As explained in Section 3, feature stores represent a data management system dedicated to ML projects, where it enables the reuse of features in different projects. The current solutions usually suffer from the increased complexity of the serving infrastructure and the strong coupling between the feature store and the trained models. In this realm, WP2 plans to overcome these challenges through introducing novel concepts and ideas for a feature that has minimum overhead on the infrastructure.
- How to enable flexible data versioning and traceability of development and data artefacts (data sets, models, parameters, test results) during data preparation, training, and operations? As discussed in Section 3, there exist currently several data version control tools to achieve better reproducibility. However, these tools still provide basic support and cannot be used with different data modalities. In this context, WP2 considers establishing a connection with many other assets important, including quality assurance code and validation reports for data, featuring engineering code, hyperparameters selected in hyperparameter tuning, evaluation metrics as well as the validation and test datasets used, explainability and bias reports, and the hardware, operating system and library versions used for training and hosting the model.
- What are good methods and tools for continuous testing and verification of ML artefacts during development, reuse, and deployment? As discussed in Section 4, uncertainties in the ML development processes may lead to distrust in these technologies. In this context, WP2 considers developing methods and tools for mapping and quantifying uncertainties to the data quality, as a major source of uncertainty in ML projects. Consequently, we enable developers to effectively deal with such uncertainties. Moreover, we plan to extend the approach from the ISO/ICE 25000 toward a standardised set of measurements based on the IML4E MLOps Framework. We are also thriving for an automated assessment of the quality which will lead to a continuous certification.

References

A4Q, 2019. AI and Software Testing Foundation Syllabus. [Online]
Available at:
https://www.gasq.org/files/content/gasq/downloads/certification/A4Q%20AI%20%26%20Software%20Testing/AI_Software_Testing_Syllabus%20%281.0%29.pdf
[Accessed 20 December 2021].

Abedjan, Z. et al., 2016. Detecting data errors: Where are we and what needs to be done?. New Delhi, Proceedings of the VLDB Endowment.

Abedjan, Z., Golab, L. & Naumann, F., 2015. Profiling relational data: a survey. VLDB Journal 24, pp. 557-581.

Armbrust, M. et al., 2021. Frequently Asked Questions About the Data Lakehouse. [Online]
Available at: https://databricks.com/blog/2021/08/30/frequently-asked-questions-about-the-data-lakehouse.html
[Accessed 20 December 2021].

Barnett, V. & Lewis, T., 1994. Outliers in statistical data. Third Edition. Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics.

Bishop, C. M., 2006. Pattern recognition. Machine learning, 128(9).

Breck, E. et al., 2019. Data Validation for Machine Learning. California, MLSys.

Chu, X. et al., 2015. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. Melbourne, ACM SIGMOD international conference on management of data.

Dallachiesa, M. et al., 2013. NADEEF: a commodity data cleaning system. New York, ACM SIGMOD International Conference on Management of Data.

Deerwester, S. et al., 1990. Indexing by latent semantic analysis. Journal of the American society for information science, 41(6), pp. 391-407.

Demontis, A. et al., 2019. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. Santa Clara, 28th {USENIX} Security Symposium.

Deng, J. et al., 2009. Imagenet: A large-scale hierarchical image database. Miami, IEEE conference on computer vision and pattern recognition.

Ehrlinger et. al., L., 2019. A Survey of Data Quality Measurement and Monitoring Tools. ArXiv, Issue abs/1907.08138.

Ehrlinger L. et. al., 2019. A DaQL to monitor data quality in machine learning applications.. International Conference on Database and Expert Systems Applications, pp. 227-237.

ENISA, 2021. Cybersecurity to the Rescue: Pseudonymisation for Personal Data Protection. [Online]
Available at: https://www.enisa.europa.eu/news/enisa-news/cybersecurity-to-the-rescue-pseudonymisation-for-personal-data-protection
[Accessed 21 December 2021].

Feldman, D., Schmidt, M. & Sohler, C., 2020. Turning big data into tiny data: Constant-size coresets for k-means. SIAM Journal on Computing, 49(3), pp. 601-657.

Géron, A., 2017. Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc..

Ghorbani, A., Kim, M. & Zou, J., 2020. A distributional framework for data valuation. Online, International Conference on Machine Learning.

Ghorbani, A. & Zou, J., 2019. Data shapley: Equitable valuation of data for machine learning. California, International Conference on Machine Learning.

Goodfellow, I., Shlens, J. & Szegedy, C., 2015. Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

Google LLC, 2020. MLOps: Continuous delivery and automation pipelines in machine learning. [Online]
Available at: https://cloud.google.com/architecture/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning
[Accessed 10 Decemer 2021].

Great Expectations, 2021. How does Great Expectations fit into ML Ops?. [Online]
Available at: https://greatexpectations.io/blog/ml-ops-great-expectations/
[Accessed 19 Dec 2021].

Gualo, F. et al., 2021. Data Quality Certification using ISO/IEC 25012: Industrial Experiences. Journal of Systems and Software.

Guazzelli, A., Stathatos, K. & Zeller, M., 2009. Efficient deployment of predictive analytics through open standards and cloud computing. ACM SIGKDD Explorations Newsletter, 11(1), pp. 32-38.

Gudivada, V., Apon, A. & Ding, J., 2017. Data quality considerations for big data and machine learning: Going beyond data cleaning and transformations. International Journal on Advances in Software, 10(1).

Hakim, D. & Darari, F., 2021. Data Completeness Impact on Deep Learning Based Explainable Recommender Systems. Online, International Conference on Information and Communications Technology (ICOIACT).

Hattemer, A., 2021. Change Data Capture is having a moment. Why?. [Online]
Available at: https://materialize.com/change-data-capture-is-having-a-moment-why/
[Accessed 20 December 2021].

Heidari, A., McGrath, J., Ilyas, I. & Theodoros, R., 2019. Holodetect: Few-shot learning for error detection. Amsterdam , Proceedings of the 2019 International Conference on Management of Data.

Ilyas, A. et al., 2019. Adversarial examples are not bugs, they are features. arXiv preprint arXiv:1905.02175.

ISO/IEC, 2008. Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- Data quality model.

ISO/IEC, 2015. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of data quality.

Johnson, J. & Khoshgoftaar, T., 2019. Survey on deep learning with class imbalance. Journal of Big Data, 6(1), pp. 1-54.

Kaggle, 2012. Titanic - Machine Learning from Disaster. [Online]
Available at: https://www.kaggle.com/c/titanic
[Accessed 2021 December 2021].

Karlaš, B. et al., 2020. Nearest neighbor classifiers over incomplete information: From certain answers to certain predictions. arXiv preprint arXiv:2005.05117.

Kläs, M. & Vollmer, A., 2018. Uncertainty in machine learning applications: A practice-driven classification of uncertainty. Västerås, Sweden, International Conference on Computer Safety, Reliability, and Security.

Kleppmann, M., 2017. Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. : O'Reilly Media, Inc..

Koh, P. & Liang, P., 2017. Understanding black-box predictions via influence functions. Sydney, International Conference on Machine Learning.

Kołcz, A., Chowdhury, A. & Alspector, J., 2003. Data duplication: An imbalance problem?. Citeseer.

Krawczyk, B., 2016. Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence, 5(4), pp. 221--232.

Krishnan, S., Franklin, M., Goldberg, K. & Wu, E., 2017. Boostclean: Automated error detection and repair for machine learning. arXiv preprint arXiv:1711.01299.

Krishnan, S. et al., 2016. Activeclean: Interactive data cleaning for statistical modeling. Proceedings of the VLDB Endowment, 9(12), pp. 948--959.

Krishnan, S. & Wu, E., 2019. Alphaclean: Automatic generation of data cleaning pipelines. arXiv preprint arXiv:1904.11827.

Kumeno, F., 2019. Sofware engneering challenges for machine learning applications: A literature review. Intelligent Decision Technologies, 13(4), pp. 463-476.

Lakshmanan, V., Robinson, S. & Munn, M., 2020. Machine Learning Design Patterns. O'Reilly Media, Inc..

Laurikkala, J. et al., 2000. Informal identification of outliers in medical data. Berlin, Fifth international workshop on intelligent data analysis in medicine and pharmacology.

LeCun, Y. & Bengio, Y., 1995. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10), p. 1995.

Ledesma, S. et al., 2018. Analysis of data sets with learning conflicts for machine learning. IEEE Access, Volume 6, pp. 45062-45070.

Liu, X., Cheng, G. & Wu, J., 2002. Analyzing outliers cautiously. IEEE Transactions on Knowledge and Data Engineering, 14(2), pp. 432-437.

Liu, Z. & Zhang, A., 2020. Sampling for big data profiling: A survey. IEEE Access, Volume 8, pp. 72713-72726.

Liu, Z., Zhou, Z. & Rekatsinas, T., 2021. Picket: guarding against corrupted data in tabular data during learning and inference. VLDB Journal, pp. 1-29.

Lwakatare, L. E., Rånge, E., Crnkovic, . I. & Bosch, J., 2020. On the experiences of adopting automated data validation in an industrial machine learning project. Online, IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).

Mahdavi, M. & Abedjan, Z., 2020. Baran: Effective error correction via a unified context representation and transfer learning. Online, Proceedings of the VLDB Endowment, pp. 1948-1961.

Mahdavi, M. et al., 2019. Raha: A configuration-free error detection system. Amsterdam , Proceedings of the 2019 International Conference on Management of Data, pp. 865-882.

Mariet, Z., Harding, R. & Madden, S., 2016. Outlier detection in heterogeneous datasets using automatic tuple expansion, Cambridge: MIT-CSAIL-TR-2016-002.

Neutatz, F., Mahdavi, M. & Abedjan, Z., 2019. ED2: Two-stage Active Learning for Error Detection. arXiv preprint arXiv:1908.06309.

Orr, L. et al., 2021. Managing ML pipelines: feature stores and the coming wave of embedding ecosystems. arXiv preprint arXiv:2108.05053.

Rekatsinas, T., Chu, X., Ilyas, I. & Ré, C., 2017. Holoclean: Holistic data repairs with probabilistic inference. arXiv preprint arXiv:1702.00820.

Russell, B., Torralba, A., Murphy, K. & Freeman, W. T., 2007. LabelMe: a database and web-based tool for image annotation. International Journal of Computer Vision.

Schaefer, L., 2021. What is NoSQL?. [Online]
Available at: https://www.mongodb.com/nosql-explained
[Accessed 20 December 2021].

Schelter, S. et al., 2019. Unit testing data with deequ. Amsterdam, International Conference on Management of Data, pp. 1993-1996.

Schelter, S. et al., 2018. Automating large-scale data quality verification. Proceedings of the VLDB Endowment.

Schofield, A., Thompson, L. & Mimno, D., 2017. Quantifying the effects of text duplication on semantic models. Copenhagen, Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.

Sessions, V. & Valtorta, M., 2006. The Effects of Data Quality on Machine Learning Algorithms. ICIQ, Volume 6, pp. 485--498.

Shvachko, K., Liang, C. & Dzinamarira, S., 2021. The exabyte club: LinkedIn's journey of scaling the Hadoop Distributed File System. [Online]
Available at: https://engineering.linkedin.com/blog/2021/the-exabyte-club--linkedin-s-journey-of-scaling-the-hadoop-distr
[Accessed 20 December 2021].

Tensorflow, 2021. Get started with Tensorflow Data Validation. [Online]
Available at: https://www.tensorflow.org/tfx/data_validation/get_started
[Accessed 19 December 2021].

Tepić, M., Abdelaal, M., Weber, M. & Rothermel, K., 2020. AutoSec: Multidimensional Timing-Based Anomaly Detection for Automotive Cybersecurity. Online, IEEE 26th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).

Toneva, M. et al., 2018. An Empirical Study of Example Forgetting during Deep Neural Network Learning. arXiv preprint arXiv:1812.05159.

Vajjala, S., Majumder, B., Gupta, A. & Surana, H., 2020. Practical Natural Language Processing: A Comprehensive Guide to Building Real-World NLP Systems. O'Reilly Media.

Visengeriyeva, L. & Abedjan, Z., 2018. Metadata-driven error detection. Bozen-Bolzano, Proceedings of the 30th International Conference on Scientific and Statistical Database Management.

Wang, R. & Strong, D., 1996. Beyond accuracy: What data quality means to data consumers. Journal of management information systems, 12(4), pp. 5-33.

Wang, T., Zhu, J., Torralba, A. & Efros, A., 2018. Dataset distillation. arXiv preprint arXiv:1811.10959.

Wells, D., 2020. What Is a Data Catalog?. [Online]
Available at: https://www.alation.com/blog/what-is-a-data-catalog/
[Accessed 20 December 2021].

Wikipedia, 2021. Pearson correlation coefficient. [Online]
Available at: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient
[Accessed 15 November 2021].

Xin, D., Miao, H., Parameswaran, A. & Polyzotis, N., 2021. Production Machine Learning Pipelines: Empirical Analysis and Optimization Opportunities. Online, Proceedings of the 2021 International Conference on Management of Data.

Yoon, J., Arik, S. & Pfister, T., 2020. Data valuation using reinforcement learning. Online, International Conference on Machine Learning, pp. 10842-10851.

Zhao, B., Mopuri, K. & Bilen, H., 2020. Dataset condensation with gradient matching. arXiv preprint arXiv:2006.05929.