Industrial Machine Learning for Enterprises

Deliverable D2.2

First Version of 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:	D2.2
Nature of deliverable:	Report
Dissemination level:	PU
Internal version number:	1.0
Contractual delivery date:	2022-05-31
Actual delivery date:	2022-05-31
Responsible partner:	Software AG

Contributors

Editor(s)	Mohamed Abdelaal (Software AG)
Contributor(s)	Mohamed Abdelaal (Software AG), Timo Sinisalmi (Basware), Luca Szegletes (BUTE), Dorian Knoblauch (Fraunhofer), Heikki Ihasalo (Granlund), Anna Korolyuk (Granlund), Harry Souris (Silo AI), Kimmo Sääskilahti (Silo AI)
Quality assuror(s)	Timo Sinisalmi (Basware), Janis Lapins (Spicetech)

Version history

Version	Date	Description
1.0	22-05-30	Version for publication

Abstract

The deliverable provides an overview of the research activities within IML4E WP2 after the first year of the project. The deliverable covers the first version of methods and tools developed to realize the objectives of the main three tasks in WP2, including data preparation automation (Task 2.1), data management and version control (Task 2.2), and continuous data quality assurance (Task 2.3). The sections of the deliverable are structured into four main themes. The first theme is an industry viewpoint discussion of existing approaches, and a motivation for the work presented in this deliverable. The other three themes cover the research tasks T2.1, T2.2, and T2.3, respectively.

Regarding data preparation automation, we introduce three different methods for automatically clean and prepare different data models, including structured data and 2D images. For instance, we propose a meta learning-based error detection method for structured data. Another contribution is a privacy-friendly preparation workflow for 2D images processed by AI models. Moreover, we develop a novel ML-based data cleaning method for tackling data with noisy labels. For data version control, we define a set of criteria and characteristics which has to be considered while designing or comparing data version control solutions. Moreover, we provide an outlook for reference implementations that we plan to build this year. Finally, we provide a catalogue of data quality attributes which can be used to continuously monitor data quality throughout the lifecycle of AI projects.

Keywords

Data preparation, data valorisation, anomaly detection, data version control, data quality assurance, ML certification, quality metrics, data cleaning, data privacy

Executive Summary

The overall objective of IML4E WP2 is to develop tools and techniques to facilitate the collection, valorisation, and processing of the data collected for machine learning applications. In other words, we strive to develop high quality and interoperable data preparation infrastructures for trustworthy ML. To this end, we divide this main objective into three tasks, involving (I) the development of a set of required technologies to accomplish the data collection and data preparation in an automated way; (II) the development of a version control procedure and system guaranteeing the versioning of data sets and intermediate files; and finally (III) the development of a continuous data quality process and ensuring 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.

This deliverable reports on the WP2 results after the first year of the project. The WP2 research tasks that were initiated at the beginning of the project, and that are covered by this deliverable, are (Task 2.1) data preparation automation, and (Task 2.2) data management and version control solutions, and (Task 2.3) continuous Data quality assurance. Before reporting on the current results of these tasks, we give in Section 2 an industry-viewpoint discussion of existing approaches to motivate the work that is presented in the remainder of the deliverable. The section complements the presentation of the state of the art in deliverable D2.1.

Section 3 to Section 6 cover the automation of data preparation techniques and tools. In particular, Section 3 introduces a new meta learning-based data cleaning method dedicated to structured data. The main contribution of this work is to exploit already-existent knowledge in historical datasets to generate a set of features leveraged to train an error detection model. Section 4 presents the specifications and criteria for developing data quality dashboards for monitoring the building energy consumption data. Section 5 elaborates on our endeavours to preserve the privacy of individuals while training AI models on digital images that include their faces. Section 6 discusses the development of an automated data cleaning method, referred to as Mosquito, which can efficiently detect anomalies in the ground truth data using formal grammars and machine learning models.

Section 7 introduces our recent work toward standardizing the data quality solutions. Specifically, we introduce a catalogue of data quality attributes which can be employed to automatically define the application-relevant quality metrics. Section 8 discusses several criteria and characteristics while designing or comparing data version control solutions. Afterwards, the section briefly presents three use cases that will be involved in the development of a data version control. Such use cases comprise (I) human pose estimation from images and versioning solution with DVC, (II) large-scale information extraction from PDFs, and (III) weather prediction.

Table of contents

1 Introduction

A main objective of IML4E WP2 is to develop techniques and tools for the automation of data preparation and data version control in machine learning pipelines. To fulfil this objective, we are conducting R&D activities in two main directions. First, we are developing techniques and tools for automated data profiling, data cleaning, and continuous quality assurance. In this context, we are eager to develop a unified data preparation infrastructure which can seamlessly deal with different data modalities, including structured data, e.g., sensor readings, financial records, medical reports, as well as unstructured data, such as digital images, videos, and log files. Moreover, we focus on developing tools and techniques to enable easy interaction between users and the data quality methods. This goal can be achieved through developing a novel data quality dashboard which can clearly visualize the various data quality problems, propose curation techniques, and enable users to monitor and to make decisions to solve the data quality problems. We envision the data quality dashboard to employ advanced data profiling, error detection, and error repair methods. This deliverable provides an overview of the work done to realize such tools and techniques with highlighting some obtained results.

The second task of WP2 is to develop a novel data version control systems which fits well with the requirements of industrial applications. The main contribution of this line of research is to develop a solution which can deal with data coming from different sources with distinct sizes. To this end, we need to enhance the flexibility of the planned solution to adaptively react to the application's requirements. The deliverable presents a set of characteristics which we shall be considered while designing or comparing various data version control solutions.

The main IML4E research questions that we tackle in WP2 are the following:

- To what extent we can automate the task of detecting errors in structured and unstructured data?
- What are the most relevant quality metrics necessary to ensure continuous quality assurance and how to efficiently and precisely measure these quality metrics?
- 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?

The results presented in this deliverable investigate all these research questions. The deliverable is structured into three main themes, namely an industry-viewpoint discussion of existing approaches, techniques for data quality and dashboards, and techniques for data version control. In particular, Section 2 gives a motivation of the WP2 work from an industrial perspective by reviewing existing methods, thereby addressing the first theme. The section complements the overview of the state of the art presented in deliverable D2.1. The second theme is addressed in the subsequent four sections. In Section 3, we elaborate on a novel error detection method which makes use of the knowledge embedded in historical data to determine the indices of the dirty data. To this end, it trains a set of base classifiers and a meta classifier which can differentiate between clean and dirty data. Section 4 introduces several requirements and criteria for developing an interactive data quality dashboard. Section 5 elaborates on our privacy-friendly data preparation method dedicated to digital images. Along a similar line, Section 6 presents several generations of the Mosquito data cleaning method which tackles the mislabelling problems while dealing with unstructured data such as PDFs, images, and log files. Section 7 provides definition of some important quality metrics for the sake of enabling the automation of continuous quality assurance. Moreover, the section provides means for properly measure such quality metrics. The third theme is addressed in Section 8 which provides a set of characteristics and question need to be addressed while designing a data version control solution suitable for a certain application.

2 Overview of Industrial Data Quality Tools

In this section, we provide an industry-viewpoint discussion of existing data quality tools to motivate the work that is presented in the remained of the deliverable. The section complements the presentation of the state of the art in deliverable D2.1.

2.1 Data Quality Dashboards

A dashboard is typically used to provide a summary and at-a-glance view of relevant information. Various data visualization methods, such as graphs, colours, and metrics, are utilized in highlighting the most important information. One often used method is key performance indicator (KPI) which presents performance against a target, for example in a percentage value. This section presents features of dashboards in three data quality tools, Informatica, SAP Information Steward and Talend Data Catalog. All the tools are positioned as leaders in Gartner® Magic Quadrant™ in year 2021.

2.1.1 Informatica

Informatica Cloud Data Profiling service focuses on creation of data profiling tasks and monitoring of data quality. The tool provides following statistics of the data:

- Number of distinct, non-distinct, and null values
- Percentage of distinct, non-distinct, null, zero, and blank values
- Documented and inferred data types
- Number of patterns
- Percentage of top pattern
- Maximum and minimum length of values
- Maximum and minimum values
- Average, sum, and standard deviation for numeric data types
- Value frequencies
- Outliers

The statistics are shown in a summary view (cf. Figure 1) and the results can be sorted by columns. In addition to the summary view, the tool user can drill down to see more detailed results, view historical results, run queries to view rows that have data quality issues and compare multiple columns. With Informatica it is also possible to add rules to the profile. There exist ready-made rules, for example replacing incorrect values or removing unwanted values, as well as possibility to create own rules.

2.1.2 SAP Information Steward

SAP Information Steward tool gives statistical analysis of data. It shows the following information on data quality:

- Value; min, max, average, and median value of column
- String length; min, max, average, and median value of column
- Completeness; percentage of nulls, percentage of blanks, percentage of zeros
- Distribution; value - distribution of records, pattern - number of patterns, word - usage of words

An example of SAP Information Steward data profiling is shown in Figure 2. Similar to Informatica, SAP Information Steward allows to profile data against rules. The tool provides library of rules, but users can also create custom rules by themselves (figure 3). Rule results are shown as score from 0 to 10 describing how much of the data passes the rule.

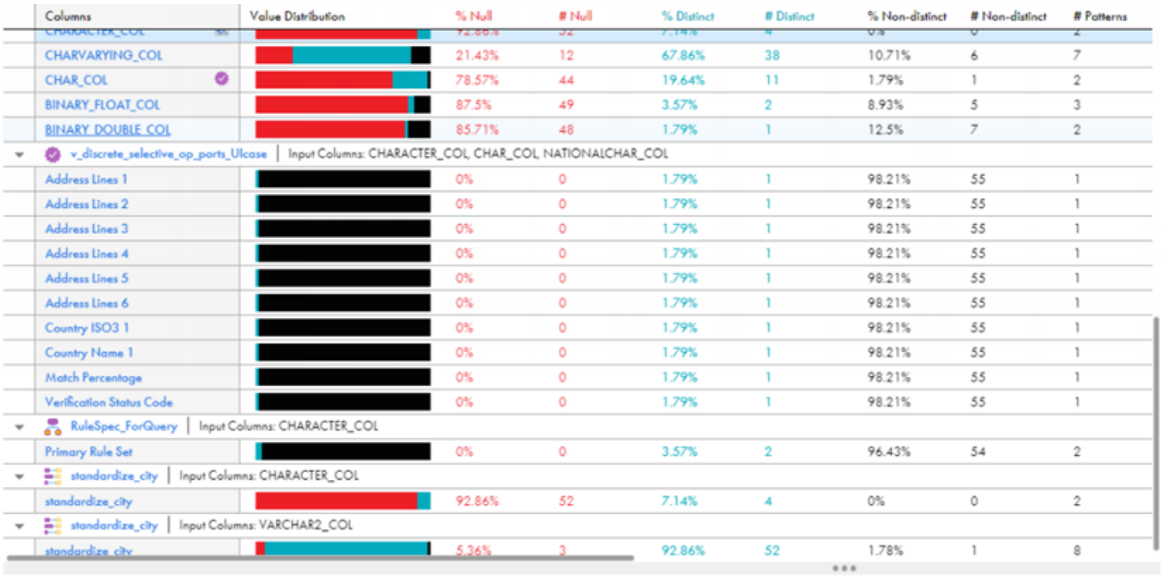

Figure 1. Informatica summary view (Informatica 2021)

Figure 2. SAP Information Steward data profiling statistics (Pinjwani 2021)

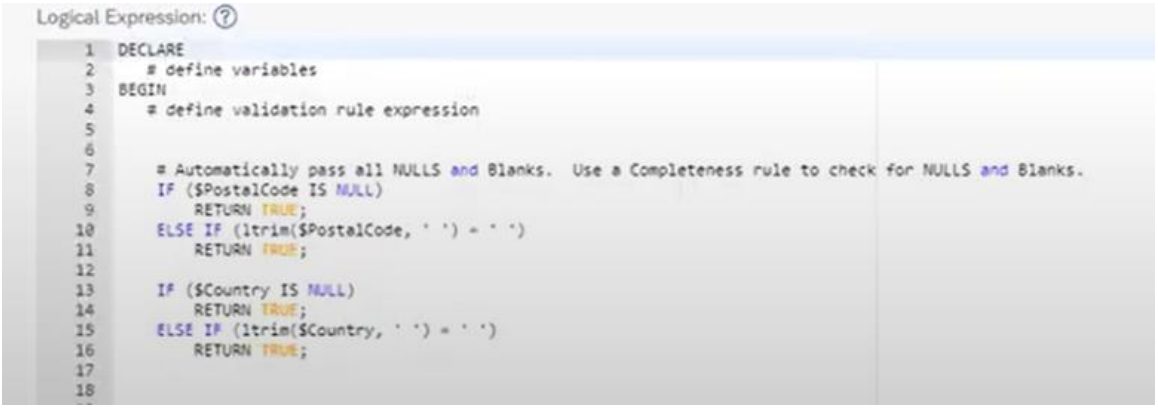

Figure 3. Creating data quality rule in SAP Information Steward (Lintelman 2020)

2.1.3 Talend Data Catalog

Talend Data Catalog reports and dashboards enable easy view of data quality trends and possible quality issues. With the tool, it is possible to make custom quality dashboards as shown in Figure 4.

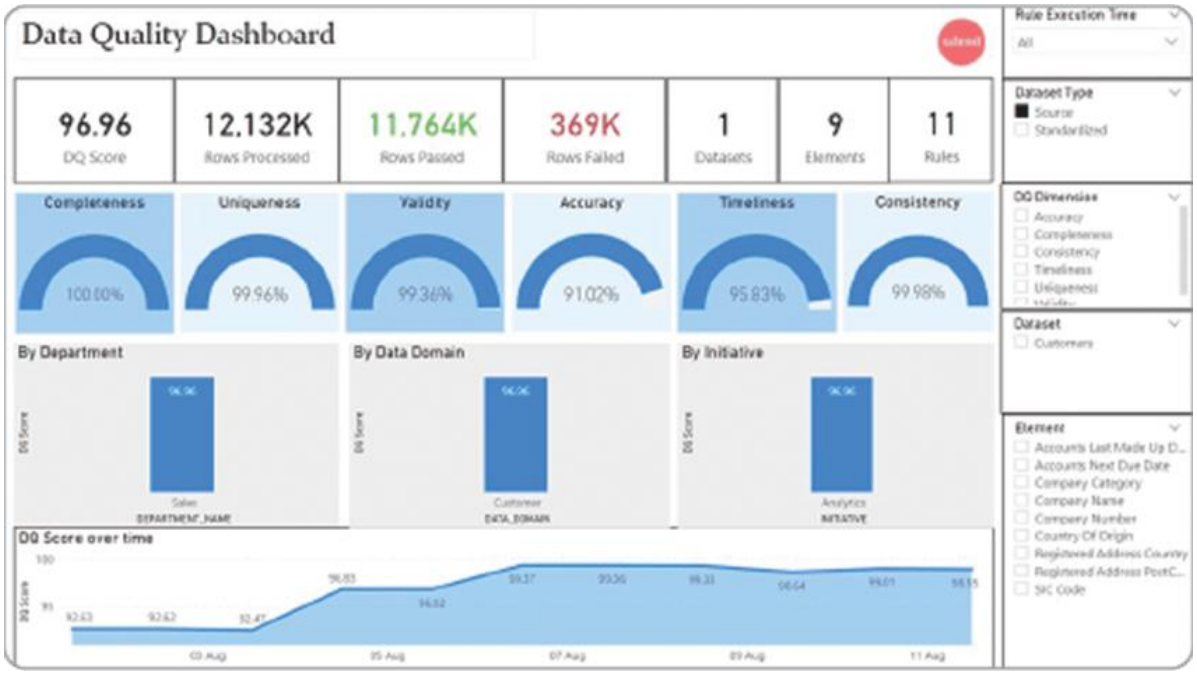

Figure 4. Talend data quality dashboard (Talend Team)

The tool can display the following data profiling details:

- Count, number of rows actually profiled
- Distinct, non-distinct=total-distinct-empty
- Duplicate, duplicate rows in database or in files.
- Valid, valid rows in database or in files.
- Empty, null rows in database or empty rows in files.
- Invalid, invalid rows in database or in files
- Average length, average length of values.
- Min length, minimum length of values.
- Max length, maximum length of values.
- Frequency, distribution of values and their frequency as a percentage
- Patterns, list of different patterns of data presentation discovered in the source and their frequency as a percentage (Talend)

2.2 Data Cleaning Platforms for Structured Data

In this section, we highlight a set of commercial data quality tools which can strengthen the quality, applicability, and value of the collected data. The selected list represents the most popular and top-rated data quality tools. Most of these tools are included in Gartner magic quadratic reports for data quality solutions[1]. The list comprises the following tools: OpenRefine, Trifacta, and Tamr. It is worthwhile mentioning that there exist several other tools, such as DataCleaner, TIBCO Clarity, Winpure, Cloudingo, Melissa Clean Suite, and IBM Infosphere Quality Stage. However, we have not considered them in this deliverable to make it more concise and to focus on the tools which highly related to our planned contributions.

[1] https://www.gartner.com/en/documents/3988016

2.2.1 OpenRefine

It was developed by Google as an open-source data tool which enables users to transform data between different formats and clean them from different data problems[2]. Data exploration is performed through two operations, namely faceting and filtering. In the former operation, OpenRefine generates a widget, for each column, that displays all the distinct values in the column and their number of occurrences. Moreover, OpenRefine enables users to define expressions, e.g., the values in a column should be greater than a threshold, on multiple columns for faceting. In this case, OpenRefine generates the widget based on the values of the expression. In the latter operation, users can select one or more values in the widget, before filtering the rows that do not contain the selected values.

Aside from data exploration, OpenRefine also enables data cleaning via an editing operation. Specifically, users can either edit one cell at a time or edit a group of cells at a time via modifying the corresponding text facet. For instance, if the widget displays two facets "USA" and "US", then modifying one of them will propagate the modification to all cells of this facet. It is worthwhile mentioning that OpenRefine enables working with the data on local machines. Accordingly, it can be used to clean confidential data without worrying about data privacy. However, it assumes that users have a relatively high level of technical know-how.

2.2.2 Trifacta Wrangler

Trifacta offers users a variety of self-service data operations, such as data discovery, cleaning, enriching, and validating[3]. Such a tool can transform structured or unstructured datasets stored in CSV, JSON, relational table formats, or SaaS application data of any size. For structured data, Trifacta can automatically detect schemas, data types, possible joins, and anomalies such as missing values, outliers, and duplicates. Moreover, it uses a proprietary inference algorithm to interpret the data transformation intent of a user's data selection. Users can define data quality rules to monitor the accuracy, completeness, consistency, validity, and uniqueness of the data. It is worthwhile mentioning that Trifacta leverages Google Cloud processing engine to accelerate the data transformation process. However, it can be slow while uploading the data and preprocessing them. A major drawback of Trifacta is lacking connectors to data sources such as Google Analytics which implies that data extraction process often needs to be handled externally. Moreover, the free version is relatively limited, and it is not feasible to try the full version before buying it.

2.2.3 Tamr

It is a tool developed by Tamr Inc., a startup focusing on large-scale data integration and cleaning[4]. Such a tool tackles problems such as duplicate records, entity consolidation, and data integration. For instance, Tamr performs entity consolidation when multiple records have data for the same entity. Specifically, Tamr employs a continuous data cleaning process, where users are involved in the process to carry out several tasks, including: (I) assessing the quality of analytics and translate user feedback into feedback on the underlying data sources; (II) generating training data for the back-end machine learning models, (e.g., de-duplication models); and (III) administrating the repairing actions on the underlying data sources given the feedback on analytics and the learned models. Such user involvement usually comes at a price where the tool requires skilled users to interact with the Tamr APIs efficiently and properly.

2.3 Data Version Control Solutions

In this section, we give an overview of selected data version control solutions. The tools selected here have been chosen based on their perceived adoption and to select tools covering multiple scales and diverse use cases, ranging from small-scale projects with structured data to large-scale projects with unstructured data. The tools selected for this overview, include DVC, LakeFS, Pachyderm, Dolt, Delta Lake, and Feast.

[2] https://openrefine.org
[3] https://www.trifacta.com/
[4] https://www.tamr.com/

In general, metadata stores, such as MLFlow and Neptune, are commonly used together with data version control tools. During the experiment run, data version information is logged to the metadata store. Metadata stores can also be used for data versioning by, for example, logging dataset snapshots as training artifacts. These approaches are not discussed here. Finally, MLOps platforms provided by cloud vendors such as Amazon SageMaker, Google Cloud Vertex AI and Azure ML provide their own solutions for data lineage tracking. These solutions are not discussed here but should be considered when using such platforms.

2.3.1 DVC

DVC is an open-source tool[5], developed by iterative.ai, that tracks datasets and machine learning projects, works with many types of storages (Amazon S3, Azure Blob Storage, Google Drive, Google Cloud Storage, local, HDFS, HTTP, network attached storage, etc.), and runs on top of Git repositories. It also supports building and running pipelines. DVC brings agility, reproducibility, and collaboration into existing data science workflows. It makes projects reproducible and shareable.

If we look at similar data versioning solutions, like Git and Git-LFS (Git large file storage), they can be challenging to use, as Git itself is not suitable for big files (100 MB is the maximum size that can be uploaded and tracked), and even Git-LFS has a file size limit of 5GB. These constraints cannot be met in typical machine learning or data engineering projects, where data can be multiple times the maximum size supported by these systems. DVC also uses *reflinks* or *hardlinks* to avoid copy operations on checkouts, thus handling large data files much more efficiently. The way that DVC is handling data versioning tasks, is by looking at the data, and creating a small metadata file, that will be versioned by Git. This file contains information about the dataset, so it can be tracked. In the meantime, DVC automatically adds the data itself to the repository's *.gitignore* file, therefore it will never be pushed to remote, instead it will be pushed to the storage you specify. Changes to the data result in changes to the metadata file created by DVC and tracked by Git.

In addition, DVC supports the creation of data pipelines, series of data processes that produce a final result, like ETL workflows or machine learning pipelines. These pipelines can also be versioned by Git, therefore allowing the reproducibility of the workflow later. Tracking metrics, updating parameters, and visualizing performance are also easy with DVC. All these tasks can be combined into experiments which can then be compared, making it very easy to experiment with different ML model parametrization for example. DVC Studio, a product also made by iterative.ai, is an official online platform for DVC. It can be used to visualize and share results, experiments, and pipelines[6]. DVC can be used as a command line tool or by using its Python API[7]. The command line tool uses similar commands to Git, so people familiar with Git commands will not have any difficulties learning DVC commands. Furthermore, many easy-to-follow tutorials are available on their website[8].

2.3.2 lakeFS

LakeFS is an open-source project used to implement versioned data lakes. The documentation describes the tool as a "project that provides a git-like version control interface for data lakes, with seamless integration to most data tools and frameworks." The tool enables data reproducibility, rollback, and automated data quality validation. The tool can be tried out in the playground environment without installing the tool[9]. An interactive tutorial on Katacoda is also available[10]. lakeFS deployment includes a central data management server. The server provides a web UI for managing and browsing the data. The command-line tool `lakectl` can be used to manage

[5] https://github.com/iterative/dvc
[6] https://studio.iterative.ai
[7] https://dvc.org/doc/api-reference
[8] https://dvc.org/doc/start
[9] https://demo.lakefs.io/
[10] https://docs.lakefs.io/quickstart/

resources from the command-line. There are multiple integrations available[11] to access data from, for example, Python, Spark, or Kubeflow Pipelines components.

lakeFS uses an object storage as the underlying object storage, typically a cloud provider's object storage. lakeFS uses a PostgreSQL database to synchronize actions on repositories. In production environments in the cloud, using a managed SQL database is recommended. lakeFS server API is compatible with the S3 object storage API. Therefore, any tool that can read data from S3, can also read from lakeFS. The S3 data path such as `s3://data-bucket/collections/foo` only needs to be supplemented with the branch name, resulting in path names such as `s3://data-bucket/main/collections/foo`.

lakeFS manages data in Git-like manner in repositories and branches. Developers can checkout data from branches created by others or create new branches for their experiments. New data is committed atomically to prevent inconsistent data views. Users can add data to a branch in a repository with lakectl, through the web UI or with tools such as `aws s3`. After data is added, it is atomically committed to the branch with a commit message. Committed data can be accessed either via the S3-compatible API, with `lakectl`, or one of the available integrations.

2.3.3 Pachyderm

Pachyderm provides a solution for data versioning and pipelines in MLOps. Pachyderm is not fully open-source but "source-available", limiting the use for purposes that would compete with Pachyderm[12]. Users can choose between the Community Edition, Enterprise Edition and Pachyderm Hub. The free Community Edition is a limited version of the Enterprise Edition. Pachyderm Hub is a fully managed service. Pachyderm is deployed as a Helm application on Kubernetes. For data storage, PostgreSQL database and object storage are required. In cloud environments, using the cloud provider's object storage and managed SQL is recommended.

Data in Pachyderm is organized in a Git-like structure that enables team collaboration through repositories, branches, commits, and rollbacks. Typically, each dataset is its own repository. A commit is an immutable snapshot of data corresponding to a change in source data or transformations. Branch in Pachyderm points to the state of its repository at a particular commit, updating as new data is added, and tells Pachyderm what input the branch depends on. Files are the actual data in the repository. Files can be any type and size. Data is stored in Pachyderm as native objects, not metadata pointers like in many data versioning tools. This ensures strong data versioning guarantees. This kind of file-based versioning provides, for example, a complete audit trail for all data and artifacts across pipeline stages, including intermediate results.

Pachyderm Pipeline System can be used to perform transformations on the versioned data. User defines a pipeline specification and creates a pipeline that waits for certain conditions to be met, for example, for new data being added to any repositories. When new data arrives, a pipeline performs an operation and processes the data. The command-line tool pachctl can be used for interacting with the Pachyderm cluster. The tool is used to manage repositories, branches, and files. The typical developer workflow involves "adding data to versioned data repositories, creating pipelines to read from those repositories, executing the pipeline's code, and writing the pipeline's output to other data repositories''. Pachyderm handles the code execution according to the pipeline specification. Data in Pachyderm is automatically mounted to the pipeline containers. Data can be accessed and exported using pachctl, writing data to an external datastore as a pipeline step, or mounting the repository to a local computer.

2.3.4 Dolt

Dolt is an SQL database that one can fork, clone, branch, merge, push and pull just like a Git repository. Dolt tracks every change made to the data in the database: who made it, what the commit message was, and what the previous values were. This makes Dolt a good fit for data provenance and auditing use-cases. In Dolt, data

[11] https://docs.lakefs.io/integrations/

[12] https://www.pachyderm.com/community-license-faq/

snapshots are automatic. Every commit is a snapshot to which users can time-travel for backup, disaster recovery, or reproducibility. Dolt also gives tools for viewing the difference between two versions. Users can update the data located in a Dolt database in a separate branch. Once the user is happy with the changes, they can merge the changes to the main branch or submit the change for peer review. This workflow ensures that data is updated in as controlled manner as code. Dolt databases including schemas and views are also easy to share with other teams.

One use-case for Dolt is to completely replace PostgreSQL or MySQL database backing an application. This is especially useful in cases when the application uses the database in read-only mode and data is added to the database in well-defined batches on some regular schedule. Dolt's branching and commit features ensure that data is added in a controlled manner. The trade-off is that Dolt is 2-20 times slower than SQL[13]. Data in Dolt is managed in repositories similar to Git repositories. A Dolt repository has a ".dolt" folder (again similar to ".git" folder), containing the state and history of the database. Dolt's storage engine implementation is based on Noms project (link) and data structures called Prolly trees[14]. Dolt repositories can be hosted either on DoltHub, local filesystem or on cloud object storage systems such as Google Cloud Storage and Amazon S3.

Dolt is used through a command-line interface "dolt". The dolt CLI has the same commands as Git and some extra commands. The CLI can be used to, for example, add and commit data to a Dolt repository, run SQL queries against tables in the repository and to start a MySQL-compatible server that can be queried with MySQL clients. In addition to the dolt CLI, Dolt can also be used via the Python library "doltpy". The library is a wrapper around the dolt CLI and can be used, for example, to connect to and interact with an existing repository. In this way, users can write and read data in any branch in the repository.

2.3.5 Delta Lake

Delta Lake is an open-source storage layer for data lakes. It uses Spark's distributed processing power to handle versioning metadata for billions of files. Delta Lake adds ACID transactions (atomicity, consistency, isolation and durability) to the data lake, allowing row-level inserts, updates and deletes. This ensures that data readers never see inconsistent data. Delta Lake can also enforce schema integrity on writes to ensure data integrity. Delta Lake stores data into so-called Delta tables. Under the hood, a Delta table consists of files (written in open Parquet file format) and a transaction log. The files are stored to a configurable storage location such as Amazon S3, Azure Blob Storage, Google Cloud Storage, or HDFS[15]. Transaction log keeps track of all the commits made to the table to provide ACID transactions. Table metadata can also be written to a Hive metastore.

Delta Lake provides unified batch and stream processing by Apache Spark Streaming integration. Delta table is both a batch table as well as a streaming data source and sink. Delta Lake provides very powerful time travel capabilities powered by the transaction log. Old snapshots of a Delta table can be queried either by version number or timestamp. By default, the retention period is 30 days. Delta Lake does not contain any server component itself, but it does require Apache Spark, typically run in a separate cluster. Delta tables can be queried and manipulated with Apache Spark's reader and writer APIs, using SQL, Scala, or Python[16]. Delta tables can also be queried by external query engines such as Presto or Trino. Delta Lake also has integrations to data processing engines such as Apache Flink and Apache Kafka, allowing reading and writing data from various external sources.

2.3.6 Feast

Feast is an open-source feature store. The project was originally developed by Gojek with Google Cloud. Today, the primary contributor to the project is Tecton, who also offers a fully managed feature store named Tecton built on top of Feast. Feature stores are in a good position to work as data versioning tools, because data is accessed and made available to training through them. Feast version 0.18 added experimental support for Saved Datasets, which allow training dataset snapshots to be persisted for later retrieval. The main use case of Saved

[13] https://docs.dolthub.com/reference/sql/latency#benchmark-data
[14] https://docs.dolthub.com/architecture/storage-engine
[15] https://docs.delta.io/latest/delta-storage.html
[16] https://pages.databricks.com/rs/094-YMS-629/images/Delta%20Lake%20Cheat%20Sheet.pdf

Datasets was to provide reference datasets for validating new training and serving data, but the snapshots can also be used for data versioning.

Feast version 0.9 and earlier was based on a complex infrastructure stack, including Kubernetes and Apache Spark. In version 0.10, the project was re-written to become SDK/CLI centric software that can deploy or configure infrastructure. Feast is installed with pip, the package installer for Python. The configuration for a Feast project is stored in a central location called a feature repository. It is the declarative source of truth for the desired state of the feature store. Users interact with the repository using the Feast CLI and the Feast Python library.

Feast uses a file-based feature registry as a central catalogue of all feature definitions and their related metadata. This registry is stored in an object store such as Google Cloud Storage or Amazon S3. For serving historical features, Feast requires an offline store used as a storage and compute system. Typical choices for an offline store include data warehouses such as Google Cloud BigQuery, Amazon Redshift and Snowflake. Note that it is not possible to query any type of data sources from all offline stores: for example, when BigQuery is used as an offline store, it is typical to query features from another table in BigQuery. For serving online features, Feast requires an online store that's optimized for low-latency access. Values are periodically materialized from an offline store to online store. Typical choices for an online store include Redis, Google Cloud Datastore and Amazon DynamoDB. See the figure below for detailed architecture of Feast.

2.4 Discussion

As one can see in the sections above, there exist several tools and techniques for data quality and data version control. However, such tools and techniques do not fit well with the requirements and specifications described in the proposal of IML4E. For instance, the reported data cleaning tools are partially automated, which implies that users have to be involved in the process and in case of lack of skilled users, the quality of such tools are negatively affected. It is important to mention that none of the methods described below consider the requirements of downstream applications, e.g., ML modelling. They simply tend to cure the data quality problems regardless of how these data will be consumed.

Similarly, the reported data version control solutions exhibit useful features for specific sector of applications. However, they lack the enough flexibility to deal with different problems which involve fluctuating requirements. Some industrial applications may change their requirements and demands according to the workload or the level of available skilled workers. In this case, the data management solutions have to dynamically react to these changes seamlessly. Moreover, the current solutions may face challenges while dealing with different data modalities. Therefore, we envision designing and developing a novel solution which can efficiently overcome these challenges.

3 Meta Learning-Based Error Detection for Structured Data

In this section, we address the challenge of automatically detecting discrepancies and noise in structured data. In general, structured data are those data originated from sensors, IoT devices, medical records, financial reports, etc. They are typically stored in relational databases which simplify various data operations, such as data search, update, and analysis. In fact, real structured data, generated from real-world applications, usually suffer from different types of discrepancies, which hinder the utilization of such data in AI applications. For instance, the data collected manually in a certain application, e.g., customers and purchase records, may suffer from missing values due to lack of information or due to data entry errors. Along a similar line, data duplicates may occur due to improper join operations. Examples of other types of discrepancies comprise outliers, rule/pattern violation, mislabelling, inconsistencies, and typos. Moreover, structured data may suffer from white noise emerged from noisy contamination and interference of the communication channels.

3.1 ML-based Error Detection

In this work, we tackle such problems through automatically identifying the contaminated records in structured data. To this end, we lean on semi-supervised Machine Learning models to differentiate between clean and contaminated records. Figure 5 depicts the different processes necessary to formulate the error detection task as a binary classification problem. At the outset, each data cell is used as an input to an automatic featurization module. The purpose of such a module is to extract a set of features which can be utilized to train a detection model. Examples of such features include metadata, statistical conclusions, word embeddings, and/or results of simple error detection methods. A subset of these features is typically selected to be labelled by an oracle who assigns a label of one to contaminated records and zero to clean records. Afterward, a detection classifier, e.g., random forest, XGBoost, or multi-layer perceptron classifier, is trained on the labelled records. To detect the contaminated records in the unlabelled data, the detection classifier is employed to predict the label of each unlabelled feature vector. Moreover, such a detection classifier can be later used to detect errors in the newly arrived serving data.

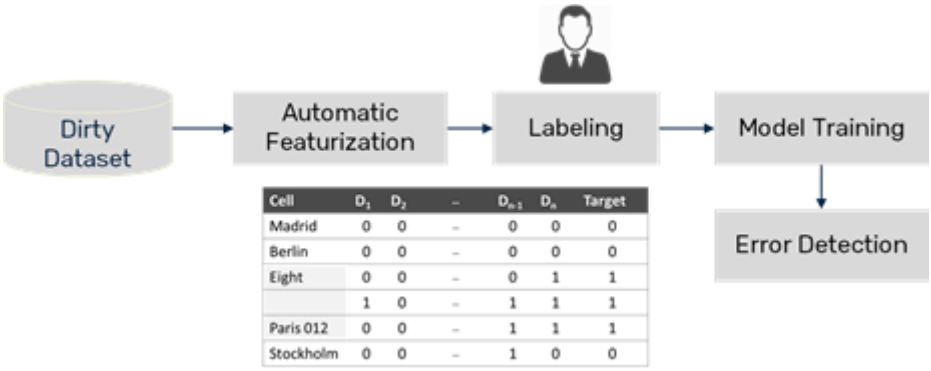

Figure 5. Error detection formulation as a binary classification problem

Despite being effective in detecting different error types which might occur at the same time in a dataset, the aforementioned ML-based error detection still suffers from several drawbacks. First, the task of feature extraction is computationally intensive where multiple values are to be computed for each cell in the dataset. The results of our experiments showed that such an error detection method is hardly applicable with relatively large datasets. Second, the error detection method cannot exploit design-time knowledge or historical artefacts. The main intuition was to develop a configuration-free error detection method. However, being completely independent of external resources prevent the method from exploiting existing knowledge which might boost the detection performance. Third, the ML-based error detection method does not recognize the error types, where the output is a binary decision for each cell of being contaminated or clean. Finally, its performance is highly dependent on the labelling budget. If the budget is limited or if no data experts exist, the ML-based error detection method performs poorly. To tackle such drawbacks, more research efforts are to be exerted and to make the ML-based error detection methods suitable in a wide range of AI applications.

3.2 Meta Learning-Based Feature Extraction

In this section, we introduce a novel ML-based error detection method which overcomes the first two drawbacks. In other words, the proposed method makes use of the concepts of meta learning to exploit the historical knowledge and to provide a knob for controlling the execution time of the error detection task. The core idea is to exploit the knowledge embedded in the historical datasets and use this knowledge while training the detection classifier. Before delving into the details of the proposed method, it is necessary to introduce the concepts behind meta learning. Generally, meta learning is used to learn new concepts and skills fast with few training examples. To this end, meta learning methods exploit pre-trained models that have been used for prior tasks to achieve higher model prediction accuracy, a faster and cheaper training process, and build more generalized models Figure 6 demonstrates the different meta learning processes. Initially, a set of base classifiers are usually trained exploiting training sets collected from prior tasks, where the sets share the same schema. For a new unseen task with few training examples, the base classifiers can generate a rich feature vector, referred to as the meta features. To generate predictions in the new task, the meta features are exploited for training the so-called meta-classifier. Accordingly, the knowledge embedded in the base classifiers can be transferred to the meta classifiers through the generated meta features.

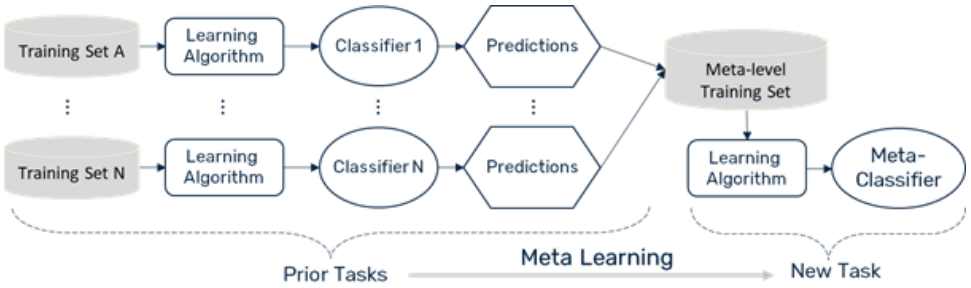

Figure 6. Basic idea behind meta learning

In fact, the utilization of meta learning to generate features for the detection classifiers is motivated by the fact that historical datasets typically share properties with new datasets within a certain application. This means that the knowledge of dirtiness profiles in historical datasets can be exploited to detect errors in similar new datasets. As an example, imagine a smart factory which includes several machines producing a certain product. The machines are embedded with a set of sensors to track the health of the machines and to predict the maintenance needs. The data extracted from these sensors can be contaminated due to sensors defect or noisy communication channels. In this case, the sensory data are to be cleaned by a team of data experts. The knowledge of such experts will be embedded in the datasets. Accordingly, such knowledge can be extracted and used for cleaning newly generated data within the factory. Moreover, if the owners of such a factory decided to establish a new branch which also comprises the same machines. Then, the data generated in the new branch will be mostly similar to the data in the original factory. In this scenario, the proposed meta learning approach can be easily applied to detect errors in the newly collected data. Figure 7 illustrates the architecture of the meta learning-based method for detecting errors using features generated using a set of historical datasets. A meta learning module takes as inputs the historical datasets and the dirty dataset. The module begins with training a set of base classifiers on the historical datasets, before using these classifiers to generate the meta features. After generating the meta features, a meta classifier is to be trained on a set of labelled features. To implement the meta-learning module, it is necessary to identify a subset of the historical datasets which highly correlate with the dirty dataset. This step is required to exclude irrelevant data from being used to generate meta features. To control the execution time of the proposed method, we can simply adjust the number of base classifiers used to generate the meta features.

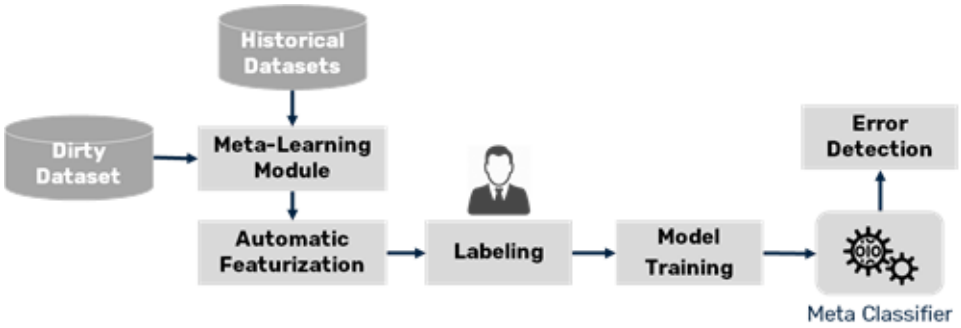

Figure 7. Architecture of meta learning-based error detection method

3.3 Conclusion and Ongoing Work

In WP2, we focus on the development of a unified framework for data processing and valorisation. One of the main problems, tackled by WP2, is to automatically clean the collected data before being used for training and serving. To this end, we introduced in this section a novel semi-supervised method for detecting errors in structured data. To consider the historical artefacts, the proposed method leans on the concepts of meta learning to transfer knowledge from the historical datasets to the dirty data. The proposed method has been evaluated with several real datasets and the results showed that the proposed method outperforms a set of baselines if the new data is similar to the historical datasets. We are currently working on the following extensions to further improve the performance in terms of the detection accuracy and runtime.

- Adopt a multioutput classifier to recognize the error type, e.g., outlier, rule violation, missing value, etc.
- Employ a data augmentation method to improve base classifiers in case of datasets with low error rates.
- Develop a data valorisation method to intelligently decide when and how to clean the collected data

The proposed methods will serve as the backbone of the unified framework for data processing and valorisation, as planned in WP2. Moreover, the proposed methods in this section will be closely aligned with the techniques and tools developed in WP3.

4 Data Quality Dashboard

This section describes the development work within data quality dashboards. In Section 2, we gave an overview to leading data quality dashboards in the market. The tools provide basic statistical information about the data quality, such as number of distinct, non-distinct and null values. In addition to statistics, tool users can create rules to find possible quality issues. As there already exists tools for basic statistical analysis, the development in this project concentrates on more advanced data profiling methods. The use case for development work is building energy consumption prediction. The following sections present typical data quality and profiling issues in this field, methods that can be used to resolve the issues and ways of visualizing the results in a dashboard.

4.1 Quality and Suitability of Energy Data

Typical data quality problems with energy meters are related to missing values and abnormal values. For missing values existing commercial tools provide solutions, but for abnormal values more advanced methods are needed. Abnormal values can be meter jams to certain consumption level or changes in a typical energy consumption level. To detect these problems, energy meter figures can be compared with historical values. In addition to assuring the quality of the data, there is a need to verify the suitability of data for energy prediction. Predictability is improved if there are no changes in the consumption pattern and the correlation between outside temperature, day of the week and consumption is similar during the analysed time period. Therefore, observation of changes in consumption patterns and correlations is vital.

4.2 Methods for Quality and Suitability Detection

Throughout the IML4E project, several different methods will be tested and evaluated to detect energy data quality issues and suitability for energy prediction. One of the tested methods will be ARIMA, that is used for predicting time series data. Figure 8 shows an illustration of ARIMA prediction. Knowing historical data, it is possible to predict the range (not exact values) in the near future – and thus detect if consumption falls into this range. This method is good, for example, for detecting outliers. The strength of the method is that it is able to catch seasonality, thus cyclic nature of consumption (hours of the day, day of the week, season of the year).

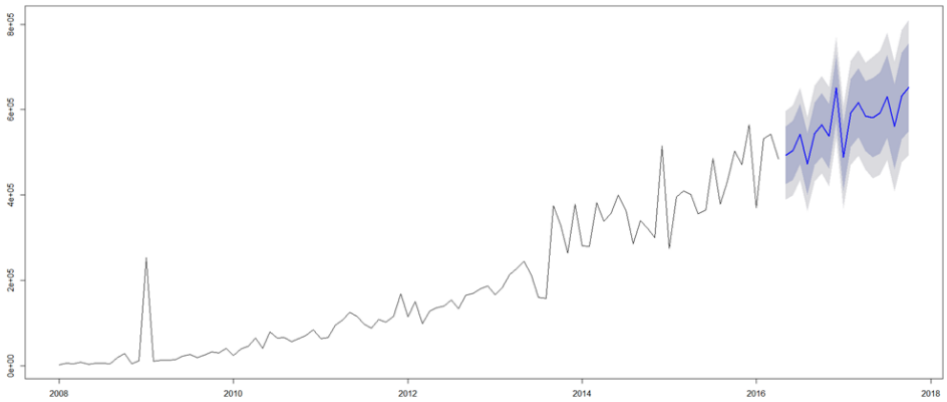

Figure 8. Forecasting using ARIMA (Pabba 2017)

Another method that will be tested is clustering (cf. Figure 9). We can create multidimensional vectors from data table - one row is one point and numbers of columns is number of dimensions. Then in the multidimensional space using for example principal component analysis (PCA), we can cluster those points into groups. If there is outlier, we can detect it and viewer of the dashboard can investigate it further. Unlike time series method, here we see each point (i.e., each row) independently. This approach may have pros and cons compared to ARIMA/time series methods, so we need to investigate which methods fits better for which task.

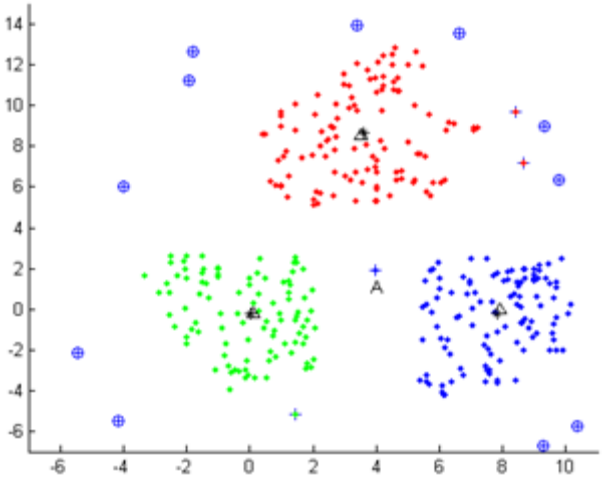

Figure 9. Clustering data (Kaur et al. 2011)

4.3 Dashboard for Energy Data Profiling

After finding suitable methods for quality and suitability detection, the results of the methods are developed into KPIs (e.g., percentage of abnormal values) and visualized in a dashboard. Several visualization techniques are examined to support the quality problem detection and root cause analysis. For example, more detailed information of results will be shown to enable deeper analysis of the reasons behind quality deviations. The dashboard is going to developed together with the end users of the tool so that it is interactive enough to support their work in the best possible way.

5 Privacy-Friendly Image Preparation for AI Pipelines

5.1 Problem Definition

In this section, we elaborate on a data preparation process dedicated to 2D images. In this context, we at the Budapest University of Technology and Economics (BUTE), jointly with use case partners from Vitarex Stúdió, develop a processing workflow for images which involves facial anonymization to offer a privacy-friendly AI model generation process. It is necessary to mention that the developed method is planned to be adopted to the pose estimation use case. In general, human 2D pose estimation is a challenging problem in computer vision, and it has a wide variety of applications. Pose estimation is usually employed in augmented reality, robotics, and health care. In pose estimation, a person's movements are tracked by finding the location of a set of selected keypoints. However, many of these keypoints can be found on the person's face, which is a very privacy-sensitive area. Unfortunately, these facial keypoints have great information value for further applications mentioned before. Therefore, they are needed to be recognized for an accurate pose estimation and are not negligible.

With the rising importance of data protection and individual privacy, face anonymization has drawn increased attention from the research community in recent years. Since the introduction of the General Data Protection Regulation (GDPR) by the European Union in early 2018, privacy protection became an indispensable task of research fields, institutions and companies using personal data. GDPR requires regular personal consent from individuals for using their privacy-sensitive data, thereby making it difficult to work with these types of resources. However, the regulation leaves space for non-consensual use of these images if the individual is unrecognizable.

5.2 Face Anonymization Techniques

Generally, face anonymization is a computer vision technique where facial information is removed from digital images and videos. In this case, a problem arises when these anonymized images are used as the input to further machine learning algorithms, such as the case of pose estimation. We lose crucial information from our data stream by applying anonymization methods as pre-processing. It is important to find the right method of anonymization, where the data stays private while preventing the machine learning algorithm from being corrupted. In this project, we examined four state-of-the-art anonymization techniques, namely: *blacked out*, *blurred*, *pixelated* and *DeepPrivacy* (Hukkelås, Mester and Lindseth, 2022). Figure 10 depicts an example to compare between the four different anonymization methods. As the figure demonstrates, the blacked out pictures are generated through filling the bounding box, containing the face, with random black and white pixels. For the blurred version, the face is blurred with a 3x3 kernel Gaussian Blur, while for pixelated images, the face is replaced with a 15x15 pixel array.

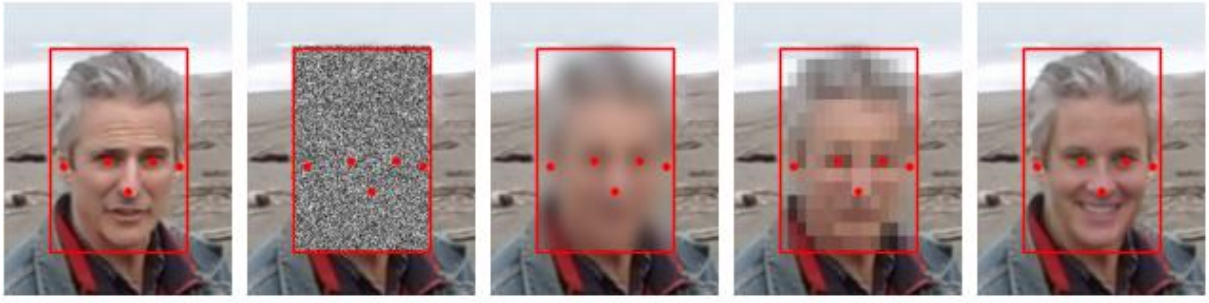

Figure 10. Examples of the different types of face anonymization techniques. From left to right: the original image from the dataset, blacked out, blurred, pixelated and DeepPrivacy modified.

DeepPrivacy provides an anonymization method by replacing individual faces on images while minimizing the change in the existing data distribution. The model removes all privacy-sensitive information and generates highly realistic faces with high accuracy. It uses Mask R-CNN (He, Gkioxari, Dollar and Girshick, 2017) for facial keypoints estimation and Single Shot Scale-Invariant Face Detector (Zhang et al., 2017) for bounding box annotation. For the face replacement, it uses a conditional GAN architecture (Mirza et al., 2014). It is worthwhile mentioning that DeepPrivacy requires only the images as an input to execute the anonymization. We explored

the effectiveness of these four face anonymization methods in our pose estimation use case. For this purpose, we anonymized the input images (remove facial information), before employing a state-of-the-art pose estimation algorithm and evaluated them on a publicly available dataset. The results showed that DeepPrivacy outperforms all other face anonymization methods (cf. Figure 11).

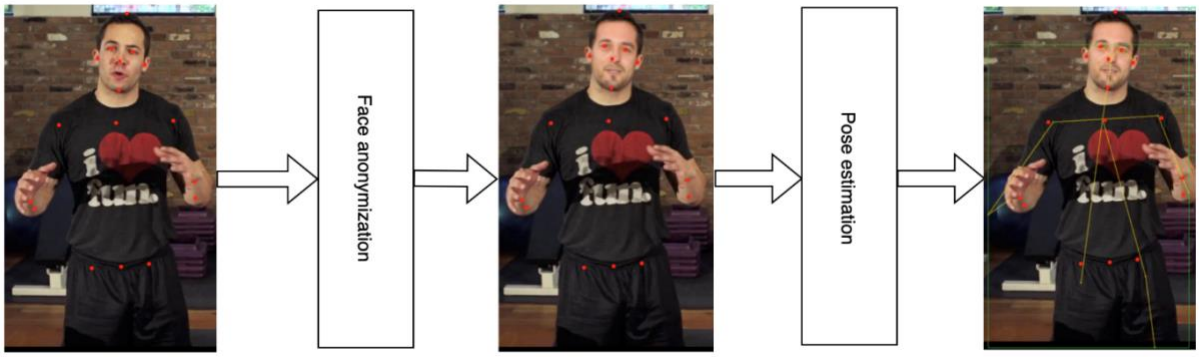

Figure 11. Workflow: Face anonymization and pose estimation stages

In a second set of experiments, we measured the success of the alteration methods. Specifically, we evaluated each unmodified image, its alterations, and the blacked-out version, on the Lightweight Open Pose network[17] (Osokin, 2022). Such a network exploits a part based (also called bottom-up) approach, that searches an image for keypoints, and then determines their pairwise relationships, thus creating the skeletons. Such a network is a modified, performance-tuned version of the original OpenPose paper (Cao et al., 2021). With a different, more lightweight backbone, and refinement stages, it is considerably faster that OpenPose, which is critical in edge devices like smartphones, where pose estimation applications are emerging. The accuracy of the optimized network is nearly identical to the original, dropping less than 1% in average precision.

5.3 Preliminary Results

To evaluate the workflow, we used the MPII Human Pose dataset (Andriluka, Pishchulin, Gehler and Schiele, 2014) that shows people in great appearance variability and complexity in over 40000 images, and in more than 800 activities. Moreover, it provides rich annotations for the main body and face keypoints. The annotations include the head, torso, and the left/right limb joints. We filtered the database for single person images in regular resting positions. We selected 92 images and their corresponding annotations from the data set. Because the pose estimation network detects facial keypoints too, and many of these keypoints were not provided with the annotations (left and right eye, nose, left and right ear), we manually annotated these. The final keypoints are drawn as red circles in Figure 12.

We used the same evaluation metrics that S. Wu et al. utilized in evaluating the performance of facial keypoint detection (Wu, Xu, Zhu and Guo, 2018). Specifically, we use the detection error (DE) defined inters of the actual and predicted values of the keypoints and the width of the face's bounding box. Equation 1 expresses the error DE where x and y are the predicted values of the keypoint,$\bar{x}$ and $\bar{y}$ are the annotated values and w is the width of the face's bounding box. Aside from the detection error, we also employ the Root Mean Square Error (RMSE), defined in Equation 2, where n is the number of keypoints recognized in the image. Moreover, there were some instances where the pose estimation failed to recognize a keypoint. To measure these events, we estimate the finding rate (FR), as expressed in Equation 3, where f is the number of the images where all keypoints were found, and N is the total number of images.

$$DE = \frac{\sqrt{\{(x - \bar{x})^2 + (y - \bar{y})^2\}}}{w} \quad (1)$$

[17] The implementation of such a network is publicly available (GitHub - Daniil-Osokin/lightweight-human-pose-estimation.pytorch: Fast and accurate human pose estimation in PyTorch. Contains implementation of "Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose" paper., 2022).

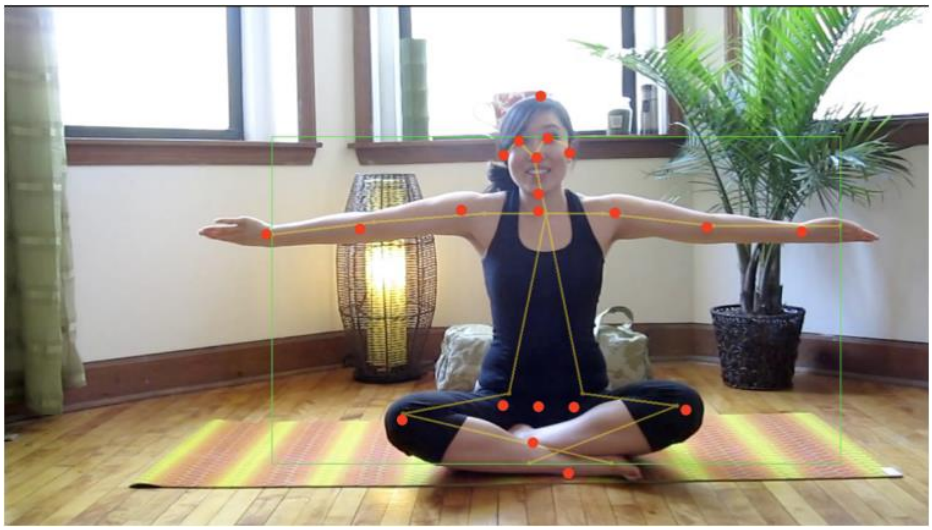

Figure 12. The pose estimation and keypoints. Annotated keypoints are marked red, estimated keypoints yellow, and bounding box for pose green.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x_i})^2 + (y_i - \bar{y_i})^2} \quad (2)$$

$$FR = \frac{f}{N} \quad (3)$$

Table 1 summarizes the obtained results of our experiment. Considering the finding rate (FR), the pose estimation worked equally well on the blurred and DeepPrivacy images. Looking through the images, it can be concluded that the pose estimation network fails to identify keypoints for annotations where some keypoints are covered, like when limbs overlap each other or the subject hides facial features because of head rotation. The evaluation metrics show that the DeepPrivacy face replacement method surpasses the performance of the traditional anonymization techniques (like blurring and pixelating). Looking at the DE and RMSE values, the performance of the DeepPrivacy images is consistently near the results of the original ones. It greatly improves over the results of the blurred method, which achieved 6.8% average DE and 8.370 RMSE, compared to 5.6% and 6.614 for the DeepPrivacy. Out of these methods, the blacked out estimation is found to be insufficient to successfully track human poses.

		Face		Full body
Type	**FR**	**DE**	**RMSE**	**RMSE**
Original	0.92	0.053	6.613	15.604
Blacked out	0.39	0.293	31.889	24.523
Blurred	0.88	0.068	8.370	17.332
Pixelated	0.85	0.075	9.272	16.157
DeepPrivacy	0.88	**0.056**	**6.614**	**14.952**

Table 1: Metric values for the pose estimation (Finding rate (FR) and average of Detection error (DE) and Root Mean Square Error (RMSE))

6 Mosquito Data Cleaner for Unstructured Data

In this section, we elaborate on a novel data cleaning method, referred to as Mosquito, which can be used to clean the data extracted from PDFs.

6.1 Problem Statement

Basware "SmartPDF AI" product extracts data from PDF documents (including scanned versions). It receives a PDF/image as an input and responds with field data as the output. Our models are supervised, using 3.3 million manually processed invoices as training set. The training data contains human errors - both instance-level, originating from single unbiased human mistakes, and population-level, originating from regular misinterpretation of the field's meaning. For some best fields, the percent of human errors in training data is 0.8%, for some worst fields the percent of errors can reach 30%. The training set requires automatic cleaning before the training has started.

6.2 General Solution

In general, it is not feasible to use boosting-like data-cleaning methods because our models are huge and expensive to train. Instead, we use a smaller model called "Mosquito" for data cleaning. Figure 13 demonstrates the code idea behind Mosquito where the anomalies are detected through comparing the ground truth with the predictions of Mosquito. In other words, our cleaning approach can be roughly characterised as "clustering-and-majority-voting-based anomaly detection". Mosquito model consists of three levels of algorithms. The first level is the representation of the invoice structure at some degree of heuristic understanding. For example, an invoice can be considered as bag-of-words, bag-of-blocks, or bag-of-key-value-pairs. This algorithm produces the invoice signature, which is a set of strings that can be quickly compared with another signature. The signature similarity is considered as a distance between invoices.

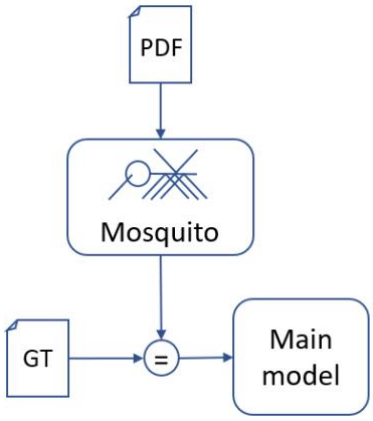

Figure 13. Cleaning GT for the main model training by comparing it with Mosquito predictions

The second level of algorithms uses a signature-based similarity distance for dataset clustering. For example, it finds 100K groups among 3.3M invoices. To this end, it uses different techniques like hierarchical clustering or pivot-based clustering. The invoices inside one group have static parts (like labels), called "anchors", and variable parts, called "fields". We use different proprietary algorithms to separate anchors from fields. The third level of algorithms uses ground truth (GT) to attach labels to the fields within each group. The problem here is that GT contains human errors, and the task becomes ambiguous. To resolve the ambiguity, we use majority voting. Specifically, each anchor candidate gets the score of correct extractions. The anchor with the largest score is considered a winner, and all GT samples that disagree with the winner choice are considered anomalies. Such anomalies are marked with lower sample weight for the main model training. This simple training approach removes immediately 30% of errors from the main model predictions, which significantly boosts the quality of the production model. One can also consider this kind of anomaly detection as applying Mosquito model to its own training set and comparing the result with the GT. If GT does not match the prediction, it can be annotated as an anomaly.

6.3 Work Plan within WP2

To increase the quality of data cleaning, Basware decided to raise the heuristic power of Mosquito algorithms within the scope of WP2. Additionally, Basware strives to apply the latest technologies in multi-modal image segmentation to introduce ML-based heuristics to the Mosquito algorithms. In the scope of WP2, we introduce three new generations of Mosquito algorithms in addition to Gen1, which has already been implemented before being involved in the activities of WP2. Figure 14 demonstrates the planned four generations of Mosquito with highlighting the main technological differences among these generations. The Gen1 heuristics (G1L1) used the "bag-of-words" invoice model. This method is the most trivial invoice understanding, which produces the primitive signature like "123Due", "456Total", where the number is x-coordinate of the word. This signature is not very descriptive, and it is vulnerable to x-shaking produced by OCR.

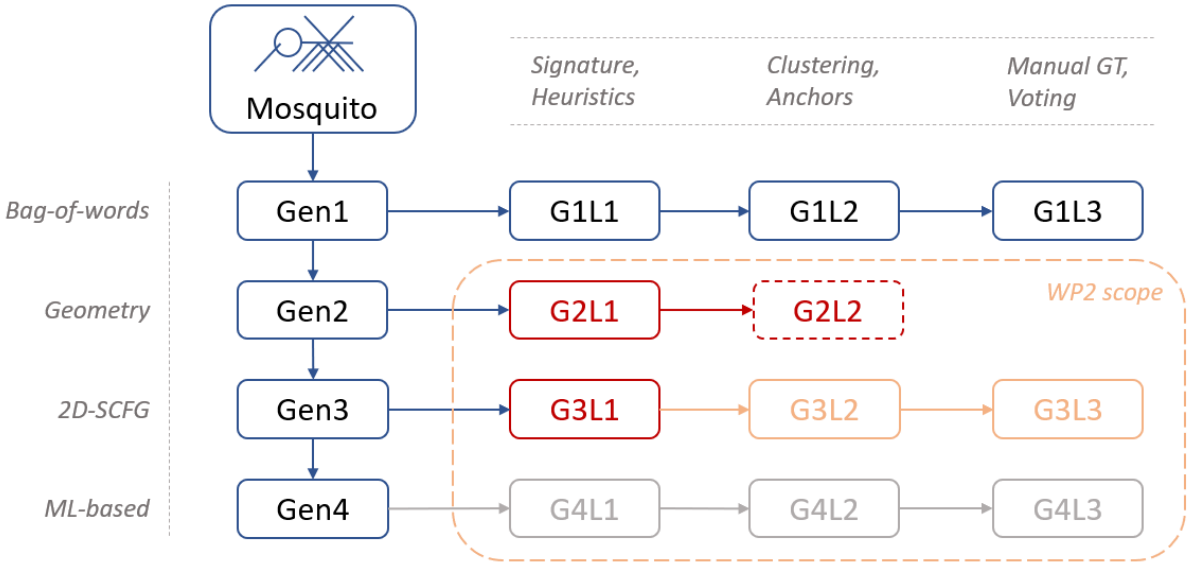

Figure 14. Mosquito algorithms family

As a result of low descriptive power, the clustering and anchor detection with this type of signature requires sophisticated hierarchical clustering techniques in G1L2. In fact, we failed to detect anchors in unsupervised manner, so anchor detection was moved to G1L3, which uses GT to spot the anchor-field pairs. The deficiencies of Gen1 approach inspired us to develop the more sophisticated heuristic representation of the invoice. The Gen2 family of algorithms uses geometry-based heuristics to represent the invoice as "bag-of-blocks". Figure 15 shows an example where the "bag of blocks" model has been used to understand the content of an invoice. This type of invoice representation enables much more descriptive signature like "14(0<3)2=Fakt", "45(1<2)2=Kund", where digits are x-position, size, and numbers of neighbours of blocks. This kind of highly descriptive signature makes such a tightly packed invoice clusters that separating them from each other can be done with the most primitive pivot-based technique in G2L2.

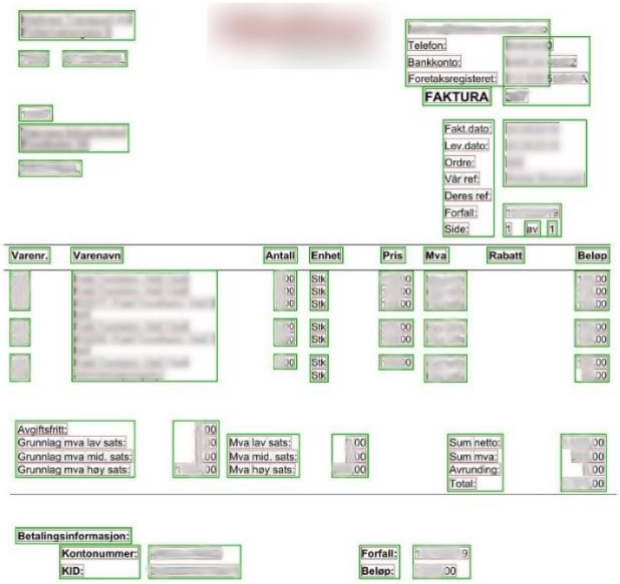

Figure 15. Mosquito G2L1 "bag-of-blocks" view of invoice

However, spotting the anchors with this technique becomes very difficult, because of the coarse-grain representation of the invoice, which is "blind" to individual fields. Therefore, we interrupted the Gen2 development and switched to even more sophisticated Gen3, which finds key-value pairs and anchors by pure algorithmic CFG-based heuristics. The G3L1 algorithm implements a grammar-based heuristic invoice understanding, which splits the invoice to key-value pairs (tabular data is considered separately). This invoice representation already contains ready-made anchors, so anchor detection becomes a trivial task. Figure 16 demonstrates an example where the "key-value-pairs" model is used to understand the contents of an invoice. The G3L1 algorithm uses 2D-SCFG probabilistic grammars for parsing the two-dimensional structure of the invoice. The parsing is done with simplified version of CYK algorithm, which produces the incomplete parse.

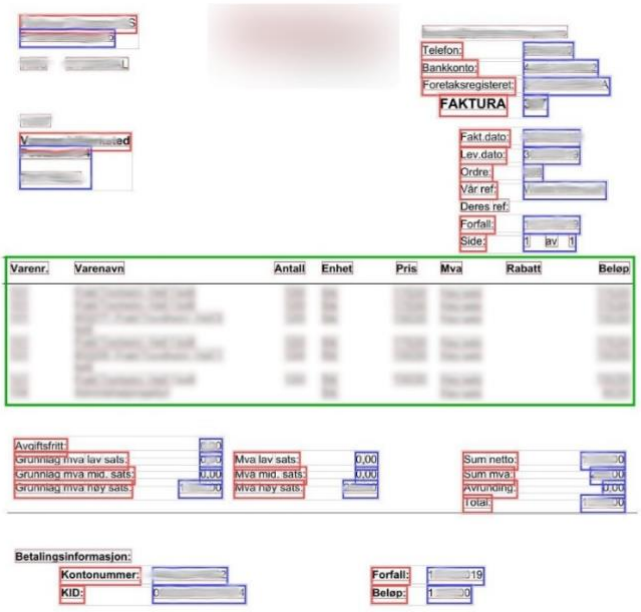

Figure 16. Mosquito G3L1 "key-value-pairs" view of invoice

It is unfortunately not feasible to define the grammar for the complete parse of the invoice due to enormous diversity of the invoice types. However, the constituents of the invoice are surprisingly simple and typical, which makes the incomplete grammar quite trivial. Figure 17 demonstrates an example of such a grammar. The G3L1 algorithm builds the semantic representation of the invoice, which makes it is possible to produce fully translation-invariant and highly descriptive set of signatures, e.g., `"faktdato[over]levdato"`, `"antall[left]enhet"`.

```
'key_val_hor': [{
   'terms': ['label', 'value'],
   'score': 1.2,
   'direction': 'right',
   'alignments': {'top': 1}
}],
'key_val_vert_list': [{
      'terms': ['key_val_hor', 'key_val_hor'],
      'score': 1.1,
      'direction': 'bottom',
      'alignments': {'left': 1, 'center': 1, 'right': 1}
   }, {
      'terms': ['key_val_vert_list', 'key_val_hor'],
      'score': 1.1,
      'direction': 'bottom',
      'alignments': {'left': 1, 'center': 1, 'right': 1},
      'keep_alignment': True,
      'keep_spacing': True
}]
```

Figure 17. Example 2D-SCFG grammar of the invoice, Chomsky normal form, fragment

This type of invoice description is completely free from any digital metrics and fully invariant to OCR damages and minor algorithm modifications, which makes a solid ground for G3L2 clustering. As mentioned above, G3L2 doesn't need to find anchors in most of the cases, because the keys are already found by G3L1. However, there are orphan fields that don't have any label. For those fields, the anchors are still to be found. Finally, the G3L3 algorithm will use ground truth information to attach the labels to the fields. Since the fields are already identified by the anchors, the assignment can be done with single data sample, which enables Mosquito single-click learning feature needed also for self-validation (outside of the scope of WP2). In case more than one sample is provided, and there is a conflict, the resolution is done with majority voting like in G1L3. The G3L1 algorithm relies heavily on the anchor-string dictionary. This dictionary is populated automatically by G3L2 and verified manually. The G3L3 adds labels to the dictionary, which makes it possible to identify fields in completely unknown invoices. It is worthwhile mentioning that the G2L1, G2L2, and G3L1 have been implemented, while the generations G3L2 and G3L3 are not yet started.

We have also started the research for ML-based generation G4L1, which is supposed to produce key-value segmentation of the invoice using multi-modal transformer-based model borrowed from Microsoft LayoutLMV2 published open-source by Microsoft[18]. The model takes both 2D raster image of the invoice and the 1D text-based representation of the invoice as inputs. The model produces the segmentation of the invoice like the one shown in Figure 18. This segmentation can later be used for clustering and anchor detection. The final value extraction can also be done by a separate ML-based model G4L3. The G4L1 model is supervised, and it needs the training data. The training data can be automatically generated by G3L1. However, there is a risk that G4L1 will learn from heuristic errors of G3L1. To avoid this, only the most probable branches of the stochastic-grammar-based parse-tree must be taken. If the parser is uncertain, it is better not to use the sample for training. The benefit of ML-based approach is that it can generalize better to the cases that CFG parser fails to parse due to grammar limitations. G4L3 is seen as the final goal of the WP2 project.

[18] https://arxiv.org/pdf/2012.14740.pdf

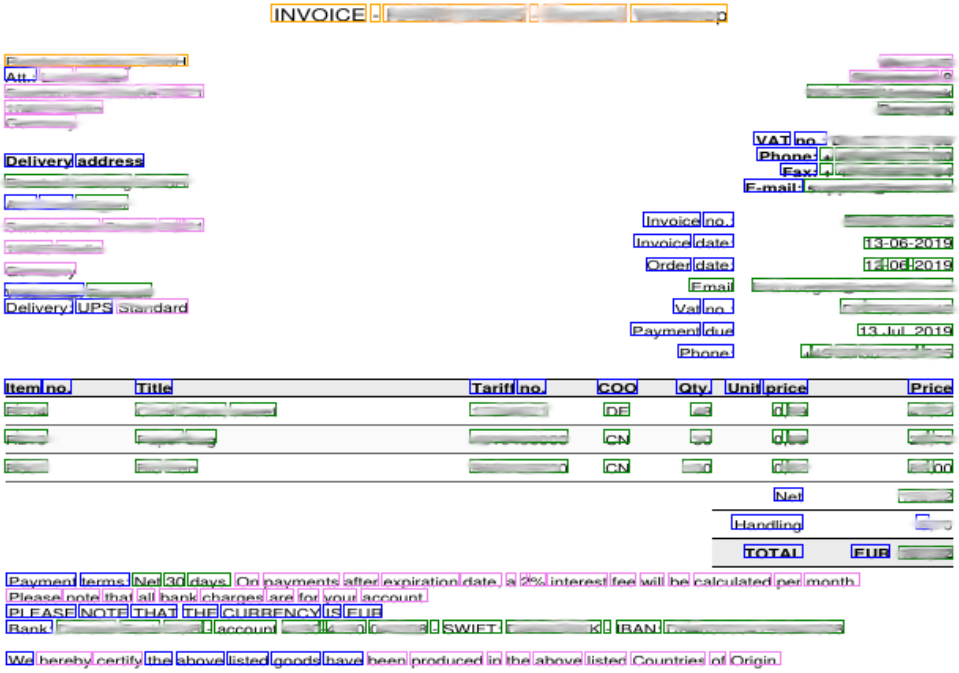

Figure18. Mosquito G4L1 “label-value-other” view of invoice, using LayoutLMV2

7 Catalogue of Quality Attributes

For assessing the quality of data in the MLOps process, its necessary to perform measurements on the dataset. In this section, we describe how a data quality catalogue can be used to quantify the quality of the data in an automated manner. Furthermore, we provide a methodology of mapping MLOps artifacts to the measurement inputs. In general, data is the core part of every ML-System, hence quality assurance in this domain has the main focus. All the attributes are taken from ISO 25012 (cf. Table 2). Quality checks and data manipulation usually take place in the first steps of each cycle. Measurements are performed on static artifacts produced in those early steps. Thus, the frequency for data-related quality attribute assessment is usually once per cycle. The ISO 25024 (ISO/IEC, 2015) standard introduces generic measurements for the listed quality attributes.

Quality Attribute	Description
Accuracy	"Data accuracy is the degree to which data has attributes that represent the actual value of a concept." (ISO 25012)
Completeness	"The degree to which subject data associated with an entity has values for all expected attributes." (ISO 25012)
Consistency	"The degree to which data has attributes that are free from contradiction and are coherent with other data in a specific context of use." (ISO 25012)
Timeliness	"The degree to which data has attributes that are of the right age in a specific context of use." (ISO 25012)

Table 2: Initial set of quality attributes from the Data domain.

7.1 Measurements

The ISO 25024 standard provides a set of measurements for Accuracy, Completeness, Consistency and Timeliness. The measurement is being performed according to the formula. The difficulty lies in retrieving the input variables. As an example, let us look at the measurements for accuracy from the ISO 25024 in Table 3. The A and B value are per se none existed in the first place, which means they need to be obtained. This can happen in serval ways, like querying a database or loading the data frame an analysing it. In general, it needs some sort of information retrieval to obtain these values.

Quality Measurement	Formula
Syntactic data accuracy	X=A/B A is the number of syntactyly correct items B is the number of items where syntactic correctness is required
Semantic data accuracy	X=A/B A is the number of semantically correct items B is the number of items where semantic correctness is required
Data accuracy assurance	X=A/B A is the number of data items measured for accuracy B is the number of data that needs to be measured for accuracy
Risk of data set inaccuracy	X=A/B A is the number of elements of the data model that accurately describe the system. B is the number of elements of the data model that describe the required accuracy within the required specification of the system.
Metadata accuracy	X=A/B A is the number of metadata that provides appropriate required information B is the number of metadata defined within the required specification of data.
Data accuracy range	X=A/B A is the number of data items having value included in a specified interval. B is the number of data items for which a range can be defined.

Table 3: Accuracy measurements taken from ISO 25024

7.2 Mapping of Raw Data to Measurement Inputs

Information required for the measurement needs to be obtained from artifacts like logs, metadata, code or raw data. During the various steps of the MLOps process, different kinds of artifacts are generated. Those artifacts contain either directly valuable information for quality assessment or they can be used as tools to generate this information. This information can then be mapped to certain measurement inputs, see Figure 19. It shows on top the MLOps process where each iteration produces raw data. This raw data is either input to active testing or directly to the extractor. The extracted information gets mapped to the inputs.

Usually, the MLOps process in industry is supported by frameworks like MLFlow or Kubeflow. Both allow extending their functionality like managing artifacts. We are utilizing this to extract raw data like logs, metadata, or trained models. In the next step, this raw data gets processed depending on the kind it can be simply parsed to extract information, or it is being used as input for other testing tools. The way a particular artifact is handled depends on by which tool it was generated. For instance, TensorFlow logs have a specific format, so it requires a dedicated parser for them. The Extractor needs to be implemented for each kind of artifact and some artifacts require the implementation or configuration of a specific active testing tool. In the end, the extracted information gets mapped to a specific API call parameter and therefore to a specific measurement value.

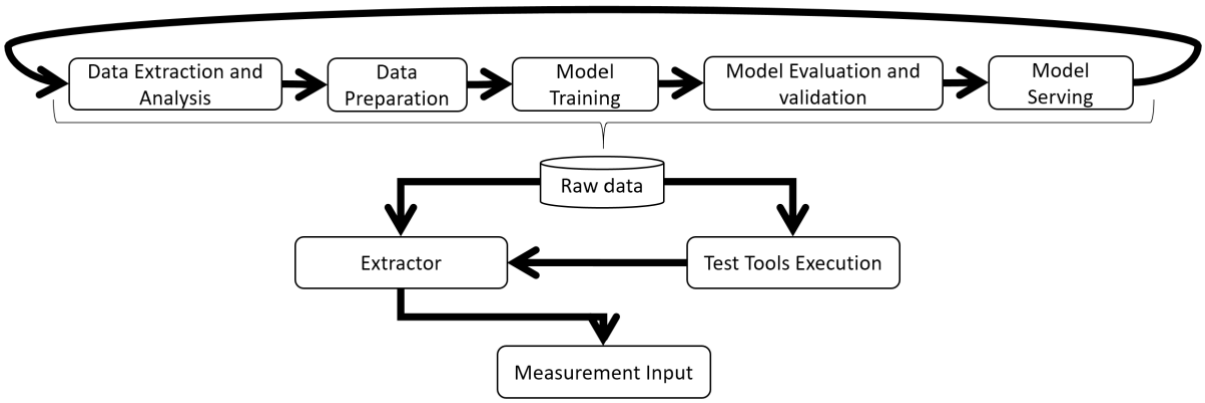

Figure 19: Mapping of the raw data to the input of the measurements

8 Data Version Control: Criteria and Use Cases

In this section, we outline our contributions in data version control. Our main goal is to help teams and organizations choose the best tools and architectures for their use cases. We begin with listing the important characteristics of data management use cases that teams should consider when comparing tools and designing architectures. We then list three example use-cases that we will consider, in this project, and finish the section with an outlook for reference implementations that we plan to build later in the project.

8.1 Data Versioning Use Case Characteristics

The choice of a suitable data versioning approach for a given project broadly depends on the use case. In this section, we list some important questions that should be thoroughly answered when comparing tools or designing architecture of data versioning solutions.

- **What is the total size of the dataset?** Data versioning approach for a dataset of size 1 MB is likely different than for 1 TB. Does the dataset contain *structured, semi-structured or unstructured data*? For example, some tools are better suited for tabular data than image data. How many files or rows are there in the dataset? If there are millions of files, it may not be feasible to keep references to them in a Git repository. How large is each file and what is their format?
- **How is the dataset updated?** How much data is added to the dataset and how often and what is the process for creating a new version of the dataset? If a new version is created every day, duplicating data between versions is less desirable than when creating a new version once a month. Are the updates append-only? For example, if new images are added to the dataset regularly, the "diffs" between versions are simple additions. Are samples in the dataset modified or deleted? If data is stored in huge CSV files and individual rows are modified between versions, this may have effect on storage requirements. What kind of data quality checks are needed when the dataset is updated?
- **How is the dataset used outside of the current project?** Is the dataset shared to other teams in the organization and used in multiple projects simultaneously? Updating the dataset for one project's needs might have unexpected consequences on the other projects. What are the requirements for access control and governance? Maybe only specific users are allowed to view datasets that contain personally identifiable information. How long should each version be persisted? How are users' requests to delete their own data handled?
- **How do employees and internal systems access the data?** For tabular data, data scientists and analysts may want to query the dataset with SQL commands. For very large datasets, it might be convenient if the dataset can be accessed through an S3-compatible API from tools such as Apache Spark. Is it feasible for employees to download the datasets to their local desktops to browse and visualize the data? Does the dataset need to stay in-place?
- **How are features derived from the dataset?** For example, if predicting transaction fraudulence depends on transactions created in the last 72 hours, the system needs to be capable of computing and accessing such online features. Is it enough to version snapshots of the raw data or should features be versioned too? Are the features shared between multiple projects or teams?
- Finally, it is important to consider **what tools are already used in the team or organization and if introducing new tools is feasible.** Is the organization committed to using Kubernetes? If not, self-hosting Kubernetes-based tools like Pachyderm is not feasible. Is Apache Spark used for data analytics or machine learning? If yes, it is much easier to get started with Delta Lake. Do your existing machine learning pipelines already read data from S3-compatible system like S3 or MinIO?

8.2 List of Selected Use Cases

8.2.1 Human Pose Estimation from Images and Versioning Solution with DVC

During the image preparation project, we used DVC, mostly for its data management and versioning capabilities. As we were working with images, a machine learning model, training data and annotations for images, it was an excellent showcase of DVC's ability to handle these tasks. To get started with DVC, we just followed the tutorial on the official website[19]. We chose to use the command line interface, but a Python API is available if one prefers that. It was very easy to integrate it into our existing Git repository, and its Git-esque commands were easy to adopt. Once initialized, we added the files to version and track, using the `dvc add` command. Using this command, DVC automatically puts the selected files into the project's .gitignore file, and started tracking only the small metadata file that it created based on the content of the original. The next step was to choose a location where DVC would keep the data. For our use case, a network attached storage (NAS) was the best fit. Setting it with `dvc remote add`, and then pushing using `dvc push`, it copied the data cached locally to the NAS remote storage we set up earlier. The remote storage directory at this point looked like on Figure 20. If another colleague would then join us, they could get the files using the command `dvc pull`, as seen on Figure 21.

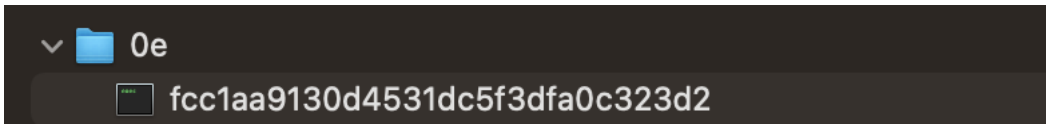

Figure 20. File structure after adding a file to dvc and pushing it

```
(base) balazsmorvay@Balazs-MBP lightweight-human-pose-estimation.pytorch másolat 2 % dvc pull
A       image_annots.json
A       big data/prepared_train_annotation.pkl
2 files added and 2 files fetched
```

Figure 21. Pulling the versioned data with DVC

In fact, our pose estimation use-case was a relatively simple one, but sometimes that represents what a certain project may need. The data we versioned was small-scale, only reaching a few gigabytes in size, however DVC can handle data at much bigger scales as well. In our case, versioning images and image annotations was crucial in a sense that we measured some metrics based on these. Changes to the mentioned data resulted in changes to the computed metrics, and we wanted to reproduce these later. Accordingly, DVC proved to be a very useful tool for our data versioning needs. It was simple to integrate in our existing repository, easy to use, and provided plenty of functionality. For our use case, Git LFS would have also been an adequate choice, but opting for DVC instead guaranteed that our solution would work in larger scales as well and having the option to use DVC pipelines and experiments can be a real advantage in the future.

8.2.2 Large-Scale Information Extraction from PDFs

The goal is to train a model for extracting information from PDFs. The dataset consists of PDF files and associated annotations, making it unstructured data. The total size of the dataset is around 1 TB. Every month, 1 million PDFs are added to the dataset, growing the dataset. Each file is around 100 kB to 1 MB in size. Annotations are stored in JSON files, and they consist of bounding boxes with associated metadata. Such annotations are modified when errors are discovered. Because annotation work is very error-prone, data quality checks must be run for new dataset versions. Specifically, PDFs are fed to the model as raw data. Any feature extraction happens as a pre-processing step, so features are not reused. The dataset is used for one project only. No special access control mechanisms are needed beyond standard data security measures.

The data, used in this use case, does not contain any personally identifiable information. Moreover, data is stored in an S3-compatible object store. Because of the size of the dataset, it is not feasible for users to download the dataset to their machines. Users want to be able to browse data in the dataset and run queries to find PDFs with specific kind of data. The team has experience of using Apache Spark for data analytics but not for machine

[19] https://dvc.org/doc/start

learning. In future deliverables of the project, we plan to describe possible solution architectures fitting this use case and implementing a reference solution.

8.2.3 Weather Prediction

In the weather prediction use case, the goal is to predict the weather for the next 24 hours using sensors and radar data. The dataset consists of high-volume tabular data and radar image files, making it a mix of structured and unstructured data. The datasets used for training are multiple TBs in size. New data is ingested to the system daily. New models only need to be trained on weekly or monthly basis when new dataset snapshots are also created. Updates to the data are append-only, but the data quality will vary a lot due to untrustful sensor readings. Since the sensor and radar readings do not include any sensitive data, no special access control mechanisms are needed beyond standard data security measures. Multiple development teams want to be able to access the dataset simultaneously. Data scientists want to be able to query data with SQL for visualization and development. In future deliverables of the project, we plan to describe the use case in more detail, discover solution architectures suitable for the use case and implement a reference solution.

9 Summary

In this deliverable, we have reported on our first results of the current WP2 tasks, namely data preparation automation (Task 2.1), data management and version control solutions (Task 2.2), and continuous Data quality assurance (Task 2.3). We started with briefly describing a set of industrial tools and platforms which are relevant to our planned contributions. We provided an overview of each tool and described the need to enhance such tools through the research and development activities in IML4E WP2. Afterward, we present our endeavours to develop a unified data preparation infrastructure for a trustworthy ML pipeline. In this context, we developed solutions for detecting errors in structured data, employing formal grammars and machine learning models to curate unstructured data, and propose a privacy-friendly preprocessing tool for digital images. Moreover, we defined a catalogue of quality attributes and methods for measuring such attributes to enable continuous quality assurance throughout the ML lifecycle.

For structured data, it is highly important to identify the contaminated records which can be extremely harmful to the downstream ML models. Therefore, we reported in this deliverable on our novel meta learning-based error detection method which uses a trained classifier to differentiate between clean and dirty records. Such a tool enables users to exploit the historical data sets which have been cleaned in the past. Moreover, it provides them with a knob to adjust the time needed to generate the list of detections. The next step in this work is to involve data valorisation into the data cleaning process. If we are able to identify the weight of each record, we can make important decisions, such as (I) whether to clean the records with lowest weights, (II) what are the best methods to cure the highly valuable records, and (III) whether we can get rid of the less-valuable records before the training phase.

After defining a set of requirements, it becomes clear how to develop a data quality dashboard suitable for the industrial applications. Therefore, we plan to start working on the data quality dashboard which can involves advanced techniques and tools for data profiling and cleaning. Similarly, we plan to start working on our data version control solution which can be adapted according to the dynamics the application scenario. Moreover, we are also eager to enable our solution from dealing with different data models, including images, relational data, audio, text, and log files.

References

Hukkelås, H., Mester, R. and Lindseth, F., 2022. *DeepPrivacy: A Generative Adversarial Network for Face Anonymization*.

Osokin, D., 2022. *Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose*. [online] arXiv.org. Available at: <https://arxiv.org/abs/1811.12004> [Accessed 4 May 2022].

He, K., Gkioxari, G., Dollar, P. and Girshick, R., 2017. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)*,.

Zhang, S., Zhu, X., Lei, Z., Shi, H., Wang, X. and Li, S., 2017. S^3FD: Single Shot Scale-Invariant Face Detector. *2017 IEEE International Conference on Computer Vision (ICCV)*,.

Andriluka, M., Pishchulin, L., Gehler, P. and Schiele, B., 2014. 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. *2014 IEEE Conference on Computer Vision and Pattern Recognition*,.

Mirza, M. & Osindero, S. (2014), 'Conditional Generative Adversarial Nets' , cite arxiv:1411.1784 .

GitHub. 2022. *GitHub - Daniil-Osokin/lightweight-human-pose-estimation.pytorch: Fast and accurate human pose estimation in PyTorch. Contains implementation of "Real-time 2D Multi-Person Pose Estimation on CPU: Lightweight OpenPose" paper.*. [online] Available at: <https://github.com/Daniil-Osokin/lightweight-human-pose-estimation.pytorch> [Accessed 4 May 2022].

Cao, Z., Hidalgo, G., Simon, T., Wei, S. and Sheikh, Y., 2021. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(1), pp.172-186.

Wu, S., Xu, J., Zhu, S. and Guo, H., 2018. A Deep Residual convolutional neural network for facial keypoint detection with missing labels. *Signal Processing*, 144, pp.384-391.

Informatica. 2021. Informatica® Cloud Data Profiling, July 2021. Cited 10.5.2022. Available: https://docs.informatica.com/content/dam/source/GUID-2/GUID-2755ADC0-A395-4CE7-91BC-30C14E970F08/11/en/CDP_July2021_DataProfiling_en.pdf

Lintelman, L. 2020. Shared Rules – A Hybrid Use Case with SAP Data Intelligence and SAP Information Steward. September 15, 2020. Cited 10.5.2022. Available: https://blogs.sap.com/2020/09/15/shared-rules-a-hybrid-use-case-with-sap-data-intelligence-and-sap-information-steward/

Pinjwani, A. (2021). Data profiling overview – SAP Information Steward. April 19, 2021. Cited 10.5.2022. Available: https://blogs.sap.com/2021/04/19/data-profiling-overview-sap-information-steward/

Talend. Talend Cloud Data Catalog User Guide, Sampling and profiling data. Cited 10.5.2022. Available: https://help.talend.com/r/en-US/Cloud/data-catalog-user-guide/sampling-and-profiling-data

Talend Team. A revolution in data quality: introducing Talend Data Quality Service. Cited 10.5.2022. Available: https://www.talend.com/blog/introducing-talend-data-quality-service/

Pabba, S. 2017. Time Series-based Forecasting using ARIMA Models. Available at https://www.hcltech.com/blogs/time-series-based-forecasting-using-arima-models

Kaur, P., Lamba, L. M. S., Gosain, A. 2011. DOFCM: A Robust Clustering Technique Based upon Density. International Journal of Engineering and Technology, 3(3):297-303