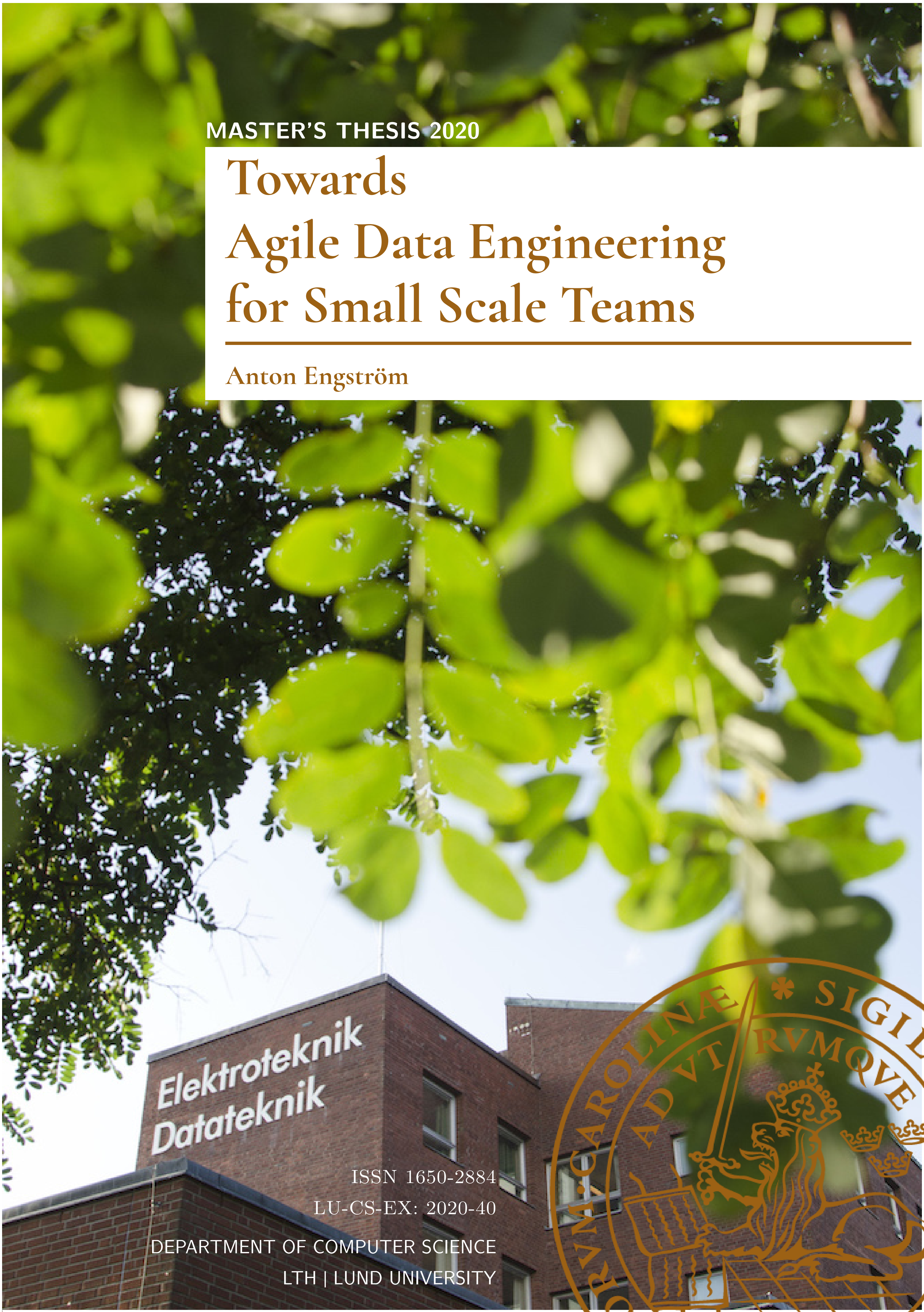
MASTER'S THESIS 2020

Towards Agile Data Engineering for Small Scale Teams

Anton Engström



ISSN 1650-2884
LU-CS-EX: 2020-40

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-40

Towards Agile Data Engineering for Small Scale Teams

Anton Engström

Towards Agile Data Engineering for Small Scale Teams

Anton Engström
anton.engström95@gmail.com

July 8, 2020

Master's thesis work carried out at Sentian Technologies AB.

Supervisors: Johan Ullén, johan.ullen@sentian.ai
Emma Söderberg, emma.soderberg@cs.lth.se

Examiner: Per Runeson, per.runeson@cs.lth.se

Abstract

Enabling production-level machine learning is difficult. In producing machine learning models for industry, a majority of the time is spent on working on the data and the infrastructure that will support the model in its production environment. In this thesis, we investigate the challenges related to data engineering for machine learning purposes. The intention is not to tackle the challenges related to big data. Instead, it focuses on a small scale context with few data sources and small datasets.

This context is studied through three iterations: a literature study, a case study of a small scale machine learning company, and a mapping of agile principles from the software engineering domain to the data engineering domain. We note that there is a gap between literature and the needs in a small context. We note that the potential of a knowledge transfer from agile principles to the data engineering domain is promising.

Keywords: data engineering, data platforms, data pipelines, machine learning, small scale, data, value creation, agile

Acknowledgements

I would like to express a very great appreciation to my supervisors, Johan Ullén and Emma Söderberg. The guidance and inexhaustible support from Johan can only be matched by the prompt and astute feedback provided by Emma.

I would also like to extend my appreciation to Per Runeson for providing advice and literature for this thesis. Thanks go to the data scientists at Sentian for participating in interviews.

Lastly, I would like to thank my family, my brother Oscar Engström especially, for the essential energy and encouragement they have supplied me with throughout my studies.

Contents

Chapter 1

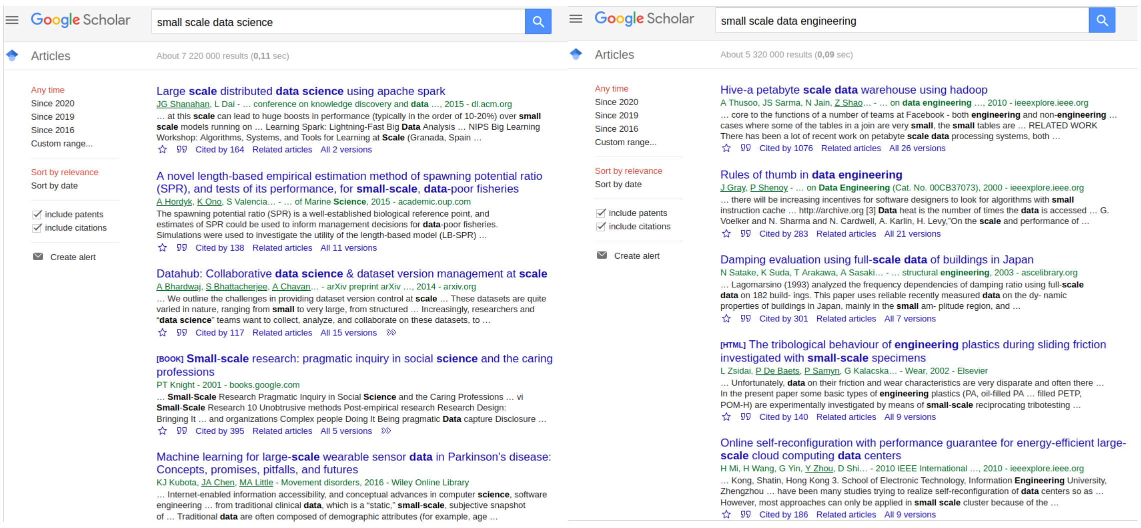
Introduction

Clive Humby, a UK mathematician, coined the term “data is the new oil” as far back as 2006. In 2017 the Economist released a report [1] stating through its title that “The world’s most valuable resource is no longer oil, but data.”. Setting the debate aside as to whether it is fair to make the comparison as they have very different properties, data not always having the potential to be useful in an economic sense or that data is not a finite resource to name a few of the criticisms [2], one can certainly say that data is an increasingly utilized and valuable resource.

While machine learning has been used for a long time by data scientists to glean knowledge from data [3], it is only recently that it has started being productionized to a larger extent. Toolsets for building machine learning models have matured in a pace magnitudes greater than the corresponding toolset related to data management [4]. However, both toolsets are needed to effectively run production-level machine learning operations.

Much like the data management toolset is far behind the machine learning toolset, the available literature on data engineering is far behind that of machine learning. In regards to small scale efforts, especially on data engineering, the lack of literature is even more exacerbated. This is unfortunate as industries move toward industry 4.0 and are implementing AI and machine learning efforts, since many of these efforts would, at least initially, be dependent on small scale data engineering.

As an example of the lack of literature, take the Google scholar search for “small scale data science” and “small scale data engineering” in Figure 1.1. The first result of each search is specifically targeted to the opposite needs of someone working in a small scale context. “Rules of thumb in data engineering” does look promising at a first glance, but as you look closer you will find that it is a very dated article on the subject of databases published in the year 2000 [5]. Of course, these two searches are not proof that there is no academic literature aimed at a small scale context, but rather that if there is, it is not easily found.



About 7 220 000 results (**0,11** sec)

Large **scale** distributed **data science** using apache spark
JG Shanahan, L Dai - … conference on knowledge discovery and **data** …, 2015 - dl.acm.org
… at this **scale** can lead to huge boosts in performance (typically in the order of 10-20%) over **small scale** models running on … Learning Spark: Lightning-Fast Big **Data** Analysis … NIPS Big Learning Workshop: Algorithms, Systems, and Tools for Learning at **Scale** (Granada, Spain …
Cited by 164 Related articles All 2 versions

A novel length-based empirical estimation method of spawning potential ratio (SPR), and tests of its performance, for **small-scale**, **data**-poor fisheries
A Hordyk, K Ono, S Valencia… - … of Marine **Science**, 2015 - academic.oup.com
The spawning potential ratio (SPR) is a well-established biological reference point, and estimates of SPR could be used to inform management decisions for **data**-poor fisheries. Simulations were used to investigate the utility of the length-based model (LB-SPR) …
Cited by 138 Related articles All 11 versions

Datahub: Collaborative **data science** & dataset version management at **scale**
A Bhardwaj, S Bhattacherjee, A Chavan… - arXiv preprint arXiv …, 2014 - arxiv.org
… We outline the challenges in providing dataset version control at **scale** … These datasets are quite varied in nature, ranging from **small** to very large, from structured … Increasingly, researchers and "**data science**" teams want to collect, analyze, and collaborate on these datasets, to …
Cited by 117 Related articles All 15 versions

[BOOK] **Small-scale** research: pragmatic inquiry in social **science** and the caring professions
PT Knight - 2001 - books.google.com
… **Small-Scale** Research Pragmatic Inquiry in Social **Science** and the Caring Professions … vi **Small-Scale** Research 10 Unobtrusive methods Post-empirical research Research Design: Bringing It … and organizations Complex people Doing It Being pragmatic **Data** capture Disclosure …
Cited by 395 Related articles All 5 versions

Machine learning for large-**scale** wearable sensor **data** in Parkinson's disease: Concepts, promises, pitfalls, and futures
KJ Kubota, JA Chen, MA Little - Movement disorders, 2016 - Wiley Online Library
… Internet-enabled information accessibility, and conceptual advances in computer **science**, software engineering … from traditional clinical **data**, which is a "static," **small-scale**, subjective snapshot of … Traditional **data** are often composed of demographic attributes (for example, age …

About 5 320 000 results (**0,09** sec)

Hive-a petabyte **scale data** warehouse using hadoop
A Thusoo, JS Sarma, N Jain, Z Shao… - … on **data engineering** …, 2010 - ieeexplore.ieee.org
… core to the functions of a number of teams at Facebook - both **engineering** and non-**engineering** … cases where some of the tables in a join are very **small**, the **small** tables are … RELATED WORK There has been a lot of recent work on petabyte **scale data** processing systems, both …
Cited by 1076 Related articles All 26 versions

Rules of thumb in **data engineering**
J Gray, P Shenoy - … on **Data Engineering** (Cat. No. 00CB37073), 2000 - ieeexplore.ieee.org
… there will be increasing incentives for software designers to look for algorithms with **small** instruction cache … http://archive.org [3] **Data** heat is the number of times the **data** is accessed … G. Voelker and N. Sharma and N. Cardwell, A. Karlin, H. Levy,"On the **scale** and performance of …
Cited by 283 Related articles All 21 versions

Damping evaluation using full-**scale data** of buildings in Japan
N Satake, K Suda, T Arakawa, A Sasaki… - … structural **engineering**, 2003 - ascelibrary.org
… Lagomarsino (1993) analyzed the frequency dependencies of damping ratio using full-**scale data** on 182 build- ings. This paper uses reliable recently measured **data** on the dy- namic properties of buildings in Japan, mainly in the **small** am- plitude region, and …
Cited by 301 Related articles All 7 versions

[HTML] The tribological behaviour of **engineering** plastics during sliding friction investigated with **small-scale** specimens
L Zsidai, P De Baets, P Samyn, G Kalacska… - Wear, 2002 - Elsevier
… Unfortunately, **data** on their friction and wear characteristics are very disparate and often there … In the present paper some basic types of **engineering** plastics (PA, oil-filled PA … filled PETP, POM-H) are experimentally investigated by means of **small-scale** reciprocating tribotesting …
Cited by 140 Related articles All 9 versions

Online self-reconfiguration with performance guarantee for energy-efficient large-**scale** cloud computing **data** centers
H Mi, H Wang, G Yin, Y Zhou, D Shi… - 2010 IEEE International …, 2010 - ieeexplore.ieee.org
… Kong, Shatin, Hong Kong 3. School of Electronic Technology, Information **Engineering** University, Zhengzhou … have been many studies trying to realize self-reconfiguration of **data** centers so as … However, most approaches can only be applied in **small scale** cluster because of the …
Cited by 186 Related articles All 9 versions

Figure 1.1: search results from google scholar as of 10th May 2020.

This lack of literature is not surprising since machine learning related skills are currently hard to acquire for companies, and as such even large companies with well funded IT departments are struggling to apply machine learning in production [4]. If large scale efforts with big funding are targeting and hog much of the talent in attempts to fill their own needs, it would be natural that the conversation is run by their perspective and interests. Additionally, it is the case that machine learning performs especially well on large and complex datasets. However, this should not be confused with the notion of big data[6], as it refers to not only large datasets but rather big ecosystems of large, often huge, datasets and the rapid growth and proliferation of data sources. There are plenty of large and complex datasets where machine learning is a feasible approach where the term big data is not really appropriate.

There is also a lack of tooling and infrastructure services in regards to data engineering. As machine learning becomes more available to small and medium-scale data science efforts through advancements in tooling, the field of data engineering needs to keep up [4]. Without well managed and high-quality data, companies will struggle to reap the rewards of machine learning since good data is the basis in which any successful data science endeavor is built.

This need for a solid foundation to build ones data science efforts on is well described by Figure 1.2. The move/store and the explore/transform levels of the pyramid in Figure 1.2 will be the focal points of this thesis. We find that driving liveness, i.e high rate of contribution to a system amongst its users, and value creation, i.e how well a system delivers value to its stakeholders, to be essential challenges in these two areas.

In this report, we exemplify some of the challenges related to small scale data engineering for machine learning and we attempt to inspire solutions to these. We illustrate that there is a need for further research on this subject and this is the goal of this thesis.

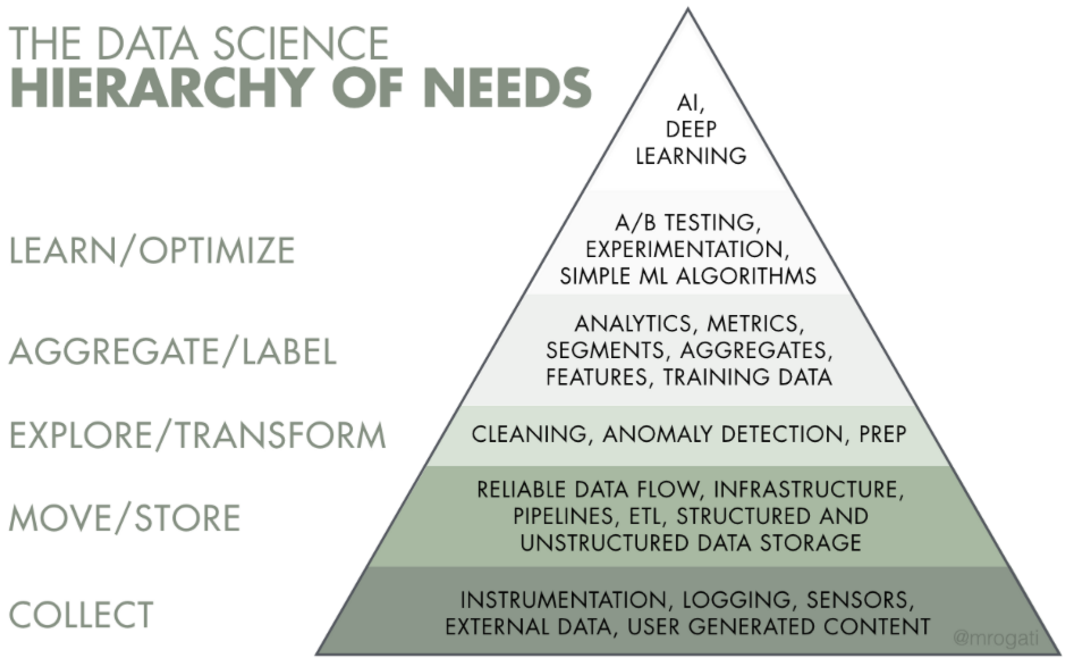

THE DATA SCIENCE
HIERARCHY OF NEEDS
LEARN/OPTIMIZE
AGGREGATE/LABEL
EXPLORE/TRANSFORM
MOVE/STORE
COLLECT
AI, DEEP LEARNING
A/B TESTING, EXPERIMENTATION, SIMPLE ML ALGORITHMS
ANALYTICS, METRICS, SEGMENTS, AGGREGATES, FEATURES, TRAINING DATA
CLEANING, ANOMALY DETECTION, PREP
RELIABLE DATA FLOW, INFRASTRUCTURE, PIPELINES, ETL, STRUCTURED AND UNSTRUCTURED DATA STORAGE
INSTRUMENTATION, LOGGING, SENSORS, EXTERNAL DATA, USER GENERATED CONTENT
@mrogati

Figure 1.2: The data science hierarchy of needs, which is depicted as a pyramid, describes the dependecies of data science. Credit Monica Rogati, Hackernoon.

1.1 Research Questions

This thesis started out with a very broad and general research question, formulated as RQ1 below.

> RQ1: *"What are the most discussed challenges currently in data engineering literature and what are their state of the art solutions? "*

As we started to understand the available literature, we found that some of the literature on data platforms especially and data versioning could be of value to Sentian. This led to RQ1 being reformulated as RQs 1.1 and 1.2 below.

> RQ1.1: *"What is the state of the art of data platform architecture and how do one drive liveness and value creation of a data platform?"*
>
> RQ1.2: *"How well can a state of the art data platform architecture be applied to a small scale machine learning context outside of big data?"*

The research questions RQs 1.1 and 1.2 were expanded during this thesis with a second broad research question, RQ2. This was done to further substantiate our answer to RQs 1.1 and 1.2 as the literature we found was deemed to be insufficient to mitigate Sentians challenges.

RQ2: *"how can data engineering be made agile for small scale teams?"*

RQ2 was then, just like RQ1, narrowed down into RQs 2.1 and 2.2.

RQ2.1: *"Can the principles behind the agile manifesto be mapped to the processes of state of the art data platforms to extract insights that may be applied to a small scale context?"*

RQ2.2: *"Are there practices in small scale data engineering that map well to the principles of the agile manifesto?"*

A guide for where the RQs will be answered throughout the report can be found in section 1.4, Covering the Report Outline, below.

1.2 Approach

The swimlane diagram in Figure 1.3 serves to outline the phases of this thesis on a high-level view. Of course, the work was carried out iteratively throughout the thesis, going back and forth between the different lanes more often than described in this chapter. The purpose of the swimlane diagram is to clarify the logic sequence of the execution of this thesis as its results are mainly theoretical.

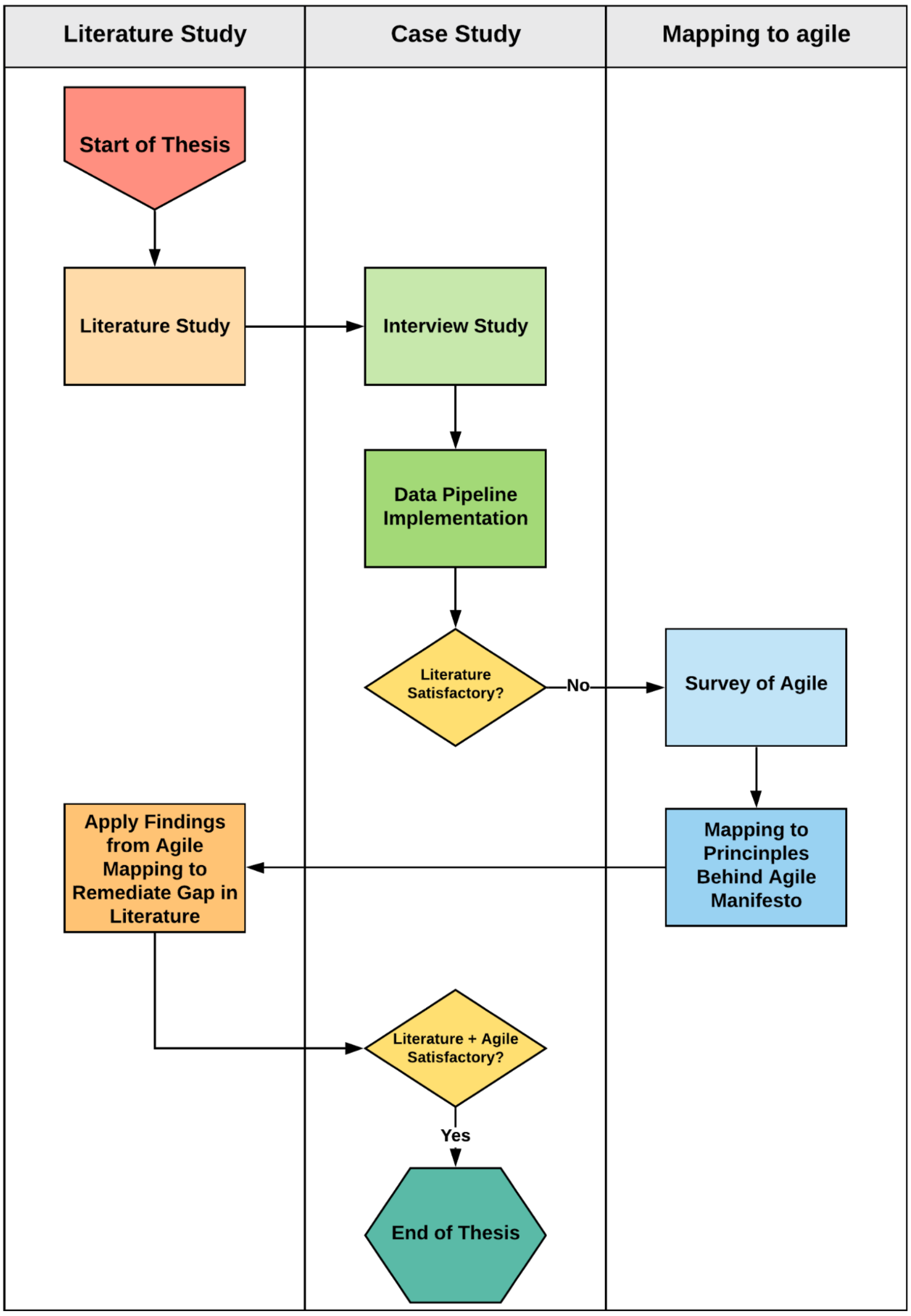

Literature Study
Case Study
Mapping to agile
Start of Thesis
Literature Study
Interview Study
Data Pipeline Implementation
Literature Satisfactory?
No
Survey of Agile
Mapping to Princinples Behind Agile Manifesto
Apply Findings from Agile Mapping to Remediate Gap in Literature
Literature + Agile Satisfactory?
Yes
End of Thesis

Figure 1.3: Swimlane diagram of thesis execution.

This thesis commenced with a literature study of current challenges, designs, and frameworks related to data engineering for machine learning. The findings from the literature study were applied in a case study of a small machine learning company that specializes in solutions for industry. The case study was performed at the company Sentian AI in two parts, both qualitatively and practically, that along with the literature study would allow for triangulation of the core challenges of small scale data engineering [7]. When applying the literature study to the case study we were interested in exploring if the current literature on data engineering for machine learning pipelines was sufficient in solving the set of challenges that the company in the case study was facing.

In comparing the results of the literature study to the context of the company in the case study, it became evident that the literature we found in the study was not sufficient in solving the challenges found in the case study. However, similarities between the state of the art solutions in data platforms and agile principles were noted during this analysis. This led to an additional phase in the approach of this thesis: a mapping of the principles behind the agile manifesto [8][9] to a current state of the art data platform and to small scale data engineering. The purpose was to explore if agile could be helpful in extracting the core concepts from the literature that was found so that they can be reapplied to a smaller context.

Throughout the thesis, it proved difficult to cover the subjects entirely through published academic literature. The field of data engineering is young and much of the conversation on the state of the art and current challenges is had through what is known as grey literature, which is common for software engineering-related fields [10]. Therefore, we have made the decision to include findings from grey literature throughout this thesis as too much relevant information would be lost without it.

1.3 Contributions

This thesis provides an exploratory look into what challenges and needs are found in the context of small scale data engineering for machine learning. It employs (1) the state of the art data platform architecture and data versioning, (2) a case study of a small specialized machine learning company, and (3) agile principles to suggest practices for designing and implementing data platforms and data pipelines in this context.

Our findings include (1) that a light version of a state of the art data platform can be implemented for a small scale context if modified, (2) that there is a gap between the needs of the small scale data engineering efforts in the case study and the literature found in the literature study, and (3) a potential for knowledge transfer from the software engineering domain through agile principles to the emerging domain of data engineering.

1.4 Covering the Report Outline

The report is structured around the three main phases of the thesis project: a literature study is presented in chapter 3, a case study of a small specialized machine learning company is presented in chapter 4, and a mapping of agile principles from the software domain to the data engineering domain is presented in chapter 5. Each chapter provides its own method, results and an accompanying analysis of those results. The final discussion in chapter 6 ties together the findings from the three project phases in order to answer RQs 1.2, 2.1 and 2.2. RQ 1.1 is primarily answered through the analysis of the literature study, although how to drive liveness and value creation of a data platform is a common theme throughout the report. Finally, we conclude the report in chapter 7.

Chapter 2
Background

This chapter introduces the concepts of agile development and the basics of data engineering as these are necessary for a reader to properly read and understand the findings and the context of this report.

2.1 Data Engineering

Data engineer is a role that has emerged and separated from the role of data scientist as the field of data science has matured and increasingly been applied in production level solutions. It deals with the infrastructure that prepares, moves and hosts data so that it can be made available for data science in production [11]. The field of data engineering is rather young but has recently received much attention and is one of the fastest-growing occupations within information technology [12].

2.2 Agile Principles

The purpose of agile software development, whether one adheres to XP, SCRUM a combination of these or some other framework, is to consider and understand the uncertainties one is facing in a specific environment and how to adapt to these as you go along [9]. There are many practices associated with agile such as pair programming, stand-ups and sprints. These do not define agile but rather these have sprung from agile philosophy as solutions to some of the problems often experienced in software development.

Although the Agile manifesto and the 12 principles behind the agile manifesto are often consulted when defining what is considered agile, there have been continuous discussion and criticisms of it being the ultimate document that defines agile. In a study made to map other definitions of agile and their emphases compared to those of the agile manifesto the words: flexible, people-centric, responsive and iterative are common [13].

It should also be noted that the concept of agile does not originate from a software context. The use of agile methodology is widespread throughout many industries [14]. Kanban boards, an agile way of managing workflows, which was adapted in 2004 at Microsoft for software development purposes has its origin in Japanese car manufacturing where it was applied in a Toyota manufacturing plant in 1953 [15]. There are many practices like this, especially from the domain of lean production in manufacturing, that have been adopted by the movement of agile software development.

2.3 Data Pipelines

As insinuated by the name a data pipeline moves data, often from one system or subsystem to another. It inputs data, may perform a series of data processing steps where the output of one process serves as input to the next one, and it outputs data as a result. The term is used both for data batch jobs as well as for streaming.

A pipeline can be represented abstractly as a directed acyclic graph (DAG). A DAG is a finite graph of nodes in which there is no way, starting at any node in the DAG, to get back to the starting point.

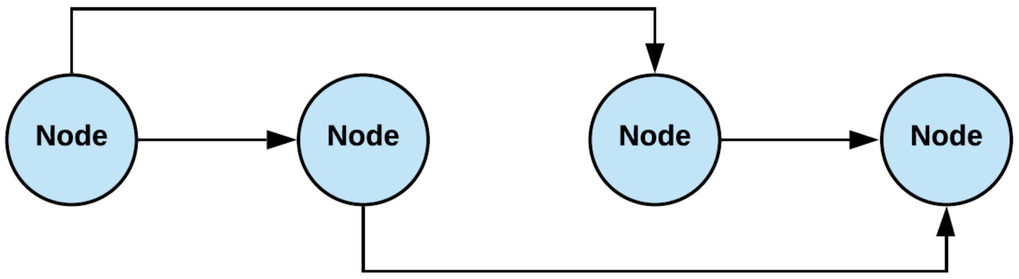


Figure 2.1: example of a DAG

In regards to moving and processing data, often in batch jobs, and storing them on a data platform, there are two main concepts: ETL and ELT (notice the order of the "L"). In short, both are a subset of the more generic term data pipelines [16]. As such they tend to be used interchangeably with the term data pipeline. ETL is a process that uses a data pipeline to Extract, Transform and Load (ETL) data into a database or a data warehouse that often stores structured data. As mentioned, ETL pipelines are implied to be batch-oriented and run on a time schedule unlike data pipelines which is the term that tends to be used in streaming contexts.

While the ETL process transforms the data and casts it to a specified format before loading it into persistent storage, Extract Load Transform (ELT) transforms the data after it has been loaded into persistent storage. It can be said that ETL is read on schema and ELT is schema on read. Meaning that using ELT, one does not apply schema, i.e. transformations, on data until it is read from its persistent storage. ELT is compatible with data lakes as these store raw data, both structured and unstructured, which is then transformed on consumption [17].

2.4 Data Understanding and Validation

Once a data engineer is assigned the task of setting up a data pipeline a significant portion is spent manually analyzing the data that it will consume. It is important to understand the properties and features of the data, but also to understand where it is heading and what its intended use is. With this context in mind, the data engineer should validate data as it enters the data pipeline which can be done in many different ways [3]. To mention a few: data types, value ranges, correlations, means and standard deviations.

For machine learning purposes where the data pipeline will feed a machine learning model data, it is useful to analyze distributions and ranges for data. A clear understanding of these will be important for monitoring the data pipeline in production and to discover when training data deviates from the data that is served to the model to make inferences (predictions). The phenomenon is known as training-serving skew and can occur for a multitude of reasons. For example, it could be that the system or behavior that the data represents has changed or it could simply be caused by a software error. This is a major source of problems in production-level machine learning and is important to consider since it negatively affects inference accuracy [3].

One way of implementing data validation is to produce a data validation schema that dictates what valid data looks like and that checks any data that is passed through it. Ideally, this should not only be done for the initial point of ingestion into the data pipeline but anywhere new data enters the pipeline. There is always the risk of bugs or human error resulting in data being improperly transformed along the way through the different processes.

2.5 Data Cleaning

A major problem with data is that it is often dirty when it arrives from its source. Dirty data can be summarized as data where parts of it are missing, i.e. missing values, wrong data and data in non-standard representations [18]. Remedying or dealing with these flaws is known as data cleaning. It might not always be possible to remedy the root cause of dirty data, which might be a bug in the code of the source system. In these cases, the data cleaning can be linked to a data validation schema as an integrated part of the data pipeline. However, considering that wrong data can be generated throughout a data pipeline from software errors, it is prudent to include checks and cleaning not only where entering data is validated.

2.6 Data Enrichment

Training and serving data from a single source may not be enough, or results could be improved, by introducing or joining new data from another source. This can provide the machine learning model with new features that help it make more precise inferences. One can also enrich data through augmentation. This is often employed when there is a lack of training data, or in the case of class inference if there is an imbalance between the amount of training data for each class. Data augmentation is often utilized in training models utilizing deep convolutional neural networks as these require vast amounts of data which is expensive to gather and label [19][20].

2.7 Containerization

Containers are a lightweight virtualization concept that bundles a software application along with its environment including configuration files, libraries and other dependecies that are required to run the application [21]. Docker is one such container technology which runs natively on Linux and Windows. Containers can be organized in clusters of host nodes where each host node contains multiple containers. One such orchestration tool that automatically handles scaling through addition and removal of Docker [22] containers according to the workload of the cluster is Kubernetes [23].

Chapter 3
Literature Study

The literature study focuses on the state of the art of ELT type data platforms and data versioning but it also covers data pipeline testing on a basic level as it is important to ensure the quality of data. Challenges related to implementation in data engineering tend to be of ad hoc nature, therefore the focus of the literature study is on the bigger perspective of how to successfully drive value from data rather than on specific technical challenges.

3.1 Method

A semi-structured literature study was employed to find related research and proposed solutions to RQ1.1 [24]. The reason for choosing to do the literature study in a semi-structured way was because solutions related to data have served many different domains. As such it is written about for many different purposes by different groups with different interests, making it hard to cover all possible source domains on the subject. Sources that aimed at solving the challenges in a context where exploratory insights from the data were sought after, such as data science, were prioritized. Thus data platforms became one of the main focuses of the literature study as this is a ubiquitous part in many solutions related to data science.

The importance of availability of all data in its raw format, to support exploratory insights, had a large impact on what kind of data platform design was to be considered. It essentially bounded the context to some sort of ELT type data platform.

A data platform solution is not complete without some sort of policy for how to manage the data that it hosts. Data management is a vast field of research, it covers a large number of topics and is an active area of research for many different communities. Since this thesis is scoped to a machine learning context there will be a focus on the topics of data versioning and data provenance. These are especially hot topics within the machine learning community [25][3][26].

Although the machine learning community has made an effort to focus on these issues in a production context recently, the scientific community has long been working on data versioning interdisciplinarily [27][28] for shareability of datasets and reproducibility of experiments. The work made by the scientific community is more mature and comprehensive than the more recent work of the machine learning community that focuses on their specific context, therefore we cover both.

In the literature study chapter we present material from 27 sources altogether on the subjects of (1) data platforms [25, 29-36] (2) data versioning [27, 28, 37-41] (3) data provenance [42-45] and (4) data pipeline testing [46-50]. Five of these sources are grey literature, four of these are on the topics of data platforms and one on data versioning.

3.2 Data Platform Architectures

To help a reader understand the three data platform architectures that are described in this chapter we provide a top-level overview of how they relate to one and another and the origin of raw data.

The data warehouse is an older architecture that solved the problem of consolidating business data that was spread out in operational databases and that could be used for business intelligence purposes. The data lake is an architecture which arrived later during the rise of NoSQL databases [29]. It is a solution that deals with challenges posed by big data and the movement of data-driven organizations. The data lake and the data warehouse architectures are not exclusive though, as noted by the creator of the data lake architecture: a data lake contains raw data that could be used to populate a downstream data warehouse [30].

The data mesh is a recent state of the art architecture, based on the original idea of the data lake, and it tries to solve some of the problems related to the scalability of data lakes. It specifically focuses on how to promote discoverability and utilization of the data in the data lake.

With regard to the raw data source, the data lake and data mesh architectures lies the closest and the data warehouse architecture lies further downstream. The further away data is from its source, the more processed it becomes and limited in its possibilities since any transformation of data narrows its use case.

3.2.1 Data Warehouse

The data warehouse is coupled with the field of business intelligence, within which it is a core component that serves the purpose of a centralized data storage [31]. It extracts, transforms and loads data from one or more source systems and makes it available for analysis. A data warehouse is made up of subsets referred to as data marts which serve specific parts of the business. The data, and its format, that is stored in a data warehouse is predefined and optimized for retrieval for specific business purposes. The data warehouse preserves historical data for large data systems over long periods of time. Unlike transactional systems for storing historical data it manages slowly changing dimensions. The purpose of a data warehouse is to answer specific questions about a business to support decision making.

3.2.2 Data Lakes

The term data lake was first coined by James Dixon, then CTO of Pentaho, in a blog post [32] 2010 in the following way: "If you think of a datamart as a store of bottled water – cleansed and packaged and structured for easy consumption – the data lake is a large body of water in a more natural state. The contents of the data lake stream in from a source to fill the lake, and various users of the lake can come to examine, dive in, or take samples."

One of the big distinctions between a data warehouse and a data lake is the order of ingestion into its repository. While a data warehouse will only ingest structured and carefully selected ETL-ed data, the data lake will ingest any data in its raw, pre transformed format. The ETL process is switched to ELT (Extract-Load-Transform) where the data will be extracted from its sources and be loaded into the data lake in its raw format.

The data in a data lake is instead transformed at the read stage where it is consumed. This allows for all data, whether structured or not, to be stored in the same place. It enables data scientists to ask ad hoc questions that were not conceived or planned for when constructing the data platform from whence the data is drawn to answer them. This allows data scientists to use their domain knowledge to pose questions that may require out of the box solutions that would otherwise have been hindered by the limited and summarized nature of data provided through data warehouses.

It is estimated that a majority of the data available through an enterprise is of semi or unstructured format, this data would not be easily available for a data scientist with a data warehouse solution [33].

Although data lakes theoretically offer a larger potential than data warehouses through a wider scope of data collection, they also run an increased risk of becoming unmanageable and hard to extract value from, they are then referred to as data swamps [34][35][25].

3.2.3 Data Mesh

The data mesh, defined by Zhamak Dehghani in [36], is an adaptation of the data lake architecture that seeks to remedy what she identifies as three architectural failure modes that lead to under-realized value of a data lake as it scales. The claim is that these failures lead to an under-realized value creation due to data inaccessibility because of poor data governance practices.

She identifies a lack of bounded context and domain driven design [37] in favor of centralized domain agnostic data ownership, which results in monolithic data platforms, as one of the key concerns with the current state of data lake architectures. This is where the data mesh significantly deviates from the data lake. It embraces the concepts of domain driven design and considers the domains a first class concern over the technical implementation of the data pipelines and the data platform itself.

This somewhat ties back to the initial philosophy of James Dixon; that a data lake should contain data from a single data source (or system) while many data lakes within an enterprise may be considered a "water garden" [30] . However, in a data mesh design the domains are the primary focus and not their source systems and as such domains can be drawn to encompass datasets from multiple data sources. These domains are then what constitute the nodes in the metaphorical data mesh similar to the data lakes of a water garden.

Dehghani describes three architectural failure modes present in current data lake designs:

1. Monolithic and centralized:

   (a) Ubiquitous data and source proliferation. It becomes hard to serve and harmonize ever-growing amounts and sources of data when you're working from a siloed engineering team.

   (b) Innovation agenda and consumer proliferation. The organizations' need for rapid experimentation leads to an increasingly tall order of data-jobs and use cases from data consumers that are hard to satisfy in a timely manner. In her experience, this leads to friction within the organization.

2. High coupling of data pipeline decomposition:

   (a) As a legacy of the ETL bound data platforms, it is common to decompose the data platform into a pipeline of processing stages as viewed in Figure 3.1. It offers some scalability in being able to work in functionally organized teams designated to different stages of the pipeline provided through the data platform. However, there is high coupling within the pipeline architecture which slows down the delivery of features. It is decomposed orthogonally to the axis of change since any change, for example of functionality or the addition of a data source, easily can affect pipeline implementation in all stages of its architecture. This lies in the sequential nature of pipeline architecture where every node, except for the output layer, feeds input for another node. The implementations of the changes to the pipeline have to be managed and their releases synchronized across multiple teams.

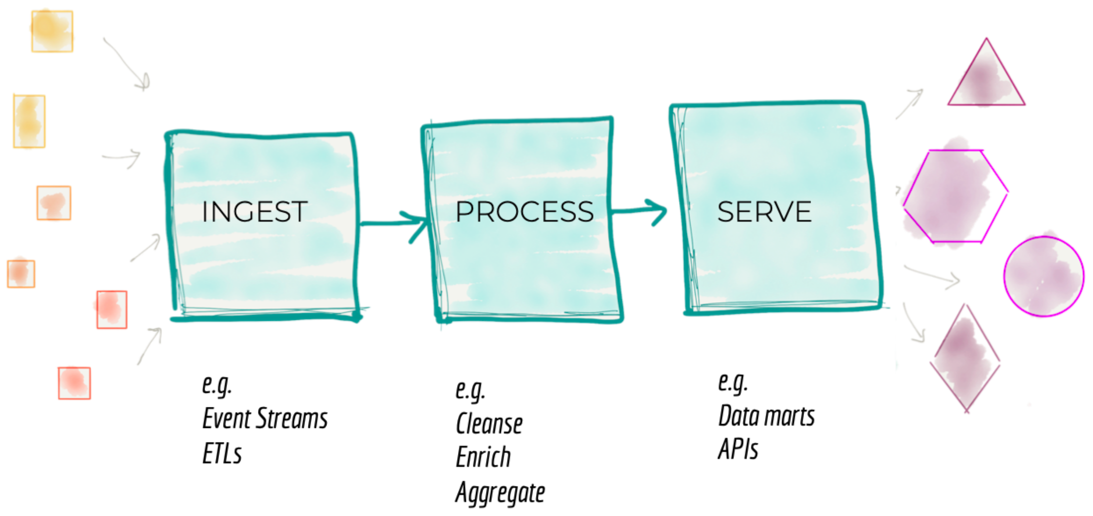


Figure 3.1: Common decomposition of a data platform into its pipeline processes, credit Zhamak Dehghani.

3. Siloed and hyper-specialized teams:

   (a) With the monolithic data lake approach, data engineers are siloed together with other data engineers who possess the same knowledge about tooling and data engineering skills. They become separated from the operational teams where the data they consume originates. The operational teams also hold no incentive to provide meaningful, truthful and correct data. To add to this data engineering teams often lack domain knowledge that they need, to understand the data but also the needs of the data consumer. This makes it increasingly hard to deliver useful data and to service the requests of data lake users.

Per the data mesh approach, every domain that provides data as a product has ownership over the data through a cross-functional domain team consisting of at least one data engineer and a data product owner. The domains themselves can be based on both source and consumption of data. Source oriented domains align with the origin of the data while consumer-oriented domains align with shared usage of specific data. The consumer-oriented domains are managed by a domain team that seeks to satisfy a specific set of use cases and often manage transformed and aggregated data from other source domains.

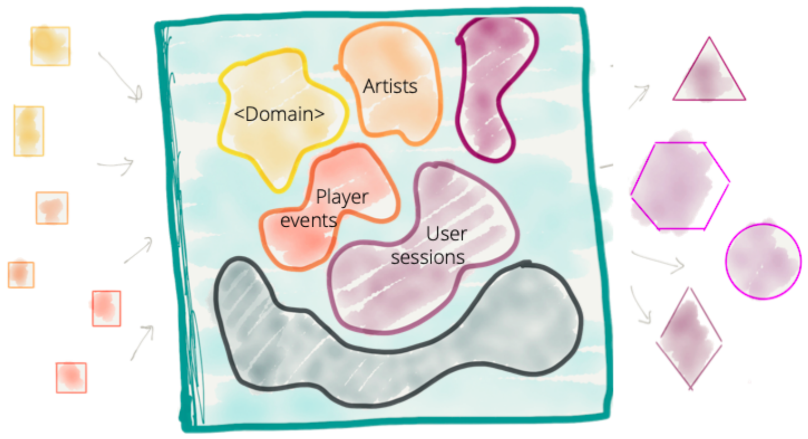


Figure 3.2: The data platform for a music streaming company, monolithic approach, credit Zhamak Dehghani.

Fig 3.2 depicts the monolithic data platform of a hypothetical music streaming company as the big blue box. It manages raw data that can be tied to domains. The boxes to the left of the monolith are domains that source raw data to the data platform. The shapes to the right are consumer domains that aggregate and transform data into enriched data sets.

With a data mesh approach the conventional monolith data platform would more closely resemble Figure 3.3 where the data belongs to its domain, managed by domain teams, rather than the centrally managed monolith that previously claimed ownership of the data.

The data engineer works with ensuring that the stipulated service level objectives (SLO) are met. As an example, a source domain may express these as timeliness and error rates of data and manage them through data cleaning performed by the domain data engineer. They work to uphold the quality of the raw data their domain provides or, if they are part of a consumer domain team, with consuming, aggregating and processing data provided by other domains. The technical implementation is done through a common data infrastructure platform that is developed and maintained by a designated team of data infrastructure engineers. As in Figure 3.4, the data pipeline implementations are now moved to and maintained by the data engineers of the domain where they are used.

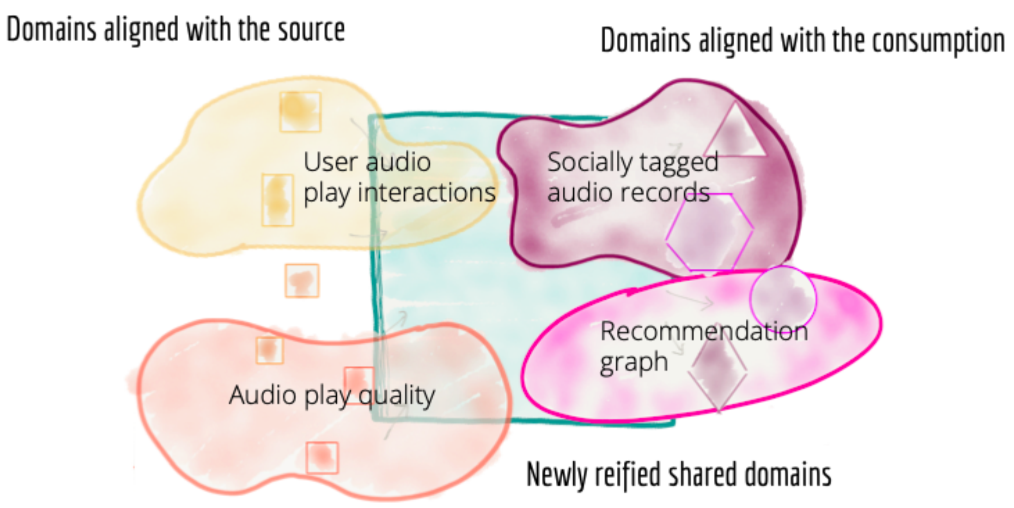

Domains aligned with the source
Domains aligned with the consumption
User audio play interactions
Socially tagged audio records
Recommendation graph
Audio play quality
Newly reified shared domains

Figure 3.3: The data platform following a data mesh approach, credit Zhamak Dehghani.

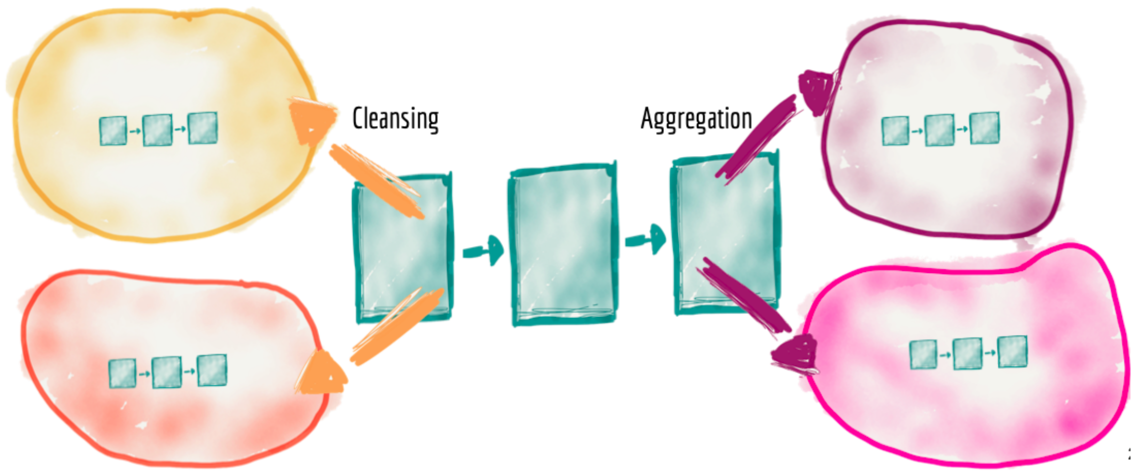


Figure 3.4: The data pipeline is considered a second class concern and its implementation is moved to its respective source or consumption domain, credit Zhamak Dehghani.

The data product owner takes responsibility for the consumption of the data sourced from their domain through tracking of key performance indicators (KPI). They consider what changes need to be made to their node in the data mesh, whether it be improvements of the data quality, the discoverability of data or something else, to drive the value their domain offers their consumers. The domains that source data, providing it for the rest of the organization, should treat the data consumers as customers and seek to delight them. The data product owner is responsible for striking the balance between the present-day competing needs of their consumers while adhering to long term perspective strategies for their data products.

Even though every domain team claims ownership of their data there are policies on an organizational level for how to handle and serve data. This field is known as data governance and parts of the field is described in the next section on data versioning. In short, every domain team should follow common formatting practices for how to store data, what to include in its description and how to trace any potential lineage if the data has been transformed. Traceability of data that has been transformed is essential to ensuring its quality for any data scientist who wishes to consume it.

The idea is that if cataloguing of the data, and its metadata, that is hosted in the data mesh is done well enough, no hand holding will be needed for consumption of the data. All of the data may be stored in a shared cloud storage location. However, a centrally located data engineering team is no longer responsible for gathering all the data and storing it in a central data warehouse or data lake. Instead the domain team cleans and formats the data according to standards issued on an organization wide level. This domain-focus approach allows for gathering and refining domain specific data along with metadata that is only contained and understood within specific domains that may now be made available for outside consumers. It moves domain specific knowledge closer to the data engineer.

The data infrastructure platform should be built avoiding domain specific concepts, it should be domain agnostic, and it should seek to hide any underlying complexity. The end goal is that it should be used as a self-serve platform by the data engineers to deliver their data products. The infrastructure team should aim to automate as much as possible of the processes performed by the data engineers. A focus for the data infrastructure team on lowering the lead time to create a new data product is suggested as a success criteria which may drive this automation. A concrete lists of some of the data infrastructure platforms capabilities is presented in the original blog post [36].

To conclude a data mesh design is a distributed design that organizes its nodes according to their domain context of data consumption or data sourcing. It moves the technical implementation and the data engineer competence to these domains which reduces siloing of data engineers. This in turn promotes cross functional teams that are better suited to take a holistic view of the data they work with, these teams are better suited to ensure that value from the data is realized. Clear ownership of data is defined which provides accountability and incentive that value is realized in practise. In the end one ends up with something like Figure 3.5 depicting a data mesh below, where the data domains are supported by the infrastructure as a platform to maximise the organizational value of the datasets in their domains under the guidance of global governance and policy standards.

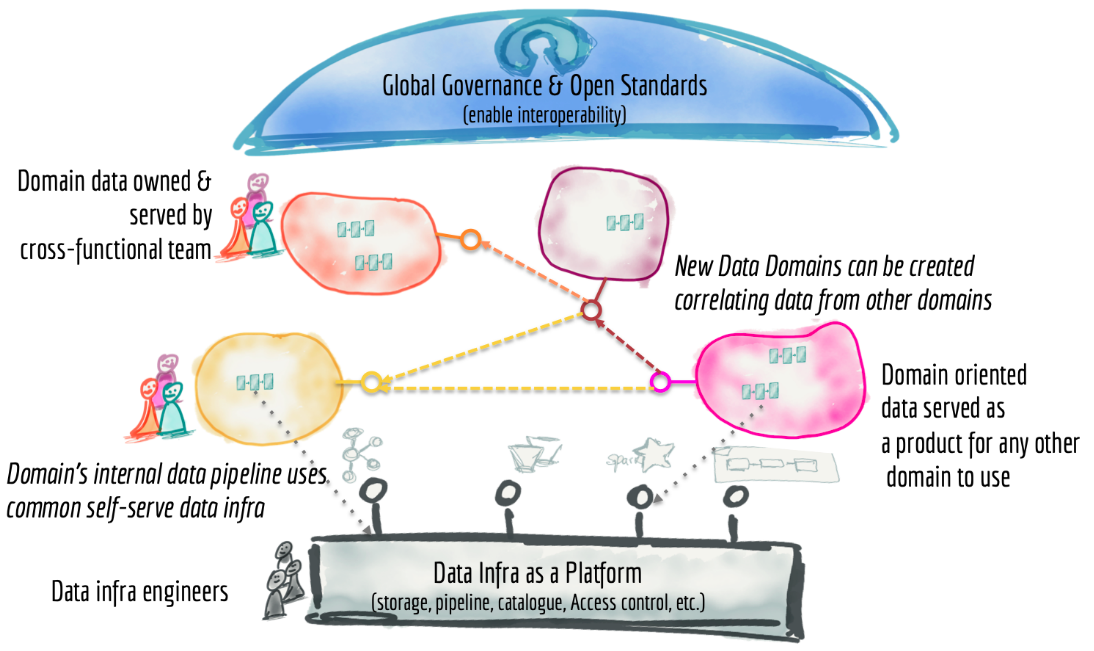

Global Governance & Open Standards
(enable interoperability)
Domain data owned & served by cross-functional team
New Data Domains can be created correlating data from other domains
Domain oriented data served as a product for any other domain to use
Domain's internal data pipeline uses common self-serve data infra
Data infra engineers
Data Infra as a Platform
(storage, pipeline, catalogue, Access control, etc.)

Figure 3.5: High level view of what a data mesh ecosystem looks like, credit Chamak Dehghani.

3.3 Data Versioning

The scientific community has long recognized the need for sharing data interdisciplinarily and that this requires cataloging [28]. In 2009 an entity-relation cataloging system from the library domain, FRBR [38], was applied to the context of scientific data [39]. In 2016 the FAIR principles were released for how to make data Findable, Accessible, Interoperable and Reusable in response to the requirement of data management plans [40]. These were still considered to not be sufficient in providing reproducibility of experiments and promoting reuse of data, a request for further research in systematic data versioning practices was voiced by [41]. The W3C Dataset Exchange Working Group further highlighted the lack of definitions of data versioning concepts and recommended practices in 2017.

In response to this The Research Data Alliance (RDA) in 2020 produced a report that delivers principles and best practices for data versioning of research datasets through documentation of use cases. The use cases were gathered from 33 different organisations within different research fields to capture a comprehensive set of scenarios with different needs and challenges [27].

RDA [27] build on the work of Hourclé [39] and his application of the FRBR framework to a scientific context. FRBR defines four levels for an entity that can be used for identification. RDA [27] kept the four levels of the FRBR framework but modified them to suit a scientific context, their adaptation is presented verbatim below.

1. A **Work** is the observation that results in the estimation of the value of a feature property, and it involves application of a specified procedure, such as a sensor instrument, algorithm or process chain;

2. An **Expression** of a work is the realisation of a work in the form of a logical data product. Any change in the data model or content constitutes a change in expression;
3. A **Manifestation** is the embodiment of an expression of a work, e.g as a file in a specific structure and encoding. Any changes in its form (e.g. file structure, encoding) is considered a new manifestation; and
4. An **Item** is a concrete entity, representing a single exemplar of a manifestation, e.g. a specific data file in an individual, named data repository. An item can be one or more than one object (e.g. a collection of files bundled in a container object).

RDA applies this framework to 39 use cases from 33 different organisations within different scientific fields to identify a number of issues related to data versioning. They identify that the recommendation made in previous literature [41], that states of datasets should be unambiguously identifiable, poses further questions. If one were to implement this, for example through checksum of the bitstream of each respective dataset, any tiny change would constitute a new version. It is not clear that simply noting a change through checksum is enough information for a user to evaluate different versions. Therefore they pose another set of six questions related to the issues they uncovered in applying their framework to the use cases. As the sixth question relates to scientific citation it will be excluded.

1. **Revision (Version control)** - What constitutes a change in a dataset?
   The previous recommendation of unambiguous identification of datasets in [41] stands and should be performed on a fine-granular level. One should make the distinction between changing the dataset and changing its metadata, an update to the metadata shall not be considered a change to the dataset itself and does not warrant a new revision.

2. **Release (Data production)** - What are the magnitude and significance of the change?
   A dataset may be modified through several iterations before it is considered ready for release, each modification sparking a new revision. A user of a previous release of the dataset needs to understand the significance of the change and if the new release is compatible with their dependencies. They recommend the work of [42] on Semantic Versioning that provides best practices on communicating the significance of a change and its compatibility with previous releases.

3. **Manifestation (Formats)** - Are the differences in the bitstream due to different representation forms?
   Data may be manifested differently, for example through different file formats, depending on what workflow it is intended to be used for. The different manifestations may be identified separately but they should identify to the same work (the first and most high level of an entity described in the four proposed levels of the framework above.).

4. **Granularity (Objects and collections)** - If the data are part of a collection and which elements of the collection have changed?
   Datasets may be generated successively and constitute parts of a collection, which is

often the case for time series. There may be multiple levels of subcollections to a collection. The level of granularity should be determined and adapted to the use case in question.

5. **Provenance (Derived products)** - How do two versions relate to each other?
   In the case that a dataset has been derived from other datasets it is important for the user to be able to trace its predecessors and understand how they relate to each other. (More on this below in a dedicated chapter called "Data provenance")

To conclude its recommendations, the report emphasizes the difference between versioning datasets based on changes of content (dataset revisions) and communicating the significance of the change in question. They identify two key concepts of data versioning as (1) being clear about which dataset is to be identified and (2) what one wants to communicate about the dataset in question to its consumer or end user.

3.4 Data Provenance

Data provenance provides a historical record of data with the purpose of reproducibility. It explains how data flowed from its origin through intermediate steps and processing to its current state [43]. It has been described briefly in the data versioning chapter above, this chapter will expand on the subject and present additional sources from literature which describe how it may be implemented.

In 2006 the first International Provenance and Annotation Workshop (IPAW'06) was held. Through a series of provenance challenges (PC1,2,3,4) posed by IPAW, the research community reached an agreement on the core representation of provenance. This led to the Open Provenance Model (OPM) [44], which is technology agnostic and defines a set of rules for what inferences are valid to make off of a provenance graph. The provenance group at World Wide Web Consortium (W3C) later took over the research of the IPAW provenance challenge workshop and released PROV-DM which is based on OPM. PROV-DM models provenance through graphing with three types of nodes and seven types of edges [43]. These can be seen in Figure 3.6.

Wang et al. [45] cover the state of the art in big data provenance, as of 2015, and propose a reference architecture. Even though they explicitly focus on challenges related to big data, their overall design of how to implement data provenance is general and applicable to a small scale context as well. The provenance subsystem they propose can be broken down into three dimensions: data, lineage and environment. In their design, data provenance captures the state of the dataset throughout execution, lineage provenance stores the computational activity and environment provenance saves the exact state (both hardware and software) of the system configuration.

In the context of production-level machine learning, provenance can look something like this [46] (Example from Microsoft): "Each [machine learning] model is tagged with a provenance tag that explains with which data it has been trained on and which version of the model. Each dataset is tagged with information about where it originated from and which version of the code was used to extract it (and any related features).".

There are other recent applications of provenance being applied to machine learning contexts such as [25] which describes how provenance data can be used in practise through

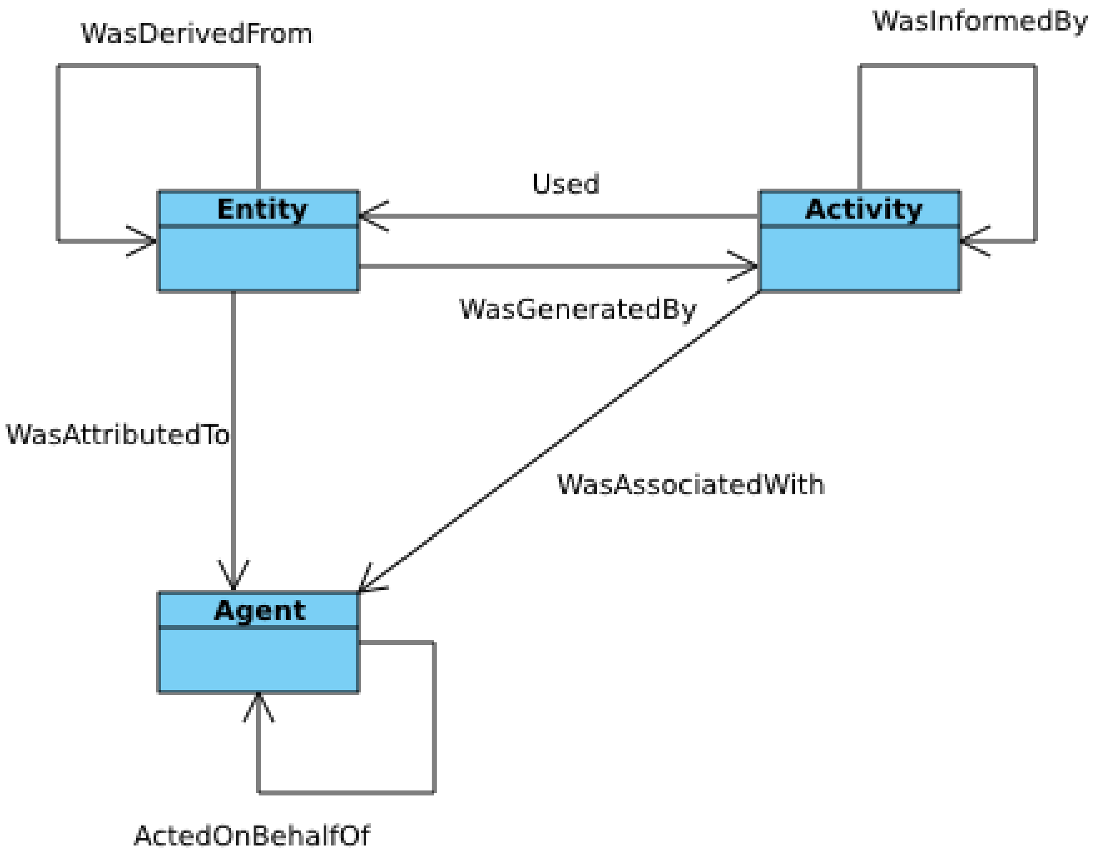


Figure 3.6: entities and edges of PROV-DM model.

their adaptation of PROV-DM, PROV-ML, in a use case in the oil industry.

3.5 Data Pipeline Testing

Testing in software development is important to the quality software [47]. Testing data pipelines is difficult because it poses the challenge of not only testing code but of testing code concurrently to data. In a recent survey of techniques used for testing machine learning systems Zhang et al. identify data as one of the components that needs testing [48].

Discovering bugs early is important, as previous inferences made by a model are often used further on as training data in updating the model, which imposes the risk that small errors in data can amplify over time through feedback loops. When testing a data pipeline, which is often a part of a machine learning pipeline, the lack of data generation logic and semantics makes testing it and tracing the origin of data bugs even harder [49].

Data validation schemas are needed throughout the data pipeline since data formats may be changed during the data pipeline lifecycle. The process of data validation can be fully automated [50]. However, the schema describing the data is in some cases a property of the data pipeline and not the data source which introduces the risk that the schema may diverge from the data source as they both evolve [3].

Hynes et al. [51] propose that although data validation and data cleaning may improve the quality of the data fed to the model during training and subsequently inference, one can still end up with a suboptimal representation of the data for a given model which will affect its performance. Their solution is to apply a new category of tools that they refer to as data linters, inspired by software linters. A data linter would take both the features present in the data and the model type into consideration to propose corrections to the representation of the data in question. An example would be that, if numeric values are being fed into a linear model, these should be normalized as it is likely to improve the quality of the models inferences.

3.6 Analysis of the Literature

In this analysis we discuss how well the literature that was found in the literature study answers RQ1.1. We also analyse the role of data versioning and data pipeline testing in driving liveness and value creation of a data platform.

Overall the literature that was found on data engineering for machine learning is often aimed at contexts of big data and the solutions it discusses assumes availability to resources and advantages tied to large scale operations [46][36]. We found little research on data platforms and data pipelines with regards to a small scale data engineering context in this literature study. Most of the academic literature that was found on data platforms focused on big data and issues related to scale, and their solutions often include tools based on distributed computing. The grey literature tends to follow the same pattern. It was difficult to find sources that specifically discuss data engineering or data science from the perspective of a small scale context. Much of literature that focuses on big data and scale does have parts that can be extracted to inspire solutions for small scale practices. However, this approach entails a lot of time spent reading and practically renders oneself doing the research and analysis that we request be done by research professionals to fill the noted gap in literature. It would be interesting to see this done in a similar fashion to how the RDA performed their 2020 best practices study on data versioning [27].

The study made by the RDA on best practices of data versioning was quite comprehensive and the literature on the subject, including data provenance as a subfield, is rather promising. However, finding literature on the subject of data pipeline testing, at any scale, was especially difficult. Zhang et al. [48], chapter 7.1, is the most extensive source on the subject that was found which gather findings from multiple studies on the subject. It primarily focus on machine learning aspects but in doing so it addresses data pipelines when discussing bug detection in data.

On the subject of data platforms, data mesh stood out in this literature study as one of the more promising works. Given that it has not been published in any academic journal, it is published on a reputable grey source via Martin Fowlers blog who has received much recognition for his work within computer science and agile software engineering. The author of data mesh, Zhamak Dehghani, is also recognized and a prevalent speaker at large tech conferences [52] .

Data mesh is especially interesting in that it takes a very holistic approach to what a data platform should be and how it should go about solving the problems it is intended to. This perspective is useful as it discusses problems not only tied to technical implementation details that often wind up being on the subject of distributed computing, which is a problem with much of the literature on the subject. But it discusses many things that are applicable to contexts of any size as it re-aims the primary focus away from technical implementation to structuring domains, ownership and collaboration. Put shortly, it specifically focuses on how to drive liveness and value creation of a data platform (RQ 1.1) by addressing people-centric issues.

Chapter 4
Case Study of Sentian

To shed light on challenges related to the application of data engineering for machine learning pipelines in a small scale context, an interview study and a practical implementation was conducted at Sentian. The interview study was performed through interviews of data scientists and a data engineer at Sentian. The implementation consisted of designing and developing a data pipeline, to gain some first hand experience of the challenges related to working with data engineering. By investigating the challenges from more than one angle, i.e. both practical and theoretical, the aim was to gain a more accurate understanding of the core of the challenges.

4.1 About Sentian

Sentian is a small, less than 20 employees, specialized machine learning company which focuses on machine learning solutions for industry. The majority of their organization is made up of data scientists with two data engineers that support these both through data pipeline implementations but also by evaluating tools or building them if there are none that suits Sentians needs.

4.2 Interview Study

The purpose of the interview study was mainly to understand the data engineering challenges that Sentian is facing and how far along they are in implementing solutions to these.

4.2.1 Method

A qualitative investigation of Sentian's current data engineering practices related to their machine learning projects was performed via semi-structured interviews [53] with one data engineer and three data scientists. The data engineer was chosen as he has a lot of knowledge about data engineering in general and because he works on developing data pipelines, tooling and infrastructure. He could probably have answered most of the data engineering related questions about Sentian, as he is the focal point for data engineering issues at the company. However, including multiple perspectives allowed for a richer understanding and aids in triangulating the core of data engineering challenges [7]. The three data scientists provide a user perspective of the tooling and infrastructure while the data engineer has more of a developer perspective. Data scientist number one is a senior full-stack data scientist, i.e. has experience of constructing whole machine learning pipelines himself. The second data scientist is a junior full-stack data scientist in this regard. The third data scientist could best be described as a deep learning specialist who focuses solely on implementing and trying out architectures in a research and development context.

The four interviewees were interviewed with interview protocol A regarding the type and size of data that they were involved in working with in customer projects and what challenges they experienced working on those datasets. The third data scientist however was not involved in customer projects and could not answer these questions and as such that interview was carried on as an open discussion on other data science related subjects. The interview with the third data scientist should be viewed as a learning opportunity about the data science domain for us and not as a result of the report which focuses on data engineering. The interviews with the data engineer and the junior data scientist followed the protocol very closely and gave good information on those subjects. Therefore we spent less time on the prepared questions, although they were asked, and dedicated more time to having the senior data scientist speak more freely about challenges related to his experiences.

The data engineer was additionally interviewed, using the semi-structured method on the subject of their ongoing development of a data platform and challenges related to it in protocol B. Unstructured interviews were held with the data engineer throughout the course of the thesis to further enrich the context and challenges through open discussions [53].

4.2.2 Results

The results of the interviews are structured in two parts, the results from protocol A and then the results from protocol B. As the interview with the third data scientist was more or less aborted and continued as an open discussion on other data science related subjects it is excluded from the results.

Results based on the semi-structured interviews conducted with interview protocol A. Sentian has rather few projects overall at any one time and have not had more than one concurrent project within any one domain so far. The datasets that they work with vary from very small, less than 100MB, to quite large but not huge, less than one TB. Furthermore, the velocity of data in production has not been demanding enough that it has posed problems that could not be solved by parallelization of computations on a single computer. In other words they do not work with what one would define as big data [54].

They often work in small teams of less than four people, where one is a data engineer or even without a data engineer at all. When no data engineer is available, one of the more senior data scientists will take on the data engineering tasks as well. It happens that entire projects are ru by a single senior data scientist.

The three interview subjects who work on customer projects all mentioned that data semantics and documentation often were lacking in the beginning of projects. They often start out in the position of having to figure out what the data represents and what certain values are measuring by themselves. It can be difficult to get access to the right person within the customer's company that can explain the data. Throughout the project, multiple deliveries of data may be delivered from the customer where the data was extracted by different people and as such delivered to Sentian in different file formats and even different data structures.

The data engineer pointed out that there may be many reasons for why it is difficult to obtain good and well documented data. The main one being that machine learning is not well understood by most people, this is also the case with Sentians customers and the management of the customers who sign off on the project. It is often understood as a blackbox solution where they provide data and Sentian will build the machine learning model which will consume the data and generate inferences for the customer. In this context the customer expects an overall solution with minimum engagement from their side. Specifying data requirements would require involvement of expensive technical expertise from the customers side which may not be available or in very short supply. Stipulating that this must be provided by the customer would make selling a machine learning solution to the customer much more difficult. At the same time, allowing a Sentian data engineer direct access would bring with it a host of problems regarding security and could risk putting a potential customer off.

One of the data scientists further points out that even well functioning organizations suffer from loss of information, due to human nature. The situations where machine learning tends to be applied are also often very complex and there is no single person who can understand them and all of the data that is generated in its processes. It should be expected that there will be investigative work and that one will have to work iteratively with the customer to find the right people who can answer specific questions. Therefore it is essential to build a good relation and to build trust with the customer, as their cooperation is essential for being able to gather the data and metadata that is needed to design a machine learning pipeline.

Sentian does not currently use a system for data versioning and sharing metadata that they gather throughout a project. They do have a tool that identifies and reuses data generated from specific transformations instead of duplicating the computations on the same data, which guarantees that a user receives exactly the same end result. It was expressed by all three of the interviewees that work on customer projects that a system for data versioning and metadata sharing would further facilitate cooperation and would make revisiting projects easier.

Results based on the semi-structured interview conducted with protocol B, and the unstructured interviews, with the data engineer.
When it comes to the technical implementation of machine learning pipelines, of which a data pipeline is an integral part, much of it is reliant on the availability of tools. The data engineer expressed that many of the tool-stacks available for data pipelines, are specifically built to solve problems of scale. They are aimed towards data engineers that work with big data and although they greatly simplify the processes, they require skills that data scientists often do not possess and that takes time to learn. This is a problem as data scientists at Sentian often construct their own data pipelines, it is a part of understanding the data and cleaning it, at least in the experimental phase. Outsourcing this process every time to the data engineer would create double work as he, too, would have to spend time understanding the data to be able to clean and process it properly. In their small scale context where resources are scarce this solution is considered too expensive.

To avoid vendor lockdown, Sentian made a business decision to use a cloud-agnostic approach to implement and deploy their machine learning pipelines. They decided to go with containerization and decided on using Docker and Kubernetes for this.

To simplify data pipeline development, Sentian intended to implement a tool similar to the data infrastructure as a platform, presented in the data mesh design (although they were not aware of data mesh at the time), where the underlying complexity is hidden from the user, in this case a data scientist. The implementation turned out to be more challenging than expected and the available literature that they found tended to focus on the challenges of implementing tools that solve problems related to scale using distributed computing practices. Distributed computing did not suit Sentians small scale context as it would impose an unnecessary distribution related computational overhead. They have been investigating how improvements could be made concerning data storage, specifically through a data platform as a potential way of structuring and sharing data internally. The data engineer noted that implementing ownership and delegating responsibility of data could be useful in adding structure as Sentians organisation grows.

A custom data pipeline tool proved to be difficult to integrate with the available machine learning workflow orchestration tools. There were two tools that Sentian were primarily interested in, ML-flow[55] and Kubeflow[56]. It was the belief of the data engineer that ML-flow focused too much on being an easy to use databricks platform, which in turn is built on Spark, and that it was over-engineered and inflexible. It did not allow for easy integration with custom in-house tools, on which Sentian is dependent, and had limited support for docker containerized solutions. ML-flow is no longer in beta but the data engineer still deems it an unfit solution for use in production at Sentian for the previously mentioned reasons. The second tool, Kubeflow, does aim to provide flexibility and is intended specifically for kubernetes. It matched many of Sentians criteria, however, it too, suffered from immaturity problems when it was evaluated at Sentian, then in v.0.3. Even though Kubeflow have made advancements in the last year, progressing their master branch to its current state, v.1.0.2, their pipeline application in this version is still in beta [56].

Sentian ended up going with a third alternative, ARGO[57], which is built for orchestrating parallel jobs on kubernetes infrastructures. It offers less of the functionality that Sentian is interested in than what ML-flow and Kubeflow does. However, ARGO is more mature and it is the experience of the data engineer so far that as it tries to do less, it does it better.

4.2.3 Analysis of the Interviews

Comparing the results of the interview study to what we initially found when searching the academic literature, Sentian is indeed in a situation where there is a gap between their needs and what kind of challenges and solutions are discussed in the data engineering domain.

Sentian's experience of problems related to receiving data from customers is not trivial to solve. The problem, which stems from the people who collect the data not being the same ones who have to clean and process it, is not unique to Sentian [58] [17]. It introduces significant obstacles such as understanding the semantics, what features are present in the data, and what the data's inherent properties are. Receiving dirty data from the customer is ubiquitous since they are not focused on remediating data related issues themselves as they expect an overall solution from Sentian which includes this. Thus it becomes the responsibility of the machine learning pipeline to make sure that incoming data is correct through data pipeline testing. This could be implemented throughout the machine learning pipeline by incorporating data validation schemas. Hopefully, as machine learning moves through the Gartner hype cycle [59] some of the problems related to customers not understanding the limitations of machine learning and its dependence on good data might be mitigated. A more widespread understanding of machine learning in industry would make it easier to motivate the customers to get more involved in the process of gathering and delivering well structured, documented and consistently formatted data.

As a small scale enterprise, it is expensive for Sentian to develop comprehensive custom implementations of machine learning frameworks in-house. The cloud-agnostic machine learning frameworks that are available, and their cloud-agnostic data pipeline framework counterparts that do not rely on distributed computing, are still immature. However, things are moving fast in this area, and tools could conceivably become more reliable in a very near future. On the subject of data platforms and improving reproducibility and reducing duplicate work, one could come a long way in a small context with simple techniques by implementing one's own light version of a data mesh. Furthermore, the design concepts of the data mesh could inspire Sentian in how to provide more structure employing clear ownership of data viewing it as a product, and how to support collaboration through cross-functional team composition. Sentian could also look into adapting or building upon one of the existing data management platforms that have been open sourced by big tech companies [4]. Even though these are not built with Sentians small context in mind they might still work as a solution to share data and metadata between data scientists.

4.3 Data Pipeline Implementation

A data pipeline was implemented to gain an understanding of challenges related to data pipeline development. It was the opinion of the supervisor at Sentian that implementation experience is essential to one's understanding of the ad hoc nature of data engineering solutions and that this understanding is needed to be able to analyze the field and its challenges effectively.

In Figure 4.1, one can see a visual outline of the data pipeline that was implemented. The wine reviews dataset [54], which is described below in section 4.3.1, was downloaded and stored locally on the machine where the data pipeline was run. Receiving a dataset to work on locally for modeling and training before setting up the production pipeline is usually how projects are started at Sentian. The data pipeline that was implemented would be classified as a so-called ELT pipeline, as the data is Extracted (downloaded from Kaggle), Loaded (persistently stored on disc) and Transformed (through data processing and enrichment) for a specific purpose. In this specific case, the data was transformed such that it could be used for a machine learning model to predict the price of wine.

As one can see in Figure 4.1, the data pipeline building process followed a standard execution of investigating, cleaning and processing. This way of working is very common and was the overall approach found in all three of the interviews with subjects who work on customer-facing projects.

Throughout this chapter we cover the work of designing and implementing this data pipeline. We end the chapter with my own analysis, where we generalize and describe the challenges that we faced and apply what we learned from the literature and interview studies

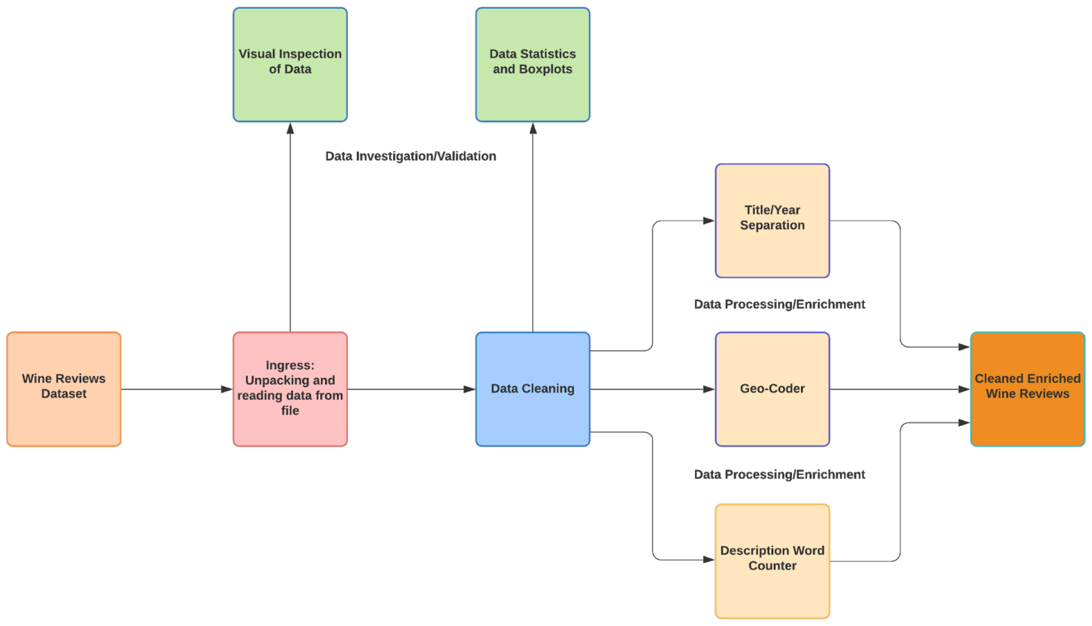

Visual Inspection of Data
Data Statistics and Boxplots
Data Investigation/Validation
Title/Year Separation
Data Processing/Enrichment
Wine Reviews Dataset
Ingress: Unpacking and reading data from file
Data Cleaning
Geo-Coder
Cleaned Enriched Wine Reviews
Data Processing/Enrichment
Description Word Counter

Figure 4.1: data pipeline overview

to the context. Since the purpose is to understand challenges related to constructing and working with data pipelines we do not focus on minute technical details. The code implementation can be found in the appendix.

4.3.1 Method

Building the data pipeline itself was performed in conjunction with performing the following tasks: (1) investigating and understanding the data (2) validating the data, i.e. making sure that the data behaves as expected (3) cleaning the data and (4) engineering features that are fit for model serving. In the following section the activities of these tasks will be covered with the engineering of features being split up into individual parts for each feature that was engineered. The choice to perform these specific tasks was based on practices described in [3] and in the interviews with protocol A.

Python was chosen as the programming language because of the availability of mature data science libraries. The data pipeline was implemented iteratively following agile principles. Functionality was implemented in line with a micro-service architecture to achieve loose coupling of components. Daily standups were held with the Sentian supervisor and a Kanban board was used to organize task supervision.

Choosing a Dataset

We chose to look for non-mathematical problems for our data pipeline implementation since the purpose behind the implementation was to work with data challenges and not to, for example, use mathematical modeling to classify the data. When choosing a data set to implement a data pipeline for, there were some aspects that were important for the scope of this thesis. The main one being limited time and resources, another one being my limited previous experience of data scrubbing.

The wine reviews dataset that we chose was rather clean and well structured compared to the real world examples that have been experienced at Sentian. Each review also contained numeric information about a wine through price and score along with other non numeric information about a wine. Predicting a numerical value based on both numerical and non-numerical values is a broad problem which serves learning well and as such we decided on preparing the data for price inference. There were still, however, some processing that needed to be done such as data cleaning, transformation and enrichment of the data further via the features that are already present. The challenge of figuring out the features and inherent tendencies of the data that can be used in modelling was also present.

The wine reviews data set covered many of the common pre-processing steps of a real world dataset with a limited amount of unstructured chaos. Chaotic and unstructured data can otherwise require specific ad hoc solutions which may consume a lot of time without generalizing well for the understanding of constructing data pipelines.

4.3.2 Data Investigation, Validation and Cleaning

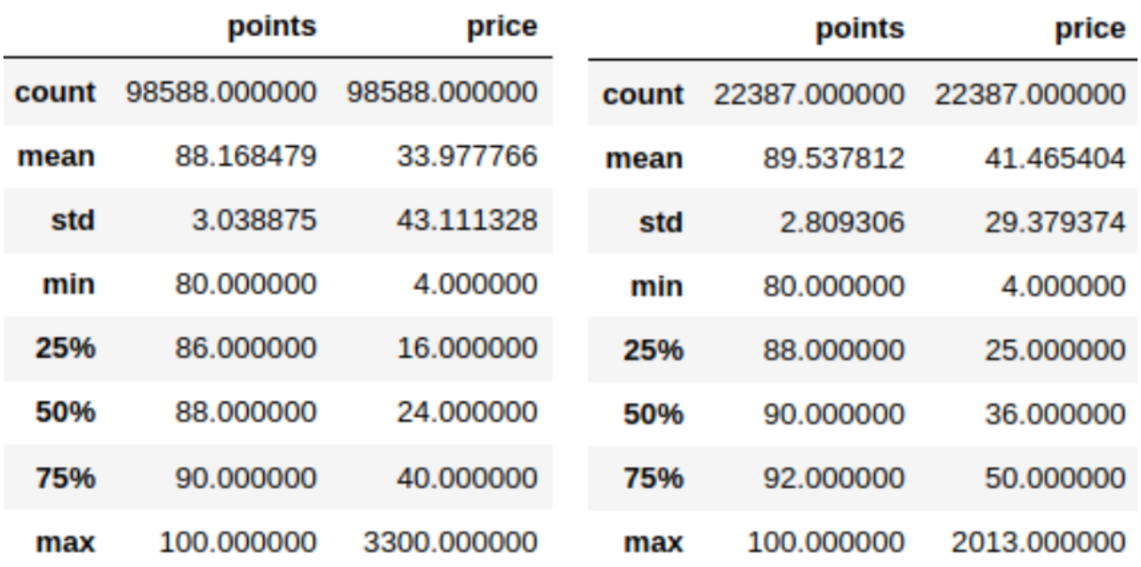
The first step in working with the data set was to understand the structure and broad properties of the data itself. This was foremostly done through simple exploratory data analysis. Much of the work lied in manually looking at the data, what features were present and how and if there are any internal relations. Locating where missing values are prevalent and whether they should be remedied through interpolation or other heuristic solutions was done early. It was done to eliminate unusable data before further processing.

An example of this would be rows where the price was missing, which unables using any of the data in the row to train supervised machine learning models on predicting price, as it has no label. This label could potentially be collected manually but one could not guarantee that the price had not changed from when the review was made. One could still use an unsupervised model for rows with missing price data that clusters them with other wines that do have price data. But as rows with missing price data were rare there would likely be little to gain from using an ensemble of both a supervised and an unsupervised model for price inference.

With the wine reviews dataset, missing values were especially common in the designation, price and region(1 2) columns. The number of rows containing missing values in any of the columns except for price and points (score) was 17 percent. To decide whether or not to drop the rows that contain missing values in any columns except for price and points, their price mean and standard deviations were analyzed.

The price distribution diverged from the set of rows not containing any missing values in the previously mentioned columns. The distribution skew from removing these rows should be kept in consideration when choosing what model to use for inference, it might be prudent to choose one that is more robust to small differences and not prone to overfitting. In Figure 4.2 we can see the difference in points and price expressed with means, standard deviation and quartiles with the count describing the number of rows from the wine reviews dataset that was used to produce the table.

Manually looking at the data inspired some possible relationships that could be explored further by visualization through the pyplot library. Since the data was sampled and authored



	points	price
count	98588.000000	98588.000000
mean	88.168479	33.977766
std	3.038875	43.111328
min	80.000000	4.000000
25%	86.000000	16.000000
50%	88.000000	24.000000
75%	90.000000	40.000000
max	100.000000	3300.000000

	points	price
count	22387.000000	22387.000000
mean	89.537812	41.465404
std	2.809306	29.379374
min	80.000000	4.000000
25%	88.000000	25.000000
50%	90.000000	36.000000
75%	92.000000	50.000000
max	100.000000	2013.000000

Figure 4.2: points and price distribution of rows with missing values (left) vs rows without missing values(right)

by humans, human bias was a good place to start.

Human bias might have an effect on the skew of the data, specifically the price and scores of a wine. A large part of the experience when trying a wine may come from circumstantial things other than specific taste attributes, such as visual cues [60]. Therefore we considered that knowledge of origin may affect the perception of a wine. The question was posed whether the experience of a wine could be influenced by one's worldview which may differ between cultures and nationalities. If it does, then that raises the question of how it affects the perception of a wine.

Most of the reviewers in the data set are from America as it is an american magazine, although some authors are based abroad. A significant part of the reviews in the dataset, 47%, are authored by the top five contributors out of a total of 19 authors. All of these top authors are american, although one of them is based in France and one in Italy. This seems to be a trend for all of the 19 authors, however it is hard to verify current residence through online sources.

Visualization of score distribution by the different authors through box plots showed that, mostly, they had similar means and spreads in their scoring of the wines they had reviewed. The same comparison yielded similar results when producing boxplots for the different types of wine and different regions as well. The difference laid primarily in the upper extremes, some authors were more hesitant than others to give exceptionally high scores. This may indicate that there is little bias in the data but that the range that they score their wines within might differ. Some might call this bias but we believe the term upper bound might be more descriptive since it is not a general skew for the whole range of values.

4.3.3 Feature Extraction of Year

Following up on the human bias investigation, one could imagine that year would impact any inference of the price of a wine. As it was, however, year was not present in the data set in its raw form but could be extracted from most of the titles. Consequently, we performed a feature expansion on the title column to extract all rows containing a title year from which the title column was split into two new columns, one containing the name and one containing the year.

4.3.4 Feature Extraction of Geographical Location

The geographical location of a wines origin was extracted through querying the Google Maps API for coordinates for (region_1, province, country). We chose not to delve deeper into origin, specifically through winery, since that might introduce noise into the data. Wineries often have multiple addresses in an area and choosing which one to represent the geographical origin was deemed to not likely contribute in predicting score or price of the wine. What is interesting from this perspective is whether a geographic region tends to produce a certain quality of wine or of a certain price range. The name of the winery could theoretically still be used to further extract information about price range and quality, assuming that it occurs often enough for an algorithm to notice the pattern.

4.3.5 Feature Encoding of Descriptions

As the descriptions all describe the same type of object, a wine, there were a lot of adjectives and categorization terms present. From a theoretical micro economic perspective, there should be certain properties of a wine that factor into its pricing. The hypothesis was that these properties could be correlated with certain words used to describe a wine and that the presence of certain words could be used by a model to glean information about the price of the wine.

To implement this we traversed the descriptions of the dataset to gather the “N” most commonly used unique words apart from stopwords and symbols. Each description of a wine was then checked against this list to count the occurrences of these common words. The words may then be encoded using a simple library implementation of the encoding that fits the specific model the best, as there are many to choose from that suit different situations.

4.3.6 Analysis of the Data Pipeline Implementation

As previously stated, the WR dataset is very clean, especially so compared to the experiences at Sentian. Despite this, it became clear during the investigation phase that deciding how to perform the most basic cleaning, such as removing missing values, potentially could significantly skew the distributions of features in the dataset. One has to have an understanding of the data, but also of the context of its consumption, to be able to implement even basic functionality of a data pipeline. To clarify this with an example, we can imagine that the use case for the WR dataset is expanded to a context where it is likely that other people would consume it with other purposes in mind. Then one should consider that many of the decisions that we made in processing the data, although they were motivated for our specific application of predicting price, could counteract, say predicting origin. In that case it could prove to be unhelpful to remove entire data rows if both (province, region_1) are present but the country is missing. It could very well be worth the effort to instead fill in the country as it would likely be easy to recover from the Google Geocoding API.

In our implementation we were spared from many of the more difficult challenges which often occur in putting a data pipeline in a production environment where additional demands are incurred. An example of such a challenge could be that if the data is streamed to the data pipeline and consumed in small batches for inference, then choosing how to process the data is no longer trivial. The model's capacity to make an inference is affected negatively by missing values. Increasing the time window for the batching so that more data can arrive may enable one to interpolate the missing values which could alleviate the performance drop from missing values. However, stating the logic for all possible cases of this is difficult and the consequences of getting it wrong can be detrimental to inference performance. There might also be time constraints from a required frequency of inferences which limits the time window for batching.

4.3.7 Future Work

In regards to enrichment, weather and climate has a documented impact on wine [61], therefore it was hypothesized that this information could be useful for a multitude of analyses. There was an idea that the year and geo location of wines could be used to gather weather data. This could be done through [62], although it is not a free option. The main limitation and the reason for not implementing this was shortage of time. There are surely many other interesting enrichments and transformations that can be made that have the potential to be useful. However, even though we do like wine (and data), endlessly improving the potential of the wine reviews dataset was not the purpose behind the implementation of this specific data pipeline.

It would be interesting to see how available tools could be applied to deploy the data pipeline, either as it was implemented or according to our design that is presented in the analysis, as a part of a machine learning pipeline in a production environment. The wine reviews dataset could be split up into multiple batches which could be fed into the machine learning pipeline, adjusting the data pipeline and the machine learning model after every new batch. One could also scrape new wine reviews from the winemag.com website and make inferences on these. It would be interesting to see a technical implementation that persists and versions the information throughout the machine learning pipeline for reproducibility. It would also be interesting to see a technical implementation of how such information could be gathered and shared through a data platform with the discussed cataloguing system functionality.

Chapter 5
Mapping to Agile practices

In analysing the literature and the case study it became evident that there is an overlap in problems addressed by the big data community and by agile software development. This spawned the second set of research questions, RQ2.1 and RQ2.2 which are presented below, in an attempt to remedy part of the gap discovered in literature.

RQ2.1: "Can the principles behind the agile manifesto be mapped to the architecture of current data platforms to extract insights that may be applied to a small scale context."

RQ2.2: "Are there practices in small scale data engineering that map well to the principles of the agile manifesto?"

When we were developing the data pipeline implementation we did so using agile practices. In this chapter we analyse one of the state of the art data platform architectures, the data mesh, through a mapping of its concepts to agile principles. This provides an agile perspective of designing a data platform which could facilitate analysis and discussions of a custom data platform implementation for a small scale context.

5.1 Method

We chose to perform the mapping of each of the twelve principles behind the agile manifesto to the two criterias (a, b) below.

(a) Mapping of agile principle in question to relevant concepts in data mesh (RQ2.1).

(b) Example of small scale data engineering practise that relate to agile principle in question and how it does so (RQ2.2).

The reason for choosing the data mesh architecture to represent the state of the art of data platforms in this mapping is because of its holistic approach. It does not only focus on the technical implementation but discusses value creation, data ownership and organization structure around the data platform as well. This can help answer not only RQ2.1 but RQ1.2 and RQ2.2 as well.

5.1.1 Choosing a Definition to Represent Agile

There are many writers on the topic of agile practices. Agile has grown to become the de facto approach for software development [63]. Amongst the authors, or cosigners if you would, of the agile manifesto you will find both the authors of extreme programming and the authors of SCRUM amongst many other well known figures within software engineering [8]. It is safe to say that the collective knowledge of agile development amongst the authors of the agile manifesto is comprehensive and that the principles behind the agile manifesto ought to be a good representation of agile principles.

5.2 Mapping of Agile Principles

Below we present our mapping of our criteria to the twelve principles behind the agile manifest. If one criteria is split into multiple points then i) and ii) is used to enumerate the points.

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

   (a) Data mesh specifies using Service Level Objectives (SLO) to ensure satisfaction amongst data product consumers which could state timeliness of new delivery of data and error rates amongst the data.

   (b) Expedite the delivery of a data product by automating the data pipeline and uploading the data product to consumers via a data platform directly as the raw data is made available.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

   (a) Data mesh allows the data lake practise of using ELT in its source domains, although its consumer domains do host processed and aggregated data. These data source domains provide the consumers of raw data products with the possibilities of performing unforeseen transformations and aggregations of data.

   (b) i. Persisting raw data at ingestion when designing data pipelines allows for unforeseen transformations and aggregations of data.

       ii. Micro service architecture decouples functionality in an otherwise highly coupled DAG such as a data pipeline.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

   (a) The data mesh approach of providing data infrastructure as a platform which is developed with the goal to shorten lead times for creating new data products supports this.

   (b) Micro service architecture and decoupling allows for constructing a data pipeline with the most basic data processing to start with. This in turn allows a data scientist to start modeling and training with one version of the data sooner. As the data pipeline functionality is developed further, the training and serving data will improve and allow for better accuracy of the model.

4. Business people and developers must work together daily throughout the project.
   (a) Data mesh adoption of domain driven design, and its management structure through domain teams to reduce siloing and promote cross functional teams.
   (b) Data is widely used to inform business decisions, as such the data engineer needs to have a communication line with business so that they are informed of available information. The end goal of the work that a data engineer performs is to support business, including them as they hold a holistic view is important.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
   (a) The role of the data product owner of domains in the data mesh enables these motivated individuals to take initiative to drive value creation.
   (b) This principle is deemed as general advice that does not change between software and data engineering contexts.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
   (a) Is enabled by data mesh by cross functional domain teams.
   (b) This principle is deemed as general advice that does not change between software and data engineering contexts.
7. Working software is the primary measure of progress.
   (a) Tracking usage of data products and how well the SLOs are met has a similar focus in data mesh.
   (b) Same as (a).
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
   (a) Data mesh consumer domains make aggregations and processed data available. A data scientist may need data that is similar to a raw data refinement that has already been performed. In this case that the data scientists use another processed dataset, it has to be well described through metadata and provide data provenance.
   (b) Save and make available any existing refined data, doing so will allow reuse by other data scientists and make it possible to continue working at a constant pace without slowing down having to wait for development of a data pipeline that will refine the raw data.
9. Continuous attention to technical excellence and good design enhances agility.

   (a) The concept of data governance policy for the data platform in data mesh guides good design of data products. Stating this through policy imbues it in work practices to be used continuously.

   (b) In a small scale context, as we mention in the interview results of interview protocol A, a data scientist might have to make their own data pipelines. If the data platform is developed with agile principles in mind, then it is likely to promote agile practices in using it. An example of this is my own application of literature regarding versioning in designing persistence of data in my data pipeline implementation analysis. If an upload of a data product to the data platform requires both an original “ground truth” file such as a .csv as well as one or more serialized object files such as a .pickle, then the data scientist will be influenced in their design of the data pipeline to create both of those entities as they will be required to upload ones work. Not only uploading processed data but raw data is ,as discussed in the context of ELT, in line with agile principle 2.

10. Simplicity - the art of maximizing the amount of work not done - is essential.

   (a) i. Data mesh source domains allow for ELT, which postpones the transformations until the specifically transformed data is needed. Combining source domains with consumer domains which provide refined data products for other users with similar needs minimizing.

      ii. Data infrastructure as a platform also maximizes work not done by providing reusable functionality and striving for automation while hiding underlying complexity.

   (b) i. Gathering domain expertise as a data engineer, or hiring it, saves time as there may be industry standard semantics.

      ii. A domain expert may have experience of common correlation and causation relations of data and knowledge of who to ask regarding specific system processes that generated poorly documented data.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

   (a) The domain teams of data mesh are self organizing. Data pipeline architecture is considered a domain-level concern. Although there are standards and some data governance through policy that must be adhered to.

   (b) Trust the data engineers to evaluate and make their own architecture and design decisions for projects. Entrust a person such as a project leader with staving off the pressures from top management to use specific hyped architectures or tools.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

   (a) Not specifically brought up in the data mesh article.

   (b) All users and contributors of a data platform should meet regularly to discuss improvements not only of data platform functionality but governance policy as well.

5.3 Data Pipeline Implementation Revisited

After having performed the agile mapping we revisited our data pipeline implementation with both a new agile perspective and a new understanding of how to apply the findings from the literature study. We used our newly gained knowledge of what parts of agile and literature could be applied in constructing a data pipeline in conjunction with the existence of a data platform. This section serves to exemplify what we, based on our findings, consider to be best practise. It is our attempt at bridging the gap we found in literature on the subject of best practices for small scale data engineering.

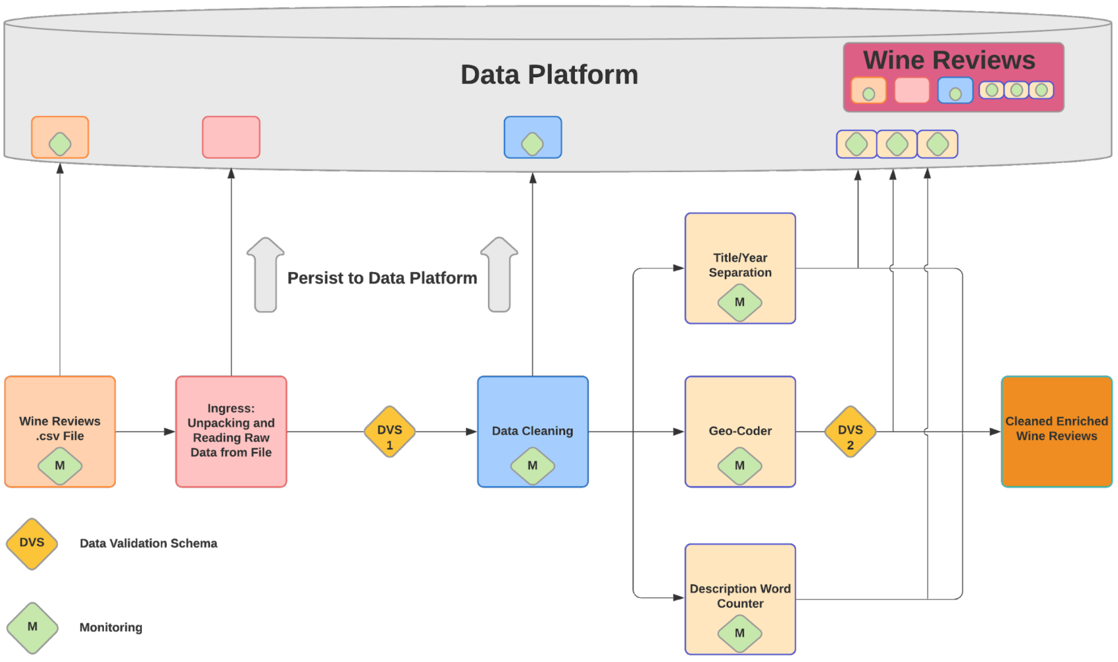


Figure 5.1: Our suggestion of the data pipeline, had it been implemented with our agile perspective and best practices.

Figure 5.1 may serve as an example of the basics of what a batch oriented data pipeline may look like with testing and monitoring according to what we found in the literature study. It may also exemplify how to apply our agile mapping and our literature study findings on data versioning and provenance through answering the questions (1) what data should be persisted, (2) how should it be persisted and (3) how should it be catalogued. It is our own view that these three questions should be answered by a data platform solution to drive liveness through discoverability and reproducibility of data.

As to not clutter Figure 5.1, data checks, some data flows and the file validation schema has been left out. The data validation schema would be implemented as a filter which does not allow data that does not meet its criteria to pass through and instead discards it. As data is discarded at a validation schema, it should be logged in a database which rows were discarded and which criteria were not met. The dataflow of this logging is not included in the figure. As we describe in the background chapter on data pipelines, simple checks on data should be included where data is processed to catch wrongful data which could be caused by software errors. Additionally the monitoring has been included as part of the process that it monitors instead of depicting it as an individual process which consumes and outputs its own dataflow. The data flow from Google Geocoding API has also not been included.

5.3.1 Data Validation Schemas and Data Monitoring

The parts of the data pipeline design that have been implemented are explained in section 4.3 on our data pipeline implementation, therefore our focus in this design on the parts of it that are new, i.e. the data validation schemas and the data monitoring. As mentioned in the background chapter 2, data cleaning can be performed, or tightly coupled, with a data validation schema. In our proposed design they are separated as DVS1 is intended to work as a filter which discards data that does not meet specific criteria. The data cleaning on the other hand includes processes where data is not only discarded but remedied if possible. An example of this is the previously mentioned example in the beginning of 4.3.6 where (province, region_1) was present but the country of origin was missing. As stated this could likely be remedied in most cases by retrieving the country by querying the Google Geocoding API with (province, region_1). The second data validation schema, DVS2, would also work as a filter and is included as the coordinates have newly entered the data pipeline. Both validation schemas would, as previously stated, log what data is discarded and why in a database.

The monitoring of the different processes collects metadata on specific intermediate states of each batch run. Some examples of what could be tracked are presented below.

- **Monitoring of file (the state before ingestion):** file format (if multiple file formats are allowed), file size, time of creation etc.
- **Monitoring of data cleaning:** distributions, means and standard deviation of price and score before normalization, what percentage of rows have been complemented through remedying missing country etc.
- **Monitoring of Geo-coder:** how many calls are made to Google Geocoding API and how many are returned. If geographical zones are defined, what percentage of the wines are from europe, america etc.

How effective the monitoring and data validation schemas are could also be subject to testing. It could be done by having someone, who preferably did not work on the data pipeline, perhaps a data scientist who will use it, introduce faulty values in the dataset that would disturb a potential model. This new dataset containing faulty values could be passed through the data pipeline in an exercise to see how well the monitoring catches for example a distribution skew or unrealistic data. This could also serve to test the efficiency of the data validation schemas. It could also be done the other way around, similar to test driven development, where the data engineer designs the monitoring and the data validation schemas to catch potentially wrongful data that the data scientist believes could be disruptive.

5.3.2 Persistence to Data Platform and Cataloguing

For the sake of choosing a ground truth, the original data file itself will be saved. For practical purposes all other data will be saved as serialized objects, in our case as a .pickle file as we use python, which is significantly faster to read than a .csv file. The enrichments will be saved individually as this enables higher granularity for a potential user to only retrieve exactly the data enrichment that they want. At the states where the data was being monitored, the metadata from the monitoring should be saved along with the data that it monitored. Parts of the metadata from the monitoring can be tagged to be included in the catalogue description of datasets and collections to help a browsing data scientist determine if they want to retrieve it.

A data scientist working on another project might be interested in using a dataset similar to theirs, perhaps to diversify their own dataset or test out their model on a different dataset. Experimenting with open datasets that are related to a problem to prepare for a potential project has been used in practise at Sentian. As described by [19][20], enriching a dataset by diversifying it with external data is a helpful and common practise among data scientists, especially within deep learning where large amounts of data is needed to train models. As machine learning is expanding within deep learning, this way of working may become more relevant to Sentian in the future. Therefore implementing cataloguing and versioning on top of a potential data platform could be worthwhile, we use our own data pipeline example in Figure 5.1 to exemplify how it could be designed.

For the cataloguing system to provide provenance and dataset discoverability, it has to have some way of versioning and structuring the datasets that are contingent to the same data source. In our case we would like to be able to access all of the data that was persisted to the data platform throughout the wine reviews data pipeline design above easily. Applying the data versioning framework of RDA [27], which is described in the literature study, we can explore one way of implementing the cataloging structure through our data pipeline design and the wine reviews dataset as an example.

- **Work,** the tasting experience of the authors.
- **Expression,** the review and score along with info about the wine origin presented by wine reviews magazine.
- **Manifestation,** the scraped wine reviews dataset that is hosted on kaggle, with the specific data schema that the person who scraped the reviews defined.
- **Item,** the wine reviews .csv file

In our specific example we are somewhat limited in which of the levels we are able to affect, since we enter this picture at the manifestation level and have no control over the work, i.e. the sampling process of the data. In fact, this is often the situation that Sentian themselves experience. The data that they are provided with is often extracted from the customers' existing database, which if done inconsistently leads to Sentian having to deal with different manifestations of data (different file types and data formats).

Specific catalogue structure can be done in many different ways. Let's imagine that it is structured by domain and that food and beverages is its own domain. Here we find different works that are related to the domain such as our specific set of wine tastings (case specific equivalent to the observation/data gathering process). The arrows represent relations which should be included in the metadata on each object to provide provenance. The three abstract levels of manifestation, expression and work are mainly serve the purpose of providing provenance. On the item level, each collection of files and each file within a collection should be versioned. Within a collection of files any dependencies between individual files must be included in the metadata. An example of this would be the geo-coordinate enrichment data which should be documented was retrieved from Google Geocoding API via (region_1, provenance, country) from the cleaned wine reviews file.

The relation between the wine reviews item and the enriched wine reviews expression is drawn differently from the other relations simply to draw attention to it crossing multiple abstraction levels, which is an important use case. In our specific example we have changed the contents of, and enriched, the wine reviews dataset with external information which leads us to having created a new expression of the wine tastings. Even though we have created an expression, for provenance reasons, the enriched wine reviews dataset is related to an item the wine reviews .csv file that we retrieved from Kaggle. Furthermore, the externally collected data on geo-coordinates should also be included in the provenance information on the expression level as its origin is the same regardless of how we manifest the information.

There are two main reasons for not persisting an enriched dataset as one object, but as separate files: (1) separating the modified original dataset from the multiple individual enrichments reduces redundancy (2) it allows for easier implementation of high granularity provenance as specific enrichments can be traced directly to its specific creator and original expression if come across as part of some other collection. A potential user can still retrieve the same complete enriched representation of a dataset through the expressions' collection. Being able to use a processed dataset representation, although it gives up some control, does save time. One can still retrieve the individual components of the collection if it is important to maintain control since the enrichments are persisted as individual files.

5.4 Analysis

In reflecting on the mapping and discussions of possible angles and perspectives for where to apply specific agile principles it dawned on us that although the mapping did lead to formulations that can be useful to follow as good practise, the discussions themselves served a far greater purpose. Using the principles behind the agile manifesto served as a guiding document not only for what to discuss but also how. It provides a common language and a basis for what level to discuss topics that are often related to technically advanced solutions in regards to which the engaged participants may come from different backgrounds and have different perspectives and vocabularies.

Evans [37], the author of domain driven design which is applied in data mesh, discusses the need for a shared language, referred to in the book as the ubiquitous language, between participants from different backgrounds to effectively define and talk about problems specific to their domain. We argue based on the findings of this thesis that the principles served this purpose and that it would be interesting to explore this further. Specifically through including more potential stakeholders of the data engineering context who are from other backgrounds, such as data product owners and data scientists.

There are surely more mappings and similarities between agile and both data mesh and data engineering in small scale contexts that can be made. We believe that the mapping that was made indicates that first of all it can be done, but also that the mapping that was made can provide a useful perspective. The mapping can be read to extract best practices from big data literature to a small scale data engineering context where it may help navigate solutions and discussions that lie in the gap.

Chapter 6
Discussion

In this chapter we make an attempt to answer the RQs: 1.1, 1.2, 2.1 and 2.2 by tying together the results from the three different phases of this thesis. We also shed light on any tangent findings that relate to data engineering for machine learning in a small scale context.

> RQ1.1: *"What is the state of the art of data platform architecture and how do one drive liveness and value creation of a data platform?"*
>
> RQ1.2: *"How well can a state of the art data platform architecture be applied to a small scale machine learning context outside of big data?"*

> RQ2.1: *"Can the principles behind the agile manifesto be mapped to the processes of state of the art data platforms to extract insights that may be applied to a small scale context?"*
>
> RQ2.2: *"Are there practices in small scale data engineering that map well to the principles of the agile manifesto?"*

Starting with RQ1.1, data lakes seem to still dominate the conversation in regards to data platform architecture. Especially so in the context of data science and machine learning endeavors. Data mesh seem to have gained some traction in the grey literature for this same context. Most of the literature on the subject of data platforms focuses mostly on scalability in large or huge scale contexts and there is a lack of best practices that can be applied regardless of scale. There are strategies to drive liveness and value creation of a data platform, such as the data product owner concept, that are discussed in literature that can be useful at all scales of operations. However, the fields of data platforms and data platform governance are still evolving. The development of new data platform architectures is continuous and the debate of which data platform governance methods are the most successful in driving liveness and value creation are not, by any means, settled.

In regards to RQ1.2, to put it short: yes we believe that state of the art architectures can be applied to a small scale machine learning contexts outside of big data. There are modifications that need to be made where the state of the art data platform architectures focus on solving big data problems, which may interfere with the needs of small scale engineering efforts. An example of this is the heavy use of distributed computing tools, which for a small scale context bring a significant overhead as described in the results from interview protocol B.

As discussed in the literature study analysis, much of the literature on ELT-type data platforms focuses on big data and distributed computing. We did find one architecture, the data mesh, which was especially promising specifically because it brought a more holistic view. In focusing less on software related technical details of how to implement scalability, which other architectures such as the data lake heavily do, we find that data mesh has larger potential to inspire solutions for a small scale context through the other aspects of its design. Data as a product and clear data ownership to drive value creation being examples of such aspects.

In the results of the case study interviews, it was expressed that the focus on distributed computing in state of the art solutions was an architectural property that interfered with their interests in a small scale context. An interest was also expressed in being able to share metadata and data semantics amongst data scientists which as we mention in the analysis of the interviews could be done via data management platforms [4]. However, these data management platforms have not been evaluated in this thesis and as such we cannot vouch for this kind of solution.

In our own data pipeline implementation analysis we focus on the practise of designing a data pipeline and which intermediate states of data throughout this data pipeline should be persisted to a data platform. We exemplify how data versioning can be applied to data and how to structure the data catalogue of a data platform to provide discoverability and the ability to ensure quality of data through data provenance. We believe that providing discoverability and provenance should be part of the technical implementation and that if executed correctly it will drive liveness and value creation.

Combining the philosophy and organizational concepts of the data mesh design with our design of a data pipeline and a data platform implementation, we believe that it is possible to design and implement a data platform based on a state of the art architecture that is suitable to a small scale context. Furthermore we believe that the findings from our mapping to agile principles can support the execution of the data platform implementation and of data pipeline implementations as well. Using the agile perspective and our mappings can serve as a solid starting point for identifying agile concepts and for starting to build on the ubiquitous language, [37], of the domain of small scale data engineering for machine learning.

Throughout the thesis we made unrelated findings that are relevant to small scale data engineering for machine learning overall but not to the research questions. The most prominent such finding is that it has been made evident to us from our experiences of this thesis that good data engineering cannot be performed in isolation. It has to be carried out in co-operation with the data source provider, which we learned from the qualitative investigation, and with the end user, which was exemplified by the findings from cleaning the wine reviews dataset. This sentiment is also found in literature. It is one of the focal points in the critique of Zhamak Dehghani, in her article on the data mesh, of how data engineers tend to be siloed off from the operational departments of an organisation.

It is brought up in the literature study by Dehghani [36] on the topic of data mesh 3.2.3 that siloing of data engineers through centralized specialized teams is a bad practise. We note that this is supported by Conway's law [64] which is very relevant to understanding the context of how a data platform may end up delivering middling results regarding its value creation. If one were to leave it up to an isolated team of data engineers to build and maintain a data platform and its contents, then one would run a high risk of ending up with a data platform that supports the needs of the data engineers rather than the needs of the users. We also find in our data pipeline implementation how decisions made when setting up the data cleaning may heavily affect the predictive capabilities of a dataset. These two findings exemplifies well why we identify that the role of the data engineer is very much that of a team support player, and that this role is best served in cross functional team environments. Of course this does not mean that the role of the data engineer is backstage, it is very much an interconnected role as its actions affect all parts of the decision making chain of a data driven organisation.

We believe that the results from this thesis can be useful in informing data engineering solutions for machine learning in small scale contexts overall but that the work should be viewed as exploratory and that it is far from complete. Things are however moving fast in this area and we remain positive that more substantial research will be carried out to start mending the gap between the current state of the literature and the needs of small scale data engineering efforts.

Chapter 7
Conclusions

We find that there is a gap between what is being written about in literature in regards to small scale data engineering for machine learning and what is applicable to the challenges and needs uncovered in the case study. The results of this thesis can help inspire implementation of data platforms and data pipelines in a small scale context and remediate the gap in literature. Most importantly we illustrate that there is a need for more research within the area of small scale data engineering for machine learning. To conclude we would like to emphasize the importance of focusing on solving challenges related to small scale efforts, to enable widespread adoption of machine learning throughout industry.

References

[1] The Economist. The world's most valuable resource is no longer oil, but data. *The Economist: New York, NY, USA*, 2017. `https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-resource-is-no-longer-oil-but-data` as of May 24th 2020.

[2] Samuel Flender. Data is not the new oil. *web: Medium*, 2019. `https://towardsdatascience.com/data-is-not-the-new-oil-bdb31f61bc2d` as of May 24th 2020.

[3] Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. Data management challenges in production machine learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1723–1726, 2017.

[4] Jesus Rodriguez. How linkedin, uber, lyft, airbnb and netflix are solving data management and discovery for machine learning. 2019. `https://towardsdatascience.com/how-linkedin-uber-lyft-airbnb-and-netflix-are-solving-data-management-and-di` as of May 24th 2020.

[5] Jim Gray and Prashant Shenoy. Rules of thumb in data engineering. In *Proceedings of 16th International Conference on Data Engineering (Cat. No. 00CB37073)*, pages 3–10. IEEE, 2000.

[6] the Research Data Alliance. Big data - defintion. `https://www.rd-alliance.org/group/big-data-ig-data-development-ig/wiki/big-data-definition-importance-examples-tools`, year=2020.

[7] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131, 2009.

[8] Martin Fowler, Jim Highsmith, et al. The agile manifesto. *Software Development*, 9(8): 28–35, 2001.

[9] Agile Alliance. What is agile software development. 2015. https://www.agilealliance.org/agile101/ as of May 24th 2020.

[10] Vahid Garousi, Michael Felderer, and Mika V Mäntylä. The need for multivocal literature reviews in software engineering: complementing systematic literature reviews with grey literature. In *Proceedings of the 20th international conference on evaluation and assessment in software engineering*, pages 1–6, 2016.

[11] James Furbush. Data engineering: A quick and simple definition. 2018. https://www.oreilly.com/content/data-engineering-a-quick-and-simple-definition/ as of June 08th 2020.

[12] Dice. Dice tech job report. 2020. https://techhub.dice.com/Dice-2020-Tech-Job-Report.html as of June 08th 2020.

[13] Maarit Laanti, Jouni Similä, and Pekka Abrahamsson. Definitions of agile software development and agility. In *European Conference on Software Process Improvement*, pages 247–258. Springer, 2013.

[14] digital.ai. State of agile report 14th addition. *web: state of agile website*, 2020. https://stateofagile.com/# as of May 24th 2020.

[15] Taiichi Ohno. *Toyota production system: beyond large-scale production*. crc Press, 1988.

[16] Ofri Raviv. What is the difference between a data pipeline and an etl pipeline? https://www.alooma.com/answers/what-is-the-difference-between-a-data-pipeline-and-an-etl-pipeline, year=2017.

[17] Mark Smallcombe. Etl vs elt: 5 critical differences. https://www.xplenty.com/blog/etl-vs-elt/#:~:text=ELT%20can%20load%20all%20data,data%20to%20transform%20and%20analyze.&text=It%20transforms%20data%20for%20integration,as%2Dneeded%20basis%20for%20analysis., year=2020.

[18] Won Kim, Byoung-Ju Choi, Eui-Kyeong Hong, Soo-Kyung Kim, and Doheon Lee. A taxonomy of dirty data. *Data mining and knowledge discovery*, 7(1):81–99, 2003.

[19] Agnieszka Mikołajczyk and Michał Grochowski. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary PhD workshop (IIPhDW)*, pages 117–122. IEEE, 2018.

[20] Zach Eaton-Rosen, Felix Bragman, Sebastien Ourselin, and M Jorge Cardoso. Improving data augmentation for medical image segmentation. 2018.

[21] Claus Pahl. Containerization and the paas cloud. *IEEE Cloud Computing*, 2(3):24–31, 2015.

[22] Docker. https://www.docker.com/, 2020.

[23] Kubernetes. `https://kubernetes.io/`, year=2020.

[24] Hannah Snyder. Literature review as a research methodology: An overview and guidelines. *Journal of Business Research*, 104:333–339, 2019.

[25] Renan Souza, Leonardo Azevedo, Vítor Lourenço, Elton Soares, Raphael Thiago, Rafael Brandão, Daniel Civitarese, Emilio Brazil, Marcio Moreno, Patrick Valduriez, et al. Provenance data in the machine learning lifecycle in computational science and engineering. In *2019 IEEE/ACM Workflows in Support of Large-Scale Science (WORKS)*, pages 1–10. IEEE, 2019.

[26] David Herron. Why git and git-lfs is not enough to solve the machine learning reproducibility crisis. 2019. `https://towardsdatascience.com/why-git-and-git-lfs-is-not-enough-to-solve-the-machine-learning-reproducibil` as of June 08th 2020.

[27] Jens Klump, Lesley Wyborn, Mingfang Wu, Robert Downs, Ari Asmi, Gerry Ryder, and Julia Martin. Principles and best practices in data versioning for all datasets big and small. version 1.1. *Research Data Alliance*.

[28] RR Fisher. Nasa heliophysics science data management policy. *Retrieved December*, 19: 2008, 2007.

[29] Keith D. Foote. A brief history of the data warehouse. 2018. `https://www.dataversity.net/brief-history-data-warehouse/#` as of May 31st 2020.

[30] James Dixon. Data lakes revisited. *James Dixon's Blog*, 2014. `https://jamesdixon.wordpress.com/2014/09/25/data-lakes-revisited/` as of May 20th 2020.

[31] Guru99. What is data warehouse? types, definition example. `https://www.guru99.com/data-warehousing.html`, year=2020.

[32] James Dixon. Pentaho, hadoop, and data lakes. *James Dixon's Blog*, 2010. `https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/` as of May 20th 2020.

[33] Pwint Phyu Khine and Zhao Shun Wang. Data lake: a new ideology in big data era. In *ITM Web of Conferences*, volume 17, page 03025. EDP Sciences, 2018.

[34] Coral Walker and Hassan Alrehamy. Personal data lake with data gravity pull. In *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*, pages 160–167. IEEE, 2015.

[35] Rihan Hai, Sandra Geisler, and Christoph Quix. Constance: An intelligent data lake system. In *Proceedings of the 2016 International Conference on Management of Data*, pages 2097–2100, 2016.

[36] Zhamak Dehghani. How to move beyond a monolithic data lake to a distributed data mesh. *Martin Fowler's Blog*, 2019. `https://martinfowler.com/articles/data-monolith-to-mesh.html` as of May 20th 2020.

[37] Eric Evans. *Domain-driven design: tackling complexity in the heart of software.* Addison-Wesley Professional, 2004.

[38] International Federation of Library Associations and Institutions. Section on Cataloguing. Standing Committee. *Functional requirements for bibliographic records*, volume 19. KG Saur Verlag Gmbh & Company, 1998.

[39] Joseph A Hourclé. Frbr applied to scientific data. *Proceedings of the American Society for information science and technology*, 45(1):1–4, 2008.

[40] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3, 2016.

[41] Tomasz Miksa, Andreas Rauber, Roman Ganguly, and Paolo Budroni. Information integration for machine actionable data management plans. *International Journal of Digital Curation*, 12(1):22–35, 2017.

[42] Tom Preston-Werner. Semantic versioning 2.0. 0. *Semantic Versioning.*, 2013.

[43] PROV Model Primer. W3c working group note. *W3C*, 2013. `http://www.w3.org/TR/2013/NOTE-prov-primer-20130430/` as of May 24th 2020.

[44] Luc Moreau, Ben Clifford, Juliana Freire, Joe Futrelle, Yolanda Gil, Paul Groth, Natalia Kwasnikowska, Simon Miles, Paolo Missier, Jim Myers, et al. The open provenance model core specification (v1. 1). *Future generation computer systems*, 27(6):743–756, 2011.

[45] Jianwu Wang, Daniel Crawl, Shweta Purawat, Mai Nguyen, and Ilkay Altintas. Big data provenance: Challenges, state of the art and opportunities. In *2015 IEEE International Conference on Big Data (Big Data)*, pages 2509–2516. IEEE, 2015.

[46] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. Software engineering for machine learning: A case study. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*, pages 291–300. IEEE, 2019.

[47] Leon Osterweil. Strategic directions in software quality. *ACM Computing Surveys (CSUR)*, 28(4):738–750, 1996.

[48] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[49] Eric Breck, Neoklis Polyzotis, Sudip Roy, Steven Euijong Whang, and Martin Zinkevich. Data validation for machine learning. In *Conference on Systems and Machine Learning (SysML). https://www. sysml. cc/doc/2019/167. pdf*, 2019.

[50] Denis Baylor, E Breck, HT Cheng, N Fiedel, CY Foo, Z Haque, S Haykal, M Ispir, V Jain, L Koc, et al. The anatomy of a production-scale continuously-training machine learning platform. In *23rd SIGKDD Conference on Knowledge Discovery and Data Mining*, 2017.

[51] Nick Hynes, D Sculley, and Michael Terry. The data linter: Lightweight, automated sanity checking for ml data sets. In *NIPS MLSys Workshop*, 2017.

[52] Zhamak Dehghani. Data mesh paradigm shift in data platform architecture. 2020. ://www.youtube.com/watch?reload=9v=52MCFe4v0UU.

[53] C Robson. *Real world research Blackwell 2nd edition*. 2002.

[54] Zack Thoutt. Wine reviews. 2017. `https://www.kaggle.com/zynicide/wine-reviews` as of June 08th 2020.

[55] Ml-flow. *web: https://mlflow.org/*, 2020. `https://mlflow.org/`.

[56] Kubeflow. `https://www.kubeflow.org/`, 2020.

[57] Argo. `https://blog.argoproj.io/`, 2020.

[58] Fahad Pervaiz, Aditya Vashistha, and Richard Anderson. Examining the challenges in development data pipeline. In *Proceedings of the 2nd ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 13–21, 2019.

[59] Gartner. Gartner hype cycle. `https://www.gartner.com/en/research/methodologies/gartner-hype-cycle`, year=2020.

[60] Gil Morrot, Frédéric Brochet, and Denis Dubourdieu. The color of odors. *Brain and language*, 79(2):309–320, 2001.

[61] Orley Ashenfelter and Karl Storchmann. The economics of wine, weather, and climate change. *Review of Environmental Economics and Policy*, 10(1):25–46, 2016.

[62] OpenWeather. Historical weather api. 2020. `https://openweathermap.org/history` as of May 27th 2020.

[63] Alexander Scheerer, Tobias Hildenbrand, and Thomas Kude. Coordination in large-scale agile software development: A multiteam systems perspective. In *2014 47th Hawaii international conference on system sciences*, pages 4780–4788. IEEE, 2014.

[64] Melvin Conway. Demystifying conways law. `https://www.thoughtworks.com/insights/articles/demystifying-conways-law`, year=2020.

Appendices

Appendix A
Interview Protocol A

This interview was conducted in Swedish, the interview protocol has been translated to English.

1. About the interviewee
   (a) Job title
   (b) Number of years of experience within relevant field
2. In what file format does the data reach you, is it consequent per project and do you recieve metadata, such as precision, of the data?
3. How large is the amount of data that is normally recieved for training a model and what is a normal velocity of the data stream in production?
4. Is there normally a structure for the recieved data and if so, to what level (structured vs unstructured)? Does it vary between the training data and live streaming data? Please exemplify.
   (a) Are you informed of which features are guided by control values and which features are not and are you informed of their dependencies?
5. What is the level of quality like of data, is there often cleaning that has to be done before the data can be used?
   (a) If yes: what types of cleaning are commonly perfomed?
6. How long time does the whole data processing take. Is choice of what model to use for inference affected by data processing lead time in regards to expected velocity and volume of data?

7. Is there further work unrelated to data processing that has to be done before one can start building a model for the data?
    (a) Is data persisted at the processed stage so that one can reuse it for future development of other models?
8. In what state do you prefer to recieve data?
    (a) Do you only work with data in local environments to train and test out models or do you also work with production environments? [From here on referred to as local laboratory environment]
        i. If they do work with production environments: How involved are you in setting up data infrastructure in the production environment and what type of infrastructure (scheduled batch-oriented, streaming, manual) is most often used?
9. Illustrate and describe a typical data pipeline and its processes. [whiteboard was used]
    (a) What activities are performed in each processing step?
    (b) What processes exist in a laboratory environment and how does it differ from a production environment?
    (c) What tools are commonly used by you for these processes and activities of the data pipeline?
10. What do you believe are the greatest challenges related to working with the data. Please describe using [their illustration] the whiteboard. If they work with data processing and infrastructure in production:
    (a) How much of the infrastructure that is built can be reused between the local laboratory environment and the production environment? What are the main differences between the local laboratory environment and the production environment?
11. Are you aware of any data versioning tools?
    (a) If yes: are you using any of them and if so, how?
12. Do you believe that a tool that simulates git [Sentian uses git] in its user experience but that versions data and data-to-code relationships would be helpful for developing models?

Appendix B
Interview Protocol B

This interview was conducted in Swedish, the interview protocol has been translated to English.

1. Who [which job titles] develops data piplines and does it depend on the level of complexity of the specific data pipeline?
2. Do you use mockups [or similar techniques] that a data scientist can provide a data engineer with if it is too advanced for the specific data scientist to deploy a data pipeline to production themselves?
3. Would it be possible to get data scientists to work according to test driven development [when developing their data pipelines], especially in the case that the data scientist is not responsible for deploying the data pipeline to production?
4. What information were you interested in gathering [on a data platform], where you only interested in structuring data provided for customer projects or internal data as well?
5. Would you agree that one could view your organization as its own domain [with regards to the data mesh design [36]] since it is rather small?
6. How are product owners [or similar role] defined in your customer projects? [With regards to the potential catch 22 of: (1) the role of product owner being shifted to Sentian as the customer often wants an overall solution and (2) Sentian often not possessing domain expertise inhouse which would be useful for someone in the role of project owner.]

Appendix C

Code Used in Data Pipeline Implementation

```
import matplotlib.pyplot as plt
import pandas as pd
import collections as c
import numpy as np
import os
from sklearn import preprocessing as pp

# Ingest data (read from file)
wine_reviews = pd.read_csv('winemag-data-130k-v2.csv', index_col = 0)

#Drops rows containing missing values in columns considered too important (see su
def cleanup_filter(df):
    df_no_missing_values = df.dropna(subset=['price', 'points', 'title', 'taster_
    return df_no_missing_values

# Expand titles -> year, name.
def feat_expander(df):
    df1 = df.title.str.extract('(\d+)')
    df1.columns = ['year']
    title = df.title.str.replace('\d+', '')
    df1['title w/o year'] = title
    return df1

# Normalizes  numeric data such as price and points.
def normalizer(df):
    x = df[['price', 'points']] #returns a numpy array
    min_max_scaler = pp.MinMaxScaler()
    x_scaled = min_max_scaler.fit_transform(x)
    df_numerics_scaled=pd.DataFrame(x_scaled, columns=x.columns)
    return df_numerics_scaled

# Remove symbols such as !,. etc. (non-words)
def clean(word):
    word = word.lower().strip()
    word = re.sub(r'[^a-z0-9]', '', word)
    return word

# Returns a counter for all words in a discription column
def word_counter(df): # count all words
    words = dict() # set: {1,2,3}, dict: {k:v,k:v}... {} => dict, dict() => dict, set
    for row in df:
        # words = count(row)
        for word in row.split():
            word = clean(word)
            words[word] = words.get(word, 0) + 1
    return words

# Remove stopwords, and lowcount words, etc
def filter_words(words, limit=2):
    stopwords = {"and", "or"}
    words = {key: value for key, value in words.items() if value >= limit and key not
    return words

# For each description of the rows in a df, count appearences of unique
# words in word_count that are deemed important
def count_words_by_row(df, word_count):
    list_of_series = list()

    for index, row in df.iteritems():
        row_dict = dict()
        for word in row.split():
            word = clean(word)
            if word in word_count:
                row_dict[word] = row_dict.get(word, 0) + 1
        row_series = pd.Series(row_dict, name=index)

        list_of_series.append(row_series)
    new_df = pd.DataFrame(list_of_series)
    new_df.fillna(0, inplace=True)
    return new_df

# Example of a test-cell for methods: clean(), count_words_by_row(), word_counter() a
#  are the methods related to  preparing the data for feature encoding of description
```

```
raw = pd.DataFrame([["hej jag heter Johan"], ["hej jag heter Anton   "], ["din ma

indata = pd.DataFrame([["hej jag heter Johan"], ["hej jag heter Anton   "], ["din
actual = word_counter(indata["description"])
expected = {"hej": 2, "jag": 2, "heter":2, "johan": 1, "anton": 1, "din": 2, "mam
assert(len(actual) == len(expected))
for key, value in expected.items():
    assert(actual[key] == value)

# df_boxplot to help investigate distributions of values in a column
def df_boxplot(df, colName, nbrOfRows, sampleSize):
    varieties_count = df.variety.value_counts()
    # Filter by greater than (gt) according to count made above of number of occu
    # values in a certain column.
    varieties_filtered = df[df.variety.isin(varieties_count.index[varieties_count
    col_values = varieties_filtered[colName].unique()
    col_scores = []
    #TODO sort according to descending for col_scores

    for title in col_values[:sampleSize]:
        col_value_row = df[df[colName] == title]
        col_scores.append((title, col_value_row['points'].tolist()))
    #print(titles_scores)
    col_values, scoress = zip(*col_scores)
    #print(titles)
    plt.xticks(rotation=90)
    plt.boxplot(scoress, labels=col_values)
    plt.show()
    pass

df_boxplot(wine_reviews, 'country', 0, 25)
df_boxplot(cleanup_filter(wine_reviews), 'country', 0, 25)

# Adds the two columns 'lat' and 'lon' which are the latitudes
# and longitudes of the country, province and region_1 of the df.
# These columns are filled with values retrieved from google maps.
def geo_adder(df):
    #Make sure we dont change the original df object.
    df = df.copy()
    latlon = list()
    for row in df.iterrows():
        geocode_result = gmaps_key.geocode({'locality': row[0],'administrative_ar

        #TODO spara geocode i en fil
        try:
            lat = geocode_result[0]['geometry']['location']['lat']
            lon = geocode_result[0]['geometry']['location']['lng']
        except (KeyError, IndexError) as e:
            lat = None
            lon = None
        latlon.append((lat,lon))
    #print(latlon)

    df['lat'] = list(zip(*latlon))[0]
    df['lon'] = list(zip(*latlon))[1]

    return df

# Example of running one of the enrichments, Geocoding, and persisting data as .pickle
if not os.path.exists('geo_coordinates.pickle'):
    #Remove the USA niagara row that messes up the unique list below and process its c
    #to later concatenate it with the df with coordinates manually. The df_new looks a
    niagara_quickfix = df.iloc[[123755]]
    wine_reviews_no_usa_niagara = wine_reviews.drop(123755)

    #Remove duplicates
    wine_reviews_no_duplicates = wine_reviews.drop_duplicates(subset ="title",keep="fi

    #Groupby uniques so we dont call google api 100k times but only once for each regi
    unique_geographical_wine_origins = wine_reviews_no_duplicates.groupby('region_1')
    #Same for the quickfix, its to get same format on them so that they can be concate
    niagara_quickfix = niagara_quickfix.groupby('region_1')[['province','country']].ag

    #Get coordinates for list of unique places and USA niagara.
    origin_coordinates = geo_adder(unique_geographical_wine_origins)
    niagara_coordinates = geo_adder(niagara_quickfix)
    #Add the df with coordinates for USA niagara
    complete_coordinates = pd.concat([niagara_coordinates, origin_coordinates], ignore
    complete_coordinates.to_pickle("geo_coordinates.pickle")

else:
    complete_coordinates = pd.read_pickle('geo_coordinates.pickle')
```

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-06-15

EXAMENSARBETE Towards Agile Data Engineering for Small Scale Teams
STUDENT Anton Engström
HANDLEDARE Johan Ullén (Sentian), Emma Söderberg (LTH)
EXAMINATOR Per Runeson (LTH)

Datainfrastruktur för maskininlärning inom industrin

POPULÄRVETENSKAPLIG SAMMANFATTNING **Anton Engström**

Industrin rör sig snabbt mot vad man kallar industry 4.0 där man med hjälp av datainsamling och smarta system går mot nästa era. För att matematiker och datavetare ska kunna integrera sina modeller som utgör hjärnan i de smarta systemen så krävs det välfungerande datainfrastruktur, men hur man utvecklar denna är ännu ett ungt område inom forskning.

När det kommer till att bygga smarta system så har man ofta en matematisk modell av det man vill styra eller samla insikter om. Under 2010-talet har det blivit mer och mer vanligt med matematiska maskininlärningsmodeller, som med en uppsättning av parametrar tränar sig själva på data för att förstå ett visst problem. Denna teknologi har till stor del hittills applicerats i sammanhang såsom sociala medier för att förutspå beteenden och för att rekomendera rätt produkter genom reklam. I dessa sammanhang så skapas enorma mängder data från användares beteenden och det skapas med en accelererande takt då allt mer data samlas in av smarta produkter såsom mobiltelefoner och internetuppkopplade tv-apparater. Detta har namngivits "Big Data" och hur man hanterar problem som uppstår i samband med den explosionsartade tillväxten av data är just nu huvudfokus inom stora delar av den akademiska litteraturen på området "data engineering" som kan översättas till datahantering på svenska.

I mitt examensarbete har jag samarbetat med ett företag, Sentian, som implementerar maskininlärningslösningar för kunder inom industri som strävar mot industry 4.0. Under mitt examensarbete så upptäckte jag att det fanns ett gap mellan problemen och lösningarna som diskuterades i litteraturen och de problem som Sentian stöter på när det kommer till just datahantering.

Jag gjorde en egen ansats på att brygga gapet genom att applicera principer från agil utveckling av programvarusystem och kartlade dessa mot mina resultat från litteraturstudien. Detta gjordes som ett försök att extrahera koncept och principer från litteraturen som fokuserade på "Big Data" till allmänna principer för datahantering. Dessa kan appliceras av Sentian vid utveckling av datainfrastruktur även i mindre sammanhang utanför domänen av "Big Data", vilket ofta är det sammanhang som de arbetar inom med kunder från industri.

Det finns stora potentiella samhällsnyttor i att stötta industrin i sin strävan mot att introducera smarta system som t.ex. kan effektivisera resursanvändningen och minska utförandet av farliga aktiviteter av människor. Resultatet är av explorativ karaktär och kan användas av företag i Sentians situation för att inspirera arbetssätt vid utveckling av datahanteringsinfrastruktur. Resultatet agerar också som en pekare mot att det behövs mer forskning inom detta området.