TU WIEN Informatics

Applying Machine Learning for Processing Time Prediction for Inbound Shipments

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Business Informatics

eingereicht von

Fabian Eichhorner, BSc
Matrikelnummer 01227328

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Priv.-Doz. Dr. Nysret Musliu
Mitwirkung: Dr. Sebastian Skritek

Wien, 14. Oktober 2021

Fabian Eichhorner   Nysret Musliu

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Informatics

Applying Machine Learning for Processing Time Prediction for Inbound Shipments

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Business Informatics

by

Fabian Eichhorner, BSc
Registration Number 01227328

to the Faculty of Informatics

at the TU Wien

Advisor: Priv.-Doz. Dr. Nysret Musliu
Assistance: Dr. Sebastian Skritek

Vienna, 14th October, 2021

Fabian Eichhorner Nysret Musliu

Erklärung zur Verfassung der Arbeit

Fabian Eichhorner, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 14. Oktober 2021

Fabian Eichhorner

Acknowledgements

First and foremost I would like to express my gratitude to my advisors Priv.-Doz. Dr. Nysret Musliu and Dr. Sebastian Skritek. Without their continuous advice, valuable feedback and encouragement, this thesis would not have been possible.

I would also like to thank my family, specifically my parents, for enabling me to pursue my study and supporting me at any time.

Additionally, I would like to thank all SEMPRE project members for their great expertise during the project. Specifically, Dr. Philip Ramprecht from Fraunhofer Austria Research Gesellschaft and the ZKW Group GmbH for providing their valuable expertise.

The research has been supported by the Austrian Research Promotion Agency (FFG), project 864792 (SEMPRE)

Kurzfassung

Die Inbound-Logistik befasst sich mit der Verwaltung und Lieferung von Waren an einen Fertigungsstandort. In dieser Arbeit liegt der Fokus auf dem Prozess von der Ankunft der Waren am Standort bis zu deren Einlagerung. Für die Kapazitätsplanung muss die Bearbeitungszeit einer Lieferung abgeschätzt werden können. Eine nicht optimale Planung führt zu einer ineffizienten Nutzung und Verschwendung von Ressourcen.

Diese Arbeit entwickelt und evaluiert Modelle zur Vorhersage der benötigten Bearbeitungszeit für eine eingehende Lieferung. Im Speziellen werden Regressionsmodelle erstellt, die bei gegebener Konfiguration einer Lieferung, eine Abschätzung über die Bearbeitungszeit dieser Lieferung vorhersagen können. Um solch ein Modell zu erstellen, müssen mehrere Probleme gelöst werden. Die Prozessstruktur ist nicht klar definiert und das Wissen über den Prozess ist, bei Domainexperten, in verschiedenen Use-Case und Methods-Time Measurement (MTM)-Modellen, verteilt. Außerdem ist nur eine sehr geringe Anzahl an Daten über solche Lieferungen verfügbar.

Um diese Probleme zu lösen, entwickeln wir zunächst Ontologien, die das vorhandene Wissen zusammenfassen und den darunter zugrunde liegenden Prozess modellieren. Konkret entwickeln wir zwei Ontologien. Eine Ontologie, die mögliche Konfiguration des Anlieferungsprozess darstellt und eine Ontologie, die diese, durch die Verknüpfung von MTM-Aufgaben, mit Transportelementen erweitert.

Auf Grundlage dieser Ontologien erstellen wir drei synthetische Datensätze. Zwei Datensätze sind basierend auf vollständig synthetischen Daten aus der MTM-Ontologie. Für den dritten Datensatz erstellen wir eine agent-based Simulation des Prozesses und messen daraus synthetische Lieferungen. Darüber hinaus analysieren wir die Features der erstellten Datensätze und prüfen diese. Anhand dieser Datensätze evaluieren wir verschiedene Regressionsmodelle und testen ihre Robustheit auf unterschiedlich großen Subsets. Im Allgemeinen analysieren wir, welche Regressionsmethoden auf MTM-Datensätze und welche Regressionsmodelle für den Simulationsdatensatz am besten geeignet sind. Außerdem vergleichen wir die Ergebnisse der verwendeten Regressionsmodelle mit den Ergebnissen eines AutoML-Regressionsmodells. Wir zeigen dabei, dass AutoML gleich gute oder bessere Lösungen für alle drei Datensätze findet.

Schließlich vergleichen wir verschiedene Merkmalsdarstellungen der Datensätze. Wir zeigen, dass eine Liste von Produkten, d.h. die Information, die während der Planungszeit

verfügbar ist, nicht ausreicht, um die Bearbeitungszeit einer Lieferung mit einem angemessenen Fehler vorherzusagen, und dass zusätzliche Information in Form von Pakettypen notwendig ist.

Abstract

The inbound-logistic domain deals with the management and movement of goods from a supplier into a warehouse. In this thesis, the focus is on the arrival of a shipment until the storage of its goods in a warehouse. For capacity planning, a company needs to estimate the processing time of a shipment, as non-optimal planning leads to an inefficient usage and waste of resources.

The aim of this thesis is to develop and evaluate models to predict the required processing time for an inbound shipment. In detail, regression models should be built that, given the configuration of a shipment, can return an estimation of the processing time of the shipment. However, to be able to build such models, several issues appear. The process structure is not clear, knowledge about it is mainly held by domain experts and split up in different use-case diagrams and Methods-Time Measurement (MTM) models. Furthermore, only a small amount of data samples of such shipments are available.

To tackle these problems, we first design ontologies to capture the knowledge and model the underlying process. Specifically, we develop two ontologies, one ontology to capture the possible shipment process configuration and one ontology extending it by linking MTM tasks to shipment elements.

Based on the gathered knowledge, we create three synthetic datasets. Two datasets are based on sampling fully synthetic data from the ontology containing the MTM information. For the third dataset, we create an agent-based simulation of the process and sample synthetic shipments from it. Furthermore, we analyze the features of the resulting datasets and review the importance of single features.

Using these datasets, we evaluate different regression models and test their robustness on different sized subsets of the datasets. In general, we analyze which regression methods work best on the MTM datasets and which regression models work best for the simulation dataset. Furthermore, we compare the results from the used regression models with the results from AutoML regression. We show that AutoML can find the same or better solutions than the handcrafted models for all three datasets.

Finally, we compare different feature representations of the datasets. We show that a list of products, which is the information that is available during planning time, is not enough to predict a shipment's processing time with a reasonable error and that additional information in the form of package types is necessary.

Contents

CHAPTER 1

Introduction

The supply chain process deals with delivering a product to a customer. One part of the supply chain is the inbound-logistic domain, which deals with the management and movement of goods from a supplier to a warehouse. As part of the receiving process, a shipment arrives at a factory, it needs to be processed and its goods need to be stored in a warehouse. For capacity planning, it is required to estimate the processing time of such a shipment. Non-optimal planning would lead to an inefficient usage and a possible backlog in the facility.

Usually, the estimation of these processing times is done by domain experts with standardized times called Methods-Time Measurement (MTM) [KB87]. MTM breaks a process down into different tasks and estimates the time for each task. However, only once the required steps are clear, an estimate can be predicted. Short product lifecycles in addition to high levels of product variation result in an increased complexity for this task. Therefore, a more dynamic solution for prediction of a shipment processing time is desired.

Knoll et al.[KPR16] propose to use a combination of mathematical models and Machine Learning (ML) models to predict the future inbound for planning scenarios. However, they offer only a theoretical framework and their approach has not been implemented and evaluated yet. A recent systematic literature review gave an overview of ML usage in the supply chain domain [TSN+21]. To the best of our knowledge, beside Knoll et al.[KPR16], none of the examined papers deals specifically with the inbound logistic domain.

What makes such a prediction task complicated is that when a shipment arrives, a company has only knowledge about the ordered items and an expected arrival time of the shipment. However, the company has no knowledge in which packaging and in which order the goods will arrive. Depending on the used packaging, workers might have to remove existing packaging and move products into another storable container. The processing time of these tasks depends heavily on the configuration of the shipment. For example, while the processing of a pallet containing products in already storable containers is fast, the processing of another pallet containing similar products in non-storable containers takes much longer.

In this thesis this process is analyzed for an automotive supplier company. For this setting, some constraints apply. A shipment arrives with a truck and consists of up to 32 pallets. The pallets are brought to a processing place, where up to 4 pallets can be handled in parallel. Each pallet has at least one package unit. A package unit always stores one product. The maximum number of package units on a pallet is limited by the package unit and pallet size. To process a pallet a worker must assign a label, depending on the package unit, change the products to a storable container and, depending on the pallet, change the pallet to a storable pallet type. Afterwards the products can be stored in the warehouse.

To understand how the process works in detail, expert knowledge is required. For the process, different use-case diagrams and MTM models are available. To merge all the existing knowledge and provide it to other project stakeholders, a knowledge base is desired. This knowledge base specifically contains the possible configurations of such a shipment and their link with existing MTM elements.

To predict processing times with a ML model, a dataset is required. For collecting such data in the factory, sensors were placed above a pallet. The occurring actions, i.e. when a package is moved, were captured. Additionally, corresponding shipping information for those pallets are available. However, due to the high time and resource costs of collecting this data, only a small amount of data samples is available. If one wants to learn to predict the shipment's processing time, this leads to the problem that not enough data is available to train the models.

One way to overcome such a problem is by creating synthetic datasets. A simulation of the process can be built upon the structured knowledge and combined with constraints, hypothesis and assumptions by domain experts. Using this simulation, a synthetic dataset of the process can be generated and further be used to train ML models. Having those datasets, questions of how the different state-of-the-art regression models perform can be answered. Furthermore, the impact of different features in these models can be observed.

Typical regression methods require that hyperparameters are learned. Additionally, the results of a model heavily depend on the characteristics of the used dataset. One way to overcome such a problem, by automatically finding and training models, is AutoML.

1.1 Problem Definition

The main problem tackled by this thesis is that there exists no automatic prediction, with a ML model, to estimate the required processing time for an inbound shipment. To solve this problem, two other problems need to be addressed:

First, the existing knowledge about this shipment process is mainly held by domain experts. Data analysts have little to no knowledge about the logistic domain and its used vocabulary. To fully understand the process, domain knowledge is required. Information about the process is split up into different use-case and MTM models. Further details about the process are only known by domain experts.

Secondly, data collection for this process is time and resource expensive. ML models require a large dataset to find robust solutions. No dataset is available to train such a model.

1.2 Aim of the Work

The main aim of this thesis is to use ML to improve the prediction of processing times for an inbound shipment. This should be done by creating a regression model to predict the required time for a given shipment order. Furthermore, the following goals should be pursued:

The first goal is to design an ontology to capture the knowledge and model the underlying process. The aim of this step is to detect the key concepts and relationship within the process. The outcome of this step should be ontologies containing information about process configurations and their MTM task linking.

The second goal is to design and model synthetic datasets. The aim of this step is to use the knowledge gathered in the ontology to build synthetic datasets. These datasets should provide a meaningful abstraction of the process and be able to be used for regression problems.

The third goal is to identify important features in the dataset. The aim of this step is to analyze the feature importance of the datasets and identify features with a high feature importance and features with a low feature importance.

Finally, the last goal is to apply and evaluate different regression algorithms, such as linear regression models and AutoML regression, on the synthetic datasets to evaluate and compare them. The aim of this task is to identify important features and compare AutoML models with handcrafted models for the specific use case. To gain insights of which model performs well for the given process, different experiments should be conducted with the models on different sizes and subsets of the dataset.

To conclude, this thesis addresses the following research questions:

- What are the key concepts in the incoming goods process and how can they be modeled?
- What is an appropriate mean to generate a synthetic dataset, which captures the incoming goods process? How can the correctness/generalization of this synthetic dataset be evaluated?

- Which are the key features for developing a prediction model of a given shipment?
- What is a good ML model to predict the required time for a given shipment? How do the results differ between an AutoML model and a handcrafted model?

1.3 Contributions

The contributions of this thesis are:

- The creation of two ontologies about an inbound logistic shipment process. One ontology capturing the possible configurations of a shipment and one ontology extending the shipment configuration with MTM task estimations.
- The generation of three synthetic datasets for regression problems. One dataset uses fully synthetic data according to the MTM ontology scheme. The second one tweaks the first one by adding randomness, noise and non-linearity. The third one uses an Agent-Based Modeling (ABM) simulation and samples data out of the simulation.
- The building and evaluation of different regression models on the three datasets. Their performance is evaluated on different sizes of datasets and the results of handcrafted models are compared with AutoML results.
- A comparison between different ways to represent features of a shipment. Finally, a comparison of two feature representations, i.e. one only including information about ordered products and one including meta information, is made.

1.4 Structure

This thesis consists of six chapters. The first chapter provides an introduction as well as the motivation behind this thesis. In the subsequent chapter, a summary of used ontologies, time prediction and regression models in the field of supply chains is given. The third chapter describes how the knowledge is captured and ontologies are created. The fourth chapter describes how this knowledge is used to create synthetic datasets. Additionally, an explorative analysis of the datasets is performed in this chapter. The fifth chapter deals with the results of the data analytics part. It includes information about the models, a feature importance analysis, how their hyperparameter were selected and how they have performed on the different datasets. A final conclusion sums up the results of this thesis and provides an outlook for possible future work.

CHAPTER 2

Research Method and Related Work

This chapter consists of two parts. First, the applied research methodology is described. Secondly, a state of the art and related work for inbound logistic time prediction is given. In detail, the related work gives an overview of current state-of-the-art methods for inbound logistics demand prediction, existing ontologies in the supply chain domain and a summary of commonly used regression methods for such problems.

2.1 Research Method

The chosen research methodology is based on the open data mining standard CRoss Industry Standard Process for Data Mining (CRISP-DM) framework [Wir00] with a few adaptions. In specifically, an additional stage of literature review has been added at the beginning because CRISP-DM is designed for nonacademic processes. The other stages are designed according to the CRISP-DM 1.0 user guide [NSN+00]. The following seven stages are part of the methodology:

Literature Review This stage is performed to gain background knowledge about prediction models, the inbound-logistic domain and specifically the interdisciplinary work on these domains. The aim is to perceive a general overview about similar solutions in other domains and an overview about current state of the art in the specific domain. The review will focus on novel approaches i.e. paper released in the last five years. Results gathered in this stage will be documented.

Business Understanding The aim of this stage is to determine business objectives and data mining goals, as well as assessing the current situation. To assess the current situation a knowledge base is build, this knowledge base is modeled with an ontology.

The ontology captures all the required knowledge and relationships within the process. The result of this stage is the definition of the research questions, the definition of the key concepts and the creation of the ontology.

Data Generation and Understanding The aim of this task is to collect the data and discover first insights of the data. This step consists of two main tasks, collecting the real-life data and creating a simulation to generate synthetic data.

For the initial collection of data, data is provided by the company as Excel spreadsheets and shipment records. First, this raw data and its available features and quantity of the data are analyzed.

Secondly, this step includes the creation of a simulation. This simulation is based on the knowledge gathered in the ontology. The simulation should be a framework representing the process, which gives a user the possibility to generate synthetic data. The framework should be able to run with different parameters and generate a specified amount of data samples.

Data Preparation The aim of this stage is to define the datasets used for the model. In this stage, multiple datasets are created. A synthetic dataset by sampling data according to the ontology schema and a dataset created from the simulation.

To create the datasets, a feature representation of each dataset needs to be defined. A feature selection algorithm is applied to remove features without information. Furthermore, the features will be encoded to be in an ML usable format.

Additionally, each dataset should be analyzed with explorative data analysis and its integrity and correctness should be checked. For the synthetic datasets, all features are validated by domain experts and the correctness verified.

Build Model In this stage, various modeling techniques are selected and applied. To find out which model works best for the given use case different regression models such as linear regression models or Multi-layer Perceptron (MLP) Regression are selected and their optimizable parameters are defined. Using random search, the optimal parameters for each of the model are defined. Additionally, the datasets are used within an AutoML framework. The output of this stage is a set of trained models.

Evaluation The results of the models are evaluated on the synthetic datasets using standardized measurements like Root Mean Squared Error (RMSE). The evaluation also consists of how different models can cope with a smaller subset of the available data. Additionally, the AutoML model is compared with the best handcrafted model. Finally, the capability of the model is evaluated on the real data set. The results gathered in this stage will be described and explained in the thesis.

Deployment Due to dependence on external factors, the deployment of the model will not be part of this thesis. However, all accomplishments and findings will be documented in the master thesis.

2.2 Related Work

A common way to estimate the processing time by a domain expert is to model it with standardized times, such as MTM [KB87]. In this method the process is split up into atomic tasks. For each of those tasks, an average time is provided by a statistic. This time has been calculated by running the task for thousands of instances and calculating the average spend time for this task. This estimation is deterministic and only works if the complete knowledge about the configuration of the shipment is available, i.e. which kind of pallet, which package units are used and how the configuration of the pallet looks like. However, in the real life that information is not available during planning time and is only available once the goods arrive. Another issue with this approach is that it strongly depends on the level of detail of the modeling of the MTM tasks.

To overcome this issue, more advanced prediction methods have been designed. A general model for predicting inbound logistic was introduced by Knoll et al. [KPR16]. They introduced a system model on the base of an inbound logistic ontology to predict future inbound. The system model consists of combinations of mathematical models and ML models. Using this model, it is possible to predict future inbound for a newly created planning scenario.

Furthermore, a recent systematic literature review analyzed artificial intelligence applications in supply chain management [TSN$^+$21]. However, besides Knoll et al. none of the discussed papers dealt with inbound logistics.

Simulation Approaches Besides ML models, simulations have been used to in the domain of inbound logistics. Smith et al. [SS19] used simulations to evaluate warehouse check-in strategies. Munoz et al. [MVVMH$^+$21] introduced an approach to measure the environmental impact of inbound logistic, where they used discrete event simulation to estimate the environmental cost of each shipment. Another simulation approach was introduced by Beyer et al. [BGYW16]. They introduced an assistance system to support the inbound logistic planning. Their approach is using an agent-based simulation to provide an overview of possible solutions and evaluating those. The results serve in an assistance system as support for an intra-logistic planner.

Furthermore, simulations have also been used to generate data. Pfeiffer et al. [PGKM16] used a discrete event simulation to generate data for a small sized flow-shop system. The data was then used with regression methods to create a predictor for manufacturing lead time.

Regression Methods for Time Prediction Regression models have been used in various prediction models [SCHP00]. Typical regression models include linear regression models, k-Nearest Neighbors (k-NN) Regression, Random Forest regression or MLP Regression [SCHP00]. However, there is not one model which provides the best results for all types of datasets. Furthermore, for such models the hyperparameter must be optimized and the best model must be selected.

Recently, AutoML has been introduced, which automatizes the selection of the best model for a given problem [HKV19]. To measure the performance of AutoML various competition have been held, where the *auto-sklearn* library gave good results [FEF+20].

Another important factor for model performance is the number of used features. The amount and the contained information of the features might influence the results [KJ13]. To determine which features are important, features can be ranked by their feature importance. One approach for such a ranking metric is by correlation [KMJ10] [HS98]. Another approach to measure the importance is by using intrinsic measurements of models. This can be done by measurements of increase in performance of tree-based models or by measuring the linear coefficients of linear regression models [KJ13]. Depending on the results of the feature importance, feature selection can be applied and the feature space can be reduced.

Most current algorithms are designed for large datasets, small datasets might lead to a reduced chance of detecting a true effect [BIM+13]. In our case only a small amount of real captured data is available. To overcome such a problem, an idea is to use additional information for inference. One idea to overcome this problem is by taking advantage of additional semantic knowledge [PPHM09]. An application of such an approach is to use the knowledge of ontologies for the inference. However, these models are rather complex to implement and require high resources. A simpler method is to use the knowledge of ontologies to create simulation models. One such approach has been introduced in the usage of an ontology-driven simulation modeling framework [BPM06].

Existing Supply Chain Ontologies The ontology engineering process might be improved by using existing ontologies, such that one could refine or extend existing ontologies [NM01].

Ontologies are commonly used in the logistic or supply chain management domain. Scheuermann et al. [SL14] conducted a survey about existing supply chain management ontologies. They analyzed over 16 ontologies. All of those 16 ontologies are not publicly available on the web.

In the specific area of inbound logistics, not many ontologies can be found. An ontology about the hierarchical structures of intra-logistic elements of storage and transporter was introduced by Negri et al. [NPF+17]. They created an example use case with "Automatisches Kleinteile-Lager" (in English: Mini-Load Systems) (AKL) interacting with an inbound logistics ontology based on an industrial ontology called eScop Manufacturing System Ontology (MSO). While these ontologies focus on a similar use-case and might bring a benefit to reusing it, unfortunately these have not been made publicly available.

From the publicly available ontologies Supply Chain Operator Reference (SCOR)Voc deals with the supply chain. SCORVoc is a vocabulary which formulizes the SCOR standard. It defines a conceptual model for supply chain related information in the form of processes [PGGC+16]. However, the structure of the modeled process differs too much from the processes in use-case diagrams and MTM modelling.

Recently, ontologies in the field of industry 4.0 have been published such as Reference Architecture Model for Industry 4.0 (RAMI) [YOSPV19], AutomationML ontology (AMLO) [KGGS+18] or Standards Ontology (STO) [GGBH+17]. These ontologies however don't deal with the specific process of incoming goods and don't provide any additional knowledge such that reusing its vocabulary would make sense.

CHAPTER 3

Business Understanding

This chapter deals with how information from an existing inbound shipment process can be collected, understood and documented such that it can be used for all stakeholders in the project.

The main goals of this chapter are to translate knowledge of domain experts into a consistent vocabulary and to create a knowledge base containing all relevant information for the process. Therefore, this knowledge acquisition is the foundation for the further steps.

To do so, first artifacts in the form of use-cases, existing processes and flow diagrams were captured. Based on this knowledge and additional expert interviews, two ontologies were designed. One ontology capturing possible configurations for the shipment process and one ontology extending the first one with a time prediction model.

Business Process Summary This thesis focus is on the inbound logistics part of the supply chain management process, in specific the process from the arrival of a shipment on a truck until the goods are ready to be stored in the warehouse. For the reviewed company, this storage is done in an automated mini-load warehouse system called AKL.

Once a shipment arrives at the facility workers are processing the shipment. To do so, the shipment data is registered in the system and its pallets are processed. One shipment can consist of up to 32 pallets. On every pallet a set of package units is stacked. The pallets can be of different kinds, however only the normed plastic pallet can be used in the AKL storage. If the pallet is not of this type, it needs to be changed. Additionally, it is possible that the pallet has a stretch wrapping or bandages. Wrapping or bandages need to be removed before storing.

The arrived goods are always stored in a package unit on a pallet. This package unit might be already ready to store, i.e. the case when it is a normed “Kleinladungsträger” (in English: Small Load Carrier) (KLT) container, or it might have to be rearranged into an AKL usable container. Multiple package units can be stacked on a pallet. Each package unit always contains only one product type. If a pallet has only one type of products, which are already in AKL conform containers, it can be directly stored.

Once all package units are processed, the processing of the shipment is finished.

Ontologies There are many definitions to what an ontology is. Noy et al. defined an ontology as: "*An ontology is a formal, explicit description of concepts in a domain of discourse (classes (sometimes called concepts)), properties of each concept describing various features and attributes of the concept (slots (sometimes called roles or properties)), and restrictions on slots (facets (sometimes called role restrictions))*" [NM01].

Out of these terms, this thesis uses the terms classes, properties and role restrictions. Properties can be further divided into data properties and object properties. Data properties relate between individuals and literals. Object properties relate between two individuals.

A typical way to formulate an ontology is by using Web Ontology Language (OWL), a specification by the World Wide Web Consortium (W3C) to create, publish and share ontologies [MvH04]. An OWL ontology consists of classes, properties, and individuals and is formulated with Resource Description Framework (RDF) triples. Such a triple consists of a subject, predicate and object. The formal writing of these triples can be done in different formats. One example is Terse RDF Triple Language (Turtle) syntax [1], which provides shorthand usage if a subject or a subject and predicate is used.

The W3C has created an updated version of OWL called OWL 2 Web Ontology Language (OWL 2). However, OWL 2 is a very expressive language for user and computer, which can make it difficult to work with [HPPsR09]. To overcome this, three sub-languages of OWL 2 have been introduced: OWL 2-EL, OWL 2-QL and OWL-RL [MvH04].

OWL 2-EL is made for ontologies that define a very large number of class instances. The sub-languages can solve standard reasoning tasks in polynomial time [HPPsR09]. OWL 2-QL enables efficient query answering over large instance population [DFH11]. OWL 2-RL is made for scalable reasoning without sacrificing too much expressive power [HPPsR09].

Objectives The main objective of this business understanding process is to create a knowledge base of the described process. A consistent vocabulary is defined, by translating knowledge of existing artifacts and collecting knowledge from domain experts. This knowledge should be usable for other project stakeholders in later stages of the data mining process.

The vocabulary used in the process is very domain specific. Many terms used by logistic domain experts are hard to understand because they are rarely used in common speech.

[1] `https://www.w3.org/TR/turtle/`: Turtle specification, last accessed 14th October, 2021

Furthermore, the knowledge is scattered over different use-case and activity diagrams. Therefore, this knowledge should be combined and be translated into an understandable and shareable knowledge base.

Additionally, all the elements interacting within the process should be identified. These include all possible configurations of shipments, such as pallets and package types, as well as all actors within the process.

Furthermore, the process can be scoped in the ontology, to define which parts are interesting and which are irrelevant. Through the level of detail of the modeling, the granularity of the process can be defined. This is essential for other project stakeholders to decide in which level of detail they are working with the process.

To fulfill all these requirements of a shareable, understandable knowledge base, the knowledge was modeled with OWL 2. OWL 2 allows formulating machine processable concepts, allows links to other ontologies and clarifies the structure of knowledge in it. By creating such an ontology, a shareable domain conceptualization is made. Furthermore, it could provide additional information for ML tasks. The ontology could also be published and reused for other projects. In this thesis it is used as knowledge base for different data generation algorithms.

Additionally, an ontology could be used for features like reasoning, however this was not the focus of the project.

Since the idea is that the knowledge can be used in later stages of the project, the ontology should be designed specifically for this. The ontology should be used by project stakeholders to understand the process. Those stakeholders are not necessarily experts in the area of ontologies. Therefore, the ontology should not contain too complex axioms, but rather be easily understandable.

Research Method The ontologies were designed in accordance with the ontology engineering process described by Kendall et al. [KM19], which is based on the popular Ontology 101 guide by Noy et al. [NM01]. An overview of the steps is depicted in Figure 3.1.

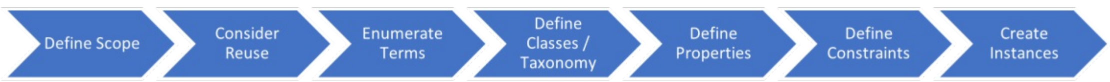


Figure 3.1: Ontology engineering process steps according to [NM01]

In detail the following steps need to be done:

Step 1. Determine the domain and scope of the ontology The first step is to decide what is included in an ontology. This depends on what's the use of the ontology.

In accordance with domain experts, it was decided that the ontology focuses on the shipment within the incoming goods process. Specifically, the scope was defined from when the shipment arrives at the processing facility until all packages and products of a shipment are ready to be stored.

The ontology should be used for business understanding and possibly be used within ML algorithms. To define the ontology, some competency questions have been defined:

- Which are possible package types within a shipment?
- Which actors are interacting with a shipment?
- Which kind of tasks accrue for a shipment and to which elements of the shipment are they linked to?

Furthermore, the aims of the ontology are defined to acquire knowledge from domain experts, make it understandable and shareable for other project stakeholders. The ontology should be used as a schema for the process. It is not the aim to use it for reasoning tasks, nor specifically checking the correctness of a shipment.

The ontology should also potentially be used with ML models. However, the focus is not to make it useable for knowledge graphs or natural language processing tasks.

Step 2. Consider reusing existing ontologies To save ontology engineering effort, ontologies should consider reusing existing ontologies [KM19].

Therefore, an analysis was conducted of existing ontologies. Common ontologies like Friend of a Friend (FOAF)[2], Dublin Core (DC)[3], Simple Knowledge Organization System (SKOS)[4] and PROV Ontology (PROV-O)[5] vocabularies as well as supply-chain ontologies (see Section 2.2) have been evaluated.

Additionally, the Linked Open Vocabulary [6] was searched for appropriate vocabulary.

However, none of these vocabularies contained vocabulary which would significantly improve the ontology. The ontologies were either too broad, i.e. common ontologies, or focused on different use cases, i.e. the supply chain ontologies.

Therefore, no existing ontology has been used.

[2] `http://xmlns.com/foaf/spec/` last accessed 14th October, 2021
[3] `https://dublincore.org` last accessed 14th October, 2021
[4] `https://www.w3.org/TR/2008/WD-skos-reference-20080829/skos.html` last accessed 14th October, 2021
[5] `https://www.w3.org/TR/prov-o/` last accessed 14th October, 2021
[6] `https://lov.linkeddata.es/dataset/lov/` last accessed 14th October, 2021

Step 3. Enumerate important terms in the ontology As a first step in the creation of the ontology, all terms were extracted from use case and flow diagrams. Additional interviews with domain experts were conducted. The aim of this step is to list all relevant terms that should appear in the ontology.

The base for this task is the business process summary, described earlier in this chapter. As a first step, nouns of the text can form a basis for classes. Verbs of this description can form the basis for property names.

Step 4. Define the classes and the class hierarchy The aim of this step is to create a class hierarchy.

We decided to use a top-down approach. In the class hierarchy, we identified the following sub hierarchies:

- PackageUnit: defines the different packages; contains several types of pallets and AKL container
- Packaging: defines the possible packaging of a *DeliveryUnit*; contains wrapping and bandages
- Task: defines an abstract task; is a hierarchical separation of the different task categories

These class hierarchies are modelled in OWL 2 as *is-a* relationships with *rdfs:subClassOf* and organized as a taxonomic hierarchy.

Step 5. Define the properties of classes In this task, the properties have been identified. The ontologies contain data as well as object properties. The properties have been defined out of the textual description and expert interviews.

In our ontologies, data properties include the descriptive properties *hasDescription*, *hasCode*, *hasName* and *hasTime*. The object properties contain all identified relationships between the classes. This includes among others *hasTask*, *containsProduct*, *hasSupplier*. The aim in the naming of the properties is to make them self-descriptive.

For easier understanding, the properties are not reflexive.

Step 6. Define constraints Only a few cardinality constraints have been modeled:

- *containsUnit* of shipment to *DeliveryUnit* has a minimal value of 1 and a maximal value 32
- *Box containsUnit* has *someValuesFrom*(i.e. at least 1) restriction
- *hasPackage* of *DeliveryUnit* has a min of 1 and a max of 32

For all other relationships no special cardinality restriction was applied.

Additionally, for all the object data properties, the *rdfs:domain* and *rdfs:range* were restricted.

Furthermore, other constraints have not been implemented to keep the ontology easily understandable.

Step 7. Create instances As a final step, an example shipment instance for the process was developed. This shipment instance was created as part of the SEMPRE-MTM ontology. Furthermore, this example shipment process was developed to test the ontology and serves as the basis for the data generation process.

3.1 Shipment Ontology - SEMPRE-CORE

SEMPRE-CORE is an ontology that describes the structure of an incoming shipment in the incoming goods process. It aims to capture all relevant shipment configurations within the process. To make the ontology usable for everybody, we refrained from using complex axioms in its design and tried to keep the ontology as simple as possible.

The practical implementation was done in Protégé [7] and is distributed as RDF triples in Turtle syntax. SEMPRE-CORE is using OWL 2-QL, because a shipment instance can have many package unit instances and OWL 2-QL is geared towards a large amount of instance data.

In the following paragraphs, we discuss a few interesting parts of the SEMPRE-CORE ontology:

Shipment The ontology consists of a central shipment class and deals with the configuration elements, i.e. which products are in which package units in the shipment, as well as the connection of a shipment to other actors like contractor, supplier and worker.

One shipment consists of at least one *DeliveryUnit*, this relationship is designed with the *containsUnit* object property. A *DeliveryUnit* has multiple *PackageUnit*s on it. A *PackageUnit* always has one *Product* type. The amount of this product is stored with the product.

Besides the pallet/package configuration, a shipment has a relationship to *Contractor*, *Supplier*, *ReceivingProcess*, *Timeslot* and *Document*.

Package units Every product is delivered and stored within a package unit. A package unit can be storable (i.e. is-a *StoragePackageUnit*) or not. Non-storable package units are of the class *DeliveryUnit*. Those include boxes, cable drums and pallets.

[7] `https://protege.stanford.edu/` last accessed 14th October, 2021

Storable package units must be a normed container compatible with an AKL. Those normed containers are called “Kleinladungsträger” (in English: Small Load Carrier) (KLT) systems and are a commonly used standard for manufacturing companies [VDA15]. Since in the observed company KLT are called after their used AKL, we settled for that naming convention for the package units. Each AKL container has a defined size and is stackable on other AKL container. On one pallet a maximum amount of each container can be stacked.

Figure 3.2 shows the recommended stacking of these containers on a pallet.

Figure 3.2: Stacked KLT container on a pallet. Image taken from [VDA15]

In SEMPRE-CORE the following *StoragePackageUnit* exist: AKLi04 (up to 1 per pallet), AKLi05 (up to 2 per pallet), AKLi11 (up to 4 per pallet), AKLi16 (up to 16 per pallet) and AKLi17 (up to 32 per pallet).

While there exist more types of package units, these selected package units have been identified by domain experts as the essential ones for the project. However, additional package types could be easily added to the ontology. They just need to have a *rdfs:subclassOf* relationship to the corresponding package unit type.

Figure 3.3 depicts the taxonomy of *PacketUnit*.

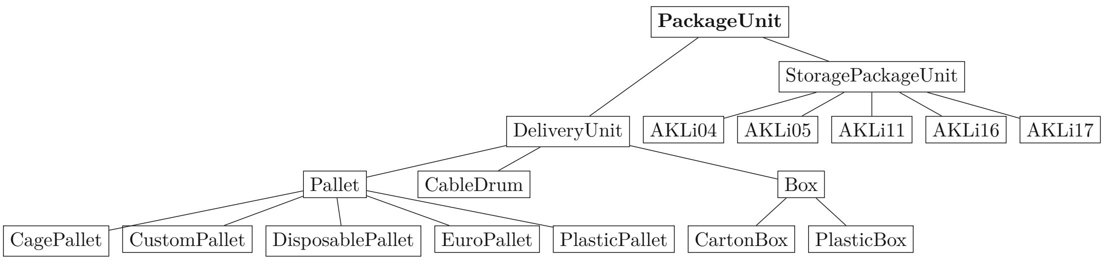


Figure 3.3: Hierarchical overview of package unit

Product A product is always part of a *PackageUnit* and has a destination *StoragePackageUnit*. Information about the product is stored with its Stock Keeping Unit (SKU) number and the product amount in the package unit.

In detail, the data property *hasSku* defines the SKU number for the product. The data property *hasAmount* defines how much products are in a package. The object property *destinationPackagedIn* is used to model the relationship to *StoragePackageUnit* as this is the package unit which is used to store the products. One product is linked to the shipment through its source *PackageUnit*.

Listing 3.1 shows an example instance of the product in Turtle syntax.

Listing 3.1: Example product instance in Turtle syntax

```
@prefix sempre: <http://dbai.tuwien.ac.at/sempre#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .

sempre:S1_P3 rdf:type owl:NamedIndividual ,
                sempre:Product ;
          sempre:hasSKU "AB123ABC";
          sempre:hasAmount 200;
          sempre:destinationPackagedIn sempre:S1_AKL1.
```

A graphical visualization of the classes and relationships of SEMPRE-CORE is shown in Figure 3.4:

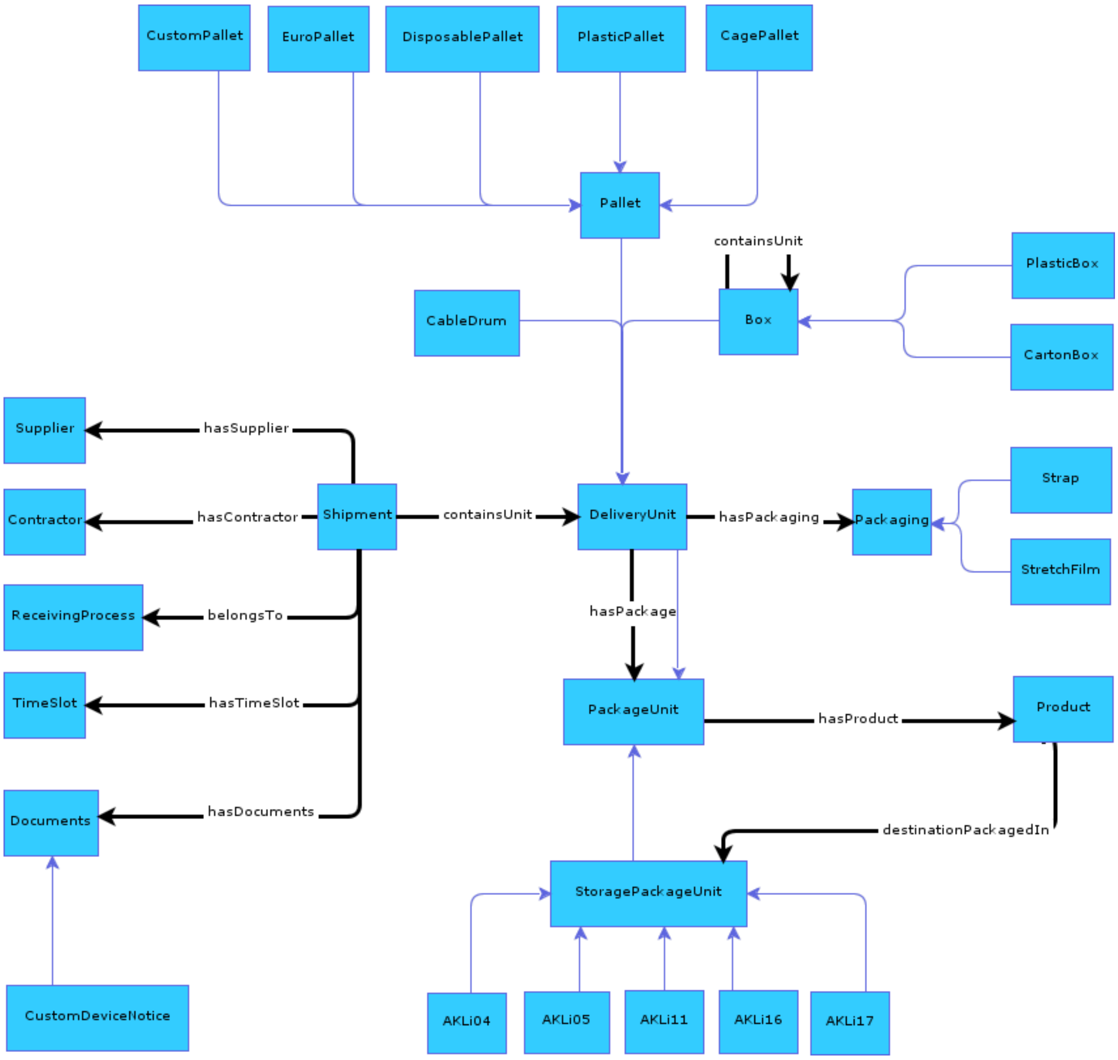


Figure 3.4: Visualization of the SEMPRE-CORE ontology

3.2 Task Extension Ontology - SEMPRE-MTM

Methods-Time Measurement (MTM) is a way to model a task by splitting it up into atomic parts and using average processing times for this atomic parts. For our process, the MTM modelling was already done by domain experts. The aim of this ontology is to link these MTM tasks with the SEMPRE-CORE ontology. With this knowledge the ontology can be used to query a time estimation of a shipment or list all tasks that need to be performed in order to process a shipment.

To include the MTM information in the ontology, we extend SEMPRE-CORE by introducing a *Task* class. First, a class for every task type has been created. Next, for every task three data properties *hasTime*, *hasCode* and *hasDescription* have been defined. These properties are restricted with the *hasValue* constraint to fixed values for each instance. Additionally, once the task has been completed by a worker, a property for *hasRealTime* can be added, to see and compare the actual time the task took with the MTM time.

Listing 3.2: Turtle formulation for task WE-04-001

```
sempre:WE-04-001 rdf:type owl:Class ;
rdfs:subClassOf sempre:WE-04 ,
            [ rdf:type owl:Restriction ;
             owl:onProperty sempre:hasCode ;
             owl:hasValue "WE-04-001"
            ] ,
            [ rdf:type owl:Restriction ;
             owl:onProperty sempre:hasDescription ;
             owl:hasValue "Baenderung von Palette entfernen"
            ] ,
            [ rdf:type owl:Restriction ;
             owl:onProperty sempre:hasTime ;
             owl:hasValue 30
            ] ;
```

Finally, these tasks are linked to SEMPRE-CORE classes, such that the existence of an instance of a SEMPRE-CORE element means that the corresponding task is created. E.g. the *stretchFilm* class is linked with the task to remove wrapping, the *EuroPallet* class is linked with the task to switch it to a storable pallet unit. This is done by using the *hasTask* relationship.

This ontology can be used to create process individuals and determine the time according to MTM measurements. If the instances for the shipment configuration exist and the corresponding MTM task instances have been created, an estimation for the total processing time can be made by summing up the *hasTime* value of all *Task* instances. This can be done, for example, with a SPARQL Protocol and RDF Query Language (SPARQL) query.

SEMPRE-MTM requires the OWL 2 profile OWL 2-RL. OWL 2-QL cannot be used because the quantification *hasValue* is used in the ontology. To get the inferred value of the *hasValue* constraints, inference through a reasoner is required.

The automatic creation of MTM task instances, when a corresponding SEMPRE-CORE instance is created, is not possible to express directly with OWL 2 axioms. It would require to create a new MTM individual and link the two individuals with the *hasTask* property. This however could be designed with more expressive languages like Semantic Web Rule Language (SWRL) rules. Another option would be to create the MTM tasks programmatically.

The complete ontology of SEMPRE-MTM and SEMPRE-CORE is visualized in Figure 3.5.

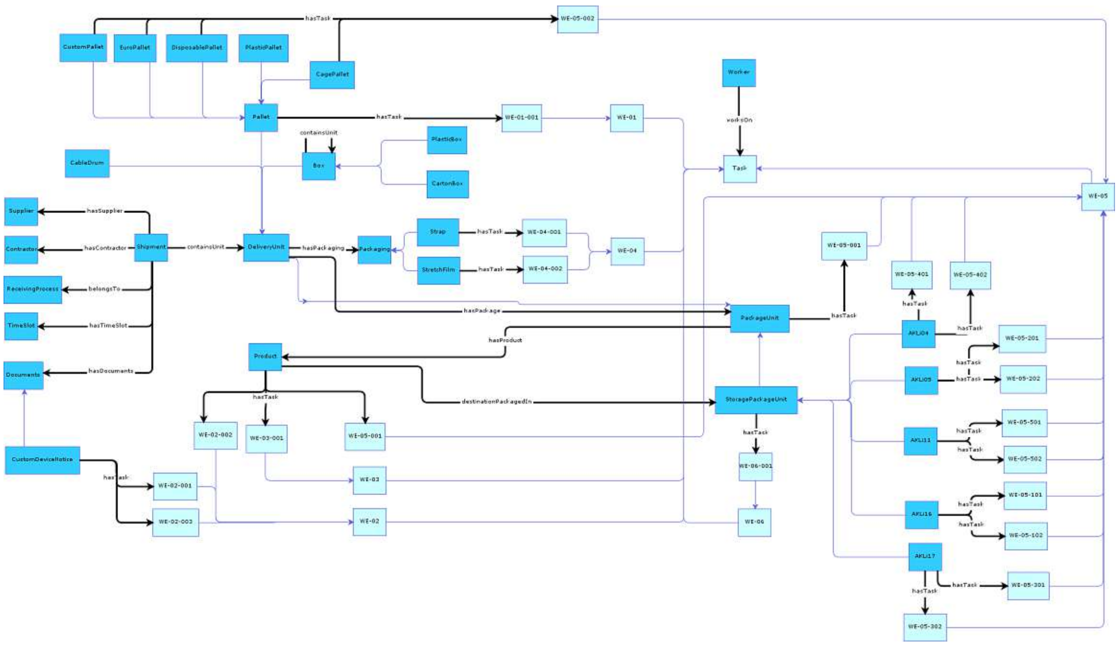


Figure 3.5: Visualization of the SEMPRE-MTM ontology

Example Process As a final step of the ontology engineering process, an example process was created. The data for the example process was randomly generated according to domain expert's assumptions. This idea of sampling synthesized data according to the shipment schema is also used for data generation in Section 4.1

The shipment process consists of a small shipment of 5 pallets and 67 packages and 18 products as well as the corresponding tasks. Three of the pallets have bandages and four of the pallets have a stretch wrapping. Four pallets have been delivered on plastic pallets, while one is a euro pallet. Of the packages, 52 are already in AKL containers, 15 packages need to be changed into them.

For this shipment, a SPARQL query was used to determine an estimation of the processing time by summing up all *hasTime* values for all connected tasks. In Listing 3.3 the SPARQL query for the example process is shown. The result of the query is the time estimation for the shipment according to MTM.

Listing 3.3: SPARQL query to get all task times for Shipment1

```
PREFIX sempre: <http://dbai.tuwien.ac.at/sempre#>

SELECT (SUM(?time) as ?totalTime) WHERE {
       sempre:Shipment1 (a|!a)* ?task .
       ?task sempre:hasTime ?time
}
```

Discussion During the development we created various versions of the ontology, at some points the ontology scope was much larger. However, many of the modeled elements did not have an impact on the processing times. In the end, we decided to focus on the elements of the specific use-case and the linking with the MTM modelling. The final ontology artifact can be used with a SPARQL query for MTM time estimation.

Unfortunately, there were no existing ontologies in the inbound logistic domain available, therefore it was not possible to link our ontology with these. Additionally, it would have been interesting to link the elements of the ontology with elements derived from Enterprise Resource Planing (ERP) systems, however no such data was available.

CHAPTER 4

Data Generation and Simulation

This chapter focuses on the generation of ML usable datasets and an explorative data analysis of the feature representation of those datasets. In specific, the following questions are answered:

- What is an appropriate way to generate a synthetic incoming goods process dataset?
- How can the correctness/generalization of this synthetic dataset be evaluated?

Based on the knowledge captured in the ontology, three synthetic datasets were created. The aim of the datasets is to offer a meaningful feature representation for shipments and be able to use these features to estimate the shipment time by solving a regression problem. This chapter describes how the synthetic datasets were created and how an analysis of the characteristics of these was performed.

The first dataset uses the ontology to create shipment instances according to the modeled MTM process. Its feature representation shows a linear relation between the features and the dependent variable. The second dataset extends the first algorithm by adding noise, randomness and an assumption that the data don't have to be in a strict linear relationship.

The third dataset uses the knowledge base from the ontology and transforms this knowledge into an ABM system. Agents are interacting according to rules and assumptions provided by domain experts. Out of these, system instances for the dataset are simulated and captured.

Motivation Since data acquisition of real-world data for this process is very resource- and time-intensive, only a few data samples can be captured. This is problematic since many ML algorithms require a large amount of data to produce robust results.

The incoming goods process is very structured and additional knowledge is available from domain experts. Thus, a high level of detail is available. This knowledge can be used to create synthetic datasets.

Synthetic datasets are commonly used to train and develop data analytics methods to generate larger data sets than available, to generate diverse data sets or to preserve anonymity in data sets [MVG19].

The aim for such a synthetic dataset is to create a feature representation $f_1, .., f_n$ such that this features can be used within a regression problem.

One way to generate a synthetic dataset is through sampling of manually defined synthetic populations. This is also called fully synthetic dataset generation [DZB18]. The assumptions about the population of random variables can be provided by domain experts. If the data can be sampled from such a population, any number of data can be generated and analyzed in detail.

Research Method The used research method is a simplified version of the three design cycle approach by Hevner [Hev07]. It consists of three cycles. The relevancy, the rigor and the design cycle. Figure 4.1 shows the ongoing flow of these cycles.

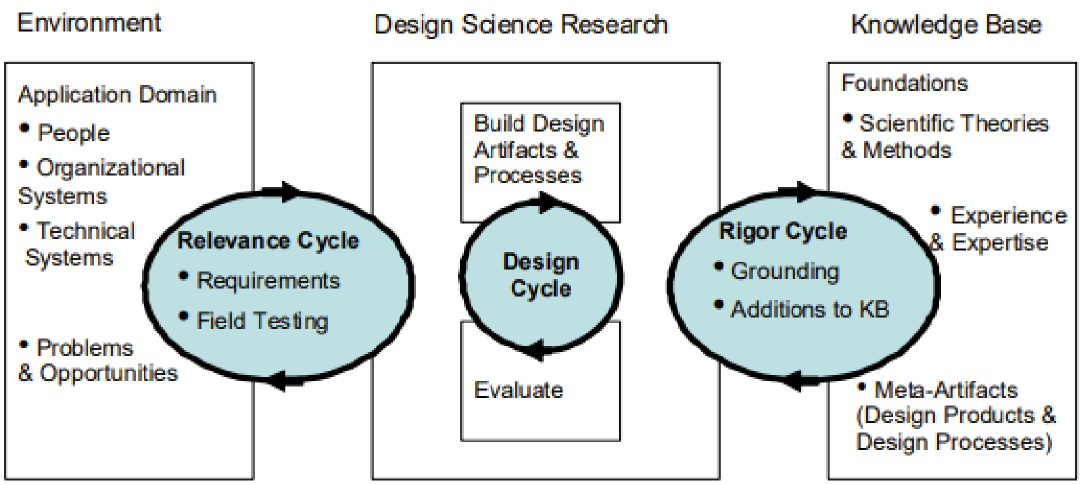


Figure 4.1: Design Science Research Cycles. Image taken from [Hev07]

Rigor Cycle The aim of this cycle is to provide past knowledge to the project. To find this knowledge, an analysis of the state of the art in the research domain is performed. This is done by performing a literature review for synthetic data generators. Additionally, this cycle also includes the knowledge from the ontology (see Chapter 3) and the specific knowledge provided by domain experts to the knowledge base.

Relevancy Cycle This cycle deals with finding the requirements and problems for the design cycle. Requirements and assumptions for the dataset are captured with structured expert-interviews.

Design Cycle This cycle has an ongoing cycle between building a model and evaluation. The model phase deals with the creation of a data generation model, based on the available knowledge and artifacts. The output of this step is a dataset which can be evaluated.

The evaluate phase deals with the verification and validation of the data. The dataset's integrity can be verified by conducting an explorative data analyses, and analyzing the correlation matrix of the dataset. The dataset can be validated by comparing the distribution of a feature with a ground truth distribution. If the results are not sufficient, additional information is required and added to the data generation model.

4.1 Linear MTM-Dataset - SEMPRE-LIN-100000

This synthetic dataset is based on the idea of creating instances according to the Sempre-MTM ontology(see Section: 3.2). Random instances of a shipment configuration are created with their corresponding MTM tasks.

The generation of the shipments are following a set of assumptions and constraints determined in accordance with domain experts:

- A shipment consists of a maximum of up to 32 pallets
- A pallet has a probability of 66% that it is wrapped
- A pallet has a probability of 66% that it has bandages
- The essential package types are i04, i05, i16, i17
- A pallet has a probability of 70% that it consists of unmixed packages
- A package has a probability of 5% to be a non-storable unit (i.e. carton package)
- Products are always in the same type of package

The pallet is either mixed or unmixed, depending on the packages on it. For unmixed pallets, the distribution between the package types can be described as:

- An unmixed pallet has a probability of 20% that it has an i04 package
- An unmixed pallet has a probability of 15% that it has i05 packages
- An unmixed pallet has a probability of 50% that it has i16 packages
- An unmixed pallet has a probability of 15% that it has i17 packages

For the mixed package types, a list of all possible configurations was created. According to the SEMPRE-CORE ontology, a maximum of 32 instances of i17 packages, 16 instances of i16 packages, 2 instances of i05 packages or 1 instance of an i04 package can be placed on a pallet. The possible configuration for a pallet consists of all possible settings (i.e. [amount_i17, amount_i16, amount_i05]) such that the sum of those packages could still fit on one pallet. For the mixed configuration, it is also possible that the pallet is not fully used.

i04 packages are excluded because due to its size this package type is only possible in an unmixed configuration. In total, 367 possible configurations of mixed pallets exist. The configuration for one pallet is then selected uniformly from all those possible configurations.

The number of products for one pallet is calculated by selecting a sample of a truncated $\Gamma_{(k=1,\theta=2)}$ distribution. This distribution is continuously falling.

This distribution ensures that most packages only have a very low number of different products. The idea to use this probability is that pallets are more probably to have multiple packages of the same product, than consisting of all different products.

The total processing time is estimated by summing up the single MTM task times of a shipment.

Methodology For the generation of one shipment, the amount of pallets is determined by taking a sample out of discrete uniform distribution $\mathcal{U}\{1, 32\}$ For each pallet a configuration is generated according to following rules:

- With a probability of 70% make the pallet unmixed otherwise make the pallet mixed
- With a probability of 66% assign the pallet a wrapping
- With a probability of 66% assign the pallet bandages
- If the pallet is unmixed assign a configuration according to the unmixed package probability
- If the pallet is mixed assign a sample out of discrete uniform distribution $\mathcal{U}\{1, 367\}$
- With a probability of 5% assign a package to a non-storable package unit
- Select the amount of different products for a pallet
- Generate MTM tasks according to SEMPRE-MTM ontology

Afterwards, an estimate of the total time for the shipment is made by summing up MTM tasks. Due to the MTM design, this estimation is a linear relationship of the shipment configuration.

According to this methodology the OWL instances are created. Figure 4.2 shows such a shipment graph with its ontology instances.

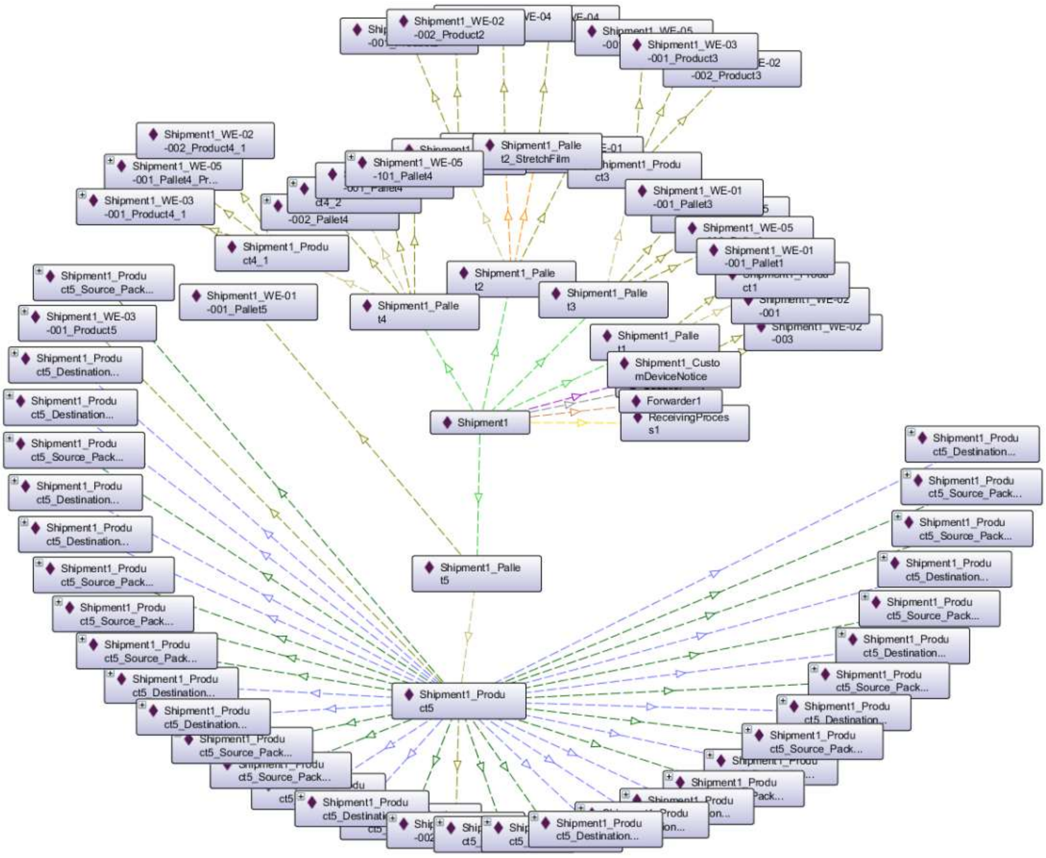


Figure 4.2: OWL instances of one shipment

Features The representation of one shipment in a graph form, has advantages in the construction, however for most ML models it is required to have a feature vector as input. To represent a shipment in such a form, quantitative measurements of the instances were selected as features. The selected meta-features count the number of corresponding instances of different package types within a shipment. Table 4.1 shows the features and an explorative analysis of the minimal value, maximal value, mean and standard deviation (std) of the feature over all 100000 instances. A more detailed description of the features can be found in Appendix A.1. The features of the dataset are linearly related to the dependent variable (i.e. total processing time).

Due to this construction a correlation exists between most features. Table 4.2 shows the correlation matrix of the dataset. The highest correlation is between amountPallets and the features amountPalletsBandaged, amountPalletsMixed since these are a subset of the amountPallets. We can verify a few basic assumptions like: *amountUnmixedSmall* has a higher correlation to *amountPalletsUnmixed* than to *amountPalletsMixed*

The integrity of all features was examined by histograms(available in Appendix A.1), by analyzing the features from Table 4.1 and the correlation matrix of the features (Table 4.2).

feature	**abbreviation**	**min**	**max**	**mean**	**std**
amountPallets	f_{l0}	1	32	16.49	9.22
amountPalletsBandaged	f_{l1}	0	30	10.88	6.38
amountPalletsWrapped	f_{l2}	0	29	10.88	6.38
amountPalletsMixed	f_{l3}	0	19	4.94	3.32
amountPalletsUnmixed	f_{l4}	0	30	11.55	6.72
amountUnmixedSmall	f_{l5}	0	352	55.35	50.33
amountUnmixedMedium	f_{l6}	0	336	92.55	60.32
amountUnmixedLarge	f_{l7}	0	26	4.61	3.81
amountUnmixedVeryLarge	f_{l8}	0	11	1.73	1.57
amountMixedSmall	f_{l9}	0	279	47.60	36.20
amountMixedMedium	f_{l10}	0	114	22.65	17.43
amountMixedLarge	f_{l11}	0	9	1.11	1.20
amountCartonSmall	f_{l12}	0	28	2.76	2.99
amountCartonMedium	f_{l13}	0	27	4.61	3.66
amountCartonLarge	f_{l14}	0	6	0.22	0.50
amountCartonVeryLarge	f_{l15}	0	3	0.08	0.29
amountDifferentProducts	f_{l16}	1	449	60.98	47.39
totalTime[sec]	f_{l17}	188.92	2208.01	867.37	399.84

Table 4.1: Feature overview of SEMPRE-LIN-100000

	f_{l0}	f_{l1}	f_{l2}	f_{l3}	f_{l4}	f_{l5}	f_{l6}	f_{l7}	f_{l8}	f_{l9}	f_{l10}	f_{l11}	f_{l12}	f_{l13}	f_{l14}	f_{l15}	f_{l16}	f_{l17}
f_{l0}	1.00	0.95	0.95	0.83	0.96	0.62	0.86	0.68	0.62	0.73	0.73	0.52	0.52	0.70	0.26	0.16	0.72	0.96
f_{l1}	0.95	1.00	0.91	0.79	0.92	0.58	0.82	0.65	0.59	0.70	0.69	0.49	0.49	0.67	0.24	0.16	0.69	0.93
f_{l2}	0.95	0.91	1.00	0.79	0.92	0.59	0.82	0.65	0.59	0.70	0.69	0.49	0.49	0.67	0.24	0.16	0.69	0.93
f_{l3}	0.83	0.79	0.79	1.00	0.64	0.41	0.57	0.46	0.41	0.89	0.88	0.62	0.34	0.47	0.17	0.11	0.61	0.90
f_{l4}	0.96	0.92	0.92	0.64	1.00	0.64	0.89	0.70	0.64	0.57	0.56	0.40	0.54	0.73	0.26	0.17	0.69	0.88
f_{l5}	0.62	0.58	0.59	0.41	0.64	1.00	0.43	0.34	0.31	0.36	0.36	0.26	0.84	0.35	0.13	0.08	0.51	0.59
f_{l6}	0.86	0.82	0.82	0.57	0.89	0.43	1.00	0.47	0.43	0.51	0.50	0.36	0.36	0.82	0.17	0.11	0.63	0.79
f_{l7}	0.68	0.65	0.65	0.46	0.70	0.34	0.47	1.00	0.34	0.40	0.40	0.29	0.28	0.38	0.38	0.09	0.45	0.60
f_{l8}	0.62	0.59	0.59	0.41	0.64	0.31	0.43	0.34	1.00	0.37	0.36	0.25	0.26	0.35	0.13	0.27	0.40	0.55
f_{l9}	0.73	0.70	0.70	0.89	0.57	0.36	0.51	0.40	0.37	1.00	0.69	0.44	0.30	0.41	0.16	0.10	0.55	0.84
f_{l10}	0.73	0.69	0.69	0.88	0.56	0.36	0.50	0.40	0.36	0.69	1.00	0.43	0.30	0.41	0.16	0.09	0.54	0.79
f_{l11}	0.52	0.49	0.49	0.62	0.40	0.26	0.36	0.29	0.25	0.44	0.43	1.00	0.22	0.29	0.10	0.07	0.36	0.52
f_{l12}	0.52	0.49	0.49	0.34	0.54	0.84	0.36	0.28	0.26	0.30	0.30	0.22	1.00	0.29	0.11	0.06	0.43	0.50
f_{l13}	0.70	0.67	0.67	0.47	0.73	0.35	0.82	0.38	0.35	0.41	0.41	0.29	0.29	1.00	0.14	0.09	0.52	0.66
f_{l14}	0.26	0.24	0.24	0.17	0.26	0.13	0.17	0.38	0.13	0.16	0.16	0.10	0.11	0.14	1.00	0.04	0.17	0.24
f_{l15}	0.16	0.16	0.16	0.11	0.17	0.08	0.11	0.09	0.27	0.10	0.09	0.07	0.06	0.09	0.04	1.00	0.11	0.16
f_{l16}	0.72	0.69	0.69	0.61	0.69	0.51	0.63	0.45	0.40	0.55	0.54	0.36	0.43	0.52	0.17	0.11	1.00	0.81
f_{l17}	0.96	0.93	0.93	0.90	0.88	0.59	0.79	0.60	0.55	0.84	0.79	0.52	0.50	0.66	0.24	0.16	0.81	1.00

Table 4.2: Correlation matrix of SEMPRE-LIN-100000 features

4.2 Nonlinear MTM-Dataset - SEMPRE-NONLIN-100000

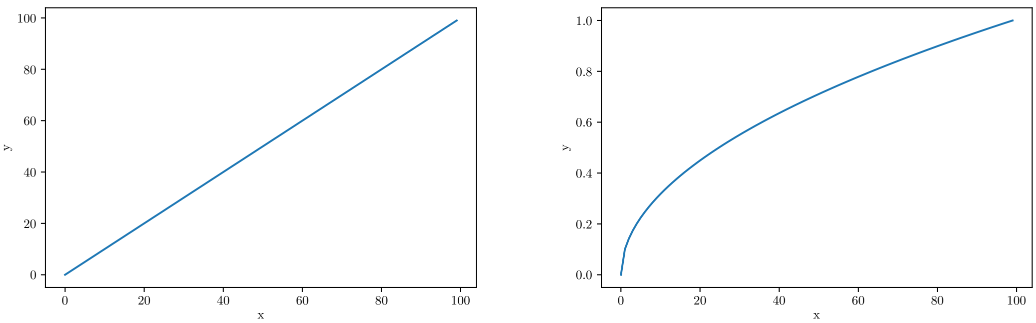
One way to increase the complexity of a synthetic dataset is to include nonlinearity of the data, noise in the inputs or noise in the target [BCSMAB13].

The SEMPRE-LIN-100000 dataset assumes the ideal case of the MTM modeling, which is not possible in reality. SEMPRE-NONLIN-100000 extends the SEMPRE-LIN-100000 such that the relation between the features cannot be represented exactly by a linear relation anymore.

To model external influences, noise and randomness are added. To the (exact) estimated value a variance of 20% is added.

To remove the strict linear relationship between features and dependent variable, the assumption was made that lower amounts have a bigger impact than higher values. The intuition behind that is that the more often the process is carried out, the closer the value comes to the standardized MTM.

To do this the features used in the MTM calculation are transformed from a linear space by the formula $f_i \to \sqrt{\frac{x}{max(x)}}$. Figure 4.3 illustrates the effect of the transformation.



(a) original value

(b) transformed value

Figure 4.3: Graphical illustration of the feature transformation

Otherwise, the configuration of a shipment is constructed in the same way as SEMPRE-LIN-100000.

Features The dataset consists of the same features as SEMPRE-LIN-100000. Besides the *totalTime* value, all features are in the same range as for SEMPRE-LIN-100000. Table 4.3 shows an explorative analysis of the features.

To verify the correctness of the dataset an explorative analysis of the features (see Table 4.3), an analysis of the correlation matrix (see Table 4.4) and histograms of the features were verified. In the correlation matrix we can verify the same basic assumptions as for SEMPRE-LIN-100000.

feature	short-name	min	max	mean	std
amountPallets	f_{n0}	1	32	16.48	9.23
amountPalletsBandaged	f_{n1}	0	29	10.87	6.38
amountPalletsWrapped	f_{n2}	0	30	10.87	6.38
amountPalletsMixed	f_{n3}	0	19	4.94	3.34
amountPalletsUnmixed	f_{n4}	0	30	11.53	6.72
amountUnmixedSmall	f_{n5}	0	352	55.56	50.47
amountUnmixedMedium	f_{n6}	0	336	92.30	60.23
amountUnmixedLarge	f_{n7}	0	26	4.61	3.83
amountUnmixedVeryLarge	f_{n8}	0	11	1.73	1.58
amountMixedSmall	f_{n9}	0	268	47.63	36.28
amountMixedMedium	f_{n10}	0	122	22.60	17.44
amountMixedLarge	f_{n11}	0	10	1.12	1.20
amountCartonSmall	f_{n12}	0	24	2.78	3.00
amountCartonMedium	f_{n13}	0	27	4.61	3.67
amountCartonLarge	f_{n14}	0	5	0.23	0.51
amountCartonVeryLarge	f_{n15}	0	3	0.09	0.30
amountDifferentProducts	f_{n16}	1	416	60.69	47.16
totalTime	f_{n17}	112.32	3385.57	1171.13	3509.92

Table 4.3: Feature overview of SEMPRE-NONLIN-100000

	f_{n0}	f_{n1}	f_{n2}	f_{n3}	f_{n4}	f_{n5}	f_{n6}	f_{n7}	f_{n8}	f_{n9}	f_{n10}	f_{n11}	f_{n12}	f_{n13}	f_{n14}	f_{n15}	f_{n16}	f_{n17}
f_{n0}	1.00	0.95	0.95	0.83	0.96	0.62	0.86	0.68	0.61	0.74	0.73	0.52	0.52	0.71	0.26	0.16	0.72	0.85
f_{n1}	0.95	1.00	0.91	0.79	0.92	0.59	0.82	0.64	0.59	0.70	0.69	0.49	0.50	0.67	0.25	0.15	0.69	0.81
f_{n2}	0.95	0.91	1.00	0.79	0.92	0.59	0.82	0.64	0.59	0.70	0.69	0.49	0.49	0.67	0.25	0.15	0.69	0.81
f_{n3}	0.83	0.79	0.79	1.00	0.64	0.41	0.57	0.45	0.41	0.89	0.88	0.62	0.35	0.48	0.18	0.11	0.60	0.84
f_{n4}	0.96	0.92	0.92	0.64	1.00	0.64	0.89	0.70	0.64	0.57	0.57	0.40	0.54	0.73	0.27	0.17	0.70	0.75
f_{n5}	0.62	0.59	0.59	0.41	0.64	1.00	0.43	0.34	0.30	0.36	0.36	0.25	0.84	0.35	0.13	0.08	0.50	0.49
f_{n6}	0.86	0.82	0.82	0.57	0.89	0.43	1.00	0.47	0.42	0.51	0.50	0.36	0.36	0.82	0.18	0.11	0.63	0.67
f_{n7}	0.68	0.64	0.64	0.45	0.70	0.34	0.47	1.00	0.33	0.40	0.40	0.28	0.28	0.39	0.38	0.09	0.45	0.52
f_{n8}	0.61	0.59	0.59	0.41	0.64	0.30	0.42	0.33	1.00	0.37	0.36	0.26	0.26	0.35	0.13	0.26	0.40	0.47
f_{n9}	0.74	0.70	0.70	0.89	0.57	0.36	0.51	0.40	0.37	1.00	0.70	0.44	0.31	0.42	0.16	0.09	0.55	0.79
f_{n10}	0.73	0.69	0.69	0.88	0.57	0.36	0.50	0.40	0.36	0.70	1.00	0.43	0.31	0.42	0.16	0.09	0.53	0.74
f_{n11}	0.52	0.49	0.49	0.62	0.40	0.25	0.36	0.28	0.26	0.44	0.43	1.00	0.21	0.30	0.11	0.06	0.36	0.47
f_{n12}	0.52	0.50	0.49	0.35	0.54	0.84	0.36	0.28	0.26	0.31	0.31	0.21	1.00	0.29	0.10	0.07	0.42	0.41
f_{n13}	0.71	0.67	0.67	0.48	0.73	0.35	0.82	0.39	0.35	0.42	0.42	0.30	0.29	1.00	0.15	0.09	0.52	0.55
f_{n14}	0.26	0.25	0.25	0.18	0.27	0.13	0.18	0.38	0.13	0.16	0.16	0.11	0.10	0.15	1.00	0.03	0.18	0.20
f_{n15}	0.16	0.15	0.15	0.11	0.17	0.08	0.11	0.09	0.26	0.09	0.09	0.06	0.07	0.09	0.03	1.00	0.10	0.12
f_{n16}	0.72	0.69	0.69	0.60	0.70	0.50	0.63	0.45	0.40	0.55	0.53	0.36	0.42	0.52	0.18	0.10	1.00	0.65
f_{n17}	0.85	0.81	0.81	0.84	0.75	0.49	0.67	0.52	0.47	0.79	0.74	0.47	0.41	0.55	0.20	0.12	0.65	1.00

Table 4.4: Correlation matrix of SEMPRE-NONLIN-100000 features

4.3 Agent-Based Simulation Dataset - SEMPRE-SIM-100000

Another possibility to create a synthetic dataset is a modeling as an agent-based simulation. ABM is a bottom-up modeling method [Bon02]. ABM allows investigations in the micro- and the macro level. The modelled agents are autonomous, heterogeneous and acting as an intelligent actor according to specified rules. In this simulation, all agents act depending on their environment and behave according to defined rules. Through the interaction of the agents' behavior can be studied over the complete system and correlation in the process is modeled implicitly.

Figure 4.4 shows an overview of the identified agents and how they are interacting. The goal of the simulation is to model the incoming goods process in detail and to generate a synthetic dataset from this simulation. Within this model, shipments can be captured in detail and different feature representations of a shipment can be created.

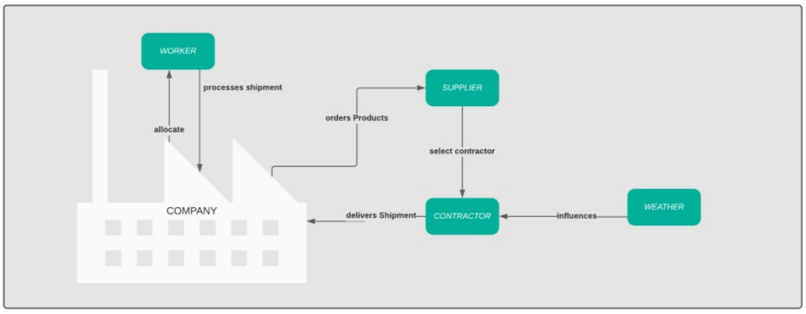


Figure 4.4: Overview of the agents in the simulation

Design Based on the developed SEMPRE-CORE ontology, the agents and configurations for the simulation were defined. Additional estimated distributions and assumptions were collected in several expert interviews with the project stakeholders and domain experts. The model was developed according to the design cycle approach through several iterations of development and evaluation. The model aims to depict the incoming goods process. A simplified version of the process that is to be examined is shown in Figure 4.5.

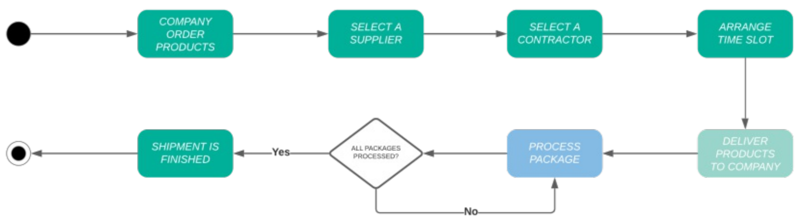


Figure 4.5: Process steps for the simulation

Within this process, the steps are executed by the different agents. The company acts as the central agent in the simulation. It orders products from a supplier and assigns a free timeslot in 3 days when the shipment should arrive. The supplier decides on the package type for the product and selects a contractor to deliver the shipment to the company. The contractor aims to deliver the shipment to the company at a random time in the given timeslot. External factors such as the weather might influence the real arrival of the contractor. Once the shipment arrives at the company, the workers process the shipments and prepare it for procurement.

Assumptions One shipment consists of up to 32 pallets. Depending on the package size, a maximum number of packages is allowed on each pallet. It is possible for a pallet to have a mixed selection of pallets or to be unmixed.

Each product is always delivered in the same packages type.

There are four different packages types:

- i17: small packages, up to 32 packages per pallet
- i16: medium packages, up to 16 per pallet
- i05: large packages, up to 2 per pallet
- i04: very large packages, up to 1 per pallet

The supplier decides whether the used packages are of plastic (i.e. storable) or carton, according to his tendency to use carton packages. If carton package are used, an additional step needs to be performed by the workers and the processing time will increase. The supplier decides randomly whether the pallet will be wrapped and/or bandaged. If there are only products of the same type on a pallet, it is a special case (i.e. unmixed pallet). These products can be stored directly and have a drastically lower processing time.

A day in the company always consists of two shifts. The first one from 6:00 to 14:00 and from 14:00 to 22:00. Every 2 hours there is a break.

The simulation runs in any number of 5 minute intervals (ticks). At the beginning of each of these intervals, the rules of all agents are executed.

Agents

Company The Company is the central actor in the simulation. There is only one company agent in each simulation. The agent orders a certain number of products from a random supplier every day. Time slots are assigned for the inbound shipment, such that only one shipment should arrive at the same time. However, if a contractor is late, multiple shipments can arrive at the same time, which could cause waiting times.

Rules:

- In each tick, there is a chance that a new shipment with random products will be ordered from a randomly selected supplier

Supplier The supplier is responsible for selecting the appropriate packaging for the products. While the packaging size is fixed for each product, the package type depends on the supplier. The supplier decides whether he uses a non-storable package (i.e. carton package) or a storable package according to his *TENDENCY_CARTON*. Additionally, he selects whether the pallet is equipped with additional packaging material (i.e. wrapping, banding) according to *TENDENCY_BANDAGES* and *TENDENCY_WRAP*.

The supplier is also responsible to select a contractor for the shipment. It is assumed that the supplier always has all products in stock. In the simulation, there can be any amount of suppliers, but at least one.

Each supplier has the following properties:

- *NAME*: Identifies the supplier
- *TENDENCY_CARTON*: Describes how likely it is that the supplier uses carton packages instead of plastic
- *TENDENCY_BANDAGES*: Describes how likely it is that the supplier will use a banding
- *TENDENCY_WRAP*: Describes how likely it is that the supplier will use a wrapping

Each supplier is acting after the following rules:

- When new products are ordered, assign packaging to the shipment and select a contractor for delivery

Contractor The contractor is responsible for delivering the products within a specified time slot. He always tries to fulfill this as good as possible, but can be delayed by his *PROBABILITY_LATE* or influenced by bad weather conditions. In the simulation, there can be any number of contractors, but at least one.

Properties:

- *NAME*: Identifies the contractor
- *PROBABILITY_LATE*: Describes the probability that the contractor will not deliver in the specified time slot

Rules:

- Deliver shipment within random range of the time-slot

Worker Employees are responsible for processing the incoming shipments. Each pallet and each package is processed individually. If the pallet is an unmixed configuration of storable containers, the processing of the individual packages is skipped. The actual time for processing a package is calculated from the employee *EFFICIENCY*, the package type, the similarity of the products, whether there are bandages or wrapping on the pallet and a small random markup to represent non-modeled factors. An employee has a status. If the status is available the worker might process packages from the queue. Every two hours, his status changes to break mode for 5 minutes, and every 8 hours, his shift ends.

There must be at least two to any number of workers in the simulation.

Properties:

- *NAME*: identifies the worker
- *EFFICIENCY*: describes how efficiently he can process a task; Influences the time needed for a package
- *WORKER STATE*: describes the current state of the worker

Rules:

- Every 2 hours, take a break for 5 minutes
- Every 8 hours, end the shift or start a new shift
- As long as packages are in the processing queue, process them

Weather Weather affects the actual time when a shipment arrives. If the weather is bad, there may be a delay in arrival. The weather acts according to a Markov model [RN20]. Each transition depends only on the last state. In each time interval there is a chance for a transition. There are four different states: SUN, CLOUD, RAIN and SNOW.

Figure 4.6 illustrates the states and transition probabilities.

Implementation The implementation was done as a standalone python microservice. The simulation allows setting different parameters and to run for an arbitrary period of time. All times of the simulation are stored in the database and can be called later.

The following parameters can be set for the simulation:

- Average time needed to process an i04, i05, i16, i17 package
- Probability of unmixed pallets

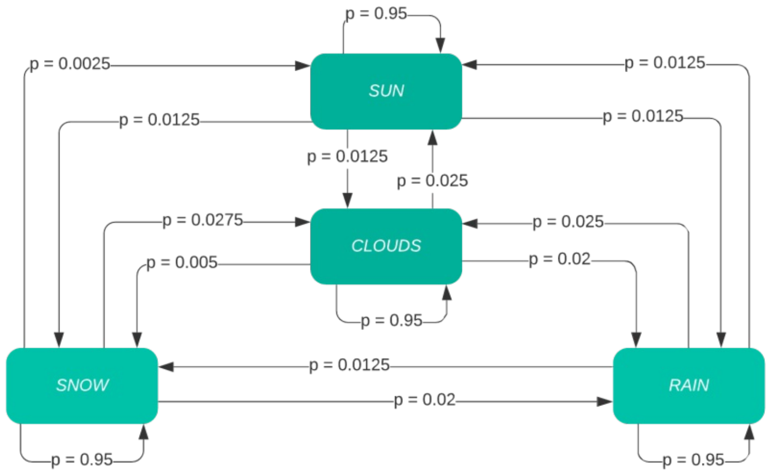


Figure 4.6: Different states of weather and their transition probabilities shows these

- How many shipments a company has on average per day
- How many products a shipment has on average
- List of workers; List of contractors; List of suppliers; List of products

To dynamically configure the simulation parameters, analyze states of the simulation and verify that the simulation is working correctly, a web application was implemented. The communication between the different microservices is done by message queue. Figure 4.7 shows an architectural overview of this web-application.

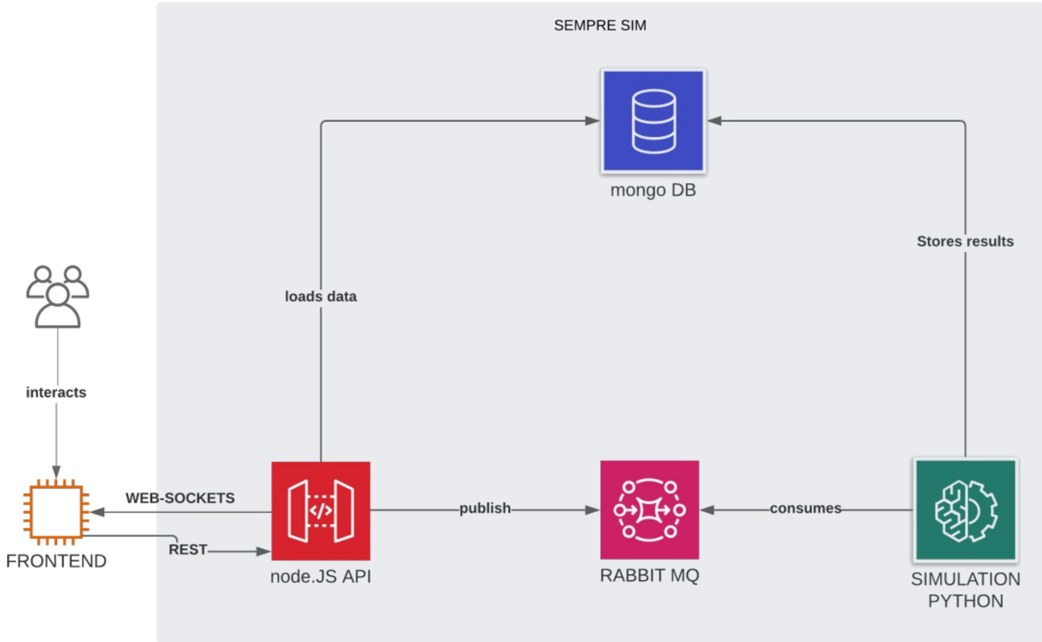


Figure 4.7: Architectural overview of the web application

Feature Construction In the simulation, a shipment is associated with information about the workers, the shipment queue and the shipment data including package and product information.

However, since not all information is known in real applications, some information was abstained and the features were limited to data that is also available in real applications. Therefore, the feature representation focuses on the different type of packages and products. Packages and products are stored in array structure and have a dynamical length. To represent this dynamical array, meta features about the packages were created. Table 4.5 shows the selected features and their statistical properties.

Table 4.6 shows the correlation matrix between the features. We can again see that the *amountPalletsWrapped*, *amountPalletsBandages* and *amountPallets* have a very high correlation. We can also observe that the amount of packages, correlates with the amount of carton container for a specific size.

These features have been selected to correspond to the features from SEMPRE-LIN-100000 and SEMPRE-NONLIN-100000.

feature	**short-name**	**min**	**max**	**mean**	**std**
amountPallets	f_{s0}	1.00	32.00	16.51	9.23
amountPalletsBandaged	f_{s1}	0.00	29.00	11.06	6.47
amountPalletsWrapped	f_{s2}	0.00	29.00	11.06	6.47
amountUnmixed	f_{s3}	0.00	32.00	13.13	10.09
amountProducts	f_{s4}	1.00	50.00	20.68	13.18
amountPackages	f_{s5}	0.00	50.00	19.07	12.35
amountSmall	f_{s6}	0.00	23.00	4.48	3.62
amountMedium	f_{s7}	0.00	19.00	2.35	2.28
amountLarge	f_{s8}	0.00	19.00	2.81	2.63
amountVeryLarge	f_{s9}	0.00	31.00	7.34	5.35
amountSmallCarton	f_{s10}	0.00	18.00	2.57	2.47
amountMediumCarton	f_{s11}	0.00	15.00	1.35	1.61
amountLargeCarton	f_{s12}	0.00	25.00	4.90	3.86
amountVeryLargeCarton	f_{s13}	0.00	25.00	4.21	3.59
totalTime	f_{s14}	50.14	3,805.30	1,446.00	863.92

Table 4.5: Feature overview for SEMPRE-SIM-100000

	f_{s0}	f_{s1}	f_{s2}	f_{s3}	f_{s4}	f_{s5}	f_{s6}	f_{s7}	f_{s8}	f_{s9}	f_{s10}	f_{s11}	f_{s12}	f_{s13}	f_{s14}
f_{s0}	1.00	0.96	0.96	0.62	0.00	0.06	0.03	0.06	0.02	0.06	0.02	0.05	0.03	0.05	0.87
f_{s1}	0.96	1.00	0.91	0.59	0.00	0.05	0.03	0.06	0.02	0.06	0.02	0.04	0.03	0.05	0.83
f_{s2}	0.96	0.91	1.00	0.59	0.00	0.05	0.03	0.06	0.02	0.06	0.02	0.04	0.03	0.05	0.83
f_{s3}	0.62	0.59	0.59	1.00	0.38	0.57	0.44	0.40	0.38	0.52	0.37	0.32	0.45	0.44	0.89
f_{s4}	0.00	0.00	0.00	0.38	1.00	0.85	0.71	0.54	0.60	0.74	0.60	0.44	0.71	0.63	0.22
f_{s5}	0.06	0.05	0.05	0.57	0.85	1.00	0.81	0.67	0.69	0.89	0.68	0.54	0.82	0.76	0.35
f_{s6}	0.03	0.03	0.03	0.44	0.71	0.81	1.00	0.45	0.48	0.59	0.84	0.36	0.56	0.51	0.26
f_{s7}	0.06	0.06	0.06	0.40	0.54	0.67	0.45	1.00	0.38	0.48	0.38	0.81	0.45	0.41	0.26
f_{s8}	0.02	0.02	0.02	0.38	0.60	0.69	0.48	0.38	1.00	0.51	0.46	0.36	0.84	0.51	0.23
f_{s9}	0.06	0.06	0.06	0.52	0.74	0.89	0.59	0.48	0.51	1.00	0.50	0.39	0.61	0.85	0.32
f_{s10}	0.02	0.02	0.02	0.37	0.60	0.68	0.84	0.38	0.46	0.50	1.00	0.35	0.47	0.49	0.22
f_{s11}	0.05	0.04	0.04	0.32	0.44	0.54	0.36	0.81	0.36	0.39	0.35	1.00	0.36	0.39	0.21
f_{s12}	0.03	0.03	0.03	0.45	0.71	0.82	0.56	0.45	0.84	0.61	0.47	0.36	1.00	0.52	0.27
f_{s13}	0.05	0.05	0.05	0.44	0.63	0.76	0.51	0.41	0.51	0.85	0.49	0.39	0.52	1.00	0.28
f_{s14}	0.87	0.83	0.83	0.89	0.22	0.35	0.26	0.26	0.23	0.32	0.22	0.21	0.27	0.28	1.00

Table 4.6: Correlation matrix for SEMPRE-SIM-100000 features

Product Representation Since the simulation provides more detailed information than the other data generators, it is possible to construct more advanced features. In this experiment, only information which is available at the time of the order is used. This information includes an array of products and their ordered amount. To vectorize this array an encoding, containing all products(i.e. 50 products in the simulation) and the amount as value, was created. Table 4.7 shows the structure and the features for this dataset.

feature	min	max	mean	std
P_0	0.00	13,065.00	2,987.28	4,388.05
P_1	0.00	19,498.00	4,515.67	6,610.91
P_2	0.00	13,725.00	3,114.49	4,587.70
P_3	0.00	14,562.00	3,103.93	4,589.77
P_4	0.00	7,531.00	1,666.29	2,453.51
P_5	0.00	21,416.00	4,894.84	7,229.79
...	...	...	...	...
P_45	0.00	24,930.00	5,606.67	8,259.57
P_46	0.00	4,423.00	991.25	1,457.57
P_47	0.00	20,666.00	4,689.23	6,905.54
P_48	0.00	1,568.00	340.64	501.70
P_49	0.00	4,489.00	1,018.09	1,492.07
totalTime	50.14	3,805.30	1,446.00	863.92

Table 4.7: Explorative analysis for SEMPRE-SIM-100000 product feature representation

4.4 Data Validation

To validate the results, the dependent variable, i.e. the *totalTime* variable, from the synthetic dataset is compared to a ground truth distribution. Unfortunately, not enough captured data was available to perform this test with real data, as a substitute a distribution provided by domain experts is used for this experiment.

To ensure that the distribution of the synthetic dataset matches the real data, the data samples of the two distributions were tested with a Kolmogorov-Smirnov test (KS-test) [Mas51]. The null hypothesis for this statistical test says that samples of the *totalTime* variable from the synthetic dataset match samples of the same variable from the real dataset. The statistical significance can be tested with a two sample test [Hod58]. This test results in a p-value, the closer this value is to 0 the more statistic significance is there to reject the null hypothesis. In the experiment, the null hypothesis is rejected if the p-value is smaller than 0.01. If the null hypothesis is rejected, the simulation was rerun with new parameters, until the resulting synthetic data was accepted by the KS-test.

Figure 4.8 illustrates this process.

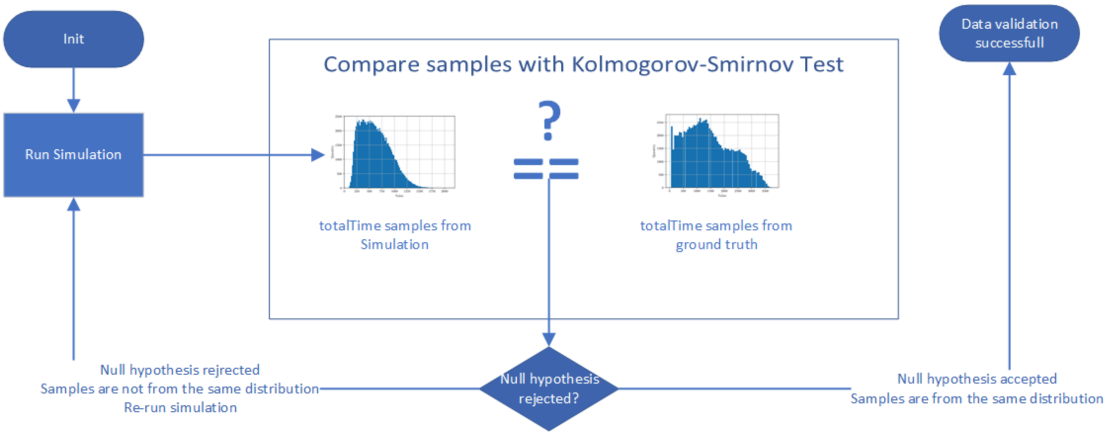


Figure 4.8: Process for data validation

CHAPTER 5

Data Analytics

In this chapter the results of the ML models are presented and evaluated. In specific, the following questions will be answered:

- Which are the key features for developing a prediction model for a given shipment?
- What is a good ML model to predict the required time for a given shipment? How do the results differ between an AutoML model and handcrafted models?

At the beginning of this chapter, the importance of the features is analyzed by evaluating their correlation coefficients, analyzing the structure of decision trees and analyzing their linear coefficients.

In the next section, the hyperparameters of the selected models are optimized for each dataset and subset. The parameter-space was searched using grid search and 5-fold cross validation.

The tuned models are then compared in the subsequent section. The models are evaluated on different subsets of the three datasets. For the experiment the whole dataset and subsets of 50000 samples, 20000 samples, 10000 samples, 5000 samples, 2000 samples, 1000 samples and 500 samples were used. The datasets were divided into 55-20-25, training/validation/testing for the experiments. For all experiments the RMSE serves as an evaluation measure.

The results of the different models are evaluated regarding their robustness and performance. The aim is to compare different models in particular with respect to their behavior on the different datasets. Furthermore, the handcrafted models are compared with an AutoML build model. During this evaluation the created AutoML model is analyzed how it has been constructed.

Lastly, the different feature representations, i.e. product feature representation and meta feature presentation, of SEMPRE-SIM-100000 are evaluated.

5.1 Feature Importance

Feature importance is a common explainability technique for ML [BXS+20]. It refers to techniques to assign scores for a single feature. Feature importance scores might help to better understand the data, better understand the impact of each feature, better understand a model or reducing the number of features.

There exist various ways to model such an importance metric. In this section, three techniques are applied to the datasets and their results are analyzed. The aim is to gain insights of which features are more important features and which features have a lesser impact. Furthermore, these results can be used for feature selection to reduce the input feature space. However, since the feature space of the datasets is already low-dimensional, the focus is not on feature selection but more on understanding the impact of the features.

The first feature importance metric analyses the statistical significance of the single features. For the second metric, a decision tree regression model is build. The decision tree splits the dataset into partitions, depending on how often a feature is used in a predictor the Mean Decrease in Impurity (MDI) is calculated. The third feature importance metric uses Lasso Regression and calculates its linear coefficient.

Correlation Analysis One measurement to check the statistical significance of a feature variable is by measuring its correlation with the dependent variable (i.e. *totalTime* variable). The correlation between the features and the dependent variable *totalTime* is examined.

Since both the features and the dependent variable are linear variables, the correlation can be calculated with the Pearson correlation coefficient. Its calculation is shown in Equation 5.1.

$$Cor(X,Y) = \frac{Cov(X,Y)}{\sqrt{Var(X)}\sqrt{Var(Y)}} \tag{5.1}$$

The values for the correlation coefficient are in $[-1, 1]$, where values close to $+1$ indicate a high linear relationship and values close to 0 indicate low or no linear relationship between the variables. Values close to -1 indicate a high negative linear relationship.

A high correlation between the variables might indicate that the feature has a high impact. However, this analysis only takes the correlation between a single variable and the dependent variable into account, while the real importance of a feature in ML models also depends on the combination with other features.

Decision Tree Decision trees partition the data with predictors into smaller groups, such that they are more homogeneous with respect to the response [KJ13]. A predictor is a set of rules for feature variables containing if statements to split the data.

There are different ways to construct a decision tree and ways to optimize the selected predictors for this split. A typical method is the Classification And Regression Tree (CART) method [BFOS84]. It iterates the dataset over all possible predictors and searches for the one that minimizes a condition. Typical conditions for regression are Mean Squared Error (MSE), half Poisson deviance or Mean Absolute Error (MAE). For our experiments, MSE was used.

By repeating this task for every sub-partition, a tree can be created. To ensure that the tree is not too big, a minimal cost-complexity pruning is performed.

For a decision tree a feature importance can be calculated as Mean Decrease in Impurity (MDI) [LWSG13]. This feature importance is calculated by the sum of all decreases in the tree due to a given variable, normalized by the number of trees [LWSG13].

For this MDI Scornet [Sco20] proved that if input variables are independent and in absence of interactions, MDI provides a variance decomposition of the output.

Linear Coefficients Linear models search for linear coefficients such that they can be used to explain the dependent variable. If the input variables are in the same scale, e.g. normalized, the size of such a coefficient can indicate the impact of a feature.

For our experiments we decided to use Lasso Regression because its regularization, tends to reduce the coefficients of non-relevant features towards 0.

SEMPRE-LIN-100000 As a first test the correlation between the dependent and the independent variables was analyzed. Table 5.1 shows these results. The highest value is the *amountPallets*, followed by *amountPalletsBandaged* and *amountPalletsWrapped*. The features *amountPalletsMixed*, *amountPalletsUnmixed*, *amountDifferentProductsPerPallet* and *amountMixedSmall* also have a very high correlation. In general, every feature has a significant correlation to the dependent variable. Thus, indicating that every feature is important and no feature should be removed. The lowest correlation is for *amountCartonLarge* and *amountCartonVeryLarge*, one reason for the low correlation could be the low number of occurrences of this package types within a shipment.

The trained decision tree has a depth of 17. The predictors which split the data into the biggest partitions are on top. Figure 5.1 illustrates the uppermost leaves of the tree. According to the tree the most splitting decision criteria is whether the shipment consists of more or less than 17 pallets. In the next layer, the same type of decision criteria can be found. The tree is split again by the amount of pallets evenly. This indicates that the *amountPallets* is the most significant feature to partition the data. In Table 5.1 this can also be seen with the very high MDI value of *amountPallets*.

The linear coefficients of the Lasso Regression for SEMPRE-LIN-100000 are also shown in Table 5.1. The highest values are for the features *amountMixedSmall*, *amountPallets* and *amountDifferentProductsPerPallet*. Thus, a high value in one of these features indicates a high *totalTime*. The feature amount unmixed features have a value close to 0, indicating

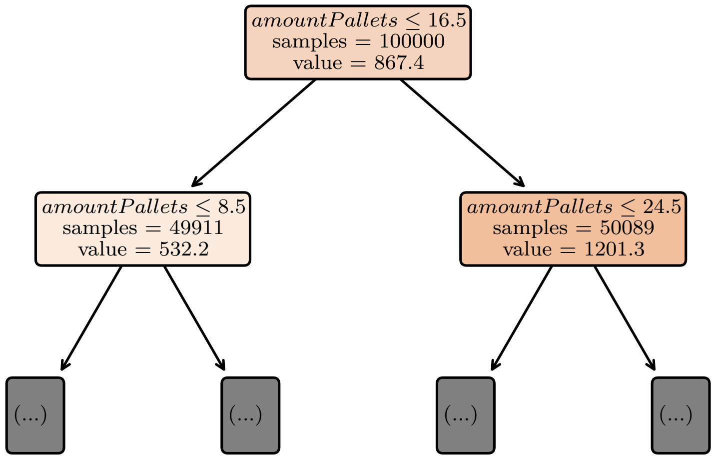


Figure 5.1: First leaves for a decision tree for SEMPRE-LIN-100000

feature	linear coefficient	correlation	MDI
amountPallets	700.42	0.96	0.92
amountPalletsBandaged	195.65	0.93	0.00
amountPalletsWrapped	239.24	0.93	0.00
amountPalletsMixed	-223.40	0.90	0.00
amountPalletsUnmixed	-351.23	0.88	0.00
amountUnmixedSmall	-0.00	0.59	0.00
amountUnmixedMedium	-0.06	0.79	0.00
amountUnmixedLarge	-0.08	0.60	0.00
amountUnmixedVeryLarge	-0.07	0.55	0.00
amountMixedSmall	753.80	0.84	0.04
amountMixedMedium	328.70	0.79	0.01
amountMixedLarge	28.39	0.52	0.00
amountCartonSmall	122.12	0.50	0.00
amountCartonMedium	149.33	0.66	0.00
amountCartonLarge	63.17	0.24	0.00
amountCartonVeryLarge	46.69	0.16	0.00
amountDifferentProductsPerPallet	812.60	0.81	0.03

Table 5.1: Different feature importance metrics for SEMPRE-LIN-100000. The three highest values for each indicator are highlighted grey

they don't have a lot of impact on dependent variable. Since an almost ideal solution for this dataset can be found with linear regression, the coefficients are directly related to the modeled MTM times.

A few of the coefficients are negative, this would mean that an increase of the feature would decrease the *totalTime*. This would not make sense the way the features are modeled. We assume that the negative value corresponds to the correlation between the variables. Table 4.2 shows that the negative variables have a high correlation with other variables.

SEMPRE-NONLIN-100000 As a second, SEMPRE-NONLIN-100000 was analyzed. It differs from SEMPRE-LIN-100000, in the way that it adds noise, randomness and non-linearity to the calculation of *totalTime*.

First the correlation matrix was analyzed. Table 5.2 shows the Pearson correlation coefficients of the variables and *totalTime* variable. The highest correlation is again for the features *amountPallets*, *amountPalletsBandaged* and *amountPalletsWrapped* however due to the noise their correlation coefficient is lower. Additionally, the feature *amountPalletsMixed* has for this dataset a slightly higher value. Besides that, the correlation coefficients only slightly differ from SEMPRE-LIN-100000 results.

The constructed decision tree for this dataset has a depth of 39 levels. Figure 5.2 shows the highest leaves in the created tree. Similar to SEMPRE-LIN-100000 the first decision criteria is the *amountPallets*. For the subsequent leaves, one leaf decision criteria uses the *amountPalletsMixed* feature. This coincides with the result from the correlation analysis, that *amountPalletsMixed* is more important in this dataset.

The linear coefficients show the three of the amount unmixed variables have a coefficient of 0. This indicates that they don't have any impact on the dependent variable. The highest coefficient values are again for the same variables as in the SEMPRE-LIN-100000 dataset: *amountPallets*, *amountMixedSmall*, *amountMixedMedium* and *amountDifferent-ProductsPerPallet*.

The *amountDifferentProductsPerPallet* has a slightly lower coefficient. One reason for this might be the added non-linearity.

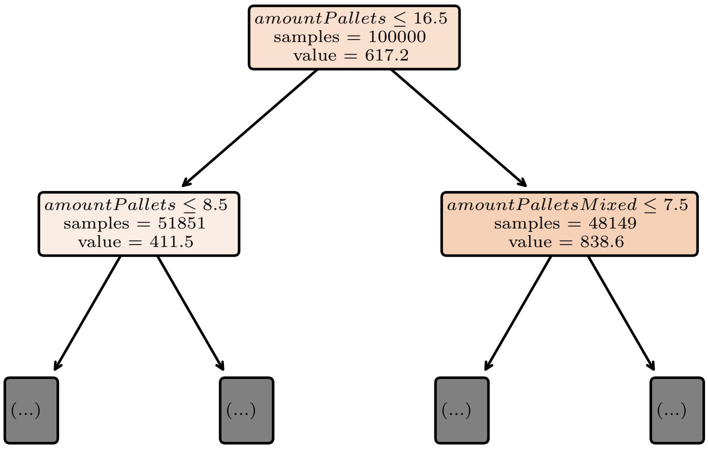


Figure 5.2: First leaves for a decision tree for SEMPRE-NONLIN-100000

feature	linear coefficient	correlation	MDI
amountPallets	331.48	0.84	0.66
amountPalletsBandaged	43.82	0.81	0.02
amountPalletsWrapped	60.45	0.81	0.02
amountPalletsMixed	-0.00	0.83	0.09
amountPalletsUnmixed	0.67	0.74	0.01
amountUnmixedSmall	11.09	0.48	0.01
amountUnmixedMedium	0.00	0.66	0.01
amountUnmixedLarge	0.00	0.51	0.01
amountUnmixedVeryLarge	0.00	0.47	0.01
amountMixedSmall	662.73	0.80	0.07
amountMixedMedium	323.19	0.74	0.03
amountMixedLarge	22.61	0.47	0.01
amountCartonSmall	8.34	0.40	0.01
amountCartonMedium	34.13	0.55	0.02
amountCartonLarge	9.97	0.20	0.00
amountCartonVeryLarge	9.45	0.13	0.00
amountDifferentProductsPerPallet	169.69	0.65	0.03

Table 5.2: Different feature importance metrics for SEMPRE-NONLIN-100000. The three highest values for each indicator are highlighted grey

SEMPRE-SIM-100000 As a third dataset, SEMPRE-SIM-100000 was analyzed. Compared to the other datasets, it has a different feature set with detailed information about the amount of mixed/unmixed pallets missing.

First, the correlation coefficients were analyzed. Table 5.3 shows the results. Again, the highest correlation is found in the features *amountPallets*, *amountPalletsWrapped* and *amountPalletsBandaged*. These three features have the highest correlation coefficient, with a significant gap to the other features. However, no feature has a correlation value close to 0, indicating that it could be removed.

Next, a decision tree for the dataset build. It consists of 55 levels. Figure 5.3 shows the two upper leaves for the decision tree. The first decision criteria is again splitting by *amountPallets*. In the second level, one decision criteria takes the *amountPallets* feature, while the other takes *amountPackages* feature.

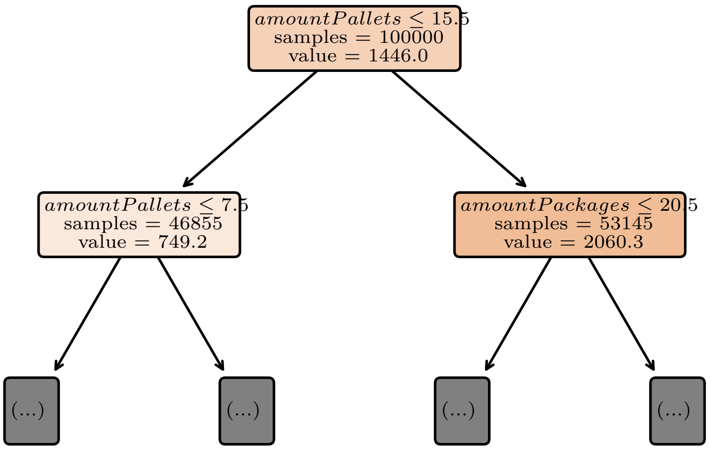


Figure 5.3: First leaves for a decision tree for SEMPRE-SIM-100000

Analyzing the linear coefficient, it's interesting to see that *amountPallets*, *amountPackages* have very high coefficients and *amountProducts* has a high negative coefficient.

In the correlation matrix of the dataset in Table 4.6, we can see a very high correlation between *amountProducts* and *amountPackages*. Therefore, we assume that this correlation influences the coefficients.

feature	linear coefficient	correlation	MDI
amountPallets	2444.93	0.87	0.74
amountPalletsBandaged	25.03	0.83	0.00
amountPalletsWrapped	-2.11	0.83	0.00
amountProducts	-498.50	0.22	0.01
amountPackages	1512.83	0.35	0.22
amountSmall	-9.92	0.26	0.00
amountMedium	16.92	0.26	0.00
amountLarge	-23.07	0.23	0.00
amountVeryLarge	0.57	0.32	0.00
amountSmallCarton	2.89	0.22	0.00
amountMediumCarton	6.96	0.21	0.00
amountLargeCarton	7.11	0.27	0.00
amountVeryLargeCarton	30.17	0.28	0.00

Table 5.3: Different feature importance metrics for SEMPRE-SIM-100000. The three highest values for each indicator are highlighted grey

5.2 Regression Algorithms and Hyperparameter Evaluation

To determine the optimal parameter for each of the models grid search with 5 fold cross validation was used. The training and test data was split into 75% trainings data and 25% test data. For each model, except linear regression, a parameter space was defined.

As scoring metric, the RMSE was used. The RMSE metric is a measure for the difference between the predicted value and the real value. It is calculated by taking the square root of the MSE. The MSE is calculated by summing up all squared residuals, i.e. the difference between predicted value and real value. Equation 5.2 shows this calculation

$$RMSE = \sqrt{\text{MSE}} = \sqrt{\sum_{t=1}^{T} \frac{(\hat{y}_t - y_t)^2}{T}} \tag{5.2}$$

The advantage of the RMSE compared to the MSE is that the values are in the same unit as the original data [KJ13].

Following we discuss the models and the selected parameter-space for each model:

Linear Regression Linear models can model the relationship of a dependent variable y and some features x A linear model can be written in the form of:

$$y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_p * x_p + \epsilon \tag{5.3}$$

where β_i indicates the learned coefficients and ϵ indicates the error term, i.e. the different between prediction and actual outcome. The objective is to find parameters that minimizes the Sum of Squared Error (SSE) between the observed and predicted response [KJ13]:

$$SSE = \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2 \tag{5.4}$$

This difference between expected and predicted variable is also called residual. The coefficients of a linear regression can be calculated as:

$$\hat{\beta} = (X^T X)^{-1} XTy \tag{5.5}$$

where X is a matrix of the input features.

The advantage of linear regression models is that they are easy to computer and that they are interpretable with their coefficients. For the linear regression model no parameter must be chosen.

Lasso Regression For two reason least square estimation is not satisfying. First, least square estimate often might have a small bias but a high variance, thus overfitting the data. Secondly, if there are a large amount of features it would make sense to eliminate features that are non-relevant.

To overcome this problems Lasso Regression was introduced [Tib96].

It introduces a regularization term to the SSE. The Lasso Regression SSE can be calculated as:

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{P} |\beta_j| \tag{5.6}$$

This regularization term serves as a penalty for to large coefficients. It ensures that coefficients are only allowed to become large if there is a proportional reduction in SSE. In effect, this method shrinks the estimates towards or even exactly to 0 as the λ penalty becomes large [KJ13].

Parameters The tunable hyperparameter for Lasso Regression is α or λ the constant that is multiplied by the regularization term. Possible values for this paramater are between 0 and ∞. 0 means the minimalization is the same as in ordinary linear regression.

The following parameter space was selected:

- λ: $[10^{-15}, 10^{-10}, 10^{-8}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 1, 5, 10]$

Table 5.4 shows the best parameters found in the various dataset configurations. An interesting point aspect in the is that for SEMPRE-LIN-100000 the regularization parameter λ tends to be very small, while for SEMPRE-NONLIN-100000 and SEMPRE-SIM-100000 this parameter tends to be big. Thus indicating the need of regularization.

Dataset	**Size**	**Best λ**
SEMPRE-LIN-100000	100000	0.0001
SEMPRE-LIN-100000	50000	0.0001
SEMPRE-LIN-100000	10000	0.0001
SEMPRE-LIN-100000	5000	0.0001
SEMPRE-LIN-100000	2000	0.0001
SEMPRE-LIN-100000	1000	1e-08
SEMPRE-LIN-100000	500	1e-15
SEMPRE-NONLIN-100000	100000	0.01
SEMPRE-NONLIN-100000	50000	0.01
SEMPRE-NONLIN-100000	10000	0.01
SEMPRE-NONLIN-100000	5000	0.01
SEMPRE-NONLIN-100000	2000	1
SEMPRE-NONLIN-100000	1000	1
SEMPRE-NONLIN-100000	500	1
SEMPRE-SIM-100000	100000	0.01
SEMPRE-SIM-100000	50000	0.01
SEMPRE-SIM-100000	10000	0.01
SEMPRE-SIM-100000	5000	0.01
SEMPRE-SIM-100000	2000	1
SEMPRE-SIM-100000	1000	1
SEMPRE-SIM-100000	500	1

Table 5.4: Best Lasso Regression parameter for different datasets and dataset sizes

k-NN Regression The k-NN Regression predicts a sample using the k-closest samples from the training set. For regression, k-NN calculates the mean of those closest samples and sets it as its predictor.

To determine which samples are the closest, their distances are measured with the Minkowski distance metric. The following equation shows this metric:

$$(\sum_{i=1}^{n} |x_i - y_i|^q)^{\frac{1}{q}} \tag{5.7}$$

If the parameter q of the Minkowski distance is set to 2, the formula is also called Euclidean distance. In our experiments, the Euclidean distance has been used.

Parameter The tunable parameters for k-NN Regression are the number of neighbors and the weights of the neighbors. This number has to be an odd number greater than 1. The weights for the neighbors can be either uniform distributed or dependent on the distance of the data points.

The following parameter space was selected:

- k: $[1, 3, 5, 7, 11, 15]$
- $weights$: $['uniform', 'distance']$

Table 5.5 shows the best parameters found in the various dataset configurations. For most datasets, the ideal number of neighbors is 11, which is the second highest parameter. For the smaller datasets, i.e. 500 sample subset, the optimal k tends to be smaller.

Dataset	Size	Best k	Best weights
SEMPRE-LIN-100000	100000	11	distance
SEMPRE-LIN-100000	50000	11	distance
SEMPRE-LIN-100000	10000	11	distance
SEMPRE-LIN-100000	5000	11	distance
SEMPRE-LIN-100000	2000	11	distance
SEMPRE-LIN-100000	1000	7	distance
SEMPRE-LIN-100000	500	3	distance
SEMPRE-NONLIN-100000	100000	11	uniform
SEMPRE-NONLIN-100000	50000	11	uniform
SEMPRE-NONLIN-100000	10000	11	uniform
SEMPRE-NONLIN-100000	5000	11	distance
SEMPRE-NONLIN-100000	2000	11	distance
SEMPRE-NONLIN-100000	1000	11	uniform
SEMPRE-NONLIN-100000	500	11	distance
SEMPRE-SIM-100000	100000	11	uniform
SEMPRE-SIM-100000	50000	11	uniform
SEMPRE-SIM-100000	10000	11	distance
SEMPRE-SIM-100000	5000	11	distance
SEMPRE-SIM-100000	2000	11	distance
SEMPRE-SIM-100000	1000	7	distance
SEMPRE-SIM-100000	500	7	distance

Table 5.5: Best k-NN parameters for different datasets and dataset sizes

MLP Regression Neural networks can be used as regression methods. A simple neural networks structure is the Multi-layer Perceptron (MLP).

For each input feature the input layer has one neuron and for each element of the output the network has an output layer neuron. Between the input and output layer is at least one level of hidden layer. Neurons are connected with some of the previous and next layer neurons. Each neuron is a linear combination of some of the incoming neurons. This linear combination is transformed by a non-linear activation function [KJ13].

A linear combination of the existing neurons can then be used to predict the outcome. This however, quickly becomes very large. To overcome this problem, the backpropagation algorithm is used.

Parameters The parameter space of our MLP model consists of the hidden layer size, the activation function which is used in the hidden layer neurons and the regularization parameter. As a solver, only ADAM optimizer has been used. Therefore, it is not needed to tune parameters like learning rate.

- *hidden_layer_sizes*: [50, 100, 200, 500]
- *activation*: ['identity', 'logistic', 'tanh', 'relu']
- *alpha*: [10^{-8}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}]

Table 5.6 shows the best parameters found in the various dataset configurations. For all tests the best activation function was 'relu', for the other parameter no clear trend is visible from the data. For some models smaller hidden layer amounts were selected as optimal parameters, for some models the biggest amount of hidden layer elements were selected. The regularization term also indicates no clear trend.

Dataset	Size	activation	hidden layer	alpha
SEMPRE-LIN-100000	100000	relu	(9, 2)	1e-05
SEMPRE-LIN-100000	50000	relu	(18, 2)	0.01
SEMPRE-LIN-100000	10000	relu	(18, 3)	1e-05
SEMPRE-LIN-100000	5000	relu	(100, 3)	1e-10
SEMPRE-LIN-100000	2000	relu	(100, 3)	1e-05
SEMPRE-LIN-100000	1000	relu	(100, 3)	1e-10
SEMPRE-LIN-100000	500	relu	(18, 3)	0.01
SEMPRE-NONLIN-100000	100000	relu	(9, 3)	1e-10
SEMPRE-NONLIN-100000	50000	relu	(100, 3)	1e-05
SEMPRE-NONLIN-100000	10000	relu	(100, 3)	1e-10
SEMPRE-NONLIN-100000	5000	relu	(100, 3)	1e-10
SEMPRE-NONLIN-100000	2000	relu	(100, 3)	1e-10
SEMPRE-NONLIN-100000	1000	relu	(100, 3)	1e-10
SEMPRE-NONLIN-100000	500	relu	(100, 3)	0.01
SEMPRE-SIM-100000	100000	relu	(100, 3)	1e-10
SEMPRE-SIM-100000	50000	relu	(100, 3)	1e-05
SEMPRE-SIM-100000	10000	relu	(100, 3)	1e-10
SEMPRE-SIM-100000	5000	relu	(100, 3)	1e-10
SEMPRE-SIM-100000	2000	relu	(100, 3)	1e-10
SEMPRE-SIM-100000	1000	relu	(100, 3)	0.01
SEMPRE-SIM-100000	500	relu	(100, 3)	1e-10

Table 5.6: Best MLP parameters for different datasets and dataset sizes

Random Forest Regression Decision trees have two well-known weaknesses, their model instability, i.e. that a tree can change drastically, and they tend to overfit for a large number of features.

One way to overcome these problems is by using Random Forests [Bre01]. The idea is to create an ensemble of decision trees. To create such an ensemble, the so-called bagging(short for bootstrap aggregation) is used. The idea of bagging is to generate a bootstrap sample from the data and train a decision tree on this data. For prediction, Random Forest takes the average mean of the single decision trees in it.

Parameter For Random Forest Regression, the parameter space includes parameters for a single tree, such as its maximal depth or its split criteria, as well as of how many trees the ensemble should consist of. The number of features was not evaluated, because the datasets are already low-dimensional.

The following parameter space was analyzed:

- n_estimators: [1, 5, 10, 50, 100, 200, 500]
- max_depth: [1, 5, 10, 50, 100, 200, 500]
- min_samples_leaf: [1, 3, 5]

In Table 5.7 the optimal hyperparameter are shown. For datasets with a high amount of samples, i.e. more than 10000 samples, the highest available number of trees seems to be best. For SEMPRE-LIN-100000, it seems that the max depth of trees is significantly higher than for the parameters of SEMPRE-NONLIN-100000. Also, the amount of minimal samples per leaf seems to be higher on SEMPRE-NONLIN-100000.

5.3 Dataset Results

For evaluation, the RMSE error was compared between different models on the different datasets. The experiments were conducted on different dataset sizes, from the full 100000 data samples down to 500 samples. Specifically, the following subset sizes were considered: 50000, 10000, 5000, 2000, 1000 and 500 data-samples.

The aim of this task was to evaluate the robustness of the models. To do so, two different criteria were analyzed. First, the difference between cross validation set and test error was analyzed. A big difference indicates an overfit of the model. Secondly, the results were compared on the different subset sizes. This shows how the model can handle smaller data sources.

Pre-Processing The created datasets have all their features already in a numerical range, therefore no encoding of the features was required. No feature selection was required either, because, as discussed in Section 5.1, all features have at least some

Dataset	Size	nr trees	max depth	min samples
SEMPRE-LIN-100000	100000	500	100	1
SEMPRE-LIN-100000	50000	500	50	1
SEMPRE-LIN-100000	10000	500	100	1
SEMPRE-LIN-100000	5000	500	100	1
SEMPRE-LIN-100000	2000	200	50	1
SEMPRE-LIN-100000	1000	200	50	1
SEMPRE-LIN-100000	500	500	10	1
SEMPRE-NONLIN-100000	100000	500	10	3
SEMPRE-NONLIN-100000	50000	500	10	1
SEMPRE-NONLIN-100000	10000	500	10	5
SEMPRE-NONLIN-100000	5000	100	10	5
SEMPRE-NONLIN-100000	2000	10	5	5
SEMPRE-NONLIN-100000	1000	200	5	3
SEMPRE-NONLIN-100000	500	10	100	3
SEMPRE-SIM-100000	100000	500	10	1
SEMPRE-SIM-100000	50000	500	10	1
SEMPRE-SIM-100000	10000	200	5	3
SEMPRE-SIM-100000	5000	50	5	5
SEMPRE-SIM-100000	2000	100	5	5
SEMPRE-SIM-100000	1000	500	5	3
SEMPRE-SIM-100000	500	50	5	1

Table 5.7: Best Random Forest parameters for different datasets and dataset sizes

importance. We perform feature min-max normalization so that all features are in the same $[0, 1]$ space.

SEMPRE-LIN-100000 In the first experiments, the SEMPRE-LIN-100000 dataset was analyzed. Figure 5.4 shows the test error of the models on the different subset sizes. The best results for this model have been gathered by the Linear Regression and Lasso Regression, which were able to find the ideal solution for all subset sizes. This was an expected outcome since the calculation of the *totalTime* variable is based on a linear equation.

MLP Regression was also able to find this linear equation and the optimal solution, however this only worked for large datasets. Once the amount of samples is decreased under 10000 samples the MLP Regression model was not able to find this linear equation again. In the smallest subset, the results were the worst model results and indicated that the model is not a good fit for such a dataset.

The Random Forest Regressor was not able to model the underlying linear equation. The model its error increases from 21.09 seconds on the largest dataset to 48 seconds on the 500 samples subset.

The k-NN Regressor had the worst performance of all models for the whole dataset, however its results only slightly got worse the smaller the subset became.

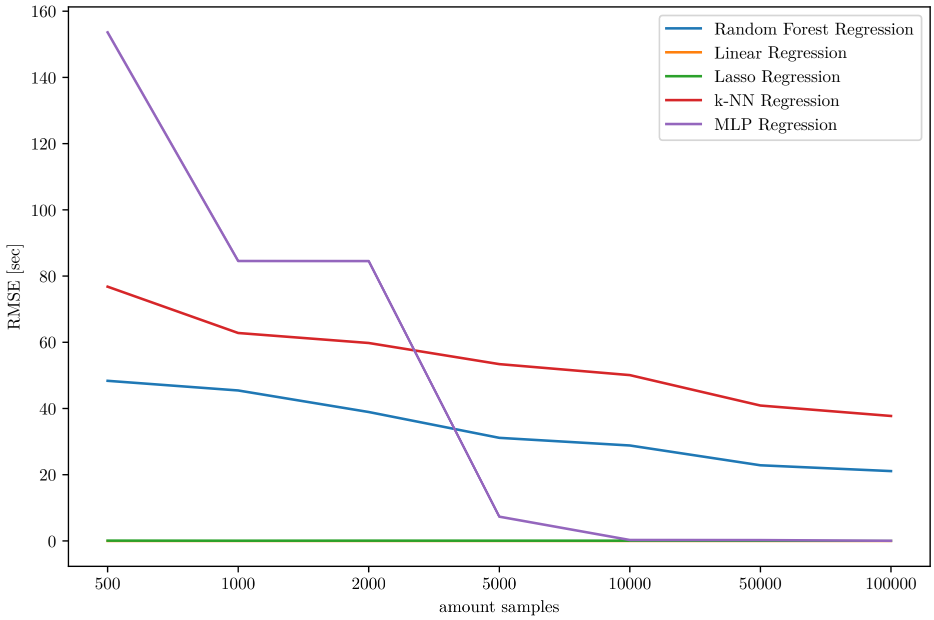


Figure 5.4: Test error of models on different sizes of SEMPRE-SIM-100000

To check the robustness a comparison of cross validation set error to test error was performed. Table 5.8 illustrates these results. The Δ_k was calculated for each dataset by calculating $|e_{k,test} - e_{k,val}|$ where k is the subset size. The linear models were able to find the linear equation of the modeled MTM. The difference between the errors was therefore very small for these models.

Model	Δ_{100000}	Δ_{50000}	Δ_{10000}	Δ_{5000}	Δ_{2000}	Δ_{1000}	Δ_{500}
Random Forest Reg.	0.41	0.74	0.58	2.05	0.25	0.22	6.49
Linear Regression	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Lasso Regression	0.01	0.00	0.01	0.01	0.01	0.02	0.01
k-NN Regression	1.29	1.28	0.41	3.29	2.68	4.14	2.95
MLP Regression	0.04	0.12	0.00	1.95	5.15	4.89	146.35

Table 5.8: Difference between validation- and test-error on different sizes of SEMPRE-LIN-100000

Random Forest Regression and k-NN Regression have a small difference in the errors. For Random Forest, the test error only changed significantly for the smallest dataset of

500 samples, indicating a small overfit. For k-NN, the difference between the errors stays similar on all subset sizes.

For MLP Regression, the validation and test error difference drastically increases for the smallest subset sizes. Thus, indicating that the model is overfitting on small sample sizes.

SEMPRE-NONLIN-100000 The same experiments have been conducted on the SEMPRE-NONLIN-100000 dataset. The results showed again that linear models such as simple Linear Regression and Lasso Regression provide the best results for the dataset, both in error to the predicted value and in the robustness of the model. This is comprehensible since the relationship between the dependent variable and the characteristics is based on a linear relation. In contrast to SEMPRE-LIN-100000, however, it is no longer possible to predict the exact value for the regression models.

These results are shown in Figure 5.5.

k-NN Regression model again has a slightly higher RMSE than the other models. MLP Regression model had similar results as the linear models, until the 500 samples subset. Random forest regression has similar to MLP an increased error in the smallest subset.

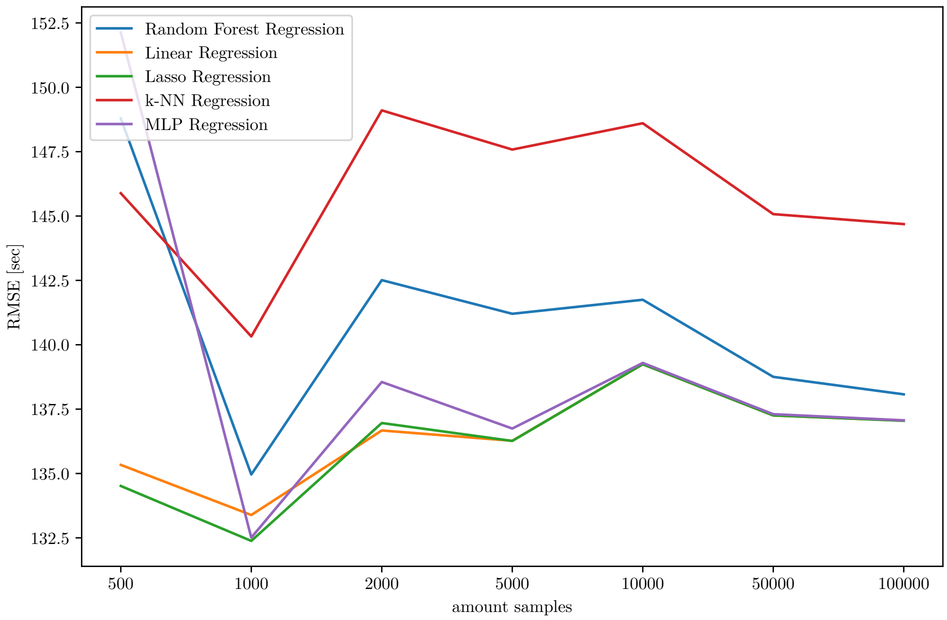


Figure 5.5: Test error of models on different sizes of SEMPRE-NONLIN-100000

When comparing the cross validation and test error for this dataset, we could see that the differences are higher than in SEMPRE-LIN-100000. We assume the reason for this is in the added noise and randomness of the *totalTime* calculation. Table 5.9 shows the results of the validation-, test-error comparison.

Another interesting aspect is that the 1000 samples dataset has for most models the lowest test error. One possible reason could be that the 1000 samples subset might have a biased split and comparatively small test set, and therefore the results are better than they should be.

Model	Δ_{100000}	Δ_{50000}	Δ_{10000}	Δ_{5000}	Δ_{2000}	Δ_{1000}	Δ_{500}
Random Forest Reg.	0.56	2.13	1.44	0.96	0.18	12.21	2.97
Linear Regression	0.63	1.92	2.00	2.67	1.93	5.38	8.11
Lasso Regression	0.63	1.92	2.00	2.55	1.45	5.39	9.96
k-NN Regression	0.01	1.54	2.44	1.70	1.13	11.82	6.41
MLP Regression	0.59	1.90	1.91	2.40	3.63	23.26	187.30

Table 5.9: Difference between validation and test-error on different sizes of SEMPRE-NONLIN-100000

SEMPRE-SIM-100000 For the SEMPRE-SIM-100000 dataset, the experiments have been conducted on the meta-feature representation.

The results show that the lowest RMSE test error was recorded Random Forest and MLP Regression. The results of these two models are clearly different from the results of Linear Regression. The linear models are not able to find as good solutions and have a significant higher RMSE.

Similar to the findings of the other datasets the MLP Regression model has good results for large datasets while the performance visibly deteriorate on small datasets.

When the cross validation and test error were compared, the trend from the other datasets prolonged. MLP error difference increases drastically for the smaller dataset sizes. Linear models and Random Forest Regression only have small divergence. Similar to MLP, k-NN Regression, the cross validation and test error difference, increases for the smaller subset. Table 5.10 shows these error differences.

Model	Δ_{100000}	Δ_{50000}	Δ_{10000}	Δ_{5000}	Δ_{2000}	Δ_{1000}	Δ_{500}
Random Forest Reg.	0.14	0.67	2.10	2.89	3.93	4.69	5.20
Linear Regression	0.25	2.94	5.87	7.25	1.55	5.18	2.35
Lasso Regression	0.25	2.94	5.88	7.25	1.58	4.94	3.50
k-NN Regression	0.40	1.00	0.84	1.83	1.56	14.10	26.91
MLP Regression	0.09	0.16	1.49	2.09	13.50	488.27	416.30

Table 5.10: Difference between validation and test-error on different sizes of SEMPRE-SIM-100000

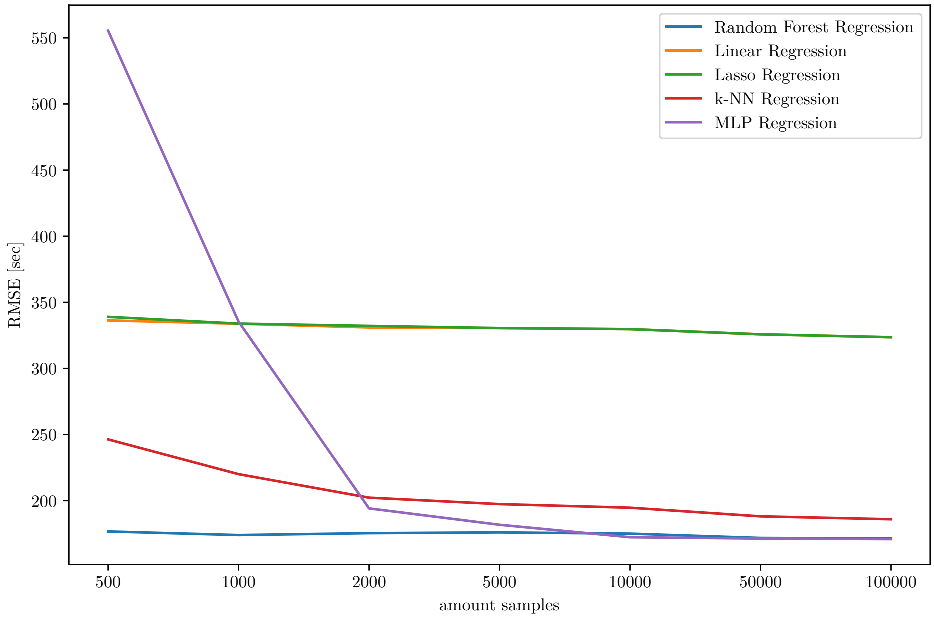


Figure 5.6: Test error of models on different sizes of SEMPRE-SIM-100000

5.3.1 Comparison to AutoML

As seen in the hyperparameter tuning section, the task to optimize a model is tedious. AutoML overcomes this by automatically producing a learned model for a dataset within a computational budget [FKE+15].

Various AutoML systems have been introduced. For this experiments we decided to use *auto-sklearn* toolkit, because the toolkit gave good results in competitions [FEF+20] and it is based on the *scikit-learn* library, which has been used to develop the handcrafted ML models.

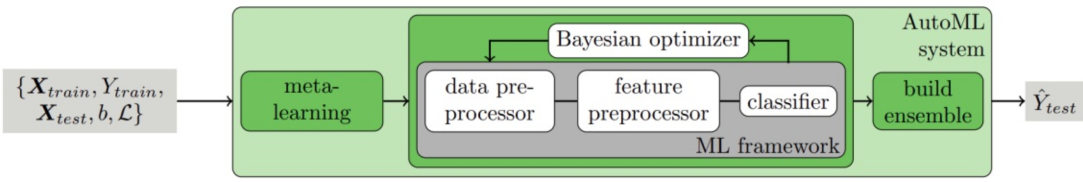


Figure 5.7: Overview of the *auto-sklearn* workflow. Image taken from [FKE+15]

Figure 5.7 shows the steps in the framework. The framework includes a meta-learning step to warm-start the Bayesian optimization procedure, the Bayesian optimization procedure to find the optimal hyperparameter and includes an automated ensemble construction step at the end [FKE+15]. For the parameters, *auto-sklearn* uses a predefined set of models and configurations as search space.

For every experiment AutoML was run for 7200 seconds. For the experiments, the search for preprocessing was deactivated to speedup the search and the same preprocessing steps as for the handcrafted models were performed.

SEMPRE-LIN-100000 The results were again conducted over different sizes of the dataset. Figure 5.8 illustrates the AutoML result as well as the handcrafted results. The performance of AutoML is similar to the linear models, such that it seems possible for the model to find the linear relation in this dataset.

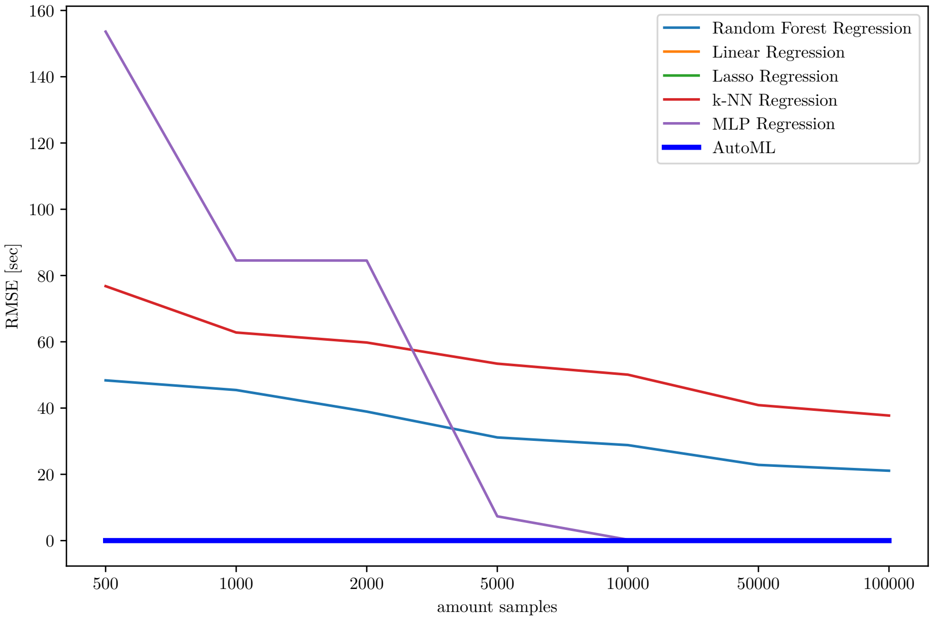


Figure 5.8: Test error of models on different sizes of SEMPRE-LIN-100000 compared with AutoML results

Table 5.11 compares the test error for the best AutoML model over different dataset sizes.

As shown in Chapter 5.3 the relationship of the dependent variable and the independent variables of this dataset is through a linear relation. Therefore, the linear regression can find an optimal solution for this problem with an error of 0 sec. AutoML can theoretically find the very same solution, however due to the preset search space and the

	AutoML	Linear	Lasso	k-NN	Random Forest	MLP
RMSE[sec]	0.00	0.00	0.08	37.74	21.09	0.05

Table 5.11: Test error for the AutoML model compared with handcrafted models on SEMPRE-LIN-100000

exclusion of ordinary linear regression, this is not possible in the used implementation. The best model found by AutoML consisted of an ensemble of multiple Automatic Relevance Determination (ARD) Regressions. In total, the found AutoML model is an ensemble consisting of 15 regression models, with the highest model weights 0.38, 0.14. The other models had relative low weights of 0.06 to 0.02. All models used ARD regression. Therefore, we assume that the AutoML model converge close to 0 but was not able to find the ideal model. Listing 5.1 depicts the structure of the found AutoML model.

Listing 5.1: Structure of ensemble of AutoML model for SEMPRE-LIN-100000

```
{
    0.38: ARDRegression,
    0.14: ARDRegression,
    ....
    0.06: ARDRegression,
    0.02: ARDRegression
}
```

SEMPRE-NONLIN-100000 For this dataset, linear models were not able to find an exact solution, however their results were still the best performing ones. The results compared to handcrafted features on different sizes of the dataset are shown in Figure 5.9.

The AutoML results are slightly better than the results from the handcrafted models. Over all larger subset sizes the AutoML model achieves the best scores, for the smaller dataset sizes linear models perform better. The exact error for the models is shown in Table 5.12.

	AutoML	Linear	Lasso	k-NN	Random Forest	MLP
RMSE[sec]	132.92	137.05	137.05	144.69	138.07	137.07

Table 5.12: Test error for the AutoML model compared with handcrafted model results on SEMPRE-NONLIN-100000

The best AutoML model again consisted of an ensemble of 6 models: 0.42 weight ARD Regression, 0.26 weight ARD Regression, 0.16 weight ARD Regression, 0.08 Linear Support Vector Regression (SVR), 0.06 Gradient Boosting Regression and 0.02 Gradient Boosting Regression.

Listing 5.2 depicts the structure of the found AutoML model.

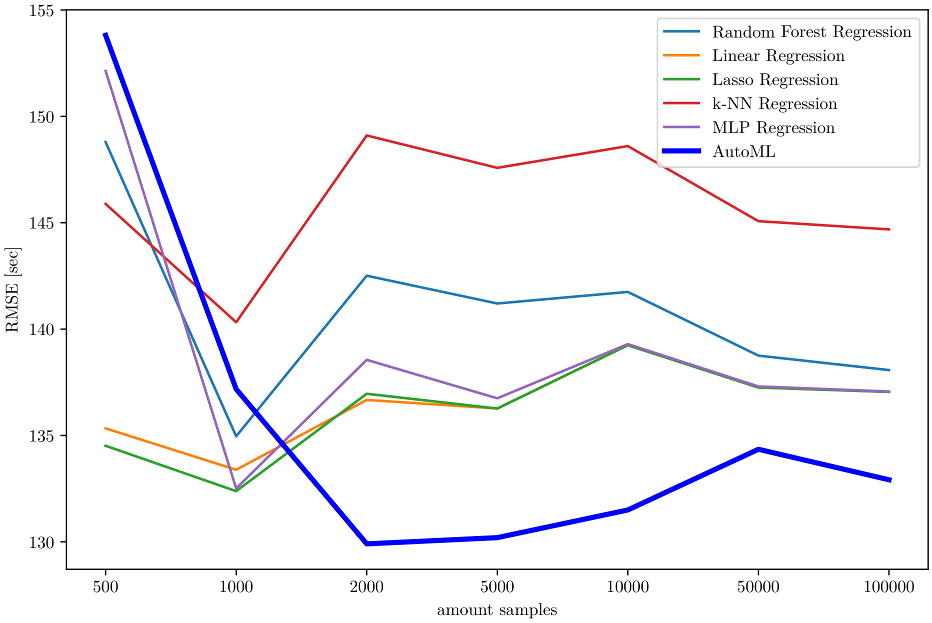


Figure 5.9: Test error of models on different sizes of SEMPRE-NONLIN-100000 compared with AutoML results

Listing 5.2: Structure of ensemble of AutoML model for SEMPRE-NONLIN-100000

```
{
        0.42: ARDRegression,
        0.26: ARDRegression,
        0.16: ARDRegression,
        0.08: LinearSVR
        0.06: GradientBoostingRegression,
        0.02: GradientBoostingRegression,
}
```

SEMPRE-SIM-100000 For this dataset the dependent variable is not in a linear relationship. The linear models perform inferior compared to MLP and Random Forest Regression. Figure 5.10 shows the comparison to the AutoML model. The AutoML model can find a slightly better model than Random Forest Regression. In contrast to the previous dataset, AutoML is also able to find a good and robust result for the small subsets.

In Table 5.13 the results are compared on the full dataset. AutoML, Random Forest and MLP have the lowest RMSE for this dataset, while linear models results are worse.

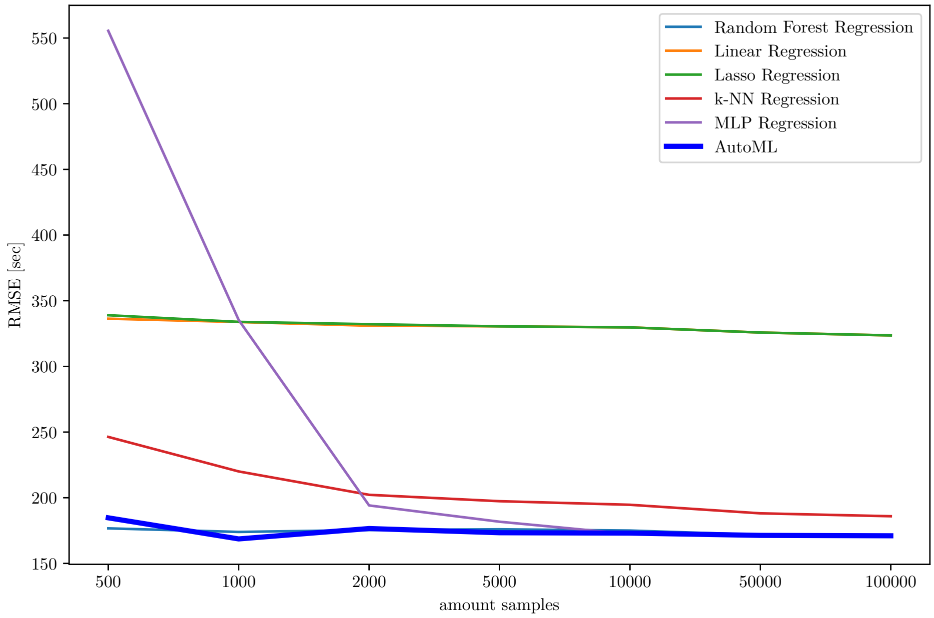


Figure 5.10: Test error of models on different sizes of SEMPRE-SIM-100000 compared with AutoML results

	AutoML	Linear	Lasso	k-NN	Random Forest	MLP
RMSE[sec]	171.03	323.59	323.59	185.92	171.35	170.89

Table 5.13: Test error for the AutoML model compared with handcrafted models on SEMPRE-SIM-100000

The AutoML ensemble consisted of a wide mix of 11 models. Listing 5.3 depicts the structure of the found AutoML model.

Listing 5.3: Structure of ensemble of AutoML model for SEMPRE-SIM-100000

```
{
    0.56: AdaBoostRegression,
    0.10: MLPRegression,
    0.10: GradientBoostingRegression,
    0.06: AdaBoostRegression,
    0.04: ARDRegression,
    0.04: SGDRegression,
    0.02: GradientBoostRegression,
    0.02: k-NNRegression,
    0.02: k-NNRegression
}
```

5.3.2 Comparison Pallet-Meta-Features and Product-Features

For the SEMPRE-SIM-100000 dataset, two different feature representation have been used. One containing of only the products and the amount of ordered products. This would be a feature representation and inference on this feature representation level would be preferred compared to the meta level feature representation. This feature representation includes information about the pallets packages within a shipment.

To compare the results the models are again tuned with the hyperparameters.

The results showed that there is a clear discrepancy between the two feature representations, all results on the meta feature representation are lower than the best results on the product feature representation level.

However, the results from the product level still show that the regression works to some extent. Thus, showing that the trade off between information available and performance.

Figure 5.11 shows the results for this experiments.

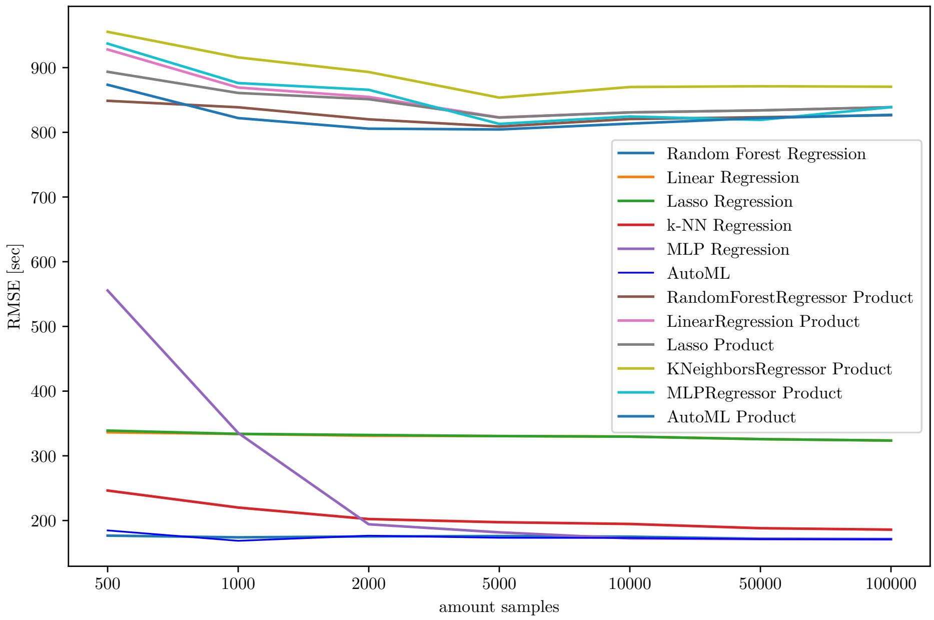


Figure 5.11: Test error of models on different sizes of SEMPRE-SIM-100000-PRODUCT compared with AutoML results

CHAPTER 6

Conclusions

In this thesis we investigated how a time prediction model for an inbound logistics process can be created and evaluated.

First, we captured the business knowledge in the form of ontologies. Based on use-case diagrams and expert interviews, we designed one ontology SEMPRE-CORE, containing relevant configuration elements for a shipment. Furthermore, we extended this with a second ontology SEMPRE-MTM to link time estimations to the shipment configurations. Based on the knowledge of the ontology, we created three datasets. The first dataset SEMPRE-LIN-100000 samples instances according to the schema of the ontology. The second dataset SEMPRE-NONLIN-100000 extends the first one by adding randomness, noise and non-linearity. For the third dataset SEMPRE-SIM-100000 we created an agent-based simulation and sampled a dataset out of it. Furthermore, we analyzed the datasets by explorative analysis and analysis of their feature importance. Based on our experiments, we concluded that all features have at least some significance. Additionally, we showed that *amountPallets* was one of the most impactful features for all datasets.

Based on our experiments on the MTM datasets, linear models gave the best results of our selected handcrafted models. On the other hand, Random Forest Regression gave the best results for the simulation dataset. Furthermore, in the experiments, we showed that MLP regression methods work well if there is enough data available. If there are less than 10000 samples the performance of MLP tends to decrease and we observed that the MLP Regressor tends to overfit on small dataset sizes. To conclude, we showed that for the given problem datasets, no handcrafted model performs best on all three datasets. While linear models were able to find good solutions for the MTM datasets, their performance on the simulation dataset was clearly worse than Random Forest, k-NN and MLP Regression.

In contrast to handcrafted models, we showed that AutoML algorithm was able to find good solutions for all three datasets. Furthermore, we saw that AutoML limited through its defined search space. For the first dataset AutoML was able to find a solution very close to the optimal solution, however it was not able to find a linear model, i.e. the optimal model, as solution for SEMPRE-LIN-100000. Finally, we showed that product feature representation doesn't have enough information and there exist a trade-off between feature entropy and model performance.

In general, this thesis showed that ML models could be useful for this kind of tasks. One field of future work would be to analyze a better integration of the ontology with ML algorithms. The direct inclusion of ontology knowledge into an ML model might increase the model robustness and performance [SPK19]. Secondly, the ontology could be used for explainability of the machine learning model. Another viable extension would be the design of more complex features. In this thesis we reduce the shipment information to meta features and product features, other feature representation like embeddings could improve the dataset and model quality. Furthermore, the models could be transfer learned to data extracted from supply chain systems like ERP systems.

List of Figures

List of Tables

Acronyms

ABM Agent-Based Modeling, 4

AKL "Automatisches Kleinteile-Lager" (in English: Mini-Load Systems), 8

AMLO AutomationML ontology, 9

ARD Automatic Relevance Determination, 59

AutoML Automatic Machine Learning, ix

CART Classification And Regression Tree, 42

CRISP-DM CRoss Industry Standard Process for Data Mining, 5

DC Dublin Core, 14

ERP Enterprise Resource Planing, 22

FOAF Friend of a Friend, 14

k-NN k-Nearest Neighbors, 7

KLT “Kleinladungsträger” (in English: Small Load Carrier), 11

KS-test Kolmogorov-Smirnov test, 39

MAE Mean Absolute Error, 42

MDI Mean Decrease in Impurity, 42

ML Machine Learning, 1

MLP Multi-layer Perceptron, 6

MSE Mean Squared Error, 42

MSO eScop Manufacturing System Ontology, 8

MTM Methods-Time Measurement, ix

OWL Web Ontology Language, 12

OWL 2 OWL 2 Web Ontology Language, 12

PROV-O PROV Ontology, 14

RAMI Reference Architecture Model for Industry 4.0, 9

RDF Resource Description Framework, 12

RMSE Root Mean Squared Error, 6

SCOR Supply Chain Operator Reference, 8

SKOS Simple Knowledge Organization System, 14

SKU Stock Keeping Unit, 17

SPARQL SPARQL Protocol and RDF Query Language, 20

SSE Sum of Squared Error, 48

std standard deviation, 27

STO Standards Ontology, 9

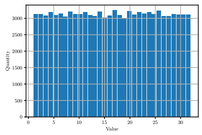
SVR Support Vector Regression, 60

SWRL Semantic Web Rule Language, 21

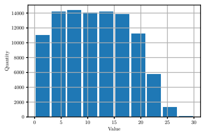
Turtle Terse RDF Triple Language, 12

W3C World Wide Web Consortium, 12

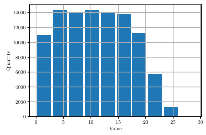
Appendix A



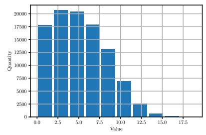
(a) amountPallets



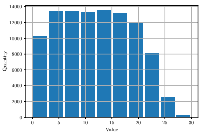
(b) amountPallets-Bandaged



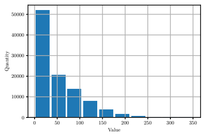
(c) amountPalletsWrapped



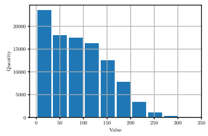
(d) amountPalletsMixed



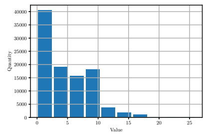
(e) amountPalletsUnmixed



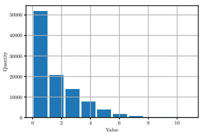
(f) amountUnmixedSmall



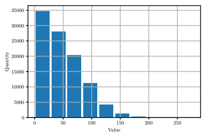
(g) amountUnmixedMedium



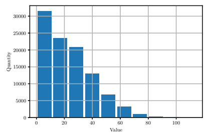
(h) amountUnmixedLarge



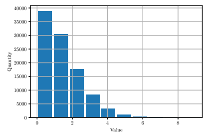
(i) amountUnmixedVeryLarge



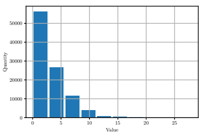
(j) amountMixedSmall



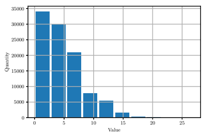
(k) amountMixedMedium



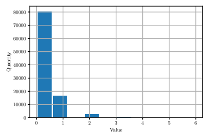
(l) amountMixedLarge



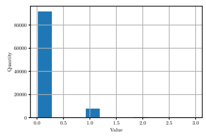
(m) amountCartonSmall



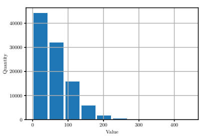
(n) amountCartonMedium



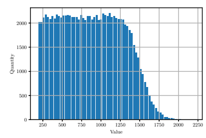
(o) amountCartonLarge



(p) amountCartonVeryLarge

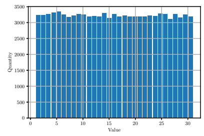


(q) amountDifferentProductsPerPallet

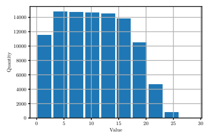


(r) totalTime

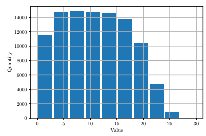
Figure A.1: Histograms for the features of SEMPRE-LIN



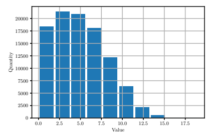
(a) amountPallets



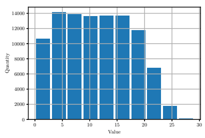
(b) amountPallets-Bandaged



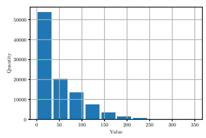
(c) amountPal-letsWrapped



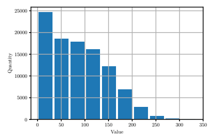
(d) amountPal-letsMixed



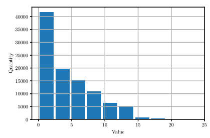
(e) amountPallet-sUnmixed



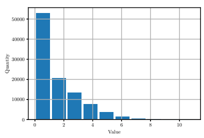
(f) amountUn-mixedSmall



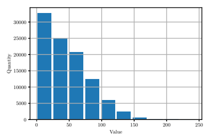
(g) amountUn-mixedMedium



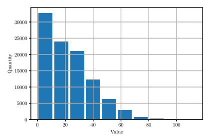
(h) amountUn-mixedLarge



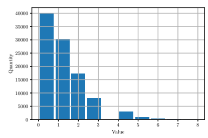
(i) amountUn-mixedVeryLarge



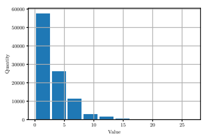
(j) amountMixedS-mall



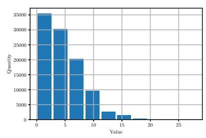
(k) amount-MixedMedium



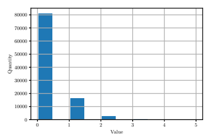
(l) amountMixed-Large



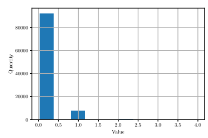
(m) amountCarton-Small



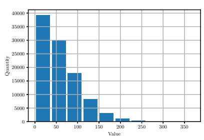
(n) amountCarton-Medium



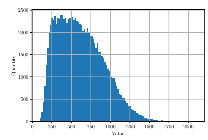
(o) amountCarton-Large



(p) amountCarton-VeryLarge

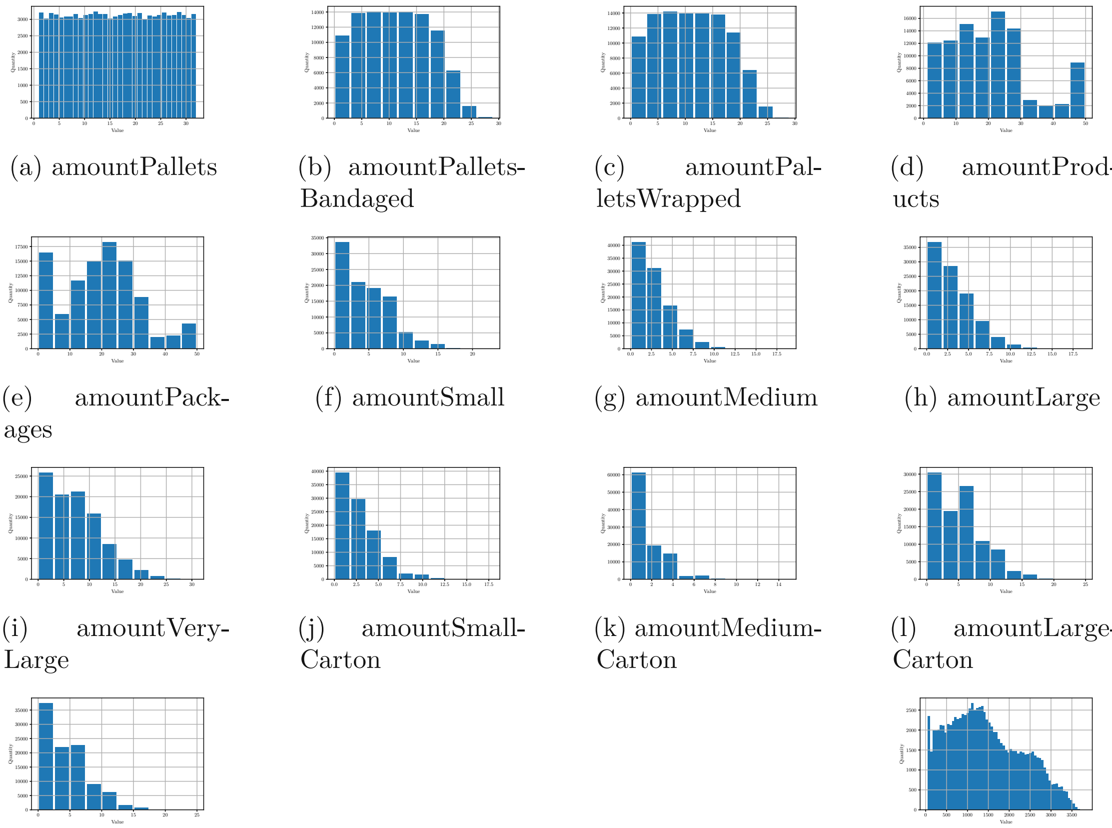


(q) amountDiffer-entProductsPer-Pallet



(r) totalTime

Figure A.2: Histograms for the features of SEMPRE-NONLIN



(a) amountPallets

(b) amountPallets-Bandaged

(c) amountPal-letsWrapped

(d) amountProd-ucts

(e) amountPack-ages

(f) amountSmall

(g) amountMedium

(h) amountLarge

(i) amountVery-Large

(j) amountSmall-Carton

(k) amountMedium-Carton

(l) amountLarge-Carton

(m) amountVery-LargeCarton

(n) amountTotal-Time

Figure A.3: Histograms for the features of SEMPRE-SIM

Bibliography

[BCSMAB13] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. A review of feature selection methods on synthetic data. *Knowledge and Information Systems*, 34(3):483–519, 2013.

[BFOS84] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[BGYW16] Theresa Beyer, Peter Gohner, Ramin Yousefifar, and Karl Heinz Wehking. Agent-based dimensioning to support the planning of Intra-Logistics systems. In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, volume 2016-November, pages 1–4. IEEE, 2016.

[BIM+13] Katherine S. Button, John P.A. Ioannidis, Claire Mokrysz, Brian A. Nosek, Jonathan Flint, Emma S.J. Robinson, and Marcus R. Munafó. Power failure: why small sample size undermines the reliability of neuroscience. *Nature Reviews Neuroscience*, 14(5):365–376, May 2013.

[Bon02] Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99(SUPPL. 3):7280–7287, 2002.

[BPM06] Perakath Benjamin, Mukul Patki, and Richard Mayer. Using ontologies for simulation modeling. In *Proceedings - Winter Simulation Conference*, pages 1151–1159. IEEE, 2006.

[Bre01] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[BXS+20] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M.F. Moura, and Peter Eckersley. Explainable machine learning in deployment. In *FAT* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 648–657, 2020.

[DFH11] John Domingue, Dieter Fensel, and James A Hendler. *Handbook of Semantic Web Technologies*. Springer Science & Business Media, 2011.

[DZB18] Ashish Dandekar, Remmy A.M. Zen, and Stéphane Bressan. A comparative study of synthetic dataset generation techniques. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11030 LNCS, pages 387–395. Springer, 2018.

[FEF+20] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: The next generation. *arXiv preprint arXiv:2007.04074*, 2020.

[FKE+15] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, volume 2015-January, pages 2962–2970. Springer, Cham, 2015.

[GGBH+17] Irlán Grángel-Gonzalez, Paul Baptista, Lavdim Halilaj, Steffen Lohmann, Maria Esther Vidal, Christian Mader, and Sören Auer. The industry 4.0 standards landscape from a semantic integration perspective. In *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, pages 1–8. IEEE, 2017.

[Hev07] R Hevner Alan. A Three Cycle View of Design Science Research. *Scandinavian Journal of Information Systems*, 19(2):87–92, 2007.

[HKV19] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[Hod58] J. L. Hodges. The significance probability of the smirnov two-sample test. *Arkiv för matematik*, 3(5):469–486, 1958.

[HPPsR09] Pascal Hitzler, Bijan Parsia, Peter F Patel-schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language Primer. *W3C Recommendation*, 27(December):1–123, 2009.

[HS98] M.a. Hall and L.a. Smith. Practical feature subset selection for machine learning. *Computer Science*, 98:181–191, 1998.

[KB87] Delmar W. Karger and Franklin H. Bayha. *Engineered work measurement : the principles, techniques, and data of methods-time measurement background and foundations of work measurement and methods-time measurement, plus other related material*. Industrial Press Inc., 1987.

[KGGS+18] Olga Kovalenko, Irlán Grangel-González, Marta Sabou, Arndt Lüder, Stefan Biffl, Sören Auer, and Maria-Esther Vidal. AutomationML Ontology: Modeling Cyber-Physical Systems for Industry 4.0. *IOS Press Journal*, 2018.

[KJ13] Max Kuhn and Kjell Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.

[KM19] Elisa F. Kendall and Deborah L. McGuinness. Ontology Engineering. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 9(1):I–102, 2019.

[KMJ10] Asha Gowda Karegowda, A S Manjunath, and M A Jayaram. Comparative Study of Attribute Selection Using Gain Ratio and Correlation Based Feature Selection. *International Journal of Information Technology and Knowledge Management*, 2(2):271–277, 2010.

[KPR16] Dino Knoll, Marco Prüglmeier, and Gunther Reinhart. Predicting Future Inbound Logistics Processes Using Machine Learning. *Procedia CIRP*, 52:145–150, 2016.

[LWSG13] Gilles Louppe, Louis Wehenkel, Antonio Sutera, and Pierre Geurts. Understanding variable importances in Forests of randomized trees. *Advances in Neural Information Processing Systems*, 26:431–439, 2013.

[Mas51] Frank J. Massey. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association*, 46(253):68, 1951.

[MVG19] Karl Mason, Sadegh Vejdan, and Santiago Grijalva. An 'On the Fly' Framework for Efficiently Generating Synthetic Big Data Sets. In *Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019*, pages 3379–3387, 2019.

[MvH04] Deborah L McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. *W3C recommendation*, 10(February):2004, 2004.

[MVVMH+21] Andrés Muñoz-Villamizar, Josué C. Velázquez-Martínez, Perla Haro, Ana Ferrer, and Roger Mariño. The environmental impact of fast shipping ecommerce in inbound logistics operations: A case study in Mexico. *Journal of Cleaner Production*, 283:125400, 2021.

[NM01] Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology, 2001.

[NPF+17] Elisa Negri, Sara Perotti, Luca Fumagalli, Gino Marchet, and Marco Garetti. Modelling internal logistics systems through ontologies. *Computers in Industry*, 88:19–34, 2017.

[NSN+00] Pete Chapman Ncr, Julian Clinton Spss, Randy Kerber Ncr, Thomas Khabaza Spss, Thomas Reinartz Daimlerchrysler, Colin Shearer Spss, and Rüdiger Wirth Daimlerchrysler. Step-by-step data mining guide. In *SPSS inc*, volume 78, pages 1–78, 2000.

[PGGC+16] Niklas Petersen, Irlan Grangel-Gonzalez, Gokhan Coskun, Soren Auer, Marvin Frommhold, Sebastian Tramp, Maxime Lefrancois, and Antoine Zimmermann. SCORVoc: Vocabulary-Based Information Integration and Exchange in Supply Networks. In *Proceedings - 2016 IEEE 10th International Conference on Semantic Computing, ICSC 2016*, pages 132–139. IEEE, 2016.

[PGKM16] András Pfeiffer, Dávid Gyulai, Botond Kádár, and László Monostori. Manufacturing Lead Time Estimation with the Combination of Simulation and Statistical Learning Methods. *Procedia CIRP*, 41:75–80, 2016.

[PPHM09] Mark Palatucci, Dean Pomerleau, Geoffrey Hinton, and Tom M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in Neural Information Processing Systems 22 - Proceedings of the 2009 Conference*, pages 1410–1418, 2009.

[RN20] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (4th Edition)*. Pearson, 2020.

[SCHP00] SRJ, Samprit Chatterjee, Ali S. Hadi, and Bertram Price. *Regression Analysis by Example*. Number 452. John Wiley & Sons, 2000.

[Sco20] Erwan Scornet. Trees, forests, and impurity-based variable importance. *arXiv preprint arXiv:2001.04295*, 2020.

[SL14] Andreas Scheuermann and Joerg Leukel. Supply chain management ontology from an ontology engineering perspective. *Computers in Industry*, 65(6):913–923, 2014.

[SPK19] Arne Seeliger, Matthias Pfaff, and Helmut Krcmar. Semantic web technologies for explainable machine learning models: A literature review. *CEUR Workshop Proceedings*, 2465:30–45, 2019.

[SS19] Dustin Smith and Sharan Srinivas. A simulation-based evaluation of warehouse check-in strategies for improving inbound logistics operations. *Simulation Modelling Practice and Theory*, 94:303–320, 2019.

[Tib96] Robert Tibshirani. Regression Shrinkage and Selection Via the Lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[TSN+21] Reza Toorajipour, Vahid Sohrabpour, Ali Nazarpour, Pejvak Oghazi, and Maria Fischl. Artificial intelligence in supply chain management: A systematic literature review. *Journal of Business Research*, 122:502–517, 2021.

[VDA15] VDA Center of Competence Packaging. VDA 4500 Small Load Carrier (KLT). https://www.vda.de/en/services/Publications/small-load-carrier-%28slc%29-system-%28klt%29.html, 2015.

[Wir00] Rüdiger Wirth. CRISP-DM : Towards a Standard Process Model for Data Mining. In *Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining*, number 24959, pages 29–39. Citeseer, 2000.

[YOSPV19] Matti Yli-Ojanperä, Seppo Sierla, Nikolaos Papakonstantinou, and Valeriy Vyatkin. Adapting an agile manufacturing concept to the reference architecture model industry 4.0: A survey and case study. *Journal of Industrial Information Integration*, 15:147–160, 2019.