Ambient temperature prediction of pharmaceutical air freight

Terpstra, B.W.C.

Award date:
2019

Eindhoven, October 19th 2018

Ambient Temperature Prediction of Pharmaceutical Air Freight

by
Bart Terpstra

Student identity number: 0791573

In partial fulfilment of the requirements for the degree of
Master of Science
In Operations Management and Logistics

Supervisors:
Dr. Y. Zhang, TU/e, IS (IE/IS)
Dr. A.E. Akçay, TU/e, OPAC (IE/IS)
Ir. Eelco de Jong, Validaide B.V.

TUE. School of Industrial Engineering
Series Master Thesis Operations Management and Logistics

Subject headings: global pharmaceutical logistics, temperature prediction, machine learning

Abstract

The global pharmaceutical logistic sector is one of the fastest growing markets in logistics and a significant part of products is transported by air freight. Part of these pharmaceutical products are distributed through a cold supply chain since they are subject to temperature restrictions. Due to increased regulations and corresponding product quality requirements, managing the cold supply chain is one the main burdens of pharmaceutical companies and distributors, and risk assessments are more important. To meet quality requirements, companies transport the products in temperature controlled environments and have started monitoring the temperatures of their cargo. This study, conducted in collaboration with Validaide B.V., aims to improve lane risk assessments of pharmaceutical air freight shipments. To do so, machine learning techniques are applied on temperature sensor data in order to predict a complete temperature profile of new shipments. Flight statistics and steps in the distribution process called 'milestones' are added to the sensor data, along with weather temperature data and several other variables. Among the tested machine learning techniques are Lasso, Random Forest, AdaBoost, Gradient Boosting, XGBoost, and stacked generalization.

Initial models used a data split that allowed them to train and test on data points of the same shipments, resulting in good predictions. Real-time modeling of shipments may therefore have high potential. The goal of this study, however, is to predict the temperature profile of a completely new shipment of which no data was seen during the training phase. A different data splitting method was applied to avoid training and testing on data points of the same shipment. Application of XGBoost has the best overall performance. The performance of this model is reasonably good, though its main bottleneck is the complexity and quality of the time-series data. The model helps to improve lane risk assessments because the predictions are better than using a simple average. Implementation of the model, however, is hard due to the required amount of data preparation.

Management summary

Introduction

The global pharmaceutical supply chain is one of the fastest growing logistic sectors in the world. Part of the products in the pharmaceutical supply chain are restricted to specific temperature conditions, which makes it an example of a cold supply chain (Kumar & Jha, 2016). The pharmaceutical cold chain is expected to grow with 70% between 2014 and 2021 (IATA, 2018). Recent years, rules and regulations related to product quality have become more strict, making the pharmaceutical cold supply chain hard to manage (Chen & Shaw, 2011). In order to comply to the rules, distributers started to ship the products in temperature controlled environments and to monitor the temperatures of their cargo. A significant part of the products is distributed by air freight over these so-called transport lanes. For these transport lanes, lane risk assessments are performed.

The aim of this study is to improve the current lane risk assessments by applying machine learning techniques on historical temperature data collected by data loggers. The goal is to find an appropriate machine learning technique that, in combination with the available data, predicts the ambient temperature to which future shipments will be exposed to during the distribution process. A prediction model is created for transport lane Luxemburg – Huntsville and product type CRT with temperature range 15-25°C. The model aims to predict temperatures from airport to airport, thus other distribution activities such as transportation to and from the airport are not included. The following research questions were formulated:

Main research question

How can machine learning be applied on temperature sensor data and other relevant data sources, in order to predict the ambient temperature profile and possible temperature excursions of future shipments to improve lane risk assessments?

During this study we will answer the following sub-questions

1. How can the required data be collected and prepared to create a workable dataset?
2. Which other data sources besides sensor data are valuable to add to the model in order to obtain a better temperature prediction?
3. How can features be selected and, if required, be constructed or transformed, to build the temperature prediction model?
4. Which machine learning technique is the most appropriate to do these temperature predictions?
5. How can the temperature prediction model be tested and validated?

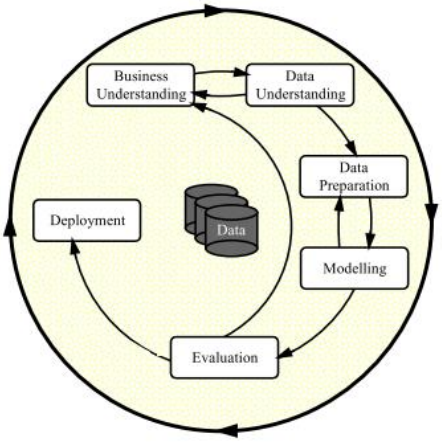
Literature review

For this study, a literature review was mainly performed to find applications of machine learning techniques and data mining on sensor data in the pharmaceutical supply chain. Literature lacks research in the use of temperature sensor data, although the industry realizes the importance of temperature monitoring. No study tried to predict a complete temperature profile of an air freight shipment. This study aims to predict a complete set of temperature values belonging to a new shipment, which makes the problem different from normal time series predictions. The measurements are influenced by a number of variables, mainly temperature related ones, and are expected to be very dependent on the milestone during which the measurement was collected.

Due to the characteristics of the data, regression models are used. This study is relatively unique since it combines large amounts of temperature sensor data and regression models to create a temperature prediction model of a complete shipment. Therefore, this study contributes to the literature by analyzing time series sensor data with regression models.

Methodology

The project will follow the steps of the CRISP-DM (Cross Industry Standard Process for Data Mining) process model (Wirth & Hipp, 2000) which is a widely accepted framework to approach data mining projects. This study followed the six phases identified in the CRISP-DM cycle as depicted in the figure below.



CRISP-DM process model reprinted from (Wirth & Hipp, 2000)

Data understanding, preparation, and visualization

Data was first collected using a web scraping tool and a weather API. Data consists of:

- Temperature sensor data;
- Milestones data (logistic process steps and flight statistics);
- General shipment information (product type);
- Historical weather temperatures.

The milestone data consisted of over 9000 unique milestones. These were converted to generic milestones corresponding to the logistic process. Many data preparation tasks were performed including filling missing data, removing irrelevant and faulty data, data transformation, data removal, and feature construction. Next, the data was visualized to get a better understanding of the data. Selections of data related to cold storage milestones were removed. Finally, a dataset was obtained consisting of 2070 shipments with in total 402917 instances. An instance or data point is one row in the dataset containing a sensor measurement and values for the other features in the columns.

Modeling

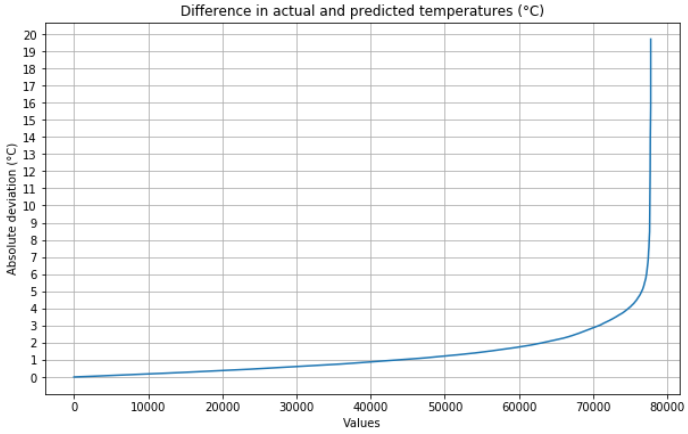
The next phase is the modeling phase. During this phase the following machine learning models were used to create a prediction model: Lasso, Random Forest, AdaBoost, Gradient Boosting, XGBoost, and stacked generalization. Two ways of splitting the data were used: random split and a split based on the years. The first split allowed the models to train and test on data points of the same shipments. This allowed these models to make good predictions. Real-time modeling of shipments may therefore have high potential. The goal of this study, however, is to predict the temperature profile of a completely new shipment, of which no data was seen during the training phase. A different data

splitting method was applied to avoid training and testing on the same shipment. Feature selection was performed using recursive feature elimination (RFE), feature importances, and sequential feature selection (SFS).

The best performing model is the XGBoost model, with performance statistics on the test set: MAE 1.22, MSE 3.13, RMSE 1.77, R^2 0.304. The overall performance of this model is reasonably good, and gives better performance than the baseline predictions, a simple average.

Evaluation

The main bottleneck is the complexity and quality of the time-series data. The model helps to improve lane risk assessments because the predictions are better than using a simple average. Most predictions are accurate within two degrees Celsius, which is visible in the graph below. Implementation of the model, however, is hard due to the required amount of data preparation.



Conclusion

The ambient temperature profile of future shipments can be predicted with a range of machine learning methods such as Random Forest, Lasso Regression, XGBoost, Gradient Boosting, and AdaBoost, applied on sensor data to which shipment milestones are linked. The prediction outcomes can be used as expected temperature profile for a new shipment to estimate temperature related risks, which in turn help to make lane risk assessments. Answers to sub-questions:

1. *How can the required data be collected and prepared to create a workable dataset?*
   Collecting with a web scraper and through an API, many preparation tasks are required.
2. *Which other data sources besides sensor data are valuable to add to the model in order to obtain a better temperature prediction?*
   Historical weather data, general shipment data, milestones data.
3. How can features be selected and, if required, be constructed or transformed, to build the temperature prediction model?
   Selection using RFE, SFS, and feature importances. Constructing by using existing features.
4. *Which machine learning technique is the most appropriate to do these temperature predictions?*
   XGBoost was found to be most appropriate.
5. *How can the temperature prediction model be tested and validated?*
   Splitting data into a training, validation, and test set. Using cross-validation during training, and evaluation of the final model on the test set.

Acknowledgements

Before you lies my master thesis, conducted in partial fulfillment of the requirements for the degree of Master of Science in Operations Management and Logistics at Eindhoven University of Technology. I wish to acknowledge the help of the following people for helping to complete this project.

Firstly, I would like to express my great appreciation to Dr. Yingqian Zhang (TU/e), who provided me with excellent feedback throughout the project and who was always willing to spend time to guide me in the right direction. Thanks to your support I never really got stuck. Secondly, I would like to thank Dr. Shahrzad Faghih Roohi (TU/e) for always providing me with useful feedback during our joint and individual meetings. Her willingness to give her time so generously has been much appreciated. Thirdly, I want to thank Dr. Alp Akçay (TU/e) and Dr. Murat Firat (TU/e) for reading and assessing my thesis. Furthermore, I would like to thank Eelco de Jong (Validaide) for his project guidance and for providing me with all the required information. I would also like to thank Daniel Lutz of Panalpina for giving me access to their data and for helping me with questions related to it. You were of great help to me for better understanding the logistic process.

Finally, I would like to thank my family for their continuous support and encouragement throughout my time as a student. Without your help finishing this project would not have been possible.

Contents

List of figures

List of tables

List of abbreviations

Abbreviation	*Definition*
A/C	Aircraft
AdaBoost	Adaptive boosting
AHP	Analytic hierarchy process
ANN	Artificial neural network
b/u	Build-up (of ULD)
CRISP-DM	Cross industry standard process for data mining
CRT	Control room temperature (15-25 °C)
FDA	Food and Drug Association
FMEA	Failure modes and effects analysis
GDP	Good Distribution Practices
GHA	Ground handling agent
HSV	Huntsville International Airport
LUX	Luxembourg Airport
MAE	Mean absolute error
MEX	Mexico City International Airport
MSE	Mean squared error
R^2	Coefficient of determination
RFE	Recursive feature elimination
RMSE	Root mean squared error
Stacking	Stacked generalization
SFS	Sequential feature selection
TME	Tranportacion Mexico Express (ground handling agent at MEX)
ULD	Unit loading device
WHO	World Health Organization
XGBoost	Extreme gradient boosting

1 Introduction

The first chapter introduces the problem that is analyzed in this study. First, background information of the subject is given followed by a problem statement. Next, the research goal and questions defined. Third, the scope and approach of the study is defined. This chapter finishes by outlining the rest of this report.

1.1 Background

The global pharmaceutical supply chain is a complex supply chain consisting of many parties such as manufacturers, distributers, retailers, and logistic service providers (Xu, Zhang, Gong, & Guan, 2014). It is an example of a cold supply chain since it contains products that need to be kept under temperature controlled conditions (Kumar & Jha, 2016). Differences between products handling and storage conditions during the distribution process make designing a robust cold supply chain complicated (Aung & Chang, 2014). In order to make sure that pharmaceutical manufacturers deliver product quality according to the product registration documents, regulatory bodies such as the European Union, World Health Organization (WHO), and the United States Food and Drug Association (FDA), came up with rules and regulations. Because rules and regulations from government agencies are becoming more and more strict, managing the cold supply chain is one of the main burdens of pharmaceutical and biotechnological companies (Chen & Shaw, 2011). Pharmaceutical companies are therefore increasingly outsourcing their logistics operations to reduce R&D expenditures to become more flexible in their production process, and to satisfy the increasing consumers' demands (Enyinda, Briggs, & Bachkar, 2009). A significant part of pharmaceutical products is distributed by air freight over large distances crossing regions with different climates, so-called 'lanes'. Therefore, to meet product quality requirements, distributors ship products in temperature controlled environments and started monitoring temperatures to improve risk assessments on these lanes. Hence, the product can be kept in the required temperature range and costly temperature excursions can be avoided. With more and more companies collecting temperature sensor data and data from other sources, analyzing this data could provide useful insights to optimize the supply chain.

1.2 Problem statement

The pharmaceutical logistic sector is one of the quickest growing markets in logistics, especially for temperature-sensitive biological products (DARA, 2018). The biopharmaceutical market is expected to grow with 41% between 2014 and 2021, reaching a total value of 1420 billion dollar (IATA, 2018). Part of this market belongs to the cold chain since it contains products that need to be kept under cold conditions (Kumar & Jha, 2016). This part of the pharmaceutical market is expected to grow with 70% in the same period, up to a value of 396 billion dollar (IATA, 2018).

An important aspect in the pharmaceutical supply chain is quality and risk management. Risks in the pharmaceutical supply chain can lead to the waste of resources and can jeopardize the delivery of medication to people requiring them (Jaberidoost, Shekoufeh, Akbar, & Rassoul, 2013). Risk management is also becoming more important since pharmaceutical products are subject to strict regulations imposed by the authorities (Jaberidoost et al., 2013). Because rules and regulations from government agencies such as the EU, WHO, and FDA, are becoming more and more strict, managing the cold supply chain is one of the main burdens of pharmaceutical and biotechnological companies (Chen & Shaw, 2011). Especially since many manufacturers of pharmaceutical products outsource logistic processes to logistics service providers (LSP's) (D'Orazio, 2015). The manufacturer must fully

comply to the regulations and is held responsible for logistics service providers' (LSP) activities (D'Orazio, 2015).

Setting up a cold supply chain is complicated because of the differences between products related to environmental requirements (Aung & Chang, 2014). Therefore, it is important for distributors to know the product requirements beforehand (Aung & Chang, 2014). In the pharmaceutical supply chain, examples of common product categories and their temperature range are: 'Freezing' (-25 to -10 °C), 'Cold' (2 to 8 °C), 'Cool' (8 and 15 °C), and 'Control room temperature' (CRT) (15 to 25 °C). The condition of pharmaceutical products and their raw materials greatly relies on the surroundings under which they are stored and handled, mainly the temperature and humidity (Kumar & Jha, 2016). Of these two influences, environmental temperature is the most critical (Chen & Shaw, 2011; Kumar & Jha, 2016; Aung & Chang, 2014). According to The International Air Transport Association (IATA) 20% of all temperature-sensitive products are damaged during transport caused by exceeding temperature ranges causing a loss of 34.1 billion dollar a year (IATA, 2018). IATA also indicates that 50% of all temperature excursions take place when the products are handled by airlines at the airports. Especially during storage product quality is affected; problems occur over four times as often during storage as during transportation (Xu et al., 2014). The losses related to these excursions may lead to abandoning pharmaceutical air cargo in exchange for ocean freight (IATA, 2018).

The temperature excursions are, as defined by the European Compliance Academy, deviations from the designated storage environment of products for any period during logistic operations (Kumar & Jha, 2016). Research has shown that pharmaceutical products lose their quality during temperature excursions because of changes in active pharmaceutical ingredients (API), for instance degradation, decomposition, polymerization, discoloration, and increased impurities (Kumar & Jha, 2016). Temperature excursions are caused by natural surroundings such as weather influence, or caused by mishandling by humans (Kumar & Jha, 2016). According to Ammann (2013), one to five percent of all events related to transportation contain temperature excursions. Whenever a temperature excursion occurs, the quality impact must be analyzed (Kumar & Jha, 2016), which requires a lot of time and is very costly. However, in many occasions, it is hard to predict the circumstances the shipment will be in (Ammann, 2013). Therefore, Kumar and Jha (2016) insist on temperature measuring and documenting throughout the product lifecycle. Their study showed that in the pharmaceutical industry, monitoring of environmental conditions with data loggers was much more common during manufacturing than during distribution activities. In order to protect the quality of pharmaceutical products, freight temperatures should be monitored. Monitoring temperature and automatic data gathering play a key role in the cold supply chain (Chen & Shaw, 2011).

Many of the current risk assessment techniques have a qualitative nature and only very few studies have focused on data-driven risk assessments, especially including temperature sensor data. As indicated by Kumar and Jha (2016) and Chen and Shaw (2011), temperature monitoring and analysis can greatly improve trade lane risk assessments and decrease the number of temperature excursions, especially since temperature has the biggest influence on product quality. Therefore, this study focuses on data-driven risk assessments by creating a temperature prediction model using sensor data and machine learning techniques.

1.3 Aim and research questions

Aim

The aim of this study is to improve the current lane risk assessment by applying machine learning techniques on historical temperature data collected by data loggers. The goal is to find an appropriate

machine learning technique that, in combination with the available data, predicts the ambient temperature to which future shipments will be exposed to during the distribution process. Furthermore, with the help of this prediction model, possible temperature excursions and their impact can be predicted, such that beforehand measures can be taken to prevent the excursion from taking place.

It is chosen to predict a shipment's complete temperature profile since this is the by Validaide preferred prediction output. Validaide wants to use the predicted temperature profiles for their platform and for the lane risk assessments they make for their customers. Therefore, creating a model that predicts a complete temperature profile generates the highest business value.

In scientific context, the objective of this study is to contribute to the search of finding a suitable machine learning algorithm that can make predictions based on temperature sensor data collected in pharmaceutical air freight. This research will also help to identify the important variables, also called features, that provide predictive power.

To reach these goals, the following main and sub-questions are formulated. The main question can be answered by providing answers to the sub-questions.

Main research question

How can machine learning be applied on temperature sensor data and other relevant data sources, in order to predict the ambient temperature profile and possible temperature excursions of future shipments to improve lane risk assessments?

During this study we will answer the following sub-questions

1. How can the required data be collected and prepared to create a workable dataset?
2. Which other data sources besides sensor data are valuable to add to the model in order to obtain a better temperature prediction?
3. How can features be selected and, if required, be constructed or transformed, to build the temperature prediction model?
4. Which machine learning technique is the most appropriate to do these temperature predictions?
5. How can the temperature prediction model be tested and validated?

1.4 Scope and approach of the project

1.4.1 Scope

The result of this study is a machine learning model that predicts the ambient temperature profile of future shipments. The goal is to predict a temperature profile of a complete shipment that takes place 'tomorrow'. The prediction is made based on all currently available data. With the model, possible temperature excursions and their impact can be identified. The prediction model is created for transport lane Luxemburg – Huntsville and product type CRT with temperature range 15-25°C. The model aims to predict temperatures from airport to airport, thus other distribution activities such as transportation to and from the airport are not included.

1.4.2 Approach

The project will follow the steps of the CRISP-DM (Cross Industry Standard Process for Data Mining) process model which is a widely accepted framework to approach data mining projects. Wirth and Hipp (2000) developed the CRISP-DM model with the goal of specifying a standardized, flexible,

efficient and reliable process that can easily be repeated by different users. According to Wirth and Hipp (2000), data mining required a standard approach to translate business problems into data mining projects. Their model provides guidelines to for the execution and documentation of data mining projects. The CRISP-DM methodology is followed throughout this project.

Four hierarchical levels structure the CRISP-DM cycle. From general to specific, these are: phases, generic tasks, specialized tasks, and process instances (Wirth & Hipp, 2000). On top of the hierarchy are the phases as depicted in figure 1.1 (Wirth & Hipp, 2000). The phases are business understanding, data understanding, data preparation, modeling, evaluation, and deployment. The sequence of the phases and execution of tasks is not fixed, and the arrows only indicate the most important and most frequent dependencies between phases. It depends on the outcome of each phase which phase will be executed next. Each phase consists of several tasks which are called generic since they should cover any data mining situation. Generic tasks consist of specialized tasks and these are the project specific tasks and therefore adjusted to the applied data mining algorithm and available data. The process instance level is structured according to the higher level tasks, but reports how the data mining process is actually executed, containing all actions, decisions, and results. The outer circle of the cycle indicates that new data mining projects can be started based on the outcome of the current project, therefore symbolizing an ongoing process.

Table 1.1 contains an overview of the phases and tasks that are executed during this project. Below, the six phases are briefly discussed.

- *Business understanding*: The first phase aims to understand the project goals and requirements and translates this information into a data mining problem definition and project plan. The eventual result of this phase is the research proposal.
- *Data understanding*: The second phase consists of data collection and getting familiar with the data. The goal is to find out what the data really consists of, gather an overview of the data, and identify what the quality of the data is. This step is closely related to the previous step since some data understanding is required to define the project's problem and to write the project plan.
- *Data preparation*: In the third phase the final dataset is prepared. The raw data is converted into the dataset that will be used as input for the prediction model. Activities include data transformation, data cleaning, selection of attributes (features), and possibly creating new attributes. Data preparation tasks are often performed multiple times and without a specific order.
- *Modeling*: In the fourth phase the machine learning algorithms are chosen and applied. Parameters are optimized to obtain the best predictive performance. The modeling step is related to the data preparation step since data problems are often found while modeling, or the data needs to be reformatted for the modeling technique. Additional features may need to be added to the dataset. Feature selection is an important task during this phase. The result of this phase are one or more working prediction models that seem to have good performance.
- *Evaluation*: In the fifth phase the previously built model or models will be more thoroughly evaluated in order to choose a final model and to make sure that the project's goals are achieved. The phase concludes with a decision on the use of the data mining results.
- *Deployment*: The last phase is the deployment of the created prediction model. The functioning of the model is reported such that users can use the model. Test shipments are used to show what the model can do. In many data mining projects, the end user carries out

the deployment tasks. In this project, Validaide will need to implement the created prediction model into their platform themselves.

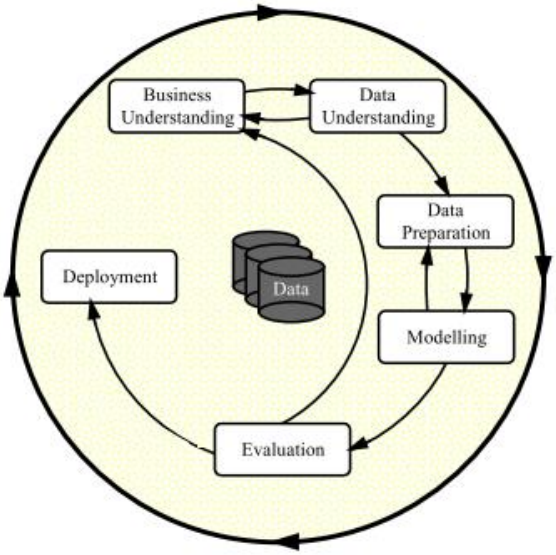


Figure 1.1: CRISP-DM process model reprinted from (Wirth & Hipp, 2000)

1.5 Thesis outline

This chapter presented background information to understand the position of this study, it identified the problem with risk assessments in pharmaceutical air freight, provided research questions, and defined the scope and approach of this study. The next chapter provides more background information about the company Validaide and the current distribution process, and describes the current literature. In chapter 3, the data collection process is presented. Furthermore, chapter 3 provides an understanding of the data and processes. Subsequently, chapter 4 outlines the majority of data preparation. The data is further described and visualized in chapter 5 and additional data preparation is performed. Chapter 6 describes the modeling phase in detail. Which models and features were used is explained. Model evaluation is described in chapter 7. Finally, conclusions and recommendations are reported in chapter 8, and chapter 9 contains a discussion.

Table 1.1: CRISP-DM phases and tasks

Task	CRISP-DM phases					
	Business understanding	Data understanding	Data preparation	Modeling	Evaluation	Deployment
Generic	Define business goal	Data collection	Create complete dataset	Create prediction models	Evaluate data mining results	Document process
Specific	Improve lane risk assessments using data mining on temperature sensor data	Collect temperature sensor data	Combine the sensor data with the milestones	Creation of models with random forest, extreme gradient boosting, adaptive boosting, gradient boost, lasso	Discuss the performance of the regression models	Document intermediate phases
Specific		Collect milestone data	Transform and add weather data to the sensor data	Apply stacked generalization	Evaluate model on test shipments	Write thesis report
Specific		Collect additional data sources weather and general shipment information	Prepare and add general data to the sensor data		Evaluate model on baseline prediction	
Specific			Anonymize data			
Generic	Translate business problem to data mining problem	Describe the data	Data cleaning	Optimize performance of the model	Review model	Describe the use of the model
Specific	Get initial understanding of the data	Visualize the dataset	Fill or remove instances with missing data	Tune parameters of each model with gridsearch cross-validation	Evaluate if the model meets the business goal	Report usage for risk assessments
Specific	Get understanding of the logistic process	Describe the data and provide descriptive statistics	Reassign shipments to the correct lane and remove shipments from different lanes	Perform feature selection with SFS and RFE		
Specific		Identify and describe the meaning of each generic milestone and the logistic process	Delete shipments with too few milestones			
Specific			Delete temperature csv-files without sensor instances			
Specific			Delete irrelevant milestones			
Specific			Standardize shipment number notation			
Specific			Remove duplicates			
Specific			Delete milestones with a too long duration			
Specific			Remove sensor devices with too short measuring intervals			
Specific			Delete milestones of shipments without sensor data			
Specific			Identify and convert to generic milestones			
Generic	Define data mining goals	Determine data quality	Data transformation			
Specific	Find a machine learning technique that predicts the ambient temperature profile a new shipment	Determine if the quality is good enough to proceed	Conversion of sensor data from column to row format			
Specific			Conversion of categorical variables to dummy variables			
Generic	Create project plan		Data reduction			
Specific			Split the dataset per lane and product type			
Specific	Create thesis proposal		Reduce the amount of data of the milestones related to cold storage			
Generic			Data extension			
Specific			Add additional features to the dataset			

2 Background information

This chapter provides background information consisting of a company description, an explanation of lane risk assessments and the logistic process, and provides insights with a literature review.

2.1 Company description

Validaide B.V. is a company that aims to connect buyers and suppliers of logistic services in global pharmaceutical air freight transportation with the help of innovative solutions. The company offers a platform that provides supplier qualification and lane risk assessments to pharmaceutical manufacturers and forwarders. The buyers and suppliers of logistic services consist mainly of manufacturers, forwarders, ground handling agents, airlines, carriers, and warehouse operators. The platform helps buyers to select the right suppliers and the suppliers can indicate their capabilities. The company currently focuses on air freight transportation of pharmaceutical products, but aims to expand their activities to more transportation modes and products such as perishables and other high-value cargo. Validaide works towards a reliable platform with standardized supplier and transport lane qualifications.

One of the companies that joined Validaide's platform is Panalpina. This Swiss company is one of the world's leading providers of forwarding and logistics services. Panalpina is specialized in intercontinental air and ocean freight and corresponding supply chain management solutions. The data used in this project is collected by Panalpina, largely on board of their own aircraft 'Spirit of Panalpina'.

2.2 Lane risk assessments

Validaide performs lane risk assessments with the help of the lane risk assessment algorithm as described in Validaide's white paper (De Jong, 2017). The algorithm calculates a risk exposure index for trade lanes by application of the failure mode effect and analysis (FMEA) method. This method identifies possible failure modes in a process and evaluates them based on the severity, probability, and detectability. The failure modes are clustered into six risk factors, based on similar root causes. The first three risk factors involve temperature exposure incurred by respectively external temperature conditions, human mishandling, or faulty storage facilities and transportation conditions. The lane risk assessment algorithm consists of the following six steps:

1. Assignment of default values for air freight per risk factor;
2. Country-specific adjustment of default value based on climate and country competences;
3. Calculating the risk adjustment based on Location capabilities;
4. Calculating the Location-based Risk Score per Risk Factor;
5. From Location-specific Risk Factors to an overall Location Risk Exposure Index;
6. Combining different flight steps (origin, flight, destination) to lane risk score.

In order to improve the lane risk assessments, the algorithm could be further complemented with quantitative data-driven models instead of solely qualitative data. This thesis can help achieve this goal.

2.3 Cold supply chain process

In order to get a better understanding of the topic and the problem description, it is useful to describe the basic shipping process and the parties that are associated with it.

Air cargo shipments take place between two or more locations. In this study, we use several definitions previously used in the DARA-project description where this thesis is part of. A transport

route is ‘a course from a specific point of origin to a specific destination’. An example is a transport route from a manufacturer in Frankfurt to a distribution center in Mexico City. Goods on a transport route can be transported over different transport lanes. A transport lane is defined as ‘a contracted specified route’. Transport lanes on a transport route can differ, since goods can be transported through different channels. A transport lane consists of a number of legs. Each leg consists of transportation of goods from one point to another. An example of a lane that consist of three legs, is transporting the goods by truck from the manufacturer to the airport, then by airplane to another airport, and then by truck to the consignee. Each lane can have a different number of transportation legs, since different transportation modes or routes can be used. This study focuses on transport lane Luxembourg (LUX) – Huntsville (HSV).

In air cargo shipments, usually the following parties are involved (Shang, Dunson, & Song, 2015): shippers, freight forwarders, carriers, and consignees. According to Shang et al. (2015). Customers, which are shippers and consignees, use forwarders for 90% of all air cargo shipments. Forwarders can use several ground handling agents (GHA), who operate at the airports. In this study, sensor and flight data is provided by logistics service provider Panalpina. Below, a number of figures show several processes that take place at the Panalpina warehouse at Schiphol airport.

Pharmaceutical products are transported to the Panalpina warehouse by truck (figure 2.1). Inside the warehouse the temperature is kept constant at room temperature ranging from 20 to 25°C. This is where most of the products are stored. Other products are stored in large refrigerators or freezers (figure 2.2), depending on the product’s requirements. Products can be protected against ambient temperatures and humidity by a range of packaging options. Examples are thermal blankets, TEMAX extra protective thermal blankets that are closed at the bottom (figure 2.3), active or passive containers (figure 2.4). The products always have a label containing information about the temperature in which they need to be stored. When the products need to be shipped, the build-up of the unit loading device (ULD) will start (figure 2.5) and then the ULD can be loaded into the airplane. Due to insufficient storage space, ULD build-up might take place outside temperature controlled areas. During transportation, the shipments are equipped with temperature and humidity data loggers. These generate the temperature data used in this thesis. Examples of these loggers are the usb-device Tempmate S1 (figure 2.6), smartPoint wireless network loggers, and Q-tags. Once the flight has landed at the destination, the ULD is offloaded from the aircraft. Offloading generally takes place immediately after landing. The ULD is then also unpacked and the goods are stored in temperature controlled storage. However, deviations from this standard procedure occur regularly due to specific circumstances.

The previously described process steps are specified by Panalpina by so-called ‘milestones’ and are registered in SmartView. This platform contains an overview of information about all Panalpina’s shipments, and also houses temperature sensor data gathered during these shipments.

Figure 2.1:Truck unloading at warehouse

Figure 2.2: Refrigerated storage

Figure 2.3: TEMAX thermal blankets

Figure 2.4: Passive container

Figure 2.5: ULD build-up

Figure 2.6: Tempmate S1 data logger

2.4 Literature research

In the pharmaceutical and other supply chains, many risk assessment and risk mitigation techniques exist. This chapter first describes temperature excursions and their key role in managing the cold supply chain. Second, assessment and mitigation techniques that use a reasonable amount of quantitative data are reported. Furthermore, applications of machine learning techniques and data mining with relation to sensor data are mentioned. Since not many machine learning techniques have been applied in the pharmaceutical industry, also applications in other industries are mentioned, such as the food supply chain. Some of the algorithms could be applied to build a temperature prediction model to improve the lane risk assessments.

As indicated in the problem description, the stability of pharmaceutical products highly relies on storing and handling conditions. Environmental temperature is the most important influencer (Chen & Shaw, 2011; Kumar & Jha, 2016; Aung & Chang, 2014). The European Compliance Academy defines temperature excursions as deviations from the designated storage environment of products for any period during logistic operations (Kumar & Jha, 2016). Temperature excursions are caused by natural surroundings such as weather influence, or caused by mishandling by humans (Kumar & Jha, 2016). The importance of temperature measuring and documenting throughout the whole product lifecycle is increasingly recognized (Chen & Shaw, 2011; Kumar & Jha, 2016). Temperature monitoring and collection can be done by several techniques that have been developed over the years (K. Y. Chen & Shaw, 2011): chart recorders, time-temperature indicators (TTI) which are labels with altering colors, data loggers, and radio frequency identification (RFID).

A way of dealing with temperature excursions is the use of early temperature excursion detection and avoiding the damage caused by them. In order to do this, Chen and Shaw (2011) applied statistical methods to control processes and total quality management (TQM). A very common method is analysis of variance (ANOVA) (Chen & Shaw, 2011). Another method is statistical process control (SPC), a very powerful way of monitoring and decreasing variation (Chen & Shaw, 2011). SPC uses collected sample data from the process to detect variations in an early stadium, such that the process can be adjusted (Chen & Shaw, 2011). This results in reduced damage to the pharmaceutical goods and makes sure that problems with the shipment are detected early and before the customer receives the goods (Chen & Shaw, 2011). SPC is also an efficient methods since it does not require personnel to physically inspect the shipment (Chen & Shaw, 2011). According to Chen and Shaw (2011), multiple control charts exist in SPC. Among these are the Shewhart control chart, the cumulative sum (CUSUM) control chart, and the exponentially weighted moving average control chart (EWMA). Every control scheme has its advantages and disadvantages, and several variations on the charts are possible.

The application of machine learning to build predictive models with temperature sensor data is a fairly new concept since the literature on this topic is very scarce. Below, several machine learning techniques are described that others applied for risk assessment in the pharmaceutical sector and other related fields. Also, other possible machine learning techniques emerging from the literature are mentioned.

For this study, it is important to find out which machine learning method will result in the best prediction model. Ahmed, Atiya, El Gayar, and El-Shishiny (2010) performed an empirical comparison of machine learning techniques for time series forecasting. They compared several regression models. The models they included in their study are: multilayer perceptron, Bayesian neural networks, radial basis functions, generalized regression neural networks, K-nearest neighbor regression, classification and regression trees (CART), support vector machines (SVM), and Gaussian processes. These are the most commonly used methods. Of each model, they used the basic variant without any additions or modifications. To find the best model, they applied each technique on the monthly M3 time series competition data. This data consists of 1045 time series, each with a length between 81 and 126 instances. Ahmed et al. also compared several pre-processing techniques, since these turn out to have a big influence on the performance of each model. The pre-processing techniques they compare are: no special preprocessing, time series differencing, and moving averages. Ahmed et al. also performed data transformations, namely log transformation, deseasonalization, and scaling. The time series

differencing pre-processing method resulted in much worse results. The other two methods performed almost equal. The two best performing methods were the multilayer perceptron, followed by Gaussian processes.

Just like the pharmaceutical supply chain, quality management in the food supply chain has also received increased attention (Ting, Tse, Ho, Chung, & Pang, 2014). Several papers emphasize the importance of temperature management during especially logistics movement, and product storage and transfer activities in order to retain quality (Hsu & Liu, 2011; Kuo & Chen, 2010; Montanari, 2008). Ting et al. propose a quality sustainability decision support system (QSDSS) with association rule mining and Dempster's rule of combination techniques to support food logistics planning and quality management. The goal is to find relations between logistics and environmental parameters and the occurrence of quality issues. Association rule mining with the Apriori algorithm is applied to discover good logistics practices. It is one of the most used data mining methods in decision support systems (Ting et al., 2014). The method discovers relations between logistics order flows and sources, for instance transportation mode, product type, and delivery period. Then, the discovered rules are grouped for certain logistic order flows with quality assurance parameters with Dempster's rule of combination. This method integrates the normalized weights of the individual flows in one solution. By using the association rules and Dempster's rule of combination, product in-transit conditions can be designed and applied by grouping route settings with large associations. In their case study, RFID technology is used to gather temperature data and the product's location during red wine distribution. 145 rules are identified on a certain transportation route, which are ordered in importance and then combined. With the model's outcomes, Ting et al. achieved great quality improvements in their case study. The number of shipment quality checks decreased from two to one per month. A great advantage of the method is that it provides continuous model learning, since every new case can be added to the database and used to train the model. A disadvantage is the model's long computation time.

Moraru et al. (2010) applied machine learning techniques in order to predict how many persons are present in a room. Their sensor data contained automatically generated data about temperature, humidity, light intensity, and pressure, variables that are also interesting in the cold supply chain. They manually added data about the number of persons in the room, the number of computers switched on, and stand of the window, based on the timestamp of the data. The window can either be closed, fully opened, or half opened. After combining these datasets, Moraru et al. analyzed the data with two classification and one regression algorithm. The two classifications methods were a decision tree and a Bayesian network, and the regression algorithm used linear regression model. The decision tree gives a good overview of the results, while the Bayesian network works better with inclusion of all the features (Moraru et al., 2010). In their study they tested their models on two different datasets: one with only the automatically obtained sensor data, and one with the sensor data and the manually collected data. To measure the performance of the models, i.e. the deviations of the predictions from the targeted values, they used the mean absolute error (MAE) and the root mean squared error (RMSE). Moraru et al. also calculated the classification accuracy of the two classification models. All three algorithms performed better when the sensor data was complemented with the manually collected data. In the study of Moraru et al., the Bayesian network performed slightly better than the decision tree. The linear regression model had more than twice the error values.

Xu et al. (2014) investigated the risk factors in the cold supply chain and came up with an artificial neural network (ANN) design to predict risks. They searched for input variables from across the whole supply chain, and eventually used data including temperature, relative humidity, employee training, employee weighbridge data which can indicate theft, and tracking of vehicles. Xu et al. use a feedforward neural network with backpropagation aimed at gradient decent. Training of the model is done by the forward and backward propagation. During forward propagation, the input information flows through the input layer, the hidden layer, and then the output layer. With backpropagation, the errors are calculated recursively between the predicted and the desired values. By adjusting the connection weights of the neurons in the layers, the error can be minimized. Xu et al. tested their model with simulation based on artificial data. They found that most problems in the supply chain occur during temporary storage, loading, and transportation of the products.

Chen and Shaw (2011) tried to predict trends in temperature deviation with the help of an exponentially weighted moving average (EWMA) control chart and a backpropagation neural network. Their research used real-time RFID monitoring, by applying an EWMA control chart and artificial neural network. The EWMA control chart helps to intervene instantly when temperature excursions occur and reduce damage to the goods (Chen & Shaw, 2011).

According to Chen and Shaw, the backpropagation network (BPN) is one of the most sturdy models for classification purposes in the cold supply chain, and it rapidly produces stable outcomes. Chen and Shaw also indicated that the CUSUM control chart in combination with ANN's lead to desirable outcomes in earlier studies. However, since other researchers already investigated these possibilities, Chen and Shaw choose to apply application of an EWMA control chart. Their study used RFID tags and temperature sensors on truck shipments, which continuously monitor the current temperature. The EWMA method uses weighting factors which decrease exponentially when the data points are getting older, thus making newer data points more weight. In the research of Shen and Shaw, application of the EMWA method helps to cluster the sensor data into specific areas, which are then used as BPN inputs in order to train the ANN.

Chen and Shaw tested their BPN model on the raw data and on the EWMA data. Their study showed that combining a BPN network with EWMA results in a more accurate BPN model and therefore, more efficient temperature monitoring.

Another example of machine learning application in the cold supply chain is the study of Aung and Chang (2014). In order to define an optimal target temperature for multi-commodity refrigerated storage in the food supply chain, Aung and Chang developed a K-means clustering model based on temperature sensor data. Multi-commodity refrigerated transport contains perishable products that have different temperature requirements in order to retain the optimal product quality. Deviation from the optimal temperatures causes product deterioration. Since it is impossible to create storage facilities for every single product, the model of Aung and Chang clusters products to the optimal refrigeration facility. To do so, they determine optimal target temperatures for refrigerated storage facilities. In machine learning clustering is considered part of unsupervised learning. The K-means algorithm applied by Aung and Chang randomly selects K objects (number of refrigerated rooms) which each represent a cluster mean. All the other objects are then assigned to one of these clusters based on their Euclidian distance from the mean. Then, a new cluster mean is calculated and the process repeats until the criterion function converges. In order to find the optimal target temperatures, they also varied the number of refrigerated rooms.

Shang et al. (2015) apply a Bayesian nonparametric model to assess and forecast risk in multimodal cargo logistics. They define transport risk as the difference between the actual arrival time and the planned arrival time of a shipment. For their study, Shang et al. use air cargo data gathered in half year from 20 airlines on 1336 routes. The data contains trip data from start to end, often containing multiple connected flights. The model they created is a Probit stick-breaking (PSBP) mixture model which calculates transport risk distributions and disruption risk probabilities. Therefore, the model focuses on risk likelihood, one of the two risk components. It does not consider the other component, risk impact. The model considers categorical and continuous demand and decision variables, predictors, such as airline, route, shipping time, and cargo weight. Weather data is not included into the model since it is not available, but it does include milestones describing the shipment status and related activities. To build the model, stick-breaking predictor weights are defined and then transformed with Probit transformation into normally distributed latent variables. Their model is executed with Markov chain Monte Carlo sampling for the latent variables and the model parameters. The inclusion of certain predictors into the model is validated with cross-validation. Their model can be used as a tool for forwarders to provide customized pricing and service, but can evaluate the performance of suppliers.

This study uses a time series dataset. A time series consists of measurements collected sequentially over time (Chatfield, 2000). In this study, sensor measurements are registered every 15 minutes. Therefore, the dataset is an example of a discrete timer series since measurements are made at fixed time intervals (Brockwell & Davis, 2002). In time series forecasting, past observations of a variable are used to create a model that describes the underlying dependencies (Zhang, 2003). Using extrapolation, future values can then be predicted with the model. Successive instances in a time series data set are usually not independent of each other, thus the order of the instances is important (Chatfield, 2000). An important concept in time series forecasting is stationarity, which means that the mean and variance remain constant over time, and that the autocovariance is independent of time (Chatfield, 2000). Common forecasting methods are linear models such as exponential smoothing, moving average, and autoregressive integrated moving average (ARIMA) (Zhang, 2003).

2.4.1 Summary

The influence of temperature excursions on the quality of pharmaceutical products and the importance of temperature monitoring in the supply chain is becoming more evident. Chen and Shaw (2011) simulated the use of SPC with a BPN to detect early deviations from the desired process. Such an application is not useful when the temperature profile of a whole shipment must be predicted. Ting et al. (2014) combined RFID with association rule mining to determine red wine distribution conditions. Association rule mining is useful for pattern finding, but not for regression. Lehikoinen et al. (2012) investigated the application of probabilistic risk assessment with a Bayesian network to evaluate environmental risk in the oil shipment industry. However, this method is not very applicable for temperature sensor data mining. Ahmed et al. (2010) tested several regression methods on the monthly M3 time series data and therefore provided insights on suitable regression techniques for time series data. Moraru (2010) used sensor data with two classification and one regression model to predict the number of persons in a room. Xu et al. (2014) identified risk factors in the cold supply chain with the help of a BPN. In order to find optimal target temperatures for multi-commodity refrigerated storage in the food supply chain, Aung and Chang (2014) developed a K-means clustering model based

on temperature sensor data. Shang et al. (2015) assessed and forecasted transport delay in multimodal cargo logistics using a Bayesian nonparametric model.

Literature lacks research in the use of temperature sensor data, although the industry realizes the importance of temperature monitoring. Recent techniques such as RFID are used more frequently, but the data is not sufficiently put into use. Also, no study tried to predict a complete temperature profile of an air freight shipment. The time series dataset that is used in this study contains measurements that belong to shipments which are in turn split into milestones, the process steps. This study aims to predict a complete set of temperature values belonging to a new shipment, which makes the problem different from normal time series predictions. Shipments have very different starting times, ending times, and no constant duration. The set of milestones have different durations and also the durations of the same milestone are different. Therefore, it is much harder to apply standard time series forecasting methods on the data. Furthermore, the measurements are influenced by a number of variables, mainly temperature related ones, and expected to be very dependent on the milestone during which the measurement was collected.

Due to the characteristics of the data regression models are used. This study is relatively unique since it combines large amounts of temperature sensor data and regression models to create a temperature prediction model of a complete shipment. Therefore, this study contributes to the literature by analyzing time series sensor data with regression models.

3 Initial data understanding

The first phase of the CRISP-DM cycle is business understanding. Chapter 1 and 2, complemented with the thesis proposal, provide the information required to finish this phase. The next phase is data understanding. In this project data understanding can be split in the following tasks: data description, data collection, data visualization, and providing descriptive statistics of the data. This chapter describes what the data consists of, and reports its collection method. Furthermore, this chapter provides data understanding of the milestones and an overview of the logistic process. In this report, an 'instance' indicates a data point. Each data point is a row in a dataset containing values for the variables in the columns.

3.1 Data description

For this project, several data sources are considered interesting. First of all, the temperature sensor data is used. This data can be given more meaning by linking it to the milestone during which the sensor instance was measured. Additionally, some general information about each shipment is available in SmartView which will also be added to the dataset. At last, the historically measured weather temperature is an interesting variable since we are trying to predict a shipment's ambient temperature profile.

3.1.1 Temperature sensor data

SmartView is a web application containing information about Panalpina's shipments. Panalpina has been collecting temperature sensor data of their pharmaceutical shipments since 2010. Data collection by Panalpina happens by placement of temperature data loggers on every single shipment. This can either be one or multiple sensors per shipment. The data is stored and can be viewed on the SmartView web application. On SmartView, a csv-file can be downloaded containing the raw temperature data of that shipment's sensors. Figure 3.1 shows a sample of a csv-file containing raw

data as it is downloaded from SmartView. The first several lines provide some general information of which only the device number is saved. At the bottom of the figure, the first 5 lines of actual sensor data are visible, containing the UTC timestamp, local timestamp, and temperature registered by two sensor devices.

The majority of the available data has departure airport Luxemburg and destination airport Mexico City or Huntsville. In the rest of this report, these locations are abbreviated with LUX, MEX, and HSV. The flight from LUX to MEX makes an intermediate stop in HSV. The initial focus of this project was on both transport lanes, but due to limited time the focus later moved to only the lane LUX-HSV. In SmartView, almost 10000 shipments belong to shipment lanes LUX-HSV-MEX and LUX-HSV.

Many shipments in the years 2010 until 2015 have an undefined transportation method. Therefore, it is unknown whether these shipments took place by road, ocean, rail, or air. However, it is very likely that these shipments were performed by air, since a direct connection by land is impossible and there are zero shipments by ocean. Therefore, an assumption will be made that all undefined shipments are shipped by air, so that this data can also be used to build the prediction model.

New shipments are continuously added to SmartView. Data was collected until the 23th of March 2018 and consists of shipments with 10 different product types. For the two transport lanes, the following number of shipments are available with and without excursions as labelled by SmartView:

- LUX-HSV: 2922 with excursions, 3754 without excursions
- LUX-HSV-MEX: 922 with excursions, 2365 without excursions

```
Local Timezone,UTC
Generated on(UTC),2018-10-15 09:39:52,Period(UTC),2018-03-20 16:35:24,2018-03-27 16:27:27
Generated on(Local),2018-10-15 09:39:52,Period(Local),2018-03-20 16:35:24,2018-03-27 16:27:27

Device,10:00:00:01:14:35,,Device,10:00:00:01:20:c2,
Channel ID,11323,,Channel ID,16582,
Channel,Temperature,,Channel,Temperature,
Unit,°C,,Unit,°C,
UTC,Local Date/Time,Value,UTC,Local Date/Time,Value
2018-03-20 16:36:37,2018-03-20 16:36:37,18.94,2018-03-20 20:39:19,2018-03-20 20:39:19,17.44
2018-03-20 16:51:39,2018-03-20 16:51:39,15.19,2018-03-20 20:54:19,2018-03-20 20:54:19,15.94
2018-03-20 17:06:39,2018-03-20 17:06:39,16.81,2018-03-20 21:09:19,2018-03-20 21:09:19,16.88
2018-03-20 17:21:40,2018-03-20 17:21:40,18.44,2018-03-20 21:24:19,2018-03-20 21:24:19,17.63
2018-03-20 17:36:38,2018-03-20 17:36:38,19.00,2018-03-20 21:39:19,2018-03-20 21:39:19,17.94
```

Figure 3.1: Sample raw temperature data

3.1.2 Milestones

In SmartView, also the milestones of the shipments are registered. As described earlier, these milestones contain information about the status of the shipment. Figure 3.2 shows how the milestones are displayed in SmartView. The name of the user that entered the milestone is removed. Examples of milestones can be found in the figure, and are for instance ULD build-up, aircraft departure, and landing at the destination. Milestones can be entered into SmartView manually by a person, or is received from the airline automatically. Since the manually added milestones lack consistent descriptions, the milestones have to be rewritten in a standard form as described in chapter 4.1.4. A portion of the shipments do not have milestones written in the system. These shipments will be removed from the dataset.

#	Timestamp	User	Description
1	2018-03-29 04:37:00		Truck opened in LUX, start unload.
2	2018-03-29 05:32:00		Cargo stored in temp controlled HLC center.
3	2018-03-29 09:05:00		Cargo build up on ULD and stored in temp controlled cell.
4	2018-03-29 14:55:00		ULD out of cell for uplift to flight.
5	2018-03-29 17:45:00		Aircraft departure
6	2018-03-30 08:26:00		ATA FLIGHT MEX
7	2018-03-30 08:50:00		OFFLOADING AIRCRAFT
8	2018-03-30 10:51:00		CARGO AT TME
9	2018-03-30 11:01:00		RELEASE OF CARGO TO CUSTOMS BROKER

Figure 3.2: Milestones in SmartView

3.1.3 Historical shipments' general data

Panalpina collects some general information about each shipment and stores it in SmartView. This information consists of the shipment's customer, flight number, product type, origin, destination, number of temperature sensors, number of excursions, and some other less relevant information. The general information of 5 shipments is depicted in figure 3.3. The columns are split halfway by a white line in order to put them in this report, and the shipments' customer names are removed.

```
ID,"Shipment ID",Product,Order,Customer,Origin,Destination,"Transport Method","Transport Company","Flight Code",
31910,"FRA 030351 HSV","+15° to +25° C",-                ."LUX (LUX Outbound)","HSV (Huntsville)",Air,-,GTI606,2018-10-10,4,4
31875,"FRA 030354 HSV","+15° to +25° C",-                ."LUX (LUX Outbound)","HSV (Huntsville)",Air,-,GTI600,2018-10-06,4,4
31779,"BRU 523571 HSV","+2° to +8° C",-                "LUX (LUX Outbound)","HSV (Huntsville)",Air,-,GTI600,2018-09-29,1,1,"20
31780,"BRU 523573 HSV","+2° to +8° C",-                "LUX (LUX Outbound)","HSV (Huntsville)",Air,-,GTI600,2018-09-29,1,1,"20
31776,"GVA 944121 HSV","+2° to +8° C",-,'                ","LUX (LUX Outbound)","HSV (Huntsville)",Air,-,GTI606,2018-10-

Flight Date",Associations,Disassociations,"Start Time (UTC)","Stop Time (UTC)","# Excursions",Sensors
"2018-10-08 18:33:03","2018-10-11 03:00:10",2,"10:00:00:01:1d:a3,10:00:00:01:21:89,10:00:00:01:19:3e,10:00:00:01:1e:26"
"2018-10-05 17:00:00","2018-10-08 01:50:46",2,"10:00:00:01:1e:02,10:00:00:01:17:19,10:00:00:01:14:5a,10:00:00:01:19:09"
.8-09-28 16:40:00","2018-10-12 17:00:32",3,10:00:00:01:12:81
.8-09-28 16:40:00","2018-10-03 20:26:58",3,10:10:00:01:04:b9
03,2,2,"2018-09-28 14:35:00","2018-10-08 17:00:44",16,"10:00:00:01:1e:40,10:00:00:01:14:1a"
```

Figure 3.3: First 5 lines of general data

3.1.4 Historical weather data

The last type of data that needs to be collected is the historical weather data. The data includes hourly weather temperature at the three locations LUX, HSV, and MEX. Data is collected of the years 2014 until March 2018.

3.2 Data collection methods

The general shipment data was downloaded from SmartView as a csv-file with all the general information of all shipments. Harder to obtain were the temperature sensor data, the milestones, and the historical weather data.

3.2.1 Web scraping tool

The temperature data cannot be downloaded or exported from SmartView at once, resulting in the fact that a person must click through the website manually in order to extract every single csv-file containing the raw data of one shipment. Just like the temperature data, exporting all milestones at

once is also impossible. These need to be extracted by entering a shipment's data in SmartView, then select the milestones as shown in tabular format in figure 3.2, copy, and then paste the lines in a new file.

Since manually obtaining the sensor data and milestones is too time consuming, a web scraping tool was created with Python which 'scrapes' all the required data from web application SmartView automatically. The scraping tool is created with Python package 'Selenium'. The steps that are followed by the web scraping tool are broadly described in appendix 1.

3.2.2 Weather API

Several websites provide access to an API to extract weather data, but most of the websites require a paid account. Dark Sky (darksky.net) provides free access to their API, although limited to 1000 API requests per day. One API request provides the hourly weather data of one day in the past, including a timestamp in Unix format and the temperature in degrees Fahrenheit. The API is accessed through a Python script and the data is saved in a separate csv-file for each location. Before saving the data, the output from Dark Sky is converted to a timestamp in UTC time format and a temperature in degrees Celsius. Because the historical weather data is only given for each hour, data interpolation is executed such that the weather data is available for every 15 minutes.

3.3 Data understanding: milestones

3.3.1 Milestone statistics

All required data is collected and the data can now be further understood. In total, we have 9991 shipments for the lanes LUX-MEX and LUX-HSV. Of these 9991 shipments, 7139 shipments contain milestones. The remaining shipments had no milestones reported in SmartView. The total number of milestones is 72197, which means that each shipment has on average 10,11 milestones. In figure 3.4, the distribution of the number of milestones per shipment is shown. Most of the shipments have 9 to 13 milestones. Shipments that have more milestones often contain a few dozen sensor battery tests. Each milestone has a milestone number, timestamp, name of the user that entered the milestone, milestone description, and the shipment number where it belongs to. The milestone number indicates the x-th milestone for a specific shipment.

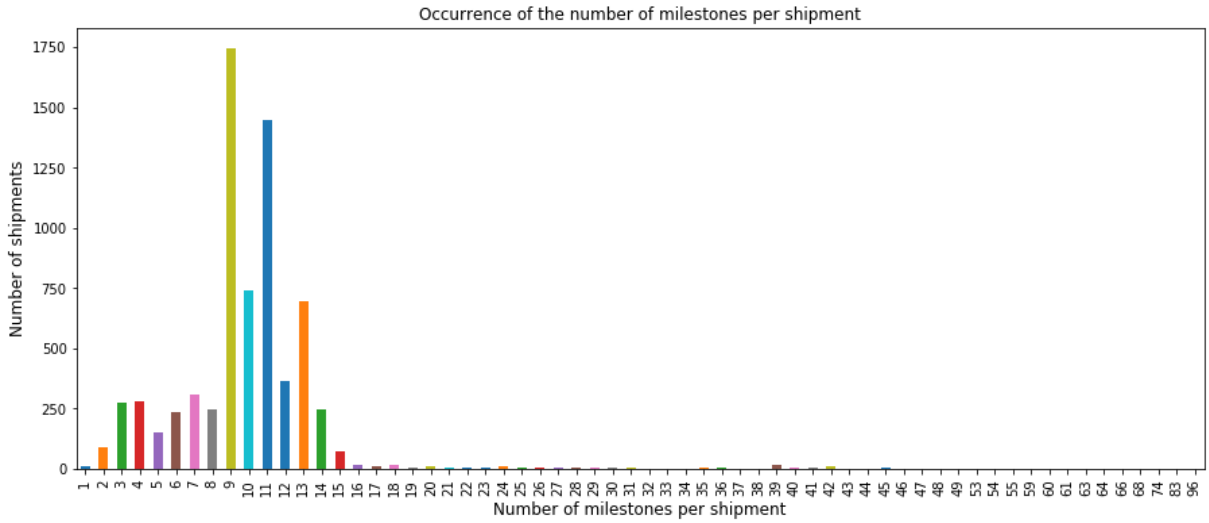


Figure 3.4: Number of milestones per shipment

The milestones are entered in SmartView automatically or by 46 different users. A great portion of the milestones is automatically entered by airline information that is linked to SmartView, and a small number of users enter the majority of the remaining milestones.

The milestone data consists of 9137 unique milestones. This is caused by the fact that users can enter the milestones manually into the system, and they do not use consist descriptions. The most often occurring milestone is 'Scheduled Time of Departure', with 3397 occurrences, and is always entered automatically into the system. Table 3.1 shows the 30 milestones that occur most often.

Due to the large number of unique milestones, it is hard to understand what the logistic process of each transport lane consists of. Therefore, the initial focus is on the milestones in order to discover a general logistic process. Data preparation on the milestones is performed to further understand the data.

Description	Occurrence	Description	Occurrence
'Scheduled Time of Departure',	3397	'Truck complete',	976
'Estimated Time of Arrival',	3286	'ULD out of cell for uplift to flight.',	813
'Actual Time of Departure',	2978	'Flight arrival',	744
'OFFLOADING AIRCRAFT',	2037	'Truck Temp Check 68F Loading Truck',	678
'DOC RELEASE TO BROKER',	1952	'ATA FLIGHT',	643
'ULD out of cell for upload to flight',	1935	'GTI606 arrived at KHSV from ELLX/LUX',	637
'RELEASE OF CARGO TO CUSTOMS BROKER',	1879	'Cargo build up on ULD and stored in schenk system',	609
'Loading Complete',	1753	'Truck loading completed.',	571
'Cargo build up on ULD and stored in temp controlled cell.',	1704	'GTI600 arrived at KHSV from ELLX/LUX',	568
'Flight Arrival',	1644	'Truck temp check 68F Loading truck',	437
'Cargo build up on ULD and stored in temp controlled cell',	1622	'GTI604 arrived at KHSV from ELLX/LUX',	431
'CARGO AT TME',	1569	'ULD out of schenk for upload to flight',	419
'ULD out of cell for uplift to flight',	1560	'Aircraft departure',	378
'ATA FLIGHT MEX',	1444	'AC Arrival',	368
'Actual Time of Arrival',	1144	'Truck temp check 68 F. Truck loading.'	343

Table 3.1: 30 most occurring milestones

3.3.2 Identification of relevant milestones

After an overall check of the milestone data, it is quickly visible that the two shipping lanes are different when it comes to the collected milestones. The main reason is that the plane from LUX to MEX makes an intermediate stop in HSV, while the flights on the lane LUX-HSV are direct flights. This results in a larger number of milestones for the lane to MEX compared to the lane to HSV because the plane arrives, departs, and unloads in HSV. The lane will be called LUX-HSV-MEX indicating the two flight legs.

With the temperature sensor data and milestones being separately collected, the goal is to eventually assign the correct milestone to every instance of the sensor data. To do so, the more than 9000 unique milestones are converted to a small set of relevant milestones for each lane. This is done separately for each lane because the relevant set is different. However, many of the milestones are similar.

Below, for each lane the set of relevant and often occurring milestones is identified. Most of the relevant milestones are easily determined by looking at a number of individual shipments, others were found while performing data preparation on the milestones as described in chapter 4.1. One very striking observation is that the aircraft departure milestone from HSV does not exist for any shipment on lane LUX-HSV-MEX.

Lane: LUX-HSV

1. Truck opened and start unloading (LUX)
2. Cargo stored in temperature controlled area (LUX)
3. Cargo build-up on ULD and stored in temperature controlled area (LUX)
4. ULD out of cell for uplift to flight (LUX)
5. Aircraft departure (LUX)
6. Aircraft arrival (HSV)
7. Aircraft offloading (HSV)
8. Cargo stored in temperature controlled area (HSV)
9. Loading truck start (HSV)
10. Loading truck complete (HSV)

Lane: LUX-HSV-MEX

1. Truck opened and start unloading (LUX)
2. Cargo stored in temperature controlled area (LUX)
3. Cargo build-up on ULD and stored in temperature controlled area (LUX)
4. ULD out of cell for uplift to flight (LUX)
5. Aircraft departure (LUX)
6. Aircraft arrival (HSV)
7. Cargo stored in temperature controlled area (HSV)
8. Aircraft arrival (MEX)
9. Aircraft offloading (MEX)
10. Cargo at TME (MEX)
11. Release to customs broker (MEX)
12. Doc release to broker (MEX)

3.4 Logistic process description

The relevant milestones used by Panalpina to describe the logistic process are now identified. To get a complete understanding of the shipping processes, the processes are explained for both transport lanes. The logistic process of transport lane LUX-HSV starts with (1) the arrival of a truck at the airport's warehouse. The trucks are not temperature controlled. Next, (2) the cargo is unloaded from the truck and stored in a temperature controlled area according to the product type. Cargo is generally stored as a whole pallet. The temperature of the warehouse is kept around 20°C. Separate refrigerated storage locations exist inside the warehouse for product types other than CRT. The next step is (3) the start of the ULD build-up. Due to a lack of space, this activity normally takes place outside refrigerated areas but inside the warehouse. Once the ULD is finished, the whole ULD is normally stored in temperature controlled conditions. This is however not a separate milestone. Once aircraft departure is near, (4) the ULD is taken out of the warehouse and is loaded into the aircraft. Subsequently, (5) the aircraft departs. The cargo bay is temperature controlled. The next step is (6) the

arrival of the aircraft in HSV. Almost immediately after the arrival, (7) the cargo is offloaded and the ULD is broken down outside the warehouse, thus not in temperature controlled surroundings. However, ULD breakdown and subsequent storage is a relatively short process mainly executed during night. ULD breakdown is not added as a milestone to the set of generic milestones since it only occurs in SmartView several times. Once the ULD is broken down, (8) the cargo is stored in temperature controlled storage, in similar manner as in the warehouse in LUX. The next registered milestone is (9) the start of truck loading. At this point the temperature sensor devices are taken of the cargo. The process finishes when (10) the truck loading has finished.

The shipping process of transport lane LUX-HSV-MEX is largely the same as the process on lane LUX-HSV. The process differs once (6) the aircraft arrives in HSV, where a part of the cargo is offloaded and some new cargo is loaded. The part of the cargo that is offloaded is (7) stored in the temperature controlled warehouse. The cargo of which temperature sensor data is collected stays on the aircraft and only this data is used in this study. Once offloading and loading are finished, the aircraft departs from HSV. However, this activity is never registered as a milestone. The next milestone is (8) the arrival of the aircraft in MEX. (9) The aircraft is offloaded, the ULD is broken down, and (10) the cargo is stored in the temperature controlled bonded warehouse of ground handling agent Tranportacion Mexico Express, from now on called the 'TME'. The TME does have cool-storage areas, however, due to size restrictions it might occur that cargo is temporarily stored outside these areas. The ULD's and containers are broken down inside the warehouse, but outside the cool-storage rooms. General cargo that is not restricted to temperatures is broken down outside the TME. Next, (11) the cargo is released to the customer's customs broker who starts with the customs clearing process for which (12) the broker receives the required documentation (indicated by 'Doc'). Cargo responsibilities of Panalpina end with the hand-over of these documents, and therefore the process ends here.

4 Data preparation

Data preparation is required to improve the understanding of the data. This chapter describes which data preparation tasks are executed on the milestones, temperature sensor data, and other data sources. How the data sources are combines to one dataset is also reported.

4.1 Milestones data preparation

The logistic processes of both transport lanes are determined by identification of the relevant milestones. The next step is data preparation of the milestones. The milestones needs to be prepared before the data can be used since the data contains missing values, duplicates, spelling related errors, and consists of 9137 unique milestones. Only a small number of milestones that indicate the logistic process are relevant for this research, and will be linked to the sensor instances based on the timestamp in a later stadium. All 72197 milestones, of which 9137 are unique, are converted to a relevant milestone describing this logistic process, now called 'generic' milestone.

At the start of the data preparation phase, the milestones dataset consists of a large csv-file with 72197 instances. One instance contains a milestone number, timestamp, user, description, and shipment number where it belongs to. Data preparation is performed with Python and Excel. Excel is mainly used to identify missing data, which is subsequently removed with Python. Within Python, the packages Pandas and Numpy are used for data analysis and preparation.

Many of the 9137 unique milestones have the same meaning or can be categorized into a generic milestone. The reason for so many unique milestones is that they are manually entered by 46 different

people, thus many variations on the ‘same’ milestones exist. Irrelevant milestones will be deleted from the dataset.

4.1.1 Missing data

The milestone file contains missing data. Some instances miss data because the web scraping tool somehow did not scrape the information, or the information was not available in SmartView. Of all shipments, 31 milestones have timestamp ‘n/a’ which means ‘not applicable’. After checking in SmartView, the timestamp of these milestones is indeed missing. Shipments containing one or more milestones without timestamps are removed from the dataset.

Furthermore, 49 shipments miss the milestone number, 26 shipments the user, 22 shipments the description, and 9 the shipment number. The missing of the shipment number is rather strange since this number is added with Python as part of the scraping algorithm. The missing data is added to the file by manually looking up the shipments in SmartView.

Also, a collection of the most recent shipments miss multiple essential milestones, and therefore, these shipments are also deleted from the dataset. It seems that these milestones were not updated by the system yet, since several weeks after data collection the milestones were registered in SmartView.

4.1.2 Shipments with insufficient number of milestones

While checking the data, it is observed that shipments belonging to the years 2013, and partially 2014, miss many of the identified set of relevant milestones. This is probably due to the fact that the SmartView system was implemented around that time, and milestones were not collected and entered into the system in the correct way. Since the logistic process was the same, keeping shipments with missing milestones leads to an unrealistically long duration of the remaining milestones that were registered for a shipment. Therefore, the shipments with too few milestones are useless and will be removed. All data from 2013 is removed because all these shipments contain at most 4 milestones. Of lanes LUX-HSV and LUX-HSV-MEX, respectively 1244 and 392 shipments are removed.

To further find shipments with too few milestones, a loop is used in Python to find shipments with subsequently, 1, 2, 3, 4, 5, 6, 7, 8, and 9 milestones. At this point the number of shipments on lane LUX-HSV and on lane LUX-HSV-MEX is respectively 4455 and 2149. A decision needs to be made on what the minimum number of milestones is for each lane. When checking shipments of lane LUX-HSV with 7 or 8 milestones, these shipments often still miss important milestones such as flight departure, flight arrival, and ULD build-up. Shipments from lane LUX-HSV with 9 milestones or more generally contain all important milestones, and therefore, the minimum number of milestones per shipment for this lane is set at 9.

For lane LUX-HSV-MEX, the set of relevant milestones is larger than for lane LUX-HSV. Therefore shipments with 9 milestones still miss too many important milestones. Shipments with up to 9 milestones are removed from the dataset. Shipments with 10 milestones often miss the milestone of the arrival of the aircraft in HSV. However, even shipments with more milestones miss this important milestone. By deleting all shipments with 10 milestones, almost a quarter of all data for this lane will be lost while the remaining milestones can still be used to model the other activities. Therefore, the minimum number of milestones for this lane is set at 10.

Now that the shipments with an insufficient number of milestones are identified, these shipments are removed from the dataset. Table 4.1 indicates the number of shipments and corresponding

number of milestones that are removed from the dataset. The total remaining number of shipments and milestones are respectively 5375 and 61004, with 8061 unique milestones.

Number of milestones per shipment	Affected number of shipments	Corresponding number of milestones
1	6	6
2	0	0
3	21	63
4	187	748
5	146	730
6	215	1290
7	302	2114
8	251	2008
9 (only LUX-HSV-MEX)	122	1098
Total	1250	8057

Table 4.1: Number of removed shipments and milestones

4.1.3 Data cleaning

A check is performed to find possible duplicate instances. A total of 878 instances are exactly the same and belong to 55 shipments. When two or more instances have the exact same values for all columns, only the first occurrence is kept.

Some of the shipments are classified to the wrong shipping lane. In SmartView most of these shipments belong to the lane LUX-HSV, but their flight number and milestone indicate that the flight actually belongs to lane LUX-HSV-MEX. This, however, is no problem when cleaning the milestone data or for combining the milestones with the temperature data, and is resolved when the general data is added to the combined dataset containing sensor data and milestones in chapter 4.4. Several other flights were wrongly classified in SmartView. These are flights to Shanghai, flights to Azerbaijan, and flights to Johannesburg with a stop in Tripoli. These 9 shipments with 83 milestones are removed from the dataset.

In order to combine the temperature sensor and milestone data, it is useful to have a generic notation of the shipment number. Of lane LUX-HSV and LUX-HSV-MEX, respectively 8 and 2 shipment numbers are written in lower letters, and these are converted to capital letters. Furthermore, the spelling is corrected of some of the shipment numbers that contain a special character.

4.1.4 Conversion to generic milestones

In the previous sections, important data cleaning was performed and useless shipments were deleted. In this section, the 8061 unique milestones are largely converted to the earlier defined set of relevant milestones as listed in section 3.3.2.

Text mining methodology

The milestones' descriptions will be converted with the help of text mining. Text mining is executed by searching for specific words in the milestone descriptions which indicate activities or are commonly used words. By simply looking at the milestone data and by using Python to search for the most occurring words, a set of words can be identified that always belong to one or more milestones. Examples of these words are truck, cargo, and departure.

First a selection of milestones is extracted by searching for one of the specific words. Second, only the unique milestones of this sub-selection are extracted. Third, by looking at the unique milestones, further milestone specific words can be identified. By doing so, code can be written to automatically categorize milestones to generic milestones. Fourth, all milestones related to the identified words are converted to generic milestones and the output is checked for errors. This is done by outputting the data in Python or writing the output to separate csv-files. At last, several milestones are grouped to generic milestones manually, or are identified as 'useless'. Milestones are classified as useless when they have no added value to the model, such as battery status updates of active containers. These milestones can later be removed when all milestones are classified. Often during the classification process, milestones are too vague or so exceptional that they need to be looked up in SmartView to see their previous and successive milestones to correctly classify them.

Which sub-selections and codes were used to classify milestones can be found in appendix 2. To summarize the text mining process:

1. Identify most occurring word or word that affects many milestones
2. Take a sub-selection of all milestones with this specific word
3. Extract the unique milestones of this sub-selection
4. Check every single unique milestone and write code such that milestones can be categorized to a generic milestone. In case too many unique milestones are found, a further sub-selection can be made.
5. Classify all milestones of the sub-selection, thus not only the unique, based on the created code
6. Check the output. If correct, start over with a new word until all milestones are classified. If not, rewrite the classification code.

Total number of converted milestones

All the 61004 milestones belonging to 5347 shipments are assigned to one of the 15 generic milestones or marked as 'useless'. Table 4.2 shows the total number of milestones assigned to each generic milestone. As indicated before, a milestone 'Aircraft departure from HSV' for lane LUX-MEX-HSV never occurs in the dataset and is therefore not listed in the table. Also the milestone 'Aircraft offloading (HSV)' has only 9 milestones assigned to it.

Furthermore, during data collection it was found that 4 shipments do not have a csv-file containing the temperature sensor data and therefore their milestones can be removed. After removing the milestones of these 5 shipments and all the 'useless' milestones, the final dataset contains 51147 milestones and 5342 shipment numbers.

The user names are anonymized to 'user_x', with x indicating a number ranging from 0 to 41.

4.2 Temperature sensor data preparation

The file containing the milestones is now prepared and the next step is combining the milestones with the temperature sensor data. First, the sensor data needs to be prepared. This chapter describes the data preparation of the sensor data and how the milestones are combined with the sensor data.

4.2.1 Removing sensor data files and converting shipment names

Since many of the shipments were removed during data preparation of the milestones, the temperature sensor csv-files of the corresponding shipment numbers also need to be deleted. Therefore, this is the first step in combining the temperature data with the milestones.

A total number of 9966 csv-files exist containing temperature sensor data of individual shipments. This number is far greater than the 7139 shipments that existed in the milestones file before cleaning, and after cleaning the number of shipments further decreased to 5342. By creating a list of all the 9966 csv filenames and comparing this list with a list of unique shipment numbers that still exist in the milestones file, the temperature csv's that need to be deleted from the map can be identified. The milestones file contains milestones of 5342 unique shipment, which means that of the 9966 temperature data files, 4624 files need to be deleted. However, some issues still exist while comparing the shipment numbers between the temperature data and the milestone data. Of all temperature sensor data files, 6 duplicates existed which had to be removed. SmartView, the platform where all the data was extracted from, also contained some errors in the shipment names, which resulted in different shipment names in the temperature sensor data files and the milestones file. After resolving these issues, the shipment names in the sensor data filenames and the milestones file are now the same.

Generic milestone or 'useless'	Number of milestones assigned
Truck opened and start unloading (LUX)	5107
Cargo stored in temperature controlled area (LUX)	5150
Cargo build-up on ULD and stored in temperature controlled area (LUX)	5082
ULD out of cell for uplift to flight (LUX)	5178
Aircraft departure (LUX)	5806
Aircraft arrival (HSV)	4838
Cargo stored in temperature controlled area (HSV)	3643
Actual aircraft arrival (MEX)	1736
Aircraft offloading (MEX)	1731
Aircraft offloading (HSV)	9
Cargo at TME (MEX)	2152
Release of cargo to customs broker (MEX)	1787
Doc release to broker (MEX)	1699
Loading truck start (HSV)	3663
Loading truck complete (HSV)	3616
Useless	9808

Table 4.2: Number of milestones classified to generic milestone

4.2.2 Sensor data transformation

The temperature data files consist of all the sensor data collected on each shipment, along with a timestamp in UTC time format and local date and time format. These time formats, however, seem to be always the same. A shipment can contain multiple sensors and the data is registered in csv-format. When multiple sensor devices are used on a shipment, the data is written in columns right next to each other, separated by commas. The sensor data files further indicate the name of the sensor device, the temperature unit, and a timestamp indicating the time it was downloaded from SmartView.

The column format is transformed into a row format, such that every single temperature sensor instance is on a single row. Also, when multiple sensor devices are used, the data is written to new rows when the instances of the previous device are written. The rows contain the UTC time, local time, the measured temperature, the sensor device number, and the shipment number. The shipment number is added in order to assign a milestone to a sensor measurement.

Since one sensor creates on average 500 rows of sensor information, it will computationally not be possible to combine all 5342 csv-files, which often contain multiple sensors, to one csv-file. This would result in a csv-file of several million rows and such a large file makes working with it in Python computationally slow. Therefore, the sensor data is converted for each file separately, and the milestones will also be linked to each file separately.

4.2.3 Sensor data cleaning

Many sensor files contain rows with missing data. The missing data was created in SmartView because the sensor data of all devices is written to a csv-file in column format. When a shipment contains multiple sensor devices, the number of rows in this column format corresponds with the number of instances of the device that has the most sensor instances. The csv-file therefore contains missing data for the sensors that have less instances. The sensor files are converted to a row format for every single device as described above. The empty rows are deleted from every csv-file.

Sensors devices can be turned on before the first milestone was registered in SmartView, and therefore these sensor instances are removed.

While combining the sensor and milestone data, which is described in the next section, it was found that many sensor files contain missing sensor instances. The sensor devices should register the temperature every 15 minutes. Because sensor instances and their timestamps are missing, it is much harder to assign milestones to the sensor data based on the common timestamp. The 5342 csv-files with sensor data are checked for missing instances and it is found that they all contain missing instances. Therefore, the missing rows of information are added to every single csv-file. A total number of 404952 rows of sensor information was added, 76 rows per shipment on average. The temperatures of these rows get the value 'xxx'. The added rows can be removed from the dataset once the milestones are assigned to the sensor data. An example of two rows that were added to a shipment is provided below in table 4.3.

UTC Time	Local Time	Temp	Device Number	Shipment Number
2015-08-29 08:36:18	2015-08-29 08:36:18	2.00	10:00:00:01:05:18	FRA 135856 MEX INSIDE
2015-08-29 08:51:18	2015-08-29 08:51:18	2.06	10:00:00:01:05:18	FRA 135856 MEX INSIDE
2015-08-29 09:06:18	**2015-08-29 09:06:18**	**xxx**	**10:00:00:01:05:18**	**FRA 135856 MEX INSIDE**
2015-08-29 09:21:18	**2015-08-29 09:21:18**	**xxx**	**10:00:00:01:05:18**	**FRA 135856 MEX INSIDE**
2015-08-29 09:36:17	2015-08-29 09:36:17	2.31	10:00:00:01:05:18	FRA 135856 MEX INSIDE
2015-08-29 09:51:16	2015-08-29 09:51:16	2.94	10:00:00:01:05:18	FRA 135856 MEX INSIDE

Table 4.3: Example addition of missing instances

4.3 Combining the sensor and milestone data

Now that the sensor data is transformed and cleaned, milestones can be assigned to the sensor instances. This process is explained in this section. The milestones are assigned to the sensor data for each shipment individually, thus separately for each csv-file. Assigning milestones to the sensor data follows the following steps which are further described below:

1. Split a shipment's sensor data csv-file into separate csv-files for each sensor device
2. Select from the milestones file the milestones that belong to the shipment
3. Assign milestones to sensor instances based on the timestamp
4. Give all other sensor instances in between these instances a milestone
5. Combine the sensor csv-files back to one csv-file for each shipment

The sensor data consists of temperature measurements with an interval of 15 minutes. Sometimes this interval is a few seconds longer or shorter. Shipments can have multiple sensor devices. In order to assign milestones to the sensor data, it was found that it is easier to split the sensor devices of a shipment into separate csv-files. Therefore, a shipment's csv-file containing the data of multiple sensors is split into a separate csv-file for each sensor device.

Next, the milestones belonging to the shipment are extracted from the file containing the milestones of all shipments. This can be done by searching for the shipment number after converting the csv-file name to the shipment number format used in the milestones file.

Once the shipment's milestones are obtained, they are assigned to the sensor data based on the timestamp. Milestones are assigned to sensor instances when the difference between the milestone timestamp and the sensor instance timestamp is less or equal to 7,5 minute, with a tolerance of 2 seconds to cover the occasionally inconsistent measurement interval. The result is that a very small number of sensor instances receive a milestone because only few instances have a timestamp close to the milestone's timestamp. As an example, table 4.4 indicates the sensor indexes and corresponding milestones that are assigned to a random shipment's sensor device. The table contains milestones written as mX, with X indicating the number of a milestone. These are abbreviations that correspond with the relevant milestones identified in chapter 3.3.2, and their exact meaning is for now not important to understand how the milestones are linked to the sensor data. Only the sensor instances with the indexes as indicated in table 4.4 have received a milestone. Eventually we want a dataset in which all sensor instances have a milestone attached. Otherwise, it is unknown during which step in the logistic process the sensor measurement was registered. Thus, the sensor instances in the range between the indexes of table 4.4 must receive a milestone.

Index	1	14	87	92	126	140	411	415
Milestone	m4	m5	m6	m7	m8	m9	m3	m2

Table 4.4: Initial milestones assigned to sensor instance index

A milestone is assigned for every next index number until the index of the next milestone is reached. For example, indexes 2 to 13 receive milestone m4. For most milestones this is a correct way to combine milestones with sensor instances, since for instance cargo is stored until the cargo build-up starts. However, milestones such as aircraft departure and aircraft arrival cannot be stretched over the period until the next milestone starts. The departure and arrival occur in only a small timeframe. The aircraft departure is the moment when the handbrake of the airplane is released at the gate, and the aircraft arrival is the moment the handbrake is enabled at the gate. Every sensor instance in between departure and arrival is assigned the milestone 'Flight'. When the milestone aircraft arrival is missing, the instances between departure and the next milestone is given 'Dummy'. These are called dummy milestones.

Dummy milestones are also written to sensor instances when milestones illogically follow each other. For instance, when the ULD uplift to flight is missing, instead of adding 'Cargo stored in temperature controlled cell (LUX)' until the aircraft departs, dummy variables are added.

After the aircraft arrival in HSV the aircraft is offloaded. This milestone, however, is only registered for 9 shipments. Most shipments on lane LUX-HSV contain the milestone 'Cargo stored in temperature controlled area (HSV)'. Therefore, when this milestone occurs after the aircraft arrival, the sensor instances in between arrival and storage receive the milestone 'Aircraft offloading (HSV)'. When this milestone is not present, the instances in between aircraft arrival and the next milestone are assigned 'Dummy', because it is unknown what exactly happens with the cargo.

The shipments on lane LUX-HSV-MEX do not contain the milestone 'Cargo stored in temperature controlled area (HSV)'. Also, the aircraft departure from HSV to MEX is always missing. Therefore, it remains unknown when the aircraft departs from Huntsville. Because a milestone cannot be assigned with certainty to sensor instances from the moment between aircraft arrival in HSV and arrival in MEX, all intermediate sensor instances receive the milestone 'Dummy'.

All other milestones are assigned to sensor indexes until the next milestone starts. After this process, still a number of instances lack a milestone due to mismatches in milestone timestamps and sensor device start and end times. For instance, a shipment contains two milestones: 'Truck opened and start unloading (LUX)' clocked at 08:00 hours and 'Cargo stored in temperature controlled area (LUX)' clocked at 10:00 hours. If a sensor device is switched on at 09:00 hours, sensor instances from 09:00 to 10:00 hours receive the milestone 'Truck opened and start unloading (LUX)' because the next milestone 'Cargo stored in temperature controlled area (LUX)' has not yet started.

With the above described steps, all sensor instances receive a milestone. The individual sensor device csv-files are combined back to one csv-file for each shipment. Each csv-file has the columns: UTC timestamp, local timestamp, measured temperature, sensor device number, shipment number, and milestone.

4.4 Splitting the dataset

To build a prediction model with the sensor and milestone data and additional data sources, it is useful to have all data in one dataset. When the 5342 csv-files are merged to one csv-file, the resulting file is about 630MB large and contains over 5,3 million rows. This makes the file unusable because when the file is opened, the computer's RAM-memory rises to 95% and no more computations are possible. Therefore, the dataset should be reduced.

To do so there are several possibilities: using a random sample of shipments, splitting the data in the two lanes, removing sensor data of milestones where no excursions occur, removing sensor instances with the unknown 'Dummy' milestones, removing product types with too few shipments, and removing shipments with too few milestones. However, data visualization must be performed in order to discover what data can be removed or should be cleaned. Since the file is too big to perform this visualization, the dataset is split up in separate csv-files for each shipping lane and product type. Data visualization is possible on the separate csv-files.

During the creation of the different datasets, it was found that 8 shipment numbers have deviating names. This was already mentioned and described in chapter 4.1.3. The numbers of the 8 shipments are converted based on their milestones such that they contain either 'HSV' or 'MEX'. By doing so, the shipment's lane can be determined just by looking at the shipment number.

It was also found that 41 sensor csv files did not contain any sensor data. After checking for errors in the code, the 41 files and their milestones were deleted. The total remaining number of shipments is 5301.

The data can now be split in the two lanes LUX-HSV and LUX-HSV-MEX. Table 4.5 shows the number of shipments per product type for each shipping lane. Based on the available amount of data, it is decided with Validaide to focus on product types cold and CRT. Separate datasets are created for the lane and product type combinations indicated in bold.

Product type	*LUX-HSV*	*LUX-HSV-MEX*
+2° to +25° C	380	292
+15° to +25° C (CRT)	**2082**	**772**
+8° to +25° C	333	6
+2° to +8° C (cold)	**779**	**262**
+2° to +30° C	2	42
+10° to +25° C	12	47
+15° to +30° C	2	0
< +40° C	1	152
-25° C to -20°C	1	0
< +30° C	1	135

Table 4.5: Number of shipments per product type

4.5 Integration with additional data sources

Besides the milestones and the sensor data, additional data was collected as described in chapter 3.1. These additional sources are the general data of each shipment which can be extracted from SmartView, and historical weather temperature data. Below is discussed how these data sources are integrated with the combined sensor and milestones data, which is from now on called the 'dataset'. Furthermore, several other variables that may have predictive value are added to the dataset.

4.5.1 Shipments' general information

The first additional data source that is added to the data is the general data of each shipment. This information was downloaded from SmartView at once for an individual lane. Therefore, the files of the individual lanes are first combined into one csv-file. The general information consists of the following 18 columns: ID, ShipmentID, Product, Order, Customer, Origin, Destination, Transport Method, Transport Company, Flight Code, Flight Data, Associations, Disassociations, Start Time (UTC), Stop Time (UTC), NrExcursions, Sensors. Only the four columns ShipmentID, Product, Customer, and NrExcursions are kept and will be added to the data. The product is the product type which indicates the temperature range in which the shipment must be transported. Panalpina's customer is often the pharmaceutical manufacturer of the cargo. NrExcursions indicates the number of excursions counted by SmartView. The 'ShipmentID' is the shipment number, which is required to add the general information to the data.

The general data requires some data cleaning. After selection of the useful variables, all the general information of the shipments that are no longer present in the dataset are removed. Of the 9991 shipments in the general information file, only the 5301 shipments that are also in the dataset are kept. A number of 31 shipment number were adjusted to make them equal to the shipment numbers in the combined data. Furthermore, 152 product types and 175 customer names are missing in the file. The product types are looked up in SmartView and are manually added to the general data. The missing customer names remain unknown. At last, it is unsure how useful the column with the number of excursions is, and these numbers need to be checked with the dataset.

The general information is added to the dataset based on similar shipment numbers. The customer names are anonymized to 'customer_x', with x indicating a number ranging from 0 to 61.

4.5.2 Historical weather data

The second additional data source is the historical weather temperature data. As described in chapter 3.2, the weather data was interpolated such that the temperature data is available for every 15 minutes. The weather temperature value of the required location that has the UTC timestamp closest to the UTC timestamp of the instance in the dataset, is added to every instance. All instances with a milestone related to temperature controlled areas, 'Cargo stored in temperature controlled area (LUX)', 'Cargo build-up on ULD and stored in temperature controlled area (LUX)', 'Cargo stored in temperature controlled area (HSV)', and 'Flight', receive for the weather temperature a value of 100. The weather temperature is irrelevant since the products are stored in the temperature controlled warehouses. Since the controlled temperature is not constant and unknown, a temperature value cannot be assigned. Assigning the value of 100 can be seen as using a global constant to fill missing values ((Han, Kamber, & Pei, 2012). The machine learning model will recognize the value 100, being much larger than the other temperatures, as a dummy value.

4.5.3 Additional variables

To create a complete dataset, several other variables are added which can be created with features from the current dataset. These variables might contribute to the creation of the prediction model, or help to perform data visualization as described in the next chapter. The variables consist of: month, year, weekday, season, sensor location on the cargo (either inside or outside the cargo), minimum and maximum temperature according to the product type, deviation from the product type, and the flight destination. The minimum and maximum temperatures of for instance product type +15 to +25°C are respectively 15 and 25°C. The deviation is the temperature measured by the sensor below the minimum or above the maximum temperatures.

5 Further data understanding and preparation

A complete dataset was prepared in the previous chapter. In this chapter, the dataset visualized and additional data preparation tasks are identified and performed.

5.1 Visualization of the dataset

In the last step of combining the sensor data with the milestones data, the total datasets for different transport lanes and product types were created. To completely understand what the data consists of, data understanding can be continued on the new datasets. Some combinations of lane and product type have a limited amount of data, such as lane LUX-HSV-MEX for product type +2° to +25°C. With the help of data visualization, it can also be determined whether there is enough data available for these lanes.

In this chapter, data visualization is performed on dataset LUX-HSV +15° to +25°C first, since this dataset with its 2082 shipments consists of the majority of all data. The same visualization can be performed the other product types and lane. However, this is a time consuming process, and therefore, we decide to continue the rest of this project with only the data of LUX-HSV +15° to +25°C. This also means that the prediction model will only be created for this dataset.

Data visualization can provide a lot of information about the data. For instance, we can determine during which milestones most excursions occur, how large and how long these excursions are, or in which season most excursions occur. More important, data visualization can show the distribution of an excursion over a milestone, which might indicate whether or not the milestones are correctly

assigned to the sensor data. Data visualization can also provide insights that lead to another approach of making a prediction model. In this chapter the following data visualizations tasks are performed:

- Number of sensor instances per shipment
- Number of sensor instances per milestone
- Distribution of shipments over time
- Occurrence of temperature excursions over time
- Size of temperature excursions
- Average temperature profile per milestone
- Average temperature profile per shipment
- Temperature excursions over time per milestone

The figures in this chapter contain a lot of coded milestones instead of their full names. The full milestone names and their coding are given in table 5.1 to understand the figures.

Coding	*Full milestone name*	*Abbreviated name*
m1	Truck opened and start unloading (LUX)	Truck unloading
m2	Loading truck complete (HSV)	Truck loading complete
m3	Loading truck start (HSV)	Truck loading start
m4	Cargo stored in temperature controlled area (LUX)	Storage LUX
m5	Cargo build-up on ULD and stored in temperature controlled area (LUX)	ULD b/u and storage
m6	ULD out of cell for uplift to flight (LUX)	Cargo uplift
m7	Flight departure from LUX	Departure
m8	Flight arrival in HSV	Arrival
m9	Cargo stored in temperature controlled area (HSV)	Storage HSV
m16	Aircraft offloading (HSV)	A/C offloading

Table 5.1: Milestones and their abbreviations

5.1.1 Number of sensor instances per shipment

Figure 5.1 shows the distribution of sensor instances per shipment. It is clearly visible that a small number of shipments contain a lot more sensor measurements than others. Of all 2082 shipments, the mean number of sensor instances is 1074, the minimum is 37, and the maximum is 42031 sensor instances. The median is 684, and the 75^{th} percentile is 1043. The shipment with the most sensor instances contains data collected by 18 sensor devices. Of all shipments, 232 shipments have more than 1500 instances and 22 shipments have less than 150 instances. The data of the 10 shipments with the least sensor instances is checked, but even these shipments contain almost all milestones.

5.1.2 Number of sensor instances per milestone

The number of sensor instances per milestone is also interesting, since it gives an idea of the duration of each milestone. Each instance means a duration of 15 minutes. As seen in figure 5.2, several milestones have significantly more sensor instances than other milestones. The milestones m5, m4, and m9 are respectively storage LUX, ULD b/u and storage, and storage HSV. Since storage is a relatively long activity, it is not strange that the dataset consist of almost 80% of these three milestones. Another milestone with relatively many instances is 'Flight', which is also a relatively long activity. A notable observation is that m2, truck loading complete, has a lot more sensor instances than m3, truck loading start. The latter lasts until the former starts. Panalpina indicates that most sensors are removed from the cargo while loading the trucks before they leave the warehouse, which means

that the cargo is not monitored on its way to the destination. However, some sensors may not be turned off and therefore they still register the temperature. These measurements are ignored.

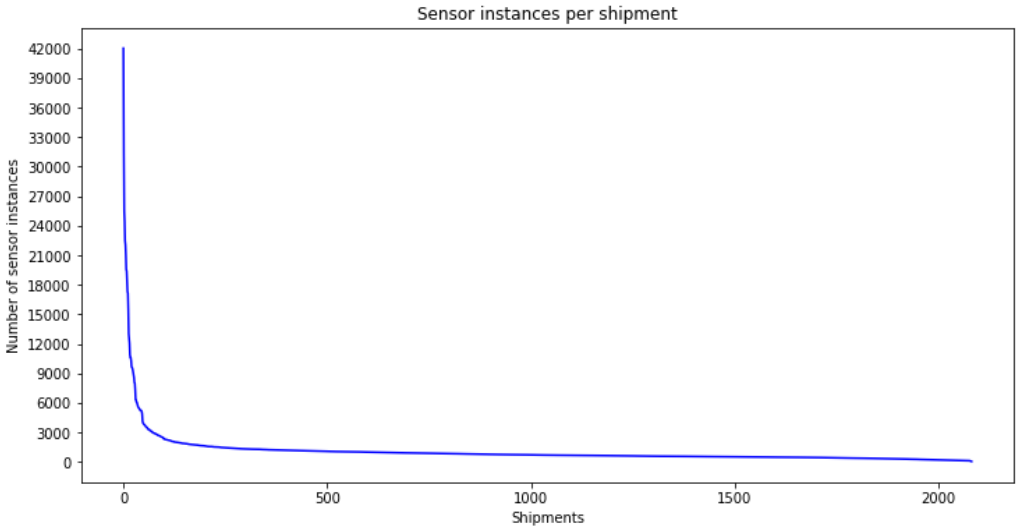


Figure 5.1: Number of sensor instances per shipment

5.1.3 Shipments over time

Figure 5.3 shows the distribution of shipments per year and month. The total number of shipments for the years 2014, 2015, 2016, 2017, and 2018, are respectively, 352, 567, 577, 486, 100. The dataset contains less data of the year 2014 because in this year less data was gathered by Panalpina. Its quality is also lower than the quality of the data of the more recent years which resulted in the removal of many shipments of the year 2014 as described in chapter 4.1. The year 2018 contains data of only few shipments because shipments were included in the dataset until the end of March. As depicted in figure 5.3, in some months more shipments occurred than in other months. This might result in a higher predictive accuracy of the model for the months with lots of data. All months contain at least around 30 shipments, thus sufficient data is expected to be available.

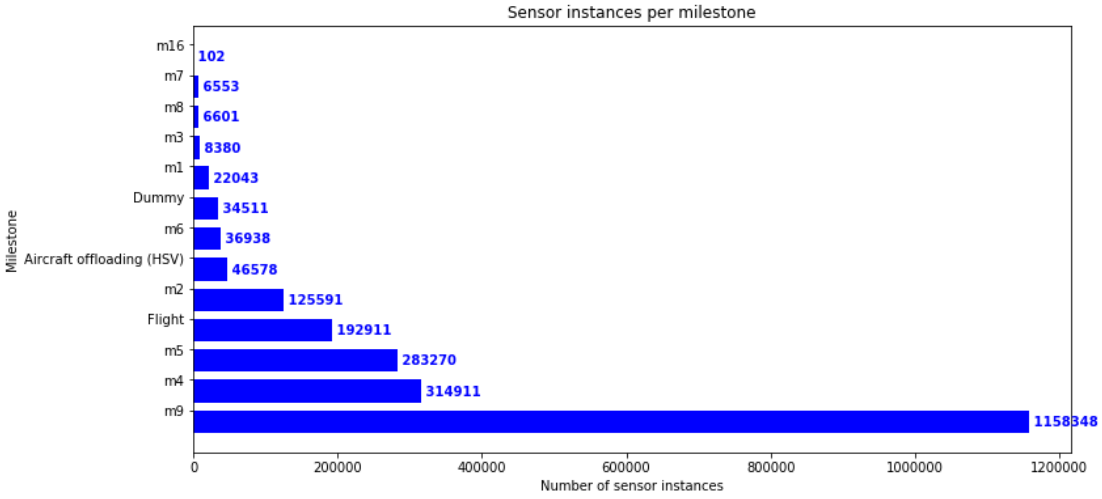


Figure 5.2: Number of sensor instances per milestone

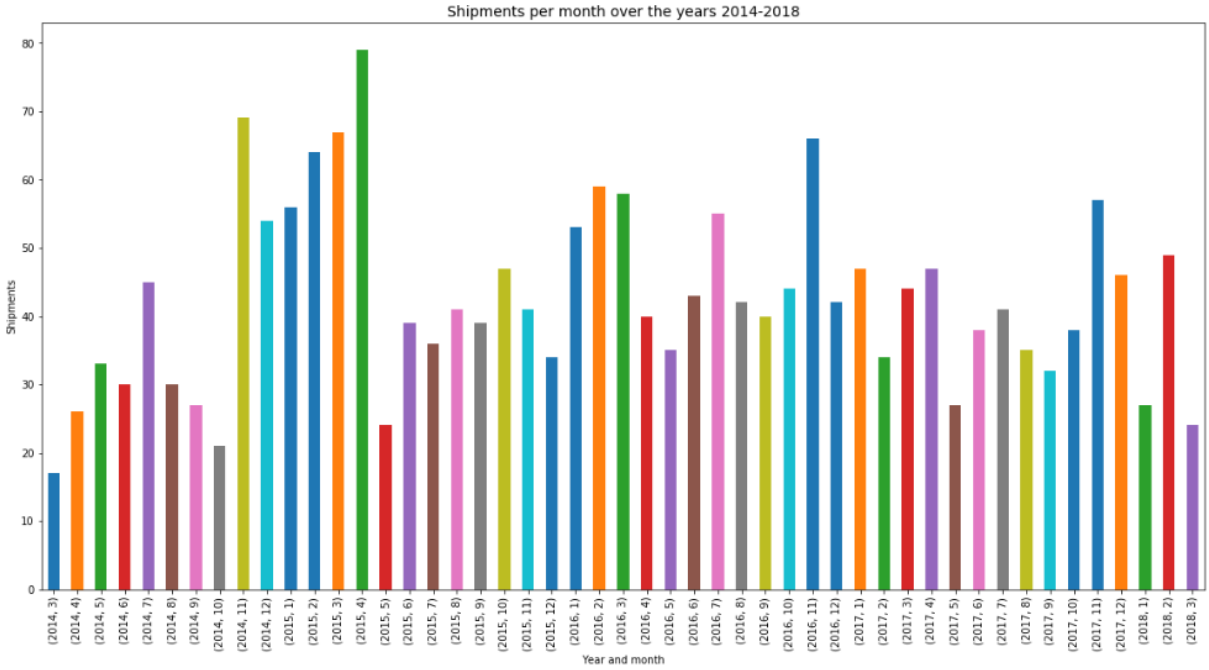


Figure 5.3: Number of shipments per year and month

5.1.4 Occurrence and size of temperature excursions

Excursion occurrence

In chapter 4.5, the deviation from the product range was added as a variable to the dataset. If a temperature was measured below the minimum allowed temperature or above the maximum allowed temperature, the deviation from the product range was registered. When a temperature excursion is described as a deviation from the product range, the total number of temperature excursions is 35993 on a total of 2236737 measurements. Figure 5.4 shows the temperature excursions and their size. For this product range, many excursions seem to occur because the temperature gets below the minimum allowed temperature. Also, many deviations from the product range are relatively large.

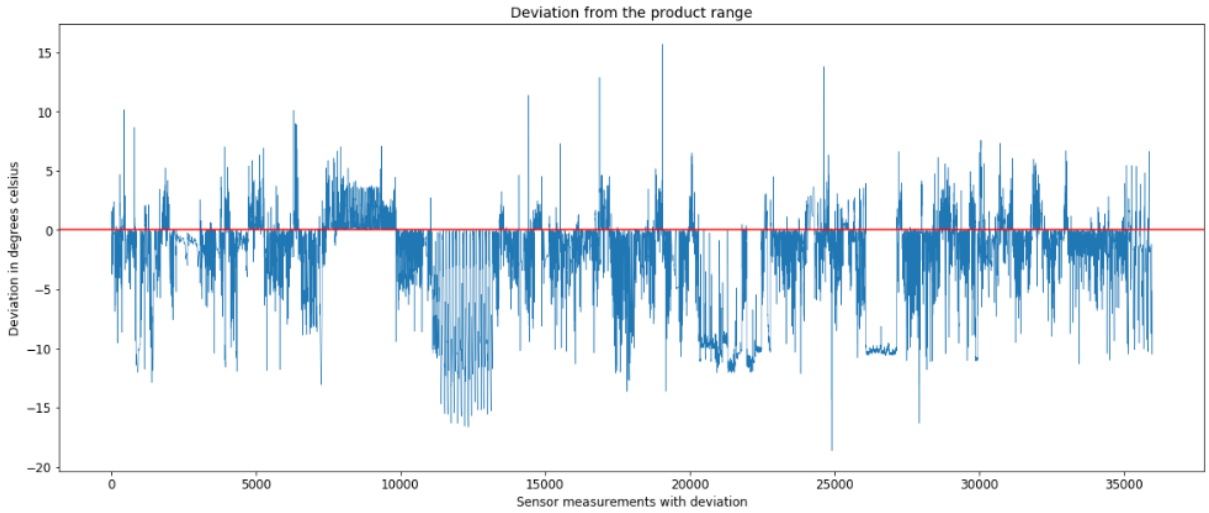


Figure 5.4: Size of temperature excursions

When looking at the distribution of the number of temperature excursions per milestone as shown in figure 5.5, several milestones have significantly more temperature excursions than others. However,

it is better to look at the number of excursions per milestone relative to the total number of sensor instances for that milestone. By summing the fractions of each milestone and setting this to 100%, the percentage of excursions relative to the other milestones can be plotted as shown in figure 5.6. This figure clearly shows that most excursions occur during activities executed outside temperature controlled areas, which is to be expected. Most excursions occur during milestone m6 cargo uplift, an activity during which the cargo is exposed to outside tarmac temperatures. Panalpina only registered 102 sensor instances for milestone m6 A/C offloading during which zero excursions occurred.

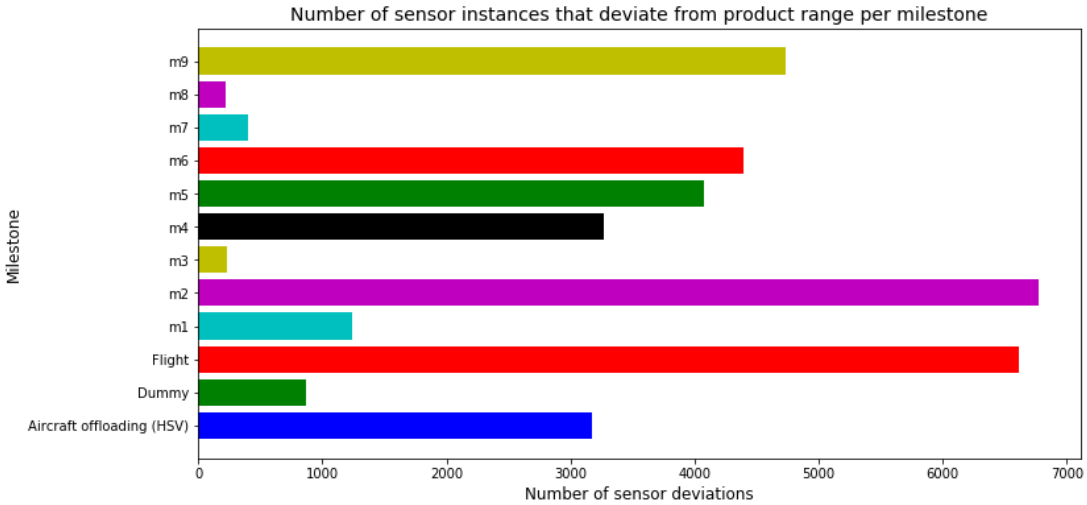


Figure 5.5: Number of temperature excursions per milestone

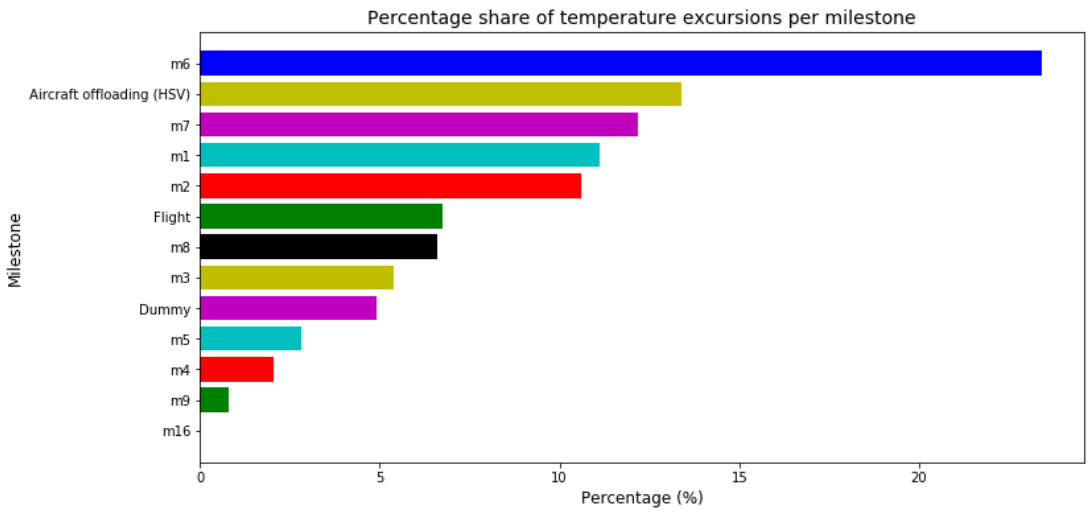


Figure 5.6: Relative percentage share of temperature excursions per milestone

By creating a graph of the number of excursions per month, as shown in figure 5.7, it is possible to visualize in which periods of the year most excursions occur. It is expected that most excursions occur during cold winter periods and warm summer periods. The figure indicates that excursions mainly occur during the months November, December, January, February and March. Besides a small peak in July, significantly less excursions occur during the rest of the year. Since the product temperature range lies between 15 and 25 °C, it makes sense that these products are more likely to be affected by cold temperatures.

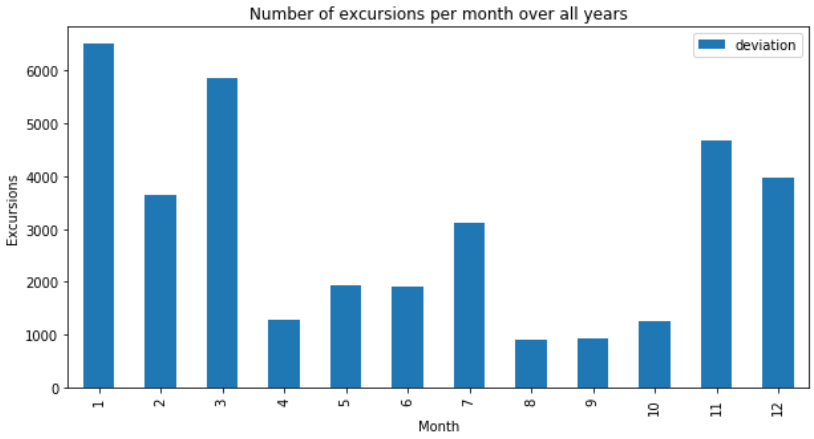


Figure 5.7: Number of excursions per month over all years

Excursion size

The average absolute deviation from the product range, including only the sensor instances with temperature excursions, is shown in figure 5.8. The figure indicates that the largest temperature excursions occur during cold storage related milestones m9 storage HSV, m5 ULD b/u and storage, and m4 storage LUX. The milestone m2 truck loading complete and 'Dummy' can both be ignored, since during m2 the sensor is not on the cargo, and 'Dummy' indicates that the milestone is unknown.

When all sensor instances are included, the average absolute deviation looks much different, as shown in figure 5.9. In this figure, the largest average absolute temperature excursions occur during A/C offloading and m6 cargo uplift. These are activities during which the cargo is much more vulnerable for temperature excursions because they take place outside of the temperature controlled areas. The average absolute deviation in the temperature controlled areas are now also much smaller (m4, m5, m9). The milestones with the largest average deviations now also correspond much better with the milestones that have relatively the most excursions.

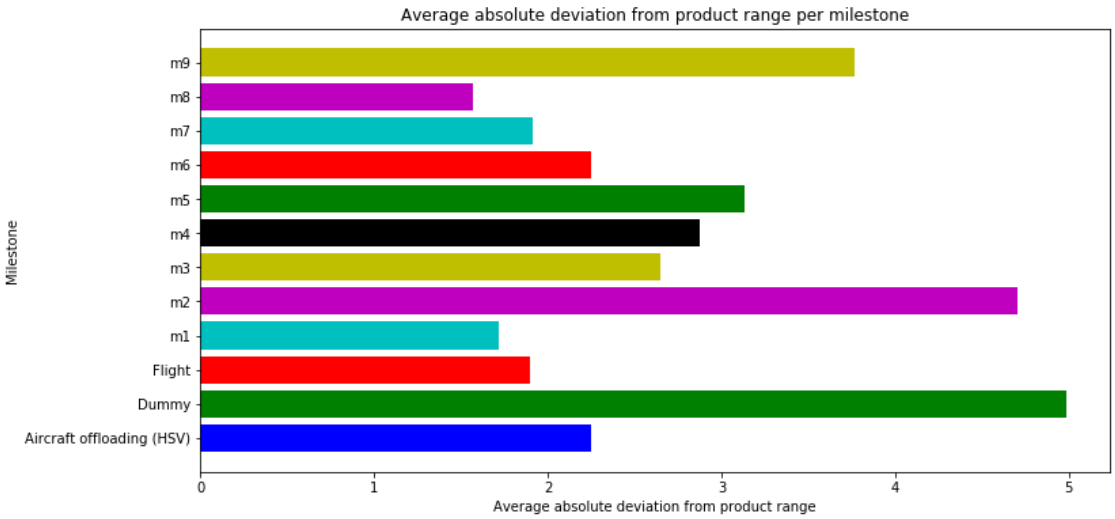


Figure 5.8: Average absolute deviation from product range of all excursions

5.1.5 Average temperature profile per milestone

To find out whether or not the milestones are correctly assigned to the sensor data and to see what on average happens during a shipment, it is useful to determine the distribution of temperatures and temperature excursions over time. For instance, it is unsure if it was correct to assign m7 departure

to only one sensor instance, instead of multiple instances. Therefore, if the next milestone 'Flight' contains a lot of excursions at the beginning of the milestone, it might indicate that the duration of m7 must be increased.

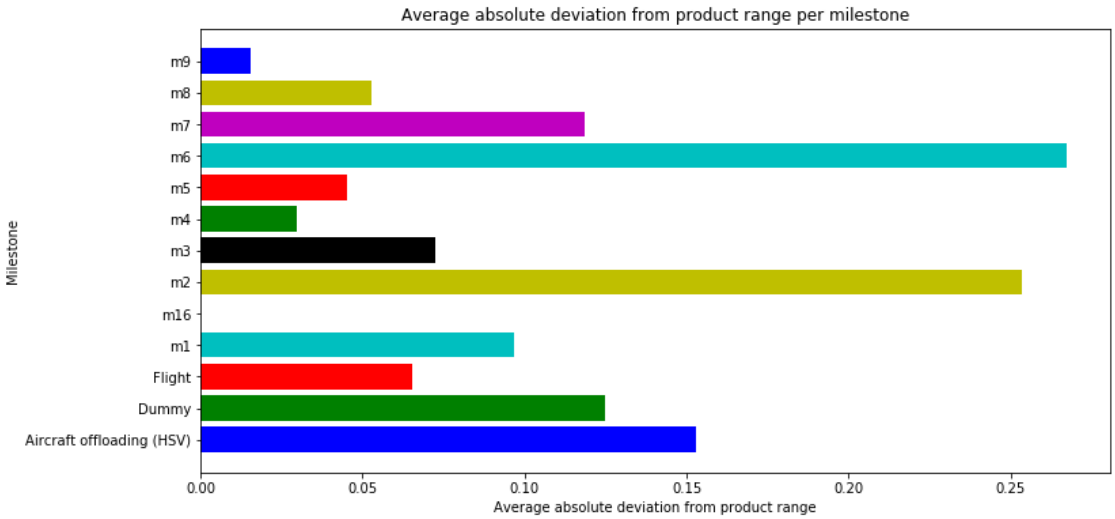


Figure 5.9: Average absolute deviation from product range of all instances

Methodology

Temperature profiles are created for each milestone during the warm months and the cold months since these differ most. The temperature profiles of all milestones are then combined to a temperature profile of a whole shipment, also for both the warm and cold period. Warm months are June, July, August, and September, cold months are November, December, January, and February.

To plot the average temperature profile for each milestone, the following steps are executed:

1. Determine the number of sensor instances of the milestone over all months.
2. Determine the number of sensor devices of the milestone over all months.
3. Determine the distribution of the number of sensor instances per device of the milestone over all months.
4. If the distribution is very spread, make a selection of devices with a certain number of sensor measurements.
5. Create a list of measured temperatures for each device in specific months. Devices can have a different number of measurements, so the length of the list varies.
6. Interpolate to 1000 values for every device. This creates lists of equal length.
7. Calculate the average temperature of every every x-th value over all devices for the specific months (blue line).
8. Determine the average temperature of all measurements for the milestone for the specific months (green line).
9. Create a graph that shows the average temperature profile per device (blue line from step 7) and the average temperature of all sensor measurements of that milestone (green line from step 8). The y-axis indicates the temperature, the x-axis the percentage that a milestone is finished.

The average temperature profiles of each separate milestone and the steps and statistics that were required to create them, can be found in appendix 3. An explanation for each temperature profile is also provided.

5.1.6 *Temperature excursions over time per milestone*

After making the average temperature profiles, it is still unsure where exactly during the milestones the temperature excursions occur. Therefore, the graph in figure 5.10 is created, which shows the percentage of excursions that have occurred against the percentage of the milestone that is finished. There is no significant difference between the cold and warm period. As an example, the black line in the figure below belongs to milestone storage HSV (m9), and shows that about 50% of all excursions belonging to this milestone occur in the first 2% of the duration of the milestone. Milestones storage LUX (m4) and ULD b/u and storage (m5), both related to cold storage activities, indicate that during storage only few excursions occur, and most occur at the start and end of the milestone. Truck offloading (m1), cargo uplift (m6) and offloading (m9) show a constant distribution of the excursions over the milestone duration, which makes sense since it is impossible to load and unload all cargo at once, and therefore different parts of the cargo are exposed to tarmac temperatures. During milestone 'Flight', a large part of the excursions occur early, but still many excursions occur later during the flight, which seems odd since the aircraft has temperature controlled compartments.

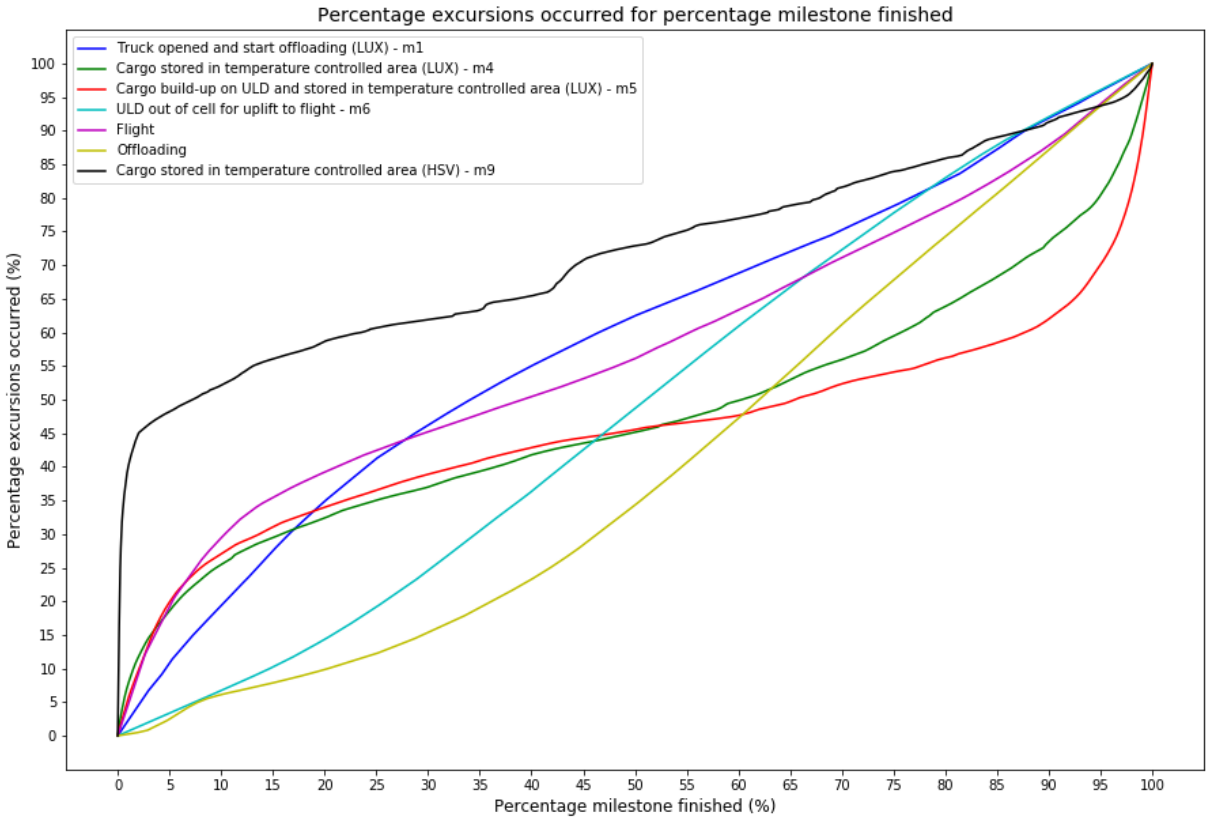


Figure 5.10: Percentage of excursions occurred against percentage of milestone finished

5.1.7 *Average temperature profile per shipment*

For every milestone of a shipment, an average temperature profile has been created for both the warm and cold periods. A complete graph of the average temperature profile for a shipment is made by combining the graphs of the average temperature profile of each milestone. These are shown in figure 5.11 and 5.12 for the warm and cold period respectively. The vertical red lines indicate the points where one milestone changes to another milestone, and the horizontal green lines indicate the average temperature during each milestone.

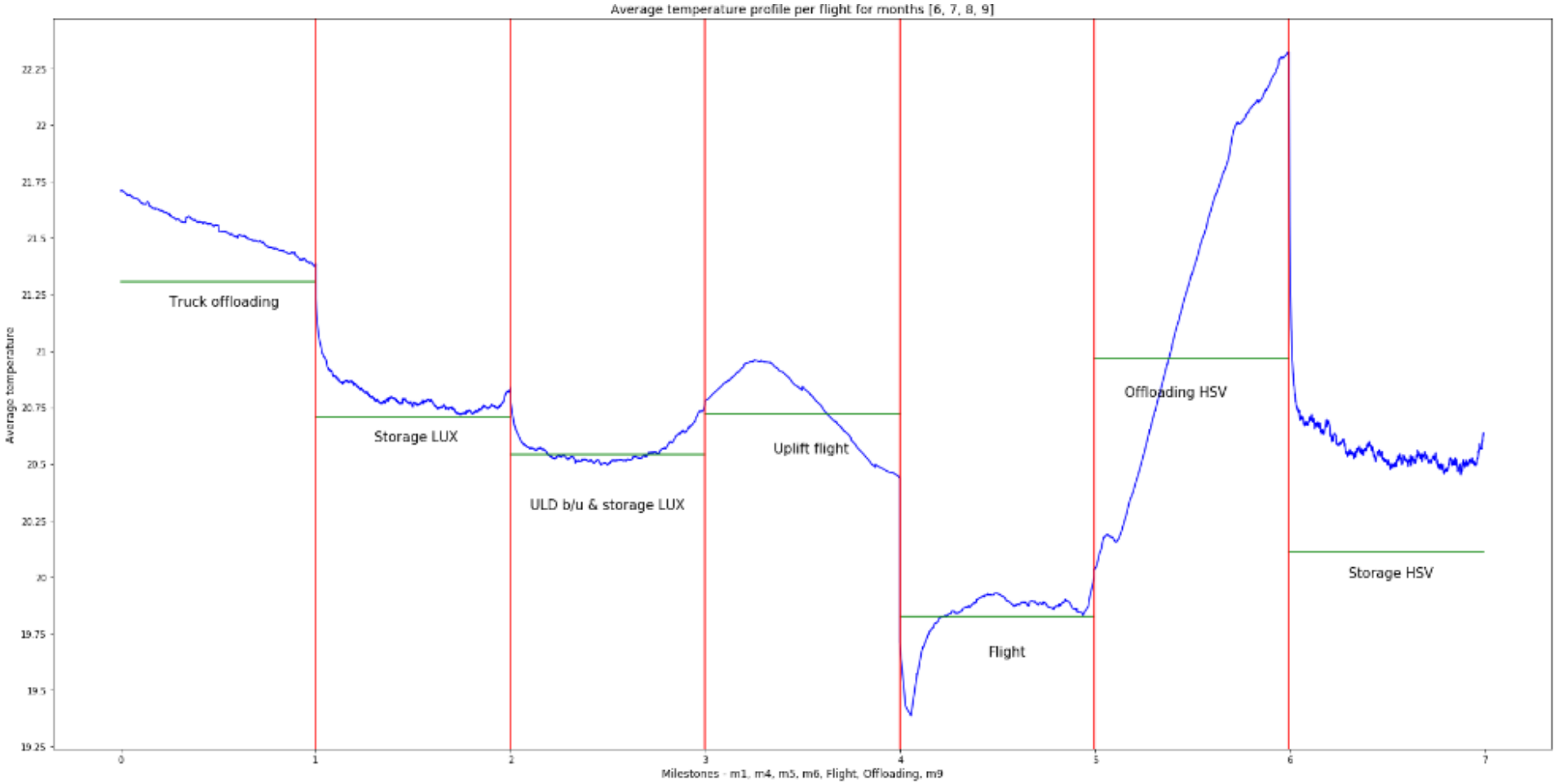


Figure 5.11: Average temperature profile per shipment of the warm period

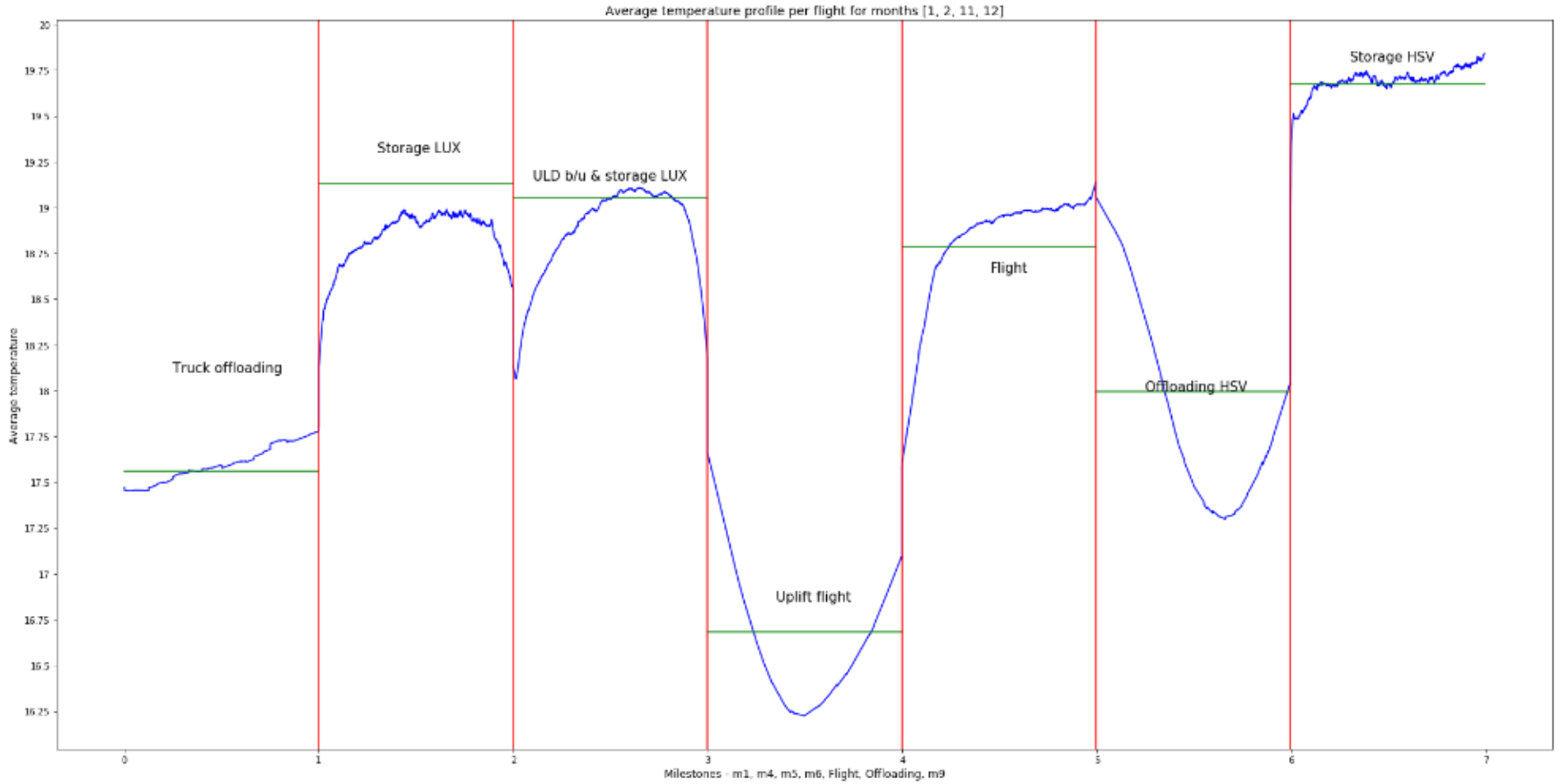


Figure 5.12: Average temperature profile per shipment of the cold period

5.2 Data preparation after visualization

The visualization as described in the previous section showed some discrepancies in the data and several other possibilities to improve the dataset. Therefore, additional data preparation is performed. The dataset contains 2236737 instances, each instance or data point is one row in the dataset containing a sensor measurement and a value for the additional variables in the columns. This chapter describes which data is changed or removed. The goal is to improve the dataset such that it more accurately represents actual shipments, hence leading to better predictions.

5.2.1 Data removal and milestone transformation

First of all, all milestones m2 'Truck loading complete (HSV)' and m3 'Truck loading start (HSV)' are removed from the dataset since the sensor devices are removed from the cargo when activity m3 starts, and therefore these sensor measurements are useless. All sensor instances labeled with milestone 'Dummy' are also removed from the dataset since these instances have no added value. A total of 168482 sensor instances are removed.

Furthermore, there were two milestones describing the aircraft offloading in Huntsville: m16 'Aircraft offloading (HSV)' and 'Aircraft offloading (HSV)'. The former was registered by Panalpina itself, and the latter was added to all instances in between milestones m8 'Aircraft arrival (HSV)' and m9 storage HSV. Only 102 sensor instances exist for milestone m16. These are joined with milestone A/C offloading.

Currently the milestones m7 departure and m8 arrival are assigned to only one sensor instance. Sometimes the milestones are assigned to two instances, one automatically entered by the airline in SmartView, and one manually by a SmartView user. Both are kept. Devices with more than 2 instances of milestone m7 or m8 are removed from the dataset. This results in a removal of 933 instances with milestone m7, and 891 instances with milestone m8. Keeping the milestones m7 and m8 as separate milestones makes no sense, especially since the timestamp is not accurate enough. The exact flight departure and arrival times are unknown, and the transition from m7 to flight and from flight to m8 is not that clear in the temperature graphs. It is expected that m7 and m8 do not have that much influence on the model on their own, and can better be joined with other milestones. Milestone m7 is converted to m6 cargo uplift, and m8 is converted to A/C offloading. The total number of converted milestones are respectively 5620 and 5710.

5.2.2 Data removal per milestone outside certain range

During data visualization, it was found that some devices have an unusual number of sensor measurements, as also described in appendix 3. Therefore, the affected milestones registered by these devices are removed from the dataset. All other milestones registered by the devices remain in the dataset.

A total of 2159 sensor devices registered milestone m1 truck offloading. Since offloading of a truck cannot take as long as some devices indicate, all devices with more than 10 instances are removed from the dataset. This means that 1897 devices are kept, with a maximum unloading duration of 2,5 hours since sensor measurements are registered every 15 minutes.

Milestone m4 storage LUX has a median of 34 sensor measurements per device. The maximum number of sensor measurements however is 4214, which means a duration of about 40 days of storage. Some devices registered only 1 instance for milestone m4. Since this storage duration of 15 minutes is not interesting for the model, the 183 devices with only one instance are removed. All

devices that registered more than 250 sensor measurements of milestone m4, are also removed from the dataset. Of the 3875 devices, 3375 are kept.

For milestone m5 ULD b/u and storage, 5261 devices registered sensor measurements. This milestone has up to 6916 sensor measurements per device, with a median of 42 measurements. All devices with up to 100 sensor measurements, which is a duration of one day, are kept. Since this milestone is related to cold storage and since it is impossible to perform ULD build-up in 15 minutes, all devices with only 1 sensor instance are removed from the dataset. A total of 4539 devices remain in the dataset.

The next milestone of which devices are removed is milestone m6 cargo uplift. The uplift to the flight is a relatively short activity. The median number of sensor measurements that were registered per device is 7. The uplift to the aircraft should not take longer than three hours. Therefore, all devices with more than 12 sensor measurements are deleted from the dataset. Of the 5608 devices, 5375 are kept.

The flight duration from LUX to HSV should be about 9 hours depending on the wind direction. Since the aircraft departure and arrival times are registered at the time the handbrake at the gate is released or applied and the aircraft does not immediately take off, a margin is included. An aircraft could be waiting for permission to take off while taxiing towards the runway, or could be waiting in front of a gate before parking. Therefore, all devices that registered a flight duration of less than 7,5 hours (30 measurements) or more than 11 hours and 15 minutes (45 sensor measurements), are deleted from the dataset. Of the 5431 devices, 5238 remain in the dataset.

The aircraft is offloaded after arrival in HSV. This activity is registered by milestone A/C offloading by 5548 devices. The median of 8 sensor instances indicates that this activity is relatively short, which makes sense. Panalpina also indicated that an aircraft is almost always immediately offloaded when it arrives. All devices that registered more than 12 sensor measurements are removed from the dataset. A total of 541 devices are removed from the dataset.

The last milestone of a shipment from LUX to HSV is m9 storage HSV. This activity has a relatively long duration of about 2,5 days. The median of all devices is 117 sensor measurements. Relatively many devices only registered 1 instance for milestone m9, 699 devices. Storage of only 15 minutes does not make sense. Therefore, these 699 devices will not be included in the final dataset. A maximum of 25185 sensor measurements were registered by a device, indicating a storage duration of about 9 months. Such devices are considered outliers. All devices with more than 800 measurements are removed as well, a total of 170 of the 4722 devices.

5.2.3 Data removal of measurements with too short time intervals

Sensor devices normally register temperature measurements every 15 minutes. Sometimes this interval differs several seconds, but not more. However, some devices registered measurements with much shorter intervals. The shortest interval is only 15 seconds. A total of 6541 sensor measurements were found to have an interval shorter than 14 minutes and are removed from the dataset.

5.2.4 Visualization with improved dataset

After the data preparation, a total number of 1503424 sensor measurements remain in the dataset. To see how the data now behaves, the new dataset is used as input for visualization. New graphs for the average temperature profile and the percentage of excursions against the percentage of the milestone that is finished, are created for both the warm and cold periods. These, however, are almost identical to figures 5.10, 5.11, and 5.12, and are therefore moved to appendix 4.

5.2.5 Milestone specific data reduction

The excursion graph gives an indication where during a milestone most excursions occur. The goal of making the excursion graph is to find irrelevant parts of the data such that these can be deleted. For instance, when milestone m9 storage HSV has an average duration of 2,5 days, and only at the early start and end of the milestone excursions occur, then all the other data in between start and end can be deleted from the dataset such that the prediction model can focus on predicting the start and end of the milestone. Keeping all the values will otherwise influence the model precision too much.

However, the excursion graph does not provide enough information to determine which parts of the data for each milestone can be deleted. In the graph, only one line for each milestone is plotted, but a milestone can have very different durations. Especially the milestones related to cold storage (m4, m5, m9) have a varying duration. When both devices with few and many measurements are averaged and plotted, this graph may look different from graphs made of devices with a number of measurements within a certain range. The excursion graph of may therefore not be accurate enough for certain milestones.

In order to more accurately make decisions on which parts of the dataset are useful, multiple excursion graphs are created for every milestone, each for a selection of devices with a certain number of sensor measurements. It is quickly found that these separate graphs are almost exactly the same for milestones that are relatively short and have a relatively constant number of measurements. These milestones are truck offloading, cargo uplift, flight, and A/C offloading. The graphs for these milestones are therefore not shown and are sufficiently represented by the lines in the original excursion graph. All data belonging to these milestones is kept since the slope of the curves is relatively constant indicating that excursions occur at a constant rate.

For the milestones related to cold storage, which are m4 storage LUX, m5 ULD b/u and storage, and m9 storage HSV, the graphs are more interesting since these show a different excursion pattern for selections of devices with different numbers of measurements. Based on these graphs, the important and unimportant parts of the data can be identified. Appendix 5 describes the process of selection in more detail. Specific parts of the dataset can be removed. The removal will be performed after the data preparations in chapter 5.3 are finished, since all instances are required to produce some of the additional variables that will be created there.

5.3 Data preparation before modeling

Before the actual modeling can start, several more data preparation tasks are performed. These include the conversion and removal of variables, but also adding additional variables. The current dataset contains 1503424 instances.

5.3.1 Current variables

Currently, the following variables are present in the dataset: timestamp in UTC format, timestamp indicating the local time, measured temperature by the sensor device, sensor device identification number, shipment number, milestone, user, customer, number of excursions as registered by SmartView, month, year, weekday, season, location of the sensor device on the cargo, minimum and maximum temperatures of the product range, the deviation of the measured temperature from the product range, airport destination, weather temperature.

5.3.2 Added variables

The prediction model will be able to make more accurate predictions when the dataset contains relevant variables based on which the model can train itself to find patterns in the data. The dataset is a time series dataset. Because we want to make a prediction of the temperature profile of a shipment for the next day, we need to add variables based on which the model can learn itself what will happen for a specific day, using information from previous shipments. The variables mentioned below are specifically created to increase the time series behavior of the dataset and the predictive power of prediction models.

Hour of the day

The hour of the day is added as a variable. The variable might be of value since it indicates the part of the day during which a milestone occurs. For instance, in the morning and evening the weather temperature is generally lower than during the afternoon. Additional variables can also be created with help of the hour variable. The hour of the day is based on the local time and not on the UTC time. It is found that the existing local timestamp is always equal to the UTC timestamp. Therefore, the local timestamp is created by considering the time zones. Summer (UTC+2) and winter time (UTC+1) in Luxembourg are taken into account.

Dayparts

Based on the hour of the day, the daypart of the day can be determined and is also added as a variable. Similar to the hour of the day, adding the daypart may be a predictor since more excursions can take place during the afternoon than during night due to temperature differences. Night is between 00:00 and 05:59 hours, morning is between 06:00 and 11:59 hours, afternoon is between 12:00 and 17:59 hours, and evening is between 18:00 and 23:59 hours. Another variable is added which splits a day in 8 dayparts. These are the same as the four dayparts, but now with prefix ‘early’ or ‘late’, indicating for instance ‘early morning’.

Ordering variables

The average temperature profiles for each milestone indicated a specific trend in the measured temperature during the activity of a milestone. Apparently, it seems to make a difference whether the instance is measured at the beginning, halfway, or at the end of the milestone. Therefore, a variable is added that indicates the i-th order of the instance of a milestone of a shipment. Another variable indicates the total number of instances of each milestone of a shipment. The fraction of the milestone that is finished is also added as a variable. This is done by dividing the order by the total number of instances for each milestone, and then adding this value as a fracture and as a percentage as variables. The percentages are rounded to the closest 5%. By adding these variables, the model can learn how far the activity of the milestone has completed.

Monthly average, minimum and maximum sensor and weather temperatures

The average, minimum, and maximum monthly measured sensor temperatures and weather temperatures are calculated for each month using the data of all years 2014-2018. The values are added to the dataset as six separate variables. The prediction model can use the mean as a starting point, and the minimum and maximum as boundary levels for making predictions.

Day of the month

A variable day of the month is added to the dataset, ranging from 1 to 31. This variable is created to order all shipments in the dataset based on the year, month, and day of the first instance of a shipment and device combination. Hence, the dataset begins with shipments of the year 2014, and ends with shipments of the year 2018. Ordering the dataset is needed to add other variables mentioned next.

Measured sensor temperature of 1 and 2 days before

For every instance in the dataset, a sensor measurement is sought that took place 24 hours ago and has the same milestone. If the measurement does not have the same milestone, the measurement is worthless. A tolerance of 1 hour is applied, which means that the value measured closest to 24 hours previously is sought in a range from 23 to 25 hours before the measurement took place. Another variable containing the measured temperature of 48 hours before the instance is also added but with a tolerance of 2 hours. The variables are called 'temp T – 1' and 'temp T – 2'. Variable T indicates discrete steps in time where each step is a day. Hence, the minus in 'temp T – 1' indicates the number of days before a specific day.

The measured sensor temperatures on T – 1 and T – 2 are expected to be good predictors for the measured temperature on T = 0 (target). This is expected since the weather temperature, the main cause of temperature excursions, generally remains relatively constant over several days. Therefore, the same weather influence on the cargo at T = 0 is expected as on T – 1 and T – 2.

Weather temperature of 1 and 2 days before

Using the same argumentation, the weather temperature of T – 1 and T – 2 are added to the dataset. The dataset containing all the weather data for both LUX and HSV consists of hourly temperatures. The weather temperature was already added to every instance of the dataset. Now the temperature of a day before (T – 1) and the temperature of two days before (T – 2) are added to the dataset. For every instance in the dataset, the weather temperature closest to 24 hours before the instance's timestamp is added to the dataset. Another variable containing the weather temperature of 48 hours before the instance is added to the dataset. Instances that have a milestone related to cold storage receive the value 100, which was previously done for the weather temperature (T = 0). The variables are called 'weather T – 1' and 'weather T – 2'.

Average, minimum and maximum measured temperature of the same closest milestone

As previously mentioned, the dataset was ordered for every combination of shipment number and sensor device. For every instance, the average, minimum and maximum measured temperature during the same milestone, of the closest previous device that belongs to a different shipment number, is added to the dataset. Due to the amount of data, the closest previous device is in many cases a device belonging to the shipment on T – 1. If this shipment does not contain the specific milestone, then the shipment on T – 2 is checked. For instance, there are 2070 shipment numbers that are ordered in time. Device A of shipment number 5 measured 10 instances of milestone 'Flight'. The closest previous (in order and thus in time) sensor device of shipment 4, device B, is checked. If this device also has the milestone 'Flight', then the average, minimum, and maximum measured temperatures of milestone 'Flight' of device B are determined. These values are then added to the 10 instances with milestone 'Flight' of device A. In this example shipment number 4 has two sensor devices. If device B does not contain milestone 'Flight', then the next closest device C that also belongs to shipment 4 is checked for

milestone ‘Flight’. If milestone ‘Flight’ is also not present in device C, the next closest device of shipment number 3 is checked, and so on.

Average, minimum and maximum weather temperature of the same closest milestone

The same procedure as for the average, minimum and maximum measured temperature is executed to obtain the average, minimum and maximum weather temperature of the same closest milestone.

Previous and next milestone

Every instance in the dataset is linked to a milestone. Since the logistic process is sequential, every milestone is always followed and preceded by the same milestone. The previous and next milestone of an instance are added to the dataset. For the first milestone m1 truck offloading, ‘none_earlier’ is added, and for the last milestone m9 storage HSV, ‘none_later’ is added to the dataset.

These variables were used in the initial models as described in chapter 6.4 but were found to have no predictive value to the models. This makes sense since the process is sequential and therefore a milestone is always preceded and followed by the same milestone.

Location

A variable indicating the city, either LUX or HSV, is added to the dataset based on where the milestone takes place. For all instances with milestone ‘Flight’, location ‘Air’ is added. Although the variable was used in the prediction models, the predictive value of this variable is questionable. The information of the location of the milestone is already implemented in the milestone since all milestones only occur in one location. The locations were added to the set of milestones in chapter 3.3.

5.3.3 Converted variables and removed data

The weather temperature during the cold storage related milestones m4, m5, and m9, was changed from the outside weather temperature to a constant of 100 degrees Celsius as described in chapter 4.5. The machine learning algorithm will recognize the value of 100 degrees as a dummy value.

Furthermore, all values of the variable with the measured temperature that are not a number, are deleted from the dataset. In order to correctly link the milestones to the sensor data, missing instances were created which received a temperature value of ‘xxx’, as described in chapter 4.2.3. These values are now removed from the dataset, which reduces the dataset with 128472 instances to 1374952 instances.

Moreover, the variable that indicates the location of the sensor on the cargo is changed. This variable previously had the value ‘inside’ or ‘outside’ of the cargo. The variable name is changed to ‘outside’ and the value that the variable has is 1 (yes) or 0 (no).

Now that all relevant variables are created, the dataset can be reduced by keeping only the relevant parts of the data of the cold storage milestones, as described in chapter 5.2.5. This reduces the dataset from 1374952 to 403485 instances.

The following variables are removed from the dataset since they were only used to produce other variables, or are found to have no predictive value for the model: sensor device, shipment number, user, customer, excursions, timestamp of local time, outside, minimum allowed temperature of the product range, maximum allowed temperature of the product range, deviation of the measured temperature from the product range. The local timestamp is removed since this value is always equal

to the UTC timestamp as mentioned earlier. The variable 'outside', which indicates whether or not (1/0) the sensor device is located on the outside of the cargo, is removed from the dataset since all remaining instances have value 1 meaning that all sensor devices are located on the outside of the cargo. The minimum and maximum product range temperatures and the deviation from the product range were added for the initial visualization and are not of value for the prediction model. The sensor device and the shipment number were used to structure the data and to create new variables, but have no value for the model. The number of excursions per shipment, a variable that originates from SmartView, contains wrong values since the number of excursions is incorrect. The user that entered the milestone and the customer of the cargo are found to be unimportant features.

5.3.4 Working dataset

The dataset that is used as input for the modeling phase contains 403485 instances belonging to 2070 unique shipments. Each instance is a row in the dataset, containing a sensor measurement and values for the other variables located in the columns. A number of 17331 temperature excursions exist in the data. In table 5.2, a list of all variables and their descriptions can be found. To see how much data is still available in the dataset, a small visualization is made. Figure 5.13 contains a plot of the temperature values in which a clear seasonality is visible. Figure 5.14 shows the number of instances in the dataset for each month, ranging from January (1) to December (12). As depicted in figure 5.15, the shipments are spread over all months of the year, each month having enough shipments. The distribution of the milestones is depicted in figure 5.16. Most instances have for the feature milestone the value 'Flight'. This makes sense since this is a relatively long activity. The number of instances of milestone m4, m5, and m9, was greatly reduced as described chapter 5.3.3. Only the beginning and the end of each cold storage milestone was kept. Milestone m1, m6, and Aircraft offloading (HSV), have the smallest number of instances, but these are also the shortest activities. Therefore, it can be concluded that all milestones are represented in the dataset.

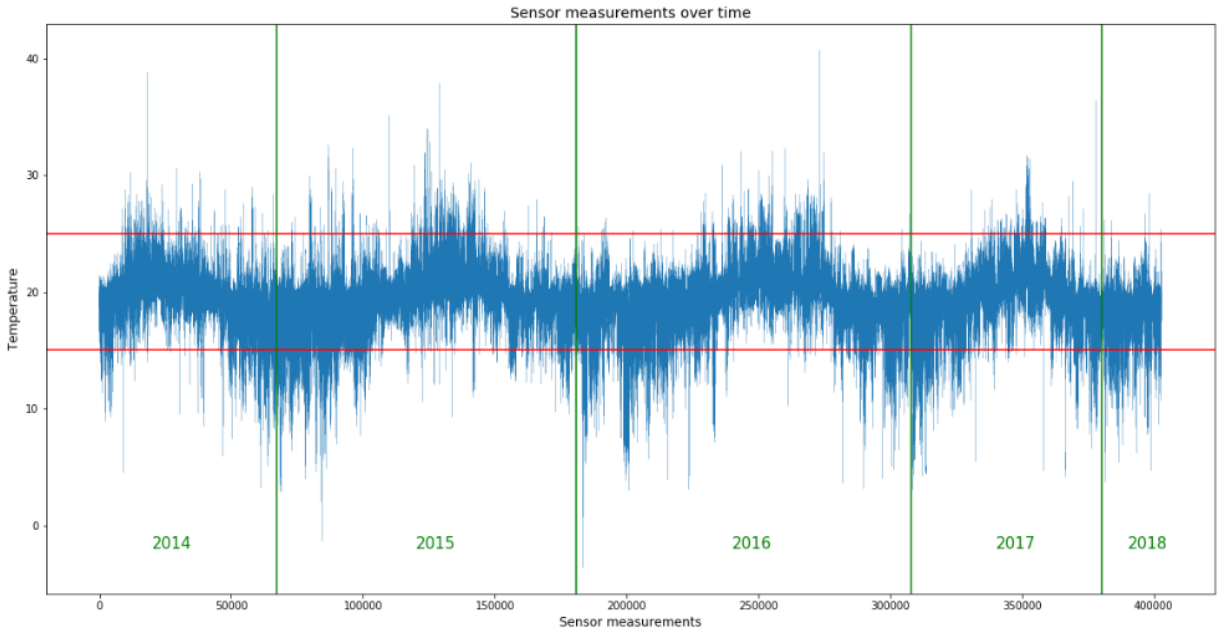


Figure 5.13: Sensor measurements over time

Variable	Description
temp (target variable)	Measured sensor temperature at T = 0
temp_t_1	Measured sensor temperature at T – 1
temp_t_2	Measured sensor temperature at T – 2
mean_w_month, max_w_month, min_w_month	Mean, maximum, minimum weather temperature during that month
mean_m_month, max_m_month, min_m_month	Mean, maximum, minimum measured sensor temperature during that month
m_mean_t_t_1, m_min_t_t_1, m_max_t_t_1	Mean, maximum, minimum measured sensor temperature during the same closest milestone
summer, fall, winter, spring	Seasons
month_x	Each month, with x a number from 1 to 12, 1 = January, 12 = December
weather	Measured weather temperature T = 0
weather_t_1	Measured weather temperature T – 1
weather_t_1	Measured weather temperature T – 2
m_mean_w_t_1, m_max_w_t_1, m_min_w_t_1	Mean, maximum, minimum weather temperature during the same closest milestone
mile_mx	Milestones m1, m4, m5, m6, m9
HSV, LUX, AIR	Location where the milestone takes place
order	The i-th order of an instance with a certain milestone belonging to a sensor device
max_order	The total number of instances with a certain milestone belonging to a sensor device
order_frac	order/max_order
order_perc	order_frac expressed as percentage, rounded to closest 5%
year_x	The year, with x: 2014, 2015, 2016, 2017, 2018
weekday_x	With x ranging from 0-6, 0 = Sunday, 6 = Saturday
(early_/late_)morning, evening, afternoon, night	Daypart, split in 4 parts, or dayparts split in 8 parts when early_ or late_ is added
day	Day of the month ranging from 1-31
prev_mx, next_mx, none_earlier, none_later	Previous and next milestone, with mx the abbreviation or name of the milestone
hour_x	Hour of the day with x ranging from 0 to 23

Table 5.2: All variables and their descriptions

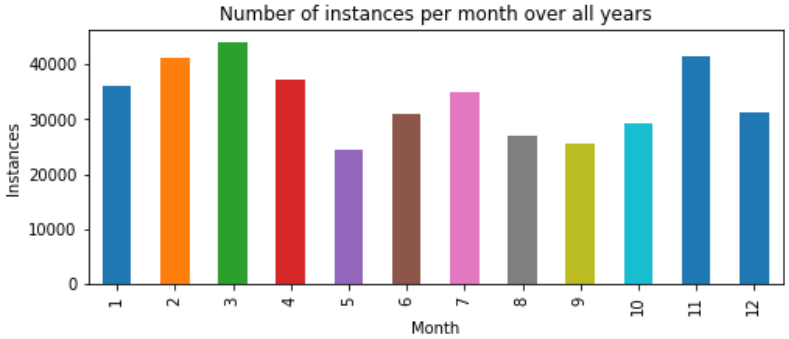


Figure 5.14: Instances per month

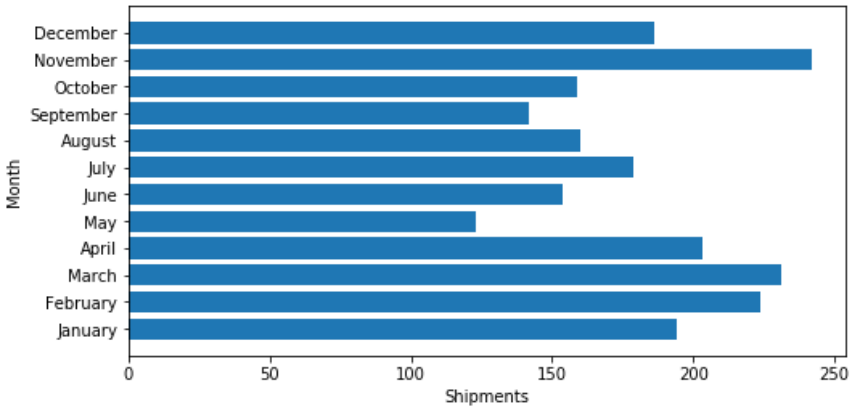


Figure 5.15: Shipments per month

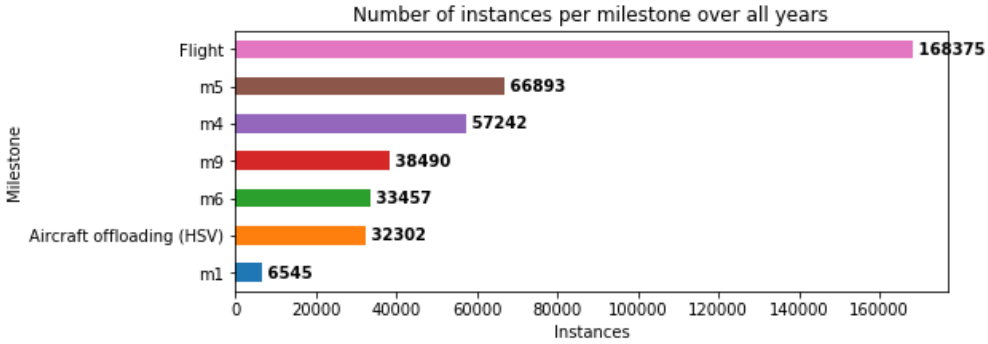


Figure 5.16: Instances per milestone

6 Modeling

This chapter describes which steps are taken during the modeling phase. The steps include some data preparation in order to apply the machine learning models on the dataset, application of the models on the dataset, feature selection, hyper-parameter optimization, and model selection. Figure 6.1 depicts these steps in the experimental setup that is used in this study. The steps are further explained in this chapter. The models are evaluated in chapter 7 'Evaluation'. Python packages scikit-learn, mlxtend, xgboost, and vecstack are used in the modeling phase. Machine learning algorithms used are random forest, lasso, gradient boosting, adaptive boosting, extreme gradient boosting, and stacked generalization.

6.1 Data preparation

Before the dataset can be used as input for a prediction model, the data requires a few more data preparation steps. These consist of dealing with missing values, normalization, conversion of categorical variables to dummy variables, and splitting the dataset in a training, validation, and test set.

6.1.1 Missing values

Since most machine learning models cannot deal with missing values, the dataset is checked for missing values indicated by the value 'NaN' which means 'Not a Number'. Adding additional variables as described in chapter 6.2 resulted in missing values for several of the variables. The variables with the sensor temperature of T – 1 and T – 2 contain respectively 295908 and 308512 missing values, a percentage of almost 75% of the values. When all instances with NaN-values are removed from the

dataset, only 50217 instances remain. This number is too small to create a prediction model with, especially since 31606 of the instances belong to only milestone m9 storage HSV and three other milestones have less than 60 instances. According to Han et al. (2012), several methods exist to deal with missing values. One of the methods is using the variable's mean of all samples that belong to the same class. In the case of the missing temperature values of T – 1 and T – 2, the similar classes milestone, month, and daypart are used. The NaN-values are replaced by the average temperature measured during that milestone during that month during that daypart. As an example, a temperature measurement 24 hours before a certain instance in the early morning in June during milestone 'Flight' does not exist in the dataset, and is replaced by the average measured temperature during a 'Flight' in the early morning in June.

Another method mentioned by Han et al. (2012) is ignoring the instances with missing values. The mean, maximum and minimum sensor and weather temperatures of T – 1 all have 181 missing values. These 181 instances are deleted from the dataset which does not affect the dataset due to its size.

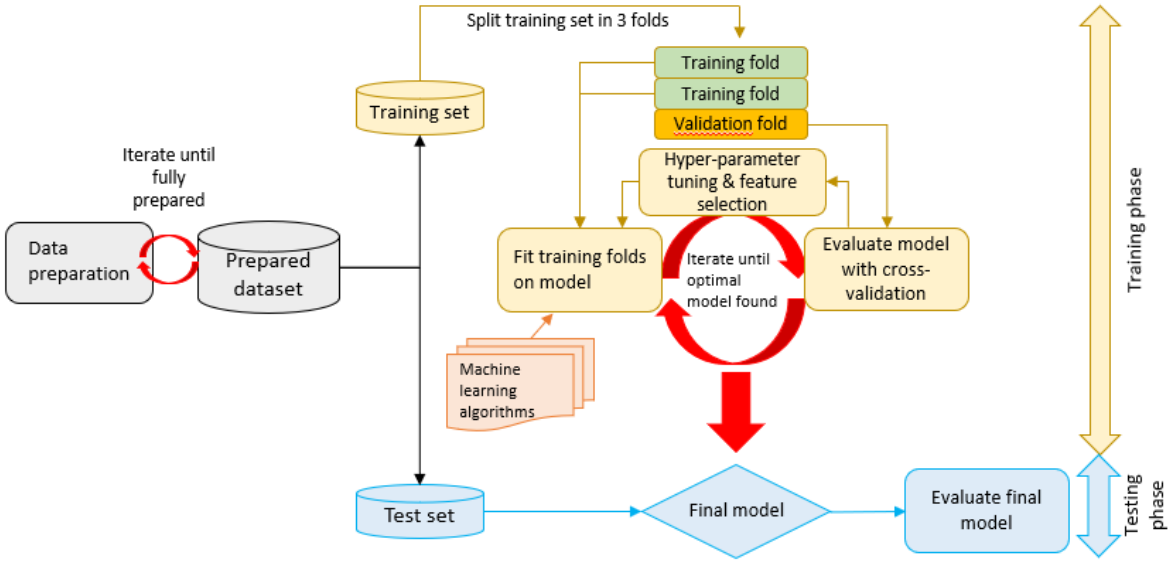


Figure 6.1: Experimental setup

6.1.2 Converting categorical variables

Not all variables are numerical but categorical instead. Before the data can be used as input for the prediction model, the categorical variables must be converted to dummy variables. All possible values of the categorical variables are changed to separate dummy variables that have the value 1 or 0. Value 1 indicates that the instance belongs to this category and 0 means it does not. The original variables are removed from the dataset.

6.1.3 Test shipments

Four test shipments are extracted from the dataset, two winter (January) and two summer (July) shipments. The test shipments have instances with all milestones and therefore describe the complete logistic process. This reduces the number of instances in the dataset with 387 to 402917. The test shipments are mainly used to visualize what a model can predict, and not specifically to test model performance.

6.1.4 Splitting the dataset

The data is split into features and targets. The features are all variables except for the measured sensor temperatures of T = 0, the targets. The model uses the features to predict the target values. The features and targets are further split into a training, validation, and test set. In the training phase, the model trains itself with the training set with the known training targets. The model therefore knows how close its predictions are and can learn the relationship between the features and targets. Once the model has trained itself, the training performance is tested with the validation set. This is done by cross-validation. Once cross-validation indicates that the model performs well during training, the model is tested on the test set. During the testing phase the model cannot 'see' the answers and will provide a prediction. Since we do have the actual target values, we can compare the predicted values with the actual values to test the predictive accuracy of the model. During the modeling phase, two data splits were used:

- Split 1 (random split): The dataset is randomly split into a training and test set, with respectively 75% and 25% of the data.
- Split 2 (year split): The training set consists of data from March 2014 to February 2017 (325137 instances, 80.7%), the test set of data from March 2017 to March 2018 (77780 instances, 19.3%). Since March 2018 only contains data until March 19th, March 2017 is included in the test set.

6.2 Feature selection

The initial feature set consists of 102 variables including dummy variables. After creating the correlation table and several initial models, the 24 hour variables and the previous and next milestone variables are removed. An overview of the 74 variables with their description can be found in table 5.2. During the modeling phase feature selection is performed. Feature selection is a procedure where the optimal set of predictive variables is chosen (Tang, Alelyani, & Liu, 2014). Only the features that contribute to the prediction of the target variable are used in the model. Feature selection helps to (Tang et al., 2014):

- Reduce overfitting: less 'wrong' decisions can be made based on irrelevant noisy variables when these are removed;
- Improve accuracy: higher predictive performance since only important variables are used to make the predictions;
- Reduced model training time: faster model training since the model has to consider less variables during training;
- Increased model interpretability: with fewer features, the model is less complex and thus better understandable.

6.2.1 Correlation

In order to see what variables might be a good predictor for the target variable, a correlation matrix is created. The pairwise correlation is performed with Pearson standard correlation method, which is a popular method when working with numerical data (Han et al., 2012). Correlation has a value between -1 and 1, and indicates the increase in a variable due to an increase in another variable by 1. A correlation of 0 means no correlation, and a correlation of -1 or 1 indicates full correlation. The most important relation is the correlation of the independent variables with the dependent variable that we want to predict: the measured ambient temperature 'temp'. Table 6.1 shows the 5 variables that correlate strongest with the target variable. Appendix 6 provides all correlations of the independent

variables with the dependent variable. The hour variables are excluded from the table, but all have values ranging from -0.03 to +0.02. The previous and next milestone variables are also excluded from the table. Their values are identical to the

It is interesting to see that the milestones do not seem to have much influence on the measured sensor temperature. Only the cargo uplift (-0.15) and storage HSV (0.116) show correlation with the measured ambient temperature. The measured temperature one and two days before (0.371 and 0.382) indicate the highest correlation. However, the mean measured temperature during a month correlates almost as high (0.345). This could indicate that the measured temperature is always very constant.

Abbreviation	*Variable description*	*Correlation*
temp_t_2	Measured temperature t – 2	0.38
temp_t_1	Measured temperature t – 1	0.37
mean_w_month	Mean weather temperature during that month	0.35
max_w_month	Maximum weather temperature during that month	0.35
mean_m_month	Mean sensor temperature during that month	0.35

Table 6.1: Top 5 most correlating variables

6.2.2 Feature selection methods

Recursive feature elimination and sequential forward feature selection

Two main feature selection methods are used in this study: recursive feature elimination (RFE) and sequential forward feature selection (SFS), both examples of wrapper models (Tang et al., 2014). Wrapper models analyze the quality of a selection of features based on the predictive accuracy of the model, which is a relatively computationally heavy process (Tang et al., 2014). RFE starts by including all variables, and removes (eliminates) variables from the dataset if removing results in better model performance. The method uses the feature weight coefficients (in case of Lasso) or feature importances (random forest and the boosting methods) to determine which variables to eliminate, and is computationally less requiring than SFS. SFS works the other way around and sequentially adds features based on a chosen performance metric to evaluate the intermediate models with. In this study the mean squared error (MSE) is used. RFE is performed with the in-built scikit-learn RFECV-function. The package mlxtend is used for SFS. Both methods use 3-fold validation.

Feature importances

Although not a feature selection method, relative feature importances can be used to determine useful variables for the machine learning models. A decision tree chooses appropriate split points in the tree. The relative rank or depth of a feature at the split points in a tree can be used to evaluate the relative feature importance for predicting the target values. The higher in the tree a feature is used on a split point, i.e. the closer to the root node, the more samples are affected by it. Therefore, the larger the fraction of samples affected by the feature, the higher the relative importance of the feature. These are expressed as features importances or regression coefficients in case of Lasso. The values indicate how much a variable improves the predictions compared to the other features. The relative feature importances can be extracted with Scikit-learn.

6.3 Model considerations

6.3.1 Supervised regression

The goal of creating the prediction model is to predict the temperature profile of a shipment on the next day. To create the temperature profile of a whole shipment, numerous temperatures need to be predicted belonging to timestamps with different milestones. In order to learn the model what temperature values to predict, the model is trained with historical data with known features (predictive variables) and targets (sensor temperatures). Because of the known input and output values, the problem is a supervised machine learning problem (Flach, 2012). Since we want to predict actual continuous values and we do not want to classify instances into a certain discrete category, we are working with a regression model (Flach, 2012). The regression model will learn the behavior of the data´s features and targets, and will so be able to make a prediction.

6.3.2 Tested machine learning models

The literature search provided only one real application of machine learning on sensor data. Aung and Chang (2014) created a K-means clustering model based sensor data to predict optimal refrigeration temperatures for multi-commodity storage in the food supply chain. This study, however, is an example of unsupervised learning and therefore not useful for our prediction model. Ahmed et al. (2010) tested several regression models on the monthly M3 time series competition data to find the optimal regression algorithms. As explained in chapter 2.4, the dataset used in this study is not comparable to a 'standard' time series dataset.

The preliminary non-exhaustive literature study did not guide us to specific machine learning models to use in this study. Therefore, we selected and tested several models. Lasso regression is a method that performs linear regression and feature selection at the same time and is therefore especially useful to reduce the number of features (Tang et al., 2014). Therefore, Lasso regression will be used in this study. Lasso belongs to the group of embedded feature selection methods (Tang et al., 2014), learning algorithms with embedded feature selection.

Ensemble models, such as averaging and boosting methods, belong to the most effective machine learning models and often surpass other techniques (Flach, 2012). For that reason, ensemble techniques are tried in this study. Ensemble techniques are often tree based models. A single classification and regression tree (CART) model usually does not have much predictive power and they suffer from high variance compared to for instance linear regression models. Combining multiple trees can result in much better predictions. Ensemble methods combine a number of base estimators built with a given learning algorithm (such as a regression tree) in order to create a single improved estimator (Han et al., 2012). Two main methods can be distinguished: averaging methods and boosting methods.

In averaging methods a number of base estimators (weak learners) is independently created with random samples of the data, whereafter the results are averaged. Weak learners are learning algorithms that make only slightly better predictions than random guessing such as small decision trees. Averaging performs better compared to a single estimator since its variance is reduced by averaging a larger number of estimators. When combining the weak learners into a single strong learner, all weak learners have equal weight. A random forest is an example of the averaging method. Bagging, for instance in random forests, is mainly a technique that reduces variance, while boosting methods mainly reduce bias (Flach, 2012).

According to Chen and Guestrin (2016), tree boosting methods have proven to be effective and are widely used. In boosting methods, base estimators are trained sequentially on the whole dataset

and at each iteration an estimator tries to reduce the bias of the combined estimator. Several weak models are combined (boosting) to create a powerful prediction model. When combining the weak learners into a single strong learner, all weak learners have different weights. The weighting depends on the boosting method that is used. Adaptive Boosting (AdaBoost), Gradient Boosting, and Extreme Gradient Boosting (XGBoost) belong to this class of ensemble methods.

6.3.3 Performance evaluation

Once a prediction model is created, the model's performance needs to be evaluated. This needs to be done to see how well a model predicts the target values, and to select the best prediction model when multiple models are created. The best method for model evaluation is to randomly split the dataset in a training, validation, and test set (Hastie, Tibshirani, & Friedman, 2017). The training set is used to fit the model, the validation set is used to determine the prediction error, and the test set is used to assess the generalization error of the final model.

Performance is expressed in performance metrics. Different performance metrics are used for regression and classification models, and some can be used for both. Evaluation metrics that can be used for regression models are mean absolute error and mean squared error (Hastie et al., 2017), accuracy, and coefficient of determination (Van der Plas, 2017). Besides accuracy metrics, prediction models can also be evaluated based on the following aspects (Han et al., 2012):

- Speed: the computational costs of the model required to make predictions
- Robustness: the ability of the model to deal with noisy and missing data
- Scalability: the ability to efficiently use the model on large amounts of data
- Interpretability: at which degree the model can be understood, specifically its functioning

Performance statistics

The different prediction models are evaluated with the help of performance statistics. These are commonly used statistics that can be calculated to indicate how well a model fits the data and how well it performs compared to other models. Below the performance statistics are described and why they are useful.

- Mean absolute error (MAE): For regressions the mean absolute error (MAE) is useful to see how large the absolute difference between the actual and the predicted values is. The MAE is very intuitive.
- Mean squared error (MSE): In order to determine how similar, on average, the predicted and target values are, MSE and root mean squared error (RMSE) can be calculated. The MSE is found by calculating the mean of the squared vertical distances between the actual and the predicted values as if they were both plotted on the y-axis. Squaring the values is done in order to include negative values. The RMSE is simply the root of the MSE and thus indicates the average difference between the predicted and actual values measured along a vertical line. The RMSE is different from the MAE since the mean is based on squared values instead of the absolute difference. Therefore, when the RMSE is higher than the MAE, this might indicate that a small number of predicted values deviate relatively much from the actual values since the errors are squared. The MSE is therefore susceptible to outliers (Flach, 2012).
- Coefficient of determination (R^2): The R^2-value indicates how well a model fits the data (goodness of fit) compared to a simple mean of the target values (Van der Plas, 2017). The R^2-value is a number between 0 and 1 which indicates the proportion of variance in the dependent variable that is explained by the variance in the independent variables in the

model. The R^2 can however also be a negative value, indicating very poor model fit. The higher the R^2-value, the more variance is explained by the variables in the model, which is desirable.

6.3.4 Baseline prediction

Before we start applying prediction models, a baseline prediction is made. The baseline prediction provides some benchmarking to see how much better the prediction model behaves compared to some average estimation (Van der Plas, 2017). The prediction models should at least perform better than the baseline prediction. The mean measured temperature of the same closest milestone (variable 'm_mean_t_t_1') is chosen as baseline variable. This variable is chosen because it should provide a reasonable estimation of the target value because it uses an average of temperatures measured during closest previous milestone. It is a better option than using the temperature of T – 1 ('temp_t_1') since that value is not averaged, and a better option than the mean measured temperature during the month ('mean_m_month') because this variable does not consider the milestone.

During the modeling phase, two different model splits (chapter 6.1.4) were used thus we also have two different baselines. The first method uses random split and the second method a split based on the years. These methods have the baseline values as indicated in table 6.2. The baseline value is more accurate for split 2, which is based on the years. A possible explanation could be less extreme weather from March 2017 to March 2018 compared to the weather in March 2014 to March 2017, and thus less extreme values in the data. Split 1 shuffles all data and therefore gives the most truthful baseline.

Method/Metric	MAE	MSE	RMSE	R^2
Split 1	1.60	5.87	2.42	-0.28
Split 2	1.48	4.72	2.17	-0.05

Table 6.2: Baseline values

6.3.5 Parameter search with exhaustive grid search

In this study several machine learning models are used. Each of them has its own set of parameters (hyper-parameters) which need to be tuned to make the models perform well. Finding the optimal set of model parameters is often a case of trial and error since optimal values vary for each application of the algorithms. Exhaustive Grid Search can be used to find the optimal parameters. This method exhaustively creates models from a grid of parameter values and tests each combination with cross-validation, and is therefore a computationally demanding method. In Python the scikit-learn module 'GridSearchCV' is used. Every combination of parameters is evaluated by 3-fold cross-validation. Using 3 folds is enough to validate the model's performance, and using more fold only results in significantly longer computation times.

6.3.6 Bootstrap method

Random forests use bootstrapping, a random sampling technique with replacement (Han et al., 2012). Many training samples can be drawn from a training set by application of bootstrap sampling without having to collect additional data. These samples can be used as training sets in combination with ensemble methods, resulting in a reduction in the prediction variance and increase in predictive performance by averaging their predictions. This is called bootstrap aggregation or bagging (Breiman, 1996a). One advantage of bagging is that the model is not overfit by increasing the number of bootstrap samples, and thus also the number of estimators.

Not all boosting methods use bootstrap sampling. The models are created sequentially and the training set is continuously updated. New models are trained on the residuals of the data, the part of the data the model had bad performance, to slowly create a strong learner. Bootstrap sampling can be used in gradient boosting, and the method is then called stochastic gradient boosting (Friedman, 2002). XGBoost is uses bagging of the samples and features (Chen & Guestrin, 2016).

6.3.7 Overfitting

Overfitting means that a model learns the relations between the features and the targets too well. If the relations in the model are learned too well, the dataset may not be a good representation of the problem anymore and may eventually provide worse results. Underfit data has a high bias, overfit data has high variance (Van der Plas, 2017). Figure 6.2 shows an example of both underfitting and overfitting. In modeling, a right balance needs to be found between the amount of bias and variance, such that the model is flexible enough to represent the features but not too flexible that it also fits for random errors.

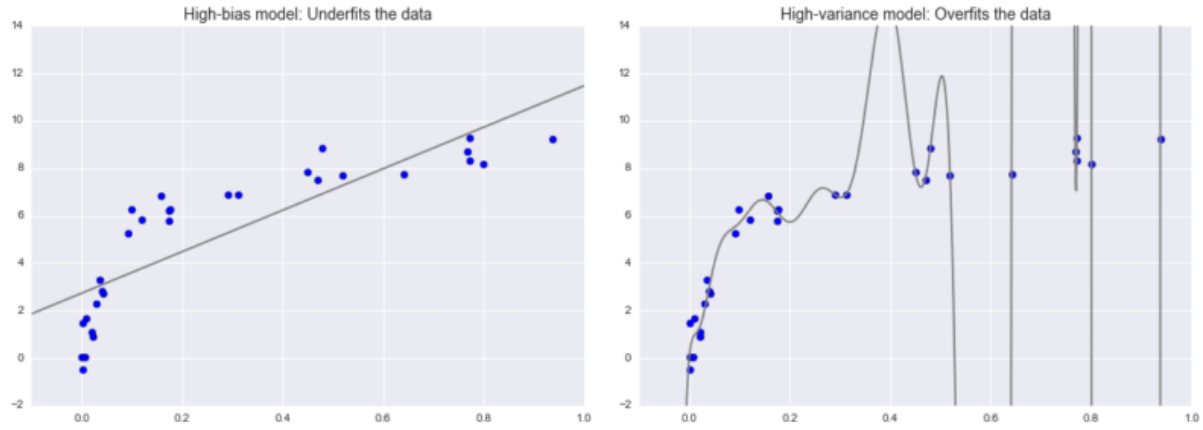


Figure 6.2: Under- and overfitting of data reprinted from Van der Plas (2017)

6.3.8 Machine learning models

Random Forest

The random forest machine learning method is an averaging ensemble method. As indicated, these methods construct more than one estimator. Usually decision trees are used as estimator. A random forest regression model consists of collections of regression trees. Each tree is trained with a random selection of the features and instances of the training data, which is why the model is called a random forest (Breiman, 2001). Besides bootstrap sampling, the random forest therefore also uses feature bagging (Chen & Guestrin, 2016). When all features are used, the random forest is equal to standard bagging. A random subset of features is selected at each split in each decision tree, which decreases correlation between the decision trees resulting in better predictions (Breiman, 2001). The separate trees are then merged into a forest which takes the averages of the predictions. This forest has a higher predictive power than the separate decision trees. The decision trees consist of a root node, branches, and leaves. The root node is the first step where a path through the tree is chosen. The branches are all follow-up conjunctions of the features that eventually lead to the leaves, which contain the regression value and the number of samples that lead to this value. In this study, the model is built with the Skikit-learn library's RandomForestRegressor class. The decision tree is the Skikit-learn

DecisionTreeRegressor. This tree is learned in all upcoming methods, and its parameters are kept at their defaults.

Tree visualization

An individual decision tree can easily be visualized and interpreted. This is also the case for averaging ensemble methods, since these methods create an 'average' decision tree based on the number of decision trees used to create the model. Therefore, it is possible to output the decision tree of the random forest. Boosting ensemble methods iterate and improve their estimates sequentially, and a final decision tree cannot be generated.

The decision trees from the random forest can be extracted with Python as an image. The complete tree is too large to easily output and interpret. Therefore, figure 6.3 shows a tree which is limited to only 3 levels and functions just to show how the random forest looks like. The root node is shown on top of the tree, and the leaves are at the bottom. The root node has the most decisive influence on the prediction. In this case, if the measured sensor temperature of two days earlier is higher than 19.5 °C, we move to the right half of the tree. The leaves indicate the predicted values and on how many samples they are based. The MSE is also shown, indicating the prediction error of the samples at a certain point in the tree.

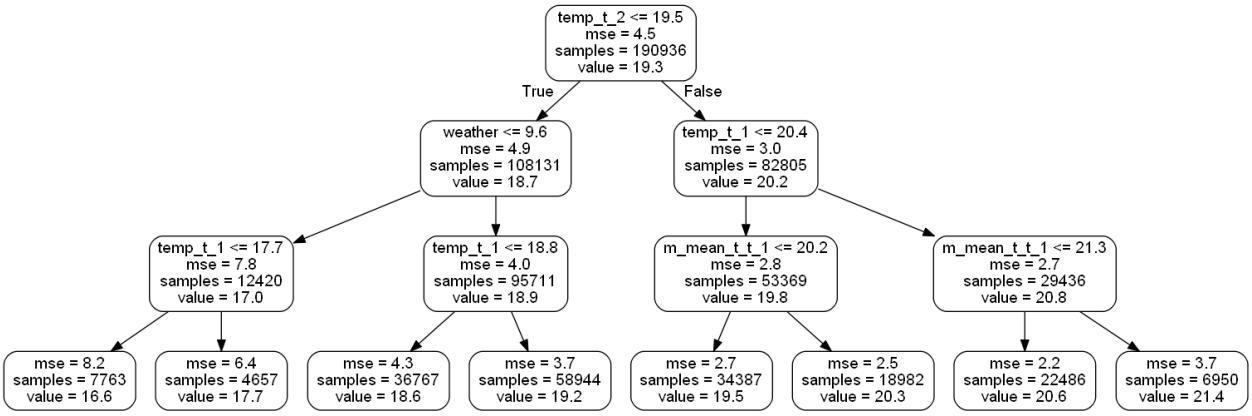


Figure 6.3: Random forest averaged decision tree with three levels

Adaptive Boosting

Boosting means that many weak (simple) learners are combined in order to make accurate predictions (Freund & Schapire, 1999). One of the most popular boosting procedures is adaptive boosting (AdaBoost) which can be used for both classification and regression problems (Han et al., 2012). AdaBoost fits a number of weak learners on repeatedly adjusted versions of the data. The predictions from the weak learners are then combined in a weighted sum to generate the final prediction. The continuous adjustments of the data at each boosting iteration, which means the application of a new weak learner, includes applying weights to each training sample. In every iteration, the sample weights are adjusted and a new estimator applied to the reweighted data. The training instances that were incorrectly predicted by the boosted model in the previous step, receive increased weights, and the instances that were correctly predicted receive lowered weights (Han et al., 2012). Therefore, when the number of iterations increases the training instances that are hard to predict become more important which forces the weak learner to focus on these instances. The learning procedure stops earlier when perfect fit is found with less estimators than those given as input

parameter. In our regression problem, the Skikit-learn library's AdaBoostRegressor class is used. This class implements the AdaBoost regression algorithm which is known as AdaBoost.R2.

Gradient Tree Boosting

Gradient Tree Boosting (Gradient Boosted Regression Trees (GBRT)) is a boosting method that can be used for both regression and classification. The method has been found to be effective in many applications (Chen & Guestrin, 2016). In Gradient Boosting, new estimators are sequentially added, each estimator correcting the errors of the existing models (Friedman, 2002). Estimators are added until no further improvements can be made, or when a restriction is set to the number of estimators.

Gradient Boosting applies Gradient Descent, a method that iteratively minimizes a function to its local optimum (Han et al., 2012). In each iteration, the value (or set of values) is compared to the value of the previous iteration subtracted with a positive constant multiplied by the slope of the function (= gradient) at the previous value. Once no further improvements can be made, the minimum is found. In Gradient Boosting, gradient descent is applied to find the optimal regression tree from a set of trees. To do so, Gradient Boosting optimizes arbitrary differentiable loss functions. In every iteration of a GBRT a regression tree is fit on the negative gradient of the given loss function. A loss function is a function whose value increases by penalizing prediction errors (Hastie et al., 2017). A suitable loss function is the squared error between the predicted and the actual values. The goal is to find a local minimum of the loss function. In Gradient Boosting a loss function must be differentiable otherwise the gradient descent method cannot be applied. Skikit-learn library's GradientBoostingRegressor class is the model that is used in this study.

Extreme Gradient Boosting (XGBoost)

XGBoost is a scalable tree boosting method that has proven its performance in many machine learning challenges such as Kaggle data mining competitions (Chen & Guestrin, 2016). XGBoost is very similar to gradient boosting but functions slightly different. The method uses a regularized learning objective which avoids overfitting (Chen & Guestrin, 2016). XGBoost is similar to Gradient Boosting if the regularization parameter is set to zero. Furthermore, XGBoost uses two additional methods reduce the possibility of overfitting (Chen & Guestrin, 2016). Shrinkage scales the weights of the trees such that newly created trees have more influence on improving the model. Feature subsampling (feature bagging) further prevents overfitting by reducing the correlation between the trees, similar to a Random Forest. Feature bagging also increase computation speed. Computation speed is further increased by using an approximate split finding algorithm instead of a greedy algorithm (Chen & Guestrin, 2016). XGBoost can also handle missing values by choosing an optimal default direction in a tree node when values are missing. The technique is applicable for al sparsity patterns (Chen & Guestrin, 2016). XGBoost efficiently uses processors and memory to speed up computations. In this study, the package 'xgboost' is used.

Lasso Regression

Lasso regression is a type of linear regression model that uses L1 regularization (Tang et al., 2014). Lasso stands for Least Absolute Shrinkage and Selection Operator (Tibshirani, 1996). The method penalizes the absolute size of the regression coefficients. The larger the penalty, the further the estimates are reduced ('shrunk') to zero. This can lead to situations where parameter estimates are actually reduced to zero, and therefore, the model automatically performs feature selection (Hastie et al., 2017). While penalizing the magnitude of the coefficients of the features, the error between the

predicted and actual values is minimized (Tibshirani, 1996). The Lasso minimization objective is the least squares objective plus alpha times the sum of absolute value of coefficients. The Lasso method leads to simpler sparse models with fewer parameters. Correlated features can be problematic with Lasso, since Lasso often keeps only one of the correlated variables and reduces the other to zero. In this study the Scikit-learn library's LassoCV class is used. LassoCV automatically selects the optimal parameter value for alpha by 3-fold cross-validation.

Stacked generalization

Stacked Generalization (stacking or meta ensembling) is an ensemble method which minimizes the generalization error by inferring the biases of the generalizers on the training set (Wolpert, 1992). The method can be used to combine multiple prediction models to create a new model. In stacking, multiple levels of prediction models can be used, each using the predictions of the level below (Wolpert, 1992). In this study, stacking is only applied on one level. A stacked model often outperforms the lower-level models due to its smoothing capability and it can combine the strength of each individual model into a single model. The different models might be able to predict certain parts of the data relatively well, and others relatively poor. The stacked model discovers where each model performs well and uses this information in the final model. The predictions from multiple first-level models are blended together and used as features to train a second-level model. Any model can be used to combine the first-level models. Stacking is most useful when the underlying models use very different algorithms und thus make uncorrelated predictions (Breiman, 1996b). The models should excel on different parts of the data. For instance, stacking multiple boosting methods might not significantly increase the performance of the stacked model. Instead, using a linear regression model, a boosting method, and random forest, might increase the performance of the stacked model since they are all very different. Inaccurate but very different models can add substantial value to the stacked model.

6.3.9 Model parameters

The different models can be tuned by changing their parameter values. Below, all parameter meanings are discussed. Some parameters are abbreviated in future performance tables. Table 6.3 indicates which model uses which parameter.

- Number of estimators (nr_est): This is the number of estimators (weak learners). Increasing the number of estimators reduces the algorithm's speed, since more samples need to be considered.
- Maximum number of features: A restriction can be set to the number of features that is used in the model. Adjusting the number of features can affect performance and computation speed. By default, most models use all features.
- Maximum estimator depth (depth): The depth of each estimator, a regression tree, can be adjusted. The depth determines the number of decisions (nodes) in the tree. The default value is no maximum depth.
- Minimum leaf size: The minimum leaf size determines the minimum sample leaf size at the end node. When this value is too small, the model might include too much noise. This parameter is not tuned in this study.
- Random state: The random state is a value that can be set such that the output of the different runs of the models are the same and the same values are chosen as training and test values. The value of the random state does not matter and is set at 42.

- Number of jobs: For some method multiple jobs can be ran in parallel which increases computation speed. By default this value is set to 1. The value can be increased or set to -1, which sets the number of jobs equal to number of CPU cores.
- Number of folds: Some methods have an implemented cross-validation method. For these methods the number of folds can be chosen.
- Learning rate (lr): Also known as the shrinkage factor, determines the contribution of each model to the weights. Its default value is 1. The learning rate can be lowered which means that the weights are increased or decreased with a smaller number at each iteration. Lowering the learning rate results is slower learning but it can result in better model performance. A trade-off exists between the number of estimators and the learning rate. The right balance must be found by trial and error.
- Loss function: A loss function must be chosen for some variables. In AdaBoost and XGBoost, the loss function is kept at the default value 'linear'. In Gradient Boosting, the default loss function least squares is used. A possible downside of least squares is that this function is sensitive to outliers due to the fact that the errors are squared.
- Regularization strength: Size of the penalty expressed as parameter Alpha. Multiple alphas can be fed to the model and the optimal alpha is selected by cross-validation. The number of alphas can be chosen, as well as their values.
- Subsample: The fraction of samples to use for fitting the individual base learners can be adjusted with the parameter 'subsample'. The default value of 1.0 indicates that all samples are used. Lowering this value result in a different type of Gradient Boosting: Stochastic Gradient Boosting. Applying Stochastic Gradient Boosting results in a lower variance but increased bias. It functions as a combination of Gradient Boosting and bootstrap averaging (bagging). The subsample is drawn without replacement. Combining shrinkage and subsampling can increase model accuracy, and can therefore be tried. A typical value of the subsample is 0.5.
- Performance metric: Performance is calculated with a metric. In the case of stacking, MSE is used.

Model parameter	*Random Forest*	*Lasso*	*AdaBoost*	*Gradient Boost*	*XGBoost*	*Stacking*
Number of estimators	X		X	X	X	
Maximum number of features	X		X	X	X	
Maximum estimator depth	X		X	X	X	
Minimum leaf size	X					
Random state	X		X	X	X	X
Number of jobs	X	X			X	
Number of folds		X				X
Learning rate			X	X	X	
Loss function			X	X	X	
Regularization strength		X				
Performance metric						X

Table 6.3: Model parameters

6.3.10 Model advantages and disadvantages

A random forests uses bootstrapping on the samples and the features (Hastie et al., 2017). By feature bagging, random forests reduce correlation between decision trees which leads to better performance. Random forests therefore do not easily overfit by increasing the number of bootstrap

samples and estimators, but it is still possible. Boosting methods on the other hand are much more sensitive to overfitting (Han et al., 2012), except for XGBoost which also applies feature bagging. The random forest also has the highest interpretability since the averaged tree can be visualized. Furthermore, a random forest is a relatively quick machine learning method because the estimators can be created in parallel (Breiman, 2001). This is not possible for boosting methods since these create estimators sequentially. The only exception is XGBoost, than can be trained in parallel.

Random forests can be affected by the number of significant and insignificant variables. With few significant and many noisy variables the random forest may not perform well. This is caused by the fact that the random forest may produce uninteresting trees based on the insignificant variables, which negatively influence the already learned relationships in the data. However, we should not only include the significant variables since then all created trees might look the same. Including some less significant variables might help the model to learn new relationships. Including too many insignificant variables includes too much noise into the model's prediction power. Boosting procedures are, compared to random forests, not affected by a large number of insignificant variables. Boosting models only assign large weights to the significant variables at each iteration, and therefore the already learned relations are affected less by the insignificant variables.

Boosting procedures are easily affected by noise (Freund & Schapire, 1999). AdaBoost will continue to try to correct errors in the training data, while the training data itself might be faulty.

The main advantage of the Lasso regression method is that it automatically performs feature selection. Lasso is also very quick in its computations. The main disadvantage of Lasso regression is that when variables are correlated, the method only keeps one variable in the model and removes the other correlated variable. This can lead to loss of information and thus loss in model performance. Furthermore, the non-zero coefficients tend to bias towards zero and are usually not statistically consistent, meaning that the coefficients change towards their true values when the sample size increases (Hastie et al., 2017).

Gradient Boosting can easily handle data of different types, has a high predictive power, and they are robust to outliers in the output space due to robust loss functions. Another advantage of Gradient Boosting is that the method is not sensitive to overfitting by using too many estimators. The main disadvantage of Gradient Boosting is that the method is sequential of nature and therefore the boosting procedure cannot be parallelized. This means that it is relatively slow.

Compared to gradient boosting, XGBoost is much faster because it is more memory-efficient and can be parallelized. XGBoost can handle data with all sparsity patterns. Gradient boosting has a wider range of applications. XGBoost is the only method that can deal with missing values.

Stacked generalization can provide better prediction results than using a single model, however, the method is computationally demanding (Breiman, 1996b).

6.4 Model creation

In the modeling phase, two types of models are created based on how the data is split as described in chapter 6.1.4. First, models were created with split 1. After evaluating these models, split 2 was applied. Next, we will first describe the overall training and testing process, and then the two different models are evaluated.

6.4.1 Creating a predictive regression model

Building a regression model consists of several steps. First, the data is split into a training and test set. Second, the features and the targets are separated from each other. A subset of features can be

manually selected or with help of feature selection methods. Third, a machine learning model is selected and created. Model parameters are chosen. Fourth, the training set is fit on the model in order to train the model. The model learns the relations in the data by making predictions and evaluating these on the known training targets. K-fold cross-validation is applied, which randomly splits the training set in K mutually exclusive folds of equal size (Han et al., 2012). Training and validation is performed K times, and in each iteration, one part is used as validation set and the others are used for training. In each iteration, a different part is used for validation. By using cross-validation, the training performance can be validated to make sure it is correct. At last, the best performing model and respective features are selected and these can be evaluated, which is the next CRISP-DM phase.

6.4.2 Modeling with random split (split 1)

Initially, models were created by using a random train and test split. All input variables were used in this modeling phase to predict the target variable: the measured sensor temperature at T = 0 (variable 'temp'). All the variables are listed in table 5.2 on page 58. The random split means that the data ranging from March 2014 to March 2018 was randomly shuffled and divided into a train set and test set with respectively 75% and 25% of the data. With the training set, cross-validation was performed. Models were created without optimizing the model's parameters. Except for Lasso regression, the performance metrics indicate a good model fit, even without optimizing the model parameters. A reason for the bad performance of the Lasso method is given in the discussion. The validation R^2-values of all models are ranging around 0.60 to 0.67. The best performing model is the Random Forest. The models were applied on the test set after training. The performance on the test set is comparable to the validation performance. Table 6.4 shows all performance statistics of the models. The training and validation scores are the average scores of the 3-fold cross-validation. The test scores indicate the performance of the model on the test set. The predictions of the models are much better than the baseline prediction, which has a MAE of 1.60, MSE of 5.87, and RMSE of 2.42.

Model (parameters)	*MAE*	*MSE*	*RMSE*	*R^2*	*Set*
Random Forest	0.36	0.45	0.67	0.901	Train
(nr_est = 30)	0.70	1.49	1.22	0.672	Val
	0.67	1.42	1.19	0.690	Test
AdaBoost	0.27	0.40	0.63	0.913	Train
(nr_est = 50, lr = 1.0)	0.71	1.71	1.31	0.623	Val
	0.68	1.63	1.28	0.644	Test
Lasso Regression	1.27	3.52	1.88	0.226	Train
(alpha = 0.01)	1.27	3.52	1.88	0.226	Val
	1.25	3.51	1.87	0.231	Test
Gradient Boosting	0.86	1.56	1.25	0.657	Train
(lr = 0.1, nr_est = 1000, depth = 5)	0.91	1.82	1.35	0.601	Val
	0.90	1.764	1.328	0.614	Test
XGBoost	0.86	1.59	1.26	0.650	Train
(lr = 0.1, nr_est = 1000, depth = 5)	0.91	1.84	1.36	0.597	Val
	0.90	1.77	1.33	0.613	Test
Random Forest with feature 'combi'	0.17	0.118	0.343	0.974	Train
(nr_est = 30)	0.44	0.763	0.874	0.832	Val
	0.36	0.574	0.758	0.874	Test

Table 6.4: Performance initial models with random split

Test shipments

As described in chapter 6.1.3, four test shipments were created, of which the data is not included in the dataset. To evaluate how well the models perform on completely new data, the models are tested on these test shipments. The models perform very poor on the test shipments. Negative R^2 values for three of the shipments indicate that the models do not represent the data of these shipments.

The bad performance on the test shipments can probably be blamed to the random data split which means that data points of all shipments are in the training, validation, and test sets. In the training phase, the model trains itself with the training folds, and then validates its performance using the validation set. The training and validation sets contain data points belonging to the same shipments. Based on these data points and several features, the models are able to learn which data points belong to the same shipment. The trained model determines to which shipments the data points in the validation set belong, and because the model was trained with other points of the same shipments, the model can make fairly accurate predictions.

However, the data of the four test shipments is completely new to the model, i.e. none of the data points of the test shipments were seen in the training phase. Therefore, the model is not able to identify a similar shipment where the data points from the test shipments can be associated with. This could be the reason why the performance is so poor compared to the training shipments. Furthermore, the model makes better predictions for summer shipments than for winter shipments. Both the winter shipments originate from January 2018 and the two summer shipments originate from July 2016. The winter shipment could be outliers and their data may therefore significantly deviate from the training data. It may have been better to choose shipments with more time in between them and from different years. The data of the test shipments is discussed later in this report. The performance statistics of the random forest on the four test shipments using all variables are given in table 6.5. The corresponding prediction graphs with the actual (blue), predicted (red), and baseline values (baseline), can be found in figures 6.4-6.7 and, in larger format, in appendix 7.

Test shipment	*MAE*	*MSE*	*RMSE*	*R^2*
Winter shipment 1	1.41	3.29	1.81	-4.40
Winter shipment 2	1.96	6.44	2.54	-0.43
Summer shipment 1	0.59	0.66	0.81	0.48
Summer shipment 2	0.76	0.86	0.93	-0.65

Table 6.5: Performance initial Random Forest model on test shipments

Feature importances

Next, the relative feature importances of each model were determined and each model underlined the importance of variables such as the ordering variables, day, and year. These are the variables which help to identify which data points belong to the same shipment. Therefore, to further test the importance of learning a model to which shipments data points belong, an additional variable indicating the unique combination of shipment number and device number, named 'combi', was added to the dataset. Using this dataset as input for a Random Forest, this model's performance was even higher, as indicated in table 6.4. Hence, the model emphasized the importance of data points belonging to the same shipment even more. The new combi variable had a relative importance of 20%. The feature importances of each model larger than zero are plotted in appendix 8.

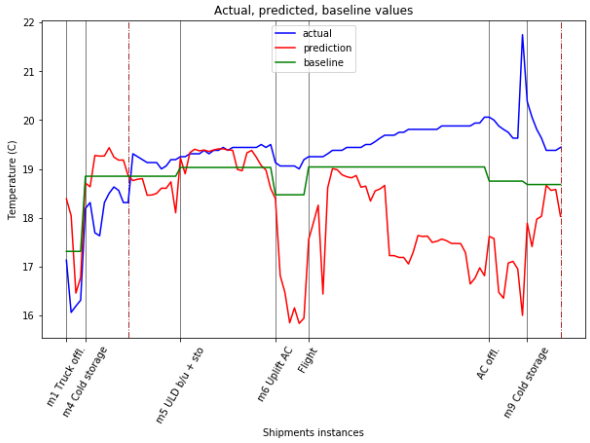


Figure 6.4: Winter shipment 1

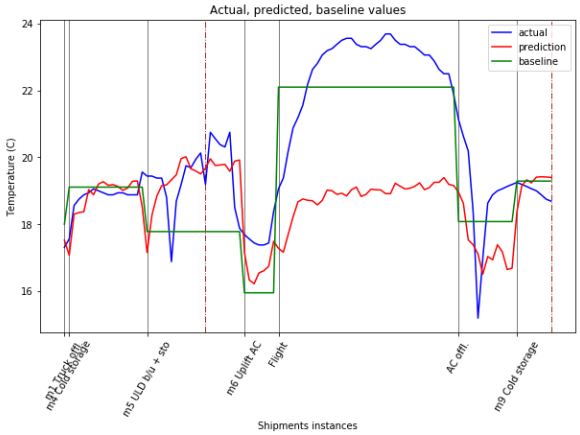


Figure 6.5: Winter shipment 2

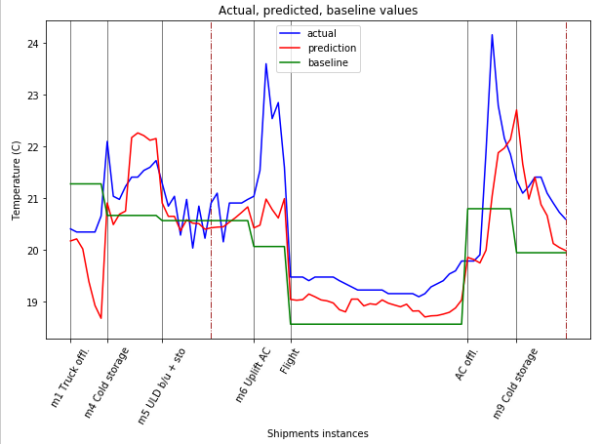


Figure 6.6: Summer shipment 1

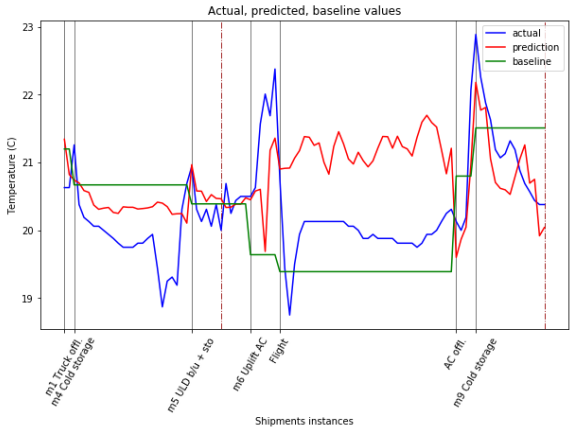


Figure 6.7: Summer shipment 2

6.4.3 Modeling with year split (split 2)

By randomly splitting the dataset, the models focused on learning to identify to which shipments data points belonged and made predictions based on that information. However, this information is not available when completely new shipments need to be predicted and the models should focus on learning other patterns. Therefore, a second method of splitting the data was introduced. The data is split based on the year, meaning that the training data consist of data belonging to March 2014 to February 2017, and the test set belongs to March 2017 to March 2018. The training data is not shuffled which means that each cross-validation fold consists of data belonging to different shipments. Using the year split, the model is trained, validated, and tested on data points belonging to different shipments.

Variable removal

Variable 'combi' was only added to confirm that randomly splitting the data was unsuccessful. The variable is useless with the year split and therefore removed from the dataset. Several other variables are removed from the complete set of variables previously used with the random split:

- Previous and next milestones: these variables have zero added value. This information is already in the data because milestones are always succeeded and preceded by the same milestone.

- Day of the month: the day of the month (1-31) cannot be of influence and can only result in models learning faulty connections from the data.
- Hours of the day: the hours of the day ranging from 0-23 are removed from the dataset. These were of help to create other variables such as the dayparts, but are found to be unimportant themselves.
- As described in chapter 5.3, the ordering variables consist of order, maximum order, order fraction, and order percentage. The order fraction is removed from the dataset because this value is also represented by the order percentage, the order fraction rounded to the nearest 5%.
- Years: the variables indicating the year are removed from the dataset since the train-test split is made based on the year.

Feature selection

Feature selection is performed on the 66 remaining variables. Since Lasso is such a fast machine learning model, RFE is used with Lasso to select the most important variables. With RFE, 53 important features are identified. However, a near optimum is reached with 12 features. Because the performance difference is so small, modeling with 12 features would be preferred, especially since it also reduces model complexity and computation time. Usually the smallest set of variables is chosen as long as the model's performance does not significantly decrease (Tang et al., 2014). The Scikit-learn method unfortunately only has the option to output the 53 features that resulted in the model with the lowest MSE. The intermediate models and their performances cannot be obtained from the method. Hence, of the near optimal model with 12 features, the corresponding MSE and feature used cannot be identified. A plot of the cross-validated MSE of the RFE process is shown in figure 6.8. All 53 with RFE identified features are indicated with a 1 in appendix 9.

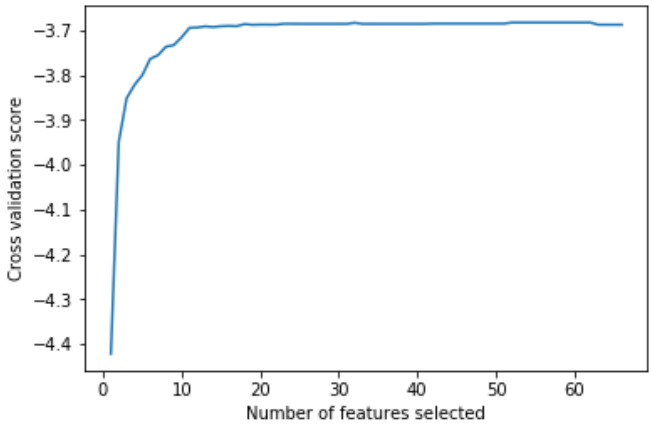


Figure 6.8: MSE of Lasso's RFE

Because the number of important features is probably lower than 53 and a near optimal solution with less features is preferred, another method is used to identify relevant variables. For the four methods Random Forest, AdaBoost, Gradient Boosting, and XGBoost, the relative feature importances are generated of all 66 features. These are also listed in appendix 9. The methods identify respectively 10, 20, 28 and 29 features as relevant (relative importance higher than 0). A set of 35 important features can be identified that cover the important features for each model.

Next, SFS is applied with the 35 features, a computationally demanding procedure. AdaBoost and Gradient Boosting have thus far not indicated to have any performance advantage over Random Forest, Lasso, or XGBoost. Therefore, these methods are discarded and we only continue with the other three. For Random Forest and XGBoost, a pipeline is used with gridsearch to test multiple parameter sets in combination with SFS. For Lasso this is not necessary since the method automatically selects its optimal alpha value by cross-validation. The pipeline is created with Scikit-learn class 'Pipeline'. Appendix 10 shows the graphs of the MSE against the number of features in the SFS process.

Performance with optimal feature set and parameters

Now that for each model the optimal feature set and parameters are determined, the performance can be evaluated. Table 6.6 identifies the optimal number of features for each model and the corresponding performance metrics. Again, training and validation scores from cross-validation as well as the scores on the test set are given. The optimal parameters of each model that were determined during training are provided. When looking at the validation and test scores, the models have very similar performance. The MAE, MSE, and R^2 are almost equal to each other. Therefore, none of the models performs significantly better than the others. The performance is better than the baseline prediction, which has a MAE of 1.48 and MSE of 2.17. The difference is however not very large. SFS with Random Forest resulted in a lower number of features compared to Lasso and XGBoost. The Random Forest also reaches its near optimal performance with less features; 8 for Random Forest compared to 12 and 15 for respectively Lasso and XGBoost.

The validation scores obtained during the training phase are lower for each model compared to the performance of the models on the test set. A possible explanation related to data quality, caused by the data splitting method, was already given in chapter 6.3.4 when the two different baselines were evaluated. This will be further discussed in the discussion in chapter 9. In appendix 9, the with SFS selected features for each model are highlighted with green.

Model (parameters)	*Nr features*	*MAE*	*MSE*	*RMSE*	*R^2*	*Set*
Random Forest (nr_est = 5, depth = 3)	15	1.21	3.38	1.84	0.260	Train
		1.26	3.66	1.91	0.200	Val
		1.24	3.24	1.80	0.278	Test
Lasso Regression (alpha = 0.01)	25	1.25	3.59	1.89	0.215	Train
		1.26	3.67	1.92	0.198	Val
		1.24	3.26	1.80	0.276	Test
XGBoost (lr = 0.1, nr_est = 50, depth = 3)	23	1.20	3.25	1.80	0.289	Train
		1.25	3.58	1.89	0.218	Val
		1.22	3.13	1.77	0.304	Test

Table 6.6: Optimal model parameters and number of features with performance metrics

Application of stacked generalization

When the Lasso, XGB, and Random Forest models were created, the models were tested on the four test shipments. The performance on the test shipments showed that sometimes the baseline predictions were more accurate than the model's predictions. The plots of the test shipments created with XGBoost can be found in appendix 11. The plots of the other two models are comparable. As mentioned earlier, stacking multiple models can result in better performance. Therefore, stacking the models with the baseline predictions is tried.

Stacking can be performed with several methods. In this study, the Python package 'vecstack' is used. The training data (325137 instances) is split into 4 folds, which is default for the vecstack method. Figure 6.9 shows the first fold of one first-level model of the stacking process. In each fold the out-of-fold part of the training data is predicted (green) using the other folds (orange). To avoid overfitting of the training data, cross-validation is used. After the 4 folds, the predicted training values are used as training features for the second-level model (blue). The complete test set (77780 values) is also predicted in every fold (green). After the 4 folds, the predicted test set values (4 arrays of 77780 values, blue) are averaged to create a single test set. The process is repeated for every other first-level model. The result is a set of train and test features for each first-level model. These features are the input for the second-level model. Appendix 12 visualizes the overall stacking process in more detail.

In this study, best performance was achieved with all three first-level models: Lasso, XGBoost, and Random Forest. The optimal feature sets for the three separate models, as identified with SFS, were merged into one set of 32 features since only one set can be used as input for all models. The Lasso's alpha is set at 0.2 and the XGBoost parameters are: maximum tree depth 3, learning rate 0.2, estimators 100. The Random Forest uses 15 estimators and a maximum three depth of 5. After stacking, three arrays of predicted train and test values remain. The mean MSE and standard deviation of each model are 3.61 and 0.26 for XGBoost, and 3.67 and 0.21 for Lasso, and 3.65 and 0.25 for Random Forest.

The baseline predictions, the values belonging to variable 'm_mean_t_t_1' containing both train and test values, are added to the stacking results to see whether they contribute to better model performance. The baseline predictions can easily be added as a fourth array to the existing three arrays from the stacking results. Hence, the second-level model has four features: the three arrays of predicted train values originating from stacking the first-level Lasso, XGBoost, and Random Forest models, and one array containing the baseline training values. The stacked model performance was found to be worse with inclusion of the baseline values. Therefore, the baseline values are not added to the stacking results, and we continue with the three arrays from the stacking results.

XGBoost is used as second-level model because it is relatively quick compared to other methods, and therefore very useful in combination with gridsearch to tune the parameters. Furthermore, XGBoost was the best performing model (table 6.6). The optimal second-level parameters are: learning rate 0.1, maximum tree depth 2, number of estimators 100.

When looking the validation scores of the stacking model (table 6.7) and the XGBoost model, it is visible that the performance of the stacked model (R^2 0.215 and MSE 3.60) is slightly worse than the performance of the individual XGBoost model (R^2 0.218 and MSE 3.58) (table 6.6). Therefore, it can be concluded that the single XGBoost model is the optimal model. The model's performance is further discussed in the next chapter 'Evaluation'.

Set/metric	*MAE*	*MSE*	*RMSE*	*R^2*
Training	1.21	3.41	1.85	0.253
Validation	1.25	3.60	1.90	0.215
Test	1.22	3.15	1.78	0.299

Table 6.7: Performance stacking

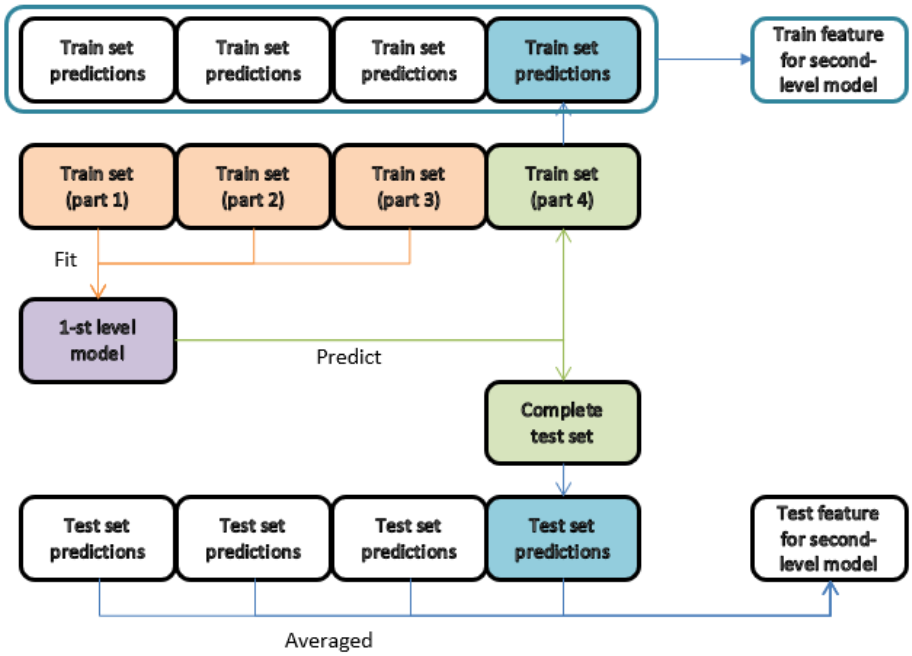


Figure 6.9: First fold of one first-level model

7 Evaluation

In the previous chapter the modeling phase was finished with the creation of the stacking model, which was found to perform slightly worse than the individual XGBoost model. According to the CRISP-DM methodology, the model and its predecessors need to be evaluated. This chapter describes the performance of the model and whether or not it suffices the business goal.

7.1 Overall performance of XGBoost model

It was found that the individual XGBoost model performs better than the stacking model. As input for the stacking model, the combined set of all (32) relevant variables for Lasso, Random Forest, and XGBoost was chosen. Changing the stacking model's input variables to the set of (23) relevant variables for the XGBoost model did not have any positive effect on the stacking model's performance.

In chapter 6.3.4, a baseline prediction was created in order to compare any model's performance against the variable containing the mean measured temperature of the same milestone on the previous day. The baseline performance had a MAE of 1.48, MSE of 4.72, and RMSE of 2.17. Compared to the baseline, the XGBoost model performs better with a MAE of 1.22, MSE of 3.15, and RMSE of 1.78. The model fit, expressed by the R^2-value of 0.299, indicates a model fit better than simple average. The model seems to perform significantly better on the test set than on the training set, which could be due to the way the data was split in a training and test set. This will be discussed in chapter 9.

Figures 7.1 and 7.2 show the absolute difference in actual and predicted temperatures. Most predictions, over 60.000 of the 77780 test values, are within 2 °C from the target value. A few hundred values have an exceptionally large deviation from the actual temperature. Such large deviations are hard to predict for any model, but have a significant effect on especially the MSE. The relatively large number of predicted values with a deviation larger than 3 °C could indicate that these are created by complete outlier shipments. Removing values with such large deviations can improve the model's performance, but it is hard to identify a sensor instance as an outlier which will be discussed later.

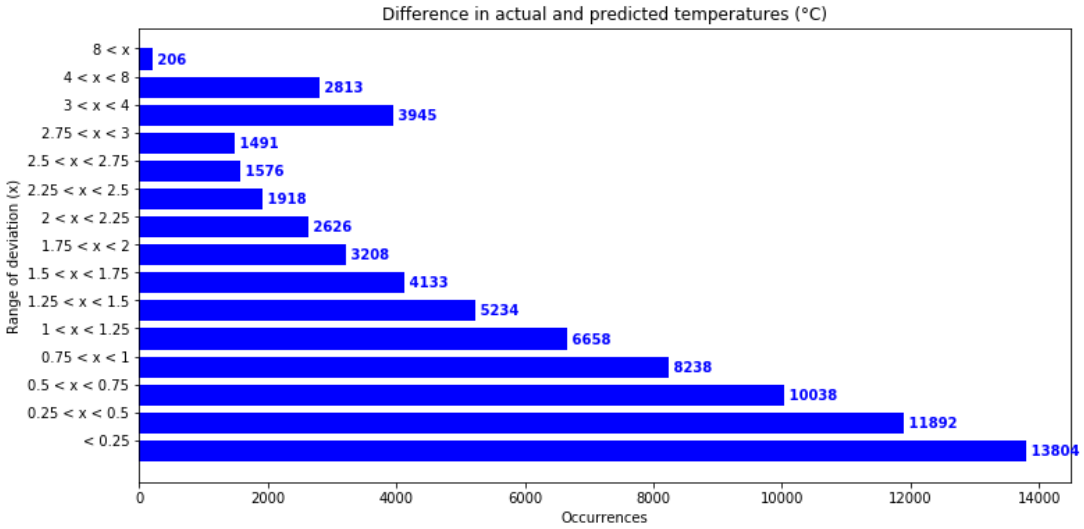


Figure 7.1: Absolute difference actual and predicted values (1)

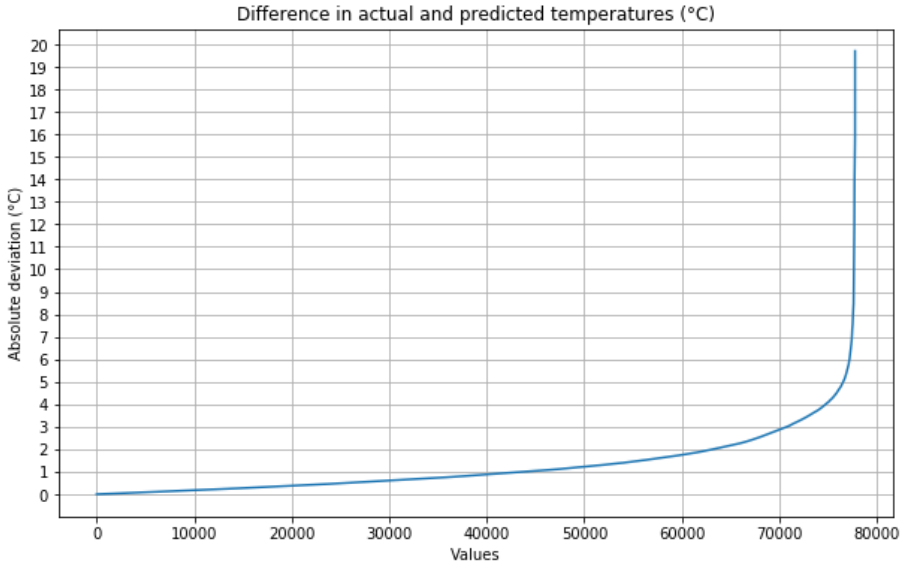


Figure 7.2: Absolute difference actual and predicted values (2)

7.2 Performance on test shipments

The performance of the Random Forest model using randomly split data (split 1) was previously tested on two winter shipments and two summer shipments. Graphs (figures 6.4-6.7 and appendix 7) showed the actual temperature values, the values predicted with the machine learning model, and the values based on the baseline average. The same shipments were predicted with the XGBoost prediction model and are displayed in figures 7.3-7.7 and in enlarged format in appendix 11. Table 7.1 shows the prediction performance values of the random forest and the stacking model on the test shipments. It can be concluded that the performance of the stacked model is better than the performance of the random forest with random train-test split.

	Random forest (split 1)			
Test shipment	*MAE*	*MSE*	*RMSE*	*R^2*
Winter shipment 1	1.41	3.29	1.81	-4.40
Winter shipment 2	1.96	6.44	2.54	-0.43
Summer shipment 1	0.59	0.66	0.81	0.48
Summer shipment 2	0.76	0.86	0.93	-0.65
	XGBoost model (split 2)			
Winter shipment 1	1.15	1.89	1.38	-2.109
Winter shipment 2	2.10	7.08	2.66	-0.569
Summer shipment 1	0.49	0.49	0.70	0.074
Summer shipment 2	0.65	0.71	0.84	0.442

Table 7.1: Prediction performance of XGBoost model on test shipments

Both models seem to perform very poor on the winter shipments. The test shipments were chosen based on their season and on the requirement that they included all seven milestones. This resulted in two winter shipments of January 2018, and two summer shipments of July 2016. The winter shipments may contain some extreme values, and, as mentioned earlier, it may have been better to choose shipments with more time in between occurrence and from different years. On the other hand, the test shipments' main purpose is to visualize what the model can do. To test whether the overall performance on winter shipments is worse compared to summer shipments, the model is separately tested on only winter and summer shipments. Of the 77780 data points in the test set, 28353 have season winter and 16155 have season summer. The overall performance of the model on the summer shipments is better. Table 7.2 summarizes the performance. The better performance on summer shipments is probably related to the product type which has a temperature range of 15-25 °C. Temperatures during the summer in Luxembourg and Huntsville are not very extreme, and therefore relatively few excursions occurred. During winter, it is much easier to get outside of the product range due to the low temperatures. Winter shipments are therefore harder to predict. It can be concluded that the two winter shipments have been poorly chosen, and are probably extreme cases.

Season/metric	*MAE*	*MSE*	*RMSE*	*R^2*
Summer	1.05	2.54	1.59	0.286
Winter	1.27	3.14	1.77	0.162

Table 7.2: Comparison of final model on only winter and summer shipments

7.3 Relevant features

This study consists of a thorough search for the optimal set of features. With RFE, the relative feature importances, and SFS, we were able to select a set of most important features for each model separately. The most relevant features are the temperature measured during the same milestone one and two days before and the weather temperature. Other important features were the average measured temperature during the same milestone on the closest previous shipment, the ordering variables that indicated where during a milestone the sensor instance was created, and the mean, minimum and maximum sensor and weather temperatures measured monthly and one day before. The relevant features are shown in appendix 9.

7.4 Meeting goals

The business goal was to improve lane risk assessments using data mining on temperature sensor data. The corresponding project goal was to find a machine learning technique that predicts the ambient temperature profile of a new shipment. The best performing model is the model using XGBoost. XGBoost is known for its scalability, speed, and robustness. Training the complete model and applying it on test shipments is a matter of seconds. The XGBoost model is considered a good choice to improve lane risk assessments because it provides better predictions than the baseline predictions and is very fast.

7.5 Real-time predictions

The initially created models with a random train-test split may have high potential to predict real-time temperatures. These models, especially when a unique number indicating the combination of shipment number and sensor device (variable 'combi') is added, have high predictive potential since in real-time, previously measured temperatures of the same shipment are known. When the model with variable 'combi' was applied on the test set, the performance indicated a model fit of 0.874 (R^2) and MAE of 0.36.

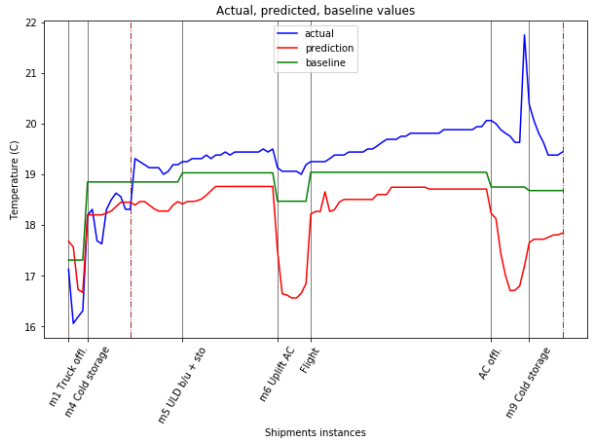


Figure 7.3: Winter shipment 1

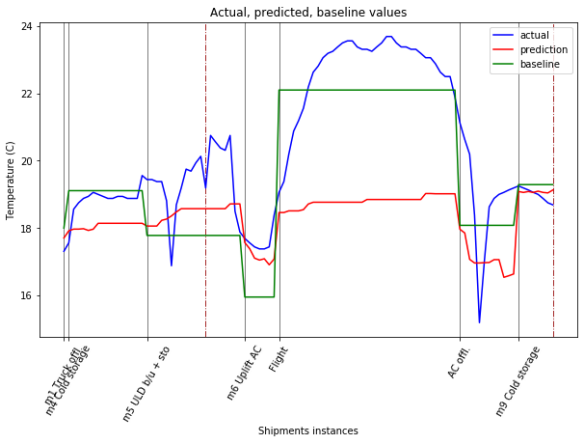


Figure 7.4: Winter shipment 2

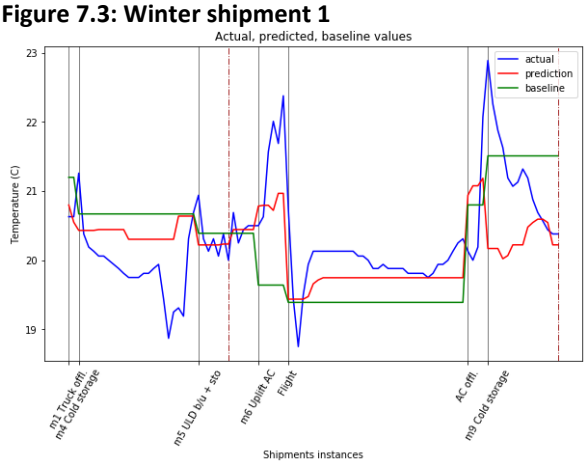


Figure 7.5: Summer shipment 1

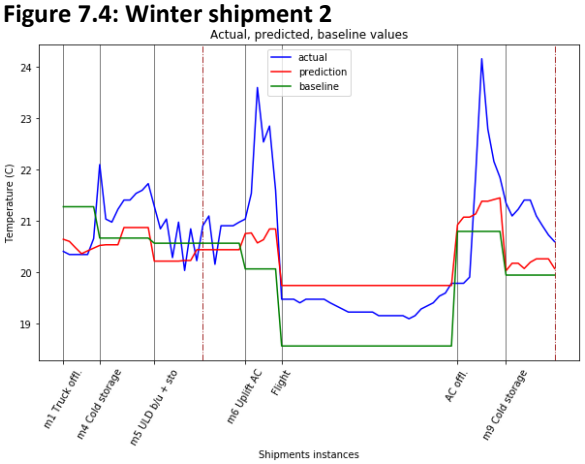


Figure 7.6: Summer shipment 2

7.6 Implementation and potential of machine learning in pharmaceutical air freight

For Validaide, implementating a machine learning model to improve lane risk assessments is hard due to the amount of data preparation that is required before modeling can start. Data is constantly being collected and these shipments need a lot of data preparation before they can be used to make predictions. Therefore, it might be easier for Validaide to use the average temperature profiles as created in chapter 5.1.6. This study, however, provided useful insights in how temperature sensor data can be used in pharmaceutical air freight, how data collection can be improved, and how machine learning algorithms can be applied.

In this study, only historical weather data was used. In order to apply the model to make forecasts of future shipments, the weather temperature feature must be replaced with weather forecast temperatures. The model will be trained with the historical weather data, but for the prediction itself the weather forecast data is used. Dark Sky provides access to future weather data through an API. This API returns the weather forecast for each hour for the next 48 hours or for each day for the next week. The data output for the next 48 hours is therefore the most useful. Data interpolation can be performed to obtain the weather forecast for every 15 minutes, instead of every hour.

8 Conclusions and recommendations

The chapter provides the main conclusion of this study. The main research and sub-questions are answered. Furthermore, limitations of this study and recommendations for future studies are given.

8.1 Main conclusion

The goal of this study was to help improve lane risk assessments by finding a machine learning technique that can predict temperature profiles of new shipments, based on the available sensor data. The literature study did not guide us to using a specific machine learning model, thus several different models were selected ourselves: Lasso, Random Forest, AdaBoost, Gradient Boosting, and XGBoost. The performance of all models is quite similar and the largest difference is in the computation times of the algorithms. XGBoost was found to have the best performance. Stacking models into a stacked generalization ensemble model did not improve performance. Overall, the performance of the XGBoost model is reasonably good. The models can help to improve lane risk assessments since the predictions are better than a simple average and better performance seems not possible with the used data. Implementation of machine learning for Validaide is hard due to the amount of data preparation that is required. Making real-time predictions for current shipments might be promising since the models that learned to which shipments specific instances belonged, had very good performance even without parameter optimization.

This study provided insights in using different regression models on temperature sensor data. The insights can be used in pharmaceutical air freight, but also in other industries. The study also shows how predictions can be made with a dataset that is not a standard time series dataset and predicts complete shipments of varying size.

8.2 Answers to research questions

Main research question

How can machine learning be applied on temperature sensor data and other relevant data sources, in order to predict the ambient temperature profile and possible temperature excursions of future shipments to improve lane risk assessments?

The ambient temperature profile of future shipments can be predicted with a range of machine learning methods such as Random Forest, Lasso Regression, XGBoost, Gradient Boosting, and AdaBoost, applied on sensor data to which shipment milestones are linked. The prediction outcomes can be used as expected temperature profile for a new shipment to estimate temperature related risks, which in turn help to make lane risk assessments.

Sub-questions

Research questions were defined in chapter 1.3 and their answers are provided below:

Q1: How can the required data be collected and prepared to create a workable dataset?

The sensor and milestone data was collected from SmartView by web scraping. Milestones were assigned to sensor instances according to their shipment number and timestamp. Historical weather temperatures were obtained through DarkSky's weather API and added to the dataset through their timestamp and the milestone's location. General shipment information was exported as a csv-file from SmartView, and added to the dataset with their similar identifier: the shipment number. Data preparation tasks involved converting the milestones to generic milestones by text mining, removing instances with missing and faulty data, and adding additional variables.

Q2: Which other data sources besides sensor data are valuable to add to the model in order to obtain a better temperature prediction?

The milestones are required to make the data more interpretable. Using the milestones as separate features did not have much added value, but their influence may be incorporated in other features. Adding the weather data proved useful, since weather related variables were good predictors for the models. The modeling phase of this study focused on only one lane with one product type. If more product types were included, then the product type originating from the general shipment data would probably have been an important feature. Other features originating from the general shipment data proved useless. Features related to the measured sensor temperature are found to be most important.

Q3: How can features be selected and, if required, be constructed or transformed, to build the temperature prediction model?

In this study, features were selected with the relative feature importances, RFE, and SFS. Of these three, SFS is the most thorough method but also computationally the most demanding method. First, RFE was used in combination with Lasso. This resulted in 53 important features, however, a near optimum was found at 12 features. These 12 features could not be identified. Therefore, a feature set was created including all variables that were found to be important according to the relative feature importances of the remaining four models. SFS was then applied on these 35 features with Lasso, Random Forest, and XGBoost, since these three models are relatively quick. For each model a best feature set was obtained. For the stacking model, all 35 variables were used.

Several features were constructed using other features in the dataset. Examples are extracting the month from the timestamp, or calculating the mean measured sensor temperature during the same closest milestone. Milestones were transformed to generic milestones. Categorical variables were changed to separate dummy variables for each value of a variable, in order to use them as input for the prediction models.

Q4: Which machine learning technique is the most appropriate to do these temperature predictions?

XGBoost was found to have the best performance. The machine learning technique proved to give the most accurate predictions, is easy to implement, and is computationally fast.

Q5: How can the temperature prediction model be tested and validated?

Four test shipments were extracted from the complete dataset. The remaining data was split into a training and test set. The training set was used to build the prediction models. Models were validated with cross-validation using the training set, by iteratively taking one of the folds as validation set and the other folds as training set. Optimal model performance was achieved by parameter tuning using gridsearch. The final models were tested on the test set but also on the four shipments including visualization of the predictions against the actual and baseline values. Performances are determined using MAE, MSE, RMSE, and R^2.

8.3 Limitations and recommendations for future research

This study has several limitations which are mentioned below. A number of recommendations for feature research are pointed out.

8.3.1 Limitations

Data quality

This study started by cleaning the data belonging to 9991 shipments, resulting in 5301 cleaned shipments that mainly belonged to lanes LUX-HSV and LUX-HSV-MEX and product types 'cold' 2-8 °C and control room temperature 15-25 °C. Modeling was performed with 2070 shipments of lane LUX-HSV and with CRT products. The quality of the sensor data itself is reasonably good. The quality of the milestones is much worse. Over 9000 unique milestones had to be converted to a generic milestone. Manually entering the milestones into SmartView gives a lot of room for errors and different interpretations of milestones. Also, the accuracy of the timestamp is doubtful, meaning that milestones may be linked to the wrong sensor instances. Furthermore, not all shipments have all milestones that were identified as part of the logistic process in this study. Moreover, the available milestones that were used to describe the logistic process do not suffice to describe the actual logistic process executed in practice. This will be further explained below in the recommendations section. At last, as shown in chapter 5, the shipments and milestones have a very different duration which makes the data inconsistent and it is therefore harder to make accurate predictions. This also led to 75% missing values of two of the most important variables: the temperature measured one and two days before. These values were filled with the average temperature during that milestone during that month. Due to the limited data quality, it was impossible to use the potential of machine learning to its full extend. The performance of machine learning models is highly dependent on the data quality.

8.3.2 Recommendations

Data collection

The logistic process could be more precisely described by the milestones in SmartView. For instance, the milestone 'ULD build-up and storage in temperature controlled area' in LUX could be split in two different milestones, one indicating the ULD build-up and one indicating the storage afterwards. Several milestones should be added at SmartView to provide a better representation of the actual logistic process: flight, A/C offloading, ULD breakdown, departure of the aircraft from HSV to MEX.

Prediction goals

Future research could focus on making real-time predictions for temperature controlled air freight shipments. As previously shown, when prediction models can determine to which shipment an instance belongs, they seem to be able to make accurate predictions. Hence, when several real-time values of a shipment are known to the model, it may have the potential to accurately predict future temperatures of that shipment.

Research could also focus on the occurrence and intensity of excursions to improve lane risk assessments. Not all temperature excursions have the same negative influence on the products and their impact is closely related to product thermal stability features. These indicate how long a product can be exposed to temperatures outside the product range before damage is incurred. Such a study could be defined as a classification problem.

Another possibility is studying the deviation from the average temperature profiles as defined in chapter 5. Since only 35993 temperature excursions were found on a total of 2236737 sensor instances (additional lanes and product types), excursions do not seem to occur very often and thus most shipments seem to follow the average temperature profile. By studying the excursions, possible risk factors can be identified.

A last suggestion is to predict the probability of occurrence of temperature excursions under different circumstances, such as the milestone, time of the day, month, and weather temperature. Measurements could be clustered into each specific circumstance and subsequently used to make predictions. Such probabilities could possibly be easier to predict and are still very useful for lane risk assessments.

Shipment and milestone duration consistency

Future research could focus on shipments and milestones that follow a 'standard' logistic process by inclusion of shipments with only the average milestone durations. Furthermore, since cargo uplift and A/C offloading are the activities during which the most and the largest excursions occur, studies could focus on these milestones.

Transport lanes and product types

The 5301 cleaned shipments mainly belonged to lanes LUX-HSV and LUX-HSV-MEX and product types cold (2-8 °C) and CRT (15-25 °C). Due to time restrictions, modeling was only performed on lane LUX-HSV and product type CRT. Future research could focus on modeling lane LUX-HSV-MEX and the other product types. Completely new transport lanes are also interesting to investigate when data is available.

Features

Machine learning models are very dependent on their predictive features. Future research could focus on feature construction. An option is using the average temperature profiles from chapter 5, which can be created for any time period and milestone. Since relatively only few excursions occurred, most shipments seem to follow a normal temperature profile and therefore these features can be of value.

9 Discussion

In this chapter the initial goal and the overall results of this study are discussed.

Milestone timestamp accuracy

Milestones are entered into SmartView manually and automatically. Manual milestones are entered by local Panalpina staff who follow specific SOP's which are tied to customer agreements. Hence, manual milestones are different from place to place, resulting in no standard way of introducing them. The Panalpina staff is also not always present and receives the data from third party subcontractors. Hence, the staff has only indirect control over the physical freight flows and processes. Since different people and subcontractors enter the milestones following different SOP's, the correctness of the timestamp linked to a milestone may be disputed.

Reduce noise

In this study, the actual measured temperature sensor values were used as training input. These values may contain some noise, which is random error or variance (Han et al., 2012). Removing the noise could have resulted in better predictions. One method is using binning (Han et al., 2012), in which surrounding values are used to perform local smoothing. According to Han et al. (2012), noise should always be removed before dealing with outliers, which is the next topic of discussion.

Outliers

Outliers are values that significantly differ from the other measurements (Han et al., 2012). Figure 5.13 in chapter 5.3.4 shows a plot of the sensor measurements. The plot shows some steep highs and lows. Several outliers were found during this project. Complete shipments were found to be not temperature controlled at all. Also, sometimes temperature controlled storage facilities seemed not temperature controlled by looking at the data. Furthermore, milestone 'Flight' contains relatively many measurements that continuously exceed the product range. In this study, all these values were retained in the final dataset. This was done since it is very hard to determine when some value is an outlier. All values were measured by the sensor devices and these do not seem to be defective. The outliers are probably caused by cargo handling errors, for instance, handling temperature restricted cargo as being not temperature restricted. This makes it very hard to identify when a measurement or set of measurements can be considered an outlier. Additional outlier detection could have been performed during this study. Especially focusing on identification of contextual outliers (season, milestone) and collective outliers (complete shipments, all instances of a milestone). However, due to time restrictions this was not done. The created prediction model shows how well predictions can be made considering all data.

Standardization and normalization of data for Lasso

Standardization of data rescales variables to have a mean of 0 and standard deviation of 1. Normalization scales all values of the variables to scale ranging from 0 to 1. The Lasso method penalizes the absolute size of the coefficient. Therefore, it is important that all coefficients have the same relative size. This can be achieved by normalization and standardization of the data. In this study, the variables were not standardized nor normalized. Possibly better results could have been achieved by either standardization, normalization or both.

Splitting the dataset

A different model performance was found by using different data splitting method into a training and test set. By looking at figure 5.3 in chapter 5.1, it is visible that on average more shipments take place during the seasons fall and spring. It may have been a better choice to randomly split the data, but keeping instances belonging to the same shipment together. By doing so, a more accurate distribution of shipments of each month would have been obtained. This would have resulted in a more constant performance on each set. Therefore, the current data split provides a possible reason for the constant better model performance on the test set.

Excursions

This study aimed to predict a complete temperature profile of a shipment. After combining the data of both lanes LUX-HSV and LUX-HSV-MEX, 2236737 instances remained. This data contained 35993 excursions. Once the data of only LUX-HSV remained for the modeling phase, this data consisted of 402917 instances of which 17331 with temperature excursions. Relative to the total number of instances, the number of temperature excursions is very low. The instances containing excursions could have been further investigated why specifically these data points have an excursion. Doing this could have provided additional insights and may have led to ideas for additional features. Furthermore, when looking at figure 5.4 in chapter 5.1, excursions seem to be relatively large when they occur, and therefore have a large influence on the predictive performance, especially the MSE. Modeling whether or not a temperature excursion will occur with this few excursions may not be very useful.

References

Ahmed, N. K., Atiya, A. F., El Gayar, N., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, *29*(5), 594–621. https://doi.org/10.1080/07474938.2010.481556

Ammann, C. (2013). Handling Temperature Excursions and the Role of Stability Data. *Pharmaceutical Outsourcing*.

Aung, M. M., & Chang, Y. S. (2014). Temperature Management for the Quality Assurance of a Perishable Food Supply Chain. *Food Control*, *40*, 198–207. https://doi.org/10.1016/j.foodcont.2013.11.016

Breiman, L. (1996a). Bagging Predictors. *Machine Learning*, *24*(2), 123–140.

Breiman, L. (1996b). Stacked Regressions. *Machine Learning*, *24*, 49–64.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*, 5–32.

Brockwell, P., & Davis, R. (2002). *Introduction to Time Series and Forecasting*. *Technometrics*. https://doi.org/10.2307/1271510

Chatfield, C. (2000). *Time-Series Forecasting*. Retrieved from www.crcpress.com

Chen, K. Y., & Shaw, Y. C. (2011). Applying back propagation network to cold chain temperature monitoring. *Advanced Engineering Informatics*, *25*(1), 11–22. https://doi.org/10.1016/j.aei.2010.05.003

Chen, T., & Guestrin, C. (2016). XGBoost : A Scalable Tree Boosting System.

D'Orazio, M. (2015). Setting standards. *Pharmaceutical Manufacturing and Packing Sourcer*, *2*.

DARA. (2018). Project Plan DARA : Data Analytics for Trade Lane Risk Assessments and Control, 1–18.

De Jong, E. (2017). Lane Risk Assessment Algorithm, 1–21.

Enyinda, C., Briggs, C., & Bachkar, K. (2009). Managing Risk in Pharmaceutical Global Supply Chain Outsourcing : Applying Analytic Hierarchy Process Model. *American Society of Business and Behavioral Sciences*, *16*(1), 19–22. Retrieved from http://asbbs.org/files/2009/PDF/E/EnyindaC.pdf

Flach, P. (2012). *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press. https://doi.org/10.1145/242224.242229

Freund, Y., & Schapire, R. E. (1999). A Short Introduction to Boosting. *Journal of Japanese Society for Artificial Intelligence*, *14*(5), 771–780.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*, 367–378. https://doi.org/10.1007/BF02938375

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques* (Third Edit). Morgan Kaufmann.

Hastie, T., Tibshirani, R., & Friedman, J. (2017). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Second Edi). Springer.

Hsu, C. I., & Liu, K. P. (2011). A model for facilities planning for multi-temperature joint distribution system. *Food Control*, *22*(12), 1873–1882. https://doi.org/10.1016/j.foodcont.2011.04.029

IATA. (2018). *How to Become CEIV Pharma Certified*. Retrieved from https://www.iata.org/whatwedo/cargo/pharma/Documents/ceiv-pharma-specifications.pdf

Jaberidoost, M., Shekoufeh, N., Akbar, A., & Rassoul, D. (2013). Pharmaceutical supply chain risks: a systematic review. *DARU Journal of Pharmaceutical Sciences*, *21*, 1–7. Retrieved from http://www.darujps.com/content/21/1/69

Kumar, N., & Jha, A. (2016). Temperature excursion management: A novel approach of quality system in pharmaceutical industry. *Saudi Pharmaceutical Journal*, *25*(2), 176–183. https://doi.org/10.1016/j.jsps.2016.07.001

Kuo, J. C., & Chen, M. C. (2010). Developing an advanced Multi-Temperature Joint Distribution System for the food cold chain. *Food Control*, *21*(4), 559–566. https://doi.org/10.1016/j.foodcont.2009.08.007

Lehikoinen, A., Luoma, E., Hänninen, M., Storgard, J., & Kuikka, S. (2012). Probabilistic Risk Assessment and Decision Support Tools for the Evaluation of Oil Transport in the Gulf of Finland, North-Eastern Baltic Sea. *International Environmental Modelling and Software Society (IEMSs)*. Retrieved from http://www.iemss.org/sites/iemss2012/proceedings/B3_1365_Lehikoinen_et_al.pdf

Montanari, R. (2008). Cold chain tracking: a managerial perspective. *Trends in Food Science and Technology*, *19*(8), 425–431. https://doi.org/10.1016/j.tifs.2008.03.009

Moraru, A., Pesko, M., Porcius, M., Fortuna, C., & Mladenic, D. (2010). Using Machine Learning on Sensor Data. *Journal of Computing and Information Technology*, *4*, 341–347.

Shang, Y., Dunson, D. B., & Song, J.-S. (2015). Exploiting Big Data in Logistics Risk Assessment via Bayesian Nonparametrics, 1–50. https://doi.org/10.1287/opre.2017.1612

Tang, J., Alelyani, S., & Liu, H. (2014). Feature Selection for Classification: A Review. *Data Classification: Algorithms and Applications*, 37–64. https://doi.org/10.1.1.409.5195

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal Ofthe Royal Statistical Society, Series B (Methodological)*, 267–288.

Ting, S. L., Tse, Y. K., Ho, G. T. S., Chung, S. H., & Pang, G. (2014). Mining logistics data to assure the quality in a sustainable food supply chain: A case in the red wine industry. *International Journal of Production Economics*, *152*, 200–209. https://doi.org/10.1016/j.ijpe.2013.12.010

Van der Plas, J. (2017). *Python Data Science Handbook* (First Edit). Sebastopol: O'Reilly Media Inc.

Wirth, R., & Hipp, J. (2000). CRISP-DM : Towards a Standard Process Model for Data Mining.

Proceedings of the Fourth International Conference on the Practical Application of Knowledge Discovery and Data Mining, 29–39. https://doi.org/10.1.1.198.5133

Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, *5*(2), 241–259.

Xu, W., Zhang, Z., Gong, D., & Guan, X. (2014). Neural network model for the risk prediction in cold chain logistics. *International Journal of Multimedia and Ubiquitous Engineering*, *9*(8), 111–124. https://doi.org/10.14257/ijmue.2014.9.8.10

Zhang, G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. *Neurocomputing*, *50*, 159–175. https://doi.org/10.1016/S0925-2312(01)00702-0

Appendices

Appendix 1: Web scraping algorithm

The web scraping tool broadly follows the following steps:

1. Open a web browsing window
2. Surf to the SmartView webpage containing an overview of all shipments
3. Access the page by entering username and password
4. Enter the origin and destination of the lane to view lane specific shipments
5. Change webpage layout from 25 to 100 shipments per page
6. Create a list of the shipment numbers shown on the page (maximum 100)
7. Open the first shipment in a new tab and go to this tab
8. Open a csv-file and write to a new line the milestone's number, timestamp, user name, and description, and add the shipment number
9. Move to the tab 'Graph' and click the download csv button such that the csv-file containing all sensor data is downloaded. The csv-file is downloaded to the download folder.
10. Close the tab of this shipment and go back to the shipment overview
11. Determine from the list with shipment number which next shipment's data needs to be scraped. If the end of the list is reached, go to step 12. If not, repeat steps 7 to 10 for the next shipment.
12. Repeat step 11 for the next shipment. If the end of the list was reached, go the next page containing the next 100 shipments
13. Repeat steps 6 to 12. Once a user-specified total number of shipments is reached, or no more shipments are available, the algorithm stops.

Appendix 2: Milestone classification by text mining

Sub-selection: Truck

One of the most occurring words in the milestones is the word 'truck' in any combination of lower and capital letters. Of the 61004 milestones, 10538 contain this word. Of these 10538 milestones, 2124 are unique. By looking at these unique milestones, several of the generic milestones can be identified. Then, a code is written in Python that searches for specific combinations of words that make sure only milestones are selected that must be classified to a specific generic milestone. Code consists of 'AND'- and 'OR'-operators in combination with words. In the code, the AND-operator is '&', the OR-operator is '|', and '!' means that the word must not occur in the milestone. Lower and capital letters are ignored and both accepted. Table A2.1 indicates the generic milestone, the word combinations used in the code, and the number of milestones that is converted to the generic milestone. As an example, the code for the milestone 'Truck opened and start unloading (LUX)' means: if a milestone contains the word 'opened' or the words 'truck' and 'offloading', or the words 'door' and 'opening', or the words 'truck' and 'arriv' and 'lux', or the words 'truck' and 'temp', then the milestone is converted to this generic milestone. The parentheses indicate a word combination, thus all words must occur. Some milestones were manually assigned to 'Useless'.

Generic milestone	Words used in code for identification	Number of milestones converted
Truck opened and start unloading (LUX)	opened | (truck & offloading) | (door & opening) | (truck & arriv & LUX) | (Truck & temp)	5105
Cargo stored in temperature controlled area (LUX)	Cargo & stored	9
Loading truck start (HSV)	(temp & loading) | (tmep & loading) | loading units onto truck | loading unit on truck | fedex | loading & starting | (loading & begins) | (loading & batt) | (Begin & loading) | Pre-loaded onto truck | All pcs placed & truck | (loading & truck & !issue)	3661
Loading truck complete (HSV)	complete | (finish & loading) | (final & loaded) | (all & loaded) | truck loaded	1755
Useless	Manually assigned	8

Table A.2.1: Generic milestone conversion 'Truck'

Sub-selection: Cargo

One of the most occurring words in the milestones is the word 'cargo' in any combination of lower and capital letters. Of the 61004 milestones, 13056 contain at least one of these two words. Of these 13056 milestones, 1398 are unique. By looking at these unique milestones, several of the generic milestones can be identified. While assigning the milestones to the generic milestones, it was found that some shipments contain a milestone 'Aircraft offloading' or something with the same meaning. This milestone and other milestones indicating what exactly happens with shipments at HSV are almost always missing. Since this is an important milestone, this milestone is added to the list of generic milestones. The identification of generic milestones with Python and the corresponding code is shown in table A1.2 below:

Generic milestone	Words used in code for identification	Number of milestones converted
Cargo stored in temperature controlled area (LUX)	(tem & HLC & !ULD) | (temp & pharm & center & !ULD) | (temp & stored & ambient & !ULD) | (tem & area|arae & !ULD) | (Cargo stored in temp controlled cell & !ULD) | (transfer & customer & arae | HLC & !ULD) | (stored in HLC & !ULD) | (customer & request & stored & !ULD) | (delay & unload & returned & !ULD) | (stored in container & !ULD)	5030
Cargo build-up on ULD and stored in temperature controlled area (LUX)	ULD | (buil|buid & up) | (stuff|stuffed|stuffing|staffed & RAP|RKN|tainer) | (RAP & malfunction & pharma) | (Loading & dry & ice) | (RAP & stored & pharma & envirotainer)	5056
Aircraft offloading (HSV)	Manually assigned	9
Cargo at TME (MEX)	TME	1324
Release of cargo to customs broker (MEX)	(Release|handover & broker)	1606
Doc release to broker (MEX)	Manually assigned	1
Cargo stored in temperature controlled area (HSV)	Zone | storage into cooler | (customer & request & removed from cooler box)	13
ULD out of cell for uplift to flight (LUX)	Manually assigned	1
Loading truck start (HSV)	Manually assigned	2

Table A.2.2: Generic milestone conversion 'Cargo'

Sub-selection: RAP

While searching for the most often occurring words in the remaining unique milestones, the word 'RAP' occurs 3893 times and is found in 3131 unique milestones. This word is therefore found in a very large portion the remaining 4557 unique milestones. For this reason, the milestones containing this word will be assigned to a generic milestone first. A 'RAP' is a type of active air cargo container from the company Envirotainer. So many unique milestones exist since these containers have their own identification numbers, such as 'RAP80713PC', but they also generate milestones in the form of status updates indicating the battery level and the temperature inside the container. The table A.2.3 below indicates how many milestones are classified to which generic milestone. Many milestones are also classified as useless and contain status updates.

Generic milestone	Words used in code for identification	Number of milestones converted
Cargo build-up on ULD and stored in temperature controlled area (LUX)	Temax	2
Useless	!(m5|m6|m12|m11|m4)*	3485
ULD out of cell for uplift to flight (LUX)	(a/c & load) | (uplift|upload & flight) | (prepared for flt) | (last & check)	72

Release of cargo to customs broker (MEX)	(broker & release) | (broker & pick up)	157
Cargo at TME (MEX)	TME	173
Cargo stored in temperature controlled area (HSV)	(stored in cell & RAP|RKN) | (unload and stored)	4

* The m's indicate generic milestone numbers used to convert milestones to generic milestones

Table A.2.3: Generic milestone conversion 'RAP'

Sub-selection: Zone

At this point 1426 unique milestones still exist in the milestones file. The next most occurring word in the set of unique milestones is the word 'zone', with 620 occurrences. This word relates to a temperature controlled area at the airport of Huntsville. Cargo is moved to one of these zones in the airport. Off all milestones that are not yet classified to a generic milestone, 3528 milestones contain the word 'zone'. After checking the unique milestones containing zone, it is found that all milestones have the same meaning and can be converted to the milestone 'Cargo stored in temperature controlled area (HSV)'.

Sub-selection: Batt

As already mentioned when searching for the word 'RAP', the milestone data contains many battery updates of active containers. These containers are either of the type 'RAP' or of 'RKN'. By searching for the word 'Batt' to find all the remaining battery status updates, 163 milestones were identified of which 153 are unique. Only 5 of the 163 milestones are related to the milestone 'Cargo at TME (MEX)', and the other 158 are classified as useless.

Sub-selection: ULD|buil

After classifying all milestones containing 'Batt', still 654 unique milestones exist. The next words used to make a sub-selection of the remaining unclassified milestones are 'ULD' and 'buil'. Of the remaining milestones, 5006 contain at least one of these two words. The table A.2.4 below shows which milestones can be identified and which words are used in Python.

Generic milestone	Words used in code for identification	Number of milestones converted
Cargo build-up on ULD and stored in temperature controlled area (LUX)	(Buil & up & !finish)	13
ULD out of cell for uplift to flight (LUX)	(out & (uplift|upload|flight|upoad|uload))	4971
Cargo stored in temperature controlled area (HSV)	(unload & returned) | ((moved |stored | transfer) & (cell|area) & !buil)	21

Table A.2.4: Generic milestone conversion 'ULD|buil'

Sub-selection: TME

The number of milestones found by searching for the word 'TME' is 647 with 55 unique milestones. By looking at the unique milestones, it is quickly found that all these milestones belong to the same generic milestone: 'Cargo at the TME'. Therefore, all 647 milestones are classified as this milestone. The TME is the bonded warehouse of ground handling agent Tranportacion Mexico Express.

Sub-selection: arr|dep

Of the remaining 16671 milestones, 542 are unique and 245 of these unique milestones are related to aircraft departure and arrival milestones. Therefore, the next sub-selection filters on either 'arr' or 'dep' in the milestone description. One shipment is found to fly from Luxembourg to Mexico City, but makes an intermediate stop at JFK airport in New York instead of HSV. This shipment (BRU 574891 MEX) will be removed from the dataset after all milestones are converted to a generic milestone. It is also found that 34 shipments on lane LUX-HSV fly off to New York after stopping at HSV. These flights can be kept since we want the data until Huntsville, and milestones of what happens after Huntsville are missing anyway. The flight departure milestone from Huntsville to Mexico City is always missing, and therefore all milestones containing 'dep' can be assigned as departing from Luxembourg. The aircraft arrivals are a bit harder to categorize, since the milestone often misses the airport name. Therefore the shipment number was used, which almost always contains either 'HSV' or 'MEX'. Some shipments had to be looked up in SmartView to determine the arrival airport. Table A.2.5 below shows which milestones can be identified and which words are used in the Python code.

Generic milestone	Words used in code for identification	Number of milestones converted
Actual aircraft arrival (HSV)	((ShipmentNr=HSV|hsv|512601|362107 |882066|196045) & arr) | (ShipmentNr=MEX & 'Actual Time of Arrival') | ((ShipmentNr=MEX|JSV|ATL) & HSV)	4833
Actual aircraft arrival (MEX)	((ShipmentNr!=HSV) & arr & mex)	9
Actual aircraft departure (LUX)	Dep & !await	5738
ULD out of cell for uplift to flight (LUX)	Out of cell	1
Useless	Scheduled | estimated | loading resumed	6075

Table A.2.5: Generic milestone conversion 'arr|dep'

Sub-selection: rest 1

After the assignment of milestones in sub-selection 'arr|dep', the remaining amount of unique milestones is 303 and the total of unallocated milestones is 7532. The milestones are combined in sub-selection 'rest 1', simply indicating the remaining milestones. Most of the milestones can be allocated to 'Loading complete (HSV)' and 'Doc release to broker (MEX)'. After allocation to the generic milestones as indicated in the table A.2.6 below, there are still 3649 milestones remaining of which 116 are unique.

Generic milestone	Words used in code for identification	Number of milestones converted
Loading complete (HSV)	load|laod & complet & !stopped	1858
Doc release to broker (MEX)	broker & doc	1695
Release to customs broker (MEX)	broker & !doc	18
Cargo stored in temperature controlled area (HSV)	(all ps placed & (in|on)) | (all containers plug & (up|in)) | (breakdown|breadown)	96
Useless	all units | erroneous | assoicated | physical	7
Cargo stored in temperature controlled area (LUX)	(temp & controlled & (area|cell)) | cgo store | (stored in & (cell|hlc))	84
ULD out of cell for uplift to flight (LUX)	(upload | uplift) | (a/c loading | ac loading) | last check | cgo removed | for aircraft loading	125

Table A.2.5: Generic milestone conversion 'arr|dep'

Sub-selection: rest 2

To retain a better overview of the remaining unallocated milestones, a new sub-selection is created called 'rest 2' as defined in table A.1.6. Most of the milestones are related to 'Actual aircraft arrival (MEX)' by searching for the abbrevation 'ata', which stands for 'actual time of arrival'. To this generic milestone, 1726 milestones are allocated. Another 4 milestones are allocated to 'Actual aircraft arrival (HSV)'.

Generic milestone	Words used in code for identification	Number of milestones converted
Actual aircraft arrival (MEX)	Ata & !hsv	1726
Actual aircraft arrival (HSV)	Ata & hsv	4

Table A.2.6: Generic milestone conversion rest 2

Sub-selection: rest 3

The remaining milestones are again grouped in a new sub-selection called 'rest 3', which consists of 1919 milestones. Of these 1919 milestones, 1714 are assigned to 'Aircraft offloading (MEX)'. Table A.2.7 shows the assignment to 'Aircraft offloading (MEX)'.

Generic milestone	Words used in code for identification	Number of milestones converted
Aircraft offloading (MEX)	Offloading & aircraft	1714

Table A.2.7: Generic milestone conversion rest 3

Sub-selection: rest 4

The remaining 205 milestones are checked one by one and assigned to the correct generic milestone based on the specific words as indicated in table A.2.8.

Generic milestone	Words used in code for identification	Number of milestones converted
Actual aircraft arrival (MEX)	Ara flight	1
Actual aircraft departure (LUX)	ATD | envirotainers | 5Y600 | (filed & plan) | (dapart|dpart|deoart|deap) | (art|departure) | (flt & lux)	68
Cargo stored in temperature controlled area (HSV)	break down start | cold box | start storing plts |netting	6
Cargo build-up on ULD and stored in temperature controlled area (LUX)	Stuff | (loading & envirotainer) | (loading & rkn)	11
Cargo stored in temperature controlled area (LUX)	star controlled | (checked & ambient)	2
Loading truck complete (HSV)	Fedex	3
Aircraft offloading (MEX)	pallet start | offloading | ofloading | offloaiding | offloding | offfloading	17
Release to customs broker (MEX)	pallet finish	6
Useless	Enter | leave | Nijmegen | (ETA & HSV) | (container & loading) | (Pod & (reed|moore)) | PMC 00019 | rollerbed | stayed on plane | (smart & point) | paperwork | load loacks | pre-cooling | pre-conditioning | (loaded & await) | delay | test	75
Truck opened and start unloading (LUX)	opened & unload	2
ULD out of cell for uplift to flight (LUX)	(loaded & a/c) | (out of & cell)	8
Doc release to broker (MEX)	Original doc	3
Cargo at TME (MEX)	TME | (ambient & MEX)	3

Table A.2.8: Generic milestone conversion rest 4

Appendix 3: Average temperature profiles per milestone

Milestone 'Flight'

The first step is to determine the number of sensor instances for this milestone, and the second step is to determine the number of devices that registered these instances. These numbers are respectively, 192911 and 5431. For the third step, a graph is created of the number of sensor measurements for every single device. The minimum, maximum, mean, and median number of instances are respectively, 4, 139, 35, and 35. Figure A.3.1 shows this distribution graph, which indicates that most devices have between 30 and 40 sensor measurements, which makes sense since a flight duration is normally fixed. The graph also shows outliers with up to 139 instances, meaning that this flight took 139 times 15 minutes which is almost 35 hours. A normal flight from LUX to HSV should take about 9 hours, depending on the wind direction. Including some room for small delays, all devices with less than 30 or more than 45 sensor instances are removed from the dataset in chapter 5.2.

Step four can be skipped since this milestone is distributed over a small range. In step 5 the temperatures of every device is stored in a list. The length of these lists vary since devices have different numbers of measurements. In step 6, the values in the list are interpolated to 1000 values. So, if a list of a device had 34 measurements, these values are interpolated to 1000 values. By doing so, all lists will have an equal length and thus the average temperature profile over all devices can be calculated much easier. Then, in step 7, the averages of all x-th measurement over all devices are calculated. This results in 1000 average values. When we put the average value of every x-th measurement over all devices in a graph (blue line), then this results in figure A.3.2 for the warm months and figure A.3.3 for the cold months. The red line indicates the average temperature of all sensor measurements with milestone 'Flight' during the respective months, as calculated during step 8. The x-axis indicates the percentage of the milestone that is finished.

The graph shows that on average, for both the cold and the warm period, the temperature increases at the start of a flight towards the average temperature over all measurements for that milestone, and at the end of a flight, the temperature further increases to a level above the average measured temperature. The increase in temperature at the start of the milestone is much greater during the cold period. The steep drop in temperature at the start of the milestone during the summer period must be ignored, since it is probably created by a small number of devices with many measurements. Since on average the number of measurements is around 35 and values are only interpolated in between these measurements, the graph shows accurate results from around 3%. The few devices with many instances then influence the graph before 3%. This effect will be removed as described in chapter 5.2 by the suggested removal of devices with less than 30 or more than 45 sensor instances.

A possible clarification for the temperature profiles could be that the temperature drops in the cold period during the cargo uplift onto the aircraft in LUX and then the temperature slowly stabilizes in the aircraft's cargo area which is temperature controlled. During the warm period, the temperature increases much less since the temperature outside the warehouse is higher. At the end of the flight, especially during the warm period, the temperature increases which might indicate that the flight actually already ended and the cargo warms up because of the tarmac temperature. The timestamp of the aircraft arrival in HSV may not be accurate enough.

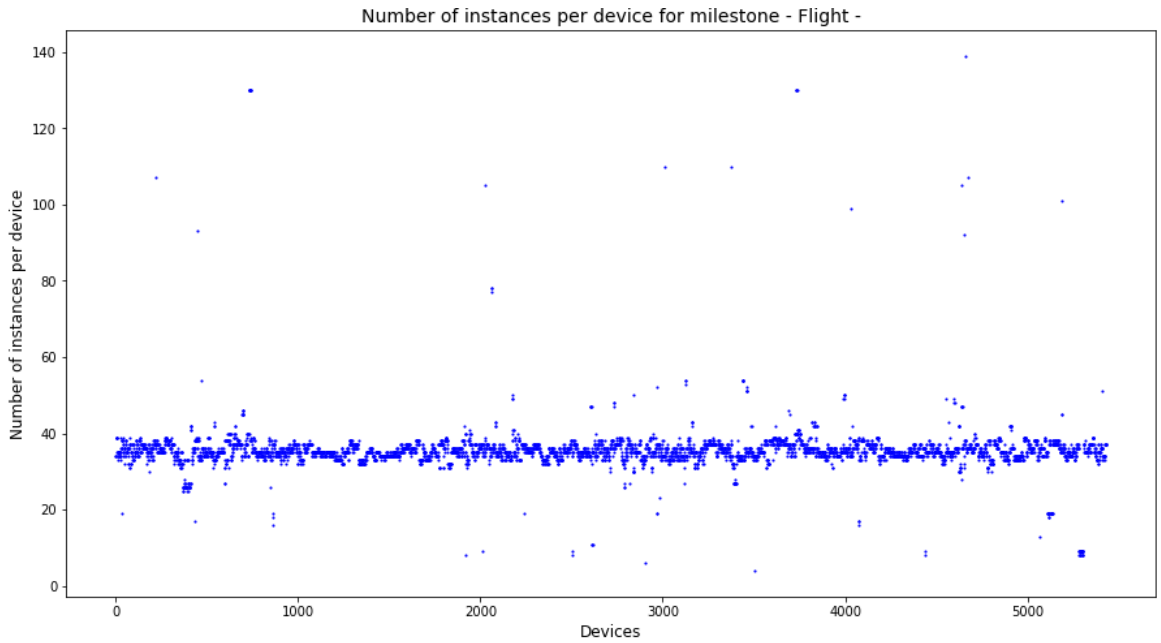


Figure A.3.1: Distribution graph of the number of instances per device for milestone 'Flight'

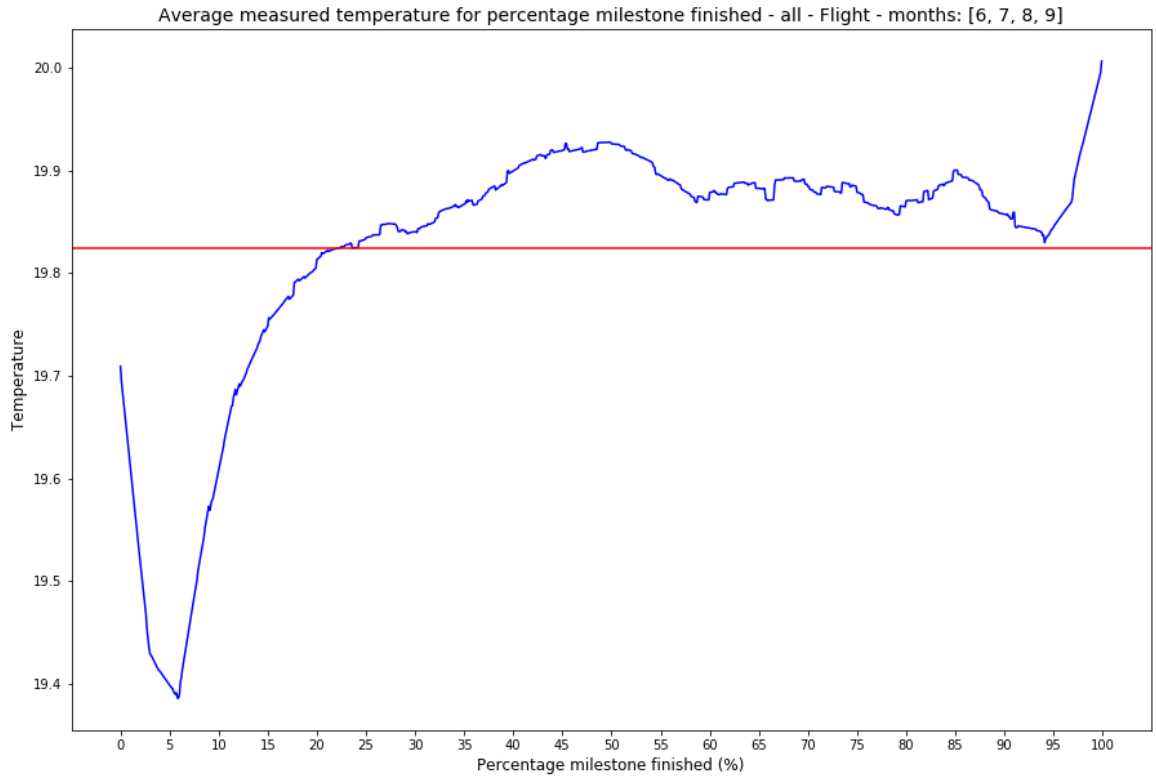


Figure A.3.2: Average temperature profile during milestone 'Flight' during the warm months

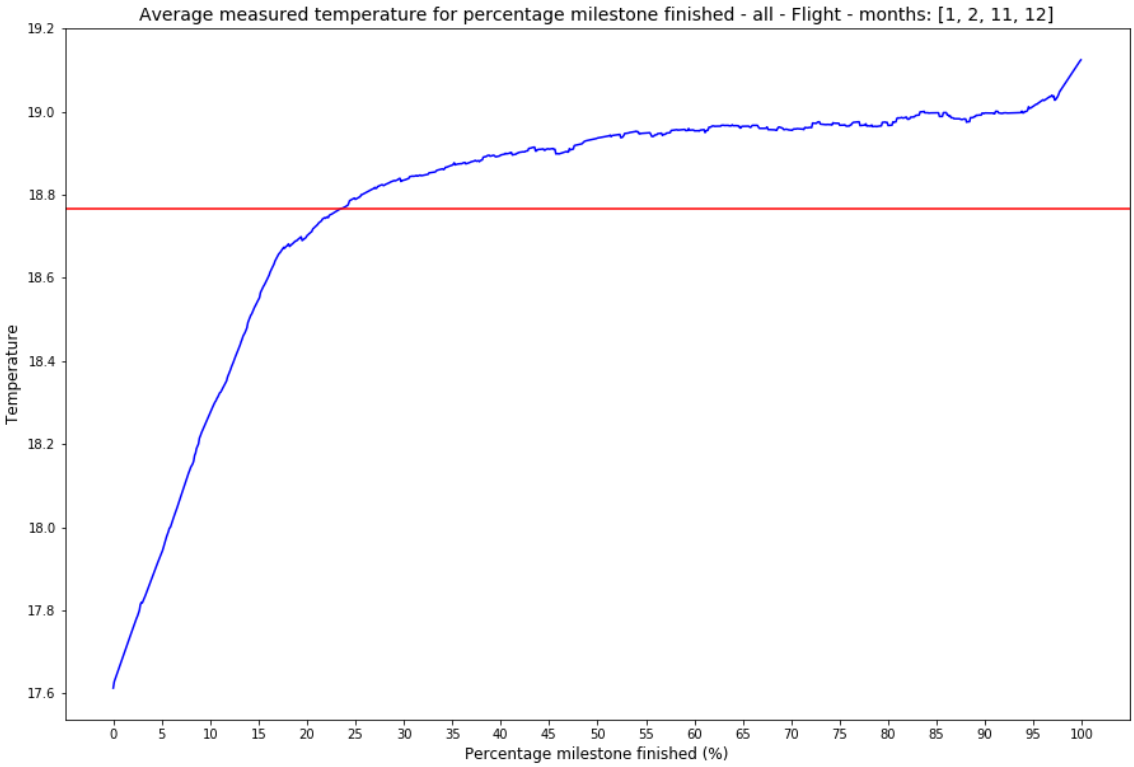


Figure A3.3: Average temperature profile during milestone 'Flight' during the cold months

Milestone m9 'Cargo stored in temperature controlled area (HSV)'

The second milestone that is visualized is 'Cargo stored in temperature controlled area (HSV)'. When the airplane arrives in HSV, it is first offloaded and then the cargo is stored in the temperature controlled warehouse in HSV (m9). Half of all data belongs to this milestone, 1158348 sensor measurements by 4734 devices. Figure A3.4 shows the distribution of the number of sensor measurements over all devices that measured this milestone. The mean number of instances per device is 244, median is 118, minimum is 1, and the maximum is 25185. As figure A3.4 indicates, most sensor devices have less than 1000 measurements and a handful of devices have a lot of measurements. After checking one of the devices, it was found that a shipment registered measurements every 15 seconds instead of every 15 minutes.

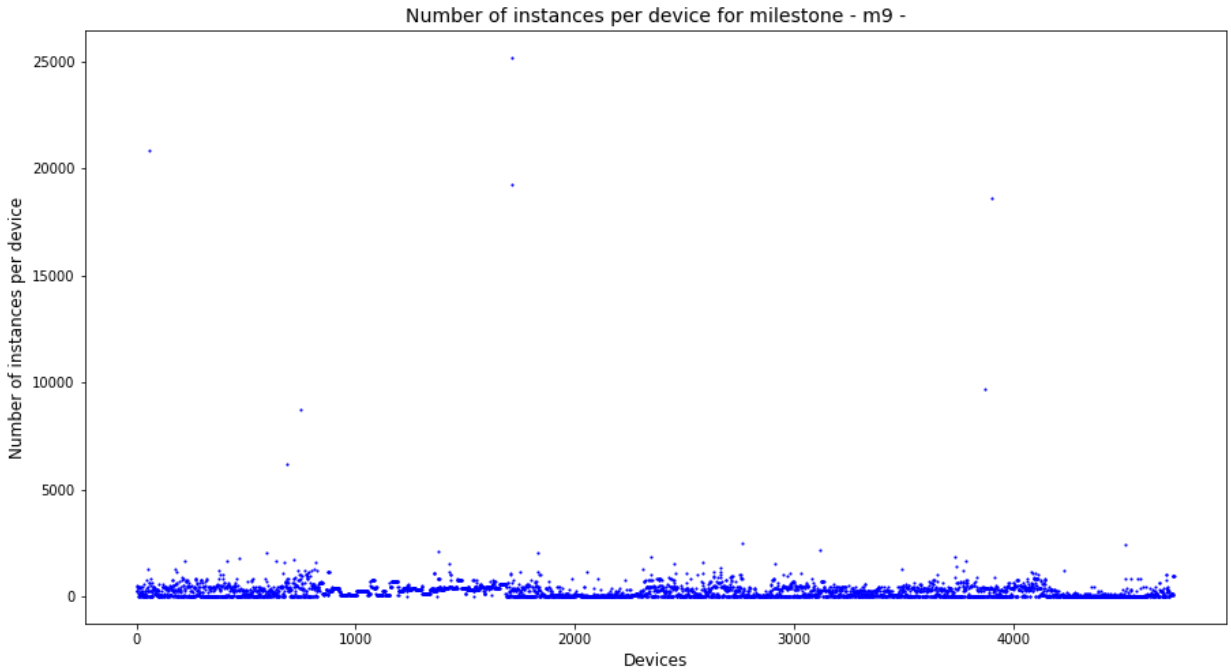


Figure A3.4: Average temperature profile during milestone m9 during the cold months

When only devices with up to 1000 measurements are included in the graph (figure A.3.5), the distribution looks much different. Many devices have less than 20 measurements, but overall the number of measurements varies a lot. This is not strange, since cargo can be stored for different durations. However, this makes it harder to create an average temperature profile. For instance, using 20 slices for both devices with 20 measurements and 300 measurements does not result in an accurate average profile. Therefore, the data is split in several subsets with a varying number of measurements. These subsets are devices with <20, 50-100, 120-180, 230-280, and 330-480 measurements.

For every subset and for both the cold and warm period, an average temperature profile is created. All graphs show the same average temperature profile. Therefore, in figures A.3.6 and A.3.7, the profile including all devices is shown for respectively the warm and the cold period. In the warm period the temperature first decreases at the beginning of the milestone, and then stabilizes about half a degree above the average temperature (red line). In the cold period, the temperature first increases at the beginning of the milestone and then stabilizes towards the average temperature. It is found that when all devices with only one instance are excluded from the graph, the curves stabilize much closer to the average temperature. The temperature decrease can be explained by high temperatures and the temperature increase can be explained by the low temperatures that occurred during the cargo offloading which happens before the cargo is stored in cold storage (m9). The cargo temperature seems to stabilize again during storage.

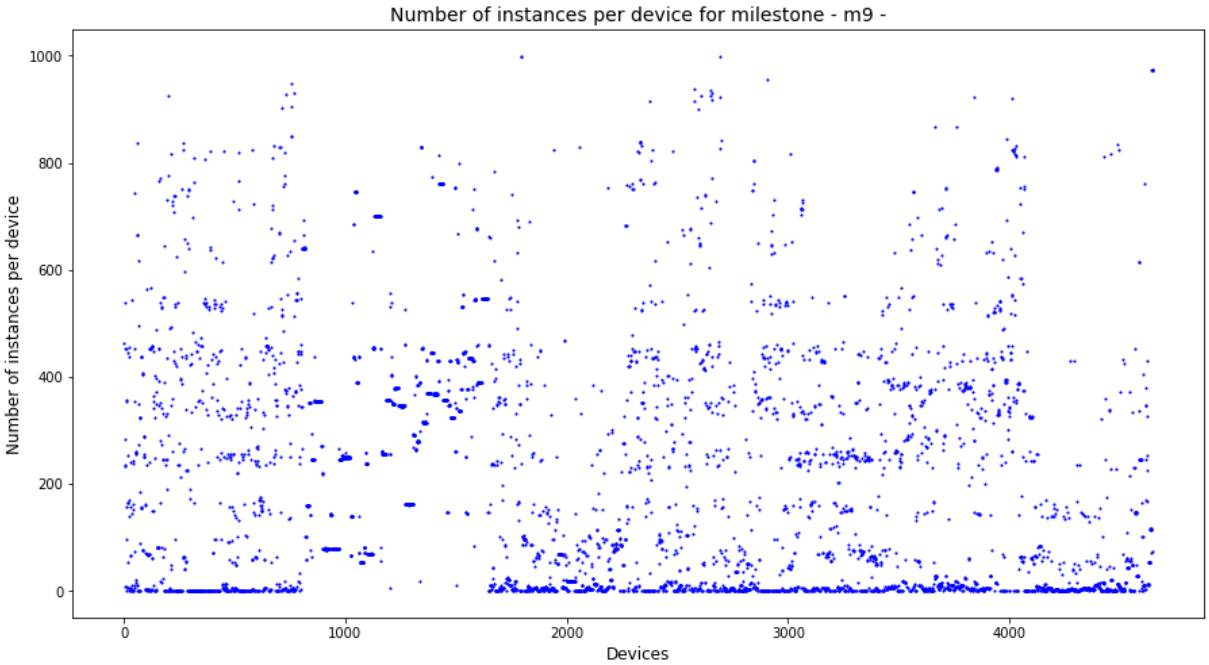


Figure A3.5: Distribution m9

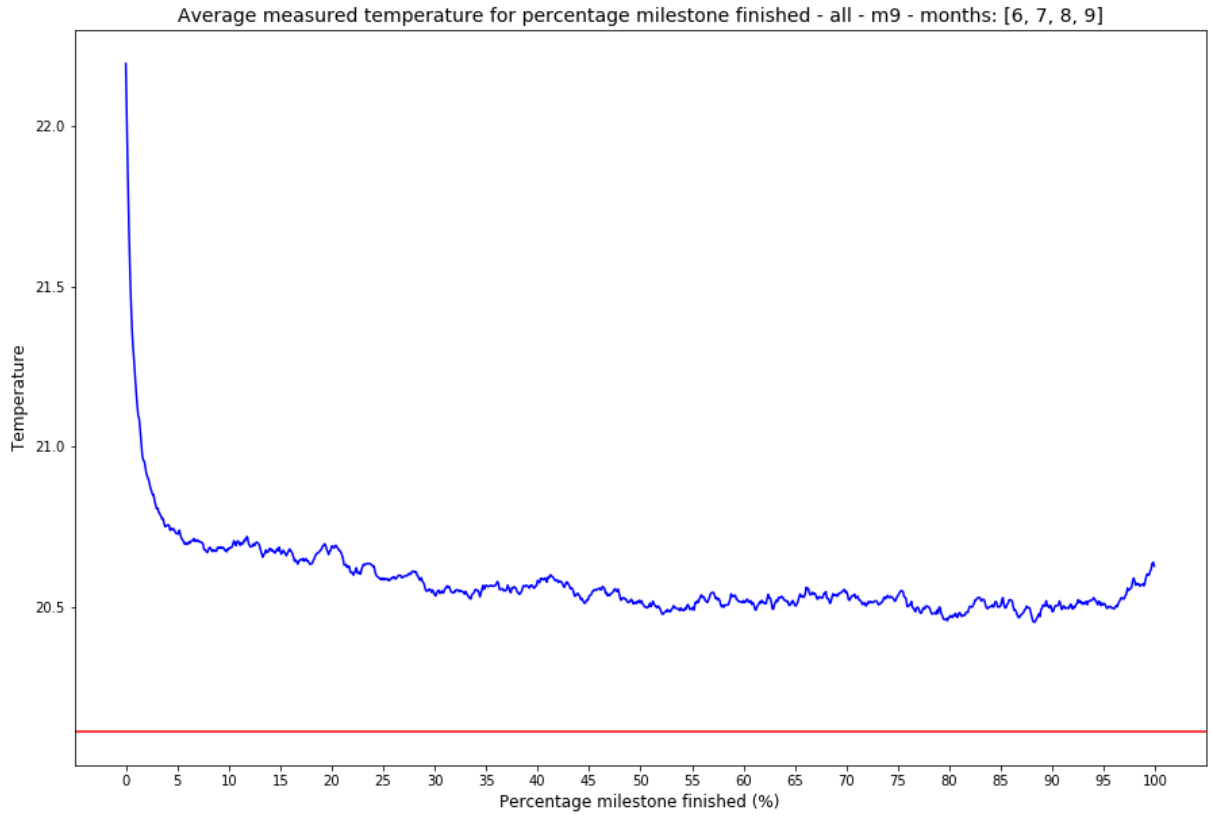


Figure A3.6: Average temperature profile during milestone m9 during the warm months

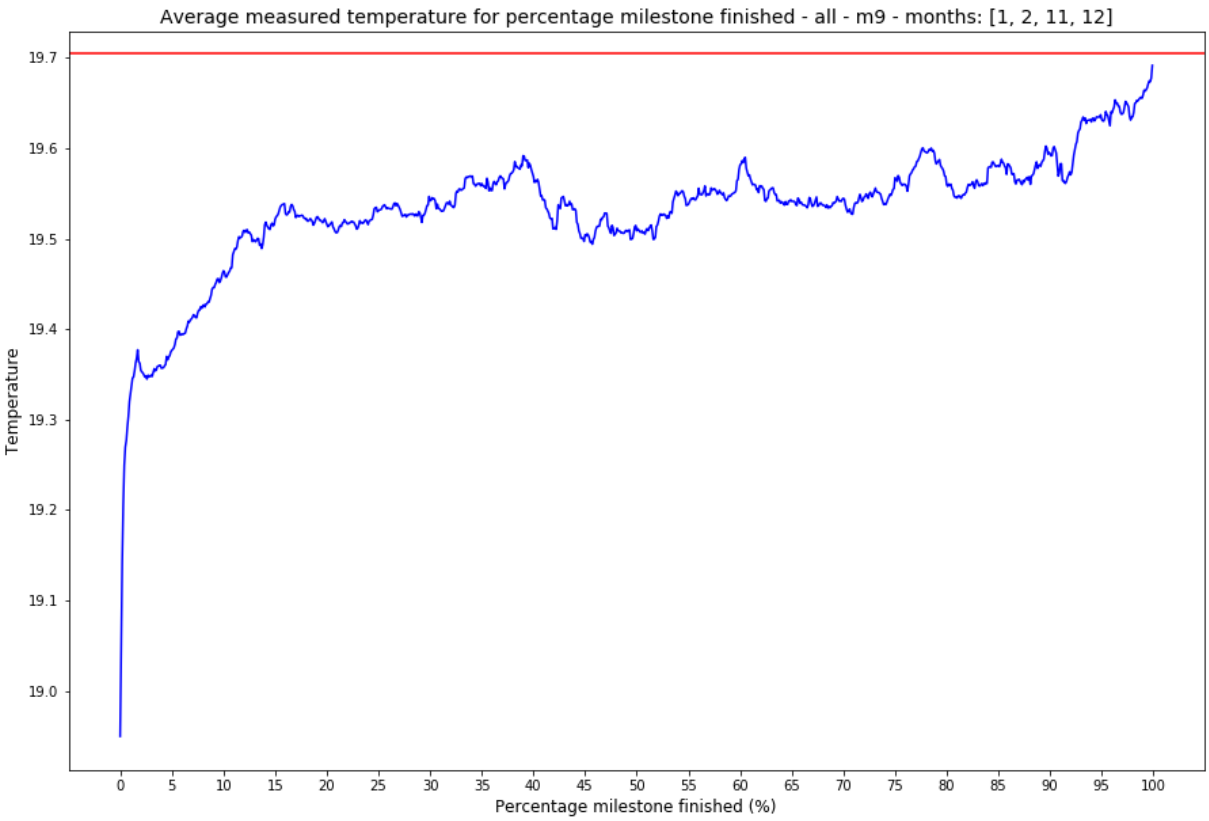


Figure A3.7: Average temperature profile during milestone m9 during the cold months

Milestone m1 'Truck opened and start unloading (LUX)'.

Milestone 'Truck opened and start unloading (LUX)' is the first activity of a shipment from LUX to HSV. Off all sensor instances, 22043 belong to this milestone measured by 2159 devices. The number of devices is lower because many sensors are put on the cargo once the cargo is offloaded, and therefore many devices start measuring the temperature from milestone m4 'Cargo stored in temperature controlled area (LUX)' onwards. The mean, median, minimum and maximum number of instances are respectively, 10, 4, 1, and 642. This, complemented with the distribution in figure A.3.8, indicates that the truck offloading is short activity, which makes sense. There are 262 devices that have over 10 measurements for the milestone, but this would mean that offloading the truck takes over 2,5 hours which is not realistic. Therefore, devices with over 10 measurements will be removed from the dataset after visualization.

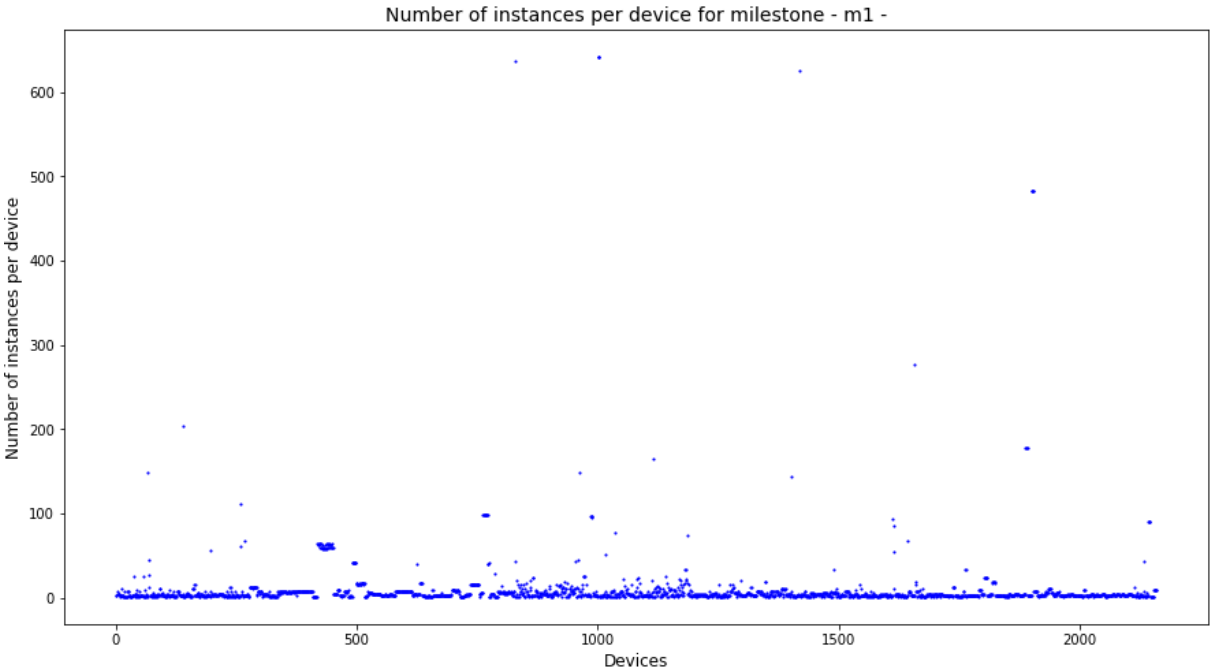


Figure A3.8: Distribution m1

A distribution table including only the devices with up to 20 measurements shows that most devices have between 1 and 7 measurements. A temperature profile is created including all devices. The temperature profiles are shown in figures A.3.9 and A3.10 for respectively the warm and cold periods. The temperature increases when the truck offloading starts in the cold period, and decreases during offloading in the warm period. In the warm period, the cargo in the truck seems to be warmer than the warehouse temperature, and therefore the cargo cools down once being stored in the warehouse. In the cold period, this seems to be the other way around.

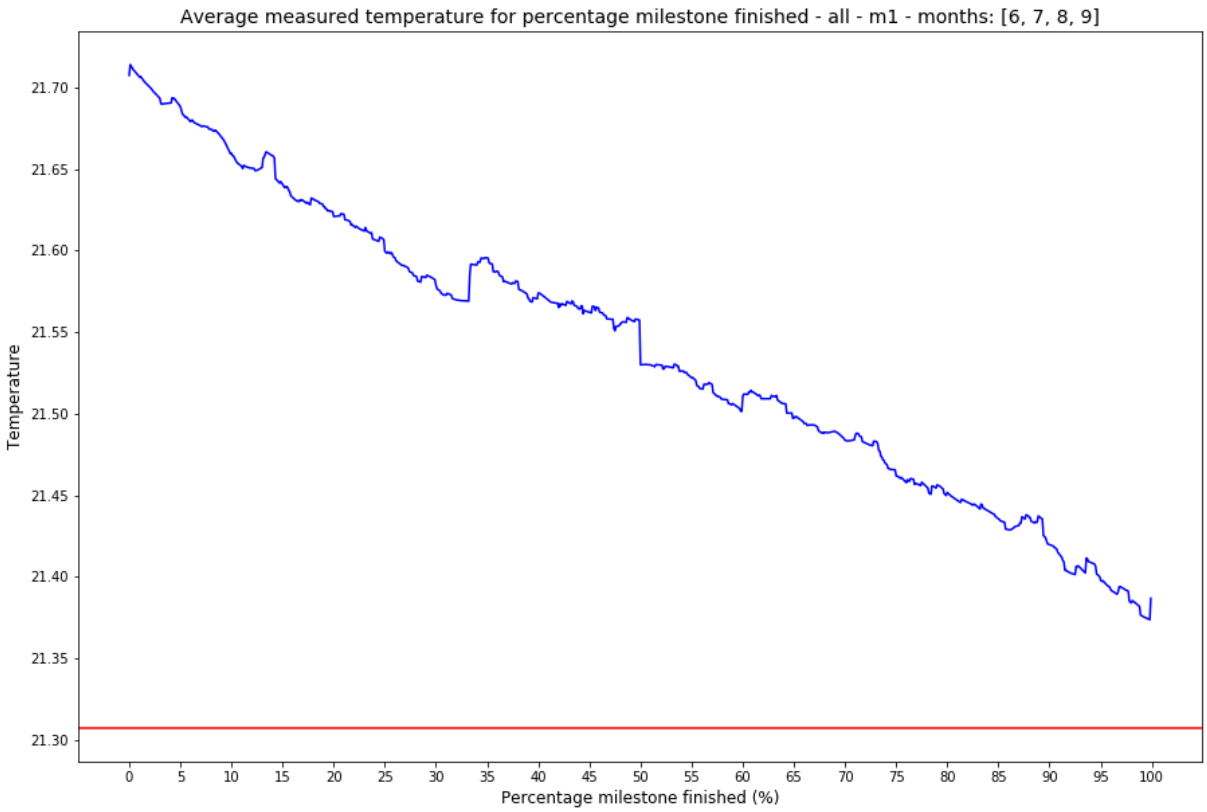


Figure A3.9: Average temperature profile during milestone m1 during the warm months

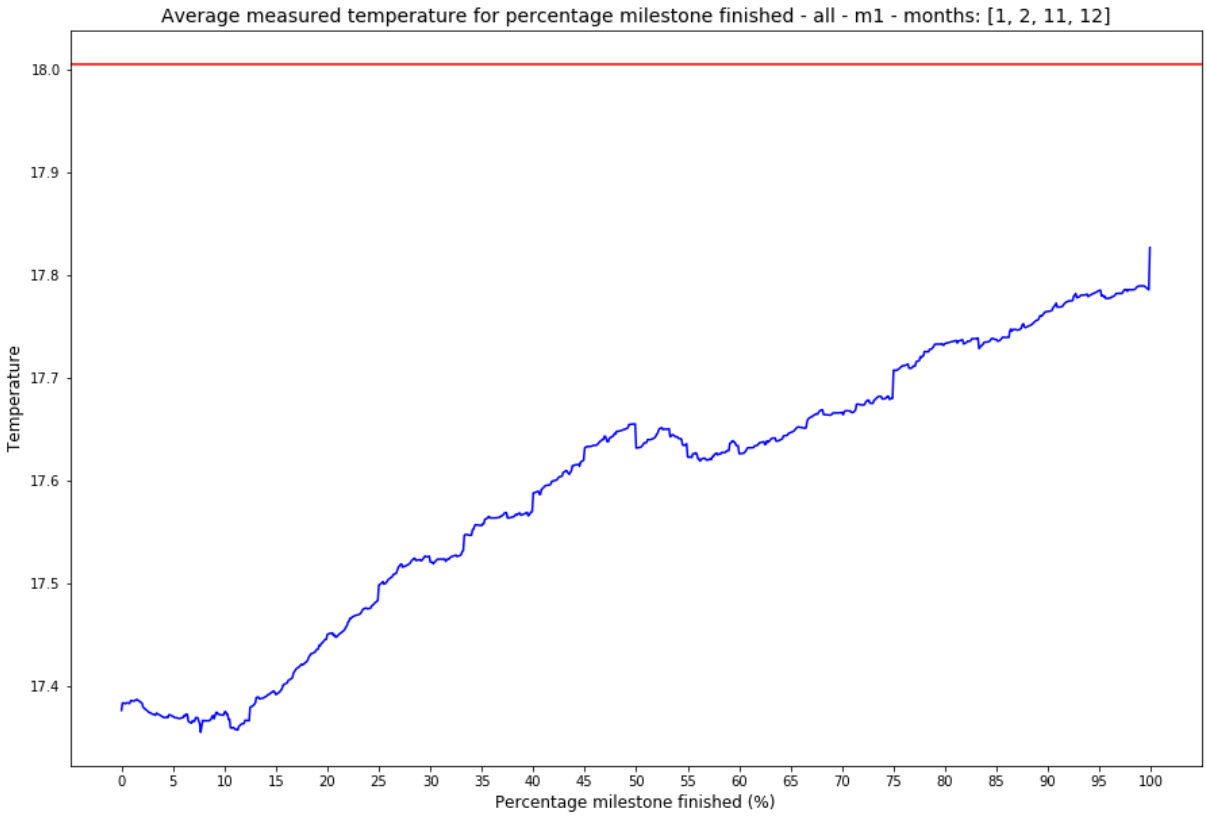


Figure A3.10: Average temperature profile during milestone 'Flight' during the cold months

Milestone m4 'Cargo stored in temperature controlled area (LUX)'.

Once the truck is offloaded, the cargo is stored in a temperature controlled warehouse in LUX. This milestone has 314911 instances generated by 3875 sensor devices. The mean, median, minimum and maximum number of instances are respectively, 81, 34, 1, and 4214. The distribution is shown in figure A.3.11, and shows that most devices have up to 100 measurements (2941 devices). A graph of the distribution including only devices with up to 100 measurements shows that the data is quite evenly distributed, although the majority of devices have less than 50 measurements. Again two graphs are created and shown in figure A.3.12 and A.3.13, one for the warm period and one for the cold period. The graph of the cold period indicates that the temperature first decreases towards the average measured temperature in the warm period, which logically follows the truck offloading since the cargo is still cooling down. The temperature increase at the start of the milestone logically follows the end of milestone m1. After the truck offloading the temperature of the cargo stabilizes around 20 degrees Celsius. At the end of the milestone, a small increase in temperature is visible. The graph of the cold period shows again a mirrored temperature profile, since the temperature first increases towards the average, and decreases at the end of the milestone. At the end of m4, when m5 ´ULD build-up and stored in temperature controlled area (LUX)' is about to start, the temperature increases slightly in the warm period, and decreases in the cold period. This might indicate that the ULD build-up, during which cargo is more susceptible for excursions, actually started slightly earlier, though the increases and decreases are small.

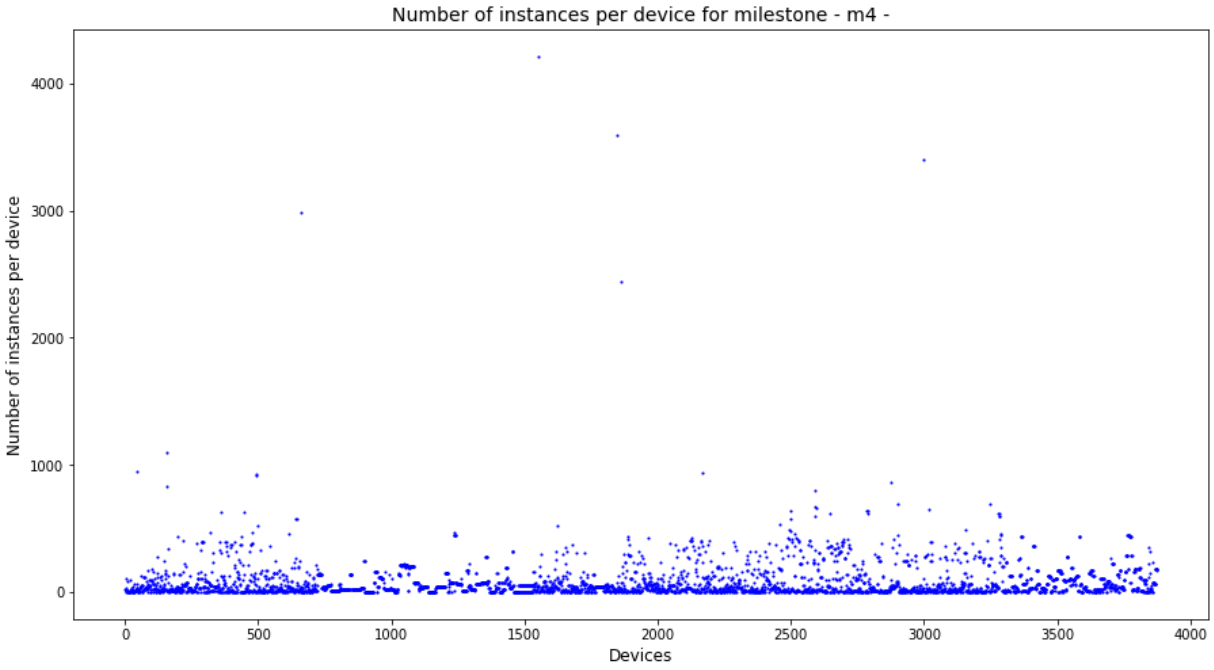


Figure A3.11: Distribution m4

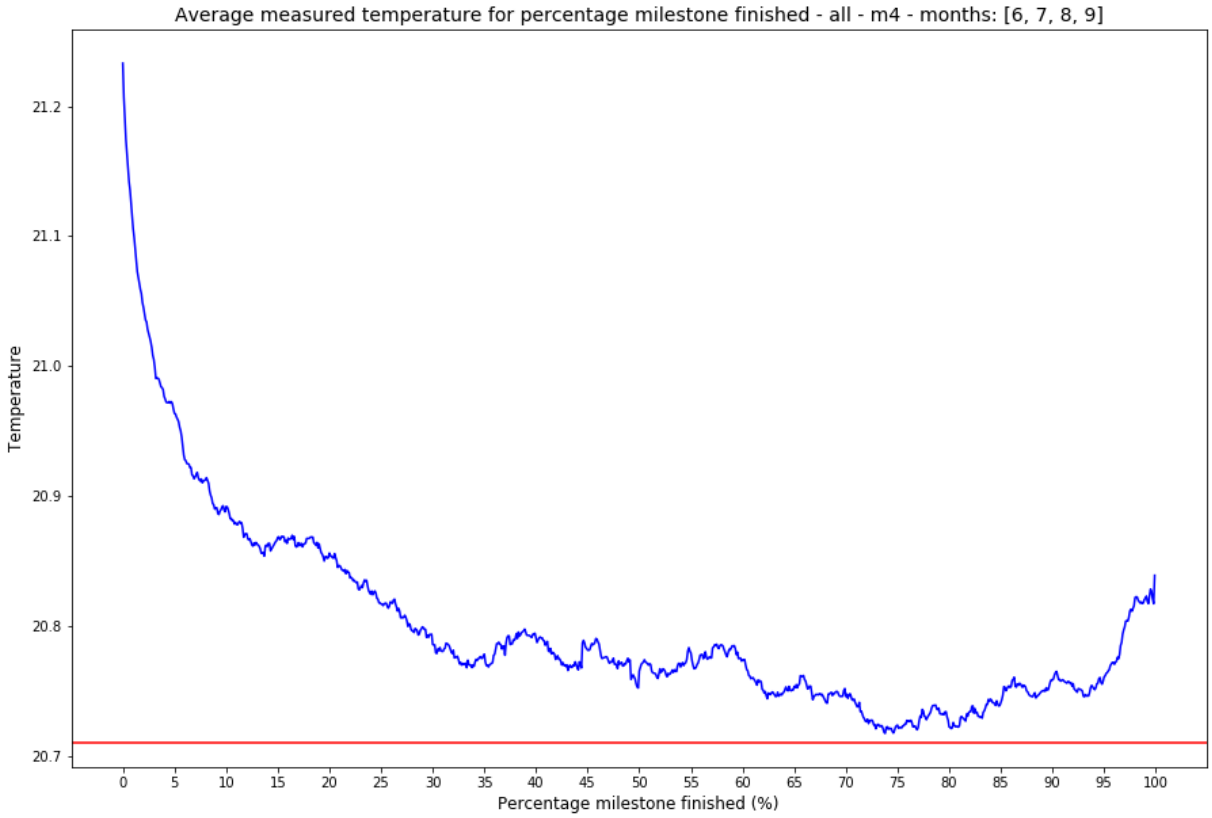


Figure A3.12: Average temperature profile during milestone m4 during the warm months

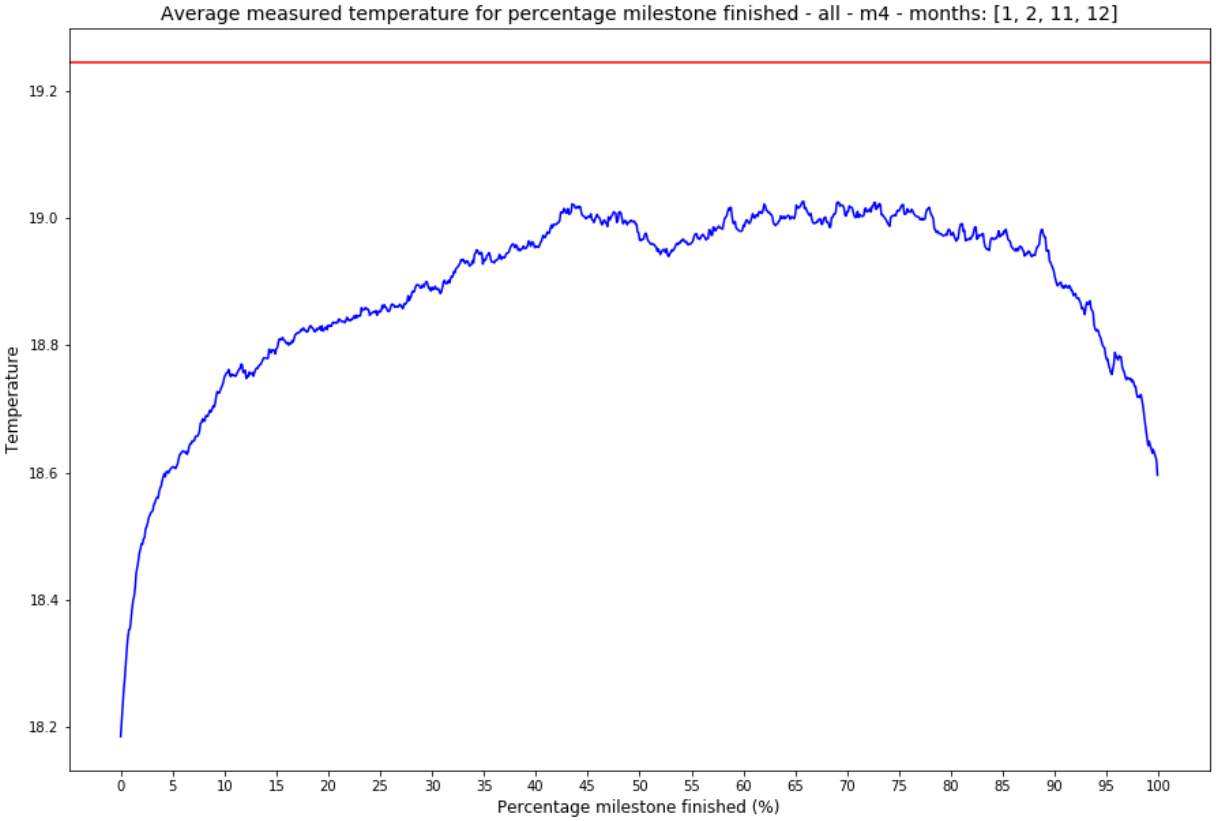


Figure A3.13: Average temperature profile during milestone 'Flight' during the cold months

Milestone m5 'Cargo build-up on ULD and stored in temperature controlled area (LUX)'.

After the cold storage of the cargo, the cargo is taken out of cold storage and the ULD build-up begins. After the ULD build-up, the cargo is again stored in a temperature controlled area. The time when the ULD is put back in temperature controlled storage is not known. Since product group 15-25°C forms the majority of all cargo transported, the whole warehouse is kept in this temperature range which means that the temperature during ULD build-up should not change. However, it is not known whether or not the temperature in the warehouse is really kept between 15 and 25°C. It is possible that during cold winters or warm summers it is harder to retain the temperature in this range.

For this milestone, 283270 measurements are registered by 5293 devices. The mean, median, minimum and maximum number of instances are respectively, 53, 41, 1, and 6916. Only one device measured more than 500 times the temperature, namely 6916 times. Only 479 devices have more than 100 measurements. Devices up to 300 measurements are shown in the distribution graph in figure A3.14.

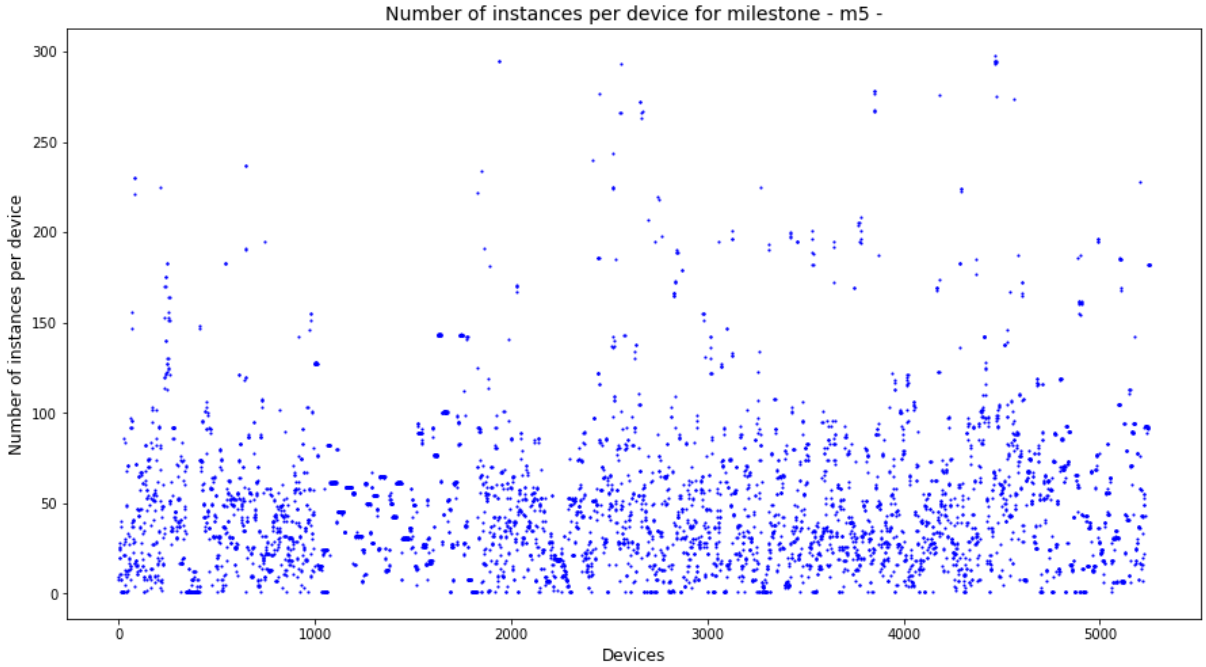


Figure A3.14: Distribution m5

The data is split in several subsets: devices with 0-10, devices with 10-30, devices with 30-60 instances, and devices with 0-100 measurements. For each subset and for both the warm and cold periods, a graph with the average temperature profile is created. The various subsets all show the same temperature graph, and therefore only the graph with all sensor devices is shown in figures A3.15 and A3.16 for respectively the warm and cold period. Both graphs are again mirroring each other. In the warm period, the temperature drops slightly at the beginning of the milestone, and increases slightly at the end of the milestone. In the warm period, the temperature increases, then stabilizes, and decreases at the very end of the milestone. Milestone m6 'ULD out of cell for uplift to flight (LUX)' starts at the end of the milestone which might explain the increase in the warm period and the decrease in the cold period.

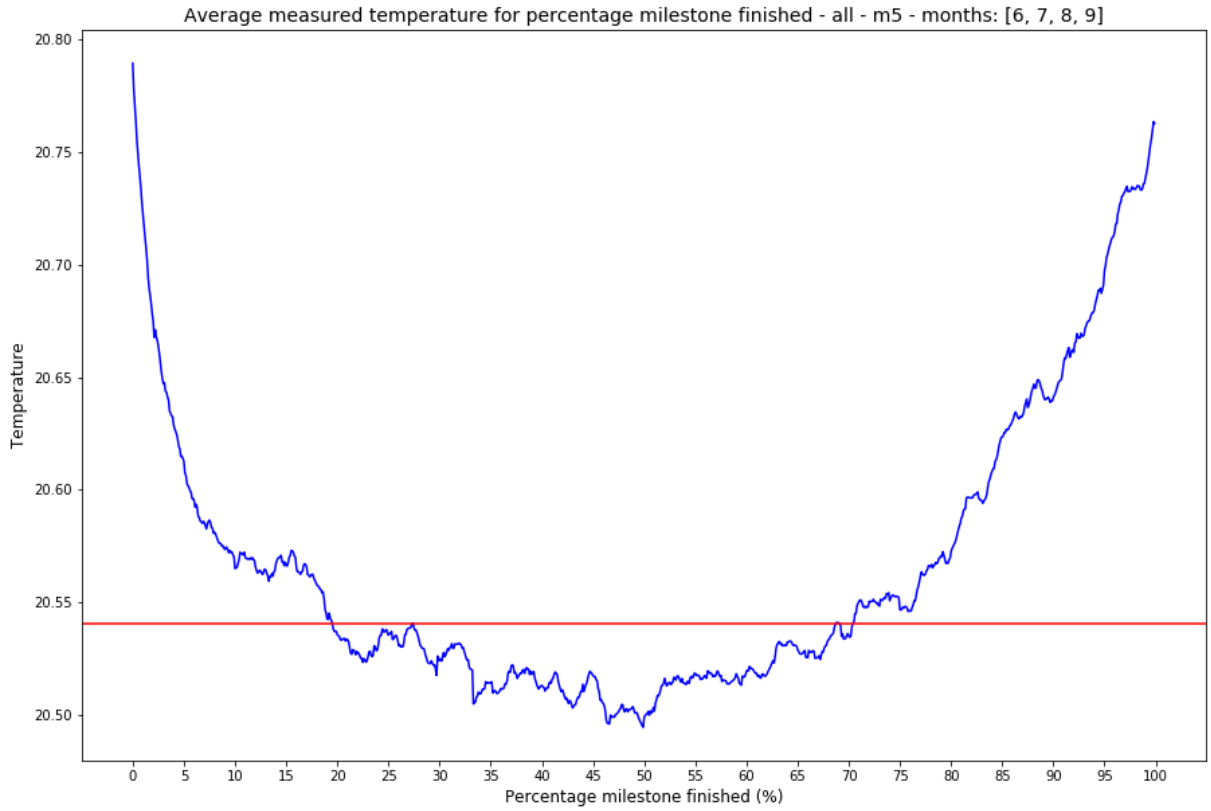


Figure A3.15: Average temperature profile during milestone m5 during the warm months

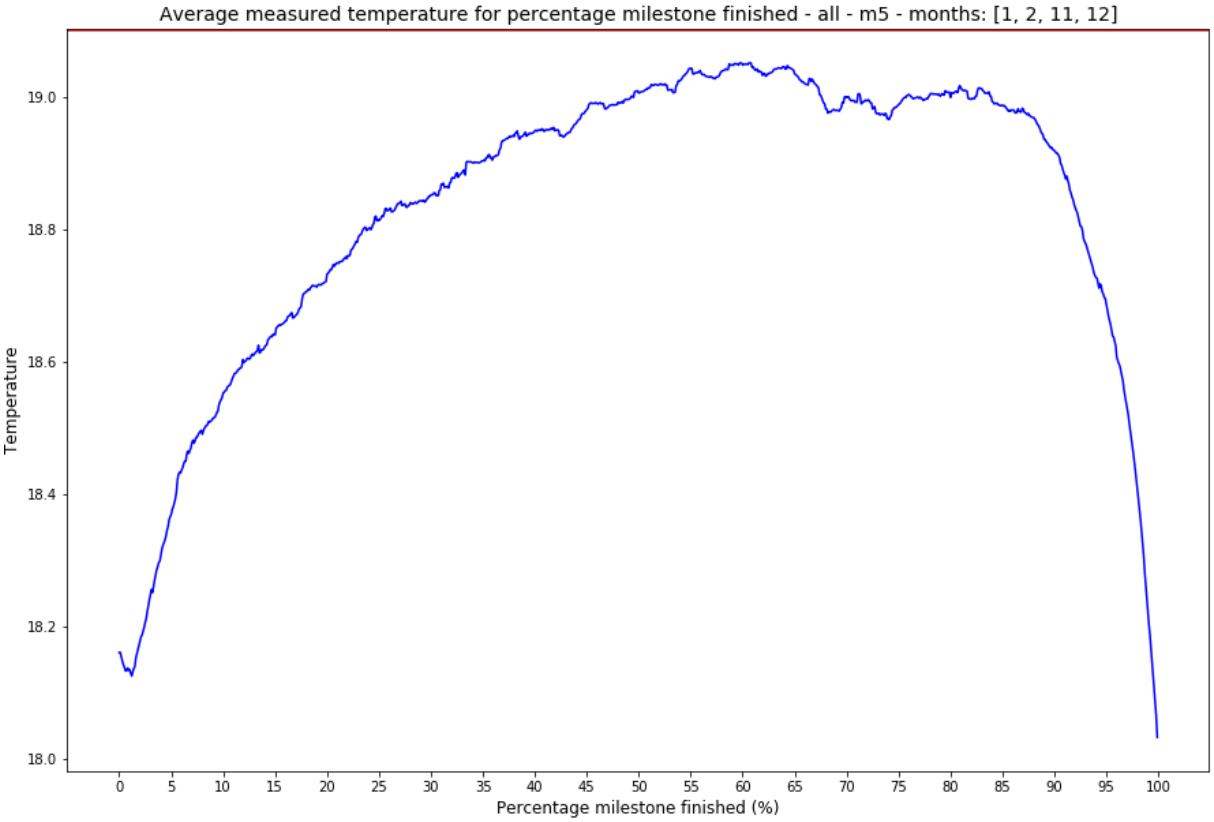


Figure A3.16: Average temperature profile during milestone m5 during the cold months

Milestone m6 'ULD out of cell for uplift to flight (LUX)'.

During this milestone, the ULD is taken out of the temperature controlled area and is loaded into the aircraft. This means that the cargo is exposed to the outside temperature while it is on the tarmac. A total number of 36938 measurements are made by 5302 sensor devices. The mean, median, minimum and maximum number of instances are respectively, 6, 6, 1, and 680. Of the 5302 devices, 4966 devices have less than 10 measurements. By including only devices with up to 25 measurements, a distribution graph is created is shown in figure A3.17. From the graph it can be concluded that the aircraft loading generally takes about 1,5 hours.

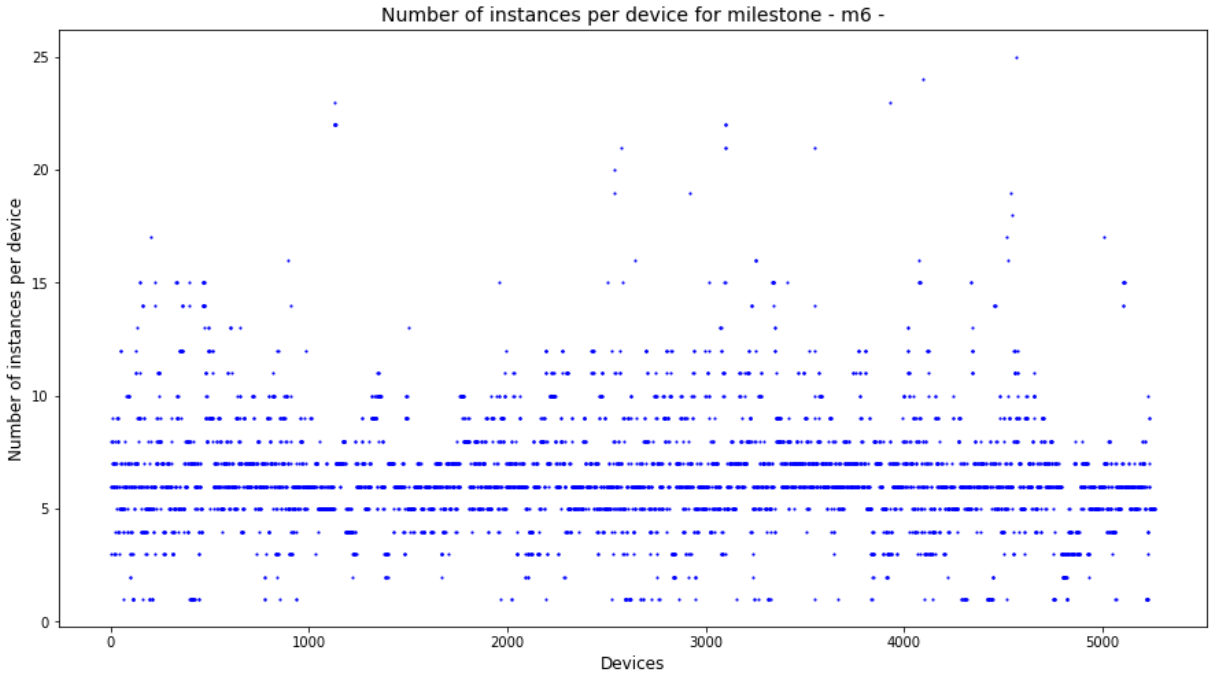


Figure A3.17: Distribution m6

The average temperature graphs showing the average temperature profile are shown in figure A3.18 and A.3.19 for the warm and cold period. As depicted from the graph of the cold period, the temperature decreases when the ULD is taken out of the temperature controlled stored, and stabilizes towards the average temperature while loaded into the aircraft. In the aircraft itself, the cargo area is temperature controlled. In the warm period, the temperature increases due to the tarmac temperature exposure, and decreases once a part of the cargo is the aircraft.

All devices with more than 12 measurements indicating that the aircraft loading takes 3 hours, are deleted from the dataset since such a duration is not realistic.

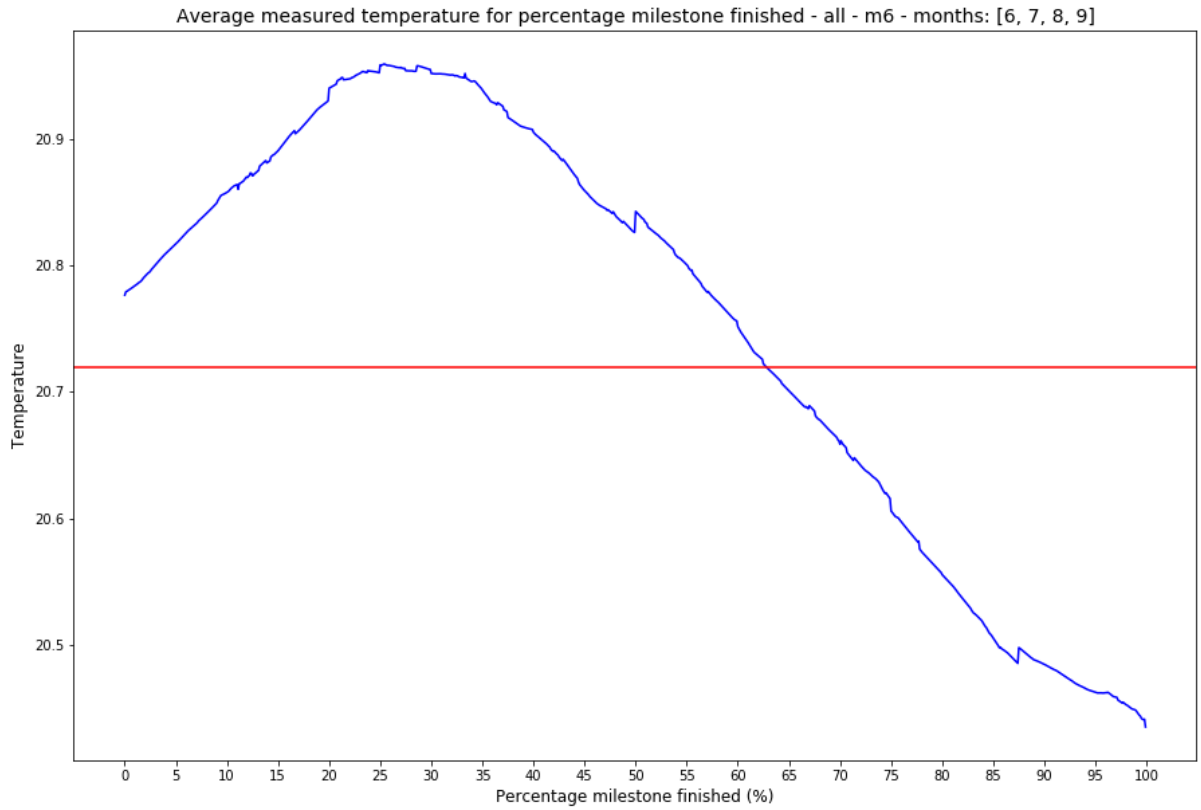


Figure A3.18: Average temperature profile during milestone m6 during the warm months

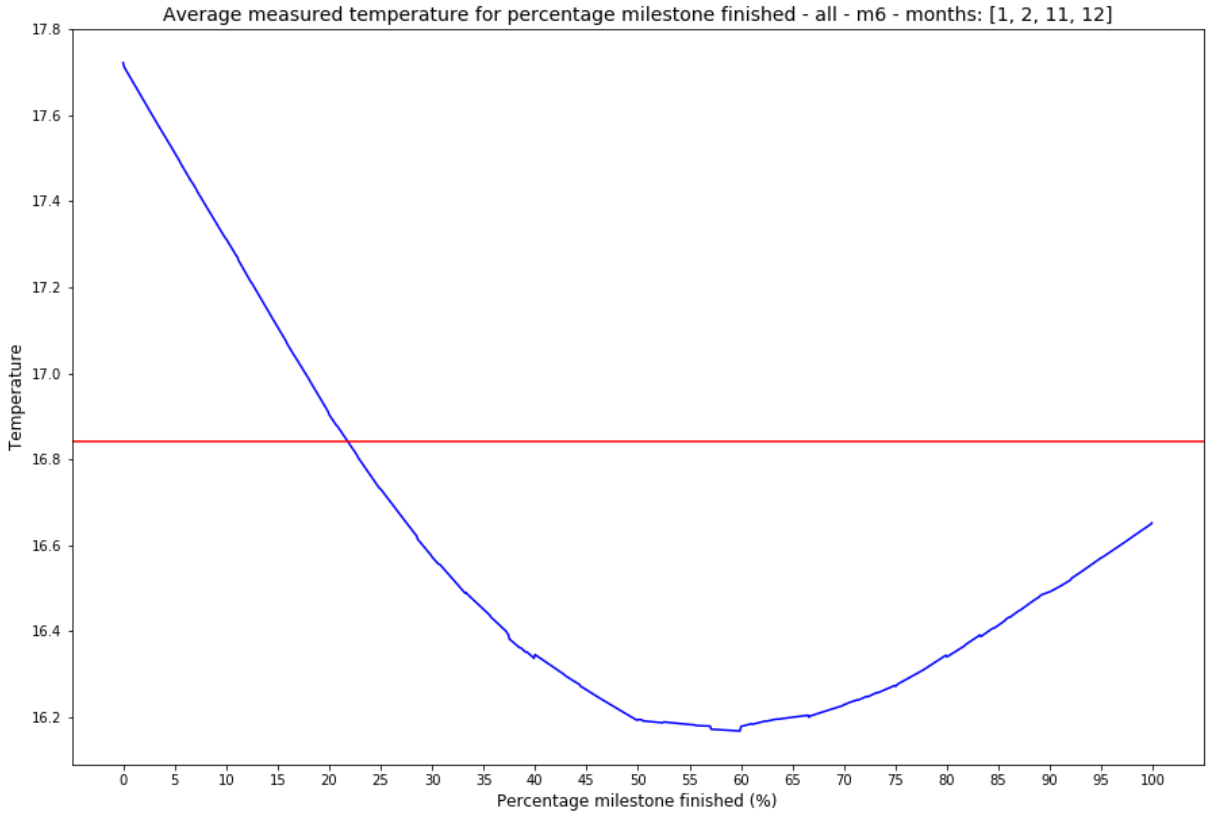


Figure A3.19: Average temperature profile during milestone m6 during the cold months

Milestone 'Aircraft offloading (HSV)'.

The aircraft's cargo is unloaded once the airplane arrives in Huntsville. Panalpina only registered this milestone with 15 devices and 102 sensor measurements. Therefore, a milestone 'Aircraft offloading (HSV)' was added in between the aircraft arrival (m8) in Huntsville and the cargo storage in a temperature controlled area in Huntsville (m9). This resulted in 46578 measurements by 5330 sensor devices. For this milestone, it is assumed that the aircraft is immediately offloaded when arriving in Huntsville. The mean, median, minimum and maximum number of instances are respectively, 8, 7, 1, and 677. Of the 5330 devices, 5214 devices have up to 20 sensor measurements. Figure A3.20 shows the distribution graph including only these devices.

For every period, a graph showing the average temperature profile is created as shown in figures A3.21 and A.3.22. On average, the temperature slightly decreases at the start of the milestone in the cold period and then increases towards the average temperature measured during offloading. The temperature increases at the end of the milestone. The graph can probably be explained by the fact that during offloading, a part of the cargo is stored in the temperature controlled area earlier than other parts of the cargo. Therefore, when at the end of the milestone almost all cargo is stored, the temperature is still increasing. In the warm period, the temperature increases linearly during offloading, however, the curve slightly curves down at the end of the milestone indicating temperature stabilization.

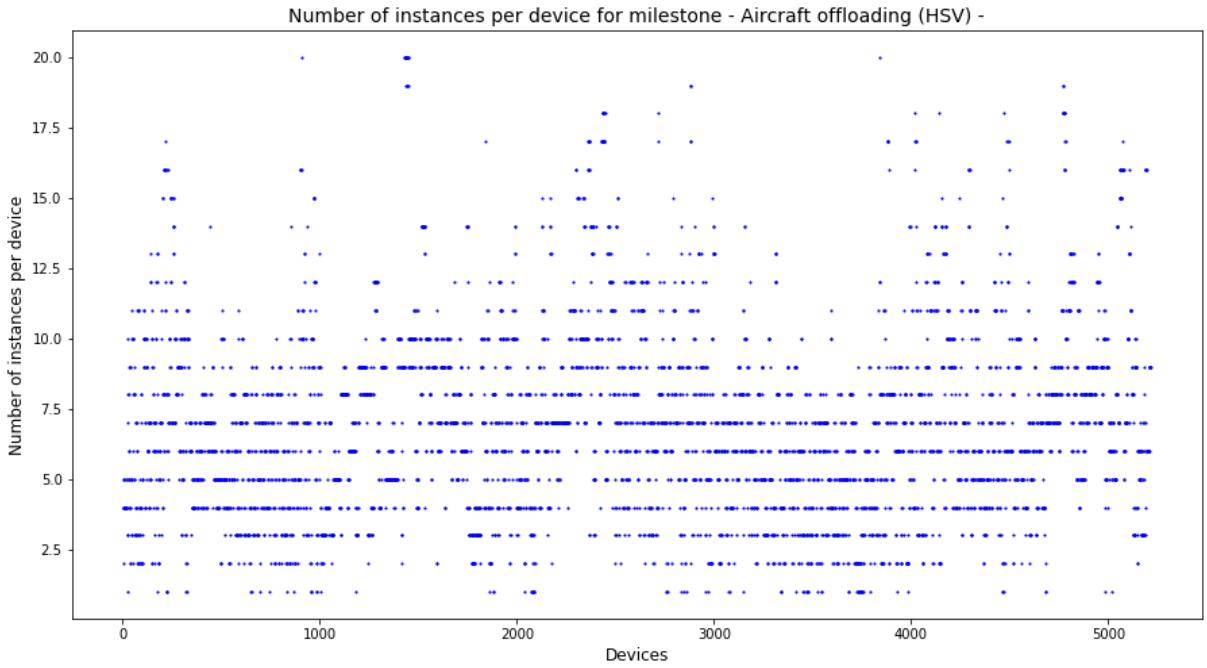


Figure A3.20: Distribution Aircraft Offloading HSV

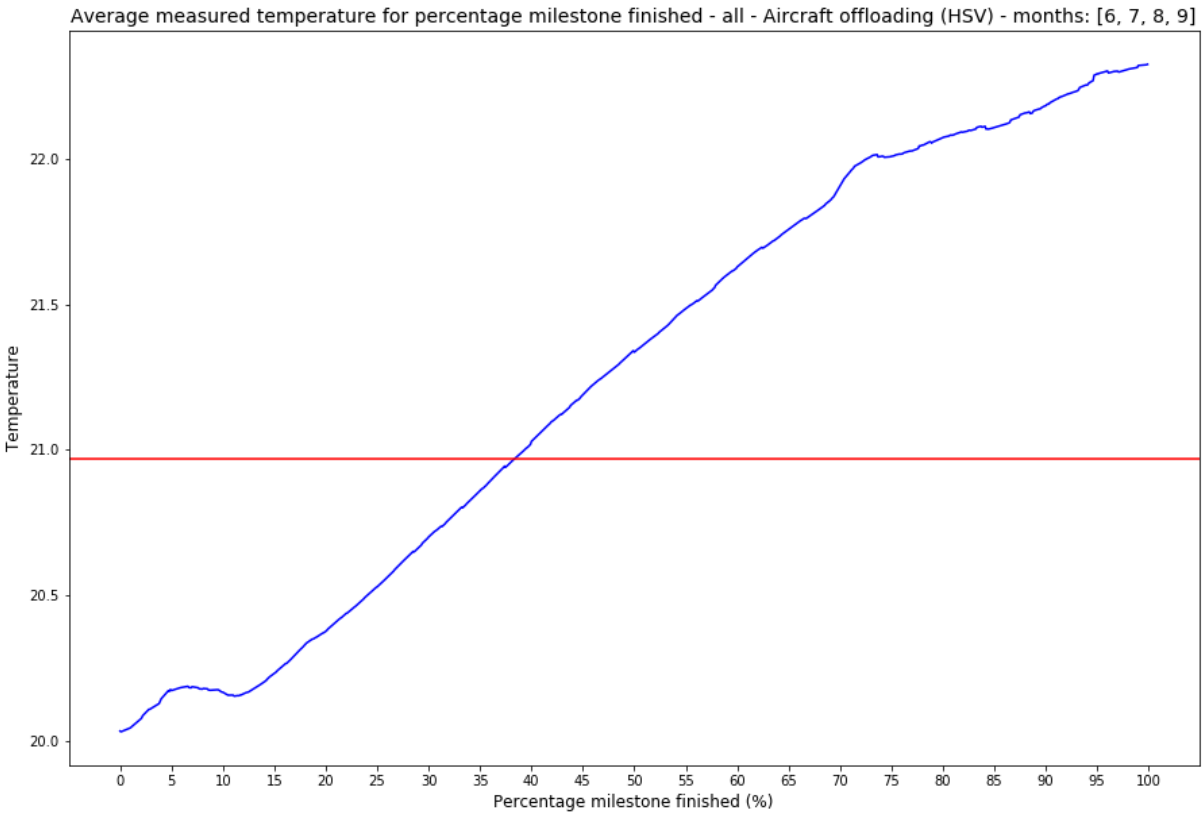


Figure A3.21: Average temperature profile during milestone aircraft offloading HSV during the cold months

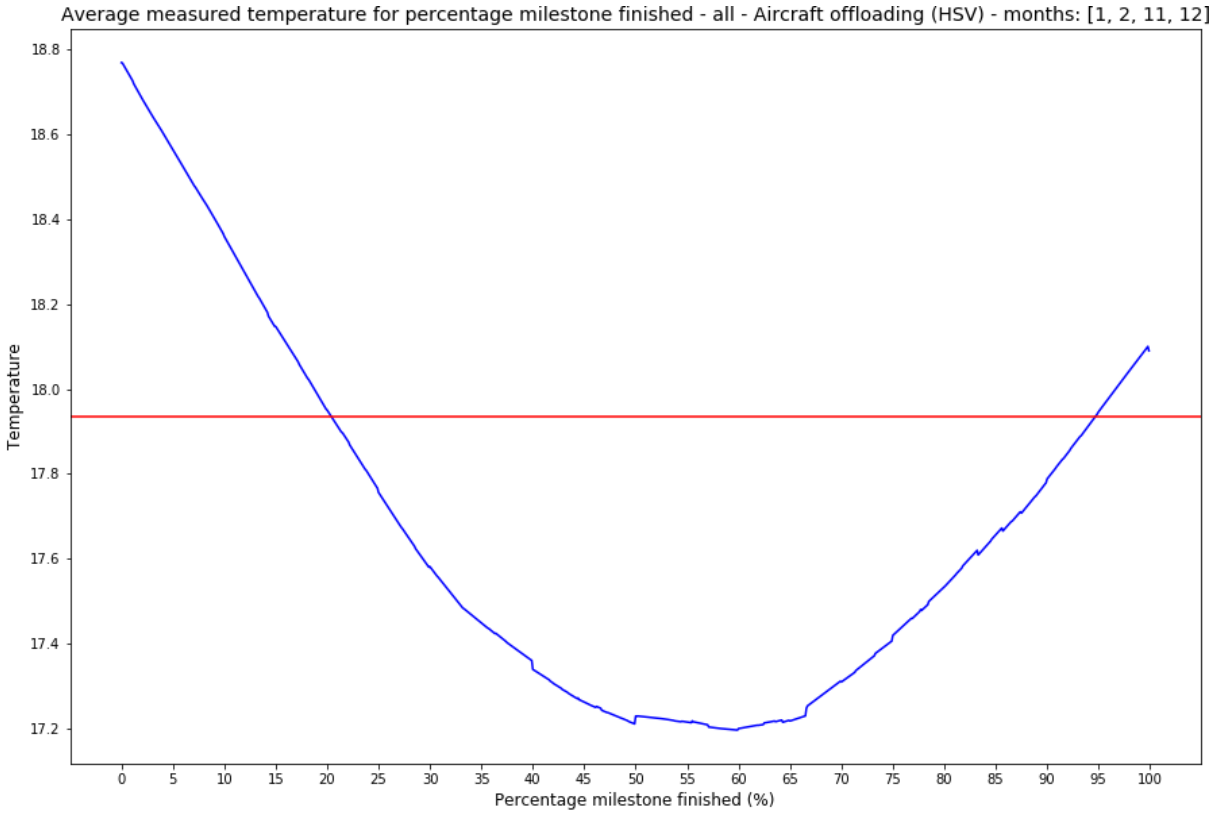
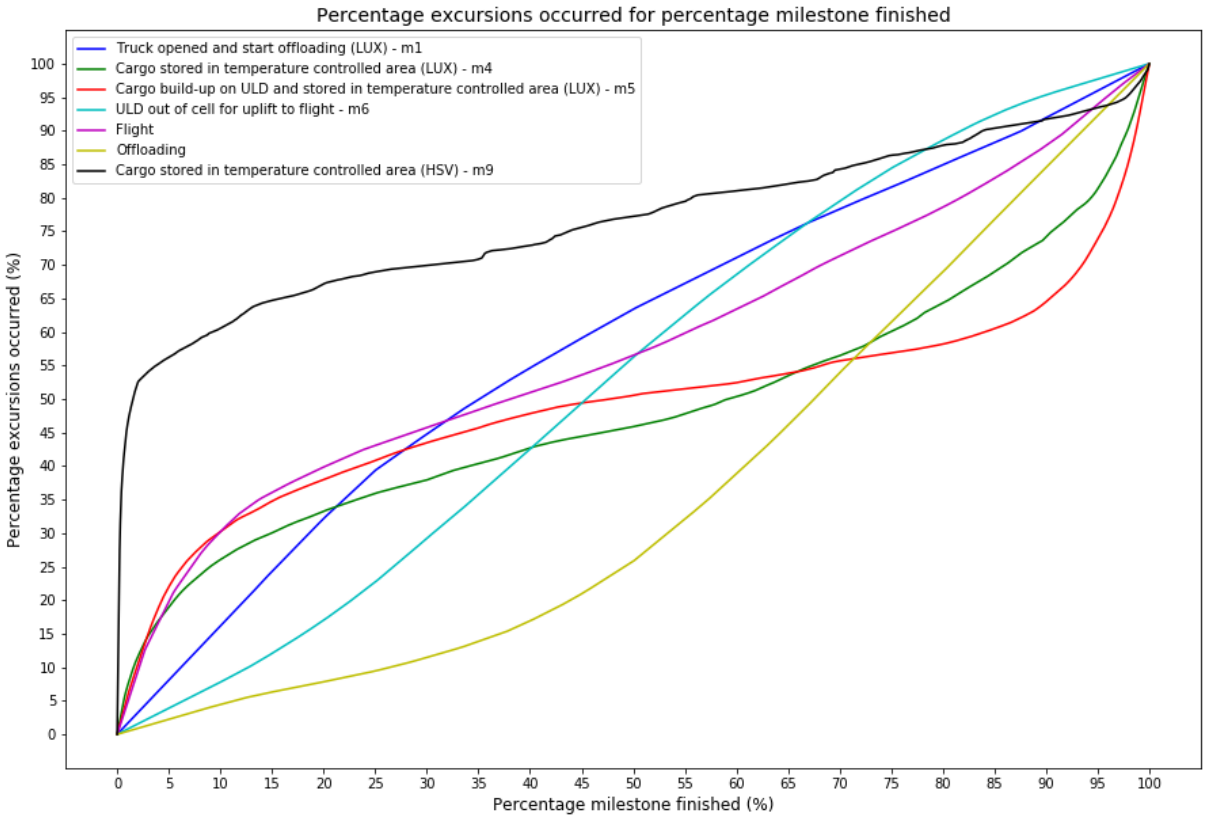


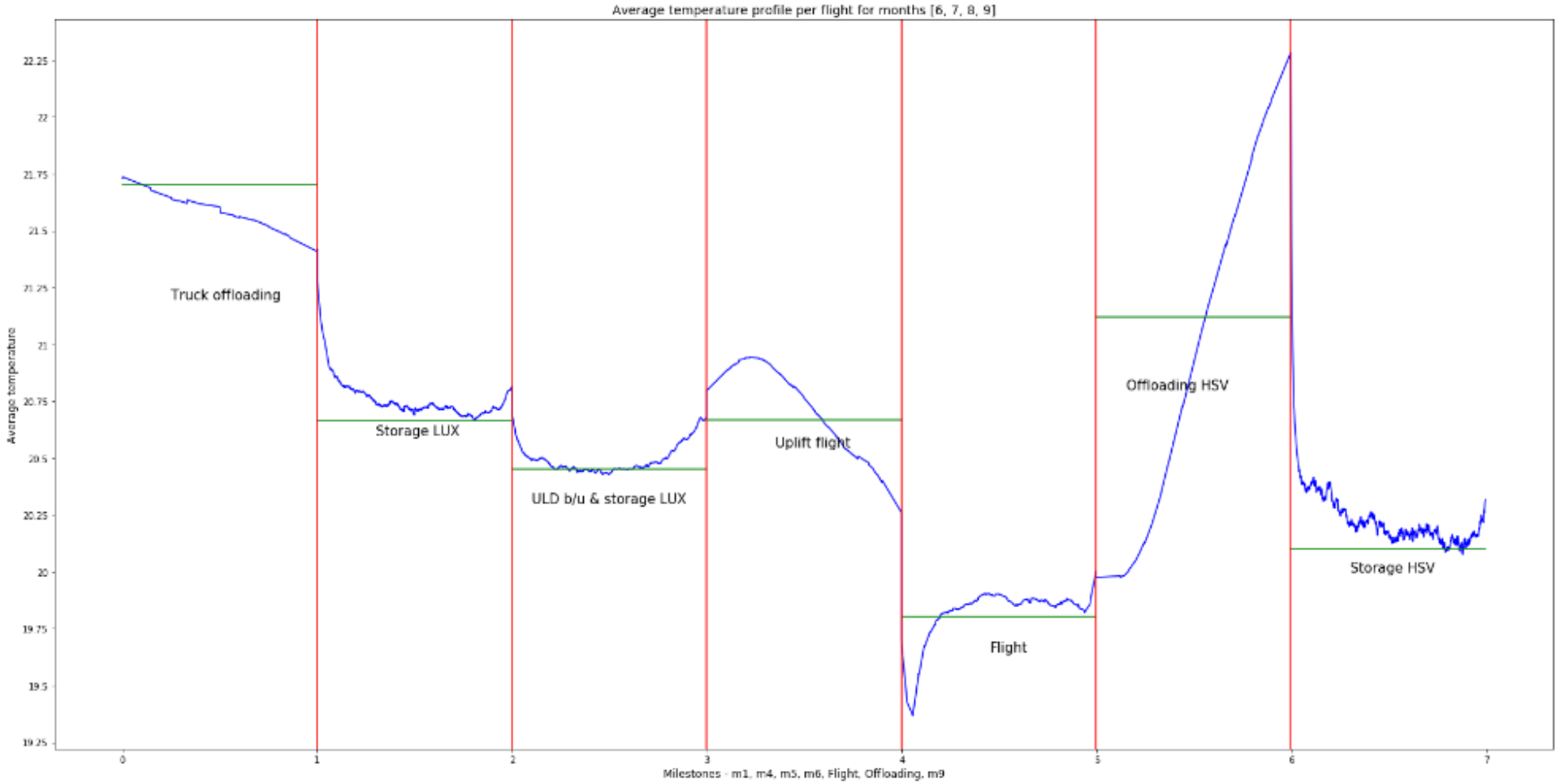
Figure A3.22: Average temperature profile during milestone Aircraft offloading HSV during the cold months

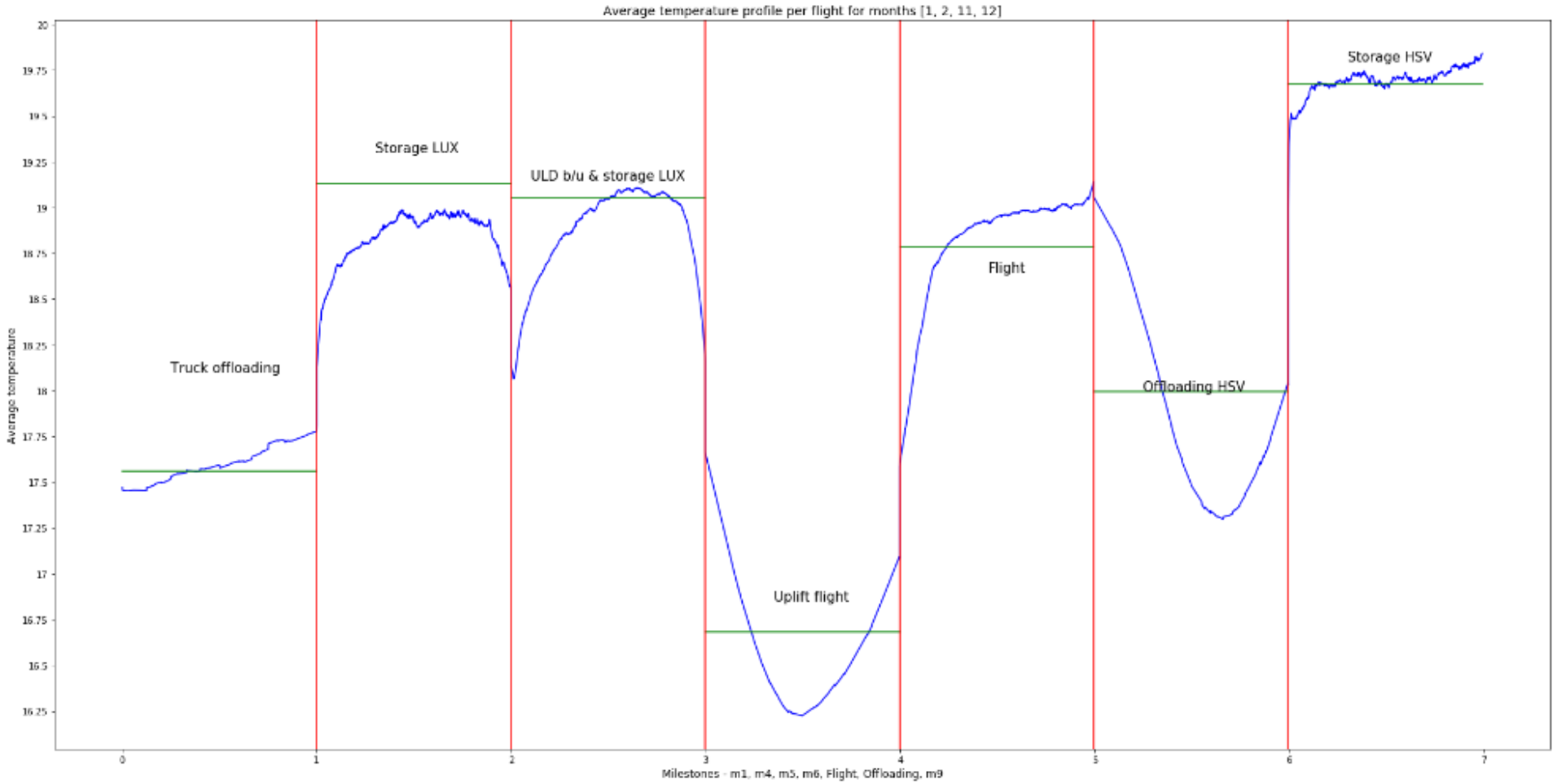
Milestone m3 'Truck loading start (HSV)' and m2 'Truck loading complete (HSV)'.

After the temperature controlled storage in Huntsville (m9), the cargo is loaded into trucks for the trip to their final destination. The sensor devices are removed from the cargo during truck loading, but keep registering temperatures because they are often not turned off. Truck loading is generally a short activity. The mean, median, minimum and maximum number of instances are respectively, 3, 2, 1, and 255. By creating a graph showing the average temperature profile during loading, it is found that the temperature increases slightly, but many truck loadings take only 15 to 30 minutes. Since the sensors are removed from the cargo, the sensor measurements generated during m3 'Truck loading start (HSV)' and m2 'Truck loading complete (HSV)' can be ignored and are removed from the dataset.

Appendix 4: Graphs of visualization with improved datasets




Average temperature profile per flight for months [6, 7, 8, 9]
Average temperature
Milestones - m1, m4, m5, m6, Flight, Offloading, m9
Truck offloading
Storage LUX
ULD b/u & storage LUX
Uplift flight
Flight
Offloading HSV
Storage HSV


Average temperature profile per flight for months [1, 2, 11, 12]
Average temperature
Milestones - m1, m4, m5, m6, Flight, Offloading, m9
Truck offloading
Storage LUX
ULD b/u & storage LUX
Uplift flight
Flight
Offloading HSV
Storage HSV

Appendix 5: Milestone specific data reduction

M4 'Cargo stored in temperature controlled area (LUX)'

The graph in figure A.5.1 shows the different graphs of milestone m4 created for devices with a number of measurements in the ranges: 0-250, 5-10, 10-15, 15-20, 20-30, 30-50, 50-70, 70-100, 100-150, 150-250. The range 0-250 includes all devices for this milestone. The brown, light green and pink lines in figure A.5.1 indicate that when the duration of milestone m4 is relatively long, it becomes more obvious that most excursions occur at the beginning of the milestone and at the end of the milestone. At the beginning and at the end of the milestone the slope of the curve is very steep, indicating that at these points a lot of excursions occur, while in between the start and end the slope is almost zero, indicating that no excursions occur. The curves of devices with up to 30 measurements (dark green, red, light blue and purple) show a much more constant slope, indicating that the occurrence of excursions is spread over the complete duration of the milestone. The data of devices with up to 30 measurements is completely kept. For the devices with more than 30 measurements, specific parts of the data is kept based on the curve for that range. The curve of devices with 30-50 measurements indicates that most excursions occur during the first 40% and last 30% of the data. Therefore, the data from 40-70% will be deleted. The curve belonging to range 50-70 measurements shows that excursions mainly occur during the first 10% and last 20% of the data. The data in between will therefore also be deleted. This process is further continued for devices with 70-250 measurements. Table A.5.1 summarizes which data is retained. The table also shows the total number of devices, the number of devices with excursions, and the average number of excursions of these devices.

M5 'Cargo build-up on ULD and stored in temperature controlled area (LUX)'

Similar to milestone m4, different graphs are created for devices with a various number of measurements. Figure A.5.2 shows the curves for devices with a number of measurements in the ranges: 0-100, 5-10, 10-15, 15-20, 20-30, 30-50, 50-70, and 70-100. The range 0-100 includes again all devices for this milestone and is therefore the same as the previously shown average line for milestone m5 as shown in figure 5.10. The same process as described for milestone m4 is repeated, but this time different parts of the data is retained. Which parts of the data is kept is shown in table A5.2.

M9 'Cargo stored in temperature controlled area (HSV)'

For milestone m9, again the same selection process is repeated. Figure A.5.3 shows the curves of devices with a number of measurements in the following ranges: 0-800, 5-10, 10-15, 15-20, 30-60, 80-100, 150-300, 300-450, and 600-800. The range 0-800 includes again all devices for this milestone and is therefore the same as the previously shown average line for milestone m9 as shown in figure 5.10. Table A.5.3 indicates which parts of the data is retained.

Devices with a number of measurements in range	Devices with excursions	Total number of devices	Average number of excursions of these devices	Part of the data that is kept in the dataset
0-250	365	3375	6	-
5-10	51	412	2	All
10-15	59	369	3	All
15-20	34	262	3	All
20-30	48	441	4	All
30-50	54	531	8	First 40% and last 30%
50-70	43	361	4	First 10% and last 20%
70-100	31	250	5	First 7% and last 10%
100-150	23	333	7	First 7% and last 7%
150-250	24	294	24	First 3% and last 10%

Table A.5.1: Splitting milestone m4 in ranges

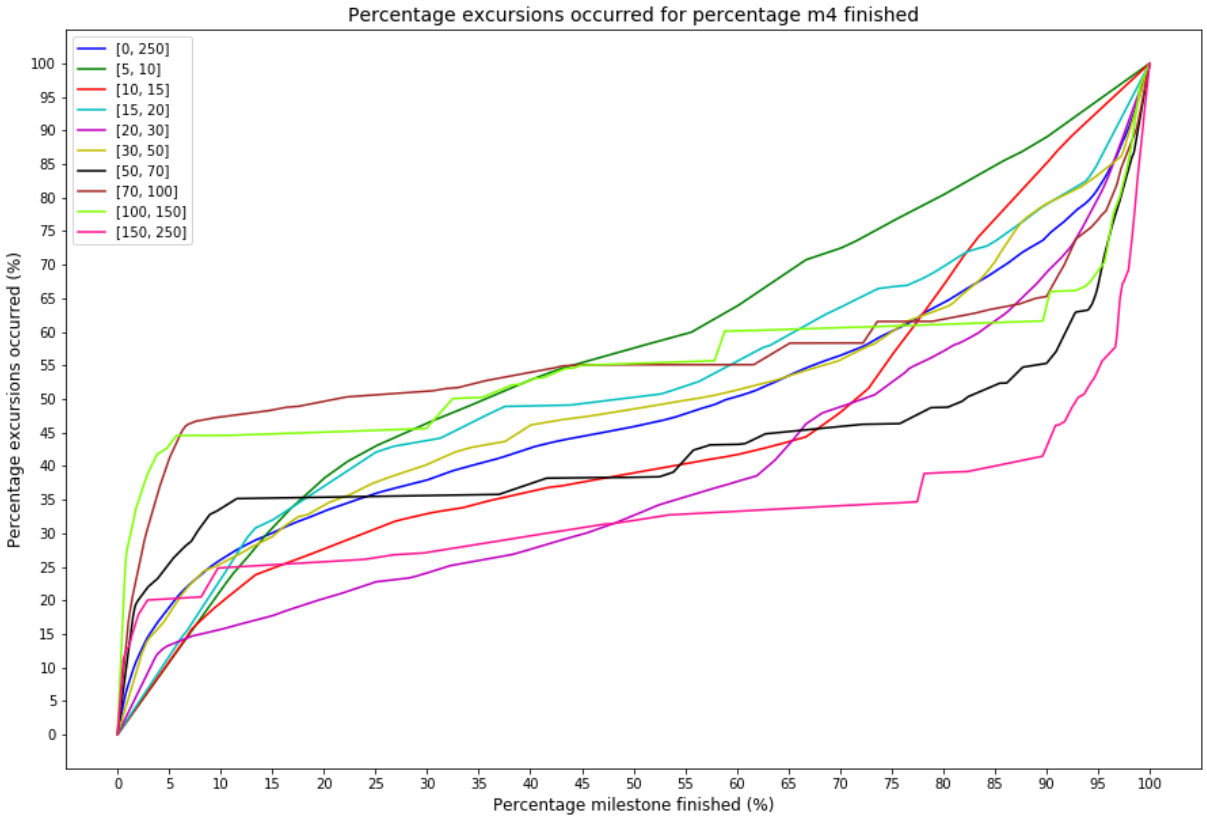


Figure A.5.1: Splitting milestone m4 in ranges

Devices with a number of measurements in range	Devices with excursions	Total number of devices	Average number of excursions of these devices	Part of the data that is kept in the dataset
0-100	564	4539	5	-
5-10	37	316	3	All
10-15	50	348	3	All
15-20	58	403	5	All
20-30	113	733	7	First 30% and last 20%
30-50	145	1274	5	First 30% and last 15%
50-70	118	1026	5	First 10% and last 10%
70-100	76	719	8	First 10% and last 10%

Table A.5.2: Splitting milestone m5 in ranges

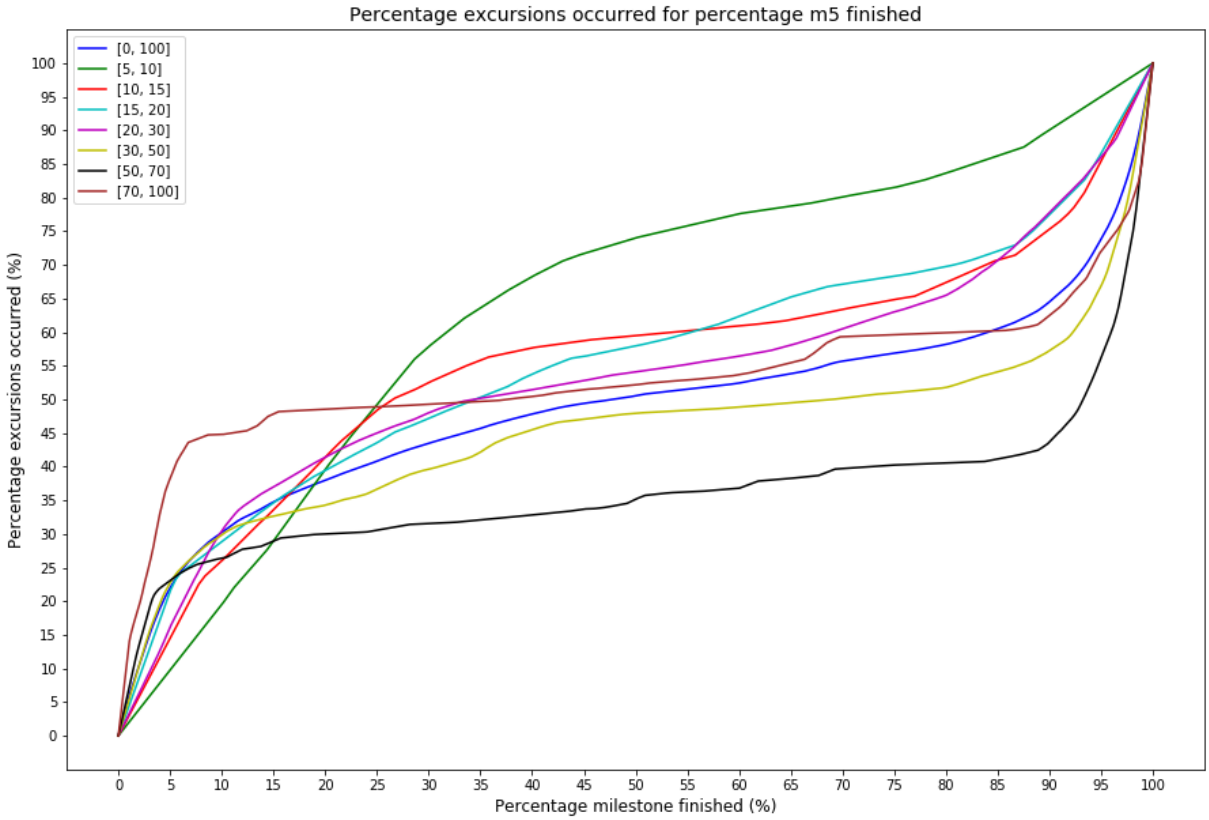


Figure A.5.2: Splitting milestone m5 in ranges

Devices with a number of measurements in range	Devices with excursions	Total number of devices	Average number of excursions of these devices	Part of the data that is kept in the dataset
0-800	307	3853	9	-
5-10	17	400	1	All
10-15	17	170	4	All
15-20	8	106	3	First 5% and last 30%
20-35	3	68	1	First 20%
35-60	12	195	1	First 5%
60-80	13	286	2	First 10%
80-100	9	110	3	First 3% and last 5%
100-150	15	175	5	First 15%
150-300	47	653	25	First 3%
300-450	93	914	8	First 2%
450-600	31	281	12	First 10%
600-700	21	121	18	First 1.5%
700-800	14	135	16	First 2%

Table A.5.3: Splitting milestone m9 in ranges

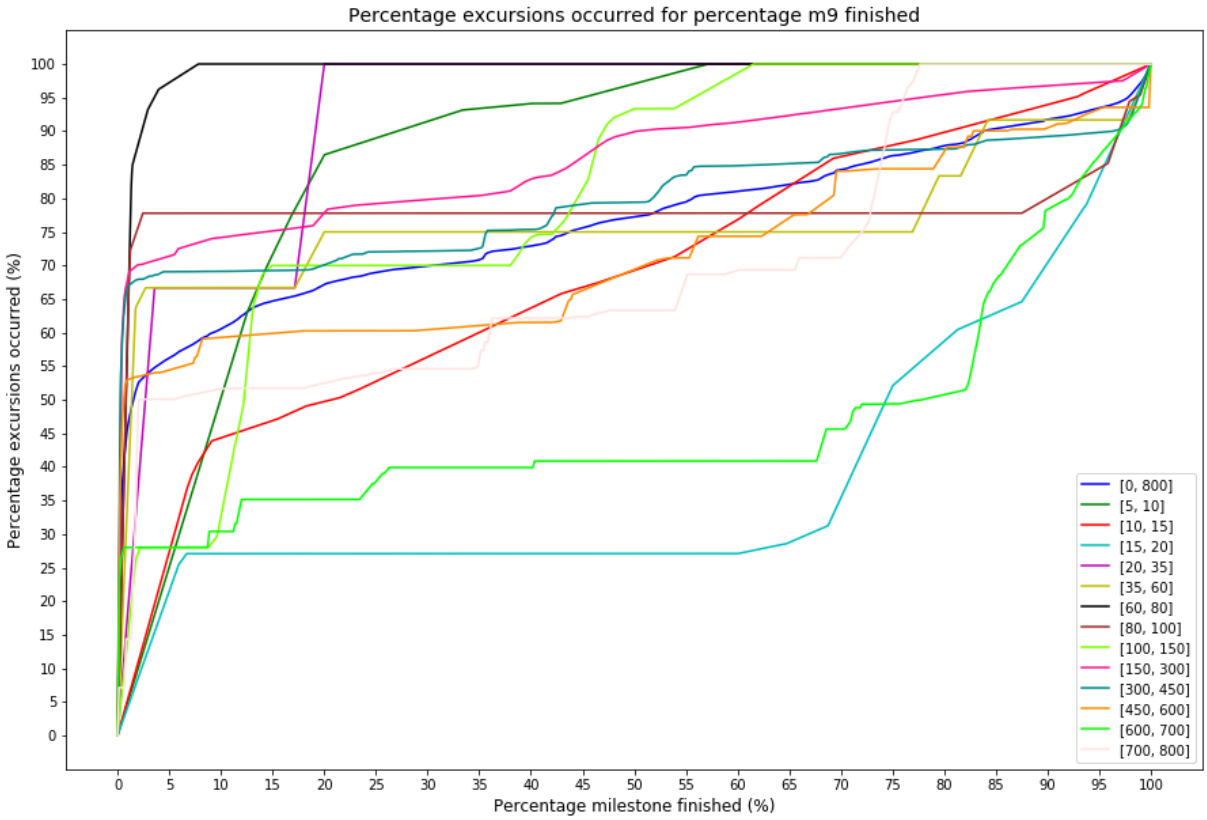


Figure A.5.3: Splitting milestone m9 in ranges

Appendix 6: Correlation of all variables

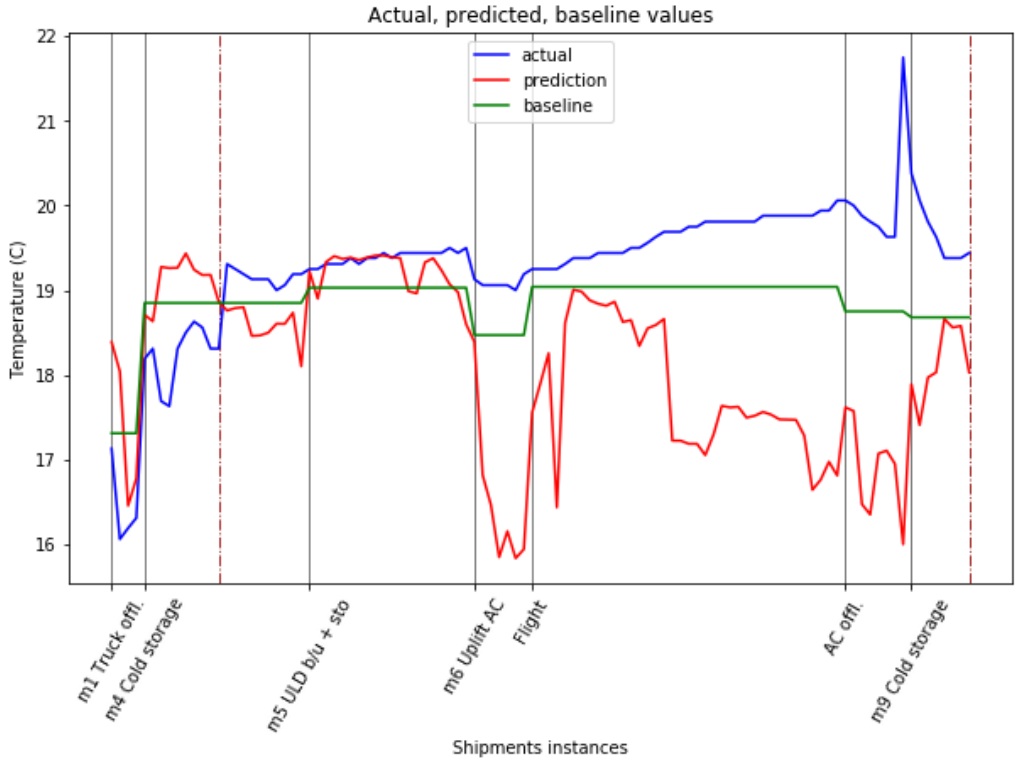
Correlation values

Variable	Correlation Value	Variable	Correlation Value
temp	1.0	weekday_4	0.02
temp_t_2	0.38	morning	0.02
temp_t_1	0.37	late_morning	0.02
mean_w_month	0.35	early_morning	0.01
max_w_month	0.35	early_afternoon	0.01
mean_m_month	0.35	mile_m1	0.01
min_w_month	0.34	night	0.01
m_mean_t_t_1	0.32	early_night	0.0
m_min_t_t_1	0.32	late_night	0.0
m_max_t_t_1	0.29	mile_Aircraft offloading (HSV)	-0.0
summer	0.27	order_frac	-0.0
min_m_month	0.25	order_perc	-0.0
month_7	0.18	evening	-0.01
max_m_month	0.16	weekday_6	-0.01
month_6	0.15	weekday_5	-0.01
month_8	0.15	LUX	-0.01
weather	0.14	afternoon	-0.01
m_max_w_t_1	0.14	weekday_0	-0.01
m_mean_w_t_1	0.14	month_10	-0.02
weather_t_1	0.14	late_afternoon	-0.02
m_min_w_t_1	0.14	year_2017	-0.02
weather_t_2	0.14	year_2016	-0.03
mile_m9	0.12	day	-0.03
max_order	0.11	weekday_2	-0.04
spring	0.09	early_evening	-0.04
HSV	0.09	mile_Flight	-0.06
order	0.06	AIR	-0.06
mile_m4	0.06	month_3	-0.08
month_9	0.06	combi	-0.08
year_2014	0.06	month_2	-0.09
weekday_1	0.05	year_2018	-0.09
month_5	0.05	month_11	-0.1
year_2015	0.04	month_12	-0.11
weekday_3	0.04	fall	-0.12
month_4	0.04	mile_m6	-0.15
mile_m5	0.04	month_1	-0.16
late_evening	0.03	winter	-0.21

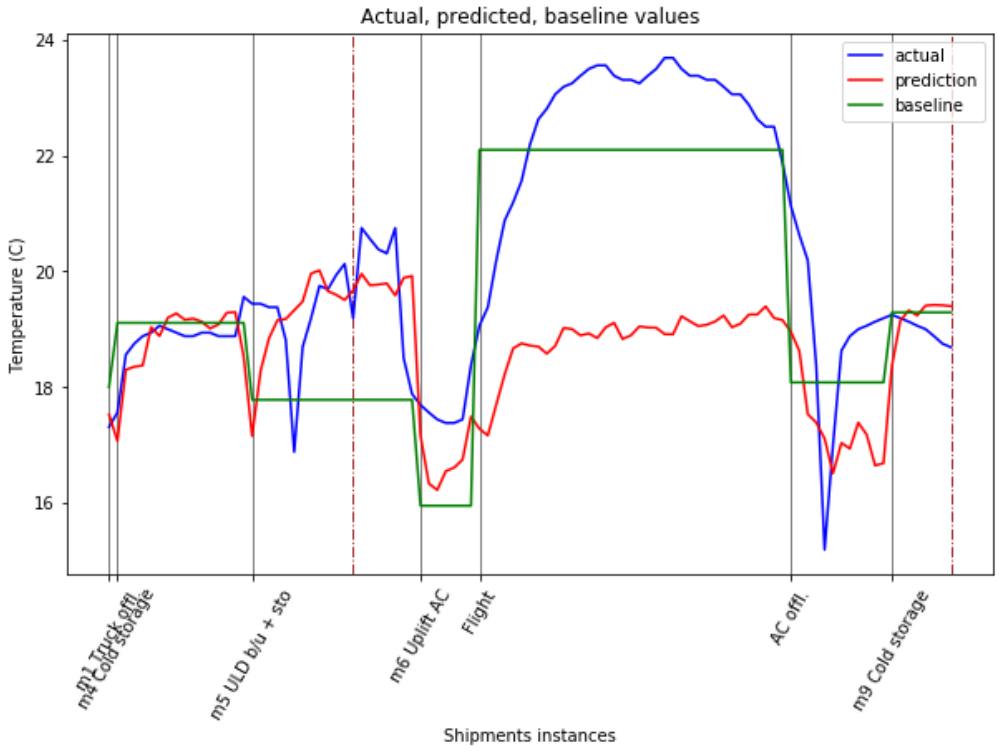
Appendix 7: Performance initial Random Forest on four test shipments

Brown dashed lines indicate where values of cold storage milestones have been removed as described in chapter 5.3.3.

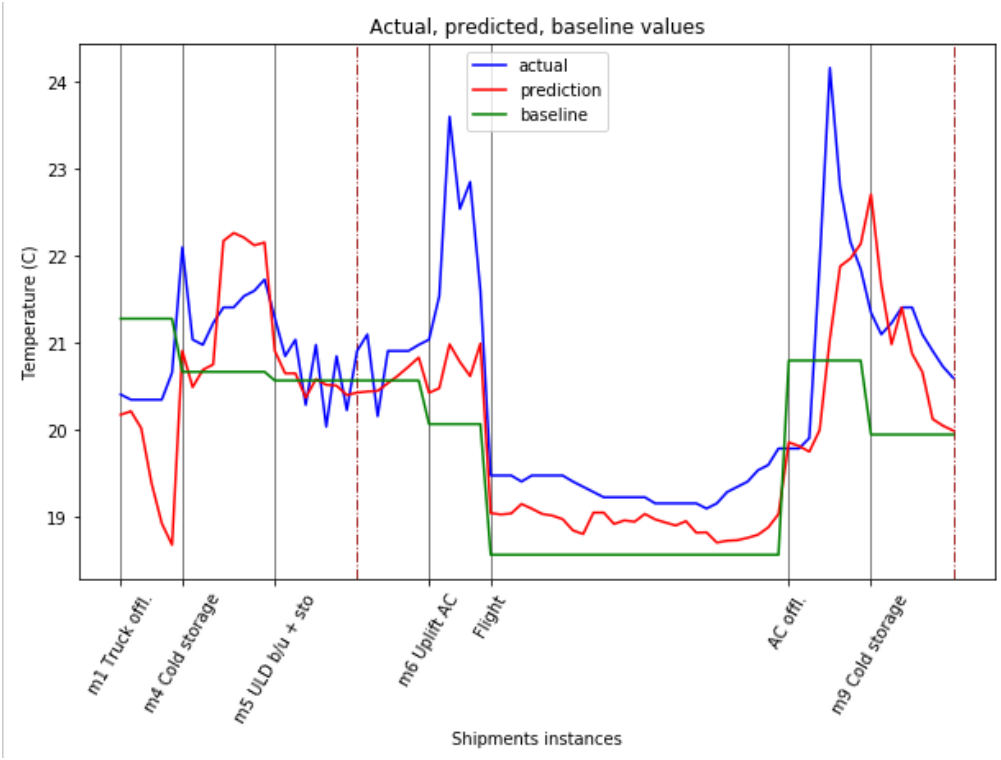
Winter shipment 1



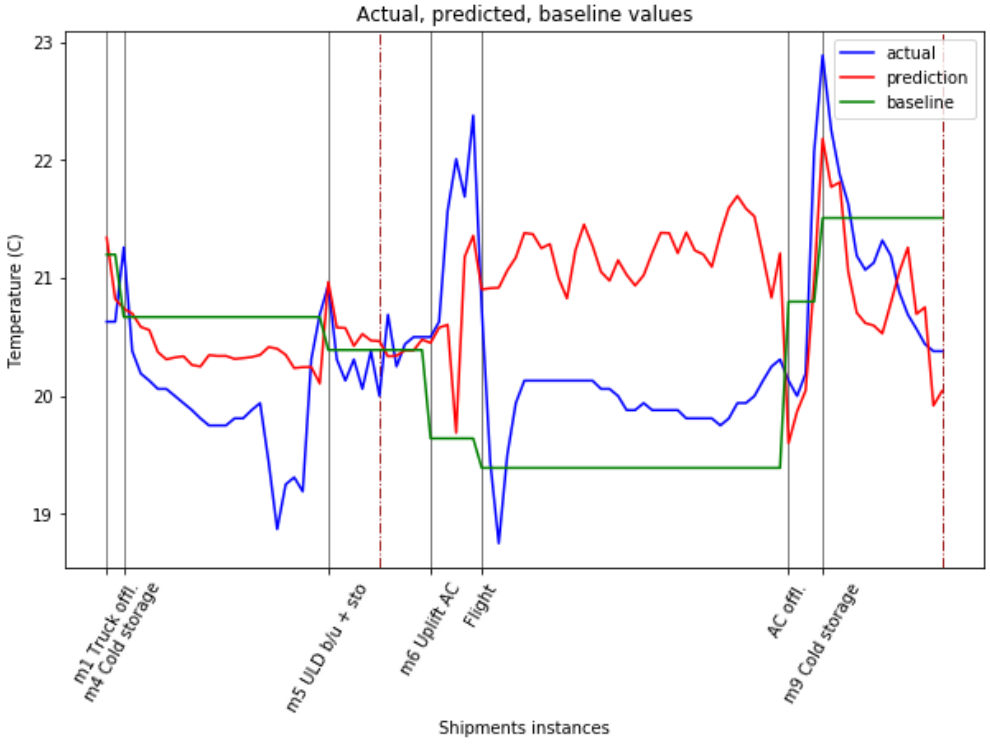
Winter shipment 2



Summer shipment 1

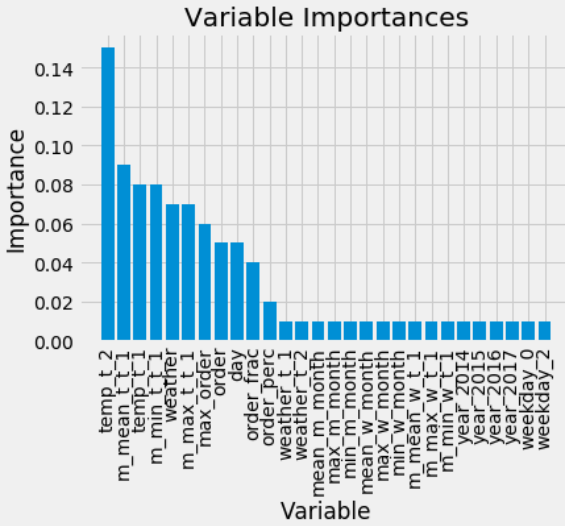


Summer shipment 2

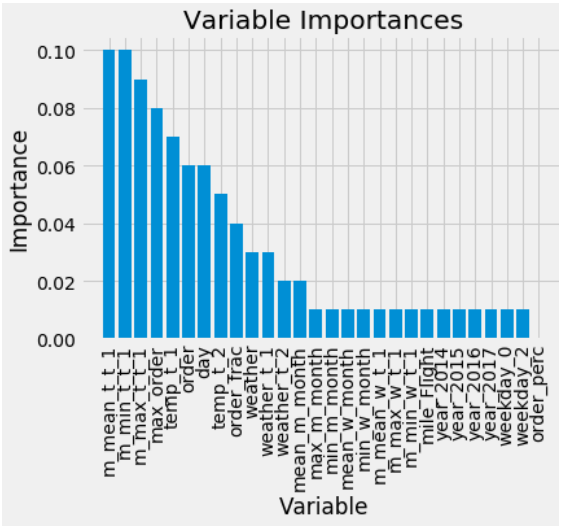


Appendix 8: Relative variable importances of models with random split

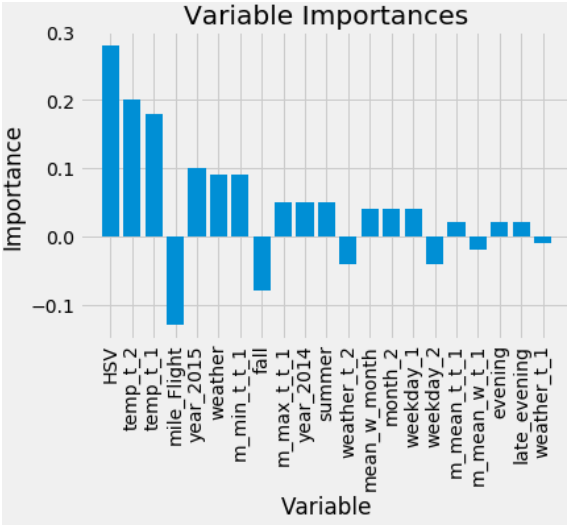
Random Forest



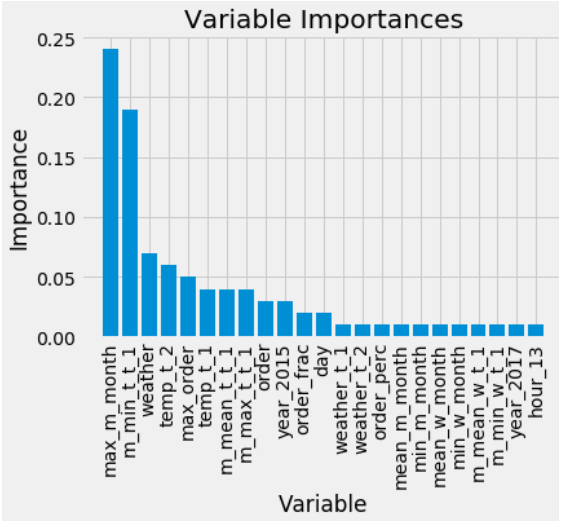
XGBoost



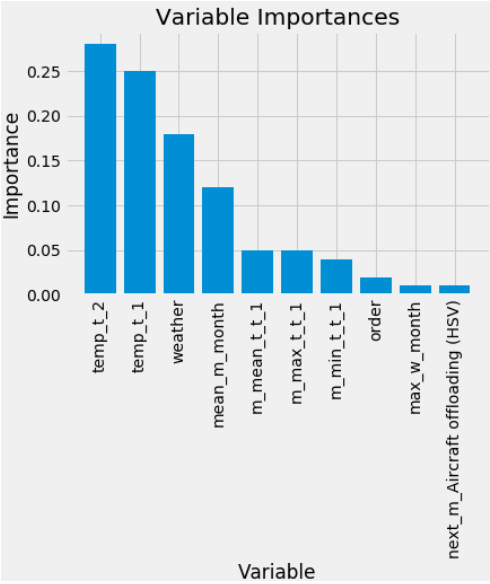
Lasso



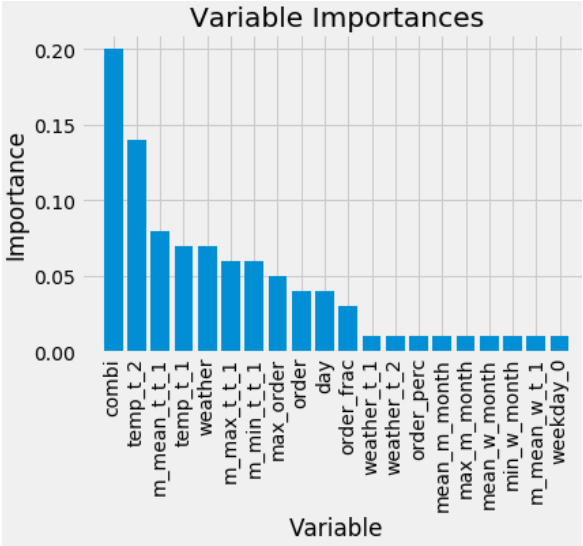
AdaBoost



Gradient Boost



Random Forest with ‘combi’



Appendix 9: Relative feature importances of models with year split

Optimal features identified with SFS are highlighted with green.

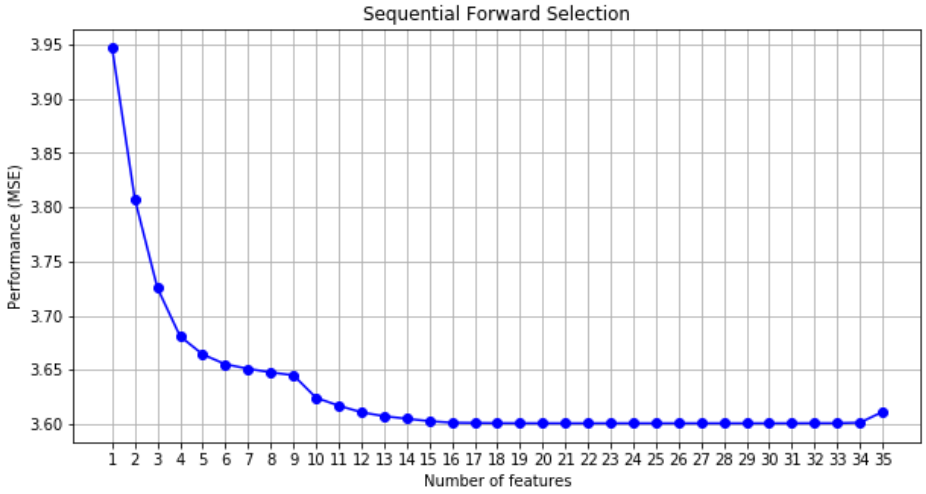
Variable	Lasso	AdaBoost	XGBoost	Gradient Boost	Random Forest
'm_mean_t_t_1'	0	0.04	0.12	0.08	0.02
'temp_t_1'	1	0.11	0.06	0.07	0.11
'temp_t_2'	1	0.11	0.07	0.07	0.48
'winter'	1	0.0	0.0	0.0	0.0
'spring'	1	0.0	0.0	0.0	0.0
'summer'	1	0.0	0.01	0.0	0.0
'fall'	1	0.0	0.0	0.0	0.0
'month_1'	1	0.0	0.01	0.0	0.0
'month_2'	1	0.0	0.01	0.0	0.0
'month_3'	1	0.0	0.0	0.0	0.0
'month_4'	1	0.0	0.0	0.01	0.0
'month_5'	1	0.0	0.01	0.0	0.0
'month_6'	1	0.0	0.0	0.0	0.0
'month_7'	1	0.0	0.0	0.0	0.0
'month_8'	1	0.0	0.0	0.0	0.0
'month_9'	1	0.0	0.0	0.0	0.0
'month_10'	1	0.0	0.0	0.0	0.0
'month_11'	1	0.0	0.0	0.0	0.0
'month_12'	1	0.0	0.0	0.0	0.0
'mile_Aircraft offloading (HSV)'	1	0.0	0.01	0.01	0.0
'mile_Flight'	1	0.0	0.02	0.02	0.0
'mile_m1'	0	0.0	0.01	0.01	0.0
'mile_m4'	0	0.0	0.0	0.0	0.0
'mile_m5'	1	0.0	0.0	0.01	0.0
'mile_m6'	1	0.01	0.0	0.0	0.0
'mile_m9'	1	0.0	0.0	0.0	0.0
'AIR'	1	0.0	0.0	0.01	0.0
'HSV'	1	0.0	0.0	0.01	0.0
'LUX'	0	0.0	0.0	0.0	0.0
'weekday_0'	1	0.0	0.01	0.03	0.0
'weekday_1'	0	0.0	0.01	0.01	0.0
'weekday_2'	1	0.01	0.01	0.01	0.0
'weekday_3'	0	0.0	0.0	0.0	0.0
'weekday_4'	0	0.0	0.0	0.0	0.0
'weekday_5'	0	0.0	0.0	0.0	0.0
'weekday_6'	0	0.0	0.0	0.0	0.0
'afternoon'	0	0.0	0.0	0.0	0.0

'evening'	0	0.0	0.0	0.0	0.0
'morning'	1	0.0	0.0	0.0	0.0
'night'	1	0.0	0.0	0.0	0.0
'early_afternoon'	1	0.0	0.0	0.0	0.0
'early_evening'	1	0.0	0.0	0.0	0.0
'early_morning'	1	0.0	0.0	0.0	0.0
'early_night'	1	0.0	0.0	0.0	0.0
'late_afternoon'	1	0.0	0.0	0.0	0.0
'late_evening'	1	0.0	0.0	0.0	0.0
'late_morning'	1	0.0	0.0	0.0	0.0
'late_night'	1	0.0	0.0	0.0	0.0
'day'	1	0.03	0.07	0.06	0.0
'mean_m_month'	0	0.01	0.01	0.01	0.0
'max_m_month'	1	0.05	0.01	0.01	0.0
'min_m_month'	0	0.01	0.01	0.0	0.02
'mean_w_month'	1	0.0	0.01	0.01	0.02
'max_w_month'	1	0.0	0.0	0.01	0.0
'min_w_month'	1	0.01	0.01	0.01	0.0
'm_max_t_t_1'	1	0.05	0.16	0.13	0.03
'm_min_t_t_1'	1	0.3	0.09	0.12	0.03
'm_mean_w_t_1'	1	0.01	0.01	0.0	0.0
'm_max_w_t_1'	1	0.01	0.01	0.01	0.0
'm_min_w_t_1'	1	0.0	0.0	0.0	0.0
'weather'	1	0.06	0.06	0.06	0.24
'weather_t_1'	1	0.01	0.02	0.02	0.01
'weather_t_2'	1	0.01	0.02	0.02	0.0
'order'	1	0.04	0.06	0.05	0.03
'max_order'	1	0.05	0.05	0.07	0.0
'order_perc'	1	0.02	0.03	0.04	0.0

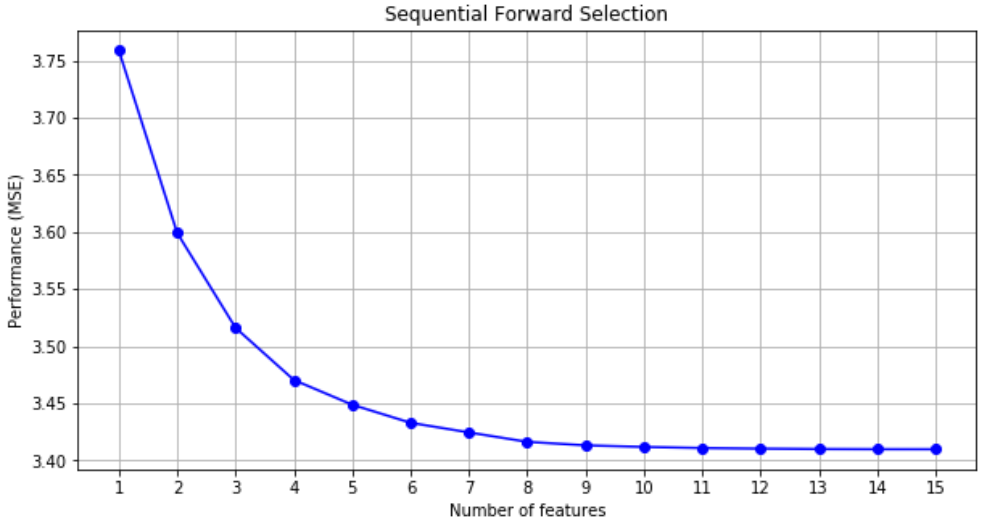
Appendix 10: Sequential forward selection plots

The figures below show the MSE of the number of features for each method.

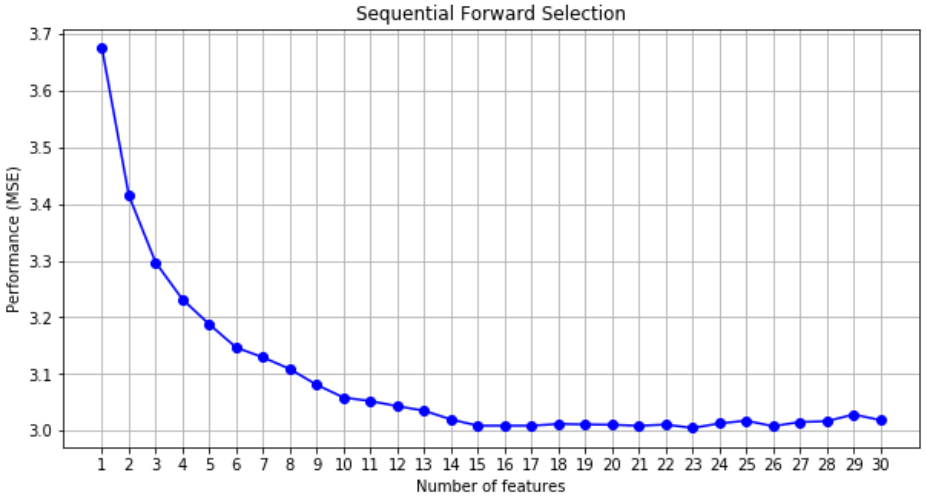
Lasso: 25 features



Random Forest: 15 features



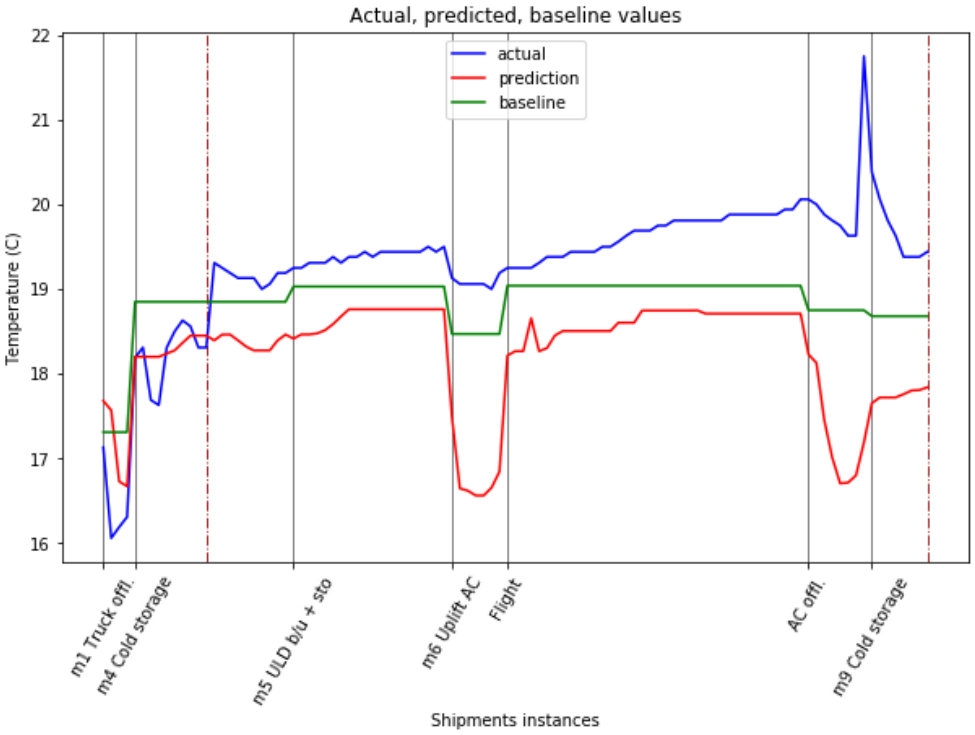
XGBoost: 23 features



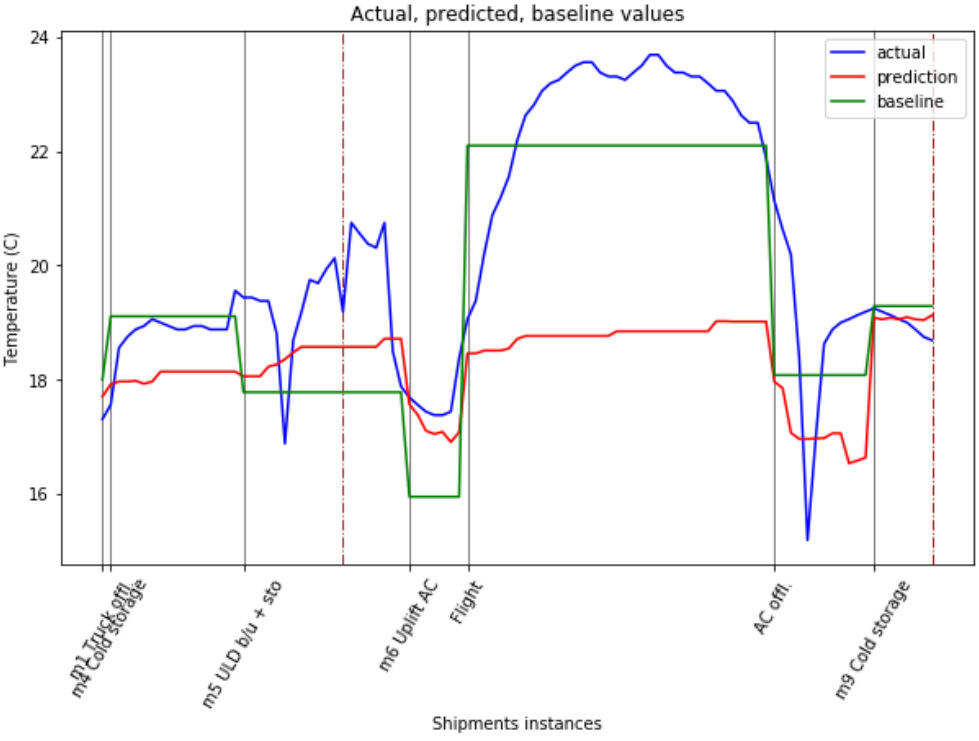
Appendix 11: Performance XGB model on four test shipments

Brown dashed lines indicate where values of cold storage milestones have been removed as described in chapter 5.3.3.

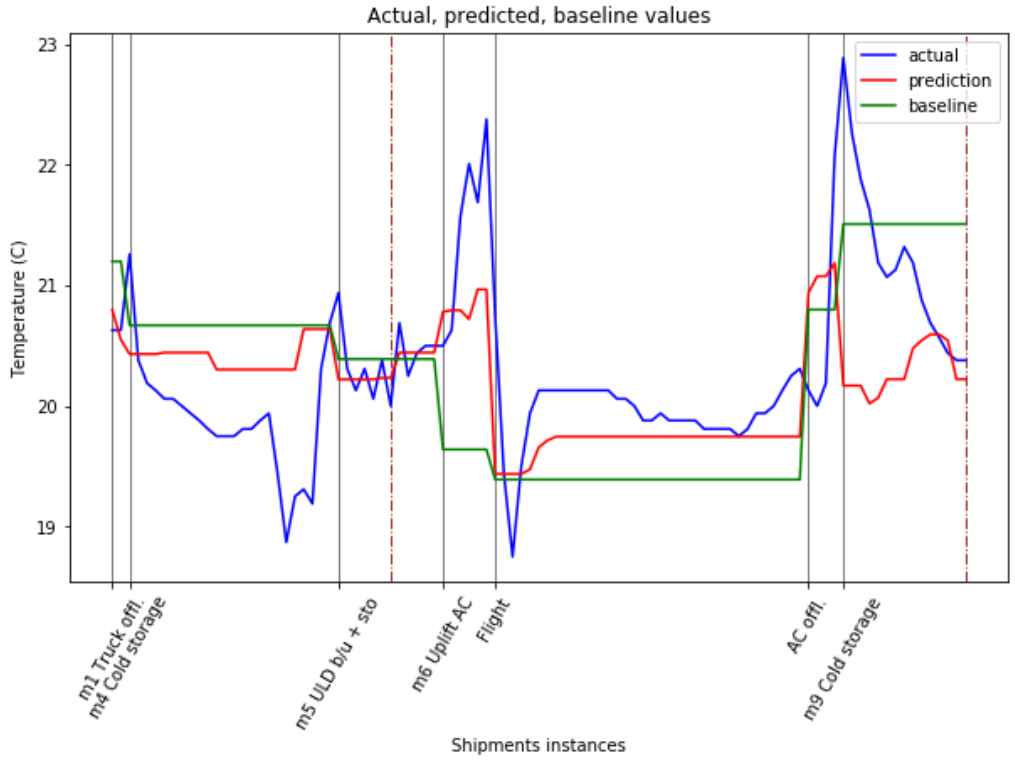
Winter shipment 1



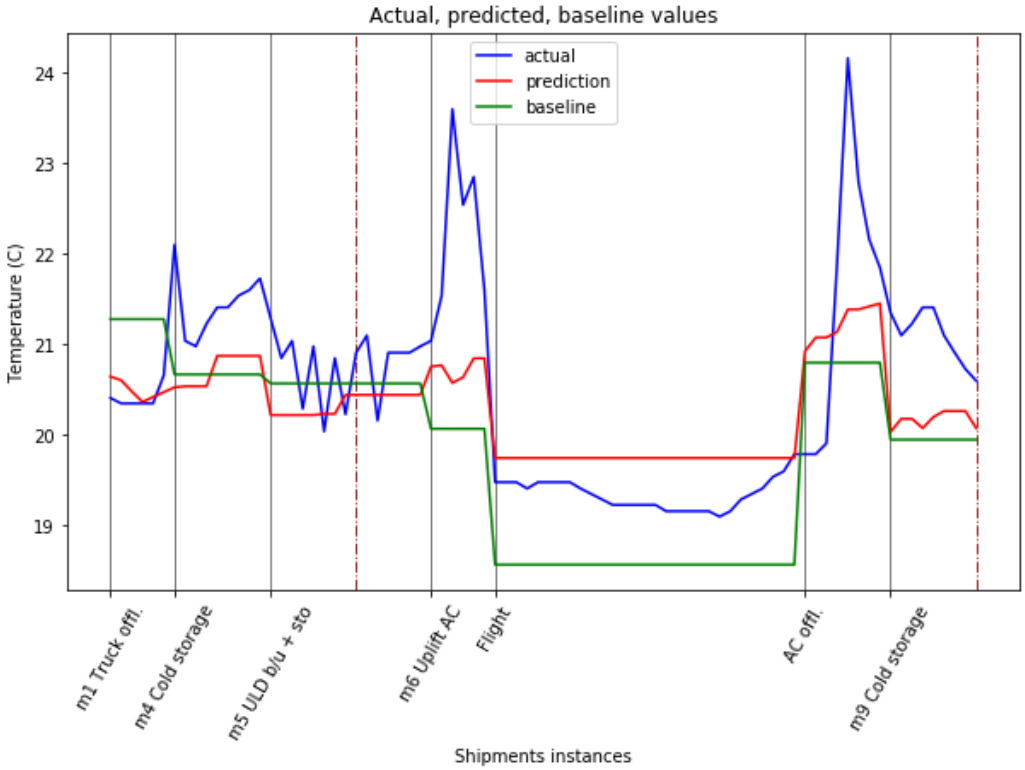
Winter shipment 2



Summer shipment 1

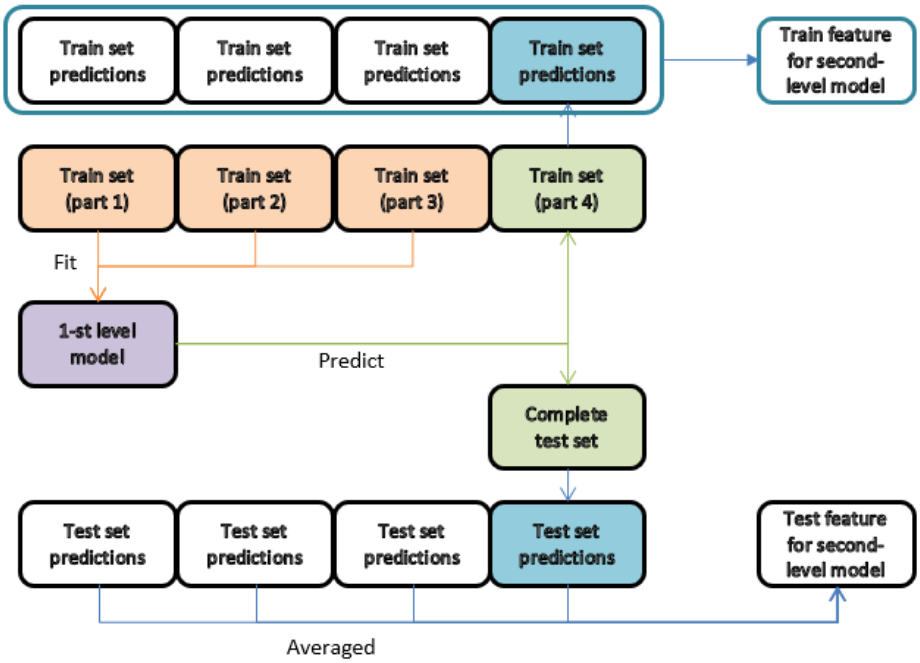


Summer shipment 2

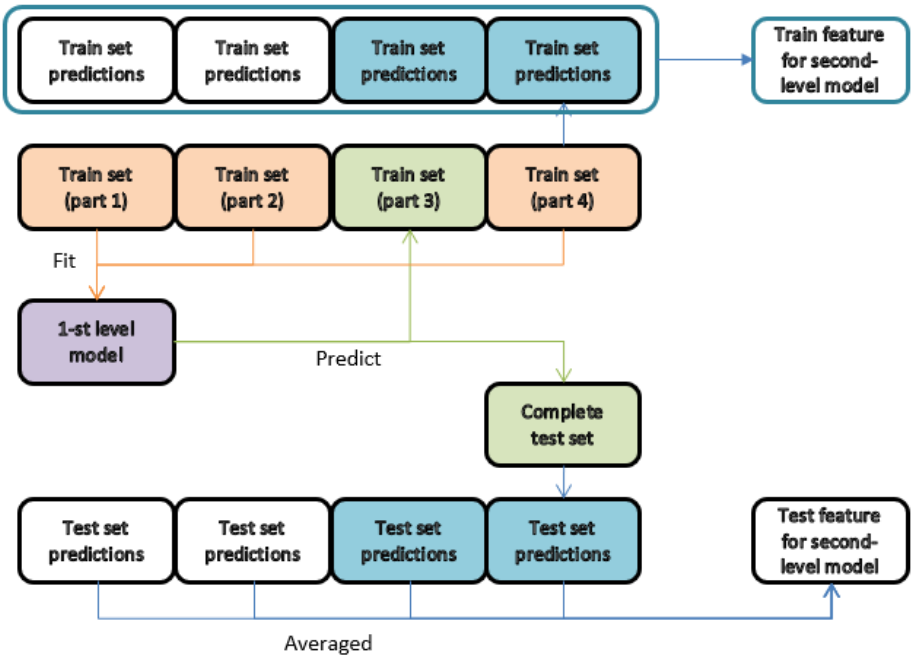


Appendix 12: Stacking process

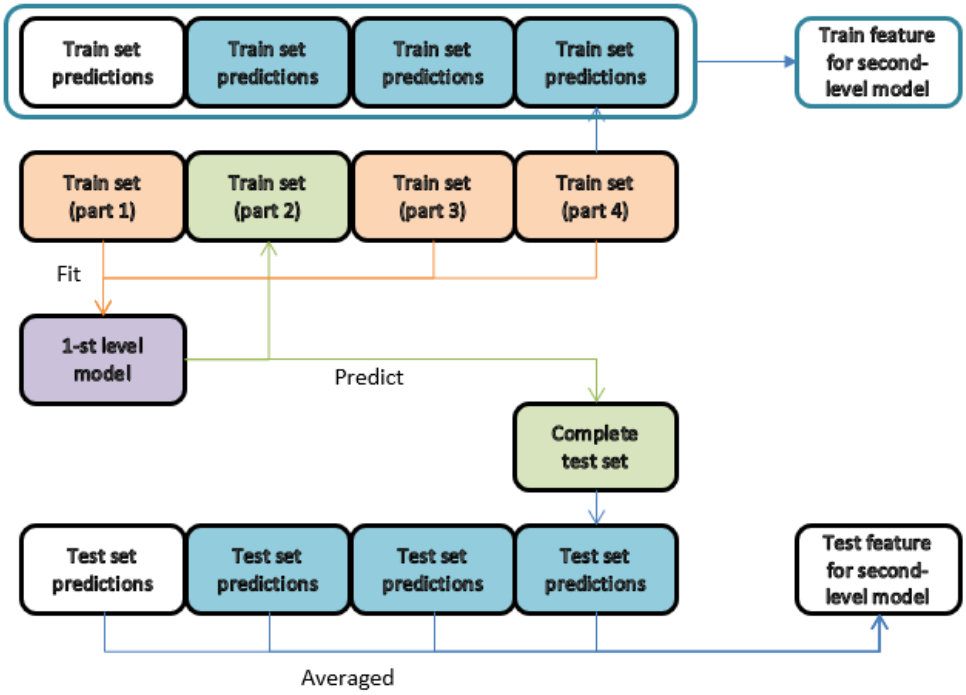
First fold of a single first-level model



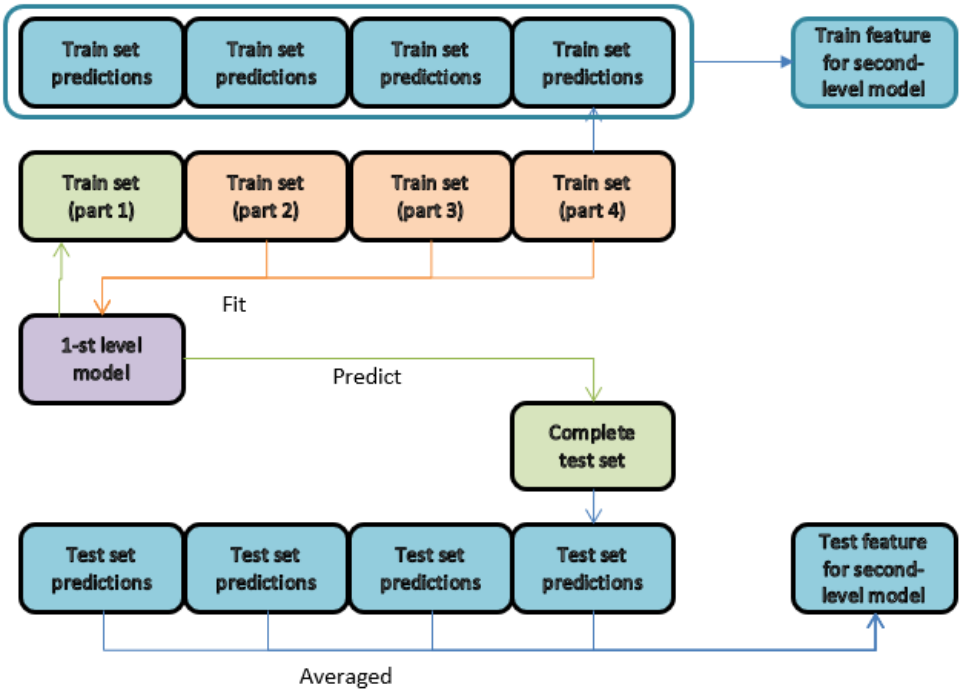
Second fold of a single first-level model



Third fold of a single first-level model



Fourth fold of a single first-level model



Combine the results of three first-level models, and use these as train and test features for the second-level model