Development of a diagnostic and prognostic model for condition-based maintenance at ASML

van der Aalst, P.

Award date:
2013

Development of a diagnostic and prognostic model for condition-based maintenance at ASML

by

Paul van der Aalst

BSc Industrial Engineering — TUE 2013

Student identity number 0637138

in partial fulfilment of the requirements for the degree of

Master of Science

in Operations Management and Logistics

Supervisors:

dr Hao Peng, TU/e, OPAC

dr. Dmitry Krushinsky, TU/e, OPAC

ir. Gert Streutker, ASML, Customer Support- Operational Services

TUE. School of Industrial Engineering.

Series Master Theses Operations Management and Logistics

Subject headings: diagnostics, monitoring, prognostics, proactive maintenance, preventive maintenance, condition based maintenance, statistical process control, data mining, CRISP-DM

ABSTRACT

This master thesis describes a research project conducted within the Customer Service department at ASML. The possible value of a certain type of machine data for diagnostic and prognostic purposes was investigated, and a decision support model has been developed around it. A combination of several data mining techniques and statistical process control was used to analyse the data and build a decision support model. Furthermore, the Cross Industry Standard Process for Data Mining (CRISP-DM) was used as a guide throughout the thesis. Due to the uncertain environment of both the data itself and application of that data, the roadmap explored during the project is an important part of the thesis. The thesis concludes with a discussion on the main findings, limitations and future recommendations.

Keywords and phrases diagnostics, monitoring, prognostics, proactive maintenance, preventive maintenance, condition based maintenance, statistical process control, data mining, CRISP-DM

PREFACE AND ACKNOWLEDGEMENTS

This report is the result of my master thesis project in completion of the Master of Science Program in Operations Management and Logistics (OML) at Eindhoven, University of Technology (TU/e). This thesis was carried out during my internship at the Customer Service department at ASML, from February 2013 till August 2013. I would like to use this opportunity to express my gratitude to many people for their help and support.

First of all, I would like to thank my first supervisor from the TU/e, Dr. Hao Peng, for guiding me through the process. Thank you for introducing me to ASML, via your contact with Gert Streutker. As a result of some unanticipated changes, she assisted me in finding a backup second supervisor. Therefore I would also like to thank Dmitry Krushinsky, my second supervisor, for his last-minute involvement in my thesis.

Within ASML, firstly I would like to thank Gert Streutker, my supervisor at ASML, for giving me the opportunity to carry out my master thesis project within ASML. His critical, but constructive, feedback was helpful in performing the research and writing the thesis. Besides, I would like to thank my office colleges Elio Cechetto, Pieter Phoelich, Jonas Rossander, and David Sigtermans for both their content related knowledge and pleasurable social working environment. I enjoyed the anecdotes and short stories, which may even prove helpful in my future career. Finally I would like to thank everybody else that I met within ASML, for their contribution to my research.

I would like to thank my dear family, friends and girlfriend, for their support and interest during my entire study and this project in particular. Finally, I want to dedicate this thesis to my mother, who passed away at the beginning of my master program. She helped me get to where I am today.

Paul van der Aalst

Veldhoven, August 2013

EXECUTIVE SUMMARY

This thesis is executed within ASML customer service. As world leader in the production of lithography systems for the semiconductor industry, ASML wants to preserve its market share by satisfying its customers. Lithographic systems are complex capital goods that require large financial investment. As a result, the opportunity cost in case of machine downtime is also high. To satisfy customers, ASML provides a maintenance support service to achieve an as high as possible availability. To achieve such availability, any form of downtime should be resolved as quickly as possible or even prevented. While currently the maintenance is mainly failure-based, ASML wants to move towards pro-active maintenance. With diagnostics and prognostics on machine status, the unscheduled downtime can be minimized. As part of this Condition Based Maintenance (CBM) research, the following research goal was developed:

"Develop a prognostic/diagnostic tool applicable to machines that find themselves in the middle part of their lifecycle (constant failure rate)."

To achieve this goal, the Cross Industry Standard Process for Data Mining (CRISP-DM) approach was followed. Via the steps, business understanding, data understanding, data preparation, modeling, evaluation, and deployment a strong judgement of the possibility of diagnostics and prognostics for CBM purposes was obtained. This executive summary will continue with short describtions and conclusions for each of the steps of CRISP-DM, after which the main findings and future recommendations are summarized.

The first step of this thesis was *business understanding*. General insight into how data is collected and stored was obtained, and the available software tools were explored. The most important result of the business understanding phase was to get a feeling about the ASML CBM environment. As a final step of business understanding, it was decided to select machine constant data as the data type to be analysed.

During the *data understanding* phase, the content and behaviour of the machine constant data was reviewed. The data proved to include only few missing data entries or impossible values (<1%). It was concluded that it is impossible to understand the behaviour of machine constants by analysing it for individual machines, but only via reviewing the population as a whole. When reviewing the behaviour of the population, the well-known bell-shape of the normal distribution was observed as the standard behaviour of machine constant values. This was an important observation for the leftover part of the CRISP-DM cycle.

The selected data set only required small adaptations before it could be used as input for the modeling phase. During the *data preparation* phase, a small sample of the total number of machine constants was selected. The requirements of these machine constants were a numeric value, positive variance, and a total number of entries of more than one-hundred. The reason for this was to achieve a normal distributed dataset, applicable to statistical process control.

During the *modeling* phase, eight different statistical sub-models were combined into one model. Based upon statistical process control, the behaviour of machines (in the form of machine constants) is searched for anomalies. Each of the eight different models considers a different form of behaviour and makes a judgement based upon the warning limits of plus and minus two times the standard deviation from the mean of the normal distribution of the particular behaviour reviewed. This model proved able to identify anomalies in machine constant behaviour and provides a list of outliers.

During the *evaluation* phase, the list of outliers obtained via the model created during the modeling phase was evaluated for diagnostic and prognostic value. The diagnostic value is defined as the possibility to identify machines that are more likely to be broken down and the identification of the exact moment that this is the case. After testing the significance of the diagnostic value of the outlier list (and with it the eight different statistical sub-models), it could be concluded that the model could add diagnostic value to the CBM

environment. There is a significant relation between machine constant anomalies and the current machine status. Regarding the prognostic value of the machine constants; even after four different attempts to find a relation between anomalies in machine constant behaviour and upcoming failures, there was no reason to assume that there is any kind of prognostic value in this type of data. All attempts resulted in insignificant results.

Because the *deployment* phase was not performed, it is part of the future recommendations. As future recommendations regarding the deployment of the diagnostic value of the model, it is suggested to validate the results in practice with the use of the expertise of the field engineers or any other expert. The developed tool is not ready for practical implementation, but could be with only a few adaptations. These adaptations mainly consist out of automating the process, making it user friendly, bug free, and able to produce a clear report for the field engineer.

As found during the evaluation phase, this thesis concludes that only diagnostic value can be extracted from the selected dataset, and that prognostics will more likely be found in another data type or possibly not at all. The concept of outliers is especially useful for the middle part of the machine life cycle (bathtub curve), because this area is described as a rather stable and homogenous environment with a low constant failure rate. This description suits the normal distribution, in which the outlier mechanism works at its best. To make a judgement with absolute certainty, the diagnostic value needs to be validated in practise and the limitations on prognostic value should be verified.

Summarizing the main thesis findings with two sentences, could be done as follows:

- A significantly successful machine constant monitoring tool was developed, to assist in the diagnostics of the CBM environment for machines in the middle part of their lifecycle.
- Four different attempts have failed in proving the prognostic value of machine constants, which leads to the recommendation not to continue the research regarding extracting prognostic value from this type of data.

TABLE OF CONTENTS

1 INTRODUCTION

This master thesis is part of the on-going cooperation between the TU/e and ASML, on Condition Based Maintenance (CBM) research. The goal of this thesis was the development of a diagnostic and prognostic maintenance tool for ASML machines that find themselves in the middle of their lifecycle. The tool aims at increasing the availability of the lithographical machine, by reducing the unscheduled downtime. A new type of unexplored data (machine constants) was used to extract machine information.

Due to confidentiality of ASML critical business information, parts of this thesis are unavailable in the published report (ASML-only appendix 12). This detailed background information is non-critical for the readability of the thesis, but is intended as support for current and future stakeholders within ASML. The report is organized in 12 chapters, starting with this introduction and ending with a description of the literature reviewed. To improve the readability of this thesis, a list of abbreviations and acronyms is given in appendix 3. In the next chapter, the thesis starts off with an introduction of ASML and their business environment for the unfamiliar reader.

2 COMPANY DESCRIPTION

2.1 COMPANY PROFILE

ASML is the world's leading provider of advanced lithography systems for the semiconductor industry. The company manufactures complex technology machines that are critical to the production of integrated circuits or chips (ASML, About, 2013). ASML designs, develops, integrates, markets and services these advanced systems used by their customers to create chips that power a wide array of electronic, communications and information technology products. Among ASML's customers, there are many well-known, market leading companies such as Samsung, Intel, Toshiba, Texas Instrument, IBM, Micron and TSMC. Founded in the Netherlands in 1984, the company is publicly traded on stock markets Euronext (Amsterdam) and NASDAQ since 1995.

ASML's corporate headquarters is in Veldhoven, the Netherlands. Manufacturing sites, R&D facilities, Technology development centres, and training facilities are located in the US, Taiwan and the Netherlands. Additional Technology development centres and training facilities are also located in Japan and Korea. By this network of facilities an optimal service is provided to its customers in more than 55 locations in 16 countries. Despite the vast growth of ASML towards their current size, the organization is still characterized by its close cooperation – internally, and with customers and suppliers worldwide. This close cooperation atmosphere reflects ASML's business strategy, which is based on technological leadership, operational excellence and customer focus (ASML, Profile, 2013).

Due to technological progress ASML is able improve their machines with such a speed that the company can follow the Moor's Law (ASML, Corporate Presentation, 2013). In general, this means that the specifications of their machines improve with a factor 2 within two years (doubling or dividing attribute values with a factor 2). An appropriate way to measure this trend is by looking at the total number of Petabytes (one Petabyte is one million Gigabytes) of computer memory produced every year thanks to the introduction of new ASML scanners. ASML's positive economic and environmental performance can be proved when the company's sales and emissions are measured against the increasing number of Petabytes made on ASML machines. This is how ASML adds value for its customers and they will continue with this proven strategy in the future (ASML, Performance and Targets, 2013).

As a side-effect of the technological progress, also the complexity of the lithography machines rises. Therefore ASML does not only sell machines, but is also able to take care of fab start-up, training, maintenance and other support services. ASML's service contract portfolio provides a flexible solution to meet any customer's need. This enables customers to focus on their core business and gain and sustain a competitive edge in the marketplace, with the use of the best lithography systems available (ASML, Profile, 2013).

ASML's largest business focuses on lithography systems for 200- and 300-millimeter diameter wafers, and the immersion lithography systems are the state-of-the-art technology in this field. Other business activities include customized solutions (e.g. for compound semiconductors), software-enhanced lithography and the commercialization of extreme ultraviolet (EUV) lithography (ASML, Organization, 2013).

Figure 1 shows the general hierarchical structure of ASML, with the Supervisory Board on top.

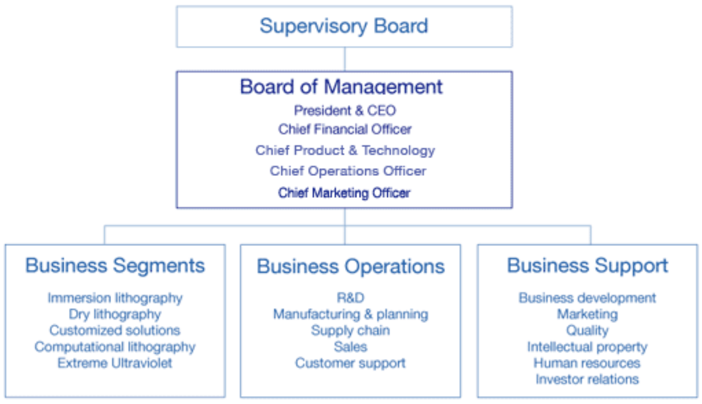


FIGURE 1. ASML BUSINESS STRUCTURE

2.2 LITHOGRAPHY SYSTEMS

ASML's core product is the lithography systems. The machines are used for the production of computer chips. The company uses laser to optically image a defined pattern upon a silicon wafer, coated with a film of light-sensitive material, via a high-detailed lens. After exposure, the unexposed parts of the wafer are etched way, to reveal the projected pattern. The wafer is then further processed by repeatedly exposing the wafer to the above steps, until the final transistor patterns is generated (ASML, Corporate Presentation, 2013). See Figure 2 for an overview of the production cycle of a wafer. The different layers on the wafer do not necessary get applied on the same (type of) machine and several non-lithographic processes take place between layers. The lithographic process can be done with several different approaches including dry lithography, immersion lithography, and EUV. The current industry workhorse, the high-end Twinscan NXT:1950i system (2011, see Figure 3), uses the EUV approach and is able to produce features down to 32 nanometres at up to 200 wafers per hour (ASML, Press Releases, 2013). With the size of a cargo container and a price tag of on average 27 million euros, this piece of cutting edge technology dominates the three main lithography system markets worldwide, namely Asia, Europe and the United States (ASML, Performance and Targets, 2013).

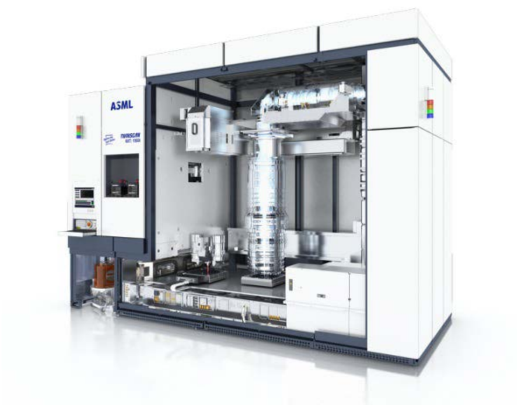


FIGURE 3. TWINSCAN NXT:1950I

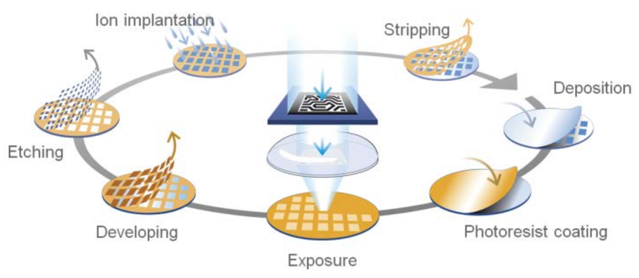


FIGURE 2. WAFER PRODUCTION CYCLE

3 Research Design

In this chapter a description of the thesis's research design is given. First, a problem statement is described (3.1). Accordingly, the research question and several sub questions are formulated (3.2). This is followed by a definition of the thesis's scope (3.3), literature overview (3.4), thesis deliverables (3.5), roadmap (3.6), and the thesis time schedule (3.7).

3.1 Problem Statement

ASML's customers continuously want to increase their productivity and decrease cost involved. The availability of the lithographic machines, provided by ASML, has a relevant influence on the productivity of the manufacturing companies. To be able to satisfy customer's requirements on system availability, ASML needs to reduce total downtime. One method for decreasing total downtime is the application of Condition Based Maintenance (CBM), which is a proactive maintenance strategy based on machine health status. By monitoring the status of machines, more efficient scheduling and preparation of upcoming failures can be achieved. To successfully apply CBM it is important to have prognostic and diagnostic tools that adequately represent the actual machine status. Solutions for applying CBM range from maintenance optimization models (Timmermans, 2012) up to data-driven prediction models (Cakir, 2011).

The latest area of ASML's interest is the diagnostics and prognostics of machine (parts) that find themselves in a particular section of their lifecycle. The failure rate of a part over its lifetime can be visualized in the well-known bathtub curve (Figure 4), which divides the lifetime into three different phases. The first phase of the lifecycle is characterized by a high initial failure rate. As a new product is developed, many issues are not yet found or solved. With updates and tuning of the product, this infant mortality decreases over time to a more steady state. While some companies do not bring their product to the market until it is free of child deceases, ASML's strategic advantage is built upon providing the customers with cutting-edge technology as soon as possible. Due to the newness of the technologies within the machines, they usually start in the first phase of the lifecycle. Customers agree with receiving unstable products, because the benefits of new technologies outweigh the drawbacks of a higher initial failure rate. This second phase of the lifecycle is characterized by a constant failure rate, because most of the initial issues are resolved and the influence of age and wear out is not yet significant. During this phase failures occur due to unusual events which are considered random. In the final phase of the lifecycle of a part the influence of age and wear out on the failure rate becomes noticeable. The failure rate increases with time due to slow deterioration of parts.

Most tools available for analysing machine status focus on the final phase of the curve, where the parts experience deterioration. The interest on this final part of the cycle can be explained by the fact that the increase in rate of failure leads to more failures (more downtime and cost), and the fact that it is possible to apply classical theoretical models in analysing the deterioration of parts over time. With this part of the cycle being covered relatively well, ASML's future focus is on tools that are able to analyse the behaviour of machines in the middle part of the curve (constant failure rate). Despite the fact that the number of failures is relatively low, these totally unexpected failures can lead to long down-times. Due to the fact that deterioration is not the main cause of failure in this part of the cycle, the application of rather new data-driven methods need to be explored. ASML supposes that certain machine data, described in paragraph 5.1, could be related to machine health status and performance and could be used to fill in the gap of CBM tools in the middle part of the lifecycle.

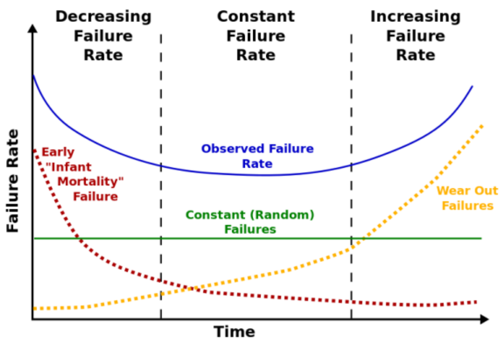


FIGURE 4. BATHTUB CURVE

Besides the possible relation between machine data and failure rate, ASML suspects that this data could also be used for optimizing the testing and calibration. A vast amount of downtime is related to the calibration and testing activities that need to be done after a machine down (part swap), or setup of a new machine. ASML felt there were opportunities in minimizing the downtime related to these activities by optimizing the selection of test executed, based on different kind of machine part swaps. This can be seen as a different problem from the actual diagnostics and prognostics of machine health status, but is still part of the whole CBM methodology. Depending on the condition of the machine, the most efficient set of tests and calibrations can be executed. Summarizing the whole problem statement; ASML wants to extent and improve their CBM approach with the use of machine data, specifically applicable to machines in the middle part of their lifecycle.

3.2 Research Question

Develop a prognostic/diagnostic tool applicable to machines that find themselves in the middle part of their lifecycle (constant failure rate).

The research goal can be achieved via the following sub-questions:

- What does the selected data represent?
  a. *How is the quality of the selected data?*
  b. *How does the selected data behave?*
  c. *Which tooling/software is applicable to the data?*

- Is it possible to extract any useful information from the selected data?
  a. *What is a suitable approach for analysing a large, rather unknown data source?*
  b. *Is it possible to apply any of the by literature suggested methods of data mining to the selected data?*
  c. *Is it possible to relate the selected data with the current machine status?*
  d. *Is it possible to relate the selected data with the future machine status and include this into a Preventive Maintenance approach?*

- What could be the future applications of monitoring the selected data?

The build-up of the research questions is focused on finding a possible solution path for analysing the specific area of the machine part lifecycle. See paragraph 12.1 for a description of prognostics and diagnostics. The first sub question searches for a general understanding and interpretation of the data given. These questions are answered during the business and data understanding phase of the thesis. The second sub question uses the generated data knowledge to select and apply a data mining method for extracting useful information from the dataset. It approaches the data from both a diagnostic and prognostic view. The third sub question focuses on an advice about the direction this research should continue after this thesis, based upon the gathered knowledge throughout the process.

3.3 SCOPE

In accordance with the research questions, the thesis scope is defined as follows:

(+) Data mining: Appropriate methodology is reviewed to analyse the data provided by ASML, following the CRISP-DM process steps described by Wirth et al. (2000).

- CRISP-DM.

(+) Pro-Active Maintenance (PAM): This cross-functional ASML project team is researching the possibilities and applications of condition-based maintenance via prognostic and diagnostic tooling. This thesis took place in an exploratory area of the PAM project for which the added value of machine constant monitoring was uncertain, but promising. Machine Constants could be the solution for machine monitoring issues, for machines in the middle of their lifecycle.

- Pro-Active Maintenance (PAM).

(+-) Data: The initial dataset consisted out of the data from 642 machines, for 89309 machine constants (see paragraph 5.1), over a period of +-1.5 month (01-Feb-13 till 15-Mar-13). This dataset includes all machines and their constants that were set and stored during the selected period. Due to the high number of machine constants, even a rather small period leads to a large dataset. The benefit of a large dataset is that the possibility of finding interesting results increases, but the drawback is the large calculation power required. The initial dataset was later on filtered during the data preparation phase of the thesis (see chapter 7), to meet the modeling requirements and project goal.

- 642 machines.
- 89309 machine constants.
- 01-Feb-13 till 15-Mar-13.

(+) Machine & Factory Constants: The machine constants are settings defined by machine software drivers, after automated tests/calibrations, which determine how the machine should perform. Factory constants are similar, but differ from machine constant in the fact that factory constants represent the influence of customer decisions on the settings of the ASML machine (for example, the edge clearance). All machine/factory constants relate to a physical attribute within the lithographical machine and influence the three basic KPI's of ASML's machines; Overlay, Imaging and Throughput (see paragraph 5.1). The focus of this research project is not on improving the specifications of the machines themselves, but trying to maximize the overlay, imaging and throughput with the use of the constants data. Throughout the thesis the term 'machine constant' will be used for both machine & factory constant, because 97.97% of the data is related to machine constants.

- Machine Constants

(-) Job (process) Constants & Test Constants: Job and Test constants are not included in the scope of the thesis, because they are not included in the data. Job constants differ from the previous mentioned machine and factory constants, because they do not represent any of the settings that can be influenced by ASML's machines. Job Constants represent the influence of non-ASML processes before or in-between wafer processing by the lithography system. Because Job constants may influence the output of the machines, it may be of future interest to include these external influences into the modeling. Test constants, usually design constants, are used as input for the test, calibration, and performance software. They are not within scope of this thesis because these constants are input data that do not fluctuate without human intervention.

(-) Background information: Due to the time limitation on a thesis project, it is not possible to get familiar with the physical meaning of the machine constants, their values and the technical relations between machine constants. It may be possible that interesting information in the data cannot be interpreted without this knowledge and therefore the suggestion that machine constants can be useful indicators of machine status could be falsely rejected.

3.4 Literature

An overview of the literature review that has been conducted for this thesis is available in chapter 12.

3.5 Deliverables

The main deliverable is the analysis of the machine constant data and a conclusion on the applicability of that data for CBM purposes. It was a serious possibility that the research into the application of machine constants led to a rejection of the suggested hypothesis, and that the main deliverable would become an advice on which direction to continue the search for prognostic and diagnostic tools for machines in the middle part of the bathtub curve. For this reason, the thesis deliverable can be described as the total roadmap from start till end with, all dead ends, described in detail. Preferably the deliverable consists out of a useful tool for diagnosis and prognosis of machine status.

Tooling Deliverable

An additional part of this research project is to investigate the application of certain tooling to the dataset. The goal of this investigation is to find out the pros and cons of several software packages available and select an approach that enables non- data experts to make use of the information stored within the data in a rather simple way.

3.6 Roadmap

This paragraph describes the different paths and directions investigated during this thesis.

Select Research Approach

There are two different possible approaches for solving the business problem, depending on the order of acquiring external background information and developing a diagnostic tool. One option is to start with examining the requirements of the final solution and the possible methods and tools that may be used to achieve the thesis goal. This approach is called *Knowledge Driven* datamining. By collecting addition information about the data (how/why/when is it collected, and what does it represent), solution requirements, and the purpose of the tool in solving the business problem, it is easier to navigate through the CRISP-DM framework (Wirth & Hipp, 2000). With a better understanding of the direction of a method towards the goal, dead-ends are easier identified and then cut off. The downside of this approach, shaped by expert knowledge, is that creative and innovative directions are closed off from the solution space due to the predefined directions.

With the second option the approach starts with analysing the data without any pre-knowledge. This approach is called *Data Driven* datamining. The output of the data analysis is then related to the additional information about the data. The advantage of this approach is that no direction is being neglected beforehand and any unexpected information may possibly be found. The downside of this approach is that it is difficult to supervise the progress towards a final result and to abort unfulfilling directions in time. The decision between both approaches depends on the preference for efficiency control or creative freedom. It is noteworthy to mention that a data driven approach is currently being used in the field by the engineers. They encounter a problem, see something in the data and try to solve it via trial and error.

Abandoning the open-mindedness may especially have a negative influence for this business problem, which is rather new and undefined. On the other hand, not defining the solution space may lead to difficulties due to the high quantity of different data-mining methods, which are closely related and suitable for almost similar problems. This relates to the issue concerning the time restriction related to this master thesis. It is impossible to examine and apply the whole variety of data-mining methods within the given time. Taking everything into account, the decision was made that the most appropriate approach for this business problem was to first start with collecting the background information and defining a clear goal for the thesis.

The intended approach was then obstructed by the fact that this background information was hard to obtain and/or understand. Machine constants relate to every single module of the machine, while knowhow about each single module is spread over several departments and even more individual people. Therefor it is not possible to understand and explain general behavior of machine constants, before starting the data mining process. The specific and complex behavior requires too much expert knowledge to acquire in a short period of time. Due to the time restrictions of this thesis it was unjustifiable to await the expert knowledge. Because clear answers regarding machine constant knowledge kept out, the preferred expert driven approach evolved into the open data-mining approach.

- Research Approach: Knowledge Driven → Data Driven

Select the Data to Analyse

The first two suggested data sources were machine parameter values and machine constants. Parameters are already used to monitor the machine status, but are unable to assist in the analysis of the particular part of the machine life cycle where a low random constant failure is applicable. Data about machine constants was therefore selected because it was a rather undiscovered area of machine data. General assumption was that machine constants should be related to machine status, because machine constants are (indirectly) related to physical attributes in the machine.

At the beginning of the thesis research the complete original machine constant dataset was part of the scope. At first, the scope of the data was further reduced to increase its homogeneousness. To do so, only a part of the existing machine types (NXT, NXE, XT, and AT) was selected. Later on it was recommended to further reduce the number of machine types to only the NXT machines, because of the lack of similarity between different machine types. Other filtering criteria like the variance, sequence number, and data format were also added during the first reduction of scope. See paragraph 7.2 for the argumentation of the adaptation of scope and the filtering criteria.

- Data Type: Machine Constants & Machine Parameters → Machine Constants
- Machine Type: All machines → NXT, NXE, XT, AT → NXT

Select Tooling and Hardware

Having no pre-knowledge in the field of data mining, selecting and getting familiar with the right tooling, for this business problem, was an important step of the thesis. Several tools like Weka, KNIME, SPSS, MATLAB, and RapidMiner were investigated. As described in paragraph 5.2, it was decided to focus on RapidMiner. The combination of large functionality (e.g. Weka & R implementation), a strong community and being an open source freeware supported the decision to select RapidMiner instead of any of its competitors.

The first plan was to perform all data preparation, modeling and evaluation steps on a non-ASML device (TU/e laptop), but due to high calculative requirements and ASML data security reasons it was decided to perform the CRISP-DM steps on two desktop computers. One performs as server (with the use of MySQL), while the other executes the analysis (with the use of RapidMiner). Using ASML equipment would also provide the extra advantage of having the environment set up for future research or application with RapidMiner and MySQL.

During the start-up phase of the thesis, in which the understanding and explanation of machine constant behaviour was still taking place, the visualization of data was explored. With the use of SpotFire (see paragraph 5.2) it was possible to visualize the data in several different ways. While exploring the many possibilities of SpotFire there could be concluded that it is a great tool to provide insight into a dataset and useful visualization options for reporting result, but that the machine constant data is to disperse and heterogeneous to be able to extract information from the visualization. Without the pre-knowledge of machine constant behaviour, even with the great possibilities of SpotFire, extracting useful information from the dataset is like looking for a needle in a haystack.

- Tooling: Weka, KNIME, SPSS, MATLAB, RapidMiner, MySQL → RapidMiner & MySQL
- Suggestion: SpotFire

Statistical Method

Largely influenced by the selected research approach, the method of analysis also evolved as the research proceeded. At the start of the thesis the main idea was to identify interesting patterns within the dataset, but because the definition of 'interesting' was unknown (due to the lack of expert knowledge available) the focus moved towards identifying abnormal machine behaviour. Abnormal machine behaviour is defined as behaviour that is different from the general behaviour of the machine population. If abnormal machine constant behaviour would indeed be related to machine status, this could become the 'interesting' pattern to look for. See paragraph 6.2 and 6.3 for more details on machine constant behaviour.

Pattern recognition (see paragraph 12.5) is a rather new, vastly growing and promising field, which made it an interesting field to investigate for ASML's business improvements. Methods like Artificial Neural Networks, Self-Organizing Maps, Statistical Classification K-Means (and X-means), Generalized Sequential Patterns, and Classification and Regression Trees (CaRTs) were studied and considered as possible pattern recognition algorithms. As the focus of the thesis changed from pattern to anomaly recognition, the following problem turned up:

Cluster Algorithms are not easily applicable in finding outliers, because they focus on separating objects into different clusters. No matter how strange an object is, it will always be assigned to any of the clusters. Only if poor machines show exactly the same strange behaviour, significantly different from the population of machines that are fine, a complete cluster of poor machines could appear. If this is the case, the algorithms could definitely add value as a statistical warning tool. If not, an extra outlier detection algorithm is required. Only in case of the X-means clustering algorithm (Cluster-Based Local Outlier Factor), such an anomaly detection tool was available (within RapidMiner). Except for the X-means and Generalized Sequential Patterns, all other pattern recognition algorithms are only described in theory (see paragraph 12.5) and not applied to the ASML business problem. Putting extra effort into finding (or developing) these outlier detections algorithms for any of the other pattern recognition algorithms seemed unattractive, because the machine constant data itself lacks quantity of homogenous data per time stamp. Generalized Sequential Patterns was not used to detect anomalies, but to identify a possible pattern leading up to an anomaly. This algorithm was applied after anomaly detection had already taken place.

With the change of focus, from pattern identification towards anomaly detection, a different field of research could be used to assist in solving the research question. The classical field of Statistical Process Control provided a framework that is easy to use, and well validated in practice (see paragraph 12.7). With the use of RapidMiner the results of several different statistical tests (8 sub models, including two X-means, described in paragraph 8.1) could be combined to find those machine constants that were behaving significantly abnormal.

Apart from the above mentioned paths taken, there was an initial interest in the relation between machine constants and the calibrations and tests executed after a swap. As described in the final part of paragraph 3.1, the focus was on trying to decrease the required time after a breakdown, without decreasing the overall reliability of the machine after it is up and running again. Because this would indeed be beneficial for ASML and its clients, this direction was shortly investigated. The reason why it moved out of scope of this thesis is because there already is a tool related to machine constants and after repair instructions, called the Automated Recovery Engine (ARE). While the role of machine constants is somewhat covered with a layer of instructions for the engineers and relational matrices used by the tool owners, it is known that ARE bases its advice on what tests and calibrations to execute on the underlying machine constants and its relation. It was impossible to extract this information for implication in this research, because of confidentiality restrictions and the difficult to extract layered information structure. Because ARE is a diagnostic tool, it does not solve an important part of this thesis regarding the implication of machine constant information into CBM.

- Statistical Method: Pattern Recognition → Anomaly Detection (via SPC)

VALIDATION METHOD

The final part of the thesis was to validate the hypothesis made with the results found via the customised SPC model. The initial planned approach was to directly validate the relation between outlier and upcoming down by expert opinion. This was later on adapted because it is rather difficult to indicate if a certain outlying machine constant could be the cause of a failure within any of the sub modules, and even harder to state that it also is the cause. Together with the fact the machine constant is spread across many different people it was decided to first do a statistical test (see paragraph 4.3) on the relation between outliers and failures, before bothering engineers with vague results. The input files for statistical analysis that were considered are the '*error eventlog'*, the '*MC eventlog'* and the '*Interrupt overview data*' (see paragraph 5.1). It was decided to select the '*Interrupt overview data*', because the '*error eventlog'* contained to much irrelevant data about all activities in and around the machine (which would be too difficult to filter) and the '*MC eventlog'* had some issues with the consistency of the data. The '*MC eventlog'* could be incomplete due to data storage restrictions, restore backup issues and relocation of data.

Evaluation the diagnostic value of the created model went according to plan and did not require any change in validation method. Evaluating the prognostic value of the model provided information required revision of the statistical model and the validation process itself. Four different trials were performed to build a strong conclusion about the prognostic value of the model. Each of the trials was based upon the knowledge and reasoning obtained during the previous trials, the diagnostic validation and this exact roadmap followed. Because of the lack of statistical evidence found, the final step of expert validation was never executed.

- Validation: Expert Validation→ Statistical Validation

Conclusion

An important part of this thesis was to describe the roadmap that was taking throughout the thesis. The reason for this is that both research question and the data to be analysed were rather undefined and unknown. The roadmap gives not only an overview of how the final result was achieved, but also describes all the dead ends that were investigated. This may help to prevent duplicate research in the future. Based on the obtained knowledge several different future recommendations are given in chapter 11. Figure 5 gives an overview of the final structure of the research design used throughout the thesis and is strongly based upon the CRISP-DM methodology. Note that the roadmap itself is symbolized by the learning cycle within the figure. Because the figure is an overview of the whole research, details are also spread throughout the whole report.

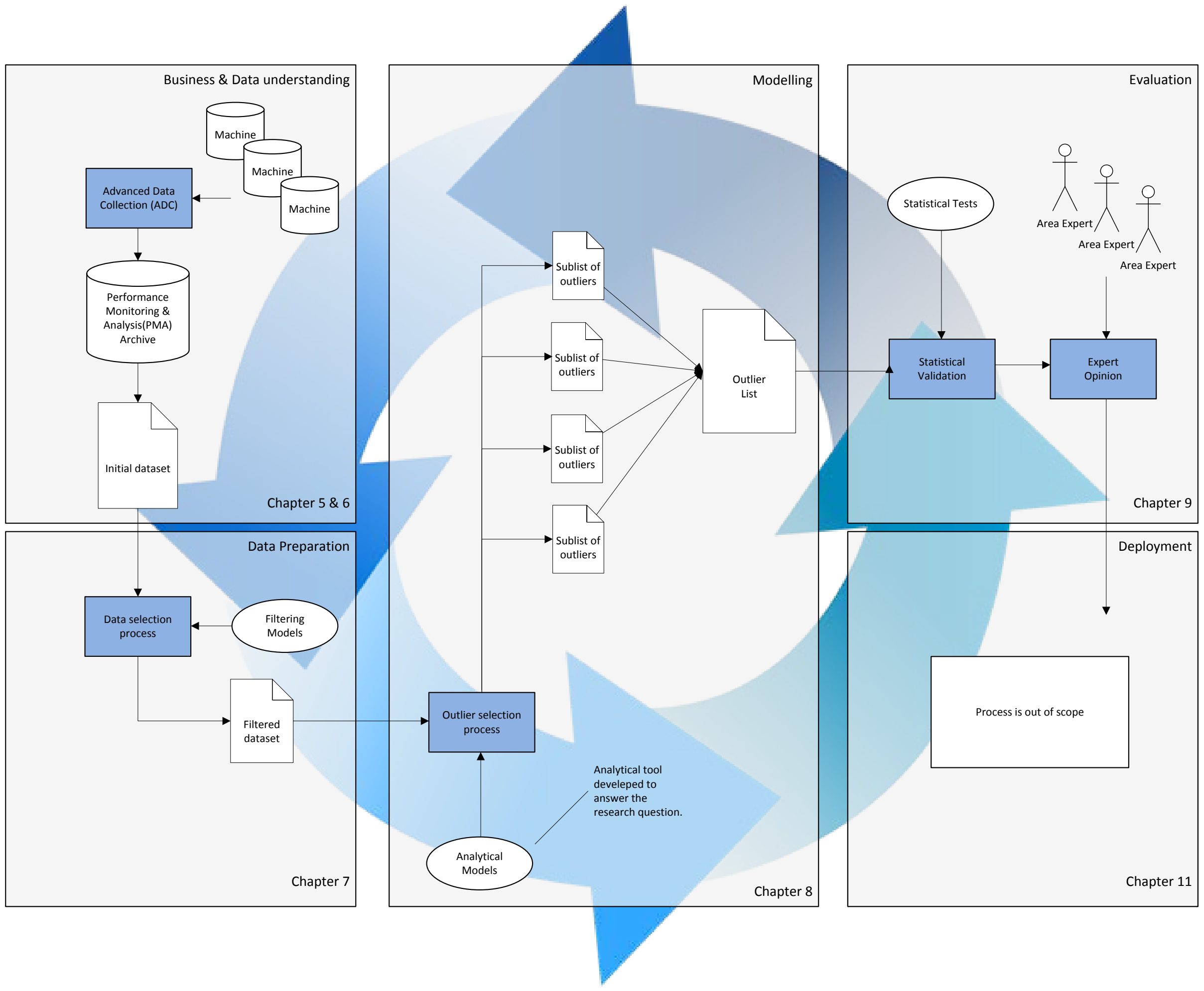


FIGURE 5. RESEARCH DESIGN

3.7 Planning

Based upon the Cross Industry Standard Process for Data Mining method (CRISP-DM), a total time duration spent on a subpart of the CRISP-DM method was made (see Table 1). Because the subparts were not executed subsequently but via several iterative steps, no indication is given of which parts were done during which time periods. The planning is kept in the thesis report, because it provides an indication on the required time per subpart. Contrary to common thought, not the modeling itself but the preceding subparts are the most time consuming. For the final part of the thesis a more detailed planning could be given (see Figure 6 and appendix 4).

TABLE 1. PLANNING

Activity	Duration
Business understanding,	4 weeks
Data understanding	4 weeks
Data preparation	6 weeks
Modeling	3 weeks
Evaluation	2 weeks
Deployment	2 weeks
Thesis completion, Defence	3 weeks
Total time	**24 weeks (week 8 till 33)**

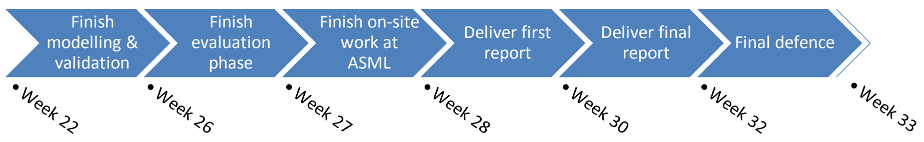


FIGURE 6. DETAILED PLANNING

4 METHODOLOGY

The methodology chapter gives an overview of the methodology used during the thesis. First the thesis approach is described (4.1). Next, the fundamental idea behind the modeling is discussed (4.2). Finally, several methods for qualifying a model are explained (4.3).

4.1 THESIS APPROACH

The theoretical framework which was the red line throughout the thesis is known as the Cross Industry Standard Process for Data Mining (CRISP-DM) (Wirth & Hipp, 2000). This model describes an approach for data mining commonly used by expert in this area. CRISP-DM breaks the process of data mining into six major phases; Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. These different phases do not form a strict sequence and even repeating steps several times may be necessary. Figure 7 displays the general structure and flow of the CRISP-DM model and the implementation of other parts of the methodology. The outer circle in the diagram represents this cyclic nature of Data Mining itself. See paragraph 12.3 for a detailed description of the sub parts of the framework.

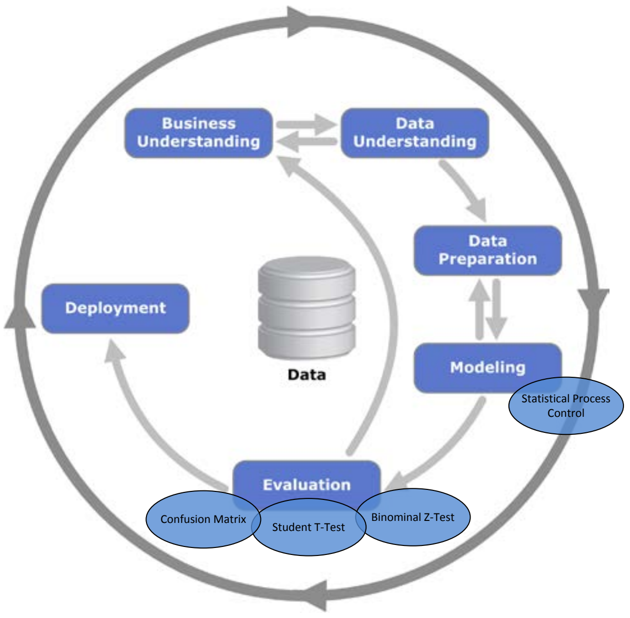


FIGURE 7. CRISP-DM FRAMEWORK AND METHODOLOGY

4.2 STATISTICAL PROCESS CONTROL

Statistical process control (SPC) is the application of statistical technics for tracking and controlling a process, meant for achieving and maintaining a highly efficient business process. If a process is declared efficient depends on the quantity of conforming products it can make (throughput) and the amount of waste (rework or trash) it produces doing so. SPC emphasizes on early detection and issue prevention instead of corrective action after the problem has already occurred (preventive versus corrective maintenance). To do so, SPC predicts significant process variation that may result in future process issues. SPC is the main methodology for the modeling part of the CRISP-DM framework (see Figure 7).

SPC uses control charts to monitor a process. The purpose of the control chart is to allow for simple detection of behaviour that is an indicative of actual process change. Due to the fact that most process characteristics continuously vary, the value of the control chart method is to distinguish unusual from normal variation (Denning, 1987). To define unusual behaviour, control charts use control limits. These upper and lower limits are the boundaries for which the process is considered under control. As long as process values are within these boundaries, there is no reason to suspect process problems or apply corrections and changes to the process control parameters. The boundaries provide a confidence interval for acceptable normal behaviour. See paragraph 12.7 for a detailed description of SPC and control charts.

Control charts should be customized on several points to fit each unique problem and provide the best advice to the process owner. These customizable settings are the selection of a confidence interval, warning limits, and the number of outliers required to label a process as out of control. The general selection of a confidence interval is largely based upon the Central Limit Theorem. This theorem states that for a sufficiently large sample drawn from any distribution (in which variance exists), the resulting distribution of the mean becomes normally distributed. For a normal distribution it is rather easy to calculate the confidence interval based on the selection of the upper and lower limits. A common setting for the control limits is the mean plus or minus three times the standard deviation (Denning, 1987) (Oakland, 2003) (Evans & Lindsay, 2010). With the Shewhart 3-sigma limits, 99.73% of the observations occur within three standard deviations of the mean. Besides the control limits, it is also possible to add warning limits to the charts. While they do not predict that the process is currently out of control, they signal that the process is behaving suspiciously and a high level of alert could be preferred. Typically, warning limits are placed at two standard deviations around the mean.

Control charts typically have the following mandatory elements to serve as guides for control activities (Evans & Lindsay, 2010):

- A central line that designates the estimated mean of the process characteristic under control;
- An upper control limit (UCL) line, which designates an outrebound that would rarely be reached by a plotted point if the process is in control. Classically, this line is designated as the "three sigma" line and stands three times the estimated process standard deviation above the central line;
- A lower control limit (LCL) line, which usually stands below the central line by a distance equal to the upper control line, usually about three times the standard deviation.

Formalizing the above described characteristics of the control chart leads to the following formulas (Oakland, 2003):

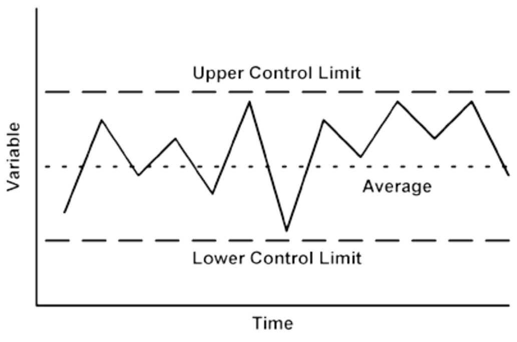


FIGURE 8. SPC CONTROL CHART

$$Upper\ Control\ Limit = \mu + k \cdot \sigma$$
$$Lower\ Control\ Limit = \mu - k \cdot \sigma$$

$$where\ \mu = population\ average,$$
$$k = significance\ factor\ (usually\ equal\ to\ 3),$$
$$and\ \sigma = population\ standard\ deviation$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}\ ,\ where\ \mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Central Limit Theorem

An important fundament of the SPC control chart is the central limit theorem. The theorem states that, given certain conditions (well-defined variance and mean), the mean of a sufficiently large number of independent random variables will be approximately normally distributed (Rice, 2007). Under some conditions, it is not even necessary for the distributions to be identical to approximate the normal distribution. By the law of large numbers, the accuracy of the approximation improves with an increase in sample size. The theorem can be formalized as follows:

The Lindeberg-Lévy Central Limit Theorem states that; suppose $\{X_1, X_2, \dots\}$ is a sequence of independent and identically distributed random variables with $E[X_i] = \mu$ and $Var[X_i] = \sigma^2 < \infty$, then as n approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converge into a normal distribution $N(0, \sigma^2)$ (Oakland, 2003).

Approximation formula:

$$\lim_{n\to\infty} \sqrt{n}\left(\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) - \mu \right) = N(0, \sigma^2)$$

4.3 Model Qualifying Criteria

During the evaluation phase, it is important to have a clear definition of what makes a model successful (see Figure 7). The main objective of the model should be the correct prediction of future machine status (prognostic value). A model can diagnose that a machine looks strange, but does not know what it will do in the future (so called *bad-weather* status), or it can predict that something will go wrong, but not conclusively when and why, or a model can pin-point the exact reason that a machine will break down. While an optimal model does the last, less concrete predictions can still be valuable. Besides the prognostics value of a model, also the diagnostic value after breakdown is interesting. High diagnostic value is achieved if a model is able to pin-point the reason for a breakdown after it already occurred. Both values can be evaluated with similar tests.

To prove the statistical quality of the model three different methods were used. The Binomial Z-test and student T-test can be applied to test if there is a significant difference between an example set with the time between a warning and breakdown, and a second set with the time between random a point in time and breakdown. In case that the difference between these two sets is significant and the warning set shows a lower mean until failure, it can be concluded that the model has predictive value. The exact accuracy of the model can then be calculated with the use of the confusion matrix, which evaluates the amount of correct and wrong predicts that a model makes.

Binomial Z-Test

The first test to qualify models is a simple test of sample means. This Z-Test compares the number of successes found in each sample to state if there is a significance difference between those means (Montgomery & Runger, 2007). The binomial test can only be used if the data is dichotomous, which means that the data can be classified into two categories, and if the number of examples in each category is at least larger then 10 $(N_x p_x > 10 \wedge N_x(1-p_x) > 10, \forall x \in 1{,}2)$. The larger the sample size the more likely it is to find a significant difference (if indeed existing). See Table 2 for an overview of the practical implementation of the binomial Z-test. A commonly used level of significance (2-tail) is 5%, which leads to a confidence region of $-1.96 < Z < 1.96$. If a Z value does not find itself within this region, the null hypothesis (*H₀*) should be rejected.

TABLE 2. BINOMIAL Z-TEST

	Sample Size	# 'success'	Sample Proportion
Group 1	N_1	D_1	$N_1/D_1 = p_1$
Group 2	N_2	D_2	$N_2/D_2 = p_2$

$$Z = \frac{p_1 - p_2}{\sqrt{(\hat{p})(1-\hat{p})(\frac{1}{N_1}+\frac{1}{N_2})}}$$

Pooling the data, we obtain $\hat{p} = \frac{D_1+D_2}{N_1+N_2}$

Null hypothesis (*H₀*): $p_1 = p_2$

Student T-Test

A more complex test for the difference between two samples is the student T-Test (Montgomery & Runger, 2007). The student T-Test is a statistical hypothesis test that can be used to determine if two sets of data are significantly different from each other. The test statistic follows the student's T distribution if the null hypothesis is supported, which is based upon the average value and variance of both samples (variance is not included in the Z-Test). The test can be customized for several different datasets (equal/unequal sample sizes and variance for independent, paired and overlapping samples), but the only types applicable to this thesis are independent samples with equal variance. In this case the *t* statistic tests whether the means of the samples are significantly different. Once a t value is determined, the associated p-value can be found in the Student's t-distribution table of value. If the corresponding p-value is below the threshold chosen for statistical significance (generally 0.05), then the null hypothesis (*H₀*) should be rejected. The student T-test requires the two populations to follow a normal distribution, which can be tested with normality tests, such as Kolmogorov-Smirnov, Shapiro-Wilk, or a visual assessment via the histogram. The following formula is used to calculate the *t* statistic for both equal and unequal sample sizes (substitute n for n_1 and n_2):

$$t = \frac{\bar{X}_1 - \bar{X}_2}{S_{X_1X_2}\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}}$$

$$S_{X_1X_2} = \sqrt{\frac{(n_1-1)S_{X_1}^2 + (n_2-1)S_{X_2}^2}{n_1+n_2-2}}$$

Where:

$\bar{X}_x = Average\ value\ for\ each\ sample.$

$$n_x = number\ of\ entries\ in\ each\ sample.$$

$$S_{X_x}^2 = the\ variance\ of\ each\ sample.$$

$$S_{X_1X_2} = the\ common\ standard\ deviation\ of\ the\ two\ samples.$$

$$n_x - 1 = the\ number\ of\ degrees\ of\ freedom\ for\ each\ sample.$$

Confusion Matrix

To qualify the result of a model it is important to statistically verify the theory behind the model. To do so, a statistical test considers the accuracy of the predictions. Hypothesis testing is an integral part of testing this accuracy. The rest requires an unambiguous definition of the null hypothesis (H_0), which usually represents the default state of the situation, person, object, or anything else analysed ('the machine is in a fine state; up and running'). The alternative implication of the null hypothesis is the unnatural state ('the machine is in a poor state; broken-down'). A model predicts if the null hypothesis should be rejected or not. If the prediction corresponds with reality, then the outcome is considered correct. However, if there is a discrepancy between the prediction and reality, the outcome is considered an error. The fact is that (almost) no model or test is free of error, due to the statistical nature of research. The error can be separated into two different types: type I and type II error. In the field of machine learning it is common to visualize the performance of a model via a confusion matrix (see Table 3). The columns represent the predictions that the model does and the rows represent the actual state manifested in reality (Stehman, 1997).

TABLE 3. CONFUSION MATRIX

	Null hypothesis (H_0) is true	Null hypothesis (H_0) is false
Reject null Hypothesis (H_0)	Type I error False positive	Correct outcome True Positive
Fail to reject null hypothesis (H_0)	Correct outcome True negative	Type II error False negative

Type I Error

A Type I error is defined as an error in which the null hypothesis is true, but is rejected by the test. Based on something that is not there, a hit is reported that incorrect. In case of a single condition test, the Type I error is similar to a false positive (or false alarm). The rate of the type I error is denoted by α (*alpha*) and represents the size of a test, usually equal to the significance level. Except for a composite null hypothesis test, α (*alpha*) is the probability of a type I error.

Type II Error

A Type II error is defined as an error in which the null hypothesis is false, but is not rejected by the test. The model fails to recognise what is present and therefore neglects the rejection of the null hypothesis and fails to believe in the truth. In case of a single condition test, the type II error is similar to a false negative. Where the Type I error uses alpha to denote the rate of error, the rate of the Type II error is represented by β (*beta*). In case of the Type II error, the rate is related to the power of a test (*1*- β). The power of a test represents the probability that the test is able to reject the null hypothesis (H_0), if it is indeed false.

The definition of the null hypothesis influences directly what kind of behaviour is called a type I or II error. One should keep in mind when defining the null hypothesis that a model can either reject or fail to reject the null hypothesis, but can never prove it to be correct. Failing to reject the null hypothesis does not provide any evidence to prove that the null hypothesis is true, only that it is not statistically false.

This theoretical framework was implemented into the thesis scope, to determine if the model (based upon the data mining procedure) is useful (see Table 4). The accuracy of a model is defined as the proportion of the results that reflect the actual real-world status (predictions that are no type I and II errors). Because the purpose of the model should be the recognition of machines that will receive issues in the future, the null hypothesis (H_0) will represent all machines that are in a normal state and that are running fine. The model analyses the null hypothesis (H_0) and predicts if it should be rejected or not. Via validation of the actual machine state (future interrupts and downs), the number of type I and II errors and the accuracy of the model can be obtained.

Noteworthy is that neglecting a poor machine status is most likely less desirable than giving a false alarm on a machine that is running normal. This is due to the fact that false alarms bring less cost than an unexpected machine down. Therefore, besides the accuracy as a percentage also the spread of the accuracy over 'normal' and 'poor' should be reviewed to decide upon the successfulness of a model.

The null hypothesis (H_0): the machine is in a normal state; up and running fine.

TABLE 4. CONFUSION MATRIX MC

		Predicted State	
		Normal(H_0)	Poor
Actual State	Poor	Type I error False Positive	Correct outcome True Positive
	Normal(H_0)	Correct outcome True Negative	Type II error False Negative

Accuracy 'poor' = True Positive / (True Positive + False Negative)

Accuracy 'normal' = True Negative / (True Negative + False Positive)

Total Accuracy = (True Positive + True Negative)/ (False Positive + True Positive + True Negative + False Negative)

5 Business Understanding

The first step of the CRISP-DM framework is to understand the business environment in which the thesis takes place. This chapter will start with an overview of what data is collected and stored and how it is done (5.1), than several business software packages are discussed (5.2), and finally an overview of the whole data structure is given (5.3). Large parts of this chapter are confidential and are only available in the ASML-only appendix 12.

5.1 Data Collection & Storage

ASML collects data about their machines in the field. The data is processed by IDP (Incoming Data processor) and is finally stored in Paris/Insight and the PMA archive (Performance Monitoring & Analysis) (ASML, Techwiki-ASML, 2013). The PMA archive is an archive, located in Veldhoven, used to store all kinds of relevant data from machines at Veldhoven or customer sites. Data in the PMA supports (any of) its two main purposes (ASML, PMA archive-Techwiki-ASML, 2013):

"Provide a historic overview of all kinds of aspects of steppers, scanners and yieldstar. This can be used by customer support to analyse problems in the field and by developers to learn about the behaviour of their functions over time and for multiple machines."

"Allow developers to analyse certain aspects over multiple machine(type)s. This is especially of interest for new developments. Therefore, with a good plan for analysis, it is in line with the intentions of PMA to request more data to be collected during a limited period of a half or a whole year."

Due to the limited capacity, the storage time of the data is also limited. Where meaningful, data is kept in the PMA archive for a sufficient length, depending on the storage capacity that is present and the actual amount of data per machine. Most data lacks such long term relevance and is therefore only stored shortly.

A detailed description of this process can be found in ASML-only appendix 12.1.

Constants

There are two types of constants, design and calibration constants. Design constants typically hold a fixed default value that is set at the design time, while the calibration constants are determined on the machine, done via a calibration, specific upgrader flag, or a setting done by the operator. Besides these two constant types, constants can also be categorized into one out of four different categories of constants; Test constants (TE), Job constants (JC), Machine constants (MC), and Factory constants (FC) (ASML, Techwiki-ASML, 2013). . Despite the fact that they are called constants, their value does not necessary stay constant, but may change over time due to calibrations.

Test constants, usually design constants, are used as input for the test, calibration, and performance software.

Job constants differ from Job, Machine, and Factory constants, because they do not represent any of the settings that can be influenced by ASML's machines.

Machine constants, which consist of both design and calibration constants, are used by the software to compensate for differences in physical dimensions, compensate for drift, and to select configuration related items. There is a set of machine constants for each hardware subsystem and for some of the software components. All machine constants relate to a physical attribute within the lithographical machine and influence the three basic KPI's of ASML's machines; Overlay, Imaging and Throughput. Overlay is a measure of the machine's ability to print layers accurately on top of each other. Successive layers or multiple processes on the same layer must be accurately aligned to the previous layer to achieve the required electrical contact between structures. Without accurate alignment the resulting chip will not perform to specification or may not work at all (ASML, Papers, 2013). Imaging can be defined as the resolution of the structures of the printed layer. A higher imaging quality allows smaller structures with accurate edges in exactly the intended shape. Throughput has the same meaning as in most companies worldwide and represents the number of usable wafers (or chips) that a machine can produce in a certain time interval. Increasing Overlay and Imaging quality, by spending more time on testing and calibrating, usually decreases the throughput. Machine Constants are stored in separate files for each recognized subsystem, which makes subsystems of Machine Constants basically independent of each other (ASML, Techwiki-ASML, 2013).

The PMA archive does not store all Machine Constants. These MCs are marked hidden and are not editable by the clients, because there is no reason for modifications by the non-experts. This is especially applicable for rather complex machine constants that are highly interrelated. For example, MC's related to the mirror alignment are considered highly interrelated. A large number of constants work together to optimize the alignment of the mirror.

Factory constants are similar to, but differ from machine constant in the fact that factory constants represent the influence of customer decisions on the settings of the ASML machine (for example, the edge clearance).

Regular Interval Calibration & Maintenance Actions

To keep the machines running successfully it is important that machine constants stay within its tolerable range of values. Because it is known that values may change over time, regular interval events are planned to recalibrate the machine constants. Software component MX (M-actions scheduler) provides functionality to execute calibration and/or maintenance actions at regular intervals.

A detailed description of the MX activities can be found in ASML-only appendix 12.1.

MC Event Log

The MC_event_log is a file where most changes in Machine constants are logged. Although this file seems very useful for finding the MC-history of a machine, there is the problem that not all changes are logged here. For some machine constants the changes are never logged, while others experience issues when, for example, the machine constant file is restored (either using the Twinscan-SW or manually) (ASML, Techwiki-ASML, 2013). For this reason, this data source is not is not usable for valid research opportunities and the machine constant change data should be retrieved directly from the PMA.

5.2 Tools

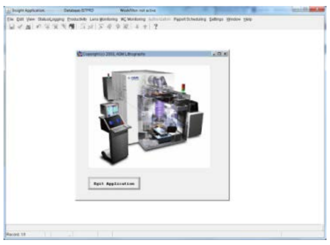
This paragraph describes the tool used during the thesis. For non-ASML tools the applicability for business cases will shortly be discussed. The desktop computer on which these tools ran during the thesis holds a Intel(R) Xeon(R) CPU, W3565 @ 3.20 GHZ, 3.19GHZ processor with 12.0 GB installed memory (RAM) on a 64-bit Operating System (Windows 7 Enterprise).

W2in

W2IN (World Wide Information Navigator) is ASML's reporting tool for PMA and OpEx, and is based on Oracle software called Oracle Business Intelligence Enterprise Edition (OBIEE). W2IN was used in this thesis to extract the business relevant machine interrupt information for validation of the prognostics value of models. A detailed description of the W2in can be found in ASML-only appendix 12.2.

Insight

Insight Application is another ASML reporting tool that uses PMA to provide people with machine information. This java language based tool does not allow extracting data in the way W2IN does, but is oriented towards retrieving detailed machine information for individual machines. For this thesis the option in the reporting tool to identify the reconciled machine state at a particular moment in time (done via the productivity tab) is the only useful mode out of many. Because this search needs to be done manually, and searching in PMA is not that fast, it was rather time consuming to add the machine state at a particular moment to the dataset. Insight uses a number code to describe the state of the machine. Additional numbers can be added to the code to further specify the exact state of the machine. The normal up-state of the machine can be defined as an alteration between productive and standby states. All other states will be defined as down-state. A detailed description of the Insight states can be found in ASML-only appendix 12.2.



RapidMiner

This tool provides an environment for data mining, text mining, machine learning, and business & predictive analytics. RapidMiner®, formerly known as YALE (Yet Another Learning Environment) is used for a wide variety of applications ranging from education to industrial applications and ranked first on usage of data mining tools for real projects in 2010 (Mikut & Reischl, 2011). RapidMiner is able to program and execute processes such as; data loading & transformation, data pre-processing, visualization, modeling, evaluation, and deployment. Build upon the Java programming language, RapidMiner makes use of the functionalities of Weka machine learning environment and the R-Project and additional functionalities can be added by plugins. RapidMiner provides a graphical user interface (GUI) to design analytical processes that generate a XML (eXtensible Mark-up Language) file that defines the analytics that the user wishes to apply to the data. Another interesting aspect of RapidMiner is its support of the PMML (Predictive Model Mark-up Language) standard, which allows the export of models designed within RapidMiner to other systems and applications. Therefore, the results of simulations executed in RapidMiner can directly be reproduced in other PMML compliant systems (Rapid-I, 2013). All these features together make RapidMiner a strong candidate for data mining purposes (Shafait, Reif, Kofler, & Breuel, 2010). The Community Edition of RapidMiner is an open-source freeware tool. For application of ASML's purpose, RapidMiner's strengths and weaknesses can be summarized as follows:

Strength:

- Intuitive Graphical user interface that provides a good overview of the analytical process.
- Most modeling applications and functionalities are included in the software.
- Open-source freeware.
- Strong online community.

- Already known within ASML.

Weaknesses:

- Error messages are poorly defined.
- Small inconveniences in the interface.
- No pre-knowledge acquired during education.

MySQL

MySQL (Community Edition) is the world's most popular freeware for open source relational database management applications (RDBMS) (MySQL, 2013). The SQL phrase stands for Structured Query Language, while My refers to the daughter of one of the co-founders. MySQL is available under the GPL license and is supported by a huge and active community of open source developers. Besides the community edition there are also several paid editions available, which offer additional functionality. Several of the world's largest and fastest-growing organizations such as Facebook, Twitter, Flickr, YouTube, Google, Adobe, Alcatel Lucent and Zappos rely on those MySQL packages to save time and money powering their high-volume web sites, business-critical systems and packaged software. The standard version of MySQL comes without a GUI tool but allows users to use the included command line tools. There is an Oracle MySQL Workbench available which provides the missing GUI. MySQL Workbench provides DBAs and developers an integrated tools environment for: Database Design & Modeling, SQL Development (replacing MySQL Query Browser), Database Administration (replacing MySQL Administrator), and Database Migration. For application of ASML's purpose, MySQL strengths and weaknesses can be summarized as follows:

Strength:

- Open-source freeware.
- Compatible with and predefined in most other applications
- Already known within ASML.

Weaknesses:

- No pre-knowledge acquired during education.

SpotFire

TIBCO SpotFire is a software platform used to visually analyse data using statistics. SpotFire strength lies in its possibility to visualize large datasets in such a way that difficult to find underlying behaviour can be identified. With ever increasing data storage in almost all market segments, the need to find proverbial needles in the haystack also increases. This is enabled by TIBCO's decision to move from the in-memory data to in-database analysis, together with a rewritten in-memory engine that takes better advantage of high-capacity, multi-core servers. This combination of innovation allows for truly high-volume data analysis (up into the tens or hundreds of terabytes of data) in a way that does not waste time extracting and moving data or handling analyses on under-powered analytic severs. The SpotFire engine is based upon the well-known agile R language, which brings a stable and scalable fundament for the SpotFire software platform. Another interesting rather new feature of the software platform is a dynamic analytic application that runs on the web, called the TIBCO SpotFire Web Player. Other companies that also make use of SpotFire are for example BP, KPN, Motorola, and Saab (Spotfire, 2013).

Strength:

- Easy to use.
- Flexible view settings.
- Already known within ASML.

Weaknesses:

- No pre-knowledge acquired during education.

- Commercial tool (paid).

5.3 Data Structure

In a data mining environment it is important to have a solid structure of storing and accessing data. A common structure is to separate the data itself from the entity that executes the data mining operations. First of all, this makes sure that the complete computational strength can be used for the execution itself instead of storing the data. Another benefit of the separate storage is that it can be stored on a server that can be approached from several different computers at a time. Uploading of the initial dataset (CSV) is done into MySQL via the SQL query language (see appendix 5), after which tools like RapidMiner and SpotFire are able to access the data easily. The information source interrupt overview is uploaded directly into RapidMiner (low quantity of data) and the Machine State information is used manually. See Figure 9 for an overview of the data flow and interaction from collection to analysis.

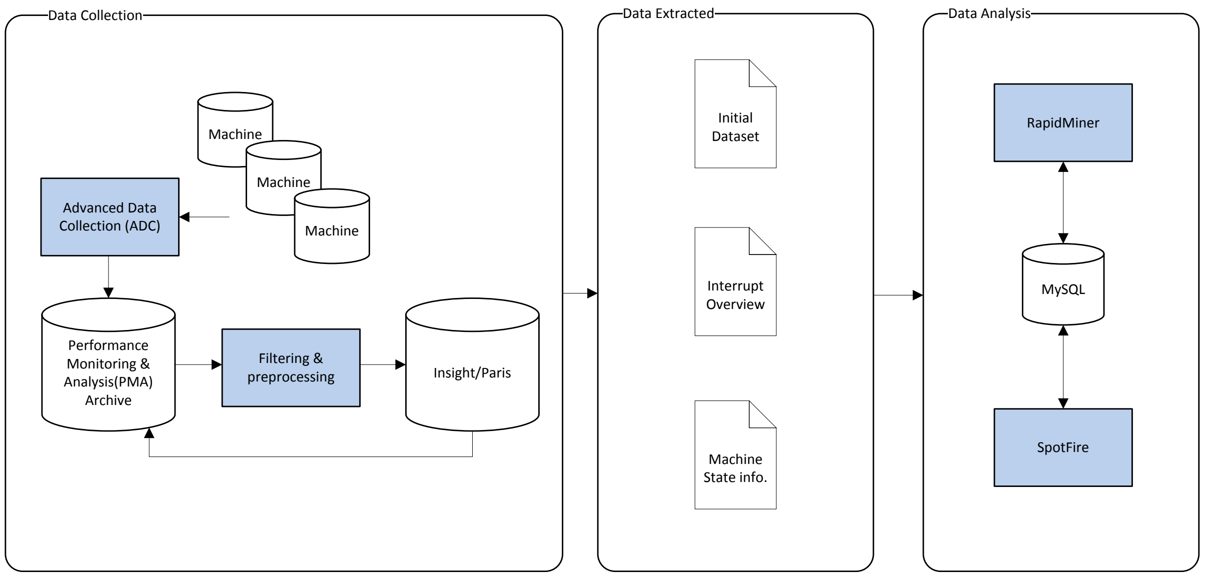


FIGURE 9. DATA STRUCTURE

6 Data Understanding

Following the CRISP-DM framework, an important step in the data mining process is to understand the data. The first part of the chapter (6.1) (6.2) focuses on the input data of the modeling phase. This is done with some basic modeling to summarize the available machine constant dataset and obtain a first insight into the behaviour of the constants themselves. Finally, the interrupt data is described and analysed (6.3). While the initial dataset is used as input of the modeling phase of the CRISP-DM methodology (see chapter 8), the interrupt file is only used during the evaluation phase (see chapter 9). The build-up knowledge was later on used to adapt the content, volume and general scope of the data and help solve the business problem.

6.1 Initial Dataset

After some manual exploration of the original initial dataset a RapidMiner model was used to get an overview of what information is available in the data. The basic analysis retrieves the dataset from the MySQL database and uses some aggregation processes to calculate the basic statistics of the data (count, unique count, missing, average, minimum, and maximum).

The initial dataset consist of the data of 642 machines, for 89309 machine constants, over a period of +-1.5 month (01-Feb-13 till 15-Mar-13). This dataset includes all machines and their constants that were set and stored during the selected period. Due to the high number of machine constants, even a rather small period leads to a large dataset. The benefit of a large dataset is that the possibility of finding interesting results increases, but the drawback is the large calculation power required. The dataset contains background information about the machine constant, time of calibration and the machine itself. See appendix 6 for the process visualization and (XML) program code of the *'basic analysis' process* that was used to analyse the initial dataset. A summary of the given data and a short description of the content of the dataset can be found in appendix 12 (ASML-only).

- 642 machines.
- 89309 machine constants.
- 1.5 months (01-Feb-13 till 15-Mar-13).

6.2 Individual Machine Constant Behaviour

After the general exploration of the data, it was important to review the actual behaviour of machine constants. With the use of basic RapidMiner® processes it is possible to analyse the behaviour of machine constants (in combination with sequence) per machine over time. Several different kinds of behaviour can be found in the filtered dataset; for example random clouds, linear trends, jumps, and outliers (see Figure 10).

Random Behaviour

The most basic behaviour that can be identified in the dataset is a random movement of numeric value over time. This random cloud shape suggests that there is no direction to which the values move. Despite the fact that the term random is used to describe this type of behaviour, this is exactly the behaviour you expect from a machine that is running fine. Especially if the random movement is a small deviation around the mean, this is no reason to assume that the machine is experiencing issues. To give predictions on future values, the average value is the best you can do for this type of behaviour. The prediction can be supported by a distribution of the historic values and its confidence interval. See Figure 10.A for an example of random behaviour of machine constants.

Linear Trends

A second typical behaviour of machine constant values is a linear drift over time. This shape suggests that the value is influenced by time and will eventually move out of specifications. While a linear trend seems rather interesting for future predictions, there are two reasons why this is not necessarily the case. First of all, it is unsure if this would also necessarily lead to an unscheduled down or if this drift can be calibrated back to its begin level, in a controlled manner. Secondly, machines that are in the middle part of their lifecycle do not show wearout behaviour, which a drift over time clearly is. So, while it may be interesting to monitor trends, they are not perfectly suited for the business case. In the initial dataset, there are several machine constants that show a significant trend in value over time (tested with the use of simple linear regression). Interesting fact is that this does not necessary occur on all machines and it is even possible to find conflicting trends (increasing and decreasing) for the same machine constant on different machines. See Figure 10.B for an example of a linear trend for machine constants.

Jumps

Another interesting behaviour is that of a jump. A jump is an abnormal large difference between two successive numeric value examples. Such an extreme change in value (increase or decrease) may indicate that something changed in the machine status. In case a solitary jump is followed by another jump, back to the initial level, the data entry will most likely also be an outlier. The difference between a jump and an outlier is that for a jump it is not necessary that any of the values before or after the abnormal change in values is also an abnormal value in the total population. The other way around, also not every outlier is the result of a jump. See Figure 10.C for an example of a jump in machine constants.

Outliers

An outlier is a single value that is significantly different from the rest of the population. Values that are unusual for a machine constant may be an indication that the machine is also in an abnormal state. An outlier can either be an extremely high or low value. Outliers may be interesting, because they are completely random and definitely not the result of slow wearout. The downside is that these kinds of random behaviour are difficult to predict. See Figure 10.D for an example of an outlier in machine constants.

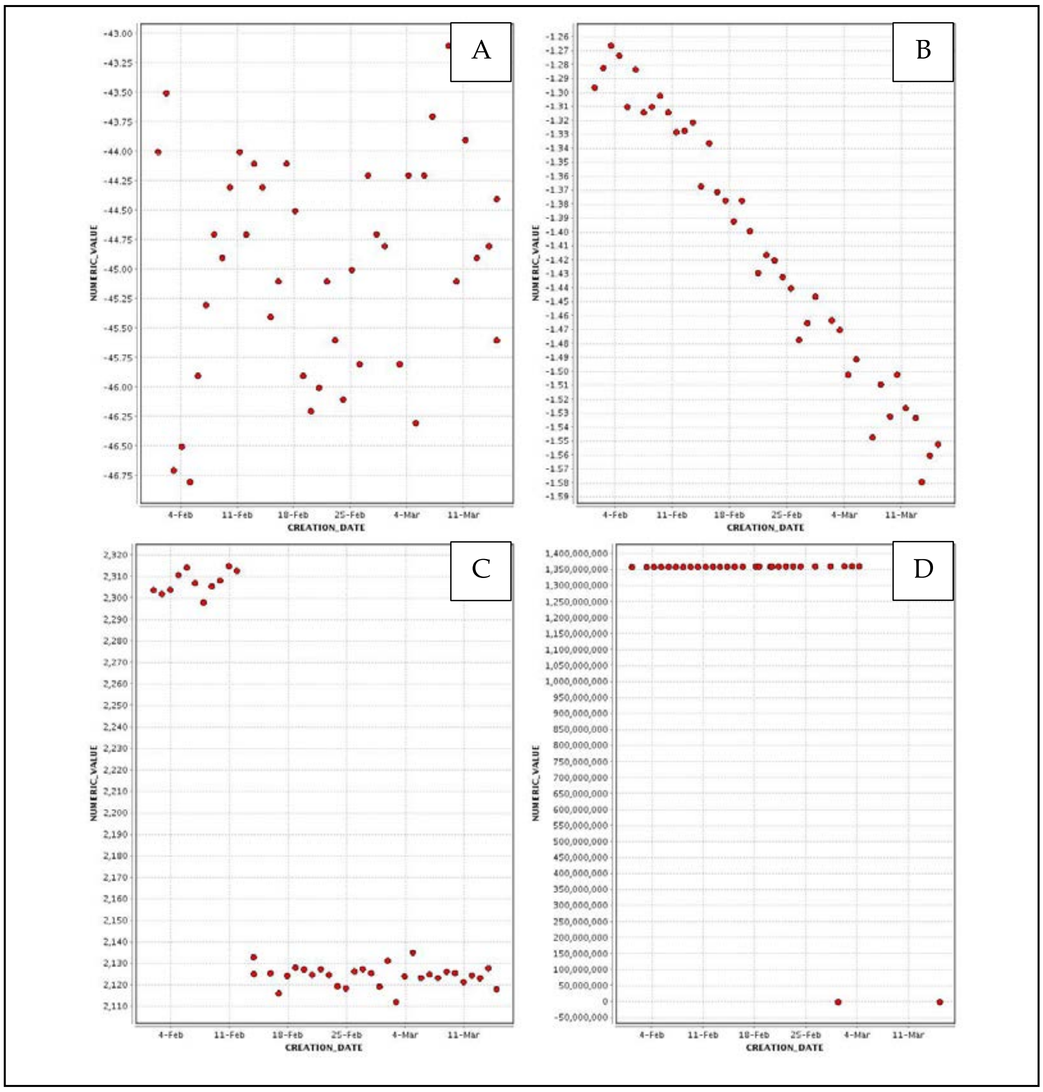


FIGURE 10. MACHINE CONSTANT BEHAVIOUR

Conclusion

For none of the above described behaviours it is clearly identifiable if it is troublesome or not. No knowledge is available to state what kind of pattern can be identified as critical. For example drifting (see Figure 10.B.), is exactly the behaviour you expect from a drift correction machine constant and a jump may just be the result of a regular planned maintenance activity. What can be concluded from this exploration of the data is that several kinds of behaviour are present within the dataset, but none of them can simply (completely data driven) be identified as an indicator of poor machine status. Based on this conclusion, the actual analysis of machine constants should focus on outlying behaviour of machines instead of the behaviour itself. The reason for this is that outlying machine constant behaviour may indicate that a machine is an outlier from the population. Abnormal behaviour (abnormal trends, jumps, or outliers) will therefore be reviewed as an indication of current or upcoming failure. The remaining part of the CRISP-DM cycle (see Figure 5), consisting out of Data Preparation, Modeling, Evaluation, and Deployment was used to validate this statement.

6.3 Population Machine Constant Behaviour

In the previous chapter the machine constant behaviour on individual machines was reviewed and it was concluded that not enough knowledge about general machine constant behaviour could be extracted with this scope. For this reason, the machine constant behaviour of equally names machines constants was reviewed (defined as the population of a machine constant). Figure 11 visualizes the distribution of values of four different machine constant populations, all having more than 50 examples $(N \geq 50)$. These four machine constants are a good representation of all the constants reviewed, and show a clear resemblance with the normal distribution. This recognizable bell shape distribution found in the histogram corresponds with the assumptions of the Central Limit Theorem and allow for the application of the Statistical Process Control (see paragraph 4.2). The fact that the values follow the normal distribution was an important requirement for the future steps of the CRISP-DM approach, especially the application of warning and control limits during the modeling (see paragraph 8.1).

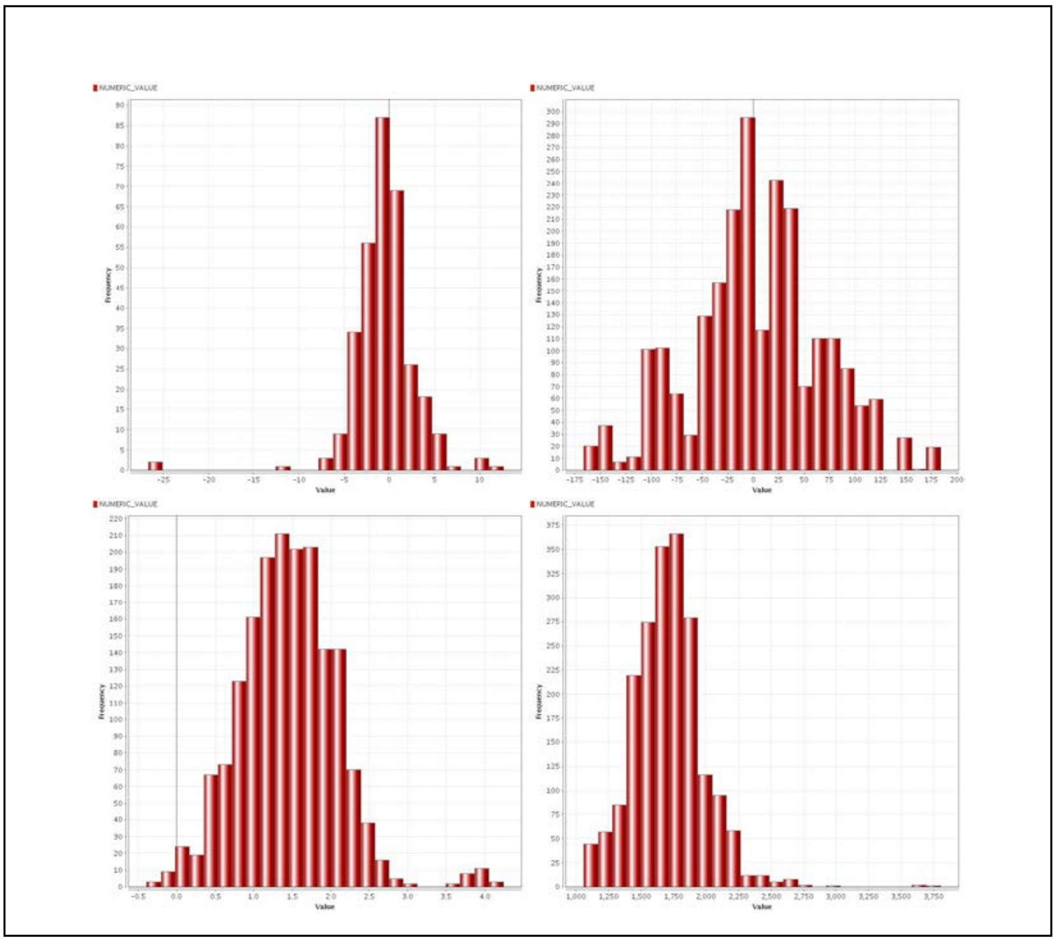


FIGURE 11. NORMAL DISTRIBUTION MC

6.4 INTERRUPT OVERVIEW DATASET

The second important data source used in this research is the interrupt overview data. The interrupt data is used to validate the output of the created model (see paragraph 9) and extracted via W2in. W2in extracts this machine information from the PMA archive, after it is manually verified. Contrary to the initial dataset discussed in the previous paragraph, the interrupt overview is not used for anomaly detection but to analyse the machine behaviour around that moment in time. The interrupt data consist out of a list of all interrupts that occurred during the selected time period, with detailed background information about these interrupts. Despite the fact that there are empty fields within the dataset (for remark, SO, and USD Time), there is no missing data. These empty fields have no missing data, because the fields are optional. The interrupt overview file was linked to the machine constants dataset, so it could be used to review the interrupt behaviour compared to the machine constant behaviour. Due to the outcome of the evaluation of the diagnostic/prognostic models (see paragraph 9.3), most of the detailed information retrievable from the interrupt file was not used. A detailed description of the interrupt overview data can be found in ASML-only appendix 12.4.

7 DATA PREPARATION

This chapter describes the activities to adapt the initial machine constant data in such a way that it can be fed into the model tool, and the interrupt overview data to serve as validation data. The knowledge about the data, discovered during the data understanding phase, can be used to change the given data into the final dataset(s). Data transformation, cleaning, reduction are commonly applied CRISP-DM steps (see paragraph 12.3) that were applied to the initial dataset. This chapter starts with a short explanation about how to handle with missing data and outliers (7.1), after which two data preparation processes will be discussed for the initial machine constant dataset (7.2)(7.3), and finally a short description on preparing the interrupt overview file is given (7.4).

7.1 MISSING DATA & OUTLIERS

A data entry is defined as a missing value if one or more variables are not included in the dataset. In case the amount of missing data is less than 10% of the total data available, and the missing data occurs randomly, the rule of thumb states that the missing data can be removed from the dataset before continuing with the analysis (Montgomery & Runger, 2007) (Hair, Black, Babin, & Anderson, 2009). If any of these restrictions is not met, than the missing data can be imputed by an estimation of the missing value based upon the rest of the dataset and the available information for that individual entry. As seen during the data understanding phase (see paragraph 6.1 & 6.3) the amount of missing data is less than 1%, which allows simple deletion of those entries from the dataset.

While in other researches the filtering of outliers is a pre-processing step, for this thesis the outliers are exactly those entries of interest for the actual data analysis. The fact that it is possible that the outlier is a data entry mistake and not the actual value of the machine constant was a risk to be taken. This risk is mentioned during the discussion of the limitations of this thesis (see paragraph 10.5).

7.2 FILTERING PROCESS

The second process to prepare the original dataset for actual analysis is the '*filtering*' process. It retrieves the whole dataset from MySQL database and starts pre-processing the data by the following editing and filtering steps:

- Renaming machine types to one consistent format (e.g. from NXT1950I to NXT1950i).
- Filter machine type equal NXT (see list of machine codes in ASML-only appendix 12.6).
- Filter sequence number equal to 1.
- Filter data format equal to numeric (N).
- Filter numeric value equal to non-missing.

When analysing data, it is important to make sure that the data analysed also represents what is assumed that it does. This is a serious issue with the data concerning machine constants, because meaning, functioning and basic settings have changed significantly over time due to radical design evolution of ASML's machines. So while the same machine constant may be mentioned in both the earliest and newest machines, assuming that a comparison is allowed is unfounded. For this reason the decision is made to focus only on the NXT machine type. These machines are rather new, have a lot of similarities and are the largest subpopulation in the initial dataset. Apart from the data mining perspective, the economic value of a tool would be the highest for the NXT machine type, because of the population size and leftover economic life time.

It is easier to analyse numeric values instead of text, because of the large variation in text values. Therefore only numeric data is selected. Text mining is a whole different area of data mining and will be placed out of scope. A future first step is to only take into account the fact that the machine constant is being update, which can easier be implemented in a numeric analysis.

This led to a rather homogenous dataset in which machine constant do have a similar meaning and only analysable constants are included. For outlier detection there were two more filters required to receive the best possible dataset. First of all, a population should consist of enough objects to allow the central limit theory to be valid (see paragraph 12.7). To achieve this, at least 100 examples of a machine constant ($n \geq 100$) should exist within the dataset to pass the filter requirements. This requirement is based upon the work by Jolliffe (1995), Smith & Wells (2006) and Maxwell (2000), taking into account that the number of examples per machine should also lead to a normal distribution, as much as possible. Putting an individual requirement on the number of machine constant examples per machine would reduce population sizes too much to benefit the quality of the analysis. Due to the fact that a normal distribution per machine is therefore not self-evident, it should be kept in mind that the individual machine analysis is less reliable then the population analysis. Machine constants that do not satisfy the data point requirement, and occur only incidental, can still be of high interest, because abnormal behaviour of a machine may be an indication that there is something wrong with that machine. This insight will be further explored during the evaluation phase of the prognostic value of models (see paragraph 9.2).

The second filtering is based on the variance of the numeric value of a machine constant. A population of a machine constant that shows absolute no variation is of no interest when searching for outliers, so the requirement filters only those machine constants that have a variance larger than 0. Despite the fact that there is no variation over time it may still be interesting to investigate the calibration moments over time, but this will be kept out of scope thus far. Finally, due to calculative power restrictions, only 40 different constants that meet all requirements were randomly selected. This was done randomly, because no preference could be given to machine constants that fulfil the requirements and raising the requirements would lead to a set of similar machine constants (e.g. X, Y, and Z space dimension axes of the same machine constant).

- Filter number of examples larger than 100.
- Filter variance larger than 0.
- Sample machine constants equal to 40 different constants.

This resulted in a pre-processed dataset with 71 different machines and a total of 26463 data points. The filtered dataset is then written back into the MySQL database, into another table to keep the original dataset intact. See appendix 7.1 for the (XML) program code of the filtering process.

7.3 Event Definition Process (Time Binning)

The second process to prepare the dataset for the final analysis is the *'event definition process'*. This process edits the creation date for two reasons; reduce false detail within the data and improving the strength of the statistical tools using cluster analysis. As stated in the description of creation date in ASML-only appendix 12.3, the exact time at which the calibration is executed, and the new value is put to practice, is uncertain due to a possible small delay between calibration and registration. Analysing the creation date on a one second scale is therefore meaningless and a round-off to a one minute scale is applied (nearest whole minute is selected). A timeframe of one minute is selected, because this is the smallest time scale that is still always able to aggregate, all calibrations that belong to a single event, into a single time stamp. Increasing the time frame could lead to false aggregation of data from different events. The benefit of the time binning process is that the number of calibrations done on one particular creation date increases, which leads to more input data for each creation date into the cluster algorithms. Where previously information was scattered across different time stamps (that clearly belonged to the same event), this information is now aggregated into a single one time stamp.

What the 'event definition' process exactly does is the following; if converts the creation date from a date format into a numeric value, represented by the number of milliseconds passed by since 01/01/1970 00:00:00 GMT, and then successively divides and rounds-off the number in such a way that it represents a number of minutes. By converting back the numeric value into a date format, a new round-off creation date is obtained. The example set is then rewritten back into the MySQL database table it was retrieved from, replacing its starting set. See appendix 8 for the (XML) program code of the 'event definition' process. Figure 12 visualizes the data preparation process of the initial machine constant dataset.

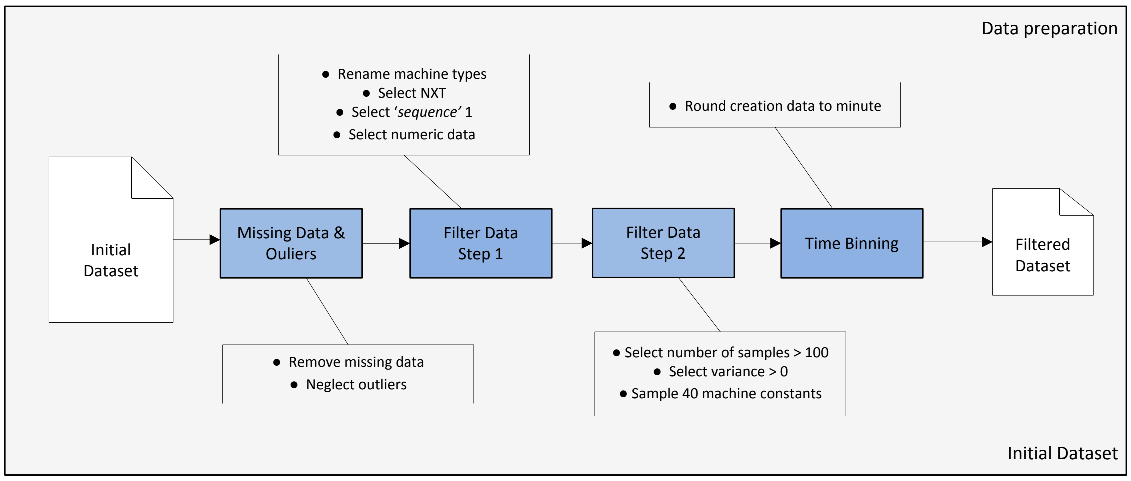


FIGURE 12. INITIAL DATASET PREPARATION

7.4 INTERRUPT OVERVIEW PREPARATION

The preparation of the interrupt overview dataset is not done in an individual pre-process, but is done in a RapidMiner process used during the evaluation phase (see paragraph 9.2). Before combining the outlier list with that of the interrupts, only those interrupts that are related to unscheduled downtime are selected (business relevant interrupt type). Unscheduled down time is the result of a breakdown, which is the event of interest during this thesis. From the interrupt overview dataset, only the machine ID and the time stamp of breakdown are used. The rest of the information is removed from the dataset before combining. See appendix 10 for the process visualization and (XML) program code of the process used during the evaluation and ASML-only appendix 12.4 for an overview of the content of the interrupt overview dataset and details about the interrupts that relate to unscheduled downtime. See Figure 13 for a visualization of the filtering steps applied to the interrupt data.

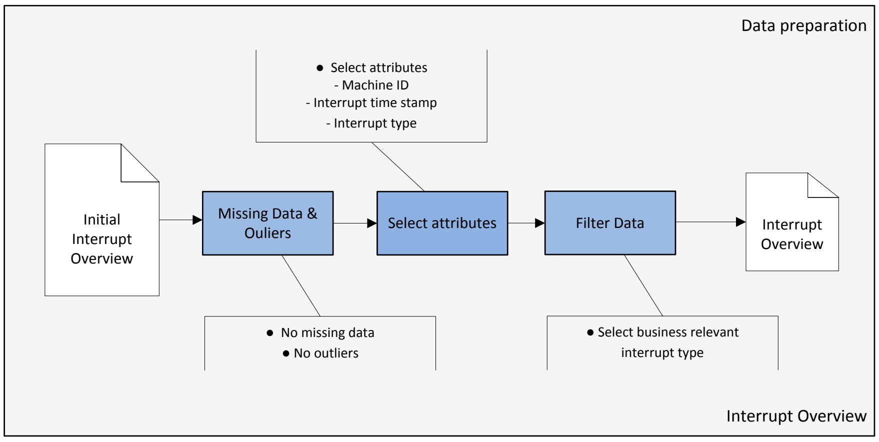


FIGURE 13. INTERRUPT OVERVIEW PREPARATION

8 Modeling

While most people think that modeling is the most time consuming part of a project, more time was spend on understanding the data and preparing it then on the modeling itself. The initial modeling of the data mining analysis was done via a combination of several different sub analyses models. Each individual model determines if a particular machine or machine constant shows unusual behaviour, either compared to its own historical behaviour or to that of the whole population of machines. The higher the absolute normalized values, the higher the number of abnormal activities, and the higher the number of sub models it receives a 'positive' abnormal indication from, the stronger the case that a machine constant on a certain machine indeed behaves in an undesirable way.

During the first phase of the creation of the models, a well-known benchmark dataset was used. The first part of the creation phase aimed at getting familiar with RapidMiner and all its possible processes, and consisted out of trial and error experiments without clear goal. The benefit of using such a set was that the expected outcome of a process of sub model is known, which allowed for early design-flaw and error detection. The benchmark dataset used was the Iris flower dataset, which consist of 50 samples from 3 different related species of flowers (Murphy & Aha, 1994). This dataset is a typical test case for many classification techniques in machine learning. It was preferred above some other similar example sets, because it is standardly available within RapidMiner. For clustering purposes the dataset is able to explain the difference between supervised and unsupervised clustering techniques, because only one out of the three different species is clearly identifiable without supervision (where the other two species blend into one cluster). In general, a test set is a safe and easy way to get familiar with a program (RapidMiner) and the datamining techniques (SOM, K-means, X-means, CBLOF). The flower dataset, with its four different attributes and three different species, is shown in Figure 14.

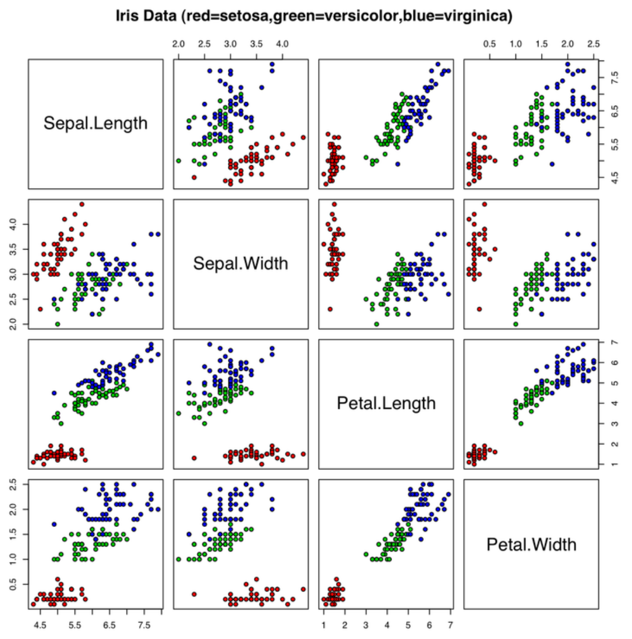


FIGURE 14. IRIS FLOWER DATASET

8.1 Statistical Models

The idea behind the statistical models is that if a machine shows significantly unusual behaviour, chances are higher that this machine will also encounter unusual events. This method is selected due to the fact that background detailed know-how about the behaviour of machine constants was limited and spread-out over several departments and even more different experts. By approaching the data from a data mining perspective, the assumption was made that the majority of machines are in a stable, normal and well-functioning state. This resulted in a similar assumption about the behaviour of the machine constants. If normal behaviour is considered stable, outliers can be identified for predicting unstable (poor) machine states. Because a machine and/or machine constant can show outlying behaviour in several ways, there is a need to analyse this behaviour with several different models. The behaviour analysable by the model is: anomalies in numeric value, size of jumps, number of jumps, trend, the total set of numeric values at a particular moment in time, and the set of calibrations that is executed per event. Each of the models will be explained in detail in this resulting part of this paragraph. These models combine the data mining framework with statistical process control (Control Charts) and pattern recognition (X-means). See paragraph 12.8 for an overview of the chemistry between the theories and methodologies. This paragraph continues with a description of the eight statistical models that can be used to construct a shortlist of examples that are identified as significant outliers. Figure 15 provides an abstract overview of the statistical model.

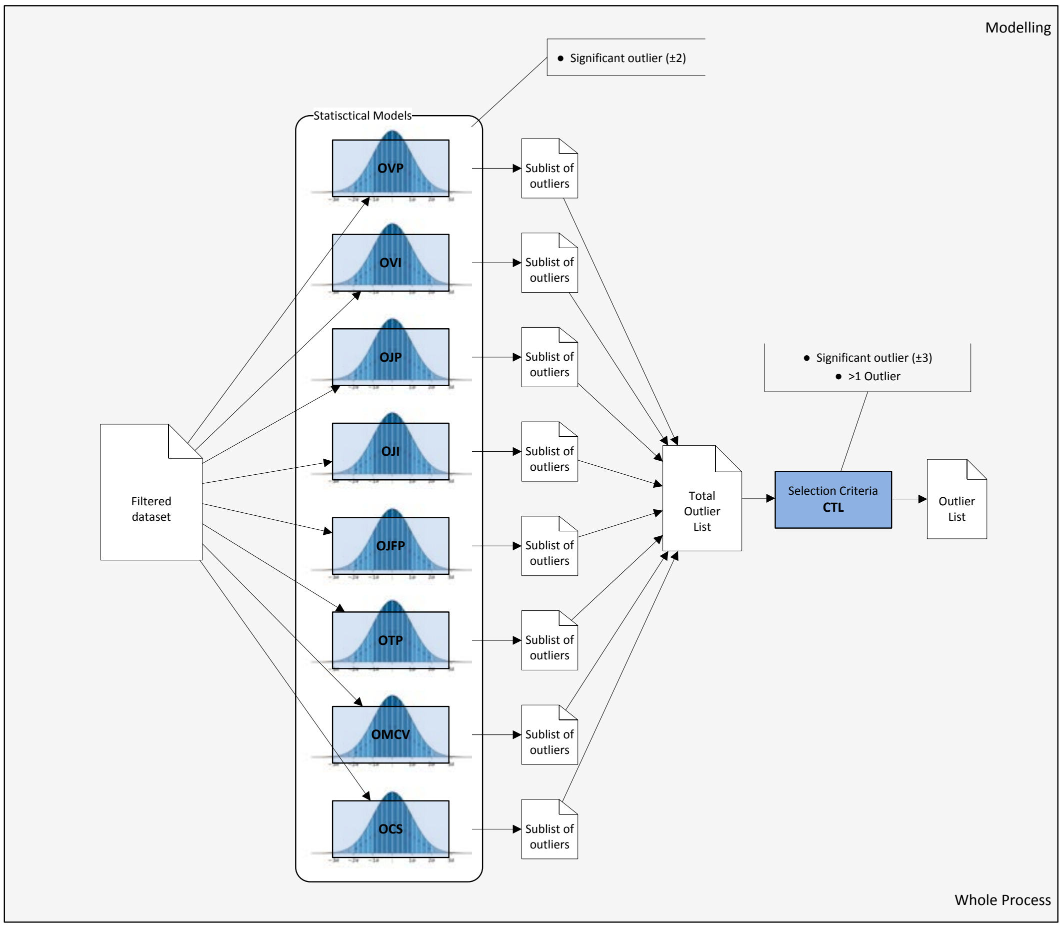


FIGURE 15. TOTAL ANALYSIS PROCESS

Outlier Value compared to Population (OVP)

This statistical model compares individual machine constant values with the whole population of machine constant values. This population consists of all equally named machine constants that are calibrated within the time scope, for which it calculates the mean (average value) and the standard deviation. Each individual value is then normalized, based on the mean and standard deviation of the whole population it belongs to. A value is considered as a possible outlier if it surpasses the normalized threshold values of ± 2, representing the warning limits of the process control charts (see paragraph 12.7). An example set of id's that fulfil the outlier requirements is the final output of this first statistical tool. This first model is an implementation of a commonly variant of the control chart, where an individual value is judged based on its distance from the average value and the common population variation. See appendix 9.1 for the process visualization and (XML) program code of the *'Outlier Value compared to Population'* model.

Outlier Value compared to Individual (OVI)

The second model is fairly similar to the first one discussed. Where the first model compares the machine constant value versus the whole population, the *'Outlier Value compared to Individual'* model only focuses on the machine constant numeric values of the respective machine it belongs to. Via an equal process of normalizing and filtering (based on the ± 2 normalized value criteria), a second set of outlying machine constant values is created. See appendix 9.2 for the process visualization and (XML) program code of the *'Outlier Value compared to Individual'* model. See Figure 16 for an abstract overview of the statistical OVP and OVI models.

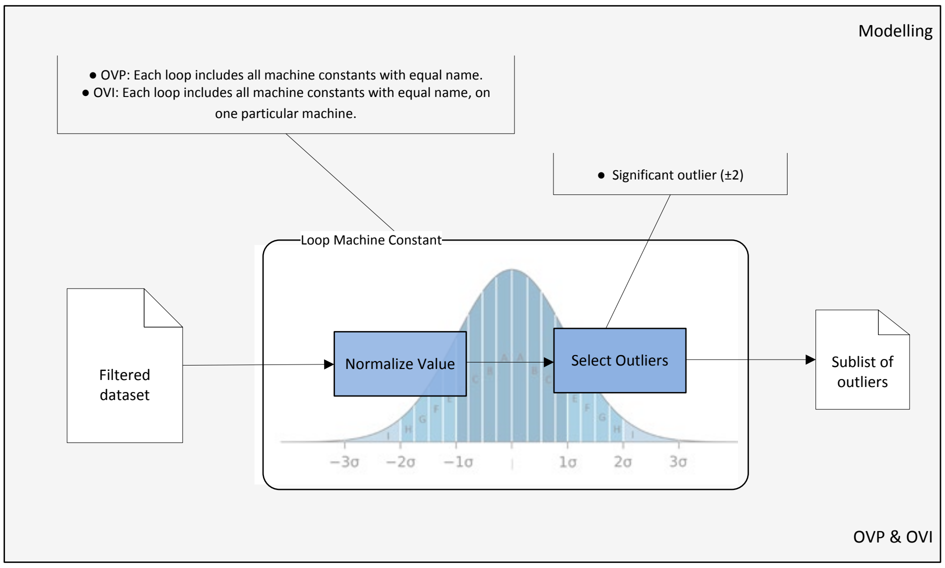

Modelling
● OVP: Each loop includes all machine constants with equal name.
● OVI: Each loop includes all machine constants with equal name, on one particular machine.
● Significant outlier (±2)
Loop Machine Constant
Filtered dataset
Normalize Value
Select Outliers
Sublist of outliers
−3σ
−2σ
−1σ
1σ
2σ
3σ
OVP & OVI

FIGURE 16. OVP & OVI MODEL

Outlier Jump value compared to Population (OJP)

The third sub-model used in the analysis of machine constants focuses on the change of value between two succeeding calibration moments. The value difference can be of such size that it can be labelled as a jump. To determine if a jump is considered an outlier (significantly abnormal jump size) this analysis compares each individual change of value with all other value differences that took place within the time frame. The mean and standard deviation of jump values is based upon the whole population of machines per machine constant. Via normalizing and filtering (based on the ± 2 normalized value criteria), the third set of outlying machine constant values is created. See appendix 9.3 for the process visualization and (XML) program code of the *'Outlier Jump value compared to Population'* model.

Outlier Jump value compared to Individual (OJI)

As in the value models, the jump analysis is also done on both the population and the individual machine. Equal settings are used to determine if a change in values is considered an outlying jump. See appendix 9.4 for the process visualization and (XML) program code of the *'Outlier Jump value compared to Individual'* model. See Figure 17 for an abstract overview of the OJP and OJI model.

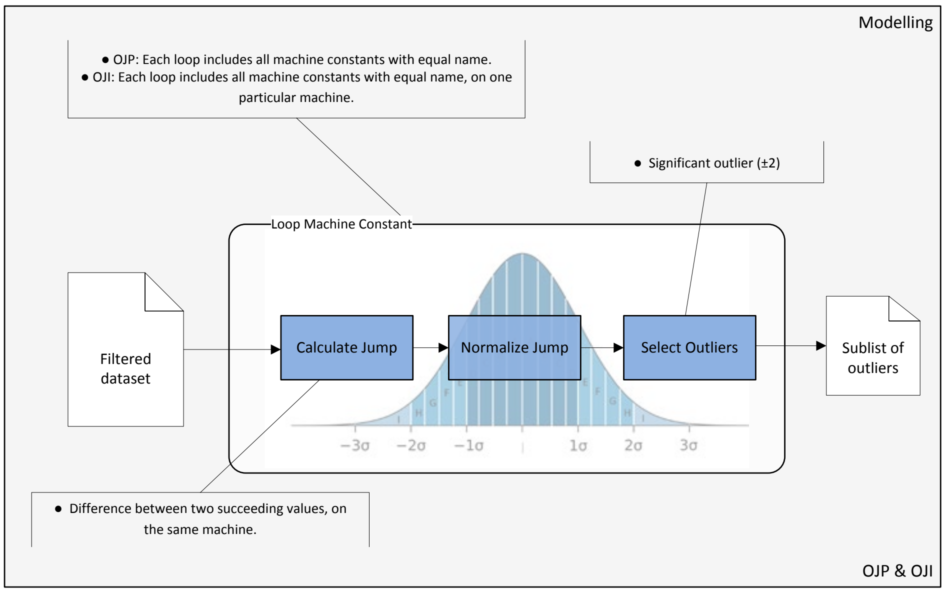

Modelling
● OJP: Each loop includes all machine constants with equal name.
● OJI: Each loop includes all machine constants with equal name, on one particular machine.
● Significant outlier (±2)
Loop Machine Constant
Filtered dataset
Calculate Jump
Normalize Jump
Select Outliers
Sublist of outliers
−3σ −2σ −1σ 1σ 2σ 3σ
● Difference between two succeeding values, on the same machine.
OJP & OJI

FIGURE 17. OJP & OJI MODEL

Outlier Jump Frequency compared to Population (OJFP)

The fifth sub-model is a bit more innovative then the previous more classical outlier filtering and analyses the frequency of jumps a machine constant produces in the given time interval. Every difference in value larger than the standard deviation is considered a jump (softer definition of a jump than applied in the third sub-model). For each combination of machine and machine constant the number of jumps is counted. This count is used to determine if a constant on a certain machine is behaving strange, compared to the whole population. Via normalizing and filtering (based on the ±2 normalized value criteria), the fifth set of outlying machine constant values is created. The difference from previous models is that this analysis compares the total behaviour (number of jumps) over time between machines instead of a single point in time. The disadvantage of this scope is that this warning cannot directly be allocated to a creation date, but only to a machine and constant. See appendix 9.5 for the process visualization and (XML) program code of the '*Outlier Jump Frequency compared to Population*' model. See Figure 18 for an abstract overview of the OJFP model.

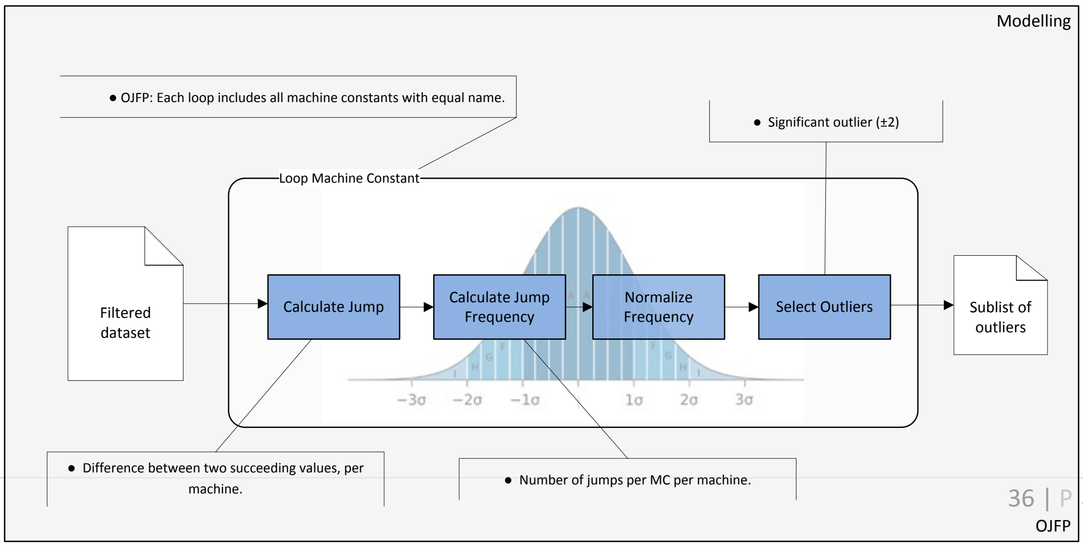

Modelling
● OJFP: Each loop includes all machine constants with equal name.
● Significant outlier (±2)
Loop Machine Constant
Filtered dataset
Calculate Jump
Calculate Jump Frequency
Normalize Frequency
Select Outliers
Sublist of outliers
−3σ −2σ −1σ 1σ 2σ 3σ
● Difference between two succeeding values, per machine.
● Number of jumps per MC per machine.
OJFP

FIGURE 18. OJFP MODEL

OUTLIER TREND COMPARED TO POPULATION (OTP)

The sixth model fits a linear trend, via simple linear regression, on the behaviour of a machine constant during the time frame on a single machine. The linear regression module returns a value that represents the regression coefficient (β_1), referred to as the weight of the trend. The weight is compared with that of the same machine constant on another machine via normalization. If the weight of the trend differences significantly from the rest of the population (+-2 normalized value criteria) the machine constant and machine are identified as possible outliers. See appendix 9.6 for the process visualization and (XML) program code of the '*Outlier Trend compared to Population*' model. See Figure 19 for an abstract overview of the OTP model.

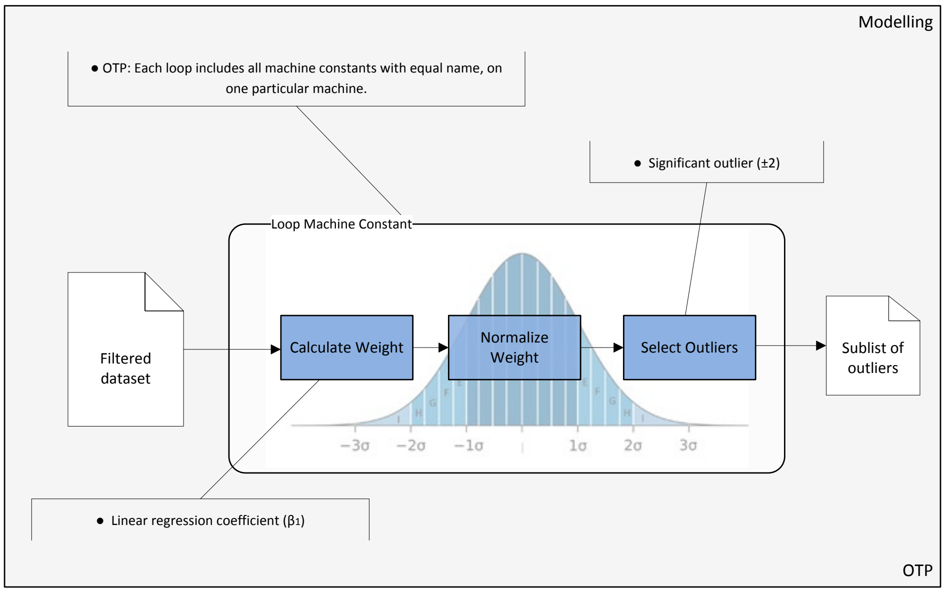


FIGURE 19. OTP MODEL

OUTLIER MACHINE CONSTANT VALUES PER TIME STAMP (OMCV)

Instead of one outlying machine constant found in the first six models, the clustering models suspect all machine constants involved during the outlying creation date. Per creation date all machine constant values are compared with those of the rest of the population. This comparison is done because it may be possible that none of the machine constant values individually can be identified as outliers, but the whole set of machine constants is showing abnormal behaviour. To identify this anomaly, the data is pivoted from different rows for the same machine and creation date, into a table were each of the constants has its own column. The values of a machine constant at a particular time are either based upon the value given to it during its last calibration, or that of the first upcoming calibration, or the average value of the population (preferred in this exact order). The combination of the X-means clustering (see paragraph 12.6) and the cluster-based outlier factor results in an example list of creation dates with an outlier value. Exactly the same steps of normalizing and filtering is also applied to the outlier value and results in a list of outliers, in the form of outlying creation dates per machine. See appendix 9.7 for the process visualization and (XML) program code of the '*Outlier Machine Constant Values per time stamp*' model.

OUTLIER CALIBRATION SELECTION PER TIME STAMP (OCS)

This second clustering model, final example set creation model, and seventh sub-model focuses on the selection of machine constants that are calibrated at the same time. To do so, the data is pivoted from different rows for the same machine and creation date, into a table were each of the 40 constants has its own column. For all 40 constants, per creation date, it is indicated if the constant is calibrated or not with a numeric value equal to binomial values 1 or 0). The combination of the X-means clustering (see paragraph 12.6) and the cluster-based outlier factor results in an example list of creation dates with an outlier value. Exactly the same steps of normalizing and filtering is also applied to the outlier value and results in the last list of outliers, in the form of outlying creation dates per machine. See appendix 9.8 for the process visualization and (XML) program code of the '*Outlier Calibration Selection per time stamp*' model. See Figure 20 for an abstract overview of the OMCV and OCS model.

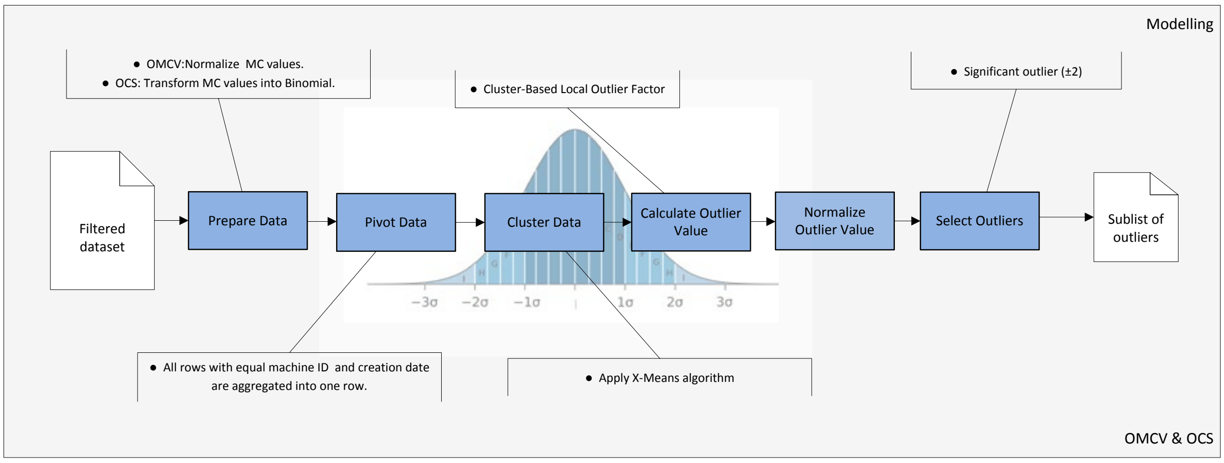


FIGURE 20. OMCV & OCS MODEL

JOIN MACHINE CONSTANT LIST (JMCL)

This sub process joins all the outlier lists, apart from the two clustering models. Via an outer join, based upon their unique example ids, the first four outlier values (if existing) are combined into one list. Due to the fact that the fifth and sixth model (jump frequency & trend) do not use the unique example ids, the same outlier value is added to all examples that have the same combination of machine number and machine constant. This process results in one outlier example list for the first five models. See appendix 9.10 for the process visualization and (XML) program code of the *'Join Machine Constant List' sub* process.

CREATE TOTAL ISSUE LIST (CTL)

The final sub process combines the list of outliers from the first five models with the list from the clustering models. As in the fifth and sixth model, the clustering models also do not use the unique example id. For this reason, the outlier values are added to the list based upon the same combination of machine number and creation date. With this final join, a complete list of all outlying values is created with the original id information (see ASML-only appendix 12.3) added to each outlie. The list is then filtered based on the following requirements: any of the first five models should have found a significant outlier (± 3) and the summation of the absolute outlier values of the all the models should exceed 6. The first filtering rule comes from Shewhart's Statistical Process Control (see paragraph 12.7) and tests the 0.9973 confidence interval. Clustering models are excluded from this first test, because they are unable to indicate one clear machine constant that is to blame. An output with only a machine number and creation date hardens the validation process. The second filtering is done to further reduce the issue list to the most likely problematic outliers, because it requires the example to show abnormal behaviour on more than one test. This is less hard than it may seem, because the models are related. For example an extreme outlier will most likely also lead to one or two extreme jumps, and if a numeric value is an extreme outlier in the population it is also more likely to be an outlier in its own individual behaviour. Furthermore, the 'outlier machine constant values per time stamp' clustering algorithm uses the same numeric values as the non-clustering algorithms. One extreme outlying machine constant may lead to an outlier classification of that whole time stamp (creation date). Requiring the models to produce a consistent judgment of the outliers increases the reliability of the model as a whole. The second filtering criterion is not based upon theory, but upon the above described reasoning, and is kept flexible to result feedback. The final output of this sub process, and also the final result, is written into an excel file and should provide advice on (future) maintenance activities. See appendix 9.11 for the process visualization and (XML) program code of the *'Create Total issue List'* sub process. See Figure 21 for an abstract overview of the JMCL and CTL process.

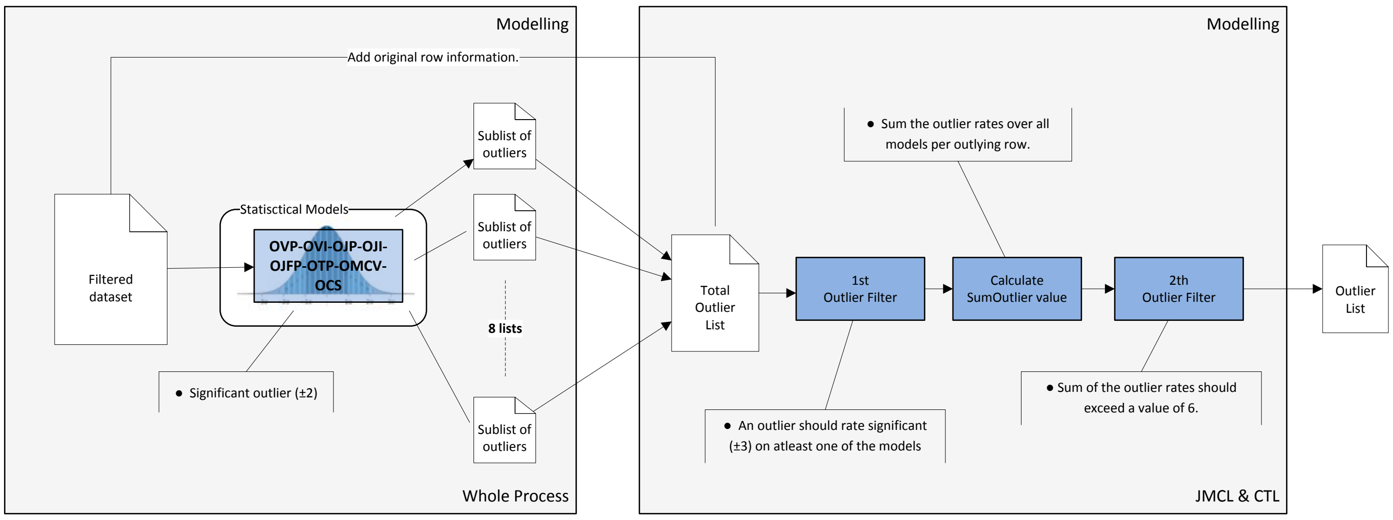

Modelling
Add original row information.
Sublist of outliers
Statisctical Models
OVP-OVI-OJP-OJI-OJFP-OTP-OMCV-OCS
Filtered dataset
Sublist of outliers
8 lists
Significant outlier (±2)
Sublist of outliers
Whole Process
Modelling
Sum the outlier rates over all models per outlying row.
Total Outlier List
1st Outlier Filter
Calculate SumOutlier value
2th Outlier Filter
Outlier List
An outlier should rate significant (±3) on atleast one of the models
Sum of the outlier rates should exceed a value of 6.
JMCL & CTL

FIGURE 21. JMCL & CTL PROCESS

Model Background Information

It is a well-considered decision not to divide the set into a training and test set. The reason for this is that the environment of the machine constants is not stable on the long run, due to continuous improvement, change and introduction of (new) machines. The latest population of machine constant values may differ from the old population, due to known non-problematic changes. Therefor the tools should not be trained by any other dataset then one that includes the most recent data. Due to the large quantity of data available and the restrictions on the selection of machine constants (N>100), the influence of behaviour in the data reviewed (which is also part of the input set) on the mean and standard deviation is insignificant. One should keep in mind that the restrictions and settings used in all the models are highly customizable. Depending on the results in practise it could be beneficial to change the control limits or add specific weight values to the individual model outliers. For example, population test will most likely be more reliable due to a larger sample size (Central Limit Theorem). This also applies to the clustering models, which are hard to qualify beforehand due to their black box characteristics. If this proves to be the case in practise, a higher weight should be attached to the reliable models or other models should be removed completely.

See Figure 22 for a RapidMiner overview of the whole model and one example of a sub model included (‘*Outlier Jump Frequency compared to population’* model) and appendix 9 for the (XML) program code.

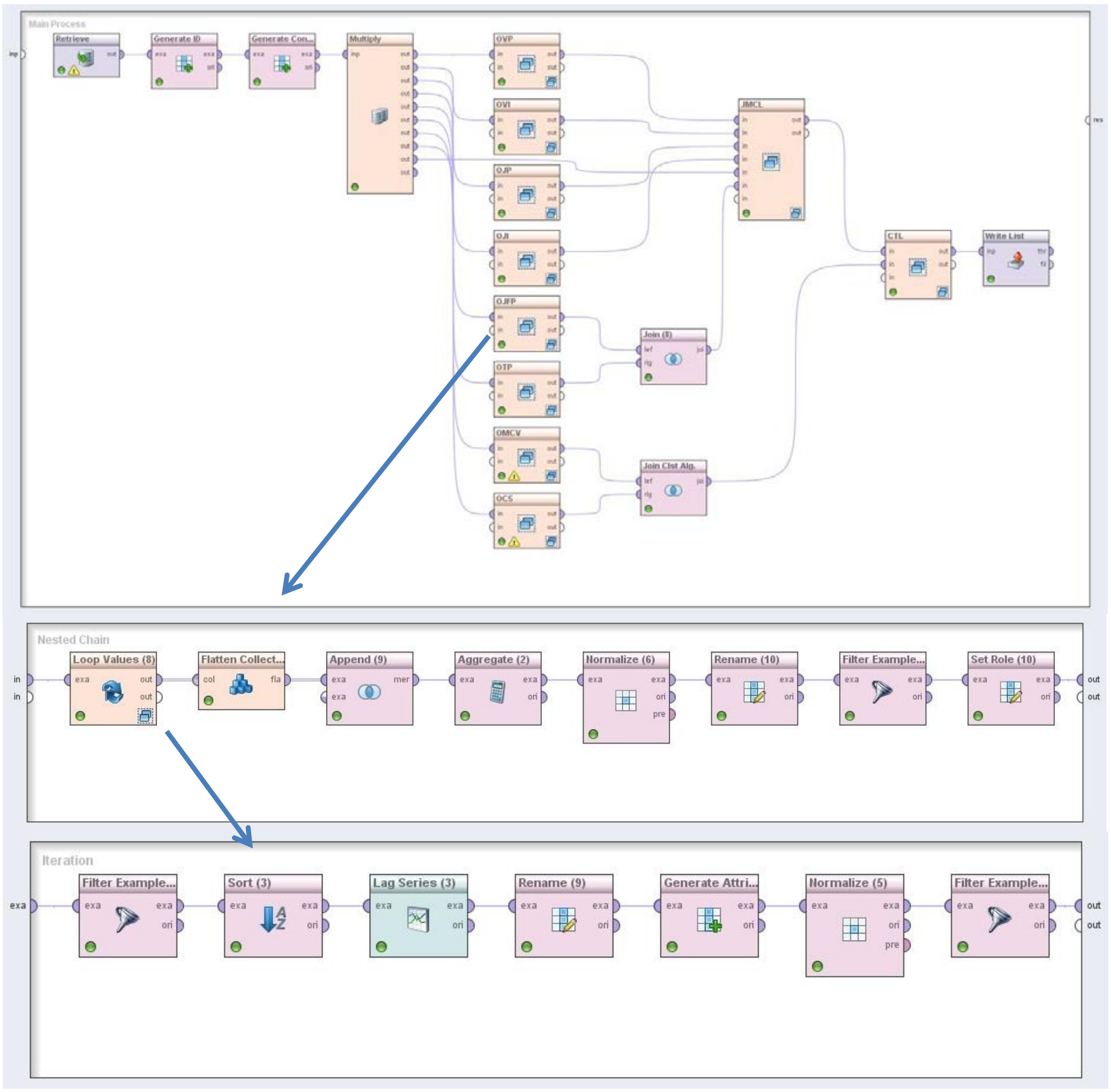


FIGURE 22. TOTAL ANALYSIS PROCESS (RAPIDMINER)

9 EVALUATION

The evaluation of the models is the second to last phase of the CRISP-DM method and the last phase performed explicitly in this thesis research. The goal of the evaluation phase is to validate if the model is able to provide exactly that information that is expected from it, and make a statement about the usability of the model in practice. As defined in the research design (see paragraph 3.2), the model should have both diagnostic and prognostic value. Especially in case of implementation into a preventive maintenance environment, the prognostic value is of high importance. The evaluation of the model starts with the analysis of the diagnostic value, by comparing the predicted with the actual machine state at the moment of a warning. The prognostic value is evaluated by comparing the predicted with the actual machine state behaviour after a warning. Evaluation of the prognostic value led to adaptations of the statistical models, wherefore four different runs (three sets of outlier and paired lists) were performed. More details about the evolution of the prognostic value analysis are given in paragraph 9.2. Throughout the evaluation chapter, a green highlighting of a cell within a data table indicates that a significant result is found.

With the use of a simple RapidMiner process (see appendix 10) the outlier lists, created by the statistical models, are combined with the interrupt file to be able to analyse the relation between outliers and failures. See paragraph 4.3 for an overview of the statistical methodology used during the evaluation phase.

The following time stamp lists were used throughout the evaluation phase:

OUTLIER LIST

The outlier list consists out of all the outliers/warnings that were found with the use of the statistical models described paragraph 8.1. Three different outlier lists were created for the evaluation. The first set, consisting out of 211 examples, was used for both diagnostic and prognostic evaluation. Second (145 examples) and third (61 examples) set are only used for prognostic evaluation and were created via adaptations of the original statistical model.

- Lists are the output of the statistical models.
- Used for evaluation of diagnostic and prognostic value.

RANDOM LIST

The random list consist out of a sample of 100 time stamps taken randomly from all machines that were part of the prepared dataset that was used as input for the analysis. The period from which the examples were taken is an extension to the original time frame, from which the outlier list was extracted. This is done to decrease the chance of selecting the same time stamps, which would decrease the possibility of finding significant differences. The random list is used to validate if the statistical model is significantly able to identify machines that are more likely to experience an interrupt. The extended time period goes from January up to May. It was decided to select an overlapping period (on both sides) to diminish the influence of external factors over time. The random list is only used during the evaluation of the diagnostic value, for the purpose of machine state comparison.

- 100 random selected machines from the initial dataset.
- Added random created time stamps, from the extended time period January up to May.
- Used for evaluation of diagnostic value (identification of machines with higher interrupt probability)

Paired List

For the diagnostic value evaluation the paired list consist out of a sample of 100 time stamps for machines from the original outlier list (duplicate machines are allowed). This means that machines that were more present in the original outlier list were also more likely to end up in the sample of the paired list. While it is not the classical definition of a paired sample, this list will be referred to as the paired list, because the content of this list can form a pair with an example in the outlier list. All the samples were selected from the same time period as the random list, for the same reasons. While the random list can only validate if the model is able to identify machines with a higher interrupt probability, the paired list is used to validate if the statistical model is also significantly able to identify the exact interrupt moment on these machines. For testing of the prognostic value of the model, the paired list consisted out of the same number of examples as the outlier lists, from the same machines, in an *up state*. Due to the three different runs executed during the prognostic value evaluation, there are three different lists of paired machine time stamps. The first set uses the same time frame as used for the Random List, and the second and third set are from the same time frame as the outlier lists (February till the mid of March).

- Random selected machines from the outlier list.
- Added random created time stamps.
- Used for evaluation of prognostic and diagnostic value (identification of interrupt time stamps on a machines)

Figure 23 gives an overview of the different data sets used during the evalution phase.

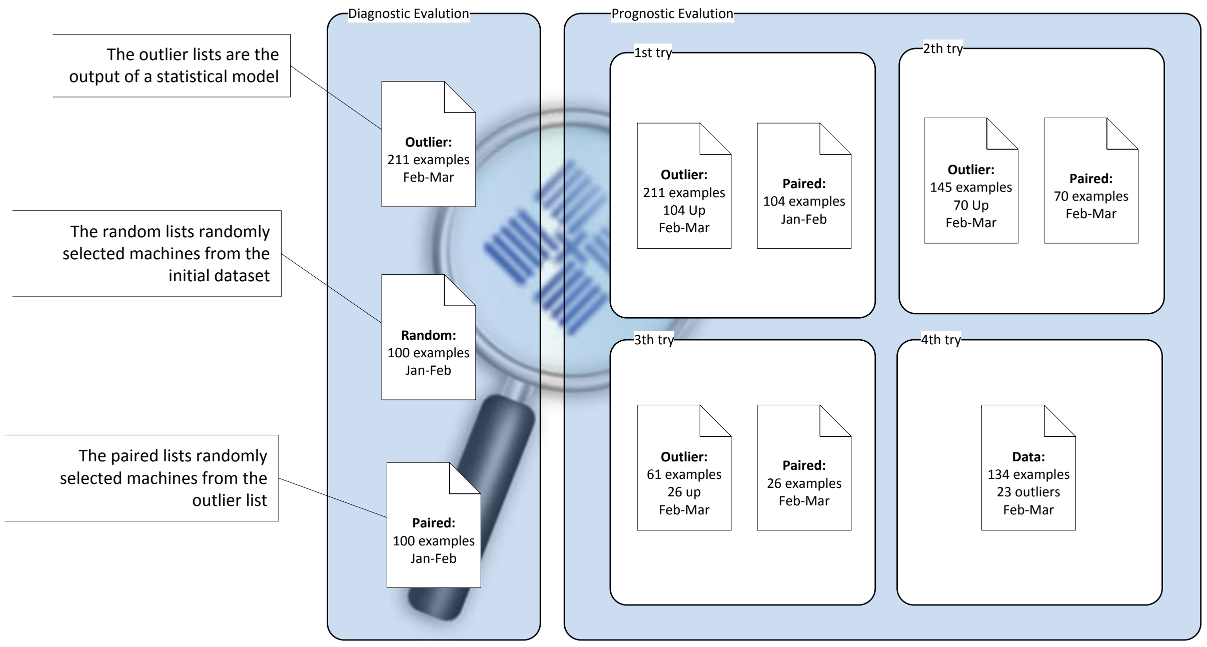

The outlier lists are the output of a statistical model
The random lists randomly selected machines from the initial dataset
The paired lists randomly selected machines from the outlier list
Diagnostic Evalution
Outlier: 211 examples Feb-Mar
Random: 100 examples Jan-Feb
Paired: 100 examples Jan-Feb
Prognostic Evalution
1st try
Outlier: 211 examples 104 Up Feb-Mar
Paired: 104 examples Jan-Feb
2th try
Outlier: 145 examples 70 Up Feb-Mar
Paired: 70 examples Feb-Mar
3th try
Outlier: 61 examples 26 up Feb-Mar
Paired: 26 examples Feb-Mar
4th try
Data: 134 examples 23 outliers Feb-Mar

FIGURE 23. DATA LISTS OVERVIEW

9.1 Diagnostic Value

The goal of validating the diagnostic value of the model (made up out of the eight sub models described in paragraph 8.1) is to prove that there is a significant relation between machine constant values and machine status. The machine status is categorised in an up and down state, which is determined by information extracted from Insight (see paragraph 5.2). When the number of downs found at the warning moment is significantly higher than at a random moment in time it is permitted to state that the model has significant analytical capabilities. To do so, the number of current down states found in the outlier list is compared with that of the random list using the Binomial Z-test (see Table 5). Conditions are met, because both categories have enough examples. At a 5% level of significance (2-tail), the critical region is $Z < -1.96$ or $Z > 1.96$, and 3.699 > 1.96, so H_0 is rejected.

TABLE 5. PREDICTION VS. RANDOM

	Sample Size (N_x)	# Down	Sample Proportion (p_x)
Outlier Group	211	106	0.502
Random Group	100	28	0.280

Condition: $(N_x p_x > 10 \wedge N_x(1-p_x) > 10, \forall x \in Outlier, Random)$

Null hypothesis (H_0): $p_1 = p_2$

Pooling the data, we obtain $\hat{p} = \frac{106+28}{211+100} = 0.431$

$$Z = \frac{0.502-0.280}{\sqrt{(0.431)(1-0.431)(\frac{1}{211}+\frac{1}{100})}} = 3.699$$

Based on the test result and the difference between sample proportion of number of downs, there can be conclude that the model is able to predict if a machine is indeed experiencing issues at that exact moment in time. So, abnormal machine constant behaviour is an indicator that the machine is broken down. To further investigate this relation the outlier list was compared with the paired list. This is done to not only prove that the model is able to predict which machines have a higher chance of experiencing issues, but also to test if, for outlying machines, the warning moment has a higher chance of being in a interrupt state then a random point in time on the same machine. At a 5% level of significance (2-tail), the critical region is $Z < -1.96$ or $Z > 1.96$, and 2.354 > 1.96, so H_0 is rejected.

TABLE 6. PREDICTION VS.PAIRED

	Sample Size (N_x)	# Down	Sample Proportion (p_x)
Outlier Group	211	106	0.502
Paired Group	100	36	0.360

Condition: $(N_x p_x > 10 \wedge N_x(1-p_x) > 10, \forall x \in Outlier, Paired)$

Null hypothesis (H_0): $p_1 = p_2$

Pooling the data, we obtain $\hat{p} = \frac{106+36}{211+100} = 0.457$

$$Z = \frac{0.502-0.360}{\sqrt{(0.457)(1-0.457)(\frac{1}{211}+\frac{1}{100})}} = 2.354$$

Based on both tests there can be concluded that the outlier model is able to recognize machine status based on the behaviour of machine constants, because in case of a warning the possibility of a machine down state is significantly higher than at a random point in time. While this does not provide any prove for the predictive value of the model, it does prove that the model does relate machine constant behaviour to current machine state. Figure 24 visualizes the evaluation process of the diagnostic value of the statistical model.

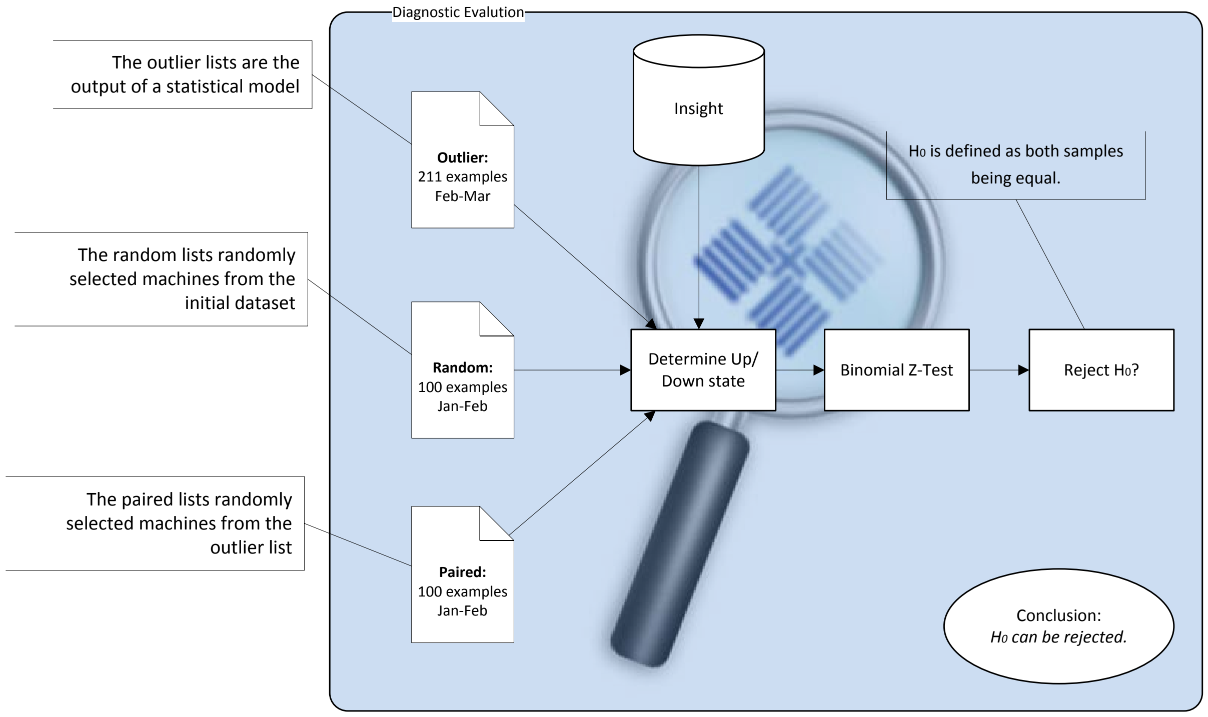

Diagnostic Evalution
The outlier lists are the output of a statistical model
The random lists randomly selected machines from the initial dataset
The paired lists randomly selected machines from the outlier list
Outlier: 211 examples Feb-Mar
Random: 100 examples Jan-Feb
Paired: 100 examples Jan-Feb
Insight
H0 is defined as both samples being equal.
Determine Up/ Down state
Binomial Z-Test
Reject H0?
Conclusion: H0 can be rejected.

FIGURE 24. OVERVIEW DIAGNOSTIC VALUE EVALUATION

When reviewing the '*SumOutlier*' value, which is the sum of the absolute standardized value of all the outlier tests, it was found that the higher the value the more likely it is that the machine is in a down state (see Table 7). This complies with the above findings that the model is able to predict the state of a machine, because the model already selects those '*SumOutlier*' values that are above a certain limit. The reason why none of the individual models are significant is because each outlier only scores an outlier value for a few of the submodels. So each for each submodel, the population consist mainly out of non-outliers. This does not fulfil the Student T conditions about normality (see paragraph 4.3).

TABLE 7. OUTLIER VALUE, UP/DOWN COMPARISON

Up/Down		SumOutlier	OVP	OVI	OJP	OJI	OJFP	OTP	OMCV	OCS
	Student T-Test	1.644E-06	0.119	0.974	0.644	0.449	0.923	0.923	0.363	0.163
Up	*Average*	7.820	2.444	0.962	3.611	1.184	0.559	0.559	0.564	0.341
	Standard Dev.	1.565	3.370	1.571	3.405	1.872	1.265	1.265	1.361	1.180
Down	*Average*	11.810	3.305	0.954	3.390	0.994	0.616	0.616	0.409	0.592
	Standard Dev.	8.075	4.534	1.563	3.540	1.642	1.489	1.489	1.090	1.407

9.2 PROGNOSTIC VALUE

After concluding that the model (see paragraph 8.1) does have diagnostic value, it was an important next step to investigate the possibility of prognostic value of the model. The goal of validating the prognostic value is to prove that there is a significant relation between machine constant values and *future* machine status. To do so, the machine behaviour in the period after the warning (outlier) was analysed (see Figure 25). It is not possible to state with certainty that a machine is no longer in a critical outlying state after a new calibration. This is based on the fact that the outlying state is not necessary left with the next calibration. For example, if a machine constant is calibrated to the same value that it received after that constant made an outlying jump, there is no reason why that jump can no longer be the cause of an upcoming failure. Therefore, all interrupts within the upcoming period of time are considered to be possibly related to that outlier.

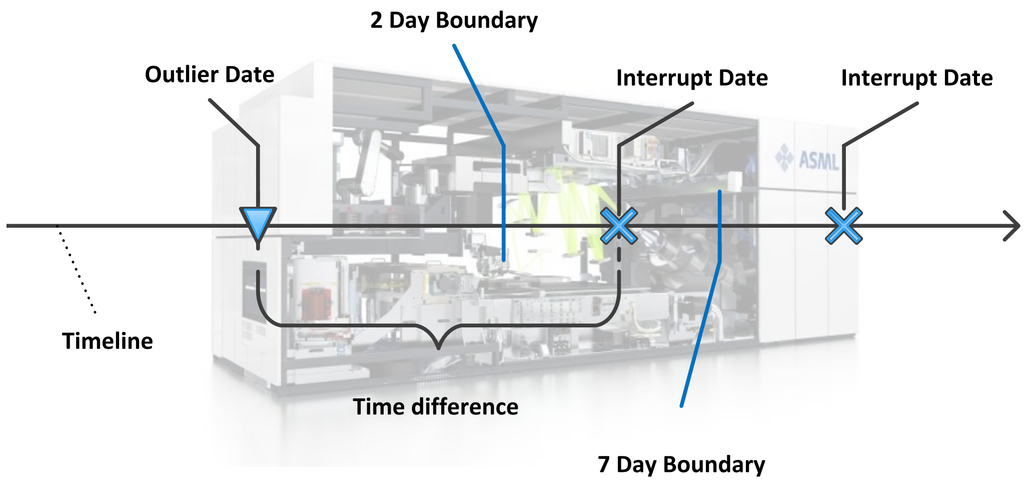


FIGURE 25. INTERRUPTS AFTER WARNING

For all examples that are listed in the outlier set, it was necessary to only select those that could have some predictive, preventive or advising value. To do so, the direction and timing of the relation between machine constants and machine status (up/down) is important. If an outlier is identified during a machine down state, then the warning is already too late to prevent that down or could even be the result of the down, instead of the cause. For prognostic purposes, it is important to understand the cause/effect relationship to make sure that no false causality is claimed. For this reason, the example set was reduced to only those warnings that occur during machine uptime.

FIRST TRY – ORIGINAL MODEL

The model can be validated and qualified via two different methods. The first step would be to tests if there is a significance difference between the outlier list and a random list concerning the chance of occurrence of a system interrupt within a certain amount of days (tested for both 2 and 7 days) after the selected point in time. This is done via a both the student t-test and the similar, but simpler, binomial test (see paragraph 4.3). The following statistics were compared between the outlier list and the paired list:

- Chance that at least one interrupt occurs within 2 (or 7) days after the time stamp.
- The number of interrupts that occur within 2 (or 7) days after the time stamp.

This led to the following result:

TABLE 8. INTERRUPT AFTER WARNING (1^{ST} TRIAL)

	Broken 2	Broken 7	# Broken 2	# Broken 7
Average				
Outlier	45.19%	69.23%	1.010	3.567
Paired	54.81%	74.04%	1.154	3.606
Variance				
Outlier	0.248	0.213	2.452	19.226
Paired	0.248	0.192	2.649	14.239
	Sample Size			
Outlier	104			
Paired	104			
T-Test (<.05)				
Outlier/Paired	0.167	0.444	0.518	0.946
Z-Test (+-1.96)				
p	0.5	0.716		
Outlier/Paired	-1.387	-0.769		

Contradictive to the hypothesis, the probability that a machine fails and the total number of failures shortly after an outlier are not significantly higher than that of the paired set, but even slightly (non-significantly) lower. This means that for the datasets used, a failure shortly after an outlier was less common than after a random point in time. Because this conflicts with the hypothesis some further investigation was required. The first step was to look if there was a difference between outliers found via any of the eight different statistical models (see paragraph 8.1). What was found is that only OVI and the OMCV model show significant difference with either the whole paired set or a modified paired set (paired with the subset)(see digital appendix A2). Surprisingly both sub models provide outlier lists that also experience significantly less failures in the upcoming time period, while the opposite effect would be expected.

As described before, the reason that the outlier list is even worse than a random sample set could lie in the fact that the relation between machine constant outlier and down is hard to specify. Even after distinguishing between machines in up and down time, it could be possible that outlying behaviour is still the result of some event in the past. One would except that the chance of finding a breakdown shortly before the time stamp would be higher for the outliers. To verify or reject this hypothesis the same analysis as before was performed on the time frame before the given time stamp (see Figure 26).

The following statistics were compared between the outlier list and the random paired list:

- Chance that at least one interrupt occurs within 2 (or 7) days before the time stamp.
- The number of interrupts that occur within 2 (or 7) days before the time stamp.

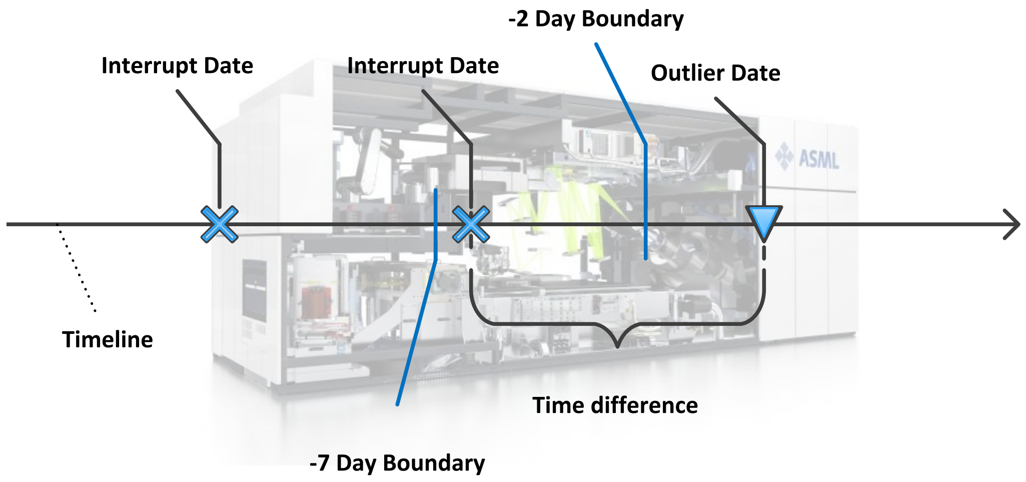


FIGURE 26. INTERRUPTS BEFORE

While it results in a non-significant difference for the overall list of outliers and those found via the OVI model (both having almost equal means)(see digital appendix A2), the outcome of the analysis of the outliers found via the OMCV model is even more surprising (see Table 9). The fact that also the chance of having a breakdown event in the short past is significantly lower for the outlier list then for the paired list, not only rejects the assumed theory, but creates a rather unexpected and difficult to explain situation. An outlier found with the OMCV model would have significantly fewer failures before and after the moment it occurs. There are a couple of reasons why this result could be found and why it should be neglected. The fact that the example set is rather small ($N < 20$), combined with the notification that 9 out of the 17 (53%) outliers come from the same machine (X) and 5 out of 17 (29%) come from another machine (Y) makes it a weak representation of the whole machine constant population. This gets even more problematic if for the same 53% there is no known breakdown before the time stamps. Another issue is that the outliers are not unrelated to each other, because if the set of all machine constant values is labelled an outlier at a certain time stamp it will most likely still be an outlier after the next calibration of a single machine constant. This is indeed the reason why the sample set consists out of only a few different machines, with time stamps shortly succeeding each other. Finally the value clustering algorithm is a black box algorithm, which is based upon a lot of value back engineering and assumed missing values. These arguments allow neglecting the statistical 'significant' results found and concluding that the clustering algorithms are unable to function in this low information per row (CREATION_DATE) environment.

TABLE 9. INTERRUPT BEFORE WARNING (OMCV)

Paired				
	Broken 2	**Broken 7**	**# Broken 2**	**# Broken 7**
Average				
Outlier	29.41%	41.18%	0.471	1.118
Paired	54.81%	73.08%	1.221	3.529

Variance				
Outlier	0.208	0.242	0.955	5.633
Paired	0.248	0.197	2.557	12.384

	Sample Size			
Outlier	17			

Subset				
	Broken 2	**Broken 7**	**# Broken 2**	**# Broken 7**
Average				
Outlier	29.41%	41.18%	0.471	1.118
Paired	58.82%	82.35%	1.176	3.412

Variance				
Outlier	0.208	0.242	0.955	5.633
Paired	0.242	0.145	1.910	11.654

	Sample Size			
Outlier	17			

Paired	104			

T-Test				
Out/Pair	0.053	0.008	0.065	0.008

Z-Test				
p	0.512	0.686		
Out/Pair	-1.94	-2.627		

Paired	17			

T-Test				
Out/Pair	0.089	0.013	0.105	0.035

Z-Test				
p	0.441	0.618		
Out/Pair	-1.727	-2.470		

Looking back to the original results of the upcoming failures after a warning, it is interesting to see that the average value of the individual models is lower than the average value of the population models (see digital appendix A2). The reason that the individual models perform '*worse*' than the population models could be the result of the validation of the Central Limit Theorem. For such a small time period, the number of samples for a machine constant on a single machine will in some cases be too small to allow strong statistical conclusions.

The final observation, based on the '*SumOutlier*' findings done during the diagnostic evaluation, is that for strong outliers ($SumOutlier > 10$), there is more breakdown behaviour after a warning then after a random point in time (see Table 10). The result is not significant, because the filtered example set is very small ($N = 20$). A non-significant result does not allow for any general conclusions, but provides a direction to continue the evaluation.

TABLE 10. INTERRUPT AFTER WARNING (SUMOUTLIER)

Paired				
	Broken 2	**Broken 7**	**# Broken 2**	**# Broken 7**
Average				
Outlier	72.73%	90.91%	1.909	5.182
Paired	54.81%	74.04%	1.154	3.606

Variance				
Outlier	0.198	0.083	6.446	18.694
Paired	0.248	0.192	2.649	14.239

	Sample Size			
Outlier	11			
Paired	104			

T-Test				
Out/Pair	0.258	0.219	0.176	0.201

Z-Test				
p	0.565	0.757		
Out/Pair	1.140	1.240		

Subset				
	Broken 2	**Broken 7**	**# Broken 2**	**# Broken 7**
Average				
Outlier	72.73%	90.91%	1.909	5.182
Paired	45.45%	81.82%	0.545	3.727

Variance				
Outlier	0.198	0.083	6.446	18.694
Paired	0.248	0.149	0.430	20.562

	Sample Size			
Outlier	11			
Paired	11			

T-Test				
Out/Pair	0.211	0.557	0.116	0.471

Z-Test				
p	0.591	0.864		
Out/Pair	1.301	0.621		

While the first run provides some insight into the lack of prognostic value of the model, it also provides some suggestions on how to improve and continue the research. These will be incorporated into the second run.

Second Try – Slim Model

Because the model used during the first run did not have any predictive value some adaptations were applied to try and improve the model. The following adaptations were made:

- Instead of putting an extra restriction on the number of machine constants per machine, the individual models are removed. This should not lead to high quantities of missed warning, because the population models test similar condition. A population outlier is more likely to be an individual outlier and vice versa.
- The black box clustering sub models proved not to be applicable to the current dataset, so they are removed from the model.
- Where the number of tests decreases, the required '*SumOutlier*' value stays the same (> 6) to select only the extreme outliers.
- The number of machine constants in the input list increases to 60 (150% of the original set), because the calculative requirements of the model is reduced. The benefit of adding more machine constants is that a larger portion of the total machine constant population is represented and that the number of outliers to be found also increases.
- The time frame is decreased to the exact time frame of the original dataset, because equal time frames allow for comparison of the time between warning and failure and it removes the possibility of seasonal or any other time related effects. The increase in the possibility of analysing overlapping time periods is taken for granted as a negative side effect.

The adapted requirement led to a dataset that consisted out of 71 different machines and a total of 38371 data points. Based upon the recommendations found during the first run, the following sub models were excluded from the model:

TABLE 11. SECOND TRY MODELS

Included models	Excluded models
Outlier Value Compared to Population (OVP)	Outlier Value Compared to Individual (OVI)
Outlier Jump Value Compared to Population (OJP)	Outlier Jump Value Compared to Individual (OJI)
Outlier Jump Frequency Compared to Population (OJFP)	Outlier Machine Constant Value per Time Stamp (OMCV)
Outlier Trend Compared to Population (OTP)	Outlier Calibration Selection per Time Stamp (OCS)

Because the model used during the second try is based upon the model used in the first try, no new visualization or program code will be given. With this slimmer model, and similar procedures used during the first run, quantitative results were obtained for the interrupt behaviour after (see Table 12) and before (see Table 13) the warning times.

TABLE 12. INTERRUPT AFTER WARNING (2TH TRIAL)

	Broken 2	Broken 7	# Broken 2	# Broken 7	MinDiff
Average					
Outlier	40.00%	62.86%	0.657	2.829	16.846
Paired	44.29%	61.43%	0.629	2.214	16.711

Variance					
Outlier	0.240	0.233	0.997	17.999	548.193
Paired	0.247	0.237	0.633	6.654	545.958

	Sample Size				

Outlier	70				
Paired	70				

T-Test (<.05)					
Out/Pair	0.611	0.863	0.853	0.306	0.973

Z-Test (+-1.96)					
p	0.421	0.621			
Out/Pair	-0.513	0.174			

TABLE 13. INTERRUPT BEFORE WARNING (2TH TRIAL)

	Broken 2	**Broken 7**	**# Broken 2**	**# Broken 7**	**MinDiff**
Average					
Outlier	41.43%	55.71%	0.643	2.700	-26.023
Paired	41.43%	61.43%	0.586	2.400	-24.261

Variance					
Outlier	0.243	0.247	0.915	13.610	822.665
Paired	0.243	0.237	0.700	8.211	803.515

	Sample Size				
Outlier	70				
Paired	70				

T-Test (<.05)					
Out/Pair	1	0.496	0.709	0.595	0.717

Z-Test (+-1.96)					
p	0.414	0.586			
Out/Pair	0	-0.686			

What can be concluded from the results is that the assumed possibility of improving the prognostic value of the original model was false. While the model no longer generates predictions with significantly lower number of failures before or after a warning, it also still does not predict significant higher number of failures in those periods. No promising adaptations of the original model exist, so it can be concluded that basing upcoming machine status on anomalies in current machine constant value is impossible.

Third Try – Unique Constants

After two failed tries, based upon the idea of anomaly detection, a rather different approach is tested during the third try. Instead of looking at the value of the machine constant, this rather simple model identified those machine constants that were only calibrated once during the whole time period. The fact that a machine constant is unique may indicate that the machine is experiencing abnormal behaviour. Because this identification process requires no analytics, the outlier list was created via the same data preparation process used during the first try (see paragraph 7.2). The only difference is that the number of examples of a machine constant in the dataset should not be larger than 100 ($N \geq 100$), but equal to only 1 ($N = 1$). The result of the filtering process were 29031 unique machine constants, spread across 61 creation dates on 31 different machines. It is interesting to note that these machine constants most be related, because the number of machine constants per time stamp is high.

Another noteworthy finding is that 57.3% of the creation dates results in a current machine down state. Comparing this with the results found during the diagnostic value evaluation (see paragraph 9.1); it is safe to state that, besides abnormal machine constant values, abnormal machine constants themselves are a strong indication that the machine is in a down state. This further strengthens the conclusion that machine constant have diagnostic value.

Continuing with the analysis of the prognostic value of machine constants, the failure behaviour before and after the warning time is compared with a paired group (see Table 14 and Table 15).

TABLE 14. INTERRUPT AFTER WARNING (3TH TRIAL)

	Broken 2	Broken 7	# Broken 2	# Broken 7	MinDiff
Average					
Outlier	42.31%	65.38%	0.692	2.808	11.850
Paired	46.15%	61.54%	1.500	3.192	10.472

Variance					
Outlier	0.244	0.226	0.905	10.694	277.351
Paired	0.249	0.237	5.250	13.925	255.251

	Sample Size				
Outlier	26				
Paired	26				

T-Test (<.05)					
Out/Pair	0.785	0.779	0.110	0.670	0.767

Z-Test (+-1.96)					
P	0.442	0.635			
Out/Pair	-0.279	0.288			

TABLE 15. INTERRUPT BEFORE WARNING (3TH TRIAL)

	Broken 2	Broken 7	# Broken 2	# Broken 7	MinDiff
Average					
Outlier	42.31%	53.85%	1.077	3.115	-26.425
Paired	46.15%	57.69%	0.846	3.385	-24.568
Variance					
Outlier	0.244	0.249	3.686	19.564	829.957
Paired	0.249	0.244	1.592	28.391	799.207
	Sample Size				
Outlier	26				
Paired	26				
T-Test (<.05)					
Out/Pair	0.785	0.785	0.618	0.847	0.819
Z-Test (+-1.96)					
P	0.442	0.558			
Out/Pair	-0.279	-0.279			

Similar to the findings during the first two tries, also the third try is unsatisfying in prognostic value. There is no significant difference between the outlier list and the paired list, for either behaviour before or after a warning. While the sample size is not as large as preferred, it is safe to conclude that a larger sample size will not improve the results (being so far away from positively significant). Having found three failed attempts to prove the relation between machine constant anomalies and upcoming failures, with the current knowledge and skill, it can be concluded that prognostic value of machine constants does not lie in the anomalies themselves.

Fourth Try – Pattern Recognition

The fourth and last model for prognostic value does not look for behaviour after an outlier, but tries to predict the outliers themselves. During the diagnostic value evaluation it was found that there is a relation between the anomalies and the direct machine status at that same moment in time. So, if a model is able to predict the moment in time the anomaly will appear, it should be able to predict a possible outlier significantly better than a random pick. So the fourth model does not contradict with the original statistical model (see paragraph 8.1), but continues with its output.

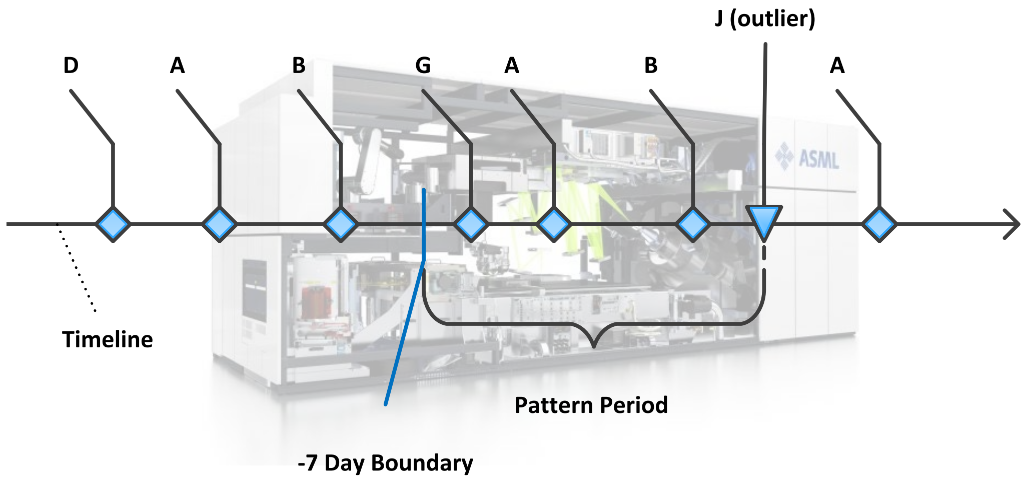


FIGURE 27. PATTERN RECOGNITION

To analyse the machine constant behaviour before an outlier, a pattern recognition algorithm is used. The Generalized Sequential Pattern (SGP) algorithm searches for common behaviour during a period of 7 days, before the outlier (see Figure 27). Each time stamp is labelled with an alphabetic character to represent the area in which the machine constant value finds itself. The areas are based upon normalization of the machine constant values that are not qualified as outliers ($Z < 2$). Figure 28 describes the area labelled *J* (which are the outliers), and the division of the leftover values following the normal distribution. Positive and negative deviations from the mean receive the same label, to increase the chance of finding a pattern. This is based upon the idea that an outlier may show slight machine constant value deviation from the population prior to the actual outlier. Because the labelling is based upon the machine constant value, compared to the whole population, there is large overlap with the OVP sub model included into the model used during the first try. See paragraph 12.6 for a detailed description of the SGP algorithm.

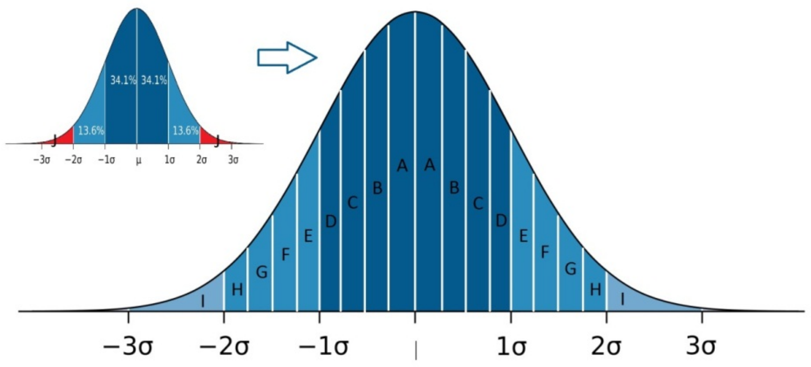


FIGURE 28. AREA DEFINITION

The dataset that is used as input for the fourth try is based upon the outlier list from the first try. Machine constants *X* and *Y* were the most occurring machine constants in downstate in the original outlier list, and therefore selected as filtering criteria for the fourth try (see Table 16). Out of the 134 timestamps that occurred during the period monitored, a total of 23 outliers were identified.

TABLE 16. GSP DATASET

Machine Constants	# Timestamps	# Outlying
X	68	15
Y	66	8
Total	**<u>134</u>**	**<u>23</u>**

The dataset was then implemented into a RapidMiner process (see Appendix 11). The GSP model is customized to find patterns that occur within one week $(time\ window = 7\ days)$ and neglects the maximum time gap $(max\ gap = 0\ days)$, because each machine constant value represents a unique event and there is no benefit in aggregating events. Figure 29 provides an abstract visualization of the pattern recognition process.

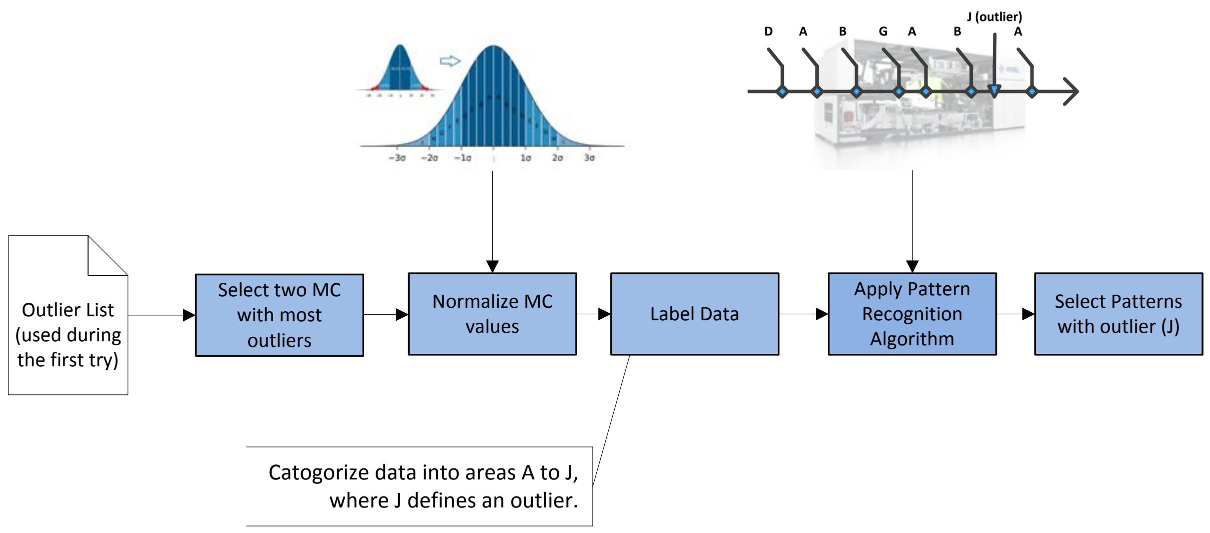

−3σ
−2σ
−1σ
1σ
2σ
3σ
J (outlier)
D
A
B
G
A
B
A
Outlier List (used during the first try)
Select two MC with most outliers
Normalize MC values
Label Data
Apply Pattern Recognition Algorithm
Select Patterns with outlier (J)
Catogorize data into areas A to J, where J defines an outlier.

FIGURE 29. PATTERN RECOGNITION MODEL

Filtering for only those sequences that include an outlying state (J) the following result was obtained:

TABLE 17. OUTLIER SEQUENCE

% Population	Sequence	# Machines	% Outliers
5.2%	$G \rightarrow J$	7	*30.4%*
5.2%	$E \rightarrow J$	7	*30.4%*
5.2%	$D \rightarrow J$	7	*30.4%*
7.5%	$J \rightarrow G$	10	*43.5%*
7.5%	$J \rightarrow F$	10	*43.5%*
6.0%	$J \rightarrow E$	8	*34.8%*
9.0%	$J \rightarrow J$	12	*52.2%*
6.0%	$J \rightarrow G \rightarrow F$	8	*34.8%*

As seen in Table 17, three different sequences that result in an outlier are identified. While it seems quite promising that there are sequences with a slightly abnormal machine constant state (D, E, and G) during the last week before a failure, for 30.4% of the outliers, it still provides low amounts of prognostic value. For example, if a machine is in a G state, the number of machines that show a pattern from that state towards any other non-outlying state is higher for any state then the outlier J (see Table 18). So, if a machine is in a D, E, or G state, the accuracy of the prediction that an outlier will occur within the next seven days is at most 5.2%. The accuracy is most likely even much lower, because the algorithm only counts the number of machines that show the pattern somewhere in the whole sequence. So even if the pattern leading up to a failure is found once, it is possible that the same state does not lead up to an outlier during other parts of the sequence. Together with the fact that not all outliers become direct machine failures, pattern recognition (implemented in this exact way) is an unpromising method for prognostic purposes.

TABLE 18. EXAMPLE SEQUENCE

% Population	Patterns
42.5%	$G \rightarrow A$
42.5%	$G \rightarrow B$
42.5%	$G \rightarrow C$
42.5%	$G \rightarrow D$
48.5%	$G \rightarrow E$
52.2%	$G \rightarrow F$
51.5%	$G \rightarrow G$
6.7%	$G \rightarrow H$
5.2%	$G \rightarrow J$

9.3 Conclusion

Throughout the evaluation phase the relation between machine constants and the current machine status is validated. Both the original model (trial 1) and the unique machine constants model (trial 3) have proven to be able to indicate if a machine is in a down state, statistically significantly better than a random guess. With an accuracy of around 50%, regarding Type II errors and True Positives (see paragraph 4.3), the model close to doubles the diagnostic value compared to having no machine constant information. The fact that the Type II error has a high value of 50% is no problem, because the information about outlying machine constants will only be used in case the machine is already down. So, false warnings do not have any negative effect on the business process. Because all the models used throughout the thesis only monitor a fraction of the total amount of machine constant, it is impossible to qualify the accuracy regarding Type I errors and True Negatives (see Table 19). Only if all machine constants are taken into account, the exact diagnostic accuracy of the model can be obtained. So far, it can be concluded that there is diagnostic value in machine constants and that this information seems worth implementing in current (Automated Recovery Engine) or new *diagnostic* tools.

TABLE 19. DIAGNOSTIC VALUE

		Predicted State	
		Healthy	Broken Down
Actual State	Broken Down	*Unknown%* Type I error	*50%* True Positive
	Healthy	*Unknown%* True Negative	*50%* Type II error

After performing four different tries to validate the hypothesis that machine constants have a *prognostic* value for future machine health, it can be concluded that there is no prove to accept this hypothesis and that it is even more likely that it can be rejected. See Figure 30 for an overview of the process of prognostic value validation. Despite with the possibility of applying completely different algorithms, approaches, or implementing expert knowledge, it is unlikely that machine constant could have any prognostic value. Applying the confusion matrix (see paragraph 4.3) to any of the above models is not useful, because it was already concluded that the models do not have any prognostic value. So if unusual behaviour of a machine constant does not lead to a direct down, it is of no harm to the machine and has no predictive value. The input data and results of all of the four trials can be found in digital appendix A to D.

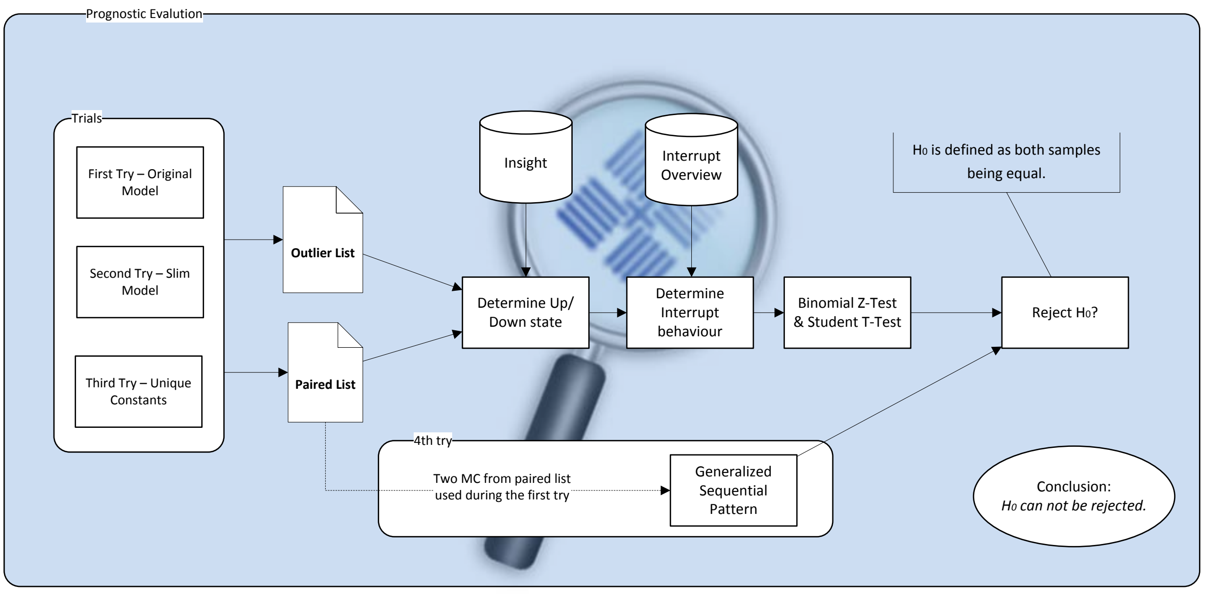

Prognostic Evalution
Trials
First Try – Original Model
Second Try – Slim Model
Third Try – Unique Constants
Outlier List
Paired List
Insight
Interrupt Overview
Determine Up/ Down state
Determine Interrupt behaviour
Binomial Z-Test & Student T-Test
H0 is defined as both samples being equal.
Reject H0?
4th try
Two MC from paired list used during the first try
Generalized Sequential Pattern
Conclusion: H0 can not be rejected.

FIGURE 30. PROGNOSTIC VALUE OVERVIEW

10 Research Conclusions

This chapter uses all obtained results and knowledge from the previous chapters to answer the research questions and goal. The structure of this chapter follows the build-up of the research question paragraph (3.2), building up towards a discussion on how far the research goal is met (10.4), and ending with an overview of the limitations of this research (10.5).

10.1 What does the selected data represent?

The data that was selected consists of machine constant data and the interrupt overview. Machine constants are settings defined by machine software drivers, after automated tests/calibrations, which determine how the machine should perform. They relate to a physical attribute within the lithographical machine and influence the three basic KPI's of ASML's machines; Overlay, Imaging and Throughput (see paragraph 3.3). The interrupt data describes all interrupts that occurred during the selected time period, with detailed background information about these interrupts.

How is the quality of the selected data?

Machine constants: Assuming that the given dataset was a good representation of the machine constant data in general, it can be concluded that machine constant data is stored correctly and reliable. The amount of missing data is negligible (<1%), because it lies below the 10% rule of thumb, and no unacceptable data entries are found. Despite the good start, there are some issues with the data that make it difficult to analyse and interpret. First of all, there is no documentation on the exact meaning of the data available. Some sort of legend or appendix with the meaning of the terms used for each column would provide some more certainty about the interpretation of the data. The second issue with the data quality is the uncertainty with the time stamps. It would be beneficial for the analytical quality if the time between the moment that an event takes place in reality and the time stamp to which it is allocated in the dataset is registered, or if the uncertainty would be described or quantified. Another noteworthy insight about the machine constant values is that a variance of 0 is rather common, while in practice this should not be possible. Machine constants relate to physical attributes of the machine, and there is always some movement (assuming that the machines are not in a zero Kelvin environment). The reason that zero variance is measured could be the cause of rounding off small values during the measurement or storage data. Because this rounding off is most likely not done deliberately, the possible negative side effects of this loss of data are probably not considered. This is something to keep in mind when mining data that is extremely detailed (up to Nano-meters). The last remark on data quality is the fact that the selected dataset is not optimal for clustering. While this is not exactly a flaw in data quality, it influences the quality of the analysis. For future application of cluster algorithms, it is preferred to use a data source that stores more machine constant information per time stamp (instead of only the value deviation).

Interrupt overview: This data source includes no missing data or unacceptable data entries. The fact that the data is manually verified means that there is space for human error. While it is most likely more reliable than non-verified data, it is false to conclude that the data is errorless. While machine error is easier to identify, because it is more structured, human error is almost unidentifiable. Besides this possible flaw, there are no known critics on this data source. The final verdict about the overall quality of the selected data (machine constants and interrupt overview) is that the data is of an on average quality, where the amount of data and percentage of missing data are fine, but the easiness of interpretation of data is below requirements.

How does the selected data behave?

Machine constants: As described in chapter 5, there is all kind of machine constant behaviour. Even for a similar machine constant it is common that the behaviour per machine is significantly different. Some examples of machine constant behaviour are the linear trend, jumps, outliers and a random cloud. It is impossible to identify bad machine states, based on its own behaviour, because there is no description available about the normal behaviour of the machine constants. It could be possible to analyse behaviour individually, if expert knowledge was able to define guidelines for machine constant behaviour. E.g. upper and lower limits of the MC values, un-allowed drifting/jumping/fluctuating of certain constants, or a predefined MC specific failure patterns. Because this information is either unavailable or does not exist at all, the best judgement of the allowed behaviour of a machine constant is the general behaviour of the whole population of machine constants (on the different machines), as validated during the evaluation (see chapter 9).

Interrupt overview: Because this data source does not show behaviour, this will not be discussed.

WHICH TOOLING/SOFTWARE IS APPLICABLE TO THE DATA?

As described in paragraph 5.2, there are many different software packages to choose from. During this explorative phase of a project, the selecting criteria for tools were a short learning time, low costs, and the required features (e.g. the X-means, Regression, and SOM models in RapidMiner). From all of those different options, the combination of MySQL and RapidMiner was selected as the most suited tools for the ASML business environment. Both tools are open source freeware with a strong issue solving community behind them, and have proven to be able to solve the business problem. This could be done in a rather short period, without any data mining experience (Note: some statistical and program code experience is advised).

One issue regarding use of tooling/software to the data is the computational strength. Even with a desktop computer with high level specifications (Intel(R) Xeon(R) CPU, W3565 @ 3.20 GHZ, 3.19GHZ processor with 12.0 GB installed memory (RAM) on a 64-bit Operating System (Windows 7 Enterprise)), it was impossible to perform the analysis the preferred way. The reason for this is that the RAM memory of the computer slowly fills during a run. As soon as the RAM memory almost reaches the maximum level, the speed of performing processes drops extremely, up to the point that the run will take an infinite amount of time to finish. The strange fact is that the developed model (see paragraph 8.1) takes about 30 minutes to run, while only a slight extension of the amount of machine constants used as input leads to an infinite running time. The cause of this extreme growth of running time is unknown and unsolved during this thesis, but it could either be the result of inefficient use of memory by RapidMiner (due to non-optimal settings) or just the result of the high requirements of the many subprocess, loops, and algorithms. Especially with the rise of big data analysis, this is certainly an issue to solve in the future. In general, try to do as much of the pre-processing in MySQL (or any other database software), because this is clearly not the strength of RapidMiner (uneasy and RAM memory consuming).

10.2 Is it possible to extract any useful information from the selected data?

Because the idea behind the machine constants is that they are related to physical attributes within the machine, the expectation was that there is some useful information about machine status in that data. During the evaluation phase, it was found that the constants do indeed relate to machine status. Practice and/or experts should validate if this information is indeed useful. So, it is indeed possible to extract information from the machine constants, and this information is most likely also useful.

WHAT IS A SUITABLE APPROACH FOR ANALYSING A LARGE, RATHER UNKNOWN DATA SOURCE?

Literature provided a suitable approach for analysing large rather unknown data sources, called the Cross Industry Standard Process of Data Mining (CRISP-DM) (see paragraph 12.3). Following a cyclic iterative process of business understanding, data understanding, data preparation, modeling, evaluation, and deployment the unknown data source can slowly be unravelled. Because of the iterative approach of CRISP-DM, it is important to keep track of the previous taken paths. Describing the dead ends protects from running in circles and provides a summary of the options evaluated, useful for future researchers. It can be concluded that CRISP-DM is a useful approach for data mining beginners, for whom it provides a strong guidance through the whole process, and experienced data mining experts, for whom it makes sure that no parts of the process are neglected due to tunnel vision towards the final result.

Regarding the decision between a Data and Knowledge Driven approach, this research question already aims at Data Driven, because the data source is defined as rather unknown. Based upon the knowledge and experience obtained during the thesis, a preference for the Knowledge Driven approach is created. Knowing what to mine for simplifies the mining process and decreases the uncertainty after results are found. For this reason, expert knowledge should be included in the process as soon as possible, or should even be the initiator of a data mining project. After the initial phase, in which the knowledge is transferred, a data mining expert can continue the research on its own, up to the evaluation and deployment phase of the CRISP-DM cycle.

IS IT POSSIBLE TO APPLY ANY OF THE BY LITERATURE SUGGESTED METHODS OF DATA MINING TO THE SELECTED DATA?

Apart from the general CRISP-DM methodology, there was one other main framework, called Statistical Process Control (SPC) (see paragraph 12.7). This classical anomaly detection method proved successful for the particular machine constant data source, and could probably also be applied to other types of machine data. Comparing behaviour of a machine constant with that of the population is the best you can do without any expert knowledge about that machine constant.

From the data mining models found in the literature (see paragraph 12.6), X-means (with CLBOF) and Sequential Pattern Recognition (GSP) where put to practise. While it is difficult to quantify to quality of the output of the models, it is most likely that these models did their task correctly. X-means provided rather unusual results, but this was the result of a small biased dataset. In case of a longer time window, the possibilities of X-means should definitely be considered. Sequential Pattern Recognition worked as required and provided an expected but unsatisfying result of no interesting sequences.

Other models, like ANN, SOM and CaRTs, seemed less suited for this particular business problem, because they focus more on separating a dataset into homogenous regions instead of anomaly detection. This could be useful to identify differences between machines, but not directly for diagnostics or prognostics. During a follow up step of anomaly detection, the result of a cluster algorithm can be used as input to a pattern recognition algorithm to find patterns in the movement of clusters. If a common failure pattern is found, the cluster models could be useful for prognostics. This could be part of the future research on machine prognostics (see paragraph 11.2).

Is it possible to relate the selected data with the current machine status?

During the evaluation phase (see paragraph 9.1) it was found that machine constants indeed statistical significantly relate to machine status. If a machine constant is strongly deviating from the population, chances are higher that the machine is currently experiencing issues. It is difficult to define the exact relation of that constant value and the machine status, because the interrupt could be the result of the abnormal value or vice versa. The machine constant can at least provide some diagnostic information to the engineer on which module of the machine (the module that relates to the machine constant) could be experiencing issues.

Is it possible to relate the selected data with the future machine status and include this into a Preventive Maintenance approach?

During the evaluation phase (see paragraph 9.2) it was found that machine constants do not relate with upcoming machine status and that it is not possible to predict future outliers via pattern recognition. Despite the fact that there are some possible limitations to this thesis (see paragraph 10.5), based upon the acquired knowledge, any implementation of machine constant monitoring into the prognostics of a CBM environment is not recommended (not to be confused with diagnostic value). If the possibility of implementing machine constants into preventive maintenance is still not forsaken, a radically different new approach should be explored.

10.3 What could be the future applications of monitoring the selected data?

Because the machine constants do have diagnostic value, but no prognostic value, monitoring machine constants does not solve the complete business problem. Still it could be valuable to implement machine constants monitoring into the machine diagnosis after an interrupt. The outlying machine constants could be an indicator for the kind of issues the machine is facing. See future recommendations (chapter 11) for a more detailed description on how to implement the research results.

10.4 Research Question

As defined in the research design (see paragraph 3.2), the following thesis goal was defined:

Develop a prognostic/diagnostic tool applicable to machine parts that find themselves in the middle part of their lifecycle (constant failure rate).

The goal of the thesis is partially met, because a conceptual process (see Figure 31) and tool (see chapter 8) were developed that relate machine constants with machine status. The problem is that this relation is only between machine constant values and current machine status (diagnostics), but not between the data and future interrupts (prognostics). The developed tool tries to find outlying machine constant values, in the assumption that this may lead to a machine interrupt.

The concept of outliers is especially useful for the middle part of the machine life cycle (bathtub curve), because this area is described as a rather stable and homogenous environment with a low constant failure rate. This description suits the normal distribution, in which the outlier mechanism works at its best. Another characteristic of this part of the life cycle is that parts do no suffer from wear out, as result of age. Wear out is usually predicted by the trend line of the wear out, while random failures are unpredictable by such a trend line. Monitoring anomalies in the form of outliers is the best alternative in case of random failures. The middle part of the curve being the main focus of this thesis, it can be concluded that the diagnostic value of the tool should be considered for implementation by ASML. While the developed tool is currently in an experimental state, and therefore not ready for real-world implementation, this could be done after a small number of adaptations. This is part of the future recommendations for ASML (see paragraph 11.1). Summarizing:

- Continue the research on, and implementation of the diagnostic value of machine constants.
- Cancel the research on the prognostic value of machine constants. To obtain certainty about this, the limitations (see next paragraph) should be investigated and rejected.

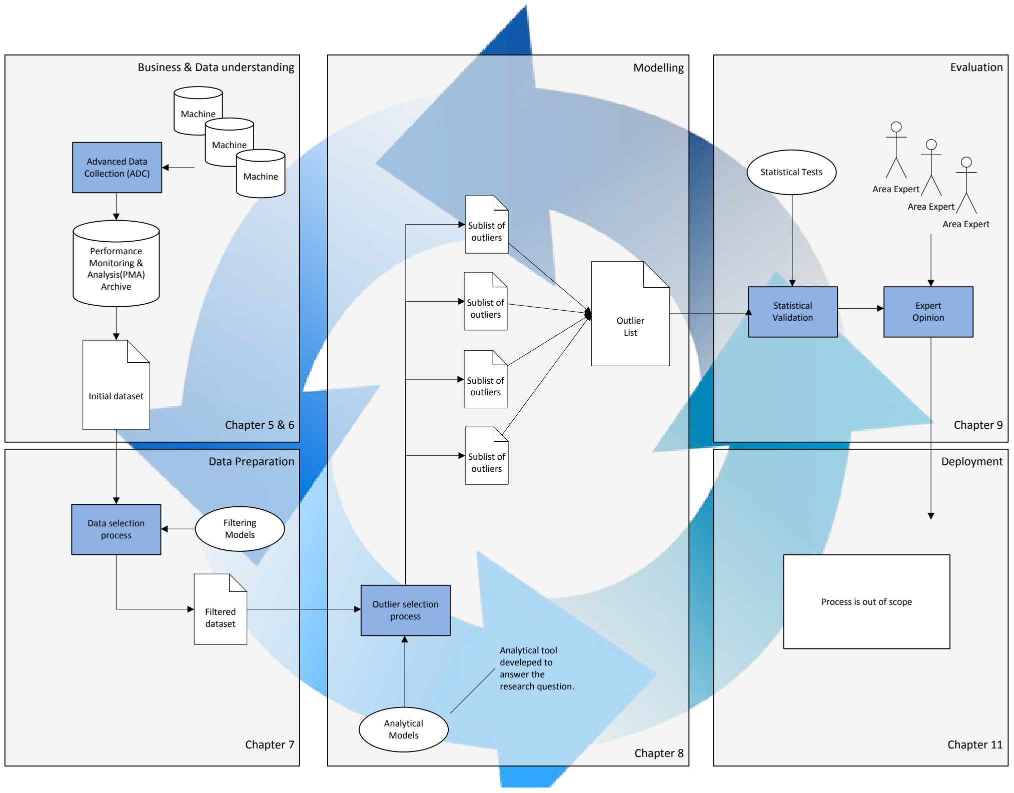


FIGURE 31. CONCEPTUAL PROCESS

10.5 Limitations

Like all research, this thesis has some limitations. It should be kept in mind that these limitations could have influenced the obtained results and conclusions. A distinction is made between research design and research question limitations, because of the difference in fundamental influence on the thesis. Design limitations are insights that were obtained throughout the thesis that could have been overcome if known in advance (hindsight), while question limitations concern fundamental issues that would bring down the whole research approach. The limitations are written down as statements, but it should be noted that they are still subjective.

Research Design Limitations

- The amount of data used was a too small sample of the total dataset to provide significant results. Both the short length of the time frame and the fact that not all machine constants could be analysed makes it impossible to find valid outliers. To be able to increase the amount of used data, the processing strength of RapidMiner should have been increased. This could either have been done via optimizing of the settings or increasing the hardware capabilities (e.g. by running the analysis on a heavier computer).
- Putting the influence of variation in machines, parts and locations out of scope immerses the machine constant values in a cloud of noise, from which no prognostic value can be obtained.
- The major part of the outlying machine constants are data entry errors. Therefore, the control limits are no longer representative for the machine constant population. This results in a list with only the data entry errors instead of real machine constant value outliers.
- The selected time frame is not representative for the general behaviour of machine constants. It could be possible that the customers are in a down turn, in which abnormal low/high amounts of issues occur due to low production, or more preventive maintenance planning (set-up calibrations).
- The validation of the models can only be done in practice or by knowledgeable experts, instead of sole statistically. So the conclusion that the model does have diagnostic and does not have prognostic value is pre mature. It could be the case that, despite the positive results, there is no practical diagnostic value in machine constants. The other way around, it could also be that there is prognostic value in the data, despite the negative results. Both could be the result of evaluation of outliers with expert knowledge, when only '*interesting*' outliers are selected.
- The dataset selected for this thesis contains only a fraction of the total information available at a single time stamp, because only the values of calibrated machine constants are stored. It is impossible to back-engineer this information to a dataset applicable for cluster algorithms.

RESEARCH QUESTION LIMITATIONS

Based on the fact that a lot of other tools also lacked significant prognostic value (e.g. list of tools validated and made by the Spanish consultant team) and the negative results found in this thesis, the following statements on the research question could be true:

- The assumption that machine constants influence machine status is not true. The physical relation between hardware and machine constants is not as direct as assumed.
- General monitoring of machine population cannot be performed, due to the heterogeneousness of the machine and machine constant population (e.g. differences in location, software/hardware versions, client customisation, and workload). Even after incorporating this information, it is impossible to compare machines successfully in any way, because the amount of homogeneous examples would become too limited. A data-driven approach is therefore unsuccessful for such a dataset, and only a knowledge driven approach could extract useful information from beneath the layer of noise.
- The influence of the storage time noise within the data (especially regarding the time stamps) predominates the underlying useful information. This cannot be simply overcome, because the amount of deviation from the real moment in time is unknown and no alternative more accurate datasets are available.

11 FUTURE RECOMMENDATIONS

The future recommendations chapter describes a vision on the activities to perform after this thesis. It starts with a discussion on final phase of the CRISP-DM cycle, called the Deployment phase. Finally several suggestions are given for the future direction of this research topic, with specific focus on the on-going research cooperation between ASML and University of Technology Eindhoven (TU/e).

11.1 DEPLOYMENT

The deployment phase of this research discusses the possibilities of implementing the model. This final phase of the CRISP-DM methodology is out of scope of this thesis and therefore discussed as a part of the future recommendations. While the intention of the research question was explanatory, it is important to define the possible benefits of monitoring machine constants and provide some insights into the implementation process itself.

During the evaluation phase it was found that the constructed model is unable to fulfil the complete requirements of the research question, namely a tool that is able to provide both diagnostic and prognostic value on machine status. Due to the fact that no positive prognostic relation was found between machine constants and future machine status, there is no such model or approach to be implemented at the current moment in time. Despite this shortcoming on the initial requirements, it is reasonable to consider the implementation of the knowledge obtained about diagnostics. The outlying machine constant could give some extra information to the field engineer on the issues that the machine is experiencing, but it was impossible to validate this theory with the current RapidMiner tool and lack of expert knowledge. The first step of the deployment phase would be to extent the time interval on which the RapidMiner tool bases its outliers and include all machine constants, instead of a rather random sample set. It is only possible to execute this step with the supported of enough calculative power, probably from a central server. Increasing the amount of data increases the possibility of finding true outliers, because it strengthens the effect of the Central Limit Theorem (see paragraph 12.7). Including all machine constants makes sure that all outliers can be found (because they are no longer out of scope of the tool) and increases the possible applicability of cluster algorithms (because there is more data available per time stamp). With only a few small adaptations, the existing RapidMiner tool can be extended and further automated (e.g. daily repeated executing, user friendliness, email notification, report building, removing bugs).

The first step before full roll-out would be doing a trial with the help of several field experts and validating the found relation between the outlying machine constant and the machine interrupt. Only if the field engineers find the machine constant information useful in practice there is an urge to continue with the tool. So, it can be stated that the current RapidMiner tool is too experimental to be implemented directly in practice, but is a strong fundament for a future real-world running support tool, and that the theoretical value should first be validated in practice before up scaling the project.

A final note on deployment is the fact that there already was a previous initiative on machine constant monitoring in the far past (2003), which was cancelled due to lack of support. This tool was a basic monitoring tool on outlying machine constant values, similar to the OVP statistical sub model (see paragraph 8.1). Due to inaccurate project documentation, only few relevant objective documents could be retrieved. The '*EDS MC Monitoring Tool'* document describes the layout of the user interface and the required behaviour of the MC monitoring tool, and the *'EPS Requirements MC monitoring tool'* document describes the high level requirements for the MC monitoring tool (see digital appendix F1 and F2). Both documents provide a list of names of people who were involved and could have some tacit knowledge on machine constant monitoring and tooling. As the initiative took place around 10 years ago, this knowledge may be out-dated or just drained away. Because the deployment phase is not executed during this thesis, this whole paragraph could be seen as a first suggestion on future research. It should be noted that this would be more based upon practical knowledge (defined as relevance) then a contribution to research itself (defined as rigour) (Katzav, 2012).

11.2 Future Research

Next to continuing from the point where this research on the possible value of machine constants in a CBM environment ended, there are several other directions the obtained knowledge could lead to.

- The experience with clustering models can be applied to parameter monitoring. In the thesis from Goknur Seyma Cakir (2011), it was found that parameter monitoring could be useful for machine prognostics. Clustering algorithms (e.g. X-means, SOM) are found to be successful in case of a multi-dimensional data environment and are therefore interesting for analysing parameters. Cakir already successfully implemented some of those algorithms, but with the vastly evolving field of data mining there are still lots of topics to investigate for possible implementation.
- Cluster models could also be applied on another dataset that contains all machine constant values of the whole machine population at a several moments in time (periodic or based on calibration moments), instead of only the values that change due to calibrations at the moment of those calibrations. For this new dataset, each time stamp contains large quantities of data for comparing machines within the population. Or cluster algorithms could be used to pre-process the input data of a pattern recognition model, to base prognostics on the movement between clusters over time.
- Enrichment of the information environment by adding data concerning the business environment, the machine (such as age of the machine, output, location, and up/down time), or the machine constants themselves (such as manufacturers specifications, and ASML internal threshold values).
- Investigate all of the uncertainties discussed during the limitations paragraph (10.5).

12 LITERATURE REVIEW

In this section an overview of the theoretical background used in this thesis is given. The literature review starts off with a short description of prognostic and diagnostic tooling in the field of condition based maintenance. After this introduction it continues with the main topic; *data mining,* and then slowly zooms into the more detailed methodology. The final part of the literature review is unrelated to data mining and focuses on Statistical Process Control. Condition Based Maintenance (CBM) is not reviewed because this thesis is part of an on-going cooperation of the University of Technology Eindhoven (TU/e) and ASML in the field of Condition Based Maintenance. More information about the CBM topic can be found in previous master theses from Bas Timmermans (2012) and Goknur Seyma Cakir (2011).

12.1 DIAGNOSTICS AND PROGNOSTICS

Diagnostics and prognostics are two different ways of using data to analyse the status of a system. Both procedures are important parts of a Condition Based Maintenance program, where system status analysis is fundamental task.

DIAGNOSTICS

The procedure with the longest history of practical application is diagnostics. The goal of diagnostics is to identify the nature or cause of a fault, which is done via detection, isolation and identification of faults when they occur (Jardine, Lin, & Banjevic, 2006). Detection should indicate when something is going wrong in the monitored system, isolation should provide an indication on where the problem is located (e.g. which component failed) and identification should detect the nature of the fault (e.g. rust on a bearing). Diagnostics is a procedure that is executed after a system has failed, following the *'fireman'* approach.

PROGNOSTICS

The goal of prognostics is to provide an accurate prediction of the future behaviour of a system, with particular interest in the prediction of failures (Jardine, Lin, & Banjevic, 2006). A commonly used prediction is the *Remaining Useful Life* (RUL), which is an indication of the expected time between now and the first upcoming failure. The prediction is defined by the expected time and likelihood that a fault will occur in the upcoming time period. Where diagnostics is a posterior event analysis, prognostics is a prior event analysis. Prognostic models can be categorized into data-driven, model-based, or hybrid.

When comparing diagnostics with prognostics, the possible economic value of prognostics is much higher. Prognostics is a better procedure to achieve a zero-downtime environment, because it is pro-active instead of re-active. By preventing failures or preparing for the upcoming events, via planning of the required human and mechanical resources, unplanned maintenance cost can be saved. Despite this advantage, diagnostics is an important backup procedure which should not be completely neglected in practice. The first reason for this is that prognostics is a procedure, which like any other prediction technique, is unable to give a prediction without uncertainty. Without this 100% certainty about upcoming failures, some unsuccessful prediction will still lead to unexpected failure. For these cases, diagnostics is the best re-active solution for achieving the shortest down time. Besides uncertainty coverage, diagnostics is also the basic information provider to the prognostic tooling. The failure mechanisms should be known and analysed before future predictions can be given. Finally, the diagnostic information is also useful as feedback for system redesign or future designs. So it can be concluded that for a strong CBM environment, a cooperation of both diagnostics and prognostics is preferable (Jardine, Lin, & Banjevic, 2006).

12.2 Data Mining & Knowledge discovery in Databases (KDD)

The field of data mining is the main area this thesis aims at. It is a broad topic, which can be summarized as a computational process of discovering patterns in large datasets. Data mining is a major part of the emerging abstract field of knowledge discovery in databases (KDD), which is concerned with the development of methods and techniques for making sense of data. Data mining's contribution to the KDD process consists of applying data analysis and discovery algorithms that produce a particular enumeration of patterns or models for the data applied to. Other important steps in KDD are data collection, data preparation, data interpretation and reporting (Fayyad, Piatetsky-Shapiro, & Smyth, 1996).

Data mining involves methods from the fields of artificial intelligence, machine learning, statistics, and database systems. The purpose of a data mining process is to retrieve implicit, previously unknown and potentially useful information from a dataset and transform it into an understandable form of output (Chen, Han, & Yu, 1996). Besides the raw data analysis, the process involves data and data management aspects, data pre-processing, model selection, post processing of discovered structures, and visualization. With the rapid growth of the amount of data that can and is being stored, the new techniques and tools that are invented, and the availability and affordability of processing power, the research area of data mining got increasingly important over the last decades. The patterns that the data mining process tries to discover can be clusters, anomalies, and dependencies within the data records. The process of analysing the data is done fully or semi-automatically. The discovered knowledge can then be applied to information management, query processing, decision making, process control, and other decision support systems (Chen, Han, & Yu, 1996).

12.3 Cross Industry Standard Process for Data Mining (CRISP-DM)

The theoretical framework which is the red line throughout the thesis is known as the Cross Industry Standard Process for Data Mining (CRISP-DM) (Wirth & Hipp, 2000). This model describes an approach for data mining commonly used by expert in this area. CRISP-DM breaks the process of data mining into six major phases; Business Understanding, Data Understanding, Data Preparation, Modeling, Evaluation, and Deployment. These different phases do not form a strict sequence and even repeating steps several times may be necessary. Figure 32 displays the general structure and flow of the CRISP-DM model. For example, while cleaning and preparing data, it may turn out that certain data is unusable, or that data from a previously unidentified source is required to be added (Zhang, Zhang, & Yang, 2003). The model also does not end after the Deployment phase of a data mining project, because often new questions and projects will arise. The outer circle in the diagram represents this cyclic nature of Data Mining itself.

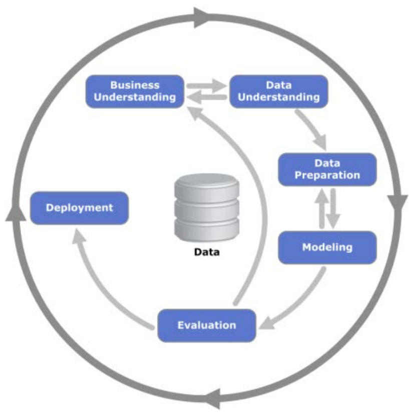


FIGURE 32. CRISP-DM FRAMEWORK

Pardo et al. (2002) explain the six phases of the CRISP-DM model in more detail:

Business Understanding

At the start of a project the highest priority is the understanding of the project objectives and requirements from a business perspective. This general understanding should then be converted into a data mining problem definition, project objectives, data to be used, and a preliminary plan. The goals of a knowledge discovery project must be identified and must be verified as actionable. This makes sure that, if the goals are met, the newly discovered knowledge can be put to use (Chen, Han, & Yu, 1996).

Data Understanding

The data understanding phase starts with an initial data selection and collection. To understand the data in more detail it is necessary, to identify data quality problems, to discover first insights into the data, or to detect interesting subsets. The exploratory data analysis consists of two parts; calculating the summary statistics, and plotting the data. A statistic is an estimation (obtained from data) of the parameters characterizing the underlying probability distribution of the whole population. By calculating the summary statistics (such as the mean and variance) for each feature and for each class a first insight of the data can be obtained. An even more useful way of an initial analysis of the data is to construct a graphical view of the data. Histograms and scatter plots of pairwise combinations of variables can successfully be applied if the dimensionality of the data is not too high. The most informative option is to plot the data projections in two dimensions. This plotting can already be considered as part of the data preparation phase, because it reduces the dimensionality in a similar way to feature extraction. Commonly used unsupervised method during the data understanding phase is the principal component analysis (PCA). If possible, the data understanding phase should be supported and validated by a data expert. When a better view of the data is obtained, hypotheses about the hidden information in the data should be formed.

Data Preparation

The data preparation phase covers all activities to adapt the initial raw data in such a way that it can be fed into the model tool. Data preparation changes the raw data into a final dataset. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. The pre-processing of the data can be done by encoding and the dimensionality reduction mentioned during the data understanding phase, using data transformation, data cleaning, data reduction, and data discretization. For these steps it may be possible to apply the a priori knowledge obtained during the understanding phase. Pre-processing should make the task of estimating simpler by reducing the number of informative features in the dataset.

Modeling

In the modeling phase, various modeling techniques are selected and applied to the dataset, and their parameters are calibrated to optimal values. Models are trained with a certain selected dataset, to be able to be applied to another set of data later on. There has been much research in the field of data mining, and many data mining techniques and systems have recently been developed. Different classification schemes can be used to categorize data mining methods and systems based on the kinds of databases to be studied, the kinds of knowledge to be discovered, or the kind of techniques to be utilized (Chen, Han, & Yu, 1996). An effective classification scheme to use is *the kinds of knowledge to be mined,* because it presents a clear display on different data mining requirements and techniques. Most important classes in this scheme are; association rules, characterization, classification, and clustering. Each of these classes can be further split up into different approaches (Chen, Han, & Yu, 1996). Data mining problems can usually be solved by several different mining techniques, involving different kinds of models. Some techniques have specific requirements on the form of data. Therefore, stepping back to the data preparation phase is often needed. It is difficult to point out a best approach beforehand, therefore it is important to try different approaches and compare the results. It is important to stress the difference between the best model and the actual best guess, which can differ due to over-fitting on the learning data. The models applied in this thesis are described in 12.6.

Evaluation

At this stage in the project, via several iterative steps, a model with high data analytical qualities is constructed. Before deploying the model into the business environment, it is important to thoroughly evaluate the model and review the whole process from start-up (Wirth & Hipp, 2000). Models used for supporting human decision making need to be interpretable in order to be implementable, because humans are not likely to base their decision on the result of so called ‘black-box’ models. The goals of accurate prediction and interpretation are somewhat contradictory, since this required high accuracy can only be obtained by quite complex models. Over time, the balance between accuracy and simplicity slowly moves towards the more complex models with high accuracy (compared to the traditional statistical approach). Providing an interpretable view of the result produced by these high-dimensional models is being viewed as a separate task and is still an active area of research. At the end of the evaluation phase, a decision on the use of the data mining model should be made (Wirth & Hipp, 2000).

Deployment

The final phase of the CRISP-DM framework is the deployment of the model made. This can be as complex as implementing a complete repeatable data mining process or as simple as writing a report. Most of the times, the actual deployment will not be carried out by the data analyst, but the customer the model was built for. For the customer to be able to do this it is of utmost importance that the analyst informs the customer about all the actions that need to be carried out in order to successfully deploy the model. Deployment of the model is usually not the end of the project, because even if the purpose of the model is to increase knowledge about the data, the knowledge gained will need to be organized and presented in a way that the customer can use it (Wirth & Hipp, 2000). Furthermore the whole project may lead to new data mining project for further investigation of business processes.

12.4 Principle Component Analysis (PCA)

The Principle component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert high dimensional data into a lower dimensional dataset. This unsupervised dimension reduction is used in very broad areas such as meteorology, image processing, genomic analysis, and information retrieval (Ding & He, 2004). It is mostly used as a tool for exploratory data analysis and predictive purposes. PCA reduces the dimension by picking up the dimension with the largest variances, which lead to a number of principal components that is always less or equal to the number of original variables. The first principle component (with the largest variance), represents the most variability of the data. Upcoming principle components decreasingly represent as much variability as possible, taking into account that the selected component is uncorrelated with previous selected components (Ding & He, 2004). Principle component analysis can be done via eigenvalue decomposition of the data correlation matrix. For better results, the data should be centered normalized or Z-scored for each attribute. The principle component analysis returns component scores, and the underlying loadings. Based upon these scores, only the first few principal components can be selected so that the dimensionality of the transformed data gets reduced. Principle component analysis is one of the simplest eigenvector-based analyses for revealing the internal structure of a dataset (Malhi & Gao, 2004).

12.5 Pattern Recognition

In machine learning, pattern recognition is defined as a set of mathematical, statistical and heuristic techniques of fundamental importance in executing 'man-like' tasks on computers (Nieddu & Patrizi, 2000). Pattern recognition is applied in many fields, such as ethology, psychology, cognitive science, traffic flow, and computer science. In practice, pattern recognition can be described as the process of assigning a certain label to a given input value (Jain, 2010). The pattern recognition algorithms aim at providing a most likely match of an input value with the rest of the dataset. This approximation is the one feature of pattern recognition that distinguishes it from pattern matching, in which an exact match with a pre-existing pattern is required. The four best known approaches for pattern recognition are; template matching (closely related to pattern matching), statistical classification, syntactic or structural matching, and artificial neural networks (Jain, Duin, & Mao, 2000). For a long time, pattern recognition problems have been solved by using a classical statistical classification approach. Due to the lack of calculation power more general modeling methods, such as neural networks and fuzzy logic, were not applicable. Compared to the statistical methods and models, these new methods are more appropriate for solving real-world pattern recognition problems with a higher accuracy and efficiency (Melin & Castillo, 2005). The three essential aspects of the design of a pattern recognition system; data acquisition and pre-processing, data representation, and decision making are easily implemented into the CRISP-DM theoretical framework (Jain, Duin, & Mao, 2000).

Clustering is an important step in exploratory data mining, and an approach commonly used for data analysis in the fields of; machine learning, pattern recognition, image analysis, and information retrieval (Pardo & Sberveglieri, 2002). It groups a set of data (without a predefined class attribute), based on the conceptual clustering principle: *maximizing the intra-class similarity and minimizing the interclass similarity* (Chen, Han, & Yu, 1996). The goal is to develop an algorithm that can discover the natural groupings in the unlabelled data. The cluster itself can be defined as a high density region in the feature space, separated by a low density region (Jain, 2010). The clusters in one dataset can differ in terms of their size, shape, and density. Together with the noise that is always present in the data, the detection of clusters can be rather difficult. The ideal cluster that can easily be identified is a set of points that is completely isolated and compact. In most real-world cases the clustering is a subjective method that requires domain knowledge to be interpreted and implemented successfully. While the human brain is excellent at identifying clusters in a low-dimension space (up to three dimensions), computing assistance is required for high-dimensional data. To solve all these difficulties with data clustering, dozens of clustering algorithms have been published and continue to appear.

The most important categories of data clustering algorithms are; hierarchical clustering, k-means clustering, distribution-based clustering, and density-based clustering (Jain, 2010).

Modeling can either be done supervised or unsupervised (Alhoniemi, Hollmen, Simula, & Vesanto, 1999) (Jain, 2010). In supervised learning, each input value (typically a vector) is linked to a desired output value. A supervised model analyses the training data and learns how to classify each new input into one of the known output values. To be able to use supervised learning, it is therefore required to have input learning data in which different classes are already recognised and indicated as such (from the moment of collection or after data pre-processing). If such a priori knowledge is not available, one should consider the use of unsupervised learning. With unsupervised learning, the goal is to try to find hidden structure in unlabelled data. Compared to the supervised learning, unsupervised leads to more of a 'black-box' modeling, in which the process from input to output is more difficult to understand and validate (Alhoniemi, Hollmen, Simula, & Vesanto, 1999). The basic methodologies behind unsupervised learning are data mining methods used for pre-processing data (Pardo & Sberveglieri, 2002), such as clustering (e.g. artificial neural networks, hierarchical clustering, mixture models, k-means) and dimensionality reduction (e.g. principal component analysis, independent component analysis, singular value decomposition). Among the artificial neural network models, the self-organizing map (SOM) is a widely used unsupervised learning algorithm (Hüllermeier, 2005).

12.6 Data Mining Models

Several models that can be used in the pattern recognition are given in the following subsections.

Artificial Neural Networks (ANN)

Neural networks are computational models with learning or adaptive characteristics. The method is based upon a model used for analysing the human brain. Natural neural networks consist of neurons and connections between them. The neurons are modeled with nodes and the connections are arcs, which together form a computational neural network. This network in combination with a computational algorithm is what makes neural networks a powerful methodology to simulate complex behaviour. The neurons form representative clusters of the data if is applied to. Neural networks models can be both supervised and unsupervised. Supervised uses both input and output data in the training phase, while unsupervised only focuses on the input data. Building an ANN takes place in two steps; training and applying (mapping). Neural networks are used to solve problems in a wide range of fields such as, system identification, control, diagnosis, time series prediction, and pattern recognition. For pattern recognition problems neural networks are an efficient solution for achieving the level of efficiency and accuracy required for real-world applications. Neural networks are able to deal with the great complexity and high dimensional spaces that are present in pattern recognition problems (Melin & Castillo, 2005). A commonly used type of artificial neural network used for pattern recognition problems is the Self-Organizing Map (SOM), also known as Kohonen-Network (mainly used for feature mapping and data clustering). Despite the obvious different underlying principles, there is a significant similarity between these neural network models and the classical statistical pattern recognition methods (Jain, Duin, & Mao, 2000).

SELF-ORGANIZING MAPS (SOM)

A Self-Organizing Map (SOM), or Kohonen Map, is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a (typically) two-dimensional map of the training sample it receives as input (Kohonen, 1990). SOM is different from ANN in the sense that it maintains the topological properties of the input space by using a neighbourhood function. This way, SOM makes it possible to visualize high-dimensional data on a 2D map (Kohonen & Honkela, 2007). Similar to ANN, SOMs are built in the steps training and mapping. Training builds the map using the example input data, while mapping automatically classifies the data based on the learned model.

A self-organizing map consists of components called neurons (or nodes), which are associated with a weight vector of the same dimension as the input data, and have their own unique position on the map (Wu, 2012). The initial weights of the neurons should be selected randomly. In the training phase, input vectors are presented to the network in a random order. For each input vector, the (weighted) closest neuron is identified. The second step is to find all the neurons that are within the predefined neighbourhood of the closest neuron and update their weight vectors to move towards the input vector. The selection of the best neuron and their neighbours can be determined by calculating the Euclidean distance between the input vector and weight vectors. So for each input value, certain parts of the map are manipulated to represent the input patterns found in the original dataset. To change the map from the original random placement to a representation of the data pattern it is important feed a large number of example vectors to the map. Input vectors can be used several times as iteration to form the final map (Kohonen & Honkela, 2007). The goal of this is to provide a low dimensional map space from a higher dimensional input space for better pattern recognition. When the map is trained and the patterns in the input data are found (if actually present) the SOM can be put to practice. During this mapping process, an input vector is appointed to the closest weight vector and therefor represented by that neuron. The unique characteristic of SOM, compared to learning vector quantization (LVQ), is that it defines a neighbourhood (Jain, Duin, & Mao, Statistical Pattern Recognition: A Review, 2000). Self-organizing maps are typically considered as type of feed-forward networks, despite the fact that the type of architecture behind it is significantly different in motivation and arrangement. While for a small number of nodes the SOM behaves similar to the K-Means method, for larger settings the SOM distinguishes itself due to its topological characteristics (Jain, Duin, & Mao, Statistical Pattern Recognition: A Review, 2000). The SOM algorithm can simply be formulized in the following list of steps (Choe, 2008):

```
1. Randomly initialize weight vectors $w_i$

2. Randomly sample input vector $x$

3. Find Best Matching Unit (BMU):

$i(x) = argmin_j \|x - \mathrm{w}_j\|$

4. Update weight vectors:

$\mathrm{w}_j \leftarrow \mathrm{w}_j + \eta h(j, i(x))(x - \mathrm{w}_j)$

$\eta$: learning rate

$h(j, i(x))$: Neighborhood function of BMU.

5. Repeat steps 2 – 4.
```

Figure 33 represents an example of the use of SOM, as the result of a data analysis done by the World Bank statistics of countries in 1992 (Neural Networks Research Center, 1997). 39 variables concerning quality-of-life factors, such as state of health, nutrition, educational service, etc., were used. SOM joins these factors and visualizes the countries on a map that presents similar circumstances in each area of the map. The different clusters on the map were automatically encoded with different bright colours, nevertheless so that colours change smoothly on the map display. As a result of this process, each country was in fact automatically assigned a colour describing its poverty type in relation to other countries.

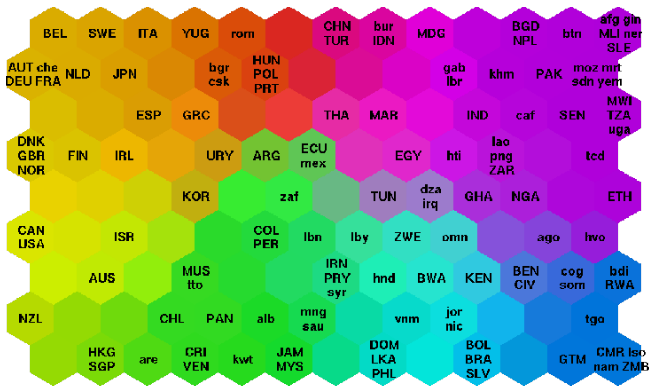


FIGURE 33. SELF ORGANIZING MAP WORLD BANK STATISTICS OF COUNTRIES

Statistical Classification K-Means

K-means clustering is a method of cluster analysis which groups the total dataset into the predefined number of cluster, to obtain the highest intra-class similarity and lowest the interclass similarity (Chen, Han, & Yu, 1996). This supervised method does so by minimizing the sum of the squared error over all the clusters. Figure 34 shows a simple example of the output after applying K-means clustering. Even for a low number of clusters (K=2) the optimization is a NP-hard problem, which can only be solved by a greedy algorithm. With exception of a dataset with distinctly separated cluster, the greedy algorithm will converge to a local instead of a global minimum. The main steps of K-means algorithm are as follows (Jain, 2010):

1. Select an initial partition with K clusters; repeat steps 2 and 3 until cluster membership stabilizes.
2. Generate a new partition by assigning each pattern to its closest cluster center.
3. Compute new cluster centers.

The user is required to provide three user-specified parameters to the K-means algorithm; (K) number of clusters, cluster initialization, and distance metric. The most important input of these three is the number of clusters to identify. No clear guidelines exist for selecting the optimal number of clusters, but a number of heuristics are available to assist. For successful clustering it is recommended to run the K-means algorithm several times, with different number of clusters. The most meaningful result can then be selected by the domain expert (Jain, 2010).

The second user selection is the cluster initialization. Different initializations can lead to different final clustering because K-means only converges to local minima. One way to overcome the local minima is to run the K-means algorithm with multiple different initializations and choose the partition with the smallest squared error. So to select the optimal setting for K and the initialization it is required to run several different K settings, with several different initializations (Jain, 2010).

The final parameter selection influences the calculation of the similarity, based on the distance between points and cluster centres. Commonly used settings are Euclidean, Mahalanobis, L1, Bregman, and Itakura-Saito distances (ranked on popularity). Euclidean is most used and is suited for finding spherical or ball-shaped clusters. The Mahalanobis specializes on hyper-ellipsoidal cluster, at the expense of higher computational cost.

Itakura-Saito. L1 is used to model the distance on a square grid, while Bregman and Itakura-Saito are non-symmetric variants of the Euclidean and Mahalanobis distances (Jain, 2010). When applying K-means to high-dimensional data the use of the squared Euclidean distance is not suitable, because of the 'curse of dimensionality'. Research has proven that one of the solutions to this problem is to use alternative distance functions such as the cosine similarity Euclidean distance. Another way to tackle this problem is to employ dimension reduction for high dimensional data, with traditional methods such as the Principal Component Analysis, Multidimensional Scaling, and Singular Value Decomposition (Jain, 2010).

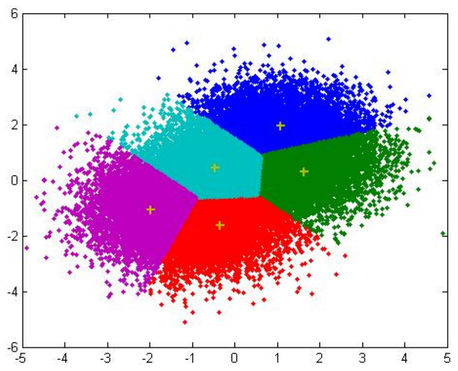


FIGURE 34. K-MEANS CLUSTERING

X-MEANS

X-Means is a slight adapted K-Means clustering algorithm, which determines the correct number of centroids (K) based upon a simple but effective heuristic (Pelleg & Moore, 2000). The heuristic starts with a minimum set of centroids and subsequently increases the number of centroids by splitting them op. Which one centroid should be split and where both new centroids should be placed, is based upon the Bayesian Information Criterion (BIC) developed by Kass and Wasserman (1995). An increase in the number of centroids leads to more precision, but has a drawback of increasing the complexity. The BIC criterion balances the trade-off between model precision and complexity. The adaptation from K to X-means improves the unsupervised learning ability of the algorithm because an input value of K is no longer required.

CLUSTER-BASED LOCAL OUTLIER FACTOR (CBLOF)

To be able to use clustering for outlier detection an extra algorithm is required. The algorithm assigns a factor to each object, called the Cluster-Based local outlier factor (CBLOF), to represent the significance of an outlier (He, Xu, & Deng, 2003). The algorithm determines the value of the factor upon the size of the cluster the object belongs to and the distance between the object and its nearest large cluster centroid. The algorithm is a cluster extension to the basic density-based local outlier algorithm introduced by Breunig et al (2000). An important part of the CBLOF algorithm consists of the distinction between small and large clusters, determined by the selection of input parameters alpha (α) and beta (β). The whole idea behind this distinction is that an object assigned to a small cluster, that finds itself far away from other clusters representing the majority of the population, is more likely to be an outlier. Alpha represents the percentage of the population that is required to call a cluster large. Beta is the factor that defines the number of times a small cluster is smaller than a large cluster. E.g. if $\alpha = 30\%$ and $\beta = 3$, a small cluster holds at most 10% of the population and a large cluster at least 30%. The higher the determined CBLOF is, the more likely it is to be a significant outlier.

Classification and Regression Trees (CaRTs)

Classification and regression trees (CaRTs) are decision tree - machine-learning - methods for constructing prediction models from data (Loh, 2011). A decision tree is a decision support tool that uses a tree-like graph to visualize the behaviour of data. They are commonly used in the field of Operations Research, due to the advantage of being easy to understand and interpret by non-experts. In data mining, there are two main types of decision trees; classification and regression (Lemon, Roy, Clark, Friedmann, & Rakowski, 2003). Classification trees predict the class to which a certain data vector most likely belongs based on the training data, while regression trees predict a certain continuous value. The two different approaches may also lead to different decisions on splitting the tree. The term CART is an umbrella term for an application of any of the two decision trees (Speybroeck, 2012). The learning phase of the modeling is done by reclusively partitioning the data space and fitting a simple prediction model within each partition. The process of top-down modeling the decision tree is an example of a greedy algorithm, which is a commonly used approach in machine learning application. The greedy algorithm splits the data by selecting the variable for which a split leads to the most homogeneous branches, after the split. This partitioning leads to the distinctive tree shape (see Figure 35) of this type of data analysis (Loh, 2011).

CART is a valuable approach for analyzing complex interactions in complex data and returning a result, interpretable even by non-statisticians. It is able to handle with missing values in both response and explanatory variables, which is the case in most datasets. The missing values of explanatory variables are based upon the values of non-missing values, called surrogate variables (Speybroeck, 2012). CART is able to work with a wide range of response variables. The downturn of this flexibility is that it is a rather blunt instrument, which divides data into two subgroups based only on one explanatory variable at a time. Another difficulty is the decision on the depth of the tree, because a large tree will give a more accurate but over fitted model while a small tree does not provide enough useful information about the behavior of the data. To solve these problems it is suggested to hypothesize the relationship between variables a priori (Speybroeck, 2012).

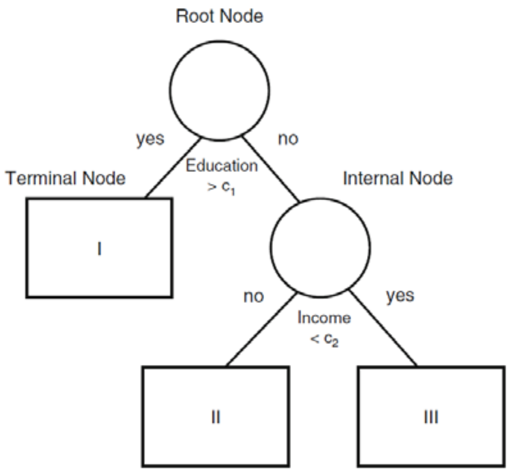


FIGURE 35. CLASSIFICATION AND REGRESSION TREE

Sequential Pattern Recognition (GSP)

To recognise a pattern in an example set, where values are delivered in a sequence, the Generalized Sequential Pattern (GSP) algorithm can be used. The GSP algorithm is able to discover all sequential patterns with a user-specified minimum support, which is the number of data-sequences that contain that exact pattern (Srikant & Agrawal, 1996). GSP analyses a dataset repetitively, where the length of the candidate sequence increases per round. The algorithm starts with selecting all single items that occur frequently to form solo sequences. The solo sequences form the starting dataset for the second up to the final repetition of sequence recognition, where only the supported sequences of previous length are used as input for the next round.

GSP can be customized on the *time window* is should analyse and the *maximum time gap* between two succeeding labels to be related to one event. Having more than one label per event is a feature that GSP can identify and represent via a different kind of sequence. E.g. Sequence $[1] \rightarrow [2] \rightarrow [5]$ is different from $[1] \rightarrow [2; 5]$, because the first sequence does not state anything about the time difference between value B and C while the second sequence clearly states that value B and C take place during the same event (within the maximum time gap). The time window itself represents the maximum time from the first event up to the last event that can be identified as one sequence. In practise, this represent the maximum time between events where for a relation between events can be claimed. See Figure 36 for a visualization of the GSP algorithm and the given example.

GSP scales linearly with the quantity of data, and has strong up scaling properties compared to the average pattern recognition algorithm (Srikant & Agrawal, 1996). This makes the GSP a suitable algorithm for sequential pattern recognition.

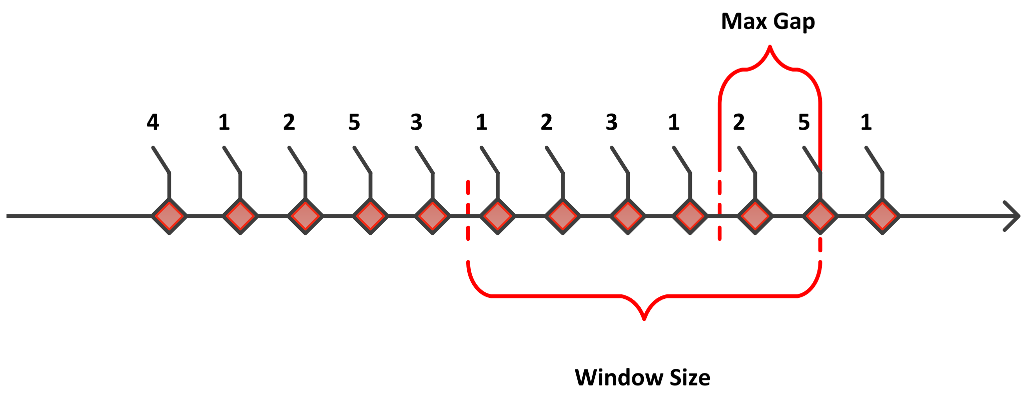


FIGURE 36. GENERALIZED SEQUENTIAL PATTERN

12.7 Statistical Process Control (SPC)

Statistical process control (SPC) is the application of statistical technics for tracking and controlling a process, meant for achieving and maintaining a highly efficient business process. If a process is declared efficient depends on the quantity of conforming products it can make (throughput) and the amount of waste (rework or trash) it produces doing so. Statistical process control can be applied to any process for which the output can be measured (Oakland, 2003). Statistical Process Control was pioneered by Walter A. Shewhart at Bell Laboratories in the early 1920s.

SPC makes use of control charts and design of experiments to focus on continuous improvement. Compared to other methods of quality control, such as inspection, SPC emphasizes on early detection and issue prevention instead of corrective action after the problem has already occurred (preventive versus corrective maintenance). To do so, SPC predicts significant process variation that may result in future process issues. Any source of variation can be classified into one of the following two classes (Oakland, 2003) (Flott, 2012):

- Common variation; sometimes referred to as "normal" or "chance" sources of variation. Considered as uncontrollable variation without identifiable cause, resulting in a noise around the stable behavior.
- Assignable variation; sometimes referred to as "special" sources of variation. Considered as an offset to the normal behavior due to an event or process occurring over time. Instead of a noise like behavior, the variation significantly influences the process and may require an intervention.

This classification of variation is very important for the prediction of process issues. While most processes have many different sources of variation, SPC should only focus on those few sources that have significant influence on the process quality, and ignore the majority of minor variation sources. The dominant assignable variation sources should be detected as soon as possible, after which the cause can be identified and removed, to return the process to its stable state. The stabilized process will still experience the common variation, but should remain within the known set of limits (for as long as another assignable source of variation is not present). Applying SPC involves three main sets of activities to improve a business process; understanding the business process via process mapping, measuring & controlling the sources of variation via control charts, and identifying & eliminating problematic sources of variation via expert process knowledge (Evans & Lindsay, 2010).

When a process is defined as out of control, with the use of the control charts, several activities can be executed to identify the source of excessive variation. Commonly used tooling includes the Ishikawa diagrams, Pareto Analysis, and designed experiments. After detailed problem identification, management can decide to try and eliminate the source of variation with newly developed standards, staff training, error-proofing or by changing the process itself. With a structural execution of the three main steps of statistical process control it is possible to reduce process issues. One should keep in mind that the success of SPC does not only require the right skills, but also depends on how suitable the process is. This represents the main disadvantage of SPC; it is difficult to judge whether or not application of SPC is appropriate for a certain business process (Oakland, 2003) (Thor, et al., 2007).

Control Charts

Control charts, also known as Shewhart charts (named after the inventor of the chart, and pioneer in the area of quality control; Walter A. Shewhart), are valuable tools for quality control. Dating back from 1924, control charts are used in statistical process control and mostly used to determine if a business or manufacturing process is under control. Control charts are particularly appealing for quality control since they can be constructed and used by persons without extensive knowledge of statistics or process behaviour (Evans & Lindsay, 2010). The purpose of this method is to allow for simple detection of behaviour that is an indicative of actual process change. Due to the fact that most process characteristics continuously vary, the value of the control chart method is to distinguish unusual from normal variation. Identifying unusual process behaviour can be based upon all kinds of process related data, such as quality measures, output measures, event counters, interval timers, and resource measures accumulated over a fixed time interval or between two related events (Denning, 1987).

To define unusual behaviour, control charts use control limits. These upper and lower limits are the boundaries for which the process is considered under control. As long as process values are within these boundaries, there is no reason to suspect process problems or apply corrections and changes to the process control parameters. The boundaries provide a confidence interval for acceptable normal behaviour. In addition to ad hoc decision making, the behaviour within the limits can be used for forecasting the future behaviour and so provide information for, or advice on, preventive maintenance (Neave, 2002). If the upper or lower limits are surpassed, it is assumed that the monitored process is not in control and an intervention should be executed. It should be noted that surpassing the limits provides no certainty about process failure, but should be interpreted as a strong indication that further analysis of the process is a necessity from a business point of view. In other words, try to find out why the process is behaving in an unusual way and if this abnormality may lead to problems (e.g. lower quality, lower output, or even total machine breakdown) (Oakland, 2003). For the preventive use of the control charts it is beneficial to add warning limits or subdividing the chart into zones for early notification. When a process finds itself in a certain part of the chart, a warning message can be generated and send to the process owner. The owner can then determine in more detail whether there is a special controlled cause for the outlier or if there is a good reason to increase the monitoring of the process or even prepare for issues. This cooperation between expert process knowledge and data driven analysis is the basic pillar of the control chart. The charts need no prior knowledge about normal activities in order to set limits, but learn what is considered normal from its observations, and then reflect this increased knowledge automatically in the confidence intervals. What is considered to be normal remains user specific, because confidence intervals depend on the observed data (Denning, 1987).

Control charts should be customized on several points to fit each unique problem and provide the best advice to the process owner. These customizable settings are the selection of a confidence interval, warning limits, and the number of outliers required to label a process as out of control. The general selection of a confidence interval is largely based upon the Central Limit Theorem. This theorem states that for a sufficiently large sample drawn from any distribution (in which variance exists), the resulting distribution of the mean becomes normally distributed. For a normal distribution it is rather easy to calculate the confidence interval based on the selection of the upper and lower limits. A common setting for the control limits is the mean plus or minus three times the standard deviation (Denning, 1987) (Oakland, 2003) (Evans & Lindsay, 2010).

The Vysochanskii-Petunin inequality states that the probability of an outcome greater than k standard deviations from the mean is at most $4/(9k^2)$. For the Normal distribution, with the Shewhart 3-sigma limits, 99.73% of the observations occur within three standard deviations of the mean. So if the process is in control, almost all of the process values will fall between the upper and lower control limits. Any observations outside the limits are an indication that there is an unanticipated source of variation, known as a special-cause variation. Since increased variation means increased issue probability, a control chart signalling the presence of a special-cause requires immediate further detailed investigation (Evans & Lindsay, 2010). It is noteworthy that even if a process is in control, there is approximately a 0.27% probability that there is a falls signal due to a value exceeding the 3 sigma control limits. This provides further support to the statement that the collaboration of expert knowhow and data driven analysis is of utmost importance. Besides the control limits, it is also possible to add warning limits to the charts. While they do not predict that the process is currently out of control, they signal that the process is behaving suspiciously and a high level of alert could be preferred. These warning limits are interesting for situations in which an out of control process is highly undesirable. Typically, warning limits are placed at two standard deviations around the mean.

Control charts typically have the following mandatory elements to serve as guides for control activities (Evans & Lindsay, 2010):

- A central line that designates the estimated mean of the process characteristic under control;
- An upper control limit (UCL) line, which designates an outrebound that would rarely be reached by a plotted point if the process is in control. Classically, this line is designated as the "three sigma" line and stands three times the estimated process standard deviation above the central line;
- A lower control limit (LCL) line, which usually stands below the central line by a distance equal to the upper control line, usually about three times the standard deviation.

Formalizing the above described characteristics of the control chart leads to the following formulas (Oakland, 2003):

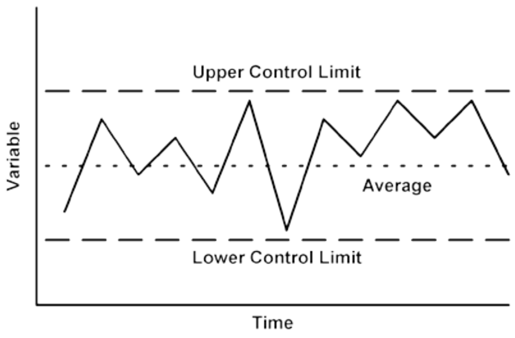


FIGURE 37. SPC CONTROL CHART

$$Upper\ Control\ Limit = \mu + k \cdot \sigma$$
$$Lower\ Control\ Limit = \mu - k \cdot \sigma$$

$$where\ \mu = population\ average,$$
$$k = significance\ factor\ (usually\ equal\ to\ 3),$$
$$and\ \sigma = population\ standard\ deviation$$

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}\ ,\ where\ \mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Shewhart's charts have proven to be quite good at detecting large changes in process variation and mean, but experience issues with important large changes that occur just within the range of the limits. The best suitable kind of data for the control charts is numeric data with Gaussian assumptions (normal distribution) (Oakland, 2003) (Evans & Lindsay, 2010). The final noteworthy fact about the control charts is that the control limits should not be confused with tolerance limits or specifications. Where control limits describe the process production capabilities, tolerance and specification limits describe the customer requirements ("voice of process" versus "voice of the customer") (Evans & Lindsay, 2010).

The Central Limit Theorem

In general probability theory, the central limit theorem (CLT) states that there is a weak convergence of the sum of individual independent stochastic variables. The most well-known application of this general theorem involves the convergence towards the normal distribution. Given certain conditions (well-defined variance and mean), the mean of a sufficiently large number of independent random variables will be approximately normally distributed (Rice, 2007). Under some conditions, it is not even necessary for the distributions to be identical to approximate the normal distribution. By the law of large numbers, the accuracy of the approximation improves with an increase in sample size. The theorem can be formalized via the following formula:

The Lindeberg-Lévy Central Limit Theorem states that; suppose $\{X_1, X_2, \ldots\}$ is a sequence of independent and identically distributed random variables with $E[X_i] = \mu$ and $Var[X_i] = \sigma^2 < \infty$, then as n approaches infinity, the random variables $\sqrt{n}(S_n - \mu)$ converge into a normal distribution $N(0, \sigma^2)$ (Oakland, 2003).

Approximation formula:

$$\lim_{n\to\infty} \sqrt{n}\left((\tfrac{1}{n}\textstyle\sum_{i=1}^{n} X_i) - \mu \right) = N(0, \sigma^2)$$

12.8 Conclusion

The field of data mining is spacious and ever increasing. With the technological progress (ironically, partially made possible by ASML itself) both the amount of data stored and the processing speed increase, which lead to more possibilities and need for data mining solutions. Via several layers of methodology the scope of the theoretical review moved from the top level of data mining in general, towards multiple possible methods that were applied in this thesis (see Figure 38). The applicability of each method strongly depends on the possibility of supervising the modeling phase and all of them have their own pros and cons.

What distinguishes this thesis from the available literature is that it combines several fields of science for solving a single business case. It shows that in case of a lack of data knowledge it is possible to apply a combination of data mining techniques (X-means, Cluster-Based Local Outlier Factor, and Generalized Sequential Pattern) and statistical process control to obtain a result that gives some insight into a previously unknown dataset.

The application of the newer data mining techniques like ANN and CaRTs are not proven to be applicable for condition based maintenance purposes (nor rejected), because the dataset is not homogenous enough for significantly strong clustering. From a theoretically perspective, they do seem interesting if the dataset is large and complex. In case of the ASML data, the information available per time stamp is difficult to compare between (even after back engineering information to a time stamp, see paragraph 8.1).

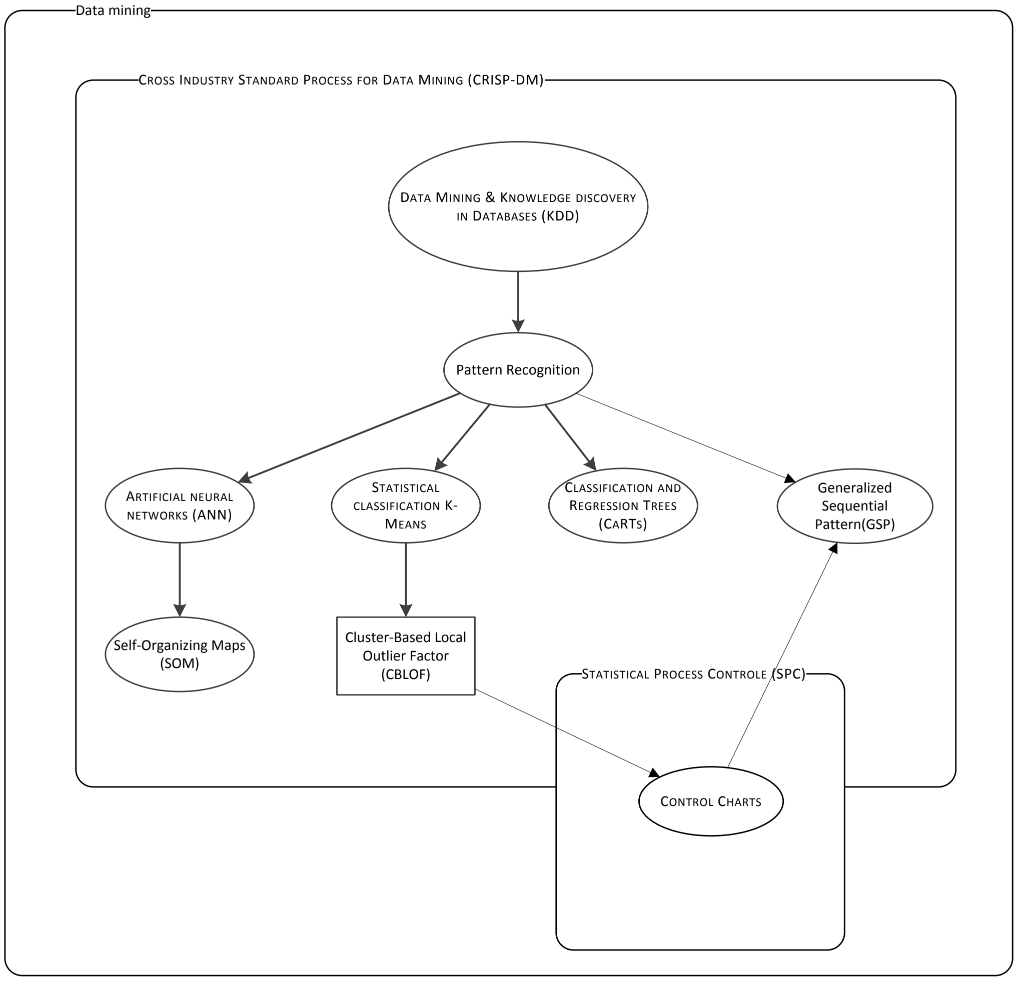

Data mining
Cross Industry Standard Process for Data Mining (CRISP-DM)
Data Mining & Knowledge discovery in Databases (KDD)
Pattern Recognition
Artificial neural networks (ANN)
Statistical classification K-Means
Classification and Regression Trees (CaRTs)
Generalized Sequential Pattern(GSP)
Self-Organizing Maps (SOM)
Cluster-Based Local Outlier Factor (CBLOF)
Statistical Process Controle (SPC)
Control Charts

FIGURE 38. THEORATICAL FRAMEWORK OVERVIEW

1 Appendix: BIBLIOGRAPHY

Alhoniemi, E., Hollmen, J., Simula, O., & Vesanto, J. (1999). Process Monitoring and Modeling using the Self-Organizing Map. *Intergrated Computer-Aided Engineering, 6*, 3-14.

ASML. (2013, March). *About*. Retrieved March 2013, from ASML: http://www.asml.com/asml/show.do?lang=EN&ctx=271

ASML. (2013). *Corporate Presentation.* Veldhoven: ASML.

ASML. (2013, March). *Organization*. Retrieved March 2013, from ASML: http://www.asml.com/asml/show.do?lang=EN&ctx=33248&rid=33384

ASML. (2013, April). *Papers*. Retrieved April 2013, from ASML: http://www.asml.nl/asml/show.do?ctx=13559&rid=34793

ASML. (2013, March). *Performance and Targets*. Retrieved March 2013, from ASML: http://www.asml.com/asml/show.do?ctx=40715&rid=40723

ASML. (2013, May). *PMA archive-Techwiki-ASML*. Retrieved May 2013, from Techwiki-ASML: www.techwiki.asml.com/index.php/PMA_archive

ASML. (2013, March). *Press Releases*. Retrieved March 2013, from ASML: http://www.asml.com/asml/show.do?lang=EN&ctx=5869&rid=44387

ASML. (2013, March). *Profile*. Retrieved March 2013, from ASML: http://www.asml.com/asml/show.do?lang=EN&ctx=272&rid=362

ASML. (2013, May). *Techwiki-ASML*. Retrieved May 2013, from Techwiki-ASML: www.techwiki.asml.com

Baldin, A. (1986). Technical diagnostic- and conditionbased maintenance for better plant availability. *Measurement, 4*, 7-22.

Baruah, P., & Chinnam, R. (2005, March). Hidden Markov Models for diagnostics and prognostics in machining processes. *International Journal of Production Research, 43*, 1275-1293.

Baruah, P., Chinnam, R. B., & Filev, D. (2006). An Autonomous Diagnostics and Prognostics Framework for Condition-Based Maintenance. *2006 International Joint Conference on Neural Networks*, (pp. 3428-3435). Vancouver.

Bebbington, M., Lai, C.-D., & Zitikis, R. C. (2007). Bathtub-type Curves In Reliability and Beyond. *Australian Statistical Publishing Association Inc., 49*, 251-265.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying Density-Based Local Outliers. *ACM Sigmod International Conference on management of data*, (pp. 99-104). Dallas.

Cakir, G. S. (2011). *Development of a Condition Based Maintenance Decision Model by Data Mining.* Eindhoven: TU/e.

Camci, F. (2009). System Maintenance Scheduling With Prognostics Information Using Genetic Algorithm. *IEEE Transactions On Reliability, 58*, 539-552.

Chen, M.-S., Han, J., & Yu, P. S. (1996, December). Data Mining: An Overview from a Database Perspective. *IEEE Transactions On Knowledge And Data Engineering, 8*, 866-883.

Chen, Z., Yang, Y., & Hu, Z. (2012, June). A Technical Framework and Roadmap of Embedded Diagnostics and Prognostics for Complex Mechanical Systems in Prognostics and Health Management Systems. *IEEE Transactions On Reliability, 61*, 314-322.

Chinnam, R. B., & Baruah, P. (2009, December). Autonomous diagnostics and prognostics in machining processes through competitive learning-driven HMM-based clustering. *International Journal of Production Research, 47*, 6739-6758.

Choe, Y. (2008). *Haykin Chapter 9: Self-Organizing Maps.* Retrieved May 2013, from Slides 09, CPSC 636-600

Collins, D. H., & Huzurbazar, A. V. (2012). Prognostic models based on statistical flowgraphs. *Applied Stochastic Models in Business and Industy, 28*, 141-151.

Dash, M., Choi, K., Scheuermann, P., & Liu, H. (2002). Feature Selection for Clustering – A Filter Solution. *International Conference on Data Mining.*, (pp. 115-122).

Denning, D. E. (1987). An Intrusion-Detection Model. *IEEE Transactions On Software Engineering, 13*, 222-232.

Ding, C., & He, X. (2004). K-means Clustering via Principal Component Analysis. *The 21st International Conference on Machine Learning*, (pp. 1-9). Banff.

Evans, J. R., & Lindsay, W. M. (2010). *Managing for Quality and Performance Excellence* (8th ed.). Mason: South-Western, Cengage Learning.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters, 27*, 861-874.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery in Databases. *American Association for Artificial Intelligence, 17*, 37-54.

Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, J., & Goedicke, M. (1992). Viewpoints: A Framework for Integrating Multiple perspectives in system development. *International Journal of Software Engineering and Knowledge Engineering, 2*, 31-57.

Flott, L. W. (2012). Quality Control - Everyday Statistical Process Control. *Metalfinishing*, 37-38.

Gentile, M., & Summers, A. E. (2006, December). Random, Systematic, and Common Cause Failure: How Do You Manage Them? *Process Safety Progress, 25*, 331-338.

Goh, K., Tjahjono, B., Baines, T., & Subramaniam, S. (2006). A Review of Research in Manufacturing Prognostics. *IEEE International Conference on Industrial Informatics., 7*, 417-422.

Gonçalves, L. F., Bosa, J. L., Balen, T. R., Lubaszewski, M. S., Schneider, E. L., & Henriques, R. V. (2011). Fault Detection, Diagnosis and Prediction in Electrical Valves Using Self-Organizing Maps. *J Electron Test, 27*, 551-564.

Hair, J. F., Black, W. C., Babin, B. J., & Anderson, R. E. (2009). *Multivariate Data Analysis: A Global Perspective.* Kennesaw: Pearson.

He, Z., Xu, X., & Deng, S. (2003). Discovering cluster-based local outliers. *Pattern Recongition Letters, 24*, 1641-1650.

Huanga, R., Xia, L., Lib, X., Liuc, C. R., Qiud, H., & Leed, J. (2007). Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. *Mechanical Systems and Signal Processing, 21*, 193-207.

Hüllermeier, E. (2005). Fuzzy methods in machine learning and data mining: Status and prospects. *Fuzzy Sets and Systems, 156*, 387-406.

Jain, A. K. (2010). Pattern Recognition Letters - Data clustering: 50 years beyond K-means. *Pattern Recognition Letters, 31*, 651-666.

Jain, A. K., Duin, R. P., & Mao, J. (2000, January). Statistical Pattern Recognition: A Review. *IEEE Transcations On Pattern Analysis And Machine Intelligence, 22*, 4-36.

Jardine, A. K., Lin, D., & Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing, 20*, 1483-1510.

Jolliffe, I. T. (1995). Sample Sizes and Central Limit Theorem: The Poisson Distrbution as an Illustration. *The American Statistician, 49*, 269.

Kass, R. E., & Wasserman, L. (1995). A Reference Bayesian Test for Nested Hypotheses and its Relationship to the Schwarz Criterion. *Journal of the American Statistical Association, 90*, 928-934.

Katzav, J. (2012). Management Science: Rigour or Relevance. Eindhoven, Noord-Brabant, The Netherlands.

Kohonen, T. (1990). The Self-Organizing Map. *Proceedings of the IEEE, 78*, 1464-1480.

Kohonen, T., & Honkela, T. (2007, December 1). *Kohonen network*, 2. Retrieved from Scholarpedia: http://www.scholarpedia.org/article/Kohonen_network

Kothamasu, R., Huang, S. H., & VerDuin, W. H. (2006). System health monitoring and prognostics - a review of current paradigms and practices. *International Journal Adv. Manufacturing Technology, 28*, 1012-1024.

Last, M., Sinaiski, A., & Subramania, H. S. (2011, March). Condition-based Maintenance with Multi-Target Classification Models. *New Generation Computing, 29*, 245-260.

Lemon, S., Roy, J., Clark, M. A., Friedmann, P. D., & Rakowski, W. (2003). Classification and Regression Tree Analysis in Public Health: Methodological Review and Comparison With Logistic Regression. *Annals of Behavioral Medicine, 26*, 172-181.

Lin, Y., Hsu, A., & Rajami, R. (2002). A Simulation Model For Field Service With Condition-Based Maintenance. *2002 Winter Simulation Conference*, (pp. 1885-1890). East Hartford.

Loh, W.-Y. (2011). Classification and regression trees. *WIREs Data Mining and Knowledge Discovery, 1*, 14-23.

Malhi, A., & Gao, R. X. (2004, December). PCA-Based Feature Selection Scheme for Machine Defect Classification. *IEEE Transactions On instrumentation and Measurement, 54*, 1517-1525.

Maxwell, S. E. (2000). Sample Size and Multiple Regression Analysis. *Psychological Methods, 5*, 434-458.

Melin, P., & Castillo, O. (2005). *Hybrid Intelligent Systems for Pattern Recognition Using Soft Computing: An Evolutionary Approach for Neural Networks and Fuzzy Systems.* Springer.

Mikut, R., & Reischl, M. (2011). Data mining tools. *WIREs Data Mining and Knowledge Discovery*, 1-13.

Montgomery, D. C., & Runger, G. C. (2007). *Applied Statistics and Probability for Engineers* (Fourth Edition ed.). Arizona: John Wiley & Sons.

Murphy, P., & Aha, D. (1994). Iris Flower Dataset. *UCI Repository of Mahine Learning Databases*.

MySQL. (2013, April). *MySQL - The world's most popular open source database*. Retrieved April 2013, from MySQL: http://www.mysql.com/

Neave, H. R. (2002). The Deming Dimension: Management for a Better Future. *Inaugural Professorial Lecture* (pp. 1-17). Nottingham: Nottingham Business School.

Neural Networks Research Center. (1997, September). *World Poverty Map*. Retrieved March 2013, from Example of application of the SOM: http://www.cis.hut.fi/research/som-research/worldmap.html

Nieddu, L., & Patrizi, G. (2000). Formal methods in pattern recognition: A review. *European Journal of Operational Research, 120*, 459-495.

Oakland, J. S. (2003). *Statistical Process Control* (5th ed.). Burlington: Butterworth-Heinemann.

Pardo, M., & Sberveglieri, G. (2002, June). Learning From Data: A Tutorial With Emphasis on Modern Pattern Recognition Methods. *IEEE Sensors Journal, 2*, 203-217.

Pelleg, D., & Moore, A. (2000). X-means: Extending K-means with Efficient Estimation of the Number of Clusters. *Proceedings of the Seventeenth Internation Conference on Machine Learning*, (pp. 727-734).

Peng, Y., Dong, M., & Zuo, M. J. (2010). Current status of machine prognostics in condition-based maintenance: a review. *International Journal Adv Manuf Technol, 50*, 297-313.

Rapid-I. (2013, April). *RapidMiner*. Retrieved April 2013, from Rapid-I Report the Future: http://rapid-i.com/content/view/181/190/

Rice, J. A. (2007). *Mathematical Statistics and Data Analysis* (3 ed.). Belmont: Thomsom, Brooks/Cole.

Shafait, F., Reif, M., Kofler, C., & Breuel, T. M. (2010). Pattern Recognition Engineering. *RapidMiner Community Meeting and Conference* (pp. 50-56). Dortmund: RCOMM'10.

Smith, Z. R., & Wells, C. S. (2006). Central Limit Theorem and Sample Size. *The Annual Meeting of the Northeastern Educational Research*, (pp. 1-22). Kerhonkson.

Speybroeck, N. (2012). Classification and regression trees. *Int Journal Public Health, 57*, 243-246.

Spotfire, T. (2013, May). *Spotfire*. Retrieved May 2013, from Spotfire About: http://spotfire.tibco.com/en/discover-spotfire/who-uses-spotfire.aspx

Srikant, R., & Agrawal, R. (1996). *Mining Sequential Patterns: Generalizations and Performance improvements.* San Jose: IBM Research Devision.

Stehman, S. V. (1997). Selecting and Interpreting Measures of Thematic Classification Accuracy. *Remote Sensing of Environment, 62*, 77-89.

Stringer, D. B., Sheth, P. N., & Allaire, P. E. (2012). Physics-based modeling strategies for diagnostic and prognostic application in aerospace systems. *J Intell Manuf, 23*, 155-162.

Thor, J., Lundberg, J., Ask, J., Olsson, J., Carli, C., Pukk Härenstam, K., et al. (2007). Application of statistical process control in healthcare improvement: systematic review. *Quality Safety Health Care, 16*, 387–399.

Timmermans, B. (2012). *Development and application of a decision model for synchronizing condition-based maintenance at ASML.* Eindhoven: TU/e.

Tobon-Mejia, D. A., Medjaher, K., Zerhouni, N., & Tripot, G. (2012, June). A Data-Driven Failure Prognostics Method Based on Mixture of Gaussians Hidden Markov Models. *IEEE Transactions On Reliability, 61*, 491-503.

Wirth, R., & Hipp, J. (2000). CRISP-DM: Towards a Standard Process Model for Data Mining. *Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining.*, (pp. 29-39).

Wu, J. (2012). *Advances in K-means Clustering: A Data Mining Thinking.* Springer.

Yu, J. (2011). A hybrid feature selection scheme and self-organizing map model for machine selection scheme and self-organizing map model for machine. *Applied Soft Computing, 11*, 4041-4054.

Zhang, S., Zhang, C., & Yang, Q. (2003). Data Preparation For Data Mining. *Applied Artificial Intelligence, 17*, 375-381.

2 Appendix: TABLE OF FIGURES

3 Appendix: ACRONYMS

ADC	Advanced Data Collection
ANN	Artificial Neural Networks
ARE	Automated Recovery Engine
BIC	Bayesian Information Criterion
BRES	Basic Remote Equipment Support
CaRTs	Classification and Regression Trees
CBM	Condition Based Maintenance
CCB	Change Control Board
CLBOF	Cluster-Based Local Outlier Factor
CLT	Central Limit Theorem
CRISP-DM	Cross Industry Standard Process for Data Mining
CTL	Create Total issue List
FC	Factory Constant
GSP	Generalized Sequential Pattern
IDP	Incoming Data Processor
JMCL	Join Machine Constant List
KDD	Knowledge Discovery in Databases
MC	Machine Constant
OCS	Outlier Calibration Selection per time stamp
OJFP	Outlier Jump Frequency compared to Population
OJI	Outlier Jump value compared to Individual
OJP	Outlier Jump value compared to Population
OMCV	Outlier Machine Constant Values per time stamp
OTP	Outlier Trend Compared to Population
OVP	Outlier Value compared to Population
OVI	Outlier Value compared to Individual
PAM	Pro-Active Maintenance
PCA	Principle Component Analysis
PMA	Performance Monitoring & Analysis
RUL	Remaining Useful Life
SOM	Self Organizing Maps
SPC	Statistical Process Control

4 Appendix: Detailed Final Phase Planning

This is a short overview of the final phase of this master thesis project. A time planning and list of activities is given:

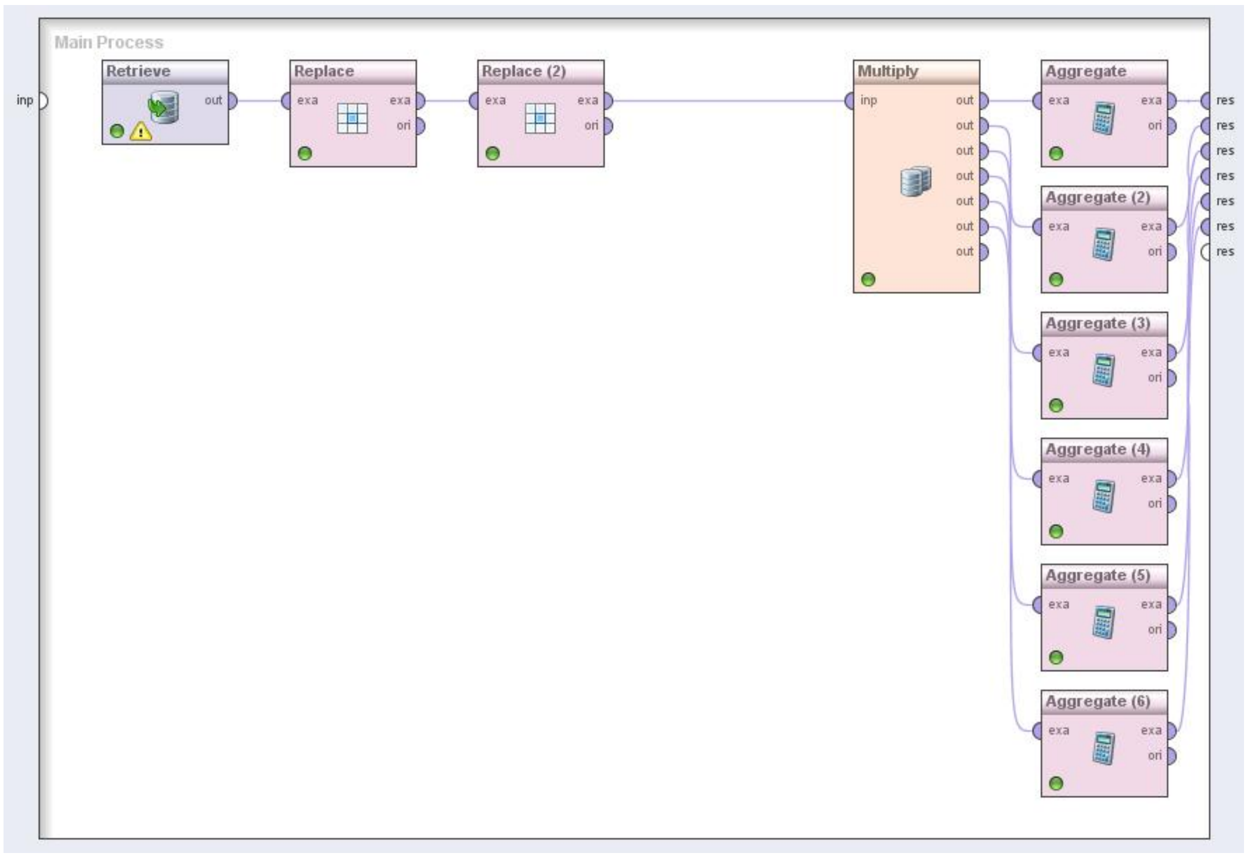
TABLE 20. DETAILED PLANNING

Activity	*Week Number*	*Deadline*
Start of final phase	Week 22	Tuesday 28th of May
Finish modeling & validation	Week 26	Friday 28th of June
Finish evaluation phase	Week 27	Friday 5th of July
Finish (required) on-site work at ASML	Week 28	Friday 12th of July
Deliver first Report	Week 30	Tuesday 23th of July
Deliver final Report	Week 32	Tuesday 6th of August
Final defence (presentation)	**Week 33**	**Tuesday 13th of August**

5 APPENDIX: MYSQL INPUT CODE

```
DROP TABLE IF EXISTS `pam_ASML`.`Machine_constants` ;

CREATE  TABLE IF NOT EXISTS `pam_ASML`.`Machine_constants` (
  `MACHINE` INT NULL ,
  `STE_CODE` VARCHAR(45) NULL ,
  `MACHINE_CONSTANT` VARCHAR(150) NULL ,
  `VALID_FROM_DATE` DATE NULL ,
  `MCS_TYPE` VARCHAR(45) NULL ,
  `MCS_TYPE_DESCRIPTION` VARCHAR(45) NULL ,
  `SEQUENCE` INT NULL ,
  `CREATION_DATE` DATETIME NULL ,
  `DATA_FORMAT` CHAR NULL ,
  `TEXT_VALUE` VARCHAR(150) NULL ,
  `NUMERIC_VALUE` DOUBLE NULL ,
  `CONSTANT_TYPE` VARCHAR(45) NULL ,
  `CONSTANT_TYPE_DESCRIPTION` VARCHAR(45) NULL ,
  `CATEGORY_CODE` VARCHAR(45) NULL ,
  `CATEGORY_DESCRIPTION` VARCHAR(150) NULL ,
  `KEEP_HISTORY` CHAR NULL,
  INDEX `machine` (`MACHINE` ASC) ,
  INDEX `constant` (`MACHINE_CONSTANT` ASC)) ENGINE = InnoDB;

Load data infile 'machine_constant-since1-feb-2013.csv'
into table pam_ASML.machine_constants
fields terminated by ','
enclosed by '"'
IGNORE 1 LINES

(`MACHINE`,
`STE_CODE`,
`MACHINE_CONSTANT`,
@tempvar1,
`MCS_TYPE`,
`MCS_TYPE_DESCRIPTION`,
`SEQUENCE`,
@tempvar2,
`DATA_FORMAT`,
@tempvar3,
@tempvar4,
`CONSTANT_TYPE`,
`CONSTANT_TYPE_DESCRIPTION`,
`CATEGORY_CODE`,
`CATEGORY_DESCRIPTION`,
`KEEP_HISTORY`)

SET VALID_FROM_DATE = STR_TO_DATE(@tempvar1,'%d-%m-%Y'),
      CREATION_DATE = STR_TO_DATE(@tempvar2,'%d-%m-%Y %H:%i:%s'),
      TEXT_VALUE = nullif(@tempvar3, ''),
      NUMERIC_VALUE = nullif(@tempvar4,'');
```

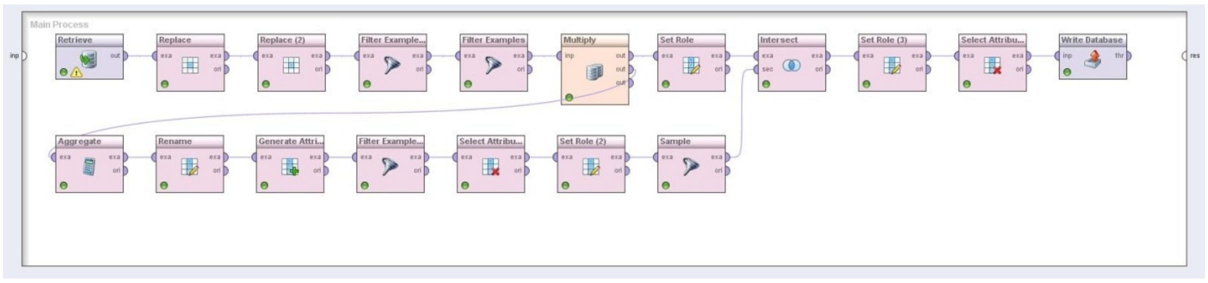
6 APPENDIX: BASIC ANALYSIS



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="5.2.008">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="5.2.008" expanded="true" name="Process">
    <process expanded="true" height="586" width="835">
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Retrieve" width="90" x="45" y="30">
        <parameter key="repository_entry" value="//DB/asml_pam/Example Sets/machine_constants"/>
      </operator>
      <operator activated="true" class="replace" compatibility="5.2.008" expanded="true" height="76" name="Replace" width="90" x="179" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="STE_CODE"/>
        <parameter key="replace_what" value="NXT1950I"/>
        <parameter key="replace_by" value="NXT1950i"/>
      </operator>
      <operator activated="true" class="replace" compatibility="5.2.008" expanded="true" height="76" name="Replace (2)" width="90" x="313" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="STE_CODE"/>
        <parameter key="replace_what" value="NXT1960BI"/>
        <parameter key="replace_by" value="NXT1960Bi"/>
      </operator>
      <operator activated="true" class="multiply" compatibility="5.2.008" expanded="true" height="166" name="Multiply" width="90" x="581" y="30"/>
      <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (5)" width="90" x="715" y="390">
        <list key="aggregation_attributes">
          <parameter key="KEEP_HISTORY" value="count"/>
        </list>
        <parameter key="group_by_attributes" value="|KEEP_HISTORY"/>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (4)" width="90" x="715" y="300">
      <list key="aggregation_attributes">
        <parameter key="CONSTANT_TYPE" value="count"/>
      </list>
      <parameter key="group_by_attributes" value="|CONSTANT_TYPE"/>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (3)" width="90" x="715" y="210">
      <list key="aggregation_attributes">
        <parameter key="DATA_FORMAT" value="count"/>
      </list>
      <parameter key="group_by_attributes" value="|DATA_FORMAT"/>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (2)" width="90" x="715" y="120">
      <list key="aggregation_attributes">
        <parameter key="CATEGORY_CODE" value="count"/>
        <parameter key="CATEGORY_DESCRIPTION" value="count"/>
        <parameter key="CONSTANT_TYPE" value="count"/>
        <parameter key="CONSTANT_TYPE_DESCRIPTION" value="count"/>
        <parameter key="CREATION_DATE" value="count"/>
        <parameter key="DATA_FORMAT" value="count"/>
        <parameter key="KEEP_HISTORY" value="count"/>
        <parameter key="MACHINE" value="count"/>
        <parameter key="MACHINE_CONSTANT" value="count"/>
        <parameter key="MCS_TYPE" value="count"/>
        <parameter key="MCS_TYPE_DESCRIPTION" value="count"/>
        <parameter key="NUMERIC_VALUE" value="count"/>
        <parameter key="SEQUENCE" value="count"/>
        <parameter key="STE_CODE" value="count"/>
        <parameter key="TEXT_VALUE" value="count"/>
        <parameter key="VALID_FROM_DATE" value="count"/>
      </list>
      <parameter key="only_distinct" value="true"/>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (6)" width="90" x="715" y="480">
      <list key="aggregation_attributes">
        <parameter key="CATEGORY_CODE" value="count"/>
        <parameter key="CATEGORY_DESCRIPTION" value="count"/>
        <parameter key="CONSTANT_TYPE" value="count"/>
        <parameter key="CONSTANT_TYPE_DESCRIPTION" value="count"/>
        <parameter key="CREATION_DATE" value="count"/>
        <parameter key="DATA_FORMAT" value="count"/>
        <parameter key="KEEP_HISTORY" value="count"/>
        <parameter key="MACHINE" value="count"/>
        <parameter key="MACHINE_CONSTANT" value="count"/>
        <parameter key="MCS_TYPE" value="count"/>
        <parameter key="MCS_TYPE_DESCRIPTION" value="count"/>
        <parameter key="NUMERIC_VALUE" value="count"/>
        <parameter key="SEQUENCE" value="count"/>
        <parameter key="STE_CODE" value="count"/>
        <parameter key="VALID_FROM_DATE" value="count"/>
        <parameter key="TEXT_VALUE" value="count"/>
      </list>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate" width="90" x="715" y="30">
      <list key="aggregation_attributes">
        <parameter key="MACHINE" value="average"/>
        <parameter key="MACHINE" value="maximum"/>
        <parameter key="MACHINE" value="minimum"/>
        <parameter key="NUMERIC_VALUE" value="average"/>
        <parameter key="NUMERIC_VALUE" value="maximum"/>
        <parameter key="NUMERIC_VALUE" value="minimum"/>
        <parameter key="CREATION_DATE" value="average"/>
        <parameter key="CREATION_DATE" value="maximum"/>
        <parameter key="CREATION_DATE" value="minimum"/>
        <parameter key="VALID_FROM_DATE" value="average"/>
      <parameter key="VALID_FROM_DATE" value="maximum"/>
      <parameter key="VALID_FROM_DATE" value="minimum"/>
     </list>
    </operator>
    <connect from_op="Retrieve" from_port="output" to_op="Replace" to_port="example set input"/>
    <connect from_op="Replace" from_port="example set output" to_op="Replace (2)" to_port="example set input"/>
    <connect from_op="Replace (2)" from_port="example set output" to_op="Multiply" to_port="input"/>
    <connect from_op="Multiply" from_port="output 1" to_op="Aggregate" to_port="example set input"/>
    <connect from_op="Multiply" from_port="output 2" to_op="Aggregate (2)" to_port="example set input"/>
    <connect from_op="Multiply" from_port="output 3" to_op="Aggregate (3)" to_port="example set input"/>
    <connect from_op="Multiply" from_port="output 4" to_op="Aggregate (4)" to_port="example set input"/>
    <connect from_op="Multiply" from_port="output 5" to_op="Aggregate (5)" to_port="example set input"/>
    <connect from_op="Multiply" from_port="output 6" to_op="Aggregate (6)" to_port="example set input"/>
    <connect from_op="Aggregate (5)" from_port="example set output" to_port="result 5"/>
    <connect from_op="Aggregate (4)" from_port="example set output" to_port="result 4"/>
    <connect from_op="Aggregate (3)" from_port="example set output" to_port="result 3"/>
    <connect from_op="Aggregate (2)" from_port="example set output" to_port="result 2"/>
    <connect from_op="Aggregate (6)" from_port="example set output" to_port="result 6"/>
    <connect from_op="Aggregate" from_port="example set output" to_port="result 1"/>
    <portSpacing port="source_input 1" spacing="0"/>
    <portSpacing port="sink_result 1" spacing="0"/>
    <portSpacing port="sink_result 2" spacing="0"/>
    <portSpacing port="sink_result 3" spacing="0"/>
    <portSpacing port="sink_result 4" spacing="0"/>
    <portSpacing port="sink_result 5" spacing="0"/>
    <portSpacing port="sink_result 6" spacing="0"/>
    <portSpacing port="sink_result 7" spacing="0"/>
   </process>
  </operator>
</process>
```

7 Appendix: Prepare Set

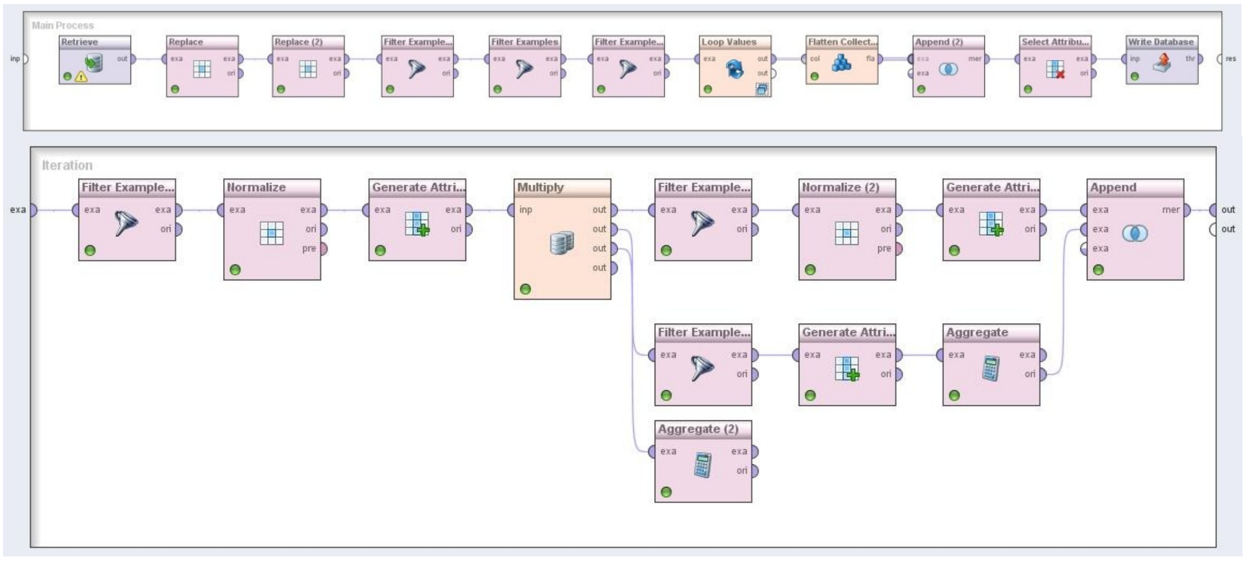
7.1 First Run



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="5.2.008">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="5.2.008" expanded="true" name="Process">
    <process expanded="true" height="531" width="1555">
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Retrieve" width="90" x="45"
y="30">
        <parameter key="repository_entry" value="//DB/asml_pam/Example Sets/machine_constants"/>
      </operator>
      <operator activated="true" class="replace" compatibility="5.2.008" expanded="true" height="76" name="Replace" width="90" x="180"
y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="STE_CODE"/>
        <parameter key="replace_what" value="NXT1950I"/>
        <parameter key="replace_by" value="NXT1950i"/>
      </operator>
      <operator activated="true" class="replace" compatibility="5.2.008" expanded="true" height="76" name="Replace (2)" width="90" x="315"
y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="STE_CODE"/>
        <parameter key="replace_what" value="NXT1960BI"/>
        <parameter key="replace_by" value="NXT1960Bi"/>
      </operator>
      <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (3)"
width="90" x="450" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="STE_CODE=NXT1950i || STE_CODE=XT1950Hi || STE_CODE=XT860K || STE_CODE=XT1000H
|| STE_CODE=NXT1960Bi || STE_CODE=XT1900Gi || STE_CODE=XT400K || STE_CODE=XT1450H || STE_CODE=XT860H || STE_CODE=XT1400F
|| STE_CODE=XT1700i || STE_CODE=XT1450G || STE_CODE=XT1900i || STE_CODE=XT400F || STE_CODE=XT850F || STE_CODE=XT1250D ||
STE_CODE=XT1700Fi || STE_CODE=XT1400E || STE_CODE=XT860F || STE_CODE=XT875F || STE_CODE=XT870G || STE_CODE=XT860G ||
STE_CODE=XT400E STE_CODE=XT870F || STE_CODE=XT875G || STE_CODE=XT1250B || STE_CODE=XT400G || STE_CODE=XT760F ||
STE_CODE=XT850G || STE_CODE=XT450F "/>
      </operator>
      <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples"
width="90" x="585" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="SEQUENCE=1 &amp;&amp; DATA_FORMAT=N &amp;&amp; NUMERIC_VALUE!=?"/>
      </operator>
      <operator activated="true" class="multiply" compatibility="5.2.008" expanded="true" height="94" name="Multiply" width="90" x="720"
y="30"/>
      <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role" width="90" x="849"
y="30">
        <parameter key="name" value="MACHINE_CONSTANT"/>
        <parameter key="target_role" value="id"/>
        <list key="set_additional_roles"/>
      </operator>
      <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate" width="90" x="45"
y="165">
        <list key="aggregation_attributes">
        <parameter key="NUMERIC_VALUE" value="variance"/>
        <parameter key="NUMERIC_VALUE" value="count"/>
        <parameter key="NUMERIC_VALUE" value="average"/>
      </list>
      <parameter key="group_by_attributes" value="MACHINE_CONSTANT"/>
    </operator>
    <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename" width="90" x="179" y="165">
      <parameter key="old_name" value="variance(NUMERIC_VALUE)"/>
      <parameter key="new_name" value="VAR_NV"/>
      <list key="rename_additional_attributes">
        <parameter key="count(NUMERIC_VALUE)" value="COUNT_NV"/>
        <parameter key="average(NUMERIC_VALUE)" value="AVERAGE_NV"/>
      </list>
    </operator>
    <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes" width="90" x="313" y="165">
      <list key="function_descriptions">
        <parameter key="COV" value="VAR_NV/((AVERAGE_NV)^2)"/>
      </list>
    </operator>
    <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (2)" width="90" x="447" y="165">
      <parameter key="condition_class" value="attribute_value_filter"/>
      <parameter key="parameter_string" value="COUNT_NV&gt;100 &amp;&amp; VAR_NV&gt;0"/>
    </operator>
    <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes" width="90" x="581" y="165">
      <parameter key="attribute_filter_type" value="single"/>
      <parameter key="attribute" value="MACHINE_CONSTANT"/>
    </operator>
    <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (2)" width="90" x="715" y="165">
      <parameter key="name" value="MACHINE_CONSTANT"/>
      <parameter key="target_role" value="id"/>
      <list key="set_additional_roles"/>
    </operator>
    <operator activated="true" class="sample" compatibility="5.2.008" expanded="true" height="76" name="Sample (2)" width="90" x="849" y="165">
      <parameter key="sample_size" value="40"/>
      <list key="sample_size_per_class"/>
      <list key="sample_ratio_per_class"/>
      <list key="sample_probability_per_class"/>
      <parameter key="use_local_random_seed" value="true"/>
      <parameter key="local_random_seed" value="1791"/>
    </operator>
    <operator activated="true" class="intersect" compatibility="5.2.008" expanded="true" height="76" name="Intersect" width="90" x="983" y="30"/>
    <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (3)" width="90" x="1117" y="30">
      <parameter key="name" value="MACHINE_CONSTANT"/>
      <list key="set_additional_roles"/>
    </operator>
    <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (2)" width="90" x="1251" y="30">
      <parameter key="attribute_filter_type" value="subset"/>
      <parameter key="attributes" value="CATEGORY_CODE|CATEGORY_DESCRIPTION|CONSTANT_TYPE|CREATION_DATE|DATA_FORMAT|MACHINE|MACHINE_CONSTANT|MCS_TYPE_DESCRIPTION|NUMERIC_VALUE|SEQUENCE|STE_CODE|TEXT_VALUE|VALID_FROM_DATE|CONSTANT_TYPE_DESCRIPTION|MCS_TYPE|KEEP_HISTORY"/>
    </operator>
    <operator activated="true" class="write_database" compatibility="5.2.008" expanded="true" height="60" name="Write Database" width="90" x="1385" y="30">
      <parameter key="connection" value="asml_pam"/>
      <parameter key="table_name" value="machine_constants2"/>
      <parameter key="overwrite_mode" value="overwrite"/>
    </operator>
    <connect from_op="Retrieve" from_port="output" to_op="Replace" to_port="example set input"/>
    <connect from_op="Replace" from_port="example set output" to_op="Replace (2)" to_port="example set input"/>
    <connect from_op="Replace (2)" from_port="example set output" to_op="Filter Examples (3)" to_port="example set input"/>
    <connect from_op="Filter Examples (3)" from_port="example set output" to_op="Filter Examples" to_port="example set input"/>
```

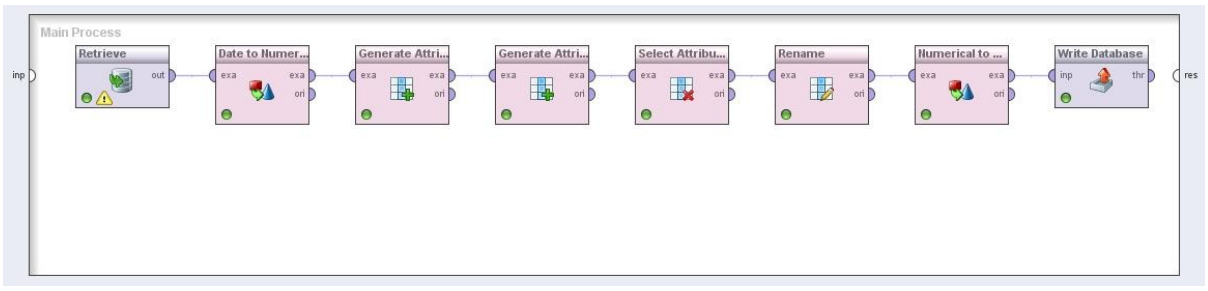
```
      <connect from_op="Filter Examples" from_port="example set output" to_op="Multiply" to_port="input"/>
      <connect from_op="Multiply" from_port="output 1" to_op="Set Role" to_port="example set input"/>
      <connect from_op="Multiply" from_port="output 2" to_op="Aggregate" to_port="example set input"/>
      <connect from_op="Set Role" from_port="example set output" to_op="Intersect" to_port="example set input"/>
      <connect from_op="Aggregate" from_port="example set output" to_op="Rename" to_port="example set input"/>
      <connect from_op="Rename" from_port="example set output" to_op="Generate Attributes" to_port="example set input"/>
      <connect from_op="Generate Attributes" from_port="example set output" to_op="Filter Examples (2)" to_port="example set input"/>
      <connect from_op="Filter Examples (2)" from_port="example set output" to_op="Select Attributes" to_port="example set input"/>
      <connect from_op="Select Attributes" from_port="example set output" to_op="Set Role (2)" to_port="example set input"/>
      <connect from_op="Set Role (2)" from_port="example set output" to_op="Sample (2)" to_port="example set input"/>
      <connect from_op="Sample (2)" from_port="example set output" to_op="Intersect" to_port="second"/>
      <connect from_op="Intersect" from_port="example set output" to_op="Set Role (3)" to_port="example set input"/>
      <connect from_op="Set Role (3)" from_port="example set output" to_op="Select Attributes (2)" to_port="example set input"/>
      <connect from_op="Select Attributes (2)" from_port="example set output" to_op="Write Database" to_port="input"/>
      <portSpacing port="source_input 1" spacing="0"/>
      <portSpacing port="sink_result 1" spacing="0"/>
    </process>
  </operator>
</process>
```

7.2 Fourth run



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="5.2.008">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="5.2.008" expanded="true" name="Process">
    <process expanded="true" height="645" width="1505">
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Retrieve" width="90" x="45"
y="30">
        <parameter key="repository_entry" value="//DB/asml_pam/Example Sets/machine_constants"/>
      </operator>
      <operator activated="true" class="replace" compatibility="5.2.008" expanded="true" height="76" name="Replace" width="90" x="180"
y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="STE_CODE"/>
        <parameter key="replace_what" value="NXT1950I"/>
        <parameter key="replace_by" value="NXT1950i"/>
      </operator>
      <operator activated="true" class="replace" compatibility="5.2.008" expanded="true" height="76" name="Replace (2)" width="90" x="313"
y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="STE_CODE"/>
        <parameter key="replace_what" value="NXT1960BI"/>
        <parameter key="replace_by" value="NXT1960Bi"/>
    </operator>
    <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (3)" width="90" x="450" y="30">
      <parameter key="condition_class" value="attribute_value_filter"/>
      <parameter key="parameter_string" value="STE_CODE=NXT1950i || STE_CODE=NXT1960Bi "/>
    </operator>
    <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples" width="90" x="585" y="30">
      <parameter key="condition_class" value="attribute_value_filter"/>
      <parameter key="parameter_string" value="SEQUENCE=1 &amp;&amp; DATA_FORMAT=N &amp;&amp; NUMERIC_VALUE!=?"/>
    </operator>
    <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (4)" width="90" x="715" y="30">
      <parameter key="condition_class" value="attribute_value_filter"/>
      <parameter key="parameter_string" value="MACHINE_CONSTANT = MCKV_CHUCK_0_LS_DRIFT_CORRECTION_LAST_UPDATE_TAG || MACHINE_CONSTANT = MCUL_BS_PE_AXIS_1_MU_AI_LINEAR_ACTUATOR_OFFSET_TAG "/>
    </operator>
    <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values" width="90" x="849" y="30">
      <parameter key="attribute" value="MACHINE_CONSTANT"/>
      <parameter key="iteration_macro" value="loopA"/>
      <parameter key="parallelize_iteration" value="true"/>
      <process expanded="true" height="645" width="1103">
        <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (2)" width="90" x="45" y="30">
          <parameter key="condition_class" value="attribute_value_filter"/>
          <parameter key="parameter_string" value="MACHINE_CONSTANT=%{loopA}"/>
        </operator>
        <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize" width="90" x="180" y="30">
          <parameter key="attribute_filter_type" value="single"/>
          <parameter key="attribute" value="NUMERIC_VALUE"/>
        </operator>
        <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (3)" width="90" x="315" y="30">
          <list key="function_descriptions">
            <parameter key="NUMERIC_VALUEX" value="abs(NUMERIC_VALUE)"/>
          </list>
        </operator>
        <operator activated="true" class="multiply" compatibility="5.2.008" expanded="true" height="112" name="Multiply" width="90" x="450" y="30"/>
        <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (6)" width="90" x="581" y="165">
          <parameter key="condition_class" value="attribute_value_filter"/>
          <parameter key="parameter_string" value="NUMERIC_VALUEX&gt;=2"/>
        </operator>
        <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (2)" width="90" x="715" y="165">
          <list key="function_descriptions">
            <parameter key="AREA" value="&quot;j&quot;"/>
          </list>
        </operator>
        <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate" width="90" x="849" y="165">
          <list key="aggregation_attributes">
            <parameter key="MACHINE" value="count"/>
          </list>
          <parameter key="only_distinct" value="true"/>
        </operator>
        <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (5)" width="90" x="581" y="30">
          <parameter key="condition_class" value="attribute_value_filter"/>
          <parameter key="parameter_string" value="NUMERIC_VALUEX&lt;2"/>
        </operator>
        <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (2)" width="90" x="715" y="30">
          <parameter key="attribute_filter_type" value="single"/>
          <parameter key="attribute" value="NUMERIC_VALUE"/>
        </operator>
        <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes" width="90" x="849" y="30">
      <list key="function_descriptions">
       <parameter key="AREA" value="if(abs(NUMERIC_VALUE)&lt;0.25,&quot;a&quot;,if(abs(NUMERIC_VALUE)&lt;0.5,&quot;b&quot;,if(abs(NUMERIC_VALUE)&lt;0.75,&quot;c&quot;,if(abs(NUMERIC_VALUE)&lt;1,&quot;d&quot;,if(abs(NUMERIC_VALUE)&lt;1.25,&quot;e&quot;,if(abs(NUMERIC_VALUE)&lt;1.5,&quot;f&quot;,if(abs(NUMERIC_VALUE)&lt;1.75,&quot;g&quot;,if(abs(NUMERIC_VALUE)&lt;2,&quot;h&quot;,&quot;i&quot;))))))))"/>
      </list>
     </operator>
     <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (2)" width="90" x="581" y="255">
      <list key="aggregation_attributes">
       <parameter key="MACHINE" value="count"/>
      </list>
      <parameter key="only_distinct" value="true"/>
     </operator>
     <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="94" name="Append" width="90" x="983" y="30"/>
     <connect from_port="example set" to_op="Filter Examples (2)" to_port="example set input"/>
     <connect from_op="Filter Examples (2)" from_port="example set output" to_op="Normalize" to_port="example set input"/>
     <connect from_op="Normalize" from_port="example set output" to_op="Generate Attributes (3)" to_port="example set input"/>
     <connect from_op="Generate Attributes (3)" from_port="example set output" to_op="Multiply" to_port="input"/>
     <connect from_op="Multiply" from_port="output 1" to_op="Filter Examples (5)" to_port="example set input"/>
     <connect from_op="Multiply" from_port="output 2" to_op="Filter Examples (6)" to_port="example set input"/>
     <connect from_op="Multiply" from_port="output 3" to_op="Aggregate (2)" to_port="example set input"/>
     <connect from_op="Filter Examples (6)" from_port="example set output" to_op="Generate Attributes (2)" to_port="example set input"/>
     <connect from_op="Generate Attributes (2)" from_port="example set output" to_op="Aggregate" to_port="example set input"/>
     <connect from_op="Aggregate" from_port="original" to_op="Append" to_port="example set 2"/>
     <connect from_op="Filter Examples (5)" from_port="example set output" to_op="Normalize (2)" to_port="example set input"/>
     <connect from_op="Normalize (2)" from_port="example set output" to_op="Generate Attributes" to_port="example set input"/>
     <connect from_op="Generate Attributes" from_port="example set output" to_op="Append" to_port="example set 1"/>
     <connect from_op="Append" from_port="merged set" to_port="out 1"/>
     <portSpacing port="source_example set" spacing="0"/>
     <portSpacing port="sink_out 1" spacing="0"/>
     <portSpacing port="sink_out 2" spacing="0"/>
    </process>
   </operator>
   <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection (2)" width="90" x="983" y="30"/>
   <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append (2)" width="90" x="1117" y="30"/>
   <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (2)" width="90" x="1251" y="30">
    <parameter key="attribute_filter_type" value="subset"/>
    <parameter key="attributes" value="CATEGORY_CODE|CATEGORY_DESCRIPTION|CONSTANT_TYPE|CONSTANT_TYPE_DESCRIPTION|CREATION_DATE|DATA_FORMAT|KEEP_HISTORY|MACHINE|MACHINE_CONSTANT|MCS_TYPE|MCS_TYPE_DESCRIPTION|STE_CODE|VALID_FROM_DATE|AREA|NUMERIC_VALUEX"/>
   </operator>
   <operator activated="true" class="write_database" compatibility="5.2.008" expanded="true" height="60" name="Write Database" width="90" x="1385" y="30">
    <parameter key="connection" value="asml_pam"/>
    <parameter key="table_name" value="machine_constants2"/>
    <parameter key="overwrite_mode" value="overwrite"/>
   </operator>
   <connect from_op="Retrieve" from_port="output" to_op="Replace" to_port="example set input"/>
   <connect from_op="Replace" from_port="example set output" to_op="Replace (2)" to_port="example set input"/>
   <connect from_op="Replace (2)" from_port="example set output" to_op="Filter Examples (3)" to_port="example set input"/>
   <connect from_op="Filter Examples (3)" from_port="example set output" to_op="Filter Examples" to_port="example set input"/>
   <connect from_op="Filter Examples" from_port="example set output" to_op="Filter Examples (4)" to_port="example set input"/>
   <connect from_op="Filter Examples (4)" from_port="example set output" to_op="Loop Values" to_port="example set"/>
   <connect from_op="Loop Values" from_port="out 1" to_op="Flatten Collection (2)" to_port="collection"/>
   <connect from_op="Flatten Collection (2)" from_port="flat" to_op="Append (2)" to_port="example set 1"/>
   <connect from_op="Append (2)" from_port="merged set" to_op="Select Attributes (2)" to_port="example set input"/>
   <connect from_op="Select Attributes (2)" from_port="example set output" to_op="Write Database" to_port="input"/>
   <portSpacing port="source_input 1" spacing="0"/>
   <portSpacing port="sink_result 1" spacing="0"/>
  </process>
 </operator>
</process>
```

8 APPENDIX: EVENT DEFINITION

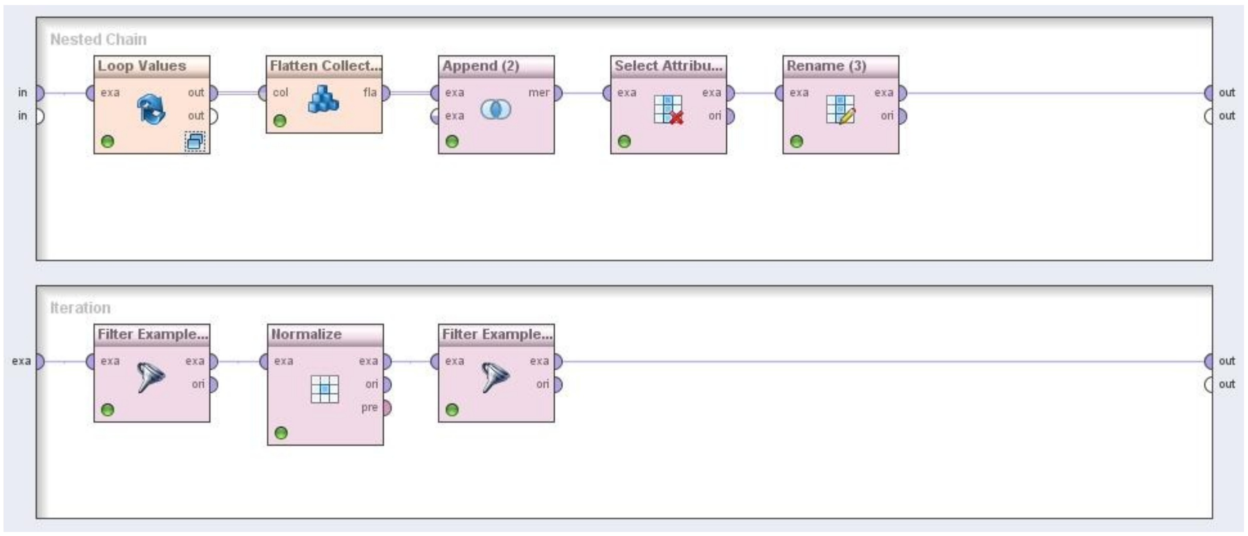


```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="5.2.008">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="5.2.008" expanded="true" name="Process">
    <parameter key="parallelize_main_process" value="true"/>
    <process expanded="true" height="637" width="1110">
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Retrieve" width="90" x="45" y="30">
        <parameter key="repository_entry" value="//DB/asml_pam/Example Sets/machine_constants2"/>
      </operator>
      <operator activated="true" class="date_to_numerical" compatibility="5.2.008" expanded="true" height="76" name="Date to Numerical" width="90" x="180" y="30">
        <parameter key="attribute_name" value="CREATION_DATE"/>
        <parameter key="millisecond_relative_to" value="epoch"/>
        <parameter key="second_relative_to" value="epoch"/>
        <parameter key="minute_relative_to" value="epoch"/>
      </operator>
      <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (2)" width="90" x="315" y="30">
        <list key="function_descriptions">
          <parameter key="ACREATION_DATE" value="round(CREATION_DATE/60000)"/>
        </list>
      </operator>
      <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (3)" width="90" x="450" y="30">
        <list key="function_descriptions">
          <parameter key="BCREATION_DATE" value="ACREATION_DATE*60000"/>
        </list>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes" width="90" x="585" y="30">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attributes" value="BCREATION_DATE|CATEGORY_CODE|CATEGORY_DESCRIPTION|CONSTANT_TYPE|DATA_FORMAT|MACHINE|MACHINE_CONSTANT|MCS_TYPE_DESCRIPTION|NUMERIC_VALUE|SEQUENCE|STE_CODE|VALID_FROM_DATE||KEEP_HISTORY|MCS_TYPE|TEXT_VALUE|CONSTANT_TYPE_DESCRIPTION"/>
      </operator>
      <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename" width="90" x="720" y="30">
        <parameter key="old_name" value="BCREATION_DATE"/>
        <parameter key="new_name" value="CREATION_DATE"/>
        <list key="rename_additional_attributes"/>
      </operator>
      <operator activated="true" class="numerical_to_date" compatibility="5.2.008" expanded="true" height="76" name="Numerical to Date" width="90" x="855" y="30">
        <parameter key="attribute_name" value="CREATION_DATE"/>
      </operator>
      <operator activated="true" class="write_database" compatibility="5.2.008" expanded="true" height="60" name="Write Database" width="90" x="990" y="30">
        <parameter key="connection" value="asml_pam"/>
        <parameter key="table_name" value="machine_constants2"/>
        <parameter key="overwrite_mode" value="overwrite"/>
      </operator>
      <connect from_op="Retrieve" from_port="output" to_op="Date to Numerical" to_port="example set input"/>
      <connect from_op="Date to Numerical" from_port="example set output" to_op="Generate Attributes (2)" to_port="example set input"/>
      <connect from_op="Generate Attributes (2)" from_port="example set output" to_op="Generate Attributes (3)" to_port="example set
input"/>
      <connect from_op="Generate Attributes (3)" from_port="example set output" to_op="Select Attributes" to_port="example set input"/>
      <connect from_op="Select Attributes" from_port="example set output" to_op="Rename" to_port="example set input"/>
      <connect from_op="Rename" from_port="example set output" to_op="Numerical to Date" to_port="example set input"/>
      <connect from_op="Numerical to Date" from_port="example set output" to_op="Write Database" to_port="input"/>
      <portSpacing port="source_input 1" spacing="0"/>
      <portSpacing port="sink_result 1" spacing="0"/>
    </process>
  </operator>
</process>
```

9 APPENDIX: TOTAL ANALYSIS

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="5.2.008">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="5.2.008" expanded="true" name="Process">
    <parameter key="notification_email" value="p.v.d.aalst@student.tue.nl"/>
    <process expanded="true" height="816" width="1464">
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Retrieve" width="90" x="45"
y="30">
        <parameter key="repository_entry" value="//DB/asml_pam/Example Sets/machine_constants2"/>
      </operator>
      <operator activated="true" class="generate_id" compatibility="5.2.008" expanded="true" height="76" name="Generate ID" width="90"
x="179" y="30"/>
      <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate
Concatenation" width="90" x="313" y="30">
        <parameter key="first_attribute" value="MACHINE"/>
        <parameter key="second_attribute" value="MACHINE_CONSTANT"/>
      </operator>
      <operator activated="true" class="multiply" compatibility="5.2.008" expanded="true" height="220" name="Multiply" width="90" x="447"
y="30"/>
```

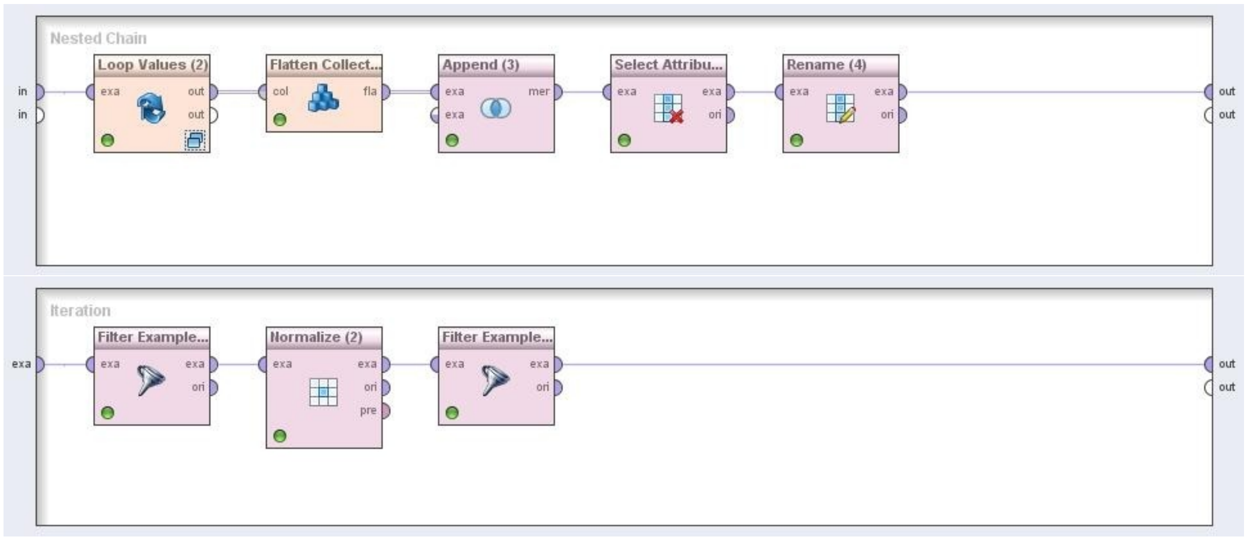
9.1 OVP



```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="76" name="OVP" width="90" x="648"
y="30">
        <process expanded="true" height="637" width="768">
          <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values" width="90"
x="45" y="30">
            <parameter key="attribute" value="MACHINE_CONSTANT"/>
            <parameter key="iteration_macro" value="loopA"/>
            <parameter key="parallelize_iteration" value="true"/>
            <process expanded="true" height="637" width="433">
              <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (4)"
width="90" x="45" y="30">
                <parameter key="condition_class" value="attribute_value_filter"/>
                <parameter key="parameter_string" value="MACHINE_CONSTANT=%{loopA}"/>
              </operator>
              <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize" width="90"
x="180" y="30">
                <parameter key="attribute_filter_type" value="single"/>
                <parameter key="attribute" value="NUMERIC_VALUE"/>
              </operator>
              <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (5)"
width="90" x="313" y="30">
```

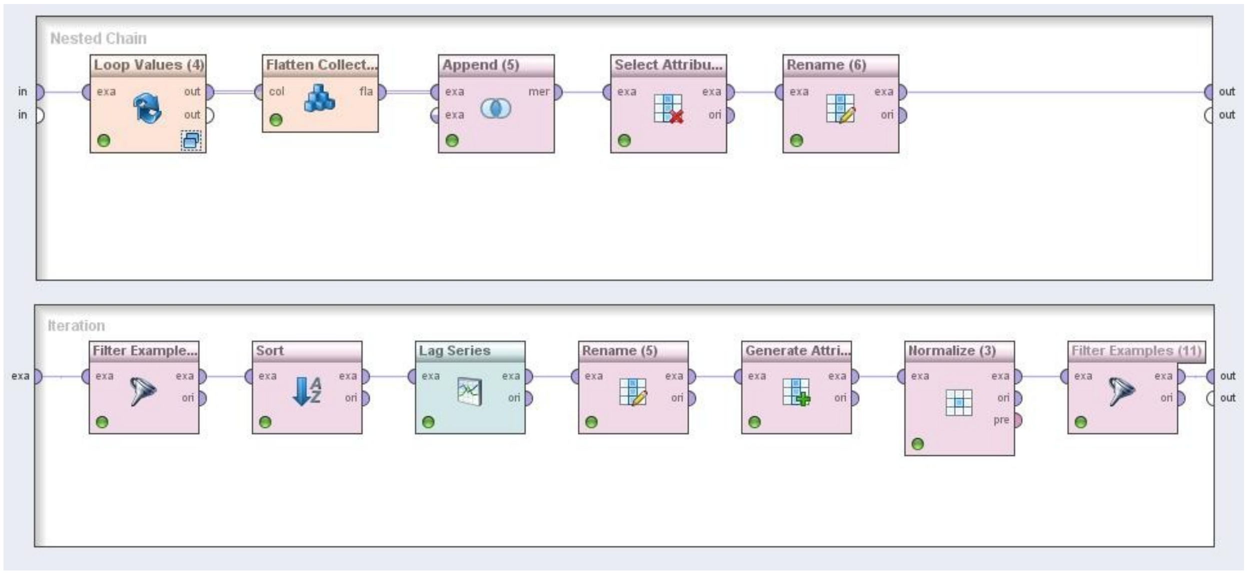
```
          <parameter key="condition_class" value="attribute_value_filter"/>
          <parameter key="parameter_string" value="NUMERIC_VALUE&gt;2.00 || NUMERIC_VALUE&lt;-2.00"/>
        </operator>
        <connect from_port="example set" to_op="Filter Examples (4)" to_port="example set input"/>
        <connect from_op="Filter Examples (4)" from_port="example set output" to_op="Normalize" to_port="example set input"/>
        <connect from_op="Normalize" from_port="example set output" to_op="Filter Examples (5)" to_port="example set input"/>
        <connect from_op="Filter Examples (5)" from_port="example set output" to_port="out 1"/>
        <portSpacing port="source_example set" spacing="0"/>
        <portSpacing port="sink_out 1" spacing="0"/>
        <portSpacing port="sink_out 2" spacing="0"/>
      </process>
    </operator>
    <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection
(2)" width="90" x="179" y="30"/>
    <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append (2)" width="90"
x="313" y="30"/>
    <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (6)"
width="90" x="447" y="30">
      <parameter key="attribute_filter_type" value="subset"/>
      <parameter key="attributes" value="id|NUMERIC_VALUE"/>
    </operator>
    <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (3)" width="90"
x="581" y="30">
      <parameter key="old_name" value="NUMERIC_VALUE"/>
      <parameter key="new_name" value="ValuePop"/>
      <list key="rename_additional_attributes"/>
    </operator>
    <connect from_port="in 1" to_op="Loop Values" to_port="example set"/>
    <connect from_op="Loop Values" from_port="out 1" to_op="Flatten Collection (2)" to_port="collection"/>
    <connect from_op="Flatten Collection (2)" from_port="flat" to_op="Append (2)" to_port="example set 1"/>
    <connect from_op="Append (2)" from_port="merged set" to_op="Select Attributes (6)" to_port="example set input"/>
    <connect from_op="Select Attributes (6)" from_port="example set output" to_op="Rename (3)" to_port="example set input"/>
    <connect from_op="Rename (3)" from_port="example set output" to_port="out 1"/>
    <portSpacing port="source_in 1" spacing="0"/>
    <portSpacing port="source_in 2" spacing="0"/>
    <portSpacing port="sink_out 1" spacing="0"/>
    <portSpacing port="sink_out 2" spacing="0"/>
  </process>
</operator>
```

9.2 OVI



```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="76" name="OVI" width="90" x="648"
y="120">
    <process expanded="true" height="637" width="768">
     <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values (2)"
width="90" x="45" y="30">
       <parameter key="attribute" value="MACHINE_MACHINE_CONSTANT"/>
       <parameter key="iteration_macro" value="loopB"/>
       <parameter key="parallelize_iteration" value="true"/>
       <process expanded="true" height="637" width="500">
         <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (6)"
width="90" x="45" y="30">
           <parameter key="condition_class" value="attribute_value_filter"/>
           <parameter key="parameter_string" value="MACHINE_MACHINE_CONSTANT=%{loopB}"/>
         </operator>
         <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (2)"
width="90" x="179" y="30">
           <parameter key="attribute_filter_type" value="single"/>
           <parameter key="attribute" value="NUMERIC_VALUE"/>
         </operator>
         <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (8)"
width="90" x="313" y="30">
           <parameter key="condition_class" value="attribute_value_filter"/>
           <parameter key="parameter_string" value="NUMERIC_VALUE&gt;2 || NUMERIC_VALUE&lt;-2"/>
         </operator>
         <connect from_port="example set" to_op="Filter Examples (6)" to_port="example set input"/>
         <connect from_op="Filter Examples (6)" from_port="example set output" to_op="Normalize (2)" to_port="example set input"/>
         <connect from_op="Normalize (2)" from_port="example set output" to_op="Filter Examples (8)" to_port="example set input"/>
         <connect from_op="Filter Examples (8)" from_port="example set output" to_port="out 1"/>
         <portSpacing port="source_example set" spacing="0"/>
         <portSpacing port="sink_out 1" spacing="0"/>
         <portSpacing port="sink_out 2" spacing="0"/>
       </process>
     </operator>
     <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection
(3)" width="90" x="179" y="30"/>
     <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append (3)" width="90"
x="313" y="30"/>
     <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (7)"
width="90" x="447" y="30">
       <parameter key="attribute_filter_type" value="subset"/>
       <parameter key="attributes" value="id|NUMERIC_VALUE"/>
     </operator>
     <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (4)" width="90"
x="581" y="30">
       <parameter key="old_name" value="NUMERIC_VALUE"/>
       <parameter key="new_name" value="ValueIndv"/>
       <list key="rename_additional_attributes"/>
    </operator>
    <connect from_port="in 1" to_op="Loop Values (2)" to_port="example set"/>
    <connect from_op="Loop Values (2)" from_port="out 1" to_op="Flatten Collection (3)" to_port="collection"/>
    <connect from_op="Flatten Collection (3)" from_port="flat" to_op="Append (3)" to_port="example set 1"/>
    <connect from_op="Append (3)" from_port="merged set" to_op="Select Attributes (7)" to_port="example set input"/>
    <connect from_op="Select Attributes (7)" from_port="example set output" to_op="Rename (4)" to_port="example set input"/>
    <connect from_op="Rename (4)" from_port="example set output" to_port="out 1"/>
    <portSpacing port="source_in 1" spacing="0"/>
    <portSpacing port="source_in 2" spacing="0"/>
    <portSpacing port="sink_out 1" spacing="0"/>
    <portSpacing port="sink_out 2" spacing="0"/>
  </process>
</operator>
```

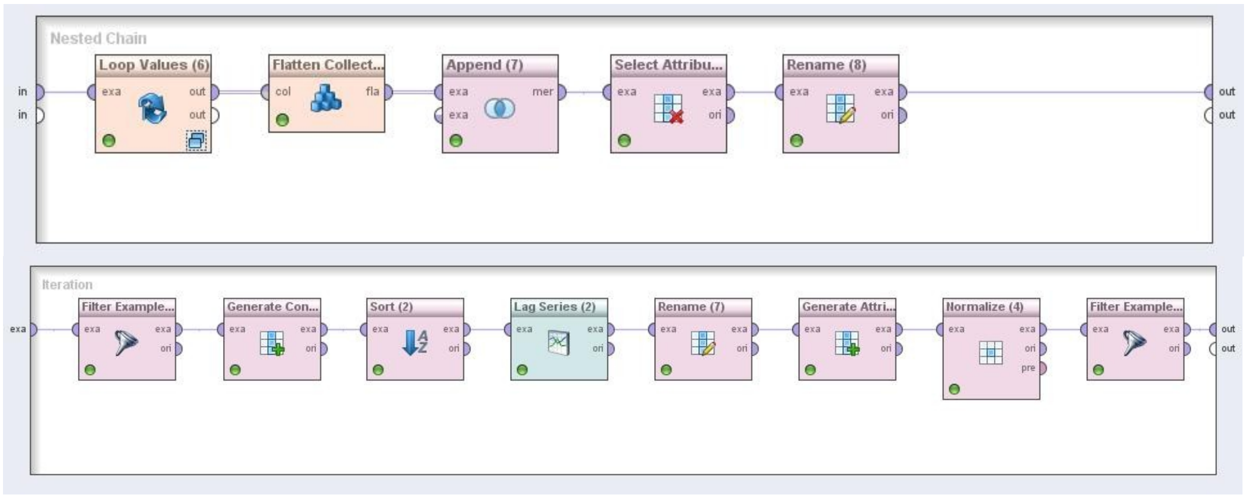
9.3 OJP


Nested Chain
Loop Values (4)
Flatten Collect...
Append (5)
Select Attribu...
Rename (6)
Iteration
Filter Example...
Sort
Lag Series
Rename (5)
Generate Attri...
Normalize (3)
Filter Examples (11)

```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="76" name="OJP" width="90" x="648" y="210">
    <process expanded="true" height="637" width="701">
     <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values (4)" width="90" x="42" y="30">
      <parameter key="attribute" value="MACHINE_CONSTANT"/>
      <parameter key="iteration_macro" value="loopD"/>
      <parameter key="parallelize_iteration" value="true"/>
      <process expanded="true" height="637" width="969">
       <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (9)" width="90" x="45" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="MACHINE_CONSTANT=%{loopD}"/>
       </operator>
       <operator activated="true" class="sort" compatibility="5.2.008" expanded="true" height="76" name="Sort" width="90" x="179" y="30">
        <parameter key="attribute_name" value="CREATION_DATE"/>
       </operator>
       <operator activated="true" class="series:lag_series" compatibility="5.3.000" expanded="true" height="76" name="Lag Series" width="90" x="313" y="30">
        <list key="attributes">
         <parameter key="NUMERIC_VALUE" value="1"/>
        </list>
       </operator>
       <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (5)" width="90" x="447" y="30">
        <parameter key="old_name" value="NUMERIC_VALUE-1"/>
        <parameter key="new_name" value="NUMERIC_VALUE_SERIES"/>
        <list key="rename_additional_attributes"/>
       </operator>
       <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes" width="90" x="581" y="30">
        <list key="function_descriptions">
         <parameter key="NumDiff" value="NUMERIC_VALUE_SERIES-NUMERIC_VALUE"/>
        </list>
       </operator>
       <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (3)" width="90" x="715" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="NumDiff"/>
       </operator>
       <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (11)" width="90" x="849" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="NumDiff&gt;2 || NumDiff&lt;-2"/>
       </operator>
```

```
          <connect from_port="example set" to_op="Filter Examples (9)" to_port="example set input"/>
          <connect from_op="Filter Examples (9)" from_port="example set output" to_op="Sort" to_port="example set input"/>
          <connect from_op="Sort" from_port="example set output" to_op="Lag Series" to_port="example set input"/>
          <connect from_op="Lag Series" from_port="example set output" to_op="Rename (5)" to_port="example set input"/>
          <connect from_op="Rename (5)" from_port="example set output" to_op="Generate Attributes" to_port="example set input"/>
          <connect from_op="Generate Attributes" from_port="example set output" to_op="Normalize (3)" to_port="example set input"/>
          <connect from_op="Normalize (3)" from_port="example set output" to_op="Filter Examples (11)" to_port="example set input"/>
          <connect from_op="Filter Examples (11)" from_port="example set output" to_port="out 1"/>
          <portSpacing port="source_example set" spacing="0"/>
          <portSpacing port="sink_out 1" spacing="0"/>
          <portSpacing port="sink_out 2" spacing="0"/>
        </process>
      </operator>
      <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection
(5)" width="90" x="176" y="30"/>
      <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append (5)" width="90"
x="313" y="30"/>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (9)"
width="90" x="447" y="30">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attributes" value="id|NumDiff"/>
      </operator>
      <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (6)" width="90"
x="581" y="30">
        <parameter key="old_name" value="NumDiff"/>
        <parameter key="new_name" value="JumpPop"/>
        <list key="rename_additional_attributes"/>
      </operator>
      <connect from_port="in 1" to_op="Loop Values (4)" to_port="example set"/>
      <connect from_op="Loop Values (4)" from_port="out 1" to_op="Flatten Collection (5)" to_port="collection"/>
      <connect from_op="Flatten Collection (5)" from_port="flat" to_op="Append (5)" to_port="example set 1"/>
      <connect from_op="Append (5)" from_port="merged set" to_op="Select Attributes (9)" to_port="example set input"/>
      <connect from_op="Select Attributes (9)" from_port="example set output" to_op="Rename (6)" to_port="example set input"/>
      <connect from_op="Rename (6)" from_port="example set output" to_port="out 1"/>
      <portSpacing port="source_in 1" spacing="0"/>
      <portSpacing port="source_in 2" spacing="0"/>
      <portSpacing port="sink_out 1" spacing="0"/>
      <portSpacing port="sink_out 2" spacing="0"/>
    </process>
  </operator>
```

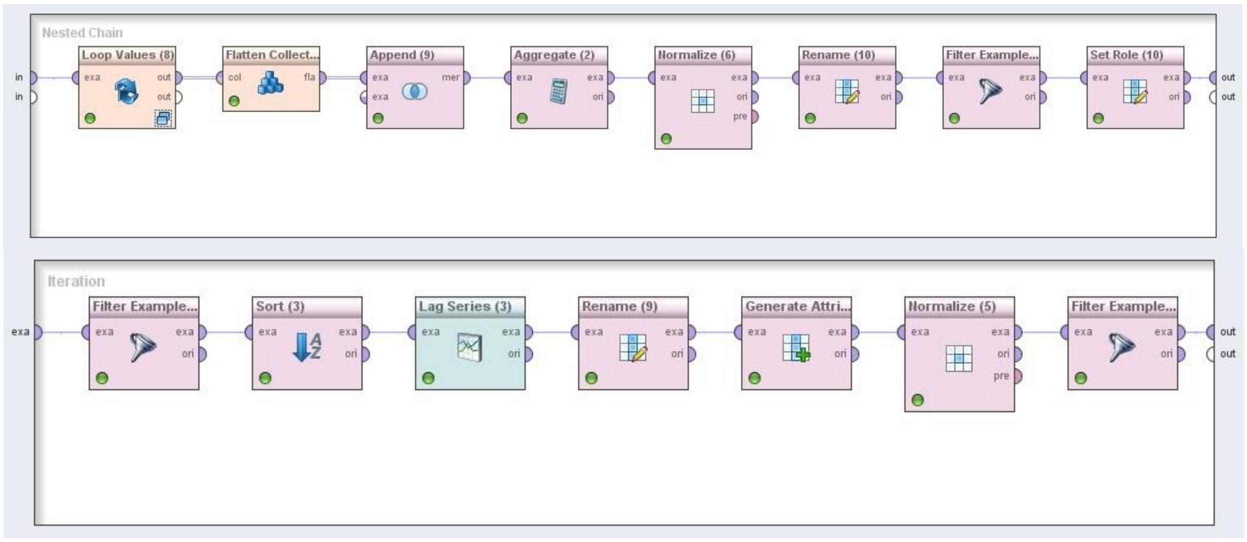
9.4 OJI



```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="76" name="OJI" width="90" x="648"
y="300">
    <process expanded="true" height="655" width="902">
     <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values (6)"
width="90" x="46" y="30">
      <parameter key="attribute" value="MACHINE_MACHINE_CONSTANT"/>
      <parameter key="iteration_macro" value="loopF"/>
      <parameter key="parallelize_iteration" value="true"/>
      <process expanded="true" height="637" width="1103">
       <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (12)"
width="90" x="45" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="MACHINE_MACHINE_CONSTANT=%{loopF}"/>
       </operator>
       <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate
Concatenation (5)" width="90" x="180" y="30">
        <parameter key="first_attribute" value="MACHINE"/>
        <parameter key="second_attribute" value="STE_CODE"/>
       </operator>
       <operator activated="true" class="sort" compatibility="5.2.008" expanded="true" height="76" name="Sort (2)" width="90" x="313"
y="30">
        <parameter key="attribute_name" value="CREATION_DATE"/>
       </operator>
       <operator activated="true" class="series:lag_series" compatibility="5.3.000" expanded="true" height="76" name="Lag Series (2)"
width="90" x="447" y="30">
        <list key="attributes">
         <parameter key="NUMERIC_VALUE" value="1"/>
        </list>
       </operator>
       <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (7)" width="90"
x="581" y="30">
        <parameter key="old_name" value="NUMERIC_VALUE-1"/>
        <parameter key="new_name" value="NUMERIC_VALUE_SERIES"/>
        <list key="rename_additional_attributes"/>
       </operator>
       <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate
Attributes (4)" width="90" x="715" y="30">
        <list key="function_descriptions">
         <parameter key="NumDiff" value="NUMERIC_VALUE_SERIES-NUMERIC_VALUE"/>
        </list>
       </operator>
       <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (4)"
width="90" x="849" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="NumDiff"/>
       </operator>
       <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (14)"
width="90" x="983" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
```

```
          <parameter key="parameter_string" value="NumDiff&gt;2 || NumDiff&lt;-2"/>
        </operator>
        <connect from_port="example set" to_op="Filter Examples (12)" to_port="example set input"/>
        <connect from_op="Filter Examples (12)" from_port="example set output" to_op="Generate Concatenation (5)" to_port="example set
input"/>
        <connect from_op="Generate Concatenation (5)" from_port="example set output" to_op="Sort (2)" to_port="example set input"/>
        <connect from_op="Sort (2)" from_port="example set output" to_op="Lag Series (2)" to_port="example set input"/>
        <connect from_op="Lag Series (2)" from_port="example set output" to_op="Rename (7)" to_port="example set input"/>
        <connect from_op="Rename (7)" from_port="example set output" to_op="Generate Attributes (4)" to_port="example set input"/>
        <connect from_op="Generate Attributes (4)" from_port="example set output" to_op="Normalize (4)" to_port="example set input"/>
        <connect from_op="Normalize (4)" from_port="example set output" to_op="Filter Examples (14)" to_port="example set input"/>
        <connect from_op="Filter Examples (14)" from_port="example set output" to_port="out 1"/>
        <portSpacing port="source_example set" spacing="0"/>
        <portSpacing port="sink_out 1" spacing="0"/>
        <portSpacing port="sink_out 2" spacing="0"/>
      </process>
    </operator>
    <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection
(7)" width="90" x="181" y="30"/>
    <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append (7)" width="90"
x="316" y="30"/>
    <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (4)"
width="90" x="447" y="30">
      <parameter key="attribute_filter_type" value="subset"/>
      <parameter key="attributes" value="id|NumDiff"/>
    </operator>
    <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (8)" width="90"
x="581" y="30">
      <parameter key="old_name" value="NumDiff"/>
      <parameter key="new_name" value="JumpIndv"/>
      <list key="rename_additional_attributes"/>
    </operator>
    <connect from_port="in 1" to_op="Loop Values (6)" to_port="example set"/>
    <connect from_op="Loop Values (6)" from_port="out 1" to_op="Flatten Collection (7)" to_port="collection"/>
    <connect from_op="Flatten Collection (7)" from_port="flat" to_op="Append (7)" to_port="example set 1"/>
    <connect from_op="Append (7)" from_port="merged set" to_op="Select Attributes (4)" to_port="example set input"/>
    <connect from_op="Select Attributes (4)" from_port="example set output" to_op="Rename (8)" to_port="example set input"/>
    <connect from_op="Rename (8)" from_port="example set output" to_port="out 1"/>
    <portSpacing port="source_in 1" spacing="0"/>
    <portSpacing port="source_in 2" spacing="0"/>
    <portSpacing port="sink_out 1" spacing="0"/>
    <portSpacing port="sink_out 2" spacing="0"/>
  </process>
</operator>
```

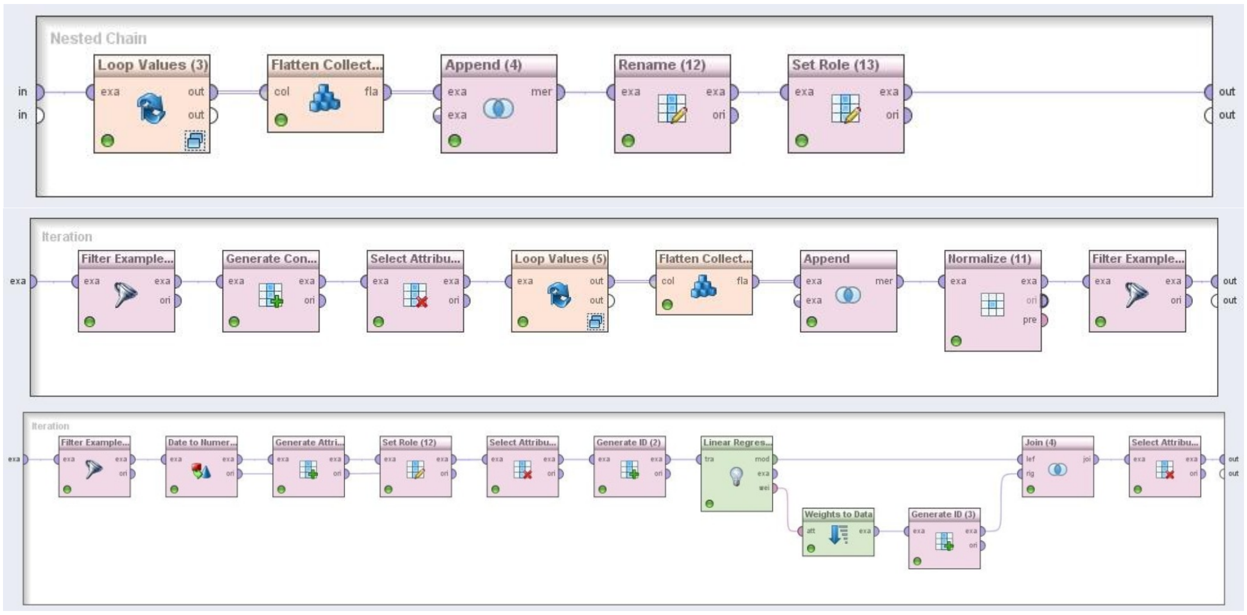
9.5 OJFP


Nested Chain
Loop Values (8)
Flatten Collect...
Append (9)
Aggregate (2)
Normalize (6)
Rename (10)
Filter Example...
Set Role (10)
Iteration
Filter Example...
Sort (3)
Lag Series (3)
Rename (9)
Generate Attri...
Normalize (5)
Filter Example...

```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="76" name="OJFP" width="90" x="648" y="390">
    <process expanded="true" height="637" width="1103">
     <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values (8)" width="90" x="45" y="30">
      <parameter key="attribute" value="MACHINE_MACHINE_CONSTANT"/>
      <parameter key="iteration_macro" value="loopH"/>
      <parameter key="parallelize_iteration" value="true"/>
      <process expanded="true" height="637" width="969">
       <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (15)" width="90" x="45" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="MACHINE_MACHINE_CONSTANT=%{loopH}"/>
       </operator>
       <operator activated="true" class="sort" compatibility="5.2.008" expanded="true" height="76" name="Sort (3)" width="90" x="179" y="30">
        <parameter key="attribute_name" value="CREATION_DATE"/>
       </operator>
       <operator activated="true" class="series:lag_series" compatibility="5.3.000" expanded="true" height="76" name="Lag Series (3)" width="90" x="313" y="30">
        <list key="attributes">
         <parameter key="NUMERIC_VALUE" value="1"/>
        </list>
       </operator>
       <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (9)" width="90" x="447" y="30">
        <parameter key="old_name" value="NUMERIC_VALUE-1"/>
        <parameter key="new_name" value="NUMERIC_VALUE_SERIES"/>
        <list key="rename_additional_attributes"/>
       </operator>
       <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (5)" width="90" x="581" y="30">
        <list key="function_descriptions">
         <parameter key="NumDiff" value="NUMERIC_VALUE_SERIES-NUMERIC_VALUE"/>
        </list>
       </operator>
       <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (5)" width="90" x="715" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="NumDiff"/>
       </operator>
       <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (17)" width="90" x="849" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="NumDiff&gt;0.50 || NumDiff&lt;-0.50"/>
       </operator>
       <connect from_port="example set" to_op="Filter Examples (15)" to_port="example set input"/>
```

```
                <connect from_op="Filter Examples (15)" from_port="example set output" to_op="Sort (3)" to_port="example set input"/>
                <connect from_op="Sort (3)" from_port="example set output" to_op="Lag Series (3)" to_port="example set input"/>
                <connect from_op="Lag Series (3)" from_port="example set output" to_op="Rename (9)" to_port="example set input"/>
                <connect from_op="Rename (9)" from_port="example set output" to_op="Generate Attributes (5)" to_port="example set input"/>
                <connect from_op="Generate Attributes (5)" from_port="example set output" to_op="Normalize (5)" to_port="example set input"/>
                <connect from_op="Normalize (5)" from_port="example set output" to_op="Filter Examples (17)" to_port="example set input"/>
                <connect from_op="Filter Examples (17)" from_port="example set output" to_port="out 1"/>
                <portSpacing port="source_example set" spacing="0"/>
                <portSpacing port="sink_out 1" spacing="0"/>
                <portSpacing port="sink_out 2" spacing="0"/>
              </process>
            </operator>
            <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection (9)" width="90" x="179" y="30"/>
            <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append (9)" width="90" x="313" y="30"/>
            <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (2)" width="90" x="447" y="30">
              <list key="aggregation_attributes">
                <parameter key="NumDiff" value="count"/>
              </list>
              <parameter key="group_by_attributes" value="|MACHINE_MACHINE_CONSTANT"/>
            </operator>
            <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (6)" width="90" x="581" y="30">
              <parameter key="attribute_filter_type" value="single"/>
              <parameter key="attribute" value="count(NumDiff)"/>
            </operator>
            <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (10)" width="90" x="715" y="30">
              <parameter key="old_name" value="count(NumDiff)"/>
              <parameter key="new_name" value="JumpFreq"/>
              <list key="rename_additional_attributes"/>
            </operator>
            <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (18)" width="90" x="849" y="30">
              <parameter key="condition_class" value="attribute_value_filter"/>
              <parameter key="parameter_string" value="JumpFreq&gt;2 || JumpFreq&lt;-2"/>
            </operator>
            <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (10)" width="90" x="983" y="30">
              <parameter key="name" value="MACHINE_MACHINE_CONSTANT"/>
              <parameter key="target_role" value="id"/>
              <list key="set_additional_roles"/>
            </operator>
            <connect from_port="in 1" to_op="Loop Values (8)" to_port="example set"/>
            <connect from_op="Loop Values (8)" from_port="out 1" to_op="Flatten Collection (9)" to_port="collection"/>
            <connect from_op="Flatten Collection (9)" from_port="flat" to_op="Append (9)" to_port="example set 1"/>
            <connect from_op="Append (9)" from_port="merged set" to_op="Aggregate (2)" to_port="example set input"/>
            <connect from_op="Aggregate (2)" from_port="example set output" to_op="Normalize (6)" to_port="example set input"/>
            <connect from_op="Normalize (6)" from_port="example set output" to_op="Rename (10)" to_port="example set input"/>
            <connect from_op="Rename (10)" from_port="example set output" to_op="Filter Examples (18)" to_port="example set input"/>
            <connect from_op="Filter Examples (18)" from_port="example set output" to_op="Set Role (10)" to_port="example set input"/>
            <connect from_op="Set Role (10)" from_port="example set output" to_port="out 1"/>
            <portSpacing port="source_in 1" spacing="0"/>
            <portSpacing port="source_in 2" spacing="0"/>
            <portSpacing port="sink_out 1" spacing="0"/>
            <portSpacing port="sink_out 2" spacing="0"/>
          </process>
        </operator>
```

9.6 OTP

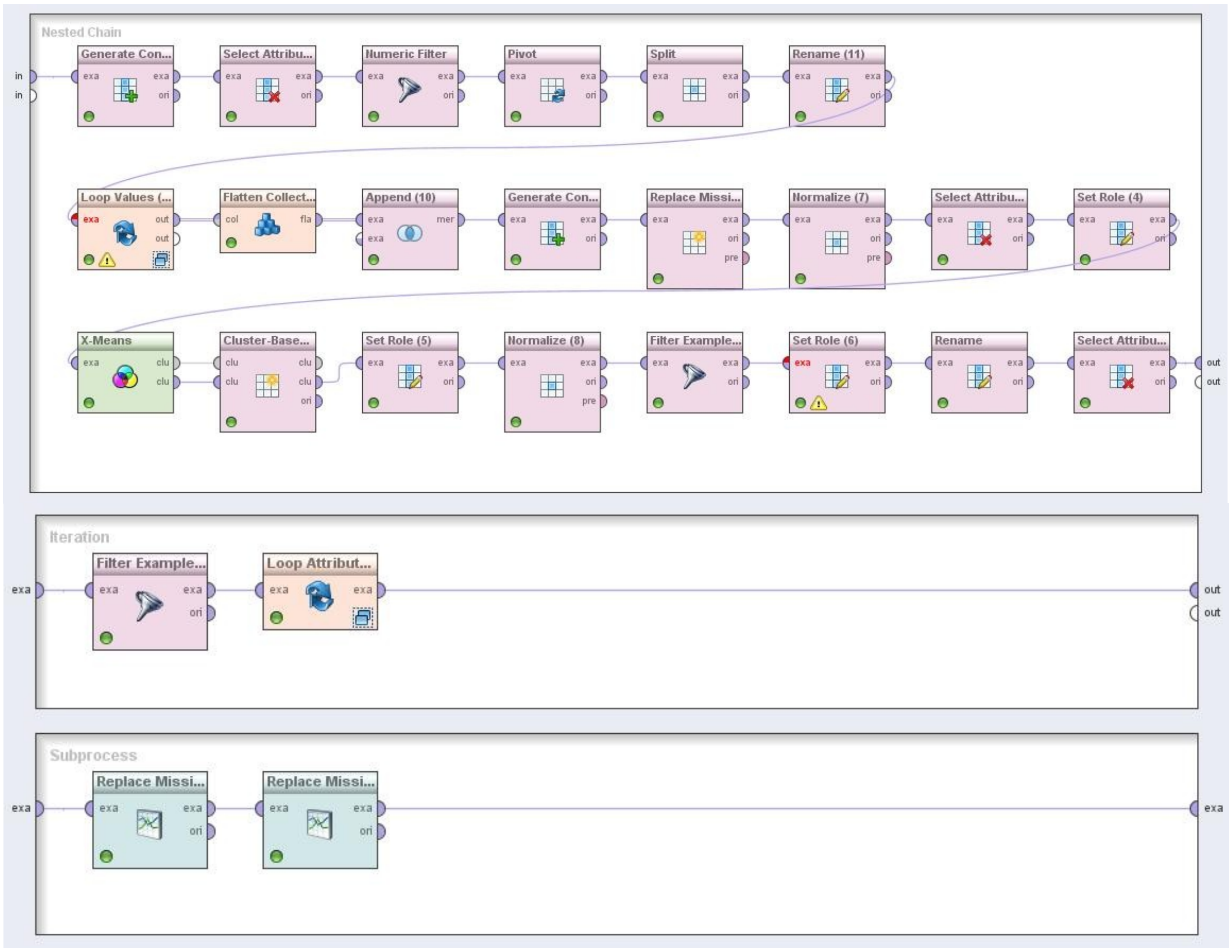


```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="76" name="OTP" width="90" x="648"
y="480">
    <process expanded="true" height="655" width="815">
     <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values (3)"
width="90" x="179" y="30">
      <parameter key="attribute" value="MACHINE_CONSTANT"/>
      <parameter key="iteration_macro" value="loopA"/>
      <process expanded="true" height="655" width="1284">
       <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (3)"
width="90" x="45" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="MACHINE_CONSTANT=%{loopA}"/>
       </operator>
       <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate
Concatenation (4)" width="90" x="180" y="30">
        <parameter key="first_attribute" value="MACHINE"/>
        <parameter key="second_attribute" value="DATA_FORMAT"/>
       </operator>
       <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes
(15)" width="90" x="315" y="30">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter                                                                                    key="attributes"
value="CREATION_DATE|MACHINE|MACHINE_DATA_FORMAT|MACHINE_MACHINE_CONSTANT|NUMERIC_VALUE|MACHINE_CONSTANT"/>
        <parameter key="include_special_attributes" value="true"/>
       </operator>
       <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values (5)"
width="90" x="447" y="30">
        <parameter key="attribute" value="MACHINE_DATA_FORMAT"/>
        <parameter key="iteration_macro" value="loopB"/>
        <parameter key="parallelize_iteration" value="true"/>
        <process expanded="true" height="637" width="1552">
         <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples
(7)" width="90" x="45" y="30">
          <parameter key="condition_class" value="attribute_value_filter"/>
          <parameter key="parameter_string" value="MACHINE_DATA_FORMAT=%{loopB}"/>
         </operator>
         <operator activated="true" class="date_to_numerical" compatibility="5.2.008" expanded="true" height="76" name="Date to
Numerical" width="90" x="179" y="30">
          <parameter key="attribute_name" value="CREATION_DATE"/>
          <parameter key="millisecond_relative_to" value="epoch"/>
         </operator>
         <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate
Attributes (3)" width="90" x="313" y="30">
          <list key="function_descriptions">
```

```
            <parameter key="Time" value="(CREATION_DATE/60000000)-22000"/>
          </list>
        </operator>
        <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (12)" width="90" x="447" y="30">
          <parameter key="name" value="MACHINE_MACHINE_CONSTANT"/>
          <parameter key="target_role" value="id"/>
          <list key="set_additional_roles">
            <parameter key="NUMERIC_VALUE" value="label"/>
          </list>
        </operator>
        <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (16)" width="90" x="581" y="30">
          <parameter key="attribute_filter_type" value="subset"/>
          <parameter key="attributes" value="MACHINE_MACHINE_CONSTANT|NUMERIC_VALUE|Time"/>
        </operator>
        <operator activated="true" class="generate_id" compatibility="5.2.008" expanded="true" height="76" name="Generate ID (2)" width="90" x="179" y="165"/>
        <operator activated="true" class="linear_regression" compatibility="5.2.008" expanded="true" height="94" name="Linear Regression" width="90" x="715" y="30">
          <parameter key="eliminate_colinear_features" value="false"/>
          <parameter key="min_tolerance" value="1.0"/>
        </operator>
        <operator activated="true" class="weights_to_data" compatibility="5.2.008" expanded="true" height="60" name="Weights to Data" width="90" x="849" y="30"/>
        <operator activated="true" class="generate_id" compatibility="5.2.008" expanded="true" height="76" name="Generate ID (3)" width="90" x="983" y="30"/>
        <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (4)" width="90" x="983" y="165">
          <list key="key_attributes"/>
        </operator>
        <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (17)" width="90" x="1184" y="30">
          <parameter key="attribute_filter_type" value="subset"/>
          <parameter key="attributes" value="MACHINE_MACHINE_CONSTANT|Weight||Time|MACHINE|MACHINE_CONSTANT"/>
          <parameter key="include_special_attributes" value="true"/>
        </operator>
        <connect from_port="example set" to_op="Filter Examples (7)" to_port="example set input"/>
        <connect from_op="Filter Examples (7)" from_port="example set output" to_op="Date to Numerical" to_port="example set input"/>
        <connect from_op="Date to Numerical" from_port="example set output" to_op="Generate Attributes (3)" to_port="example set input"/>
        <connect from_op="Date to Numerical" from_port="original" to_op="Generate ID (2)" to_port="example set input"/>
        <connect from_op="Generate Attributes (3)" from_port="example set output" to_op="Set Role (12)" to_port="example set input"/>
        <connect from_op="Set Role (12)" from_port="example set output" to_op="Select Attributes (16)" to_port="example set input"/>
        <connect from_op="Select Attributes (16)" from_port="example set output" to_op="Linear Regression" to_port="training set"/>
        <connect from_op="Generate ID (2)" from_port="example set output" to_op="Join (4)" to_port="left"/>
        <connect from_op="Linear Regression" from_port="weights" to_op="Weights to Data" to_port="attribute weights"/>
        <connect from_op="Weights to Data" from_port="example set" to_op="Generate ID (3)" to_port="example set input"/>
        <connect from_op="Generate ID (3)" from_port="example set output" to_op="Join (4)" to_port="right"/>
        <connect from_op="Join (4)" from_port="join" to_op="Select Attributes (17)" to_port="example set input"/>
        <connect from_op="Select Attributes (17)" from_port="example set output" to_port="out 1"/>
        <portSpacing port="source_example set" spacing="0"/>
        <portSpacing port="sink_out 1" spacing="0"/>
        <portSpacing port="sink_out 2" spacing="0"/>
      </process>
    </operator>
    <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection" width="90" x="585" y="30"/>
    <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append" width="90" x="720" y="30"/>
    <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (11)" width="90" x="849" y="30">
      <parameter key="attribute_filter_type" value="single"/>
      <parameter key="attribute" value="Weight"/>
    </operator>
    <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (10)" width="90" x="983" y="30">
      <parameter key="condition_class" value="attribute_value_filter"/>
      <parameter key="parameter_string" value="Weight &lt;= -2 || Weight &gt;= 2"/>
    </operator>
```

```
          <connect from_port="example set" to_op="Filter Examples (3)" to_port="example set input"/>
          <connect from_op="Filter Examples (3)" from_port="example set output" to_op="Generate Concatenation (4)" to_port="example set
input"/>
          <connect from_op="Generate Concatenation (4)" from_port="example set output" to_op="Select Attributes (15)" to_port="example
set input"/>
          <connect from_op="Select Attributes (15)" from_port="example set output" to_op="Loop Values (5)" to_port="example set"/>
          <connect from_op="Loop Values (5)" from_port="out 1" to_op="Flatten Collection" to_port="collection"/>
          <connect from_op="Flatten Collection" from_port="flat" to_op="Append" to_port="example set 1"/>
          <connect from_op="Append" from_port="merged set" to_op="Normalize (11)" to_port="example set input"/>
          <connect from_op="Normalize (11)" from_port="example set output" to_op="Filter Examples (10)" to_port="example set input"/>
          <connect from_op="Filter Examples (10)" from_port="example set output" to_port="out 1"/>
          <portSpacing port="source_example set" spacing="0"/>
          <portSpacing port="sink_out 1" spacing="0"/>
          <portSpacing port="sink_out 2" spacing="0"/>
        </process>
      </operator>
      <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection
(4)" width="90" x="313" y="30"/>
      <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append (4)" width="90"
x="447" y="30"/>
      <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (12)" width="90"
x="581" y="30">
        <parameter key="old_name" value="Weight"/>
        <parameter key="new_name" value="TrendPop"/>
        <list key="rename_additional_attributes"/>
      </operator>
      <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (13)" width="90"
x="715" y="30">
        <parameter key="name" value="MACHINE_MACHINE_CONSTANT"/>
        <parameter key="target_role" value="id"/>
        <list key="set_additional_roles"/>
      </operator>
      <connect from_port="in 1" to_op="Loop Values (3)" to_port="example set"/>
      <connect from_op="Loop Values (3)" from_port="out 1" to_op="Flatten Collection (4)" to_port="collection"/>
      <connect from_op="Flatten Collection (4)" from_port="flat" to_op="Append (4)" to_port="example set 1"/>
      <connect from_op="Append (4)" from_port="merged set" to_op="Rename (12)" to_port="example set input"/>
      <connect from_op="Rename (12)" from_port="example set output" to_op="Set Role (13)" to_port="example set input"/>
      <connect from_op="Set Role (13)" from_port="example set output" to_port="out 1"/>
      <portSpacing port="source_in 1" spacing="0"/>
      <portSpacing port="source_in 2" spacing="0"/>
      <portSpacing port="sink_out 1" spacing="0"/>
      <portSpacing port="sink_out 2" spacing="0"/>
    </process>
  </operator>
```

9.7 OMCV



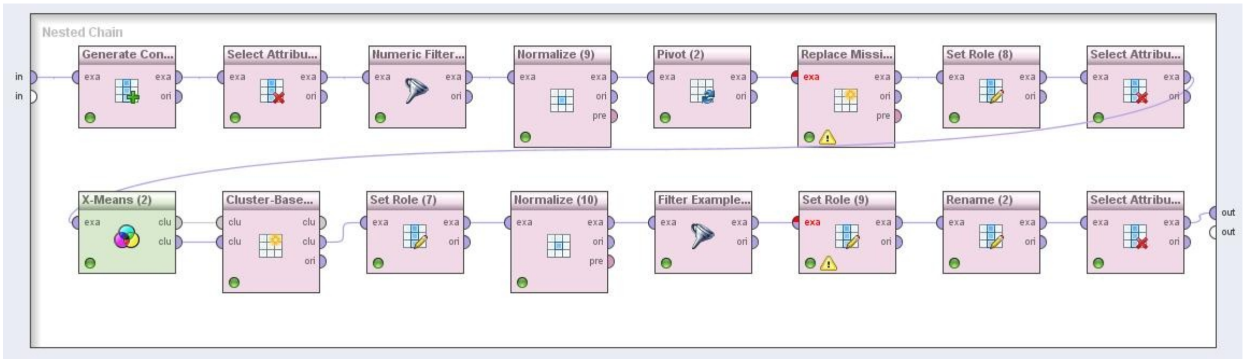
Nested Chain
Generate Con...
Select Attribu...
Numeric Filter
Pivot
Split
Rename (11)
Loop Values (...
Flatten Collect...
Append (10)
Generate Con...
Replace Missi...
Normalize (7)
Select Attribu...
Set Role (4)
X-Means
Cluster-Base...
Set Role (5)
Normalize (8)
Filter Example...
Set Role (6)
Rename
Select Attribu...
Iteration
Filter Example...
Loop Attribut...
Subprocess
Replace Missi...
Replace Missi...

```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="76" name="OMCV" width="90" x="648" y="570">
    <process expanded="true" height="637" width="1103">
      <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate Concatenation (3)" width="90" x="45" y="30">
        <parameter key="first_attribute" value="MACHINE"/>
        <parameter key="second_attribute" value="CREATION_DATE"/>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (5)" width="90" x="179" y="30">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attributes" value="MACHINE|MACHINE_CONSTANT|MACHINE_CREATION_DATE|NUMERIC_VALUE"/>
      </operator>
      <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Numeric Filter" width="90" x="313" y="30">
        <parameter key="condition_class" value="no_missing_attributes"/>
        <parameter key="parameter_string" value="NUMERIC_VALUE=?"/>
      </operator>
      <operator activated="true" class="pivot" compatibility="5.2.008" expanded="true" height="76" name="Pivot" width="90" x="447" y="30">
        <parameter key="group_attribute" value="MACHINE_CREATION_DATE"/>
        <parameter key="index_attribute" value="MACHINE_CONSTANT"/>
        <parameter key="consider_weights" value="false"/>
        <parameter key="skip_constant_attributes" value="false"/>
      </operator>
      <operator activated="true" class="split" compatibility="5.2.008" expanded="true" height="76" name="Split" width="90" x="581" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="MACHINE_CREATION_DATE"/>
        <parameter key="split_pattern" value="_"/>
      </operator>
```

```
     <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (11)" width="90"
x="715" y="30">
       <parameter key="old_name" value="MACHINE_CREATION_DATE_1"/>
       <parameter key="new_name" value="MACHINE"/>
       <list key="rename_additional_attributes">
         <parameter key="MACHINE_CREATION_DATE_2" value="CREATION_DATE"/>
       </list>
     </operator>
     <operator activated="true" class="loop_values" compatibility="5.2.008" expanded="true" height="76" name="Loop Values (10)"
width="90" x="45" y="165">
       <parameter key="attribute" value="MACHINE"/>
       <parameter key="iteration_macro" value="MACH"/>
       <process expanded="true" height="637" width="299">
         <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (19)"
width="90" x="45" y="30">
           <parameter key="condition_class" value="attribute_value_filter"/>
           <parameter key="parameter_string" value="MACHINE=%{MACH}"/>
         </operator>
         <operator activated="true" class="loop_attributes" compatibility="5.2.008" expanded="true" height="60" name="Loop Attributes"
width="90" x="179" y="30">
           <parameter key="iteration_macro" value="ATT"/>
           <process expanded="true" height="637" width="299">
             <operator activated="true" class="series:replace_missing_series_values" compatibility="5.3.000" expanded="true" height="76"
name="Replace Missing Values (Series)" width="90" x="45" y="30">
               <parameter key="attribute_name" value="%{ATT}"/>
             </operator>
             <operator activated="true" class="series:replace_missing_series_values" compatibility="5.3.000" expanded="true" height="76"
name="Replace Missing Values (2)" width="90" x="179" y="30">
               <parameter key="attribute_name" value="%{ATT}"/>
               <parameter key="replacement" value="next value"/>
             </operator>
             <connect from_port="example set" to_op="Replace Missing Values (Series)" to_port="example set input"/>
             <connect from_op="Replace Missing Values (Series)" from_port="example set output" to_op="Replace Missing Values (2)"
to_port="example set input"/>
             <connect from_op="Replace Missing Values (2)" from_port="example set output" to_port="example set"/>
             <portSpacing port="source_example set" spacing="0"/>
             <portSpacing port="sink_example set" spacing="0"/>
           </process>
         </operator>
         <connect from_port="example set" to_op="Filter Examples (19)" to_port="example set input"/>
         <connect from_op="Filter Examples (19)" from_port="example set output" to_op="Loop Attributes" to_port="example set"/>
         <connect from_op="Loop Attributes" from_port="example set" to_port="out 1"/>
         <portSpacing port="source_example set" spacing="0"/>
         <portSpacing port="sink_out 1" spacing="0"/>
         <portSpacing port="sink_out 2" spacing="0"/>
       </process>
     </operator>
     <operator activated="true" class="flatten_collection" compatibility="5.2.008" expanded="true" height="60" name="Flatten Collection
(10)" width="90" x="179" y="165"/>
     <operator activated="true" class="append" compatibility="5.2.008" expanded="true" height="76" name="Append (10)" width="90"
x="313" y="165"/>
     <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate
Concatenation (10)" width="90" x="447" y="165">
       <parameter key="first_attribute" value="MACHINE"/>
       <parameter key="second_attribute" value="CREATION_DATE"/>
     </operator>
     <operator activated="true" class="replace_missing_values" compatibility="5.2.008" expanded="true" height="94" name="Replace
Missing Values" width="90" x="581" y="165">
       <list key="columns"/>
     </operator>
     <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (7)" width="90"
x="715" y="165"/>
     <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes
(10)" width="90" x="849" y="165">
       <parameter key="attribute_filter_type" value="subset"/>
       <parameter key="attributes" value="CREATION_DATE|MACHINE_CREATION_DATE_3||MACHINE"/>
       <parameter key="invert_selection" value="true"/>
     </operator>
     <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (4)" width="90"
x="983" y="165">
       <parameter key="name" value="MACHINE_CREATION_DATE"/>
```

```
        <parameter key="target_role" value="label"/>
        <list key="set_additional_roles"/>
      </operator>
      <operator activated="true" class="x_means" compatibility="5.2.008" expanded="true" height="76" name="X-Means" width="90" x="45" y="300"/>
      <operator activated="true" class="anomalydetection:Cluster-Based Local Outlier Factor (CBLOF)" compatibility="1.3.000" expanded="true" height="94" name="Cluster-Based Local Outlier Factor (CBLOF)" width="90" x="179" y="300"/>
      <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (5)" width="90" x="313" y="300">
        <parameter key="name" value="outlier"/>
        <list key="set_additional_roles">
          <parameter key="MACHINE_CREATION_DATE" value="regular"/>
          <parameter key="cluster" value="regular"/>
        </list>
      </operator>
      <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (8)" width="90" x="447" y="300">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="outlier"/>
      </operator>
      <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (20)" width="90" x="581" y="300">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="outlier&gt;2"/>
      </operator>
      <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (6)" width="90" x="715" y="300">
        <parameter key="name" value="MACHINE_CREATION_DATE"/>
        <parameter key="target_role" value="id"/>
        <list key="set_additional_roles"/>
      </operator>
      <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename" width="90" x="849" y="300">
        <parameter key="old_name" value="outlier"/>
        <parameter key="new_name" value="ClusterValue"/>
        <list key="rename_additional_attributes"/>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (12)" width="90" x="983" y="300">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attributes" value="|ClusterValue|MACHINE_CREATION_DATE"/>
      </operator>
      <connect from_port="in 1" to_op="Generate Concatenation (3)" to_port="example set input"/>
      <connect from_op="Generate Concatenation (3)" from_port="example set output" to_op="Select Attributes (5)" to_port="example set input"/>
      <connect from_op="Select Attributes (5)" from_port="example set output" to_op="Numeric Filter" to_port="example set input"/>
      <connect from_op="Numeric Filter" from_port="example set output" to_op="Pivot" to_port="example set input"/>
      <connect from_op="Pivot" from_port="example set output" to_op="Split" to_port="example set input"/>
      <connect from_op="Split" from_port="example set output" to_op="Rename (11)" to_port="example set input"/>
      <connect from_op="Rename (11)" from_port="example set output" to_op="Loop Values (10)" to_port="example set"/>
      <connect from_op="Loop Values (10)" from_port="out 1" to_op="Flatten Collection (10)" to_port="collection"/>
      <connect from_op="Flatten Collection (10)" from_port="flat" to_op="Append (10)" to_port="example set 1"/>
      <connect from_op="Append (10)" from_port="merged set" to_op="Generate Concatenation (10)" to_port="example set input"/>
      <connect from_op="Generate Concatenation (10)" from_port="example set output" to_op="Replace Missing Values" to_port="example set input"/>
      <connect from_op="Replace Missing Values" from_port="example set output" to_op="Normalize (7)" to_port="example set input"/>
      <connect from_op="Normalize (7)" from_port="example set output" to_op="Select Attributes (10)" to_port="example set input"/>
      <connect from_op="Select Attributes (10)" from_port="example set output" to_op="Set Role (4)" to_port="example set input"/>
      <connect from_op="Set Role (4)" from_port="example set output" to_op="X-Means" to_port="example set"/>
      <connect from_op="X-Means" from_port="cluster model" to_op="Cluster-Based Local Outlier Factor (CBLOF)" to_port="cluster Model"/>
      <connect from_op="X-Means" from_port="clustered set" to_op="Cluster-Based Local Outlier Factor (CBLOF)" to_port="clustered set"/>
      <connect from_op="Cluster-Based Local Outlier Factor (CBLOF)" from_port="clustered set" to_op="Set Role (5)" to_port="example set input"/>
      <connect from_op="Set Role (5)" from_port="example set output" to_op="Normalize (8)" to_port="example set input"/>
      <connect from_op="Normalize (8)" from_port="example set output" to_op="Filter Examples (20)" to_port="example set input"/>
      <connect from_op="Filter Examples (20)" from_port="example set output" to_op="Set Role (6)" to_port="example set input"/>
      <connect from_op="Set Role (6)" from_port="example set output" to_op="Rename" to_port="example set input"/>
      <connect from_op="Rename" from_port="example set output" to_op="Select Attributes (12)" to_port="example set input"/>
      <connect from_op="Select Attributes (12)" from_port="example set output" to_port="out 1"/>
      <portSpacing port="source_in 1" spacing="0"/>
      <portSpacing port="source_in 2" spacing="0"/>
      <portSpacing port="sink_out 1" spacing="270"/>
      <portSpacing port="sink_out 2" spacing="0"/>
    </process>
  </operator>
```

9.8 OCS



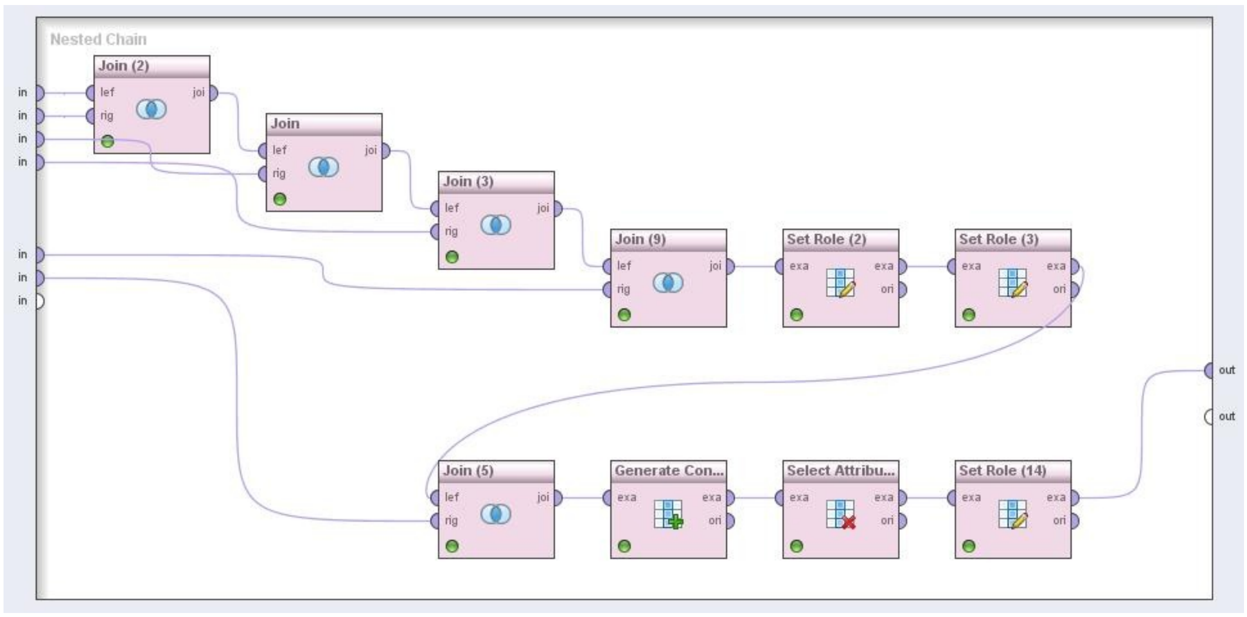
Nested Chain
Generate Con...
Select Attribu...
Numeric Filter...
Normalize (9)
Pivot (2)
Replace Missi...
Set Role (8)
Select Attribu...
X-Means (2)
Cluster-Base...
Set Role (7)
Normalize (10)
Filter Example...
Set Role (9)
Rename (2)
Select Attribu...
in
in
out
out

```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="76" name="OCS" width="90" x="648" y="660">
    <process expanded="true" height="637" width="1103">
      <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate Concatenation (8)" width="90" x="45" y="30">
        <parameter key="first_attribute" value="MACHINE"/>
        <parameter key="second_attribute" value="CREATION_DATE"/>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (11)" width="90" x="180" y="30">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attributes" value="MACHINE_CONSTANT|MACHINE_CREATION_DATE|NUMERIC_VALUE"/>
      </operator>
      <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Numeric Filter (2)" width="90" x="315" y="30">
        <parameter key="condition_class" value="no_missing_attributes"/>
        <parameter key="parameter_string" value="NUMERIC_VALUE=?"/>
      </operator>
      <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (9)" width="90" x="450" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="NUMERIC_VALUE"/>
        <parameter key="method" value="range transformation"/>
        <parameter key="min" value="1.0"/>
        <parameter key="max" value="1.000000001"/>
      </operator>
      <operator activated="true" class="pivot" compatibility="5.2.008" expanded="true" height="76" name="Pivot (2)" width="90" x="580" y="30">
        <parameter key="group_attribute" value="MACHINE_CREATION_DATE"/>
        <parameter key="index_attribute" value="MACHINE_CONSTANT"/>
        <parameter key="consider_weights" value="false"/>
        <parameter key="skip_constant_attributes" value="false"/>
      </operator>
      <operator activated="true" class="replace_missing_values" compatibility="5.2.008" expanded="true" height="94" name="Replace Missing Values (3)" width="90" x="714" y="30">
        <parameter key="attribute" value="NUMERIC_VALUE"/>
        <parameter key="default" value="zero"/>
        <list key="columns"/>
      </operator>
      <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (8)" width="90" x="849" y="30">
        <parameter key="name" value="MACHINE_CREATION_DATE"/>
        <parameter key="target_role" value="label"/>
        <list key="set_additional_roles"/>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (13)" width="90" x="983" y="30">
        <parameter key="attribute_filter_type" value="subset"/>
      <parameter key="attributes" value="CREATION_DATE"/>
      <parameter key="invert_selection" value="true"/>
    </operator>
    <operator activated="true" class="x_means" compatibility="5.2.008" expanded="true" height="76" name="X-Means (2)" width="90"
x="45" y="165"/>
    <operator activated="true" class="anomalydetection:Cluster-Based Local Outlier Factor (CBLOF)" compatibility="1.3.000"
expanded="true" height="94" name="Cluster-Based Local Outlier Factor (2)" width="90" x="179" y="165"/>
    <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (7)" width="90"
x="313" y="165">
      <parameter key="name" value="outlier"/>
      <list key="set_additional_roles">
        <parameter key="MACHINE_CREATION_DATE" value="regular"/>
        <parameter key="cluster" value="regular"/>
      </list>
    </operator>
    <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize (10)" width="90"
x="447" y="165">
      <parameter key="attribute_filter_type" value="single"/>
      <parameter key="attribute" value="outlier"/>
    </operator>
    <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (21)"
width="90" x="581" y="165">
      <parameter key="condition_class" value="attribute_value_filter"/>
      <parameter key="parameter_string" value="outlier&gt;2"/>
    </operator>
    <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (9)" width="90"
x="715" y="165">
      <parameter key="name" value="MACHINE_CREATION_DATE"/>
      <parameter key="target_role" value="id"/>
      <list key="set_additional_roles"/>
    </operator>
    <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (2)" width="90"
x="849" y="165">
      <parameter key="old_name" value="outlier"/>
      <parameter key="new_name" value="ClusterEvent"/>
      <list key="rename_additional_attributes"/>
    </operator>
    <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes
(14)" width="90" x="983" y="165">
      <parameter key="attribute_filter_type" value="subset"/>
      <parameter key="attributes" value="|ClusterEvent|MACHINE_CREATION_DATE"/>
    </operator>
    <connect from_port="in 1" to_op="Generate Concatenation (8)" to_port="example set input"/>
    <connect from_op="Generate Concatenation (8)" from_port="example set output" to_op="Select Attributes (11)" to_port="example set
input"/>
    <connect from_op="Select Attributes (11)" from_port="example set output" to_op="Numeric Filter (2)" to_port="example set input"/>
    <connect from_op="Numeric Filter (2)" from_port="example set output" to_op="Normalize (9)" to_port="example set input"/>
    <connect from_op="Normalize (9)" from_port="example set output" to_op="Pivot (2)" to_port="example set input"/>
    <connect from_op="Pivot (2)" from_port="example set output" to_op="Replace Missing Values (3)" to_port="example set input"/>
    <connect from_op="Replace Missing Values (3)" from_port="example set output" to_op="Set Role (8)" to_port="example set input"/>
    <connect from_op="Set Role (8)" from_port="example set output" to_op="Select Attributes (13)" to_port="example set input"/>
    <connect from_op="Select Attributes (13)" from_port="example set output" to_op="X-Means (2)" to_port="example set"/>
    <connect from_op="X-Means (2)" from_port="cluster model" to_op="Cluster-Based Local Outlier Factor (2)" to_port="cluster Model"/>
    <connect from_op="X-Means (2)" from_port="clustered set" to_op="Cluster-Based Local Outlier Factor (2)" to_port="clustered set"/>
    <connect from_op="Cluster-Based Local Outlier Factor (2)" from_port="clustered set" to_op="Set Role (7)" to_port="example set
input"/>
    <connect from_op="Set Role (7)" from_port="example set output" to_op="Normalize (10)" to_port="example set input"/>
    <connect from_op="Normalize (10)" from_port="example set output" to_op="Filter Examples (21)" to_port="example set input"/>
    <connect from_op="Filter Examples (21)" from_port="example set output" to_op="Set Role (9)" to_port="example set input"/>
    <connect from_op="Set Role (9)" from_port="example set output" to_op="Rename (2)" to_port="example set input"/>
    <connect from_op="Rename (2)" from_port="example set output" to_op="Select Attributes (14)" to_port="example set input"/>
    <connect from_op="Select Attributes (14)" from_port="example set output" to_port="out 1"/>
    <portSpacing port="source_in 1" spacing="0"/>
    <portSpacing port="source_in 2" spacing="0"/>
    <portSpacing port="sink_out 1" spacing="126"/>
    <portSpacing port="sink_out 2" spacing="0"/>
   </process>
  </operator>
```

9.9 Join

```
<operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join Clst Alg." width="90" x="849"
y="615">
    <parameter key="join_type" value="outer"/>
    <list key="key_attributes"/>
   </operator>

   <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (8)" width="90" x="849"
y="435">
    <parameter key="join_type" value="outer"/>
    <list key="key_attributes"/>
   </operator>
```

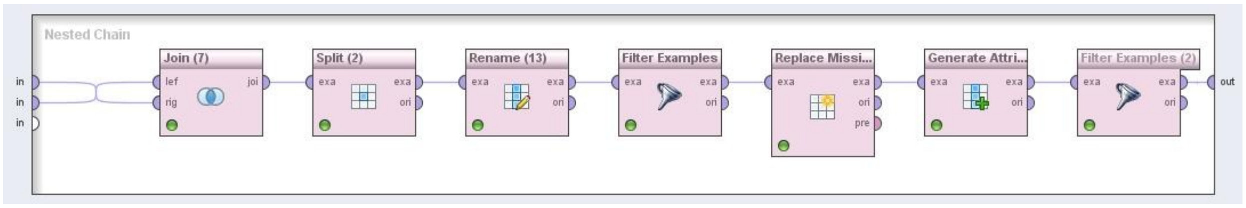
9.10 JMCL



```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="166" name="JMCL" width="90" x="983"
y="120">
    <process expanded="true" height="637" width="1213">
     <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (2)" width="90" x="45"
y="30">
      <parameter key="join_type" value="outer"/>
      <list key="key_attributes"/>
     </operator>
     <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join" width="90" x="179" y="75">
      <parameter key="join_type" value="outer"/>
      <list key="key_attributes"/>
     </operator>
     <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (3)" width="90" x="313"
y="120">
      <parameter key="join_type" value="outer"/>
      <list key="key_attributes"/>
     </operator>
     <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (9)" width="90" x="447"
y="165">
      <list key="key_attributes"/>
     </operator>
     <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (2)" width="90"
x="581" y="165">
      <parameter key="name" value="id"/>
      <list key="set_additional_roles"/>
     </operator>
     <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (3)" width="90"
x="715" y="165">
      <parameter key="name" value="MACHINE_MACHINE_CONSTANT"/>
      <parameter key="target_role" value="id"/>
```

```
          <list key="set_additional_roles"/>
        </operator>
        <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (5)" width="90" x="313"
y="345">
          <parameter key="join_type" value="left"/>
          <list key="key_attributes"/>
        </operator>
        <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate
Concatenation (2)" width="90" x="447" y="345">
          <parameter key="first_attribute" value="MACHINE"/>
          <parameter key="second_attribute" value="CREATION_DATE"/>
        </operator>
        <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes
(18)" width="90" x="581" y="345">
          <parameter key="attribute_filter_type" value="subset"/>
          <parameter key="attributes"
value="CATEGORY_CODE|CATEGORY_DESCRIPTION|CONSTANT_TYPE|JumpFreq|JumpIndv|JumpPop|MACHINE_CONSTANT|MACHINE_CREA
TION_DATE|NUMERIC_VALUE|STE_CODE|TrendPop|VALID_FROM_DATE|ValueIndv|ValuePop"/>
          <parameter key="include_special_attributes" value="true"/>
        </operator>
        <operator activated="true" class="set_role" compatibility="5.2.008" expanded="true" height="76" name="Set Role (14)" width="90"
x="715" y="345">
          <parameter key="name" value="MACHINE_CREATION_DATE"/>
          <parameter key="target_role" value="id"/>
          <list key="set_additional_roles"/>
        </operator>
        <connect from_port="in 1" to_op="Join (2)" to_port="left"/>
        <connect from_port="in 2" to_op="Join (2)" to_port="right"/>
        <connect from_port="in 3" to_op="Join" to_port="right"/>
        <connect from_port="in 4" to_op="Join (3)" to_port="right"/>
        <connect from_port="in 5" to_op="Join (9)" to_port="right"/>
        <connect from_port="in 6" to_op="Join (5)" to_port="right"/>
        <connect from_op="Join (2)" from_port="join" to_op="Join" to_port="left"/>
        <connect from_op="Join" from_port="join" to_op="Join (3)" to_port="left"/>
        <connect from_op="Join (3)" from_port="join" to_op="Join (9)" to_port="left"/>
        <connect from_op="Join (9)" from_port="join" to_op="Set Role (2)" to_port="example set input"/>
        <connect from_op="Set Role (2)" from_port="example set output" to_op="Set Role (3)" to_port="example set input"/>
        <connect from_op="Set Role (3)" from_port="example set output" to_op="Join (5)" to_port="left"/>
        <connect from_op="Join (5)" from_port="join" to_op="Generate Concatenation (2)" to_port="example set input"/>
        <connect from_op="Generate Concatenation (2)" from_port="example set output" to_op="Select Attributes (18)" to_port="example set
input"/>
        <connect from_op="Select Attributes (18)" from_port="example set output" to_op="Set Role (14)" to_port="example set input"/>
        <connect from_op="Set Role (14)" from_port="example set output" to_port="out 1"/>
        <portSpacing port="source_in 1" spacing="0"/>
        <portSpacing port="source_in 2" spacing="0"/>
        <portSpacing port="source_in 3" spacing="0"/>
        <portSpacing port="source_in 4" spacing="0"/>
        <portSpacing port="source_in 5" spacing="54"/>
        <portSpacing port="source_in 6" spacing="0"/>
        <portSpacing port="source_in 7" spacing="0"/>
        <portSpacing port="sink_out 1" spacing="216"/>
        <portSpacing port="sink_out 2" spacing="18"/>
      </process>
    </operator>
```

9.11 CTL



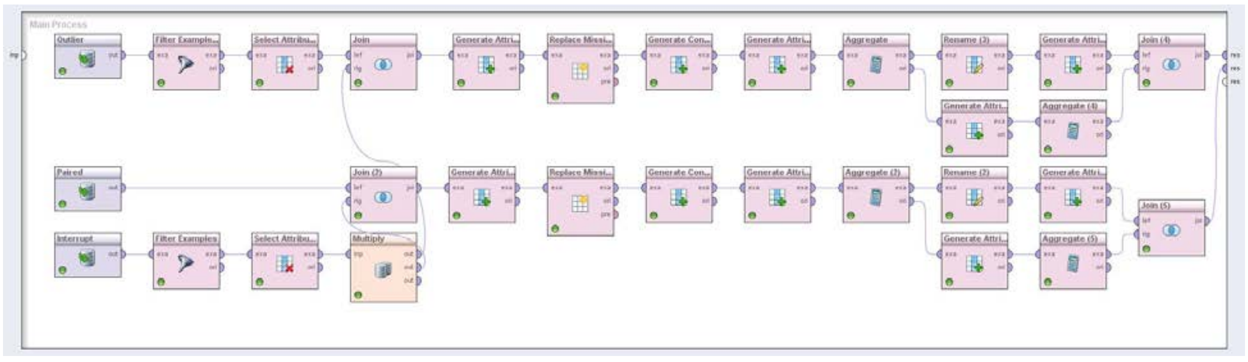
```
<operator activated="true" class="subprocess" compatibility="5.2.008" expanded="true" height="94" name="CTL" width="90" x="1184"
y="300">
    <process expanded="true" height="655" width="1069">
     <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (7)" width="90" x="112"
y="30">
       <parameter key="join_type" value="outer"/>
       <list key="key_attributes"/>
     </operator>
     <operator activated="true" class="split" compatibility="5.2.008" expanded="true" height="76" name="Split (2)" width="90" x="246"
y="30">
       <parameter key="attribute_filter_type" value="single"/>
       <parameter key="attribute" value="MACHINE_CREATION_DATE"/>
       <parameter key="include_special_attributes" value="true"/>
       <parameter key="split_pattern" value="_"/>
     </operator>
     <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (13)" width="90"
x="380" y="30">
       <parameter key="old_name" value="MACHINE_CREATION_DATE_1"/>
       <parameter key="new_name" value="MACHINE"/>
       <list key="rename_additional_attributes">
         <parameter key="MACHINE_CREATION_DATE_2" value="CREATION_DATE"/>
       </list>
     </operator>
     <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples"
width="90" x="514" y="30">
       <parameter key="condition_class" value="attribute_value_filter"/>
       <parameter key="parameter_string" value="JumpFreq &gt;= 3 || JumpFreq &lt;=-3 || ValuePop &gt;= 3 || ValuePop &lt;=-3 ||
ValueIndv &gt;= 3 || ValueIndv &lt;=-3 || JumpPop &gt;= 3 || JumpPop &lt;=-3 || JumpIndv &gt;= 3 || JumpIndv &lt;=-3 || ClusterEvent
&gt;=3 || ClusterValue &gt;=3"/>
     </operator>
     <operator activated="true" class="replace_missing_values" compatibility="5.2.008" expanded="true" height="94" name="Replace
Missing Values (4)" width="90" x="648" y="30">
       <parameter key="attribute_filter_type" value="subset"/>
       <parameter key="attribute" value="id"/>
       <parameter key="attributes" value="JumpFreq|JumpIndv|JumpPop|TrendPop|ValueIndv|ValuePop||ClusterEvent|ClusterValue"/>
       <parameter key="default" value="zero"/>
       <list key="columns"/>
     </operator>
     <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate
Attributes (2)" width="90" x="782" y="30">
       <list key="function_descriptions">
         <parameter key="SumOutlier" value="sum(abs(JumpFreq)
,abs(ValuePop),abs(ValueIndv),abs(JumpPop),abs(JumpIndv),abs(ClusterEvent),abs(ClusterValue),abs(TrendPop))"/>
       </list>
     </operator>
     <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (2)"
width="90" x="916" y="30">
       <parameter key="condition_class" value="attribute_value_filter"/>
       <parameter key="parameter_string" value="SumOutlier&gt;=5"/>
     </operator>
     <connect from_port="in 1" to_op="Join (7)" to_port="right"/>
     <connect from_port="in 2" to_op="Join (7)" to_port="left"/>
     <connect from_op="Join (7)" from_port="join" to_op="Split (2)" to_port="example set input"/>
     <connect from_op="Split (2)" from_port="example set output" to_op="Rename (13)" to_port="example set input"/>
     <connect from_op="Rename (13)" from_port="example set output" to_op="Filter Examples" to_port="example set input"/>
     <connect from_op="Filter Examples" from_port="example set output" to_op="Replace Missing Values (4)" to_port="example set
input"/>
     <connect from_op="Replace Missing Values (4)" from_port="example set output" to_op="Generate Attributes (2)" to_port="example set
input"/>
```

```
        <connect from_op="Generate Attributes (2)" from_port="example set output" to_op="Filter Examples (2)" to_port="example set input"/>
        <connect from_op="Filter Examples (2)" from_port="example set output" to_port="out 1"/>
        <portSpacing port="source_in 1" spacing="0"/>
        <portSpacing port="source_in 2" spacing="0"/>
        <portSpacing port="source_in 3" spacing="0"/>
        <portSpacing port="sink_out 1" spacing="180"/>
        <portSpacing port="sink_out 2" spacing="0"/>
      </process>
    </operator>
```

9.12 Write List

```
    <operator activated="true" class="write_excel" compatibility="5.2.008" expanded="true" height="76" name="Write List" width="90" x="1318" y="300">
      <parameter key="excel_file" value="C:\Users\paalst\Desktop\Final.xlsx"/>
      <parameter key="file_format" value="xlsx"/>
    </operator>
    <connect from_op="Retrieve" from_port="output" to_op="Generate ID" to_port="example set input"/>
    <connect from_op="Generate ID" from_port="example set output" to_op="Generate Concatenation" to_port="example set input"/>
    <connect from_op="Generate Concatenation" from_port="example set output" to_op="Multiply" to_port="input"/>
    <connect from_op="Multiply" from_port="output 1" to_op="OVP" to_port="in 1"/>
    <connect from_op="Multiply" from_port="output 2" to_op="OVI" to_port="in 1"/>
    <connect from_op="Multiply" from_port="output 3" to_op="OJP" to_port="in 1"/>
    <connect from_op="Multiply" from_port="output 4" to_op="OJI" to_port="in 1"/>
    <connect from_op="Multiply" from_port="output 5" to_op="OJFP" to_port="in 1"/>
    <connect from_op="Multiply" from_port="output 6" to_op="OMCV" to_port="in 1"/>
    <connect from_op="Multiply" from_port="output 7" to_op="OCS" to_port="in 1"/>
    <connect from_op="Multiply" from_port="output 8" to_op="OTP" to_port="in 1"/>
    <connect from_op="Multiply" from_port="output 9" to_op="JMCL" to_port="in 5"/>
    <connect from_op="OVP" from_port="out 1" to_op="JMCL" to_port="in 1"/>
    <connect from_op="OVI" from_port="out 1" to_op="JMCL" to_port="in 2"/>
    <connect from_op="OJP" from_port="out 1" to_op="JMCL" to_port="in 3"/>
    <connect from_op="OJI" from_port="out 1" to_op="JMCL" to_port="in 4"/>
    <connect from_op="OJFP" from_port="out 1" to_op="Join (8)" to_port="left"/>
    <connect from_op="OTP" from_port="out 1" to_op="Join (8)" to_port="right"/>
    <connect from_op="OMCV" from_port="out 1" to_op="Join Clst Alg." to_port="left"/>
    <connect from_op="OCS" from_port="out 1" to_op="Join Clst Alg." to_port="right"/>
    <connect from_op="Join Clst Alg." from_port="join" to_op="CTL" to_port="in 2"/>
    <connect from_op="Join (8)" from_port="join" to_op="JMCL" to_port="in 6"/>
    <connect from_op="JMCL" from_port="out 1" to_op="CTL" to_port="in 1"/>
    <connect from_op="CTL" from_port="out 1" to_op="Write List" to_port="input"/>
    <portSpacing port="source_input 1" spacing="0"/>
    <portSpacing port="sink_result 1" spacing="90"/>
  </process>
 </operator>
</process>
```

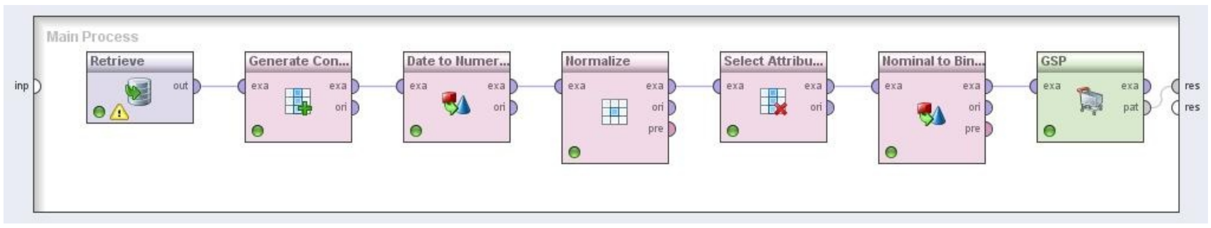
10 Appendix: Predictive Analysis



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="5.2.008">
```

```
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="5.2.008" expanded="true" name="Process">
    <process expanded="true" height="457" width="1639">
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Paired" width="90" x="45"
y="210">
        <parameter key="repository_entry" value="Paired3"/>
      </operator>
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Interrupt" width="90" x="45"
y="300">
        <parameter key="repository_entry" value="Interrupt"/>
      </operator>
      <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples"
width="90" x="179" y="300">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="INTERRUPT TYPE=Reliability || INTERRUPT TYPE=Availability"/>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (2)"
width="90" x="313" y="300">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attributes" value="MACHINE|TIMESTAMP"/>
      </operator>
      <operator activated="true" class="multiply" compatibility="5.2.008" expanded="true" height="94" name="Multiply" width="90" x="447"
y="300"/>
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Outlier" width="90" x="45"
y="30">
        <parameter key="repository_entry" value="Outlier3"/>
      </operator>
      <operator activated="true" class="filter_examples" compatibility="5.2.008" expanded="true" height="76" name="Filter Examples (2)"
width="90" x="179" y="30">
        <parameter key="condition_class" value="attribute_value_filter"/>
        <parameter key="parameter_string" value="Up/Down=Up"/>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes"
width="90" x="313" y="30">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attributes" value="MACHINE|CREATION_DATE|"/>
      </operator>
      <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join" width="90" x="447" y="30">
        <parameter key="join_type" value="left"/>
        <parameter key="use_id_attribute_as_key" value="false"/>
        <list key="key_attributes">
          <parameter key="MACHINE" value="MACHINE"/>
        </list>
      </operator>
      <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate
Attributes" width="90" x="587" y="30">
        <list key="function_descriptions">
          <parameter key="Time" value="date_diff(CREATION_DATE,TIMESTAMP)/144000000"/>
        </list>
      </operator>
      <operator activated="true" class="replace_missing_values" compatibility="5.2.008" expanded="true" height="94" name="Replace Missing
Values" width="90" x="715" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="Time"/>
        <parameter key="default" value="value"/>
        <list key="columns"/>
        <parameter key="replenishment_value" value="60"/>
      </operator>
      <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate
Concatenation" width="90" x="849" y="30">
        <parameter key="first_attribute" value="MACHINE"/>
        <parameter key="second_attribute" value="CREATION_DATE"/>
      </operator>
      <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes
(10)" width="90" x="983" y="30">
        <list key="function_descriptions">
          <parameter key="TimeDiff" value="if(Time&lt;=0,60,Time)"/>
      </list>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate" width="90" x="1117" y="30">
      <list key="aggregation_attributes">
        <parameter key="TimeDiff" value="minimum"/>
      </list>
      <parameter key="group_by_attributes" value="MACHINE_CREATION_DATE|"/>
    </operator>
    <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (3)" width="90" x="1251" y="30">
      <parameter key="old_name" value="minimum(TimeDiff)"/>
      <parameter key="new_name" value="MinDiff"/>
      <list key="rename_additional_attributes"/>
    </operator>
    <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (7)" width="90" x="1251" y="120">
      <list key="function_descriptions">
        <parameter key="Broken2" value="if(TimeDiff&lt;=2,1,0)"/>
        <parameter key="Broken7" value="if(TimeDiff&lt;=7,1,0)"/>
      </list>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (4)" width="90" x="1385" y="120">
      <list key="aggregation_attributes">
        <parameter key="Broken2" value="sum"/>
        <parameter key="Broken7" value="sum"/>
      </list>
      <parameter key="group_by_attributes" value="MACHINE_CREATION_DATE|"/>
    </operator>
    <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (4)" width="90" x="1385" y="30">
      <list key="function_descriptions">
        <parameter key="Broken2" value="if(MinDiff&lt;=2,1,0)"/>
        <parameter key="Broken7" value="if(MinDiff&lt;=7,1,0)"/>
      </list>
    </operator>
    <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (4)" width="90" x="1519" y="30">
      <parameter key="use_id_attribute_as_key" value="false"/>
      <list key="key_attributes">
        <parameter key="MACHINE_CREATION_DATE" value="MACHINE_CREATION_DATE"/>
      </list>
    </operator>
    <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (2)" width="90" x="447" y="210">
      <parameter key="join_type" value="left"/>
      <parameter key="use_id_attribute_as_key" value="false"/>
      <list key="key_attributes">
        <parameter key="MACHINE" value="MACHINE"/>
      </list>
    </operator>
    <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (3)" width="90" x="581" y="210">
      <list key="function_descriptions">
        <parameter key="Time" value="date_diff(CREATION_DATE,TIMESTAMP)/144000000"/>
      </list>
    </operator>
    <operator activated="true" class="replace_missing_values" compatibility="5.2.008" expanded="true" height="94" name="Replace Missing Values (2)" width="90" x="715" y="210">
      <parameter key="attribute_filter_type" value="single"/>
      <parameter key="attribute" value="Time"/>
      <parameter key="default" value="value"/>
      <list key="columns"/>
      <parameter key="replenishment_value" value="60"/>
    </operator>
    <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate Concatenation (2)" width="90" x="849" y="210">
      <parameter key="first_attribute" value="MACHINE"/>
      <parameter key="second_attribute" value="CREATION_DATE"/>
    </operator>
    <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (11)" width="90" x="983" y="210">
      <list key="function_descriptions">
        <parameter key="TimeDiff" value="if(Time&lt;=0,60,Time)"/>
      </list>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (2)" width="90" x="1117" y="210">
      <list key="aggregation_attributes">
        <parameter key="TimeDiff" value="minimum"/>
      </list>
      <parameter key="group_by_attributes" value="|MACHINE_CREATION_DATE"/>
    </operator>
    <operator activated="true" class="rename" compatibility="5.2.008" expanded="true" height="76" name="Rename (2)" width="90" x="1251" y="210">
      <parameter key="old_name" value="minimum(TimeDiff)"/>
      <parameter key="new_name" value="MinDiff"/>
      <list key="rename_additional_attributes"/>
    </operator>
    <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (8)" width="90" x="1251" y="300">
      <list key="function_descriptions">
        <parameter key="Broken2" value="if(TimeDiff&lt;=2,1,0)"/>
        <parameter key="Broken7" value="if(TimeDiff&lt;=7,1,0)"/>
      </list>
    </operator>
    <operator activated="true" class="aggregate" compatibility="5.2.008" expanded="true" height="76" name="Aggregate (5)" width="90" x="1385" y="300">
      <list key="aggregation_attributes">
        <parameter key="Broken2" value="sum"/>
        <parameter key="Broken7" value="sum"/>
      </list>
      <parameter key="group_by_attributes" value="|MACHINE_CREATION_DATE"/>
    </operator>
    <operator activated="true" class="generate_attributes" compatibility="5.2.008" expanded="true" height="76" name="Generate Attributes (6)" width="90" x="1385" y="210">
      <list key="function_descriptions">
        <parameter key="Broken2" value="if(MinDiff&lt;=2,1,0)"/>
        <parameter key="Broken7" value="if(MinDiff&lt;=7,1,0)"/>
      </list>
    </operator>
    <operator activated="true" class="join" compatibility="5.2.008" expanded="true" height="76" name="Join (5)" width="90" x="1519" y="255">
      <parameter key="use_id_attribute_as_key" value="false"/>
      <list key="key_attributes">
        <parameter key="MACHINE_CREATION_DATE" value="MACHINE_CREATION_DATE"/>
      </list>
    </operator>
    <connect from_op="Paired" from_port="output" to_op="Join (2)" to_port="left"/>
    <connect from_op="Interrupt" from_port="output" to_op="Filter Examples" to_port="example set input"/>
    <connect from_op="Filter Examples" from_port="example set output" to_op="Select Attributes (2)" to_port="example set input"/>
    <connect from_op="Select Attributes (2)" from_port="example set output" to_op="Multiply" to_port="input"/>
    <connect from_op="Multiply" from_port="output 1" to_op="Join" to_port="right"/>
    <connect from_op="Multiply" from_port="output 2" to_op="Join (2)" to_port="right"/>
    <connect from_op="Outlier" from_port="output" to_op="Filter Examples (2)" to_port="example set input"/>
    <connect from_op="Filter Examples (2)" from_port="example set output" to_op="Select Attributes" to_port="example set input"/>
    <connect from_op="Select Attributes" from_port="example set output" to_op="Join" to_port="left"/>
    <connect from_op="Join" from_port="join" to_op="Generate Attributes" to_port="example set input"/>
    <connect from_op="Generate Attributes" from_port="example set output" to_op="Replace Missing Values" to_port="example set input"/>
    <connect from_op="Replace Missing Values" from_port="example set output" to_op="Generate Concatenation" to_port="example set input"/>
    <connect from_op="Generate Concatenation" from_port="example set output" to_op="Generate Attributes (10)" to_port="example set input"/>
    <connect from_op="Generate Attributes (10)" from_port="example set output" to_op="Aggregate" to_port="example set input"/>
    <connect from_op="Aggregate" from_port="example set output" to_op="Rename (3)" to_port="example set input"/>
    <connect from_op="Aggregate" from_port="original" to_op="Generate Attributes (7)" to_port="example set input"/>
    <connect from_op="Rename (3)" from_port="example set output" to_op="Generate Attributes (4)" to_port="example set input"/>
    <connect from_op="Generate Attributes (7)" from_port="example set output" to_op="Aggregate (4)" to_port="example set input"/>
    <connect from_op="Aggregate (4)" from_port="example set output" to_op="Join (4)" to_port="right"/>
    <connect from_op="Generate Attributes (4)" from_port="example set output" to_op="Join (4)" to_port="left"/>
    <connect from_op="Join (4)" from_port="join" to_port="result 1"/>
    <connect from_op="Join (2)" from_port="join" to_op="Generate Attributes (3)" to_port="example set input"/>
    <connect from_op="Generate Attributes (3)" from_port="example set output" to_op="Replace Missing Values (2)" to_port="example set input"/>
    <connect from_op="Replace Missing Values (2)" from_port="example set output" to_op="Generate Concatenation (2)" to_port="example set input"/>
    <connect from_op="Generate Concatenation (2)" from_port="example set output" to_op="Generate Attributes (11)" to_port="example set input"/>
    <connect from_op="Generate Attributes (11)" from_port="example set output" to_op="Aggregate (2)" to_port="example set input"/>
    <connect from_op="Aggregate (2)" from_port="example set output" to_op="Rename (2)" to_port="example set input"/>
    <connect from_op="Aggregate (2)" from_port="original" to_op="Generate Attributes (8)" to_port="example set input"/>
    <connect from_op="Rename (2)" from_port="example set output" to_op="Generate Attributes (6)" to_port="example set input"/>
    <connect from_op="Generate Attributes (8)" from_port="example set output" to_op="Aggregate (5)" to_port="example set input"/>
    <connect from_op="Aggregate (5)" from_port="example set output" to_op="Join (5)" to_port="right"/>
    <connect from_op="Generate Attributes (6)" from_port="example set output" to_op="Join (5)" to_port="left"/>
    <connect from_op="Join (5)" from_port="join" to_port="result 2"/>
    <portSpacing port="source_input 1" spacing="0"/>
    <portSpacing port="sink_result 1" spacing="0"/>
    <portSpacing port="sink_result 2" spacing="0"/>
    <portSpacing port="sink_result 3" spacing="0"/>
   </process>
  </operator>
</process>
```

11 Appendix: Pattern Recognition



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<process version="5.2.008">
  <context>
    <input/>
    <output/>
    <macros/>
  </context>
  <operator activated="true" class="process" compatibility="5.2.008" expanded="true" name="Process">
    <process expanded="true" height="456" width="969">
      <operator activated="true" class="retrieve" compatibility="5.2.008" expanded="true" height="60" name="Retrieve" width="90" x="45" y="30">
        <parameter key="repository_entry" value="//DB/asml_pam/Example Sets/machine_constants2"/>
      </operator>
      <operator activated="true" class="generate_concatenation" compatibility="5.2.008" expanded="true" height="76" name="Generate Concatenation (2)" width="90" x="179" y="30">
        <parameter key="first_attribute" value="MACHINE"/>
        <parameter key="second_attribute" value="MACHINE_CONSTANT"/>
      </operator>
      <operator activated="true" class="date_to_numerical" compatibility="5.2.008" expanded="true" height="76" name="Date to Numerical" width="90" x="313" y="30">
        <parameter key="attribute_name" value="CREATION_DATE"/>
        <parameter key="time_unit" value="second"/>
        <parameter key="millisecond_relative_to" value="epoch"/>
        <parameter key="second_relative_to" value="epoch"/>
        <parameter key="minute_relative_to" value="epoch"/>
        <parameter key="day_relative_to" value="epoch"/>
        <parameter key="month_relative_to" value="epoch"/>
      </operator>
      <operator activated="true" class="normalize" compatibility="5.2.008" expanded="true" height="94" name="Normalize" width="90" x="447" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="CREATION_DATE"/>
        <parameter key="method" value="range transformation"/>
        <parameter key="max" value="43.0"/>
      </operator>
      <operator activated="true" class="select_attributes" compatibility="5.2.008" expanded="true" height="76" name="Select Attributes (5)" width="90" x="581" y="30">
        <parameter key="attribute_filter_type" value="subset"/>
        <parameter key="attributes" value="|AREA|MACHINE_MACHINE_CONSTANT|CREATION_DATE"/>
      </operator>
      <operator activated="true" class="nominal_to_binominal" compatibility="5.2.008" expanded="true" height="94" name="Nominal to Binominal" width="90" x="715" y="30">
        <parameter key="attribute_filter_type" value="single"/>
        <parameter key="attribute" value="AREA"/>
      </operator>
      <operator activated="true" class="generalized_sequential_patterns" compatibility="5.2.008" expanded="true" height="76" name="GSP" width="90" x="849" y="30">
        <parameter key="customer_id" value="MACHINE_MACHINE_CONSTANT"/>
        <parameter key="time_attribute" value="CREATION_DATE"/>
        <parameter key="min_support" value="0.05"/>
        <parameter key="window_size" value="7.0"/>
        <parameter key="max_gap" value="0.0"/>
        <parameter key="min_gap" value="0.0"/>
        <parameter key="positive_value" value="true"/>
      </operator>
      <connect from_op="Retrieve" from_port="output" to_op="Generate Concatenation (2)" to_port="example set input"/>
      <connect from_op="Generate Concatenation (2)" from_port="example set output" to_op="Date to Numerical" to_port="example set input"/>
      <connect from_op="Date to Numerical" from_port="example set output" to_op="Normalize" to_port="example set input"/>
      <connect from_op="Normalize" from_port="example set output" to_op="Select Attributes (5)" to_port="example set input"/>
      <connect from_op="Select Attributes (5)" from_port="example set output" to_op="Nominal to Binominal" to_port="example set input"/>
      <connect from_op="Nominal to Binominal" from_port="example set output" to_op="GSP" to_port="example set"/>
      <connect from_op="GSP" from_port="patterns" to_port="result 1"/>
      <portSpacing port="source_input 1" spacing="0"/>
      <portSpacing port="sink_result 1" spacing="0"/>
      <portSpacing port="sink_result 2" spacing="0"/>
    </process>
  </operator>
</process>
```