CODEN:LUTEDX/(TEIE-5403)/1-59(2018)

Industrial Electrical Engineering and Automation

Monitoring of a Veneer Lathe Knife by the use of an Industrial Internet of Things-Platform

Nils Lindqvist

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Monitoring of a Veneer Lathe Knife by the use of an Industrial Internet of Things-Platform

by Nils Lindqvist

Supervisor: Gunnar Lindstedt

Assisting supervisors: Jens Molin and Ulf Yngvesson

Examinator: Ulf Jeppsson

LUND INSTITUTE OF TECHNOLOGY
Lund University
Division of Industrial Electrical Engineering and Automation

NOVOTEK

Abstract

The number of devices connected to the Internet grows constantly. This information entity has been labeled the Internet of Things (IoT). One important aspect of this is the industrial applications, sometimes labeled the Industrial Internet of Things (IIoT). Collecting and analyzing the massive amounts of data that industry generates will only become more and more important as technology and the need for efficiency increase.

Novotek is a company with long and extensive experience of industrial IT and automation. Together with their customer Quant Service they are launching a project for predictive maintenance. This aims to monitor several different industrial sites using an industrial platform and the IIoT framework. The monitoring will allow for tracking of machine status and maintenance needs from both near and afar.

One of the sites for this project is a veneer production line for composite wood products. As a part of the monitoring and predictive maintenance project, this report looks at the possibility of using the ThingWorx IIoT platform's analytics functionality to determine the need for maintenance of the cutting knife on a veneer lathe. The goal is to look at its uses for monitoring and predictive maintenance for this particular case but also as a general method. The process for this will be twofold. Since the project uses the IIoT framework one part is how to collect the data from the site and then passing it through the platform and to the analytics program. The second part is the machine learning and statistical methods and algorithms used to analyze the data for predictions. For benchmarking, it will be compared to another analytics product.

The results of the project are not conclusive concerning the knife predictions. Development of the measurement setup is needed. The IIoT platform does however show potential in being used for the intended purpose.

Keywords: Process Monitoring, Industrial Internet of Things, Automation, Machine Learning, ThingWorx, Industrial IT, Wood Veneer Production.

Acknowledgements

Firstly, I would like to thank Novotek for giving me the opportunity to do this project. Without your dedication and contribution this study would not have been possible. A special thank you to Jens Molin and Ulf Yngvesson who supervised me. Your enthusiasm and support is what has made it so engaging to work with.

Additionally, I would like to thank Quant Service and Vikon for providing process access, data and information. Especially Juha Ketolainen, for all his answers concerning the veneer lathe and guiding me at the production site, and Glenn Strängberg, for his expertise and knowledge on the measuring process.

I would also like to thank Gunnar Lindstedt and Ulf Jeppsson at Lund University. Not only for their involvement in this project, but also for inspiring me in previous courses and sparking my ideas.

Lastly, I would like to thank my family for all their support.

Malmö, 28 March 2018

Nils Lindqvist

Acronyms

IoT – Internet of Things. Framework for connecting data generating devices via the Internet.

IIoT – Industrial IoT. The industrial applications of IoT

LVL – Laminated Veneer Lumber. A composite wood construction material.

FFT – Fast Fourier Transform. A signal processing method.

API – Application Programing Interface. A set of subroutine definitions, protocols, and tools for building application software.

ROC – Receiver Operating Characteristic. A measure of how well a machine learning algorithm can predict true positive outcomes for a binary goal.

MCC – Matthews Correlation Coefficient. A measure of how well a machine learning algorithm can predict the outcome of binary goal.

RMSE – Root Mean Square Error. A measure of the difference between the predicted and actual values of the target of a predictive model.

json – JavaScript Object Notation. A platform independent data format used in software applications. Originally for JavaScript

xml – Extensible Markup Language. A platform independent data format used in software applications.

csv – Comma separated value. File format for storing data in rows and columns.

Data parameter acronyms

TOTG – Total vibration acceleration.

RESG – Stochastic part of TOTG.

PERG – Periodic and stationary part of TOTG.

TOTV – Total vibration velocity.

RESV – Stochastic part of TOTV.

PERV – Periodic and stationary part of TOTV.

SPI – One quartile of the noise friction spectrum. Indexed 1 through 4.

Table of Contents

1 Introduction

1.1 Background

The number of connected, data generating devices is growing constantly. These do not only connect more people, but also industrial machines to each other. This interconnectedness is believed to pave the way for smart factories and production systems that can make decisions for themselves. Add to this that almost half of all jobs today are predicted to disappear due to automation and it is not hard to imagine that another industrial revolution, Industry 4.0, is coming (Marr 2016).

This rapid increase of data generating automated industry has created a need for both automated and remote process monitoring. Novotek has for over 30 years been a leading supplier of products and solutions to digitalize and streamline the manufacturing industry. Together with a customer that delivers third-part-maintenance, Novotek is now realizing a pilot project with a new platform for Industrial Internet of Things (IIoT). A part of this is to further develop the predictive maintenance of advanced processing machines.

One of the sites that receive third-part-maintenance is a plywood and composite wood manufacturing plant. The knife that peels logs into veneer needs continuous maintenance in the form of sharpening and replacement several times per day. This project aims to evaluate if the current system for this can be optimized, or perhaps even be replaced, by a more efficient method.

1.2 Research questions

The aim of the project is two-fold. One is to investigate whether it is possible to detect when the peeling knife of a veneer lathe is dull by analyzing overall vibrations in the lathe machine. The second part is to evaluate generic tools for connecting industrial data producing devices and performing analytics on the data they generate. In other words, to evaluate an IIoT platform as a tool for process monitoring.

The base requirement is to identify when the knife needs maintenance and notify the operator. To accomplish this, two things must be found in the data:

A. When is the knife dull?
B. What in the data signifies when the knife is starting to get dull?

When the method is able to make predictions for the most basic maintenance suggestion, the next step is to optimize it. It should be able to monitor the data and advice on the optimal point for exchange or sharpening of the knife. This means finding a point before veneer quality is compromised but without halting the process too frequently.

1.3 Delimitations

This is not a true IoT solution or implementation. This is a study focusing on if the IoT framework and an IIoT platform can be used for Process Monitoring purposes. So, while the platform is used for passing data, the Analytics part of the platform will be the main focus for evaluation.

Since the algorithms are already implemented in the analysis programs there will only be a short discussion on the algorithms used for the project. The intention is to give the reader some idea of

how the machine learning and statistical methods work. For a comprehensive understanding, secondary sources are recommended.

There will only be a short discussion on wood working and vibration sensors. The goal is not to develop a monitoring method specifically for this process, but to investigate if an analysis tool of the type used in this study can be applied to an arbitrary problem.

Documenting the aspects of the tools used is a large undertaking. Therefore, the description of the program used for comparison, called CSense, will be less detailed than the description of the ThingWorx platform, which is the main focus.

1.4 Copyright and trademarks of companies and their products

2 Methodology

The methods used for predictions and monitoring can be divided into two categories. One is statistical and machine learning algorithms implemented in the analytics program. These will be used to analyze the data for patterns in order to make a predictive model. The other part is programmatic methods used for data communication and categorization inside of the ThingWorx IIoT platform and the analytics program.

To have some basis for comparison, another analytics program called CSense will also perform analysis on the data. The two analytics programs are somewhat different when it comes to purpose and implementation. Nevertheless, they are both employed for predictive analytics and this will establish a baseline for comparison.

The approach to the task is to evaluate the methods for process monitoring analysis by using the veneer knife as a case study. The methods are to be:

- Documented concerning general functionality.
- Described concerning how they are applied to the task at hand.
- Evaluated. Both for the case of the veneer knife but also as a general method.
- Supplied with recommendations concerning which methods and tools are suitable for predictive maintenance.

Chapters 3 through 7 deal with explaining the veneer machine, the sensors, the software used and the machine learning and statistical methods that are employed for analysis. The content of these chapters deal primarily with methodology and theory. Results, discussion and conclusion is covered in chapters 8 through 10.

3 Veneer machine and sensors

The information pertaining to the lathe, the veneer production line and the sensors all originate from talking to the engineers and operators involved in running the machine and performing the monitoring of the bearings and other machine parts. Information was also gathered during a visit to the production site. All material in this chapter originates from these correspondences and observations, unless otherwise specifically noted.

3.1 Production line

The veneer machine is part of a production line that produces Laminated Veneer Lumber (LVL), which is a type of composite wood product for all types of construction and which is made from several sheets of veneer (Canadian Wood Council 2016).

The majority, 97 %, of the log material is spruce with the remaining 3 % being pine. The type of wood affects the process somewhat, but it is more a question of the logs being hardwood or softwood. Spruce and pine are sufficiently similar and the change of material does not affect the process very much for this particular case.

The process is quite advanced. First, the logs, which have been pre-peeled of bark, are soaked in warm water for 8 hours to achieve the right temperature and hardness. Then they are conveyed to a queue for the lathe. Before entering the veneer lathe, the logs are scanned to determine diameter, needed round-up and how to cut to get the most amount of veneer out of them. After that they are inserted into the lathe that peels the logs into a long mat of veneer. This length of veneer is scanned for optimal cutting points for the “guillotine”, which cuts it into sheets of equal length. The scan also determines the moisture level and quality of the veneer and sorts the sheets accordingly. These piles are then transported by truck to another line where they are dried and subsequently glued into LVL.

Once the logs are put on the conveyer all the steps in the peeling process are automated, with an operator overseeing the process from a control booth above the lathe.

3.2 Lathe

The peeling cycle in the lathe starts as soon as the log has been scanned and the residual of the previous log has been removed. Once inserted, the log is held in place at the ends by two spiked chucks which are inserted with force into the log, turning it into a rotating axis. The knife consists of a two-part blade, which is fixed to a separate frame and is controlled by a servomotor. The log rotates in a fixed position and the knife is moved towards the log to enable the peeling. The peeling cycle can be divided into several states. Apart from insertion of a new log and emptying of the remainder of the old log, there is a round-up operation and a peeling operation. Round-up is when the log is peeled in a way to make it as uniformly cylindrical as possible. This allows for better veneer and the uniform shape makes the veneer calculations easier. After round-up is done the peeling starts. This peels the log into a long mat of veneer that exits the lathe on the other side. Once the leftover log is removed the cycle starts over. A complete cycle for one log takes about 12 seconds.

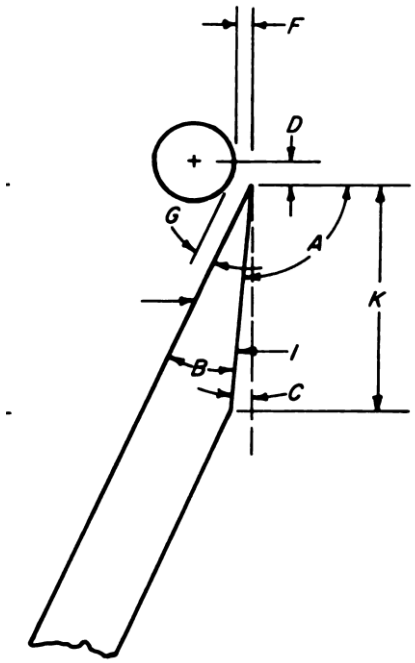

Figure 3.2.1: Lathe Knife Cutting Parameters with the letters indicating different gaps and angles. Important measures are clearance (or flank) angle, C, and knife gap, F.

While the process is automated when it is up and running, any problems in the peeling as well as setting the parameters for the lathe is handled manually by the operator. The parameters for the cutting are primarily: (1) the rotational speed of the lathe, which is specified in veneer meters produced per minute for both round-up and peeling (usually the same speed), (2) the clearance-angle (also referred to as the flank-angle in wood cutting literature (Astakhov 2010 p 68)) and its off-set, and (3) the knife-gap and its counter-weight pressure bar. The clearance-angle is the angle formed between the knife's cutting-edge plane and the vertical tangent of the log at the cutting point (Astakhov 2010 p 68), C in figure 3.2.1 (Lutz 1977 p 48). The clearance-angle is central to re-sharpening and must be adjusted accordingly (Astakhov 2010 p 63). When the blade has been re-sharpened or exchanged, the clearance-angle and offset must be reduced, since the blade is more brittle. For the same reason, the speed is also lowered when the knife has been sharpened or exchanged. Knife-gap is a measure of the horizontal distance between the knife tip and the pressure bar (Lutz 1977 p 63), F in figure 3.2.1. The size of the gap varies with the veneer thickness, and to a certain extent, the type of wood that is being peeled. A common measure is to set the gap to 80 % of the veneer thickness, though tests have shown that a smaller gap might be better (Lutz 1977 p 64), which is also the method used on the lathe in question for this report. The pressure bar is set to a counter-pressure against the veneer passing against the bar and out of the machine. All of these parameters are set by the operator, partly by routines based on tests and process experience but also according to the personal experience of the operator in question. This speaks to the craftsman-like nature of the peeling process, which is controlled manually to a higher degree than the log pre-processing and the veneer post-processing. Temperature is also a factor (Lutz 1977 p 65) and if the wood is to cold or to hot it can cause knife damage, veneer damage or production stops. This is part of the reason for why the logs are pretreated in hot water.

3.2.1 Production issues and maintenance

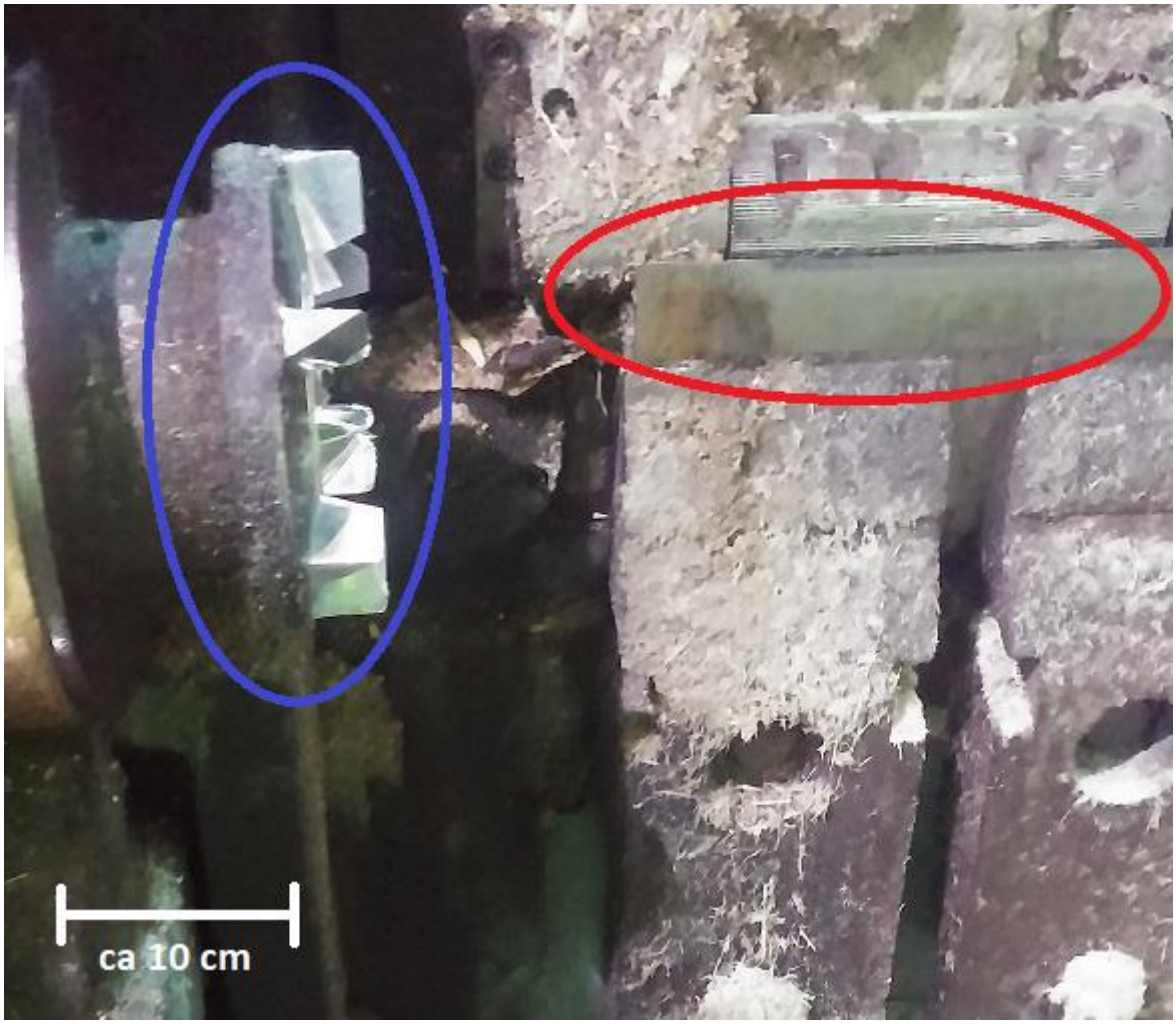

Figure 3.2.2 Left chuck, marked with blue, and part of the knife, marked with red, after cleaning for sharpening.

When there is a problem with the process the operator must notice this visually or by sound. Common problems are either spin-out or some issue with the knife. Spin-out is when the chucks (seen in figure 3.2.2) of the lathe start to dig into the log instead of rotating it. The operator must notice this by the lack of produced veneer and that the log is not rotating in the machine. Since the knife arm is still being pressed against the static log, it is important that the operator notices this quickly, so that the knife is not damaged. A log that has suffered spin-out is forfeit and must be discarded without yielding any veneer. From a data point of view, the problem with spin-out is that it is still recorded as produced veneer. The electrical motor is still running and produced distance is calculated from the rotation of the chucks and not the actual amount leaving the machine. When it comes to problems with the knife, this is primarily due to it hitting nails or rocks lodged in the log. These will chip the blade and the operator must notice this in the produced veneer that leaves the machine. For this purpose, there is an illuminated inspection point that is visible from the operator booth, seen in figure 3.2.4. From a data point of view, these events cannot be foreseen and they need to be discounted in the predictions concerning whether the knife is sharp or dull.

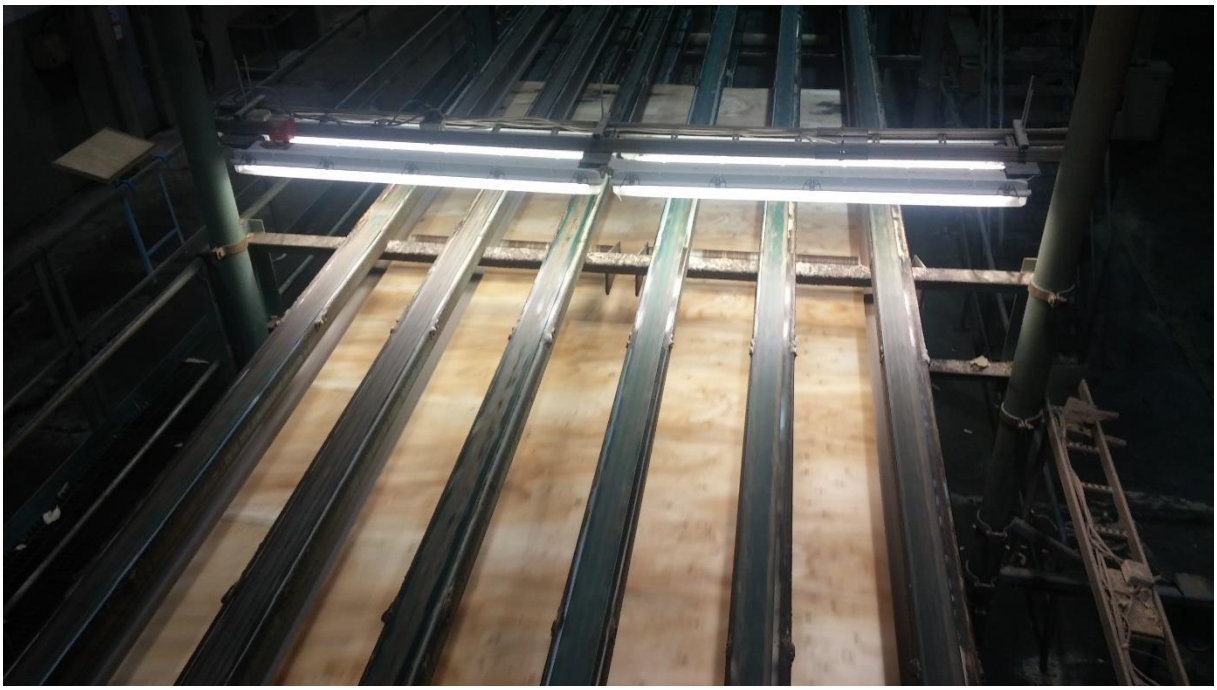

Figure 3.2.3: Inspection point for newly produced veneer.

The maintenance of the knife is done by set intervals of produced veneer length. When the lathe has produced 10’000 meters of veneer, it is sharpened manually by the operator. It is up to the individual operator to decide whether it has been sufficiently sharpened. After an additional 5000 meters, meaning a total produced length of 15’000 meters of veneer, the blades are exchanged for a new pair. The spent blades are sent off-site to be properly resharpened.

All production stops are logged in an application connected to the lathe. Times for starts and stops are logged automatically, as well as the duration of the stop. The operator enters the reason for the stop from several options and also enters the produced veneer length manually, in the cases where the knife has been sharpened or exchanged.

3.3 Sensors

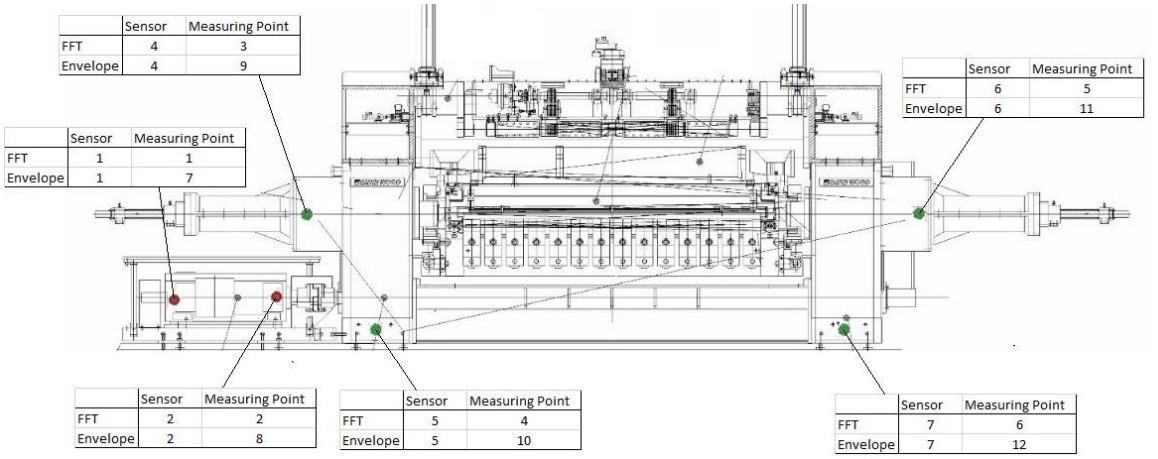

Figure 3.3.1: Schematic showing the lathe from the side where the logs enter the machine. Measuring points and sensor numbers are indicated in the boxes with the signal processing method stated for each measuring point.

The lathe is equipped with six vibration velocity sensors, seen on the lathe schematic in figure 3.3.1, and one inductive sensor that measures rotations per minute (RPM) of the electric motor. There are also several vibration velocity sensors on the hydraulic pumps for the lathe, but data from these will not be utilized for predictions concerning knife sharpness. The sensors are developed by Vikon, which is also the company who has implemented the measurement solution. There are two forms of vibration sensors, one for horizontal vibration and one for vertical vibration. The sensors need to be mounted with these directions in mind. Some specifications for them can be seen in table 3.3.1. These apply to both horizontal and vertical vibration velocity sensors.

Sensitivity	29 mV/mm/s
Frequency Range	4.5 - 2000 Hz
Accuracy	$\pm$5%
Temperature Range	-40 to +100 °C

Table 3.3.1: Specifications for the vibration velocity sensors.

The sensor measuring RPM was originally designated as sensor 3, hence the gap in sensor numbering. This gap is eliminated in the data files so, for example, measuring point 3 refers to sensor 4. This can be seen in figure 3.3.1. Sensors 1 and 2 are located on the electric motor, as seen in figure 3.3.2. Sensor 4 is located behind the housing for the bearing of the left chuck, seen in figure 3.3.3. Sensor 5 is at the left side bottom of the frame, near the main axle and on the back side, where the veneer leaves the machine. It can be viewed in figure 3.3.5. Sensors 6 and 7 are on the opposite, right side of the lathe with sensor 6 placed on the right bearing housing, corresponding to where sensor 4 is placed, and sensor 7 is placed on the back side of the frame, similarly to where sensor 5 is placed. These are seen in figure 3.3.5. The sensors' positions are primarily chosen to monitor the bearings of the machine and detect early signs of wear and damage.

3.3.1 Sensor measurements

The measurements are gathered by going through all the sensors, one by one, at set time intervals. This is done at frequency ranges 1000 Hz and 2000 Hz for the lathe sensors when processed with Fast Fourier Transform (FFT) and at 2000 Hz when handled with envelope techniques. Measuring point index 1 to 6 is for FFT and 7-12 is for data using envelope technique. FFT is an algorithm commonly used for signal processing (NTI Audio 2018). Envelope techniques are used for vibration monitoring to detect damages in bearings (Howieson 2003). Initially, both the 1000 Hz and 2000 Hz data were used. Later in the project, Vikon stopped recording data at 1000 Hz range and therefore only the 2000 Hz range is used in the data analysis. Every measurement operation results in a collection of parameter values for that sensor with a timestamp, a batch. This batch is part of a larger batch, which contains all of the measurement operations. For example, the measurements done on sensor 2 is not taken simultaneously as sensor 4. This means they will have different timestamps even if they are part of the same main batch (measurement cycle through all the sensors).

The data from the sensors is written to an xml file, a platform independent data format (W3Schools 2018a), on site. For each sensor measurement the program logs a timestamp (date and time of the measurement), the motor RPM, the bandwidth and the frequency range. The data recorded in the

xml file are the 50 frequencies with the highest amplitudes in the vibration spectrum for that sample, with amplitude given in velocity (mm/s), and 13 so called Q-parameters. The Q-parameters are divided into three categories: (1) rotational parameters, which are a type of aggregate vibrational measurements, (2) shock and impact parameters and (3) friction parameters. The rotational parameters measure total acceleration and velocity, calculated as the root mean square (rms-value). Acceleration is sensitive to high frequency vibration and velocity to lower frequencies. They are also divided between acceleration and velocity that has a regular pattern, periodic or stationary, and stochastic vibrations with no discernable pattern. The shock and impact parameters measure peak amplitude in relation to rms of the velocity, as well as some other probability distribution measures in the form of kurtosis (how probable extreme outcomes are) and skewness (the asymmetry of the probability distribution of a random variable) of the signal. The friction parameters measure vibration energy noise intensity in the spectra, divided into four parts; low frequency noise, medium low frequency noise, medium high frequency noise and high frequency noise.

A sample of a complete round of measurements is sent at even intervals for use in the monitoring of the machine parts, but the data is also stored locally at the site, reaching back 3 months, and can be accessed remotely. Initially the logging interval was 10 minutes. This was later changed to 3 minutes in order to generate more usable data.

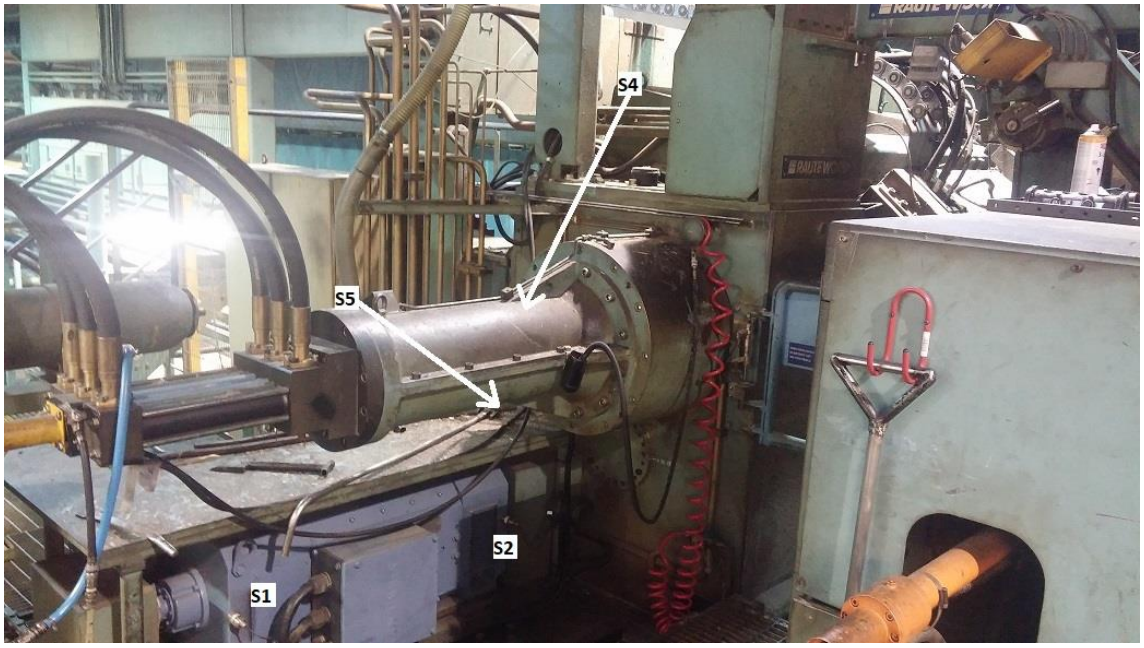

Figure 3.3.2: View of the motor sensors and indication where the left chunk and left main shaft sensors are located.

Figure 3.3.3: Detail of the left chunk sensor.

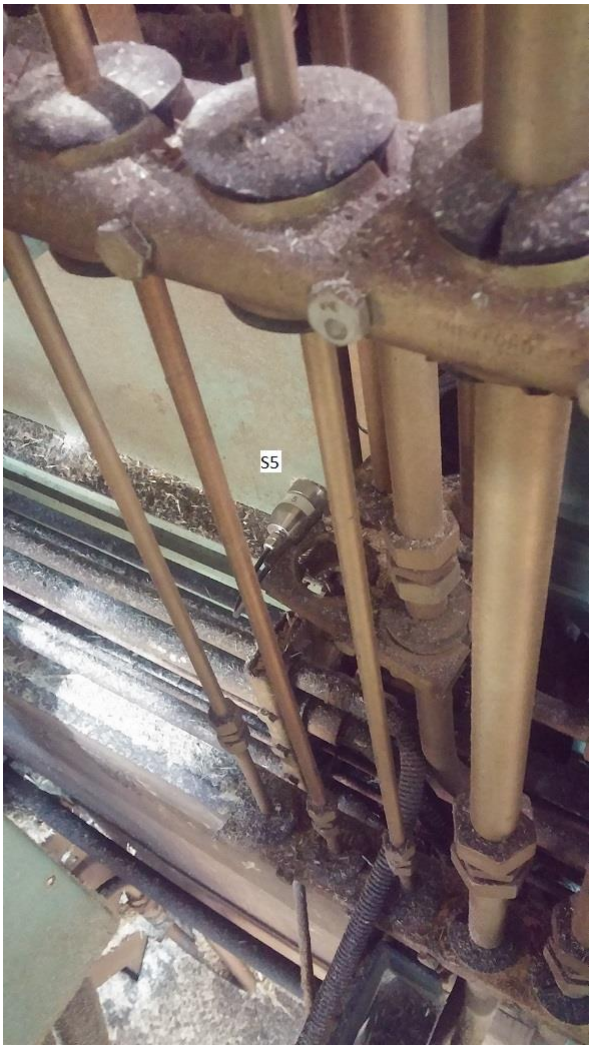

Figure 3.3.4: Detail of left main shaft sensor.

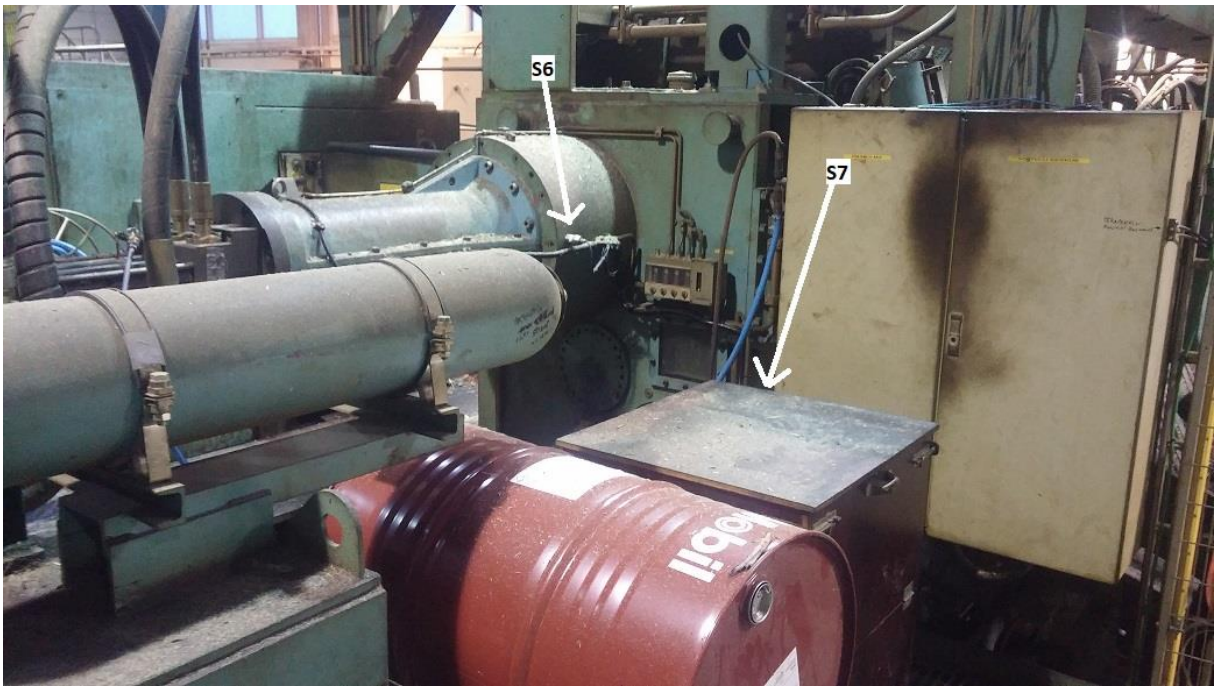

Figure 3.3.5: View indicating where the right chunk and right main shaft sensors are located.

3.4 Hypothesis on how to detect dullness of veneer knife

As can easily be seen, the sensors are not placed particularly close to the knife. Therefore, it becomes quite a challenge to predict any changes in the state of the knife. The working hypothesis is that a knife that is duller will produce more vibrations, not just in the knife but throughout the entire lathe.

While there are no sensors mounted on the knife or the knife carrier, there are sensors measuring vibration velocity on parts related to the cutting. The pair of sensors on the outside of the frames on the housing for the chucks, sensors 4 and 6 (measuring points 3 and 5), might possibly detect vibrations due to dullness. Similarly, the pair of sensors on the main shaft, sensors 5 and 7 (measuring points 4 and 6), might also be affected by vibrations due to dullness. If it is possible to detect vibration patterns giving frequency or Q-parameter shifts in some of these sensors when the knife is dull, then it will be possible to create an alert system for when the knife needs sharpening.

4 IIoT and ThingWorx

4.1 Industrial Internet of Things

The Internet of Things (IoT) refers to the connection of objects, physical and virtual, to the internet. This goes beyond more conventional connected devices, such as desktop computers and smartphones, and includes almost any data generating and connectable device, referred to as a Thing. Industrial Internet of Things (IIoT) is the application on industry. IIoT uses this connectivity on sensor data, machine-to-machine communication and automation technology, to optimize industrial processes. The rationale being that smart machines are better than humans at evaluating and communicating their situation and status (TechTarget 2018).

Another justification for the IoT approach is to better enable responses to incoming information in real time. Traditionally, data is often first collected, then stored in a data base and then queried and analyzed with some business logic. This obviously creates a delay in the information. An IoT-platform therefore takes an event-based approach to both architecture and business logic (PTC 2018a).

There are two main parts of the IoT, the “Edge” and the “Cloud” (Biron & Follett 2016 p 3-4). The Edge is where the real Things are, the entities creating and sending the data (Biron & Follett 2016 p 21). The Cloud is where the IoT solutions application is located. The IoT does not look at things locally and globally, from the local perspective. Instead it places its perspective in the Cloud and from there reaches out towards the Edge to gather information and return suggestions and commands. However, the model for the application is usually constructed from the Edge and up to the Cloud, since the Edge is where the need for the solution arises (Biron & Follett 2016 p 39).

4.2 The ThingWorx Platform and the ThingWorx Model

ThingWorx is an industrial platform adapted for the IIoT and designed for managing and organizing industry and business over all scales. The software is developed and supported by the company PTC (PTC 2017a). It builds on the Apache Tomcat Web server, which is an open-source implementation of the Java Servlet and other Java web server communication functionality (Apache Software Foundation 2018).

ThingWorx uses what PTC calls model-based application development. The purpose of this is to create reusable building blocks and avoid time consuming debugging and updating of code. The ThingWorx Model provides a semantic layer for the Things and data inside an application. This is done by creating your solution with a Thing-centric mind frame, making all parts of the industrial process into entities that are connected to a varying degree. The structure inside ThingWorx is similar to object-oriented programming and it has equivalents to programming concepts like inheritance (PTC 2018a).

Instead of writing data base queries, an end user should be able to ask questions of specific Things, pertaining to properties and location of a said Thing at a certain time or time period. ThingWorx uses RESTful web service calls to communicate this between the Thing-representations in the platform and the actual Things (PTC 2018a).

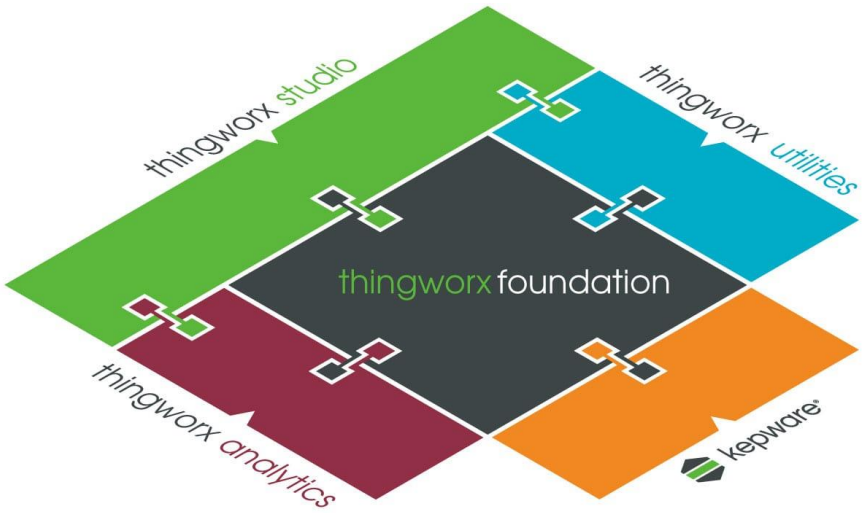

Figure 4.2.1: The ThingWorx platform and its components.

The ThingWorx platform is divided into several components, seen in figure 4.2.1 (PTC 2017a), with different applications in mind. They are all tied together with ThingWorx Foundation, which also contains the core functionality. ThingWorx Analytics is the extension which focuses on analyzing data (PTC 2018b). There are also extensions for various other ends including, ThingWorx Studio for Augmented Reality functionality, ThingWorx Utilities which provides extended device management and monitoring of connected products and ThingWorx Industrial Connectivity which provides interoperability for automation devices and software applications and is powered by the Kepware OPC-platform, another PTC product (PTC 2017a). In this project the use of the ThingWorx platform will be limited to Foundation, Analytics and some of the extensions described below.

4.3 ThingWorx Foundation

To create and modify applications, ThingWorx uses a graphical interface, called the Composer, which is accessed through a web browser, see figure 4.3.1. This composer acts as an interface to the server instance of ThingWorx Foundation (PTC 2018a).

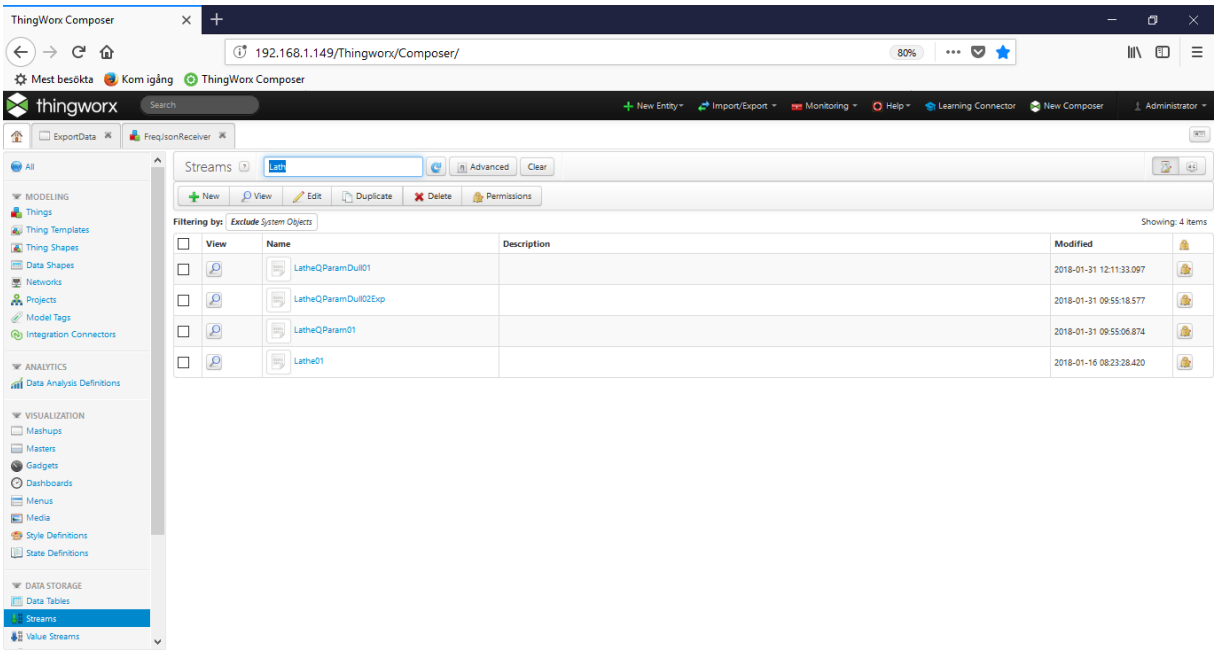

Figure 4.3.1: The ThingWorx Composer in Mozilla Firefox

Inside the composer it is possible to create and connect Things, which of course is the basic purpose of any IIoT-platform. Things can be basically anything, but in the case of production industry and IIoT, it is commonly production machines and sensors monitoring production. To exemplify with the veneer lathe that is being monitored in this project, it is possible to represent parts of the machine and its sensors as individual Things, or you could consider all of it as one Thing with the parts as properties of the machine. Alternatively, both is possible, with the parts tied to the main machine through services.

Properties are simply variables tied to the Thing and services are operations or functions relating to the Thing. Some services are supplied as default for different types of Things while others can be written by the user. To write services, ThingWorx makes use of a simplified version of JavaScript to create programming scripts. This means that it is not entirely code-free and some programming background is helpful when using the Composer. Furthermore, there are System Development Kits for several mainstream languages (PTC 2018a). These will be discussed later on. In many ways the Composer can be viewed as an Integrated Development Environment similar to what is used with other programming languages.

Part of what makes it easier to use for those who do not have a software developing background are the categorizations. There are built-in categories and by adding tags and creating Thing templates and shapes it is easy to group entities. Templates represent a Thing type, so that the developer easily can create several identical Things, while shapes contain abstract definitions. All Things have a template and templates can implement many different shapes. They also provide an easy way to change a property or a service for all Things of the same type (PTC 2018a).

Another essential category is Data Storage. Just as Things represent the parts of your process the Data Storage types contain historical and current data for the Things and the process. These can be independent or tied to a specific Thing and there are types adapted for both time series and non-time series data. The data in these containers is what is usually exported for use with ThingWorx Analytics (PTC 2018a).

To facilitate interaction with the application by the user, ThingWorx allows the developer to create so-called Mashups. These are graphical user interfaces tied to entities and their functions. Mashups can be used to create Human-Machine-Interfaces (HMIs) for the application, with buttons, displays, diagrams and similar objects being tied to different entities (PTC 2018a). Certain functions actually work better when done with the Mashup's widgets, as the Mashup objects are called, than with functions used in service scripts. An example would be when exporting data from a data storage container to a Comma Separated Value-file (csv file), where the Data Export widget preserves the order of the fields (i.e. matrix columns) but the extension for exporting csv files in a script service does not.

The ThingWorx platform can be used for a wide variety of implementations, but that is beyond the scope of this report. For this project the platform was used to receive data through a Remote Thing, a template for Things meant to connect to applications outside ThingWorx, and then store this data in appropriate containers and data shapes. The data was then filtered and combined before being exported to a format acceptable to ThingWorx Analytics.

4.4 ThingWorx Edge System Development Kits

Apart from the main platform, ThingWorx also has system development kits (SDKs) for connecting external implementations, via PTCs own 'AlwaysOn' protocol (PTC 2018a). There are supported SDKs for C, Java, .NET, iOS and Android. The main SDK is the one for C, which is understandable considering that a lot of Things are running embedded systems. The main reason for the SDKs is to easily facilitate communication between existing programs and ThingWorx applications, or for functions not supported in ThingWorx. Of course, they can also be used if the developer prefers to implement some functionality using any of the supported SDKs (PTC 2018c).

There is also the possibility of using analytics at the Edge, so called Edge Analytics. This is implemented with the ThingWatcher, which is a Java API that can be implemented with the Java SDK or in a standalone Java program. The ThingWatcher contains reduced analytics that can be used for detecting signal deviation (PTC 2018b).

For the purpose of this project, the Java SDK was used to convert data from the format of xml files to json objects, another platform independent data format (W3Schools 2018b), and then send it to ThingWorx. The Java application was primarily used to transfer large amounts of historical data to be used for modeling purposes, but it can also be used to pass live data or simulate the passing of live data. The ThingWatcher will not be utilized.

5 Machine Learning and ThingWorx Analytics

5.1 ThingWorx Analytics

ThingWorx Analytics is one of the modules of the ThingWorx platform. Its purpose is to discern information out of data in the ThingWorx applications. The idea is to make available tools used for machine learning and statistical analysis available through a simple user interface. This in turn allows users and developers that do not have an in-depth knowledge of data science to analyze data they store in their applications (PTC 2017a).

5.1.1 ThingWorx Analytics Server and Extension

ThingWorx Analytics requires quite a bit of setting up before it can be connected and used with the ThingWorx platform. Apart from needing a working installation of ThingWorx Foundation and access to the Composer, it also needs a running instance of ThingWorx Analytics Server. Furthermore, the ThingWorx Analytics Extension needs to be applied to Foundation. Optimally, ThingWorx Foundation and Analytics Server will run on different server machines. The Analytics Extensions has two purposes. One is supplying a means to connect Foundation to Analytics Server. This creates appropriate Things representing the Analytics Server instance and makes the Analytics functionality accessible through Mashups. The second part is supplying functionality that allows the use of models for real time predictions. This function only requires the Extension (PTC 2018 b).

5.1.2 Analytics Builder

The first major part of ThingWorx Analytics is the Analytics Builder. This is where data sets are uploaded for modelling and analysis purposes. The Analytics Builder connects to the Analytics Server via an Analytics Server Thing that contains all the needed information. This Thing is auto-generated with Analytics Extension by entering certain parameters for the Analytics Server. Figure 5.1.1 gives an overview of the connection.

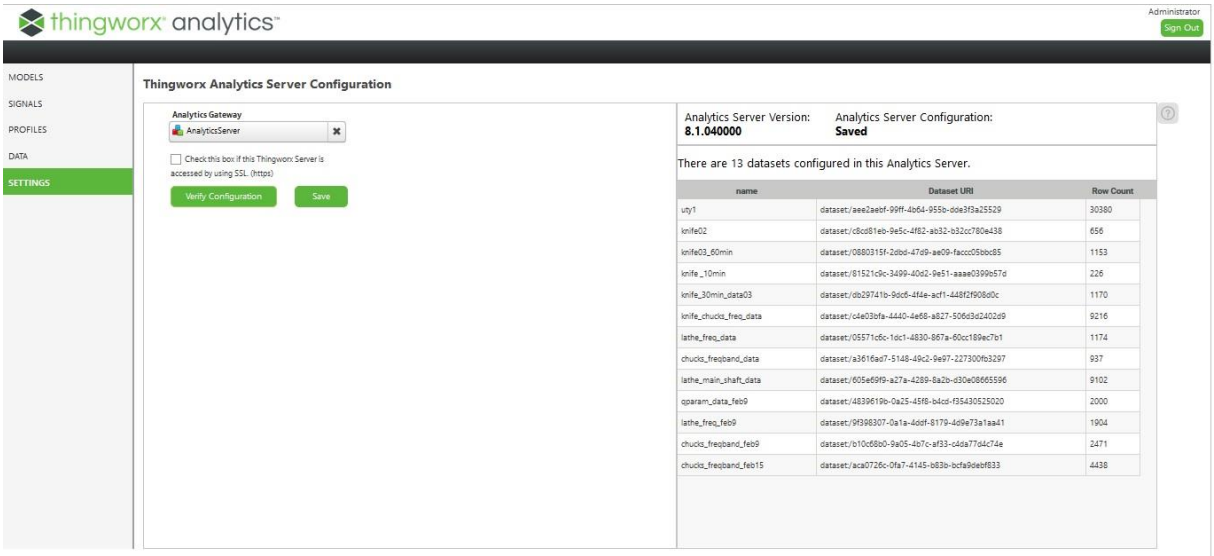

Figure 5.1.1: The settings window of the Analytics Builder. It shows the connection information and also a list of datasets.

On the left-hand side all the functions of the Analytics Builder can be seen. *Data* is where data sets can be uploaded and viewed. This is also where filters are most easily viewed and created. *Signals* is used to determine significant machine learning features affecting a chosen goal for analysis. *Profiles* investigates combinations of features in relation to a chosen goal. Lastly, *Models* is where modelling is done using statistical and machine learning algorithms (PTC 2018 b).

5.1.3 Datasets and Filters for Analytics Builder

The Analytics Builder takes datasets in two files. A Dataset Field Configuration file, in the form of a json file, is needed to define the Dataset properties. For the configuration there are seven properties available for each feature, also referred to as fields in ThingWorx Analytics. The mandatory properties are Field Name, Data Type and Op Type. The first two are just the basic information: naming and type declaring the feature. Op Type states how the variable can be used, meaning for example if it is numerical, informational or temporal. For appropriate types it is also possible to state minimum and maximum values, time sampling interval for time-series data, if the field is constant for all samples as well as acceptable values for strings (PTC 2018 b).

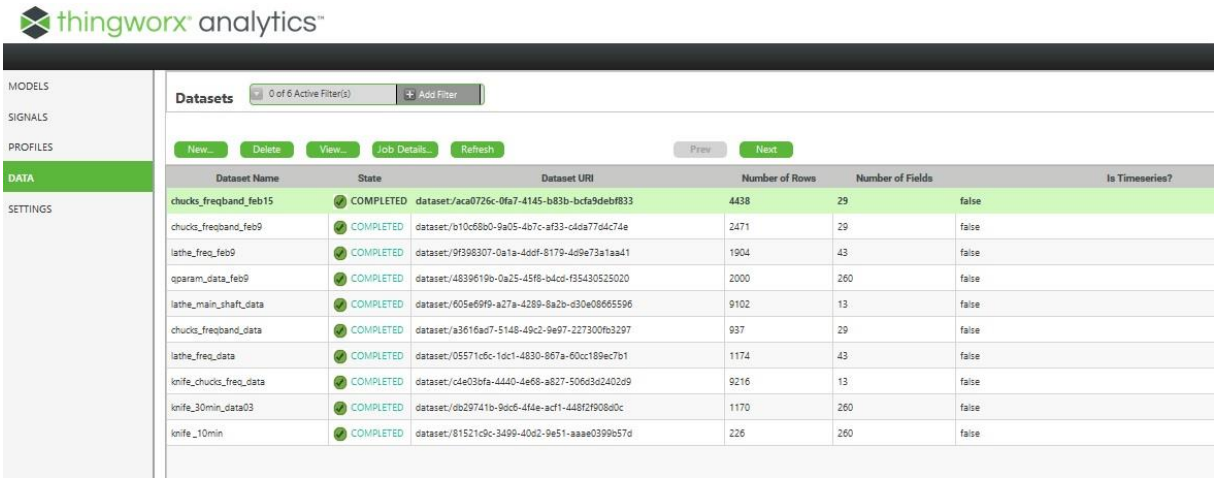

Figure 5.1.2: Data tab. Showing all datasets and some of their basic information

The dataset itself is defined in a csv file, with columns for machine learning goals and features. If the csv file has a header for each column, this needs to be noted when uploading the dataset or the Builder will give an exception. The idea is to feed information from connected Things, but any data that is in the right format can be used to generate models (PTC 2018 b).

To perform the analysis, appropriate goal fields must be included in the data set. Several goals can be defined but these will have to be used separately from each other and will be handled like any other feature if not stated as the goal. Which features are to be used as goals is not set in the data but defined for each operation performed on the data set inside Analytics Builder. Goals can be of Boolean type, continuous numerical values or groupable information. The last type can be either categorical, like hair color, or ordinal, such as economic status (high, medium, low) (PTC 2018 b).

Once uploaded successfully, the dataset will appear in the tab shown in figure 5.1.2. Further information can be accessed by viewing individual jobs, as seen in figure 5.1.3. If certain properties

are not defined, they are ascribed a default value. Minimum and maximum are recorded for continuous features (PTC 2018 b).

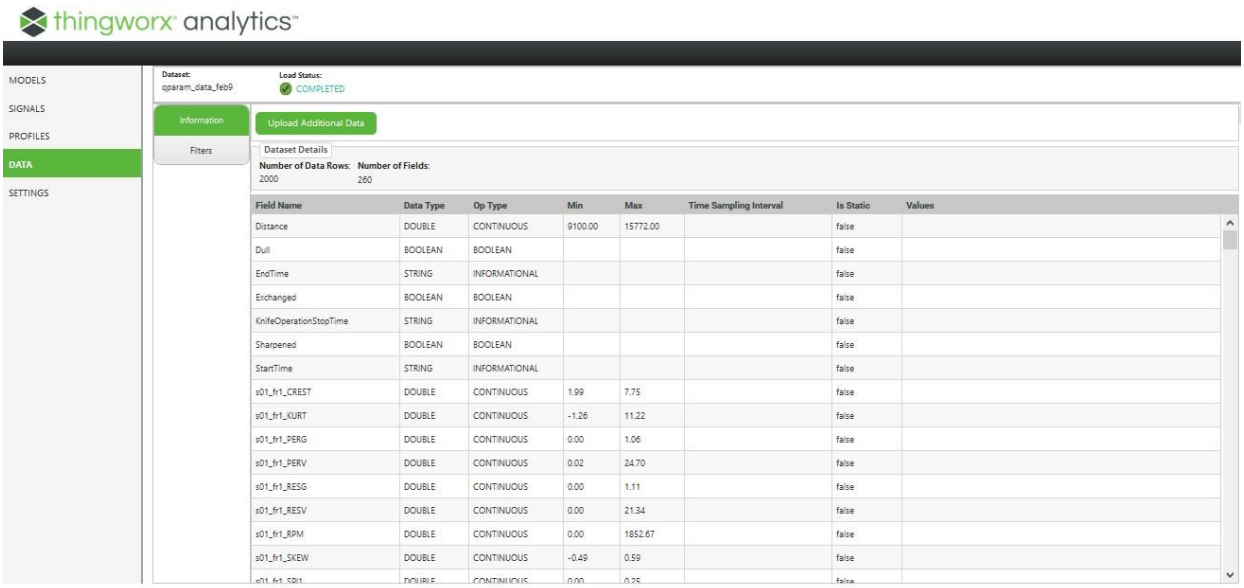

Figure 5.1.3: Overview of a specific dataset. It shows size of the dataset and information for individual features.

To facilitate different analyses, Analytics Builder also allows for filtering the data. Figure 5.1.4 shows an overview of the filter tab. By default, there is a filter called “all_data” which represents the entire dataset. This is because a filter must always be chosen. When new filters are created, these indicate how much of the whole dataset the filtered data contains. If selected, it also shows which parameters have been filtered and for which values (PTC 2018 b).

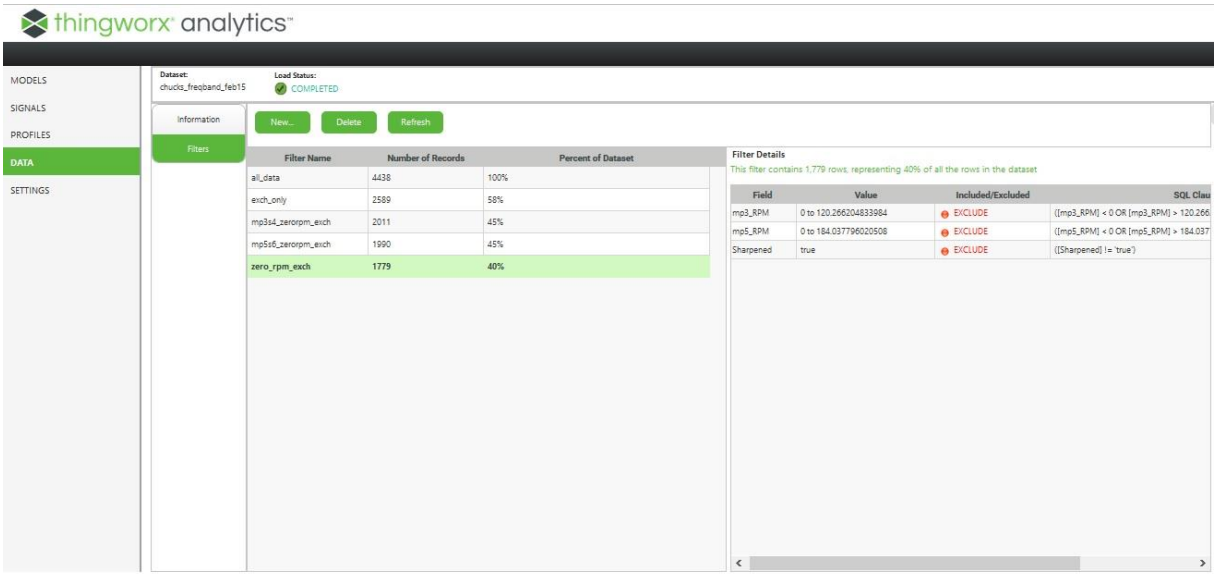

Figure 5.1.4: Filter tab of a dataset. It shows an overview of the filters and how of much data they contain.

The filtering is not entirely up to the user. Analytics Builder suggests certain intervals for each field and these cannot be altered. Filters can be used on data for all the different operations in Analytics Builder. It is also possible to exclude features when executing the different analysis jobs.

5.1.4 ThingWorx Analytics Builder Signals

ThingWorx Analytics Builder Signals is the most fundamental analysis in Analytics Builder. It checks which of the data features that are most statistically significant towards the defined goal. For example, if the features are sensors with several frequency bands and our goal is to detect when an alarm is triggered, Signals will rank the features (sensors bands) according to which one is most significant for the alarm occurring (PTC 2018b).

Signals are calculated using the concept of mutual information. This is done by computing Information Entropy for the feature and the goal. If there are large numbers of correlated features these may crowd other features that are perhaps less individually significant but still provide new information. Analytics Signals handles this by weighing features with their correlation to features that are already selected (Peysakhov 2017 15:25 min). This means that if modelling is done using only the more significant features, information loss is low.

Looking at figure 5.1.5, the predictive strength is indicated by the green bars and numbers next to the feature names in the lower left frame. Predictive strength is a normalized measure for the variable's ability to predict the goal, given between 0 (bad) and 1 (perfect). On the right, individual features can be looked at to show which values of the feature are most significant in relation to the goal. For each of those feature values the average value of the goal is shown along with how that average compares to the average for the entire data set (PTC 2018b). This information will be more relevant for a numeric goal than for a Boolean goal.

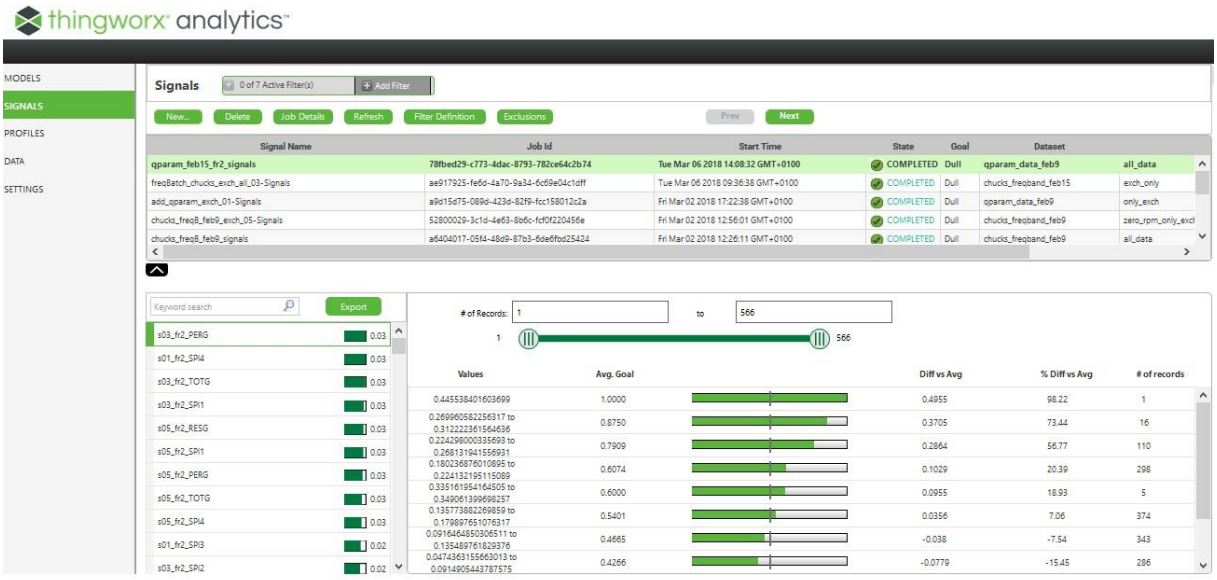

Figure 5.1.5: Signals tab. Predictive strength of individual features is displayed on lower left. Each feature can be selected for further information based on its strength in relation to the stated goal.

5.1.5 ThingWorx Analytics Builder Profiles

Another Analytics Builder function is Profiles. Profiles investigates if there are any groups of features that are significant for the stated goal. This may be any combination of features, but it may only appear once, meaning that feature 1 together with feature 2 excludes feature 2 together with feature 1. Using Profiles, we can see if the goal correlates with a set of features (PTC 2018b). An example would be monitoring the heat inside a wall, where temperature on both sides may affect it in combination. Profiles shows combinations, but it is still possible that the individual features, shown in Signals, are more important.

5.1.6 ThingWorx Analytics Builder Modeling

Modeling is the main functionality of the Analytics Builder and it allows the user to create predictive models that can be deployed in the main ThingWorx platform. An overview of the Models tab can be seen in figure 5.1.6. After finding patterns in the data ThingWorx can make predictions about where the process is heading and make suggestions. In most cases modeling is a two-part process. First a model is trained using machine learning and statistical algorithms. Then the model is validated using part of the data excluded in the training. Model validation is included by default, with a validation hold out of 20%, but it is not mandatory (PTC 2018b).

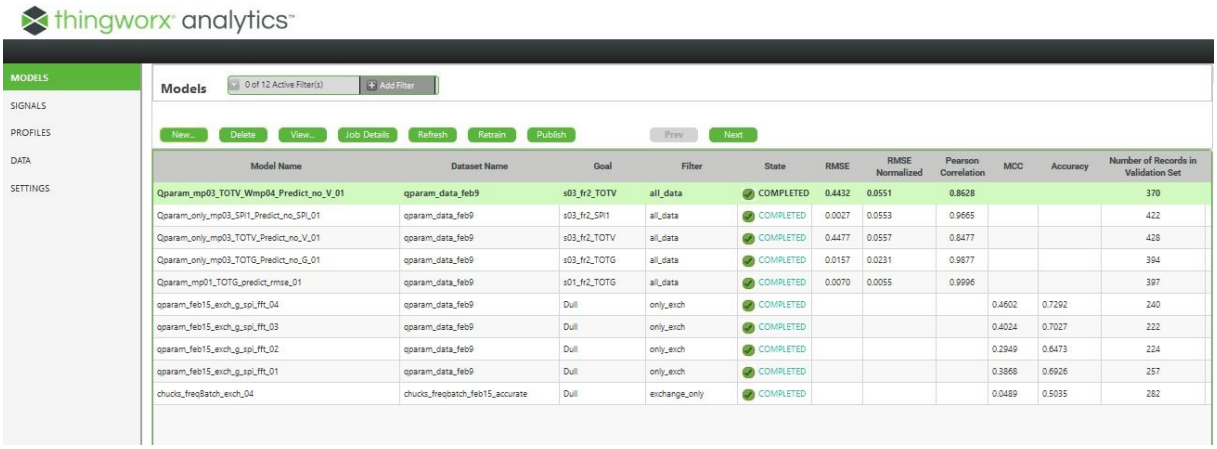

Figure 5.1.6: Models tab. Here it is possible to train new models and access all the predictive models that have been trained. Depending on the chosen goal for the model training, different statistical and machine learning measures are shown.

A relatively large variety of techniques, called learners, are available for model training. These will be discussed in section 5.2, along with the different comparison metrics used by Analytics Builder. The different learners are combined using ensembles. The ensemble technique is the chosen strategy for combining and evaluating learners. Average ensemble technique is when each learner scores predictions separately and then the scores are averaged. When Best is selected, only the best performing learner is used for scoring (PTC 2018b).

After the model has been trained and validated, several comparison metrics are displayed. The results can be viewed in a separate tab and what appears is also dependent on the type of goal chosen. After the model is trained it can be used for Predictive Scoring. Predictive scoring means examining a data set and making predictions based on similarities to records used in model training.

Each record is assigned a predictive score reflecting how good the prediction is. Scoring can be done on the exact same data that was used for training but ideally the user will upload additional data. Even if no additional data is used the scoring will not be perfect and the model will make a guess based on the sample. Another possible analysis done by Predictive Scoring is Important Fields. Here the user can specify a desired number of fields that will be displayed together with a weight. The weight indicates how influential the feature is on the goal (PTC 2018b).

5.1.7 ThingWorx Analytics Manager

Analytics Manager is where models can be used once they have been trained and published in Analytics Builder. It also allows use of computational models from external applications, such as Microsoft Excel or PTC's Mathcad. As previously mentioned, Analytics Manager does not require Analytics Server but does require that the Analytics Extension has been installed with ThingWorx Foundation. This means that if the developer or user has some other computational tool that they prefer they can still use this with ThingWorx. For every type of analysis tool an Analysis Provider is needed. This handles communication between the Analytics Manager and the analysis tool. Even models created with Analytics Builder require this, despite being part of the ThingWorx framework. For this purpose, PTC has developed ThingPredictor, which is run similar to a separate server program. The ThingPredictor handles predictive scoring, mentioned in section 5.1.6, for real-time use with the published model. While this might seem tedious it can allow the use of Analytics Builder models without having an Analytics Server connection, if an exported model is uploaded. If a model has been published it is available via an URL link when viewing the model. This will download an xml file containing the model in Predictive Model Markup Language (PMML) (PTC 2018b).

Once an Analysis Provider is working, Analysis Models can be imported. For models developed in Analytics Builder this is simply done by pressing the publish button once the model training is complete, see figure 5.1.6. After this it is possible to connect to a Thing in the main platform representing the predictive model. This is done partially in Analytics Manager and partially in the ThingWorx Composer. The Analytics Extension provides services that can be used to create Thing Templates and Things for this purpose. The Thing's properties are then mapped to the Analysis Models inputs and results. This allows for Analysis Jobs to be automatically generated for the most recent data (PTC 2018b).

5.1.8 Monitoring in ThingWorx

When the previous steps are finished, data can be visualized by creating applications with mashups, as discussed in chapter 4. This ties together the whole process. Data is received by the main platform and sent on to the analysis tool via the Analysis Provider. Once it has scored the data, the results can be displayed, and actions can be taken if needed. It is also possible to feed simulated data to the model to determine what would happen in different scenarios. This last part is what truly makes the model predictive (PTC 2018b).

5.2 Machine learning and statistical algorithms and measures

Analytics Builder uses a relatively large variety of learner techniques for model training. These are (PTC 2018b):

- Linear Regression
- Logistic Regression
- Neural Networks
- Decision Trees
- Gradient Boosting
- Random Forests

Furthermore, various measures are used to evaluate and compare the results of the modelling. A short description of these algorithms and measures will follow. For a more in-depth discussion the reader is encouraged to seek out works on machine learning and mathematical statistics.

5.2.1 Linear Regression

Linear Regression means trying to fit a linear equation to observed data points (Lacey 1998). In the simple case of one independent and one dependent variable this means the equation is a line, as seen in equation (1) and figure 5.2.1 (Sewaqu 2010).

$$Y = kX + m \tag{5.2.1}$$

Here *X* is the dependent variable and *Y* is the independent variable. *k* denotes the relationship between these two variables, in this case the inclination of the line. *m* is the value of *Y* when *X* is 0.

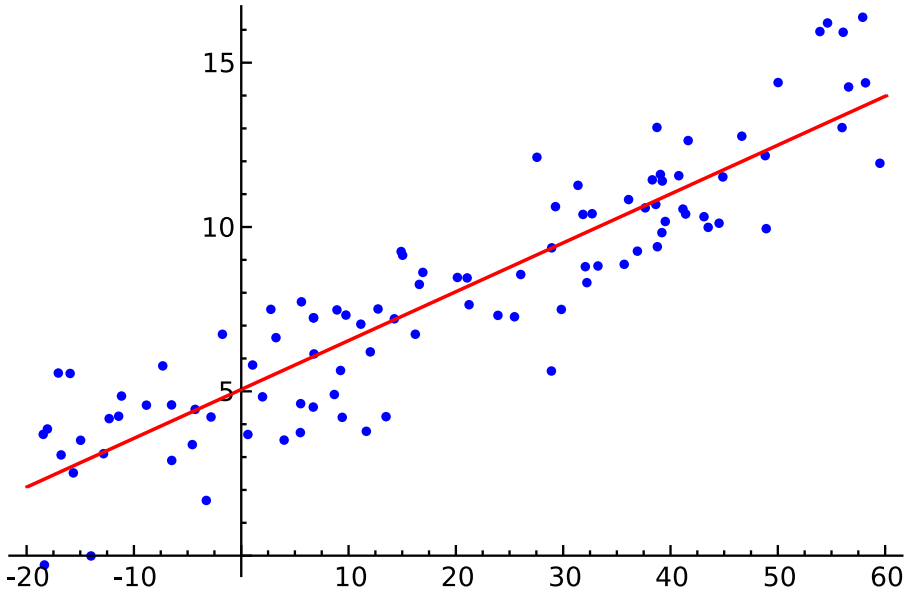

Figure 5.2.1: Linear Regression. Blue dots represent measured values and the red line is the obtained pattern.

The most common method for doing this is using the least squares method, which means minimizing squares with opposite corners on the observed data point and the line (Lacey 1998). For data with more independent variables the geometric comparison is not valid, but the principle for computation is still the same.

5.2.2. Logistic Regression

Logistic Regression is very similar to Linear Regression. The main difference lies in that the dependent variable is categorical, for example a Boolean of true or false value. This can be seen as a categorization of samples. Given a certain combination of observed values for features the method tries to ascertain the status of the dependent variable (Pennsylvania State University 2018).

5.2.3 Neural Networks

Neural Networks, or more precisely Artificial Neural Networks, are learning algorithms inspired by biological neural networks. Idealized neurons are studied to create machines that can learn and recognize patterns. These neurons are modelled as multiple input single output (MISO) units. Each input has a weight attributed to it to rate the ingoing information. This is modelled in two parts. First, we have equation (5.2.2) (MacKay 2003 p471)

$$a = \sum_i w_i x_i \tag{5.2.2}$$

where i is the number of inputs, w_i are the weights, x_i are the inputs and a is called activation and is the weighted input signal. Secondly, we have the output y, also called activity, which is a function of the activation. A common choice is the linear logistic function, which together with (5.2.2) gives equation (5.2.3) (MacKay 2003 p471-472).

$$y = \frac{1}{1+ e^{-a}} \qquad y \in (0,1) \tag{5.2.3}$$

This single neuron can be seen as a neural network on its own. If we also have a target value, t, for our output the idea is to train the network so that it learns a model of the relationship between input, $\boldsymbol{x}$, and target, t. This is a matter of calibrating the weights, $\boldsymbol{w}$, by minimizing an error based on the difference between the target and the output. For predictive purposes this means that once the neural net has been trained so that its output matches the target as accurately as possible, it can then state what will happen for a given set of feature inputs (MacKay 2003 p473-476). To be useful, the artificial neurons are connected into networks of several neurons. This allows them to become more than singular amplifiers and have a memory function (MacKay 2003 p505-507).

Hidden Neurons can be introduced between the input and output layer. These do not correspond to observed variables and can play other roles in the probabilistic model. Often, they function as feature detectors used to spot patterns in variable shifts and high-order correlations (MacKay 2003 p525-526).

Neurons can be combined into multilayer perceptrons, which contain several levels of rows of feedforward neutrons. These are given by one input layer, one output layer and one or several hidden layers. A neuron in a layer sends its output forward to every neuron in the next layer, but not to its neighbors in the same layer. This type of feedforward network can be seen in figure 5.2.2 (Glosser 2013) (MacKay 2003 p527).

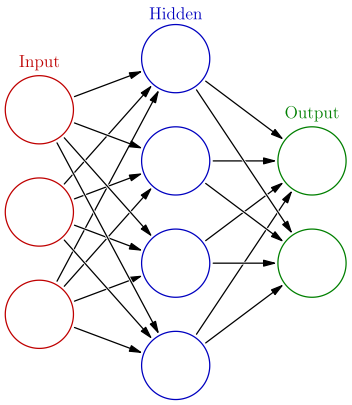

Figure 5.2.2: A Multilayer Neural Network with one hidden layer.

While Linear Regression produces a flat plane for three-dimensional space, a Neural Network of the type mentioned above will give a membrane-like surface. This means that a Neural Networks can fit more complex functions to the data, which is useful when looking at more complex input-output relations (MacKay 2003 p528).

5.2.4 Decision Trees

Decision Trees tries to determine different scenarios depending on the possible values of the feature variables, as seen in figure 5.2.3 (Manske 2005). Each level contains all possible values of the current variable and it splits into new branches for each of those. The boxes are called leaves and show the current value of the dependent variable, the goal. When branched out this gives a blue print which allows the model to state the value of the goal depending on which leaf the feature values lead to (PTC 2017b 5:55 min).

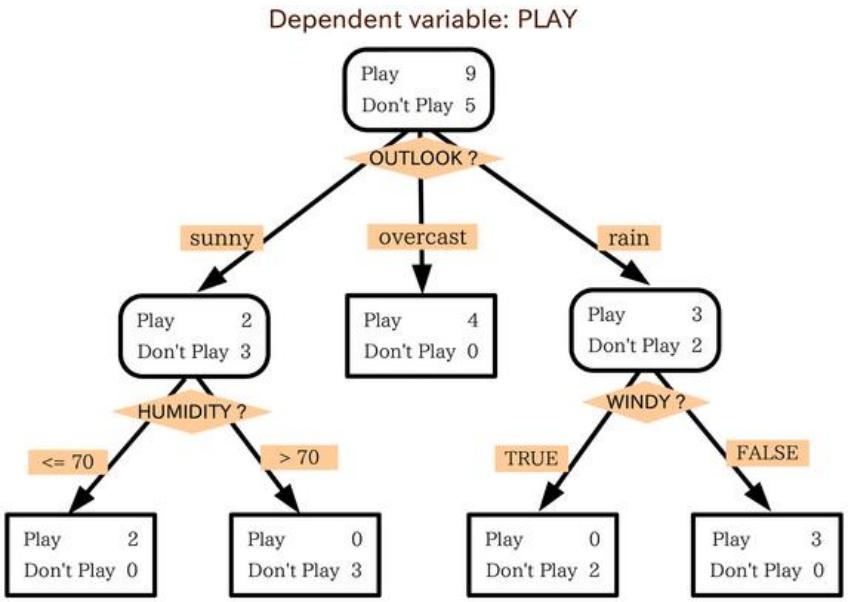

Figure 5.2.3: Decision Tree. Each level represents possible values for a given variable. For every value the tree splits into new branches.

5.2.5 Gradient Boosting Machine

Gradient Boosting Machines are a type of ensemble technique that uses several weak learners to boost predictions. That a learner is weak means that it gives only a rudimentary classification. The rationale behind this is that it is often easier the find several rough rules of thumb, than finding one single accurate rule for prediction (Schapire 2003). Often weak Decision Trees are used, meaning trees that are not fully grown in order to improve computation time. The Gradient Boosting is done by adding a new learner for each iteration and then training it based on the error for the whole ensemble so far. There is a risk of overfitting the model to the training data, but Gradient Boosting Machines make up for it with high flexibility and easy implementation (Natekin & Knoll 2013).

5.2.6 Random Forests

Random forests are another ensemble technique that is specific for Decision Trees. A Random Forest uses several deep or fully grown trees to make predictions. Another difference from a Gradient Boosting Machines, which uses Decision Trees, is the use of random stochastic samples of data. The randomness is used with the aim of getting better generalization and less overfitting to the training data (Ho 1995).

5.2.7 Comparison Metrics

Analytics Builder uses several comparison metrics, differing depending on the type of goal used. For a continuous numerical goal, Root Mean Square Error (*RMSE*) and Pearson Correlation is shown. *RMSE* is a common statistic measure and is computed with the difference between the observed and predicted values of the goal. *RMSE* is also displayed normalized between 0 and 1, referred to as *RMSE* Normalized. The Pearson Correlation, or just Correlation, looks at how much the predicted and observed value of the goal vary in unison. Furthermore, a scatter plot is displayed, figure 5.2.4 (PTC 2018b), with predicted results plotted as a function of actual values. Color denotes if the results where over-predicted, accurately-predicted or under-predicted (PTC 2018b).

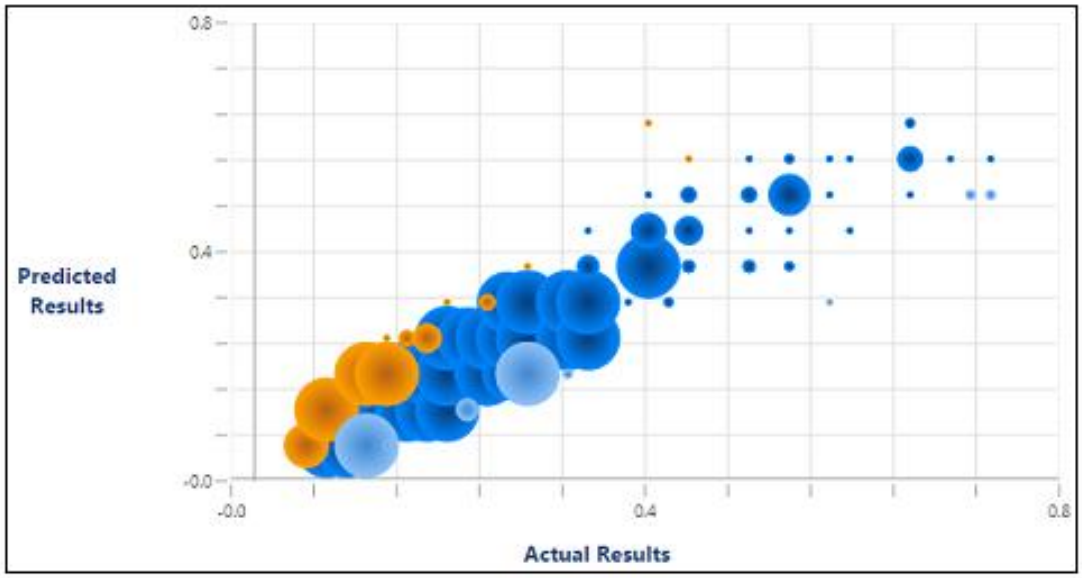

Figure 5.2.4: Scatter plot for continuous goal variable. Color denotes accuracy of prediction and size denotes number of predictions.

For binary goals a confusion matrix, figure 5.2.6 (PTC 2018b), and a Receiver Operating Characteristics (*ROC*) curve, figure 5.2.5 (PTC 2018b), are displayed along with a numerical *ROC* value and the Matthews correlation coefficient (*MCC*) (PTC 2018b).

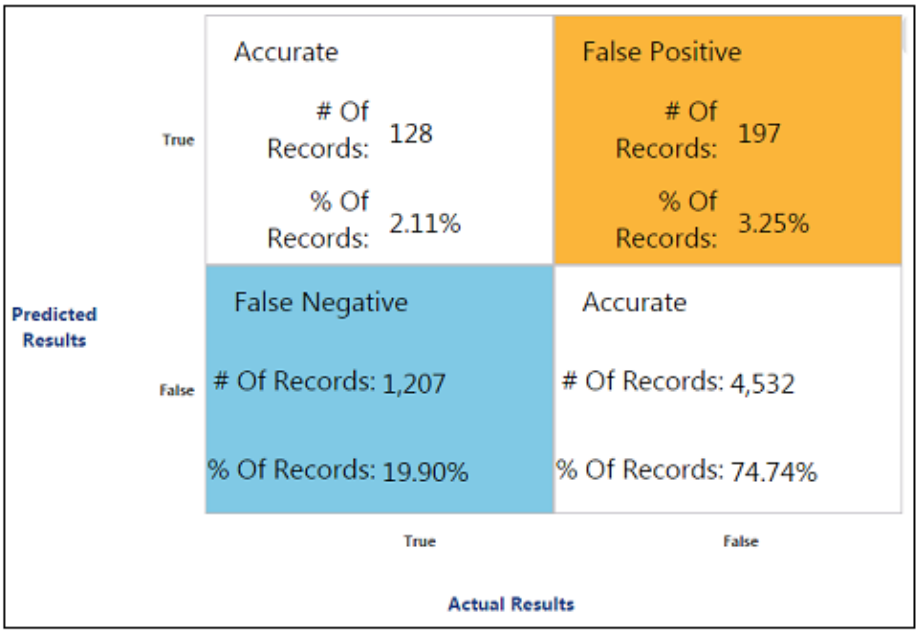

Figure 5.2.5: Confusion Matrix. Showing numbers of and rates of true and false positives and negatives.

The confusion matrix shows the outcomes of the predictions in relation to the actual results. Number of records and percentage of the validation set are shown for true positive (prediction and actual result true), false positive (prediction true and actual result false), true negative (prediction and actual result false) and false negative (prediction false and actual result true) (Fawcett 2005).

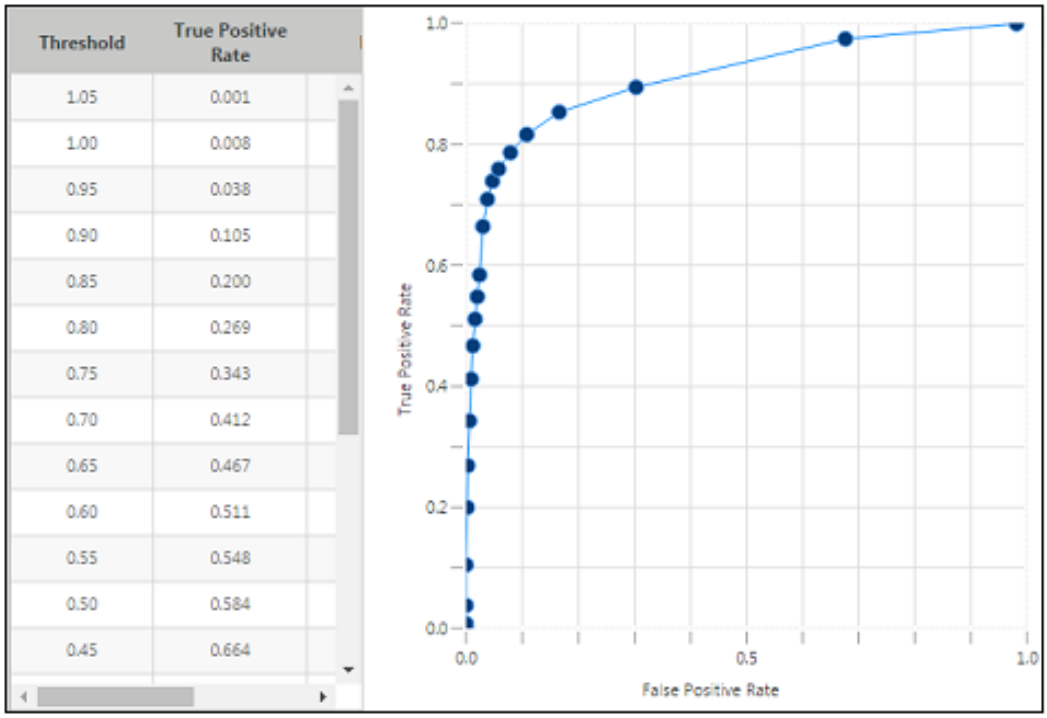

Figure 5.2.6: ROC curve. The curve has the desired appearance, by going up quickly, showing that true positives significantly outnumber false positives.

The *ROC* curve plots true positives as a function of false positives for various thresholds. The threshold is mostly significant when values between false (0) and true (1) are meaningful. It is desirable that the curve follows the left (west) and top (north) of the plot as much as possible for a high true positive ratio. Visualizing the *ROC* can give a more intuitive feeling for the prediction performance than simply calculating the true false positive rate (Fawcett 2005).

MCC is a measure of the correlation between true and false positives and negatives. It can be calculated from the confusion matrix according to equation 5.2.4:

$$MCC = \frac{tp \times tn - fp \times fn}{\sqrt{(tp+fp)(tp+fn)(tn+fp)(tn+fn)}} \quad (5.2.4)$$

with *tp* being true positive, *tn* being true negative, *fp* being false positive and *fn* being false negative. An advantage of the *MCC* is that it takes into account the entire result population and only scores high if the model predicts well on both outcomes. It will also signify if there are no classifiers on one side, for example $tn = fn = 0$, by being undefined (Chicco 2017).

6 Comparing analytics tools and CSense

To able to make some comparative evaluation of ThingWorx Analytics, another analytics program will also perform analysis on the data. This will establish a baseline.

CSense is a product from General Electric (GE) that provides predictive analytics, with similar goals and uses as ThingWorx Analytics. It is closely tied to GE's IIoT platform, Predix, and their HMI system, iFix (General Electric 2016). For the moment though, CSense is not sold commercially but GE has plans to include it in their cloud analytics.

The main components for analytics and monitoring in CSense are Proficy Troubleshooter and Proficy Cause+. The Troubleshooter has separate versions for continuous process data and for discrete and batch process data. These take data in csv format, just like ThingWorx Analytics, but they do not need a data configuration file that states data shape and use. This is due to the Troubleshooter having data preparation as a built-in function, with more options than Analytics Builder's filter function (GE Digital 2010). It does, however, not take data in Boolean values. For this it is needed to configure the Booleans as binary variables, 0 and 1.

There are several tools for visualizing and inspecting the data once it has been uploaded and prepared. One useful feature is the possibility to compute a correlation matrix for all or some of the features. Another is viewing trends and histograms of individual fields. It can also build decision trees that allows the user to investigate how the process ends up in different states (GE Digital 2010).

Troubleshooter uses statistical methods and decision trees, just like ThingWorx Analytics. There is, however, no mention of using decision tree ensembles like random forest or gradient boosting machines. For the Continuous Troubleshooter, the statistical methods used are not described in detail and it is not possible to adjust parameters for these. It tries to accomplish two things when doing Continuous modeling. One is to create a non-linear model following the target. The other is to create rules for the parameters relating to the target. When using CSense for process control, this can be very useful (GE Digital 2010).

When it comes to the Discrete and Batch Troubleshooter, there is the possibility for creating models using Principal Component Analysis, Partial Least Squares and Decision Trees. For batch processes it recommends using Principal Component Analysis and for discrete processes it recommends a Decision Tree (GE Digital 2010).

Principal Component Analysis (PCA) is a method to handle processes with a large number of variables that are correlated to a varying degree. By performing linear algebra calculations on the covariance matrix (how the variables vary together) PCA reduces several variables down to a few that still capture the majority of the system characteristics (Wise & Gallagher 1996). Part of this is showing how much of the variability that is captured in each of the calculated principal components. A limit on how much of the system needs to be captured can be decided by the monitoring strategy. It is also possible to see which of the original variables make up the principal components and track any deviations back to the original parameters. This means that an operator can observe the reduced system and only look at the original larger number of parameters when there is a problem (Wise & Gallagher 1996).

Partial Least Squares (PLS) combines PCA with linear regression. This means that it looks at both correlation and covariance of the variables. One of the main benefits of PLS is that it can handle several variables and several predicted variables (Wise & Gallagher 1996).

For real-time use, GE delivers Proficy Cause+. Cause+ focuses on process monitoring and control and makes use of the models developed with Proficy Troubleshooter. It allows you to receive and analyze real-time data against set alarm levels and can deliver messages and suggestions based on this. It can also log historical data that allows users and technicians to look at events from previous shifts (GE Digital 2011). In this way, it can perform similar functions to ThingWorx Foundation when dealing with process monitoring. For the use in this project, Proficy Continuous Troubleshooter will be the primary tool for comparison. This choice was made for easier comparison to ThingWorx Analytics and due to the process data.

7 The Veneer Lathe Modeling and Prediction Application Procedure

Here the application used for processing and analyzing data is explained. It consists of several parts and a path of multiple steps is followed to go from the data onsite to predictive scoring results. This will result in some repetition from the previous chapters, but for sense of clarity all components will be covered. An overview of how the parts are connected can be seen in figure 7.1.

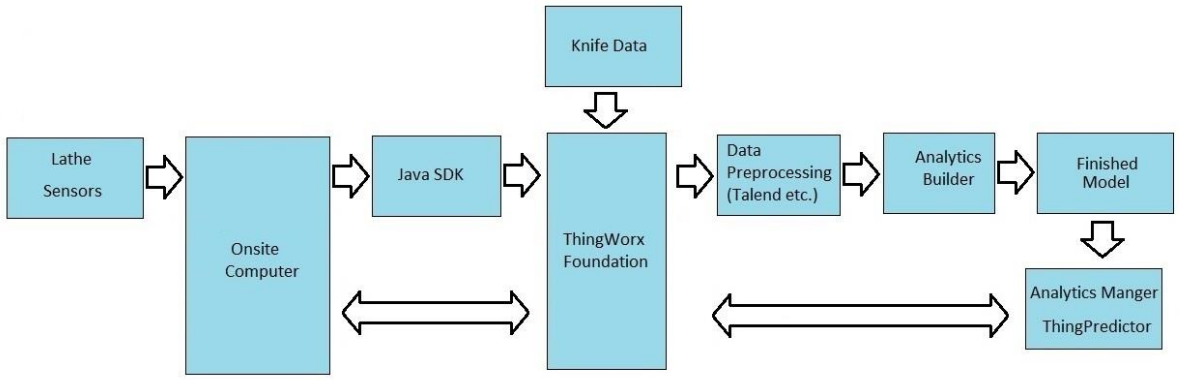

Figure 7.1: A flow diagram of how the data is moved and processed. The upper part shows the steps from the production site up to the finished model. The lower section illustrates real-time flow when the model has been deployed.

As explained in chapter 3, there is a computer on-site that gathers measurements from the mounted sensors. This is done at even intervals, going through each of the sensors on the lathe and on the pumps and gathering data with timestamps for each sensor. These gathered measurements are processed and recorded as Vikon's Q-parameters and as the most significant frequencies inside xml files. This stored data is then transferred from the onsite computer to the computer running the application offsite. Once downloaded, the sensor data is transformed from xml format to json objects before being sent to a Remote Thing in the ThingWorx platform. The data format transformation and sending it to ThingWorx is done with a Java program that is written for this purpose, which utilizes ThingWorx's Java SDK. ThingWorx Thing properties can be of xml type, but this project is a part of a larger project for predictive maintenance that receives data in json format. The choice to use json was made to make the application easier to integrate with the larger project.

The main part of the ThingWorx platform, ThingWorx Foundation, is run on a Windows Server 2016 machine. Here the sensor data is received by the Remote Thing with the json property. The data is then stored in Data Container Streams with Data Shapes appropriate for the stored data. For example, one Stream holds the frequency data and another one stores the Q-parameters. The other part of the data, which contains information on knife exchange and sharpening, is also read to the platform. The knife data is in csv format to begin with and can be read using a ThingWorx service extension called CSV Parser.

Figure 7.2: The basis for combining the knife data with the sensor data. E represents a planned exchange after producing 15'000 meters of veneer. S stands for sharpened and denotes that the knife is sharpened at 10'000 produced meters. U E is for an unforeseen exchange relating to knife damage. The blue boxes are for sensor data samples included before the event. These are marked as dull. The purple boxes are for included samples after a knife event. These are marked as not dull, meaning the knife is sharp.

Once both parts of the data, sensor and knife events, are inside the platform they are combined. This is done by taking an entry for a scheduled knife event and then combining it with the sensor data. The basis for the combination is taking a few entries *before* the production stop timestamp and marking them as dull and then taking an equivalent number of samples *after* the production start timestamp and labeling these as sharp. Here, scheduled events refer to not including instances where the knife was exchanged due to some type of damage. It only includes those events when it has produced the set veneer length for sharpening and exchange. An overview of this can be seen in figure 7.2. The sensor data entries queried in relation to a knife event are also checked to make sure that they do not coincide with another knife event. This might happen if an unforeseen event occurred in a short time span after another event. Once the data has been combined, it is exported to a new csv file via a Mashup using the Data Export Widget. This data file then undergoes some minor alterations with a data preparation program, named Talend as well as some editing in Notepad++. The final step before going to ThingWorx Analytics is configuring the Data Configuration json file, which specifies the type and use of the data.

When all of the preparation of the data is done, it is uploaded as a Data Set in ThingWorx Analytics Builder. Here the data can be handled and analyzed according to the methods and functions explained in chapter 5. The analysis done in CSense uses the same csv files but with the differences noted in chapter 6. ThingWorx Analytics Server runs on a Linux Virtual Machine on the same server that runs ThingWorx Foundation. CSense is run on a separate Windows Server.

Once the model is created and validated, it is published to Analytics Manger. Here ThingWorx can send data in real-time for predictive scoring. This is done by generating Things that represent the model and are connected via the ThingPredictor analysis provider. The result of the scoring job is then returned to the Thing in ThingWorx and it can be utilized for whatever is desired.

8 Results

The results are divided between ThingWorx in section 8.1 and 8.2 and CSense in section 8.3. Please note again the previously mentioned discrepancy between sensor index and measuring point index. For example, that sensor 4 is measuring point 3.

8.1 Knife Dullness Prediction with ThingWorx

During the visit to the veneer production site it became clear that sharpening is done manually by the operator and that it is up to him or her to decide when the knife is sufficiently sharp. Therefore, it was assumed to be more appropriate to primarily look at entries close to the exchange of the knife since this might more uniformly represent dull and sharp states.

The first analysis is seen in figure 8.1.1 where Analytics Builder Signals has been run on the Q-parameter data. The results show that the sensor mounted on the left chuck housing is deemed most significant. Also, the opposite chuck sensor is listed. Two features from the left sensors on the motor, sensor 1, are listed. Sensor 1 is the farthest from the cutting process, so this is somewhat surprising. The most important features seem to be those dealing with vibration acceleration, parameters ending in G, and vibration in the low, SPI1, and high, SPI4, parts of the friction noise spectra. The predictive strength for the top features, 0.03, implicates that they have a low correlation with the goal. Analytics Builder Profiles did not yield any combinations containing more than one significant feature.

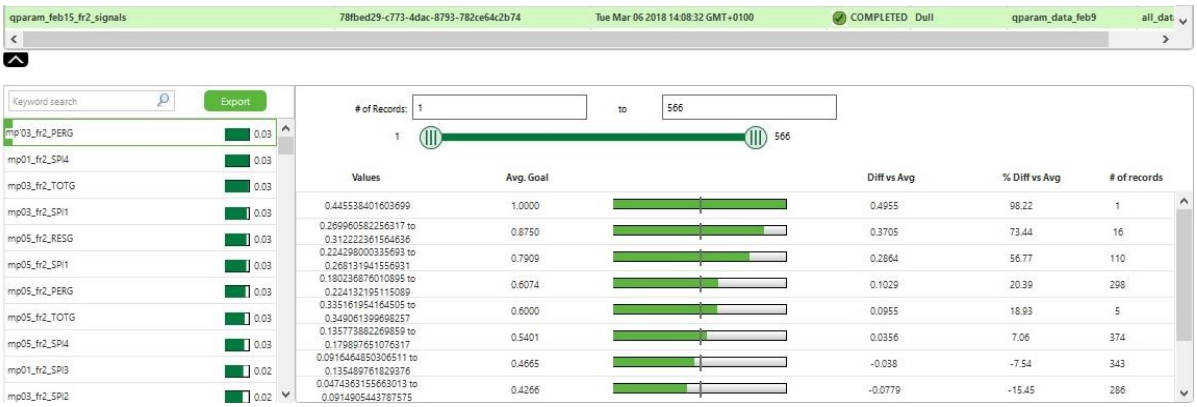

Figure 8.1.1: Signals for the Q-parameters with the Boolean variable Dull as the goal. The most relevant part is the frame in the lower left showing important features and their predictive strength on the goal.

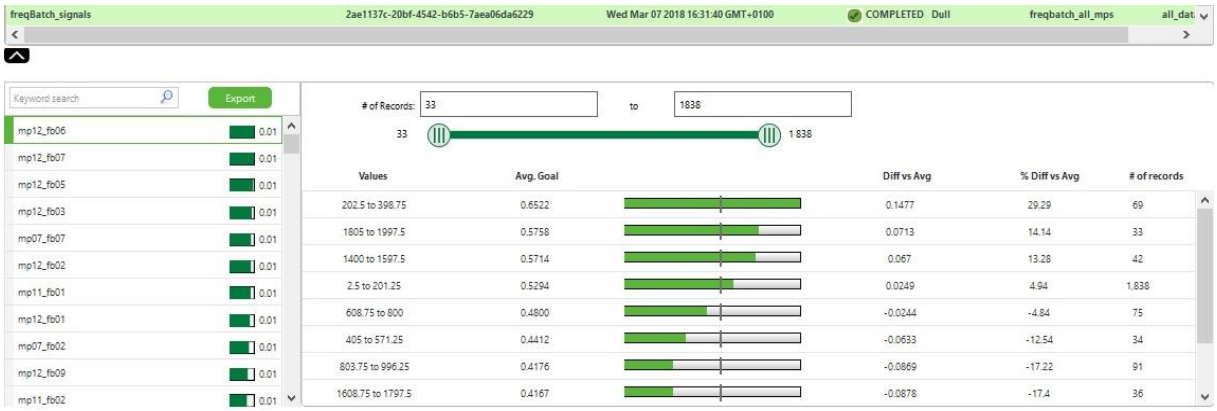

Figure 8.1.2: Signals for the frequency batch data with the Boolean variable Dull as the goal. The most significant part is the frame in the lower left showing important features and their predictive strength on the goal.

In figure 8.1.2, Signals was calculated using frequencies. This shows less predictive strength than for the Q-parameters. In section 3.3.1 it is noted that the frequencies are recorded according to amplitude value. It is therefore interesting to note that the frequency position indicates that high amplitude is not what is most significant for the goal. It does, however, show completely different sensors than the Q-parameter Signals.

Model Name	Dataset Name	Goal	Filter	State	RMSE	RMSE Normalized	Pearson Correlation	MCC	Accuracy	Number of Records in Validation Set	Create Date
qparam_GradBoost_exch_03	**qparam_data_feb9**	**Dull**	**only_exch**	**COMPLETED**				**0.3036**	**0.6468**	**252**	**Fri Mar 09 2018 12:59:35 GMT+0100**
qparam_GradBoost_exch_02	qparam_data_feb9	Dull	only_exch	COMPLETED				0.3266	0.6654	254	Fri Mar 09 2018 12:58:53 GMT+0100
qparam_GradBoost_exch_01	qparam_data_feb9	Dull	only_exch	COMPLETED				0.3250	0.6624	237	Fri Mar 09 2018 12:41:06 GMT+0100
qparam_RanFore_exch_03	qparam_data_feb9	Dull	only_exch	COMPLETED				0.3344	0.6626	246	Fri Mar 09 2018 12:36:30 GMT+0100
qparam_RanFore_exch_02	qparam_data_feb9	Dull	only_exch	COMPLETED				0.4232	0.7113	239	Fri Mar 09 2018 12:36:05 GMT+0100
qparam_RanFore_exch_01	qparam_data_feb9	Dull	only_exch	COMPLETED				0.3022	0.6513	238	Fri Mar 09 2018 12:33:38 GMT+0100
qparam_DecTree_exch_03	qparam_data_feb9	Dull	only_exch	COMPLETED				0.2302	0.6148	244	Fri Mar 09 2018 12:27:08 GMT+0100
qparam_DecTree_exch_02	qparam_data_feb9	Dull	only_exch	COMPLETED				0.2817	0.6419	215	Fri Mar 09 2018 12:26:43 GMT+0100
qparam_DecTree_exch_01	qparam_data_feb9	Dull	only_exch	COMPLETED				0.1124	0.5536	233	Fri Mar 09 2018 12:20:24 GMT+0100
qparam_LogReg_exch_03	qparam_data_feb9	Dull	only_exch	COMPLETED				0.2341	0.6140	228	Fri Mar 09 2018 11:36:22 GMT+0100
qparam_LogReg_exch_02	qparam_data_feb9	Dull	only_exch	COMPLETED				0.2446	0.6292	240	Fri Mar 09 2018 11:35:11 GMT+0100
qparam_LogReg_exch_01	qparam_data_feb9	Dull	only_exch	COMPLETED				0.3146	0.6540	237	Fri Mar 09 2018 11:31:01 GMT+0100
qparam_NN_exch_03	qparam_data_feb9	Dull	only_exch	COMPLETED				0.0000	0.5043	230	Fri Mar 09 2018 11:14:22 GMT+0100
qparam_NN_exch_02	qparam_data_feb9	Dull	only_exch	COMPLETED				0.2821	0.6410	234	Fri Mar 09 2018 11:13:54 GMT+0100
qparam_NN_exch_01	qparam_data_feb9	Dull	only_exch	COMPLETED				0.2731	0.6372	215	Fri Mar 09 2018 11:08:01 GMT+0100

Figure 8.1.3: A comparison of using different learning algorithms for modeling. All of them where retrained two times to investigate consistency of results. NN stands for Neural Network, LogReg for Logistic Regression, DecTree for Decision Tree, RanFore for Random Forest and Grad Boost for Gradient Boosting Machines. The models use Q-parameters and all measuring points are included. Dataset with circa 1000 samples.

Figure 8.1.3 looks at different learners being trained on data using only exchanges and Dull as the goal. All of them are retrained to investigate consistency in the accuracy. Linear Regression has been opted out in favor of Logistic Regression since the goal is a Boolean variable.

Model Name	Dataset Name	Goal	Filter	State	RMSE	RMSE Normalized	Pearson Correlation	MCC	Accuracy	Number of Records in Validation Set
qparam_mod_learners_exch_03	**qparam_data_feb9**	**Dull**	**only_exch**	**COMPLETED**				**0.2400**	**0.6197**	**234**
qparam_mod_learners_exch_02	qparam_data_feb9	Dull	only_exch	COMPLETED				0.3262	0.6638	232
qparam_mod_learners_exch_01	qparam_data_feb9	Dull	only_exch	COMPLETED				0.2577	0.6266	241

Figure 8.1.4: Overview of results for Dull model training with Q-parameters for all sensors and measuring points. Learners used for training are Logistic Regression, Random Forest and Neural Net. They are also filtered for samples only relating to the exchange of the knife and not manual sharpening.

After looking at several learners some modeling was done using combinations of learners. These can be seen in figure 8.1.4 and detailed results for one model is displayed as an ROC curve in figure 8.1.5 and confusion matrix in figure 8.1.6. The figures show that the model predicts correctly about two thirds of the time.

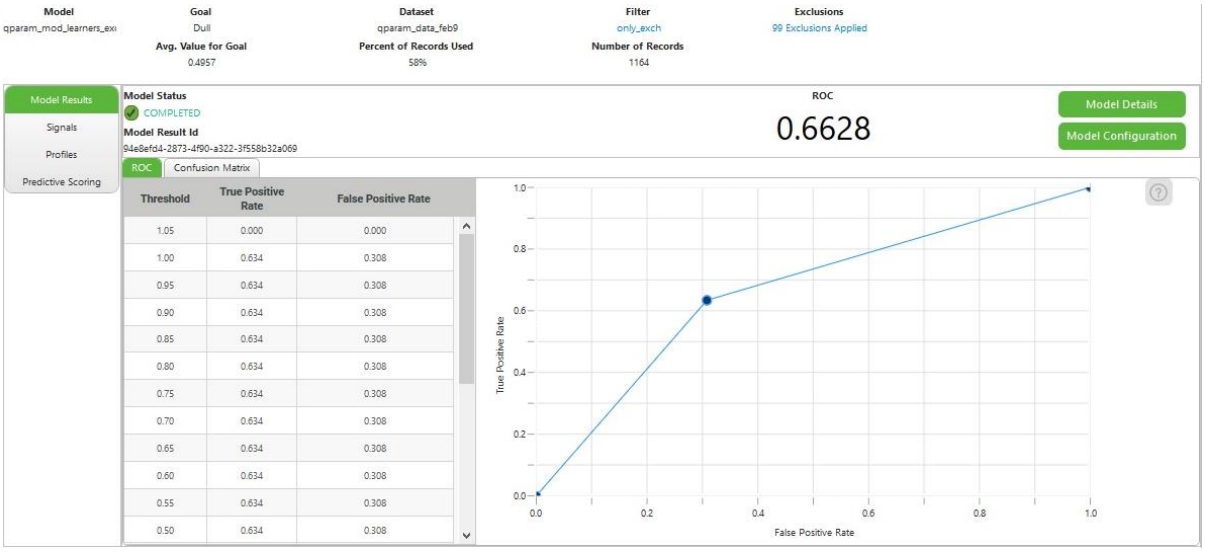

Figure 8.1.5: ROC curve for model validation of a model using Logistic Regression, Random Forest and Neural Network as learners.

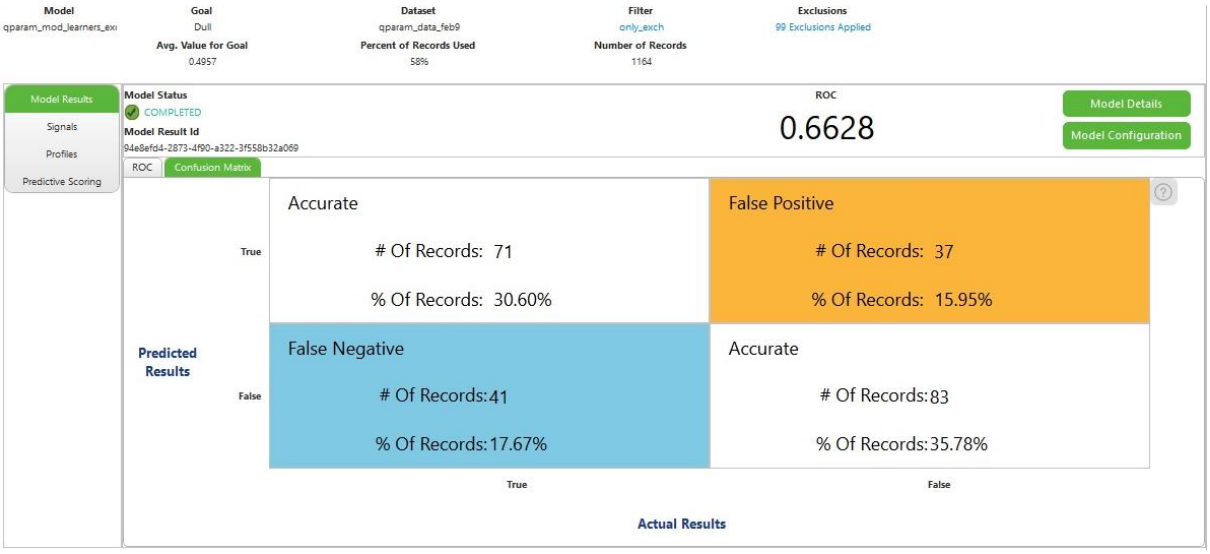

Figure 8.1.6: Confusion Matrix for model validation of a model using Logistic Regression, Random Forest and Neural Network as learners.

Model Name	Dataset Name	Goal	Filter	State	RMSE	RMSE Normalized	Pearson Correlation	MCC	Accuracy	Number of Records Validation Set
qparam_feb15_exch_g_spi_fft_04	qparam_data_feb9	Dull	only_exch	COMPLETED				0.4602	0.7292	240
qparam_feb15_exch_g_spi_fft_03	qparam_data_feb9	Dull	only_exch	COMPLETED				0.4024	0.7027	222
qparam_feb15_exch_g_spi_fft_02	qparam_data_feb9	Dull	only_exch	COMPLETED				0.2949	0.6473	224
qparam_feb15_exch_g_spi_fft_01	qparam_data_feb9	Dull	only_exch	COMPLETED				0.3868	0.6926	257
chucks_freqBatch_exch_04	chucks_freqbatch_feb15_accurate	Dull	exchange_only	COMPLETED				0.0489	0.5035	282
chucks_freqBatch_exch_03	chucks_freqbatch_feb15_accurate	Dull	exchange_only	COMPLETED				0.0461	0.5244	307
chucks_freqBatch_exch_02	chucks_freqbatch_feb15_accurate	Dull	exchange_only	COMPLETED				0.0434	0.5220	295
chucks_freqBatch_exch_01	chucks_freqbatch_feb15_accurate	Dull	exchange_only	COMPLETED				0.0202	0.5141	284

Figure 8.1.7: Overview of results for Dull model training. The bottom four uses the frequency batch data and the top four are filtered for Q-parameters indicated in the result from Signals in figure 8.1.1. Both look at sensors 4 and 6, which are mounted on the outside of the chucks.

Modeling was done using some of the parameters indicated by Signals, namely the G- and SPI-parameters for the sensors mounted on the chucks. These yield marginally better results, seen in figure 8.1.7. The bottom four modeling results in the same figure are for using frequency data, which yield results comparable to chance.

Model Name	Dataset Name	Goal	Filter	State	RMSE	RMSE Normalized	Pearson Correlation	MCC	Accuracy	Number of Records in Validation Set
dull_chucks_exch_zero_rpm_outliers_08	qparam_data_feb9	Dull	chucks_f2_zero_rpm_exch_outliers	COMPLETED				0.4032	0.7023	383
dull_chucks_exch_zero_rpm_outliers_07	qparam_data_feb9	Dull	chucks_f2_zero_rpm_exch_outliers	COMPLETED				0.3647	0.6745	381
dull_chucks_exch_zero_rpm_outliers_06	qparam_data_feb9	Dull	chucks_f2_zero_rpm_exch_outliers	COMPLETED				0.3584	0.6790	377
dull_chucks_exch_zero_rpm_outliers_05	qparam_data_feb9	Dull	chucks_f2_zero_rpm_exch_outliers	COMPLETED				0.3710	0.6814	408
dull_chucks_exch_zero_rpm_outliers_04	qparam_data_feb9	Dull	chucks_f2_zero_rpm_exch_outliers	COMPLETED				0.3545	0.6779	149
dull_chucks_exch_zero_rpm_outliers_03	qparam_data_feb9	Dull	chucks_f2_zero_rpm_exch_outliers	COMPLETED				0.2056	0.5822	146
dull_chucks_exch_zero_rpm_outliers_02	qparam_data_feb9	Dull	chucks_f2_zero_rpm_exch_outliers	COMPLETED				0.5036	0.7520	125
dull_chucks_exch_zero_rpm_outliers_01	qparam_data_feb9	Dull	chucks_f2_zero_rpm_exch_outliers	COMPLETED				0.2585	0.6333	150

Figure 8.1.8: Models trained with different amounts of data for the goal Dull. Models 05 through 08 have more data and more consistent accuracy. The models are trained using Q-parameters for the chucks (sensors 4 and 6) and are filtered to include entries when the motor is running, entries with only samples close to knife exchange and removing outliers in the parameters.

In figure 8.1.8, modeling is done with the same configuration but the bottom four models are with less data. The modeling process uses one of each learner and then decides on which produces the best results. Figure 8.1.9 shows the ROC curve for one of the trainings done with the larger quantity of data and figure 8.1.10 shows the confusion matrix for the same case.

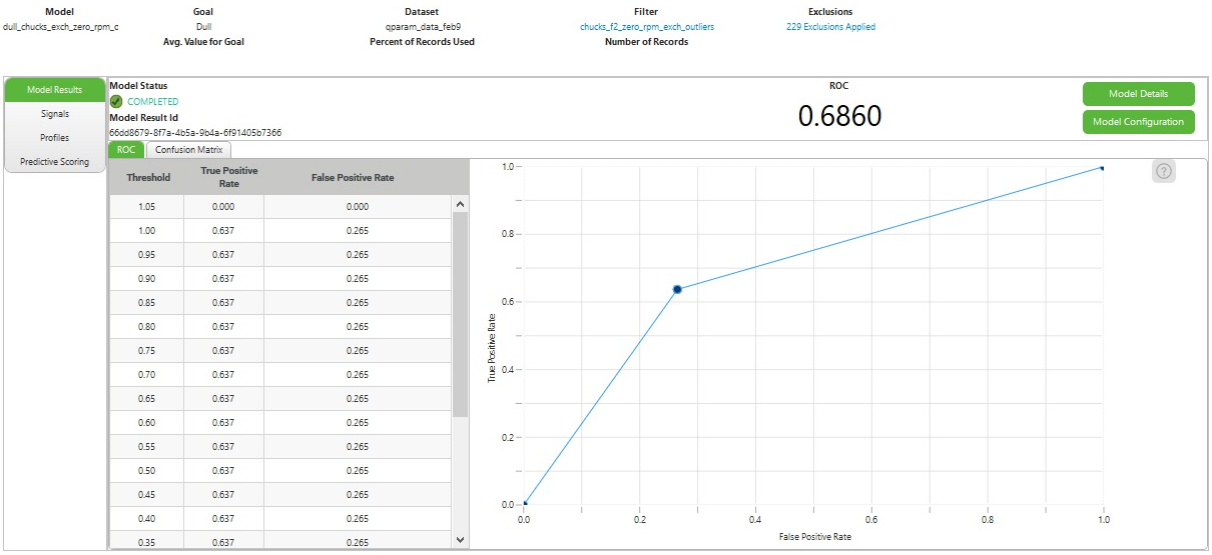

Figure 8.1.9: ROC curve for model 05 in figure 8.1.8. The model is trained using Q-parameters for the chucks, (sensors 4 and 6) and is filtered to include entries when the motor is running, entries with only samples close to knife exchange and removing outliers in the parameters.

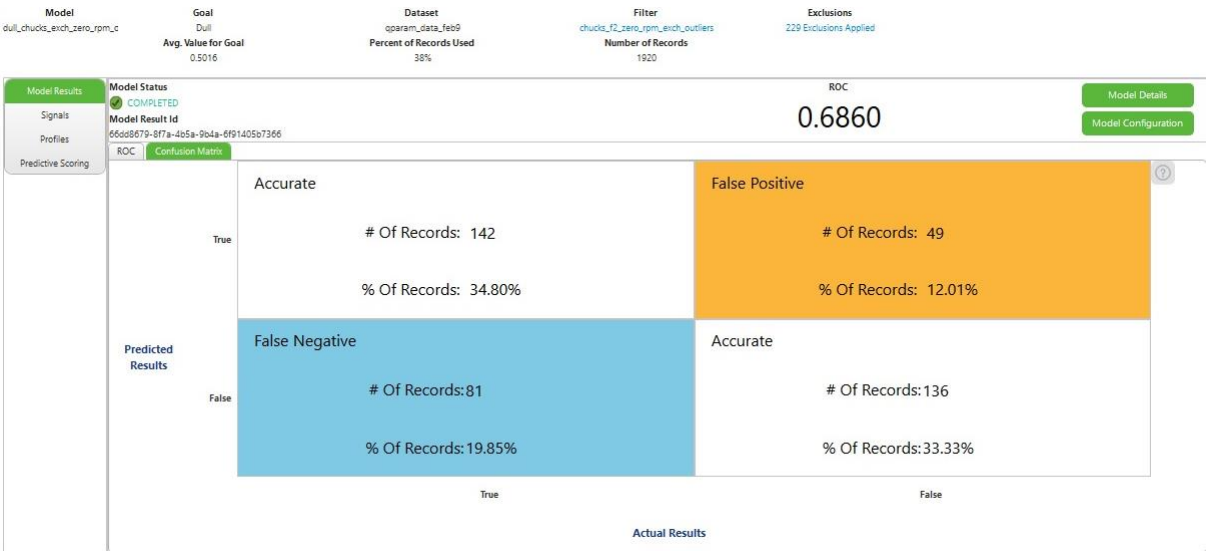

Figure 8.1.10: Confusion Matrix for model 05 in figure 8.1.9. The model is trained using Q-parameters for the chucks (sensors 4 and 6) and is filtered to include entries when the motor is running, entries with only samples close to knife exchange and removing outliers in the parameters.

8.2 Q-parameter Prediction with ThingWorx

Because predictions on knife dullness with high accuracy appears to not be possible with the current set up, some predictions were also done on Vikon's Q-parameters in order to evaluate the statistical and machine learning applications more thoroughly. This changed the goal from Boolean to continuous, meaning that the comparison metrics and displayed results appear differently.

Q-parameters ending in G are in the gravitational constant g. V-parameters and SPI-parameters are in speed mm/s. CREST, KURT and SKEW are dimensionless measures.

Model Name	Dataset Name	Goal	Filter	State	RMSE	RMSE Normalized	Pearson Correlation	MCC	Accuracy	Number of Records in Validation Set
Qparam_mp03_TOTV_Wmp04_Predict_no_V_01	qparam_data_feb9	mp03_fr2_TOTV	all_data	COMPLETED	0.4432	0.0551	0.8628			370
Qparam_only_mp03_SPI1_Predict_no_SPI_01	qparam_data_feb9	mp03_fr2_SPI1	all_data	COMPLETED	0.0027	0.0553	0.9665			422
Qparam_only_mp03_TOTV_Predict_no_V_01	qparam_data_feb9	mp03_fr2_TOTV	all_data	COMPLETED	0.4477	0.0557	0.8477			428
Qparam_only_mp03_TOTG_Predict_no_G_01	qparam_data_feb9	mp03_fr2_TOTG	all_data	COMPLETED	0.0157	0.0231	0.9877			394

Figure 8.2.1: Model results for predictions using Q-parameters. From the bottom these are: Total Vibration Acceleration TOTG on sensor 1 using all other sensors and their Q-parameters. Total Vibration Acceleration TOTG on sensor 4 (measuring point 3) using only Q-parameters from the same sensor and no other G-parameters. Total Vibration Velocity TOTV on sensor 4 (measuring point 3) using only Q-parameters from the same sensor and no other V-parameters. SPI1 on sensor 4 (measuring point 3) using only Q-parameters from the same sensor and no other SPI-parameters. And at the top, Total Vibration Velocity TOTV on sensor 4 (measuring point 3) using only Q-parameters from sensors 4 and 5 (measuring point 3 and 4) and no other V-parameters.

8.2.1 Total Acceleration Prediction

Predictions where done using total vibration acceleration, TOTG, as the goal. Signals showing important features is displayed in figure 8.2.2. Note the high predictive strength and the important features of listed parameters 3 through 5. Important features 1 and 2 are components of TOTG and therefore not very useful for testing predictive modeling. Figure 8.2.3 shows model results for TOTG for the same sensor but using only features from this measuring point and excluding G-parameters. Note the accuracy in the blue rectangle at the bottom of the figure.

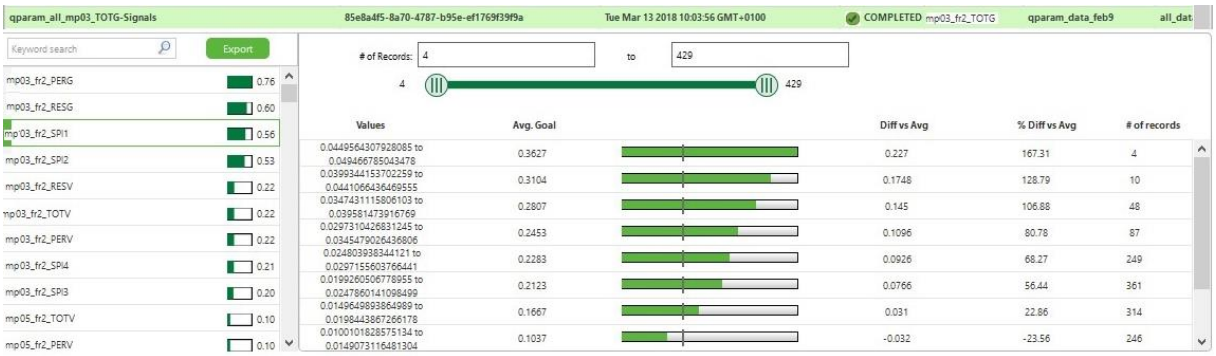

Figure 8.2.2: Signals for TOTG of sensor 4 (measuring point 3). The most important features for the goal are listed in the lower left frame. Note that the top two fields are components of the total acceleration.

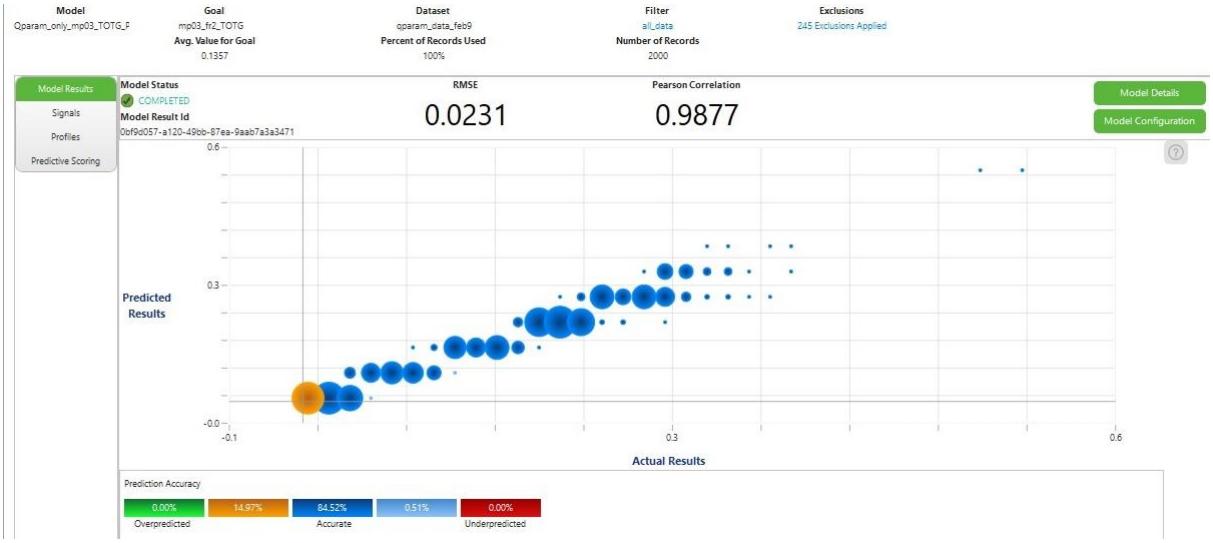

Figure 8.2.3: Scatter plot for model results of model using Total Vibration Acceleration TOTG on sensor 4 (measuring point 3) as goal. Model training is done using only Q-parameters from the same sensor and no other G-parameters. Learners used are Linear Regression, Decision Tree, Gradient Boosting Machine and Neural Net.

The TOTG model was published to Analytics Manager and Scoring Jobs where done with the use of the Analytics Provider ThingPredictor. Results for the predictions and the parameters used can be seen in table 8.2.1. Table 8.2.2 lists important fields for the scoring, which can be compared with figure 8.2.2.

RPM	TOTV	PERV	RESV	CREST	KURT	SKEW	SPI1	SPI2	SPI3	SPI4		Actual TOTG	Predicted TOTG
734.6	2.77	2.48	1.24	3.35	0.056	-0.003	0.04	0.004	0.002	0.0008		0.284	0.293
696.1	1.30	1.04	0.78	3.09	-0.33	-0.06	0.02	0.002	0.001	0.0005		0.161	0.145
0	0.87	0.77	0.42	4.05	2.68	0.296	0.004	0.0002	0.0001	0.00009		0.026	0.030
465.4	1.13	0.95	0.61	4.67	2.87	-0.89	0.01	0.001	0.0009	0.0006		0.080	0.093
785.6	1.55	1.19	0.996	3.39	0.07	0.09	0.02	0.003	0.002	0.0009		0.197	0.190

Table 8.2.1: Showing Analytics Manager Predictive Scoring Job results for five samples of Q-parameters. Total Vibration Acceleration TOTG is the predicted parameter. Acceleration components have been excluded for a more challenging prediction.

Actual TOTG	Predicted TOTG		Important Field 1 Name	Important Field 1 Weight	Important Field 2 Name	Important Field 2 Weight	Important Field 3 Name	Important Field 3 Weight
0.284	0.293		SPI2	0.273	SPI1	0.228	SPI3	0.200
0.161	0.145		SPI2	0.287	SPI1	0.247	SPI3	0.213
0.026	0.030		SPI2	0.324	SPI1	0.280	SPI3	0.216
0.080	0.093		SPI2	0.300	SPI1	0.262	SPI3	0.217
0.197	0.190		SPI2	0.279	SPI1	0.241	SPI3	0.209

Table 8.2.2: Showing Analytics Manager Predictive Scoring Job Important Features. The most significant features for the scoring is listed along with their percentage weights showing their influence on the goal.

8.2.2 Total Velocity Prediction

Predictions were also made by looking at the total vibration velocity, TOTV, of sensor 4 (measuring point 3) using only values from that sensor.

Figure 8.2.4 shows the results from Signals done with total vibration velocity, TOTV, done on sensor 4. Note that the predictive strength is lower than when comparing to acceleration in section 8.2.1 for parameters that are not components of the same.

Figure 8.2.5 shows the model results for prediction on the TOTV parameter of the same sensors. Note the accuracy in the blue rectangle at the bottom of the figure.

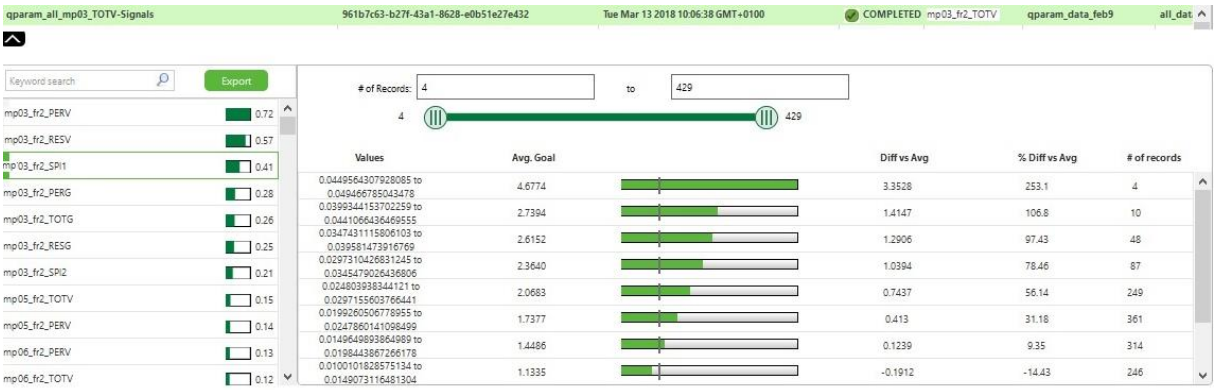

Figure 8.2.4: Signals for TOTV of sensor 4 (measuring point 3). The most important features for the goal are listed in the lower left frame. Note that the top two fields are components of the total velocity.

Figure 8.2.5: Scatter plot for model results of model using Total Vibration Velocity TOTV on sensor 4 (measuring point 3) as goal. Model training is done using only Q-parameters from the same sensor and no other V-parameters. Learners used are Linear Regression, Decision Tree, Gradient Boosting Machine and Neural Net.

8.3 Dull and Q-parameter prediction with CSense

The first analysis done in Proficy Troubleshooter is the calculation of a correlation matrix. The correlation matrix for all the features is quite large. For this reason, only a part of it will be presented here. The matrix is calculated from all the chosen features in the data and is therefore not directly dependent on a chosen goal. The correlation matrix for sensor 4 (measuring point 3) can be seen in table 8.3.1. Most interesting is looking at the correlations of TOTG and TOTV to the other features and comparing these with the results from Signals in figures 8.2.2 and 8.2.4. They show high correlations for the features that are ranked high in predictive strength in Signals.

	CREST	KURT	PERG	PERV	RESG	RESV	RPM	SKEW	SPI1	SPI2	SPI3	SPI4	TOTG	TOTV
CREST	100.00	60.93	29.75	3.87	34.08	5.50	26.80	-2.04	26.16	26.22	28.83	25.04	31.49	4.34
KURT	60.93	100.00	-2.77	-1.37	7.28	10.99	-12.62	-38.49	2.65	-7.74	-5.48	-7.51	0.92	2.77
PERG	29.75	-2.77	100.00	78.59	97.36	70.81	91.96	8.35	93.77	94.65	94.32	89.77	99.66	77.53
PERV	3.87	-1.37	78.59	100.00	74.29	90.24	65.34	-1.14	86.43	67.13	67.34	63.79	77.56	99.01
RESG	34.08	7.28	97.36	74.29	100.00	71.47	87.40	2.31	90.75	92.95	93.56	87.83	98.90	74.81
RESV	5.50	10.99	70.81	90.24	71.47	100.00	57.49	-6.92	77.95	63.80	62.29	60.99	71.51	95.37
RPM	26.80	-12.62	91.96	65.34	87.40	57.49	100.00	17.41	78.55	94.51	92.23	92.03	90.83	63.94
SKEW	-2.04	-38.49	8.35	-1.14	2.31	-6.92	17.41	100.00	-1.69	14.04	11.78	14.36	6.10	-3.23
SPI1	26.16	2.65	93.77	86.43	90.75	77.95	78.55	-1.69	100.00	80.10	80.56	73.21	93.29	85.35
SPI2	26.22	-7.74	94.65	67.13	92.95	63.80	94.51	14.04	80.10	100.00	97.72	96.51	94.58	67.30
SPI3	28.83	-5.48	94.32	67.34	93.56	62.29	92.23	11.78	80.56	97.72	100.00	96.72	94.61	66.96
SPI4	25.04	-7.51	89.77	63.79	87.83	60.99	92.03	14.36	73.21	96.51	96.72	100.00	89.60	64.07
TOTG	31.49	0.92	99.66	77.56	98.90	71.51	90.83	6.10	93.29	94.58	94.61	89.60	100.00	77.06
TOTV	4.34	2.77	77.53	99.01	74.81	95.37	63.94	-3.23	85.35	67.30	66.96	64.07	77.06	100.00

Table 8.3.1: Correlation matrix for the Q-parameters of sensor 4 (measuring point 3). Correlation with TOTG has been highlighted, specifically for first, second and third quartile of the noise vibration spectrum.

8.3.1 Knife Dullness Prediction

Modelling was done with Dull as a target. Results for two different filters can be seen in figures 8.3.1 and 8.3.2. Note the very low accuracy of the non-linear model. The rules score higher, seen in the lower right frame. CSense automatically divides the goal into interval and here Normal means Dull = false (or 0) and High means Dull = true (or 1).

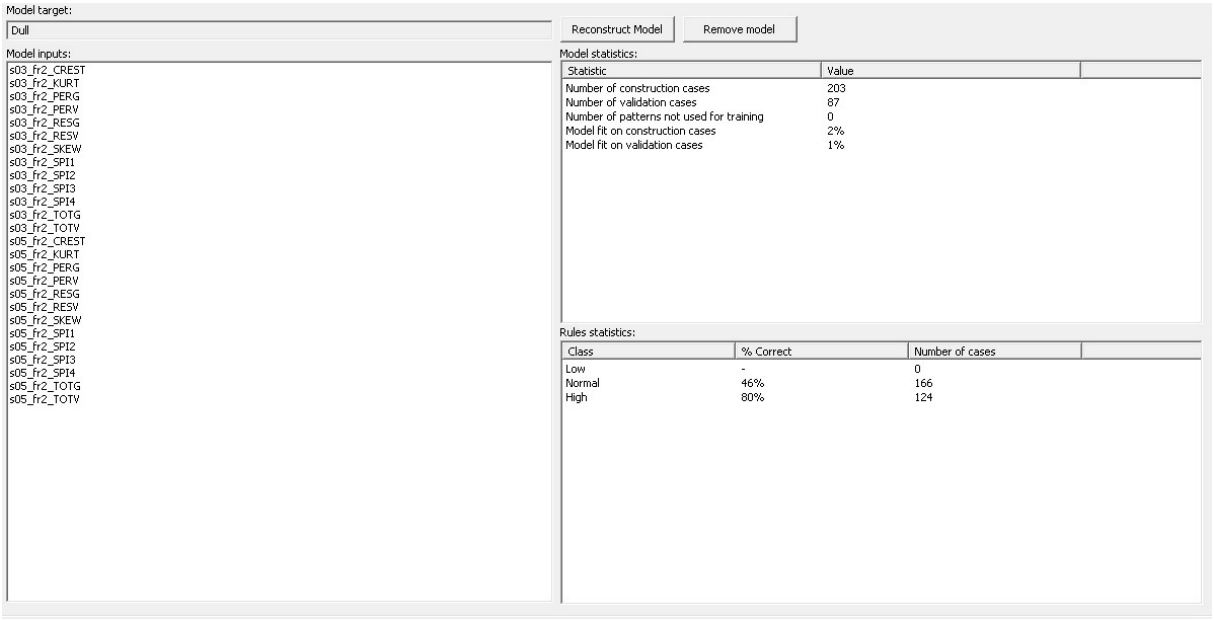

Figure 8.3.1: Modeling results for Dull as target and filtering out entries where RPM is zero. Model uses the sensors mounted on the outside of the chucks, sensors 4 and 6 (measuring points 3 and 5). Note model fit in the upper right frame.

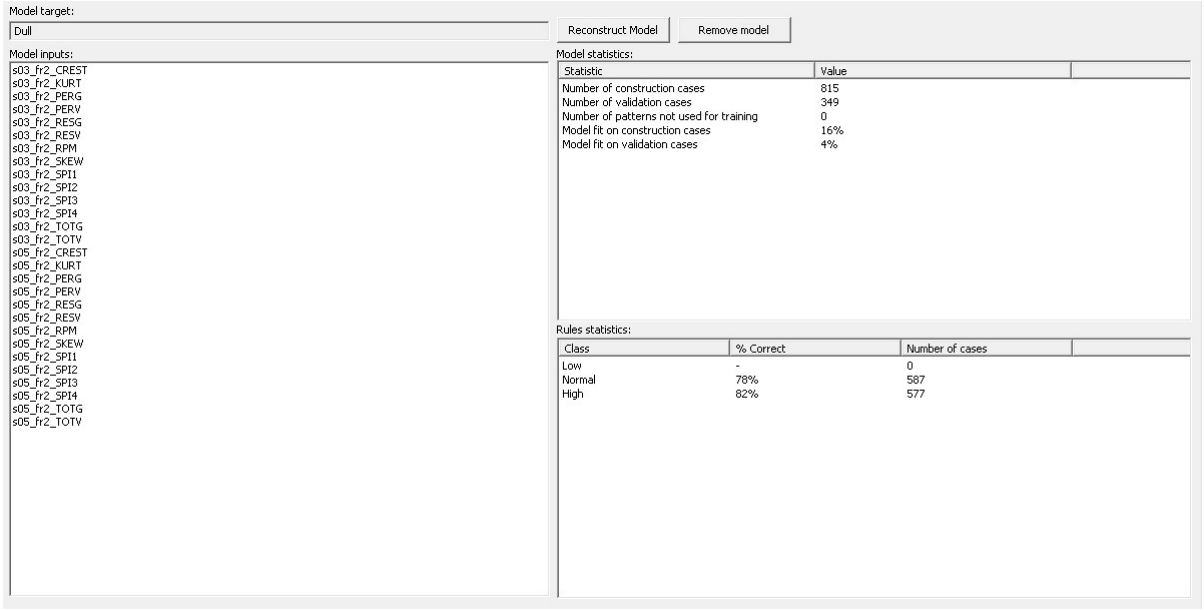

Figure 8.3.2: Modeling results for Dull as target and filtering for entries where the blade was exchanged. Model uses the sensors mounted on the outside of the chucks, sensors 4 and 6 (measuring points 3 and 5). Note model fit in the upper right frame.

8.3.2 Q-Parameter Prediction

A Decision Tree for TOTG of sensor 4 (measuring point 3) can be seen in figure 8.3.3. The limits are set as Low = 0.125, High = 0.221 and Normal in between.

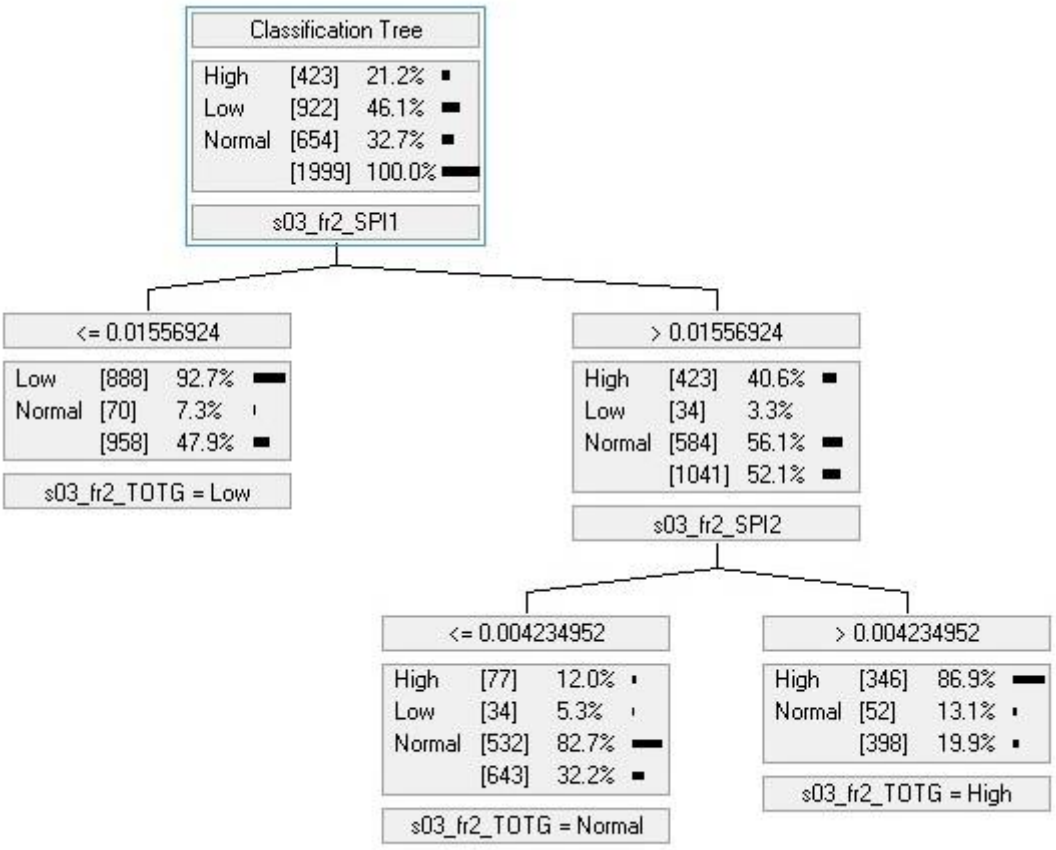

Figure 8.3.3: Decision Tree for TOTG of sensor 4 (measuring point 3) as the target. SPI1 and SPI2 are the variables that are most significant for the value of TOTG.

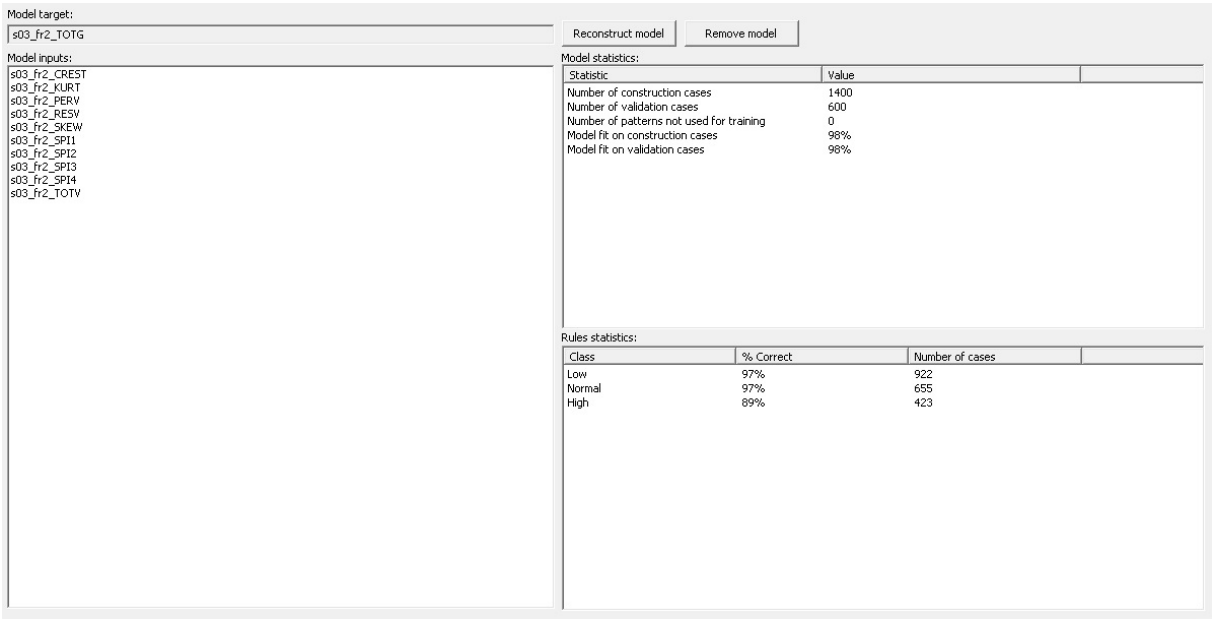

Figure 8.3.4: Modeling results for TOTG as target for sensor 4 (measuring point 3). Note the high model fit in the upper right frame.

Figure 8.3.4 shows modeling results for using TOTG of sensor 4 (measuring point 3) as the target. In figure 8.3.5 the same model's output and actual value of the target TOTG is seen in comparison.

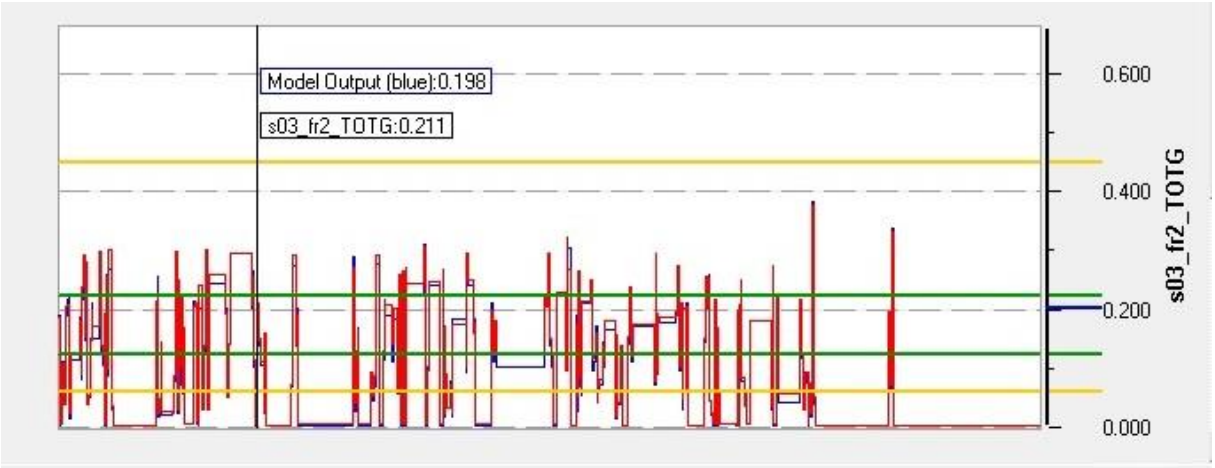

Figure 8.3.5: Showing model output (blue) and actual value (red) of TOTG as result of the model in figure 8.3.4.

Modeling was also done with TOTV as target for the same sensor. Results from this can be seen in figure 8.3.6 and the same comparison of the model output and actual value of the target is seen in figure 8.3.7.

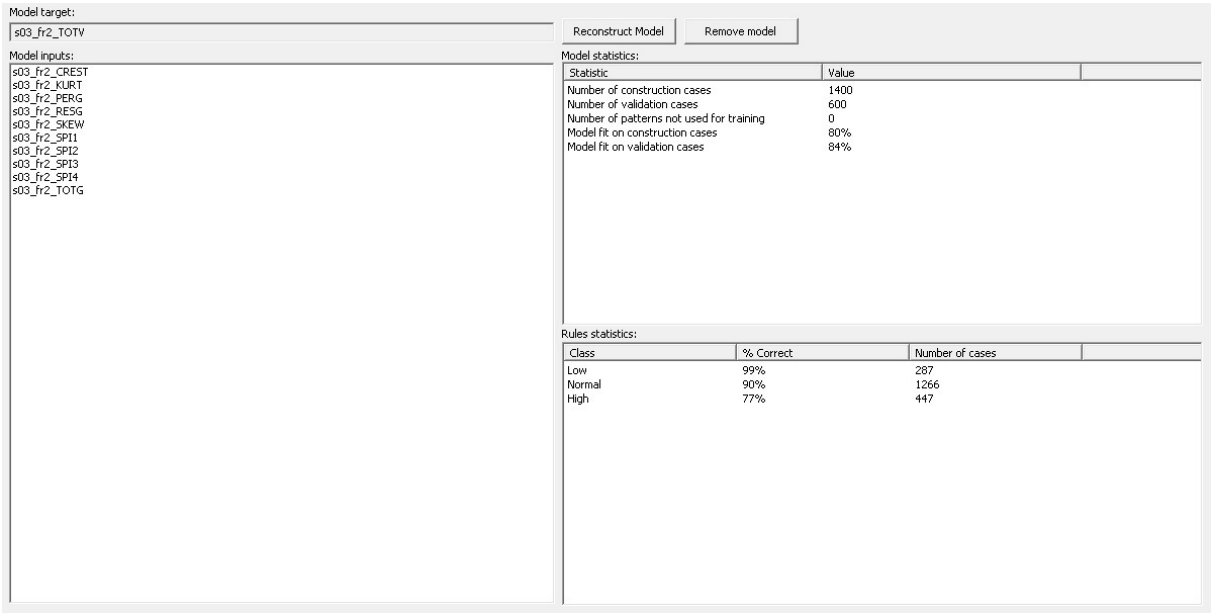

Figure 8.3.6: Modeling results from using TOTV for sensor 4 (measuring point 3) as a target. Model fit is seen in the upper right frame.

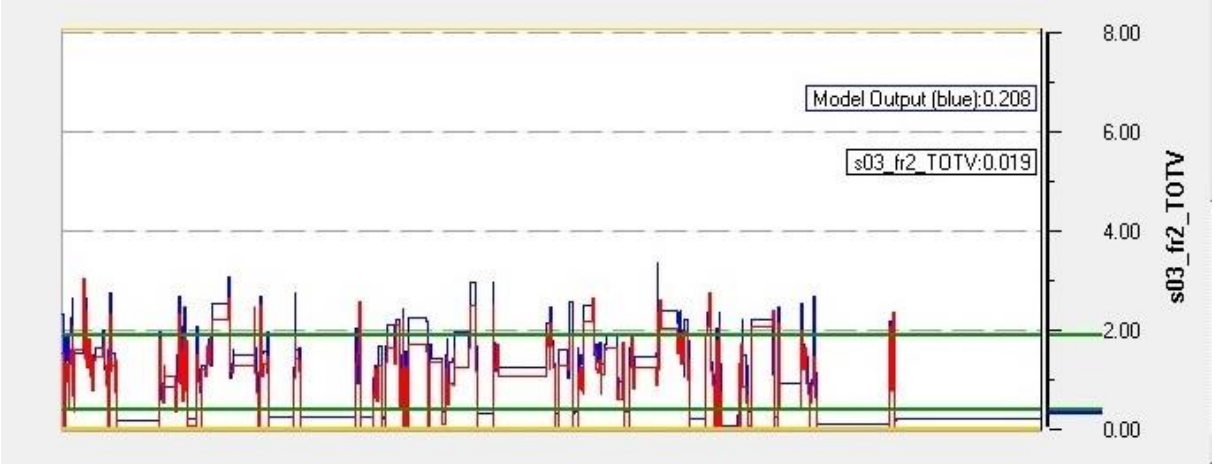

Figure 8.3.7: Showing model output (blue) and actual value (red) of TOTV as result of the model in figure 8.3.6.

9 Discussion

9.1 Knife Prediction

As is seen in the results, chapter 8, the prediction of whether the knife is dull or not does not produce adequate results. Several different learners and filters have been tried. Looking at the constellation that produced the best results, figures 8.1.9 and 8.1.10, an accuracy of about 0.68 for ROC is achieved. This implies that it guesses incorrectly about one third of the times and this is not good enough for online commercial use. Figure 8.1.10 implies that the model has more difficulty predicting when the knife is sharp. On the other hand, the results are quite similar in figure 8.1.6 indicating that this might not be that significant. The ROC curves in figures 8.1.5 and 8.1.9 do not go as quickly as desired to the north part of the graph.

Looking at important features for the prediction of dullness (figure 8.1.1) it is interesting to note that vibration acceleration is deemed significant. Most significant here is PERG, which is the part of the acceleration signal that is stationary and with a recurring pattern. It is sensitive to vibration in the high frequency range. That this parameter is deemed important seems logical since it is expected that vibration from dullness would increase slowly. Also, low frequency friction noise, SPI1, and high frequency friction noise, SPI4, are important. The fact that different ends of the spectra are implicated may suggest that one is occurring when the knife is sharp and the other when it is dull. This is unfortunately not possible to verify from the results given in Analytics.

Using frequency data was even less successful, as can be seen in figure 8.1.7. This might be due to the batch-like nature of the data. As noted, the frequencies with the top amplitude are recorded for each measurement. They are not recorded in any frequency bands. Furthermore, csv format does not allow easily connecting of frequencies and amplitudes in pairs. Filtering on low amplitude was not successful. This was due to the semi-automatic nature of Analytics Builder's filter function, which removed to many entries for the low amplitude interval.

When comparing with CSense results for Dull as target (figures 8.3.1 and 8.3.2), ThingWorx Analytics is much more successful. This might be due to Analytics being more configured for Boolean goals or perhaps those algorithms suit the data better. It might also be that the data should undergo more preparation in Proficy Troubleshooter, but no obvious solution presents itself. For the reasons given below it is, however, not prudent to make any certain conclusion about the knife predictions.

Looking at how often samples are taken, this is not likely to have been a problem. Since the knife events are somewhat spread out more frequent measurements are not needed. A sample every third minute is likely adequate when the knife events are about three hours apart. The goal is not to detect damages but gradual dullness. When the samples are taken are more of a problem. As noted in section 3.2 the peeling cycle has several operational modes. The one that is of interest is the mode where actual peeling is done, and the knife is in constant contact with the log. Since there is currently no way of knowing which mode the data represents, predictions are made harder. Looking at RPM only tells if the machine is running or not, so filtering on this is likely not enough. A further idea was to include the number of produced veneer meters into the data and then try to model on this. The meter data was, however, included too late in the project for there to be enough data.

At a late stage in the project the possibility to have a conversation with the company that makes the lathe was presented. They had done vibration measurements of the peeling process during research and development. The goal was then to develop a construction that would prevent the knife from vibrating due to deteriorating sharpness. This will naturally make it much more difficult to use

vibrations for this type of prediction and is probably one of the main reasons it does not seem to work. It is of course possible that they have not managed to eliminate vibrations to this extent. There is support in the results for this last assumption. Even though the knife predictions do not reach as high an accuracy as desired, guessing correctly two out of three times implicates that there is a discernable pattern.

Another possible reason for the insufficient prediction is the placements of the sensors. Initially, it was believed that several sensors where fitted on or at least very close to the knife. This was, however, not the case. The sensor placement is optimized for other measurements concerning bearing and machine part damages. Combining this with the goal of reducing dullness vibration it is perhaps not reasonable to expect better results.

One last thing to consider is the assumption of dullness. Since there is no trial data for when the knife is confirmed to be dull the assumption that it is dull before an exchange had to be done. This assumption might not be entirely true and it is possible that the distance set is well before the knife is measurably dull. One of the operators mentioned during the site visit that at a previous production plant he worked at, the limit for exchanging the veneer lathe knife was 30'000 produced meters, which is twice the length compared to the exchange limit in this case. This casts further doubt on the assumption that the knife actually is dull when exchanged.

9.2 Q-parameter Prediction

The predictions on the Q-parameters work a lot better than the Dull predictions. This is natural since the vibration measurements are likely connected to some degree, as implicated by the correlation matrix from CSense (table 8.3.1) and Analytics Builder Signals (figures 8.2.2 and 8.2.4). The predictive strength implicated in the Signals figures are much higher than for Dull as the goal. It is quite interesting that the modeling for TOTG is much more successful than for TOTV, figure 8.2.3 vs 8.2.5. This is a bit unexpected since the sensors are vibration velocity and therefore one would expect this to have an effect in all the parameters. This means that TOTG and not TOTV is the derived parameter. Signals, however, indicate higher predictive strengths for the features in relation to TOTG. When comparing to the results of Signals for TOTG and TOTV, the better modeling outcomes for TOTG is logical. For TOTV, perhaps overfitting to the data might be a problem.

When looking at important features, Signals in figure 8.2.2 and scoring results in table 8.2.2, it is prominent that the two features with the highest predictive strength are also noted as the two most important fields for the scoring, if in reverse order. The reversion is explained by the fact that the predictive strengths and weights are close to each other.

In comparison to modeling done in CSense, ThingWorx Analytics produces less successful results for Q-parameter prediction. Analytics modeling is perhaps more directed at binary and categorical goals, since this is the type of goal used in Analytics Builder tutorials. CSense was much more successful in relation to that type of goal. This is most likely due to CSense being very useful for modeling signals and using TOTG and TOTV as target can be seen as modeling them as signals from the other parameters. When it comes to important features the two programs are in agreement, which can be seen in scoring result for important features in table 8.2.2 and in the correlations for TOTG in table 8.3.1.

9.3 ThingWorx Evaluation

The claim that ThingWorx Analytics "…eliminates the need for developer or user expertise in data modeling, complex mathematics, or machine learning" (PTC 2017c p2) is not entirely true. Processing

the data requires some data science knowledge, but arguably process experience is perhaps even more important. Furthermore, some understanding of the machine learning and statistical algorithms used is needed to configure modeling properly. It does, however, require less knowledge than CSense and certainly less than most numerical and statistical computation tools.

The integration between the main ThingWorx platform and ThingWorx Analytics is quite good but still needs more development, especially for the step when one wants to export data from ThingWorx Foundation to Analytics. Ideally, ThingWorx Foundation should be able to supply Analytics Builder with the data and data configuration directly. This could be done via the Data Storage Containers using functionality in the composer, such as Services. Furthermore, Analytics Builder should include more possibilities for preprocessing the data. Visualization of the data can be handled by the main ThingWorx platform.

Once the model is trained and published the integration is much better. The fact that there are Things and Thing Templates that can be auto-generated for the connection between Analytics Manager and Foundation, speeds up the process significantly. The need for the ThingPredictor for scoring jobs is, however, a bit cumbersome. This functionality should optimally be included in the Manager and not run in a standalone process. It is also a great benefit that all sorts of analysis providers can be connected to ThingWorx. Another plus is that once a model has been developed a connection to Analytics Server is not necessary. This allows ThingWorx developers to run scoring jobs on the model and use this in their application by running ThingWorx Foundation and the Analytics Extension alone.

When it comes to visualization and process overview, CSense provides a lot more than ThingWorx out of the box. It is possible to achieve the same visual aids and variable manipulation in ThingWorx, but this has to be developed by the application developer. Conversely, this of course allows for a much more tailored process monitoring application. As always there is a tradeoff between flexibility and preexisting solutions.

The failed knife prediction cannot be blamed on ThingWorx Analytics or CSense. As noted there are many other more significant reasons for not achieving high enough model accuracy. Looking at the TOTG prediction it seems ThingWorx can be a good tool for process monitoring if data with proper labels are available. Labeling here refers to what type of operational mode or situation the data is taken from. This will establish a basis for comparison. When certain patterns appear, the program can tell where the process is and, more importantly, where it is going. The old saying “garbage in garbage out” is important to keep in mind. It is not reasonable to expect meaningful analytics results if you do not have the right data. CSense does obtain a higher accuracy for the Q-parameter prediction, but with more preprocessing of the data and more tweaking of learner and modeling parameters this gap will close.

9.4 Future Development

One of the most obvious improvements that could be made is putting a sensor on the floating knife carrier. This would give sensor readings as close as possible to the point of interest. Putting sensors directly on the knife is probably not feasible due to the process nature. They could be damaged by the peeling and logs entering and leaving the lathe. Also, there is a lot of dirt and wood chips this close to the knife.

The fact that the operational state of the lathe is not registered with the samples is a major problem. If it was possible to filter out only entries where actual peeling is being done it is likely that a higher

accuracy could be achieved. This would give data that primarily holds entries where the knife is in contact with the log. This was discussed during the project, but unfortunately the measuring system was not able to sync the PLC's signals for operational mode with the sensor data. The PLC readings would only be useful if it is possible to make sure that they are taken at the same time as the sensor readings.

The abovementioned problem of assuming when the knife is dull is another source for error. To produce optimal models, specific trial runs should ideally be done where data can be recorded for all interesting operational modes. When dealing with a production line that runs 24 hours and 7 days a week, this is of course not easy to organize. One also has to consider damages and costs to the machine being used in undesirable operational modes. There is a possible workaround concerning this. If it was possible to somehow measure the sharpness of the knifes directly, this could be done on the knifes that have been removed after an exchange. This could then be entered into the knife event data and perhaps also be related to the produced meters signal.

It is, however, worth considering if the prediction method is viable for knife maintenance. Since the wood type used is fairly constant the use of produced length is perhaps the best in relation to cost-benefit. Other types of analysis, such as on the produced veneer, might also prove more successful for knife state predictions. If the desire is to automate the whole process more development of the manual aspects of the peeling needs to be handled. Using produced distance is, however, not really a hindrance to this.

When it comes to the IIoT platform ThingWorx, and ThingWorx Analytics in particular, it is constantly being refined by PTC. This project uses version 8.1 but in the moment of writing a version 8.2 and 8.2.1 has been released. The area of connected industrial devices is only likely to grow and this type of process monitoring with it.

10 Conclusions

The study exemplifies many of the challenges and opportunities of using industrial data for monitoring and predictive maintenance. Trying to predict the status of the veneer knife with high accuracy ultimately proved to be a failure, but the problem lies more with the data collection than the tools and methods of analysis. It is important to both collect data from the right place and make sure that the data contains the information needed. For the lathe neither of these are really fulfilled. The measurements are done relatively far away from the desired point of monitoring, and type of measurements taken are not ideal for the monitoring, due to the design of the machine. The design of the lathe is of course a huge hindrance to the predictions. Nevertheless, the reached accuracy of 0.68 would imply more than random guessing and the improvements stated in section 9.4 are likely to lead to more successful results.

The tests done on the vibration parameters do imply that this type of setup can be successfully used for monitoring. More comprehensive and properly labeled data is needed, but it is very likely that ThingWorx and ThingWorx Analytics could be used to monitor a process and give predictions concerning where it is headed.

Another aspect is having and using the right tools. When it came to the true or false classification of whether the knife is dull or sharp, Analytics performed a lot better. On the other hand, when trying to model one of the Q-parameters as a signal CSense was more successful. This is especially true for the velocity vibration parameter. Quite naturally, all tools for analysis have strengths and weaknesses and it is up to the developers using these to apply them in the right places.

Despite the inability to make knife dullness predictions as accurately as desired, the study shows the potential for applying IoT to industrial problems. Being able to not only monitor where a process is at currently, but also use that information to discern where the process is going is very valuable. When smart connected machines can ‘talk’ to each other the possibility for automation grows far beyond what is conceivable when simply displaying parameter values to an operator. The Industrial Internet of Things is still young but will only get larger and is predicted to play an important part in the next industrial revolution.

11 Bibliography

Apache Software Foundation (2018), *Apache Tomcat*
http://tomcat.apache.org/ [2018-03-28]

Astakhov, V. P.,(2010) *Geometry of Single-Point Turning Tools and Drills – Fundamentals and Practical Applications*, Springer-Verlag London Limited, New York, USA.

Biron, J. & Follett, J. (2016) *Foundational Elements of an IoT Solution*, O'Reilly Media, Sebastopol, USA.

Canadian Wood Council (2018) *Laminated Veneer Lumber*
http://cwc.ca/wood-products/structural-composite/laminated-veneer-lumber/ [2018-01-26]

Chicco, D. (2017) 'Ten quick tips for machine learning in computational biology', *BioData Mining*.
https://biodatamining.biomedcentral.com/articles/10.1186/s13040-017-0155-3 [2018-03-12]

Fawcett, T. (2005) 'An introduction to ROC analysis', *Pattern Recognition Letters*, Volume 27, Issue 8, June 2006, Pages 861-874.

GE Intelligent Platforms (2010) *Proficy Troubleshooter*
https://www.geautomation.com/de/download/proficy-troubleshooter [2018-03-16]

GE Intelligent Platforms (2011) *Proficy Cause+*
http://www.geautomation.com/download/proficy-cause [2018-03-16]

General Electric (2016) *CSense 6.0 Datasheet*
https://www.ge.com/digital/sites/default/files/csense-from-ge-digital-datasheet.pdf [2018-03-16]

Ho, T. 'Random Decision Forests', *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Montreal, QC, Canada, 14–16 August 1995. pp. 278–282.

Howieson, D. (2003) *Vibration Monitoring: Envelope Signal Processing*, SKF Reliability Systems @ptitudeXchange, San Diego, USA. Available at: http://www.maintenance-engineering.eu/downloads/public/envelope%20bearing.pdf

Lacey, M. (1998) Yale Univeristy Department of Statistics *Linear Regression*,
http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm [2018-03-08]

Lutz, J. F. (1977). *Wood Veneer: Log Selection, Cutting and Drying*, Forest Products Laboratory, Forest Service, U.S. Department of Agriculture, Madison, WI, USA.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press

Marr, B. (2016) *Why Everyone Must Get Ready For The 4th Industrial Revolution*, Forbes Magazine. https://www.forbes.com/sites/bernardmarr/2016/04/05/why-everyone-must-get-ready-for-4th-industrial-revolution/#23781ef3f90b [2018-03-27]

Natekin, A. & Knoll, A. (2013) 'Gradient Boosting Machines, a Tutorial', *Front Neurorobot*, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3885826/ [2018-03-08]

NTI Audio (2018) *Fast Fourier Transform FFT*
https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft [2018-03-22]

Pennsylvania State University (2018) *Lesson 6: Logistic Regression*, https://onlinecourses.science.psu.edu/stat504/node/150 [2018-03-08]

Peysakhov, M. (2017) *Expert Session: ThingWorx Analytics Signals*, https://community.ptc.com/t5/ThingWorx-Expert-Sessions/Expert-Session-ThingWorx-Analytics-Signals/td-p/533744 [2018-03-07]

PTC (2017a) *ThingWorx Platform Product Brief*
Accessible at: https://www.ptc.com/-/media/Files/PDFs/ThingWorx/ThingWorx-Platform_Product-Brief_Jun-2017.pdf

PTC (2017b) *Expert Session: Decision Tree, ThingWorx Analytics Builder*, https://community.ptc.com/t5/ThingWorx-Expert-Sessions/Expert-Session-Decision-Tree-ThingWorx-Analytics-Builder/td-p/532273 [2018-03-08]

PTC (2017c) *ThingWorx Analytics Product Brief*
Accessible at: https://www.ptc.com/-/media/Files/PDFs/ThingWorx/ThingWorx-Analytics_Product-Brief_Jun-2017_FINAL.pdf

PTC (2018a). *ThingWorx*
https://support.ptc.com/help/thingworx_hc/thingworx_8_hc/ [2018-02-06]

PTC (2018b). *ThingWorx Analytics*

https://support.ptc.com/help/thingworx_hc/thingworx_analytics_8/ [2018-02-06]

PTC (2018c) *ThingWorx Edge*
http://support.ptc.com/help/thingworx_hc/thingworx_edge/ [2018-02-13]

Schapire R.E. (2003) The Boosting Approach to Machine Learning: An Overview. In: Denison D.D., Hansen M.H., Holmes C.C., Mallick B., Yu B. (eds) *Nonlinear Estimation and Classification. Lecture Notes in Statistics*, vol 171. pp 149-171. Springer, New York, NY, USA.

TechTarget (2018) IoT Agenda *Industrial Internet of Things*
http://internetofthingsagenda.techtarget.com/definition/Industrial-Internet-of-Things-IIoT [2018-02-06]

W3Schools (2018a) *Introduction to XML*
https://www.w3schools.com/xml/xml_whatis.asp [2018-04-11]

W3Schools (2018b) *JSON - Introduction*
https://www.w3schools.com/js/js_json_intro.asp [2018-04-11]

Wise, B. M. & Gallagher N. B. (1996) *The process chemometrics approach to process monitoring and fault detection*, Eigenvector Research, Manson, WA, USA.

Figures

Glosser (2013) *Colored Neural Network*, Wikimedia Commons
https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg [2018-03-08]

Manske, M. (2005) *Decision Tree Model*, Wikimedia Commons
https://commons.wikimedia.org/wiki/File:Decision_tree_model.png [2018-03-08]

PTC (2017a) *ThingWorx Platform Product Brief*
Accessible at: https://www.ptc.com/-/media/Files/PDFs/ThingWorx/ThingWorx-Platform_Product-Brief_Jun-2017.pdf

PTC (2018b) *Confusion Matrix,*
https://support.ptc.com/help/thingworx_hc/thingworx_analytics_8/#page/thingworx_analytics_8%2Fanalytics-builder-models-view-results.html%23 [2018-03-08]

PTC (2018b) *ROC Curve*,
https://support.ptc.com/help/thingworx_hc/thingworx_analytics_8/#page/thingworx_analytics_8%2Fanalytics-builder-models-view-results.html%23 [2018-03-08]

PTC (2018b) *Scatter Plot*,
https://support.ptc.com/help/thingworx_hc/thingworx_analytics_8/#page/thingworx_analytics_8%2Fanalytics-builder-models-view-results.html%23 [2018-03-08]

Sewaqu (2010) *Linear Regression*, Wikimedia Commons
https://commons.wikimedia.org/wiki/File:Linear_regression.svg [2018-03-08]