# On the Challenges of Real World Data in Predictive Maintenance Scenarios: A Railway Application

Sebastian Kauschke[1], Frederik Janssen[2] and Immanuel Schweizer[3]

[1] kauschke@ke.tu-darmstadt.de
[2] janssen@ke.tu-darmstadt.de
Knowledge Engineering Group & Telecooperation Group
Technische Universität Darmstadt, Germany
[3] schweizer@cs.tu-darmstadt.de
Telecooperation Group
Technische Universität Darmstadt, Germany

**Abstract.** Predictive maintenance is a challenging task, which aims at forecasting failure of a machine or one of its components. It allows companies to utilize just-in-time maintenance procedures instead of corrective or fixed-schedule ones. In order to achieve this goal, a complex and potentially error-prone process has to be completed successfully.
Based on a real-world failure prediction example originated in the railway domain, we discuss a summary of the required processing steps in order to create a sound prediction process.
Starting with the initial data acquisition and data fusion of three heterogeneous sources, the train diagnostic data, the workshop records and the failure report data, we identify and elaborate on the difficulties of finding a valid ground truth for the prediction of a compressor failure, caused by the integration of manually entered and potentially erroneous data.
In further steps we point out the challenges of processing event-based diagnostic data to create useful features in order to train a classifier for the prediction task. Finally, we give an outlook on the tasks we yet have to accomplish and summarize the work we have done.

## 1 Introduction

Predictive maintenance (PM) scenarios usually evolve around big machinery. This is mainly caused by those machines being both expensive and important for production processes of the company they are used in. A successful predictive maintenance process for a machine can help at preventing this, aid in planning for resources and material, and reduce maintenance cost and production downtime. In order to benefit from PM, a constant monitoring and recording of the machine status data is required.

 In: R. Bergmann, S. Görg, G. Müller (Eds.): Proceedings of the LWA 2015 Workshops: KDML, FGWM, IR, and FGDB. Trier, Germany, 7.-9. October 2015, published at http://ceur-ws.org

Usually, historical data is used to train a model of either the standard behaviour of the machine, or - if enough example cases have been recorded - a model of the deviant behaviour right before the failure. These models are then used on live data to determine whether the machine is operating within standard parameters, or - in the second case - if the operating characteristics are similar to the failure scenario. If the model is trained correctly, it will give an alarm in due time. An overview of various PM methods is given in [4].

In our example case, the machines are cargo trains. These trains are pulling up to 3000 tons of cargo, so a lot of parts are prone to deterioration effects. In this paper we will present a workflow which will help us discover the necessary information needed to use a classifier to predict a specific failure case, the main air compressor failure. It is relevant for pressured air that is used in many pneumatic applications in a train, e.g. for braking or opening/closing doors and occurred quite often in the two year period of the historical data we received. This allows us to have a fair amount of example instances to train the classifier upon.

This paper is organized as follows. Section 2 will give an introduction to the problems and challenges. In Sect. 3 we will introduce the data sources in detail, followed by the challenge of finding a suitable ground truth therein (Sect. 4). From there on we will elaborate on how to extract meaningful features (Sect. 5) and propose a labelling process (Sect. 6) until we conclude our findings in Sect. 7 and give an outlook on further work.

## 2 Problem Definition and Challenges

We want to predict the failure of the main air compressor in a complex system containing many components. Therefore we will build a predictive model using a supervised learning approach on historical data, and apply it to new(er) data.

We were supplied by *DB Schenker Rail AG*[4] with data from three distinct datasets, from which we are going to extract the exact points in time when the failures have happened: (i) the diagnostic data recorded on the trains, (ii) the maintenance activity data gathered in the workshops and (iii) the failure report data with information from the hotline regarding the time and cause of the failure. Furthermore, we will create features that are descriptive and discriminative for the compressor failure, label the instances and train a classifier that will give a warning before the actual failure happens. We will have to face the following challenges:

1. Deal with large amount of diagnostic data that has unusual properties, i.e., inhomogeneous data distribution[5].
2. Extract a valid ground truth from three given datasets to find out exactly when the compressor failures happened by searching for indications for that type of failure and recognizing unnecessary workshop layovers (i.e. through premature indication of failure by the diagnostic system).

[4] www.rail.dbschenker.de

[5] Most often failures cases of machines are extremely rare. However, extracting instances that describe the regular operation of the machine comes more or less for free, in predictive maintenance, we have to deal with a very skewed distribution of the classes.

3. Recognize errors, incompleteness and imprecision in the data and derive methods to deal with them.
4. Create meaningful features that emphasize the underlying effects that indicate an upcoming failure.
5. Set up a labelling process for time-series data that enables classifiers to anticipate impending failures early enough to allow for corrective reactions.
6. Define an evaluation procedure that maximises the outcome with respect to given criteria in order to achieve optimal prediction.

## 3 Data

In this section, we will give a short introduction to the three information sources that were available, and make assumptions about the quality and the completeness of the data. In comparison to our effort in 2014 ([1]), we were supplied with more data (span of two years) from different databases, which enables us to gather more precise information and tackle a variety of new issues. With this information we will then filter the occurrences of the compressor failure scenario and determine the exact date of the failures (Sect. 4).

### 3.1 Diagnostics Data

The diagnostics data is recorded directly on the train in the form of a logfile. It shows all events that happened on the train, from the manual triggering of a switch by the train driver to warning and error messages issued by the trains many systems.

We have access to data from a complete model range of train's, the class *BR185*. This class was built in the 1990s, so it has internal processing and logging installed, but the storage capacity is rather limited. Back then, predictive maintenance was not anticipated, and therefore the logging processes were not engineered for this application. This becomes abundantly clear, when we consider the steps necessary to retrieve the logfiles from the train. It has to be done manually by a mechanic, each time the train is in the workshop for scheduled or irregular maintenance tasks.

In the two years we are using as historic data, around 21 Million data instances have been recorded in a fleet of 400 trains.

**Diagnostic Codes and System Variables.** As already mentioned, the diagnostic data is in the form of a logfile. It consists of separate messages, each having a code to identify its type. When we refer to a diagnostic code, we usually mean a message with this specific code. In total there are 6909 diagnostic codes that can occur.

To each diagnostic code, a set of system variables is attached. Those are encoded as Strings. Since the whole system is event based, the variables are not monitored periodically, but only recorded when a diagnostic message occurs. Which variables are encoded is depending on the diagnostic message that was recorded. This implies that some variables will be recorded rarely and sometimes not for hours or days. Overall, there are 2291 system variables available: simple booleans, numeric values like temperature or pressure and system states of certain components.

The event-based nature makes handling values as a time series difficult, for some sort of binning has to be done in order to achieve regularly spaced entities. In our case, a bin size of one day was used. Especially when relying on attributes that involve temperature or pressure measurements it is impossible to create a complete and fine-grained time-series of their values, because they appear too sparsely.

There is one speciality about these diagnostic messages: They have two timestamps, one for when the code occurred first (*coming*), and one for when it went away (*going*). This is designed for status reporting messages that last a certain time. Most codes only occur once, so they do not have a timespan, others can last up to days. For example there is code 8703 (*Battery Voltage On*) which is always recorded when the train has been switched on, and lasts until the train is switched off again.

Because of the two entries *coming* and *going*, there are usually two measurements of a variable for each diagnostic code entry. To handle these variables correctly, we separate them from the diagnostic messages and use both values as a separate measurement.

### 3.2 Failure Report Data

The Failure Report Data contains information when a failure has been reported by a train driver. In the driving cabin there is the Multi-Function-Display (MFD), which shows warnings or error messages. When a critical problem occurs, the information in conjunction with a problem description is shown to the driver. If he is unable to solve the issue, he will call a hotline. The hotline operator will try to help find an immediate solution. In the case of an unsolvable or safety-critical problem, the hotline operator will schedule a maintenance slot for the train and organise the towing if necessary.

The information recorded here is the date of the hotline call, the stated reason of the caller, and the information if the issue had an impact on the overall train schedule, i.e. the train was delayed more than ten minutes.

The start and end date of the train being in the workshop for repairs will be added to this database afterwards (manually). In general, the textual description given by the train driver and hotline operator are free text inputs and not consistent. The easiest possible way of finding out instances of a failure type is to search these text descriptions for certain keywords. In our case, 159 compressor failures on 95 trains were recorded.

Since this data is added manually and the dates are filled in afterwards, there is no guarantee that it is correct in any way.

### 3.3 Workshop Data

Compared to the Failure Report Data, the Workshop data is gathered in a much more controlled environment. Every maintenance activity is recorded here, from the replacement of consumables up to the more complex repair activities. Each entry has a date stamp as well as an exact id to each activity predefined in a database. All activities are divided into systems, as well as tagged with a price. The information, if a certain action was ”corrective” or a ”scheduled replacement” is also available.

The correct tracking of the maintenance records is necessary for invoices, so it is plausible to assume that these are handled more carefully than the failure report data.

They are manually entered into the system and it can not be guaranteed that they resemble the exact activities that have been applied to the train.

### 3.4 Quality issues

All of the three datasets have an issue in common: they may not be complete (missing entries, descriptions or dates) or are filled with false values (wrong dates, typos that make it hard to find a keyword). Whether this is caused by human error, negligence or processes that do not cover enough details is not important. In any case, these problems need to be dealt with in a way that renders each dataset as useful as possible.

## 4 Finding the ground truth

Our primary goal is to reconstruct the ground truth, in order to be able to create good labels for a classification process. In the special case at hand, this means finding out when a failure has happened exactly. In this section, we will show how much information is contained in each of the datasets described in Sect. 3 w.r.t. the ground truth, and combine them in such a way that the optimal result based on our current state of knowledge is received:

- the time of failure given as an approximation of the day,
- information on when the train was in the workshop afterwards, and
- a list of layovers that were unnecessary.

### 4.1 Pure Diagnostic Data

When we look at the information we can retrieve from the pure diagnostic data, it seems reasonable to think that we can extract the point in time when the train driver received the precise error message that led him to report the failure.

In reality, however, the messages displayed on the MFD can not be retrieved from the diagnostic data. They are generated from it with a certain internal programming logic. Unfortunately, as it remains unknown how this is done, the combinations of codes needed to display a certain message also is not accessible. Given the original documentation, we would be able to identify the causes, which could help find the exact reasons the train driver called the hotline, and also evaluate which of those messages occur before real failures and which before unnecessary layovers (see Sec. 4.4).

When we take all diagnostic messages that explicitly state a malfunction of the main compressor into account, a result as depicted in Fig. 1 can be achieved, each square indicating one failure day. Hence, for the train in our example a total of six failure indications are present.

Because the underlying reasons that caused this messages are unclear, we proceeded by taking further knowledge into account and refine these findings in subsequent steps.

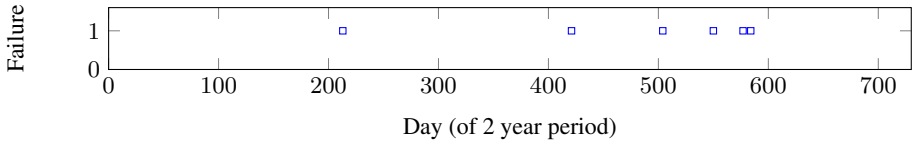


**Fig. 1.** Discovered failures for one exemplary train using only diagnostics data

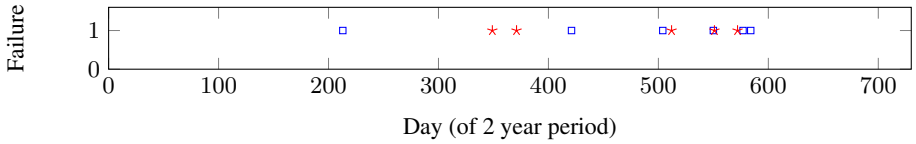


**Fig. 2.** Discovered failures using workshop data (red) compared to Fig. 1 (blue)

### 4.2 Workshop data

Using the workshop data, we can determine the point in time when a certain part has been replaced, and if the replacement was corrective or scheduled (mandatory). This greatly improved the identification of the true failures. Still, depending on how maintenance is handled, it only gives us a rough estimate of the point in time the failure actually took place. Note that maintenance procedures are not always carried out directly. Some types of failures require the train to be sent to a certain workshop, as not all workshops are equipped to handle every repair procedure. This may cause some days or even weeks in delay before the train finally is repaired. Therefore, the workshop date is not precise enough for a valid labelling.

A comparison of the extracted failure points can be seen in Fig. 2 depicted as red stars, showing a certain overlap with the findings from Fig. 1, but also completely unrelated entries. On average, red stars are 24 days away from blue boxes. If we only take pairs that are less than 21 days apart into account, the average distance is 6.5 days. But those are only 6 out of 10 instances, which leaves room for improvement and consequently leads us to the next step.

### 4.3 Failure Report Data

Utilizing failure report data, we are able to increase our understanding of when the actual breakdown has happened. The date of the reporting is noted here, and with high confidence it also states the correct day. We encountered some irregularities, for example the reporting date being behind the date the train was then brought into the workshop. We can still use this information to narrow down the exact day of the failure, but can not narrow it down to anything more fine grained than a day, because the precision of the timestamp that is recorded in the reporting system is not high enough. Therefore, we decided to take one day as the smallest unit a prediction can be done for. Since we expect to predict failures weeks in advance, this is not crucial. However, when failures have to be predicted that are appearing within minutes, the proposed method is not suitable any more.

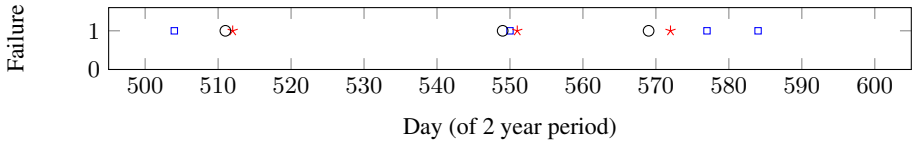


**Fig. 3.** Discovered failures using failure report data (black) compared to Figs. 1 (blue) and 2 (red)

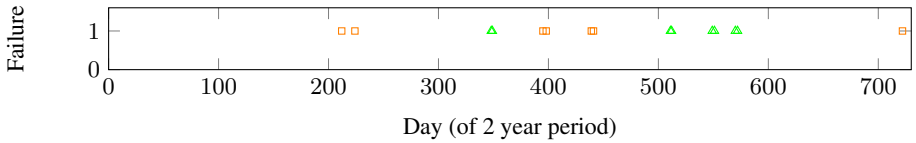


**Fig. 4.** Discovered failures (green) and unnecessary layovers (orange)

In Fig. 3 we now look at a smaller part of the timeline from day 500 to 620 (for better visibility). It is obvious that the failure report dates (black circles) are related to the workshop dates (red star), but not always to the diagnostic data (blue squares).

Therefore, we can conclude that only the combination of a failure report and a following repair is truly indicative of a failure. The diagnostic messages seem to indicate failures, but, surprisingly, after most of them the train is not affected negatively. Comparing the failure report dates with the repair dates an average distance of 1.6 days is yielded when events that are more than 21 days apart are discarded.

### 4.4 Unnecessary workshop layover

Unnecessary workshop layovers mostly happen because of the train drivers concern with safety, or them being overly cautious. As we were told by domain experts, the programming logic that drives the MFD's error and warning display is usually very sensitive, therefore generating a certain amount of false positives.

This may cause the train driver to trigger unnecessary maintenance actions. In the workshop the mechanics will then check the system, conclude that there is no failure and cancel the scheduled replacement. With the workshop data and the failure report data combined, we are able to differentiate the necessary from the unnecessary activities and exclude them from the pool of failures. This emphasizes the strong need for combining the different data sources by using expert knolwedge, as only then high-quality datasets can finally be built. In Fig. 4 the detected unnecessary layovers in comparison to the correct repairs are shown.

### 4.5 Missed and double repairs

Related to the unnecessary workshop activities are the missed repairs. Sometimes the train might arrive in the workshop with a given failure to check, and the repair crew may not be able to reproduce the symptoms, hence declaring this an unnecessary activity. A few days later the train might get submitted for the same reason again, and often only then the crew will actually repair or replace the component.

This effect has two implications, the first being that the time between those two layovers should not be used to train the model, because it may contain data where it is not certain if the failure is near or not. Second, it is also not clear whether the replacement that was made in the second attempt was actually well reasoned, or the maintenance crew decided to simply replace the part in order to eliminate the interference from happening again. These events are not documented as such, and we can only avoid negative influence on the training by removing the instances from the training set completely.

In the case of double repairs, we treat layovers caused by the same reasons that appear in a less than two weeks time as a single one, therefore assuming that the reasons to bring the train in were correct in the first place. Unfortunately, we can not prove if this assumption is always correct, but a discussion with the domain experts assured us that it is usually the case. With this 14 day range, the total number of compressor failure cases is reduced from 159 to 135.

## 5 Extracting meaningful features

In this section, we will describe the techniques we used to extract features from the given datasets. A thorough review of how to create features from various types of data can be found in [2]. We will give a short overview of the different types of attributes we extract and describe which features we create from them. As we have to do aggregation because of event based data, we chose the smallest possible bucket size of one day. As mentioned before (cf. Sect. 4.3), technical limitations apply. Furthermore, a failure should be predicted weeks in advance, and it is very likely that signs of deterioration are developed over a longer time, so one day seems to be appropriate.

### 5.1 Diagnostic messages and Status variables

As mentioned in Sect. 3.1, a diagnostic message has a pair of timestamps and values, one for *coming* and one for *going*, possibly spanning a certain duration. We use the values from one diagnostic code occurrence as source for multiple features. The information of a trains' runtime during one day is used to scale the attributes.

A status variable may have a certain amount of states, each defined by a number. For each of the states a feature is generated. The states behave like the diagnostic codes, the machine is in a certain state for a certain time span, therefore the features for both types are equal:

1. *Total duration of code/status*: All durations summed up for the whole day
2. *Frequency*: How often one diagnostic code/status occurs during one day
3. *Average duration*: Total duration divided by the frequency

These attributes cover the primary properties of the appearance of diagnostic codes and states. Other statistical values might be useful, e.g., variance of the average duration. It is planned to conduct further experiments including other statistics in the future, however, we are confident that these three statistics have the highest impact on the quality of the features.

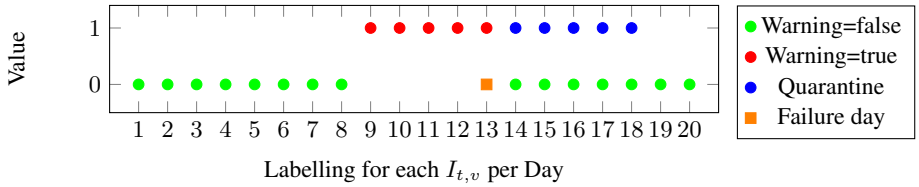


**Fig. 5.** The label ($warning = true|false$) assigned to instances before and after the failure

### 5.2 Numeric values

Numeric values occur in a wide range of applications, for example the measurement of temperature values. For these variables we use standard statistical measurements:

1. *Average*: Arithmetic mean of all recorded values in one day
2. *Maximum*: Maximum recorded value in one day
3. *Minimum*: Minimum recorded value in one day
4. *Variance*: Variance of all recorded values in one day

These attributes cover important properties of numerical values, more complex ones may be evaluated in later experiments.

### 5.3 Time normalization

Since a train does not have the same runtime each day, we scale the time-based values that are absolute (duration, occurrences) to the total uptime per day, in order to increase comparability between days. As a result we achieve frequency (occurrences per hour) as well as average duration per hour.

## 6 Labelling

In this section we will describe the labelling process with emphasis on the preprocessing steps. We will describe why large amounts of the data were not used for training and evaluation because of inconsistencies and information gaps.

### 6.1 Aggregation

As proposed in related literature (e.g. [2, 5, 6]), we will use a sliding window approach to label the instances. The goal of this is to calculate not only a certain feature-vector $A_t$ for a given day $t$, but instead calculate the trend leading towards this point in time. For this, we use linear regression with the window size $v$ for each day $t$ such as to create a vector $I_{t,v} = linreg(A_{t-v}, ..., A_{t-1}, A_t)$, in order to represent the behaviour of the system in the last $v$ days. The gradient of the linear regression is then used as the attribute. Each of those vectors represents one instance, as can be seen in Fig. 5. The labels are then assigned as follows:

**Step 1:** Label all instances as $warning = false$

**Step 2:** For each failure on train $B$ and on day $S$, label the instances $B_{S-w}...B_S$ as $warning = true$

The value $w$ represents the "warning epoch". The optimal value of $w$ will be determined experimentally, and depends on the specific type of failure. The optimal value for $v$ will also be determined experimentally.

### 6.2 Quarantine area

Because of the nature of the sliding window, we need to assure that - right after a part has been repaired - we will not immediately create instances with $warning = false$. For example, given a window size of $v = 8$ and a failure/repair on day $F$: if we create $I_{(F+2),8}$ the window will date back to 6 days before the failure and incorporate the measurements from those days. The calculated features would be influenced by the behaviour before the maintenance. Therefore we introduce the quarantine interval, also of length $v$. All instances in this interval may be affected by the failure and have to be treated accordingly, in our case removed. The quarantine interval prevents instances that are influenced by the effects of the failure, but labelled as $warning = false$ (see Fig. 5).

### 6.3 Unnecessary layover area

In Sect. 4.4 we elaborated on how we detect unnecessary layovers. Apparently these result from values in the diagnostics system which caused it to issue a warning on the MFD. Thus, some sort of non-standard behaviour has been detected. Compared to our ground truth we can state that - although abnormal - the records do not correlate with the failure we are trying to detect. We do not want these to affect the training of the classifiers, so we create a buffer area around those dates. The buffer area affects all instances from $I_{t-v}...I_{t+v}$. The instances inside this area will not be used for training.

### 6.4 Removal of instances

As stated before, the diagnostic data we built the instances upon is not recorded continuously, but on an event-triggered basis. For example, data is not recorded when the train is switched off. To address this issue, the concept of *validity* was introduced. If no data was recorded on a given day, this day is regarded as *invalid*. The same applies, when no mileage was recorded on a day. It can happen that a train is switched on and records data, even when it does not actually drive. Most often this happens in situations where the train is moved to another rail, hence, we consider a mileage of less than 10 km per day as *invalid*, since driving less than 10 km definitely is no cargo delivery.

The last attribute that has an influence on the *validity* is the information, if a train was in the workshop at a given day. In workshop layovers, usually problem detection gear is attached and some diagnostic programs are executed, causing the train to emit more diagnostic messages than usual. In order to keep this artificially created information from influencing the process, workshop days are also handled as *invalid*.

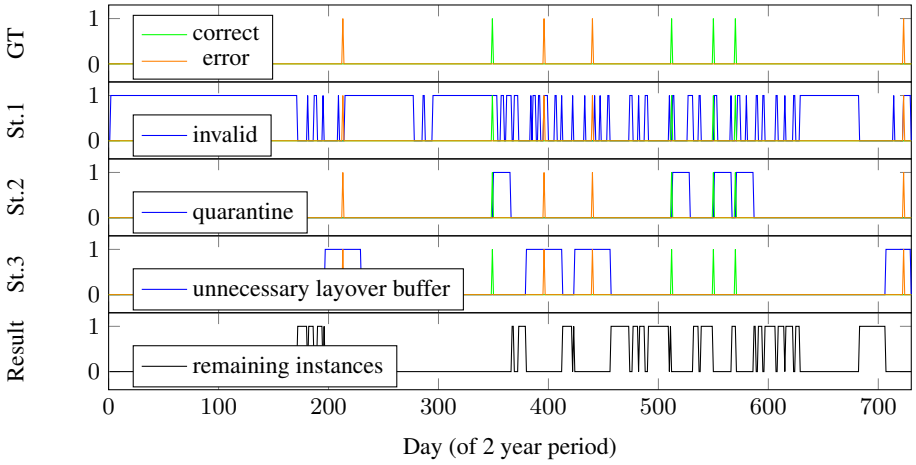


**Fig. 6.** Stepwise removal of invalid and unreliable values

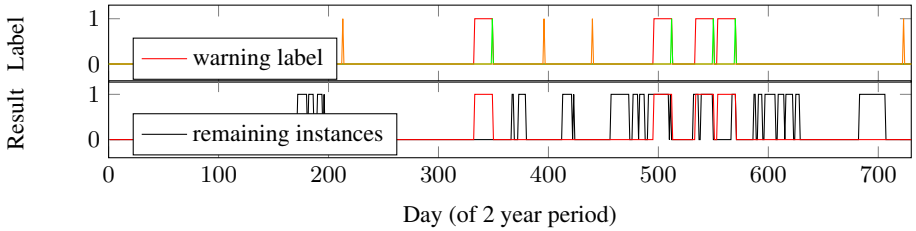


**Fig. 7.** Remaining instances compared to positive labels

In Fig. 6 the sequence of removal steps is displayed. In the first part of the figure, the ground-truth (GT) resulting from the process of Sect. 4 is shown. During a period of 2 years, we calculate the conditions for an instance for each day. In steps 1-3 those criteria are displayed, the status being true (1) when the condition applies.

The first step of the removal process eliminates all the invalid instances (St.1). In the second step, we remove all instances that appear in the quarantine period defined in Sect. 6.2. Finally, we remove data in the unnecessary layover buffer area from Sect. 6.3 in step 3. This is done in order to eliminate all negative training influences those instances might have.

At the end of this process we are left with a significantly smaller number of instances, as can be seen in the *Result* column of Fig. 6. In comparison to the actual labels we assign to those instances, we can see in Fig. 7 that a significant number of the “warning=true” instances was removed during the process. The quality of those remaining instances with respect to our labelling is highly increased when employing these steps, since potentially problematic, useless or erroneous instances are completely removed.

## 7 Conclusion

We have presented a preliminary process for converting a predictive maintenance scenario into a classification problem.

We dealt with domain-specific data, special characteristics of that data and presented preliminary solutions for the resulting challenges. We showed the necessity of incorporating domain expert knowledge in the process that proved to be successful for labelling the instances correctly. Several of the clean-up steps would not be possible without knowing the specific properties of the domain at hand.

Unfortunately, preliminary results in terms of classification accuracy were yet not promising. However, we are confident that with a further refinement of the presented procedures we will achieve better results soon. We will continue our work in the future with the following steps:

1. Discussion of validation methods and the implications they have (cf. [7])
2. Usage of sophisticated feature selection methods in order to improve classifier performance
3. Evaluation of classifier performance and parameter optimization
4. Solving the problem of of skewed class distribution
5. Evaluation of different approaches for converting predictive maintenance scenarios into classification problems (cf., e.g., [3])

## References


1. Sebastian Kauschke, Immanuel Schweizer, Frederik Janssen, and Michael Fiebrig. Learning to predict component failures in trains. In *Proceedings of the LWA 2014 Workshops: KDML, IR and FGWM*, 2014.
2. Silvain Létourneau, Chunsheng Yang, Chris Drummond, Elizabeth Scarlett, Julio Valdes, and Marvin Zaluski. A domain independent data mining methodology for prognostics. In *Essential technologies for successful prognostics : proceedings of the 59th Meeting of the Society for Machinery Failure Prevention Technology*, 2005.
3. David Martinez-Rego, Oscar Fontenla-Romero, and Amparo Alonso-Betanzos. Power wind mill fault detection via one-class ny-svm vibration signal analysis. In *Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 - August 5, 2011*.
4. Ying Peng, Ming Dong, and MingJian Zuo. Current status of machine prognostics in condition-based maintenance: a review. *The International Journal of Advanced Manufacturing Technology*, 50(1-4):297–313, 2010.
5. Ruben Sipos, Dmitriy Fradkin, Fabian Moerchen, and Zhuang Wang. Log-based predictive maintenance. In *KDD '14 Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*.
6. Marvin Zaluski, Silvain Létourneau, Jeff Bird, and Chunsheng Yang. Developing data mining-based prognostic models for cf-18 aircraft. In *Journal of Engineering for Gas Turbines and Power*, volume 133, 2011.
7. Indre Zliobaite, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, 98(3):455–482, 2015.