Master Thesis

Automatic detection of anomalies in times series data

Big data for smart maintenance

Richard Boon

1. ABSTRACT

In this thesis techniques in machine learning and data mining are explored for their applicability in a large dataset produced by complex radar systems. This dataset consists of a wide variety of data: logging measurements of many sensors and operations performed by the radar system. All data has one thing in common: they are all time series.

The thesis focusses on automatic detection of anomalies in these time series. Anomalies are defined in space and time. Anomalies in time are detected by learning normal behaviour from historic data. Anomalies in space are detected by comparison of the behaviour of similar components. Deviations from the normal behaviour in time and/or space are marked as anomalies. These anomalies can provide feedback for diagnosis, validation and prognosis of the radar system.

Multiple case studies have been identified in detecting anomalies which make use of techniques in machine learning and data mining. In this thesis three case studies are explored in detail.

The first case study focusses on usage patterns of the radar system. Using decision tree a predictive model is created for the usage of the radar system, based on its historic usage. The model can be used for diagnosis in usage of the radar.

The second case study focusses on validation of similar hardware components in the radar system. Using clustering techniques the behaviour of the components were compared to each other. We discovered that a specific hardware component in the radar is distinguishable by its behaviour using clustering and classification techniques. This result came as a surprise as we expected that these hardware components behaved similarly and thus should not be identifiable by its behaviour. It is expected that this is caused due to production variances in the components. The process can be applied as a validation to check the stability of the production.

The third case study focussed on analysis of sensor data. Long Short Term (LSTM) Recurrent Neural Networks were used as generative models for detecting anomalies in the time series. The neural network learns from historic data to generate new sensor values. These are compared with the real sensor values and the reconstruction error is used as anomaly score. This proved to be effective method for univariate time series. However, multivariate time series remain challenging. These models can be used as automatic diagnostic tools for sensor data.

Table of Contents

List of Tables

List of Figures

2. INTRODUCTION

In modern world, organizations are acquiring more and more data due to storage of data becoming cheaper, advances in the internet of things allowing to record all kinds of data and the huge amount of value which can be obtained from data for the organizations. The data can reveal valuable information about their customers and products. However, this faces companies with the complex problems of analysing and using this enormous amount of data efficiently. Techniques in artificial intelligence, data mining and machine learning try to solve these problems. Anomalies in the data are often especially of interest for companies as these can help to reveal intrusions, failures or new trends in their data.

In this work, techniques in machine learning and data mining are applied to real data sets of complex radar systems. The study is performed at Thales, a company producing radar systems for defence. These radar systems are equipped with many sensors and complex systems for controlling and monitoring the radars, which generate a wide variety of data that is stored in data logs. These data logs contain, among other things, commands sent by the user to the radar, logging of operations performed by the radar, generated alarm messages indicating potential failure of systems and sensor readings such as temperature and humidity.

In this work a broad overview is given of different techniques in data mining and machine learning. The data is explored which consists of a broad range of time series data: discrete and continuous, uni- and multivariate data. Multiple case studies are identified in which the techniques of machine learning and data mining are applied. These case studies all resolve around finding anomalies in the data in an automated manner.

Three case studies are explored in detail: unauthorised usage of the radar system, anomalous behaviour of similar hardware components in the radar and time series of sensor data.

In chapter 3 a broad overview is given in techniques in machine learning and data mining. Chapter 4 and 5 explores the company Thales where the study is taken and the dataset which is used in this study. Chapter 6 gives an overview of the goals. In chapter 7 six case studies are described and linked to techniques in machine learning and data mining. In chapters 8, 9 and 10 three of these case studies are further explored.

3. BACKGROUND INFORMATION

In this chapter related background information is explored. First we give a basic introduction to maintenance of systems, which is one of the potential usages of the dataset. Next, we will explore techniques in machine learning and data mining. Finally, an introduction to data warehousing is given, a technique related to data mining and an expected useful tool.

3.1. System Maintenance

Reliability of products has always been of importance for companies to keep customer satisfaction high. To provide for reliable products, a good product design is essential. However, over time reliability of components decreases due to tear. Therefore maintenance is required to repair and/or prevent failures and keep the products reliable throughout the usage over time.

Corrective maintenance is the most basic form of maintenance. Hereby maintenance is only performed when a component breaks down and stops working. This can lead to long down-time of the production cycle and high costs. Therefore, it is desired to intervene before this happens.

Using *periodic/preventive maintenance*, where after a certain period maintenance is scheduled, can prevent these failures. This period can be in a specified time or be measured in usage of the system, for example after a vehicle has driven a certain amount of kilometres. A downside is that in order to maintain high reliability of components, preventive maintenance must be performed often. This leads to unnecessary maintenance and replacement of components and creates high costs. Furthermore, the maintenance is performed by humans who can make misjudgements in the state of a component and/or overlook signs of tear in components.

A more sophisticated and cost-effective approach is to perform predictive maintenance, in particular by *condition-based maintenance* (CBM). The system is extended with sensors to collect data about the condition of the system and this data is processed to determine the health of the system. Maintenance is only performed when there are indicators of the system degrading or abnormal system behaviour (Heng, 2009).

3.1.1. Condition based maintenance

In CBM, sensors are installed in the system to monitor its components and perform maintenance based on the monitored system's health. CBM consist of three steps (Jardine, 2006):

1. *Data acquisition*: is the process of capturing relevant condition information which can indicate the health status of the system. This data can be very versatile ranging from temperature, oil analysis, vibration data, humidity data, etc.
2. *Data processing*: cleaning the data from errors/noise and analysing the data to improve its understanding and interpretation. The processing of data is very dependent on the type of data. For example, analysis of vibration data and oil data apply different techniques.

3. *Maintenance decision making*: based on processed data a decision must be made whether maintenance is required. Decision support can be divided into *diagnostics* and *prognostics*. In the former is focussed on detection, isolation and identification of faults when they occur. The latter focusses on predicting faults before they occur.

These additional steps in CBM bring extra costs to the maintenance cycle which would not exist in other forms of maintenance. Therefore, CBM is only cost-effective for expensive systems, which have high cost during down time, are under full service contracts and when remote monitoring is possible (Paul van Kempen, 2016).

In the past CBM has been successfully put into practice using vibrations signals, oil diagnosis and thermal monitoring. CBM has mostly been applied to mechanical systems and less so to electrical systems (Jardine, 2006).

There are roughly two approaches to acquire a decision whether maintenance is necessary (Medjaher, 2012):

- Model based prognostics
- Data driven prognostics

In model based prognostics mathematical models of the system are created. These models simulate the degradation process of the system and use this information to estimate the remaining life of components. These models require expert knowledge of the system and its operating conditions. Creating these models is often difficult and time-consuming and is usually only created for critical components of systems. For example, common model based approach is crack growth modelling using a variation of Paris Law, which is a formula that relates crack growth in materials with the amount of load it is enduring (Medjaher, 2012).

The other approach is data-driven prognostics, where models are created from data. Condition data of the system captured by the sensors are logged over time and failures of components are recorded. When sufficient data is captured about the system, techniques from data mining and machine learning, such as classification, clustering and regression analysis, can be applied to generate a prediction model of the system. These models can take various forms depending on the algorithm used to generate them, such as a (hidden) Markov model, mathematical equations, neural network, finite state machine, etc. These techniques will be explained in more detail in chapter 3.2 Machine learning & data mining.

Note however that both approaches are driven by data and it is difficult to separate an approach in one class. In practice, a model is usually a mix between domain knowledge and system data. In this study we primarily focus on data-driven models. Furthermore, the system under investigation contains a lot of condition monitoring data as will be explained in chapter 4.

3.1.2. Remaining Useful Life estimation models

The failure of assets is a variable which is difficult to predict. It is dependent on its age, operational environment conditions and observed health information (Si, 2011). The figure below shows the ‘bathtub’ curve displaying the typical failure distribution of a component. It contains three regions: infant mortality, useful life and wear out period. In the beginning a component has a high failure probability due to manufacturing defects. During the useful life period a component has a very low failure rate until, due to tear and wear, the failure rate

rises. Remaining useful life (RUL) estimation models are prognostic models which try to determine this period before the wear out period of a component starts; this can be seen in figure below. RUL estimation models are the basis to apply condition based maintenance.

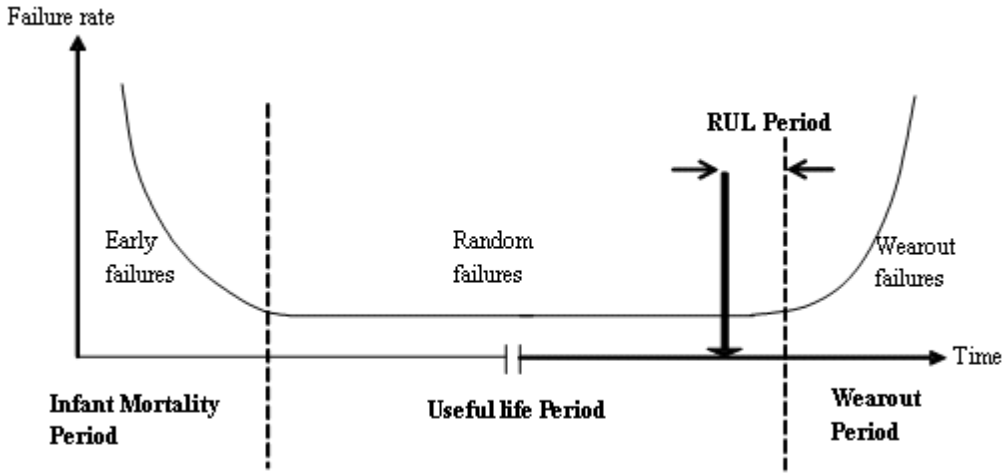


Figure 1: Typical failure rate of components; showing the typical 'bathtub curve'

RUL estimation models can be provided with two kinds of data:

- Event data: recorded failure data of components.
- Condition monitoring data: periodic measure about the condition of a component.

Acquiring event data can be difficult, especially for new product of which no failure data is known or critical assets which are not allowed to run to failure. As an alternative, RUL models based on condition monitoring data have been developed. These models describe the state of a component and determine its distance to a predefined threshold, which indicates failure of the component. The threshold is often determined by a domain expert (Peng, 2010), (Sikorska, 2011).

3.2. Machine learning & data mining

Machine learning is a subfield in computer science in which systems find patterns and/or make predictions on data, without being explicitly programmed how to do this nor having any background knowledge of the characteristics of the data. It is closely related and has overlap with data mining. Data mining is more focussed on the exploration of data.

Algorithms in machine learning can be divided by *supervised* and *unsupervised* learning. In supervised learning, algorithms are learning from examples. They are fed with input data, including their expected outcome and (try to) learn the relation between the input and output. It usually requires many examples to learn a relationship. Unsupervised learning algorithms, on the other hand, do not learn from examples but uses a big data set in which it tries to find relations and patterns.

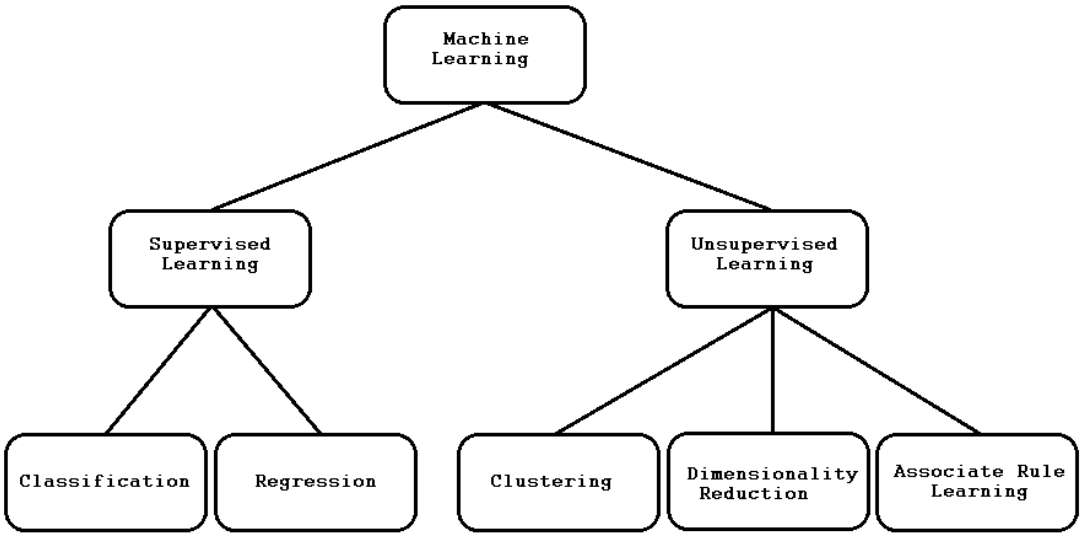


Figure 2: Division of machine learning techniques.

Figure 2 shows the distinction of supervised and unsupervised learning and makes an additional distinction of algorithms based on their goals:

3.2.1. Classification

Classification is the process of classifying the data into different defined categories. The input data for classification algorithms are large sets of records with each record containing an attribute set and a class label. The classification algorithm generates a function which maps the relations of attributes with the class label (Pang-Ning Tan, 2006).

A classic example is a spam-filter which tries to distinguish emails into spam and not-spam. The input for the classification algorithm is a dataset of emails, labelled spam or not-spam. Attributes of these emails can be the length of the email, number of spelling errors, keywords in the email, etc. Based on the dataset the algorithm tries to find a relationship between the attributes and class label. For example, emails containing spelling errors are more likely to be spam.

Once a model has been generated by the algorithm from the dataset, this model can be applied to new, unknown entries to try and classify them. This process has been visualized in Figure 3. In steps 1 and 2 a classifier model is learned based on a training set of labelled emails. In steps 3 and 4 new (unlabelled) emails are processed through this generated model and labelled as either spam or not-spam.

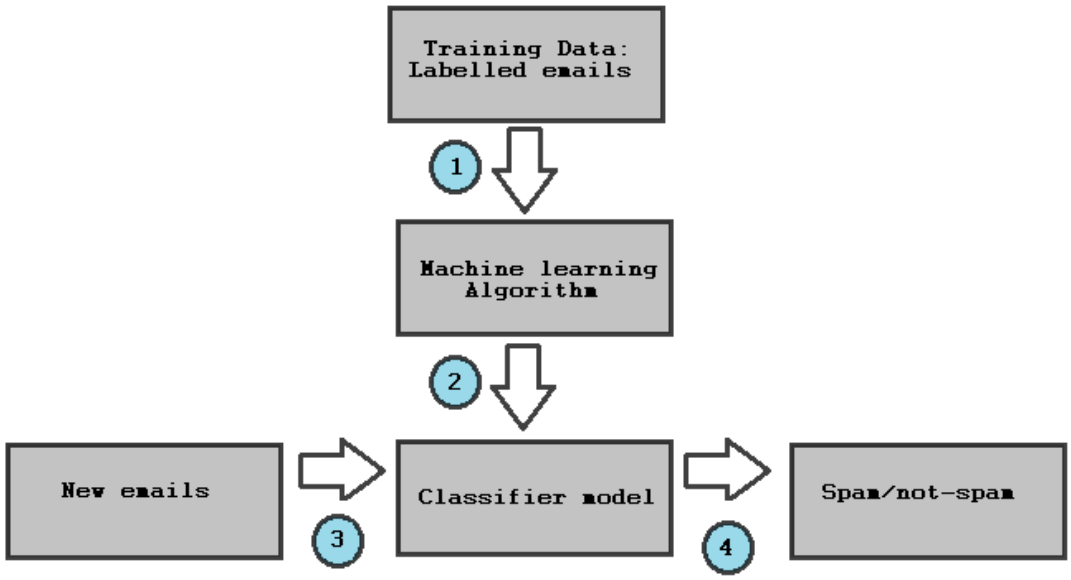


Figure 3: Classifying spam/not-spam

3.2.2. Clustering

Clustering is the task of dividing the data into groups, called *clusters*, of closely related objects, where objects in one cluster are more similar to each other than objects in another cluster. What defines the similarity between objects in a cluster can be defined by the user and determines the type of clustering algorithm (Pang-Ning Tan, 2006). A common similarity function (or inversely, the distance function) is the Euclidean distance, but a user can specify its own similarity function based on its domain and desired results (Xu, 2005).

Clustering is similar to classification as they both try to separate the data into groups, but unlike classification, clustering does not have predefined groups. Furthermore, clustering algorithms are performed on unlabelled datasets therefore clustering is sometimes also called unsupervised classification. Clustering is often used for exploration purposes of the dataset.

The result of clustering is most easily shown when we have a set of two dimensional points, visualized on Figure 4 shown below. We can clearly see four different groupings of the points. Applying clustering techniques with Euclidean distance as similarity measure, these four different clusters can automatically be identified. Clustering can also be applied to sets of higher dimensions or other types of data, for example graphs, when these clusters can be less apparent.

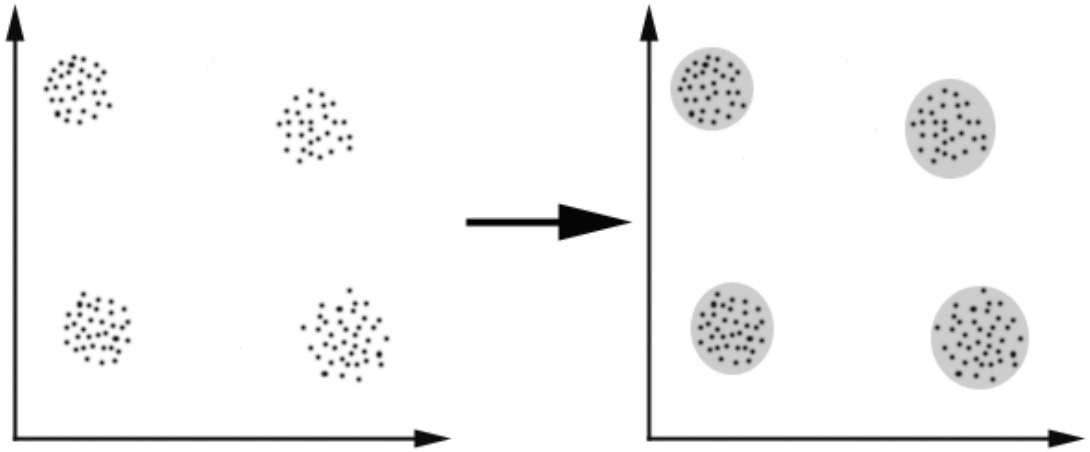


Figure 4: Applying clustering

An example where clustering is used, can be found in biology. Biologists have spent many years creating a taxonomy of all the creatures living on this world. To aid them in this task, biologists have used clustering to discover groups which share the same features. For example: applying clustering with the features: 'are warm-blooded', 'produce milk' and 'are vertebrates'; will reveal a large group of animals, namely the group of mammals. More recently, clustering is being applied on the genetic information of animals.

The result of cluster analysis depends on what you define as a cluster and the type of clustering technique you use. There are many definitions of clusters. Examples of definitions of clusters are (Rakesh, 2009):

- *Well-separated*: clusters are defined by distance between points. Distance between points in the same cluster is less than the distance between points in two separate clusters.
- *Graph based*: in case the points are represented by a graph, clusters are defined as a clique in the graph (a set of vertices such that each pair of distinct vertices are adjacent).
- *Density based*: clusters are defined by their density region. A cluster is a region consisting of high density of objects surrounded by a region low density of objects.

A rough way to make a distinction between different clustering techniques is to classify them on:

- Hierarchical clustering: the dataset is divided into clusters with a hierarchy. Objects can be assigned to a child-cluster which can belong to a parent-cluster, thus object can belong to multiple clusters.
- Partitioning clustering: the dataset is divided into non-overlapping clusters, thus each object is assigned to exactly one cluster.

There are hundreds of variants of clustering algorithms. Some of the most important clustering algorithms include:

- K-means (Hartigan, 1979)

- Support Vectors (Ben-Hur, 2001)
- DBSCAN (Ester, 1996)

Explaining their exact working is beyond of the scope of this study, in (Xu, 2005) an overview of clustering algorithms is given.

3.2.3. Regression

Regression is also in the class of supervised learning. Regression analysis tries to learn a function for predicting numeric values, as opposed to classification which works on categorical values. It takes as input a number of *independent* variables and outputs one or more numeric *dependent* variables. The model calculates based on the values of the independent variables the dependent variables. Regression is used for prediction, forecasting and to explore relations in datasets.

As a simple example take a regression model of the price of a house. The value to predict (*dependent* variable) is the house price. Variables which have influence on the house price (*independent* variables) are size of the house, the condition of the house, its location, etc. (Abernethy, 2010). Figure 5 gives an example where linear regression is applied to learn a function of the price of the house based its size (red line).

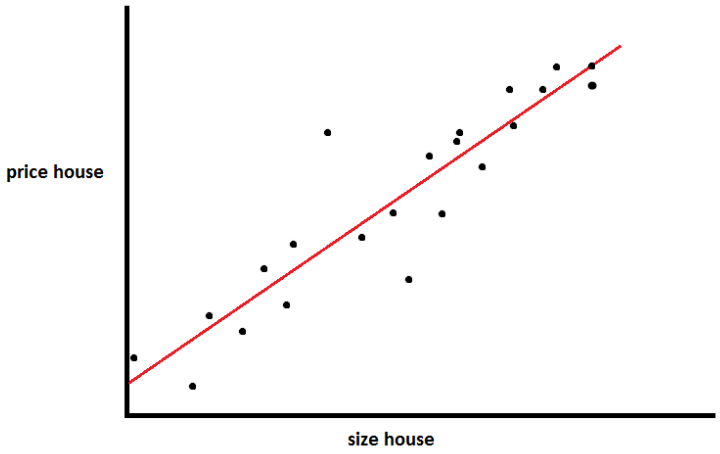


Figure 5: Regression, house price example

3.2.4. Associate rule mining

Associate rule mining tries to discover relations between variables in large data sets. It originates from market-basket analysis, where relationships between product sales of customers are being discovered, i.e. which products are often bought together by customers. These relationships are expressed as rules in the form $X \Rightarrow Y$, where X and Y are sets of items.

The result of applying associate rule learning will be illustrated with a market-basket example. In this example the dataset consists of a large table showing the all products sold with each transaction, see figure below for an example of how the dataset could look like.

TID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}

Figure 6: Market-basket dataset

Applying associate rule learning on this dataset would find multiple rules, one of which is: {diapers} ⇒ {beer}. This rule can be interpreted as follow: 'When a customer buys diapers, he/she will also likely buy beer'. These rules are often accompanied with *support* and *confidence* values which describe how reliable a rule is (Hipp, 2000).

Support indicates how many times the item-set appears in the complete dataset. In this example the item-set {diapers, beer} appears three times, out of the total of five transactions. Thus, we have a support of 3/5.

Confidence indicates how often the rule is true. In this example there are 4 transactions in which a customer buys diapers. In 3 of these transactions he/she also buys beer and the rule is true. Thus, we have a confidence of 3/4 for this rule. Notice that we don't count for the complete data set, but only for the set where the rule can be applied, i.e. transactions which contains diapers.

Support and confidence are used to limit the number of generated rules only to those that are of significance and interest for the user. The rule {Bread} ⇒ {Cola} for example, is not of interest as it only occurs one time in the dataset and thus has a low support. One can already generate many of these rules in this very small dataset.

A common strategy for association rule mining algorithms is to split the problem up into two parts (Pang-Ning Tan, 2006):

1. Generating frequent item sets: these are sets of items which occur often in the database, i.e. above a specified support threshold.
2. Generating rules: from all generated frequent item sets in the previous step, generate rules which have a confidence value above a specified threshold.

Many association rule algorithms and variations have been developed, some of the well-known algorithms are:

- Apriori (Agrawal, 1994): traverses with breadth first search through the search space, generates candidate item-sets each iteration from which frequent item-sets are extracted.
- FP-growth (Han, 2000): traversers with depth first search through the search space, no candidate item-sets are needed.

3.2.5. Dimensionality reduction

The task of dimensionality reduction is to reduce the set of variables to a set of principal variables. This can be necessary as datasets can become quite large in size and dimension which make the usage of any of the techniques described above computational intensive and time consuming. Furthermore, in some cases of classification or regression can be done more

accurately in a reduced space than its original space. This is caused by the 'curse of dimensionality' where the number of required observations to create a reliable model grows exponentially with the number of inputs (Ding, 2002), (Eklöv, 1999).

A simple way to achieve dimensionality reduction is by discarding variables which contain no valuable information in the application or are highly correlated to another variable. The main technique for dimension reduction is principle component analysis where the set of variables are replaced with a new set of variables such that the variance is maximized (Wold, 1987).

3.3. Model assessment

Algorithms in supervised machine learning, classification and regression, first learn themselves a model from a training set of labelled data. Next, they try to estimate the label on new, unlabelled data using this trained model. This new data can consists of records never seen before in the training data, in which case the model needs to apply the learned generalizations from the training data. One problem with training a model is *overfitting*, in which the model learns the training data too well and results in poor generalization. For example, overfitting occurs when we create a model which has 100% accuracy on the training data but only 50% accuracy on new data, while 75% accuracy on both data sets would be possible.

Overfitting is a common problem in machine learning and can take different forms. (Domingos, 2012) decomposes overfitting in *bias* and *variance*. "Bias is a learner's tendency to consistently learn the same wrong thing. Variance is the tendency to learn random things irrespective of the real signal."

This is illustrated in Figure 7 as dart throwing on a board. Imagine we have a large set of training data which we split to learn multiple models. Due to the different training sets the models will have some differences. Each dart (cross) on the board represents a model that is learned from the training data. The bulls eye of the dart board is the target model which we want to learn, i.e. a model with 100% accuracy. Models further away from the centre will have a larger prediction error. In the figure you can see that models with either high variance or high bias can result in models with a large error.

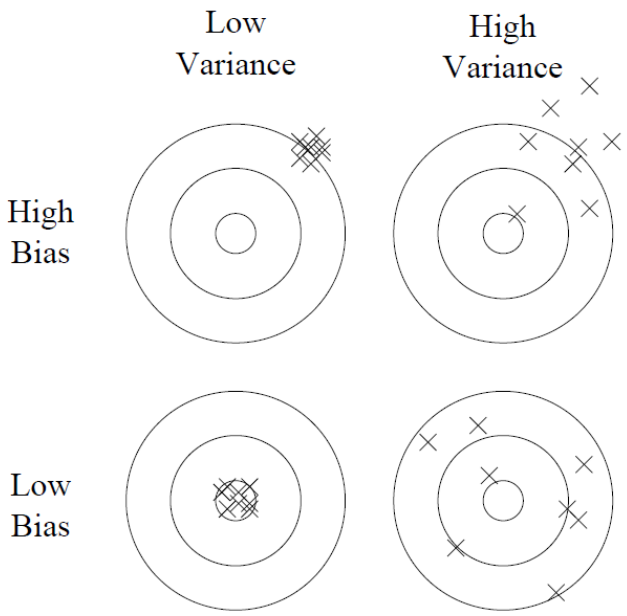


Figure 7: Bias and variance in dart-throwing, by (Domingos, 2012)

There are multiple methods to combat overfitting. Two commonly used and easy to understand are (Domingos, 2012):

- *Holdout method*: separate the training data set in two parts: learning set and validation set. Use the learning set to learn the model and the validation set is only used to test how well the learned model performs. Usually the validation consists around 25% of the total set and the other 75% is used for training.
- *K-fold cross validation* is similar to the holdout method. The dataset is separated in k sets of equal length. One of the sets is used as validation set while the other k-1 sets are used for training. This process is repeated k times such that each subset is used once for validation.

Another common pitfall in training your model is d*ata leakage*. Data leakage is the introduction of data outside of the training data set to create a model. An example of leakage is a model that uses the target itself to learn a model, which can result in the model to make useless conclusion like “it rains on rainy days”. Data leakage leads to overestimation of the model’s performance (Kaufman, 2012).

3.4. Anomaly detection

Detecting abnormal behaviour, or anomaly detection, is a problem that has been studied extensively in computer science and has been applied in many different domains such as fraud-detection in credit-cards, insurance or health-care, intrusion detection in cyber-security or fault detection in critical systems (Chandola, 2009).

In anomaly detection we need to compare 'items' with each other in order to identify if an 'item' is anomalous or not. What we define as an 'item' is of large influence on the techniques applicable to discover anomalies.

Furthermore, we need to find properties of these items which can distinct normal items from anomalous items. There are many possible properties and finding properties which can correctly distinguish normal and anomalous items is a difficult process.

In the work of (Chandola, 2009) different anomaly detection techniques have been identified and the assumption about how anomalies can be detected:

- Classification based techniques:
    - A classifier that can distinguish between normal and anomalous classes can be learned in the given feature space.
- Nearest neighbour:
    - Normal data instances occur in dense neighbourhoods, while anomalies occur far from their closest neighbours.
- Clustering based:
    - Normal data instances belong to a cluster in the data, while anomalies do not belong to any cluster.
    - Normal data instances lie close to their closest cluster centroid, while anomalies are far away from their closest cluster centroid.
    - Normal data instances belong to large and dense clusters, while anomalies either belong to small or sparse clusters.
- Statistical:
    - Normal data instances occur in high probability regions of a stochastic model, while anomalies occur in the low probability regions of the stochastic model.
- Information theoretic:
    - Anomalies in data induce irregularities in the information content of the data set.
- Spectral based:
    - Data can be embedded into a lower dimensional subspace in which normal instances and anomalies appear significantly different.

3.5. Data warehousing

A data warehouse is a system for reporting and data analysis. Businesses often use data warehousing to make business decisions supported by (historical) data. Data warehousing combines information of multiple sources and transforms the data from different sources to make them compatible with each other. Furthermore, data warehousing often involves huge amount of historical data.

There are multiple tools available for data warehousing. Especially the available tools for exploring a database make data warehousing an interesting tool to aid us in our study, because this study involves the exploration of how Thales might use the dataset for machine learning and data mining. These tools are optimized to query large amount of (temporal) data and allow the user to easily visualize this data. In the next section we will give some more details about data warehousing and how to apply data warehousing.

3.5.1. Data warehouse

(Jensen, 2010) defines a data warehouse as a subject oriented, integrated, time variant, non-volatile collection of data in support of management's decision making process:

- Subject oriented: a data warehouse is designed around the important subjects that concern the business, for example product sales, to allow easy analysis of them.
- Integrated: collection of multiple data sources, these sources can originate from outside of the enterprise.
- Time variant: it shows the evolution over time, and not just the most recent data. It has a time dimension.
- Non-volatile: deletion or updates are rarely applied to existing data in the data warehouse, mostly new data is added.
- Support of management decision making: the goal of a data warehouse is to allow managers to make business decisions based on data.

Ideally, only one data warehouse is present at an enterprise. Data warehouse integrates all these sources to one understandable format using one unit. It is the job of the Extract-Transform-Load (ETL) process to extract the data from different sources, clean the data and transform the data into an integrated format before loading the data into the data warehouse. Data warehouses are often implemented using a technique called multidimensional modelling which will be explained in the next section (Inmon, 2005).

3.5.2. Multidimensional modelling

With multidimensional modelling a model determines the logical structure of a database. It is a variation of the well-known relational-database model. It is a technique intended to support end-user queries in a data warehouse.

Multidimensional modelling introduces the concepts of *dimensions* and *facts.*

Dimensions are used for selecting and grouping of the data. The dimensions indicate the granularity used to analyse the data and each dimension contains multiple levels. Examples of dimensions are 'date' and 'location'. Date can have the levels 'year', 'month', 'week' and 'day' and location can have the levels 'country', 'state' and 'city' for example.

Facts contain the actual data to be analysed and consists of a combination of dimensions and an actual measure. An example of a fact table can be the product sales for a company. This fact table contains each sale (the measure) for the product sold by the company. It could have the dimensions time (when the product was sold), location (where the product was sold) and type (what kind of product).

Users can apply multidimensional modelling using the relational database model. Facts and dimensions are represented as tables. The levels of the dimension tables can be stored as separate tables or as columns in the table (this is called a snowflake and star schema respectively). The fact tables can become very large and can contain millions of rows, whereas the dimension tables stay relatively small. In the figure below an example is given using a star schema to implement a multidimensional model. At the centre is the fact table sales, which has the dimensions location, time and book.

BookID	Book	Genre
7493	Tropical Food	Cooking
9436	Winnie the Pooh	Children's books

Book (dimension table)

BookID	CityID	DayID	Sale
9436	854	2475	20
7493	854	2475	5
7493	876	3456	2
9436	876	3456	11
9436	876	2475	18

Sales (fact table)

CityID	City	State
876	Arlington	Virginia
854	Boston	Massachusetts

Location (dimension table)

DayID	Day	Month	Year
2475	March 1, 2009	March 2009	2009
3456	March 13, 2009	March 2009	2009

Time (dimension table)

Figure 8: Multidimensional model, by (Jensen, 2010)

The data warehouse consists of many of these fact tables, which contain interesting information for the enterprise and which can share the dimension tables.

Storing the data using multidimensional modelling has the advantage of being able to easily and efficiently query the fact table using different granularities. For example a user can acquire the sales for each location each year. This could lead to the observation that, for example, one location is lacking in sales which allow the user to query for a specific location the sales each month or each day.

4. CONTEXT OVERVIEW

In this chapter we will give a short overview of the company Thales at which the study takes place and the available data to analyse.

4.1. Thales

Thales Group is a multinational company specialized in designing and building systems in aerospace, transport, defense and security. Thales Nederland is a subdivision of Thales Group and is located in Hengelo, Enschede, Huizen, Delft and Einhoven. Thales Hengelo is specialized in naval systems, logistics and air defense systems. This study will be performed at Thales Hengelo. In the rest of this report, 'Thales Hengelo' will be abbreviated with 'Thales'.

4.2. Radars

Thales produces different types of radars:

- *Search radars*: used to detect and locate multiple objects in their environment.
- *Track radars*: used to accurately locate an object for fire guidance.
- *Identification Friend or Foe (IFF):* used to identify if object is friendly or not.

Radars studied in this paper are search radars, specifically product A and product B shown in the pictures below.

(classified)

Figure 9: Product A

(classified)

Figure 10: Product B

The radars are installed on ships that get deployed for defense missions. These missions can take long periods of time, i.e. multiple months, and take place various locations, resulting in different operating conditions of the radars. During these missions, the radar systems are of critical importance for the success of the mission. Furthermore, during missions only limited spare parts for the radar and expert knowledge can be brought along the ship, in case maintenance is needed. Therefore, it is of upmost importance that the radar systems are reliable.

4.3. Radar data

The radars produce two sets of data:

- Operational data
- Technical data

Operational data contains information about objects detected by the radar, radar frequency settings and ship locations which are used by the radar to achieve its goal, i.e., monitor its surroundings. The radars are able to produce a vast amount of operational data in short periods of time. This type of data is classified and not available for use in this study.

Technical data contains information about the state and health of the radar and its components. This is available for use and will be the main source for analysis. The exact content of the technical data will later be specified in chapter 5.

The available data contains technical data about product A and product B prototype radars described earlier. These are stationed at Thales in Hengelo and are currently being used for testing. Since (classified) 2016, technical data has been logged of the product A radar and since (classified) 2016 technical data has been logged of the (prototype of) product B radar. Data collection continues for these radar systems located in Hengelo.

Product A has limited storage capacity and can only store its technical data for a short time before running out of memory. Once this happens, the oldest data is thrown away in order to store the newly generated data. Therefore a script has been created to download and store the data of the radars in Hengelo to the main servers. Product B has much better storage capabilities and is able to store its technical data approximately up to at least 6 months. A tool has been developed by Thales to translate the binary data files into human readable format; multiple formats are available including XML and JSON.

4.4. Limitations

When using this data set for analysis one has to keep a couple of things in mind. Firstly, because the radars are currently in their test phase the data contains many system tests. In this test phase all components are thoroughly tested. Therefore the dataset contains the effects of seemingly random events produced by the system test engineers for testing purposes. Currently, it is difficult/almost impossible to trace which tests have been performed at a certain point in time.

Secondly, both radars are very new. Product A is currently in use by only one customer. Product B has not yet been deployed in the field, only at the test-site of Thales. Therefore there is no field data available and limited information about the failure of components in the radars.

Thirdly, the operating conditions on naval ships can be very different than the current operating conditions of the radars at the test site of Thales. However, the test site does capture the operating conditions of the two radars that will be deployed by the Royal Netherlands Air Force well. Furthermore, an agreement has been made with the Royal Netherlands Air Force to share the technical data log to Thales once the two product B radars have been deployed.

Acquiring historic data of similar radar systems already deployed in the field can become difficult task because defense organizations may not be keen on sharing technical and especially operational data of radars as this can reveal confidential information. Furthermore, captains of ships may not want to share data of their ship since it might lead to loss of face when, for example, it is revealed that components in his/her ship are not well maintained.

4.5. Current usage of technical data log

Currently, Thales is using the technical data log only for manual diagnosis of failures which are not correctly diagnosed automatically: whenever a customer faces a system defect and is not able to fix the defect themselves, Thales customer services is contacted. The technical

data log containing data of a few days before the defect happened is sent to Thales to analyse. Usually Thales' experts are able to find the exact cause of the defect in a short period of time.

It is expected that much more useful information can be gained from this data and one of the goals of this report to explore these possibilities. In chapter 7 multiple of these use cases for the data logs are explored and elaborated.

4.6. Other data sources

Besides the technical data log, there are other data sources which can be useful. Here these data sources will be listed. Some of these sources are readily available, others will require effort to acquire.

- Reports of health checks done by Thales.
- Failure reporting, analysis and corrective action system (FRACAS): this contains a classification of the failure and a root cause analysis. Via serial numbers they can be linked to components in the radar.
- Customer complaints which are registered via Jira.
- Online database of weather conditions.

5. EXPLORATION DATASET

In this chapter we will take a closer look into the technical data log. The content of the most important message types are explained. This data log is our main source of information during this study.

5.1. Message types

The software running on top of the radar consists of multiple subsystems. These subsystems are connected to each other via a middleware layer. Via this middleware layer the subsystems can send a wide variety of messages to fulfil their tasks. The system provides a data management capability, which is responsible for storing messages for off-line diagnostics or performance evaluation. It creates the technical log that we use for analysis.

The technical data log is ordered chronologically. Each time a subsystem sends a message through the middleware layer, the middleware layer adds a header to the message. The relevant attributes of the header are:

- Management interface: indicates which subsystem has generated the message. However, for some message types this value is not present.
- Parameter name: indicate the type of message.

Next we will explore some of the most important types of messages which reside in the technical data log, namely the following message types:

- Alarm
- Sensor status
- Notification
- BIT report & detailed condition status
- Usage record state
- Technical state
- Battle short state
- EIC status

5.1.1. Alarm message

Alarm messages indicate a failure in the system and the system can contain many different types of alarms. Some examples of when an alarm message is generated are:

- Two (sub) systems are unable to create a connection.
- There is a problem to execute the download software script.
- The measured air temperature is higher than the specified maximum operational temperature in this zone.
- A system receives a timeout.
- Invalid checksum in data has been discovered.

The product A system uses around (classified) and product B (classified) different type of alarms for failure detection.

Relevant attributes for an alarm are:

- Component_instance_name and management_interface_name indicates which software component has generated the alarm. It contains the same information as management_interface in the header of the message, i.e. management_interface is the concatenation of these two elements.
- Alarm_name and alarm_name_location indicates which failure is detected.
- The alarm status which can be either raised or cleared.

The system produces many alarm messages, but one single alarm message does not directly have to indicate a failure of a component. On contrary, some alarm messages are expected to arise during system operation. For example, during start-up of the radar not all components are activated at the same time, which causes many (expected) alarm messages to arise.

Thales has created an **advanced rule based reasoning capability** to recognize failures in the system based on the alarm messages. This capability is used for diagnostics and contains a lot of domain knowledge about the radar. It is able to precisely isolate the exact cause of the failure and provide repair instructions. Therefore, the customer (and Thales) does not have to analyse long lists of alarm messages during failures but instead can use this diagnostic capability.

5.1.2. Sensor status message

Sensor status messages are periodically generated messages which report the value of a specific sensor installed in the radar. The frequency of these messages depends on the exact sensor, but most messages are generated every (classified) minutes. The following sensor measures are available:

- Temperature: temperature of air, material or liquid
- Humidity: humidity of air.
- Pressure: pressure of air or liquid.
- Flow rate: rate of an air or liquid flow.
- Blower: rotation rate of a centrifugal fan.
- Valve: status of a liquid valve.
- Door: status of door, hatch, cover (open or closed).

Besides the measured value of the sensor, the sensor status message also contains:

- Resource name: where the sensor is installed.
- Min: the minimum value for which the sensor is healthy.
- Max: the maximum value for which the sensor is healthy.
- Condition: has one of the values “Normal”, “Warning” or “Failed”. When the value of a sensor is close to its minimum or maximum value, then its condition will be set as

"Warning". When it exceeds the min or max values then it will enter "Failed" state. Otherwise it will be in "Normal" state.

5.1.3. Notification message

Notification messages contain a wide variety of information about the system. System engineers let the system produce notification messages whenever the system makes operations which potentially could be useful information for debugging. However, notification messages are currently rarely used by Thales except for debugging. This is because for failure detection, alarm messages are more valuable and usually provide enough information to isolate the fault.

When the diagnostic capability, mentioned earlier, is not able to provide for correct fault isolation, domain experts are needed to investigate the problem in more detail. In these cases the notification messages can provide some additional information to help isolate the cause of a failure.

Similarly to an alarm message, a notification has a component_instance_name and management_interface_name, which indicate where the notification was generated. The notification_name indicates the type of notification. A notification also has an extra_data attribute, which is an array of additional notes about the specific notification type, specified by the programmer.

5.1.4. BIT report and detailed condition state

BIT (Built-In Test) report messages contain a report of diagnostics; it reports the number of observed faults including fault isolation and effects. A BIT report is the result of the advanced rule based reasoning system, mentioned in section 5.1.1, and contains its conclusions. Below an example of a bit report message is shown.

Further details of these faults can be found inside the detailed condition status message. The detailed condition state messages reports the health of a resource in a fine grained manner, by indicating the health status of a large amount of low-level elements of the resource.

(classified)

Figure 11: Example of BitReport with 1 fault.

Specification of the faults is only partly shown

5.1.5. Usage record state and info

The usage record state reports the usage of the resource, for example the number of hours a component has been used. Each usage record state message also has a usage record info message which gives additional details of the component and its parameters.

5.1.6. Technical state

A technical state message is a message generated when a subsystem changes from state. A subsystem can either be in an 'on' or 'off' state.

5.1.7. Battle short state

Battle short state messages are to enable/disable protection provisions for the radar systems implemented in the software. Disabling these protection provisions can be useful during battle situations when, for example, the user does not want part of the system to shutdown because a temperature gets too high.

5.1.8. EIC status

An EIC (Electronic Identification Card) status message reports information of hardware components installed in the radar system. Relevant attributes in an EIC status message are:

- Resource name: indicates the hardware component which is reported.
- Part number: part number of the component.
- Serial number: serial number of the component.

6. GOALS

The main goals of Thales are maximizing availability of the radar systems and minimizing their maintenance costs. Here we split this up in validation, diagnosis and prognosis of the system and explain how these are linked with the main goals of Thales. The result of this study should aid in achieving one of these goals.

6.1. Validation

Design validation: the goal is detection of design failures in the absence of hardware defects. A design failure is a deviation of the intended system behaviour. Only at limited points in time the design is manually validated. By means of data analysis the design can be validated continuously. This behaviour can be expressed as a measurement which needs to be in an expected range, based on the design. For example, temperature of a component is much higher or is fluctuating much more than expected in a particular mode.

Usage validation: the goal is detection of abnormal system usage by its users. Abnormal system usage is a sequence of system interactions which do not satisfy the assumptions/expectations. For example, switching the system on/off repeatedly in a short period of time or exchanging multiple hardware parts at once. This can show inappropriate use of the system which may cause damage. Thales can use this to improve customer training.

6.2. Diagnosis

In diagnosis the goals is to improve the detection of hardware failures and the isolation of their faults. The existing diagnostics capability, mentioned in section 5.1.1, correlates alarms to detect and isolate faults. It ignores alarms which only exist for a brief moment in time. Detection and isolation can be improved by using the history of alarms and by using other information such as notification messages, temperatures, etc. The goal is to detect failures which are not detected by alarms only. These failures may be detected by the expected behaviour based on past data or data of similar parts/systems operating in the same circumstances.

6.3. Prognosis

In prognosis the goal is to predict the remaining useful life of hardware components, via a data-driven approach. One must be able to predict the remaining useful life of hardware components with a margin large enough to allow for intervention before the (predicted) failure. For example, an increasing frequency of intermittent alarms could indicate a particular degradation of a hardware component.

7. CASE STUDIES

We have identified the following case studies related to data mining and machine learning which are applicable in the context of Thales. The case studies revolve around finding anomalous behaviour in the system. What is defined as anomalous behaviour differs per case study. This chapter will first describe how one can look at different definitions of anomalous behaviour. Next, the case studies are described with the goals they are trying to achieve, and applicable techniques which can help in achieving these goals. In Table 1 an overview of all case studies can be found along with the goals and applicable techniques.

7.1. Definition of an anomaly

The case studies described below are centred on finding anomalous behaviour. An anomaly is defined as 'something that deviates from what is standard, normal or expected'. Thus, in order to find anomalies in a system, we need to establish a baseline of standard, normal or expected behaviour. We can create this baseline based on different dimensions:

Time: define normal behaviour based on the previous observed behaviour of the system. If new behaviour differs greatly from historic data, it will be marked as an anomaly. For example, in historic data the temperatures may always fluctuate between two limits. Based on this historic data, we can define normal behaviour of the temperature between these two limits. Anything outside will be defined as an anomaly. Of course, one can also look into more detail of the historic data and take into account more details of what defines normal behaviour, for example, the slope of the temperature graph.

Spatial: define normal behaviour based on the behaviour of similar components under the same operating conditions. In this domain a component can, for example, be a TX or RX tile as there are multiple of these components installed in a radar. We can also compare complete radars of the same type. A component behaviour can be marked anomalous if it behaves differently than the majority of similar components at the same time.

Expert knowledge: define normal behaviour based on the experience and knowledge of domain experts. For example, a domain expert can determine that the slope of a temperature in the radar should never exceed a specified limit.

In this study we are mostly interested to learn a model of normal behaviour from the time and spatial dimensions. These can be learned from the available data log described in chapter 5; whereas models learned from expert knowledge requires extensive involvement of domain experts.

7.2. Case study 1: Usage of radar system

This case study is focussed on the usage of the radar systems by humans. This includes the operation of the radar system and also its maintenance. It is important that radar systems are used in a proper manner, because incorrect usage of the system can lead to failure of the system. For example, heavy usage of the radar system leads to more wear and tear of the system, therefore it is desired to keep such usage to a minimum and prevent unnecessary usage, to prolong the lifetime of a radar system. Furthermore, unexpected usage of the radar

system may also reveal unauthorised usage of the radar system. For example, someone accessing the radar at the Thales site in the middle of the night may be suspicious.

The goal of this case study is detecting anomalies in usage of the radar. We define an anomaly in usage based on the historic usage of the radar. For example, we expect to find a pattern of usage during the week and non-usage in the weekends. Unexpected usage in weekends and unexpected non-use during the week are examples of anomalies we hope to find.

Usage of the radar system can be tracked via the commands sent by the user to the radar system. Via these messages one can determine when someone is operating the radar. Additionally, human presence at the radar can be detected via sensors installed in the doors at the radars.

Via EIC status message maintenance of the radar can be tracked. Whenever we see a serial number changing, we know that the part has been replaced for some reason. Although the reason for replacement might be difficult to trace, simply knowing that a component has been replaced can provide for interesting usage information of the radar.

Via this data we can deduce and label at what times people were operating the radar, maintaining the radar and when there were people physically present at the radar. From this data we hope to find a pattern in usage of the radar, so that we can predict the likelihood of someone using the radar at a certain date. When we observe usage of the radar, while the prediction model would not expect such usage, an anomaly has been found.

In this case study we try to answer the following question: Can we predict usage of the radar based on historic data?

7.3. Case study 2: Anomalous TX/RX-tiles

This case study is focussed on the TX and RX tiles installed in the radar systems. These are the core components of the radar and are responsible for sending and receiving radio frequency signals respectively.

Due to the radars containing multiple of the same components, we can compare them both in the temporal and spatial domain. It is expected that these TX and RX tiles are behaving in a similar fashion and it is the goal of this case study to confirm or refute this. When one or multiple tiles are behaving differently we mark it as anomalous. With the help of a domain expert, the reason of its anomalous behaviour must be tracked.

In this case study we try to answer the following research questions:

1. What are features on which one can compare TX/RX tiles?
2. What techniques are applicable to compare the tiles features?
3. Can we find anomalous behaviour between TX/RX tiles with these techniques and features?

7.4. Case study 3: Analysis of radar sensors

This case study is focussed on the sensors installed in the radar system, which produce a continuous stream of numeric data. The type of data depends on the sensor and includes:

- Temperature: temperature of air, material or coolant.
- Humidity: humidity of air.
- Pressure: pressure of air or liquid.
- Flow rate: rate of an air or liquid flow.
- Blower: rotation rate of a centrifugal fan.
- Valve: status of a valve.

Monitoring of these sensors values is currently done automatically by the radar system, with simple range checks: when a measurement is not within a predefined range, an alarm is generated, which is processed by the diagnostics capability. However, measurements can behave abnormally while they stay within the predefined bounds.

Therefore, the behaviour of measurements in time can be monitored in a manual way, by looking at the sensor values for a moment in time or plotting these values over time. However, this requires Thales employees to continuously keep checking and verifying the behaviour of the sensors, which is a process prone to error and requires continuous human resources.

Furthermore, if one of these sensors observes abnormally behaviour, it is desired to get to know this as soon as possible to prevent any potential damage from happening and to be able to find and find the cause of this behaviour. For example, the temperature sensors can observe abnormal high temperatures fluctuations. If this is not observed within a short period of time, the thermal stress may damage the system. Additionally, recalling what the system was doing and especially what the system engineers were doing in the past what might be the cause of this abnormal behaviour becomes more difficult over time.

In order to prevent such scenarios from happening, it is desired to automatically observe the behaviour of the sensors and notify engineers when abnormal behaviour is occurring. Ideally, this system is able to predict that abnormal behaviour is about to occur in the (near) future, based on learned features which are indicators of this abnormal behaviour.

The goal of this use case is to develop a prototype of the described system. The main research questions which this case study tries to answer are:

1. What is normal and abnormal behaviour in sensors installed in the radar, and how can we learn this automatically?
2. What are available techniques to automatically detect abnormal behaviour?
3. How can we implement and integrate a warning system at Thales?

7.5. Case study 4: Start-up behaviour of radar system

This case study is focussed on the start-up behaviour of the radar system. The start-up of the radar consists of starting multiple subsystems and each subsystem will generate a sequence of messages. We can view the complete sequence of messages during the start-up as its fingerprint. Because the start-up is a non-deterministic process, multiple start-ups won't

produce the exact same fingerprint every time. Furthermore, because of constant development of the software running on the radar, the start-up fingerprint will also changes over time. However, it is expected that the fingerprints of normal start-ups have similar properties. For example, the order of produced messages might not be exactly the same, but we expect to find general patterns in the order of subsystems that are activated.

In this case study we focus on analysing these start-ups and how we can compare them with each other. We are interested in their similarity and development over time. Research questions this case study tries to answer are:

1. How can we determine in the technical data log the radar system is starting, and when do we know it's finished with the start-up procedure?
2. What are useful features we can track during its start-up?
3. How can we compare different start-ups with each other and see the development of start-ups over time, and spot anomalies?

7.6. Case study 5: Compare radar-systems

This case study is similar as case study 2 but instead of looking at the TX/RX tiles, compare complete radar systems with each other. Thus, in this case study we look at a higher abstraction level.

7.7. Case study 6: Sporadic alarms

This case study is focussed on the sporadic alarms. These are alarms which are raised at seemingly random times in the system, and a couple of moments later are cleared again. Why these alarms are occurring is not always known. Thales is interested to discover more about these alarms and their properties. For example, how often are they occurring? Are sporadic alarms occurring with increased frequency over time?

The rule based reasoning system, explained in section 5.1.1, simply ignores these alarms. However, it is expected that these sporadic alarms may provide help in diagnosing the fault. Thus, the sporadic alarms might also improve the current rule based reasoning system. This case study tries to answer the following research questions:

1. How can we distinguish sporadic from normal alarms?
2. What are features on which one can compare sporadic alarms?
3. Are sporadic alarms related to failure of components and the overall health of the radar system?

7.8. Applicable techniques

As explained in the beginning of this chapter, all case studies are trying to discover anomalous behaviour, thus techniques in anomaly detection are applicable to all use cases. To achieve this other techniques in data mining and machine learning, explained in chapter 3.2, can be of use.

Classification and regression are only applicable in supervised environment, i.e. where we have a set of labelled examples. Classification applies to categorical data whereas regression to numeric data. Thus, case study 1, 2, 4 and 5 may benefit from classification techniques to determine if instances belong to the classes of anomalous and non-anomalous. Case study 3 contains sensor data which is numeric of nature, thus regression is more appropriate.

Dimensionality reduction only makes sense to apply when data is of high in dimension, i.e. when the data contains many features. Thus this seems like a useful technique for case study 2, 4, 5 and 6 as we are dealing with many different types of messages.

Similarly, clustering is often used as an exploration tool on high dimensional data. This might help us in case study 2, 4, 5 and 6. Via clustering we can discover whether or not multiple clusters are appearing based on the features.

Associate rule mining might be useful in case study 6 to find relations between the sporadic alarms and failures of the system.

In Table 1 an overview of the case studies and the related techniques for each use case is given. Note that this serves as a basic guideline and starting point for which techniques can be used to solve each case study, based on the literature study described in chapter 3. It does not imply that these are the only techniques to solve each case study. The last column shows the goal of the case study. The following acronyms are use: P = prognosis, D = diagnosis, UV = usage validation, DV = design validation.

	Classification	Clustering	Regression	Dimensionality reduction	Associate rule mining	Pattern recognition	Anomaly detection	Goal
CS 1	x					x	x	UV, D
CS 2	x	x		x			x	DV, D, P
CS 3			x			x	x	DV, P
CS 4	x	x		x		x	x	DV, D
CS 5	x	x		x			x	DV, P, D
CS 6		x		x	x	x	x	D, P

Table 1: Overview case studies

We were not able to explore all case studies due to limited time. In the next chapters three chapters the case studies 1, 2 and 3 are further explored in detail. Each chapter can be read independently from each other and thus in any order. During each case study an agile methodology was used, where newly obtained results were discussed weekly with supervisors/domain experts.

8. CS1: USAGE OF RADAR SYSTEMS

In this case study the usage of the radar system is explored and a prototype is created to detect anomalous usage of the radar. The main component of the prototype is a classifier, which tries to determine for a given date and a set of features whether the radar will be in use or not, based on its historic usage.

8.1. Goals

The goal of this case study is two-fold:

- Firstly: the goal is to monitor the usage of the radar and detect when anomalous usage of the radar is occurring. This anomalous usage of the radar could indicate unauthorized access to the radar and is therefore of interest from a security point of view.
- Secondly: this case study serves as an introduction and practice to get more familiar with techniques described in chapter 3.2 and software libraries used to apply these techniques, namely Pandas (Pandas: a python data analysis library), Numpy (Jones, 2001) and Scikit-learn (Pedregosa, 2011).

8.2. Prototype

In the figure below an overview of the prototype is given. The main component is the classifier which creates a model of normal use of the radar system based on historic data. The model is used to determine unexpected usage for future data.

The first three steps consist of preparing the data and training the classifier. In steps 4 and 5 the learned classifier is used to evaluate whether or not anomalous usage of the radar is occurring. In the last step new incoming data is used to update the classifier by appending the new data to the historic data and starting process again:

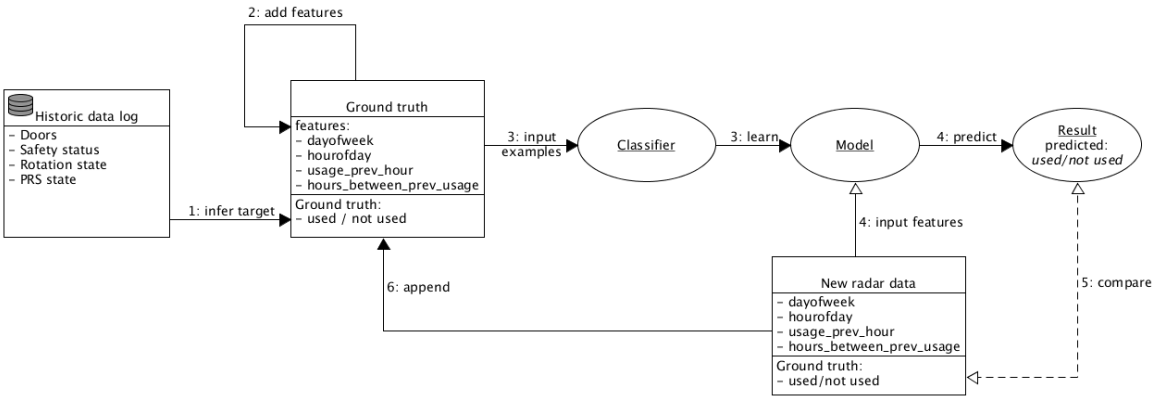


Figure 12: System process overview

1. In the first step, the ground truth for the classifier is determined, i.e. the moments in time when the radar was in use and when it was not. Unfortunately, this is not directly available. Therefore, this is inferred from the technical data log by observing the arrival times of a set of selected messages which indicate a person is present at the

radar. For example, when someone opens/closes a door in the radar, a sensor in this door will generate a message from which we can infer that someone is present at the radar at that moment of time. The set of selected messages can be found in Table 2 along with a short explanation of why it was selected.

Each day is divided into 24 slots of 1 hour and each slot is labelled as 'in use' if one or multiple of these messages occurred in this period. If no messages occurred, the slot is labelled as 'not in use'.

Observed message	Explanation
Doors	Someone opens or closes a door at the radar
PRS state change	Someone sends a command to the radar[1]
Rotation state change	Someone sends a command to the radar[1]
Safety status	Someone flips a safety switch at the radar

Table 2: Observed messages to determine ground truth

2. For each labelled slot features are added. The list of added features can be found below.

Feature	Type
Day of week	Integer between 0 and 6 indicating the day of the week: 0 = Monday, …, 6 = Sunday
Hour of day	Integer between values 0 – 23
Usage previous hour	0 = previous hour system was not used 1 = previous hour system was used
Hours between previous usage	Positive integer indicating the number of hours since the last time the radar was in use

Table 3: Features of classifier

3. The set of examples with their features and class labels have been extracted in the previous two steps, now the classifier can learn a model. This model is also evaluated to get an estimate of its performance. More information about the classifier and its performance can be found in the next two sections.
4. After a model has been generated with satisfactory performance, the model can predict new values. The radar can continuously send its features to the model which will tell if it is expected to be in use or not.
5. The prediction will be compared to the truth. If the prediction is correct, the radar is being used as expected. If the prediction is not correct, the radar is not being used as

[1] A command can only be sent from within the radar in the radar under study.

expected and some additional action may be needed. For example, warn security about a potential intruder.

6. The new data is added to the set of examples. This allows it to generate a new and (potentially) better model since it has more examples to learn from. To not waste too much computation power to generate a new model after each hour, a threshold should be defined after which enough new examples have been gathered.

8.3. Classifier

The classification task in this case study is binary: the radar is either in use or not. There are many different types of classifiers. In this case study a decision tree classifier has been applied. The main reason being as it is simple to understand and interpret the learned model via a decision graph.

8.4. Dataset

A classifier was learned for both the product A and product B radar. In the table below some more information about the data sets can be found:

	Product A	Product B
Start day	(classified)	(classified)
Last day	13/12/2016	9/1/2017
Total examples	5911	1992
In use	640	453
Not in use	5271	1539

Table 4: Data sources

It is immediately apparent that there are much more examples of none usage than usage of the radar. This is to be expected as employees will be working around 8 hours a day, for 5 days a week. During this period the radar *can* be in use. The other 16 hours a day, and during the weekends the radar is expected not to be in use.

8.5. Performance classifier

The learned classifiers were tested using 5-fold cross validation. Due to the class imbalance in the data set, the classifiers were also learned with a random under-sampled data set. This has shown to improve the performance of the classifiers according to (He, 2009). With random under-sampling all examples of the minority class (‘in use’) are selected and an equal number of the majority class (‘not in use’) are randomly selected. This thus results in fewer examples but the classes are evenly distributed. In the Table 5 the performance of the

learned classifiers can be found. For each classifier its accuracy, precision, recall and F1 score is given. In appendix 13.1 the confusion matrixes can be found.

$$\text{Accuracy} = \frac{true\ positives + true\ negatives}{true\ positives + true\ negatives + false\ positives + false\ negatives}$$

$$\text{Precision} = \frac{true\ positives}{true\ positives + false\ positives}$$

$$\text{Recall} = \frac{true\ positives}{true\ positives + false\ negatives}$$

$$\text{F1 score} = 2 * \frac{Precision * Recall}{Precision + Recall}$$

	Product A	Product A under-sampled	Product B	Product B under-sampled
Accuracy	0.930	0.857	0.852	0.882
Precision	0.705	0.820	0.642	0.853
Recall	0.605	0.916	0.794	0.923
F1 score	0.651	0.865	0.710	0.887

Table 5: Classifier scores

Both datasets improve from random under-sampling. Without the under-sampling, both data sets get a fairly high accuracy but suffer from lower recall and precision. With the imbalanced datasets, the classifier is more inclined to label new data as 'not used'. This causes a rise in the true negatives and false negatives compared to the true positives and false positives. This explains the lower recall without using random under-sampling.

8.6. Validation

To validate the anomaly detection prototype, a couple of labelled anomalies have been selected and further investigated to see if they were truly anomalous.

- Non usage during Christmas: false positive as the radar is not being used due to Christmas.
- Usage in the middle of the night: no observations found of usage in the middle of the night.
- Usage at start/end of day: likely false positive which can be explained by employees starting sooner or working overtime.

8.7. Conclusions and future work

It has been shown that with only a few simple features a reliable model can be created to detect normal and anomalous usage of the radar. This model can be used as warning system for security officers to warn for any potential unauthorized use of the radar.

Because the model learns from historic data, new use of the radar is automatically learned over time, for example when the radar gets deployed on another ship or location.

The best classifier was obtained using a small set of features and applying under-sampling on the imbalanced dataset. Future works lies in improving and integrating the model in a warning system. The model can be improved in several ways:

Firstly, adding more relevant features into the model. For example, a feature indicating whether the day is a national holiday or not is likely to improve its performance.

Secondly, testing the model performance with the use of over-sampling instead of under-sampling. In over-sampling new samples of the under-represented class are generated, which has been shown to provide for more accurate results than under-sampling (Batista, 2004).

Thirdly, while the decision tree classifier allows us to get insights in how the classifier is operating, it is not a classifier which has state of the art performance. Therefore, using different classifiers or an ensemble of classifiers can improve the model (Rokach, 2010).

Finally, the historic usage of the radar from which the model is learned, is inferred from the data log. This inferred historic usage can contain errors which results in the model learning from incorrect data and thus learning an incorrect model. Any improvements in the historic data usage of the radar will also improve the model. This can be improved by using more messages to infer that someone is present at the radar.

9. CS2: ANOMALOUS TX/RX TILES

In this case study the behaviour of TX and RX tiles are studied. The TX and RX tiles are one of the core components of the radar and are responsible for sending and receiving radio frequency signals respectively.

9.1. Goals

The goal of this case study is to explore and learn more about the behaviour of TX/RX tiles installed in the radar and find useful techniques to automatically analyse them. This is part of a higher goal of Thales to apply condition based maintenance on their radar systems explained in chapter 3.1 System Maintenance. During this case study we are interested in finding anomalous behaviour of the TX/RX tiles, which might be indicators of a failure of hardware components. What is defined as anomalous and how the behaviour of TX/RX tiles are captured, will be explained in the next sections. To achieve this goal, the following research questions will be answered in this case study:

1. What are features on which one can compare TX/RX tiles?
2. What techniques are applicable to compare the TX/RX tiles features?
3. Can we find anomalous behaviour between TX/RX tiles with these techniques and features?

9.2. Product A

This case study will focus on the product A radar, specifically on its TX and RX tiles. Product A is a radar which can be equipped with multiple TX and RX tiles, depending on its desired performance.

The radars used in this case study are equipped with 6 TX tiles and 10 RX tiles. TX tiles are installed in the cells 2, 3, 4, 7, 8 and 9. The RX tiles are installed in cells 1 through 10.

Thales has been collecting data of this radar type on two test towers since (classified) 2016. During this period the radars have been extensively tested and all technical data has been logged. In (classified) one of the radars has been replaced. Therefore there are three different datasets of product A available.

9.3. Datasets

To capture the behaviour of the TX and RX tiles the data logs of the radars have been analysed. These data logs are collections of binary files which contain all messages produced by the system during its operation time. In order to analyse these messages, the binary files have been converted into readable JSON format and loaded into a database. More details on the content of the data log and its messages can be found in section 5 Exploration dataset.

Many of these messages are unrelated to the behaviour of TX/RX tiles as they originate from other components. Therefore a filter has been applied which only selects the relevant messages for the TX/RX tiles. These relevant messages have been selected with the help of domain experts and consist of a group of alarm messages.

This selection of alarm messages will be the focus of this case study and will be used to measure the behaviour of the TX/RX tiles. Recall that the data of an alarm message is a time series of the alarm status which can be either raised or cleared.

The data log contains the moment in time when an alarm status changes. During the pre-processing, the moment in time when an alarm is raised and later cleared is combined to calculate the duration the alarm was in a raised state. Furthermore, the following attributes are extracted from an alarm message and stored in a separate database:

- Alarm type: alarm type of the alarm message
- Tile number: the cell where the tile is installed, see **Error! Reference source not ound.**.
- Tile type: type of tile, can either be ‘TX’ or ‘RX’.
- Tile: name of the tile, which is a combination of the tile number and type, e.g. ‘txtile 2’.
- Raised: timestamp when alarm is raised.
- Cleared: timestamp when the alarm is cleared.
- Duration: duration of the alarm in seconds (cleared – raised)

This is done for all three datasets, which results in 3 different databases. In the table below some general information about the three filtered databases are listed. In Figure *13* an overview of the pre-processing of the data can be seen. Due to the generic design of the process, it was easily repeated for all three datasets. Furthermore, the process is expected to work for other radar types.

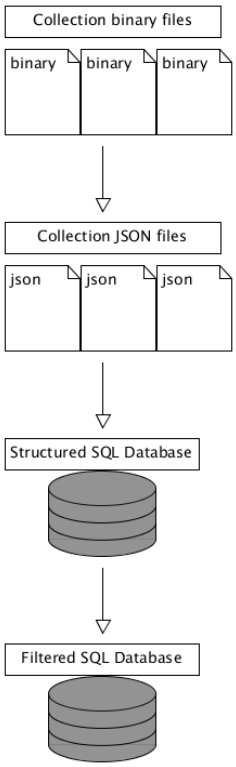


Figure 13: Overview of pre-processing of data. First binary data files are converted into JSON. Next, the JSON is read into a structured database. Finally, all relevant messages for this case study are filtered into a separate database.

	Dataset 1	Dataset 2	Dataset 3
Location	Test tower north	Test tower north	Test tower south
Timestamp first message	(classified)	(classified)	(classified)
Timestamp last message	(classified)	(classified)	(classified)
Total number of alarms	(classified)	(classified)	(classified)
Total TX alarms	(classified)	(classified)	(classified)
Total RX alarms	(classified)	(classified)	(classified)
Alarm types of TX	(classified)	(classified)	(classified)
Alarm types of RX	(classified)	(classified)	(classified)

Table 6: General information of the filtered datasets.

In the following sections, we focus on dataset 2, unless noted otherwise. The main reason being that this dataset captures multiple hardware failures which will be discussed next.

9.4. Hardware failures

During the time the radar under study has been operating on the test site of Thales, there are a few known instances of TX/RX tiles failing, see table below.

Date	Component	Serial number
(classified)	(classified)	(classified)

Table 7: TX/RX hardware failures

Unfortunately, the number of known hardware failure is very low and is insufficient to apply supervised learning techniques directly on the hardware failures. Furthermore, the exact cause of the failures was not known at start of the case study, and during the case study it became known that detecting the failure directly via an alarm message would not be possible due to the type of failures. However, it might still be possible to detect the failure indirectly.

9.5. Abnormal behaviour

Ideally, we want to link abnormal behaviour to the failures of the hardware components. Unfortunately, this is not possible due to the low amount of known hardware failures. Instead, we define abnormal behaviour in space and time:

9.5.1. Space

Radars are equipped with multiple TX and RX tiles. Each TX tile is the same hardware component and thus it is expected that these components act in a similar fashion at the same time under same operating conditions. The same is true for RX tiles. Any tile that behaves differently than the majority of the tiles is considered anomalous. This assumption forms the basis of our analysis in this case study.

9.5.2. Time

Another definition of an anomaly in the behaviour of TX and RX tiles can be compared to their historic behaviour. However, in the dataset available, test engineers have been performing various tests on the radar to verify correctness of the radar. These test include deliberately disrupting the system by, for example, disconnecting a cable to test whether the diagnostic capability of the radar works correctly. Because of these tests, comparing the behaviour of a TX/RX tile over time becomes very difficult as the system is operating in different operating conditions over time.

9.6. Obstacles

In the previous section, multiple obstacles in the dataset were discussed, namely:

- Low number of known hardware failures, makes it impossible to apply supervised learning techniques directly.
- Dataset contains experiments of test engineers. This pollutes the dataset and makes it more difficult to create a model of normal behaviour.

Additionally, the following facts provide for even more challenges in using the data:

- Dataset may contain more unknown subtle hardware failures, which may not yet affect system performance but which can pollute the dataset and influence our experiments if they are present.
- Product A is still under active development by Thales. This means that besides the hardware failures and errors caused by the test engineers, the dataset could also contain design errors and software bugs not yet discovered by Thales.

9.7. Basic statistics

The table below shows the result of some basic statistics over the complete dataset for TX tiles. For each TX tile the cumulative sum of an alarm raised is given in seconds. The rows

show the different alarm types and columns show the corresponding TX tile. The last row shows the total sum of all alarm types.

	TX tile						
Alarm type	**2**	**3**	**4**	**7**	**8**	**9**	**total**
SUB0_4	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_6	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_7	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_8	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_9	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_10	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_11	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_12	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_14	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_15	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_16	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_17	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_18	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_19	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_20	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_21	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_23	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_24	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_25	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_26	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_29	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_32	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_33	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_35	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_37	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_38	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_44	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_88	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_89	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_95	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_96	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_99	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
SUB0_102	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)
Total	(classified)	(classified)	(classified)	(classified)	(classified)	(classified)	

Table 8: Duration TX alarms raised in seconds per TX tile

Looking at the total sum per TX tile, it seems like all TX tiles are behaving similar, as each tile is around 1/6 of the total. However, comparing TX tiles on each individual alarm type shows large differences between tiles for some alarms (e.g. SUB0_7 and SUB0_21), while other alarm types are showing tiles behaving similar (e.g. SUB0_89 and SUB0_96).

Note that these statistics are a summary of over multiple months. These results are too generalized to make any conclusions about anomalies in the TX tiles. Instead, the alarms need to be studied in finer detail.

9.8. Visualize alarms over time

The alarms have been visualized over time to get a better understanding how alarm messages for the TX tiles are behaving. Figure 14 shows the status of 3 alarm types over time for all 6 TX tiles. The figure shows 6 subgraphs: one for each TX tile. The coloured lines represent the duration an alarm was raised. Each colour represents a different alarm type. For readability, only 3 alarm types have been shown and a very small time window of 2 minutes is visualized.

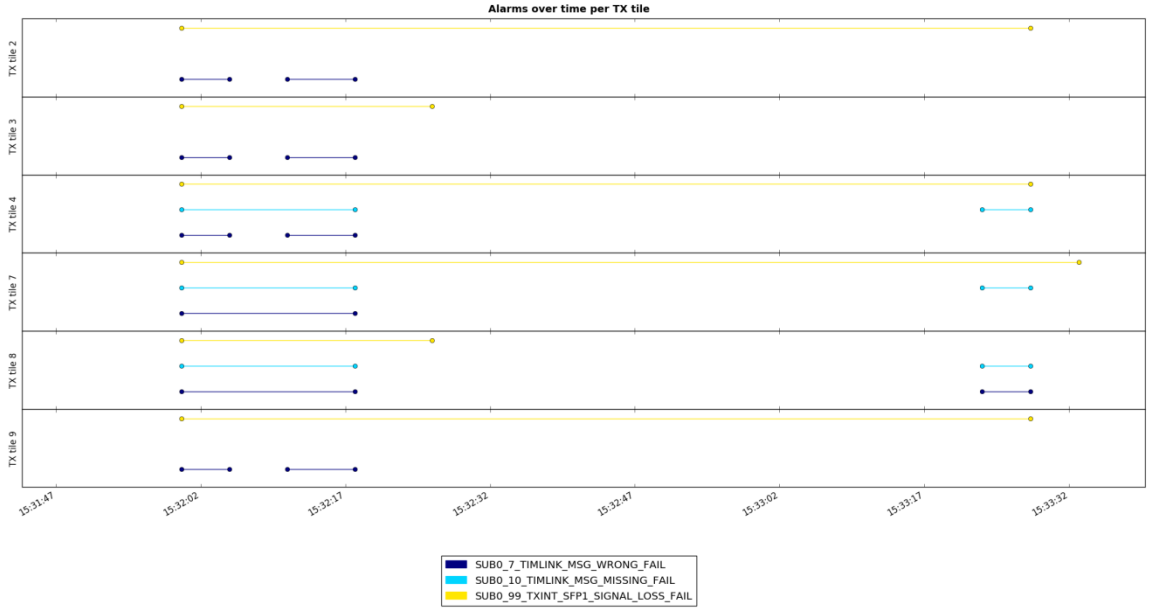


Figure 14: Alarms over time per TX tile

Notable from the figure is that the TX tiles seem have many similarities, but also some differences:

Similarities:

- All TX tiles start raising multiple alarms around 15:32:00. At this time the system was started.
- Whenever an alarm is raised by a TX tile, the same alarm type is often raised at the same moment by other tiles. Furthermore, their duration is often of similar length.

Differences:

- Tile 7 and 8 one period where alarm sub0_7 is raised, while the other tiles have a short interruption in this alarm. Furthermore, TX tile 8 has alarm sub0_7 shortly raised at the end.
- Alarm sub0_99 is much shorter for TX tile 3 and 8.
- TX tile 2, 3 and 9 don't have alarm sub0_10 raised in this period.

Analysing other moments in time shows similar results: the tiles show many similarities and some subtle differences in their behaviour. The similarities between the TX tiles are a confirmation of our assumption of the behaviour of TX tiles, i.e. the tiles are behaving in a similar fashion.

However, in this case study we are interested in the tiles that show differences in their behaviour, as it is expected that these differences are indicators of failures. In order to highlight these differences in our figure, we want to show only the alarms which are not shared between all TX tiles. To achieve this, we can give each alarm an additional property indicating how often the alarm type is '*shared'* with the other tiles. With 'shared' we mean the following:

> Whenever an alarm is raised, 'shared' is the sum of tiles which have the same alarm type raised within a specified time window of the moment the alarm is raised.

The length of the time window influences how sensitive the 'shared' property is to the moment in time an alarm is raised. A long time window allows the distance between two alarm messages from different TX tiles to be larger, and therefore it is more likely that the 'shared' property will be larger. The inverse is true for smaller time windows.

In our dataset a time window of 30 seconds is used: 15 seconds before and after the alarm is raised. This window has been chosen after carefully observing the distance of arrival time of messages during start-up of the system.

This additional property allows us to easily show the differences in alarms over time between the TX tiles. For example, by setting the 'shared' property to 1, we highlight all unique alarms for the tiles. This is shown in Figure 15 for a complete day. It also allows us to easily filter for alarm messages which present at most tiles but are missing at one or more other tiles.

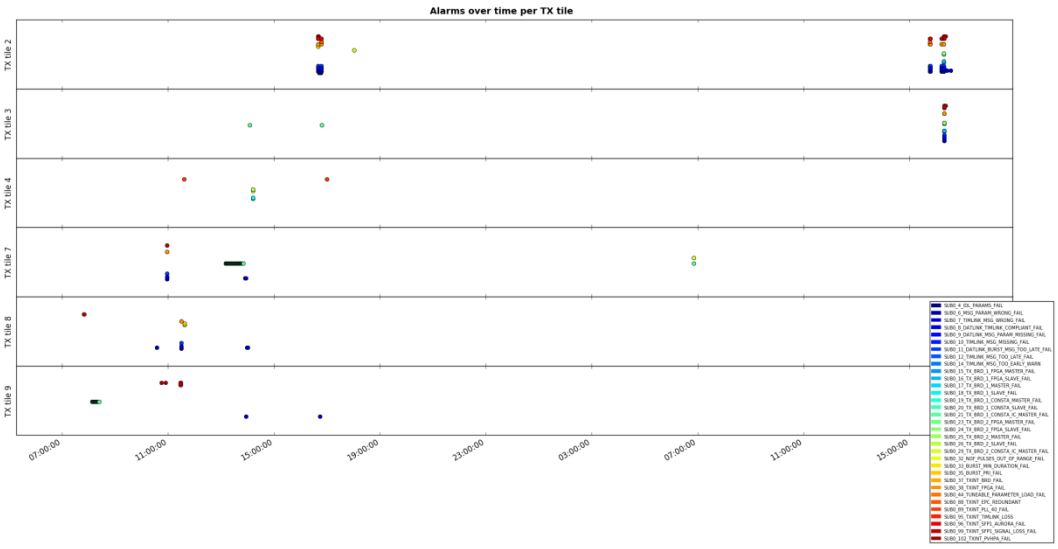


Figure 15: Alarms over time per TX tile, shared is 1.

9.9. Clustering fingerprints

In the next step we continued exploring the behaviour of TX tiles. For each TX tile we created a simple fingerprint of its behaviour during the day. The fingerprint consists of the cumulative sum of the duration of that day for each alarm type. This results in 6 fingerprints for each day

the radar is operating (as there are 6 TX tiles installed). Each fingerprint is a vector of length 41, because the dataset contains 41 different alarm types.

The idea is that because normal working tiles are behaving similarly, their fingerprints will also look similar and thus they will start forming large clusters. Malfunctioning tiles however will separate from the large cluster and thus stand-out.

Experiments were done using agglomerative hierarchical clustering. In agglomerative hierarchical clustering each observation (fingerprint) start as a separate cluster. Observations are iteratively combined into a new cluster. In each iteration two observations with the smallest distance are combined into a new cluster. This continues until all observations have been combined into one cluster or when the distance becomes larger than a specified threshold (Pang-Ning Tan, 2006). Multiple distance functions were tested:

- Manhattan: $\sum_{i=1}^{n}|q_i - p_i|$
- Euclidean: $\sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$

Where n is the total number of alarm types in each fingerprint and p and q are the two fingerprints from which we are measuring their distance.

Furthermore, multiple linkage functions were tested. The linkage function determines which distance to use between sets of observations.

- Ward: minimizes the total within-cluster variance.
- Average: uses the average of the distance of each observation of the two sets.
- Single (minimum): uses the minimum distances of each observation of the two sets.
- Complete (maximum): uses the maximum distances off each observation of the two sets.

Combinations of the above mentioned distance and linkage functions were used along with multiple distance thresholds. In these generated clusters we searched for any outliers, i.e. clusters with an abnormal large distance between the other clusters.

Furthermore, for each cluster the set of observations of which it consists is checked on its distribution. TX tiles are expected to operate similarly and therefore it is expected that the set of observations for each cluster consists of evenly distributed TX tiles. For example, if a cluster consists of 100 observations, we expect that approximately 1/6th of the observations originate from TX tile 2, 1/6th originate from TX tile 3, etc. If a cluster consists of observations of mainly one tile, it is marked as an anomaly.

Unfortunately, we were not able to find any results with this method. There appeared no large clusters with unevenly distributed observations. Furthermore, we were not able to identify any anomalous clusters. This can be caused due to the fingerprints still being too general and not able to capture subtle behaviour differences between normal and anomalous tiles.

9.10. Visualizing fingerprints

Next, we continued with the fingerprints of tiles. These fingerprints are expanded with some additional features for each alarm type:

- Max duration

- Minimum duration
- Mean duration
- Median duration
- Count of alarms raised
- Single raises: alarms where an alarm is raised, but no corresponding cleared message is present.

These fingerprints consisted of a very high number of dimensions: for each alarm type 7 features are recorded. With 41 different alarm types, each fingerprint consists of 7*41 = 287 dimensions. To visualize these fingerprints, multiple dimension reduction techniques have been applied, to bring back the number of dimensions to 2. In Figure 16 the results are shown.

The following dimension reduction techniques have been applied:

- Principal component analysis (PCA) (Wold, 1987)
- Linear discriminant analysis (LDA) (Izenman, 2013)
- t-distributed Stochastic Neighbour Embedding (TSNE) (Maaten, 2008)

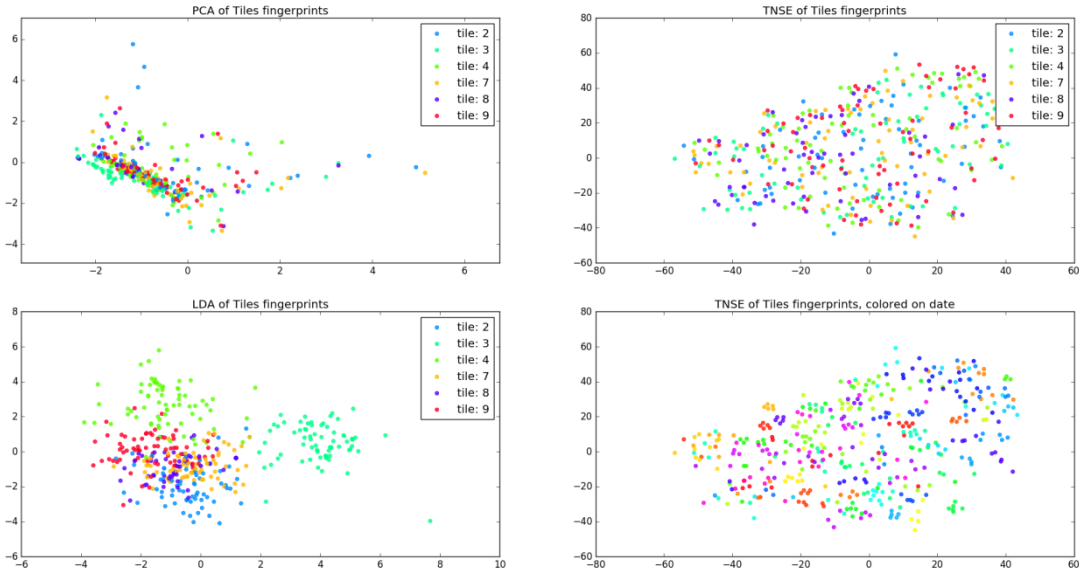


Figure 16: Visualization of fingerprints on two dimensions. In all panes the fingerprints are colored based on the tile number, except the bottom right pane, where the fingerprints are colored base on their date.

The results of the reduction techniques are shown in Figure 16, where each colour represents a TX tile. Using PCA no clusters are appearing. However, LDA shows different clusters appearing based on tile number. Especially TX tile 3 is well separated from the other clusters.

TNSE does not show any clustering based on tiles. However, when we colour the fingerprints bases on their dates, we do see fingerprints of the same day grouping together. This can be seen in the bottom right pane, where many small clusters of 6 observations appear.

The clusters appearing using LDA is rather remarkable, as this suggests that it is possible to distinguish the TX tiles based on their fingerprints. This would be a contradiction to the assumption that TX tiles are behaving similar. This is further explored in the next section.

9.11. Applying classification on fingerprints

Previous visualization has shown clusters based on tile. Next, we try to apply a decision tree classifier to discover more about which features are characteristic for each TX tile. The decision tree classifier is fed the fingerprints described in the previous section and it tries to classify to which TX tile the fingerprint belongs. This model has been validated using 5 fold cross-validation, similar as in the previous case study. In Table 9 the results of the classifier can be found. The last column, support, shows the number of samples that are true for the class.

TX Tile	Precision	Recall	F1-score	Support
2	0.58	0.63	0.61	73
3	0.91	0.92	0.91	73
4	0.84	0.71	0.77	73
7	0.64	0.67	0.65	73
8	0.53	0.49	0.51	72
9	0.68	0.74	0.71	73
Avg / total	0.70	0.69	0.69	437

Table 9: Classification report TX tiles based on fingerprints.

Striking is the high precision and recall. All TX tiles score above 50%, with TX tile 3 even reaching 91% precision. While one would expect the ratios to be around 17% if the tiles behave very similar. The complete decision tree model can be found in appendix 13.2.

The same has been repeated for RX tiles, however, these scored far lower with precision/recall around 15%. The other two datasets, described in section 9.3, showed similar results: a much higher precision and recall for the TX tiles than expected, but a low precision and recall for the RX tiles.

After careful analysis of the decision trees together with domain experts, we noticed that there are 3 alarm types of TX tiles which are recurring in all three models and seem to have a high impact on the model, namely:

- SUB0_7
- SUB0_10
- SUB0_99

All three alarm types indicate a malfunction in the data links of the TX tiles. To measure the impact of these alarms on the model, two additional decision tree models have been generated: one without the three alarms and one with only the three alarms. The results can be found in Table 10 and Table 11 respectively.

TX Tile	Precision	Recall	F1-score	Support
2	0.31	0.45	0.37	66
3	0.69	0.64	0.67	73
4	0.59	0.70	0.64	73
7	0.44	0.23	0.30	73
8	0.29	0.42	0.34	48
9	0.50	0.33	0.40	73
Avg / total	0.48	0.47	0.46	406

Table 10: Classification report TX tiles without the three alarms.

TX Tile	Precision	Recall	F1-score	Support
2	0.67	0.68	0.67	65
3	0.99	0.96	0.97	72
4	0.81	0.81	0.81	72
7	0.69	0.75	0.72	72
8	0.21	0.12	0.16	48
9	0.72	0.86	0.78	72
Avg / total	0.71	0.73	0.72	401

Table 11: Classification report TX tiles with only the three alarms.

The large drop in accuracy for the classifier without the 3 alarms and the fact that a classifier with only the three alarms still gives high accuracy shows that these 3 alarms are mainly responsible for the striking high precision and recall. An explanation of this result is likely to be found in these alarms.

9.12. Potential causes of unexpected behavior

The striking results of the classifier were discussed with several employees of Thales. Their expertise ranging from different field: from test manager, designer of hardware systems and software engineers. The exact cause has not yet been verified but there are some ideas of what could be the cause:

RX tiles have been produced much more than TX tiles and therefore more mature. Due to the still relative low production volume of TX tiles, they could still contain small production variances not yet discovered. These variances could be the reason why TX tiles are behaving dissimilar and thus explains the high performance of the classifier. It also explains why the RX tile classifier behaves poorly.

Furthermore, one of the domain experts has done many tests on data link connections.

If the production variances are in fact the reason of the high performance of the classifier, the classifier could serve as a tool for validation. Low performance classifier would then indicate small production variances.

9.13. Conclusions

In this case study we have seen that the behavior of TX/RX tiles can be measured with a set of alarm messages. Features of these alarm messages are its timestamp, alarm type and duration. The tiles show many similarities in their alarms, but subtle timing differences.

By creating a summary of the behavior of each tile per day, we can compare the tiles over time with each other. Using clustering techniques has proven to be unsuccessful. However, extending the summary and applying dimensionality reduction showed formation of groups. This led to the use of a classifier which was able to distinguish the tiles with accuracy much higher than expected. This high accuracy might be explained due to small production variances in the TX tiles.

9.14. Future work

For the future it still remains to be validated if the production variances in the TX tiles are in fact the cause of the high accuracy of the classifier. By switching the TX tiles from their position on the radar it is possible to discover if the same TX tile is still identifiable by its characteristics described by the classification model. If this is the case, then we have verified that these characteristics are originating from the TX tile and not the underlying hardware of the radar system. This would improve our confidence the reliability of the classifier as verification tool.

Furthermore, we can add additional features to the classification model to improve the fingerprint of a tile. This improved fingerprint thus described the behavior of a tile during a day in finer detail. These additional details might allow the model to discover anomalous tiles based on the added features.

In this case study we focused on techniques in clustering and classification for the detection of anomalies. However, another approach in associate rule mining is also possible, which is described in chapter 3.2.4. Via process mining we could create a model of common patterns of alarm messages. These common patterns will be classified as normal and large deviations

in these patterns can be marked as anomalous. Furthermore, the patterns of alarm messages of each of the tiles can be compared with each other.

10. CS3: ANALYSIS OF RADAR SENSORS

This case study focusses on continuous data produced by the sensors in the radar. These sensors monitor the various operating conditions of the radar, for example the temperature and humidity at various places in the radar and provide for feedback about the state of the radar system. In this case study methods for automatic anomaly detection for continuous time series are explored.

10.1. Goal

Currently, data of sensors is analysed on an ad-hoc basis by domain experts. Usually whenever a failure occurs, current sensor readings are checked for correctness to help identify the problem. However, this has the risk of missing anomalous behaviour due to human error. Anomalous behaviour of the sensor is usually defined by its historic behaviour, its *context*, and thus requires the expert to analyse this historic behaviour to be able to detect the anomaly.

Furthermore, the radar is equipped with many different sensors: product A already contains (classified) different temperature sensors, these numbers increase with more advanced radars, such as product B. This makes the analysis of sensors a labour intensive and repetitive job and therefore also more sensitive to human error.

Finally, as the sensors capture the current operating conditions of the radar, anomalies in these sensors can be an early indicator of a failure. Therefore, it is important to be able to detect any anomalies as soon as possible.

The goal of this case study is to discover techniques for *automatic* analysis of sensors to combat these problems. Figure *17* shows an example of two different anomalies in time series. The left pane shows a series which contains several outliers. These anomalies are relative simple to detect using a threshold. Thales already has such a system in place, where an alarm message is generated whenever a sensor exceeds a value, which is specified by domain experts. The right pane shows a signal which resides within the safety thresholds, however strange behaviour is still occurring in the signal. This anomalous behaviour only becomes apparent when looking over the subsequence and taking into account its historic behaviour. Therefore these types of anomalies are more difficult to detect. This case study will focus on the detection of the latter type of anomalies.

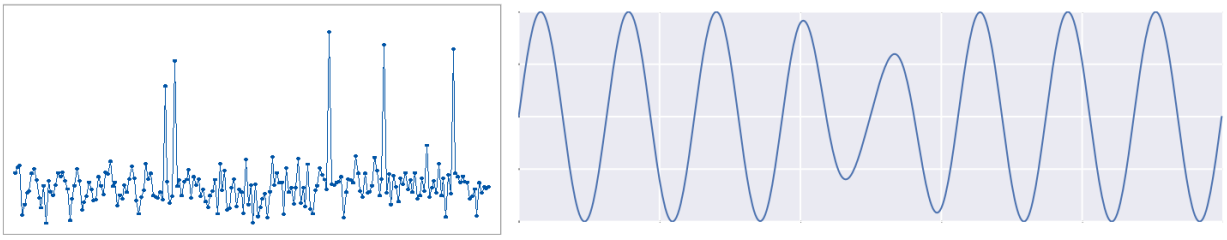


Figure 17*: Two different anomalies. Left pane shows several point outliers; right pane shows a time series with a context anomaly.*

10.2. Dataset

This cases study will use the data log of the product A radar. This data log contains time series data of the following sensors:

- (classified) temperature
- (classified) humidity
- (classified) blower rate
- (classified) flow rate
- (classified) pressure

Readings of these sensors are logged around every (classified) minutes over a period of multiple months.

10.3. Related work

There has been done extensive research on detection of anomalies, from many different domains and application areas. In chapter 3.4 a high level categorization is given on existing techniques. An extensive review of anomaly detection techniques is outside of the scope of this study, for more information on this we refer to the work of (Chandola, 2009) and (Patcha, 2007).

This case study will focus on anomaly detection using artificial neural networks, specifically Long Short Term Memory (LSTM) neural networks. An artificial neural network is a reasoning model based on the human brain. It consists of a large set of basic information processing units, called *neurons*, which are densely connected with each other. Neurons can have multiple input units from other neurons and produce one single output. This output can be split over multiple branches which transmit the same output. These outputs can in turn be used by other neurons as input. The complete set of interconnected neurons is called an artificial neural network.

Neural networks are connected by *links*. Each link has a *weight* associated with it, which determines the strength of the connection. During the learning phase of the neural network, these weights are repeatedly adjusted towards the desired behaviour of the neural network. The neural network learns from examples to nudge the weights in the right direction. An algorithm which is often used to train the network is Back Propagation Through Time (Werbos, 1990). In Figure 18 an example of a neural network is given, each circle represents a neuron and the arrows represent the links between the neurons (Negnevitsky, 2005).

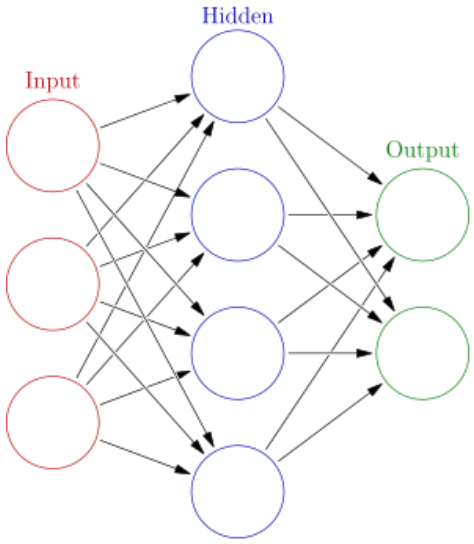


Figure 18 Architecture of a typical neural network

Recurrent neural networks (RNN) are a type of neural network which allows the formation of cycles between neurons. This allows the model to 'remember' previous input data and is therefore especially useful in sequential data.

However, RNN suffer from the *exploding and vanishing gradient problem*. This is a major obstacle for the performance of the model, where the training of the network takes too much time due to extreme increased learning rate. A detailed description of the problem is outside of the scope of this paper, but can be found along with methods which try to overcome this problem in the work of (Hochreiter S. , 1998).

Long Short Term Memory (LSTM) is a type of RNN which try to overcome the vanishing gradient problem. In LSTM the neurons are more advanced. Very briefly, they are equipped with a 'memory cell' that allows them to store information unchanged for many time steps. The memory is controlled with an additional 'input', 'forget' and 'output' gate. These gates are also associated with a weight which the network needs to learn. The network is therefore able to learn when to when to remember, forget and forward historic information. The LSTM network is able to overcome the vanishing gradient problem because it makes use of an addition operation on its input instead of multiplication. More details about the LSTM network can be found in (Hochreiter S. a., 1997).

RNN and LSTM have proved to be able achieve state-of-the-art performance on many machine learning problems related to sequences, for example in handwriting/speech recognition, translation and image caption. Very recently the use of Long Short Term Memory (LSTM) neural networks have been used successfully as generative model to discover anomalies (Marchi, 2017), (Malhotra, 2016), (Blaisten-Barojas, 2016).

In LSTM anomaly detection a LSTM network is trained to reconstruct the normal time series. The idea is that the network won't be able to reconstruct anomalies and thus these anomalies will be revealed in the reconstruction error. This is under assumption that the training data does not contain any anomalies or relative few anomalies. It is thus a semi-supervised algorithm, where the model is fed with normal examples to learn from.

There are properties for LSTM which makes them attractive tools to use in our case study:

- Semi-supervised: the model does not require labelled data to learn.

- Neural networks can easily be extended to allow multivariate data.
- No (extensive) pre-processing of the data is required or any feature engineering.
- Capable of detecting novel anomalies which have never occurred in its history.

Some disadvantages are:

- Training of a neural network often is computational expensive. However, once a model has been generated, new data can efficiently be evaluated.
- A neural network contains many hyper-parameters which need to be tuned to find the optimal model. There exists no strict guidelines in finding this optimum and can therefore be a difficult process.

10.4. Experiment with generated data

First, an experiment was performed where the above described anomaly detection technique was applied on a generated data set. The data was generated from two combined sine waves with some added noise sampled from a uniform distribution, resulting in a dataset of a total of 1000 samples. The dataset is split into a training of 600 samples and validation set of 400 samples. In the validation set an anomaly is inserted. A LSTM network consisting of 3 fully connected hidden layers was trained to predict the next value given the previous 100 values. This window of the previous 100 values was selected to include the periodicity of the combined sine wave, see Figure 19.

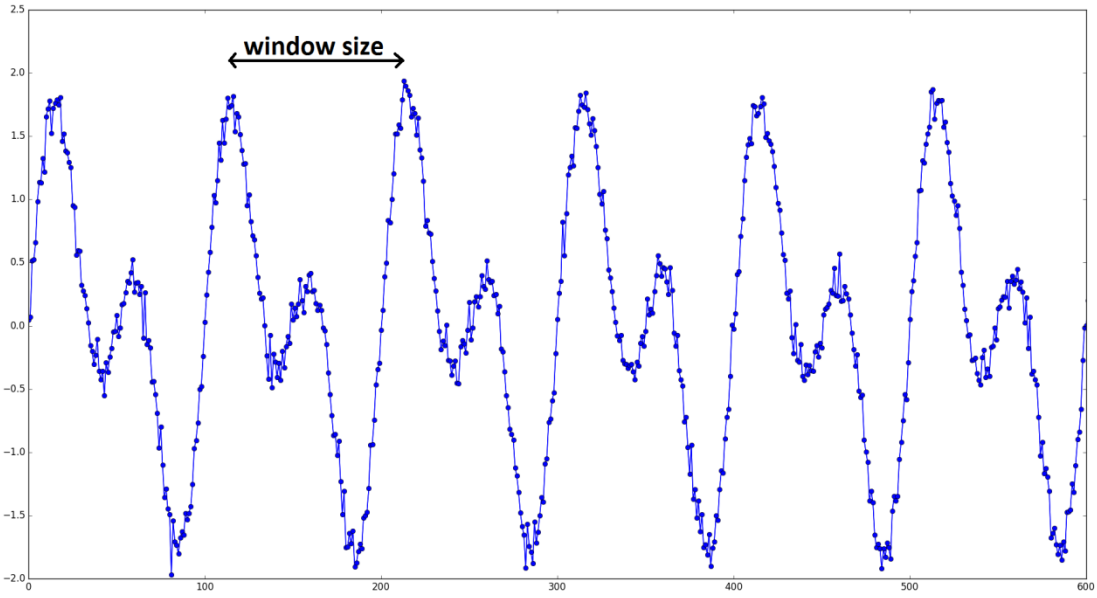


Figure 19: Generated training dataset and window size large enough to contain the periodicity of the series.

The network is trained on the training set and then validated using the validation set. In Figure 20 the results of the trained network are shown on the validation set. The top pane shows the original data, while the middle pane shows the reconstruction using the LSTM network; the bottom pane shows the squared difference between these two.

The LSTM network is able to reproduce the signal very well except around point 200. Here the inserted anomaly occurs which is clearly visible in the bottom pane. This demonstrates the effectiveness of LSTM as anomaly detectors. Note that the LSTM only took 600 samples to train the network.

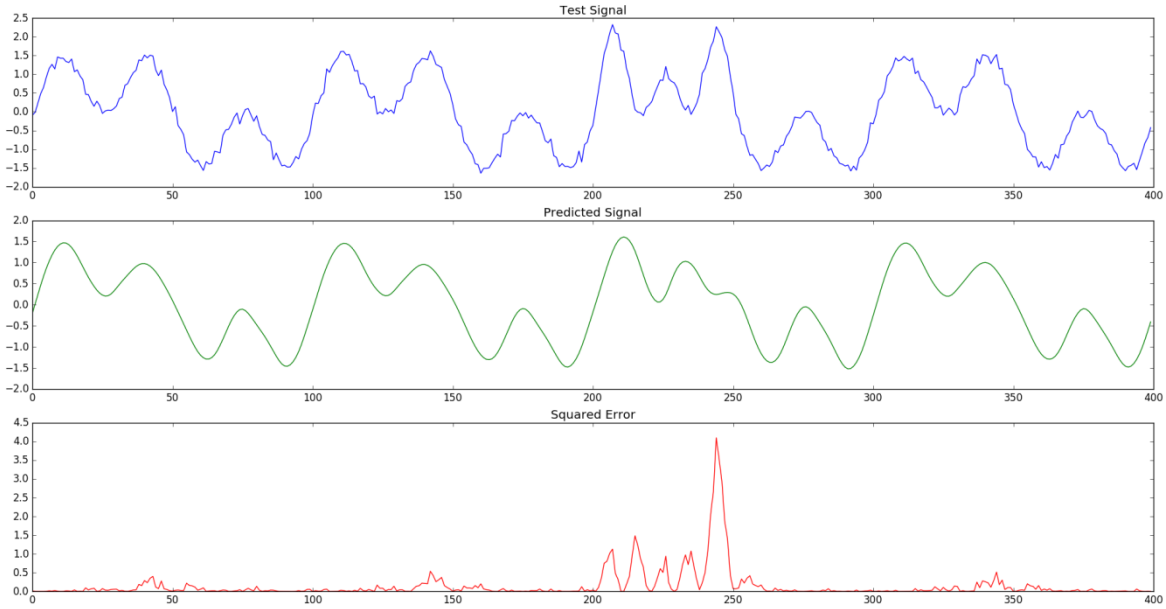


Figure 20: Experiment with generated sine wave with an anomaly

Next we will try to apply the same technique on the field data of the radar. However, before this is possible, some obstacles need to be overcome which will be discussed next.

10.5. Obstacles in field data

10.5.1. Radar status

Some of the sensors are dependent of system state and thus the system state is needed to reconstruct the time series. This is illustrated in Figure 21. In the upper panel the temperature of coolant is shown over time. The lower panel shows when the radar is sending radio frequency signals, as indicated by a high signal. In the figure the correlation between the temperature and operating condition is shown: each time the radar starts sending the temperature of the coolant drops. This somewhat counter-intuitive behaviour of decreased temperature during load can be explained by the fact that this sensor measures the temperature of coolant of a cooler which only is active when the radar is transmitting.

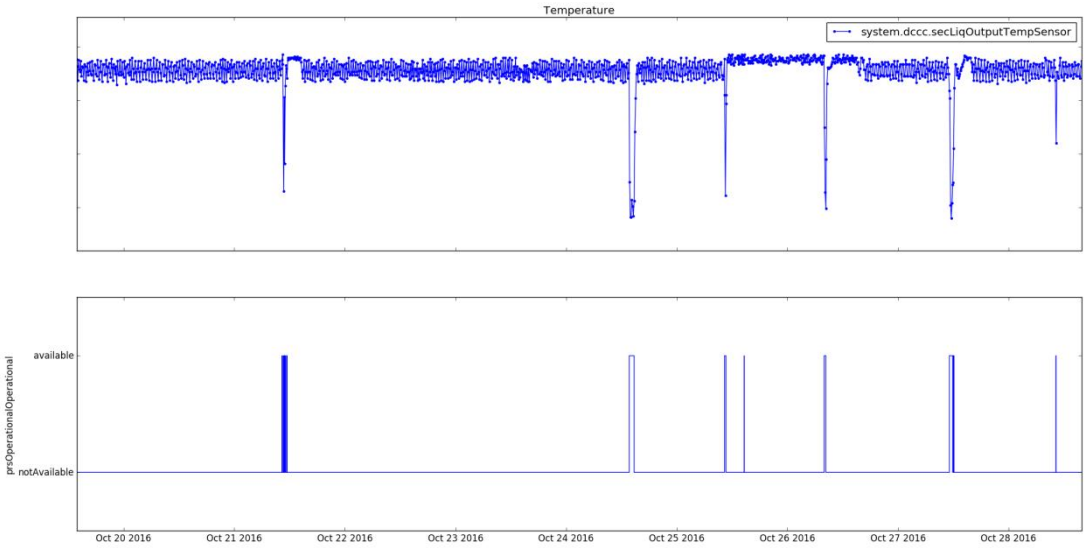


Figure 21: Temperature of coolant is dependent on system state.

Here it is evident that these spikes are not predictable without knowledge of the current state of the radar. For this reason the LSTM will also be fed a history of the states of the radar. A total of 14 states have been included.

10.5.2. Sampling

The time steps between data points are not always the same, however the network has no knowledge of time between data points. To solve this problem the data points are resampled into equal time steps of (classified) minutes which can be seen in Figure 22.

Another solution could be by adding an additional feature indicating the time that has been passed since previous data point. However, this makes the learning process of the network more complex as it requires the network to also learn the relation of different time steps. Therefore, the first solution has been chosen.

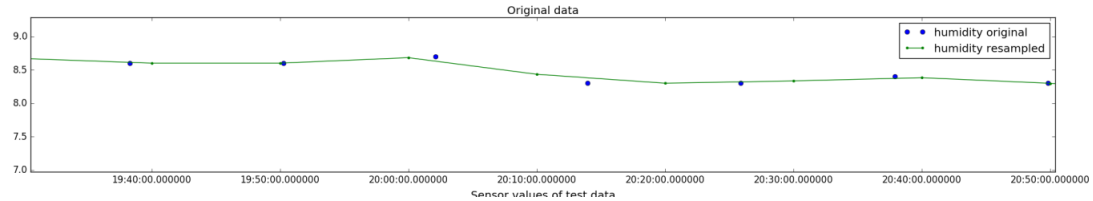


Figure 22: Linear interpolation to create equal time steps. Blue points show the recorded data and green line shows the linear interpolation between these points. The green points are fed into the LSTM and have an equal time step of (classified) minutes.

10.5.3. System states

There are several ways we can feed the network its history of the system state of the radar. A naïve solution would be to be to simply feed the network a *snapshot* of the system state at the sampling rate of the target time series. However, this results in only an approximation of the system state and some information about the system state can be lost. This will now be illustrated.

System states can change any moment in time. For example, when we feed the network the history of the previous hour, with samples of its state every (classified) minutes (thus a window of 6 points). In these (classified) minutes it is possible that the system changes its state many times. With the snapshot solution the network can miss changes of system states and only approximates the real system state with (classified) minute accuracy. This approximation gets smaller if we decrease the time steps between consecutive data points.

This is shown in Figure 23. The blue line shows the actual system state over time, while the red line shows what is fed into the network. It shows that the network doesn't see the change from state s0 to s1, instead it only sees the state changing form s0 to s2. Furthermore, its approximation of the duration in each state is not accurate. This becomes especially apparent when the system changes from state S2 to S3. The network thinks the radar is in S3 form (classified) minutes while in reality it is much shorter. This improves when we reduce the time steps between data points as can be seen in the right pane. However, this also results in more much more data points for the network which increases its computational complexity.

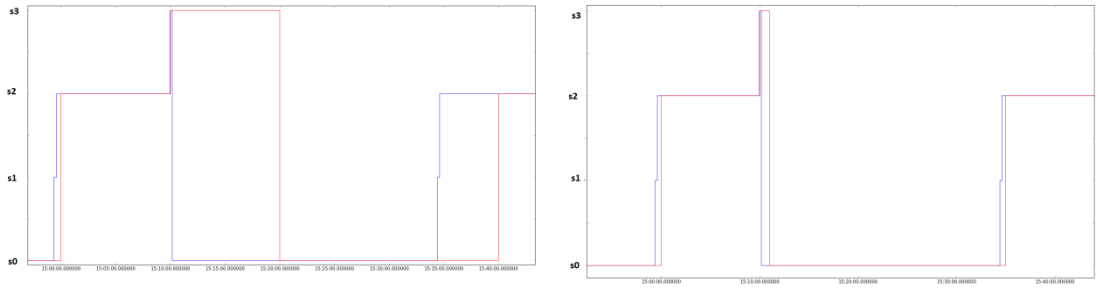


Figure 23: Missing states of radar in resampled data. Left time steps are (classified) minutes; right 1 minute. Blue line shows the actual system state, red its approximation.

Another solution which gives a better approximation of reality is as follows. Instead of feeding a snapshot of the system state, we can feed the network a summary how long it has been in each state between two data samples. In this solution the network will know exactly how long the radar has been in each state. However, there is still loss in information about the order of states the system has been.

Both methods for feeding the network the system state of the radar have been implemented. However, no significant differences have been found in the results between these two methods. The results with field data will be discussed next.

10.6. Experiments with field data

With these solutions were able to overcome the above mentioned obstacles and repeat the experiment with field data. The field data has been normalized as this often improves the learning rate of the LSTM. Furthermore, training of the LSTM network has been done with selected parts of the dataset, where no anomalies occurred during training.

We experimented with multiple sizes of the LSTM network. We settled for a network with 3 fully connected hidden layer as this gave the best results. To prevent the network from overfitting on the training data a dropout is added between layers (Srivastava, 2014).

Time window was manually chosen to include one period, in case of humidity sensor this is at least 12 hours. Different time windows were tested, the best window was selected: 12 hours.

Figure 24 shows the result of the network on a humidity sensor. The figure shows that the LSTM network is able reconstruct the time series very well. On the last day an anomaly occurs which is clearly visible in the reconstruction error (bottom pane). However, this humidity sensor is an easy case as it is relatively independent of the system state because the air dryers are always on. The network has learned to ignore the system state as similar results were acquired when repeating the experiment without feeding the network the system state.

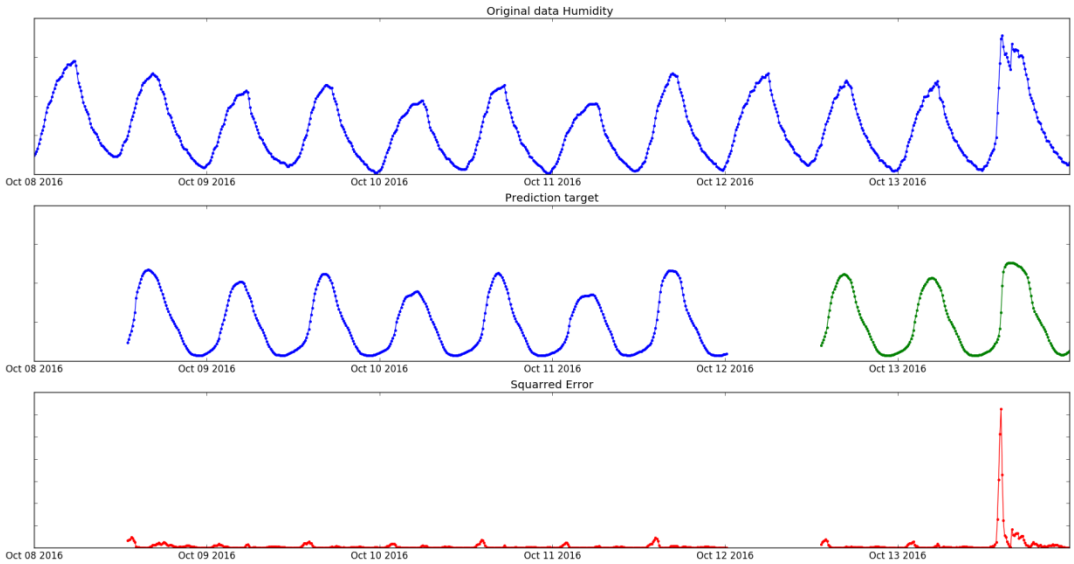


Figure 24: LSTM trained on humidity senor with a time window of 12 hours. The top pane shows the original data; middle pane shows the reconstruction; and the bottom pane shows the squared error between the original and reconstruction. Blue indicates training data and green the validation data.

A more difficult use case would be the temperature sensor in the radar, as it is dependent on the system state, as described above. In Figure 25 the results of reconstructing the temperature sensor are shown with and without system states.

The top pane shows the original values of the temperature sensor. The downwards spikes are the moments in time the radar starts transmitting radio signals. Without the system states the network simply results to the average temperature and is not able to predict any of the downwards spikes, which is to be expected. Only when a spike downwards spike is lasting for more than a few points the network reconstructs this as a small dip.

With the system states the network is able to reconstruct the downward spikes much better; however the reconstruction is a moment too late. This results in a large error and thus it performance is as bad as a LSTM network without states.

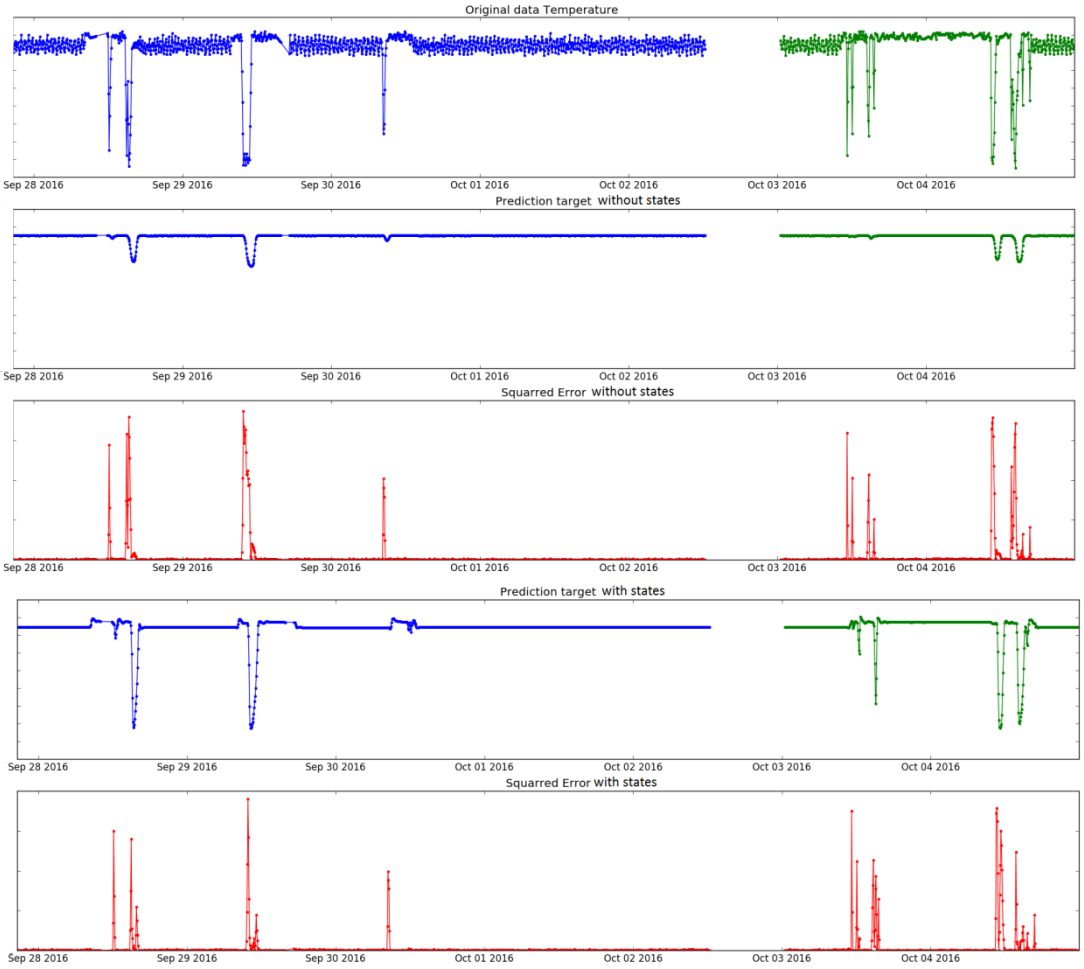


Figure 25: Comparison of temperature prediction without and with system states. The upper panel shows the actual train and test data. Panel 2 and 3 show the reconstruction and error respectively without system states. The bottom two panels show the reconstruction and error respectively with system states.

10.7. Additional experiment

Due to the disappointing performance of the network on its temperature prediction, an additional experiment has been performed to test the LSTM on multivariate data. The goal is to see if the network is able to detect an anomaly in that is introduced by a control signal.

Figure 26 shows the result of the experiment. The network is trying to predict the sine wave, colored in green. This sine wave is dependent on a control signal, shown as the black line. Its prediction is shown as the red line and the error is shown on the pane below. Whenever the control signal is turned on, the sine wave will jump up. This control signal is repeatedly randomly turned on and off.

The figure shows that during the training phase, shown on the left side of the figure, the network is able to learn the correlation of the control signal and sine wave. In the test phase and anomaly is introduced at the end, where the correlation between the target signal and control signal ends.

The network is fed a history of the target signal and the current control signal. The network is able to detect the anomaly. This demonstrates the LSTM networks ability to learn from multivariate data.

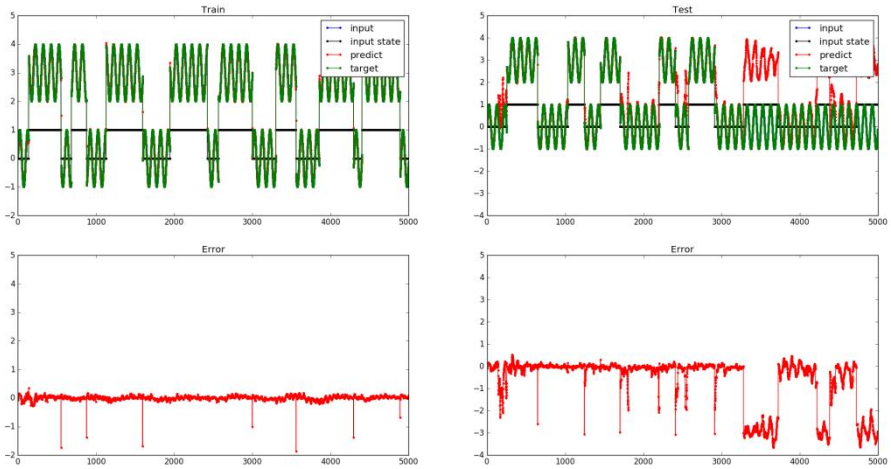


Figure 26: Anomaly with multivariate data.

10.8. Conclusions

LSTM networks have shown to be effective in learning the pattern of time series when presented with non-anomalous data and the time series is not dependant on external factors. Using the prediction error of the LSTM network allows for effective automatic detection of anomalies.

However, when the time series are dependent on external factors like the system state, training the network becomes more difficult. More research has to be done to on effective methods to better integrate external factors into the prediction. Several ideas are discussed in the future works.

10.9. Future work

Here, we list several ideas which can improve the model:

- Separate the prediction of spikes due to system states in separate model. Use the LSTM only for predicting the periodicity of the signal, which might improve its prediction accuracy.
- Predict multiple time steps ahead, instead of one. Use the combined prediction to calculate an anomaly score. This can make the system more robust against noise/single outliers which are not relevant.
- Apply better sampling techniques in the change of states, for example count the number of times the system changed state in the time step.
- Apply data augmentation techniques to increase the number of examples for the network to learn normal behaviour, this has been proven effective in (Cui, 2015).

11. CONCLUSIONS

The goal of this study was to explore and apply techniques in data mining and machine learning to learn more about the enormous dataset of field data of radar systems. Central was automatic discovering of anomalies in the data. We have explored the field of data mining and machine learning. Multiple machine learning and data mining techniques have been described, namely: classification, regression, clustering, dimensionality reduction and associate rule learning.

Multiple case studies have been identified in detecting anomalies with these techniques, three of which were explored in more detail:

- Creating a model of expected use of the radar.
- Comparing similar hardware components to detect failures.
- Discovering anomalies in time series via generative models.

Using a classifier unexpected use of the radar system can be automatically detected, allowing for a warning system for security officers of unauthorized use of the radar. The classifier has good performance with only a few features. Applying under sampling improves its performance. This model can be used by Thales to automatically detect suspicious usage of the radar which can be interesting from a security point of view.

Similar hardware components in the radar system have been compared with each other in order to detect failures. Using alarms originating from these components a fingerprint of the behaviour for each day of the component can be made. Using a classifier to try and identify the components, anomalous components can be revealed as they will be behaving differently than the rest. This classifier can be used as validation tool for the maturity of TX/RX tiles.

Finally, we have showed that it is possible to discover anomalies in univariate continuous time series using LSTM network as generative model with real field data. However, if the time series is dependent on several external factors, this process becomes more difficult and requires more extensive research. These models can be applied by Thales as automatic monitoring tools of the various sensors installed in the radar system.

12. BIBLIOGRAPHY

(n.d.). Retrieved 01 10, 2017, from Pandas: a python data analysis library: http://pandas.sourceforge.net

Abernethy, M. (2010, 04 27). *IBM DeveloperWorks*. Retrieved 11 14, 2016, from http://www.ibm.com/developerworks/library/os-weka1/

Agrawal, R. a. (1994). Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB, vol. 1215*, 487-499.

Batista, G. P. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 20-29.

Ben-Hur, A. D. (2001). Support vector clustering. *Journal of machine learning research 2, no. Dec*, 125 - 137.

Blaisten-Barojas, E. (2016). Unsupervised Anomaly Detection in Sequences Using Long Short Term Memory Recurrent Neural Networks.

Budalakoti, S. A. (2006). Anomaly detection in large sets of high-dimensional symbol sequences.

Chandola, V. A. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 15.

Cui, X. G. (2015). Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 1469-1477.

Ding, C. X. (2002). Adaptive dimension reduction for clustering high dimensional data. *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference*, 147 - 154.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM 55*, 79-87.

Eklöv, T. P. (1999). Selection of variables for interpreting multivariate gas sensor data. *Analytica Chimica Acta 381, no. 2*, 221-232.

Ester, M. H.-P. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Knowledge Discovery and Data Mining, vol. 96, no. 34*, 26–231.

Han, J. J. (2000). Mining frequent patterns without candidate generation. *ACM Sigmod Record, vol. 29, no. 2*, 1-12.

Hartigan, J. A. (1979). Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics) 28, no. 1*, 100 - 108.

He, H. a. (2009). Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering 21*, 1263-1284.

Heng, A. S. (2009). *Intelligent prognostics of machinery health utilising suspended condition monitoring data.*

Hipp, J. U. (2000). Algorithms for association rule mining—a general survey and comparison. *ACM sigkdd explorations newsletter 2*, 58-64.

Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6*, 107-116.

Hochreiter, S. a. (1997). Long short-term memory. *Neural computation 9*, 1735-1780.

Inmon, W. H. (2005). *Building the data warehouse.* John wiley & sons.

Izenman, A. J. (2013). Linear discriminant analysis. *Modern multivariate statistical techniques*, 237-280.

Jardine, A. K. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing 20, no. 7*, 1483-1510.

Jensen, C. P. (2010). *Multidimensional Databases and Data Warehousing.* Synthesis Lectures on Data Management.

Jones, E. T. (2001). *SciPy: Open source scientific tools for Python*. Retrieved 01 10, 2017, from http://www.scipy.org

Kaufman, S. S. (2012). Leakage in data mining: formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD) 6*, 15.

Maaten, L. v. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research 9*, 2579-2605.

Malhotra, P. A. (2016). Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv*.

Marchi, E. F. (2017). Recurrent Neural Network-Based Autoencoders for Acoustic Novelty Detection. *Computational Intelligence and Neuroscience*.

Medjaher, K. D.-M. (2012). Remaining useful life estimation of critical components with application to bearings. *IEEE Transactions on Reliability 61, no. 2*, 292-302.

Negnevitsky, M. (2005). *Artificial intelligence: a guide to intelligent systems.* Pearson Education.

Pang-Ning Tan, M. S. (2006). *Introduction to Data Mining.* Addison Wesley.

Patcha, A. a.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer networks*, 3448-3470.

Paul van Kempen, R. v. (2016). *Ontwikkel paden voor Predictive maintenance in service business.* Adversitement.

Pedregosa, F. G. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825-2830.

Peng, Y. M. (2010). Current status of machine prognostics in condition-based maintenance: a review. *The International Journal of Advanced Manufacturing Technology 50, no. 1-4*, 297-313.

Rakesh, V. (2009). *Hypertextbookshop.* Retrieved 10 25, 2016, from Data minig book: http://www.hypertextbookshop.com/dataminingbook/public_version/contents/chapters/chapter004/section001/green/page002.html

Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review 33*, 1-39.

Salfner, F. M. (2010). A survey of online failure prediction methods. *ACM Computing Surveys (CSUR) 42, no. 3*, 10.

Si, X.-S. W.-H.-H. (2011). Remaining useful life estimation–A review on the statistical data driven approaches. *European Journal of Operational Research 213, no. 1*, 1 - 14.

Sikorska, J. Z. (2011). Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing 25, no. 5*, 1803-1836.

Srivastava, N. H. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 1929-1958.

Weiss, G. M. (1998). Learning to Predict Rare Events in Event Sequences. *KDD*, 359-363.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*, 1550-1560.

Wold, S. K. (1987). Principal component analysis. *Chemometrics and intelligent laboratory systems*, 37-52.

Xu, R. a. (2005). Survey of clustering algorithms. *IEEE Transactions on neural networks*, 645-678.

13. APPENDIX

13.1. Confusion matrixes

Confusion matrixes for the classifiers described in section 8 CS1: Usage of radar systems.

Product A	Actual 'not used'	Actual 'used'	Total
Predicted 'not used'	5109	253	5362
Predicted 'used'	162	387	549
Total	5271	640	

Confusion matrix product A

Product A under sampled	Actual 'not used'	Actual 'used'	Total
Predicted 'not used'	511	54	565
Predicted 'used'	129	586	715
Total	640	640	

Confusion matrix product A under sampled

Product B	Actual 'not used'	Actual 'used'	Total
Predicted 'not used'	1338	93	1431
Predicted 'used'	201	360	561
Total	1539	453	

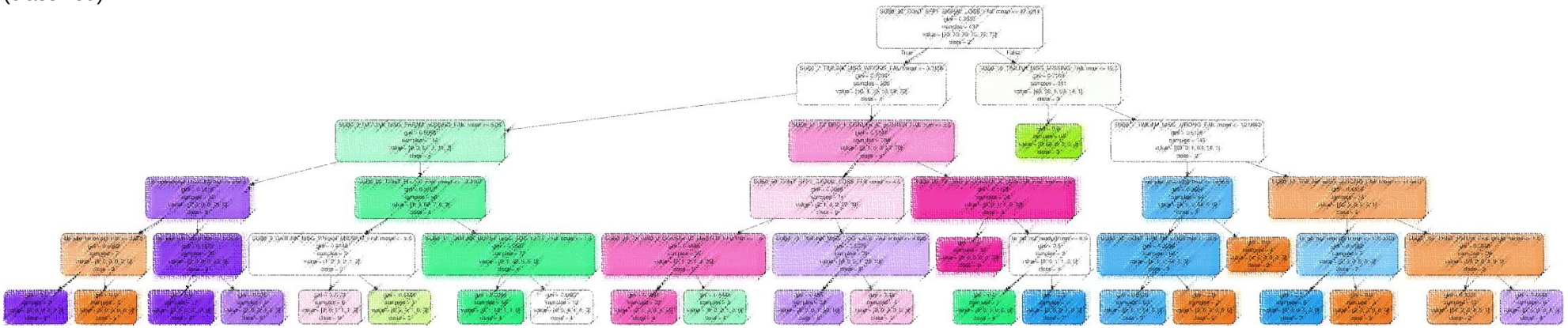
Confusion matrix product B

Product B under sampled	Actual 'not used'	Actual 'used'	Total
Predicted 'not used'	381	35	416
Predicted 'used'	72	418	490
Total	453	453	

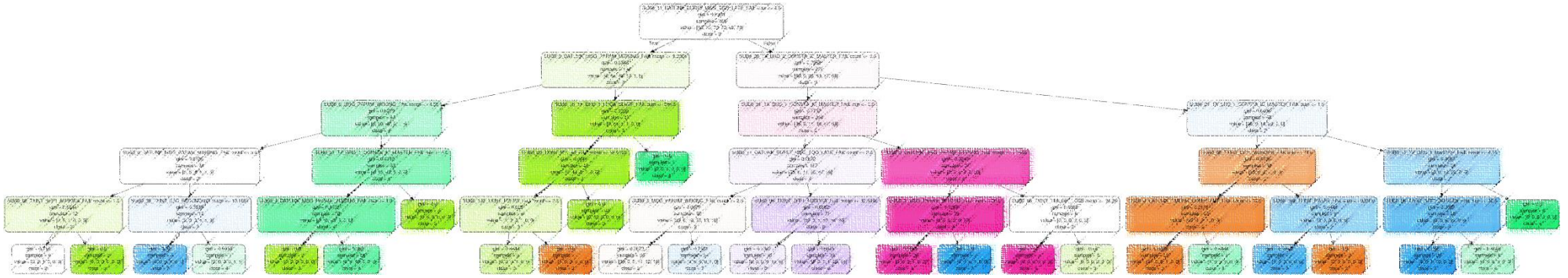
Confusion matrix product B under sampled

13.2. Decision tree model TX tiles

(classified)



Decision tree classifier for TX tiles



Decision tree classifier for TX tiles without 3 data link alarms