FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Predictive Modeling Using Railway Event Alarms System

Carlos Miguel da Cunha Rodrigues

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Prof. João Pedro Carvalho Leal Mendes Moreira

Second Supervisor: Eng. Hélder Ribeiro

July 30, 2015

U.PORTO

FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES | 2014/2015

A Dissertação intitulada

"Predictive Modeling using Railway Event Alarms System"

foi aprovada em provas realizadas em 20-07-2015

o júri

Presidente Professor Doutor Pedro Luís Cerqueira Gomes da Costa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Pedro Manuel Henriques da Cunha Abreu
Professor Auxiliar Convidado do Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra

Professor Doutor João Pedro Carvalho Leal Mendes Moreira
Professor Auxiliar do Departamento de Engenharia Informática da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Carlos Miguel da Cunha Rodrigues

Faculdade de Engenharia da Universidade do Porto

Abstract

Intelligent monitoring systems are being implemented along industries to automatic diagnosis procedures. Event alarms form on board systems stored by the monitoring systems were used as raw data with the objective of mine value information. The main goal of the project is to be capable of predicting failures in doors system of the train vehicles in order to replace periodical maintenance by preventive maintenance and reducing costs. The data was preprocessed and labeled transforming it in an unbalanced multi-class classification problem. It required a feature engineering process that may have been the key for the models understand properly the data. State of art classification algorithms were used, SMOTE technique implemented and finally used Stacking methods to achieve the best results.

Acknowledgements

I would like to thank my mother Margarida for all the conditions that she provided. Also to my friends, specially to Xico, Bernie, Saraiva and Driqs for the support during this project.

To my supervisors Professor João Pedro Moreira and Eng. Hélder Ribeiro for all the patience, advice and especially for their availability!

And to Professor Adriano Carvalho who introduced me to the fantastic company Nomad Tech.

Carlos Rodrigues

"You should be glad that bridge fell down.
I was planning to build thirteen more to that same design"

Isambard Kingdom Brunel

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Sensor networks are broadly used across industries in order to monitor equipment conditions. This is due to the growing capacity to store data and increase computational power, whilst having low cost and high flexibility and efficiency. In the railroad industry, conducting preventative maintenance on the rail cars is essential to maintaining the efficiency and safety of the rail network [1]. Intelligent monitoring systems are being implemented along industries for automatic diagnosis procedures. Commercial train vehicles are now highly instrumented with GPS (global position system) and communication systems, as well as on-board systems monitoring the life of various subsystems in the vehicles. These sensors provide a real time flow of data consisting of geo-referenced alarms, called events, along with their spatial and temporal coordinates. This data is transferred wirelessly towards centralized servers where it is stocked and exploited [2]. After it has extracted all this information, it bids and orders it on a database that there are a potential to mine this information in order to extract value.

1.1 Motivation

Failure prediction refers to the problem of finding patterns in data that do not conform to expected behaviour. These non-conforming patterns are often referred to as anomalies, outliers, discordant observations, exceptions, aberrations, surprises, peculiarities or contaminants in different application domains. Of these, anomalies and outliers are two terms used most commonly in the context of anomaly detection; sometimes interchangeably. Anomaly detection finds extensive use in a wide variety of applications such as fraud detection for credit cards, insurance or health care,intrusion detection for cyber-security, fault detection in safety critical systems, and military surveillance for enemy activities. Predictive Maintenance is about predict machine failures and plan maintenance measures accordingly.

With the success of this project is expected to:

- **Avoid Unplanned Maintenance**- Implement predictive maintenance to predict future equipment malfunctioning and failures and minimize the risk for unplanned disasters putting your business at risk.

- **Improve Maintenance Planning**- Optimize your maintenance schedules, reduce costs by cautiously allocating maintenance resources and reduce mean time-to-repair.
- **Reduce Maintenance Costs**- Don't waste money through over-cautious maintenance. Only repair equipment when repairs are actually needed.
- **Identify Failure Causes**- Find causes for equipment malfunctions and work with suppliers to switch-off reasons for high failure rates. Increase return on your assets.

Many studies predict failure with unfeasible data. Namely data that can not be extracted easily or without huge extra costs and derived problems. For instance, knowing the mechanic system of the doors and using the data of the pneumatic valves appears to be the best approach to predict those failures. However, to have this information, it would be necessary to the railway companies to install more data sensors systems which also would need maintenance. Consequentially, solving the problem this way would actually increase the problem.

The motivation of this project is to predict components failures in the doors systems using the alarm event data from the on board systems. This is the data available in almost single train vehicles from the companies that can only have slight variances due to different on board systems providers (brands).

1.2 Objectives

At an abstract level, an anomaly is defined as a pattern that does not conform to the expected normal behavior. A straightforward anomaly detection approach, therefore, is to define a region representing normal behavior and declare any observation in the data which does not belong to this normal region as an anomaly. Nut several factors make this apparently simple approach very challenging:

- Defining a normal region which encompasses every possible normal behavior is very difficult. In addition, the boundary between normal and anomalous behavior is often not precise.Thus an anomalous observation which lies close to the boundary can actually be normal, and vice-versa.
- In many domains normal behavior keeps evolving and a current notion of normal behavior might not be sufficiently representative in the future.
- Availability of labeled data for training/validation of models used by anomaly detection techniques is usually a major issue.
- Often the data contains noise which tends to be similar to the actual anomalies and hence is difficult to distinguish and remove.

Considering all these points, the objective of this project is to chose a model capable of have these notions and, therefore **predict door failure** with enough time to act defensively. We propose

a data mining approach to process the data, machine learning algorithms to be trained with the data and considerations in regard of the final evaluation and the inherent data problems are exposed in the following chapters. We identify as the principal problems the unbalance of the data and the low quality predictors which make difficult for most of the models to predict correctly. The huge amount of predictors versus the number of samples in the dataset adds difficulties to the model train process and the predictors missing theoretically available in the datasheet not present in the dataset can be a problem in the future when a generalization of the model to others vehicles is required. Also is important to refer that this was a classification problem because the data was labeled by a specialist.

1.3 Company Introduction

This Thesis is developed along with Nomad Tech, a joint venture of Nomad digital, a specialist in fleet connectivity (such as WiFi) and EMEF, the Portuguese Railways company for rolling stock maintenance. They provided resources and their own data logs in order to accomplish the task. The data is from a Norwich company who uses their remote condition motorization software.

Nomad Tech is installed in UPTECH, in Oporto and it has a 15th member team.

1.4 Document Structure

The document is organized as follows. Chapter 2 introduces the company for this work. It also describes the problem and the engineer requirements of the project. In chapter 3 it is a small survey of recent works done on this industry. On of the papers presented is also a suvey with more details of the research done. Chapter 4 is the state-of-art of classification problems, describing the basic information of the most know families of algorithms. The approach that we propose to solve the problem is presented on chapter 5, being the most extended chapter of all because it have described all the steps done in order to pre-process the data, train the models and evaluate them. It is followed by chapter 6 that has some considerations to take in account with regard to improvement of the model results. Then in the end, chapter 7 where all the results are discussed regarding the inherent problems of the data.

Chapter 2

Maintenance systems in Railway Industry

Nomad Tech presents itself as a global provider of technological solutions to the railway industry. Maintenance, energetic efficiency, and power electronics are areas in focus granting their clients the knowledge of the state of their fleets, improving the security, quality and performance or even reducing costs.

Figure 2.1: Nomad Tech

In this chapter it will be introduced Nomad Tech, The company who support this thesis with resources, data, work place . Questions such as: what it is Nomad Tech? what does the company do?

2.1 Company Business

Nomad Tech global capabilities can provide strategic, engineering and technological solutions to help support rolling stock maintenance optimization. Nomad Tech solutions are strongly supported by the railway knowledge of its staff and proven results in the field.

The Nomad Tech approach for the Reliability Centred Maintenance (RCM) methodology allows the maintainers to leave behind the traditional maintenance approach. based in standard suppliers maintenance plans and periodic systems/equipment overhauls, and embrace a top notch maintenance methodology which allows the achievement of a proactive maintenance cycle and optimized working plan procedures, both focused on safety, reliability, availability and cost reduction.

Figure 2.2: Train monitored by Nomad

The company is deeply experienced in training and support the rolling stock maintainers in the application of the Reliability Centred Maintenance (RCM) methodology, with a unique, practical and simple approach and extremely cost-effective, assuring the KPI's improvement.

Beyond its IT resources expertise, Nomad Tech has highly skilled staff with decades of real life experience in Rolling Stock and the Rail Industry, allowing to truly understand the customer's real needs and expectations, thus delivering all-around solutions, culminating in online, real-time, decision-aid support tools for the train drivers, the maintenance workshop, and the fleet management teams.

They deliver customized tools according to the customer main goals, assets and data. When delivering a product, Nomad Tech supports the Customer in the implementation and integration of the tools in its business strategy.

As a technology company, Nomad has the need to adapt to the new trends and news of technological market. Data mining is a concept that fits in the Nomad Tech solutions. It can replace old algorithms/methods for schedule maintenance or build decision support systems based on old data. Data mining approaches can find hidden patterns and it build models free from human errors

To innovate, Nomad has several partnerships which highlights Toshiba and FEUP allowing me to develop this project.

2.1.1 NT Maintain

This family of solutions combines dynamically, the application of RCM maintenance methodology (reliability centered maintenance) to ROCM tool (Remote Online Condition Monitoring) developed by Nomad Tech, highly customizable; presents a general and universal

view of the customer's fleet status, sense of predictive way - in real time - (potential) failures on trains (systems and components), constituting a support tool to maintain the surrounding material that operators and Maintenance managers can remotely access and / or can be connected to other management tools that the customer already has. It is thus possible to paradigm shift in maintenance, abandoning the merely reactive and periodic procedures, and adopt rather a supported and progressively, a proactive approach, optimized and that takes into account the real state of components and systems, enabling the extent of its useful life, without compromising - and even improving - safety indicators, reliability, availability and cost. These solutions were for example applied to the entire fleet of Alpha tilting trains in Portugal allowing an increase in the fleet availability of 20% and accuracy in the order of 30%; these impressive success rates have led to Nomad Tech to Norway, where a version of this system for the entire fleet of NSB was developed - Norwegian Railways.

There are two more products available, NT Eco and NT Power with different focus, however I emphasized NT Maintain solution because maintenance is the problem addressed.

2.2 Problem

At regular intervals train sets have scheduled depot visits, to have preventive and corrective maintenance performed but also to inspect, detect and analyse faults etc. During operation, on-board staff and drivers report faults through handheld devices and via telephone to operational dispatch.

The preventive maintenance includes regular inspections and maintenance procedures, based on a generic average wear which is translated into the maintenance interval in km's or days.

In reality the actual wear and the maintenance needs rarely fit the generic model. Therefore from an economic perspective, this implies that maintenance is sometimes performed too early and sometimes too late. This creates unnecessary maintenance and operating costs and lower availability. Tools for monitoring vehicle status exist but are not utilized to the extent that they could.

Updated status monitoring and analytics as a cornerstone in improving availability, reliability and reducing cost as well as improving safety there fore the objectives/end state vision point to:

- Reduce maintenance costs by performing maintenance when needed rather than after a fixed interval
  - Increase maintenance intervals
  - Reduce maintenance costs through fewer manual inspections
  - Validate performed maintenance or changes to maintenance intervals
- Detect faults before causing breakdowns (figure 5.1)
- Increase plan horizon maintenance

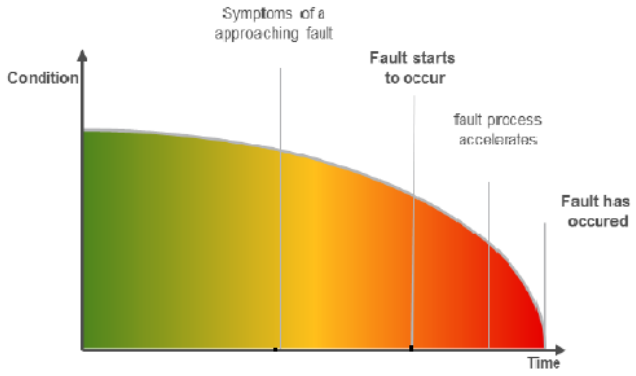


Figure 2.3: Acceleration of fault sequence

In the next section it will be present the requirements demanded by companies who contract Nomad Tech services. The entities will be preserved as anonymous.

2.3 Requirements

2.3.1 Client Requirements

In order to improve the company's performance, Nomad Tech gathers information about the clients requests. On the one hand they have existing customers asking for improvements on their actual system. On the other hand they have future possible clients who provide the systems requirements either by email or by official documents. This client network allows them to understand the needs that are common to each client. Table 2.1 are shown the requirements of three different clients.

Requirements		
Client 1	Client 2	Client 3
Predictive Component Failure	Pattern	Statistical Analysis Functions
-	Statistical Trend	What If Analysis
-	-	Trending

Table 2.1: Anonymous Clients Requirements

2.3.2 Company Requirements

Along with the client requirements it is important to understand also the needs of the company. With the specification of both it will be possible to understand the problem, normalize it and build a solution that fits both the client and the company. It consists of an improvement of their software that can be adapted to their maximum current and future projects.

The first requirement is the capability to recognize patterns using data mining techniques. It uses data from the past in order to predict the future, in this case to predict some failure before it happen. Data Mining uses machine learn algorithms that learn from the examples what is the normal behavior and the abnormal that origins failures. The end state vision is to replace the traditional maintenance methods for these systems, in other words, replacing schedule maintenance for situational maintenance. Planned maintenance has two huge problems, namely due to the fact it happens in interval of time with the same length, for example, first week of every month or 6 in 6 weeks. The problems associated are the replacement of equipment in good condition or the late replacement of worn equipment. Both case causes extra expenses to the companies and in a large scale "every penny counts".

Adaptability and integration capacity in the Nomad platform are the second requirement. The first propriety demands that the new data mining module can be used in different fleets and produce always acceptable results. Nomad has its software implemented in different types of vehicles and so it is important to understand it in order to choose the best approach. If not they have the risk to develop a biased solution.

Finally, the software module must use data mining concepts to operate and predict the failures. The program must produce results fast enough to be possible to replace the component without further costs such as delays. The interval must be large enough to plan a maintenance intervention and regularize the situation.

Chapter 3

Data Mining on Railway Systems

The availability and reliability of the railway industry is a challenge to the companies. Decision making, optimization and maintenance are right now based in technical and economic information as well as knowledge and experience. The instrumentation of railway vehicles as well as infrastructure by smart wireless sensors has provided huge amounts of data that, if exploited, might reveal some important hidden information that can contribute to enhancing the service and improving capacity usage. In order to improve their services, railway companies have been fund applicative research and projects in this context. The table 3.1 from paper [2] has a survey of what was done on the train vehicle subsystems in the the recent years. It is divided in the paper (the references in the table are from its paper), vehicle subsystem and the methodologies used in the paper.

Reference	Subsystem	Methodologies Used
[12], [17], [25]	Doors	Ontology-based methods, Neural networks, Classification, fuzzy logic, statistical learning
[5]	Axle	Statistical methods
[29], [31], [46]	Rolling element bearings	Envelopment analysis, Squared envelope spectrum, 2nd order cyclostationary analysis, Spectral kurtosis, Empirical mode decomposition, Minimum entropy deconvolution
[20], [30], [43]	Wheels	Dynamic modelling, Signal processing, Wavelet transform methods, Fourier Transform, Weigner-Villa Transform

Figure 3.1: Examples of recent research work along with the methodologies used for the condition inspection of various train vehicle subsystems

3.1 DB Schenker Rail AG

DB Schenker Rail AG is a rail freight company that has such a closely meshed network throughout all of Europe and beyond. The company aim to make transports better, more efficient and more reliable with customer-oriented and tailored solutions.

In order to accomplish that the company got into Data Mining area and Failure prediction. They claim to record a continuous logfile of diagnostic data for further analysis. It can have a significant impact in costs reduction or increase train availability. Each year this company transport more than 400 million tons of goods using a fleet with around 3500 trains. Reliability is a crucial issue, since delivery dates have to met, and complications may lead to delays and increase transport cost. Mechanical failures of the train itself is one of the main reasons not reaching destination on time. Failures of a mechanical component are costly, since not only the component itself has to be replaced or repaired, but also the train is standing.

Accordingly, the company developed a study [3] to predict the component failures. As input they used the logfile, in order to analyze which failures are important for the predictive task and could be actually predicted, for instance a component has any sensor attached it is impossible to get information and predict it status. So, on their study, they chosen as specification to focus on failures caused due to deterioration, component failures that has a significant cost or affect the train motion and components with status data. The 3 most frequent failures on DB schenker Rail AG data base were:

- instand setzen (repair motor control, found 142 times)
- instand setzen (repair guiding system, found 126 times)
- LZB Empfangsantenne einstellen (repair antenna of guiding system, found 93 times)

After it, they used some heuristics where they pre-processed the data, merged the data, labeled it and post processed it. The next step was to use some classification algorithms as JRIP, J48, RandomForest, and SMO. Having those milestones done they did experiments and result analysis where they conclude that predict components failure is indeed possible. However the highest accuracy was obtained when they treated the problems, i.e. every diagnostic code separately, as a different classification problem. Also prediction quality would significantly increase when given the data quality is improved. Their approach to feature engineering is similar to the case study of Sammouri in the previous section 3.2

3.2 Pattern recognition approach in doors

In this section, it is presented a paper [2] with an approach to predict train vehicle component failures with 6 month data from TrainTracer database. They use the state of art approach, preprocessing the data by creating features and label those features. They do not consider the problem of unbalanced data to the process but they realize that it can not take into account the

repartition of the classes of the problem so they use the recall and precision measures to have a better understand of the results. The SVM method with radial kernel, K-Nearest Neighbour, Naive Bayes and Neural Networks were trained using cross validation.

The obtained results have shown the effectiveness of the proposed approach to predict infrequent target events. The attribute selection method has led to an increase in the performance of the four classifiers, which highlighted also the fact that many of the initial attributes were misleading the classification process and can be considered as noise which needed to be pruned out to sharpen the performance [2].

Chapter 4

Predictive Modeling

4.1 Classification Algorithms

In this section, it is described briefly the algorithms thAat we used in chapter 5. It was conducted some pre-eliminate experiences in order to chose suitable algorithms. The Naive Bayes method represents a important family used in many Data Mining works however it was revealed not suitable for our data.

4.1.1 Tree based Models

Tree based models are models which provide as result a model based on logical tests on the input variables. The aim of this models is to partition data into smaller, more homogeneous groups. This partitioning consist of nested if-then statements, logical tests on the variables to find the which one will maximize the gain of the model. In each partition the same prediction is assigned to its cases [4]. Tree based models are known by their computational efficiency, interpretable models , intrinsic feature selection and embedded handling of unknown variable values [5].

- Test on numerical predictiors take the form $x_i < \alpha,\ with\ \alpha \in \Re$
- Tests on nominal predictors take the form $x_j \in \{v_1, ..., v_m\}$
- Each path from the top (root) node till a leaf can be seen as a logical condition defining a region of the predictors space.
- All observations “falling” on a leaf will get the same prediction the majority class of the training cases in that leaf for classification trees
- The prediction for a new test case is easily obtained by following a path from the root till a leaf according to the case predictors values

Trees uses the Recursive Partitioning algorithm to determine when to stop growing the tree (termination criterion), which value to put on the leaves and how to find the best split test. This algorithm uses typically the Gini index, the Gain ratio or the entropy as error rate criterion.

4.1.1.1 C5.0

C5.0 is the new improved version of C4.5 classification method [6]. It can produce a tree or a rule-based model. Employs a top-down, greedy search and it uses *information theory* [7] to split. The information gain is based on the decrease in entropy after a dataset is split on an attribute.

$$gain(split) = info(prior\ to\ split) - info(after\ split)$$

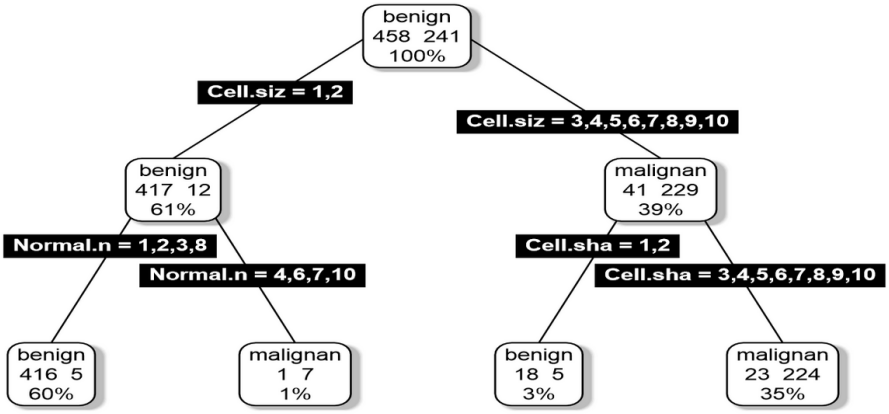


Figure 4.1: Classification Tree

4.1.2 Artificial Neural Networks

IT is a family of statistical learning models inspired in the biological animal brain [8]. An artificial neural network (ANN) is composed by a set of units (neurons) that are connected. These connections have an associated weight. Each unit has an activation level as well as means to update this level. Some units are connected to the outside world. We have input and output neurons. Learning within ANNs consists of updating the weights of the network connections [5].

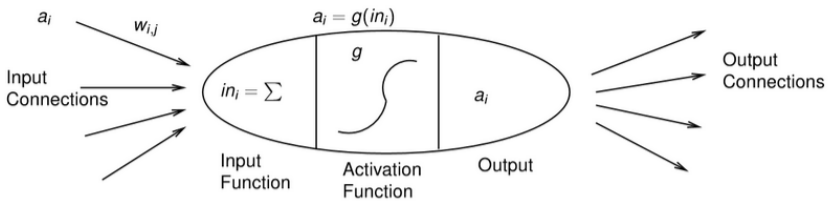


Figure 4.2: Neuron

The neuron 4.2 receives the input impulses and calculate its output as a function of these impulses. To calculate it use a linear computation of the inputs and weights and then a non linear computation, namely an activation function. There are different activation functions such as the

Step function, the Sign Function and the Sigmoid Functions. The two most known ANNs algorithms are Feed Forward networks and Recurrent networks.

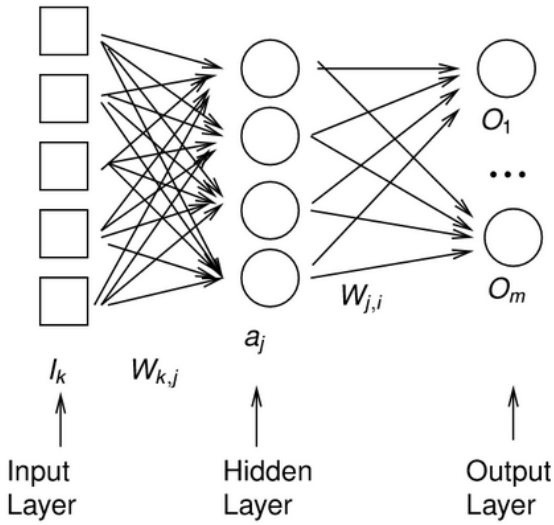


Figure 4.3: Feed-forward Multi-Layer Architectures

Figure 4.3 is an example of a multi-layer Feed Forward network [9] and have an implementation in R. Is it multi-layer because it has hidden layers. The most famous algorithm for learning ANNs is **The Backpropagation Algorithm**:

- Each example is presented to the network
- If the correct output is produced nothing is done
- If there is an error we need to re-adjust the network weights
- This adjustment is simple in perceptrons as there is a single connection between the input and output nodes
- In multilayer networks things are not that simple as we need to divide the adjustments across the nodes and layers of the network

4.1.3 Support Vector Machines

Support Vector Machines (SVM) obtains a very simple and linear separation of the cases (binary problems) however in most real world problems the cases are not linearly separable. Original data is mapped in a higher dimension coordinate system where the classes are linearly separable.

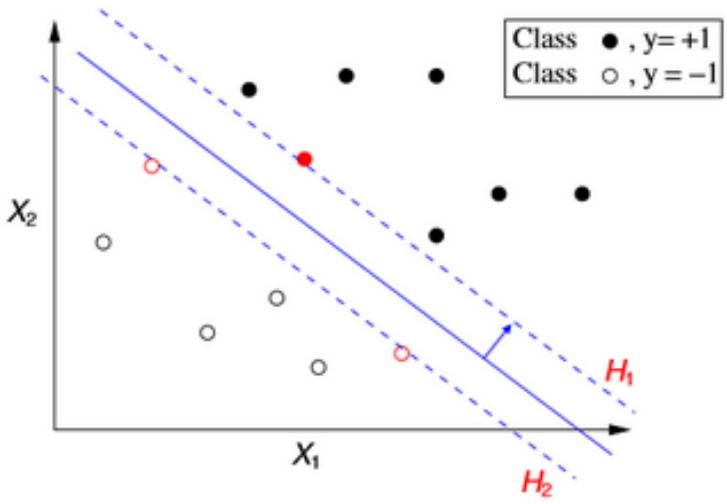


Figure 4.4: SVM problem

There is an infinite number of hyperplanes separating the classes and the objective is to chose the one who ensures the better classification accuracy on unseen data. The approach is to search for the **maximum margin hyperplane**. Figure 4.4 shows an example, all the cases that fall on the hyperplanes H1 and H2 ae called the support vectors but removing all other cases would not change the solution. So a very high dimension space is needed to find the maximum margin hyperplane [5].

When the problem is multi-class it resumes to solve several binary classification tasks, essentially finding the support vectors that separate each class from all others.

4.1.3.1 Kernel

SVM use quadratic optimization algorithms to find the optimal hyperplane that maximizes the margin which separates the classes. However it is necessary to move the data in to higher dimensional spaces where the computation of the quadratic method is heavy. Therefore, the kernel trick is used applying kernel function replace the complex calculations of dots into quadratic optimization.

4.1.4 Ensemble Models

Ensembles are collections of models that are used together to address a certain prediction problem. For some complex problems it is hard to find a model that explains all observed data so averaging over a set of models typically leads to significantly better results [5]. The idea is to employ multiple learners and combine their predictions. If we have M models with uncorrelated errors, simply by averaging them the average error of a model can be reduced by a factor of M [10].

4.1.4.1 Random Forest

Random Forest is an ensemble model that combines the idea of sampling the cases and the predictors. Sampling the cases consists in Bagging [11] and sampling the predictors consists in generate a diverse set of models by using different random chosen predictors.

Random Forests consist of sets of tree-based models where each tree is obtained from a bootstrap sample of the original data and uses some form of random selection of variables during tree growth.

4.1.4.2 AdaBoost

Adaptive Boosting is another ensemble algorithm. To improve the performance of a base algorithm, it uses an iterative process where each new iteration of the algorithm the new models are built to try to overcome the errors made in the previous iterations. The weights of the training cases are adjusted at each iteration [12].

4.2 Data Spliting

Given a fixed amount of data, it is important to understand where and how to use the data to build the predictive model. Generally the initial data must be divided, the data used to build the model can not be the same to evaluate the model performance otherwise the result would provide an unbiased sense of model effectiveness [4].

The idea is to divide the initial data set in two partitions, generally one is called **training set** and the other one **testing set**. Another split is also done in the training set splitting it in the new training set and **validation set**, figure 4.5. The testing set is kept aside and exclusively used to measure the final model performance while the validation set measures mid term performance in the model creation. Some cases where the data available is not large enough, the test set is avoid or there are cases where the test does not have sufficient power or precision to make reasonable judgements [13].

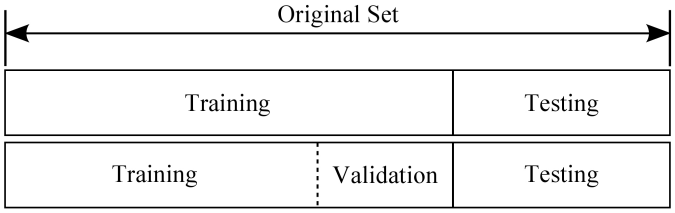


Figure 4.5: Data Split

If it is necessary a test set there are some methods to split the samples. On one hand we can use nonrandom approaches, for example, if we want to predict football matches outcomes on the Portuguese league and then we test in different leagues to understand how well the model generalizes.

On the other hand, in the most cases, it is propense to use random sampling to create similar data, homogeneous training and testing set. The simplest way is to take a random sample to create the data sets. Another way is to have in account the outcome, stratified random sampling, it is used when the outcome class is not balanced, for instance, class A is 90% of the data. In those cases, random sampling within subgroup is done to keep the same outcome balance in every subset. It is as well possible to do it but taking in account the predictors values. *Dissimilarity sample* approaches based on maximum and minimum is the most common processes[14].

4.2.1 K-Fold Cross-Validation

The data is randomly partitioned into k sets of roughly equal sizes. The first subset is held out to validate the model prediction and the others are used to train. The first subset is returned to the training set and procedure repeats with the second subset held out, and so on. Then a performance summary is done to understand and evaluate the model utility.

There are no rules to define the k, however the choice is normally 5, 10 or 20. How larger is the k value the difference in size between the train set and the test set gets smaller. Also the bias gets smaller with large k values. The problems that come with large k values is the computing time increment and if the data set is not large enough the subsets created may not describe the population. In figure 4.6 is an example how to cross validation works with 3 folds.

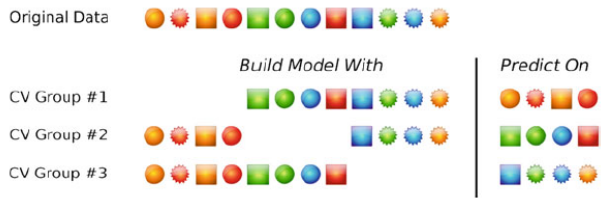


Figure 4.6: 3 fold Cross Validation

Cross Validation is applied to: validating the robustness of a particular mining model, evaluating multiple models from a single statement.

4.2.2 Windowing

Windowing is the process to visualize temporal data. Usually in temporal data the order of the data needs to be considered because the future instances may depend on the past ones. Two known methods are Growing Window or Sliding Window.

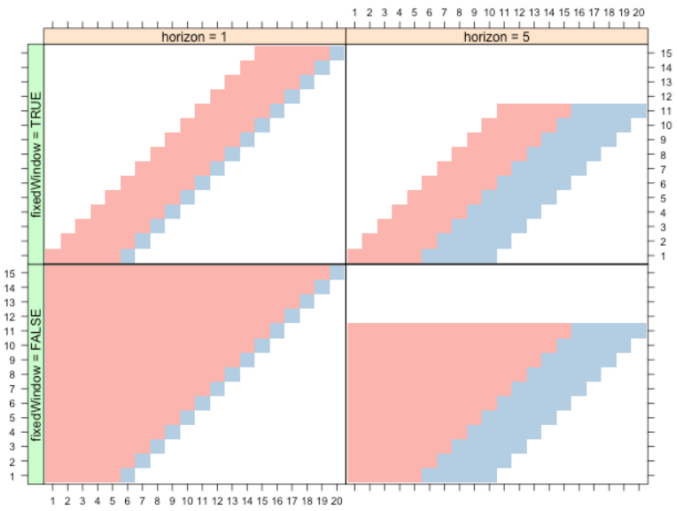


Figure 4.7: Sliding Window vs Growing Window

Figure 4.7 illustrate the sliding window that is when we do not have a fixed window and the Growing window where the window is fixed. Despite it both have two parameters, window length and horizon. Horizon is what you pretend predict so the length of this parameter represent the number of samples targeted to prediction. The difference between both is that in sliding window the train set has always the same size and it is moving forward as the model is predicting. On the other hand in the growing window, as the name suggest, the training set increases over time.

The choice of window length involves a balance between two opposing factors. A shorter window implies a smaller data set on which to perform your estimations. A longer window implies an increase in the chance that the data-generating process has changed over the time period covered by the window, so that the oldest data are no longer representative of the system's current behavior.

To put it in other terms, shorter windows increase your parameter risk while longer windows increase your model risk. A short data sample increases the chance that your parameter estimates are way off, conditional on your model specification. A longer data sample increases the chance that you are trying to stretch your model to cover more cases than it can accurately represent. A more "local" model may do a better job.

Your selection of window size depends, therefore, on your specific application including the potential costs for different kinds of error. If you were certain that the underlying data-generating process was stable, then the more data you have, the better. If not, then maybe not.

4.3 Performance Measures

Classification problems can generate two types of predictions, probabilities (between 0 and 1 and sum to 1) or a predicted class, which can be seen in the form of a discrete category, for example a **yes** or **no** prediction. The last type is usually used when a decision needs to be done. Filtering spam in emails is an example where the prediction needs a definitive judgement for each email.

However, the probability can be very important for gauging the model's confidence about the predicted classification. An email with a 51% probability to be spam will be classified the same way as an email with 98% probability. Although both emails would be judged the same way by the filter we are more confident that, the second message was, in fact, truly spam [4].

4.3.1 Accuracy

A classification problem is, typically, evaluated by a confusion matrix as illustrated in figure 4.8 [15]. The rows are the actual class, the true class of a sample, and the columns are the predicted class by the model. In the following matrix, TP stands for True Positives and it is the number of positive examples correctly classified while TN (True Negative) are the negative examples correctly classified. FP (False positive) is the number of negative examples incorrectly classified as positive and FN (False Negative) is the number of positive examples incorrectly classified as negative. Accuracy is a standard measure of the predictive model, basically it is the total percentage of what the model has predicted correctly.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

	p' (Predicted)	n' (Predicted)
p (Actual)	True Positive	False Negative
n (Actual)	False Positive	True Negative

Figure 4.8: Confusion Matrix

4.3.2 Precision and Recall

From the confusion matrix in figure 4.8, it is also possible to extract other performance measures, **Precision** and **Recall** [16].

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

The mail goal for learning from imbalanced data sets is to improve the recall without hurting the precision. However these two measures can be discordant when the true positive for the minority class improves, the false positive can inherently increase as well, this effect cause a precision reduction. The F-value metric is a measure that ensemble the trade-off of precision and recall and returns a value which characterize the "goodness" of a classifier in the presence of rare cases. The F-value is trade-off among TP,FP and FN [16].

$$F - value = \frac{(1+\beta^2) * recall * precision}{\beta^2 * recall * precision}$$

Usually β has a value of 1 and the F-value can be called as F1-score.

4.3.3 ROC

The ROC (Receiver Operating Characteristic) curve is a standard technique to classify the model performance taking in attention the trade-offs between thr TP and FP. The AUC (Area Under the Curve) is an accepted performance metric for a ROC curve. [17] [18]

On the ROC curve the X-axis represents de percentage of FP = FP/(TN + FP) and the Y-axis is the percentage of TP = TP/(TP + FN). The ideal Point on the ROC curve would be when it reachs the y = 1 and x = 0, that represents a classifier that predicted correctly all positive samples and no negative instances are misclassified as positive.

Comparing ROC curves can be useful in contrasting two or more models with different predictor sets (for the same model), different tuning parameters (i.e., within model comparisons), or complete different classifiers (i.e., between models). The optimal model should be shifted towards the upper left corner of the plot. Alternatively, the model with the largest area under the ROC curve would be the most effective. The ROC curve is only defined for two-class problem but has been extended to handle three or more classes [18],[19].

4.4 One-Against-All

One-against-all (OAA) is a standard approach of breaking multiclass classification problems into several binary classification ones. It does so, by building each classifier for a specific class (positive), it converts all the remaining classes into negative. That is, it builds n models for a dataset with n classes. For each binary problem, we need to build a classifier for a specific class. This classifier only makes prediction for a test example whether it belongs to the specific class or not. After n predictions are available for a test example, the classifier classifies it into the class where it has the highest probability [20],[21].

OAA has an obvious drawback. When OAA creates the training set for a specific class, it converts all the rest examples belonging to other classes into an arbitrary class Y. This conversion will create the imbalanced issue, which makes the problem more complicated.

There are two more approaches, one-against-one (OAO) and and all-at-once (AAO) which will not be discussed because it is out of scope. The packages used in the experiments are implemented with OAA approach.

4.5 Unbalanced Data

Data in unbalanced when the class of interest is much smaller or rarer than normal behaviour (majority class). What is the correct distribution for a learning algorithm ? Natural distribution is not always the best distribution for learning a classifier[22].

Also, the imbalance in the data can be more characteristic of "sparseness" in feature space as shown in the subsection 5.3.4. This problem is prevalent in many applications, including: fraud detection, risk management, text classification, medical diagnosis, and maintenance monitoring. but there are many others. It is worth noting that in certain domains (like those just mentioned) the class imbalance is intrinsic to the problem. Various re-sampling strategies have been used to overcome this issue such as random oversampling with replacement, random undersampling, focused oversampling, focused undersampling, oversampling with synthetic generation of new samples based on the known information, and combinations of the above techniques[22],[23].

The under and over sampling method have their various shot-comings. The random undersampling can potentially remove certain important examples, and random oversampling can read to overfitting. However, there was an improvement on this area and new techniques emerged such as SMOTE (Synthetic Minority Oversampling Technique) or Tomek Link approach[24],[25].

4.5.1 SMOTE

SMOTE, figure 4.9 consists on oversampling by replicating the minority class and by operating in the "feature space" rather than "data space"[15]. Namely, oversampling by taking each minority class sample and introducing synthetic examples taking in account the k nearest neighbours. The k nearest neighbours are chosen randomly and the synthetic samples are generated by taking the difference between the feature vector (sample) under consideration and its nearest neighbour. Multiply this difference by a random number between 0 or 1, and add it to the feature vector under consideration. For nominal cases, the majority vote is used in the nominal neighbours. It causes a generalization of minority class[15].

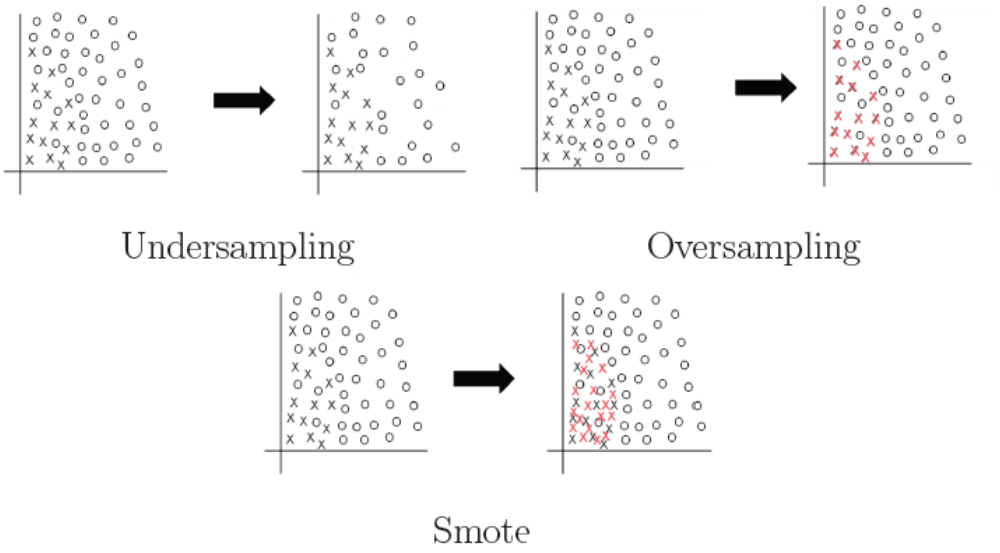


Figure 4.9: Sampling methods

There are also other methods to solve the unbalanced data problem such as **Ensemble methods** like Balance Cascade that explores the majority class in a supervised manner and Easy Ensemble which learns different aspect of the original majority class in unsupervised manner. Also cost based approaches were developed.

A study to compare all those techniques was done using different data sets. They concluded that SMOTE and its combinations with Tomek Link and ENN (Nearest Neighbours in edited data)[26] provided the best results [27].

Chapter 5

Practical Case of Predictive Analytics Based On Alarms Triggered By The Onboard System

5.1 Problem Scope

In this case study, it was chosen data from a Nomad ROCM project with the Norges Statsbaner AS, known in English as the Norwegian State Railways and commonly abbreviated NSB. This is a government-owned railway company which operates most passenger train services in Norway. Their fleet has 69 trains class 74/75. The data set was downloaded from the vehicles 74-06, chosen "randomly" by a Nomad Engineer.

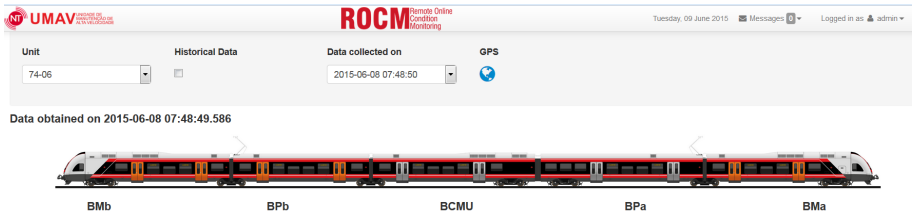


Figure 5.1: NSB 74-06 view in ROCM

That particular fleet was chosen because it has the most standard data. Therefore a case study with that data could be further expanded to other fleets. Also the relation between Nomad Tech and Norwegian State Railways is good and the request of additional information is possible. The particular vehicle was chosen randomly between the ones which were functional and the time line length was 1 year. The previous years are filed.

The data was downloaded from a mySQL database with specified parameters and in CSV (comma separated value) format. The query will not be exposed due to confidential information about Nomad data organization.

5.2 Available Data

The data set used has a time-span of 1 year, the first day is 23 of February of 2014 at 00:00. All the alarm events triggered since that time were registered and are in the file. The data has the following attributes:

- **Id:** It is a numeric attribute who identifies which code it is. Despite being numeric it has no value or scale
- **Veracity:** It is a nominal attribute, that can be a character between A and E, where the A represents the most dangerous and E is the least one.
- **Begin:** It is a Date attribute. Represents the time where the signal started
- **End:** It is a Date attribute. Represents the time where the signal turn off
- **System:** It represents the system where the signal belong, for instance: door system, brakes system, etc

Summarizing, each row of the data frame has an event defined by five characteristics (columns), id, veracity, begin time, end time and the system. There might happen different events at the exact same time even if the system is the same. Additionally, the same event can be triggered more than once each day or never ever happened in the register. Finally the total number of rows of the data frame was 197210.

5.2.1 Data Overview

The first analysis conducted with the data was to print a missing value map. 30% of the feature "End" were missing. The rest of the data was complete. For now the missing data is automatically completed by NA's. The two most common approaches for those cases in the pre-processing state of art is to remove the missing data or as an alternative, fill the missing data with statistical values such as median or mean. However, it depends always on the problem context and we are talking about times, data times so the second approach is invalid. The approach I suggest would be to fill the data with the start time inherently giving a length of 0 seconds to the event. It makes sense because it will not compromise future events, i.e., if we gave a length for instance of 10 seconds and during that 10 seconds the same event trigger it would be a nonsense. But this kind of nonsense actually happens in the data due to electronic or data capture failures that are uncontrollable by us and we do not want to increase it. Those cases are inherent to the system errors and it is out of the scope.

Then statistical information was retrieved like median, mean, statistic mode, maximum and minimum for each Id code, using the begin and the end time of each row to create a feature called "duration". With this analysis was possible to observe that most of the events had a minimum duration of 0.0 seconds and the mean was biased because there were cases where the duration reached huge values, we can call it outliers.

Finally there is the need to transform the available data in a classification problem.

5.3 Feature Engineering

Feature engineering is another topic which doesn't seem to merit any review papers or books, or even chapters in books, but it is absolutely vital to ML success. [...] Much of the success of machine learning is actually success in engineering features that a learner can understand. — Scott Locklin, in "Neglected machine learning ideas"

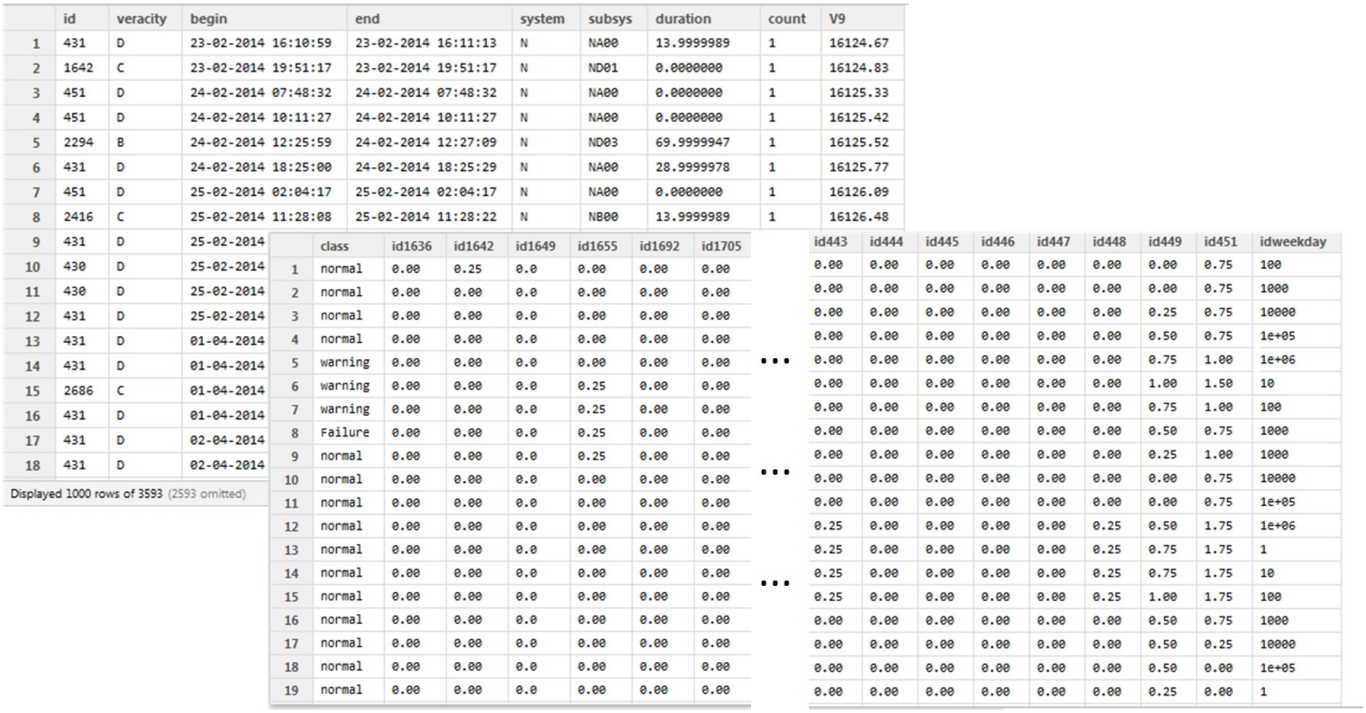
Feature engineering is an informal topic, but one that is absolutely known and agreed to be key to success in applied machine learning. It is about extracting the most of the data to get the best possible results from a predictive model.

The feature in the data influences directly the results achieved once it is the input of your algorithms, so it is important to transform it in the way that your machine learning algorithms can understand. Also giving a well engineered feature set you can get better results without using the optimal parameters of the model. Spend time and give emphasis to feature engineer actually pays off. The idea about this process is to design the features that best characterize the underlying problem and represents the raw data and so it will improve inherently the model accuracy.

Example of features:

- **Images:** colours, textures, contours, ...
- **Signals:** frequency, phase, samples, spectrum, ...
- **Time series:** ticks, trends, self-similarities, ...
- **Biomed:** dna sequence, genes, ...
- **Text:** words, POS tags, grammatical dependencies,

So what is feature engineering? It can mean many things to different problems but in this case it is about signals. Two different feature processes were done. Remembering that each row in the data set represents an event, the first process was to group the events (signals) by frequency in a column as shown in the left table of figure 5.2. In other words, the number of times a particular signal on a given day occured. Then we transpose the information in the frequency column to an horizontal matrix where the columns were the events and the lines the date. In the cells we have a value that represents the occurrence of an Id. It is illustrated in figure 5.2 the final matrix after the engineering done in the next two subsections 5.3.1 and 5.3.2.



	id	veracity	begin	end	system	subsys	duration	count	V9
1	431	D	23-02-2014 16:10:59	23-02-2014 16:11:13	N	NA00	13.9999989	1	16124.67
2	1642	C	23-02-2014 19:51:17	23-02-2014 19:51:17	N	ND01	0.0000000	1	16124.83
3	451	D	24-02-2014 07:48:32	24-02-2014 07:48:32	N	NA00	0.0000000	1	16125.33
4	451	D	24-02-2014 10:11:27	24-02-2014 10:11:27	N	NA00	0.0000000	1	16125.42
5	2294	B	24-02-2014 12:25:59	24-02-2014 12:27:09	N	ND03	69.9999947	1	16125.52
6	431	D	24-02-2014 18:25:00	24-02-2014 18:25:29	N	NA00	28.9999978	1	16125.77
7	451	D	25-02-2014 02:04:17	25-02-2014 02:04:17	N	NA00	0.0000000	1	16126.09
8	2416	C	25-02-2014 11:28:08	25-02-2014 11:28:22	N	NB00	13.9999989	1	16126.48
9	431	D	25-02-2014						
10	430	D	25-02-2014						
11	430	D	25-02-2014						
12	431	D	25-02-2014						
13	431	D	01-04-2014						
14	431	D	01-04-2014						
15	2686	C	01-04-2014						
16	431	D	01-04-2014						
17	431	D	02-04-2014						
18	431	D	02-04-2014						

Displayed 1000 rows of 3593 (2593 omitted)

	class	id1636	id1642	id1649	id1655	id1692	id1705
1	normal	0.00	0.25	0.0	0.00	0.00	0.00
2	normal	0.00	0.00	0.0	0.00	0.00	0.00
3	normal	0.00	0.00	0.0	0.00	0.00	0.00
4	normal	0.00	0.00	0.0	0.00	0.00	0.00
5	warning	0.00	0.00	0.0	0.00	0.00	0.00
6	warning	0.00	0.00	0.0	0.25	0.00	0.00
7	warning	0.00	0.00	0.0	0.25	0.00	0.00
8	Failure	0.00	0.00	0.0	0.25	0.00	0.00
9	normal	0.00	0.00	0.0	0.25	0.00	0.00
10	normal	0.00	0.00	0.0	0.00	0.00	0.00
11	normal	0.00	0.00	0.0	0.00	0.00	0.00
12	normal	0.00	0.00	0.0	0.00	0.00	0.00
13	normal	0.00	0.00	0.0	0.00	0.00	0.00
14	normal	0.00	0.00	0.0	0.00	0.00	0.00
15	normal	0.00	0.00	0.0	0.00	0.00	0.00
16	normal	0.00	0.00	0.0	0.00	0.00	0.00
17	normal	0.00	0.00	0.0	0.00	0.00	0.00
18	normal	0.00	0.00	0.0	0.00	0.00	0.00
19	normal	0.00	0.00	0.0	0.00	0.00	0.00

id443	id444	id445	id446	id447	id448	id449	id451	idweekday
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.75	100
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.75	1000
0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.75	10000
0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.75	1e+05
0.00	0.00	0.00	0.00	0.00	0.00	0.75	1.00	1e+06
0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.50	10
0.00	0.00	0.00	0.00	0.00	0.00	0.75	1.00	100
0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.75	1000
0.00	0.00	0.00	0.00	0.00	0.00	0.25	1.00	1000
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.75	10000
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.75	1e+05
0.25	0.00	0.00	0.00	0.00	0.25	0.50	1.75	1e+06
0.25	0.00	0.00	0.00	0.00	0.25	0.75	1.75	1
0.25	0.00	0.00	0.00	0.00	0.25	0.75	1.75	10
0.25	0.00	0.00	0.00	0.00	0.25	1.00	1.75	100
0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.75	1000
0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.25	10000
0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.00	1e+05
0.00	0.00	0.00	0.00	0.00	0.00	0.25	0.00	1

Figure 5.2: Feature Engineering

5.3.1 Track

A failure in the door systems is something binary, either it is fail or not. But between the state of normal behavior and the state of failure behaviour there are a degradation of the component 5.3 which can not be measure with this data. In order to have it in account we decide to create the features with a *synthetic degradation*, in other words, we created the features and each one spreed the path of degradation to the next ones, if the value of the feature is 0, nothing change, if the value is, for instance α, we will have α/N_i where i is the number of instance for where the *synthetic degradation* will spread. We believe it will be an important consideration to achieve good results.

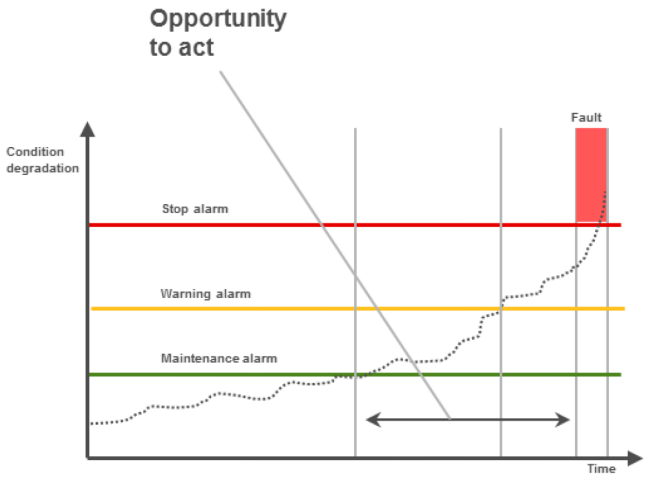


Figure 5.3: Condition Degradation

The process done to create this degradation path was to do the average of each feature of the pattern (row) with a window length of 4. So taking a pattern of a day, the attributes can have only integer values which represents the occurrence of that attribute in the day. To create the "track" we split the value of each attribute in equal parts to the features of the next 3 days as illustrated in the figure 5.4.

	begin	class	1636	1642	1649	1655	1692	1705
1	2014-02-23	normal	0	1	0	0	0	0
2	2014-02-24	normal	0	0	0	0	0	0
3	2014-02-25	normal	0	0	0	0	0	0
4	2014-04-01	normal	0	0	0	0	0	0
5	2014-04-02	normal	0	0	0	0	0	0
6	2014-04-03	normal	0	0	0	0	0	0
7	2014-04-04	normal	0	0	0	0	0	0
8	2014-04-05	warning	0	0	0	0	0	0
9	2014-04-07	warning	0	0	0	1	0	0
10	2014-04-08	warning	0	0	0	0	0	0
11	2014-04-09	Failure	0	0	0	0	0	0
12	2014-04-09	normal	0	0	0	0	0	0
13	2014-04-10	normal	0	0	0	0	0	0

	begin	class	1636	1642	1649	1655	1692	1705	1727	1728
1	2014-04-01	normal	0.00	0.25	0.0	0.00	0.00	0.00	0.00	0.00
2	2014-04-02	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00
3	2014-04-03	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00
4	2014-04-04	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00
5	2014-04-05	warning	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00
6	2014-04-07	warning	0.00	0.00	0.0	0.25	0.00	0.00	0.00	0.00
7	2014-04-08	warning	0.00	0.00	0.0	0.25	0.00	0.00	0.00	0.00
8	2014-04-09	Failure	0.00	0.00	0.0	0.25	0.00	0.00	0.00	0.00
9	2014-04-09	normal	0.00	0.00	0.0	0.25	0.00	0.00	0.00	0.00
10	2014-04-10	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00
11	2014-04-11	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00
12	2014-04-12	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00

Figure 5.4: "Track Phenomenon"

5.3.2 Dates

Many algorithms can not understand dates, therefore the second process was to get useful information in dates. Since trains perform services (from the starting station to the end station) with different frequencies per weekday, for instance the Mondays and the Fridays have more services than the other days, the weekly day was considered an important information to have in the business point of view. A dummy variable was created to represent the weekday, figure 5.2 shows the correspondent match.

Dummy Variable match	
1	Sunday
10	Monday
100	Tuesday
1000	Wednesday
10000	Thursday
100000	Friday
1000000	Saturday

Table 5.1: Weekday

Figure 5.5 is an histogram of this new feature, **weekday**. We can notice that, as expected, there are a slight variation in the number of weekdays. The reason to have no signal in some specific day could be as simple as the vehicle was stopped in maintenance or the occurrence of strikes for example.

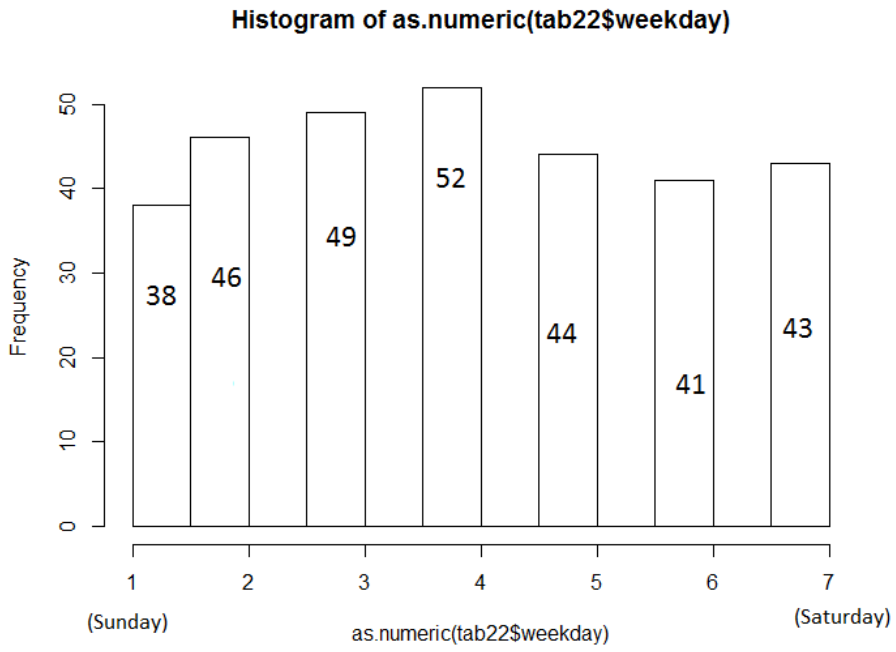


Figure 5.5: Weekday histogram

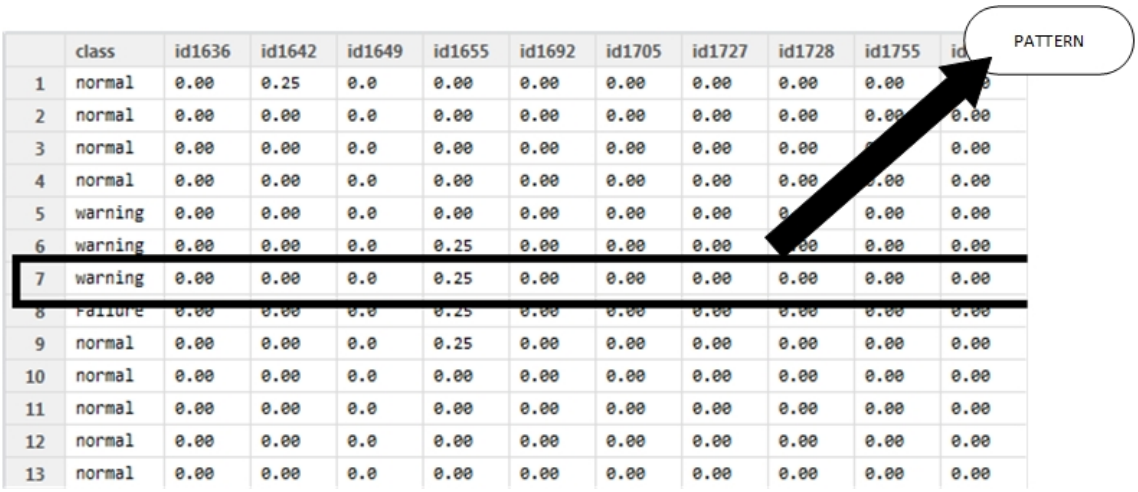
The month was considered irrelevant because the data set only had a year of size at most.

5.3.3 Labeling

"*To leverage predictive modeling for planning maintenance, you need data capturing the information whether there has been need for maintenance activities in a particular situation. Typically, this is equivalent to the fact that the equipment failed at some point in time. This information has to be complemented by data describing the state of your equipment at that point of time.*" By RapidMiner Studio

The present data contains rows that we call patterns. A pattern, figure 5.6 is a day of that vehicle. A day where no Id occured is a vector of zeros and if a event happens X times then the correspondent cell will have that value.

Due to the lack of information to allow us to label that data as being a real failure or not in Nomad sources, NSB specialist gave us a list of events id's identified by them which cause a real failure. The days where those events happen were classified as **Failure**, the rest of the days were classified as **Normal**.



	class	id1636	id1642	id1649	id1655	id1692	id1705	id1727	id1728	id1755	id
1	normal	0.00	0.25	0.0	0.00	0.00	0.00	0.00	0.00	0.00	
2	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00		0.00
4	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	.00	0.00
5	warning	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0	0.00	0.00
6	warning	0.00	0.00	0.0	0.25	0.00	0.00	0.00	00	0.00	0.00
7	warning	0.00	0.00	0.0	0.25	0.00	0.00	0.00	0.00	0.00	0.00
8	Failure	0.00	0.00	0.0	0.25	0.00	0.00	0.00	0.00	0.00	0.00
9	normal	0.00	0.00	0.0	0.25	0.00	0.00	0.00	0.00	0.00	0.00
10	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
11	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00
13	normal	0.00	0.00	0.0	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Figure 5.6: Pattern

However, identifying a failure in the day that happened is redundant and nonprofitable, in other words, it is too late. The main goal is to identify it before it happens giving opportunity to do a corrective intervention. A threshold of 3 days was defined by me and approved by a specialist to be the time span that allows to correct the possible failure, those 3 days were re-label as **warning**. As example, we can look to figure 5.2 in the second table. The first column is the outcome column, what we want to predict. As described, the outcome has three classes, normal, warning and failure, and we can also observe the example in the row number 8 where it was labeled as failure and as consequence the 3 rows before were labeled as warning.

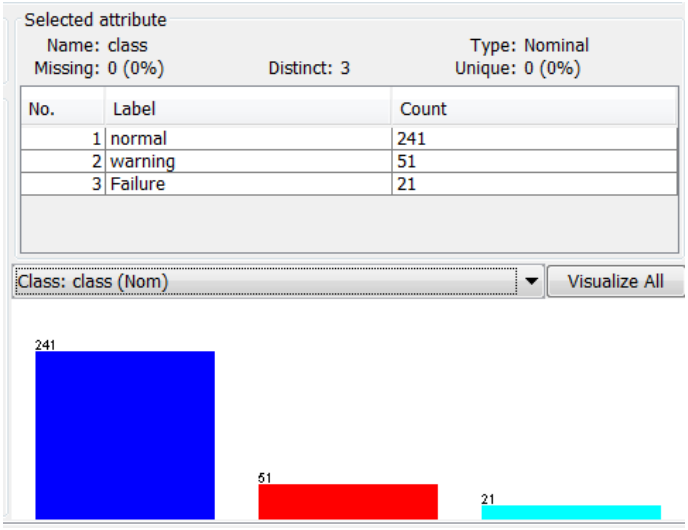


Figure 5.7: Class Distribution

After the labeling process the distribution of the class's per patterns is represented in figure 5.7. The normal class is the most common class as expected in this type of problems. Failure class is the minority class and the warning class is the intermediate class but it is much less frequent than the majority class. The warning class is the goal because it is the "clue to the failure". The normal class has also an important role since misclassfying those prediction could lead to unnecessary maintenance interventions, in other words, extra costs antithetical to the intended purpose.

5.3.4 Predictors Quality

Predictors are the model ingredients and so it is important to understand them and their viability. Due to feature engineering, it is possible to generate predictors that only have a single unique value, "zero-variance predictor". It can lead the model to crash or the fit can be unstable.

In addition, predictors can only have a few unique values which can occur with very low frequencies as illustrated in table 5.2 (four random predictors as example) or in the histograms 5.9 and 5.10 where it it possible to observe the few unique numeric values that are highly unbalanced.

Predictors		Frequency	0 Value Freq (%)
Error Id's	Values		
id445	0	289	92.3
	0.25	16	
	0.5	4	
	0.75	4	
id444	0	309	98.7
	0.25	4	
id1728	0	285	91.1
	0.25	28	
id3074	0	307	98.1
	0.25	2	
	0.75	2	
	1	2	
id2628	0	287	91.7
	0.25	22	
	0.75	4	

Table 5.2: Predictors Frequency Examples

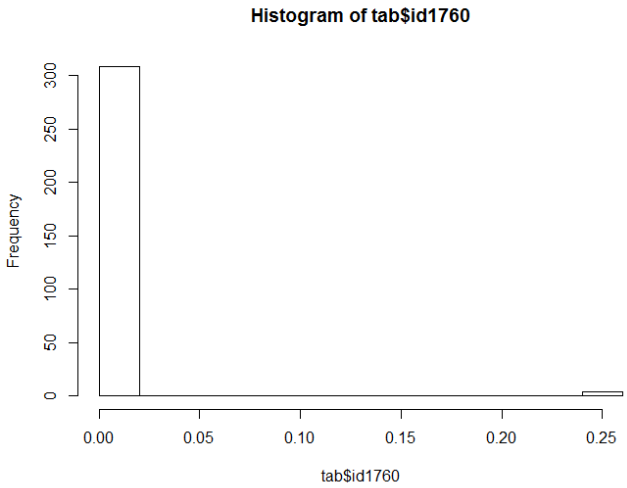


Figure 5.9: Histogram Id 1760

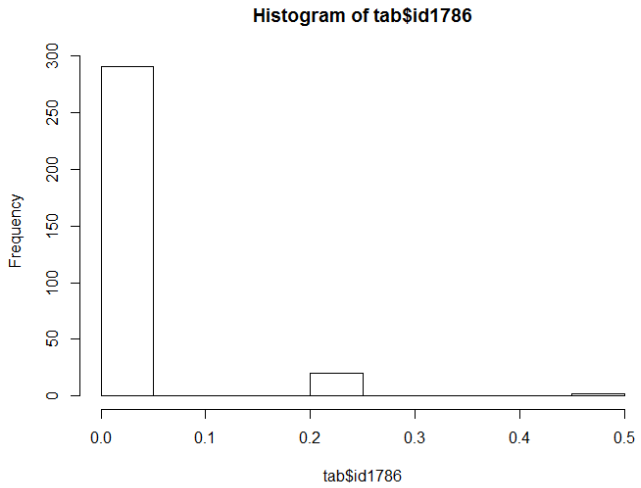


Figure 5.10: Histogram Id 1786

As we can observe, all the predictors except the "weekday" have a major class with a much higher frequency and indeed it is possible to happen problems such as, use a predictor with only the value zero in the training set, the different algorithms will react in different ways except decision trees where it is not a problem due to their inherent feature selection.

In the*caret Package* there is a function, **nzv** which decides either to attribute the "near zero-variance predictor" to a variable by the position according to the threshold defined. The function returned TRUE to all the predictors, which means they are all "bad predictors". However the function was updated and the threshold method reformulated and now the result is different, some of the predictors are assumed in fact non problematic. A future work could be to analyse this output. The consideration will remain the first one in this thesis, without despising this second result, the objective is to predict components failures and it includes "coincidences" and not only the most frequent failures. Likewise we can not guarantee that predictors distribution is random, and so the test that for a predictor is true for another time span might be false. Or for one predictor the test is true and for other one be false.

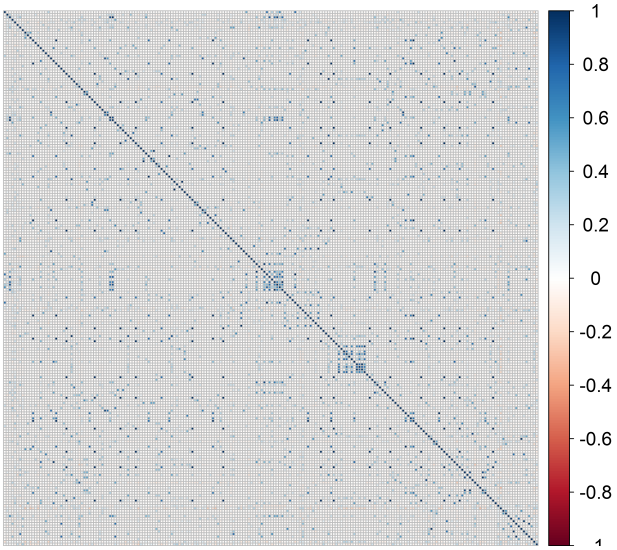


Figure 5.11: Correlation Matrix

Another analysis done was the predictors correlation. We used the package caret to find the matrix correlation 5.11 of all the predictors, it is a pairs comparison. The matrix has values between -1 and +1. Values near 0 means that the predictors have a weak or null correlation, and the higher values in module means the opposite. However the elimination of the high correlated predictors have not lead to results improvement. The cutoff tried was 0.9.

5.4 Data Visualization

"Data visualization is the presentation of data in a pictorial or graphical format. For centuries, people have depended on visual representations such as charts and maps to understand information more easily and quickly." [28]

"Data visualization is an art and a science unto itself..." [28]

As more and more data is collected and analyzed, decision makers at all levels welcome data visualization software that enables them to see analytical results presented visually, find relevance among the millions of variables, communicate concepts and hypotheses to others, and even predict the future. Because of the way the human brain processes information, it is faster for people to grasp the meaning of many data points when they are displayed in charts and graphs rather than poring over piles of spreadsheets or reading pages and pages of reports.

As said in section 5.3.3, NSB provide us with a report file with the fail Id's. Also, They provide with a list of possible Id's that could be related or lead to those failure but there nothing to prove, no technical report. A scatter plot was done to understand the data of 4 different doors. In the X-axis is the date with a length of one year and in the Y-axis is the Id's.

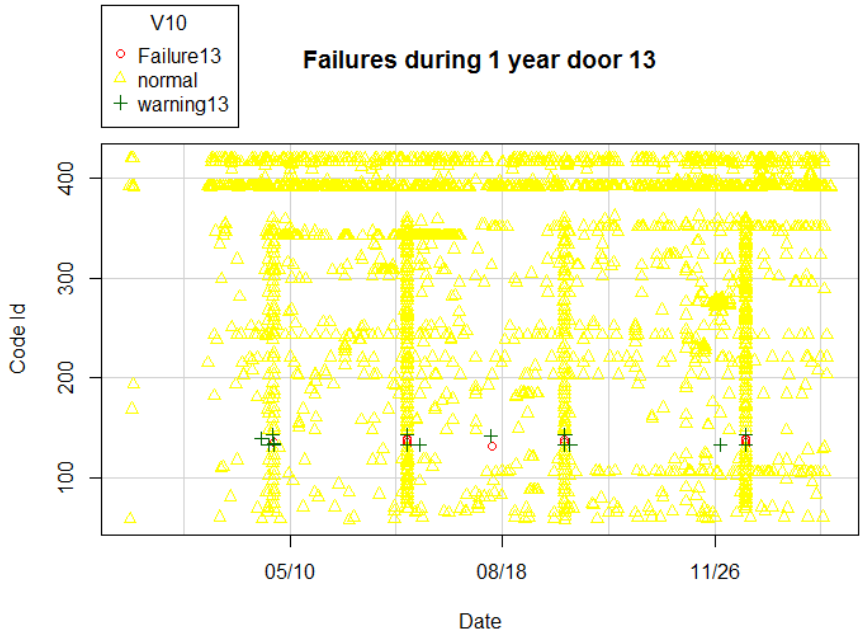


Figure 5.12: Door 13

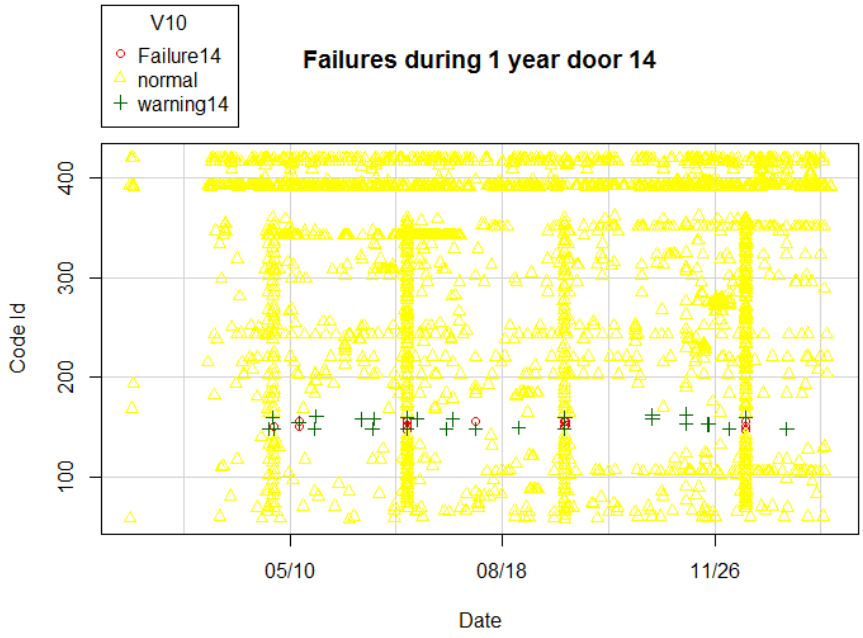


Figure 5.13: Door 14

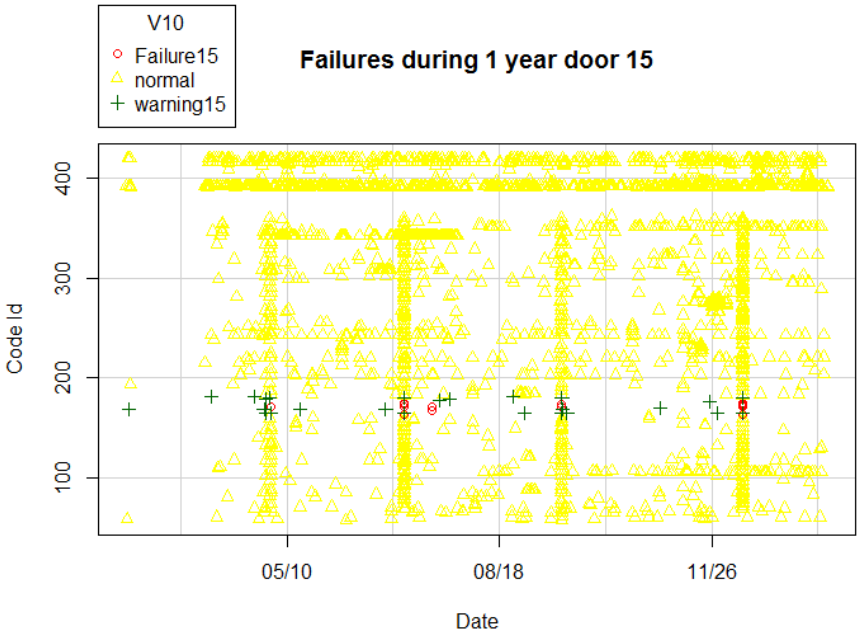


Figure 5.14: Door 15

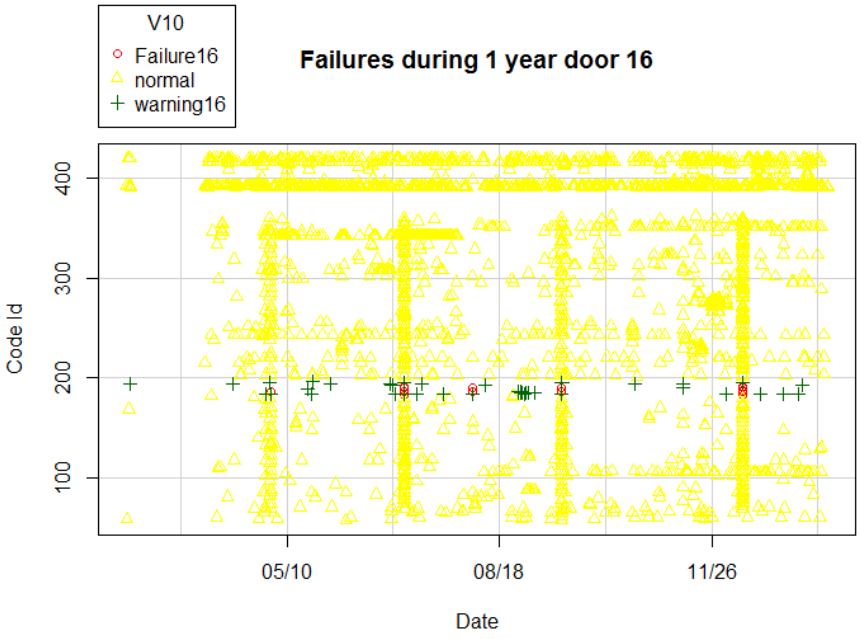


Figure 5.15: Door 16

Analysing those plots 5.12, 5.13, 5.14 and 5.15 we can notice four vertical patterns and in the top an horizontal pattern and very dense area. The dense area correspond to 2 or 3 signals that

appear with a very high frequency and looking almost continuous. The vertical patterns correspond to days with huge amount of different events.

To understand it, a specialist opinion was asked, Id's data-sheet was consulted and GPS (Global Positioning System) coordinates to contextualize it. It lead us to realize that the horizon patter was a very frequent and common alarm but innocuous. While the four vertical patterns suggested by the specialist and confirmed with the GPS were maintenance days. The vertical pattern imply a fail of all the door systems at the same time, an abnormal case if the train was on service. So the GPS confirmed that on those days the train was indeed in the same place, the workshop. Due to this analyse, the days considered of maintenance were excluded to not lead models to false or useless predictions.

5.5 Experimental Setup

In this section It will be explained in global point of view the experimental setup. The objective is to make decision and normalize the procedure in order to preserve two key factors, the experiences reproducibility and reliability. The first factor guarantee that anyone can reproduce the experience any time and will have the same results, the other assure that the results are meaningful by defining baselines and precise procedures to not have biased results. The figure 5.16 is a general overview containing the key point that will be explained along the next sections and subsetions.

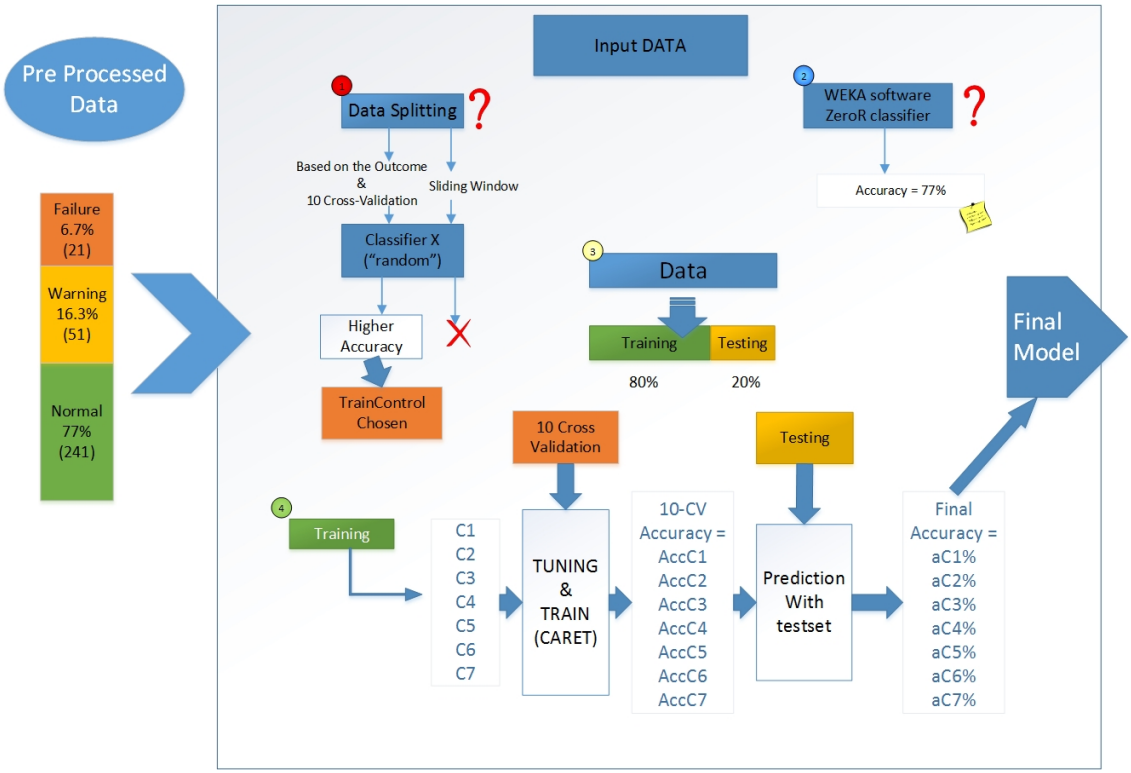


Figure 5.16: Experimental Setup Overview

Since the project scope is to have a solution capable to predict a failure and not to compare algorithms or suggest a new one we will avoid some of the possible or expected criticisms and clarify how we surround it such as:

- *The criterion used to select the data set may bias the results.* The data set was previously picked by a person or team in the company excluding myself from any decision.
- *whether the selection of learners is representative enough and whether the selected learners are properly configured to work at their best performance*. To avoid the lack of attention to properly configure a algorithm in favour of other(s) we delegate that task to the **Tune** function of the R package Caret [29]. It used pre-defined and supposedly meaningful values to find the best available configuration over a training set.
- It *is still impossible to determine the maximum attainable accuracy for a data set, so that it is difficult to evaluate the true quality of each classifier.* It is impossible to validate this hypothesis and it is out of the scope. So due to the time available to the project and resource we will use a small set of classifiers and from different families to try to represent a few fields like machine leaning, statistic,... and have a good diversity. It is also reasonable to assume that when the number of classifiers increases, some of them will achieve the largest possible accuracy.
- *The lack of standard data partitioning, defining training and testing data for cross validation trials. Simply the use of different data partitionings will eventually bias the results, and make the comparison between experiments impossible* [30]. In this project the data set uses the same partitioning for all the classifiers for the same iteration in order to compare the results. So it will not bias the results favouring any classifier.

5.6 Data Split

The method chosen to split the data in training set and test set was *random splitting based on the outcome*. As the figure 5.7 illustrate in the previous section the data is unbalanced so we considered this method the most appropriate to the data. Also in the overview picture 5.16, on the left we can see some numbers, the class Normal has 241 cases, the Warning has 51 occurrences and Failure 21 totalizing 313 samples. Usually it is used 70% or 80% of the data to build the training set and the rest to the test set. We used 80%. And as said before both of the subset have a class percentage equal to the original set, in other words, the normal class has a 77% of occurrence in the original set and it is also kept that value in both, the training and the testing sets, same for the other classes. However we do not guarantee the same distribution to all the predictors in the subsets created.

The function **createdatapartion** in the package Caret was used since it does it automatically. Five different partitions were created (without replacement), the first four were aggregated as the training set and the last one was the test set.

The sliding window is also a procedure that could bring some improvements since it takes in account time series. Caret package also provide the tools to create a sliding window. However we decided to use 10 fold cross validation instead of windowing in the tuning process to model validation. A combination of different values were tried with windowing parameters in a pre experience to chose the evaluation method although windowing did no achieve better results then cross validation.

5.7 Model Tuning

"Each problem is unique, you are very likely have not seen it before and you cannot know what algorithms to use, what data attributes will be useful or even whether the problem can be effectively modelled." by Jason Brownlee

5.7.1 Baseline

"If other algorithms do not give better accuracy than the baseline, what lesson should we take from it? Does it indicate that the data set does not have prediction capability?"

To any problem it necessary a useful point for comparison to understand whether a model or a change is adding value. Data mining is an experimental science and so after you define your baseline, you can add or change the data attributes, the algorithms you are trying or the parameters of the algorithms and you will know if you have improved you approach or not and also how much you improved.

A baseline is the simplest possible prediction. If it is a classification problem, usually the majority class is selected and used as result for all predictions. We used the algorithm ZeroR available in Weka to calculate our problem baseline. All the data set was used as input and as expected we had a 77% of accuracy as baseline that correspond to the percentage of samples that are labeled as **normal**.

5.7.2 Algorithms

As mentioned in *Experimental Setup section*, we chose one algorithm from each most known families from different areas. Some algorithms picked at the begin such as Naive Bayes produced a baseline accuracy or worst, they were discarded or changed by others from the same family. Understand the reason and try to make it work is out of the scope since we have plenty of alternatives and it would consume precious project time. The list of final algorithms we have used is:

1. **C5.0** from the decision trees family, package C50

2. **rf**, Random Forest an ensemble algorithm, package randomForest

3. **NNET**, a neuronal network implementation

4. **PDA**, penalized discriminant analysis from the LDA family

5. **svmPoly**, from the support vector machine family, polynomial kernel

6. **LogitBoost**, Boosted Logistic Regression from caTools package

7. **gbm**, Stochastic Gradient Boosting from gbm package

5.7.3 Tuning parameters

Machine learning algorithms are parameterized so that they can be best adapted for a given problem. Like selecting the best algorithm for a problem you can not know before hand which algorithm parameters will be best for a problem. So it is necessary to investigate empirically with controlled experiments.

In order to get the parameters for each algorithm that produces the best available result we used the R package Caret which provide an interface to handle this problem with the function **train**. It provides a grid search method for searching parameters combined with various methods for estimating the performance of a given model. An example is provided in figure 5.18. It identify that there are 3 classes to predict, the number of samples provided in the data set. As output is possible to observe all the combinations done for that classifier and which was the final model chosen. We can verify that the model with the higher accuracy was indeed chosen correctly by the function train and the parameters combination that maximize the accuracy are trials = 10, model = tree and winnow = FALSE. The model achieves an accuracy of 0.7900641 in the 10 cross validation. It is also possible to plot the details to examine the relationship between the estimates of performance and the tuning parameters 5.17.

```
C5.0

251 samples
209 predictors
  3 classes: 'Failure', 'normal', 'warning'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 1 times)

Summary of sample sizes: 226, 226, 225, 225, 226, 226, ...

Resampling results across tuning parameters:

  model  winnow  trials  Accuracy   Kappa      Accuracy SD  Kappa SD
  rules  FALSE    1      0.7895897  0.2002751  0.04229145   0.2130896
  rules  FALSE   10      0.7859103  0.2766582  0.05206898   0.2064226
  rules  FALSE   20      0.7897564  0.3018069  0.05831667   0.2020562
  rules   TRUE    1      0.7732564  0.1870837  0.03920270   0.1692978
  rules   TRUE   10      0.7815897  0.1843796  0.04736555   0.2038533
  rules   TRUE   20      0.7695897  0.1571698  0.05956602   0.2255846
  tree   FALSE    1      0.7897564  0.2203116  0.05345573   0.2411218
  tree   FALSE   10      0.7900641  0.3625533  0.06314548   0.1803603
  tree   FALSE   20      0.7860769  0.3669654  0.06962870   0.1935491
  tree    TRUE    1      0.7692564  0.2014007  0.04658585   0.1586818
  tree    TRUE   10      0.7814359  0.1919606  0.05169032   0.2193838
  tree    TRUE   20      0.7694359  0.1818569  0.06014022   0.2167114

Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were trials = 10, model = tree and winnow = FALSE.
```

Figure 5.17: Function Train

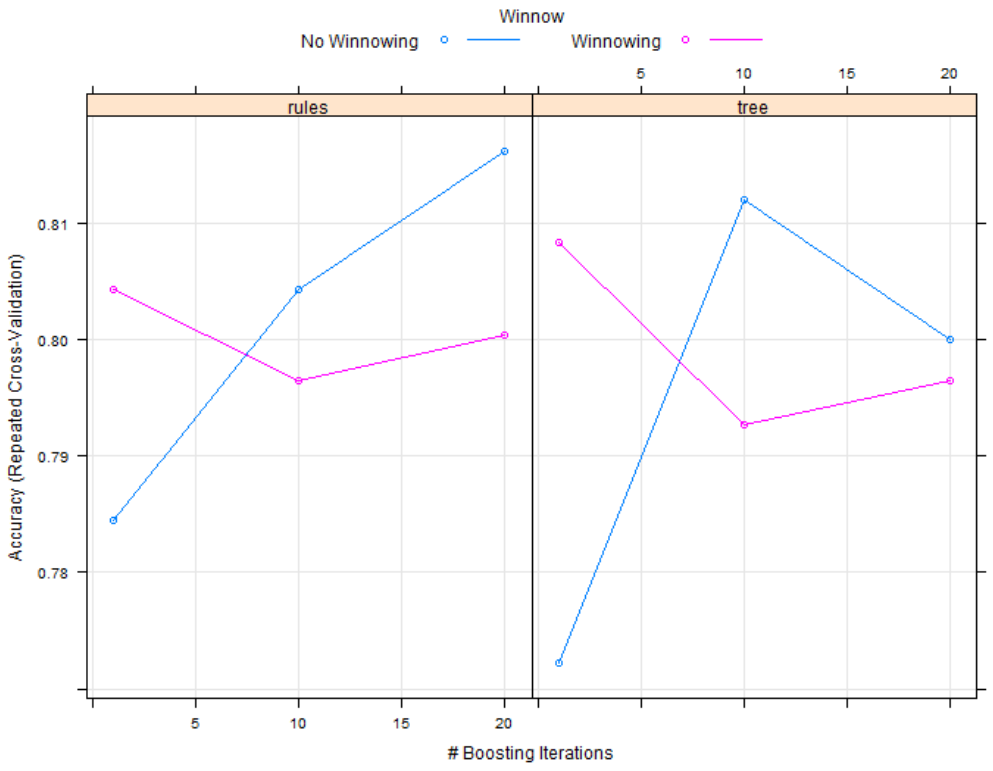


Figure 5.18: Parameters

This function develops the parameter tuning, selecting the values which maximize the accuracy according to the validation selected (timeslices, k-fold, etc.). The caret package also allows to define the number of values used for each tunable parameter, although the specific values can not be selected. We let the function parametrize all automatically. We only specified the 10 fold cross validation as the validation method. This function only allow to model tuning multiclass problems maximizing accuracy or kappa. Only have official implementation in dual class problems to maximize other evaluation parameters as recall, precision or ROC curve.

The train function allows as well to personalize the model tuning in two different ways. On one hand we can just increase the grid of option and it does all automatically or on the other hand we can provide values that the model will test performing all the possible combinations.

Tuned Parameters		
Algorithm	Final Parameters	Accuracy & Kappa
C5.0	trials = 10 model = tree winnow = FALSE	0.7900641 0.3625533
Random Forest	mtry = 214	0.7782179 0.3624744
Neuronal Network	size = 1 decay = 0.1	0.8172949 0.4417007
PDA	lambda = 2	0.7775897 0.2338405
svmPoly	sigma = 0.003592967 C = 0.5	0.7972692 0.287496
LogitBoost	nIter = 31	0.8261901 0.4047760
GBM	n.trees = 150 interaction.depth = 3 shrinkage = 0.1	0.7822308 0.3399710

Table 5.4: Tuned Parameters

The table 5.4 summarizes the final model of all algorithms with the function **train** and using a seed with the value 3. The algorithm which achieves the best accuracy is *Boosted Logistic Regression*, in the second place follows random forest. If you are working on a classification

problem, you may want to look at the Kappa statistic, which gives you an accuracy score that is normalized by the baseline. The baseline accuracy is 0 and scores above zero show an improvement over the baseline. Using the predefined parameters of the function train to tune was good enough since trying more combination would increase the computation time and if the results improve due to that it would not be significant enough.

5.8 Model Evaluation

Contrary to a common misconception, there is nothing wrong in using the whole available labeled dataset as the training set for building a model. What is deeply wrong is using some part of this dataset to evaluate the same model. So one can train and use all-data as long as one does not evaluate the model does obtained. This is, of course, totally unacceptable for research, which is all about evaluating, comparing, and benchmarking, but for practical applications it is also completely unimaginable to accept any model without reliable performance estimates. The solution to this dilemma may be to evaluate one or more models built on a smaller training subset using a separate validation or test set, and use the obtained indicators as performance estimates for another model, built on the whole dataset, using exactly the same modeling procedure (i.e., the classification algorithm, its parameter settings, and all other details that impact the produced model being unchanged). The right way to look at the evaluation procedures is therefore often as methods of evaluating a modeling procedure rather than the actual model delivered for the application, where the term “modeling procedure” encompasses the classification algorithm and everything else other than the dataset that affects the generated model. When evaluating a modeling procedure, an appropriate evaluation procedure is needed to keep training and validation or test sets separate. This evaluation procedure can be applied to calculate one or more performance indicators that will serve as performance estimators for the model built on the complete dataset using exactly the same modeling procedure. The final model not only can, but also should be built using as much data as is available (and can be handled within the existing computational constraints, if any), to maximize performance on new data.

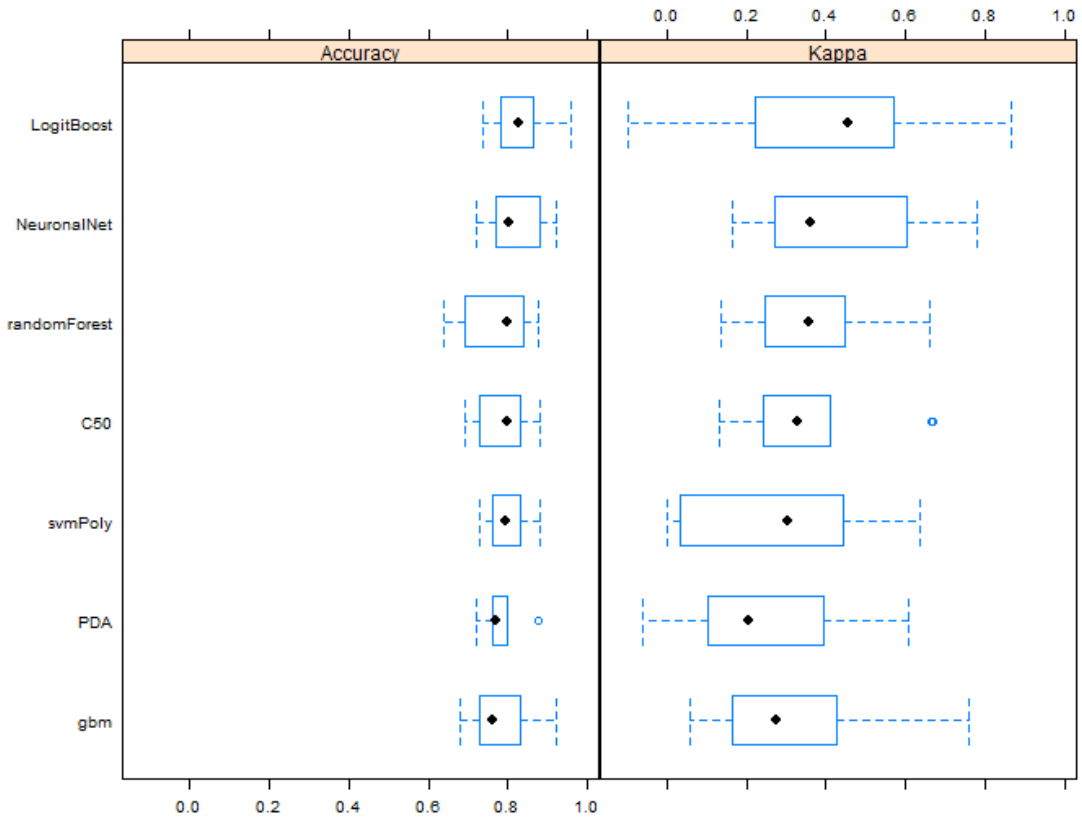


Figure 5.19: Evaluation Boxplot

With the algorithms and the respective final model parameters we can observe a boxplot illustrated in the figure 5.19. Due to the 10 cross validation done to evaluate the model was possible to present some statistical values referent to accuracy and kappa. Models with a large variation between the maximum and the 3rd quartile or the minimum and the first quartile are considered untrustable.

5.9 Results

In the experimental work we tested seven different classifiers with the optimal parameters given in table 5.4. The test was performed in the test set hold out before, and the results kept were the accuracy, kappa, precision and recall. The authors of the package caret use a one-against-all approach to output evaluation measures and they call to recall Sensitivity and to precision positive prediction value (POS Pred Value). The warning class is our target class and since the approach is OAA we would get the two measures, recall and precision, to all the classes. We decided to only extract those measures for the warning class because as a target class is where we have the important information and the measures to quantify whether we improved or not and whether we satisfyed the problem requirements. For instance, for the majority class those measures are

"meaningless" because they have always high scores and the variations are almost imperceptible. While the target class has less cases, one miss classification cause a big variation in the scores changing drastically the model precision or recall because that difference worth money.

5.9.1 Final Models

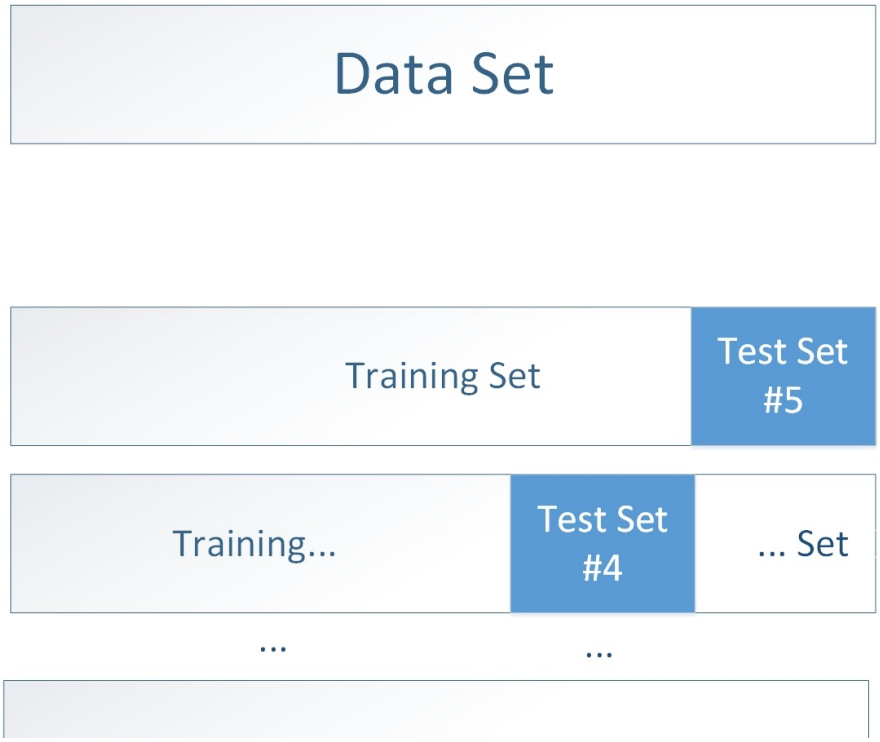


Figure 5.20: Iteration Process

In order to do not benefit one model due to "luck" we run the procedure 5 times. Namely, one model can perform better than the other in a test set and in the next *n* test sets perform worst so to no benefit an occasional good perform once we decided to apply all experimental work five times, each time with a different test set with unreplicated blocked data. We called it iteration process, being iteration 1 to all the procedure (which consists in use the train set to tune, train, and cross validation) for the test set 1, iteration 2 for the second test set... The figure 5.20 illustrate the unreplicated bocked data partition. It is also important to understand that the split procedure was the same explained before. We only guarantee that the data set is unreplicated but it is not done in a temporal order, it is totally random. So the iteration process correspond to a repetition of the points 3 and 4 in figure 5.16.

In the results table 5.5 is possible to observe that high scores are achieved but by different algorithms in each iteration. The models which achieve the best accuracy also achieve the best kappa since the kappa is a measure that quantify the accuracy gain in the test sample having in

account the baseline accuracy of the test set. NNET model appears to be the most constant always with good scores.

Recall is a very important measure in behalf of predict the system breakdown. The measure represents the trade off between all the predictions of warning class which are actually warning and warnings that are classified as other class. Idealistically if the value is 1 we would predict all the warnings so if this value is low less cases we will predict which is the scope of the problem. Cases such as the algorithm C5.0 in iteration 4 are unacceptable because it is to guess wrong all real failures, there are no gain.

Precision is considered also an important measure since it captures the wrong warning class prediction that can lead to unnecessary maintenance interventions. Other classes cases classified wrongly as warning lead to a decrease on the precision value. In order avoid it necessary to left out algorithms which have very low performances on this measure and recall measure

Once again we can infer that NNET has a great problem interpretation since it has as recall lower value 0.6 and 0.5 as precision lower value. It is the unique algorithm able to performer above the 50 % on those two measures. C5.0 has the best score in precision twice however has also the worst performance one time. GBM accomplish two times the best precision score and the highest score in the experiments and C5.0 the worst score.

5.9.2 Statistical test

We did the non-parametric Friedman rank test [31] to investigate if the there are statistical differences between the final models on the recall and precision measure. It ranks the algorithms for each data set (iteration) separately, the best performing algorithm getting the rank of 1, the second best rank 2... In case of ties, average ranks are assigned. The null hypothesis states that there are no significant difference when $p < \alpha$, and usually using $\alpha = 0.05$. If the test is reject it one or more classifier is significantly better than another. Also the test was run taking in account the recall measure and accuracy.

We used the Friedman rank from package Stats and test in order to do this statistical comparision. And the package PMCMR has the function to do the post hoc Friedman test. It is necessary to guarantee that the data is unreplicated and take into account that "comparisons using a single data set are pestered by the biased variance estimations due to dependencies between the samples of examples drawn from the data set..." [32].

There are three families of statistical test but we used non-parametric ones since parametric tests do not assume normal distributions or homogeneity of variance. Also this tests are refereed in the article [32] which they conclude to be stronger than others tests studied.

After applied the Friedman rank test we got a $p > 0.05$ and so we do not reject the null hypotheses. If the null-hypothesis is rejected, we can proceed with a post-hoc test. The Nemenyi test [33]. Table 5.6 has the Friedman rank for the experiments.

	Algorithm	Iteration1	Iteration2	Iteration3	Iteration4	Iteration5
Accuracy	C5.0	0.7937	0.7742	0.871	0.7619	0.871
	rf	0.8095	0.7419	0.8548	0.7937	0.8548
	NNET	**0.8413**	**0.8710**	0.8387	0.7937	0.8387
	PDA	0.7937	0.8065	0.8226	**0.8254**	0.8226
	svmPoly	0.7937	0.8226	0.8387	0.7937	0.8387
	LogitBoost	0.807	0.8491	0.8889	0.7966	0.8889
	GBM	0.7778	0.7419	**0.9032**	0.7143	**0.9032**
Kappa statistic	C5.0	0.3276	0.3589	0.6248	0.1	0.6248
	rf	0.4149	0.3525	0.5592	0.3018	0.5592
	NNET	**0.5464**	**0.6026**	0.5317	**0.3933**	0.5317
	PDA	0.34	0.4206	0.4141	0.3324	0.4141
	svmPoly	0.2507	0.5129	0.471	0.2909	0.471
	LogitBoost	0.3549	0.4316	0.6157	0.334	0.6157
	GBM	0.3776	0.1047	**0.7343**	0.2125	**0.7343**
Sensitivity (Recall)	C5.0	0.18182	0.30000	0.40000	0.00000	0.40000
	rf	0.27273	0.50000	0.50000	0.20000	0.50000
	NNET	**0.6364**	**0.7000**	0.7000	**0.60000**	0.7000
	PDA	0.36364	0.30000	0.10000	0.30000	0.10000
	svmPoly	0.18182	**0.7000**	0.40000	0.30000	0.40000
	LogitBoost	0.40000	0.60000	0.57143	0.37500	0.57143
	GBM	0.27273	0.10000	**0.9000**	0.40000	**0.9000**
POS pred value (Precision)	C5.0	0.40000	0.42857	**0.80000**	0.00000	**0.80000**
	rf	0.42857	0.35714	0.71429	0.50000	0.71429
	NNET	**0.5833**	0.7000	0.5833	0.50000	0.5833
	PDA	0.50000	**0.75000**	0.50000	**0.75000**	0.50000
	svmPoly	0.50000	0.5000	0.50000	0.42857	0.50000
	LogitBoost	0.57143	0.42857	0.66667	0.37500	0.66667
	GBM	0.37500	0.25000	0.6429	0.33333	0.6429

Table 5.5: Results

	C5.0	rf	NNET	PDA	svmPoly	LogitBoost	GBM
Iteration 1	6.5	4.5	1	3	6.5	2	4.5
Iteration 2	5.5	4	1.5	5.5	1.5	3	7
Iteration 3	5.5	4	3	7	5.5	2	1
Iteration 4	7	6	1	4.5	4.5	3	2
Iteration 5	5.5	4	3	7	5.5	2	1
Friedman Rank	**7** (6)	**4** (4.5)	**1** (1.9)	**6** (5.4)	**5** (4.7)	**2** (2.4)	**3** (3.1)

Table 5.6: Friedman Rank using Accuracy

Chapter 6

Model Improvements

6.1 SMOTE

Unbalanced data sets are a problem which can affect many models. In oder to deal with this problem there are some state-of-art procedures refereed in section 4.5. Unfortunately our data set in unbalanced having three classes, normal class with 241 cases (77%), warning class with 51 cases (16.3%) and finally failure class with 21 of the total cases (6.7%) and it is also unbalanced in relation to the predictors as showed in section 5.3.4.

In order to overcome the unbalance class problem we tried the SMOTE approach with the same test set used for the iteration 1. SMOTE is a synthetic over sampling which we applied in the training set to enhance and emphasize the rules created by our models. In other words, while the models have only a few cases of the minorities class it can be difficult to the algorithm to understand and create proper rules to predict unseen data. Increasing the number of cases it can lead to a improve on the results.

To do it we used the software WEKA where we uploaded the training data from R as illustrated in figure 6.1. We chose the SMOTE technique and applied it to our two minority classes, warning and failure. The over sampling parameter was 200 and the nearest neighbours parameter was 5 also the random seed was set as 1. This process was done two times, the first was applied directly to the warning class and then to the failure class. As consequence the SMOTE remove the cases that he consider to be on the boundary of the classes, in other words, the examples which are too much similar and from different classes.

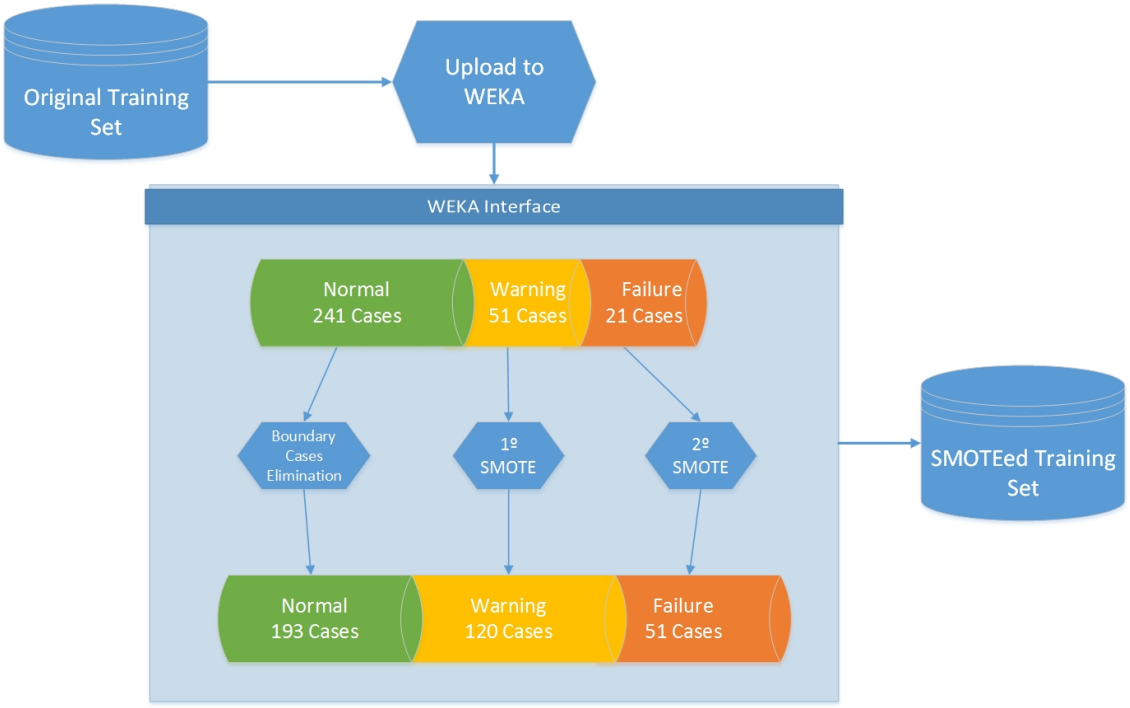


Figure 6.1: SMOTE_process

After "SMOTEing" the data we got 364 instances, 193 normal cases, 120 warning cases and 51 failure cases. The target class is warning however we considered important to also over sample the class failure since it represent a uninformative class but important to the model do not misclassify with the other two. Giving this new set as the new training set for the same test set we did all the iteration (train, tune and cv) and tested. The results are in table 6.1 and it can be compared with the results of the iteration 1 from the table 5.5.

Metrics	Models						
	C5.0	rf	NNET	PDA	svmPoly	LogitBoost	GBM
Accuracy	0.8095	0.8413	0.8254	0.746	0.8571	0.7931	0.8571
Kappa	0.4466	0.5510	0.5772	0.4217	0.6043	0.4276	0.5756
Recall	0.36364	0.54545	0.6364	0.6364	0.6364	0.5000	0.4545
Precision	0.44444	0.60000	0.5000	0.3889	0.5833	0.4166	0.7142

Table 6.1: SMOTE with test set 1

6.2 Ensemble techniques

Stacking, Blending and Stacked Generalization are all a category of ensemble learning with different names. In traditional ensemble learning, we have multiple classifiers trying to fit to a training set to approximate the target function. Since each classifier will have its own output, we will need to find a combining mechanism to combine the results. This can be through voting (majority wins), weighted voting (some classifier has more authority than the others), averaging the results, etc. This is the traditional way of ensemble learning [10].

In stacking [34], the combining mechanism is that the output of the classifiers (Base model $M_1 ... M_x$) will be used as training data for another classifier (Meta Learner) to approximate the same target function. Basically, the idea is to let the meta classifier to figure out the combining mechanism as shown in the figure 6.2.

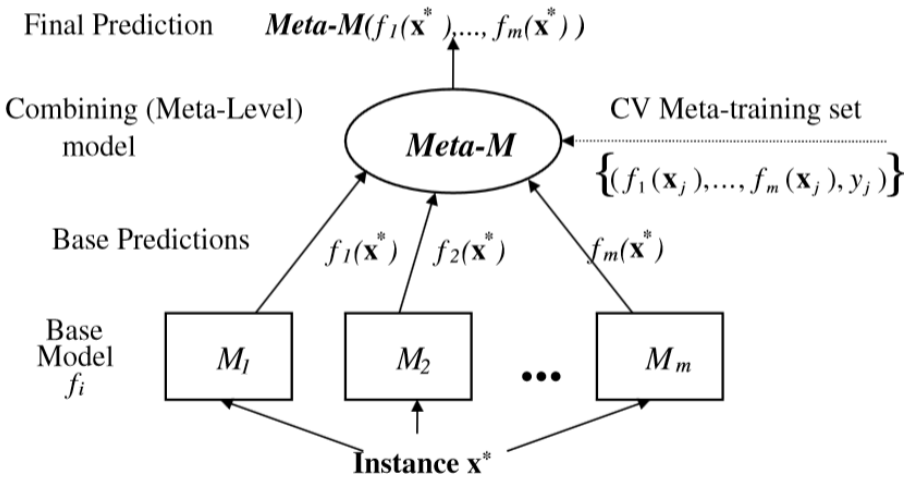


Figure 6.2: Stacking

6.2.1 Homogeneous and Heterogeneous

"Two heads are better than one."

Stacking is very powerful to eliminate the noise of the data set. Each classifier interprets data in each own way and so there are models that are sensible to a type of data and other to another type. When we chose the same algorithm to be the base classifiers, for instance the models M1, M2,... from the image 6.2 to be a SVM models we are talking about homogeneous ensemble. On the other hand if the base classifiers are different we call it heterogeneous ensemble, figure 6.3.

To test if we could achieve a model improvement we tried both methods, the homogeneous was conducted using four Random Forest as baseline classifiers and meta learner. The random forest meta learner was kept as meta learner in the heterogeneous experience and as base line was used the C5.0, random forest, nnet and GBM algorithm (same algorithms from the previous chapter subsection 5.7.2). The idea of homogeneous is to mainly to enhance the rules that we could get with a simple random forest model and therefor we expect a slight improvement from here. In the heterogeneous we expect a significant improvement in the model evaluation metrics essentially in recall and precision due to different baseline algorithms.

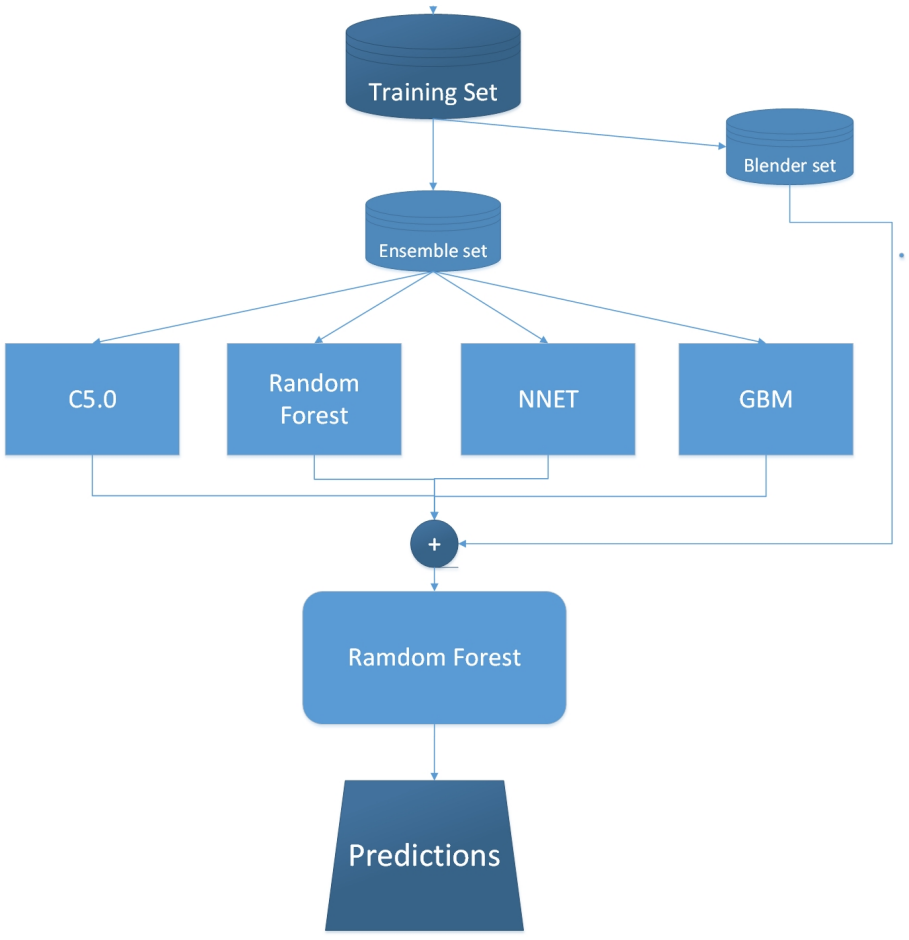


Figure 6.3: Heterogeneous Ensemble Architecture

We did the experiments with stacking using the iteration 1 process, i.e. using the training set and test set number 1 for all the experiences. Stacking have to training processes, the first train is done to obtain the predictions and use it as predictors for the second train phase which is done on the meta learn level. So the training set is divided in two training sets with the same size and the same outcome class distribution, the one used in the baseline level is called ensemble dataset and the data used in the meta learn level is called blender dataset as illustrated in figure 6.3.

Four experiences were conducted, two of them using the normal training set using the heterogeneous and the homogeneous stacking and the other two using the "SMOTEed" training set also using hetero-homogeneous stacking.

Input Data	Type	Accuracy	Recall	Precision
Normal training set	Homogeneous	0.8254	0.36364	0.57143
	Heterogeneous	0.8413	0.6364	0.5833
SMOTEed training set	Homogeneous	0.8889	0.6364	0.7778
	Heterogeneous	0.8889	0.6364	0.7778

Table 6.2: Stacking Experiences

We can observe in table 6.2 the results of those four experiments. The SMOTEed data as training set achieved the best results so far. Besides the two results from the SMOTEed training set in relation to the evaluation metrics being the same, in figure 6.4 we can notice a slight difference on the predictions. One warning case was classified as failure in the homogeneous approach while in the heterogeneous approach it was classified as normal.

Homogeneous (left) vs Heterogenous (right)
(SMOTEes Data)

```
Confusion Matrix and Statistics

          Reference
Prediction Failure normal warning
   Failure       2      0       2
   normal        1     47       2
   warning       1      1       7

Overall Statistics

               Accuracy : 0.8889
```

```
Confusion Matrix and Statistics

          Reference
Prediction Failure normal warning
   Failure       2      0       1
   normal        1     47       3
   warning       1      1       7

Overall Statistics

               Accuracy : 0.8889
```

Figure 6.4: Confusion Matrix's

Chapter 7

Conclusion

In this document, we proposed a methodology for the prediction of failures with event alarms data from onboard systems.

7.1 Conclusion

First we processed the data-creating feature capable to describe the data in the best way possible to characterize the problem. Those features retained the inherent problems that come with this data and might influence the model understand of the problem such as the unbalance of the predictors' values, unbalance of the target classes, the huge number of predictors vs samples. Also, the lack of information about what really represents the Id of the fail events restrained the capability of filtering those features without losing information, so a more detailed study about the Id's might increase the results.

To create the models, it was used the well-known package caret, function train. The evaluation of the tuning process was done be the function train a 10 fold cross validation over a windowing. Such decision is due to the fact that multiple experiments were conducted using the sliding window or the growing window with multiple combinations of the parameters that reached results always two percentage points behind the cross validation results.

In the evaluation using the test set, the nnet algorithm stood out in the overall results being always in top positions. Neural network achieved two times the best accuracy, three times the best recall and one time the best precision. Since accuracy does not have into account the unbalance in the data, Recall and Precision are then two interesting metrics to evaluate the model. For a given class, the recall indicates the percentage of samples of this class that have been successfully labeled while precision indicates the percentage of samples attributed to their real class. The accuracy presented is always from the confusion matrix of the model while precision and recall are the measures of the target class (warning) in an approach one-against-all. However, statistically, no model distinguished itself from the others in a pairwise comparison. When the unbalance of the classes was taken in account using SMOTE, the results improved in all important evaluation measures. Finally, using the stacking technique in the over sampled training data set, namely four

base classifiers and a meta learner (random forest), the results reached the highest scores of all experiments conducted.

Conclusively, the satisfactory results of this project lead it to be included in a business proposal to a Swedish Nomad Client. And their answer was positive asking for a implementation of the system using their data.

7.2 Future Work

As feature work we propose to conduct a detailed study about the features to filtering it in the way that it could increase a result performance. Some simple techniques were experimented in this work such as the filter Relief or predictors correlation filter however it did not improve the results so we kept all the available features.

Also and the most important is to study the model generalization. This data set have not all the available Id code errors that exist in the datasheet, it only have the codes that occurred at least once in the data time span. This study will allow to understand whether it is necessary to train a model for each train from the same fleet or if it is possible to generalize it without losing quality.

References

[1] Hongfei Li, Buyue Qian, Dhaivat Parikh, and Arun Hampapur. Alarm prediction in large-scale sensor networks - A case study in railroad. *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013*, pages 7–14, 2013. doi:10.1109/BigData.2013.6691771.

[2] Wissam Sammouri, Etienne C^, Latifa Oukhellou, Patrice Aknin, and Charles-eric Fonlladosa. Pattern recognition approach for the prediction of infrequent target events in floating train data sequences within a predictive maintenance framework. 2014.

[3] Sebastian Kauschke, Immanuel Schweizer, Michael Fiebrig, and Frederik Janssen. Learning to Predict Component Failures in Trains. (September 2014):8–10.

[4] Max Kuhn and Kjell Johnson. *Applied Predictive Modeling*. 2013. URL: http://link.springer.com/10.1007/978-1-4614-6849-3, doi:10.1007/978-1-4614-6849-3.

[5] Luis Torgo. *Data Mining with R: Learning with Case Studies*. 2010.

[6] Quinlan R. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[7] Joy A. Thomas Thomas M. Cover. *Elements of Information Theory, 2nd Edition*. 2006.

[8] Warren; Walter Pitts McCulloch. A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, 1943. doi:10.1007/BF02478259.

[9] Daniel and Svozil; Vladimir KvasniEka; Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 1997.

[10] Gavin Brown. Ensemble Learning. 2009. URL: http://eprints.pascal-network.org/archive/00004769/, doi:10.1007/978-0-387-73003-5_293.

[11] L. Breiman. Bagging predictors. *Machine Learning*, pages 123–140, 1996.

[12] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. *13th International Conference on Machine Learning*, 1996.

[13] Annette M. Molinaro, Richard Simon, and Ruth M. Pfeiffer. Prediction error estimation: A comparison of resampling methods. *Bioinformatics*, 21(15):3301–3307, 2005. doi:10.1093/bioinformatics/bti499.

[14] Peter Willett. Universities of Leeds , Sheffield and York Dissimilarity-Based Algorithms For Selecting Structurally Diverse Sets Of Compounds. 6, 1999.

[15] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. arXiv:1106.1813, doi:10.1613/jair.953.

[16] Michael Buckland and Fredric Gey. The relationship between Recall and Precision. *Journal of the American Society for Information Science*, 45(1):12–19, 1994. doi:10.1002/(SICI)1097-4571(199401)45:1<12::AID-ASI2>3.0.CO;2-L.

[17] D. G. Altman and J. M. Bland. Diagnostic tests 3: receiver operating characteristic plots.

[18] Christopher D. Brown and Herbert T. Davis. Receiver operating characteristics curves and related decision measures: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 80(1):24–38, 2006. doi:10.1016/j.chemolab.2005.05.004.

[19] Jialiang Li and Jason P. Fine. ROC analysis with multiple classes and multiple tests: Methodology and its application in microarray studies. *Biostatistics*, 9(3):566–576, 2008. doi:10.1093/biostatistics/kxm050.

[20] Daisuke Tsujinishi and Shigeo Abe. University Repository : Kernel Why Pairwise Is Better than One-against-All or All-at-Once. pages 693–698, 2015.

[21] Alina Beygelzimer, John Langford, and Bianca Zadrozny. Machine Learning Techniques - Reductions Between Prediction Quality Metrics. *Performance Modeling and Engineering*, pages 3–28, 2008. URL: http://www.springerlink.com/index/10.1007/978-0-387-79361-0, doi:10.1007/978-0-387-79361-0.

[22] Gary M Weiss and Foster J Provost. Learning When Training Data are Costly: The Effect of ClassDistribution on Tree Induction. *J. Artif. Intell. Res. (JAIR)*, 19:315–354, 2003. arXiv:1106.4557, doi:10.1613/jair.1199.

[23] N V Chawla, N Japkowicz, and a Kolcz. Editorial: special issue on learning from imbalanced data sets. *ACM SIGKDD Explorations*, 6(1):1–6, 2004. URL: http://www.acm.org/sigs/sigkdd/explorations/issues/6-1-2004-06/edit_intro.pdf.

[24] P. Hart. The condensed nearest neighbor rule (corresp.). *Information Theory, IEEE Transactions on*, 14(3):515–516, May 1968. doi:10.1109/TIT.1968.1054155.

[25] Ivan Tomek. Two Modifications of CNN, 1976. doi:10.1109/TSMC.1976.4309452.

[26] Dennis L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. *Systems, Man and Cybernetics, IEEE Transactions on*, SMC-2(3):408–421, July 1972. doi:10.1109/TSMC.1972.4309137.

[27] Olivier Caelen Andrea Dal Pozzolo and Gianluca Bontempi. Comparison of balancing techniques for unbalanced datasets. Technical report, Machine Learning Group University of Bruxelles, Belgium.

[28] SAS. Data Visualization. URL: http://www.sas.com/en_us/insights/big-data.

[29] Max Kuhn. Building predictive models in R using the caret package. *J. Stat. Softw., 28(5)*, pages 1–26, 2008.

[30] Joaquin Vanschoren, Hendrik Blockeel, Bernhard. Pfahringer and Geo rey Holmes. A new way to share, organize and learn from experiments. *Machine Learning*, pages 127–158, 2012.

[31] Lothar Sachs. *Angewandte Statistik*. Springer, 1997.

[32] J Demšar and J Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, 7:1–30, 2006. doi:10.1016/j.jecp.2010.03.005.

[33] Douglas A. Hollander; Wolfe. *Nonparametric Statistical Methods*. 1999.

[34] Alexander K Seewald, Alex Seewald At, Alexsee Ai, and Univie Ac. Meta-Learning for Stacked Classification. 2000.