Remote Condition Monitoring: Application to Railway Traction Systems

Rafael Mendes Borges
rafael.m.borges@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa, Portugal
November 2018

Abstract

Maintenance is a crucial aspect for any transportation operator. The predictive maintenance strategy improves fleet availability while reducing costs. This approach aims to anticipate equipment failures, allowing to schedule maintenance activities accordingly, preventing downtime and the severe consequences of breakdowns during operation. This thesis presents a data-driven approach based on data science tools for failure prediction applied to railway traction systems, using real data collected by a remote condition monitoring system. The first steps include the exploratory data analysis and dimensionality reduction. Then, the engineering problem is represented from a data science point of view, for which the expert knowledge perspective is considered. Lastly, a scheme for failure prediction is designed and machine learning algorithms together with strategies for imbalanced datasets are used to learn from data. Important results were achieved as all failures were correctly predicted in the test set, meaning that every fault is anticipated. This allows repairing the system just before its breakdown, avoiding failures during operation and their potentially serious repercussions. However, the false positive rate (0.40) is higher than desired, leading to unnecessary maintenance operations, but these are expected to have a lower impact than not predicting a true failure. From an operation’s perspective, this solution represents a cost reduction if a repair costs at least 2.22 times more than a maintenance activity, which is clearly below the estimated relative cost (3.7 – 6.1). Future work directions are discussed, from alternative learning methods to solutions which were designed but not performed due to time constraints.

Keywords: failure prediction, predictive maintenance, data-driven approach, railway traction systems, remote condition monitoring

1. Introduction

Predictive maintenance (PdM) tries to predict the time of failure, allowing maintenance to be planned accordingly. This technique requires intelligent systems capable of predicting failures based on condition monitoring [1]. One of the approaches, usually called a model-driven approach, consist of using rule engines based on failure modes. Another type of methods is known as a knowledge-based approach employing methods like fuzzy logic and expert systems [2]. The current paradigm of PdM, generally referred to as a data-driven approach, takes advantage of data science, machine learning and big data tools. These include machine learning algorithms like neural networks, decision trees and support vector machines (SVMs) [3].

This work addresses a real industry problem for which an engineering solution is being sought. Some trains from Portuguese Railways (CP) received a refurbishment on its traction converter. In addition, these trains were equipped with a remote condition monitoring (RCM) system, allowing the extraction of relevant data, in particular from the traction system. Since this is the first time a refurbishment like this was done in this train series, there is an interest in deepening the knowledge on the new system, specially regarding its failures and modern approaches of maintenance.

This work presents a PdM approach using data science and machine learning methods. Data collected from two different train systems is used to predict failures on the traction system. The data was collected over time representing several variables that monitor the train. Also, events were recorded, namely those representing failures. Some of these events have also several variables associated (variables of context), which are collected every time a failure is detected. Our goal is to predict four failure events of the traction system. The idea of this work is to use a learning from data approach, as opposed to the traditional ones based on failure modes and rule engines [2]. This work follows the usual steps involved in a data science project, of course adapted to the circumstances that are particular to this case.

PdM approaches using machine learning tools have been discussed in various studies, some of which address problems common to this work.
In [4], a machine learning approach for failure prediction is proposed. The authors start by using principal component analysis for dimensionality reduction of the dataset. Then, SVMs with strategies for imbalanced data are used to solve the classification problem. Since the dataset is imbalanced, appropriate performance metrics are used. The authors of [5] propose an approach to predict train wheel faults by combining multiple classifiers using decision trees and Naïve Bayes algorithms. This work proposes a different evaluation method, joining the common metrics for imbalanced data with a reward function to reinforce early and penalize late alerts. A log-based PdM approach for failure prediction in medical devices is proposed in [6]. The data comes from equipment event logs, labeled as normal/fault. The classifier intends to label both single events and events in an interval.

2. Background

2.1. Train

The data used in this work comes from a passenger train operated in Lisbon suburban area. Regarding the nomenclature used, train or unit refers to four rail cars; vehicle stand for two rail cars (motor car + trailer car); M1 and M2 designate each vehicle that composes the train. M1 refers always to the same vehicle and the same is valid for M2. The motor cars are on the edges and the trailers in the middle. The traction motors are placed in the two motor cars. The train has four traction groups: M1 has two and M2 the remaining two. To simplify the notation, the traction group #i of vehicle Mj is referred to as Mj,Gi.

2.2. Sources of Data

The available data is collected by an RCM system installed in the train used for this work. Several files containing the data were provided. These are divided by system – central control unit (CCU) and traction control unit (TCU) – and types of data. The CCU is the core of the train control system (supervision, control and command of the principal train parameters) while the TCU is specially designed for the control and command of a traction converter.

The information can be divided into variables, events and variables of context. The CCU variables are mostly real-valued parameters of the train with an associated timestamp and GPS data, which are recorded whenever the train is working, at a certain sample rate. Events represent failures and, for that reason, they are stored whenever the train detects a failure. Each log has the fault description, the start and end timestamps, and the GPS coordinates relative to the location where the event started. Both CCU and TCU have specific events. Finally, the variables of context are exclusive from the TCU and they are collected every time a TCU event starts. They represent the values of several parameters and also flags indicating internal commands and states of the traction system. Their values correspond to the instant when its associated event starts.

2.3. Problem Formulation

The problem addressed by this thesis was presented from a railway engineering point of view. As it is always necessary for real problems, a challenging process is followed to adapt the problem from the expert knowledge to the data science viewpoint. Figure 1 summarizes a possible interpretation of the steps involved. Once a suitable formulation is found, the initial problem can be adequately represented as a data science problem without compromising the real engineering context.

Fig. 1: Several iterations are necessary to find an appropriate representation of the problem, which satisfies the requirements from both expert knowledge and data science domains. Problem formulation cycle.

2.4. Exploratory Data Analysis (EDA)

The EDA is a way of looking at the data, allowing a better comprehension of its main characteristics. It can be done using multiple techniques, usually making use of graphical methods. The analysis made is based on [7]; the main statistics, histograms and other figures representing the data were used in the analysis.

2.5. Dimensionality Reduction

The principal component analysis (PCA) is probably the most widely used technique for dimension reduction of datasets. This technique analyzes a data table containing observations described by many dependent variables, generally inter-correlated. Its goal is to extract the relevant information from the data table in order to represent it as a set of new orthogonal variables known as principal components. PCA generates an approximation of a numeric data table, in terms of the product of two smaller matrices, which capture the fundamental patterns of the data table [8].

The data table is represented by the matrix $A \in \mathbb{R}^{m \times n}$, where $m$ is the number of observations and $n$ the number of features. In general, before performing the analysis, the matrix $A$ is preprocessed by adjusting its columns so that they
have zero mean and unit norm. It is done for each variable by subtracting its mean and diving by its norm. This standardization rescales the data in order to compensate for unequal scaling in different features [9]. PCA finds a matrix \( Z \in \mathbb{R}^{m \times n} \) of rank \( k < \min(m, n) \) that best approximates \( A \) in the least-squares sense. The constraint of the rank can be encoded implicitly by expressing \( Z \) in a factored form as \( Z = XY \), with \( X \in \mathbb{R}^{m \times k} \) and \( Y \in \mathbb{R}^{k \times n} \), whose factorization is not unique. The problem is reduced to choosing \( X \) and \( Y \) in order to minimize \( \|A - XY\|_F^2 \) (Frobenius norm of a matrix) [10].

The PCA technique can be used for numeric matrices not containing missing entries. When only some entries are observed, this method cannot be used. PCA is also not suitable for data tables containing non-numeric features, such as Boolean or categorical data. Generalized low rank models (GLRM) is a method which generalizes PCA on an arbitrary dataset composed of different data types, e.g. numerical, Boolean and categorical, allowing to have missing entries [11].

Consider that the matrix \( A \) has missing entries so that only the entries \( A_{ij} \), for \((i, j) \in \Omega \subseteq \{1, \ldots, m\} \times \{1, \ldots, n\}\) are observed, and that the entries of \( A \) can have different data types. For each data type, a loss function is given, which describes the approximation error incurred when a feature value \( a \) is represented by the number \( u \in \mathbb{R} \). In addition, one must provide regularizers, which are used to prevent the overfitting to the observations.

The problem can be adapted to allow a similar approach to that commonly used in PCA, which is the standardization of the data. In this technique, instead of performing this on the data, the loss functions are rescaled, preserving the sparsity of the dataset. Finally, \( x_i \in \mathbb{R}^{1 \times n} \) and \( y_j \in \mathbb{R}^m \) are defined to be the \( i \)th row of \( X \) and the \( j \)th column of \( Y \), respectively. Therefore, \( x_i y_j \in \mathbb{R} \) denotes a inner product. These considerations lead to a final formulation of the GLRM problem [10],

\[
\begin{align*}
\text{minimize} & \quad \sum_{(i,j) \in \Omega} L_{ij} (x_i y_j + \mu_j, A_{ij}) / \sigma_j^2 \\
& \quad + \sum_{i=1}^m r_i(x_i) + \sum_{j=1}^n \tilde{r}_j(y_j),
\end{align*}
\]

with variables \( x_i \), \( y_j \) and \( \mu_j \), where, for each feature \( j \),

\[
\begin{align*}
\mu_j &= \arg \min_{\mu} \sum_{i: (i,j) \in \Omega} L_{ij}(\mu, A_{ij}) ,\\
\sigma_j^2 &= \frac{1}{n_j - 1} \sum_{i: (i,j) \in \Omega} L_{ij}(\mu_j, A_{ij} ).
\end{align*}
\]

A solution of this problem gives an estimate \( \hat{A}_{ij} = x_i y_j \) for the value of the entries \((i,j) \notin \Omega \) that were not observed. Considering a matrix where most entries are missing (it is the case of interest in this work), this method is used to guess all its entries, given just a few of them [10].

In this work, two different loss functions were used: for real-valued data, the quadratic loss, \( L(u, a) = (u - a)^2 \), which is a simple and commonly used loss function; for Boolean data, the Hinge loss, \( L(u, a) = \max(1 - au, 0) \), representing the value “true” with +1 and “false” with -1. The Hinge loss is chosen since it punishes misclassifications. The regularizers used were \( r(x) = \gamma \|x\|_2 \) and \( \tilde{r}(y) = \gamma \|y\|_2 \), to prevent overfitting. For the case of \( \tilde{r}(y) \), a sparsity inducing regularizer was also tested, \( \tilde{r}(y) = \gamma \|y\|_1 \), to verify if inducing sparsity on matrix \( Y \) could reduce the problem to only some features while preserving significant information present in the original data.

Note that the each row of \( X \) represents an example by a vector in \( \mathbb{R}^k \). The solution to the problem gives a compressed and real-valued representation that may be used to efficiently store the information present in the original dataset [11].

The authors of [10] developed a package to solve this problem. The approach to solve it uses alternating minimization, which iteratively keeps one of \( X \), \( Y \) fixed and optimize over the other, then switch and repeat. The overall problem is non-convex due to bi-linearity, which is why in general it cannot be solved globally. Even though, it can be locally solved by alternating minimization, where each subproblem is convex and can be solved efficiently. Despite the wide usage of bi-linear representation and alternating minimization, global optimality guarantees for such methods are still lacking [11, 12].

2.6. Model Selection

When selecting a model, a way of testing the different values for each parameter is required in order to choose the most adequate ones. One of the most used techniques is the k-fold cross-validation, which is utilized in this work. The dataset is split into training and test sets. Then, for model selection, the training set is split into \( k \) folds (subsets of equal size). The number of folds \( k \) is chosen to be equal to the number of models to test. For each model, \( k-1 \) folds are used for training and one fold is used for test. The process is repeated \( k \) times such that each fold is used once as test set. After this, the model achieving the best performance is selected. Then, the entire training set is used to train the algorithm and the final evaluation is done using the test set, guaranteeing that the final model is tested utilizing data that was never used before [13].

2.7. Classification

Two different classifiers are used in this work: support vector machines (SVMs) and decision trees.
SVMs are appropriated to deal with not too large datasets, which is the case. They are fast and provide linear models. SVMs are not explainable, but it is not a requirement. High accuracy values can be obtained, specially when using kernels, but losing speed. Decision trees are fast training and explainable algorithms while achieving high accuracy values. Even though, decision trees have high variance since they are sensitive to the training data. This variance can be reduced using adequate techniques, like bagging, which is described later.

SVMs divide the data by defining a hyperplane in a high-dimensional feature space. Its objective consists of maximizing the separability of the two classes. The hyperplane is expressed in the input space using the support vectors, which are the closest points to the hyperplane. The optimal hyperplane is the one that maximizes the margin between samples from the two classes. Since the boundary between classes is usually not linear, the problem can be solved by mapping the data from the input space into a high dimension feature space, where the data can be separated by a hyperplane. This is done using kernel functions [14]. Two kernels are used in this work: linear and radial basis function.

Decision trees divide the training data into smaller subsets so that the label variables in each subset are as homogeneous as possible. The classifier is assumed to be constant in each subset and the class is chosen by majority vote. For a given data point, the label is found by going through the tree and making decisions based on the feature values. There are noisy labels both on training and test sets, which consist of examples leading to misclassification because they have the same attributes but different labels. This motivates the use of measures of impurity. Training a tree consists of finding a tree which minimizes the impurity.

2.8. Appropriate Metrics for Imbalanced Data
It is very common to have a dataset whose classes are not equally distributed. Moreover, frequently the most important samples belong the classes having less points. This is exactly what happens in failure prediction, in general. The samples representing faults are expected to be much less but more important than the ones associated with a normal condition. These problems require a high detection in the minority class rather than the majority class. For this reason, simple predictive accuracy is not a good measure. An important concept when defining performance metrics is the confusion matrix, as shown in the Figure 2. It is a representation of the predicted label compared to the true class.

The most common metrics for imbalanced data are derived from the interpretation of the confusion matrix and two examples are precision \(\frac{TP}{TP+FP}\) and recall \(\frac{TP}{TP+FN}\). Using an example from the context of failure prediction, precision measures what proportion of predicted failures was actually correct. Recall measures what proportion of real failures was identified correctly. Ideally, both metrics should be close to one. For each particular problem, one of these metrics can be considered as the more relevant.

Other tools are frequently used for classifier performance evaluation. A generic method, not specifically designed for imbalance data, uses the receiver operating characteristic (ROC) curve. It represents the classifier performance over combinations of true positive and false positive error rates. The true positive rate (TPR) is the same as the recall, and the false positive rate is defined as FPR = FP/(FP + TN). The objective of the ROC curves is to be in the upper left hand of the ROC space. A traditional metric for the ROC curves is the area under the curve (AUC), that should be as close to one as possible. The AUC provides an aggregate measure of performance across all thresholds. The curve is obtained from the scores extracted from the classifier. Different confusion matrices are created, from which the points on the ROC space are generated. These points are then interpolated. This is a general method and it is not the most adequate for imbalance data, as the ROC curve may provide a too optimistic view of the algorithm performance. For this case, an alternative is commonly used, the precision-recall (PR) curve. It is obtained similarly to what is done for the ROC curve, but computing precision and recall. Contrary to what happens with the ROC curve, the AUC of the PR curve does not make a relevant contribution. Generally, the comparison can be made by considering that the curves closer to the upper right corner have a better performance [15].

2.9. Strategies for Imbalanced Data
Many techniques were developed to mitigate the problems with imbalanced datasets. Four different strategies were chosen to be used in this work: SVMs with weighted classes, undersampling,
SMOTE and balanced bagging. The first two methods are classic and simple strategies, which is why they are tested. Two more advanced techniques are also used.

SVMs with weighted classes consist of an adjustment done to the soft margin of the SVM algorithm, in a way that the weights for each class become inversely proportional to the class frequencies. This intends to increase the penalty for minority class misclassification to prevent the minority class samples from being overwhelmed by the majority class.

The sampling methods consist of the modification of an imbalanced dataset to get a balanced distribution. In general, a balanced dataset achieves better performances compared to imbalanced datasets [16]. The first sampling technique used in this work consists of undersampling the majority class by randomly removing samples from it until reaching the desired class size. The percentage of undersampling can be adjusted, for which cross-validation is used. The majority class can be reduced to have a size similar to the minority class.

The authors of [17] proposed a combination of undersampling the majority class and oversampling the minority class, called synthetic minority oversampling technique (SMOTE). The oversampling of the minority class is done creating synthetic data points. These are introduced along the line segments joining any or all the $k$ minority class nearest neighbors. According to the amount of oversampling needed, the necessary neighbors from these $k$ neighbors are randomly chosen. Using this method, the decision region of the minority class becomes more general. The undersampling of the majority class is done by randomly removing samples from it until achieving the desired class size. Finally, the selection of the parameters (undersampling and oversampling rates, and the number of neighbors) can be done using cross-validation.

Undersampling the majority class is a good strategy but has a problem which is its high variance. This leads to different performances of classifiers over individual bootstrap examples. The solution proposed in [18] is to use the bagging (bootstrap aggregating) variance-reduction ensemble method. The balanced bagging method can be summarized as follows: an ensemble composed of a given number of models is built, each induced over an undersampled bootstrapped sample of the training data. The bootstrap samples are generated by sampling randomly with replacement and then the undersampling of the majority class is done removing instances randomly until the classes are balanced. When a new instance is to be classified, each model makes a prediction, and the final prediction is obtained as the majority vote. In general, this approach will improve classifier performance when the models of the ensemble are high-variance, which is the case. This method is usually applied with decision trees as the base classifiers, which will also be the case in this work. Finally, cross-validation can be used to select the parameters needed: the number of models and the size of the bootstrap samples.

3. Implementation

3.1. Exploratory Data Analysis (EDA)

After extracting the data from the files using MATLAB, the EDA was performed. This analysis was very important as it allowed to detect various problems with the data. At the same time, it provided an useful overview of the data, in particular concerning the most important events. In summary, some of the contributions of EDA were:

- It was found that the first days of the CCU variables contained faulty data and these had to be removed.
- Each type of data has its own missing pattern, except the TCU variables of context which doesn’t have missing data.
- Regarding the CCU variables, 2 out of 19 are constant; for the case of the TCU variables of context, these are 485 out of 939.
- The ranges of the variables are very diverse, as they represent many different parameters (most of them with unkown meaning).
- The important events were registered 55 times (one of these four events never occurred). M1 registered 71% of these events.

3.2. Problem Formulation

Several steps were needed to achieve the final representation of the problem. The main details which were being figured out and included are presented below:

1. There are two severe and two very severe events which represent the failure to be predicted. Each of the four traction groups has two individual events (one severe and one very severe).
2. The very severe event is always preceded by a very severe event; a severe event can happen without any very severe event.
3. In some situations, a train reset done by the driver may recover from a severe event (perhaps preceded by a very severe event) to a normal condition.
4. There are two types of failure: permanent and non-permanent. It is possible to identify a permanent failure reading its indication in the end timestamp, but only from raw files (non pre-processed by the RCM system).
5. The repair of a permanent failure is only done by maintenance and is not visible from the data available; maintenance information is needed.

The considerations above lead to the diagram of states shown in the Figure 3. These states can be inferred from the data and are later used as the label for the machine learning algorithms.
Division of data into blocks. Each block, obtaining the label consists of finding the state of the correspondent traction group, according to the diagram of states.

The CCU variables are collected continuously, while the remaining types are registered only when failures occur. This implies that most rows of the dataset contain only the CCU variables and these correspond to a small fraction of the features. Most features are related to the TCU variables of context and their frequency is not comparable. This implies that their columns have missing entries for most rows of the dataset. The case of events is different; they are never missing entries since the timestamps not having a specific event are coded as zeros in the respective column. Therefore, events do not originate missing entries. Considering the relative size of the variables of context (85.4%) in the number of features, it is easily understood the reason why most entries (85.24%) of the dataset are missing. Finally, the dataset has 780852 rows and 479 columns.

3.4. Dimensionality Reduction

This dataset has two significant characteristics: the high number of features and also the large amount of missing entries. These two challenging properties motivate the use of GLRM, which is used to both dimensionality reduction and imputation of missing values. In order to use GLRM, several inputs to build the model need to be considered. The inputs to be selected are: the rank \(k\), the regularizer \(\hat{r}(y)\) and the scales of both regularizers, \(\gamma\) and \(\tilde{\gamma}\). The k-fold cross-validation technique is applied for model selection.

Due to the time requirements of GLRM on such a large dataset, some non-ideal solutions had to be used. The first consists of using only a small randomly chosen subset from the entire dataset and then applying the k-fold cross-validation technique to this small portion of the dataset. In addition, since there are several possibilities to combine the parameters to choose, the number of folds would be enormous. The strategy used consist of refining gradually the parameters until finding the best model, by applying k-fold cross-validation as many times as necessary. Of course, to a certain degree and not refining indefinitely. The performance metric to evaluate each model is the normalized reconstruction error.

The first tests allowed to conclude that the quadratic regularization performs better than the sparsity-inducing, which is why the latter was discarded. Regarding the rank \(k\), the training error decreases when increasing the rank, as expected. However, the behavior of the test error is not as expected, that is decreasing at first but then increasing. For some tests, it slightly decreases in the beginning and then increases; in other cases, it is always increasing or oscillating. This unexpected behavior can be caused by optimization errors or by
the randomness of the samples in the test set. Possibly the data used for test in cross-validation is particularly bad in some cases but in other test sets it is especially good. This problem would be mitigated if more data could be used for cross-validation, which is not possible due to time constraints. Finally, the quadratic regularization leads to a well-conditioned problem, which is why it is preferred.

Figures 5 and 6 present the training and test errors (respectively) for the final refinement using quadratic regularization on both X and Y. It shows what was discussed above. The training errors have an expected behavior while the test errors do not. From these results, the final model is selected to have rank \( k = 25 \) and quadratic regularization with scales \( \gamma = 0 \) and \( \tilde{\gamma} = 2 \) for \( r(x) \) and \( \tilde{r}(y) \) respectively. This represents a considerable reduction on the number of features when comparing to the original dataset (25 instead of 479 columns), meaning that the size of matrix \( X \) is 5.22% of the original dataset size.

The value obtained for the regularization scale on \( X \), \( \gamma = 0 \), implies that the regularization term on \( X \) vanishes. We are aware that it can be polarizing our solution. Nevertheless, this value was obtained from cross-validation using the normalized reconstruction error as the metric for evaluation, which is why we consider it as an adequate solution.

![Fig. 5: The training errors decrease while increasing the rank \( k \), as expected. Cross-validation training errors using quadratic regularization.](image)

After selecting the model, the final step consists of applying the algorithm with these inputs using the entire dataset to get the low rank approximation. Then, the matrix \( X \) is used in the place of the original dataset. This representation allows to have a much more compact dataset without missing entries. Again, GLRM time restrictions motivated a non-ideal approach and the stop criteria had to be relaxed, otherwise the fitting process would take too much time. This causes the low rank approximation to be much less accurate than desired.

3.5. Classification

This Section addresses the final goal of this work, which is failure prediction. This challenge requires a new dataset, that is built from the original dataset.

The scheme for failure prediction is shown in Figure 7, representing how to extract the data from the original dataset. The green block represents the data points which are used to predict if there will be a failure afterwards. The label of the upcoming data is the blue block. The orange square is the label obtained for prediction, which is one if there is any failure on the blue block and zero otherwise. The data points from the green block are then vectorized to be a row on the new dataset for prediction whose corresponding label is the value of the orange block. These steps are repeated by sliding on time, an operation that is represented by the dashed blocks. A new row results from these dashed blocks. The dataset for prediction is built moving forward on the original dataset until reaching the end, that is when the blue block reaches the last data point. Note that this operation is done separately for each traction group, using the correspondent data.

The time window for prediction and the horizon to be predicted are set to be one week. Then, the sliding window step used is one day, allowing to have more weeks to train the algorithm. Finally, no delay is used, meaning that the time windows are consecutive: data from one week is used to predict if there is a fault in the next week. These blocks are defined to begin always at midnight, meaning that this tool can be used whenever a new day begins.

![Fig. 7: New rows for the prediction dataset are generated by applying the same method when moving forward in time. Scheme for prediction.](image)
As expected, the weeks do not have the same number of data points. The size of each week varies significantly, from 6,000 to 18,000 data points, approximately. When building the dataset for prediction, the ideal is to have synchronous data points for every week. This makes the features of the prediction dataset corresponding to equivalent times of the day, in order to preserve the time characteristics. However, it doesn’t happen having weeks with so different sizes. Figure 8 represents the data points for some weeks over time. There are big holes inside several weeks, and these are propagated along some days because of the sliding window of one day.

Fig. 8: Several weeks have big gaps; these are propagated because of the sliding window of one day. Even the other weeks are very irregular. Distribution of data from weeks of M1,G1.

The limitations related to GLRM imply that no solution involving oversampling can be used, which would be the desired way to deal with the problem of synchronization. The solution found to cope with this problem is undersampling. The week is divided into slots, whose size can be selected. For each slot, it is calculated the minimum number of points in all weeks. Then, for the weeks to have the same number of data points, it is necessary to remove points in all weeks until all of them having the same number of points. The minimum can be zero, implying that all points are removed from that slot. In order to minimize the damage of removing data, the algorithm chooses the points using a semi-random criteria. Firstly, it removes randomly points originally coming from the CCU data. Then, if more points need to be removed, it randomly removes points originally coming from the TCU data. This allows to keep as much TCU data as possible, as it is the more relevant one (traction specific data).

Figure 8 shows the distribution of data from where several big holes are visible. Even using big slots (order of hours), at least one week would not have any point in each slot. A solution needs to be designed to overcome the obstacle. It can be done into two parts. Firstly, the weeks containing big holes are removed and not used. For this, a minimum number of data points is used as the criterion. Most weeks having big holes coincide with longer maintenance periods. The second part of the solution consists of choosing a longer slot duration. Removing the weeks containing big holes doesn’t solve the problem by itself since one week not having data in a slot is enough to force all the data in that slot to be removed. Therefore, the solution consists of using a slot of two hours, which is much bigger than desired but, considering the context, it is not so long that the synchronization is severely affected. Figure 9 shows the distribution of data after applying this undersampling approach. After performing undersampling, the week size is much smaller than the mean week size before removing data. It represents a reduction of more than 50%.

At this point, the dataset obtained by applying undersampling is ready for classification. Regarding the label, the dataset is more or less balanced (57%/43%). The division of this dataset into training and test was not done by randomly dividing it into training and test sets. The approach employed was using the most recent weeks of each traction group as the test set. This has two purposes: firstly, not using weeks for training and test containing data points in common, which could compromise the confidence of the results. This happens because of the sliding window, as explained before, which makes the same data to be in several weeks. Even using the most recent weeks as the test set, some weeks have to be removed to prevent the test set to contain data in common to the training data, again because of the sliding window. Secondly, this way simulates what would be the real application of the algorithm: receiving unseen data from the train and trying to make a prediction for the future based on historical data.

4. Results
This Section presents the main results of this work, obtained using the dataset for failure prediction.

SVMs were tested without imbalance methods, for which a precision and recall of 0.33 and 0.40 were obtained using a linear SVM, respectively. These
values are very far from the desired, which is why it makes sense to try an alternative approach. Even though the prediction dataset is more or less balanced, the imbalanced methods discussed before were tested. Figure 10 presents the results of the imbalance methods. These results represent an improvement when compared to the previous ones not using strategies for imbalanced data. The accuracy seems to not be significantly affected, but the same doesn’t apply for precision and recall, which can be considered as the most important parameters. The bagging technique achieves the highest values for both precision and recall, and also accuracy (0.70). The precision still low but the recall is the maximum (1.0). Although bagging cannot be considered as an optimal solution due to its low precision, its recall value is very important. It means that all weeks having failures are correctly labeled, which is a relevant feature of this prediction algorithm. Figure 10 shows also both the ROC and PR curves. These curves confirm bagging as the best solution among the strategies tested, achieving the highest AUC in the ROC curve and being the closest curve to the upper right corner in the PR curve.

(a) Bagging has the highest AUC (0.83), outperforming the remaining methods: weighted SVM (0.57), undersampling (0.53) and SMOTE (0.57). ROC curves.

(b) The bagging curve is the closest to the upper right corner, outperforming the other strategies. PR curves.

Fig. 10: Both ROC and PR curves verify bagging as the best solution. ROC and PR curves for imbalanced methods (markers representing the observed values).

One must conclude that the results are far from the expected for this type of application. Even so, the analysis of the results must take into account all the limitations. For example, it is not known how noisy the data is. As happened with the temperature data discarded at the EDA stage, other sources can be introducing relevant noise which was not detected due to the huge amount of variables (most of them with unknown meaning).

5. Conclusions
This work addressed the problem of failure prediction on railways traction systems, for which a data-driven approach was used to design a solution, making use of data science and machine learning tools. This work resulted from the desire of finding an innovative solution for a real engineering problem. For this reason, it involved a challenging process of interpretation. Naturally, this kind of processes with real data give rise to several difficulties. The EDA was very important. It allowed to discard irrelevant variables and, at the same time, previously unknown problems were detected.

The process of joining all the types of data into a single dataset end up with a considerable number of features and a huge percentage of missing entries. These characteristics motivate the use of GLRM, both for dimension reduction and missing imputation. Despite being a powerful tool to perform these steps, GLRM has also some drawbacks, in particular the time required for each fit. These time restrictions motivated some undesired changes to have results in a reasonable time in order to proceed to the next phases of the work, implying a much less accurate approximation. Nevertheless, GLRM allowed to find a low rank approximation with a much lower number of features and, at the same time, performing the imputation of an extreme number of missing entries.

When designing the solution for failure prediction, an oversampling approach was considered, allowing to maintain the synchronism of the data. Even so, this strategy could not be adopted, again because of time constraints. An alternative approach consisting of undersampling was used, in which the new dataset was built from the approximation of the original dataset. Many consequences resulted from this solution. Firstly, several weeks were not used due to the irregularity of the data in time, with some weeks having much less data than the others. Even discarding those weeks, more than 50% of the data was removed when synchronizing the data, despite using a relaxed synchronism criterion.

Even though the dataset used for failure prediction was relatively balanced, the use of methods for imbalanced data considerably improved the results. The precision and recall for the best-performing technique (bagging) achieved 45% and 100%, respectively. This represents a lower precision than desired but a maximum value for recall, which is a
very important feature for a failure prediction algorithm. The optimal recall means that all weeks having failures are correctly predicted. This guarantees that all failures are anticipated, allowing to repair the system before its breakdown. Moreover, it avoids the serious consequences of a failure occurring during operation, which can force the train to stop, causing high financial losses and comprising all the operation schedule, which is even worse since it is a passenger train. On the other hand, the precision value below what one might expect implies that the algorithm incorrectly labels some weeks as having failures. This leads to unnecessary maintenance operations, negatively affecting the train availability. Nevertheless, these negative consequences are expected to have a lower impact than not predicting a true failure.

One can compare this solution with a baseline scenario of planned maintenance, assuming some simplifications: the baseline scenario only performs periodic maintenance activities and all failures imply repairs; all failures predicted by the proposed solution lead to maintenance activities, avoiding repairs for the cases of correct predictions. From a quantitative perspective, the test set contains 15 weeks of normal condition and 5 weeks having failures. The proposed solution detected all failures. However, this solution incorrectly predicted 6 weeks as having failures, adding 6 unnecessary maintenance activities to those of periodic maintenance. Considering a baseline scenario of planned maintenance, the 5 failures would lead to 5 repairs (which are all avoided using the proposed solution). Even tough, the baseline scenario would require only the periodic maintenance activities. From an operation’s perspective, this solution represents a cost reduction if a repair costs at least 2.22 times more than a maintenance activity. From domain knowledge, we know an estimate for this relative cost, which is between 3.7 and 6.1. Therefore, considering the results on the test set, the proposed solution represents a cost reduction when compared to the planned maintenance strategy.

Despite having achieved significant results, one would expect to obtain better results if the preferred alternatives could be used. All the code developed is prepared to perform those alternatives, but the time constraints did not allow to go ahead with those approaches. Having more time would certainly contribute to attain a better solution, but the data itself plays also an important role. Firstly, using more data would improve the model. This work was restricted to the data available by that time. In addition, the quality of the data is also a crucial aspect as noisy data can severely influence the results. One must conclude that important results were achieved, especially considering all the limitations. Finally, most of this work is not specific for railways. These tools can be applied in a wide range of domains and applications.

Further improvements could be achieved by considering the temporal correlations. In addition, there are various other methods that can be used for different phases of this work. Regarding the strategies for failure prediction, other approaches could be tested, such as outlier detection and time series analysis.

References