IN2DREAMS

Contract No. 777596

INtelligent solutions 2ward the Development of Railway Energy and Asset Management Systems in Europe

D5.3: Visual Analytics of Railway Data and Models

DUE DATE OF DELIVERABLE: 30/11/2018

ACTUAL SUBMISSION DATE: 19/11/2018

Leader/Responsible of this Deliverable: Wolfgang Jentner - UKON

Reviewed: Y

Document status		
Revision	Date	Description
0.1	30/05/2018	First Issue of the TOC
0.2	01/08/2018	UNIGE Completed Sections 1.1, 2.2, 2.4, 2.6, and 2.7
0.3	19/09/2018	UNIGE Completed Section 2.3
0.4	31/09/2018	UNIGE Completed Section 2.5 its last contribution
0.5	15/10/2018	UKON Completed the Executive Summary, Section 2.1, & 3
0.6	17/10/2018	UKON Added Abbreviations Table
1.0	06/11/2018	UKON Implemented feedback from EE, RFI, and CEFRIEL
2.0	19/11/2018	Final version after TMT approval and quality check

Project funded from the European Union’s Horizon 2020 research and innovation programme		
Dissemination Level		
PU	Public	X
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

Start date of project: 01/09/2017 - Duration: 24 Months

Executive Summary

The general objective of WP5 is to study, design and develop data analytics solutions for knowledge extraction from railway asset data. This objective will be achieved through the following tasks:

- Definition of data analytics scenarios (T5.1 - D5.1);
- Development and demonstration of tools and methodologies aiming at extracting knowledge from data analytics algorithms, and contemporarily making them interpretable in an easier way (T5.4 - D5.3 & T5.3 - D5.4);
- Study and develop the proof-of-concept of metrics and methods/tools to measure the accuracy of analytics algorithms (T5.2 - D5.2).

This deliverable builds upon D5.1 and reports the work that has been conducted for each of the defined scenarios. The scenarios focus on relevant railway assets whose malfunction and maintenance policies have an impact on the KPIs targeted by the SHIFT2RAIL program. The cross-scenarios cover many aspects of the railway ecosystem while the five specific-scenarios focus on a single particular aspect.

- **Cross-Scenario 1: Visualizations in the Control Center**
  The work described in CS1 details optimizations and enhancements of the visualizations for various systems at RFI. It is further described how interactive visualizations are introduced to provide additional information to the users. The work carried out prepares the inclusion of the blockchain-technology developed in WP4 and the data-driven-models as described SS3 for a demonstrator (D5.4);
- **Cross-Scenario 2: Marketplace of Data and Data Monetization**
  CS2 is discontinued because of the absence of a Marketplace for Data which will not be developed in WP4;
- **Specific-Scenario 1: Track Circuits**
  Described is the visual analytics approach and the development of interpretable gray-box models to enable predictive maintenance. The results can be exploited in IN2SMART WP8. The defined metrics are evaluated in T5.2;
- **Specific-Scenario 2: Train Delays and Penalties**
  This scenario details the development of a hybrid-model consisting of interpretable and experience-based models that often lack accuracy, and less interpretable data-driven-models whereas the combination can mitigate the disadvantages for each type of model;
- **Specific-Scenario 3: Restoration Time**
  Describes the development of a fully data-driven-model solution and the usage of diagnostic models in combination with visualizations to extract knowledge about the functional dependencies between input and output variables;
- **Specific-Scenario 4: Switches**
  Discontinued because of time and resource constraints;
- **Specific-Scenario 5: Train Energy Consumption**
  The work on the predictive models is conducted in WP6. Their quality will be reported in D5.2.

Abbreviations and Acronyms

Abbreviation	Description
AI	Artificial Intelligence
AMS	Asset Management System
CBI	Computer-Based Interlocking system
CBM	Condition-Based Maintenance
CS	Cross-Scenario
IM	Infrastructure Manager
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
ML	Machine Learning
PM	Predictive Maintenance
PoC	Proof of Concept
RFI	Rete Ferroviaria Italiana
RTS	Railway Transport System
SR	Strukton Rail
SS	Specific-Scenario
SVG	Scalable Vector Graphics
TCS	Track Circuit System
TMS	Traffic Management System
TO	Train Operator
UNIGE	University of Genoa
UKON	University of Konstanz
VA	Visual Analytics
WP	Work-Package
WS	Work-Stream

Table of Contents

List of Figures

List of Tables

1 Introduction

Ultimately, the operational performance of a train system directly depends on the many individual decisions operators make in their daily job routine. Consequently, it stands to reason to support decision makers as optimal as possible, especially for positions where decisions of a single operator have potentially large impacts on complex train schedules.

In complex systems, it is rather the rule than the exception that the information and knowledge needed to make informed decisions is distributed over different parts of one system or even entirely different systems. Having to put together information from heterogeneous environments is challenging and error-prone, which in turn creates uncertainties in the operators workflow. As Ellis and Dix [26] point out, “decision making under uncertainty can result in cognitive biases and irrational decisions”. Such effects, also as described by Kahnemann and Egan [41], should be minimized, especially in the light of the consequences operational decisions can have in TMS environments.

While it is impossible for an operator to oversee all consequences of his actions, incorporating our knowledge of previous decisions and developments of operational scenarios can help structuring the information necessary for decision making. For example, an operator could be provided with solution suggestions based on previous decisions in similar scenarios. To use this kind of knowledge, models have to be created which encode the records of past situations and compare these events in real-time to present ones. As well, conclusions provided by such models do not necessarily consist of singular solutions, but allow to choose from options optimized for different outcomes. For example, delay management could optimize for passenger impact, overall punctuality or most economic solution. Yet, regarding the responsibility resting on typical users in their decision making, it is also important for an expert to be able to verify the reasons why a model is suggesting a solution. Consequently, it is of importance to efficiently communicate the decision criteria to users to build trust [39].

In this deliverable, the examples of several scenarios illustrate novel approaches to aid operators and planners in various scopes of responsibility by providing them both a comprehensive overview on a situation and condensed information for decision making. To do so, the *Visual Analytics* [45] approach is employed which aims to interactively bring together algorithmic models and visualization displays for efficient, informed decision making. Before the Visual Analytics approach is introduced in Section 1.2, Section 1.1 provides an overview on the challenges in the creation of models for knowledge representation. Following, sections 2.1 to 2.7 illustrate challenges, progress and intended solutions for the scenarios defined in D5.1.

The scenarios introduced in sections 2.3, 2.4 and 2.5 detail the fundamental models for the status identification and extraction from the Track Circuit System, the role and analysis of train delays and penalties and the estimation of recovery time in case of failures. The results of these scenarios are the fundamental building blocks for the visualizations in the control center presented to the operators. Section 2.1 covers the Visual Analytics approach in the control center, which provides visual and interactive access to the knowledge extracted in the aforementioned scenarios.

1.1 Rule-Based Methods for Knowledge Extraction

As already described in D5.1, sometimes, even if a data analytics model performs well, having solely a model able to make predictions is not enough. In these cases more than the answer itself it is important why the model made a certain decision since a single metric, such as classification accuracy, is an incomplete description of most real-world tasks [23].
In predictive modeling, you have to make a trade-off: do we simply want to know what is predicted (i.e. the probability that a switch will brake or a score for the effectiveness of some train dispatching solution) or do we want to know why that prediction was made, possibly paying for the interpretability with a drop in accuracy? In some cases we do not care why a decision was made, only the assurance that the predictive performance was good enough but in other cases, knowing why can help understand more about the problem, the data and why a model might fail. Some models might not need explanations, because they are used in a low risk environment, meaning a mistake has no severe consequences, (e.g. a train prediction system) or the method has already been extensively studied and evaluated. The necessity for interpretability comes from an incompleteness in the problem formalization [23], meaning that for certain problems or tasks it is not enough to get the answer (the what). The model also has to give an explanation about how it came to the answer (the why), because a correct prediction only partially solves the original problem. The following reasons drive the demand for interpretability and explanations [23, 58]:

- Human curiosity and learning. Humans have a mental model of their environment, which gets updated when something unexpected happens;
- Find meaning in the world. We want to reconcile contradictions or inconsistencies between elements of our knowledge structures;
- Data analytics models are taking over real world tasks, that demand safety measurements and testing;
- By default most data analytics models pick up biases from the training data;
- The process of integrating machines and algorithms into our daily lives demands interpretability to increase social acceptance;
- Explanations are used to manage social interactions;
- Only with interpretability data analytics algorithms can be debugged and audited.

If you can ensure that the data analytics model can explain decisions, the following traits can also be checked more easily [12, 22–25, 37, 64]:

- Fairness: Making sure the predictions are unbiased and not discriminating against protected groups (implicit or explicit). An interpretable model can tell why it decided that a certain person is not worthy of a credit and for a human it becomes easier to judge if the decision was based on a learned demographic (e.g. racial) bias;
- Privacy: Ensuring that sensitive information in the data is protected;
- Reliability or Robustness: Test that small changes in the input don't lead to big changes in the prediction;
- Causality: Check if only causal relationships are picked up, meaning a predicted change in a decision due to arbitrary changes in the input values is also happening in reality;
- Trust: It is easier for humans to trust a system that explains its decisions compared to a black box.

The most straightforward way to get to interpretable data analytics is to use only a subset of algorithms that create interpretable models [84]. Very common model types of this group of interpretable models are:

- Linear models: linear models have been used since a long time by statisticians, computer scientists, and other people tackling quantitative problems. Linear models learn linear (and therefore monotonic) relationships between the features and the target. The linearity of the learned relationship makes the interpretation easy [35];

- Decision trees: tree-based models split the data according to certain cutoff values in the features multiple times. Splitting means that different subsets of the dataset are created, where each instance belongs to one subset. The final subsets are called terminal or leaf nodes and the intermediate subsets are called internal nodes or split nodes. For predicting the outcome in each leaf node, a simple model is fitted with the instances in these subsets. Trees can be used for classification and regression [35];
- Rule-based models: for example, the RuleFit algorithm [29] fits sparse linear models which include automatically detected interaction effects in the form of binary decision rules.

1.1.1 Linear Models

Linear models [10, 35, 59, 71, 78] learn linear (and therefore monotonic) relationships between the features and the target. The linearity of the learned relationship makes the interpretation easy. Moreover their linearity makes the estimation procedure straightforward and, most importantly, these linear equations have an easy to understand interpretation on a modular level (i.e. the weights). That is one of the main reasons why the linear model and all similar models are so widespread in academic fields like medicine, sociology, psychology, and many more quantitative research fields. Linear models also come with some assumptions that make them easy to use and interpret but which are often not satisfied in reality. The assumptions are: Linearity, normality, homoscedasticity, independence, fixed features, and absence of multicollinearity.
The interpretation of a weight in the linear model depends on the type of the corresponding feature:

- Numerical feature: for an increase of the numerical feature by one unit, the estimated outcome changes by the corresponding weight for that feature;
- Binary feature: a feature, that for each instance takes one of two possible values. One of the values counts as the reference level. A change of the feature from the reference level to the other level changes the estimated outcome by the corresponding weight for that feature;
- Categorical feature with multiple levels: in this case the problem is overparameterized, for this reason the categorical feature must be coded with the one-hot-encoding schema and, in this way, we can exploit again the approach described for the binary features.

Linear models have obviously also disadvantages. Linear models can only represent linear relationships. Each non-linearity or interaction has to be hand-crafted and explicitly given to the model as an input feature. Linear models are also often not that good regarding predictive performance, because the relationships that can be learned are so restricted and usually oversimplifies how complex reality is. The interpretation of a weight can be unintuitive because it depends on all other features. A feature with high positive correlation with the outcome and another feature might get a negative weight in the linear model, because, given the other correlated feature, it is negatively correlated with in the high-dimensional space. Completely correlated features make it even impossible to find a unique solution for the linear equation.
Linear models have been extensively studied and extended to fix some of the shortcomings.

- Lasso is a method to pressure weights of irrelevant features to get an estimate of zero. Having unimportant features weighted by zero is useful, because having less terms to interpret makes the model more interpretable;
- Generalised Linear Models allow the target outcome to have different distributions. The target outcome is no longer required to be normally distributed given the features, but Generalised Linear Models allow you to model for example Poisson distributed count variables. Logistic regression, is a Generalised Linear Model for categorical outcomes;
- Generalised additive models are Generalised Linear Models with the additional ability to allow non-linear relationships with features, while maintaining the linear equation structure.

It is possible to apply all sorts of tricks to go around some of the problems related to linear models:

- Adding interactions: it is possible to define interactions between features and add them as new features before estimating the linear model. The RuleFit algorithm can add interactions automatically;
- Adding non-linear terms like polynomials to allow non-linear relationships with features;
- Stratifying data by feature and fitting linear models on subsets.

Finally in reality we might not have just a handful of features, but hundreds or thousands and, in this case interpretability goes downriver. There are also situations with more features than instances and in this case it is not possible to fit a standard linear model at all. The most automatic and convenient way to introduce sparsity is to use the Lasso method. Lasso stands for Least Absolute Shrinkage and Selection Operator and when added to a linear model, it performs feature selection and regularisation of the selected feature weights. Lasso is not the only solution, a big spectrum of methods can be used to reduce the number of features in a linear model.
Methods that include a pre-processing step:

- Hand selected features: it is possible to use expert knowledge to choose and discard some features. The big drawback is, that it can't be automated and you might not be an expert;
- Use some measures to pre-select features: an example is the correlation coefficient. You only take features into account that exceed some chosen threshold of correlation between the feature and the target. Disadvantage is that it only looks at the features one at a time. Some features might only show correlation after the linear model has accounted for some other features. Those you will miss with this approach.

Step-wise procedures:

- Forward selection: fit the linear model with one feature. Do that with each feature. Choose the model that works best. Now again, for the remaining features, fit different versions of your model by adding each feature to your chosen model. Pick the one that performs best. Continue until some criterium is reached, like the maximum number of features in the model;
- Backward selection: same as forward selection, but instead of adding features, start with the model that includes all features and try out which feature you have to remove to get the highest performance increase. Repeat until some stopping criterium is reached.

In the case when we want to apply linear models to classification problems (e.g. binary $\{0,1\}$ classification problems) using, as a model, a simple linear combination some problems arise.

- A linear model does not output probabilities, but it treats the classes as numbers (0 and 1) and fits the best linear model (if you have one feature, it's a line) that minimises the distances between the points and the model. So it simply interpolates between the points, but there is no meaning in it and you cannot interpret it as probabilities;
- Also a linear model will extrapolate the features and give you values below zero and above one, which are not meaningful and should tell you that there might be a more clever approach to classification;
- Since the predicted outcome is not a probability but some linear interpolation between points there is no meaningful threshold at which you can distinguish one class from the other;
- Linear models don't extend to classification problems with multiple classes.

A solution for classification is logistic models. Instead of fitting a straight line or hyperplane, the logistic model uses a non-linear function, the logistic function to squeeze the output of a linear equation between 0 and 1. The interpretation of the logistic models weights differs from the linear model case, because in logistic models the outcome is a probability between 0 and 1, and the weights don't affect the probability linearly, but are squeezed through the logistic function.

Here are the interpretations for the logistic model with different feature types:

- Numerical feature: for an increase of one unit of the feature the estimated odds change (multiplicatively) by a factor proportional to the exponential function of the corresponding weights;
- Binary categorical feature: one of the two values of the feature is the reference level. A change of the feature from the reference level to the other level changes the estimated odds (multiplicatively) by a factor proportional to the exponential function of its corresponding weights;
- Categorical feature with many levels: again, as for linear models, the feature must be coded in multiple binary features.

1.1.2 Decision Trees

Linear models and logistic models fail in situations where the relationship between features and outcome is non-linear or where the features are interacting with each other. Decision trees can fill this gap [48, 61, 66, 67, 79, 86, 89]. Tree-based models split the data according to certain cutoff values in the features multiple times. Splitting means that different subsets of the dataset are created, where each instance belongs to one subset. The final subsets are called terminal or leaf nodes and the intermediate subsets are called internal nodes or split nodes. For predicting the outcome in each leaf node, a simple model is fitted with the instances in this subset (for example the subsets average target outcome). Trees can be used for classification and regression. There are a lot of tree algorithms with different approaches for how to grow a tree. They differ in the possible structure of the tree (e.g. number of splits per node), criteria for how to find the splits, when to stop splitting and how to estimate the simple models within the leaf nodes. Classification and regression trees is one of the more popular algorithms for tree induction.
The interpretation of decision trees is simple: starting from the root node you go to the next nodes and the edges tell you which subsets you are looking at. Once you reach the leaf node, the node tells you the predicted outcome. All the edges are connected by a logic AND.
The tree structure is perfectly suited to cover interactions between features in the data. The data also ends up in distinct groups, which are often easier to grasp than points on a hyperplane like in linear model. The interpretation is arguably pretty straightforward. The tree structure also has a natural visualization, with its nodes and edges. Trees create good explanations as defined here. The tree structure automatically invites to think about predicted values for single instances in a counterfactual way: if a feature would have been bigger / smaller than the split point, the prediction would have been different? The created explanations are contrastive, because you can always compare the prediction of an instance with relevant (as defined by the tree) what-if-scenarios, which are simply the other leaf nodes of the tree. If the tree is short, like one to three splits deep, the resulting explanations are selective. A tree with a depth of three needs a maximum of three features and split points to create the explanation for the prediction of an instance. The truthfulness of the prediction depends on the predictive performance of the tree. The explanations for short trees are very simple and general, because for each split, the instance either falls into one or the other leave and binary decisions are easy to understand. There is no need to transform features. In linear models it is sometimes necessary to take the logarithm of a feature. A decision tree can handle a feature regardless of monotonic transformations.
Handling of linear relationships, that's what trees cannot do. Any linear relationship between an input feature and the outcome has to be approximated by hard splits, which produces a step function. This is not efficient. This goes hand in hand with lack of smoothness. Slight changes in the input feature can have a big impact on the predicted outcome, which might not be desirable.

Trees are also quite unstable, so a few changes in the training dataset might create a completely different tree. That's because each split depends on the parent split. And if a different feature gets selected as the first split feature, the whole tree structure will change. It does not generate confidence in the model if the structure flips so easily.

1.1.3 Rule-Based Models

A decision rule is a simple IF-THEN statement consisting of a condition and a prediction. A single decision rule or a combination of several rules can be used to make predictions [11, 17, 29, 30, 49, 88].
Decision rules follow a general structure: IF the condition is true THEN make a particular prediction. Decision rules are probably the most interpretable prediction models. Their IF-THEN structure semantically resembles natural language and the way we think, provided that the condition is built from intelligible features, the length of the condition is short (number of *feature=value* pairs combined with an AND) and there are not too many rules. In programming it's very natural to write IF-THEN rules. New in machine learning is that the decision rules are learned through an algorithm.
A decision rule uses at least one *feature=value* statement in the condition, with no upper limit on how many more can be added with an AND. An exception is the default rule that has no explicit IF-part and that applies when no other rule applies, but more will be detailed later.
The usefulness of a decision rule is usually summarized in two

- Support of a rule: the percentage of instances to which the condition of a rule applies is called the support;
- Accuracy of a rule: the accuracy of a rule is a measure of how accurate the rule is in predicting the correct class for the instances to which the condition of the rule applies.

Usually there is a trade-off between accuracy and support: by adding more features in the condition, we can achieve higher accuracy, but lose support.
To create a good classifier for predicting the value of a house you might need to learn not only one rule, but maybe 10 or 20. Then things can get more complicated:

- Rules can overlap: what if I want to predict the value of a house and two or more rules apply and they give me contradictory predictions?;
- No rule applies: what if I want to predict the value of a house and none of the rules apply?

There are two main strategies for dealing with multiple rules: decision lists (ordered) and decision sets (un-ordered). Both strategies imply different solutions to the problem of overlapping rules.

- A decision list introduces an order to the decision rules. If the condition of the first rule is true for an instance, we use the prediction of the first rule. If not, we go to the next rule and check if it is true and so on. Decision lists are one-sided decision trees, where the first rule is the root node and with each rule, the tree grows in one direction. Decision lists solve the problem of overlapping rules by only returning the prediction of the first rule in the list that applies;
- A decision set resembles a democracy of the rules, except that some rules might have a higher voting power. In a set, the rules are either mutually exclusive, or there is a strategy for resolving conflicts, such as majority voting, which may be weighted by the individual rule accuracies or other quality measures. Interpretability suffers potentially when several rules apply.

Both decision lists and sets can suffer from the problem that no rule applies to an instance. This can be resolved by introducing a default rule. The default rule is the rule that applies when no other rule applies. The prediction of the default rule is often the most frequent class of the data points which are not covered by other rules. If a set or list of rules covers the entire feature space, we call it exhaustive.

By adding a default rule, a set or list automatically becomes exhaustive.
There are many ways to learn rules from data and this document is far from covering them all. The algorithms are chosen to cover a wide range of general ideas for learning rules, so all three of them represent very different approaches.

- OneR learns rules from a single feature. OneR is characterized by its simplicity, interpretability and its use as a benchmark;
- Sequential covering is a general procedure that iteratively learns rules and removes the data points that are covered by the new rule. This procedure is used by many rule learning algorithms;
- Bayesian Rule Lists combine pre-mined frequent patterns into a decision list using Bayesian statistics. Using pre-mined patterns is a common approach used by many rule learning algorithms.

There are many benefits of IF-THEN rules in general.

- IF-THEN rules are easy to interpret. They are probably the most interpretable of the interpretable models. This statement only applies if the number of rules is small, the conditions of the rules are short and if the rules are organised in a decision list or a non-overlapping decision set;
- Decision rules can be as expressive as decision trees, while being more compact. Decision tree often also suffer from replicated sub-trees, that is, when the splits following a left and a right child node have the same structure;
- The prediction with IF-THEN rules is fast, since only a few binary statements need to be checked to determine which rules apply;
- Decision rules are robust against monotonous transformations of the input features, because only the threshold in the conditions changes. They are also robust against outliers, since it only matters if a condition applies or not;
- IF-THEN rules usually generate sparse models, which means that not many features are included. They select only the relevant features for the model. For example, a linear model assigns a weight to every input feature by default. Features that are irrelevant can simply be ignored by IF-THEN rules;
- Simple rules like from OneR can be used as baseline for more complex algorithms.

Nevertheless, there are also downsides in IF-THEN rules in general.

- The research and literature for IF-THEN rules focuses on classification and almost completely neglects regression. While you can always divide a continuous target into intervals and turn it into a classification problem, you always lose information. In general, approaches are more attractive if they can be used for both regression and classification;
- Often the features also have to be categorical. That means numeric features must be binned, if you want to use them. There are many ways to cut a continuous feature into intervals, but this is not trivial and comes with many questions without clear answers. How many intervals should the feature be divided into? What's the splitting criteria: fixed interval lengths, quantiles or something else? Dealing with binning continuous features is a non-trivial issue that is often neglected and people just use the next best method (like I did in the examples);
- Many of the older rule-learning algorithms are prone to overfitting. The algorithms presented here all have at least some safeguards to prevent overfitting: OneR is limited because it can only use one feature (only problematic if the feature has too many levels or if there are many features, which equates to the multiple testing problem), RIPPER does pruning and Bayesian Rule Lists impose a prior distribution on the decision lists;

- Decision rules are bad in describing linear relationships between features and output. That's a problem they share with the decision trees. Decision trees and rules can only produce step-like prediction functions, where changes in the prediction are always jumps and never smooth curves. This is related to the issue that the inputs have to be categorical (in decision trees, they are implicitly categorized by splitting them).

An important algorithms for rule fitting purposes is the RuleFit algorithm [29]. The RuleFit algorithm fits sparse linear models which include automatically detected interaction effects in the form of binary decision rules.
The standard linear model doesn't account for interactions between the features so it is convenient to have a model that is as simple and interpretable as linear models, but that also integrates feature interactions. RuleFit addresses this issue and fits a sparse linear model with the original features and also a set of new features which are decision rules. These new features capture interactions between the original features. RuleFit generates these features automatically from decision trees. Each path through a tree can be turned into a decision rule by combining the split decisions to a rule.
These are trees that are trained to predict the outcome of interest, so that the splits are meaningful for the task at hand and not arbitrary. Any algorithm that creates a lot of trees can be used for RuleFit, like a Random Forest [14] for example. Each tree is disassembled into decision rules, which are used as additional features in a linear Lasso model.
RuleFit also comes with a feature importance measurement, which helps to identify linear terms and rules that are important for the prediction. The feature importance is calculated from the weights of the regression model. The importance measure can be aggregated for the original features (which appear once untransformed and possibly in many decision rules).
RuleFit also introduces partial dependence plots to plot the average change of the prediction by changing one feature. The partial dependence plot is a model-agnostic method, which can be used with any model, and it has its own part in the book.
The question that raises now is what are the advantages and disadvantages of RuleFit and how it is possible to interpret it?
The interpretation is analogue to linear models.
The advantages are:

- RuleFit adds feature interactions automatically to linear models. Therefore it solves the problem of linear models that you have to add interaction terms manually and it helps a bit with the issue of modeling non-linear relationships;
- RuleFit can handle both classification and regression tasks;
- The created rules are easy to interpret, because they are binary decision rules. Either the rule applies to an instance or not. Good interpretability is only guaranteed as long as the number of conditions within a rule is not to big. A rule with 1 to 3 conditions seems reasonable to me. This translates into a maximum depth of 3 for the trees in the tree ensemble;
- Even if there are many rules in the model, they do not apply to each instance, so for one instance only a handful of rules are important (non-zero weights). This improves local interpretability;
- The RuleFit proposes a bunch of useful diagnostic tools. These tools are model-agnostic, that's why you will find them in the model-agnostic section: feature importance, partial dependence plots and feature interactions.

While the disadvantages are:

- Sometimes RuleFit creates many rules which get a non-zero weight in the Lasso model. The interpretability degrades with higher number of features in the model. A promising solution is to force feature effects to be monotonic, meaning that an increase in a feature has to result in an increase of the predicted outcome;
- An anecdotal drawback: the papers claim good performance of RuleFit - often close to the predictive performance of Random Forests! - yet in the few cases where I personally tried it, the performance was disappointing;
- The end product of the RuleFit procedure is a linear model with additional fancy features (the decision rules). But since it is a linear model, the weight interpretation is still unintuitive.

1.2 Visual Analytics Methods for Knowledge Extraction

The challenges in the IN2DREAMS project comprise a variety of tasks substantially differing in nature: Processing and presenting, analysis and decision making, simulation and interpolation are very different duties whose integration cannot be achieved with conventional solutions. In addition, extracting relevant and meaningful information from heterogeneous data sources is notoriously complex and cumbersome.
Researchers have been trying to solve these problems through either automatic data analysis or interactive visualization approaches. However, only the combination of both approaches allows to leverage both the computational power of modern algorithms and machines as well as a user's experience and unmatched ability to perceive and interpret patterns.
Visual Analytics (VA) is an interdisciplinary approach towards complex data analysis scenarios based on this combination of man and machine. VA "combines automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex datasets", a definition given by Keim et al. as summary of the VisMaster EU research project [45]. Besides direct knowledge generation, following Visual Analytics principles also fosters a user's constructive reflection and correction of conducted analyses, resulting in improvements for processes and models, and ultimately, of decisions taken and knowledge generated by the users.
VA combines multiple research areas and subjects including data management and analysis, spatio-temporal data processing, statistics, human-computer-interaction and visualization [46]. It is intended to allow to derive insights from large, in-homogeneous and ambiguous datasets and enables both to confirm expected results as well as finding unexpected coherence. Users can quickly come to comprehensible, verifiable results and are able to communicate their findings and derived consequences for action efficiently. The Visual Analytics process has been described and modeled extensively. A refined view is provided by Sacha et al.[73] with the introduction of the Knowledge Generation Model. This model for VA defines and relates human and machine concepts embedded in a three-loop framework [73]. The model is shown in Figure 1 and consists of the VA process model on the left hand side and is related to human knowledge generation process on the right hand side. The model clearly conveys that lower-level processes, which are part of the exploration loop, are guided by higher-level analytic activities, which are part of the verification and knowledge generation loops.
The application of VA principles for knowledge extraction to specific scenarios is a process which can not be generalized, but that is dependent on the domain problem. Yet, VA workflows follow general rules which can be implemented for any data-analytical model creation process as applied in the train scheduling models applied in Cross-Scenario 1 in Section 2.1, for example as provided by Sacha et al. [72]. To incorporate expert knowledge and to explore parameter spaces, already at this point dedicated VA solutions should be deployed. Figure 2 illustrates the complex steps necessary for the creation of valid models with realistic prediction or classification results.

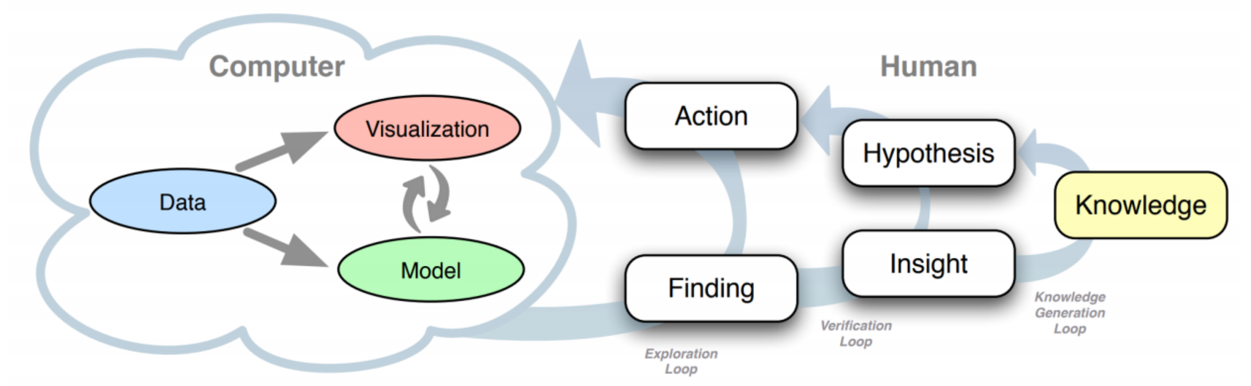


Figure 1: The Knowledge Generation Model for Visual Analytics. It describes the relation between the algorithmic processing of data and human data exploration and analysis through interactive visualization approaches. In combination, an expert goes through several loops to arrive at a conclusion.

Besides the model creation process itself, models already deployed in applications also have to adapt to changing situations and regimes. In other words, the models have to be able to learn from the decisions an operator makes, and, through active learning processes [77], incorporate the same into the digital knowledge representation. Consequently, the wealth of available sensor information and predictions generated by the models applied in the various scenarios can be employed to create a holistic view on a train system, merging information usually separated in different systems (e.g. maintenance, scheduling, delay management and ticket sales).
The integration of the collected data into models through interactive machine learning entails the adaptation of these models to process spatially and temporally changing data sources. Thus, changing spatial and temporal uncertainties have to be considered and adaptive interpolation methods implemented for the whole prediction process. Due to these circumstances, the development of suitable, progressive Visual Analytics systems as introduced by Stolperer et al. [81] is necessary to ensure the understanding of the whole process from data cleaning over model building to validation tasks by the experts. Progressive VA approaches enable analysts to intervene in the model learning process. By inspecting partial results directly when available, experts can influence the modelling process and put emphasis on interesting subspaces based on their expertise and domain knowledge. This way, the ML process can be steered early on and computationally costly, but ineffective parts of the decision space can potentially be avoided. This process heavily relies on the visualization of intermediary results and the interaction with the same, and consequently is very much suited for the application of VA approaches.

2 Achievements for WP5 Scenarios

WP5 targets many systems and domain-specific, real-world problems of the operators and asset managers working at RFI. As in D5.1 “Data Analytics Scenarios”, we divide this into 7 different scenarios whereas the Cross-Scenarios (1, 2) target multiple systems at once and the Specific-Scenarios (1-5) seek solutions to specific problems. However, the final demonstrator (D5.4) will merge the solutions of Cross-Scenario 1, Specific-Scenario 3, as well as the solutions in blockchain technology of Work-Package 4 (WP4). In the following, we detail the problems and requirements for each scenario, report on our progress and achievements, and discuss the findings and solutions.

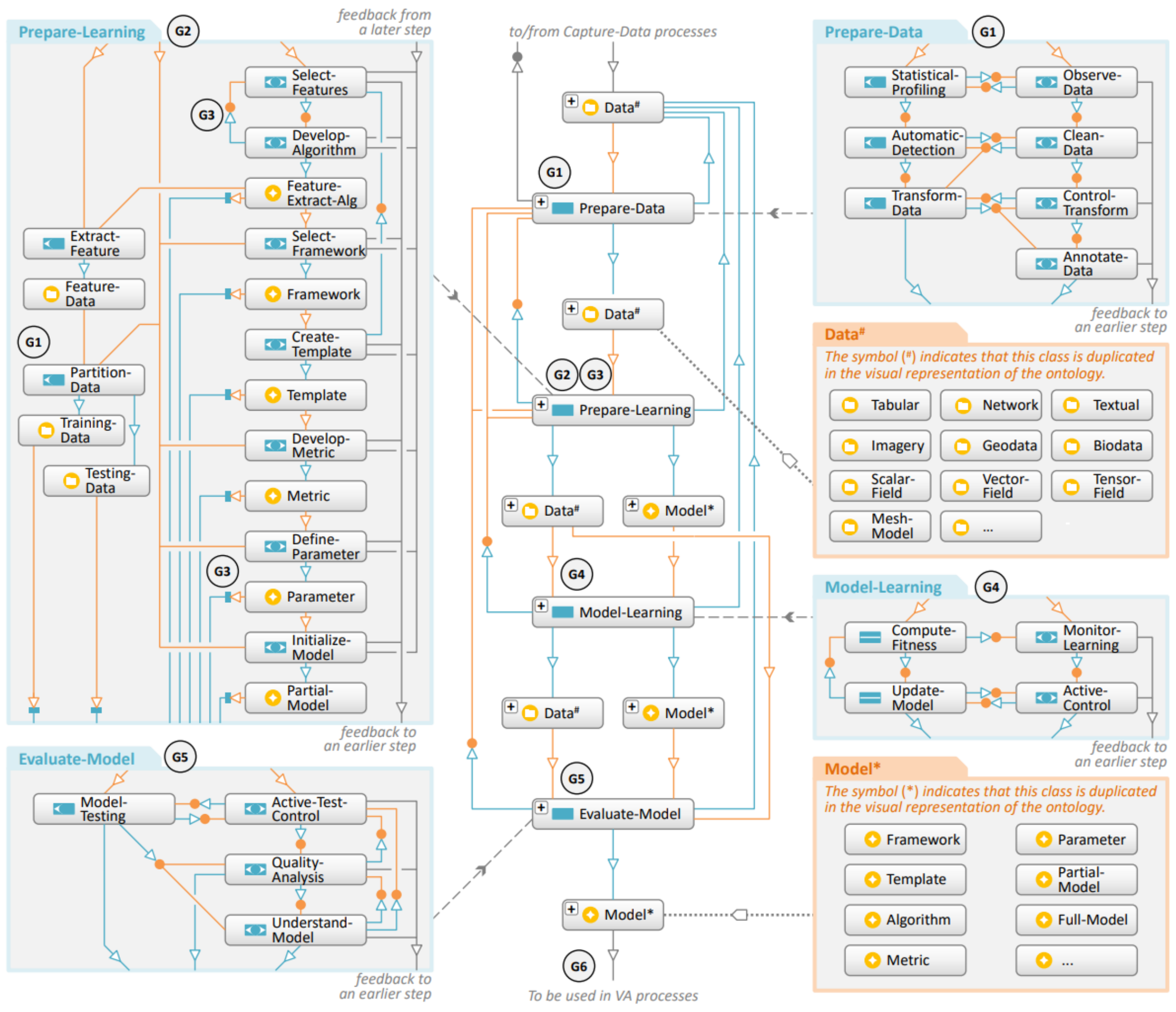


Figure 2: The ontology for VA-assisted machine learning. The major steps are Examining/Preparing Data (G1), Examining and Understanding the machine learning model (G2), feature and parameter analysis (G3), the learning process (G4), quality and result analysis (G5) and comparative analysis (G6).

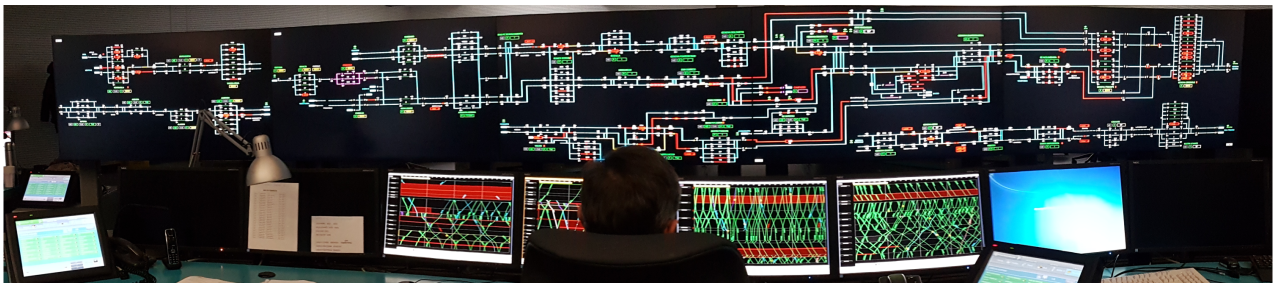


Figure 3: Cross-Scenario 1: The working station of one operator. In this picture the systems of Task 1 (lower four monitors) and 3 (upper five monitors) are being shown.

2.1 Cross-Scenario 1: Visualizations in Control Center

Cross-Scenario 1 (CS1) provides visualization techniques for the control room. More specifically, this involves the Traffic Management System (TMS, Figure 3) and the Asset Management System (AMS, Figure 4).
The goal is to improve the existing systems and enhance them with state-of-the-art visualization and visual analytics (VA) techniques. Especially the VA-part prepares to include the models generated in Specific-Scenario 3 (Section 2.3) in combination with the blockchain technologies developed in Work-Package 4 (WP4). This will provide the user with the ability to reason about the input of the models, their output and the model itself enabling her/him to receive a good understanding about the reasoning of a prediction model (Explainable-AI) as well as the impact of the outcomes. In general, this shall enable the operators and managers to gain a better overview of the systems as well as understand proposed resolutions and automatic predictions better. This also involves past decisions and available uncertainties. CS1 targets different systems, we therefore divide this scenario into three tasks:

1. Decision Support System for Rail-Conflict Resolution
2. Alert Management and Prioritization System for AMS
3. Improving the TMS and Directing the Awareness of the Operator

Each system respective to its task supports the user in various domain-tasks. Therefore, each of the tasks are developed and enhanced separately. However, the final PoC will include the models (Specific-Scenario 3, 2.3) and the blockchain-technologies (WP4) in more than one system. The integration will vary on the users necessities. For all tasks we follow the Design-Study Methodology of Sedlmair et al. [76]. We therefore report our progress in 5 Phases which describe essential changes in the design and features:

Phase 1 State-of-the-Art, Requirement Specification and Vectorization

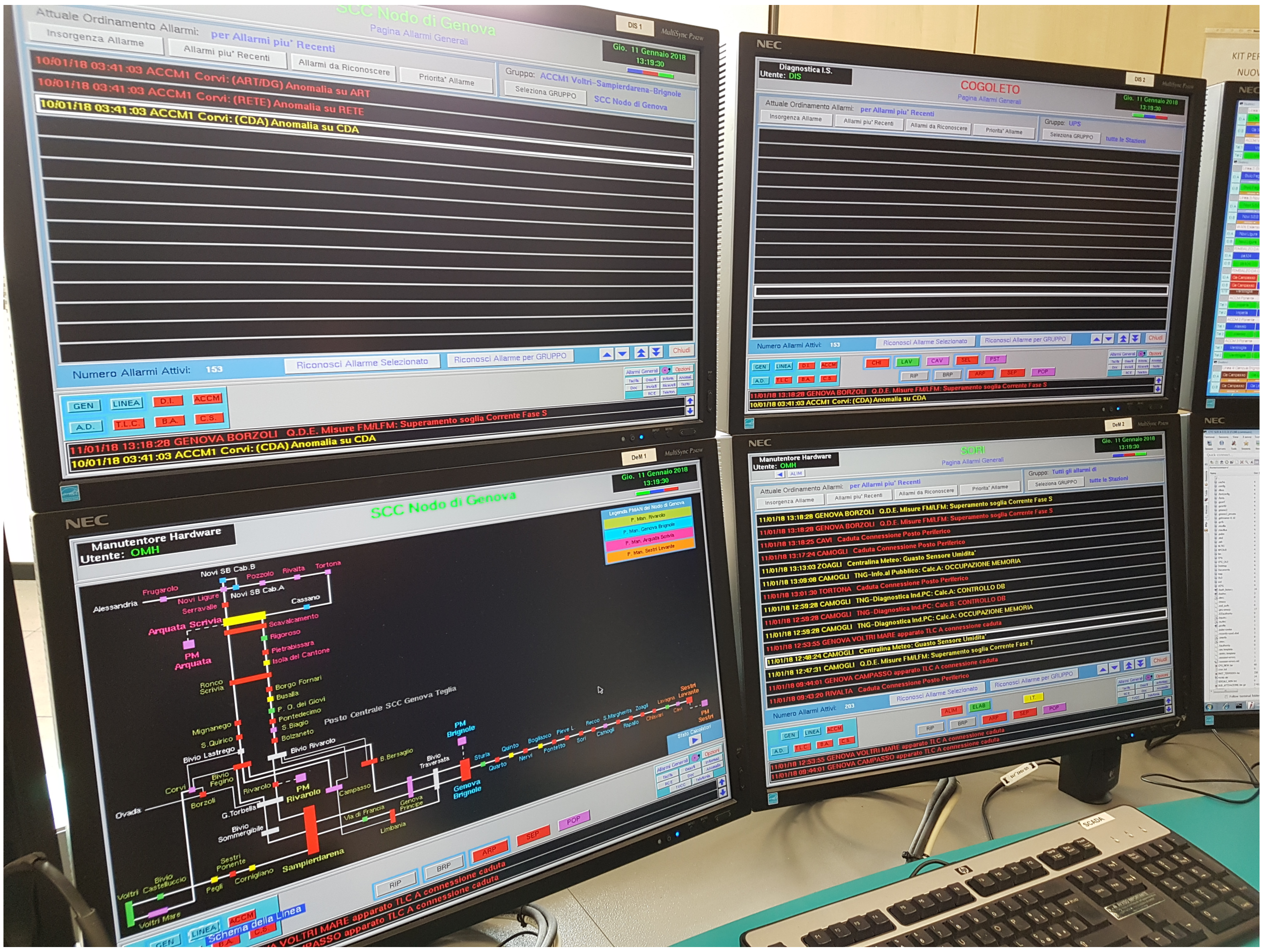
Phase 2 Adding Interactions and First Improvements

Phase 3 Datafication

Phase 4 Including Machine-Learning-Models

Phase 5 Finalization of PoC

Each of the phases includes a review-cycle with the end-users from RFI. Their feedback is being evaluated and included in the next phase, respectively.



```
10/01/18 03:41:03 ACCM1 Corvi: (ART/DG) Anomalia su ART
10/01/18 03:41:03 ACCM1 Corvi: (RETE) Anomalia su RETE
10/01/18 03:41:03 ACCM1 Corvi: (CDA) Anomalia su CDA

11/01/18 13:18:28 GENOVA BORZOLI  Q.D.E. Misure FM/LFM: Superamento soglia Corrente Fase S
10/01/18 03:41:03 ACCM1 Corvi: (CDA) Anomalia su CDA
```

```
11/01/18 13:18:28 GENOVA BORZOLI  Q.D.E. Misure FM/LFM: Superamento soglia Corrente Fase S
11/01/18 13:18:28 GENOVA BORZOLI  Q.D.E. Misure FM/LFM: Superamento soglia Corrente Fase S
11/01/18 13:18:25 CAVI  Caduta Connessione Posto Periferico
11/01/18 13:17:24 CAMOGLI  Caduta Connessione Posto Periferico
11/01/18 13:13:03 ZOAGLI  Centralina Meteo: Guasto Sensore Umidita'
11/01/18 13:09:08 CAMOGLI  TNG-Info.al Pubblico: Calc.A: OCCUPAZIONE MEMORIA
11/01/18 13:01:30 TORTONA  Caduta Connessione Posto Periferico
11/01/18 12:59:28 CAMOGLI  TNG-Diagnostica Ind.PC: Calc.A: CONTROLLO DB
11/01/18 12:59:28 CAMOGLI  TNG-Diagnostica Ind.PC: Calc.B: CONTROLLO DB
11/01/18 12:59:28 CAMOGLI  TNG-Diagnostica Ind.PC: Calc.A: OCCUPAZIONE MEMORIA
11/01/18 12:53:55 GENOVA VOLTRI MARE apparato TLC A connessione caduta
11/01/18 12:48:24 CAMOGLI  Centralina Meteo: Guasto Sensore Umidita'
11/01/18 12:47:31 CAMOGLI  Q.D.E. Misure FM/LFM: Superamento soglia Corrente Fase T
11/01/18 09:44:01 GENOVA CAMPASSO apparato TLC A connessione caduta
11/01/18 09:43:20 RIVALTA  Caduta Connessione Posto Periferico

11/01/18 12:53:55 GENOVA VOLTRI MARE apparato TLC A connessione caduta
11/01/18 09:44:01 GENOVA CAMPASSO apparato TLC A connessione caduta
```

Figure 4: Cross-Scenario 1: Parts of the working station of an asset manager. The lower left monitor shows an overview of an area displaying alarms generated at various train-stations. The remaining three monitors represent filtered lists of alarms for critical systems. Task 2 targets the topological overview system (lower left).

Phase 1 included three visits at different control-centers of RFI (Milan, Florence & Genoa). The purpose of these visits incorporates demonstrations of the systems, interviews with different operators to understand their tasks and domain-specific problems as well as a first requirement analysis and the features and improvements the operators wish for. From the pictures taken, we selected typical representatives for each system and vectorized these photos using Scalable Vector Graphics (SVG) [6]. SVG allows a seamless rendering on different monitors and enables us to present our current states on a website providing RFI an easy access to the visualizations. Feedback can then be received using video-calls.
In Phase 2, we added interaction possibilities and first improvements to our visualizations. This is mainly done using JavaScript with the D3-library [4]. The overall framework architecture is powered by Angular [2] which supports a development environment, stateful pages and provides many utilities.
Phase 3 describes the datafication. This enables us to assemble the interactive system with real data provided by RFI. Furthermore, it allows additional interactions that require re-rendering of the visualization. The information can be recomputed from the data and be plotted. Aside from this, feedback from RFI is implemented which was obtained during a feedback session at the end of Phase 2.
At the time of writing, we are currently in the beginning of Phase 4. The kick-off for this phase dates on 2018-09-27 at the 5th Work-Stream 2 (WS2) meeting held in Konstanz. At the same event, another feedback-session was conducted with RFI. This feedback is currently being evaluated and implemented as part of Phase 4.
Phase 5 will cover the finalization of the PoC (D5.4) and will again implement detailed feedback from RFI. Furthermore, the different systems of each task will be interconnected. This linking-and-brushing [44] will allow the user to seamlessly switch between the visualization systems and keeping track of the context in multi-screen environments.

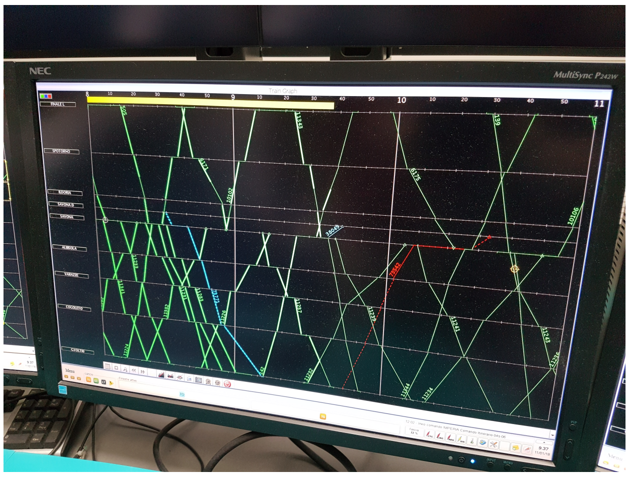
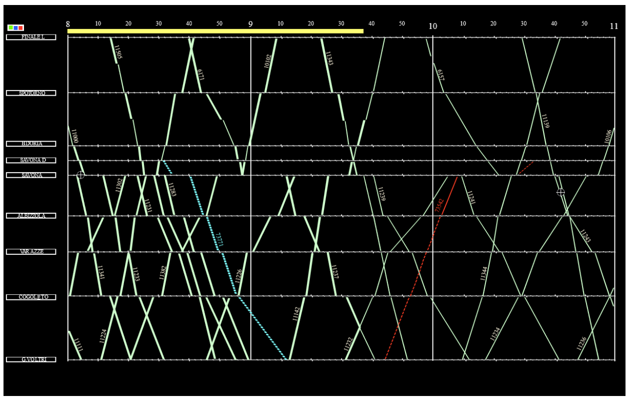
In the following, we report on the achievements and progress of the systems for each task in detail.

2.1.1 Task 1 - Decision Support System for Rail-Conflict Resolution

The purpose of this system is to give the operator a temporal overview of the train-schedule within a specific region she/he controls. The future train-schedule is predicted with a rule-based system based on the fix, programmed schedule of the trains.

2.1.1.1 Phase 1

Figure 5 shows the train schedule system of a specific region. The x-axis represents the time. The y-axis shows the different stations. The colored lines represent trains based on a schedule or real-data. Green are person-trains, blue are freight-trains, and red represent single locomotives. Everything right of the yellow bar shows future trains and there schedule according to a rule-based system. The yellow crosses (Figure 5a) represent conflicts. For a conflict the operator may decide which of the two trains can go first. The underlying rule-based system tries to resolve the conflict automatically. The operators stated that they typically wait with the resolution of a conflict if it is 20 or less minutes into the future. Otherwise the conflict may disappear due to false predictions of the system. Furthermore, the users state that the resolution of conflicts is typically based on their experience which also already includes changing an automatically resolved conflict by the system. The experienced is gained through the daily routine and the daily schedules. Most of the conflicts appear at the same time and with the same trains. This is due to daily delays in peak-commuting times or other periodic events. The users complain that the current system is not able to present more sophisticated resolutions as well as showing the impact of a resolution. Optimized models (Specific-Scenario 2, Section 2.4) output global metrics such as train delays or costs. The operator should be able to see how these global metrics change based on the decision and how the decision impacts the predicted schedule.

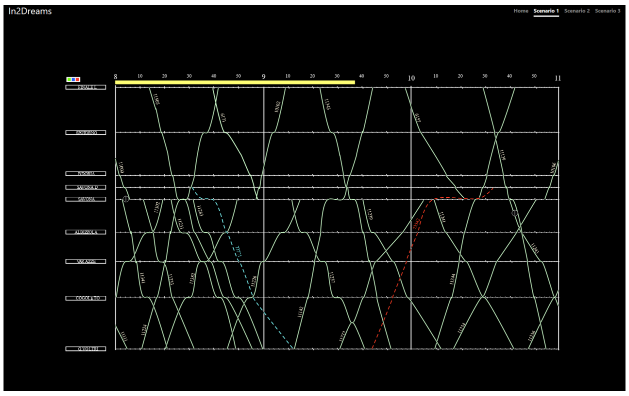
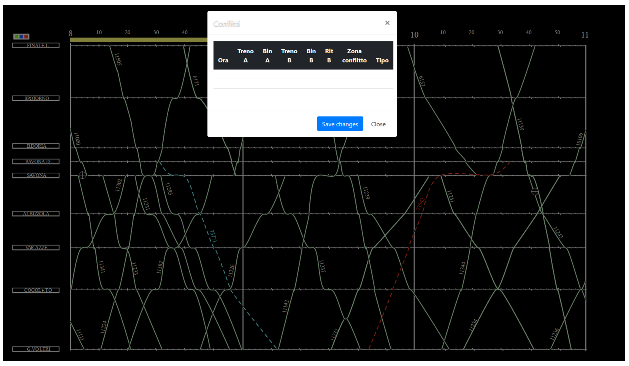



(a) The original photo. (b) The vectorized image.

Figure 5: Cross-Scenario 1, Task 1, Phase 1: The x-axis represents the time. The y-axis shows the different stations. The colored lines represent trains based on a schedule or real-data. Green are person-trains, blue are freight-trains, and red represent single locomotives. Everything right of the yellow bar shows future trains and their schedule according to a rule-based system. The yellow crosses (only left image) represent conflicts.

The vectorized image (Figure5b) copies the information showed in the photo.

2.1.1.2 Phase 2

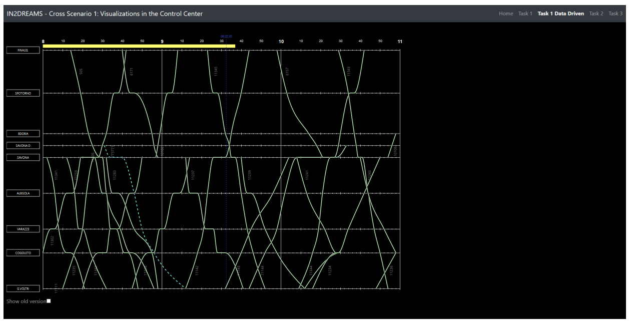
(a) The lines are interpolated and are outlined with a black, thin border to improve their traceability. (b) A new overlay-visualization is prepared to display the global metrics of prediction models.

Figure 6: Cross-Scenario 1, Task 1, Phase 2

Phase 2 introduces an interpolation of the lines with the goal to introduce their traceability (Figure 6a). This is especially important when multiple trains have a temporal overlap within one station. To underline this, the lines are plotted with an additional black, thin border which further shows when one line branches off to the next station. The user can also hover the lines to highlight them whereas the opacity for the rest of the lines is reduced. Figure 6b shows an overlay which prepares the inclusion of the predictions by the models generated in Specific-Scenario 2 (Section 2.4). The current state-of-the-art system merely lets the user pick one train which is allowed to go first. The selection may be pre-selected by the rule-based system however

without an explanation why this automated decision was made. The prepared overlay allows the inclusion of global metrics such as train delays or costs. Furthermore, the user will be able to see how the decision impacts the schedule which is displayed in the background. This is important for the operator as specific changes in the schedule may have a higher impact on the decision than the global metrics.
A video-conference call with RFI was conducted in the end of this phase. RFI agrees to the changes made and that the traceability has been improved. RFI wishes for additional information within this visualization: besides the stations the operators should also know which platform is occupied by a specific train. This allows a better scheduling of the trains.

2.1.1.3 Phase 3


(a) The interpolation was improved to better represent the temporal aspect. An additional reference time-line (blue) adjusts to the user's cursor providing a reference of several trains at the same time and allows their comparison.

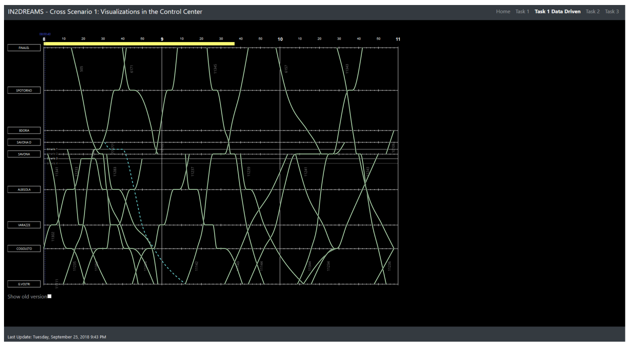

(b) When selecting the stations, additional information is provided by displaying the platforms and the respective trains. The lines are animated allowing to trace this additional information and compare it to the more compact representation.

Figure 7: Cross-Scenario 1, Task 1, Phase 3

In Phase 3 we created a data files that represent the previous train schedule. The data is serialized in a JSON-format. This step allows us to add further interactions such as panning and zooming the visualization. Each of these interactions causes a replotting of the data, this information must therefore be data-driven and cannot origin from a static image. Also, this allows us to simply modify scenarios or representing a different region.
We further improved the interpolation of the lines to improve their accuracy with respect to the temporal-dimension.
In this version, the curves in the interpolation are more narrow and do not cause a major part of the line to be bent.
Finally, in this phase we introduce the interactive plotting of additional information in response to the feed-back gained after Phase 2. Visualizing all platforms at the same time causes the visualization to be cluttered. Therefore, we decide to show this information only when the user interacts with the system. Selecting the station lets the platforms of a station appear. One of the platforms overlays the horizontal line that is shown in the static mode. All platforms are labeled (e.g., binario 1). The interaction further triggers an animation that moves the train-lines onto the respective platform. This animation supports the traceability and allows the user to better remember which train occupies which platform when the selection is dismissed. Unused platforms appear in a darker color to easily let the user spot opportunities for rescheduling trains onto other platforms.

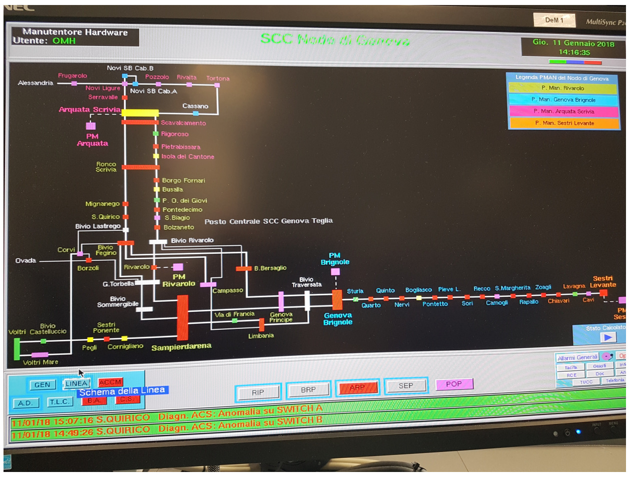
2.1.1.4 Phase 4

Although the main task of the final PoC targets the restoration time, we aim to also include this information in this interactive visualization. The operator will likely receive limited information only necessary to her/him. At least, the information will contain the restoration time with an uncertainty or time window. Further details may be provided on demand.

2.1.2 Task 2 - Alert Management and Prioritization System for AMS

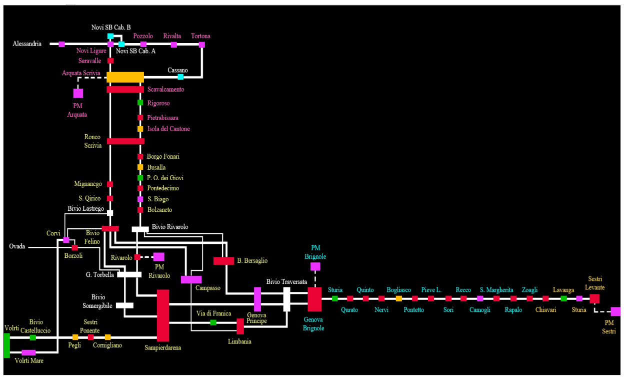
The user of this system, called asset manager, is required to oversee a large region of stations and rail-network, track alarms generated by various systems within these stations and is further required to schedule maintenance and repair teams. We specifically focus on the overview visualization (Figure 4, lower left monitor).

2.1.2.1 Phase 1



11/01/18 15:07:16 S.QUIRICO Diagn. ACS: Anomalia su SWITCH A
11/01/18 14:49:26 S.QUIRICO Diagn. ACS: Anomalia su SWITCH B

(a) The original photo.



(b) The vectorized image.

Figure 8: Cross-Scenario 1, Task 2, Phase 1

A through demonstration and interviews with asset managers were conducted in a visit at RFI's premises in Genoa. The used system was developed about 18 years ago and requires specific hard and software to be operated.
We observed that the system is rather unresponsive as each interaction causes the system to freeze for 1-2 seconds until the system responds. The overview screen is highly distractive and cluttered. The colored blocks represent train stations or logically important points within the rail-network. Their color represents a state that indicates the alert level. The system collects many alarms, more specifically around 15.000 for each day. The alarms belong to a specific asset or system. Logically, these systems can be modeled in a hierarchy, e.g., network; switch; asset; network-adapter of the asset. However, in the current system these hierarchies are not displayed or modeled. Additionally, there is no prioritization of alarms. An alarm of a failing ticket-machine triggers the same alert as a failing railroad switch. A work around is visible in Figure 4. Besides the overview visualization there are three monitors showing lists of alarms. These lists have a filter applied to a specific set of assets or alarms. These lists then have to be observed and when a new alarm appears an immediate reaction is required. The overview-screen, however, is not suitable to distinguish different priorities of systems or alarms. Moreover, each alarm should be acknowledged by the asset manager.

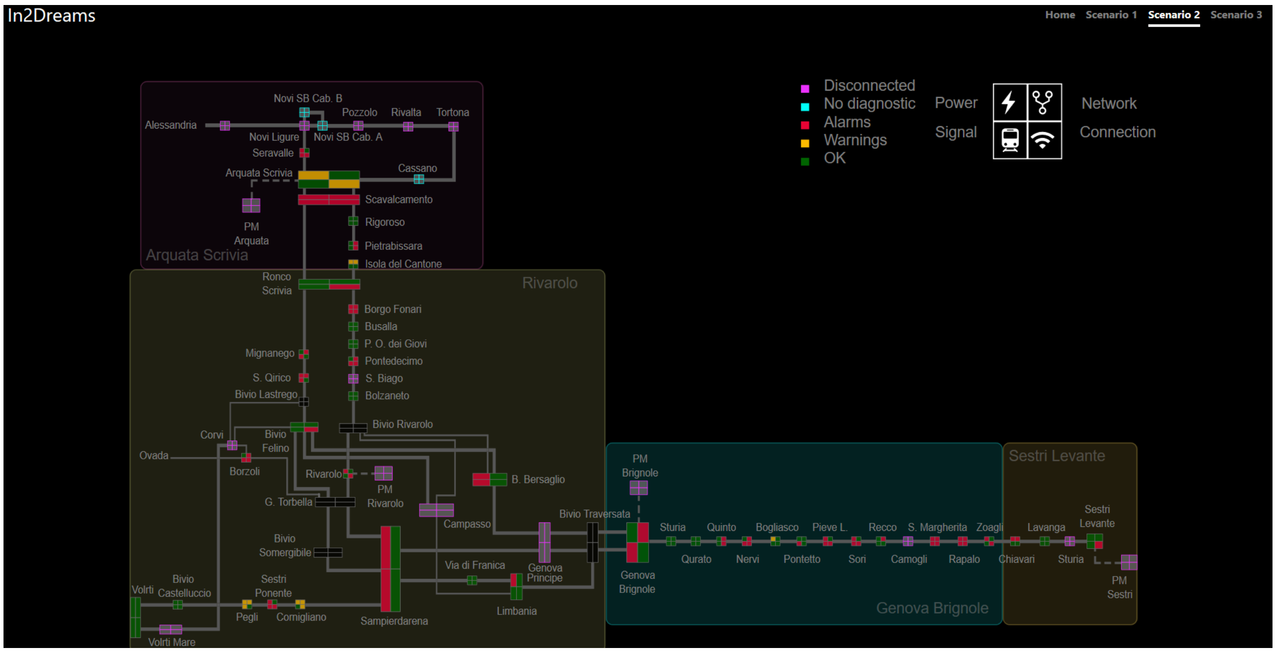


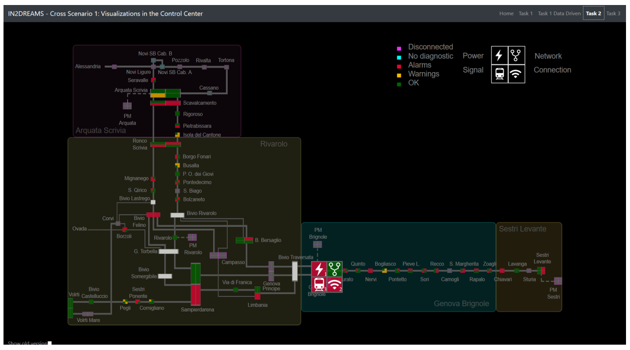
Figure 9: Cross-Scenario 1, Task 2, Phase 2

An unacknowledged alarm triggers a blinking-animation in the system. The system displays a high amount of alarms, mostly for non-critical systems. The additional unresponsive behavior of the system causes the asset managers to not acknowledge incoming alarms which essentially effects a highly blinking, distractive overview.
As in the previous task, the first phase included the transformation of a photo to the vectorized image (Figure 8).

2.1.2.2 Phase 2

Phase 2 already introduces many visible modifications (Figure 9). First and foremost, a reduction of colors and contrast is visible. In the previous version, colors of the station labels are used for regions within the rail-network and the state of the stations. Using similar and bright colors for more than one variable may confuse a user. The reduction of contrast and brightness provides a less tiring visualization. This is important as an asset-manager has to watch and observe this screen in particular. The color of the station labels is replaced by bounding rectangles that contain a very unobtrusive, moderate coloring which is highly different to the colors of the stations.
The same applies to the lines of the rail-network that connects the stations. This information is less important to the asset-manager but should still be available. The user is therefore able to highlight various lines by hovering them.
The previously colored areas representing a train station are now divided into four rectangles. This split provides a semantic categorization of the assets into the four highest and most important categories according to RFI. The placement of these categories is displayed in the legend on the top-right of Figure 9. Each of the tiles can be colored differently. This glyph visualization gives the user access on what kind of system is producing an alarm without the need to interact with the system. Furthermore, if more than one system accross the categories are failing this causes more of the area to be red raising a higher awareness.

2.1.2.3 Phase 3


(a) An extended version of the glyph is shown when the user hovers a station. The number of alarms is added to each tile of the glyph.

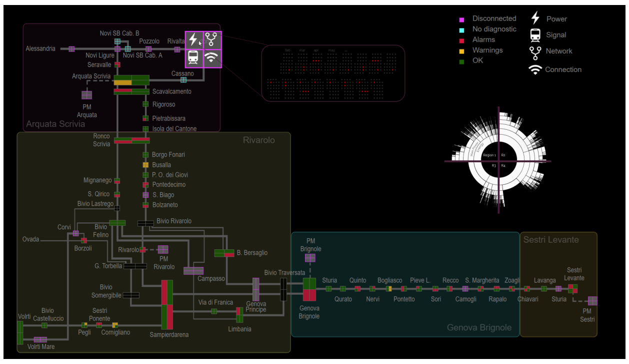

(b) Clicking on a station or an asset triggers a calendar visualization and a sunburst visualization representing the hieararchy.

Figure 10: Cross-Scenario 1, Task 2, Phase 3

The resulting system of Phase 2 provides a richer information space without the necessity to interact with the system and yet the overview seems to be less cluttered. However, this does not allow the user to navigate the hierarchy and explore lower levels. We support this through interaction. The user can hover a station to trigger a small overlay visualization showing the glyph with more details (Figure 10a). The additional information contains the icons as shown in the legend to provide an easy reference. Furthermore, the number of alarms within the respective category is shown in the lower right corner of each tile. The glyph is currently extended such that the user can navigate the hierarchy and reach the underlying levels through a rubber-sheet metaphor [75]. Figure 11 shows how this navigation works. The user may click on the red, upper, right tile (network) and the tiles are laid out again. The other three tiles are shrunk to the side of the glyph while the area of interest is expanded revealing the underlying levels of the hierarchy. Now the user can see that there is one warning (yellow) regarding the WiFi, the database systems as well as the security cameras are working and there are two alarms in for the train-schedule displays.
Clicking on a station reveals a temporal view called calendar vis (Figure 10b). This view aids the user in reasoning about the failure of an asset as she/he might be able to detect temporal patterns when the asset fails. Additionally, this view can also provide information about the restoration time of previous repairs for this asset. Another view is the hierarchical view that provides a similar information as the interactive glyph, however it displays all the information of the whole hierarchy at once. The chosen technique is called sunburst visualization [80].

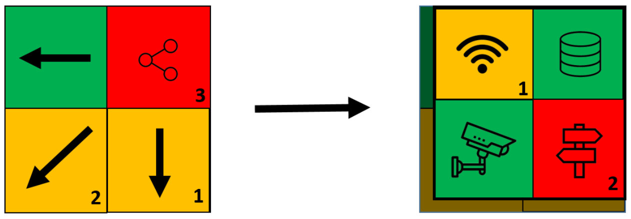


Figure 11: Cross-Scenario 1, Task 2, Phase 3: Rubber-Sheet Navigation

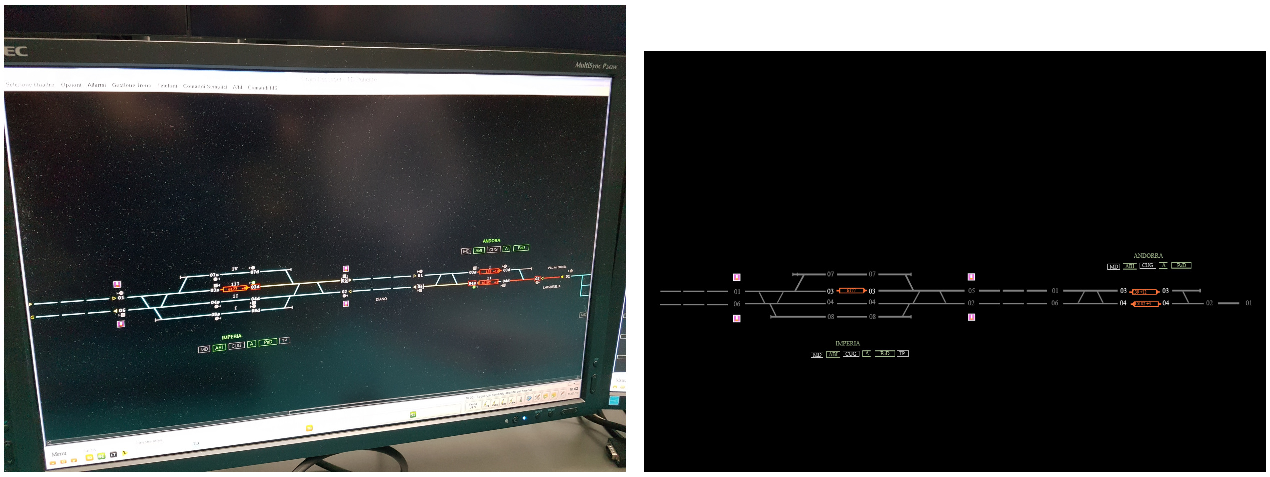
2.1.2.4 Phase 4

This task has the closest relation to the task of the PoC as the restoration time of an asset is of high interest to the asset manager. The restoration time is predicted by a decision tree. While many of the input parameters are provided, the user may wish to modify some of these parameters interactively as unforseen events happened and the provided parameters are obsolete. For example the Actual Beginning Time (see Table 12) of the reparation of the asset may be delayed and needs to be adjusted. There might be cases where the next possible setting is not optimal regarding the restoration time, therefore the user must be empowered to interactively find the best parameter settings.
Besides the input and output, the model itself can be visualized supporting the user in understanding the model better. This is a trust-building measure and helps the user to understand the predictions of a model better.
The foundations are already created in Phase 3. The historical views help the user understand how previous repairs of a specific asset were conducted and what problems may have occurred. In combination with prediction models the user can better estimate the restoration time for an asset.

2.1.3 Task 3 - Improving the TMS and Directing the Awareness of the Operator

This task focuses on the status screens as shown in Figure 3 (upper screens). The purpose of this system is to inform the operator of the current state of the rail-network and especially the trains running in the area the operator oversees. In contrast to Task 1, this system does not provide any historical or future lookahead regarding the train schedule. It therefore focuses on the spatial orientation, accuracy, and ease.

2.1.3.1 Phase 1



(a) The original photo. **(b) The vectorized image.**

Figure 12: Cross-Scenario 1, Task 3, Phase 1

As visible in Figure 3 this system makes heavy use of a multi-screen environment. The operators we interviewed with explained that through training and experience they can quickly find and identify trains at a specific location. However, the position on the screens does often not match the real-world position. Distances in between stations are heavily distorted in order to fit even larger areas onto several monitors. Furthermore, a vertical expansion is not possible due to the layout of the monitors. This may result in heavy distortions which may also include rotations of tracks such that a north-south direction in the real-world may be plotted

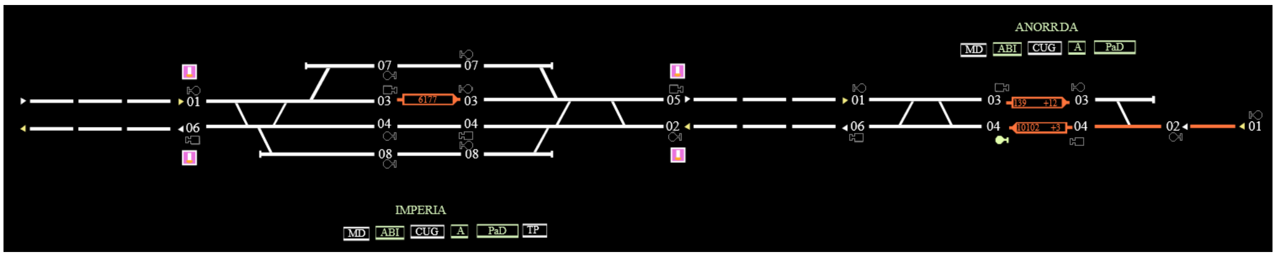


Figure 13: Cross-Scenario 1, Task 3 Phase 2: Reduction of information

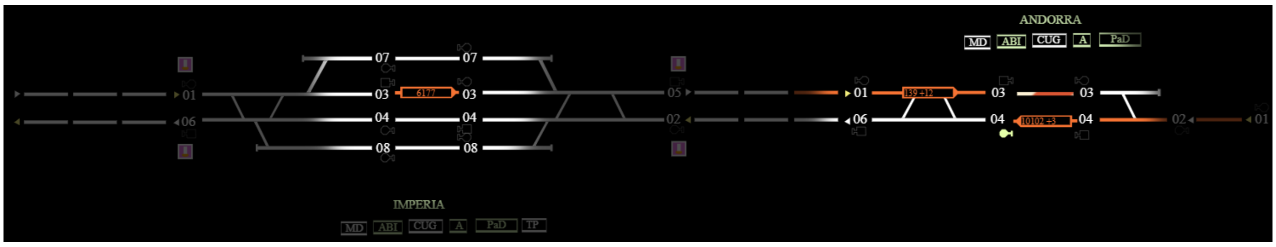
horizontally. Similar layout algorithms are for example used to create metro-maps for cities [87]. Besides the trains itself the system also visualizes occupied segments of trains. This is important as the TMS will not allow to route a train on a currently occupied rail-segment or if this segment is already claimed by another train. When a train is about to occupy a segment, this segment starts blinking in orange. When the segment is occupied the color of the segment changes to orange. This blinking animation is, however, often hardly visible as the operator cannot oversee all of the area at once. Furthermore, due to singular usage of the colors white and orange this may not be perceived. Operators also mentioned that the visualization propagates a lot of information and seems thus cluttered. Especially, trainees often have difficulties orienting themselves within this heavily distorted map. Through observation and interviews we learned that the operators are mostly working with the systems covered in Task 1. A typical use case to use this system is to verify the current situation when a conflict appears (Task 1). Then the operators may check the area where the conflict appears to make a more informed decision on which train can go first. As this system only shows the current situation, conflicts which typically appear and are resolved ahead of time, are not shown. The visualizations of Task 1 and 3 are not visually connected so the user-task "finding the position of the conflict on the map" is difficult for less experienced operators. The operators further mentioned that many transitions between the systems for a longer period are tiring as, besides the cognitive performance, it is also physically challenging because it often requires to move the head or body.
RFI demands solutions to reduce the number of monitors such that the same area is covered in less space. This is extremely challenging because it requires more distortions which increases the complexity of spatial orientation. Moreover, this imposes that overall the size of the visualization is reduced making it harder reading, for example, the train numbers that are being displayed. This also means that even though less physical movement is necessary it possibly may not improve the situation as the more dense plotting in combination with the rather distant screens is exhausting for the eyes and our perception.

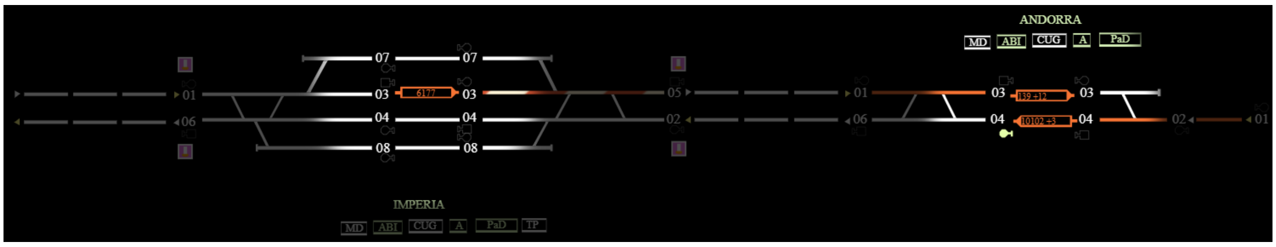
2.1.3.2 Phase 2

As a preliminary measure we modified the symbols of the signals in the visualization because we learned that in the daily business of the operators the signals are mostly out of interest and only in specific cases their state needs to be checked. The shape of the symbols differs describing different signal-systems. We removed the filling of the symbols only leaving the outline and plot this outline in a dark, grey color which makes them unobtrusive. The feedback from RFI emphasizes the effect of less occlusion due to the reduction of symbols. RFI points out that a linking between the visualization of Task 1 and 3 is necessary to ease the workflow of the operator.

2.1.3.3 Phase 3

Phase 3 introduces four extensions. The datafication allows us to run a stateful system where trains can be moved. Figure 14 shows three states where two trains are progressing in the network.


(a) Train 139 is occupying the next segment.


(b) Train 139 is progressing to the next segment. Train 6117 is occupying the next segment.

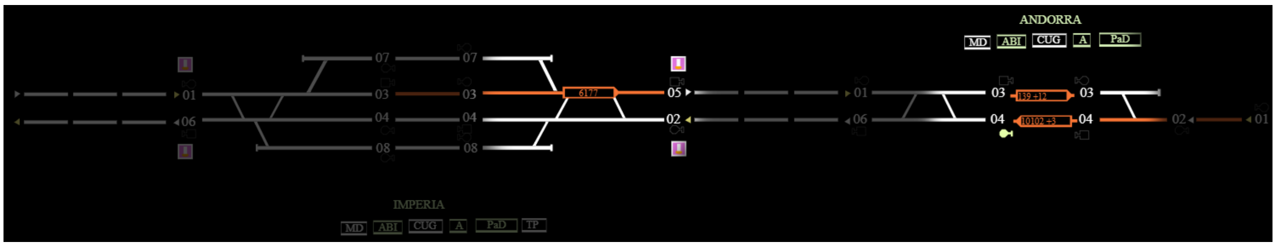

(c) Train 6177 is progressing to the next segment.

Figure 14: Cross-Scenario 1, Task 3, Phase 3: A stateful system.

The difference that is most apparent is the circular highlighting around the trains. This highlighting allows the user to spot the areas of interest much faster while the remaining network and other details are still visible and thus, no context is lost. The highlighting is currently only applied on the trains but can be extended to any part of the visualization and vary in size. Future systems may also change the highlighting depending on their current state or provide the opportunity to the user to change the areas of interest.
The state-of-the-art system uses flashing cue when a new railway-segment is claimed for a train. Blinking however, attracts the user's attention very much, especially in form of a bright and saturated color [38]. This is a good cue for exceptional cases such as when alerts or warnings are triggered. However, in a frequently occurring scenario such an overused cue may distract the user, cause stress, and fatigue. We therefore changed this cue to a white-orange-striped overlay which moderately moves. While this animation still attracts the user's awareness it is less alerting to the user.

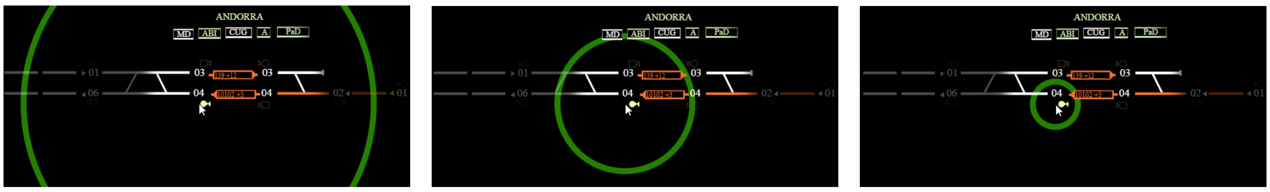


Figure 15: Cross-Scenario 1, Task 3, Phase 3: Attract the user's attention.

The fourth extension is shown in Figure 15. The enhancement is designed for exceptional cases when the user's awareness is required at a certain location on the screen and at a specific time. The animation draws a large circle which shrinks down to a specific location on the screen. We call this "inverted ping". Even on multiple screens the eyes follow the circle's boundaries automatically towards the point of interest. As mentioned before, this feature should not be used too frequently. For example, Figure 14 shows purple icons. The meaning of these icons is the temperature of the railway-segment. If the temperature reaches a threshold the icon is shown alerting the user that trains may have to pass this segment slower or less frequently. In a complex visual environment the simple placement of the icon may happen unrecognized by the user. Additionally, the "inverted ping" animation could be used to steer the user to this segment. This should however only happen when the icon is added or removed and not regularly when the icon is displayed. The end-users welcome the aforementioned extensions as they visibly improve the steering of the users awareness. The system was presented on a multi-screen environment. Again, RFI highlighted the importance on a linked environment that improves the relations of the systems of Task 1 and 3.

2.1.3.4 Phase 4

Similar to the other tasks, Phase 4 focuses on the linking and brushing to the other tasks and the inclusion and extension to the PoC (D5.4). The linking and brushing will be mostly implemented in combination with Task 1 as both systems are used the operator's workstation in the TMS. As also the operator should be aware of the restoration time, especially when it concerns the trains and their routing. On the other hand, the operator of the TMS does not need excessive information as the asset manager would need. The focus will lie on compact visualizations that are easily interpretable and require less interaction.

2.1.4 Conclusion

For this scenario we chose three systems used by the operators in a daily fashion. We optimize these visualization systems to present the same or more information without cluttering the visualizations. We introduce interactions to let the user more conveniently request additional information. The development is conducted with regular feedback-sessions in collaboration with the end-users (RFI). This iterative development ensures that the users are not overwhelmed with too many new changes and that the incremental changes can be evaluated separately and be refined in the next phase(s). The current state of the progress prepared the inclusion of specific prediction models that are developed and described in the following scenarios. The final inclusion will be shown in the proof of concept (D5.4).

2.2 Cross-Scenario 2: Marketplace of Data and Data Monetization

Because of absence of actual data, since the Marketplace of Data does not yet exist and will be not developed in WP4, we decided to not to develop further the Cross-Scenario 2.

2.3 Specific-Scenario 1: Track Circuits

2.3.1 Introduction

As described in D5.1 the main objective of this scenario is to develop data-driven models for online status identification and behaviour modelling of the Track Circuit System (TCS), in order to enable diagnostics and prognostics functionalities. In this context, the exploitation of gray-box models and the development of metrics and KPIs able to assess models performance (i.e. understanding under what conditions they can be

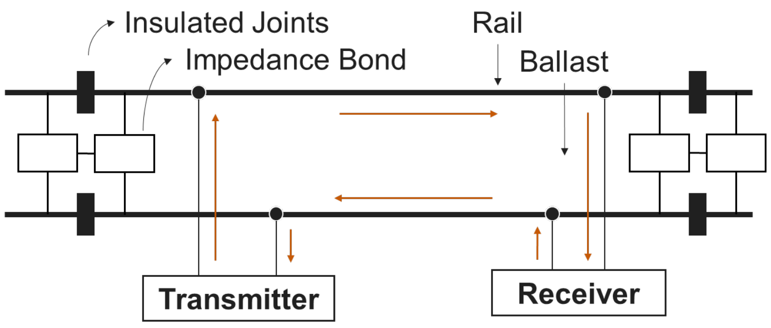


Figure 16: Current flow in the track circuit.

effectively be applied in real operations) represent a core task in this scenario. Moreover, the work done inside this scenario, together with the maintenance operators, revealed the possibility to develop and study some interesting features more related to Visual Analytics (VA), which is a core topic of IN2DREAMS WP5. VA processes are designed in close cooperation with domain experts in order to create semi-automatic work-flows, which allow to leave the operators in full control, while supporting them with context-sensitive and automatically extracted information. In the context of TCS maintenance, the application of a VA approach could support the maintenance process, improving decision making before and during maintenance operations, speeding up and renewing condition-base and preventive maintenance processes. This possibility has arisen from the practical needs of the maintenance operators which emerged during the course of work. On the other side, the development of the data-driven model for anomaly detection has been slowed by some technical issues related to the frequency of data acquisition (this will be discussed later in Section 2.3.4). Thus, the work on the model will be completed in a later stage of the project and reported in the next deliverable once the technical issues will be solved and some evidence will be achieved.
Based on what have been stated in D5.1, the deliverable is organized as follows. Section 2.3.2 contains a detailed description of the system involved and the identified problem. The first processing step, which enable visual analytics functionalities, is described in Section 2.3.3, while in Section 2.3.4 the work related to the anomaly detection model development is presented. Finally, results achieved in all the activities described above are contained in Section 2.3.5.

2.3.2 Problem and System Description

The TCS is a device used to detect the presence (or the absence) of a train on a rail track. Moreover, it allows the transmission of digital cab signalling data, in the form of binary information, with the purpose of enabling automatic train protection functions. The TCS is composed by an electronic board, which communicate with the Computer-Based Interlocking system (CBI) enabling trains management functions, and a physical component which includes the electrical network and other equipment like transmission cables or directional relays.
In order to enable train detection functions, each section of the rail track is electrically separated from the other sections and a different TCS is associated to each section. The electrical network is mainly composed by a transmitter which transmits a current signal through the two rails to a receiver, where the signal is measured (Figure 16). Moreover, current leakage between adjacent sections through the railway track is prevented by insulated joints, while impedance bonds allow direct current to flow but they block inside the section alternating current (which is used for the detection functions). With this setup, if the track is free (no train present on it) the current signal flow through the rails to the end of the section. If a train occupies the track section, the wheels give rise to a short-circuit which prevents the signal to reach the receiver (Figure 17).

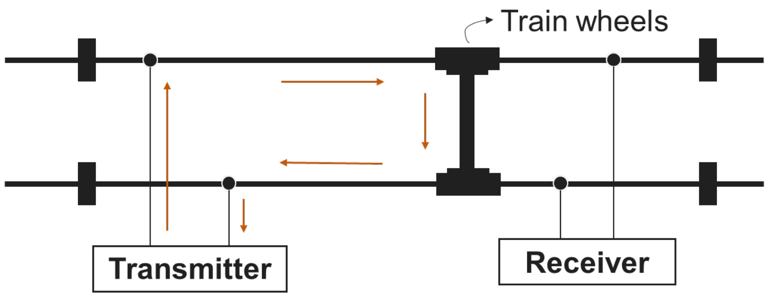


Figure 17: Current flow in the track circuit when a train is passing on the track section.

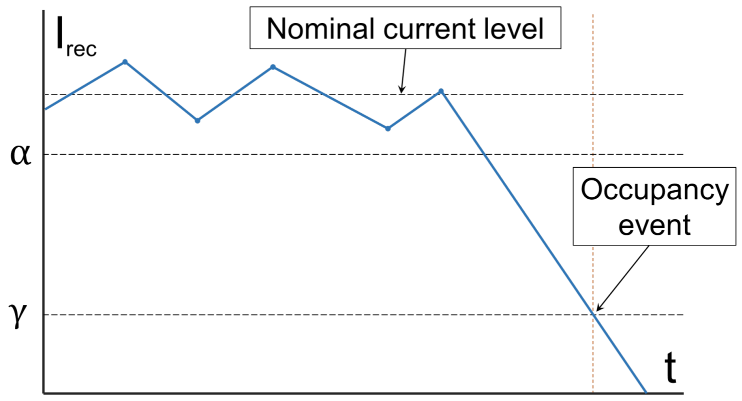


Figure 18: Train detection behaviour of a track circuit: when the current signal I_{rec} goes below the threshold γ, the track is reported as occupied.

2.3.2.1 Train Detection

Track occupancy is observed from TCS through the measure of the current signal at the receiver. Considering a healthy TCS in normal conditions, signal level lies in a defined range of values (above α threshold, as depicted in Figure 18) while it sharply decreases to a nil value when a train enters in the track section. The system identifies a track as occupied if the signal level falls below the threshold γ. Thus, TCS are tuned such that, even in the case of small current deviations, the presence and absence of a train are correctly reported.

2.3.2.2 Track Circuit Failures

Track circuits have been designed to be robust to many different problems, and their performances are durable over time. Despite this, though the analysis of historical data, it is possible to demonstrate that some degradation effects, undesired behaviours or physical impairment exist and unexpected failures may occur. For example, some defects affecting a rail can cause an increase in the rail resistance resulting in a current signal at the receiver too low. This kind of behaviour can affect the right functioning of the train detection system causing a functional failure. The TCS have to meet two different type of requirements [40]:

- *Safety requirement*: the track section have to be reported as occupied when a train is present on it. Failures related to this requirement are referred as False Negative (*FN*);
- *Operational requirement*: the track section have to be reported as free when a train is not present on it. Failures related to this requirement are referred as False Positive (*FP*) or False Occupancy (*FO*).

While the safety requirement is guaranteed by the CBI system itself, operational requirement is less restrictive and *FO* failures occur quite often. Since *FO* consequences could include traffic disruption and penalties this type of failures represents the main issue investigated in this work. *FO* failures are mainly related to TCS faulty behaviour which results in current level drop. In that condition, even if some degree of fluctuation is acceptable (thanks to the tolerance of the receiver threshold system) when the current level becomes too low, the section will be reported as occupied, even if there is actually no train present.

2.3.2.3 Faults affecting the TCS

Before going in depth with the specific system involved and the analysis conducted in this work, a description and categorization of the main faults affecting the track circuit system is required. As proposed in [40], the following categories of faults can be identified:

- **Insulation imperfection.** To prevent the AC current signal of one section from spreading to adjacent sections energizing their relays, insulated joints are used. Problems occur when insulated joints degrade during time or when conductive objects lie over them. In the case of an insulation problem, the circuit leaks current and, thus, the current signal is too low;
- **Rail conductance impairment.** The proper functioning of a TCS is based on the conductance properties of the rails and if the Impedance of the rail is characterized by a too high resistance the current level at the receiver decreases. This anomalous behaviour could be caused by a damaged or degraded rail, by an insufficient quality of the bonds in jointed track or by disturbance currents along the rail;
- **Ballast condition.** The condition of the ballast determines the resistance that currents encounter when flowing from one rail to the other rail or to the ground. Because the effect of a decreasing ballast resistance is similar to that of a train shunt, it is important that the ballast resistance is sufficiently high and constant. Due to environmental disturbances (mainly related to weather conditions) and aging, the ballast resistance will fluctuate over time. This behaviour could again lead to a current drop causing *FO* failures;
- **Train shunt imperfection.** The proper functioning of a TCS requires that every train short-circuits the section in order to guarantee train detection functionalities. A good train shunt can be inhibited by different causes of which the two most important ones are: contamination between the rail surface and the wheels and lightweight trains. Even if this kind of faults is related to *FN* failures (the system is unable to comply with safety requirement) they are reported here for completeness;
- **Circuit-related and other faults.** Even if track circuits have a high reliability, their components (e.g., relays, cables, and power supply) can break. Moreover, some malfunctions could affect the electronic board of the TCS (i.e. Critical Error in the board software). With the exception of cables and similar equipment degradation, these faults are abrupt and no parameters are available in order to monitor their source (i.e. relays or electronic boards). For this reason, these kind of faults are not treated further.

In [40], a categorization of these faults is given in relation with their evolution during time and their spatial dependencies. For what concerns the evolution in time, four types of behaviour exist: abrupt (*A*), linear (*L*), exponential (*E*), intermittent (*I*). On the other side, the following spatial dependencies are presented: no correlation with other sections (D_1), correlation with sections on the same track (D_2), train-specific correlation (D_3), correlation with all nearby sections (D_4). Results of the study are summarized in Table 1, where we kept only faults related to *FO* and we have added the circuit equipment degradation fault.

Table 1: TCS Faults types and features.

Fault (F)	Problem	Cause	Signal level	Time Trend	Spatial Dependency
1	-	Healty	ok	-	-
2	Insulation imperfection	Insulated joint defect	*low*	*L/E*	D_1
3		Conductive objects	*low*	*A*	D_1
4	Rail conductance impairment	Mechanical defect	*low*	*E*	D_1
5		Electrical disturbances	*low*	*I/A*	D_2
6	Ballast condition	Ballast degradation	*low*	*L/E*	D_1-D_2
7		Ballast variation	*low/ok/high*	*A/L/E/I*	D_4
8	Circuit-related faults	Circuit equipment degradation	*Unknown*	*Unknown*	*Unknown*

2.3.2.4 Observed Parameters

In the system analyzed in this scenario, the following parameters are available for the monitoring of the track circuits behaviour:

- *Received Signal Level* (R_1). This parameter is important because it verifies that the received signal level is within the operating range of the receiver circuit. It is acquired as the ratio between the value of the signal and the maximum range of the receiver in percentage;
- *Shunt Level* (S). As with the Received Signal Level this value indicates the track signal level (but with a higher resolution). When the track circuit is initialized or calibrated a value of 162% is assigned to the actual received signal level. This is the main parameter;
- *Variance* (V). This parameter can be used to determine whether the coupling unit is tuned correctly or a coupling unit has failed. Variance can also indicate whether interference from another source is becoming a problem in the reliable operation of the track circuit;
- *Raw Signal Level* (R_2). The Raw Signal Level is the total signal received by the TCS (it includes noise and adjacent track signals). As for R_1 is expressed in percentage with reference to the maximum range of the receiver.

2.3.2.5 TCS Redundancy

In order to meet the safety requirement TCS is composed by two different electronic boards, Primary (*P*) and Backup (*B*). With this setup, when the primary board is active and fails, the system automatically transfers all the safety and operational functionalities to the Backup board and vice-versa. Moreover, in order to guarantee the proper functioning of both P and B boards, they are interchanged periodically during scheduled maintenance actions and/or after the recalibration process (which will be discussed in the following section).

2.3.2.6 TCS Calibration

When the track circuit is initially adjusted with high ballast resistance, the threshold level of the track circuit's receiver is equal to nominal current of the track circuit ($\alpha > \gamma$). In this condition, the threshold selected represents an overdrive value of 100% (or 1). With only an overdrive of 100% (or 1), the track circuit would become occupied if the ballast resistance changes to any value lower than the one assumed at the initial adjustment value and also if the current signal presents some small variations. In order to enable it to operate over a changing ballast value, the overdrive has to be added to the track circuit and it is obtained by changing the threshold of the receiver. The magnitude of overdrive is selected as a function of different factors such as frequency, shunting sensitivity, minimum ballast and rail-to-rail voltage. An example is depicted in Figure 19, where an overdrive value of 1.62 (or 162%) is obtained by adjusting the receiver threshold to 62 mA (in this situation the threshold α is represented by a Shunt Level value of 150%).

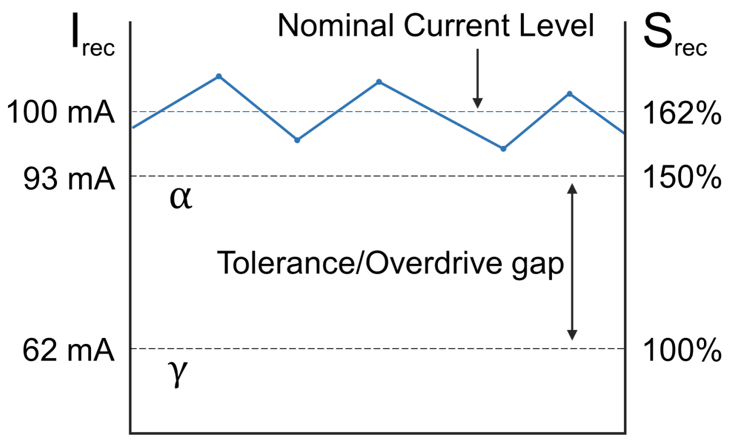


Figure 19: Comparison between current signal I_{rec} and Shunt Level S_{rec}: here an overdrive value of 1.62 (or 162%) is obtained by adjusting the receiver threshold to 62 mA (in this situation the threshold α is represented by a Shunt Level value of 150%).

Once the track circuit is adjusted with high ballast resistance, the received signal strength can change as the ballast changes. Thus, from the latter condition, if the track circuit is recalibrated, the threshold of the receiver will be changed. For example, if the ballast value is low and the Shunt Level value is different from its original value, performing recalibration will adjust the threshold of the receiver to a new value: the recalibration process allows the operators to change the threshold γ depending of the signal level and the ballast condition in order to keep the system reliable. The downside of the process is that the Shunt Level value is reset to its original value and this can hide ballast degradation and aging effects (low current signal at the receiver). Recalibration is the main action performed during the preventive maintenance routine.

2.3.2.7 TCS Monitoring and Corrective Maintenance Actions

During their operations, track circuits are monitored in two different ways:

- TCS specific information and parameters' value could be monitored from a specific panel which is accessed only when a maintenance action has to be performed (not for preventive actions);
- Alarms, coming from the central automatic control system, are monitored in real-time. These alarms include the ones related to TCS (i.e. electronic board failure or *FO* alarms). More details about data coming from the central automatic control system are contained in D5.1.

Main corrective maintenance actions which are performed on TCS (*FO* failures and causes taken into account for this study):

- Recalibration, which is used to guarantee that TCS works properly, mainly to face Faults of type 6 and 7, but it does not resolve the cause of the problem;
- Insulated Joints cleaning, which solves Faults of type 2 or 3.

Other specific actions are performed when problems cannot be solved with a standard approach or for problems which are very rare (i.e. cables and equipment replacements). On the other side, main corrective maintenance actions which are performed on TCS to solve *FO* failures for circuit-related and other faults, are:

- Relay replacement;
- Electronic board reset or replacement;
- Power Supply replacement.

As stated before failures related to these actions are outside of the scope of this work.

2.3.3 Data Processing and Visual Analytics

Definition of the problem, identification and understanding of the system behaviour, data exploration and data acquisition process have been conducted in close collaboration with the maintenance operators working on the line selected for this study. This collaboration highlighted the possibility of studying an analysis process and a data representation in order to support, through visual analytics, maintenance operators with context-sensitive and automatically extracted information. In this context, the following requirements have been identified from the operators:

- Visualization of TCS parameters trend. At the moment the system allows to monitor a single track circuit, without a visual representation and without the possibility of accessing historical data. Thus, a visual representation of historical data is required in order to make more reasoned considerations on track circuits health and plan more efficient maintenance strategies. Moreover, the real behaviour of the TCS should be visualized: observations associated with an occupancy should not be considered and data coming from *P* and *B* boards should be merged to identify observations which belongs to the active board (at the moment the system does not provide information about which of the two boards is working);
- Alarms and Events reporting. The system does not provide statistical summary or reporting of system failures, alarms and main events. Alarms and events data are available from the central automatic control system and used only for real-time monitoring. A high-level knowledge of the status and past history of the system would be extremely useful to support operators in their decisions in the maintenance workflow.

Thus, the analysis conducted in this work led to a two-fold contribution: on one side, it allowed to meet that requirements identified with the maintenance operators, on the other, it represents the main step, for what concerns data understanding and preprocessing, for the development of an interpretable data-driven model for the identification of faulty track-circuits. In the following sections the data process is described with results related to visual analytics task identified.

2.3.3.1 TCS logs data processing

The aim of TCS data processing is to obtain a dataset with all the track circuits related parameters (Shunt Level, Variance, Receive Level, Raw Signal Level) representative of the TCS behaviour. The analysis focuses on Shunt Level (Figure 20) because it is both the most relevant parameter and the one that can be treated more easily in order to clean data. Two main issues have to be solved:

- Observations relative to Primary and Backup boards are acquired separately (Figure 21 and 22) and a single pattern has to be extracted during time from the board which is active;
- As depicted in Figure 20, raw data contains train passage observations (i.e. observations with a value equal to zero or with a value sampled during a drop to zero) and this fact makes difficult both the visualization and the analysis of the behaviour in normal condition.

The final data format is represented in Table 2.

2.3.3.2 Central Control System logs data processing

A typical log record is composed of six main parts as showed in Table 3. In case of alarms, an additional file in the log record indicates the alarm code (WSTRCODE) and it is to classify it properly. This field allows to filter the logs based on a certain subset of alarm/events codes of interest. From these log files, a cleaning

Table 2: TCS logs data variables after the processing step.

Variable Name	Description
Timestamp	Date and time of the observation
Station	Reference station (TCS board rack location)
Track ID	TCS unique identufier
Primary/Backup	Reports to which board (primary or backup) the observation is referred
Shunt Level	TCS status parameter
Variance	TCS status parameter
Raw Signal Level	TCS status parameter
Receive Level	TCS status parameter
Direction	Direction in which the TCS is set
Active	True if the board is currently working, else False
Occupied	True if the track is occupied by a train, else False

Table 3: Central Control System logs data variables after the processing step.

Variable Name	Description
FMT	Summary of all the information contained in the log
TS	Timestamp related to the event/alarm reported
CP	Location of the event (i.e. Station)
CX	More detailed location of the event (i.e. Station + Track ID)
STY	Log type (i.e. event or alarm)
EQ	Event unique code

and extraction process is applied in order to obtain the two main sources of information in which we are interested: the occupancy events and alarms related to the TCS. The latter group of extracted information contains only a subset of relevant alarms which has been selected in collaboration with the operators.

2.3.3.3 Active Boards

In order to find which board is active, considering a single observation, the concept is to look at n previous observations of Shunt Level and count the number of zero values: If the number of zero values is greater than a value m (for example 3) the board is reported as Inactive, otherwise it is reported as Active (the value m depends on the frequency of acquisition). Assuming we have an observation each minute, if 3 or less observations are found it is due to an occupancy (less than 3 minutes of occupancy) but if more than 3 observations are found it is supposed to be due to an inactive board (an occupancy cannot last for more than 4 minutes, neither for track circuits lying in the stations).

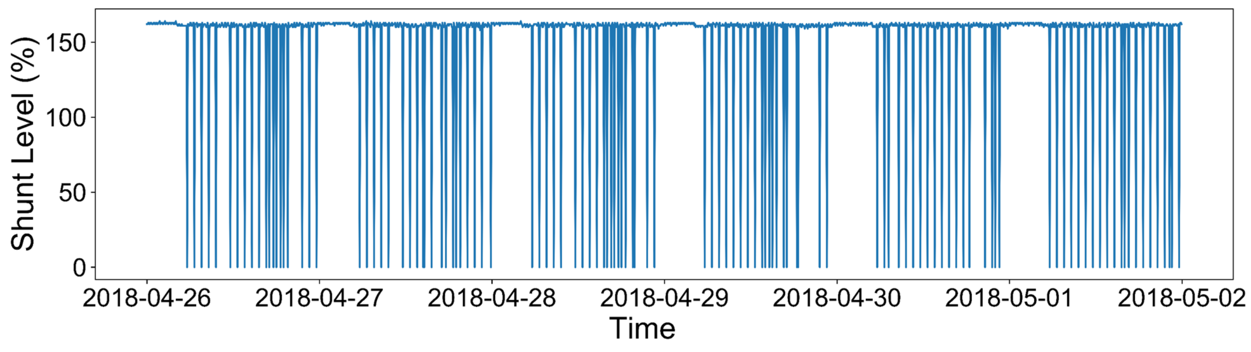


Figure 20: Shunt Level values for a specific track circuit (raw data).

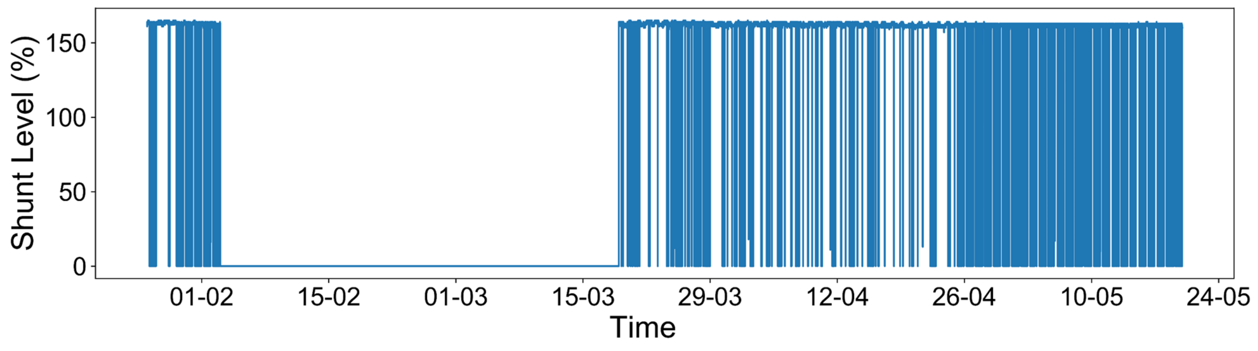


Figure 21: Shunt Level values of a specific track circuit observed from the Primary board.

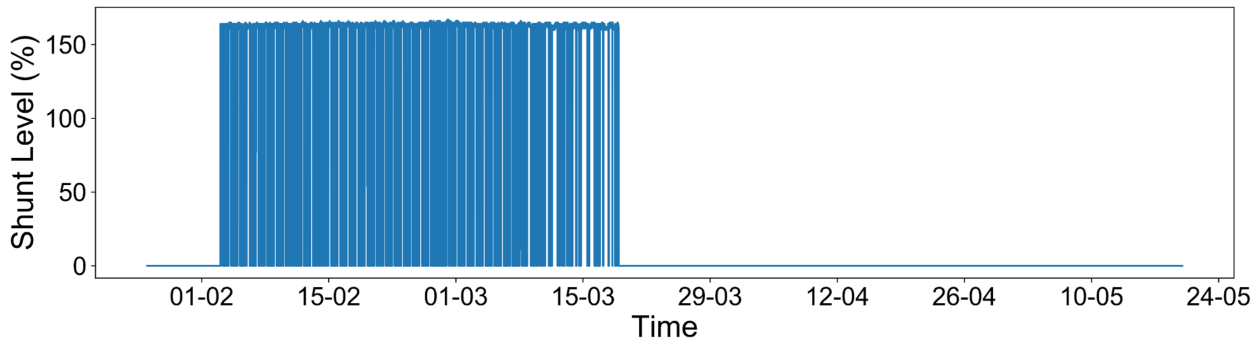


Figure 22: Shunt Level values of the same track circuit of Figure 21 observed from the Backup board.

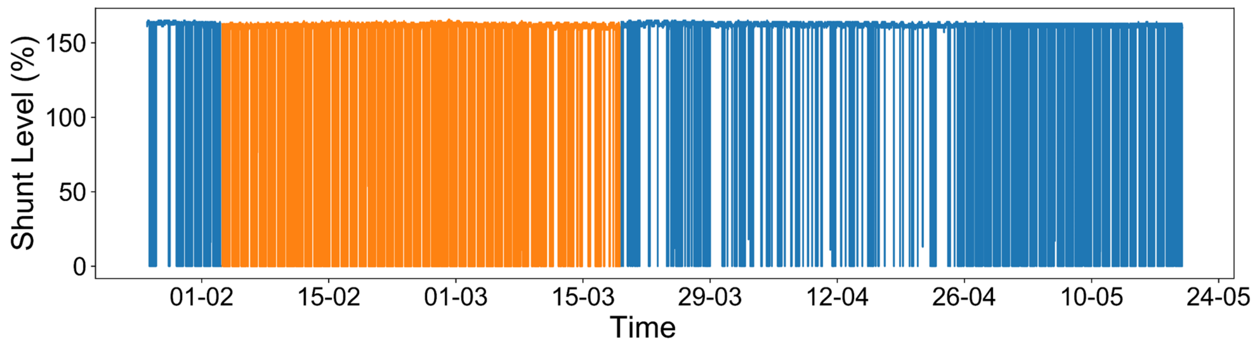


Figure 23: Shunt Level observations from a specific track circuit relative to the board which is active: Primary (P) in blue, Backup (B) in orange.

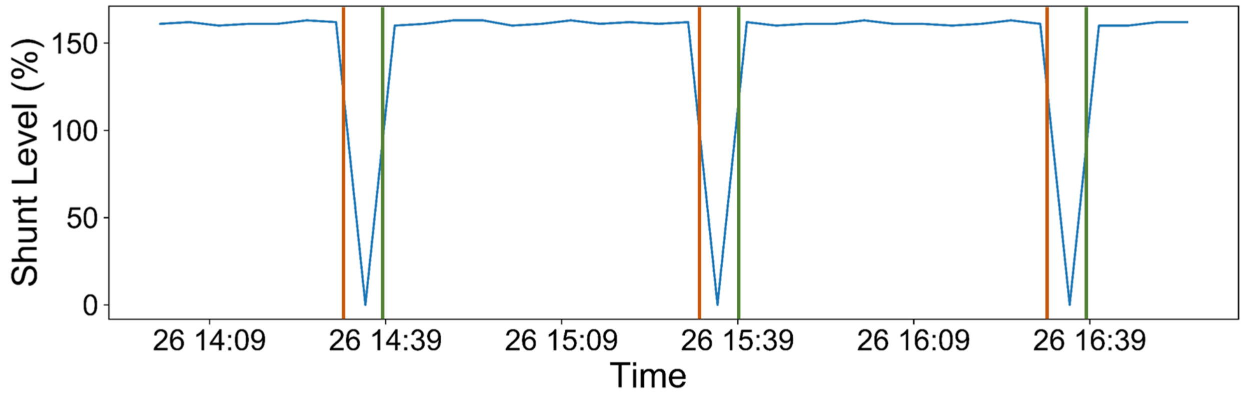


Figure 24: Shunt Level observations from a specific track circuit relative to the board which is active and occupancy events for the same time window: occupancy-on events are represented in red while occupancy-off events in green.

2.3.3.4 Occupancy Events

In order to identify the normal behaviour of a TCS, using a simple filter with a threshold (e.g. selecting observation with 100%) is not enough to achieve the desired precision. For example, if an occupancy occurs immediately after an observation the Shunt Level value could be recorded while is going down to zero from its normal value. In this condition, considering a normal value of 160%, a value of 120% could be identified as normal while it is due to the signal drop caused by a train approaching. To achieve a reliable estimate of real occupancies, Shunt Level observations have to be correlated to occupancy events (*occupancy on* and *occupancy off* events) from Central Control System logs, as depicted in Figure 24. Thus, for each Shunt Level observation X, we look in the *occupancy events* dataset for occurrences where *TS* (*occupancy on*) is greater than *Timestamp* $-\delta$ and *TS* (*occupancy off*) is less than *Timestamp* $+\delta$ (with reference to Table 2 and 3). Thus, if one or more occurrences exist the track is reported as occupied and the observation will not be taken into account for the analysis of the track circuit behaviour.
The parameter δ has to be tuned considering different factors, such as occupancies duration and frequency of acquisition (at the moment a value of 10 seconds has been tested). To note that the above process represents the simplest technique to find real occupancies but it does not take into account all the aspect of the problem and thus can improved.

2.3.4 Anomaly Detection Approach

The final goal of this use case is the development of interpretable and gray-box models of the track circuits behaviour in order to identify faulty assets (i.e. assets which suffer from a fault and may therefore be subject to *FO* failure). These models will support maintenance operators with additional and useful information in order to take decision about maintenance planning, starting a shift toward predictive maintenance. Moreover, we are also interested in finding metrics and KPIs able to understand if our models can be effectively used in real operations or if the developed models are not reliable enough. Once these KPIs will be available, they could be exploited for the evaluation of the quality of the predictive models developed inside IN2SMART WP8, in which this scenario is analysed with slightly different objectives.

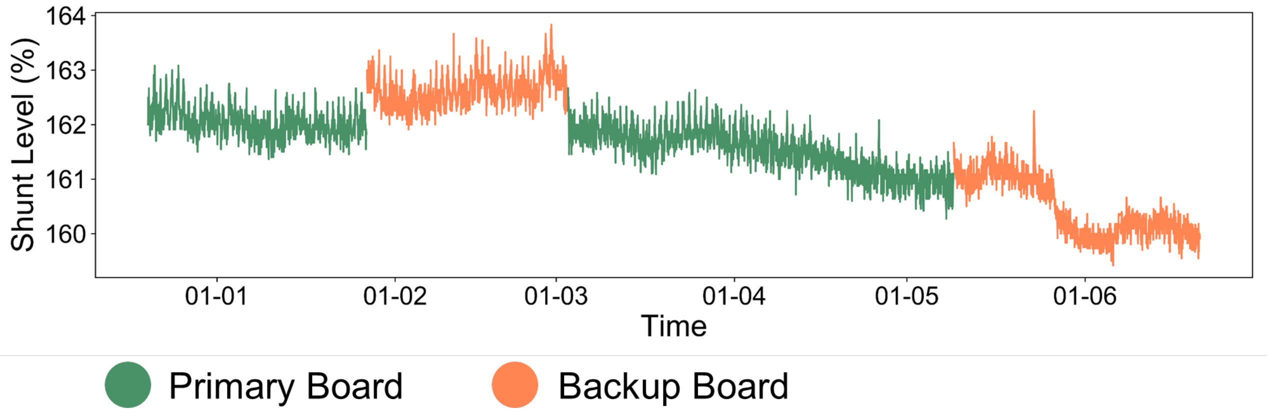


Figure 25: Example of a possible representation of the Shunt Level trend over time after the data processing step: color information is added to identify which board is active in a specific moment.

For what concerns the anomaly detection models, two possible approaches could be identified depending on the time window used for the analysis. The first approach, which is the most promising, is based on the analysis of the TCS behaviour in the neighbourhood of the moments in which a train is about to enter, or leave, the track section (short time window). The second approach consists in analysing the long term behaviour of the TCS, trying to identify unusual patterns in a larger time window. While the first approach could identify different type of faults, also intermittent and abrupt ones, the second one is focused to faults characterized by a slow degradation in time, which could be linear or exponential.
Data quality and in depth data exploration activity, in order to identify the best and more effective approach to the problem, have been performed up to now. Considering the first approach, on which we mainly focused, the main issue we faced is related to the data frequency (one observation every five minute) which it has been verified is not high enough to understand TCS behaviour inside the occupancy event time window (the occupancy could last less than ten second). In this context, we are working with operators on the field in order to develop a solution which should allow an update in the TCS logs acquisition frequency (e.g. one observation every seconds): this could widely change possible results achievable in this scenario. Finally, we started to work with the second approach and results will be presented in detailed in the next deliverable.

2.3.5 Results and Conclusions

Due to the complexity of the system a great effort has been made in order to identify and understand the problem, with the help of the operators, to acquire a detailed knowledge of the assets behaviour (fault and failure) and their interaction with operators (maintenance process). Results achieved in this context, as described in Section 2.3.2, lay the foundation for the development of effective KPIs and represent a fundamental step for all the other foreseen activities. The knowledge of the system allowed the identification of relevant variables and information and the definition of a standard data format, which is viable both for descriptive analytics and data visualization (meaningful and readable for the operators) and for data modelling. The new format is achieved through a specific process which is described in Section 2.3.3. To note that these results are shared between IN2DREAMS WP5 and IN2SMART WP8, where this scenario is exploited for other studies.

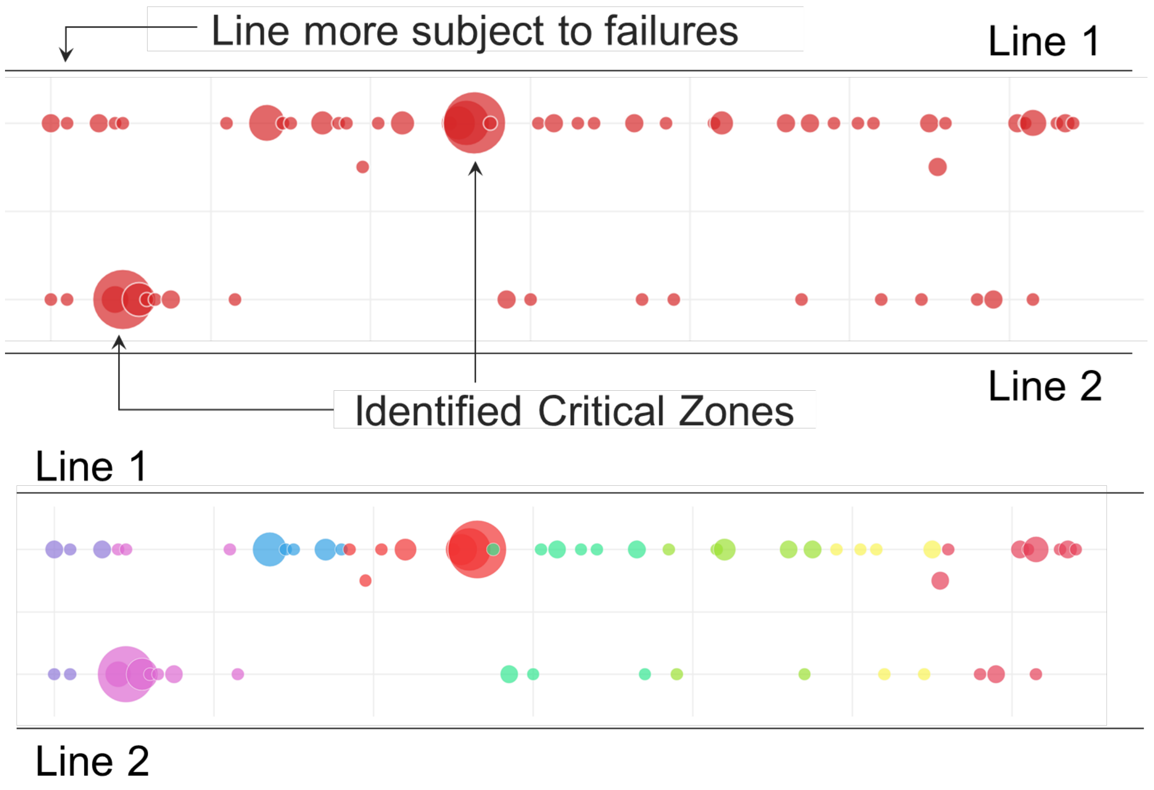


Figure 26: Bubble plot of alarms occurrences (FO alarms) along the railway lines. Three different kind information are represented: geographical (bubble distribution) quantitative (bubbles dimension) and categorical (the color represents the zone to which an asset belongs).

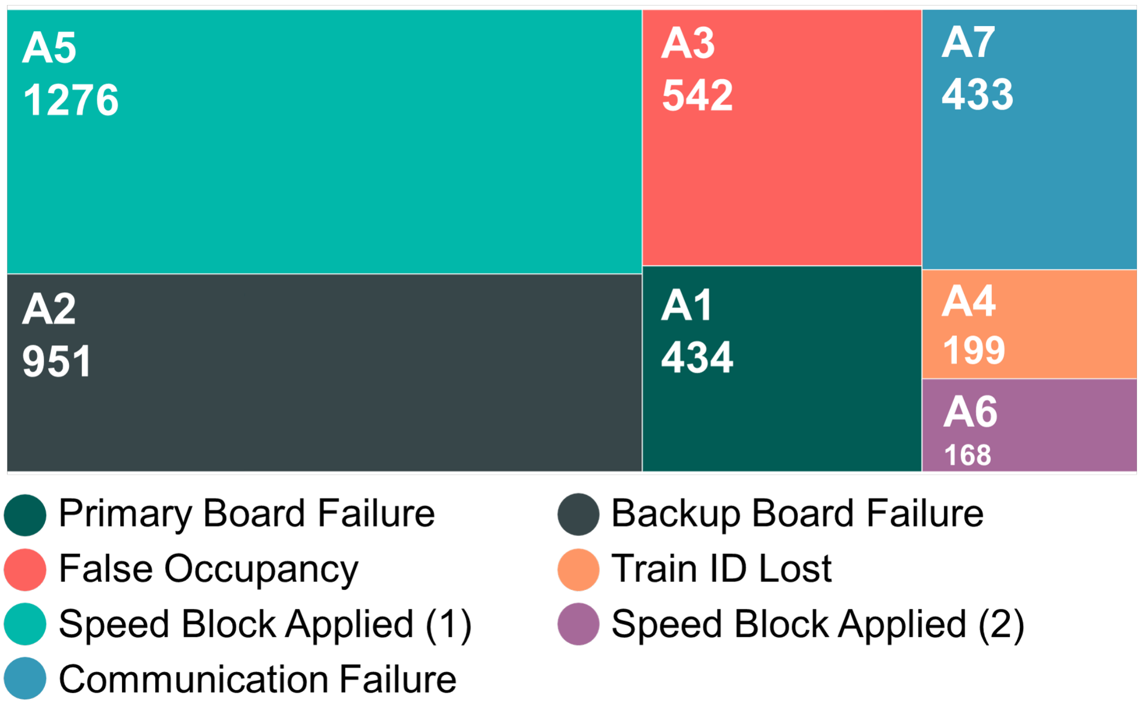


Figure 27: Heat-map of alarms coming from the Central Control System: the size of the boxes is proportional to number of occurrences for each specific alarm

In order to take advantage of the available data in the maintenance process and fulfill operators' requirements different solutions have been identified. The two major points are reported below:

- The data process developed allows to understand TCS normal behaviour, which is represented by four parameters (Shunt Level, Variance, Received Level and Raw Signal Level). As depicted in Figure 25 the trend over time of these parameters can be now visualized (also in a near-real-time fashion) and used as a diagnostic tool to monitor TCS status. Moreover, additional information could be provided to the operators (e.g. information about which board is active or the actual direction);
- The historical collection of the alarms can be exploited for reporting about the status of the whole railway section object of this work, supporting decisions about maintenance planning at a strategic level. Visual analytics techniques enable efficient and understandable reports which can highlight different aspect of the problem and provide additional information in a very simple way: for example, it is possible to provide geographical information together with quantitative information, as shown in Figure 26, and communicate quantitative data in a way which they can be understood in a simple and immediate way (e.g. using heat-maps, as shown in Figure 27).

For what concern predictive modelling, some experiments have been performed with the short-time-window approach described in Section 2.3.4, but no relevant results have been achieved due to the sampling frequency of the signals, which is too low. Thus, during the next steps of the work, for the development of predictive models we will focus on a different approach (focusing on wider time windows) and, at the same time, we will continue to work with the operators involved in order to modify the data acquisition process (to increase sampling frequency) and to develop KPIs for models and preventive maintenance evaluation. Finally, due to the satisfactory results achieved with the used of visual analytics techniques we will proceed on this branch of the work to improve results achieved up to now and to study new solutions for the opportunities which will emerge.

2.4 Specific-Scenario 2: Train Delays and Penalties

We investigate the problem of analyzing the train movements in Large-Scale Railway Networks for the purpose of understanding and predicting their behaviour. We focus on different important aspects: the *Running Time* of a train between two stations, the *Dwell Time* of a train in a station, the *Train Delay*, and the *Penalty Costs* associated to a delay. Two main approaches exist in literature to study these aspects. One is based on the knowledge of the network and the experience of the operators. The other one is based on the analysis of the historical data about the network with advanced data analytics methods. In this work, we will propose an hybrid approach in order to address the limitations of the current solutions. In fact, experience-based models are interpretable and robust but not really able to take into account all the factors which influence train movements resulting in low accuracy. From the other side, Data-Driven models are usually not easy to interpret, nor robust to infrequent events, and require a representative amount of data which is not always available if the phenomenon under examination changes too fast. Results on real world data coming from the Italian railway network will show that the proposed solution outperforms both state-of-the-art experience and Data-Driven based systems in terms of interpretability, robustness, ability to handle non recurrent events and changes in the behaviour of the network, and ability to consider complex and exogenous information.

2.4.1 Introduction

Railway Transport Systems (RTSs) play a crucial role in servicing the global society and the transport backbone of a sustainable economy. A well functioning RTS should met the requirements defined in the form of the 7R formula [62, 70]: Right Product, Right Quantity, Right Quality, Right Place, Right Time, Right Customer, and Right Price. Therefore, an RTS should provide: (i) availability of appropriate products (the provisioning of different categories of train), (ii) proper number of executed transportation tasks (enough trains to fulfill the request), (iii) proper quality of execution of transportation tasks (safety, correct scheduling, and effective conflicts resolution), (iv) right place of destination according to a timetable (correct transportation routes), (v) appropriate lead time (reduced *Train Delays*), (vi) appropriate recipients (focused on different customer needs and requirements), and (vii) appropriate price (both from the point of view of the customers and the infrastructure managers).

In this work we focus on the problem of analyzing the train movements in Large-Scale RTSs for the purpose of understanding and predicting their behaviour. Hence, we will study four important aspects: the *Running Time*, the *Dwell Time*, the *Train Delay*, and the *Penalty Costs*. The first one is the amount of time a train spends in travelling between two consecutive stations. The second one is the amount of time a train spends in a station. The third one is the difference between the actual arrival (or departing time) and the scheduled one in each of the stations composing the itinerary of a train. Finally, the fourth one is the penalty that the Infrastructure Managers (IMs) and the Train Operators (TOs) have to pay because of the delays in proportion to their responsibilities. These aspects are of paramount importance in the context of an RTS. Studying them, and being able to predict their behaviour, allows to improve the quality of service, the train circulation, and the IMs and TOs management costs. More specifically, in relation with the 7R formula, it allows to improve the Right Quantity (improving circulation improves the network capacity without requiring massive public investments in new physical assets), the Right Quality (it helps the operators to understand how much a train needs from one checkpoint to another, to provide a timely resolution of the conflicts on the network and, to correctly schedule all the trains), the Right Time (efficiently predict the train transits improves the ability of the operators to maintain the correct train circulation), and the Right Price (it helps to minimize the penalties for the IMs and TOs).

A large literature covering the aforementioned problems already exists [31]. However, the majority of the works focus just on a single aspect of the train movements. The *Running Time* and *Dwell Time* have been exploited mainly to retrieve train positions and track occupations [18, 42], or to detect train conflicts [33], or

to perform a correct dispatching [7, 47, 52]. The *Train Delay* prediction is the most investigated aspect of train movements [8, 9, 27, 34, 43, 53, 57, 85]. Some works study how the *Train Delays* propagate in subsequent stations [19], for online track conflict predictions [34], and for deriving dependencies between trains [28, 82]. For what concern the study and the prediction of the *Penalty Costs* in [54] it has been studied the relation between *Penalty Costs* and *Train Delays* in the Britain's railway.
To the best knowledge of the authors, there is no work in the literature which deals comprehensively with all the aspects of the train movements as we will propose in this work.
From a methodological point of view, the models adopted in literature to solve the train movements related problems can be grouped in two categories [31]. Models in the first category, called Experience-Based Models (*EBMs*), attempt to exploit the knowledge of the network in order to derive a model which takes into account the physical characteristics and limitations of the network (e.g. speed limits, usury, and slopes) and the trains (e.g. acceleration, weight, and number of wagons) together with the experience of the operators [15, 20, 28, 31, 32, 34]. Models in the second category, called Data-Driven Models (*DDMs*), are based on the analysis of the historical data about the network coming from the most recent Railway Information System with advanced analytic methods [31, 42, 63]. Both *EBMs* and *DDMs* have strengths and weaknesses. *EBMs* are usually low computational demanding, easy to interpret, and robust. A the same time, *EBMs* are usually not very accurate, hard to modify in order to contemplate complex phenomena (e.g. congestion of the network and weather conditions), and not dynamic (they tend to oversimplify the phenomenon not taking into account behaviour's drifts). On the other side, *DDMs* are much more accurate but they are also much more computational demanding (at least for building them and sometimes also for making predictions), often not easy to interpret (interpretability in learning from data is a crucial issue nowadays), not really robust (they do not handle well infrequent events), and not very dynamic (if the phenomena under examination change too fast with respect to the possibility to collect enough data about it).
For these reasons, in this work we propose an hybrid approach, that we will call Hybrid Model (*HM*), taking the best from *EBMs* and *DDMs*. In particular, the proposed *HM* will be interpretable (the *HM* will be easy to understand from an operator point of view), robust and dynamic (*HM* will handle well both infrequent events, like the passage of Freight trains, and fast changes of the train movements phenomena, like a timetable modification), easily extensible (it will be able to take into account complex phenomena like the congestion of the network and exogenous factors like the weather conditions), and able to take into account the knowledge about the network and the experience of the operators.
The rest of the paper is organized as follows. Section 2.4.2 describes the RTS train movements related problems. Section 2.4.3 focuses the attention on the particular case of the Italian RTS. Section 2.4.4 presents the actual *EBM* and *DDM* exploited in the Italian RTS. In Section 2.4.5 we present our contribution: the *HM*. In Section 2.4.6 we compare the performance of the *HM* against the *EBM* and the *DDM* on a set of real world data provided by Rete Ferroviaria Italiana (the Italian IM) showing the effectiveness of the proposed approach both in terms of dynamicity, interpretability, and robustness. Section 2.4.7 concludes the scenario.

2.4.2 Problem Description

A railway network can be easily described with a graph. Figure 28 depicts a simplified railway network where a train follows an itinerary characterized by a station of origin (station *A*), a station of destination (station *F*), some stops (stations *A*, *D*, and *F*) and some transits (checkpoints *B*, *C*, and *E*).
We call checkpoint a station without differentiating between a station where the train stops or transits and between actual stations and points of measure. In fact, not all checkpoints are actual stations since in long railway sections it is often needed to add a point of measure for following the trains with a better granularity. The railway sections are the pieces of the network between two consecutive checkpoints, note that railway sections have also an orientation (e.g. transit *D* to *E* is different from transit *E* to *D*).

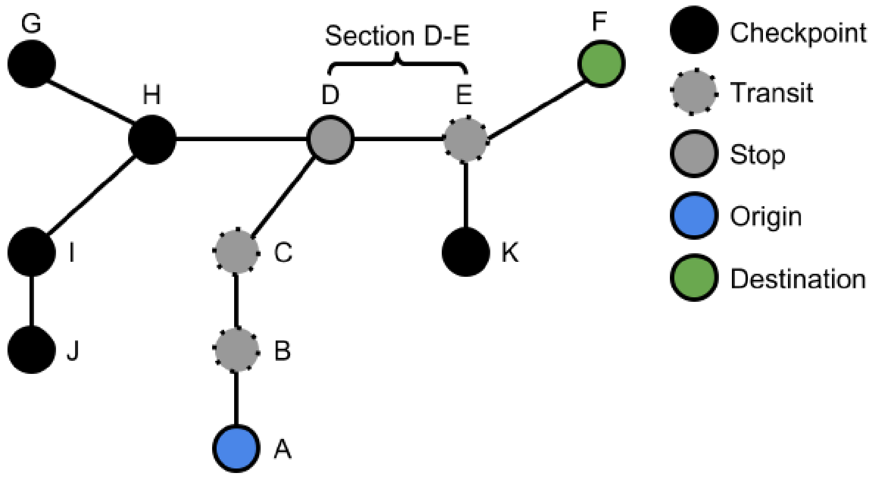


Figure 28: A railway network. The itinerary of a train is depicted with the grey nodes where *A* is the origin station and *F* is the destination.

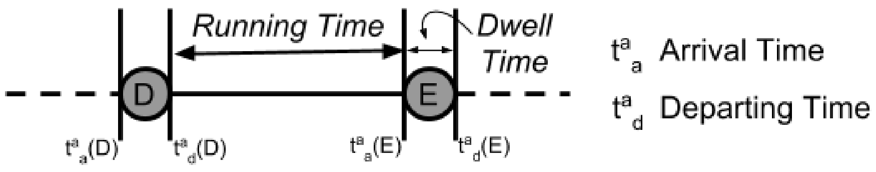


Figure 29: *Running Time* and *Dwell Time*.

For any checkpoint in the itinerary, the train is scheduled to arrive and depart at different specified times, defined in the timetable, respectively t^s_a and t^s_d. Usually, the time references included in the timetable are approximated with a precision of 30s. The difference between the scheduled time and the actual time, either for arrival (t^a_a) or for departure (t^a_d), is defined as *Train Delay*. If the delay is greater than 30s, then a train is considered as delayed. Note that, for the origin station there is no arrival time, while for the destination station there is no departure time. We define the *Running Time* as the amount of time needed to depart from the first of two subsequent checkpoints and to arrive to the second one (see Figure 29, for railway section *D* to *E* the scheduled *Running Time* is $t^s_a(E) - t^s_d(D)$ while the actual *Running Time* is $t^a_a(E) - t^a_d(D)$) and the *Dwell Time* is the difference between the departure time and the arrival time in a fixed checkpoint (see Figure 29, in checkpoint *D* the scheduled *Dwell Time* is $t^s_d(D) - t^s_a(D)$ and the actual *Dwell Time* is $t^a_d(D) - t^a_a(D)$).
Furthermore, each train has an unique identifier from which it is possible to retrieve the category of the train (e.g. Regional, Freight, and High Speed). Analogously, each checkpoint has an unique identifier from which it is possible to retrieve the category of the network (e.g. Node, High Speed, and Second Complementary Network). Train, network category, time of the day, and other factors allow to compute the Penalty Costs associated to a delayed train.
Based on these definitions, it is possible to describe the train movements related prediction problems that we will face in this work.

2.4.2.1 Running Time and Dwell Time Prediction

The prediction of the *Running Time* and *Dwell Time* are the first problems that we address. For a specific train, the problem is to predict the *Running Time* for all the subsequent railway sections it will traverse and the *Dwell Time* for all the subsequent checkpoints in which it will stop, updating these predictions every time it reaches the next checkpoint. Providing an accurate prediction of the *Running Time* and the *Dwell Time* allows to provide to the operators a clear understanding of how much time a train needs to complete the itinerary. Moreover, as we will describe later, the *Running Time* and the *Dwell Time* predictions can be exploited as a building block for the *Train Delay* predictors (see the *EBM* of Section 2.4.4.1).

2.4.2.2 Train Delay Prediction

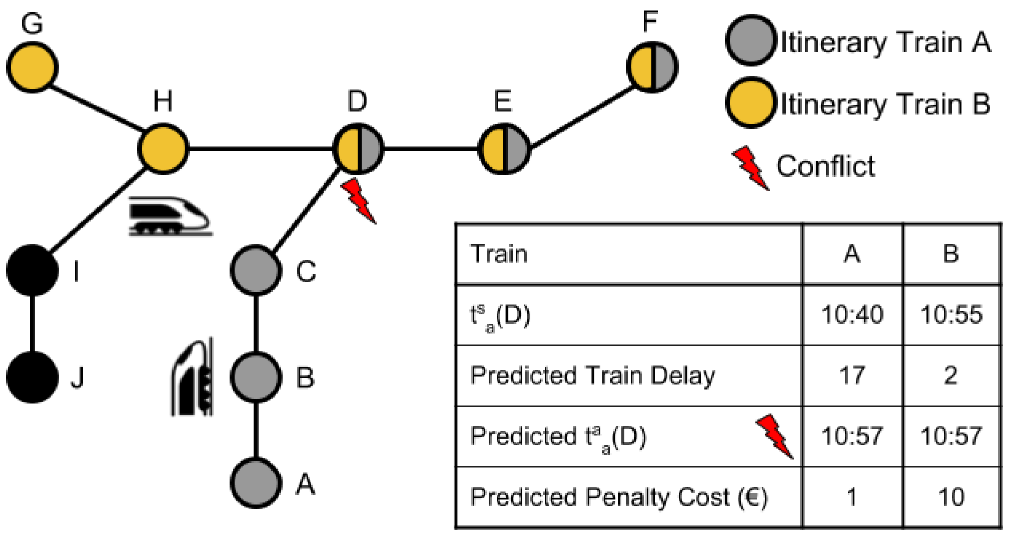
The *Train Delay* prediction is the problem of forecasting the arrival and departing delay of a train for all the subsequent checkpoints in its itinerary, updating this predictions every time it reaches a new checkpoint. The prediction of the future delays is a problem of paramount importance and yields several benefits: a reliable information for the passengers currently on the trains or waiting in a checkpoint, a better exploitation of the railway network while maintaining the safety of the passengers and avoiding resource conflicts, better train rescheduling and dispatching, and more.
Note that, *Train Delay* prediction can be seen as a standalone task (see the *DDM* of Section 2.4.4.2) or it can be retrieved from the combination of the *Running Time* and the *Dwell Time* predictions (see the *EBM* of Section 2.4.4.1).

2.4.2.3 Penalty Costs Prediction

In an RTS the IMs and the TOs have to pay penalties, when trains are delayed, in proportion to their actual responsibilities. For this reason, predicting the *Penalty Costs* is a strategic issue: an effective prediction system can be exploited to choose the best dispatching solution which minimizes both *Train Delays* and *Penalty Costs*. However, this problem is rather complex since the *Penalty Costs* computation is the result of a complex procedure that has to be fully understood.
Currently, in every State a document of management principles (for example, in Italy is the PIR[1]) defines the rules, agreed between the State, the IMs (e.g. Rete Ferroviaria Italiana is an Italian IM), and the TOs (e.g. Trenitalia is an Italian TO), that must be followed to solve the conflicts when one or more trains are delayed and the associated *Penalty Costs* that IMs and TOs have to pay based on their responsibilities. Such rules define the level of priority of each train based on different variables such as the category of the train and time of the day. For instance, during the daily commuter time slot, some Regional trains could have the same priority as the High Speed trains, even if the latter have usually higher priority. In order to enforce the IMs to follow these rules, if a train is delayed, the priorities also influence the *Penalty Costs* associated to a *Train Delay*. Consequently, in order to compute the *Penalty Costs*, it is required to retrieve a series of information regarding the trains and their itinerary. Although a deterministic relation exists to compute the penalties, not all these variables are known at the time of the train transit. The final penalty is usually agreed after the train has completed its journey (even after months). For example, the percentage of responsibility may be the results of a legal dispute between the IM and the TO.
More in details, the *Penalty Costs* is the result of a deterministic combination of the following variables:

- the category of the train (e.g. a train belonging to the market service delayed of one hour has larger *Penalty Costs* than a Freight train affected by the same delay);
- the operational category of the train (e.g. if the itinerary is scheduled in the timetable, or it is created/modified in the last few days before the actual train transit);
- the type of railway section (similarly to the category of the train, the High Speed Lines are affected by a higher *Penalty Costs*);

[1] `http://www.rfi.it/rfi/SERVIZI-E-MERCATO/Accesso-alla-rete/Prospetto-informativo-della-rete`



Train	A	B
t^s_a(D)	10:40	10:55
Predicted Train Delay	17	2
Predicted t^a_a(D)	10:57	10:57
Predicted Penalty Cost (€)	1	10

Figure 30: Handling the conflicts using *Train Delay* and *Penalty Costs* prediction models. Exploiting the *Penalty Costs* prediction would result in stopping *Train A* because is less expensive for the IM. Exploiting, instead the delay would resort in stopping *Train B* for reducing the grater delay of *Train A*.

- the amount of delay of the train (e.g. the average and maximum delay for Regional trains, or just the delay in the final checkpoint for Freight trains);
- the percentage of responsibility of the IMs, of the TOs, and of the exogenous factors (e.g. flooding and strikes).

2.4.2.4 Example

In this section, we present an example to show the usefulness of the predictive models described above. Let us suppose to have two trains travelling the simplified railway network depicted in Figure 28, with two different itineraries as depicted in Figure 30. The first train, *Train A*, travels along its gray itinerary from checkpoint *A* to *F*, while the second one, *Train B*, travels its yellow itinerary from *G* to *F*. The two trains share three checkpoints in their itineraries (checkpoint *D*, *E*, and the destination *F*). The timetable has been constructed in order to give the correct headway to the trains for safety and regularity purposes. Suppose also that *Train A* is in checkpoint *B*, and that *Train B* is in checkpoint *H*.
Exploiting the *Train Delay* predictor, we discover that both trains will arrive at approximately the same time in checkpoint *D*, leading to a conflict. Then, we have to decide which one of the two trains should have the priority over the other. Exploiting just the *Penalty Costs* prediction would result in stopping the *Train A* because is less expensive for the IM, while exploiting just the *Train Delays* prediction would resort in giving the priority to the *Train A* for reducing its grater delay. Considering instead both *Penalty Costs* and *Train Delays* predictions, would result in a more aware decision. In this case, the most reasonable solution is to stop *Train A* since it will negligibly increase its delay (few additional minutes) to make *Train B* go forward, possibly regaining some delay which is instead very costly for the IM (so, probably, it is a more important train).

2.4.3 The Italian Railway Transportation System

In this document, we consider the specific case of the Italian RTS, which is substantially handled by just one IM, Rete Ferroviaria Italiana (RFI), which provided to us both the knowledge of the network and the data

needed for this study.
According to the *International Union of Railways*, the Italian RTS is in the Top 3 and the Top 10 largest RTS respectively in Europe and Worldwide. RFI controls every day ≈ 10.000 trains travelling along the national railway network of ≈ 25.000km. Every train is characterized by an itinerary composed of an average of ≈ 12 checkpoints. This means that the number of train movements is greater than or equal to ≈ 300.000 per day. This results in more than one message per second and more than 10GB of messages per day to be stored.
Note that, every time a message describing the itinerary of a particular train is retrieved, the predictive models can take advantage of this new information both to make better predictions and to updated the model itself. This allows to have always the best performing models which exploits all the available information, and to follow the effects of small or big changes in the timetables that occur during the year.
Apart from the daily messages of the train movements, RFI is also able to provide all the information about the travelling trains and network characteristics needed to compute the *Penalty Costs* according to the PIR (see Section 2.4.2.3).
Finally, other exogenous information regarding the network can be retrieved from many Italian freely available data sources which can help in improving the accuracy of the *DDM*s. In this work, we will take into consideration the weather information (see e.g. [68, 69]) since in previous works it has been shown to be an effective solution for improving the *DDM* accuracy [63].

2.4.4 The Actual Systems

This section describes two different state-of-the-art approaches employed in RFI to tackle the problems described in this paper. In particular, RFI exploits both a *EBM* which is quite similar to the one described in [34] (although the latter includes process mining refinements which potentially increase its performance) and a *DDM* [63] that produces better predictions of the *Train Delays* with respect to *EBM*.

2.4.4.1 The Actual Experience-Based Model

The actual RFI *EBM* performs the predictions based on the knowledge of the railway network and the experience of the operators. It focuses mostly on the problem of predicting the *Running Time*. The *Dwell Time* is considered fixed to the difference between the scheduled departure and arrival time in a station. The *Train Delays* and the *Penalty Costs* are derived from the predicted *Running Times* and the fixed *Dwell Times* assuming that the percentage of responsibility for a delay is always 100% of RFI.
More in details, the idea of the *EBM* is to analyze the amount of time that a train needs to traverse each railway section of the network, taking into account the speed limits, the state of the network, the type of train etc. The timetables are produced taking in consideration such physical constraints and a working margin is kept for dealing with delays. Then, for each railway section and each train category, a coefficient, called *Gaining Time*, is computed which represents the time that be can regained in case of delay (the *Gaining Time* takes into account also a possible smaller *Dwell Time*). The *Gaining Time* is static, i.e. it does not change based on the state of the network, weather conditions, etc. The *Gaining Time*, is exploited to solve the *Train Delay* prediction problem. When predicting a delay, it is assumed that a delayed train is always able to regain, in a given railway section, an amount of time equals to its *Gaining Time*. Then, when RFI predicts the *Train Delay* in a subsequent checkpoint it subtracts from the current delay all the *Gaining Times* of the railway sections between the actual station until the considered checkpoint. Once the delay is computed, the *Penalty Costs* can be derived straightforwardly if, as in RFI, it is assumed that 100% of the delay costs is to impute to RFI, thanks to the deterministic formula that can be found in the PIR.
The *Gaining Times* of the RFI *EBM* do not depend on the time of the days, on the fact that it is a weekend or a weekday, on the train actual delay, on the network congestion, on the weather conditions since no easy

relation can be retrieved. On the other side, the RFI *EBM* is quite robust and easy to understand from an operator perspective even if not very accurate and dynamic.

2.4.4.2 The Actual Data-Driven Model

Given the low accuracy of the *EBM*, in RFI it has been decided to exploit also the *DDM* developed in [63]. The *DDM* does not take into account the knowledge of the railway network nor the experience of the operators, but it is based just on the historical data about train movements, weather conditions, and weather forecasts. For this purpose, the *DDM* exploits advanced analytic methods able to extract accurate models of the future behaviour of each train. The advantage of these methods is that there is no need of any a-priori knowledge of the underline physical system but, most of the time, they produce non-parametric models that are not easy to interpret nor supported by any physical intuition or interpretation. Moreover, in general, a great amount of historical data is needed in order to build an accurate model and it is not so easy to make these systems strongly dynamics. In fact, if for example the timetable changes, they require at least one month of data before achieving a reasonable accuracy.
The RFI *DDM* is composed of many *DDM*s that, working together, make it possible to perform a regression analysis on the past delay profiles in order to predict the future ones. In particular, for each train and for each checkpoint composing its itinerary, a set of *DDM*s is built to predict the delay in all the subsequent checkpoints. Consequently, the total number of *DDM*s to be built for each train is $\approx n(n-1)$ where n is the number of checkpoints visited by the train. These *DDM*s work together to estimate the delays of a particular train during its entire journey. For a single train, every time it arrives at (departs from) a specific checkpoint included in its trip, the *DDM*s take as inputs its previous sequence of arrival and departure *Train Delays*, *Running Times*, and *Dwell Times* to predict delay for all the subsequent checkpoints. These *DDM*s are also able to take into account the state of the congestion of the network and other exogenous variables (e.g. the weather information) [63]. The *DDM*s can be built using many different learning algorithms, exploiting the Random Forest (RF) usually leads to better results [14].
Unfortunately, the RFI *DDM* has some drawbacks. Many historical information about the trains are requested before performing the prediction, otherwise it performs badly (e.g. on new trains or after changes in the timetable). Moreover, each model composing the *DDM* is specific for one particular train and checkpoint limiting its interpretability on a larger scale (it cannot group similar trains or trains in the same category) and the complexity of the *DDM* is higher with respect to *EBM* (too many models to build). Finally, the *DDM* does not integrate the knowledge and the experience of the operators nor gives to the operators an interpretation of the *Train Delay* phenomenon.

2.4.5 The Proposed Hybrid Model

In this work, we propose an *HM* to perform the *Running Time*, *Dwell Time*, *Train Delay*, and *Penalty Costs* predictions, merging together the *EBM* and the *DDM* to exploit their strengths and limit their weaknesses. The goal is to build accurate, dynamic, robust, and interpretable models able to provide insights for both solving the train conflicts and minimizing the *Train Delays* and the *Penalty Costs*.
Similarly to the *EBM*, the *HM* relies, on the top, on an interpretable model able to encapsulate the experience of the operators in the form of a decision tree and, at the bottom, the leafs, instead of being defined relying on the physical knowledge of the network as in the *EBM*, are constructed following the ideas of the *DDM* where the historical data about the network and other exogenous information (e.g. weather) are exploited via advanced analytic methods. Moreover, contrary to the *DDM*, the *HM* does not implement one model for each train and, contrary to the *EBM*, the *HM* does not group all the trains just based on their category and railway section. In fact, the *HM* groups the trains based on a series of similarity variables, defined together with the RFI operators, which allow to have, from one side, robust statistics, thanks to the possibility to

Table 4: Description of the *HM* top level decision tree feature set.

Feature Name	Categorical	Description
Railway Section	Yes	The considered railway section
Railway Checkpoint	Yes	The considered railway checkpoint
Train Type	Yes	The considered Train Type
Daytime	No	The time of the day with an hourly granularity
Weekday	Yes	The day of the week
Last Delay	No	The last known delay with the following granularity in minutes ($[0, 2]$, $(2, 5]$, $(5, 10]$, $(10, 20]$, $(20, 30]$, $(30, 60]$, $(60, 120]$, $(120, \infty)$);
Weather Conditions	Yes	The weather conditions (Sunny, Light Rain, Heavy Rain, Snow).

Table 5: Description of the *HM* bottom level RF feature set.

Feature Name	Categorical	Description
Weather Information	Yes	Weather condition (Sunny, Light Rain, etc.) in all the checkpoints of the train itinerary (for the already traveled checkpoints we use the actual weather while for the future checkpoints we use the predicted weather conditions)
Past *Train Delays*	No	Average value of the past *Train Delays* in seconds & Last known *Train Delay*
Past *Dwell Times*	No	Average value of the past differences between actual and scheduled *Dwell Times* in seconds & Last known difference between actual and scheduled *Dwell Time*
Past *Running Times*	No	Average value of the past differences between actual and scheduled *Running Times* in seconds & Last known difference between actual and scheduled *Running Time*
Network Congestion	No	Number of trains traversing the checkpoints of the train itinerary in a slot of 20 minutes around the actual and scheduled times respectively for the past and future checkpoints
Network Congestion Delays	No	Average *Train Delay* of the trains traversing the checkpoints of the train itinerary in a slot of 20 minutes around the actual and scheduled times respectively for the past and future checkpoints

learn from a reasonable group of train, but also a rich feature set, to be able to capture the variability of the phenomena. The proposed *HM* is then able to be extremely dynamic: grouping the trains increases the number of historical data to exploit during the leaf creation and follow, in a reasonable amount of time, timetable changes and new train schedules, thanks to the robustness introduced by the *HM* experience based top level structure.
We exploited the above mentioned approach for predicting both the *Running Time* and the *Dwell Time*. For what concern the *Train Delay*, instead, we opted for the same solution of the actual RFI *EBM* (see Section 2.4.4.1). In fact, in order to predict the *Train Delay* at a desired subsequent checkpoint, we sum all the needed *Running Time* and *Dwell Time* predictions to the current train time and then we compute the difference between the estimated and the scheduled train time. Finally, in order to predict the *Penalty Costs*, we made use of the *HM* described in the previous paragraph to predict an auxiliary variable, the *Responsibility*, which is the percentage of responsibility of the IM for the delays. Then, combining the *Train Delay* and the *Responsibility* predictions, we were able to predict the *Penalty Costs* exploiting the deterministic relation described in the PIR.
The work has been conduced side by side with the RFI operators taking into account their needs and their working environment which is constrained, in terms of complexity of the solution, to something that can provide simple and effective insights.
In the subsequent subsections, we will first present in details how we constructed the above mentioned *HM* decision tree based top structure and its Data-Driven based bottom structure (see Section 2.4.5.1), and then we will describe how this *HM* has been exploited for predicting the *Running Time*, the *Dwell Time*, the *Train Delay*, and the *Penalty Costs* (see Section 2.4.5.2).

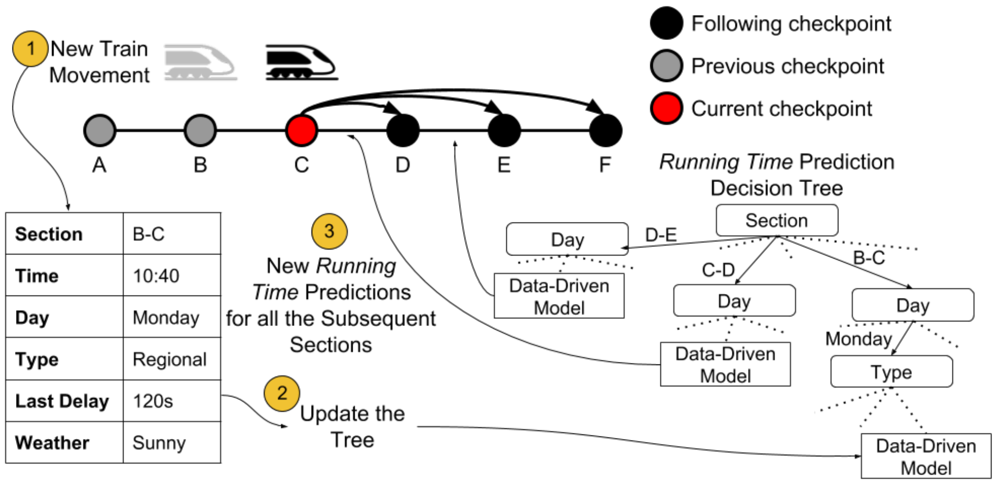


Figure 31: The proposed *HM* for the *Running Time* prediction: updating the model every time a new movement is recorded and predicting the future *Running Time* in the subsequent sections.

2.4.5.1 Hybrid Decision Tree

As described before, the *HM* exploits, as a basic structure, a top level experience based decision tree and a bottom level Data-Driven model which is able to easily take into account the network congestion state and other exogenous information, like the weather conditions, which are not easy to model with the experience. The top level structure can be easily adapted to the prediction task under examination. For instance, for the *Running Time* we are interested in considering each railway section separately, instead for the *Dwell Time* prediction it is better to differentiate each of the checkpoints. The variables that we consider in the top level structure, defined with the RFI experts, are a subset of the ones reported in Table 4. Then, as leafs of the tree, instead of plugging an estimate of the quantity that we want to predict based on the experience of the operators and the knowledge of the network, we exploit a Data-Driven model able to learn from the historical data regarding all the trains which fall in that particular leaf (basically all trains which share similar characteristics and itinerary) plus additional complex features. In particular, each leaf is a RF regressor [14] (following the experience of the *DDM* developed in [63]), which predicts the quantity that we want to estimate based on a series of features designed with the RFI experts and based also on the lesson learned with the *DDM* [63]. This feature set is reported in Table 5.
The whole *HM* is constructed and updated incrementally as soon as a new train movement is recorded. In the top level decision tree, a new leaf is added each time we record a new train movement which belongs to a previously unexplored branch of the decision tree. Then, the RF regressor in the leaf is learned based on all the past train movements which fall in that particular leaf. In order to follow the changes in behaviour of the phenomena we forgot the train movements older than three months. The predictions phase, instead, is simpler: we just visit the tree considering the information that we want to predict and we exploit the correct RF regressor to make the actual prediction.
As described at the beginning of Section 2.4.5, the *HM* will be exploited for predicting:

- the *Running Time*: in this case we exploit, in the *HM* top level decision tree, all the variables of Table 4 except the one relative to the checkpoints since *Running Time* is a property of the railway sections and do not depend on the checkpoints;
- *Dwell Time*: in this case we exploit, in the *HM* top level decision tree, all the variables of Table 4 except the one relative to the railway sections since *Dwell Time* is a property of the checkpoints and do not depend on the railway sections;

- *Responsibility*: in this case we exploit, in the *HM* top level decision tree, all the variables of Table 4.

Figure 31 depicts an example of use of the *HM* for the *Running Time* prediction problem. As one can see from Figure 31, every time a new movement is recorded the *HM* is updated based on the information inside the train movement record and we exploit this new information about the travel of the train to update all the predictions about the subsequent railway sections.

2.4.5.2 Train Movements Predictors via Hybrid Model

In this section we describe how the previously described *HM* has been exploited for predicting the *Running Time*, the *Dwell Time*, the *Train Delay*, and the *Penalty Costs*.

2.4.5.2.1 Running Time Prediction

In this case we apply the *HM* described in Section 2.4.5.1 and we directly predict the values of the *Running Times*. Every time a train movement is recorded, the model and the predicted future *Running Times* are updated based on this new information.

2.4.5.2.2 Dwell Time Prediction

Regarding the *Dwell Time* prediction we exploit exactly the same approach described for the *Running Time* prediction. Note that, the only difference between the two models stays in the feature set of the *HM* top level decision tree (see Section 2.4.5.1).

2.4.5.2.3 Train Delay Prediction

In order to predict the *Train Delays*, instead of building another *HM*, we exploit, similarly to the *EBM*, the *Running Time* and *Dwell Time* predictors as building blocks. Each time a prediction is required, we predict all the *Running Times* of the sections and all the *Dwell Times* of the checkpoints between the current checkpoint and the one for which we request the *Train Delay* prediction. Then, the desired result is obtained by summing all these times to the current time and subtracting from the results the scheduled time.

2.4.5.2.4 Penalty Costs Prediction

In order to compute the *Penalty Costs* of a particular *Train Delay*, we have to combine two quantities. First, we obtain the predicted *Train Delay* exploiting the approach described in Section 2.4.5.2.3. Then, we predict the *Responsibility* with a new *HM* as described in Section 2.4.5.1. Once these two predictions are available, we combine them with the deterministic relation described in the PIR, obtaining the *Penalty Costs* prediction. Specifically, we compute the *Penalty Costs* P of a train as follows:

$$P = P_U \sum_{j \in \mathcal{I}} m_j r_j C_T C_N C_D, \tag{1}$$

where P_U is the unitary costs, $\mathcal{I}$ is the set of checkpoints composing the itinerary, m_j are the minutes of delay at section j, r_j is the percentage of responsibility for section j, C_T is coefficient relative to the type of the train T, C_N is the coefficient relative to the type of railway network N, and C_D is a coefficient which depends on the average and maximum delay registered for the train. The parameters m_j and C_D are estimated with the *Train Delay* predictor. The parameters r_j are estimated with the *Responsibility* predictor. More details about Eq. (1) can be found in the PIR.

2.4.6 Experimental Evaluation

In this section we test the proposed *HM*, presented in Section 2.4.5, against the actual RFI *EBM*, presented in Section 2.4.4.1, and *DDM*, presented in Section 2.4.4.2.
All the experiments have been conducted on a virtual machine in the Google Cloud Platform[2] (GCP). The machine is the *n1-standard-8* characterized by 8 core and 30GB of RAM and 500GB of SSD disk space. Each experiment has been repeated 30 times in order to ensure the statistical robustness of the results.

2.4.6.1 Available Data

The experiments have been conducted exploiting the real data provided by RFI:

- data about train movements which contains the following information: Date, Train ID, Checkpoint ID, Actual Arrival Time, Arrival Delay, Actual Departure Time, Departure Delay and Event Type. The Event Type field can assume different values: Origin (O), Destination (D), Stop (F), Transit (T);
- data about the delay responsibilities: for every delay the percentage of RFI responsibility is available;
- timetables, including planning of exceptional train, cancellations, and *Gaining Time* of each section.

For the purpose of this work, RFI provided the access to the data of 12 months (the whole 2016 solar year) of train movements of one big Italian Region (Liguria). The data are relative to more than 2.500 trains and 146 checkpoints. The dataset contains $4.127.380$ train passages.
From the PIR, freely available on the RFI website[1], we retrieved all the information needed to compute the *Penalty Costs* as described in Section 2.4.5.2.4.
We also exploit, as exogenous information, the weather conditions from the weather stations in the area. For each checkpoint we consider the closest weather station to the railway station/line. We collect the data relative to the solar radiation and precipitations for the same time span of the train passages from Italian national weather service databases, which are publicly accessible for the Liguria Italian Region at [68]. From this data it is possible to extract both the actual and the forecasted weather conditions (Sunny, Rain, Heavy Rain, and Snow).

2.4.6.2 Key Performance Indicators

In the experiments, we exploit the following Key Performance Indicators (KPIs) for measuring the quality of the different models (in parenthesis we report the prediction problem where they have been applied). These KPIs have been designed together with RFI based also on the lesson learned during the exploitation of the *DDM* [63]:

- AASk (*Running Time* prediction): the Average Accuracy for a particular Section k. AASk is computed as the averaged absolute value of the difference between the predicted and the actual *Running Times* in minutes;
- AAS (*Running Time* prediction): is the average over the different sections k of AASk;
- AACk (*Dwell Time* prediction): the Average Accuracy for a particular Checkpoint k. AACk is computed as the averaged absolute value of the difference between the predicted and the actual *Dwell Times* in minutes;
- AAC (*Dwell Time* prediction): is the average over the different checkpoints k of AACk;
- AAiCTk (*Train Delay* prediction): the Average Accuracy at the i$-th$ subsequent Checkpoint for Train k. For a particular Train k, the absolute value of the difference between the predicted delay and its actual *Train Delay* is averaged, at the i$-th$ subsequent checkpoint with respect to the actual checkpoint in minutes;

[2]Google Compute `https://cloud.google.com/products/`

Table 6: Comparison between *HM* and *EBM* for *Running Time* prediction. (n) is the number of train passages in the section.

k	n	AASk	
		EBM	*HM*
1	7344	1.1	**0.9**
2	10672	1.7	**0.8**
3	22082	1.2	**0.9**
4	1013	1.4	**0.4**
5	25228	0.5	**0.4**
6	18090	0.8	**0.5**
7	398	3.2	**2.9**
8	12671	1.2	**0.6**
9	29357	1.4	**0.9**
10	5614	2.7	**1.5**
. . .			
AAS Regional		1.3	**0.8**
AAS High Speed		0.8	**0.6**
AAS Freight		1.9	**1.2**
AAS		1.3	**0.9**

Table 7: Comparison between *HM* and *EBM* for *Dwell Time* prediction. (n) is the number of train passages in the checkpoint.

k	n	AACk	
		EBM	*HM*
1	49134	1.7	**0.7**
2	61888	**0.1**	0.3
3	22210	1.4	**1.2**
4	23629	2.4	**1.8**
5	29652	2	**1.6**
6	29271	1.3	**1**
7	33350	1.2	**0.9**
8	22508	0.5	**0.2**
9	33418	**0.8**	1
10	24307	**0.5**	0.9
. . .			
AAC Regional		**1.1**	**1.1**
AAC High Speed		**0.5**	0.7
AAC Freight		2.5	**1.5**
AAC		1.1	1

- AAiC (*Train Delay* prediction): is the average over the different trains j of AAiC;
- AAP (*Penalty Costs* prediction): is the Average Accuracy over the different trains between the predicted and actual *Penalty Costs* in Euros.

2.4.6.3 Results

In this section we compare the proposed *HM* for predicting *Running Times*, *Dwell Times*, *Train Delays*, and *Penalty Costs* against the *EBM* and *DDM*, by using the data described in Section 2.4.6.1 and the KPIs described in Section 2.4.6.2

2.4.6.3.1 Running Time Prediction

In this first set of experiment we compare the *HM* with the *EBM* on the *Running Time* prediction problem. We could not compare them also with the *DDM* since it does not provide a solution for this problem [63]. Table 6 reports the AASk for a subset of the railway sections and the AAS also considering the different train types[3]. From the results it is possible to observe that:

- *HM* clearly outperforms the *EBM*;
- the improvement is more evident for Freight and Regional trains, instead for High Speed trains the two approaches provide similar results.

In order to show the ability of the proposed solutions to handle changes in the timetable, Figure 32 reports the value of AAS during the 2016. From Figure 32 it is possible to observe that:

- *HM* is constantly better with respect to the *EBM* during the whole year;
- *HM* needs really little time to learn a good model, for example in January after 10 days of data it reaches almost its optimal accuracy;
- *HM* and *EBM* exhibit an increase in the error in June (days from 180 to 210), this is motivated by a change in the timetable happened the 12th of June.

[3]Because of confidentiality issues we cannot report the results and the ids for all the sections and all the checkpoints available.

Table 8: Comparison between *HM*, *EBM*, and *DDM* for *Train Delay* prediction. (n) means the number of days that the train transit according to our dataset. (–) means not available since data is not enough to build the model.

AAiCTk		*EBM*	*DDM*	*HM*	*EBM*	*DDM*	*HM*	*EBM*	*DDM*	*HM*	*EBM*	*DDM*	*HM*	
k \ i	n	1st			2st			3st			4st			
1	349	1	0.6	**0.5**	1.2	**0.7**	1	1.7	**1.1**	1.5	2.1	**1.4**	2.1	
2	346	1	0.5	**0.4**	1.2	**0.8**	0.9	1.5	**1**	1.3	1.8	**1.2**	1.6	
3	345	0.5	0.4	**0.3**	0.9	0.6	**0.4**	1.1	0.8	**0.6**	1.2	1	**0.7**	
4	308	0.9	**0.5**	**0.5**	1.5	**1**	1.2	1.7	**1.3**	1.4	1.9	**1.4**	1.8	
5	235	0.9	0.9	**0.7**	1.5	**1.3**	**1.3**	2	**1.5**	1.8	2.5	**1.8**	2.3	
6	175	0.7	**0.4**	0.5	1.1	**0.7**	1	1.4	**0.7**	1.3	1.8	**1**	1.7	· · ·
7	169	0.7	**0.4**	**0.4**	1	**0.6**	0.9	1.3	**0.6**	1.2	1.6	**0.8**	1.6	
8	129	2.4	3.4	**1.6**	5.1	6.2	**3.9**	7.8	9	**6.4**	9.9	11.3	**8.2**	
9	14	1.4	–	**1.1**	2.1	–	**1.7**	2.8	–	**2.2**	3.1	–	**2.6**	
10	2	1.8	–	**1.1**	3.8	–	**1.9**	5.9	–	**3**	7.6	–	**4**	
· · ·														
AAiC Regional		1.2	**0.8**	0.9	2.1	**1.5**	1.7	3	**2.2**	2.5	3.8	**2.8**	3.3	
AAiC High Speed		0.7	0.7	**0.5**	1.2	1.1	**1**	1.6	**1.4**	**1.4**	2	**1.7**	1.8	
AAiC Freight		1.9	3.5	**1.6**	3.6	5.2	**3.1**	5.3	6.9	**4.7**	6.9	8.2	**6.1**	
AAiC		1	0.9	**0.8**	1.8	**1.5**	1.6	2.5	**1.8**	2.3	3.2	**2.1**	2.9	

Table 9: Comparison between *HM* and *EBM* for *Penalty Costs* prediction.

	EBM	*HM*
AAP Regional	4.15	**2.49**
AAP High Speed	0.2	**0.14**
AAP Freight	0.11	**0.1**
AAP	4.44	**2.71**

Note that, even if the *HM* has a Data-Driven core, it is still robust and the *EBM* and much more dynamic of any *DDM*.

2.4.6.3.2 Dwell Time Prediction

For what concerns the *Dwell Time* prediction problem, the approach, the results, and the comments are quite similar to the one made for the *Running Time* prediction problem. Table 7, analogously to Table 6, reports the AACk for a subset of the checkpoints and the and AAC also considering the different train types[3]. From Table 7 it is possible to observe that:

- in this problem *EBM* and *HM* provide similar results, *HM* being slightly better;
- similarly to the result for the *Running Time* prediction problem the *HM* approach results to be particularly effective for the Freight trains.

We do not report the equivalent of Figure 32 since results are basically the same.

2.4.6.3.3 Train Delay Prediction

In this section we compare the *HM* with both the *EBM* and the *DDM* for the *Train Delay* prediction problem. Table 8 reports the AAiCTk for a subset of the trains and subsequent checkpoints and the AAiC also considering the different train types[3]. From the results it is possible to observe that:

- both the *HM* and *DDM* perform better with respect to the *EBM* approach;
- the *HM* better predicts the delays in the subsequent checkpoint ($i = 1$);
- the *DDM* better predicts the delays when the distance from the actual checkpoint is larger;

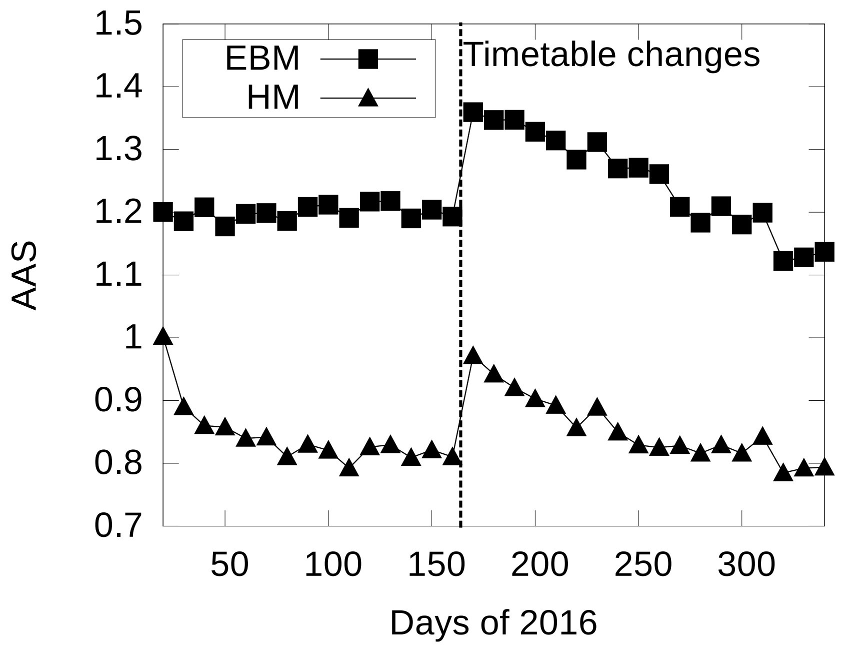


Figure 32: AAS for the *Running Time* during the year.

- *DDM* is not able to perform the prediction for the trains for which we have too little information (i.e. infrequent trains) while *HM* is always able to provide an answer;
- for what concern the Freight trains, *DDM* provides the largest error while *HM* improves of $\approx 20\%$ over also the *EBM*.

2.4.6.3.4 Penalty Cost Prediction

In this section we compare the *HM* with the *EBM* on the *Penalty Costs* prediction problem. We could not compare them also with the *DDM* since it does not provide a solution for this problem [63].
Table 9 reports the AAP considering the different train types[3]. From Table 9 it is possible to observe that:

- the *HM* is much more effective with respect to the *EBM* for all the train categories;
- the difference is much more evident for the Regional trains which are also the most expensive in terms of *Penalty Costs* for RFI.

2.4.6.4 Computational Requirements

Finally, we compare the computational requirements of the different models. Figure 33 depicts both the scalability varying the number of cores (left) and the trade-off between accuracy and computational requirements (right) for the *Train Delay* prediction case (AAiC with $i = 1$). The time reported on the axis is the time needed for performing the analysis of all the 12 months of data provided by RFI.
From Figure 33 we can observe that:

- *EBM* and *HM* have a similar scalability, the computational time decreases smoothly when more cores are added to the computation;
- *DDM*, when 8 cores are exploited, requires $100\times$ the time with respect to *EBM* and *HM* (we did not execute *DDM* with less than 8 cores because the computation required more than 10 hours);

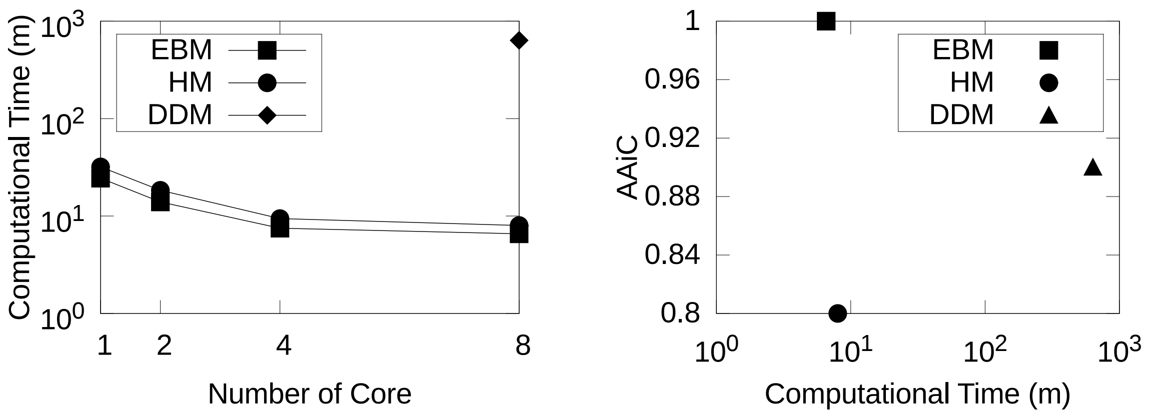


Figure 33: Computational Time Evaluation.

- *EBM* and *HM* have similar computational requirements, *HM* being just slightly slower with respect to *EBM*;
- *HM* provides clearly the best trade-off between accuracy and computational requirements.

In conclusion, *EBM* is the fastest method but, with a small additional computational effort with respect to *EBM*, *HM* is able to deliver a model which is extremely more accurate with respect to *EBM* and *DDM*.

2.4.7 Conclusions

In this work we dealt with the problem of understanding and predicting the train movements in Large-Scale Railway Networks. In particular, our purpose was to predict the *Running Time* of a train between two stations, the *Dwell Time* of a train in a station, the *Train Delay*, and the *Penalty Costs*, four important aspects which fully characterize the train movements and that were never studied together before. For this purpose, we proposed, for the first time, an hybrid approach which is able to merge together two approaches adopted in literature: the one which develops models based on the knowledge of the network and the experience of the operators and the one based on the analysis of the historical data about the network with advanced analytic methods. The result is a dynamic, interpretable, and robust hybrid data analytics system able to handle non recurrent events, changes in the behaviour of the network, and ability to consider complex and exogenous information like weather information. Basically, the proposed approach is able to take the strengths of the two original approaches and to limit their weaknesses. Results on real world data coming from the Italian railway network shown that the proposed solution outperforms both state-of-the-art experience and Data-Driven based systems.

2.5 Specific-Scenario 3: Restoration Time

2.5.1 Introduction

Every time an infrastructure asset is affected by a failure, it is clear that it will affect not only the single asset functional behaviour, but also the normal execution of railway operations.
Modern railway networks have to guarantee that security systems, trains and assets -and consequently maintenance operations- are able to provide an effective and efficient service to their customers.
The general aim is to study and prove the application of advanced visual and rule-based data analytics in the railway ecosystems.

A fully data-driven approach is here proposed for the solution of two similar problems in the scope of railway transportation system: predicting time to restoration for future maintenance actions (RFI scenario) and detecting the underlying functional dependencies and useful relationships between the function restoration time of an asset and various features describing its initial conditions (SR scenario).
The information outputted by the models can be very useful because it could be used by the traffic management system to reroute trains through safer paths, minimizing the risks of any problem.

2.5.1.1 Problem description

The general objective of this work is to study, design and develop data and visual analytics solutions for knowledge extraction from railway asset data. The two scenarios that will be presented cover the same aspect of the railway system -the time to restoration- and exploit similar data, but they differ in terms of aims:

- In the first scenario, a set of predictive models for forecasting purposes have been developed, based on both data provided by RFI about maintenance/repair actions and weather data;
- In the second scenario, different diagnostic models have been designed to capture, understand and visualize the knowledge enclosed into maintenance reports (provided by SR) and historical weather conditions, without predictive aims.

2.5.1.2 State of the art

Nowadays, vast amounts of data are being generated in many fields, including the railway one; data analytics and machine learning are tools able to collect, clean, process, analyze, and gain useful insights from them.
The task of data analytics comes in as there is a need to extract concise and possibly actionable insights from the available data for application-specific goals: the raw data may be arbitrary, unstructured, or even in a format that is not immediately suitable to be processed by an automated computer program; to address this issue, data analysts use different processing techniques to collect, clean and transform raw data into standardized formats.
Applications of data analytics are often closely connected to supervised learning problems. The concept of machine learning was born in the Nineties. It can be seen as a union of Artificial Intelligence, Computational Intelligence and Statistics [83]. In this field a lot of algorithms for supervised and unsupervised learning were born, e.g. Support Vector Machine, Neural Networks, Decision Trees and Statistical Pattern Recognition.
The learning process can be viewed as the process of generating automatically new knowledge from past experience/data [56]. In this context, the term “supervised” indicates the presence of the outcome variable to guide the learning process. Typically, the development of a machine learning algorithm follows two or three fundamental phases: training, validation (when the learner is parametric) and testing. These phases correspond, in their more basic applications, to so-called data “sets”, i.e. subsections of the original data: training set, validation set and test set. A typical example of supervised learning is the regression case: an outcome measurement is provided and the objective is to predict it based on a set of known features; the learner observes the outcome and the feature measurements for a set of objects in the historical data (training) and, using this “experience”, a model is built to predict the outcome for new unseen objects (in the test set).
Learners can be black-box -i.e. they hide the algorithm’s details from the user and just allow parameter adjustment, so that “the algorithm learns whatever it learns” [5]- or white-box -the algorithm’s structure is revealed. A good white-box learner can be defined as the one that can accurately predict the outcome based on the problem specific metrics and can be easily interpreted.
In recent times, as public agencies responsible for areas such as criminal justice, health and welfare are increasingly using algorithms and software to steer or make decisions on life-changing events, research insti-

tutes are focusing more and more on the social implications of artificial intelligence; the debate around how these systems can be “opaque” seems to be of greater importance than ever. In some cases, algorithms are now capable of accomplishing almost incredible tasks and handling an unfathomably complex world better than a human can, but, exactly because they can, the way they work risks to become unfathomable too. For these reasons, the employment of white box algorithms is also assuming ethical connotations [1].
In the scope of railway systems, machine learning is often synonymous with Condition Based Maintenance (CBM) and Predictive Maintenance (PM), other than Traffic Management System (TMS). Thanks to the rapidly expanding scale of manufacturing and asset maintenance industries, they are now adapting to the wider applications of advanced algorithms on consumer generated big data [3].
In fact, according to [90], maintenance of railway assets is not regarded anymore as something that needs to be done, but more and more as a professional business delivering very important products for rail operations:

- Availability: the time that the infrastructure is available for operations per calendar period. A part of the time the infrastructure is out of service due to planned preventive maintenance actions. Another part of the time the infrastructure can be unavailable due to infrastructure failures (corrective maintenance), possession over-runs or external factors, such as vandalism and bad weather;
- Reliability: the time that the infrastructure is available for operations during the periods agreed. In other words, with regard to the reliability, only the unplanned maintenance and repair are considered. The reliability depends on e.g. the asset quality and maintainability, the amount of preventive maintenance, and the failure restoration times.

Various data analytics applications have been developed to guide maintenance planning: data-driven decision support models have been built either to study maintenance performance [60] and to achieve the availability target in both the scheduled and the condition based maintenance regimes [65].
Maintenance issues are also strictly connected with speed of line-haul movement between terminals: among many determinants of overall network velocity, a key driver is service interruption, including lowered operating speed due to track conditions [50].
Using huge volumes of historical detector data, in combination with failure data, maintenance action data, train type data and weather data, several analytical approaches have been explored, including correlation analysis, causal analysis, time series analysis and machine learning techniques to automatically learn rules and build failure prediction models.
Additionally, the analytics and models can also be used with diagnostic intent -as in the second scenario of this work- detecting the root cause of several failure modes, which can be proactively used by maintenance organizations to optimize trade-offs related to maintenance schedule, costs and shop capacity.
Arjen Zoeteman, researcher in Life Cycle Costing for Rail Infrastructures at TU Delft, in 2001 pointed out that available data about failures in rail systems were of low quality [90]. Failure types were missing, failure frequencies, restore times and speed restrictions were not consistent. However, his efforts in improving the performance of railway systems using data led to the recognition of the importance of the subject and helped in opening a dialogue between the construction and maintenance staff.
Interest in predicting time to restoration resulted in a need for actionable and tidy data about repair actions -not only in railway scope- so that in the last decade a literature about maintenance and repair times for components in technological facilities was born [16].
Indeed, information about restoration time is crucial not just regarding infrastructures. For instance, greater accuracy in predicting the time needed to repair software defects would be useful for devising better testing plans, schedules, and for allocation of testing resources, helping in keeping projects on time and within budget [36].

2.5.1.3 Proposed approach

As the objective of the previously mentioned EU funded projects is the development of tools and methodologies aiming at extracting knowledge from data analytics algorithms and at the same time making them interpretable in an easier way, white-box (and grey-box) learning algorithms are the best choice for this work. In particular, two algorithms have been employed: Decision Trees and Random Forests.

2.5.1.3.1 Decision Trees

Decision trees are often the best way to understand the main functional dependencies between the variables in the first instance.
Decision trees are trees where each node represents a feature (or attribute), each link (or branch) represents a decision (or rule) and each leaf represents an outcome.
Other than being white-box and nonparametric, they tend to mimic the human thinking so they make it quite simple to understand the data and produce some good visual interpretations -especially useful in SR scenario.
Tree models where the target variable can take a discrete and limited set of values are called classification trees; in these tree structures, leaves represent class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.
Building a decision tree is especially a matter of choosing which attribute to test at each node.
In classification trees, a measure of information gain can be defined for this purpose, using Entropy or Gini Impurity.
Given a binary categorization, C, and a set of examples, S, for which the proportion of examples labelled 0 by C is p_0 and the proportion of examples categorized as 1 by C is p_1, then the entropy of S is:

$$Entropy(S) = -p_0 log_2(p_0) - p_1 log_2(p_1) \tag{2}$$

Therefore, given an arbitrary categorization C into categories c_1, ..., c_n, and a set of examples, S, for which the proportion of examples in c_i is p_i, then the entropy of S is:

$$Entropy(S) = \sum_{i=1}^{n} -p_i log_2(p_i) \tag{3}$$

In simple words, entropy characterizes the (im)purity of an arbitrary collection of examples. [56] Please note that, when p gets close to zero (i.e., the category has only a few examples in it), then the $log_2(p)$ becomes a big negative number, but the p part dominates the calculation, so entropy is nearly zero. As entropy calculates the disorder in the data, this low score is good and it reflects the intent to reward categories with few examples in. Similarly, if p gets close to 1 (i.e., the category has most of the examples in), then the $log_2(p)$ part gets very close to zero and so the overall value. Hence we see that both when the category is nearly (or completely) empty, or when the category nearly (or completely) contains all the examples, the score for the category gets close to zero.
Intuitively, the Gini Impurity can be understood as a criterion to minimize the probability of misclassification:

$$GiniImpurity(S) = \sum_{i=1}^{n} p_i(1 - p_i) \tag{4}$$

Similar to the Entropy, the Gini Impurity is maximal if the classes are perfectly mixed. In practice, Gini Impurity and Entropy typically produce very similar results.
The information gain of an attribute A can be seen as the expected reduction in impurity caused by knowing the value of the attribute. Given a collection of examples, S, information gain is calculated as:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{5}$$

or

$$Gain(S, A) = GiniImpurity(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GiniImpurity(S_v) \tag{6}$$

So, defined these measures, the algorithm goes as follows: it chooses the root node to be the attribute, A, which scores the highest for information gain relative to S; for each value v that A can possibly take, it draws a branch from the node; for each branch from A corresponding to value v, calculates S_v. Then, if S_v contains only examples from a category c, then the algorithm puts c as the leaf node category which ends that branch. Otherwise, it puts a new node in the decision tree, where the new attribute being tested in the node is the one which scores highest for information gain relative to S_v. The algorithm terminates either when all the attributes have been exhausted, or the decision tree perfectly classifies the examples.
Regression trees work in a similar way: the basic idea is again measuring impurity and reducing it. While Entropy does not suit regression problems, Gini Impurity is the most common splitting criteria for regression trees -and the one used in this work.
Different measures of disorder could be used in quantitative cases, e.g. MSE (mean squared error). Given a quantitative (response) variable Y:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{Y}_i - Y_i)^2 \tag{7}$$

MSE is one of the measures used in this work to evaluate models.
Decision trees suffer from overfitting, because they are trained to stop when they have perfectly classified all the training data, i.e., each branch is extended far enough to correctly categorize each example. To avoid this problem, the most popular approach is post-pruning some of the branches from the complete tree. Therefore, it is clear that pruning raises the issue of determining the correct tree size. On the other hand, decision tree learning is robust to errors in the data: it will function well in the light of errors in the classification instances provided or missing values for certain attributes for certain examples.
Given the characteristics of the two scenarios in this work, decision trees are a good method for these learning tasks. In both RFI and SR cases, the outcome -repair time- is a quantitative variable, so the kind of tree used is the regression one.

2.5.1.3.2 Random Forests

The natural next step after Decision Trees is Random Forest, an ensemble algorithm which, in simple words, builds multiple Decision Trees and merges them together to get a more accurate and stable prediction.
The general idea behind ensemble algorithms is that a combination of learning models increases the overall result. In fact, Random Forests overcome the limitations of single trees, first of all the problem of overfitting. In random forests, tree predictors are combined such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. Thanks to these properties, error for forests converges to a limit as the number of trees in the forest becomes large [14].
To inject randomness into the ensemble algorithm, different techniques can be used. An example is bagging [13], where to grow each tree a random selection (without replacement) is made from the examples in the training set. Another example is random split selection [21] where at each node the split is selected at random from among the K best splits.

In classification cases, each tree casts a unit vote for the most popular class, which represents the output; in regression problems, numeric results are calculated by each tree and the output value is either the mean, the median or the mode of them, depending on the algorithm.
In many applications random forests produce very satisfying results and have important advantages over other techniques in terms of ability to handle highly non-linear data and robustness to noise; furthermore, one of the most interesting points boil down to the interpretability of random forests in the eyes of variable importance measures.
Random Forest -as implemented in the R package used in this work- provides two different importance measures, mean decrease accuracy (MDA) and mean decrease Gini (MDG), that can be used for ranking variables and for variable selection. MDA quantifies the importance of a variable by measuring the change in prediction accuracy when the values of the variable are randomly permuted compared to the original observations. MDG is the sum of all decreases in Gini impurity due to a given variable (when this variable is used to form a split in the random forest).
For this reasons, in RFI scenario (the predictive one) Random Forests were particularly useful, though they can't be considered completely white box.

2.5.1.3.3 Error Estimation

In this work, both K-Fold Cross Validation and Out-Of-Bag techniques have been used to estimate error.
In K-Fold Cross-Validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single one is retained as validation data for testing the model, and the remaining $k-1$ subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as validation data. The k results can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-sampling is that all observations are used for both training and validation, and each observation is used for validation exactly once. 10-fold cross-validation is commonly used [55], but in general k remains an unfixed parameter.
In random forest algorithms, when using bagging, each tree is grown on a new training set, which is drawn, with replacement, from the original training set. Bagging, other than enhancing accuracy, can be used to give ongoing estimates of the error of the combined ensemble of trees. These estimates are done out-of-bag (OOB): each observation x_i is used to test only the aggregations of trees which were built on training sets not containing x_i. The study of error estimates for bagged classifiers gives empirical evidence to show that the out-of-bag estimate is as accurate as using a test set of the same size as the training set [14].

2.5.2 RFI - Rete Ferroviaria Italiana

The main objective of this scenario is to estimate the time to restoration for future planned and urgent maintenance operations, based on historical data about repair actions in Liguria -provided by RFI- and weather data -provided by Regione Liguria.
The predictive models that will be designed will be able to exploit the knowledge enclosed into maintenance reports so to predict the time needed to complete an action over an asset in order to restore its functional status. Moreover, historical weather conditions data will be included in the analyses in order to take into account the atmospheric factors affecting railway maintenance and repair operations (e.g. temperature, rainfall, solar radiation, wind intensity).

Name	Data Type
Type	Factor 'P'= Planned Maintenance 'S'= Extraordinary Maintenance
Condition	Factor 'C'= Allowed 'NC'= Not Allowed 'Rn'= Renounced 'R'= Requested
Beginning Station	Factor: 69 levels
Included/Excluded	Factor 'I'= Included 'E'= Excluded
End Station	Factor: 66 levels
Included/Excluded	Factor 'I'= Included 'E'= Excluded
Track	Factor 'D'= Left 'P'= Right 'L'= Entire Line
Planned Beginning Time	Chron (planned works only)
Planned End Time	Chron (planned works only)
Planned Repair Time	Chron (planned works only)
Actual Beginning Time	Chron
Actual End Time	Chron
Actual Repair Time	Chron

Table 10: Structure of the initial dataset

2.5.2.1 Data description and basic statistics

2.5.2.1.1 Historical data

The analysis will consider reports of maintenance works from 01-06-2017 to 31-01-2018. There are 4945 observations and 13 variables. This initial structure of the dataset can be seen in Table 10.
From now on "Repair time" will indicate the numeric response variable -unit of measurement: minute-, derived from the already existing "Actual repair time". As shown in the statistics and in the figure below (Figure 34 and Table 11), there are few big outliers in the distribution of the variable. For this reason, observations with a value greater than 540 minutes (the highest datum still within 1.5 interquartile range of the upper quartile) have not been considered (129 observations out of 4945).

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	33	82	247.5	201	83160

Table 11: Summary of Repair time

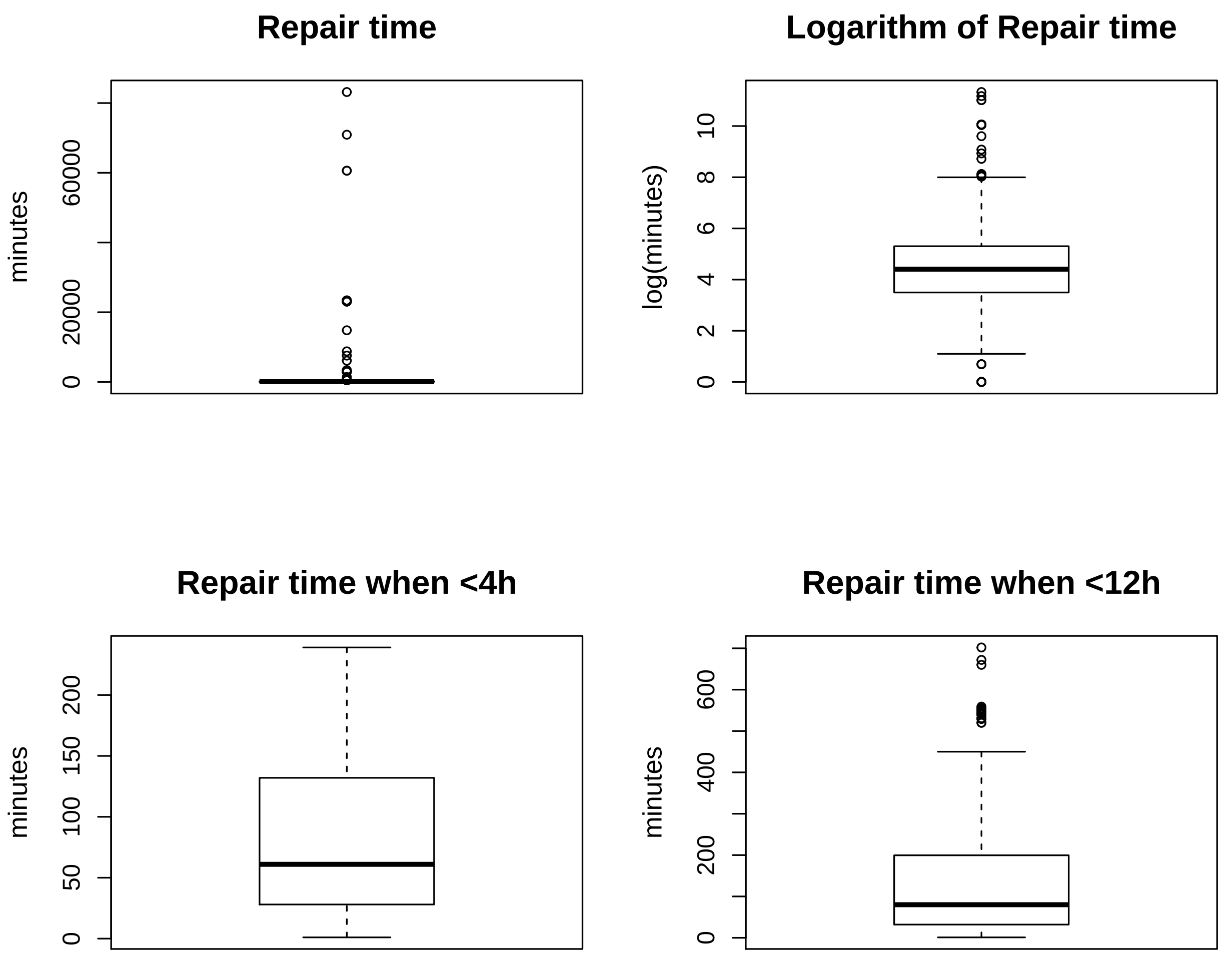


Figure 34: Distribution of Repair time.

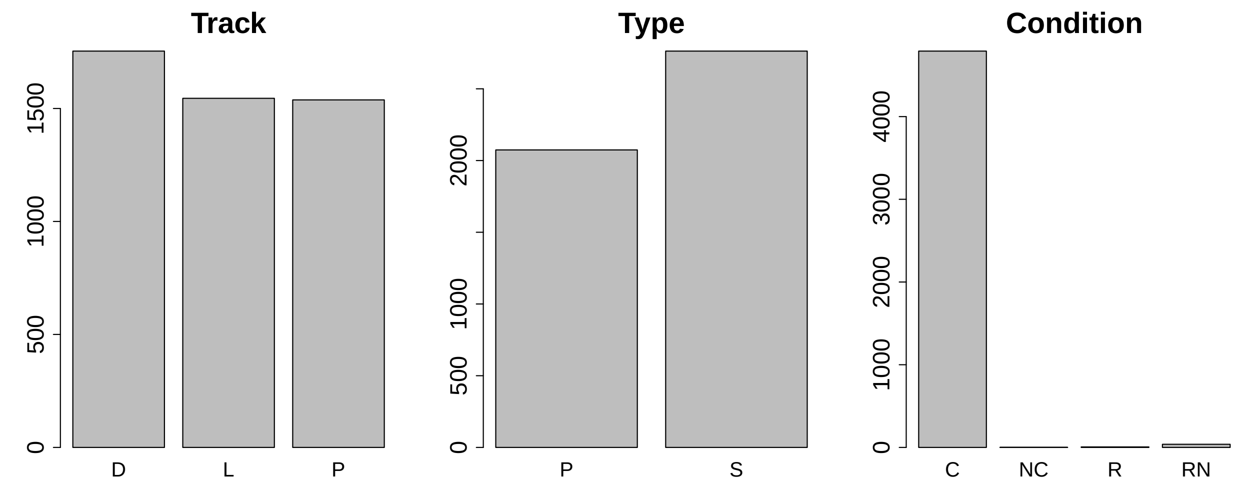


Figure 35: Barplots of cathegorical variables Type, Track and Condition

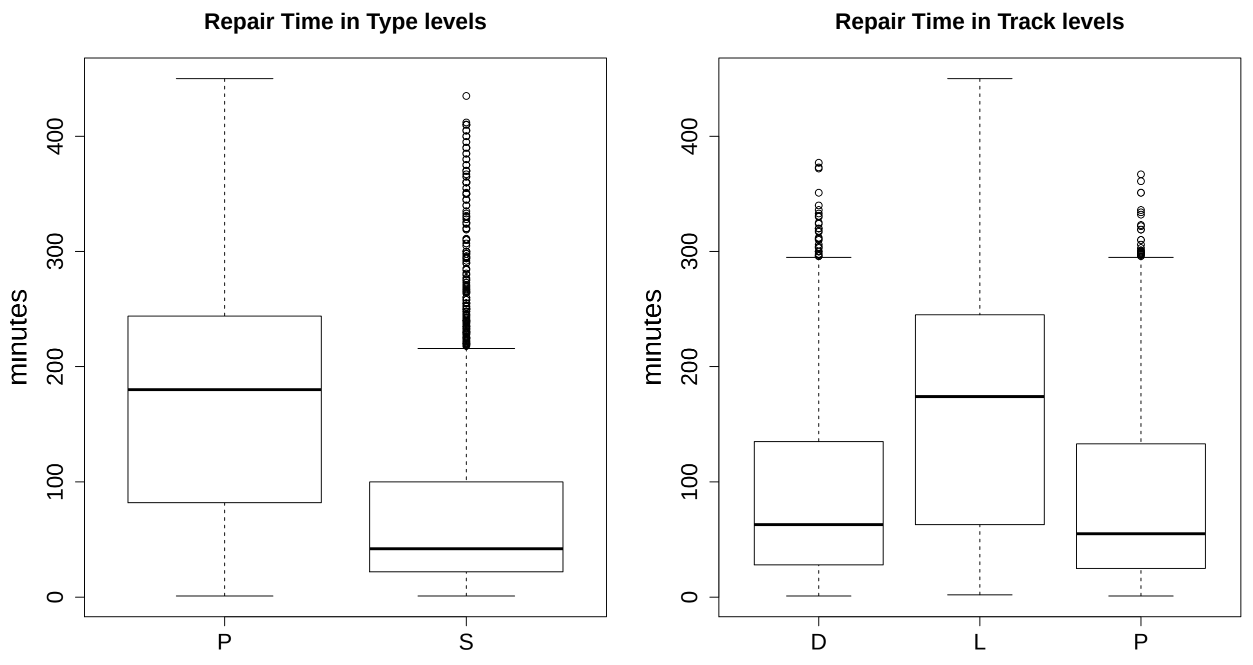


Figure 36: Repair Time in different levels of the variables Type and Track

Briefly looking at the most relevant categorical variables, in Figure 35 it can be seen that Track and Type have quite an even distribution; almost every maintenance work, instead, is labelled as Allowed.
The distribution of Repair Time in the levels of Type and Track (Figure 36) shows that the response variable is not statistically independent from both the factors: planned works seem to have longer duration on average than the extraordinary ones, and, as would be expected, when maintenance involves a single track it is usually shorter than on the entire line. Not surprisingly, the distinction between right and left track doesn't seem to be effective on repair time.
Eventually, about the variable Planned repair time, it is interesting to note that the correlation with the actual one is 0.91, so it can be considered very predictive.

2.5.2.1.2 Extraction of other relevant features

It is possible to extract some new features from the original ones; it will be considered, especially:

- The day of the week of the beginning date;
- The hour of the day of the beginning date;
- The province of the beginning station.

While, at a first glance, the day of the week seems relevant just concerning the distinction between Sunday and other days (first graph in Figure 37), Hour of the day and Province appear to be quite influential in the variations of the response variable (Figure 37 and 38).
Please note that, as partially mentioned, most of the stations are in Liguria (Genova, Savona, Imperia) and the few whose province is Alessandria (Piedmont) are actually very near to Liguria's border.

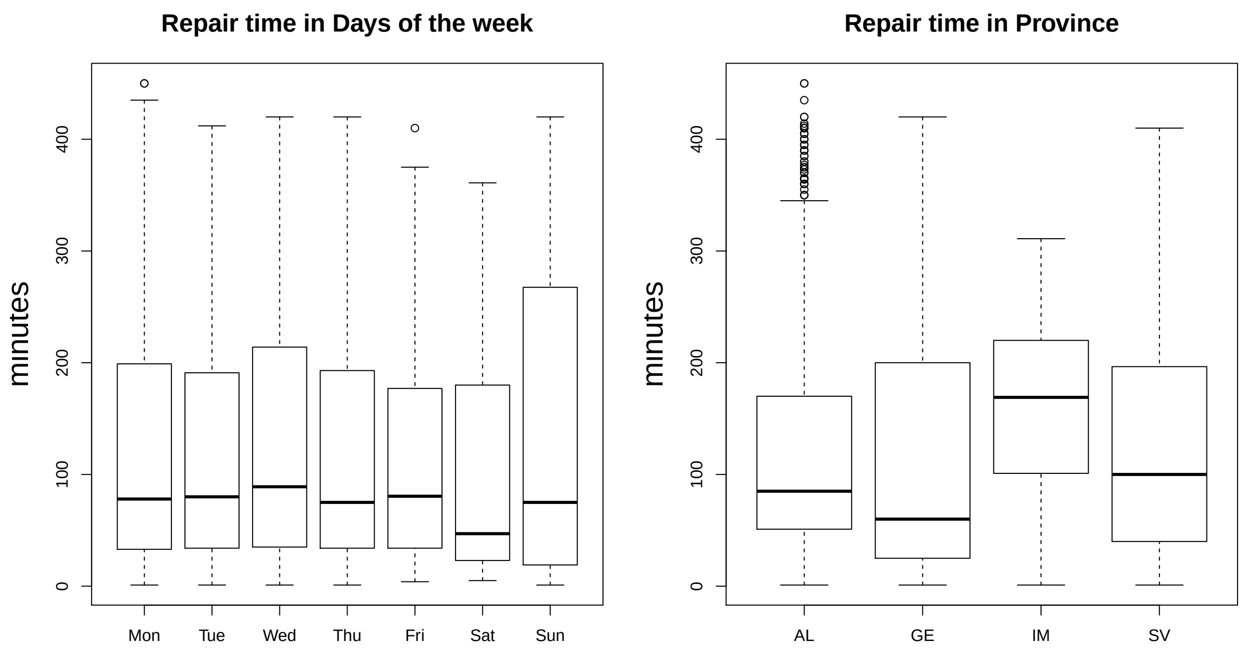


Figure 37: Repair Time in different levels of the variables Day of the week and Province.

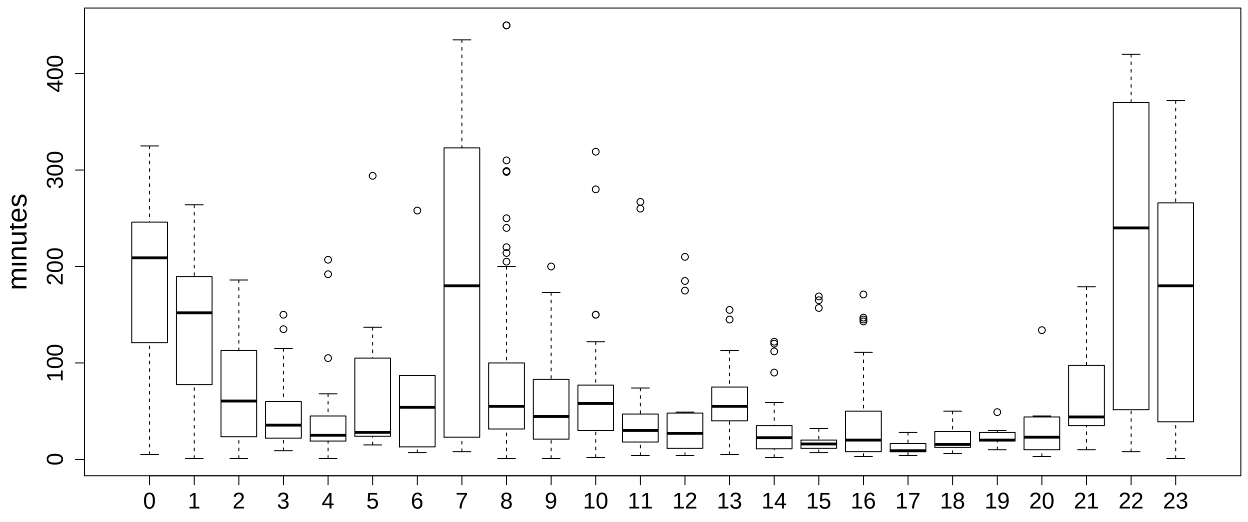


Figure 38: Repair Time in different levels of the variable Hour of the day.

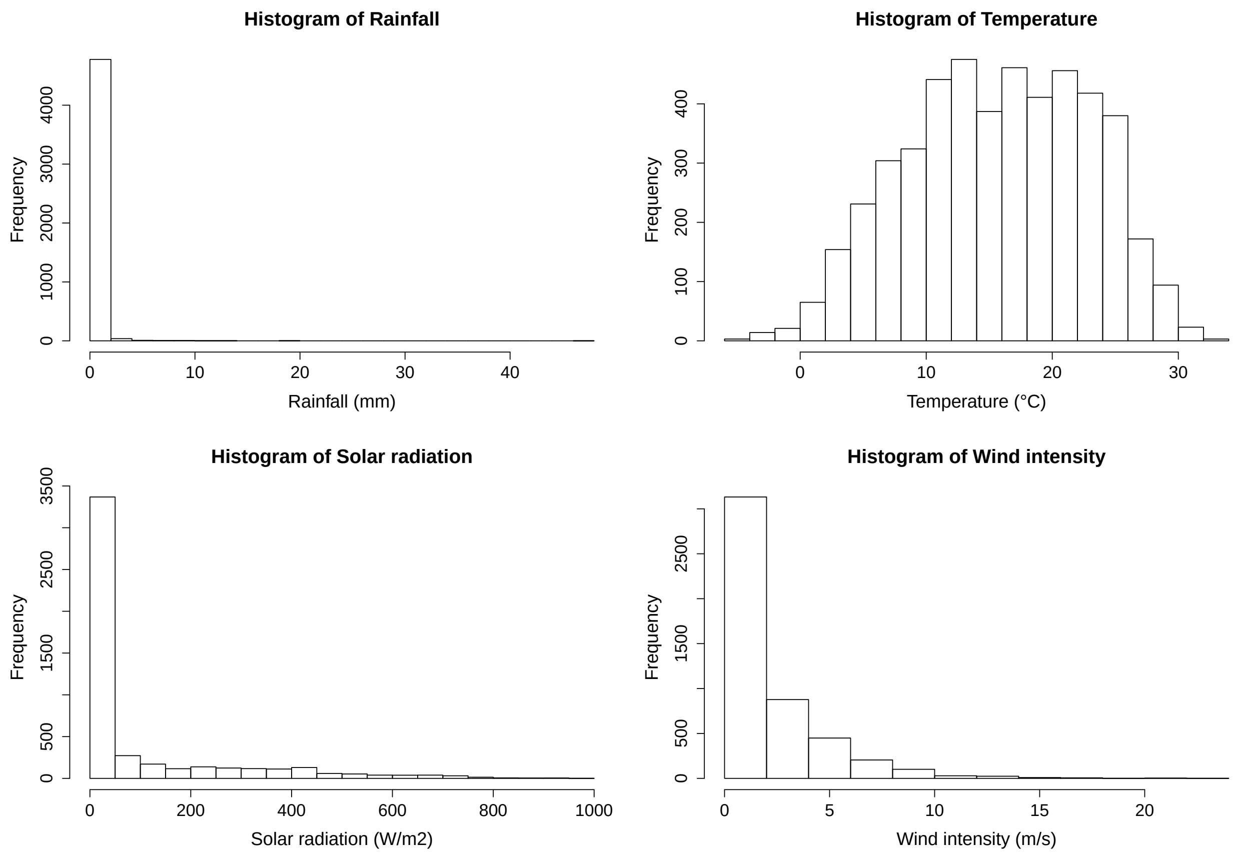


Figure 39: Distribution of weather variables

2.5.2.1.3 Weather data

Historical weather data have been retrieved from Regione Liguria - Meteorological Service website. For each month, 4 datasets -one for each variable: rainfall, temperature, solar radiation and wind intensity- representing hourly weather measurements have been considered.
The datasets include information coming from different weather stations all over Liguria. As previously mentioned, the geographical locations of the railway stations have been divided into provinces; for every province, a weather station has then been chosen, in order to associate historical maintenance data with corresponding weather conditions.
All of the weather data collected by Regione Liguria is archived by the UTC time. So, to convert it into local time, it has been shifted one hour forward (or two when DST is observed).
As can be seen in Figure 39, the distribution of Rainfall, Wind intensity and Solar radiation is concentrated around lowest values -in the Solar radiation case, this is due to the fact that during the night this value is always 0. Temperature, instead, shows a more homogeneous distribution.
The effect of these variables on repair time is not immediately visible from the data: for instance, the correlation between Solar radiation and Repair time is about -0.35, while Rainfall and Temperature seem to be almost uncorrelated with the response. Despite this fact, these features could still be influential in a non-linear way, as can be partially seen in Figure 40.

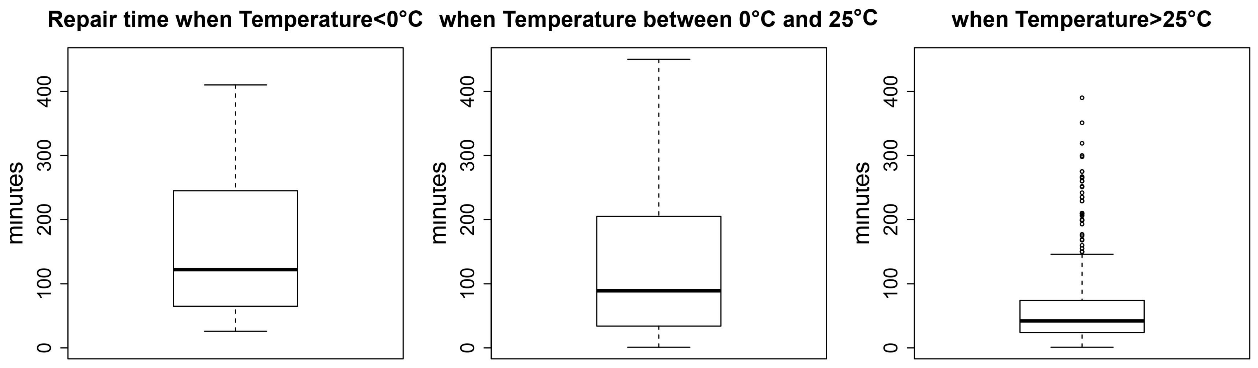


Figure 40: Distribution of Repair time in different subsets of the variable Temperature

2.5.2.1.4 The structure of the final dataset

Now the dataset has 23 variables and 4837 observations. The structure of the final dataset can be seen in Table 12.
In the modeling phase all these variables will be used except Beginning and End Time (both Planned and Actual) as well as Beginning and End station, whose amounts of levels are too large.

2.5.2.2 Regression trees

The first models considered are regression trees. Decision tree algorithms are white-box, nonparametric and, therefore, make no assumptions regarding the distribution of input data, so they can be one of the easiest ways to understand the problem in the first instance.

2.5.2.2.1 Planned maintenance

The table and figures below (Table 13 and Figures 41 to 43) show some interesting results about an exploratory regression tree trained with data referring to planned works (labelled as ‘P’ in the Type variable). The dataset has been divided into a training set -75% of data- and a test set -25% of data.
The feature importance ranking can be seen in Table 13; it is calculated using mean decrease Gini (also called mean decrease in node impurity). The most relevant variable is, as would be expected, Planned Repair Time. It can also be seen from the pruned tree (Figure 42 and 43): the feature Planned Repair Time is used in a large amount of splits, and branches with lower values of planned repair time always correspond to lower values of actual repair time in the leaves. Other important features are Solar Radiation, Hour of the Day, Province, Track and Temperature; these results point out the importance of a thorough extraction and addition of features of interest.
Figure 41 shows how the cross-validated estimate of relative error -i.e., in this case, $1 - R^2$- changes due to the changes of size of the tree. This plot suggested that the size could be reduced to about 15 leaves: a good choice for pruning is often the leftmost value for which the mean lies below the horizontal dashed line, which represents the highest cross-validated error minus the minimum cross-validated error, plus the standard deviation of the error. On the lower axis, ‘cp’ stands for Complexity Parameter of the tree, a measure of the amount by which further splitting would improve the relative error.
Pruning decision trees reduces the complexity of the final algorithm, hence improving predictive accuracy by the reduction of overfitting.

Name	Data Type	Source
Type	Factor 'P'= Planned Maintenance 'S'= Extraordinary Maintenance	RFI
Condition	Factor 'C'= Allowed 'NC'= Not Allowed 'Rn'= Renounced 'R'= Requested	RFI
Beginning Station	Factor: 69 levels	RFI
Included/Excluded	Factor 'I'= Included 'E'= Excluded	RFI
End Station	Factor: 66 levels	RFI
Included/Excluded	Factor	RFI
Track	Factor 'D'= Left 'P'= Right 'L'= Entire Line	RFI
Planned Beginning Time	Chron (planned works only)	RFI
Planned End Time	Chron (planned works only)	RFI
Planned Repair Time	Chron (planned works only)	RFI
Planned Repair Time - numeric	Integer (planned works only)	Derived
Actual Beginning Time	Chron	RFI
Day of the Week (beginning time)	Factor: 7 levels	Derived
Hour of the Day (beginning time)	Integer: 1,2...23,24	Derived
Month of the Year (beginning time)	Factor: 8 levels	Derived
Province (beginning station)	Factor: 'GE','SV','IM','AL'	Derived
Rainfall (beginning time)	Numeric	Regione Liguria
Temperature (beginning time)	Numeric	Regione Liguria
Solar Radiation (beginning time)	Numeric	Regione Liguria
Wind Intensity (beginning time)	Numeric	Regione Liguria
Actual End Time	Chron	RFI
Actual Repair Time	Chron	RFI
Actual Repair Time - numeric	Integer	Derived

Table 12: Structure of the final dataset

Variable	Mean decrease in node impurity
Planned Repair Time - Numeric	44
Solar Radiation	19
Hour of the Day	14
Province	8
Track	5
Temperature	4
Actual Beginning Time	2
Month of the Year	1
Day of the week	1
Wind Intensity	1

Table 13: Variable Importance

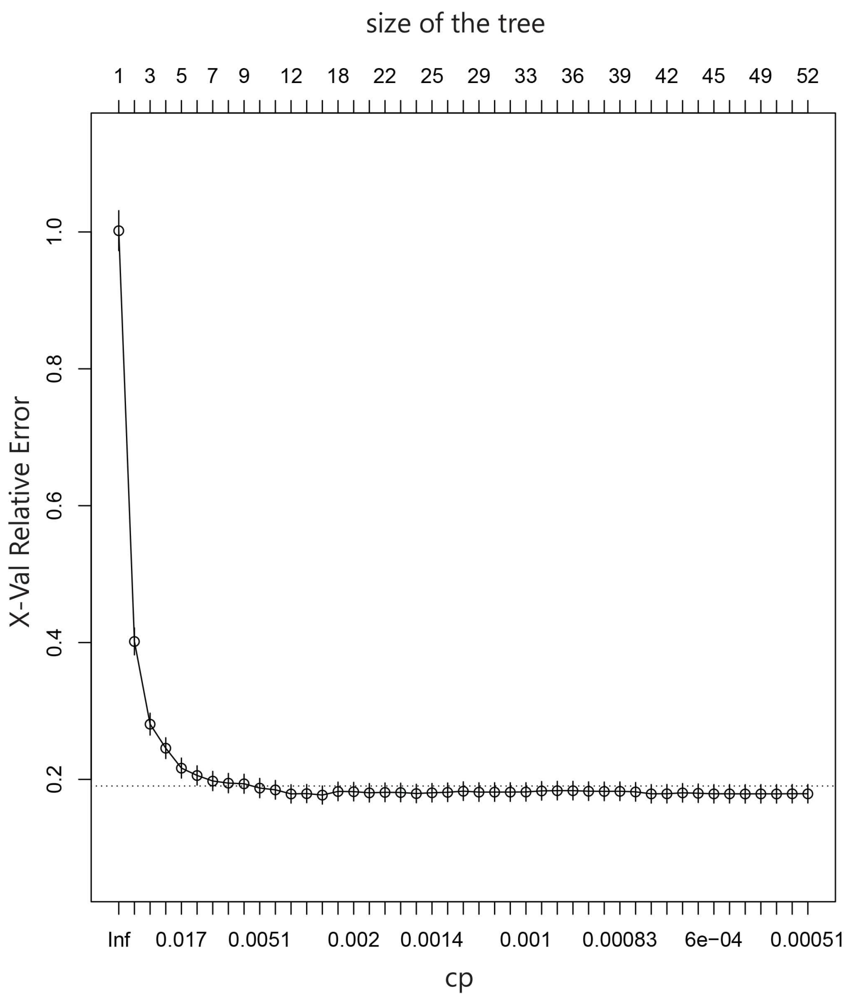


Figure 41: Cross-validation results

```
Median of absolute percentage error: 14%

Mean of absolute percentage error: 42%
```

Table 14: Mean and Median of Absolute Percentage Error

Please note that in this case R^2 -here estimated about 0.8- is a measure of how well the model fits the data on which it is built -not of how well it would perform on an independent test set. In any statistical model, including conventional regressions, R^2 is an overly optimistic prediction of model performance on previously unseen data.
The scatter plot in Figure 44, instead, shows how the model performs on the test set (Figure 44). As the dots show a clear uphill pattern moving from left to right and are concentrated around the diagonal red line, it is possible to say that the model captures the existence of some information inside data.
As the probability of having a certain error decreases as the error increases, a similar conclusion can be drawn by looking at Figure 45. In fact, the graph is a histogram showing the distribution of the absolute percentage error, which is defined as the absolute difference between the true values and the predicted values as a percentage of the true values. This measure expresses how close the estimates are to the real values. In particular, the histogram includes the values of percentage error on the x-axis, and the frequency of that particular value of percentage error on the y-axis.
For a better visualization the biggest outliers (30 out of 504 values of absolute percentage error are greater than 100%) have not been reported in this graph.
Eventually, in Table 14 mean and median of absolute percentage error can be seen. Please note that median is a more robust measure when big outliers are involved in calculation.

2.5.2.2.2 Extraordinary works

While the ideal size of the tree (Figure 46) is not really different from the one of the previous model, feature importance and predictive accuracy vary substantially when works are not planned and the time to restoration is not previously scheduled.
Table 15 suggests that the most influential features are Track and Hour of the Day, followed by Province, Wind Intensity and other variables relative to the actual beginning time and the weather conditions. The role of these features can be seen in detail in Figure 48 and 49.
Even if the histogram (Figure 47) utterly suggests that some information has been captured, both the scatter plot (Figure 50) and the mean/median of absolute percentage error (Table 16) show that predictive accuracy is not satisfying.

2.5.2.3 Random Forests

As previously mentioned, Random Forests are an ensemble learning method, that operate by constructing a multitude of decision trees at training time and outputting -in the regression case- the mean prediction of the individual trees. Random forests are usually more accurate than single trees and, furthermore, correct for decision trees' habit of overfitting.
In first instance, two random forest models have been built to predict repair time, one for planned and one for extraordinary works. As for the previous modeling, both datasets have been divided into a training set (75% of data) and a test set (25% of data).

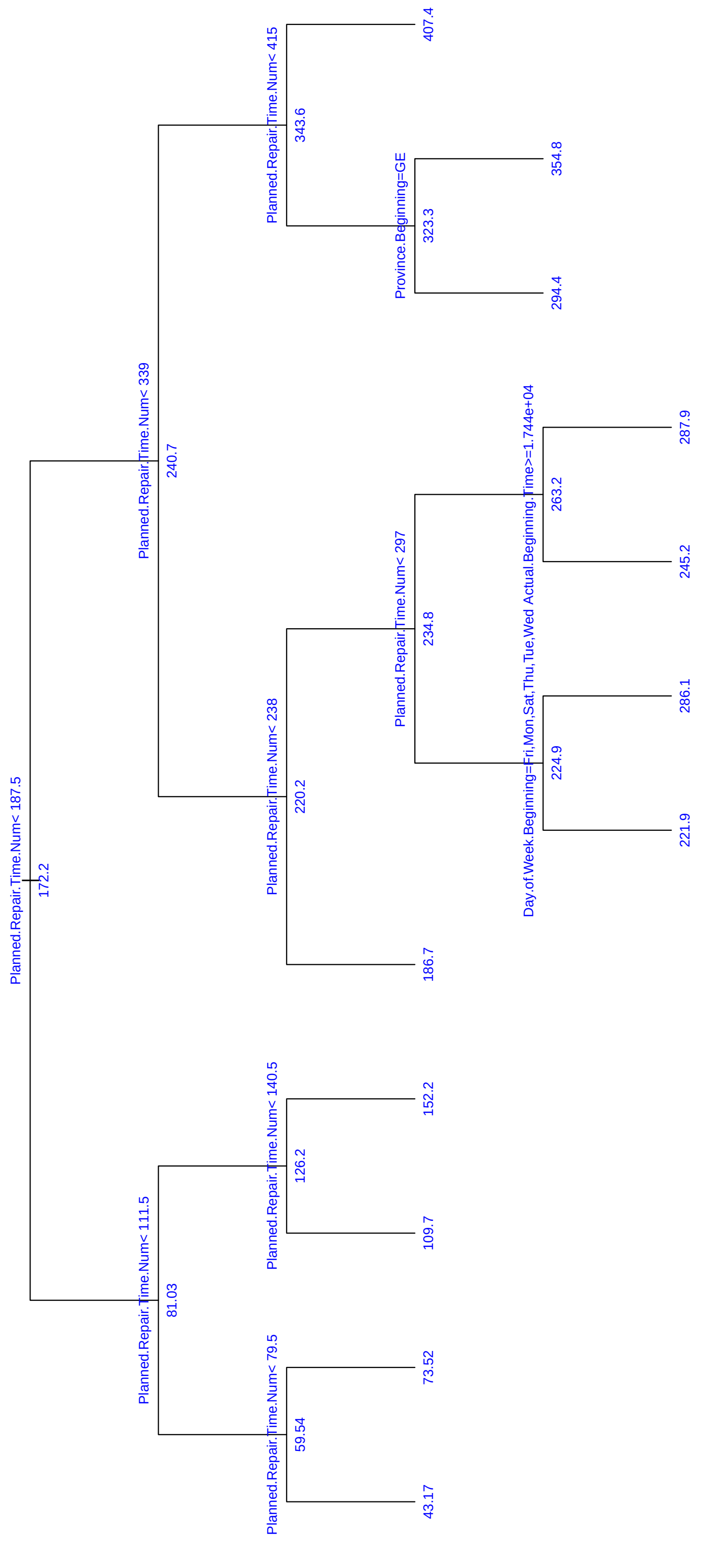

Planned.Repair.Time.Num< 187.5
172.2
Planned.Repair.Time.Num< 111.5
81.03
Planned.Repair.Time.Num< 79.5
59.54
43.17
73.52
Planned.Repair.Time.Num< 140.5
126.2
109.7
152.2
Planned.Repair.Time.Num< 339
240.7
Planned.Repair.Time.Num< 238
220.2
186.7
Planned.Repair.Time.Num< 297
234.8
Day.of.Week.Beginning=Fri,Mon,Sat,Thu,Tue,Wed
224.9
221.9
286.1
Actual.Beginning.Time>=1.744e+04
263.2
245.2
287.9
Planned.Repair.Time.Num< 415
343.6
Province.Beginning=GE
323.3
294.4
354.8
407.4

Figure 42: Pruned Tree for planned maintenance

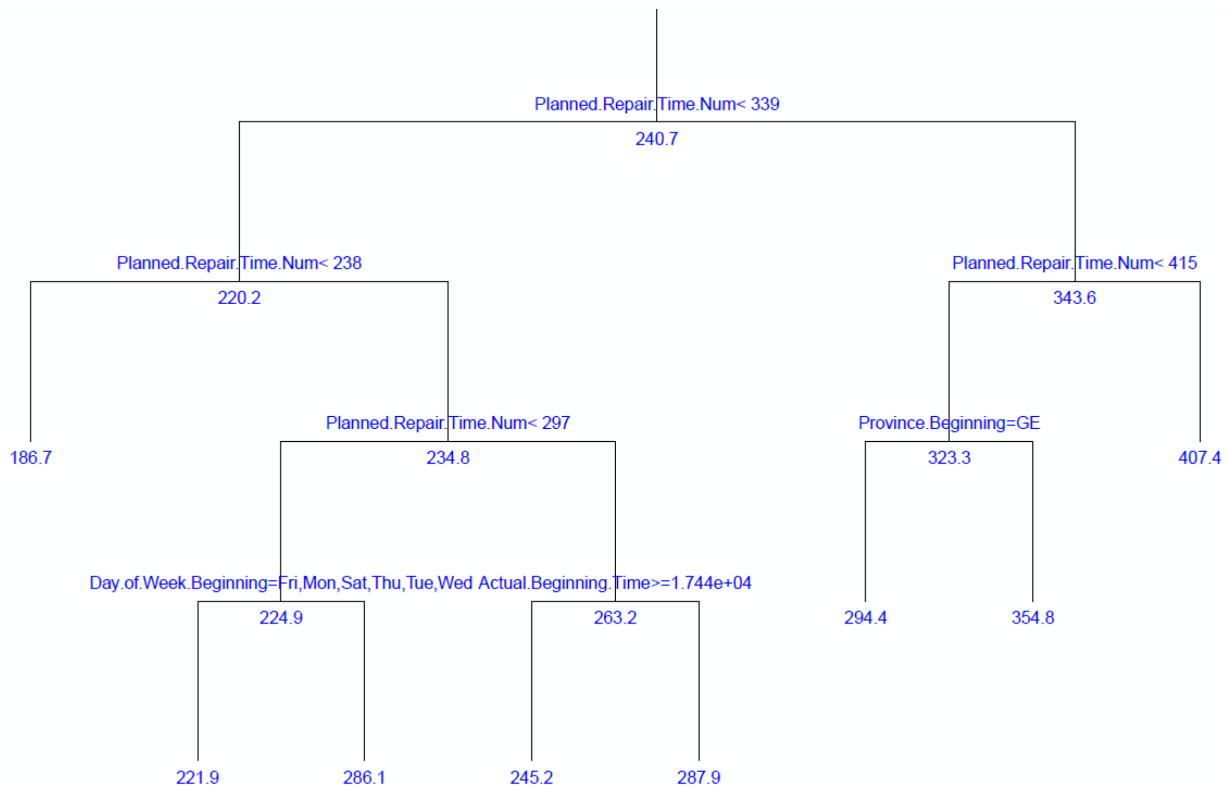

Planned.Repair.Time.Num< 339
240.7
Planned.Repair.Time.Num< 238
220.2
Planned.Repair.Time.Num< 415
343.6
186.7
Planned.Repair.Time.Num< 297
234.8
Province.Beginning=GE
323.3
407.4
Day.of.Week.Beginning=Fri,Mon,Sat,Thu,Tue,Wed
224.9
Actual.Beginning.Time>=1.744e+04
263.2
294.4
354.8
221.9
286.1
245.2
287.9

Figure 43: Detail of Pruned Tree for planned maintenance

Variable	Mean decrease in node impurity
Track	23
Hour of the Day	23
Province	14
Wind Intensity	10
Month of the Year	7
Actual Beginning Time	6
Day of the weeek	6
Temperature	5
Solar Radiation	3
Rainfall	1
Wind Intensity	1

Table 15: Variable Importance

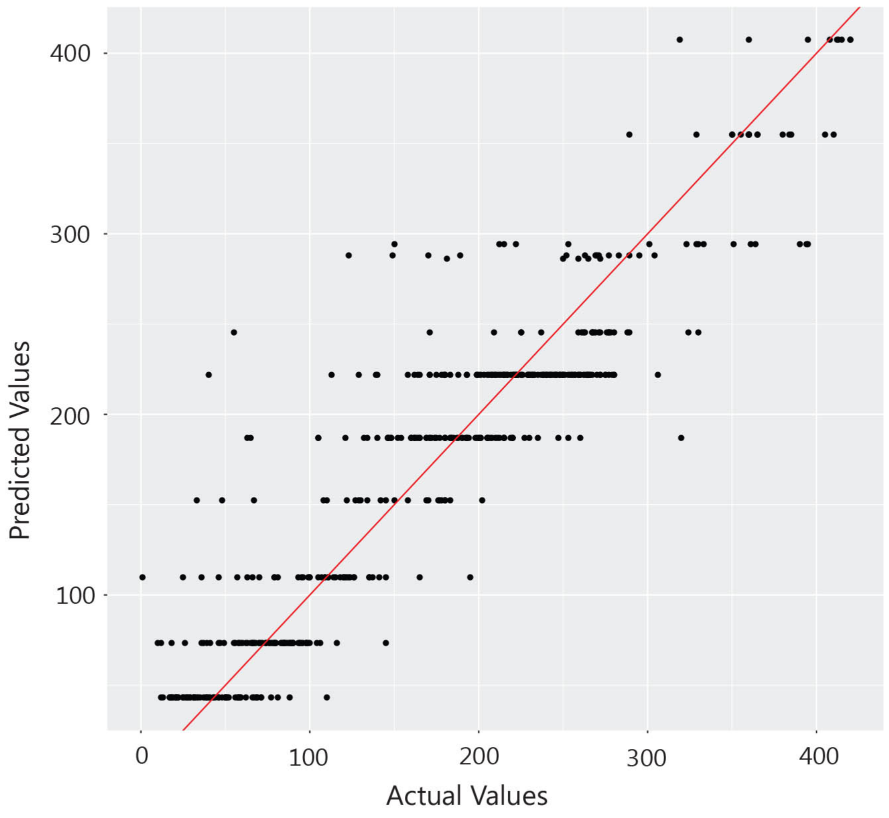


Figure 44: Scatter plot of actual vs predicted values (test set)

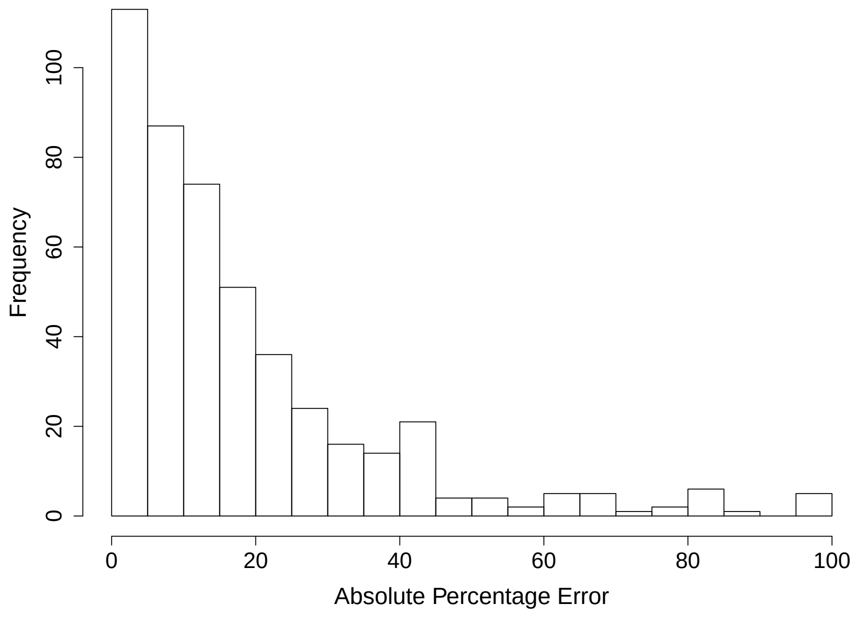


Figure 45: Histogram of absolute percentage error

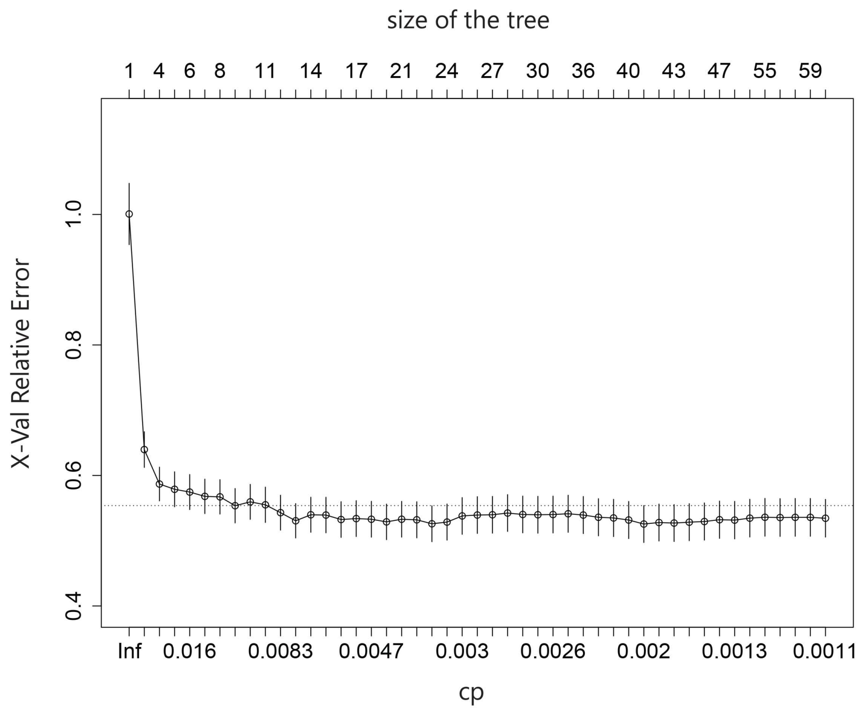


Figure 46: Cross-validation results

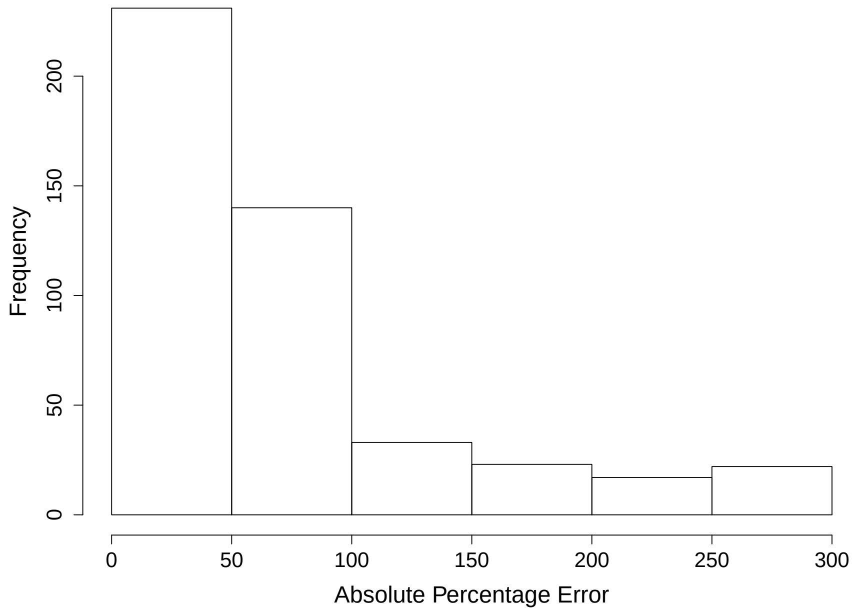


Figure 47: Histogram of absolute percentage error

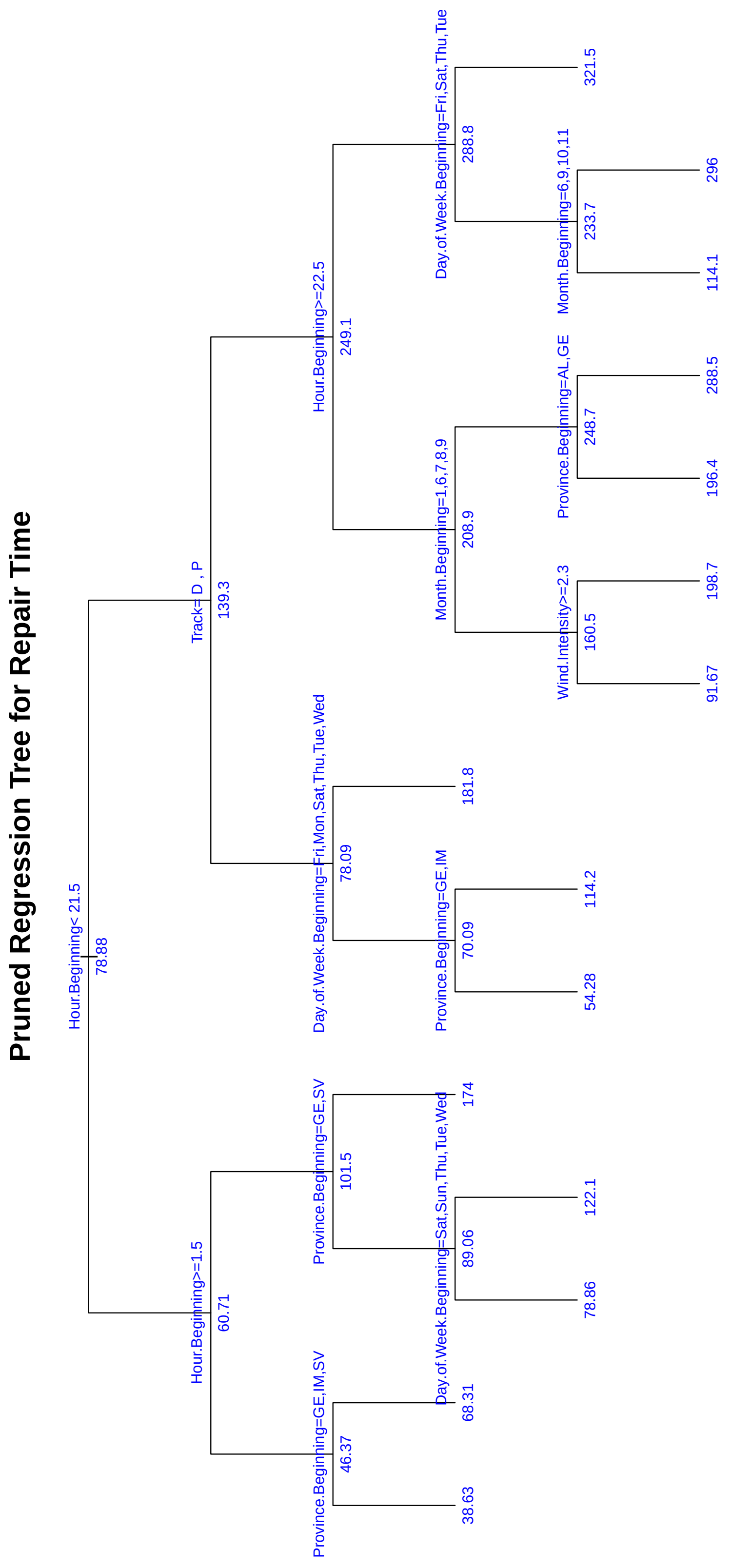


Figure 48: Pruned Tree for extraordinary works

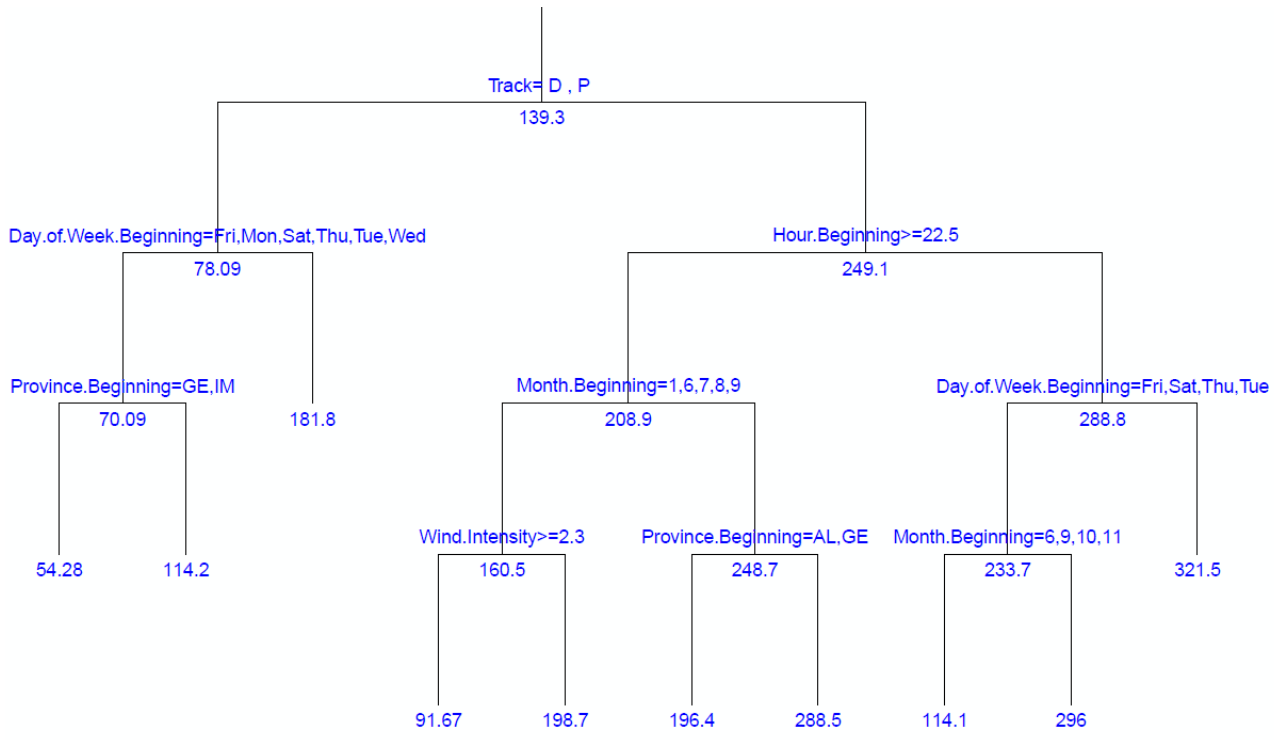


Figure 49: Detail of Pruned Tree for extraordinary works

```
Median of absolute percentage error: 58%

Mean of absolute percentage error: 132%
```

Table 16: Mean and Median of Absolute Percentage Error

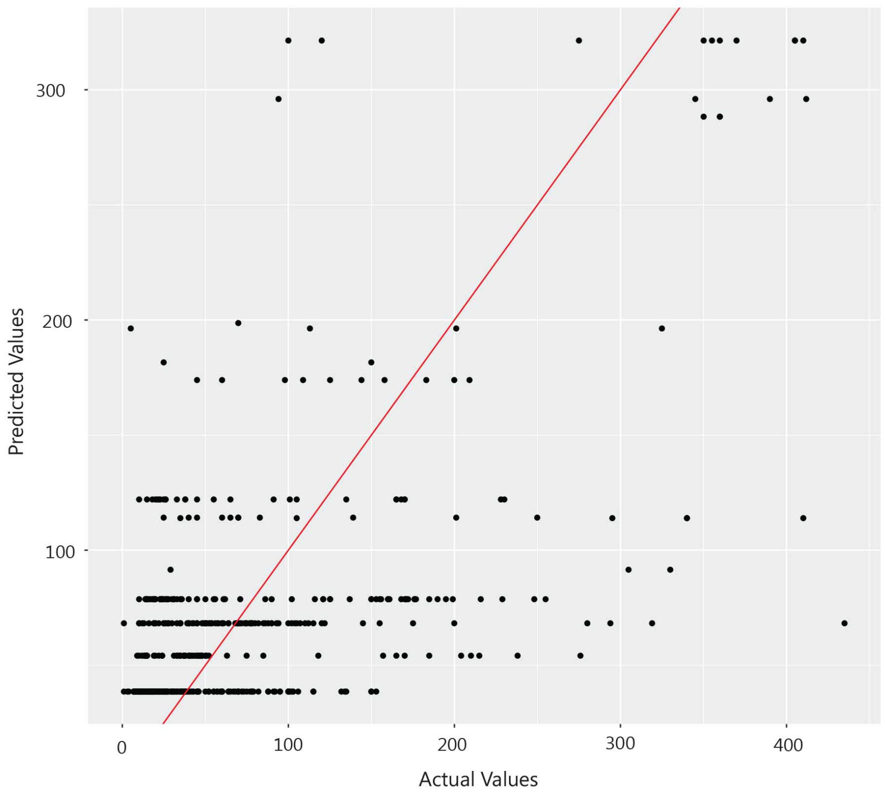


Figure 50: Scatter plot of actual vs predicted values (test set)

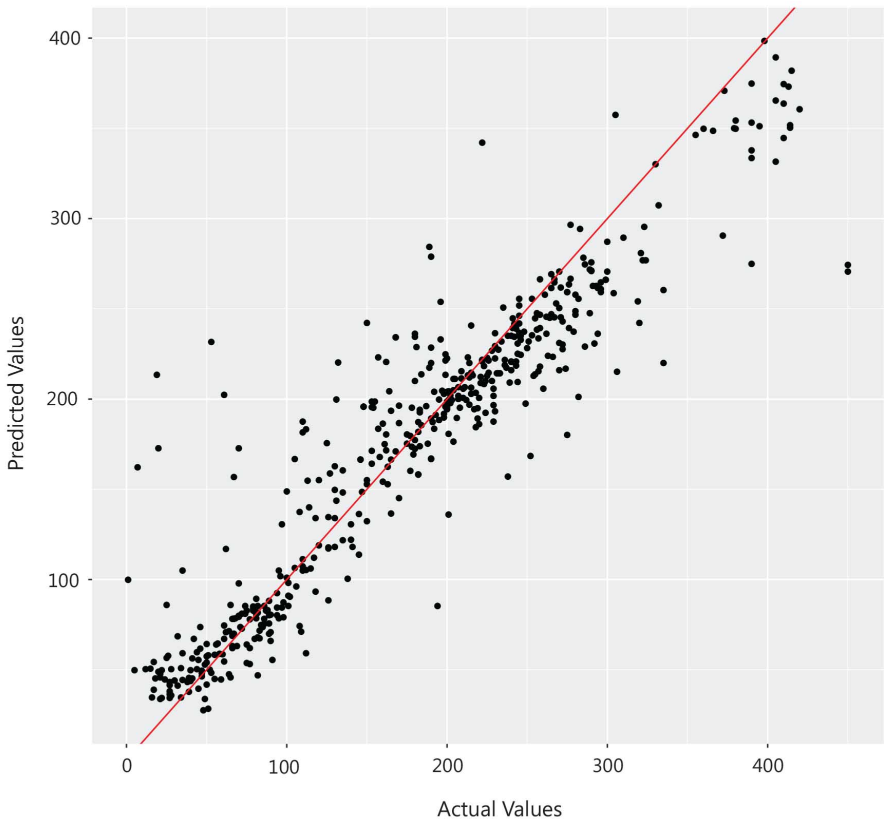


Figure 51: Scatter plot of actual vs predicted values (test set)

The number of trees is set to 500, one of the most common options; the number of variables tried at each split is set to $\lceil\sqrt{r}\rceil$, given r the number of features.

2.5.2.3.1 Planned maintenance

```
Type of random forest: regression
Number of trees: 500
No.~of variables tried at each split: 4
Mean of squared residuals: 1122.874
Var explained: 88.83%
```

Mean of Squared Residuals and Proportion of Variance Explained are calculated as out-of-bag (OOB) estimates, hence they are the mean predictions of MSR and R^2 on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample.
As would be expected, R^2 is much higher in this model than in the single tree one. The scatter plot, instead, shows how the model performs on the test set (Figure 51). As the dots are evidently concentrated around the diagonal red line the model can be considered very satisfying. Furthermore, dots which deviate from the bisector have quite an even distribution, which does not highlight any bias in the residuals.
The feature ranking is similar to the previous one, confirming the importance of Planned Repair Time (Table 17 and 18). The two rankings were built using mean decrease in impurity (as for the trees) and mean decrease in accuracy; the results are similar but not identical, especially as regards Province and Solar Radiation, which however can be considered relevant in both cases.

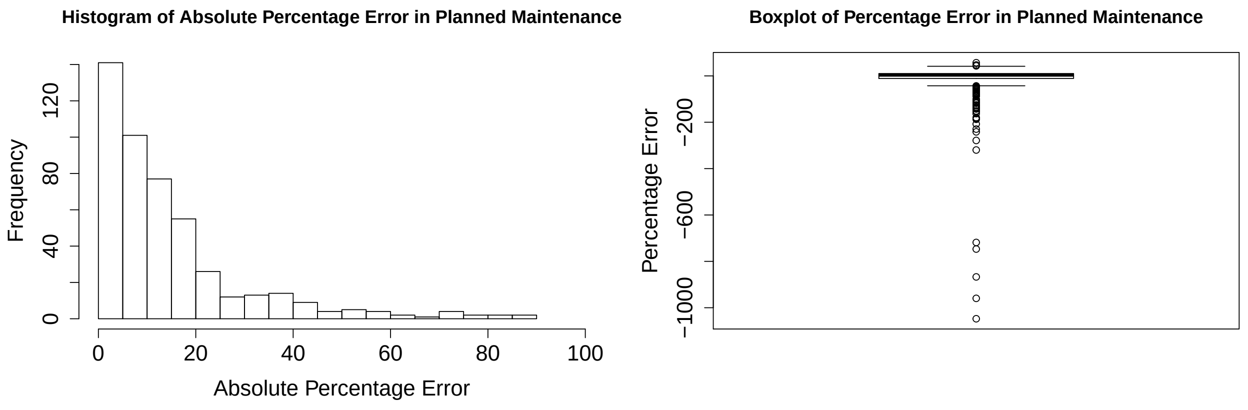


Figure 52: Two graphs of percentage error (test set)

Variable	Mean decrease in accuracy
Planned.Repair.Time.Num	70.6
Hour.Beginning	32.0
Province.Beginning	29.4
Track	25.8
Actual.Beginning.Time	23.2
Day.of.Week.Beginning	23.0
Month.Beginning	22.0
Wind.Intensity	21.6
Temperature	19.4
Solar.Radiation	17.0
Incl..Escl	14.2
Rainfall	8.5
Condition	8.5
Incl..Escl.1	3.4

Table 17: Variable Importance (mean decrease in accuracy)

Variable	Mean decrease in node impurity
Planned.Repair.Time.Num	7720030
Hour.Beginning	2490630
Solar.Radiation	1917280
Track	604470
Actual.Beginning.Time	509470
Temperature	495970
Day.of.Week.Beginning	394040
Month.Beginning	356630
Wind.Intensity	335740
Province.Beginning	322750
Condition	693560
Incl..Escl	67990
Rainfall	64440
Incl..Escl.1	12520

Table 18: Variable Importance (mean decrease in node impurity)

```
Median of absolute percentage error: 10%

Mean of absolute percentage error: 26%
```

Table 19: Mean and Median of Absolute Percentage Error

The histogram in Figure 52 is similar to the previous one (Figure 45) too; the second graph in the figure is a boxplot of the (not absolute) percentage error, which shows in detail how often the model overestimates or underestimates actual values, including the outliers not considered in the histogram (23 out of 504 values of absolute percentage error are greater than 1 and have not been reported for a better visualization).
As can be seen both from the boxplot above and from the difference between mean and median, the distribution of the (absolute) error is completely uneven and its concentration is especially around the 0 value, though there are some great negative outliers. This would suggest that overestimates are much more frequent than underestimates, but this can be not completely true: model often overestimates little values, so that the (negative) difference with the predicted values, expressed as a percentage of a little number, seems a more relevant value than the errors calculated as a percentage of grater values, more often underestimated. Eventually, it has been noticed that the observations with higher percentage error have values of Planned Repair Time which differ from Actual Repair Time more than the average.

2.5.2.3.2 Extraordinary works

```
Type of random forest: regression
Number of trees: 500
No.~of variables tried at each split: 4
Mean of squared residuals: 3120.265
Var explained: 58.32%
```

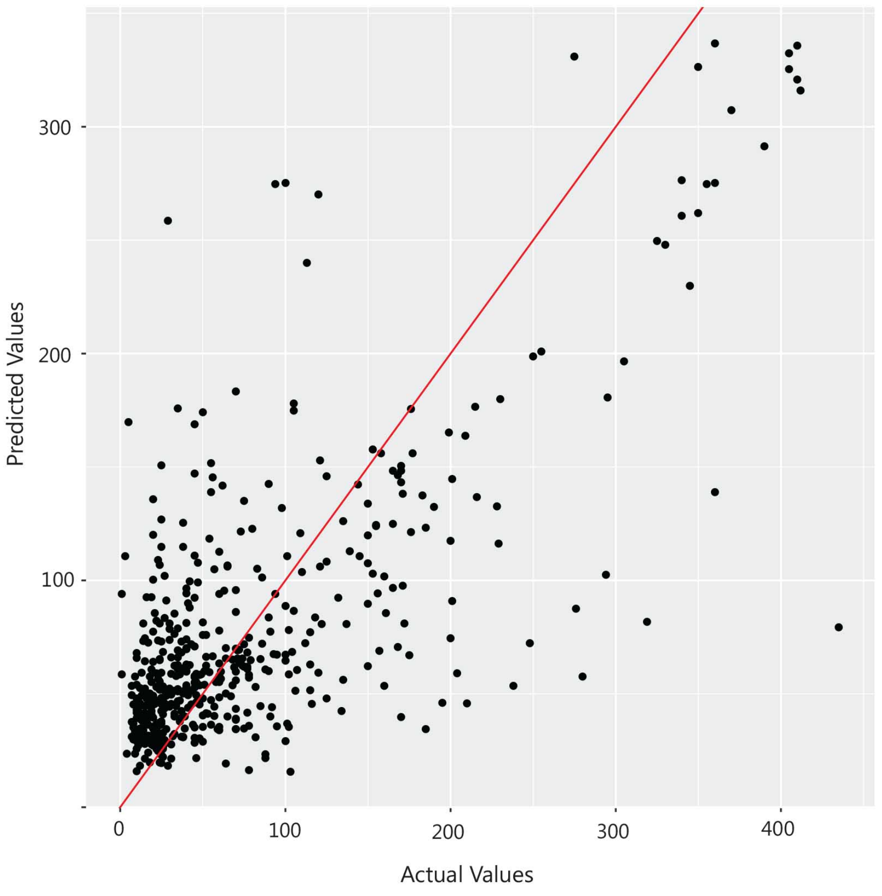


Figure 53: Scatter plot of actual vs predicted values (test set)

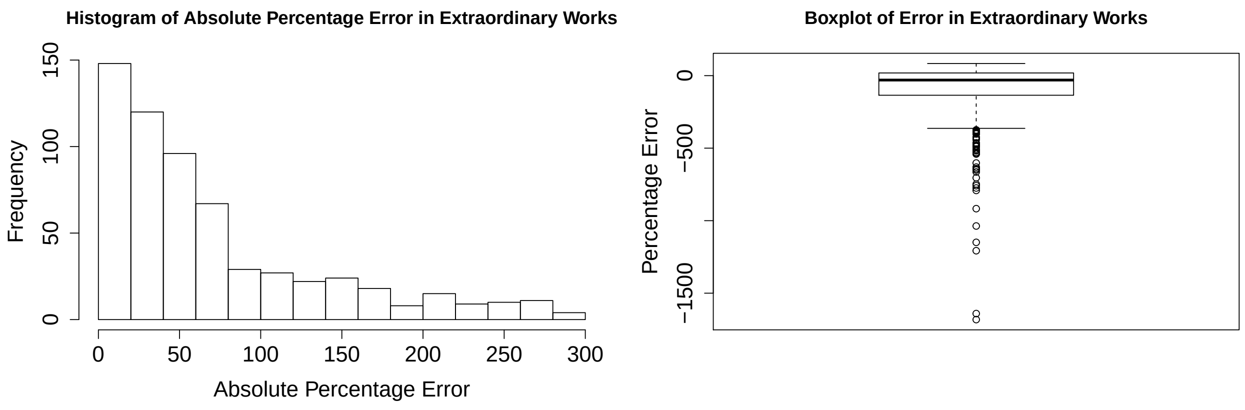


Figure 54: Two graphs of percentage error (test set)

The importance ranking of the variables (Tables 21 and 22) is similar to the single tree one: track, beginning time and weather conditions become much more important as repair actions are not scheduled. The low position of the variable Rainfall can be partially explained with the fact that Solar Radiation provides a similar information -but more complete as marks the difference between night and daylight.
It is also interesting to note that the variables relative to inclusion or exclusion of the involved stations are not in the very last positions in both the rankings.
As would be expected, the results with the extraordinary works are much worse than the results with planned ones. MSE, R^2 and all the graphs and statistics suggest that the model, though more satisfying than the single regression tree, has a predictive accuracy which cannot be compared with scheduled maintenance. Explained variance is about 40% lower than the planned works one and the median of absolute percentage error (26%) is almost three times the other one -but about half the error of the single tree. The scatter plot shows a quite indefinite shape of dots, though an uphill trend is still roughly visible.

2.5.2.3.3 Extraordinary works divided into provinces

It has been observed that the beginning conditions (in terms of time, weather and space) are really influential in the prediction of time to restoration in extraordinary works. For this reason, taking into account Beginning Station and End Station could enhance significantly the performance of the predictors. While using these two factors in the previous random forest models -which were trained on data relative to different provinces- caused some issues due to the too large amount of levels, training new province-specific models could work. Data are divided into four datasets, each containing observations from the same province, and then each further divided in training and test set.
Province-specific random forests -when beginning and end stations are involved in models- show much better performances than the generic ones in the prediction of extraordinary works (Tables 23 - 26).
In fact:

- Mean of squared residuals is lower in all the comparisons;
- In the models involving data from Savona and Imperia -which are the smallest datasets- the proportion of variance explained is significantly higher, from 9 to 30 percentage points. As regards Genoa and Alessandria, the proportions are quite similar, though a little decrease is still visible;
- While the two models on Alessandria's data show similar performances, in the other cases the median and mean of percentage error is not comparable: by taking into account beginning and end stations, relative error is remarkably reduced.

Furthermore, the importance of the additional features is confirmed in the variable ranking (Figure 55). This graph regards Genoa's data, but almost identical results were obtained with other provinces.
However, it is interesting to observe that this kind of approach does not enhance the performance of the predictor when trained (and tested) with planned maintenance data, neither in terms of accuracy or importance ranking. Other possible splits of the data have been tried, but, for the scheduled maintenance, the best model found in this work is the one involving the entire dataset.

```
Median of absolute percentage error: 26%

Mean of absolute percentage error: 73%
```

Table 20: Mean and Median of Absolute Percentage Error

Variable	Mean decrease in accuracy
Track	72.8
Hour.Beginning	48.1
Province.Beginning	44.2
Day.of.Week.Beginning	34.33
Month.Beginning	32.6
Wind.Intensity	31.6
Actual.Beginning.Time	27.7
Temperature	25.7
Solar.Radiation	24.4
Incl..Escl.1	8.1
Incl..Escl	6.4
Rainfall	5.7
Condition	1.4

Table 21: Variable Importance (mean decrease in accuracy)

Variable	Mean decrease in node impurity
Hour.Beginning	3184730
Track	2379270
Solar.Radiation	1525520
Actual.Beginning.Time	1503400
Wind.Intensity	1404040
Province.Beginning	1402581
Day.of.Week.Beginning	1381970
Temperature	1310180
Month.Beginning	1018650
Incl..Escl	118350
Incl..Escl.1	116250
Rainfall	108760
Condition	11030

Table 22: Variable Importance (mean decrease in node impurity)

Without Beginning and End Station	With Beginning and End Station
Mean of squared residuals: 2749	Mean of squared residuals: 2727
% Var explained: 40.94	% Var explained: 41.42
Mean of abs. percentage error: 106%	Mean of abs. percentage error: 52%
Median of abs. percentage error: 64%	Median of abs. percentage error: 16%

Table 23: Results from Genoa's data

Without Beginning and End Station	With Beginning and End Station
Mean of squared residuals: 4702	Mean of squared residuals: 2427
% Var explained: 51.04 Mean of abs. percentage error: 283%	% Var explained: 81.22 Mean of abs. percentage error: 32%
Median of abs. percentage error: 41%	Median of abs. percentage error: 13%

Table 24: Results from Savona's data

Without Beginning and End Station	With Beginning and End Station
Mean of squared residuals: 2628	Mean of squared residuals: 1935
% Var explained: 65.08	% Var explained: 74.41
Mean of abs. percentage error: 168%	Mean of abs. percentage error: 37%
Median of abs. percentage error: 44%	Median of abs. percentage error: 17%

Table 25: Results from Imperia's data

Without Beginning and End Station	With Beginning and End Station
Mean of squared residuals: 3687	Mean of squared residuals: 3366
% Var explained: 66.37	% Var explained: 68.38
Mean of abs. percentage error: 87%	Mean of abs. percentage error: 83%
Median of abs. percentage error: 32%	Median of abs. percentage error: 29%

Table 26: Results from Alessandria's data

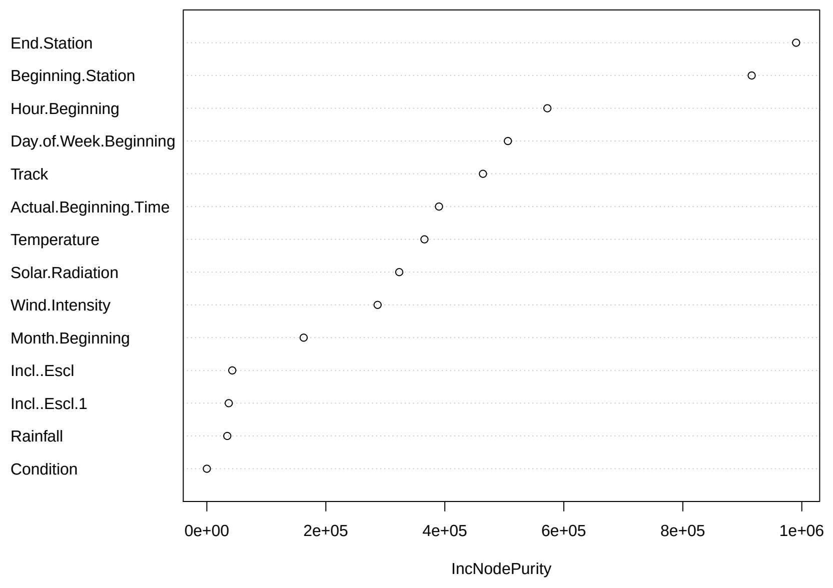


Figure 55: Variable importance ranking (Genoa)

2.5.3 SR - Strukton Rail

This scenario was previously analyzed in [74].
Initially, a predictive approach had been proposed; it showed that it was possible to capture interesting functional dependencies between input and output variables, though the average percentage error in prediction was not satisfying in the perspective of an actual real-world application. The noise in data and the few examples of failures for some of the many different failure types affected the predictive accuracy, so a diagnostic approach is here proposed instead.
Strukton Rail is one of the most qualified railroad contractor in Northern Europe for all rail systems and rail works; in this scenario data from Netherlands are exploited.

2.5.3.1 Data description and basic statistics

The analysis will consider reports of maintenance works from 01-01-2010 to 31-12-2015. There are 17243 observations of 47 variables. Each observation identifies a single failure/repair action, for which a large amount of information about time, place, mechanic's characteristics and weather is provided.
At the very first step, Strukton receives a notification from the Infrastructure Manager that a failure on an asset has been detected. The notification includes several information, such as a priority level, the ID and location of the asset to be repaired, etc. After the notification has been received, mechanics are informed that a failure on an asset has been reported; when mechanics arrive on the asset location, travelling from the closest headquarter, they communicate that they reached the asset location and start inspecting the asset for assessing its status. Based on this first inspection, the mechanics perform a ”soft” forecast on the kind of action and the time needed to complete the intervention. Then the timestamp in which the mechanics start the repair intervention is recorded.
The function is restored when the mechanics have completed the intervention so that the line has been freed and trains can travel again over it.
This structure of the dataset (Table 27) is the result of a process of feature extraction and selection.
In addition to the original variables (provided by Strukton Rail), in fact, weather data have been retrieved from the Royal Netherlands Meteorological Institute (KNMI) and some other useful features have been derived.

2.5.3.1.1 SR variables

Considering the problem as a regression one, there are three response variables:

- **Response time** is the time needed by the mechanics to start operating on the asset from the moment in which the failure notification has been received;
- **Repair time** is the time needed by the mechanics to perform the repair on the asset;
- **Function restoration time** is the time needed to complete the intervention and free the railway line.

The scatterplots (Figure 56) show different associations between the three variables: both Repair and Response time seem to be positively correlated with Function Restoration time, while not with each other. This could be expected -as Function Restoration time is influenced by both the other two variables- and is confirmed by correlation coefficients: 0.74, 0.59 and 0.06 (respectively, looking at the graphs).
It is interesting to note that, in both the first and the second plot, dots show two different patterns: ‘short’ repair time distribution is almost independent from function restoration time, while as values grow the positive linear association become clear.
Furthermore, these graphs highlight some anomalies, e.g. a few observations have values of repair time grater than function restoration time, which is meaningless. This information, combined with that contained in Table 28, can be used to detect and delete some outliers.

Name	Data Type	Source
Geographic Code	Factor: 139 levels	SR
Failure Type	Factor: 28 levels	SR
Failure Type Dutch	Factor: 143 levels	SR
Object Type	Factor: 10 levels	SR
Object ID	Factor: 6964 levels	SR
Technical Department	Factor: 14 levels	SR
Part Code	Factor: 187 levels	SR
Action Carried Out	Factor: 76 levels	SR
Failure Main Group	Factor: 4 levels	SR
Failure Cause	Factor: 60 levels	SR
Longitude	Numeric	SR
Latitude	Numeric	SR
Mechanic ID	Factor: 533 levels	SR
Diagnosis Time	Numeric	SR
Year	Factor: 6 levels	Derived
Month	Factor: 12 levels	Derived
Day	Numeric	Derived
Hour	Numeric	Derived
Day of the Week	Factor: 7 levels	Derived
Zone	Factor: 3 levels	Derived
Year of Activity	Integer	Derived
Mechanic's Past Actions (in the zone)	Integer	Derived
Past Actions - Normed (in the zone)	Numeric	Derived
Past Actions - Difference from mean (in the zone)	Numeric	Derived
Past Actions (same priority)	Integer	Derived
Past Actions - Normed (same priority)	Numeric	Derived
Past Actions - Difference from mean (same priority)	Numeric	Derived
Past Actions (total)	Integer	Derived
Past Actions - Normed (total)	Numeric	Derived
Past Actions - Difference from mean (total)	Numeric	Derived
Past Failures in the Week	Integer	Derived
Past Failures in the Month	Integer	Derived
Past Failures in Six Months	Integer	Derived
Open Failures in the Zone	Integer	Derived
Wind Speed	Numeric	KNMI
Temperature	Numeric	KNMI
Dew Point	Numeric	KNMI
Global Radiation	Numeric	KNMI
Wind Direction	Factor: 10 levels	KNMI
Sunshine duration	Numeric	KNMI
Rainfall duration	Numeric	KNMI
Rainfall	Numeric	KNMI
Response Time	Numeric	SR
Repair Time	Numeric	SR
Function Restoration Time	Numeric	SR

Table 27: Structure of the dataset

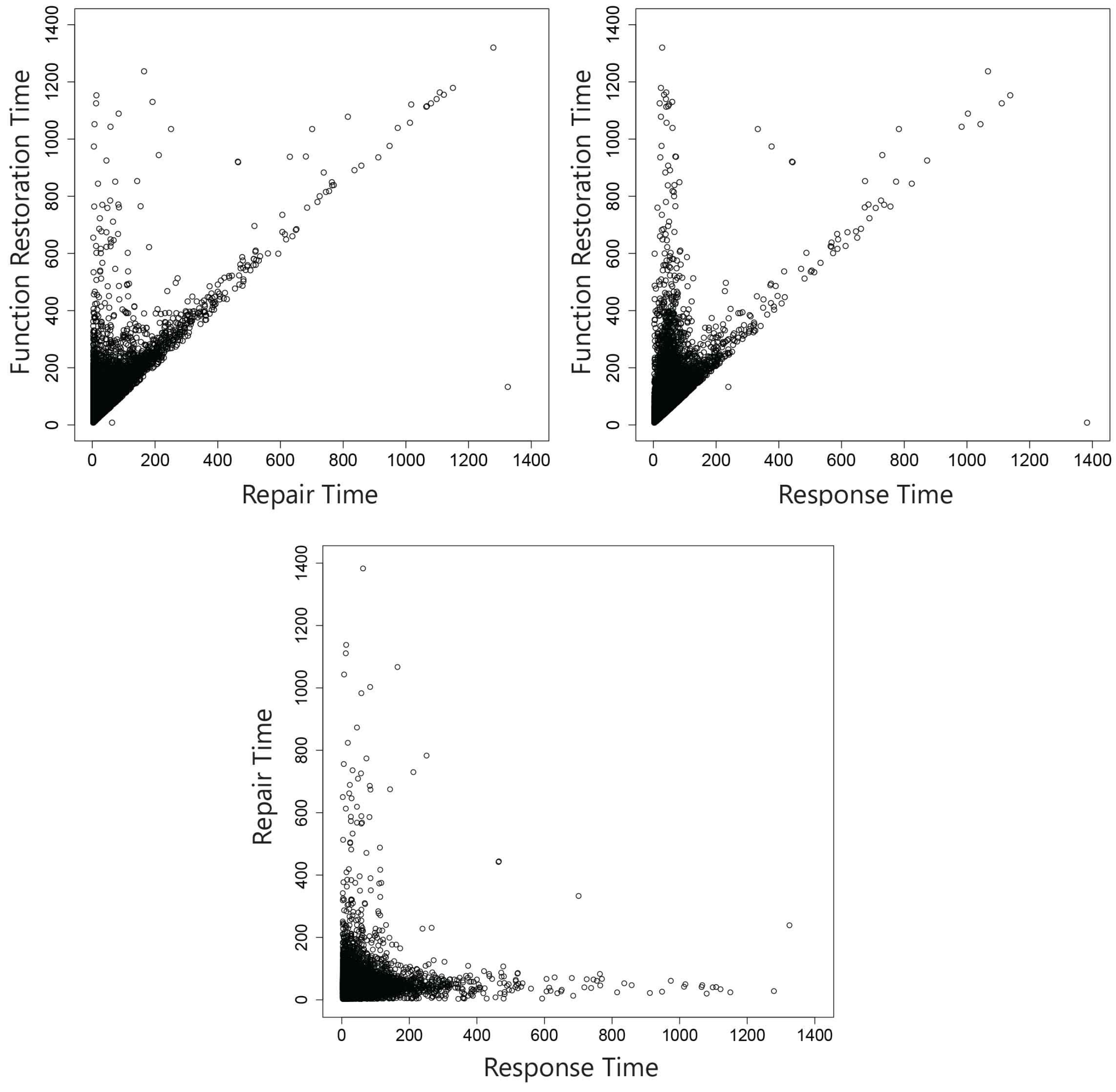


Figure 56: Scatterplots of response variables

```
Min.~1st Qu.~Median Mean 3rd Qu.~Max.
0 27 42 47.2 57 1383

Min.~1st Qu.~Median Mean 3rd Qu.~Max.
0 2 11 31 37 1325

Min.~1st Qu.~Median Mean 3rd Qu.~Max.
2 45 70 88.6 106 1320
```

Table 28: Summary of Response, Repair and Function restoration time

2.5.3.1.2 Weather variables

As previously noted in RFI's case, weather conditions can noticeably affect time to restoration, and therefore taking them into account can enhance the performance of data-driven models.
Weather data consisted of 39 variables fro which 8 have been extracted by selecting or recombining the original ones, in order to obtain the most relevant features and to avoid redundant ones. The entire data refers to Netherlands, where Strukton is the responsible for asset maintenance.
The two datasets -SR data and weather data- have been linked by correlating the geographical locations of the assets to the locations of the weather stations, so to find the closest one for which it is possible to extract the most accurate weather information related to each asset.

2.5.3.1.3 Derived variables

It is possible to extract some new features relative to time and location from the original ones; it will be considered, especially:

- The month, day and hour of the beginning date;
- The day of the week of the beginning date;
- The "Zone" of the beginning station.

The variable "Zone" -based on the latitude and longitude coordinates of the failure locations- identifies three different areas. In fact, by looking at the map (Figure 57), a peculiar tripartite distribution is evident.
The three zones have similar size but some different characteristics, as can be seen in Figure 58 and 59.
In order to explain and predict the repair time, mechanic's experience can be another useful piece of information. It can be partially described by the number of past repair actions in which the mechanic was involved, so 9 features of this kind have been computed. They can be divided into three subgroups: the first one relates to the past actions in general; the second one refers to the past actions in the geographical area of the asset failure under examination (zone); the third group is related to the past actions on failures with the same priority.
Another group of features has been extracted from the failures that occurred before a certain notification is received. These "Past Failures" features are computed for different time horizons -a week, a month and six months before the notification under examination.
Eventually, the variable "Open Failures" has been derived by considering all the unresolved maintenance/repair actions that still have to be completed at a certain time.

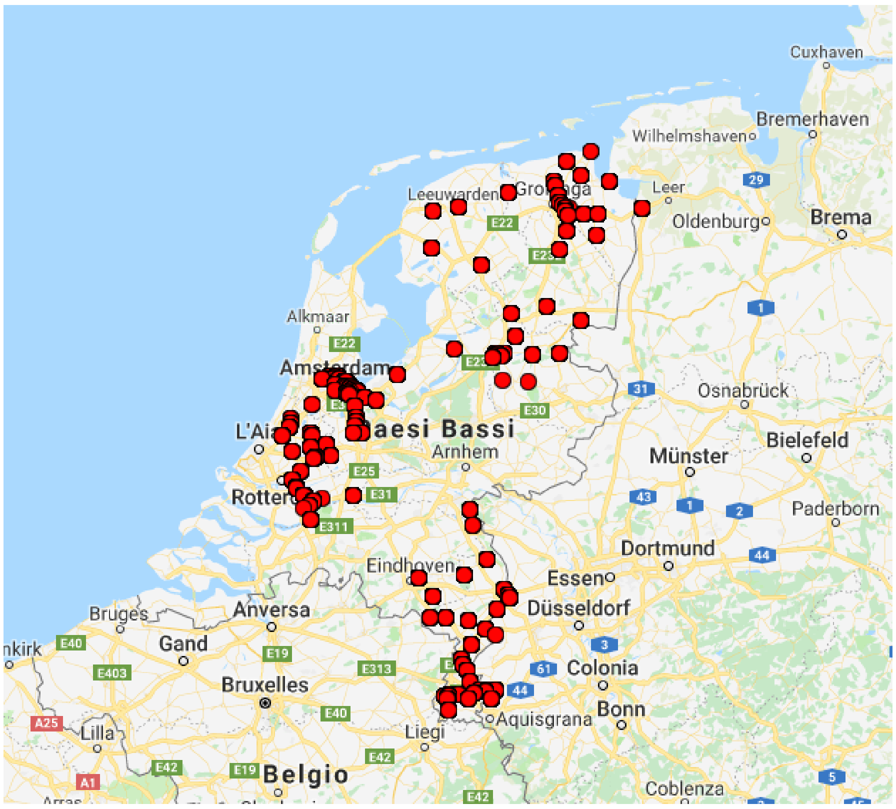


Figure 57: Map of failure locations

2.5.3.1.4 Data preparation

As already shown in the statistics and in the figures above, there are few outliers in the distribution of the response variables. For this reason, observations with a value greater than 360 minutes (400 for Function Restoration time) and less than 2 minutes have not been considered (about 4000 observations out of 17243).

2.5.3.2 Regression trees

As already said, the objective of this scenario was to estimate the significance of the different parameters on the time to restoration. Thanks to regression trees, it has been possible to realize an effective visualization of the role of the variables. As models were built with a descriptive purpose more than a predictive one, some overfitting has been tolerated. Furthermore, feature ranking methodology made it possible to estimate the relevance of each single input parameter of the models with the real outputs.
Data are no more divided into test and train set: as the aim is not predictive, there is no need to measure models' performance on previously unseen data, so all data can be considered training data. Descriptive accuracy of the developed algorithms is connected, instead, with their ability to manage the trade-off between fidelity to data (overfitting) and generalizable significance of the detected dependencies.
Many different splits of the dataset have been tried in order to reach the right compromise between gain and loss of information due to size -and therefore impurity- reduction. The best split has been found to be -similarly to RFI's case- a zone-specific one: for each response variable, three different models have been built, one for every level of the variable Zone.

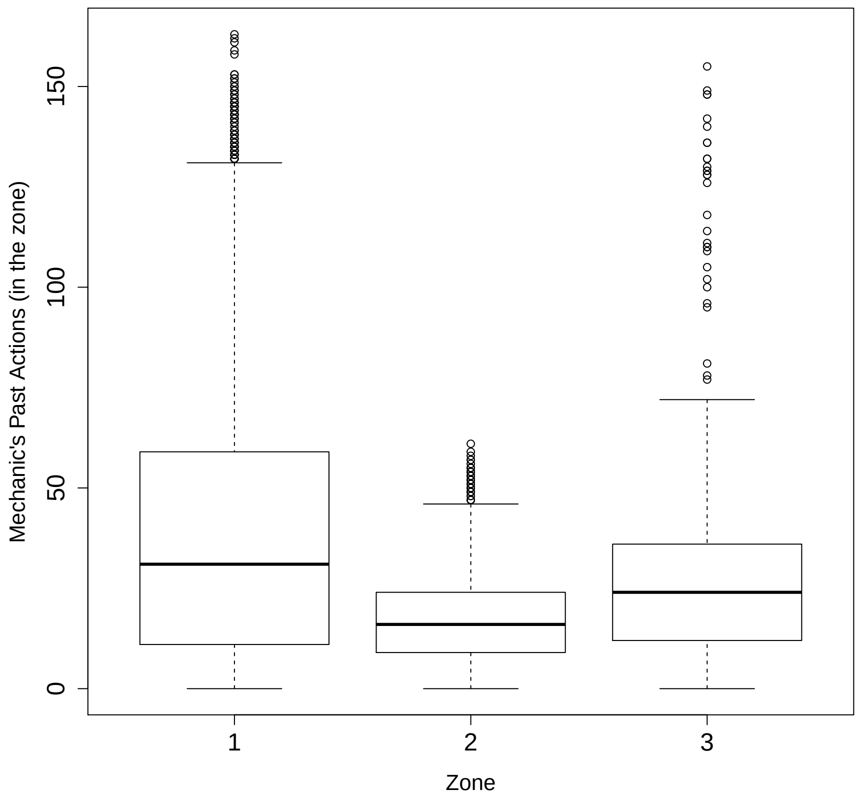


Figure 58: Past Actions and Zone

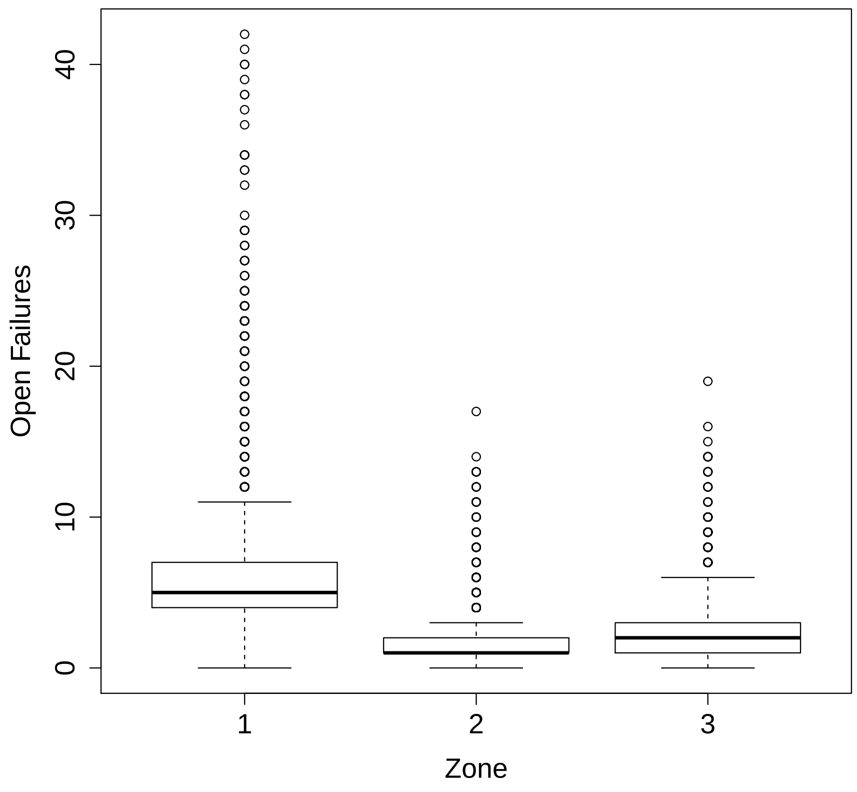


Figure 59: Open Failures and Zone

2.5.3.2.1 Repair Time

In the aforementioned previous analyses [74], the Repair Time was the most difficult quantity to predict. In all the simulations, the performance of the associated data-driven models were the less satisfactory.
Also in the diagnostic models here presented the performance relative to this quantity -intended as the ability of the models to generalize other than describe available data- is the worst. This can be seen in Figure 60, which shows how the error varies according to the size of the tree: the cross-validated error barely declines under 100% error at the very first split, but it rises almost immediately, making overfitting utterly visible - and reaching an error value of about 1.5. Results are similar in the three zones, so just one of the graphs is reported.

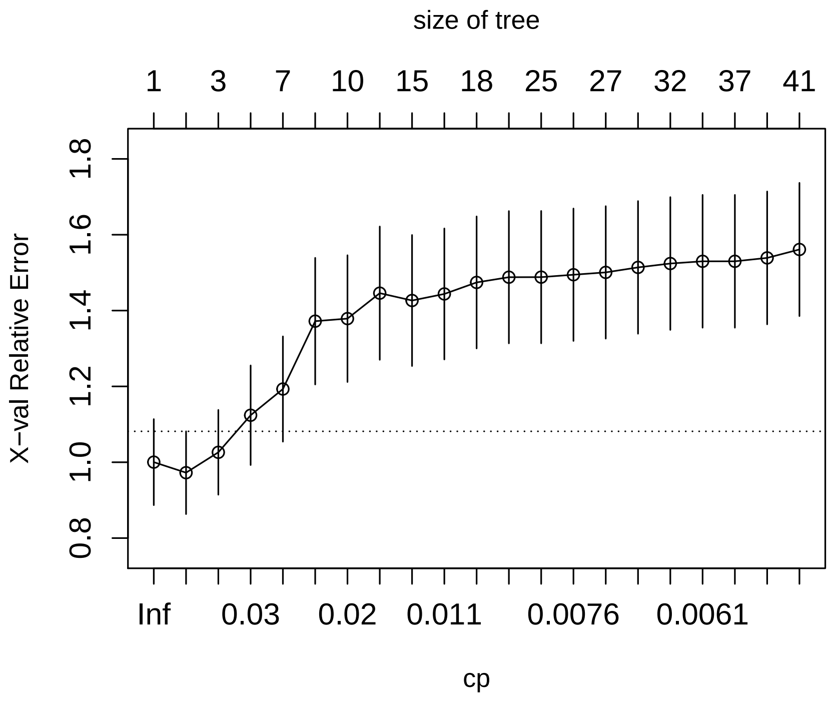


Figure 60: Cross-validated relative error and size of the tree (Zone 3)

Figures 61, 62 and 63 represent the trees describing the data of the three zones.
As the intent was to use these trees to visualize data and their connections, a part of the work was focused on the research of correct visualization tools. The two objectives of the visualization were partially in contrast with each other: while comprehension should be easy and direct, without too much complex or redundant content interfering with it, the description of nodes and branches -and the subsets involved- should be accurate and meticulous.
After different attempts, quite detailed and at the same time legible trees have been produced, also thanks to specific packages.
The final leaves show the average repair time and the number of observations in the final branches. Colours vary from green to red as the values of the response variable increase. It is interesting to note how the different splits identify from left to right different subsets with increasing values of repair time. Furthermore, please note that the size of the final leaves is quite unbalanced, with values varying approximately between 10 and 800. These differences can be analyzed with particular attention to the smaller sizes; in fact, trees and forests are often used also to detect anomalies: the sooner a very small subset is isolated in a leaf, the most anomalous the subset is [51].

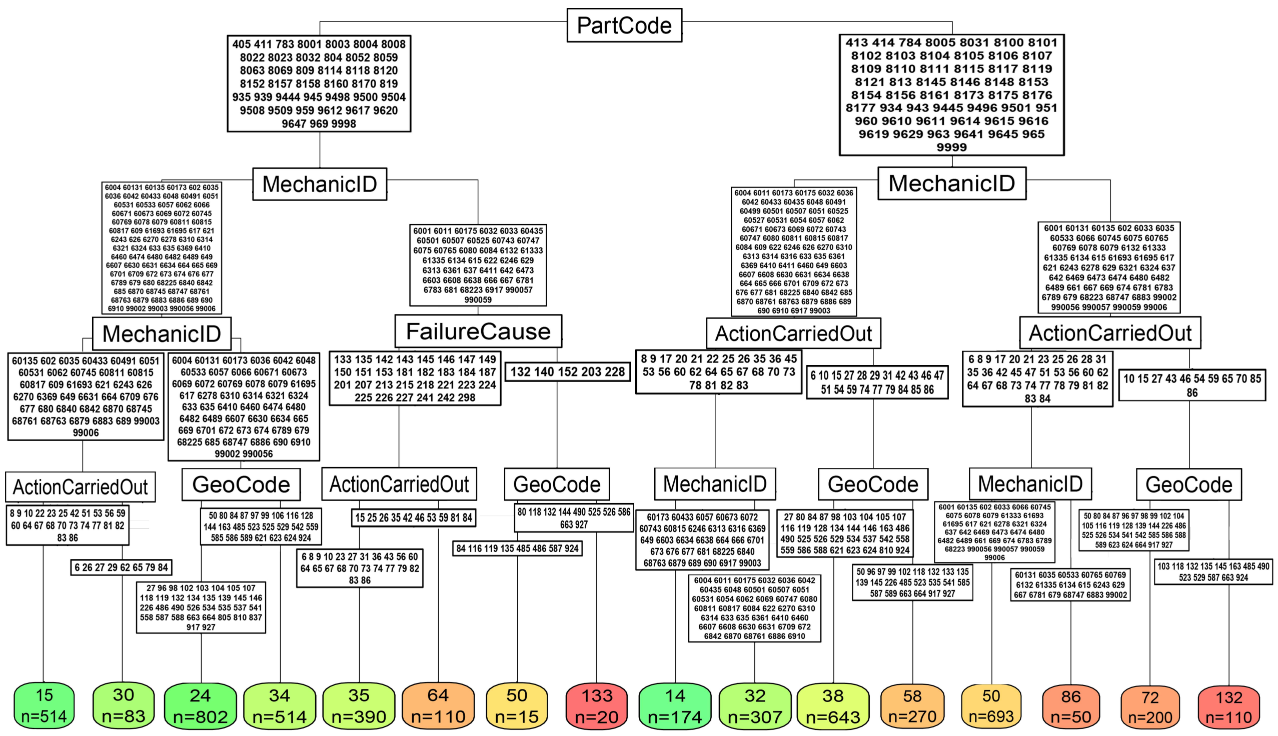


Figure 61: Tree of repair time, Zone 1

It is significant that the ‘Part Code’ node comes at the top of the tree in all the zones and that, in the importance ranking, this variable is always in one of the top three positions. In general, the other most important features are Geographic Code, Failure Type Dutch (more specific than Failure Type), Mechanic ID, Failure Cause and Action Carried Out. The influence of technical characteristics is not surprising, and seems coherent with the fact that some features relative to mechanics’ experience are in high importance positions in every ranking. Between the weather variables, temperature and rainfall (and wind in the third zone) are the most influential.
Different pruning levels have been tried in order to reach more or less specific visualizations. Looking in more detail (Figure 64, 65) interesting similarities and significant differences can be seen in the split of the part codes and failure causes as the zone changes.

- As regards the partcode splits, it must be noted that different zones have some different partcodes, so comparison cannot be completely accurate; however, it is evident that certain codes (e.g. 8001, 8002, 8003 and many others) correspond systematically to lower values of repair time and certain codes (e.g. 8101, 8170, 9611 etc.) correspond to longer times. A few codes, anyway, appear in different branches as the zone changes, less consistently;
- Failure Causes can be compared more easily and a quite systematic behaviour is evident, as the splits in Figure 64 and 65 are almost identical.

2.5.3.2.2 Response Time

As can be seen in Figures 66 and 67, the performance associated with the variable Response Time -and also with Function Restoration, as will be shown in the next pages- is not the same in different zones, and the Zone 1 model outperforms the other two.

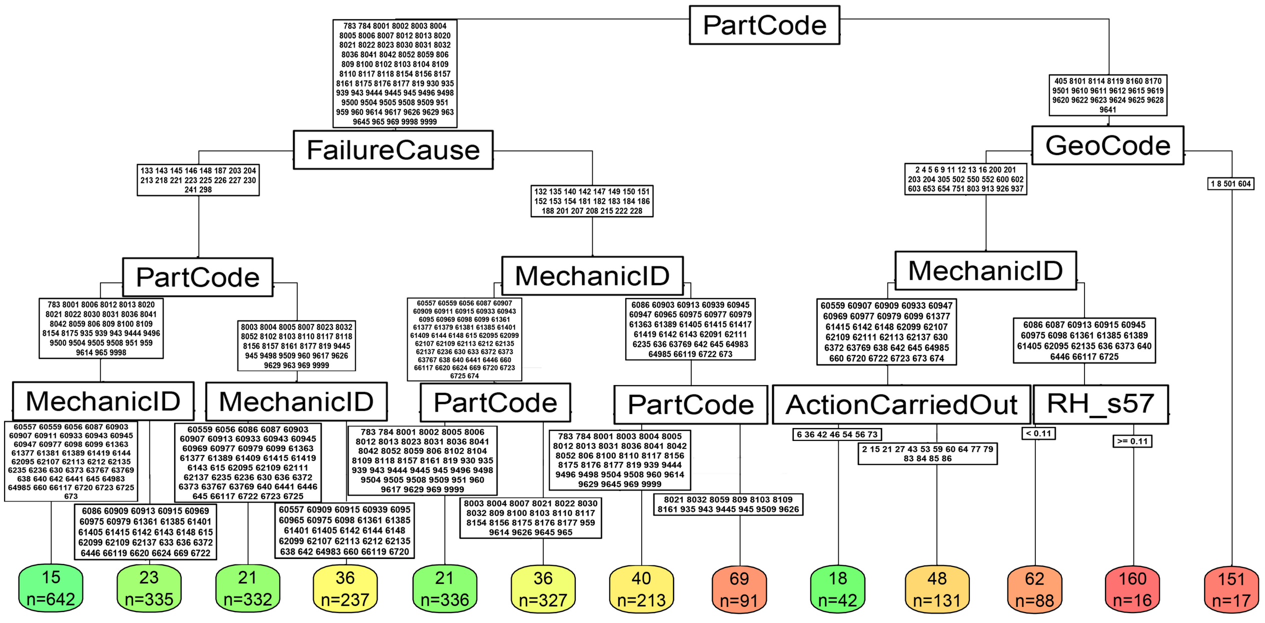


Figure 62: Tree of repair time, Zone 2

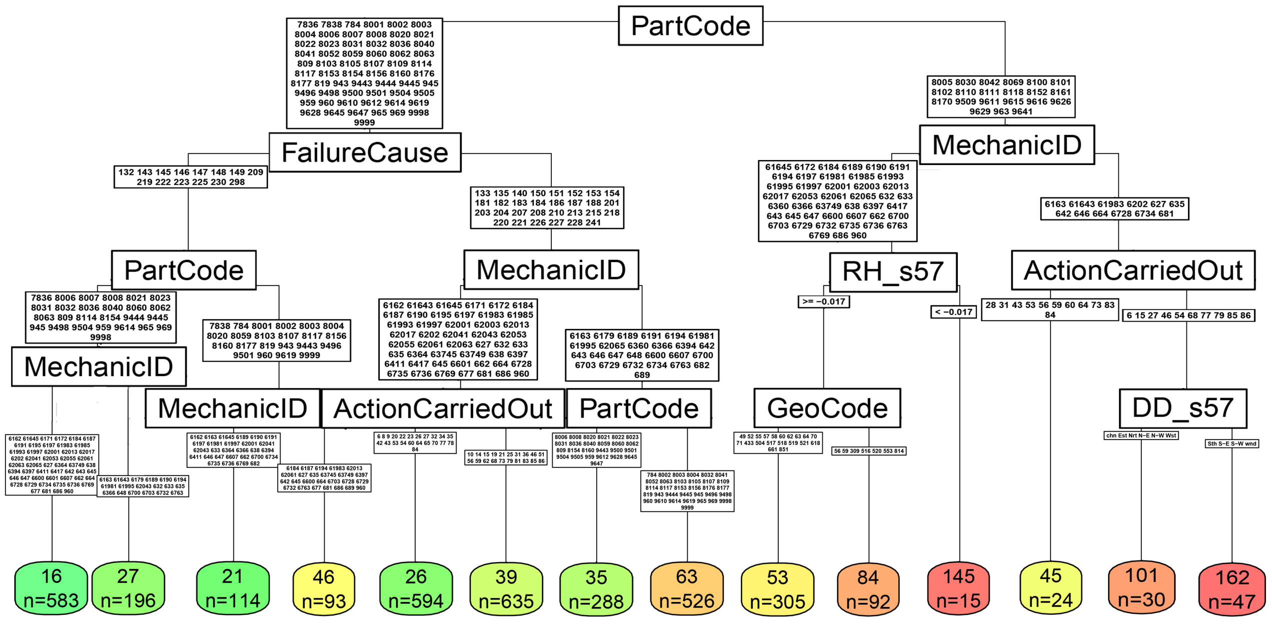


Figure 63: Tree of repair time, Zone 3

Variable	Mean decrease in node impurity
Part Code	5718453
Failure Type Dutch	5385223
Geographic Code	5299649
Mechanic ID	5096325
Action Carried Out	4458908
Failure Cause	2590437
Month	2380255
Rainfall	2006347
Object ID	1784495
Diagnosis Time	1587553
Past Actions - Normed (same priority)	1470584
Past Actions Difference from mean (same priority)	1469033
Object Type	1324140
Latitude	1321793
Past Actions - Difference from mean	1289806
Past Actions - Normed	1086578
Rainfall Duration	1081368
Past Actions (same priority)	1023389
Temperature	982643
Past Actions - Normed (Zone)	903564
Dew Point	854316
Day	814062
Sunshine Duration	801657
Past Actions (Zone)	780481
Day of the Week	752652
Failure Type	709722
Longitude	700064
Wind Speed	520322
Past Failures in the Week	318597
Failure Main Group	297220
Wind Direction	283841
12046106 Past Actions - Difference from mean (Zone)	239033
Hour	223792
Past Failures in the Month	181707
Technical Department	110354
Year of Activity	87987
Global Radiation	84206
Past Actions	76956
Year	65719
Past Failures in Six Months	36525

Table 29: Variable importance - Zone 1

Variable	Mean decrease in node impurity
Geographic Code	2861331
Failure Type Dutch	2371546
Part Code	2273050
Mechanic ID	2106038
Failure Cause	1099114
Action Carried Out	995704
Failure Type	787991
Rainfall	622828
Object ID	562957
Past Actions - Normed (Zone)	542999
Rainfall Duration	509819
Temperature	475314
Global Radiation	387264
Day of the Week	351604
Wind Direction	347064
Longitude	307345
Wind Speed	303705
Sunshine Duration	282612
Month	262589
Object Type	247650
Pat Actions (same priority)	230569
Dew Point	224770
Past Actions - Normed	204405
Past Actions - Difference from mean (same priority)	167567
Diagnosis Time	146541
Past Actions Difference from mean (Zone)	145010
Past Actions (same priority)	123313
Latitude	110500
Year of Activity	86645
Failure Main Group	84460
Year	82124
Past Actions (Zone)	57979
Past Failures in the Week	47779
Technical Department	46594
Past Actions - Difference from mean	45710
Day	37865
Hour	28086
Past Actions	8374
Past Failures in the Month	604

Table 30: Variable importance - Zone 2

Variable	Mean decrease in node impurity
Geographic Code	4400421
Part Code	3764296
Failure Type Dutch	3450569
Mechanic ID	3235184
Action Carried Out	2217742
Diagnosis Time	1761837
Failure Cause	1522351
Wind Speed	1192196
Rainfall	1191593
Failure Type	986948
Past Actions (Zone)	952128
Temperature	901352
Dew Point	781256
Longitude	779063
Day	614681
Year	583459
Object ID	570371
Latitude	553063
Month	538195
Past Actions Difference from mean (Zone)	467017
Wind Direction	466126
Year of Activity	444903
Past Actions - Normed (same priority)	421977
Object Type	417610
Past Actions - Difference from mean (same priority)	414417
Past Actions - Normed (Zone)	402325
Day of the Week	320400
Hour	311559
Global Radiation	270036
Past Actions - Difference from mean	265273
Sunshine Duration	89366
Past Failures in the Month	85373
Technical Department	79960
Past Actions - Normed	60507
Past Failures in Six Months	59484
Past Failures in the Week	48638
Past Actions	33473
Rainfall Duration	16936

Table 31: Variable importance - Zone 3

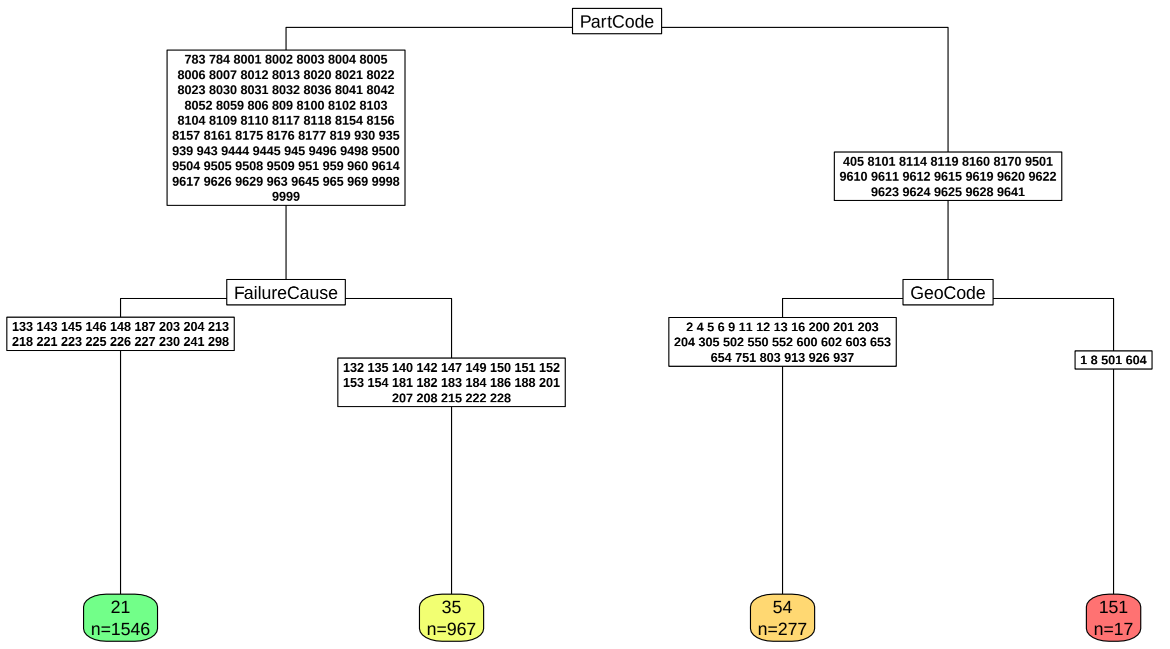


Figure 64: Tree of repair time, Zone 2 (detail)

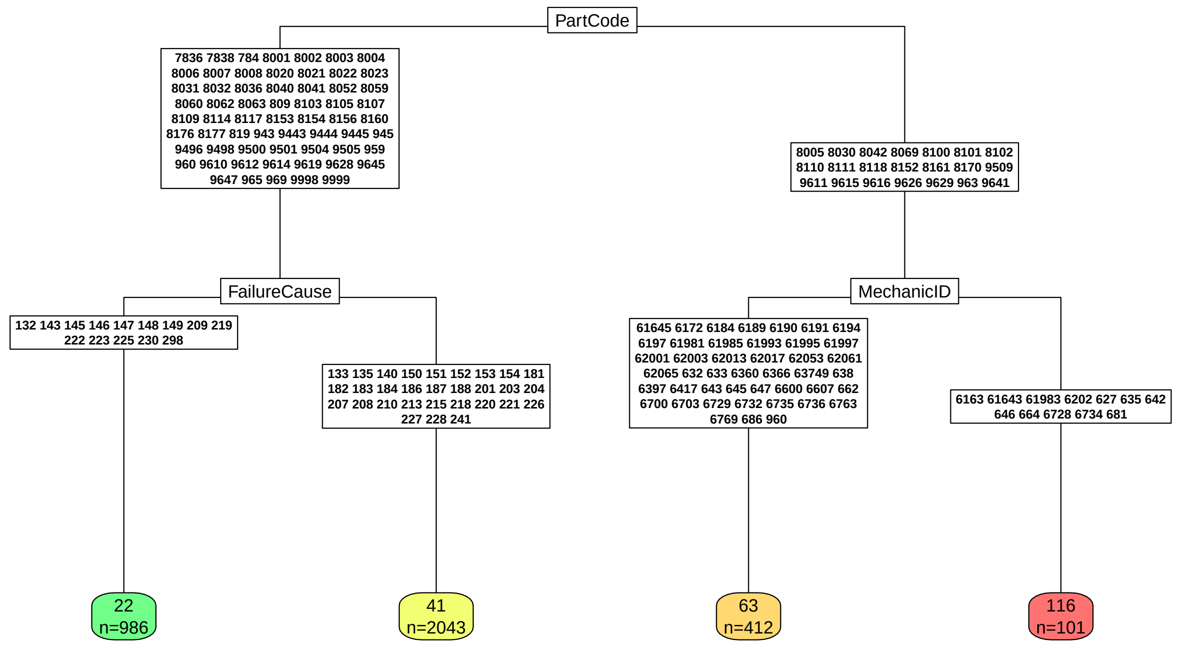


Figure 65: Tree of repair time, Zone 3 (detail)

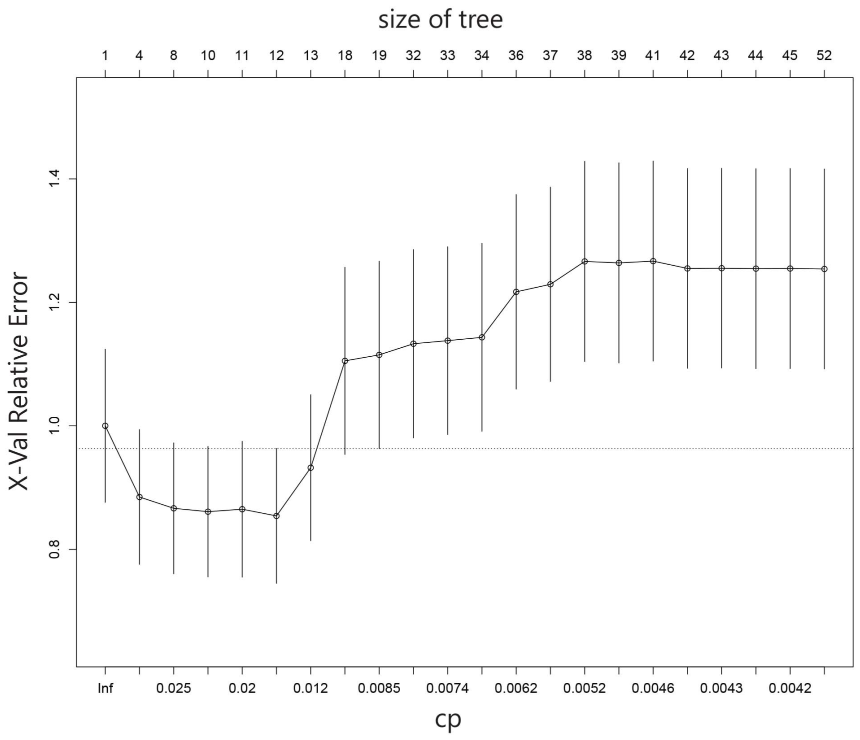


Figure 66: Cross-validated relative error and size of the tree (Zone 1)

In fact, in Figure 66 a great decrease in the cross-validated relative error can be seen, and overfitting starts affecting the model only when the size exceeds the amount of 12 leaves. In Figure 67, relative to Zone 3 -but really similar to Zone 2 results- relative error is instead continuously increasing.
The feature ranking is similar in the three zones: the Geographic Code is one of the most important variables and the weather features are very relevant in all the areas. Furthermore, Failure Type Dutch is in the top positions in both Zone 2 and 3, while not in the first zone.
These results differ from the Repair Time ones: Response Time is significantly affected by weather conditions more than by technical details. Actually, it can be quite intuitive that the time needed by the mechanics to start operating on the asset from the moment in which the failure notification has been received depends much more from external agents than from the characteristics of the asset.
The ranking of the weather features, however, is not identical in the three zones; as for Repair Time, Zone 1 seems to be more affected from the rain and less from the wind, which plays a more important role in Zone 2 and Zone 3. Temperature, instead, appears fundamental in all the areas.
In Figure 71, a detail of the Zone 1 model shows how the variable Open Failures affects the average response time, demonstrating the usefulness of this extracted feature.
It is interesting to note that the Open Failures split involves a subset of data where the average value of response time is higher than in the other branch of the first split; furthermore, the final leaves generated by this split have utterly unbalanced sizes, about 1 to 30 ratio. This facts could suggest that unresolved maintenance actions -and therefore the line congestion derived from them- tend to affect the response time especially when it is already longer than the average and to make the difference in a particular way when the level of congestion is anomalous.

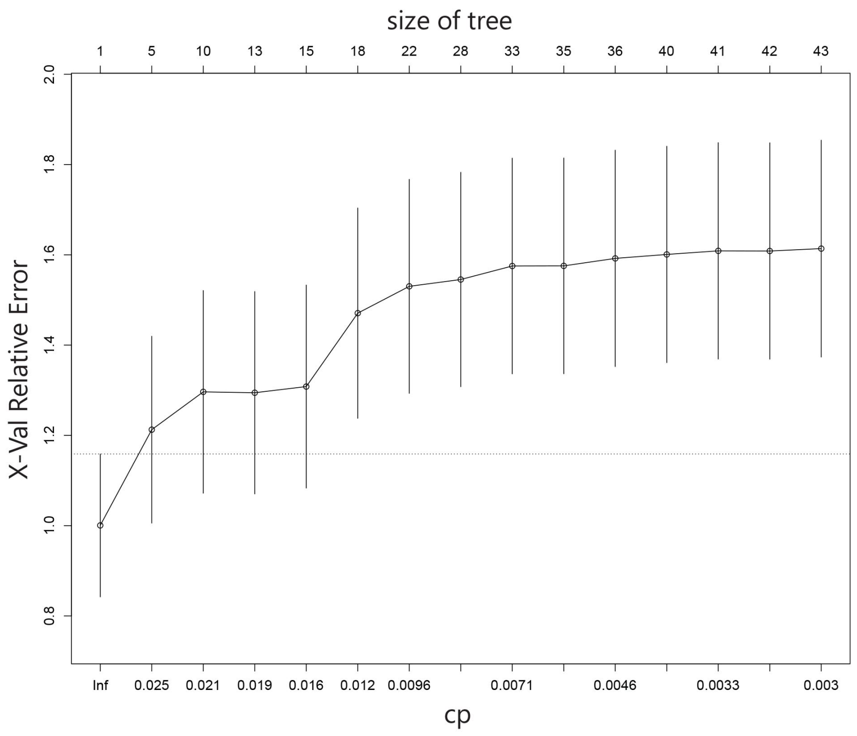


Figure 67: Cross-validated relative error and size of the tree (Zone 3)

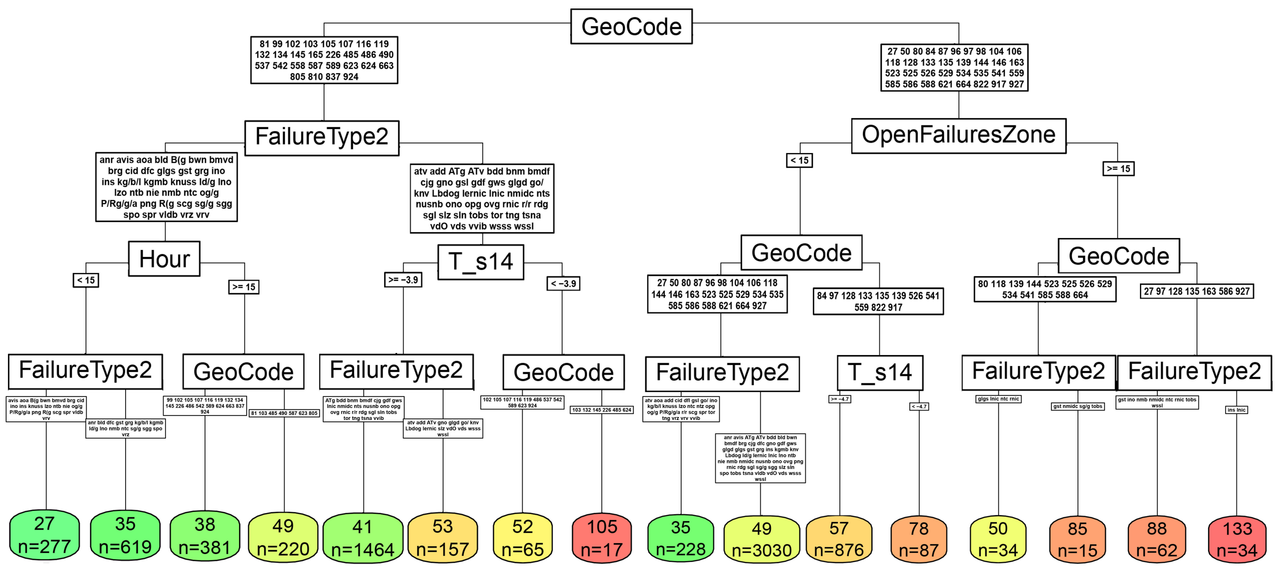


Figure 68: Tree of response time, Zone 1

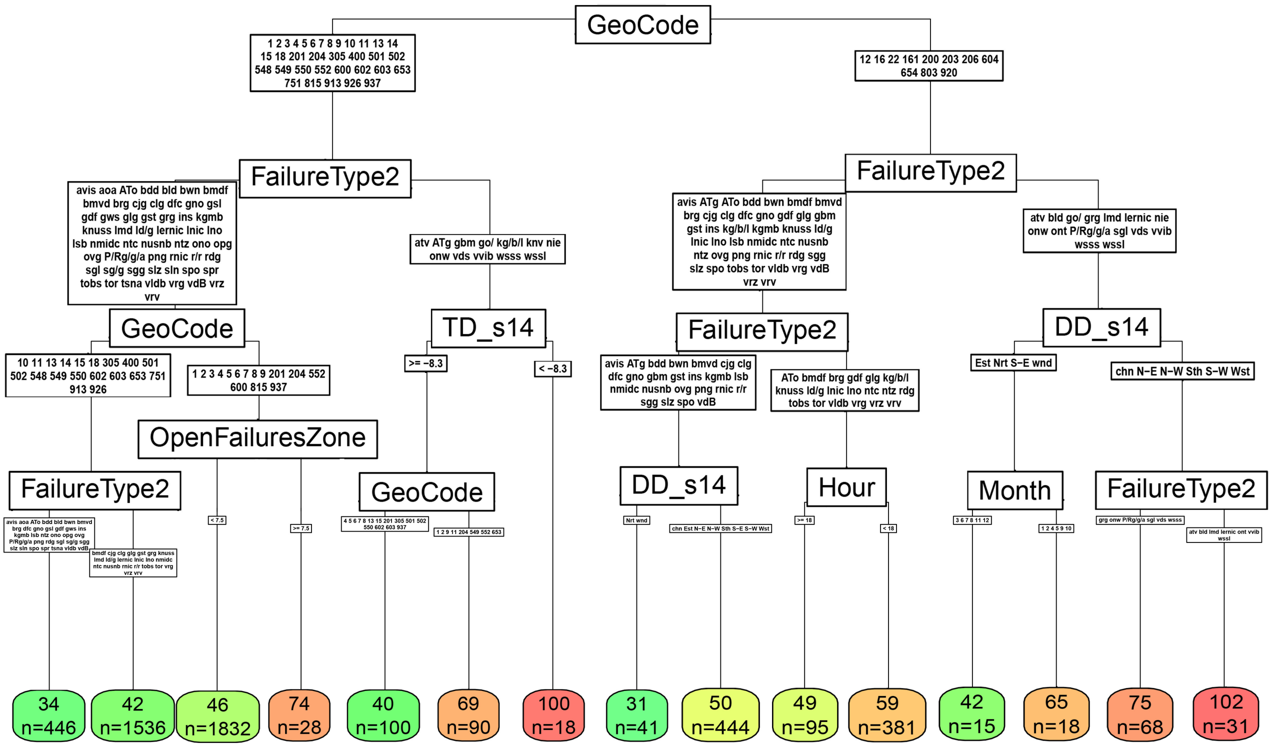


Figure 69: Tree of response time, Zone 2

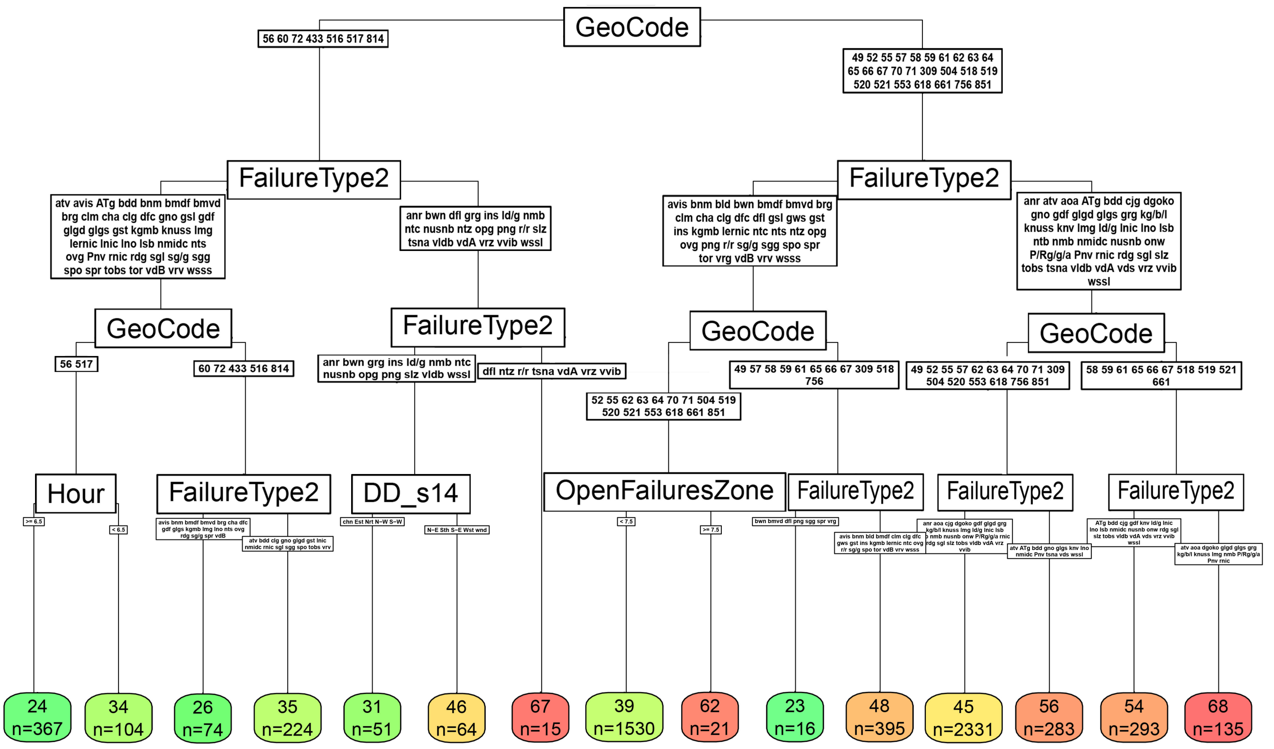


Figure 70: Tree of response time, Zone 3

Variable	Mean decrease in node impurity
Rainfall Duration	5400471
Temperature	4825519
Geographic Code	3715352
Dew Point	3608482
Rainfall	3443582
Day of the Week	3108017
Open Failures (Zone)	2990304
Wind Speed	2778887
Day	1463593
Failure Type Dutch	1317006
Wind Direction	1190420
Global Radiation	1168304
Latitude	1010350
Hour	818811
Object ID	785418
Longitude	745151
Sunshine Duration	630309
Failure Type	437907
Month	435034
Technical Department	361136
Object Type	267819
Year	33198

Table 32: Variable importance - Zone 1

Variable	Mean decrease in node impurity
Geographic Code	1630080
Wind Direction	1449348
Failure Type Dutch	545608
Wind Speed	533079
Temperature	366622
Sunshine Duration	292439
Object ID	272044
Failure Type	232208
Month	207238
Dew Point	186096
Hour	179779
Latitude	147337
Open Failures (Zone)	144468
Longitude	128134
Rainfall	114565
Day of the Week	111562
Object Type	96752
Rainfall Duration	87814
Global Radiation	84012
Technical Department	68608
Day	32498
Year	23479

Table 33: Variable importance - Zone 2

Variable	Mean decrease in node impurity
Geographic Code	1469241
Temperature	1214444
Failure Type Dutch	1186063
Wind Speed	664149
Rainfall	590212
Longitude	409382
Sunshine Duration	357500
Failure Type	296746
Rainfall Duration	266996
Hour	260897
Object ID	258066
Wind Direction	250876
Global Radiation	240097
Dew Point	212850
Object Type	211317
Latitude	202691
Day	195588
Day of the Week	137054
Month	121850
Open Failures (Zone)	121564
Technical Department	59600
Year	9791

Table 34: Variable importance - Zone 3

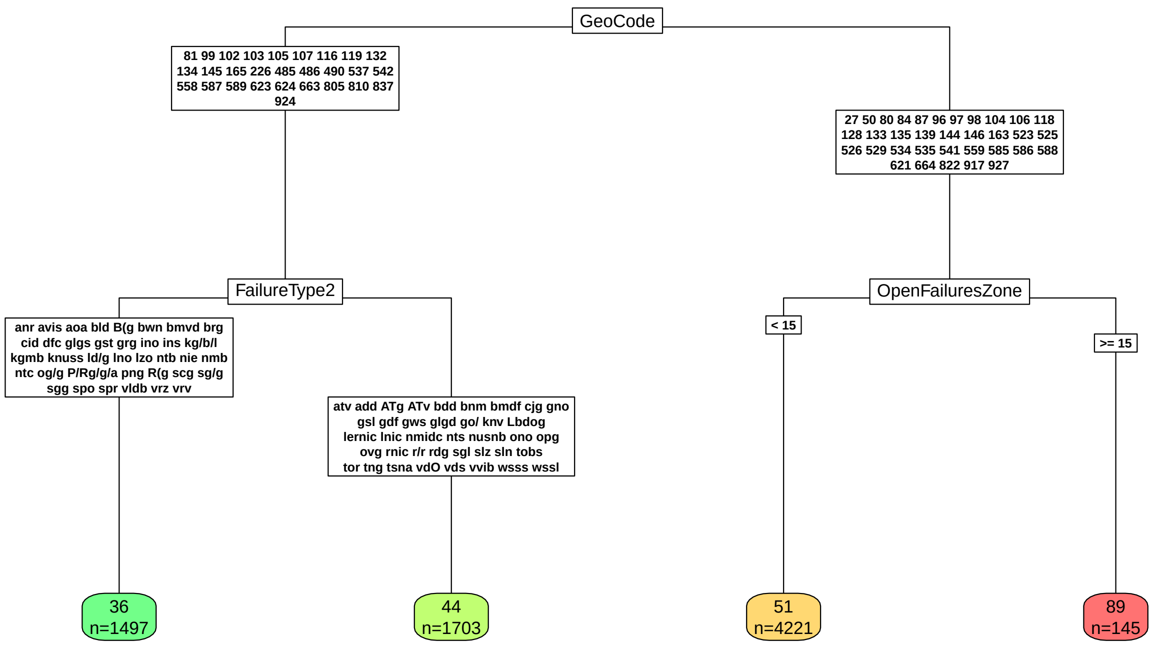

GeoCode
81 99 102 103 105 107 116 119 132 134 145 165 226 485 486 490 537 542 558 587 589 623 624 663 805 810 837 924
27 50 80 84 87 96 97 98 104 106 118 128 133 135 139 144 146 163 523 525 526 529 534 535 541 559 585 586 588 621 664 822 917 927
FailureType2
anr avis aoa bld B(g bwn bmvd brg cid dfc glgs gst grg ino ins kg/b/l kgmb knuss ld/g lno lzo ntb nie nmb ntc og/g P/Rg/g/a png R(g scg sg/g sgg spo spr vldb vrz vrv
atv add ATg ATv bdd bnm bmdf cjg gno gsl gdf gws glgd go/ knv Lbdog lernic lnic nmidc nts nusnb ono opg ovg rnic r/r rdg sgl slz sln tobs tor tng tsna vdO vds vvib wsss wssl
OpenFailuresZone
< 15
>= 15
36 n=1497
44 n=1703
51 n=4221
89 n=145

Figure 71: Tree of response time, Zone 1 (detail)

2.5.3.2.3 Function Restoration Time

The total time needed to complete the maintenance or repair intervention, as previously mentioned, is connected with both response and repair time. So, as would be expected, results relative to this variable present various similarities with the previous ones.
While Zone 1 and Zone 3 have a similar distribution of the cross-validated error (Figure 72), which decreases substantially until the size of the tree remains under about 10 leaves, in Zone 2 error presents a lower variability (Figure 73).
Unlike the previous models, the three trees referred to the function restoration time have different variables in the node at the top.
However, the pruned trees in Figure 77, 78 and 79 show how the first splits involve the same variables though they have a different disposition: Part Code, Mechanic's ID and Action Carried Out.
According to the mean decrease in node impurity, the features Mechanic's ID and Diagnosis Time are really relevant in all the three models, along with the features mentioned in 2.5.3.2.1. In fact, technical details of both the asset and the kind of intervention become again more relevant than collateral conditions, such as the weather ones, when the global duration of the repair action is considered.
Some common characteristics underpin all the models here illustrated:

- Weather importance seems to vary according with the zone considered: rainfall is the most significant condition in Zone 1, while wind plays a more relevant role in Zone 3. Temperature, instead, is almost equally important in all the areas;
- Models relative to Zone 1 generally outperform the other ones in terms of cross-validated error;
- The size of the final leaves is highly unbalanced in most cases, and the trees often tend to detect and isolate some small groups of observations with peculiar characteristics and, usually, extreme values of the response variable.

The use of regression trees in this chapter made it possible to explore the data in a thorough and deep way, highlighting specific dependencies and peculiarities without losing an overall view. This would not have been so easy with the only use of the classical statistical methods, especially as regards the visualization's aspect. The role of weather conditions, mechanic's experience and single failure causes or part codes have been described both in a graphic way and in a quantitative one; other than the points made in the previous paragraphs, meticulous analyses can be made also by laypeople on single branches in order to determine exactly which characteristics of the assets or of the failures are decisive in restoration time variations. These analyses could be exploited to plan and manage line possessions in an informed way.

2.5.4 Conclusions and future perspectives

In this work, various models were built based on real-world data.
In both studies the analyses demonstrate the reliability of the achieved results.
Therefore, it is possible to draw the following considerations:

- The predictive models developed in the first scenario can be considered satisfying. Random Forests resulted a good choice and the general results achieved are very interesting; the trained algorithms perform quite accurate predictions of time to restoration on previously unseen data, both for planned and extraordinary maintenance actions;
- Concerning the second scenario, the diagnostic models proposed are easy to interpret and immediately actionable. The cross-validated estimates of the error confirm that the diagnostic aim was more appropriate than the predictive one, based on the available data.

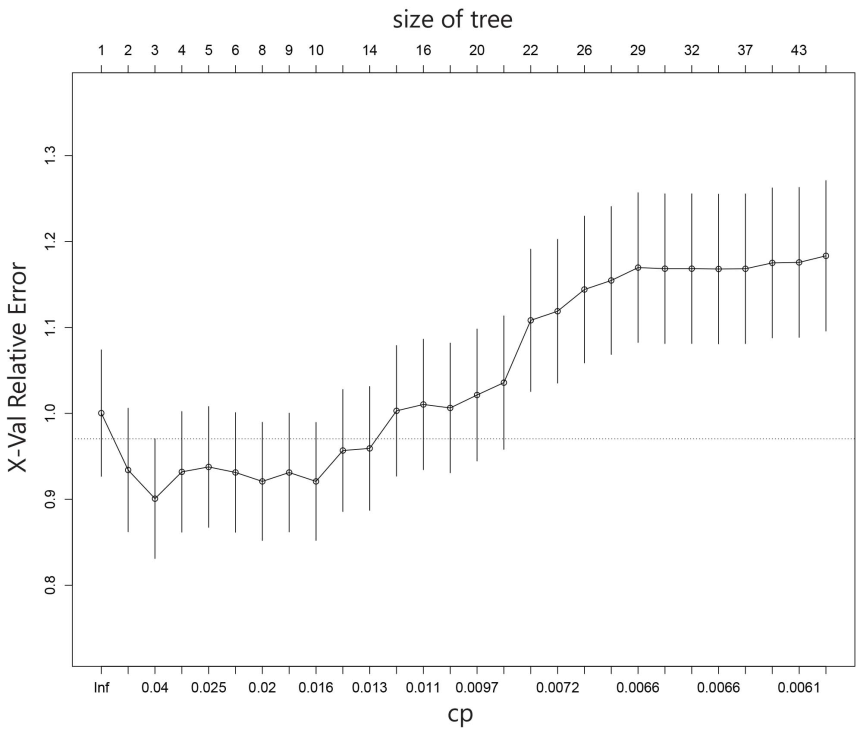


Figure 72: Cross-validated relative error and size of the tree (Zone 1)

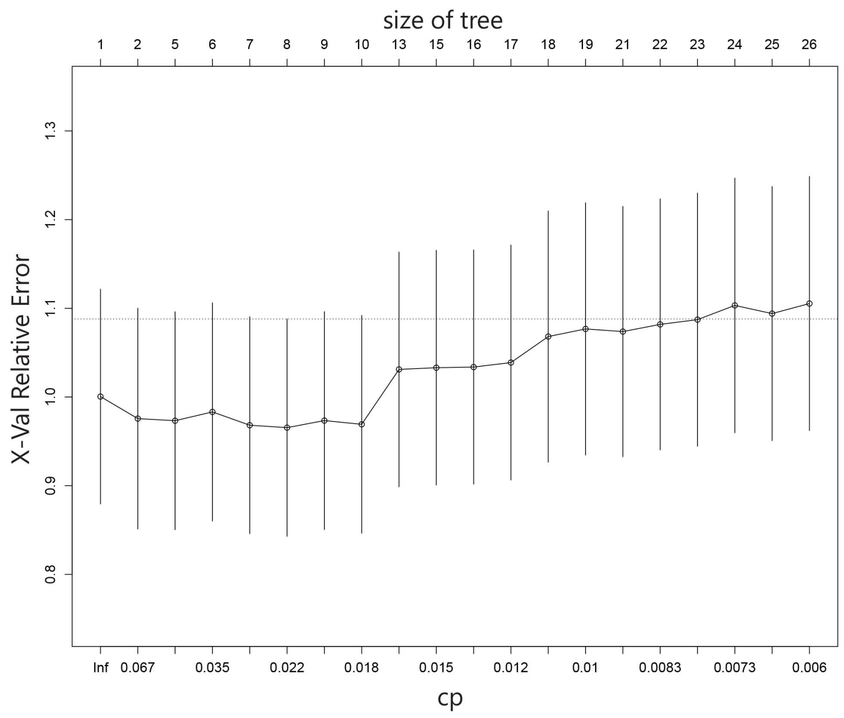


Figure 73: Cross-validated relative error and size of the tree (Zone 2)

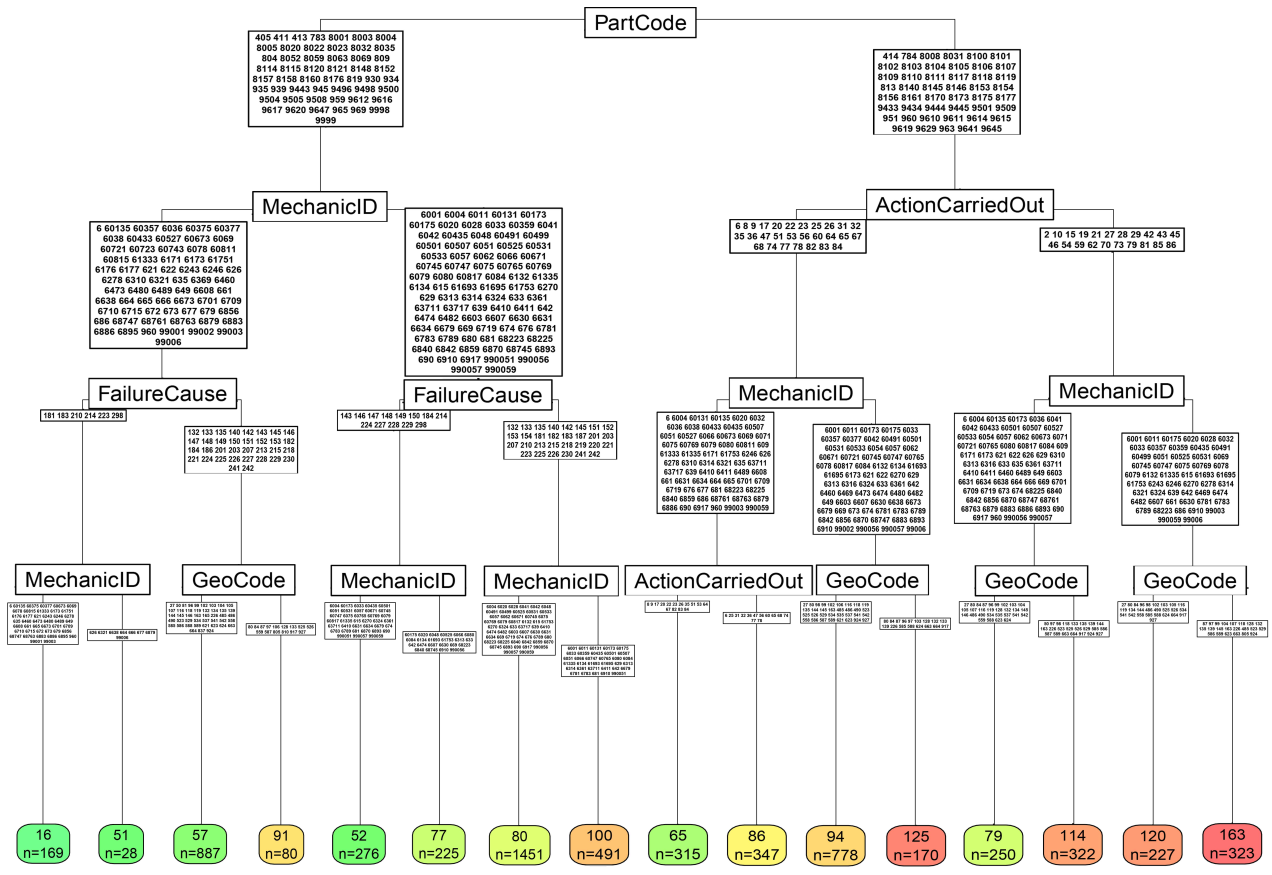


Figure 74: Tree of restoration time, Zone 1

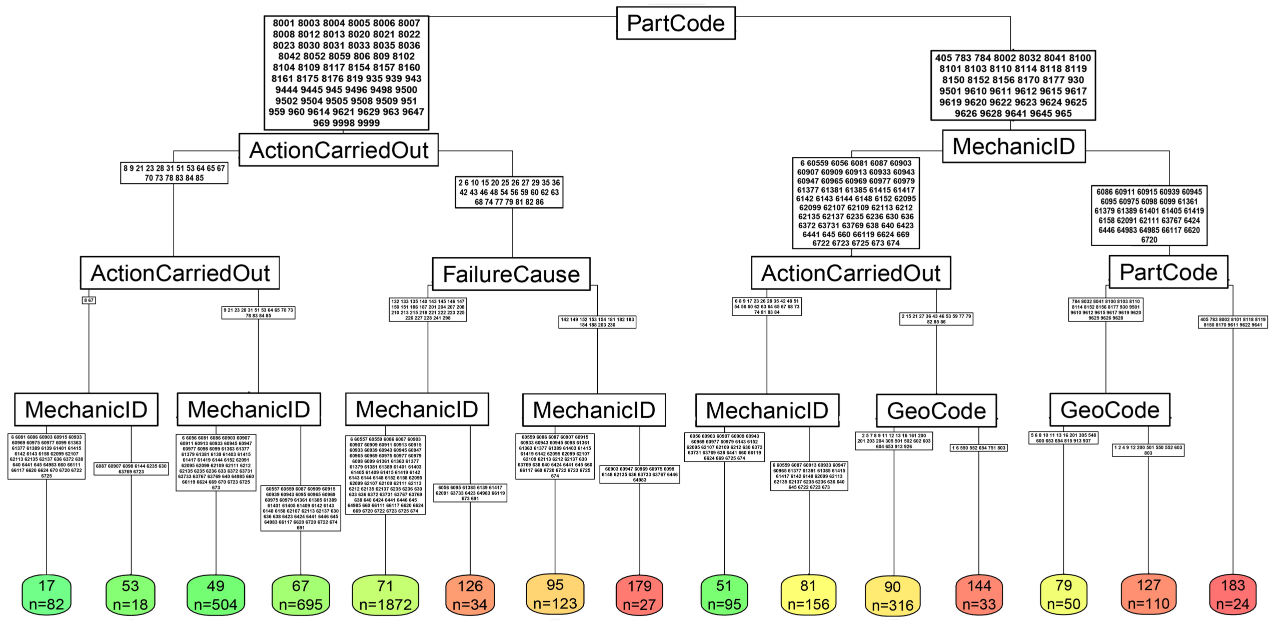


Figure 75: Tree of restoration time, Zone 2

Variable	Mean decrease in node impurity
Mechanic ID	14440222
Part Code	11223125
Geographic Code	9468236
Diagnosis Time	8713387
Failure Type Dutch	7457569
Action Carried Out	5958067
Failure Cause	4867516
Rainfall Duration	3951659
Dew Point	2872431
Wind Direction	2711325
Longitude	2587335
Object ID	2518884
Temperature	2504470
Failure Type	2464687
Latitude	2361075
Past Actions - Normed (same priority)	1690633
Global Radiation	1617133
Rainfall	1599123
Past Actions (Zone)	1555454
Month	1542107
Past Actions - Normed (Zone)	1449671
Day of the Week	1417079
Past Actions (same priority)	1411358
Past Actions - Difference from mean (same priority)	1333806
Past Actions - Difference from mean (Zone)	1182355
Wind Speed	1123994
Past Failures in the Week	1097194
Past Failures in the Month	1089590
Open Failures (Zone)	1056858
Past Failures in Six Months	1030395
Object Type	1024729
Day	977824
Hour	782953
Failure Main Group	608065
Sunshine Radiation	589260
Year of Activity	329971
Past Actions - Normed	308707
Past Actions	197615
Year	90303
Technical Department	87602
Past Actions - Difference from mean	10449

Table 35: Variable importance - Zone 1

Variable	Mean decrease in node impurity
Mechanic ID	5389728
Diagnosis Time	5037524
Geographic Code	4164375
Part Code	3891730
Failure Cause	2156727
Failure Type Dutch	1881605
Action Carried Out	1790935
Wind Speed	1606117
Dew Point	1429770
Sunshine Duration	1183283
Rainfall Duration	1125203
Past Actions - Normed	614242
Month	575149
Past Actions - Difference from mean (Zone)	425690
Object ID	398168
Past Actions - Normed (Zone)	391617
Past Actions - Normed (same priority)	368056
Past Actions - Difference from mean	362070
Wind Direction	361528
Failure Type	348569
Failure Main Group	309445
Hour	278130
Object Type	235522
Longitude	221823
Past Actions - Difference from mean (same priority)	194701
Rainfall	193141
Latitude	182003
Past Failures in the Week	167836
Year	165492
Year of Activity	165492
Temperature	161978
Global Radiation	144593
Past Actions (same priority)	137592
Past Failures in the Month	99623
Day of the Week	91957
Day	85032
Open Failures (Zone)	76867
Technical Department	73086
Past Actions	71976
Past Failures in Six Months	60626
Past Actions (Zone)	49624

Table 36: Variable importance - Zone 2

Variable	Mean decrease in node impurity
PartCode	7674926
Mechanic ID	6726293
Diagnosis Time	6361600
Failure Type Dutch	5207802
Geographic Code	4959113
Action Carried Out	4356728
Failure Cause	3675377
Failure Type	1555942
Object Type	1401239
Longitude	1390062
Latitude	1326941
Object ID	1289646
Wind Direction	1150418
Temperature	1132747
Wind Speed	1101381
Past Actions - Difference from mean (same priority)	1098718
Past Actions (Zone)	1049506
Dew Point	1016944
Hour	872887
Failure Main Group	788397
Past Actions - Difference from mean (Zone)	752338
Month	649657
Past Actions - Difference from mean	638289
Past Actions - Normed (Zone)	619202
Sunshine Duration	504639
Past Actions (same priority)	361134
Year of Activity	359088
Global Radiation	286013
Year	281467
Rainfall	266128
Technical Department	241479
Day of the Week	225496
Rainfall Duration	222142
Open Failures (Zone)	199855
Past Actions - Normed (same priority)	151241
Past Actions - Normed	122891
Past Failures in the Week	30024
Past Failures in Six Months	27471
Past Actions	14325
Day	10387

Table 37: Variable importance - Zone 3

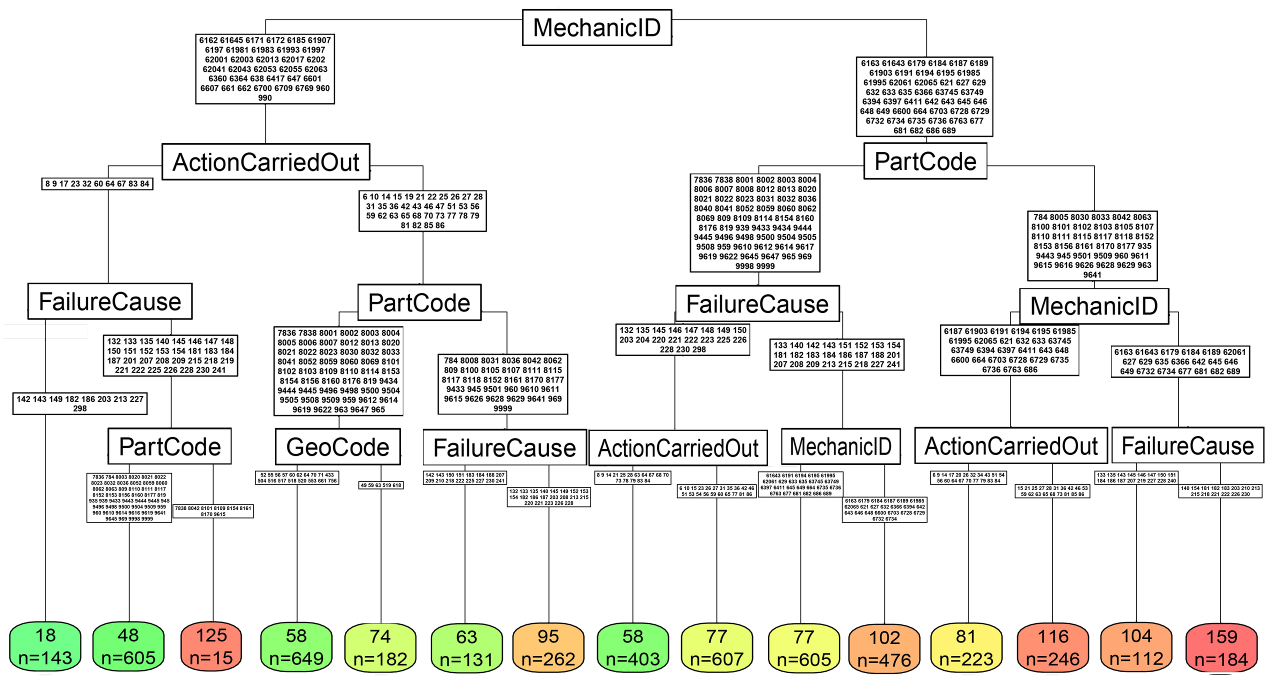


Figure 76: Tree of restoration time, Zone 3

It is interesting to note that -maybe not so surprisingly- in both cases the most effective split of the dataset was an area-specific one, despite the fact that Liguria and Netherlands obviously differ in dimensions, structure etc.

In both the scenarios, the quality of the models could be significantly enhanced by the availability of better data. In RFI's case, the dataset was quite tidy, but limited in number: both the amounts of observations and features were relatively small. SR data, instead, were complex and numerous, but quite noisy and, for some of the many different failure types, few examples were provided.

2.6 Specific-Scenario 4: Switches

Because of time and resource constraints, we decided to not to develop further the Specific-Scenario 4.

2.7 Specific-Scenario 5: Train Energy Consumption

The predictive models of this scenario will be developed in WP6 and their quality will be assessed by WP5 in D5.2.

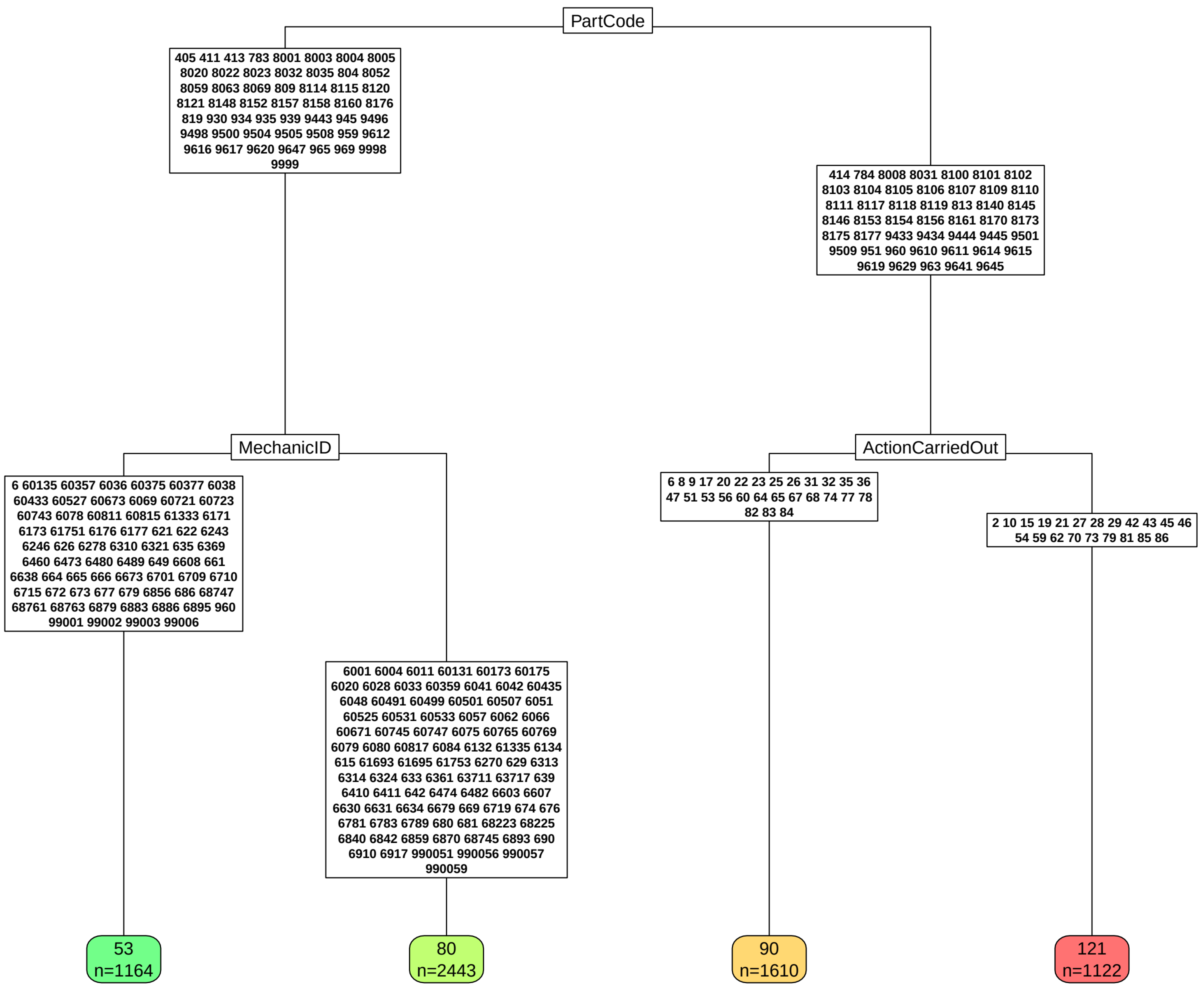


Figure 77: Tree of restoration time, Zone 1 (detail)

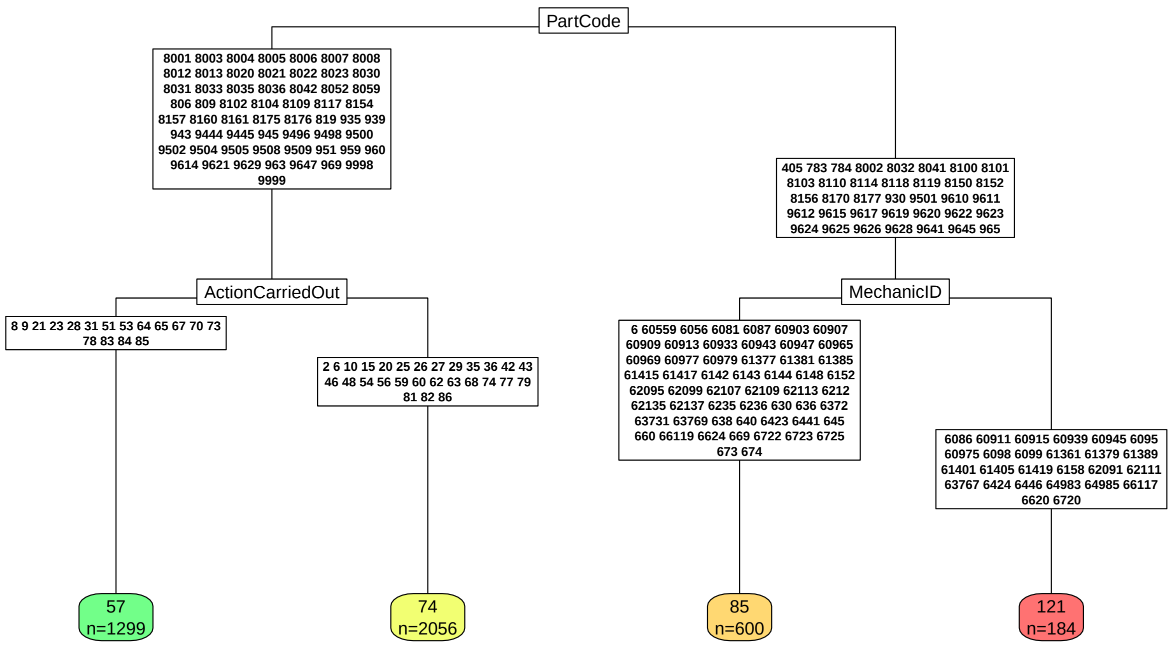


Figure 78: Tree of restoration time, Zone 2 (detail)

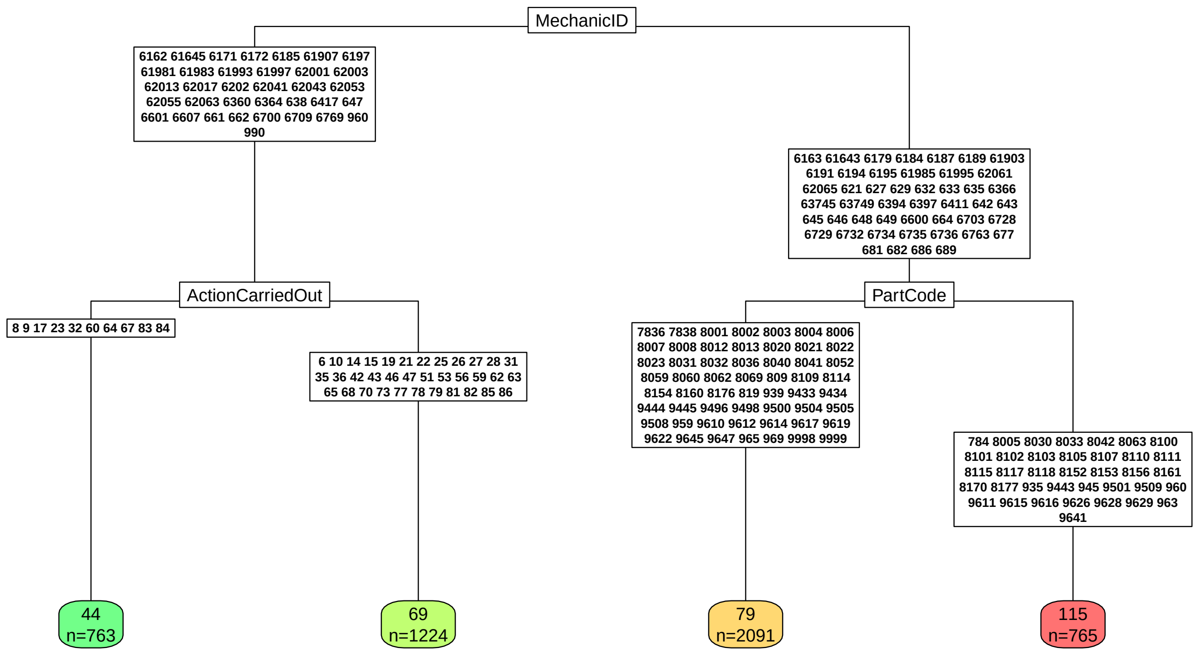


Figure 79: Tree of restoration time, Zone 3 (detail)

3 Conclusions

The general objective of WP5 is to study, design and develop data and visual analytics solutions for knowledge extraction from railway asset data. This deliverable reports to solutions and work-in-progress-solutions on relevant railway assets whose malfunction and maintenance policies have an impact on the KPIs targeted by the SHIFT2RAIL program. This deliverable presents the solutions and reports on the executed work in the scenarios that were defined in D5.1. The work reported in CS1 focuses on the visualizations used at RFI and optimizes and enhances them. Cross-Scenario 2 is discontinued because of the absence of the Marketplace of Data. Specific solutions and data-driven-models are developed in Specific-Scenario 1, 2, and 3. The carried out work prepares the final demonstrator (D5.4) which will combine the developments of CS1, SS3 and the blockchain-technologies developed in WP4. Cooperation with IN2DREAMS WS1 WP6 is assured by partner EVOLUTION ENERGIE and the coordination with other SHIFT2RAIL recipients is assured by the collaboration with IN2SMART and IN2RAIL.

References

[1] AI Experts want to end Black-Box Algorithms in Government. https://www.wired.com/story/ai-experts-want-to-end-black-box-algorithms-in-government.

[2] Angular (v6). https://angular.io/. Accessed: 2018-10-03.

[3] Bringing Artificial Intelligence to the Rail Industry. https://dataconomy.com/2015/11/bringing-artificial-intelligence-to-the-rail-industry.

[4] Data-driven documents, d3 (v5). https://d3js.org/. Accessed: 2018-10-03.

[5] God is in the machine. https://www.the-tls.co.uk/articles/public/ridiculously-complicated-algorithms/.

[6] Scalable vector graphics, svg (v2). https://www.w3.org/TR/SVG2/. Accessed: 2018-10-03.

[7] T. Albrecht. Reducing power peaks and energy consumption in rail transit systems by simultaneous train running time control. *WIT Transactions on State-of-the-art in Science and Engineering*, 39, 2010.

[8] J. Barta, A. E. Rizzoli, M. Salani, and L. M. Gambardella. Statistical modelling of delays in a rail freight transportation network. In *Proceedings of the Winter Simulation Conference*, 2012.

[9] A. Berger, A. Gebhardt, M. Müller-Hannemann, and M. Ostrowski. Stochastic delay prediction in large train networks. In *OASIcs-OpenAccess Series in Informatics*, volume 20, 2011.

[10] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[11] C. Borgelt. An implementation of the fp-growth algorithm. In *International workshop on open source data mining: frequent pattern mining implementations*, 2005.

[12] L. Bottou. From machine learning to machine reasoning. *Machine learning*, 94(2):133–149, 2014.

[13] L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[14] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[15] O. Brünger and E. Dahlhaus. Railway timetable & traffic-analysis, modelling. In *Simulation, Eurail press*, 2008.

[16] L. C. Cadwallader. Review of maintenance and repair times for components in technological facilities. In *INL/EXT-12-27734, Idaho National Laboratory*, 2012.

[17] W. Cohen. Fast effective rule induction. In *International Conference on Machine Learning*, 1995.

[18] W. Daamen, R. M. P. Goverde, and I. A. Hansen. Non-discriminatory automatic registration of knock-on train delays. *Networks and Spatial Economics*, 9(1):47–61, 2009.

[19] A. D'Ariano, T. Albrecht, J. Allan, C. A. Brebbia, A. F. Rumsey, G. Sciutto, and S. Sone. Running time re-optimization during real-time timetable perturbations. *Timetable Planning and Information Quality*, 1:147–156, 2010.

[20] A. D'Ariano, M. Pranzo, and Ingo A Hansen. Conflict resolution and train speed coordination for solving real-time timetable perturbations. *IEEE Transactions on Intelligent Transportation Systems*, 8(2):208–222, 2007.

[21] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, 2000.

[22] M. Donini, L. Oneto, S. Ben-David, J. Shawe-Taylor, and M. Pontil. Empirical risk minimization under fairness constraints. *arXiv preprint arXiv:1802.08626*, 2018.

[23] F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. In *arXiv preprint arXiv:1702.08608*, 2017.

[24] C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Innovations in theoretical computer science conference*, 2012.

[25] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[26] Geoffrey Ellis and Alan Dix. Decision Making Under Uncertainty in Visualisation? In *IEEE workshop on Visualization for decision making under uncertainty*. IEEE, October 2015.

[27] W. Fang, S. Yang, and X. Yao. A survey on problem models and solution approaches to rescheduling in railway networks. *IEEE Transactions on Intelligent Transportation Systems*, 16(6):2997–3016, 2015.

[28] H. Flier, R. Gelashvili, T. Graffagnino, and M. Nunkesser. Mining railway delay dependencies in large-scale real-world delay data. In *Robust and online large-scale optimization*, 2009.

[29] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

[30] J. Fürnkranz, D. Gamberger, and N. Lavrač. *Foundations of rule learning*. Springer Science & Business Media, 2012.

[31] Faeze Ghofrani, Qing He, Rob MP Goverde, and Xiang Liu. Recent applications of big data analytics in railway transportation systems: A survey. *Transportation Research Part C: Emerging Technologies*, 90:226–246, 2018.

[32] R. M. P. Goverde. A delay propagation algorithm for large-scale railway traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(3):269–287, 2010.

[33] R. M. P. Goverde and L. Meng. Advanced monitoring and management information of railway operations. *Journal of Rail Transport Planning & Management*, 1(2):69–79, 2011.

[34] I. A. Hansen, R. M. P. Goverde, and D. J. Van Der Meer. Online train delay recognition and running time prediction. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1783–1788, 2010.

[35] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[36] R. Hewett and P. Kijsanayothin. On modeling software defect repair time. *Empirical Software Engineering*, 14(2):165, 2009.

[37] P. W. Holland. Statistics and causal inference. *Journal of the American statistical Association*, 81(396):945–960, 1986.

[38] Weiyin Hong, James Y. L. Thong, and Kar Yan Tam. Does animation attract online users' attention? the effects of flash on information search performance and perceptions. *Information Systems Research*, 15(1):60–86, 2004.

[39] Wolfgang Jentner, Rita Sevastjanova, Florian Stoffel, Daniel A. Keim, Jürgen Bernard, and Mennatallah El-Assady. Minions, Sheep, and Fruits: Metaphorical Narratives to Explain Artificial Intelligence and Build Trust. In *Workshop on Visualization for AI Explainability*, 2018.

[40] R. Babuška K. Verbert, B. De Schutter. Fault diagnosis using spatial and temporal information with application to railway track circuits. *Engineering Applications of Artifical Intelligence*, 56:200–211, 2016.

[41] Daniel Kahneman and Patrick Egan. *Thinking, fast and slow*, volume 1. Farrar, Straus and Giroux New York, 2011.

[42] P. Kecman and R. M. P. Goverde. Process mining of train describer event data and automatic conflict identification. *Computers in Railways XIII: Computer System Design and Operation in the Railway and Other Transit Systems*, 127:227, 2013.

[43] P. Kecman and R. M. P. Goverde. Online data-driven adaptive prediction of train event times. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):465–474, 2015.

[44] Daniel A. Keim. Information visualization and visual data mining. *IEEE Trans. Vis. Comput. Graph.*, 8(1):1–8, 2002.

[45] Daniel A. Keim, Jörn Kohlhammer, Geoffrey Ellis, and Florian Mansmann. *Mastering The Information Age - Solving Problems with Visual Analytics*. Eurographics, 2010.

[46] Daniel A. Keim and Jim Thomas. Scope and Challenges of Visual Analytics, 2007. Tutorial at IEEE Visualization.

[47] H. Ko, T. Koseki, and M. Miyatake. Application of dynamic programming to the optimization of the running profile of a train. *WIT Transactions on The Built Environment*, 74, 2004.

[48] S. B. Kotsiantis. Decision trees: a recent overview. *Artificial Intelligence Review*, 39(4):261–283, 2013.

[49] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[50] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur. Improving rail network velocity: A machine learning approach to predictive maintenance. *Transportation Research Part C: Emerging Technologies*, 45:17–26, 2014.

[51] F. T. Liu, K. M. Ting, and Z. H. Zhou. Isolation forest. In *IEEE International Conference on Data Mining*, 2008.

[52] P. Lukaszewicz. *Energy consumption and running time for trains*. PhD thesis, Doctoral Thesis). Railway Technology, Department of Vehicle Engineering, Royal Institute of Technology, Stockholm, 2001.

[53] N. Marković, S. Milinković, K. S. Tikhonov, and P. Schonfeld. Analyzing passenger train arrival delays with support vector regression. *Transportation Research Part C: Emerging Technologies*, 56:251–262, 2015.

[54] F. P. G. Marquez, R. W. Lewis, A. M. Tobias, and C. Roberts. Life cycle costs for railway condition monitoring. *Transportation Research Part E: Logistics and Transportation Review*, 44(6):1175–1187, 2008.

[55] G. McLachlan, K. A. Do, and C. Ambroise. *Analyzing microarray gene expression data*. John Wiley & Sons, 2005.

[56] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[57] S. Milinković, M. Marković, S. Vesković, M. Ivić, and N. Pavlović. A fuzzy petri net model to estimate train delays. *Simulation Modelling Practice and Theory*, 33:144–157, 2013.

[58] T. Miller. Explanation in artificial intelligence: Insights from the social sciences. In *arXiv preprint arXiv:1706.07269*, 2017.

[59] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2012.

[60] A. Morant, P. O. Larsson-Kråik, and U. Kumar. Data-driven model for maintenance decision support: A case study of railway signalling systems. *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 230(1):220–234, 2016.

[61] M. Norouzi, M. Collins, M. A. Johnson, D. J. Fleet, and P. Kohli. Efficient non-greedy optimization of decision trees. In *Advances in Neural Information Processing Systems*, 2015.

[62] T. Nowakowski. Analysis of modern trends of logistics technology development. *Archives of Civil and Mechanical Engineering*, 11(3):699–706, 2011.

[63] L. Oneto, E. Fumeo, G. Clerico, R. Canepa, F. Papa, C. Dambra, N. Mazzino, and D. Anguita. Advanced analytics for train delay prediction systems by including exogenous weather data. In *IEEE International Conference on Data Science and Advanced Analytics*, 2016.

[64] L. Oneto, S. Ridella, and D. Anguita. Differential privacy and generalization: Sharper bounds with applications. *Pattern Recognition Letters*, 89:31–38, 2017.

[65] A. P. Patra. *Maintenance decision support models for railway infrastructure using RAMS & LCC analyses*. PhD thesis, Luleå tekniska universitet, 2009.

[66] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[67] J. R. Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[68] Regione Liguria. Weather Data of Regione Liguria. `http://www.cartografiarl.regione.liguria.it/SiraQualMeteo/script/PubAccessoDatiMeteo.asp`, 2018.

[69] Regione Lombardia. Weather Data of Regione Lombardia. `http://www.arpalombardia.it/siti/arpalombardia/meteo/richiesta-dati-misurati/Pagine/RichiestaDatiMisurati.aspx`, 2018.

[70] F. Restel. The markov reliability and safety model of the railway transportation system. In *Safety and Reliability: Methodology and Applications-Proceedings of the European Safety and Reliability Conference*, 2014.

[71] C. Robert. Machine learning, a probabilistic perspective, 2014.

[72] D. Sacha, M. Kraus, D. A. Keim, and M. Chen. VIS4ML: An Ontology for Visual Analytics Assisted Machine Learning. *IEEE Transactions on Visualization and Computer Graphics*, 2018.

[73] D. Sacha, A. Stoffel, F. Stoffel, B. C. Kwon, G. P. Ellis, and D. A. Keim. Knowledge generation model for visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1604–1613, 2014.

[74] P. Sanetti. *Data Analysis for Naval and Railway Transportation Systems*. PhD thesis, University of Genoa, 2017.

[75] Manojit Sarkar, Scott S. Snibbe, Oren J. Tversky, and Steven P. Reiss. Stretching the rubber sheet: A metaphor for viewing large layouts on small screens. In *ACM Symposium on User Interface Software and Technology*, pages 81–91. ACM, 1993.

[76] Michael Sedlmair, Miriah D. Meyer, and Tamara Munzner. Design study methodology: Reflections from the trenches and the stacks. *IEEE Trans. Vis. Comput. Graph.*, 18(12):2431–2440, 2012.

[77] B. Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[78] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[79] S. Si, H. Zhang, S. Keerthi, D. Mahajan, I. Dhillon, and C. J. Hsieh. Gradient boosted decision trees for high dimensional sparse output. In *International Conference on Machine Learning*, 2017.

[80] John T. Stasko and Eugene Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *INFOVIS*, pages 57–65. IEEE Computer Society, 2000.

[81] C. D. Stolper, A. Perer, and D. Gotz. Progressive visual analytics: User-driven visual exploration of in-progress analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1653–1662, 2014.

[82] T. H. Tsai, C. K. Lee, and C. H. Wei. Neural network based temporal feature models for short-term railway passenger demand forecasting. *Expert Systems with Applications*, 36(2):3728–3736, 2009.

[83] V. N. Vapnik. *Statistical learning theory*. Wiley New York, 1998.

[84] A. Vellido, J. D. Martín-Guerrero, and P. J. G. Lisboa. Making machine learning models interpretable. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2012.

[85] R. Wang and D. B. Work. Data driven approaches for passenger train delay estimation. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 535–540, 2015.

[86] G. Williams. Decision trees. In *Data Mining with Rattle and R*, 2011.

[87] Alexander Wolff. Drawing subway maps: A survey. *Inform., Forsch. Entwickl.*, 22(1):23–44, 2007.

[88] H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. *arXiv preprint arXiv:1602.08610*, 2016.

[89] Y. Yuan and M. J. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and systems*, 69(2):125–139, 1995.

[90] A. Zoeteman and E. Braaksma. An approach to improving the performance of rail systems in a design phase. In *World Conference on Railway Research*, 2001.