Transportation Research Part C 101 (2019) 35–54

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

# Predictive maintenance using tree-based classification techniques: A case of railway switches

Zaharah Allah Bukhsh[a,*], Aaqib Saeed[b], Irina Stipanovic[a], Andre G. Doree[a]

[a] Department for Construction Management and Engineering, Faculty of Engineering Technology, University of Twente, Enschede, Netherlands
[b] Department of Mathematics and Computer Science, Technical University of Eindhoven, Eindhoven, Netherlands

A R T I C L E I N F O

*Keywords:*
Predictive maintenance
Decision support
Machine learning
Railway infrastructure
Data-driven
Switches and crossings
Classification
LIME

A B S T R A C T

With growing service demands, rapid deterioration due to extensive usage, and limited maintenance due to budget cuts, the railway infrastructure is in a critical state and require continuous maintenance. The infrastructure managers have to come up with smart maintenance decisions in order to improve the assets' condition, spend optimal cost and keep the network available. Currently, the infrastructure managers lack the tools and decision support models that could assist them in taking (un) planned maintenance decisions effectively and efficiently. Recently, many literature studies have proposed to employ the machine learning techniques to estimate the performance state of an asset, predict the maintenance need, possible failure modes, and such similar aspects in advance. Most of these studies have utilised additional data collection measures to record the assets' behaviour. Though useful for experimentation, it is expensive and impractical to mount monitoring devices on multiple assets across the network. Therefore, the objective of this study is to develop predictive models that utilise existing data from a railway agency and yield interpretable results. We propose to leverage the tree-based classification techniques of machine learning in order to predict maintenance need, activity type and trigger's status of railway switches. Using the data from an in-use business process, predictive models based on the decision tree, random forest, and gradient boosted trees are developed. Moreover, to facilitate in models interpretability, we provided a detail explanation of models' predictions by features importance analysis and instance level details. Our solution approach of predictive models development and their results explanation have wider applicability and can be used for other asset types and different (maintenance) planning scenarios.

## 1. Introduction

The railway is second rapidly growing transport modal having the steady progress growth rate of +1.5% (EuroStat, 2016). As the railway networks get busier and more developed, the demands of availability, improved service quality, and reliable infrastructure have become more critical (de Bruin et al., 2017). With the rapid deterioration due to extensive usage, limited maintenance interventions due to budget cuts, and growing service demands, the need for infrastructure maintenance is continuously growing (ERF, 2013; European Railway Agency, 2014). Therefore, the infrastructure managers have to make maintenance decisions with the objectives to improve the assets' condition, spend optimal cost and keep the network available.

* Corresponding author.
*E-mail addresses:* z.allahbukhsh@utwente.nl (Z. Allah Bukhsh), a.saeed@tue.nl (A. Saeed), i.stipanovic@utwente.nl (I. Stipanovic), a.g.doree@utwente.nl (A.G. Doree).

https://doi.org/10.1016/j.trc.2019.02.001
Received 28 June 2018; Received in revised form 29 January 2019; Accepted 1 February 2019
Available online 14 February 2019

Traditionally, the maintenance decision is derived from system failures or repetitive schedules (Marquez, 2007). The former strategy results in increased cost and longer operational delays, while the latter leads to the extra cost of undue maintenance. In condition-based maintenance, a widely adopted strategy, the decision to perform the maintenance is driven by the asset condition state (Al-Douri et al., 2016). In practice, these decisions are mainly based on expert's judgements, available budgets, and repetitive schedules. The literature has numerous Mathematical Deterioration (MD) models that estimate the service lifetime of an asset. These models require multiple types of data e.g. asset's age, loading, weather, etc. for an accurate estimation of asset's service limit. Since the mentioned attributes are bound to change, MD models must be calibrated continuously. The MD models are able to provide sufficient decision support to infrastructure managers for planned maintenance only. But, in case of unplanned maintenance, the decisions need to be made efficiently in order to mitigate the problem with minimal network disruptions. Currently, the infrastructure managers lack the tools and decision support models that could assist them in taking (un) planned maintenance decisions effectively and efficiently. To facilitate and accelerate the process of maintenance decision-making with less uncertainty involved, we propose to leverage machine learning techniques by utilising a large amount of historical data of visual inspection, condition state, and maintenance records. Depending on the acquired data, the predictive models can estimate the performance state of assets, possible failure states, the best time for maintenance, and estimated operational delay minutes.

Machine Learning (ML) have brought multiple success stories from a number of industries ranging from health-care, finance, manufacturing, and marketing. ML is an umbrella term used to represent multiple computational methods for finding hidden patterns in data and estimating predictions (Cabitza and Banfi, 2018). Few ML solution for predictive maintenance of the railway assets has been reported in the literature. For instance, Li et al. (2014) developed the failure prediction models using heterogeneous data to improve the overall railway network velocity. Similarly, Kauschke et al. (2016) used the diagnostic logs for predicting the component failure of a cargo train. Another ML-based approach for detecting the metro train door failure is presented in (Manco et al., 2017). de Bruin et al. (2017) have employed the recurrent neural networks to identify and detect the failures in railway track circuits by using the signals from multiple track circuits. Based on the electrical power consumption of a switch engine, Bohm (2017) presented an approach to predict the remaining useful lifetime of a switch. A vision-based method based on a convolutional neural network is utilised by Chen et al. (2018) for detecting defects of the fasteners supported on the catenary device.

The aforementioned studies highlight the increasing trends towards the data-driven predictive maintenance solutions for rolling-stocks and rail infrastructure. However, the proposed methodologies are dedicated to specific problem context and cannot be generalised to other asset types. Moreover, most of these studies have employed additional data collection measures e.g. condition monitoring devices to record the assets' behaviour. Though useful for experimentation, it is expensive and impractical to mount monitoring devices on multiple assets across the network as also mentioned by de Bruin et al. (2017). In addition, the models developed using the sophisticated ML techniques (e.g., deep learning models, neural networks) are accurate, but their results are difficult to interpret and communicate to infrastructure managers. Considering the infrastructure managers' needs, the objective of this study is to develop maintenance prediction models that use existing data from railway agency and yield interpretable results.

The tree-based classification techniques for the maintenance prediction of railway switches and an approach for their interpretation is introduced in this study. The choice of tree-based classification techniques is motivated by the fact that they are comparatively easy to interpret and have empirically shown to provide optimal results for small and structured datasets (Hastie et al., 2009; Chen and Guestrin, 2016). Among numerous other types of assets, we choose to illustrate our models on railway switches. Switches and crossings regulate the traffic flow and are most critical points of the network. A switch consist of multiple components, e.g. points, frog, joints, etc and are vulnerable to many possible failures e.g. fatigue crack, wear failure, material deformations, etc. An unexpected failure in switches may leads to enormous consequences. For instance, a poor condition of the switches led to derailment of train in UK on 10 May 2002, which caused seven casualties. Similarly, a major switch failure on 21 August 2018 halted the traffic around Schipol, one of the busiest airport of Europe, and Amsterdam for 6 h. Ideally, an early warning system can facilitate in timely maintenance of switches in order to avoid such incidents altogether. As also mentioned earlier, the early failure prediction models demands the additional monitoring devices to be mounted on the switches, which is expensive and not practical for infrastructure owners. Therefore, it is paramount to explore the possibilities to improve the current maintenance planning systems in order to bring efficiency in maintenance decision-making process. Many railway agencies employ the SAP Enterprise Resources Planning (ERP) system to record inspections details and plan maintenance activities. We have utilised data from in-use business process of SAP ERP for the development of maintenance prediction models. The data consist of switches' basic properties and condition states, accompanied with specific inspection, maintenance triggers, and historical maintenance plans. The data is a provided by a railway agency though have been anonymised due to confidentiality reasons.

The primary contribution of this work is in proposing a practical solution approach for the efficient maintenance planning for discrete types of infrastructure assets. The example of such assets on railway network are bridges, tunnels, switches and crossing, drainage, and slopes. In this study, we develop tree-based classification models for railway switches only. These models are able to predict maintenance need, activity type and trigger's status using the existing data of a railway agency. To enable the transparency and trust of infrastructure managers on these models, a detailed explanation of models' outcomes is provided. A feature level importance analysis is performed to illustrate which features positively or negatively contributes to model's predictive power. Moreover, we have employed Local Interpretable Model-Agnostic Explanations (LIME) framework to provide a single instance level explanations.

The rest of the paper is structured as follows: Section 2 gives an overview of SAP maintenance request process and data sources. A brief introduction of tree-based classification approaches is provided in Section 3. Section 4 presents a development methodology and evaluation techniques of maintenance classification models. Section 5 presents the results of the models and discuss their performance differences. To interpret the models' predictions, the features importance and instance level interpretability details are

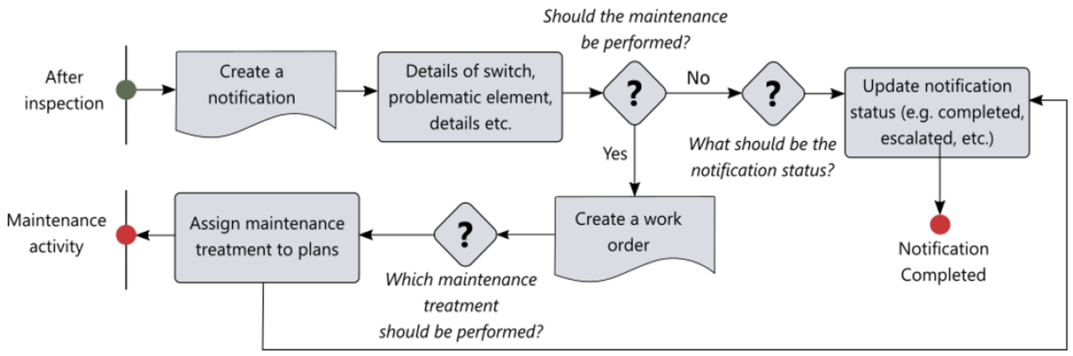

Should the maintenance be performed?
After inspection
Create a notification
Details of switch, problematic element, details etc.
?
No
?
Update notification status (e.g. completed, escalated, etc.)
Yes
What should be the notification status?
Maintenance activity
Assign maintenance treatment to plans
?
Create a work order
Notification Completed
Which maintenance treatment should be performed?

**Fig. 1.** Maintenance Request Process (MRP).

discussed in Section 6. Finally, Sections 7 and 8 presents discussion on the results and the conclusion of this study, respectively.

## 2. Overview of maintenance requests and data sources

SAP ERP is a software solution implemented in many railway agencies. The software manages the data and defines the key processes of a business. Among other processes, the Maintenance Request Process (MRP) is implemented by plant maintenance module of SAP ERP. An overview of MRP along with details of related data sources are explained in this section.

### 2.1. Maintenance request process

Railway switches are one of the most critical components on the railway network due to their intensive use, shorter lifespan, and importance for the operations. To keep the switches safe and running, the visual inspections are performed periodically. A MRP is initiated, if a switch is found to be in a poor state. Fig. 1 shows the detailed steps of a MRP.

As a result of an inspection, the condition state of each switch is updated in the system. If a switch is found to have a poor condition, the inspection manager generates a maintenance trigger (termed as *notification*). The created notification highlights the detected problem with a description, possible cause, and the proposed solution. Based on the reported details and respective switch properties, a maintenance engineer has to decide *if a maintenance should be performed or it can be delayed*. With the decision to perform the maintenance, a work-order is generated highlighting on *which type of maintenance activity to perform* in order to mitigate the problem. Based on the maintenance need decision, the *status of the notification* (also referred as maintenance trigger) is updated. In case a maintenance was performed then the status of notification is termed as completed. Otherwise, a notification attain the status as under observation, escalated to a manager, and technical non-compliance. The status of notification shows the actions taken by the responsible personnel as a result of a maintenance request.

This chain of actions as a result of maintenance trigger requires decision-making on mainly three stages, as highlighted in above paragraph with italics and with '?' sign in Fig. 1. We aim to develop classification models that could learn from historical data of notifications, work-orders and switches properties to predict the possible answer to these decision questions.

### 2.2. Datasets

We have used four different data sources where two of them are generated through the MRP. In the following, the details of each data source are provided:

1. *Asset register* consists of basic details of 802 switches constituting features like functional location, year of installation, direction, object type, and technical details.
2. *Condition data* provides the most recent information regarding the condition state of the switches. The condition state of a switch is determined by visual inspection, where a scorecard is used to rank the asset condition from good (represented by *1*) to very poor state (represented by *4*).
3. *Notifications file* consists of maintenance triggers that are generated for 802 switches from 2011 to 2017. As discussed earlier, each notification is created after a visual inspection and provides the details of a detected problem, its description, and causes along with a possible mitigation measure.
4. *Work-orders file* consists of different types of maintenance works, their description, and planned execution date. The list of maintenance activities provided in work-orders files are either initiated by direct orders due to availability of funding, cyclic planned activities, or unplanned triggers (i.e. notifications) reported as a result of inspection. We have selected only those maintenance activities that has associated notification code in order to predict the specific activity types on the basis of reported problem.

**Table 1**
Description of dataset.

| Data files | Asset register | Condition data | Notifications | Work-orders |
|---|---|---|---|---|
| *Datatype* | Multivariate | Multivariate | Multivariate | Multivariate |
| *Number of instances* | 802 | 4759 | 13,548 | 12,771 |
| *Number of attributes* | 44 | 10 | 22 | 22 |
| *Size of data (KB)* | 349 | 113 | 1510 | 1020 |
| Prediction problem | Data used for maintenance need and trigger's status prediction | | | |
| | | Data used for maintenance activity type prediction | | |

Table 1 provides a description of the each of the data file in terms of data types, size, number of instances and number of attributes (or features). Almost all the data files have mixed features types constituting of real number, integer values, text and categorical distribution. The data sources are interconnected with a unique equipment number, notification number, and work-order number. These unique numbers have enabled us to trace a single switch across the different data files. In order to get the basic characteristic of each switch, the notification file is combined with asset register on the basis of unique equipment number. The resulting file is further combined with condition data. For the *maintenance need, and trigger's status prediction*, the data from asset register, condition and notifications files are used as also depicted in Table 1. For the *maintenance activity type prediction*, all the four data files are used. The notifications are combined with assets register and condition data in a similar way as mentioned above. The resulting file is then combined with work-order file on the basis of order number. This results in selection of only those notifications which has resulted in certain maintenance activity. It is important to note that for each of the prediction tasks, we have created a distinct dataset by following the maintenance request process (see Fig. 1) and decision aspects of the practice.

## *2.3. Exploratory data analysis*

Before proceeding with exploratory analysis, basic data cleaning operations are performed. This includes replacing blank data fields with feature mode, renaming features' name, and conversion of data types (e.g. strings to categorical codes). In the following, few analytics and general remarks over the data are discussed.

Among the total 802 switches, most of the switches are found to lie within the age range of *10* to *35* years having a condition score of *1* or *2*. The Fig. 2 presents a box plot outlining the switches condition states with respect to their age. As expected, the switches having the older age have poorer condition states as compared to the younger switches. The box with corresponding whisker and outliers represent the total spread of the condition score relative to their age. The thick line in the rectangular box shows the median age with respect to each condition states. Switches having condition state *1* have data range between *1* to *35* years with median value of 15 years. The switches with condition state *2* have a median value near 20 years of age, which means even with large spread half of the switches with this condition are younger than *20* years. Except for the outliers, at conditions score *3* the median age range of the switches is very high as compared to others because of deteriorating asset condition. Though, it is important to note that we had *25* times more data points having condition state *1* compared with condition state *3*. On overall, the switches with older age shows a good condition state. Further analysis on the number of notifications with respect to switch age group is performed. The assumption is that the older switches will generate the higher number of unplanned maintenance triggers due to their deteriorating state. To validate this, we have computed a ratio of a number of notification per number of switches in the dataset with respect to their age range. The ratios are computed because our dataset consists of varying number of switches belonging to each age range. Fig. 3 shows a bar plot of a number of switches and number of notifications per age group. As mentioned earlier, most of the switches belong to the age range of 11–20 years, whereas, older switches tend to have a large number of notifications generated as compared to younger switches (see for age range 'Above 40' and '31–40' years).

Fig. 4 presents the total number of unplanned maintenance triggers (notification) generated from 2011 to 2017. The relatively low number of notifications in the year 2011, 2012 and 2013 is due to the gradual adoption of new SAP ERP system by the railway

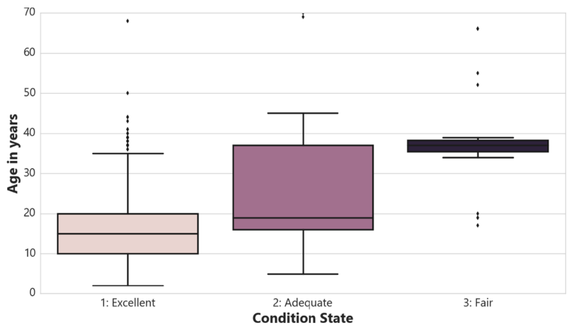


**Fig. 2.** Condition states of switches with respect to age.

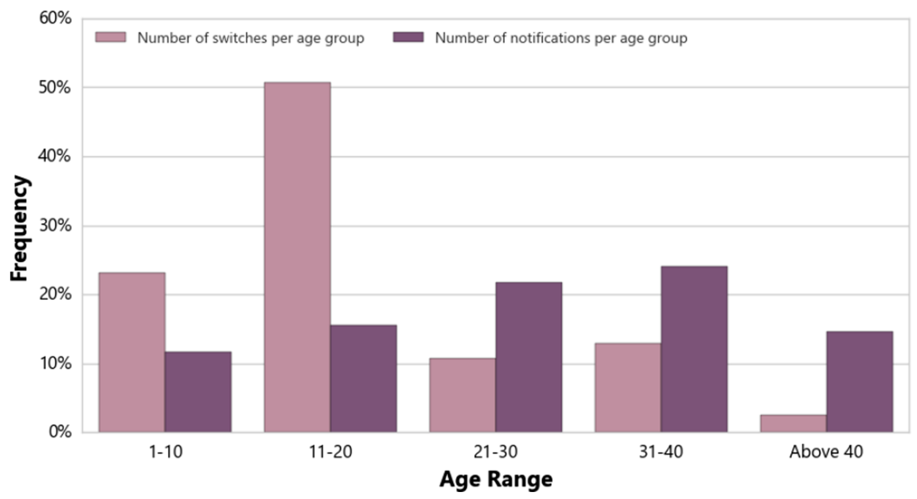


**Fig. 3.** Ratio of switches and notifications generated with respect to age ranges.

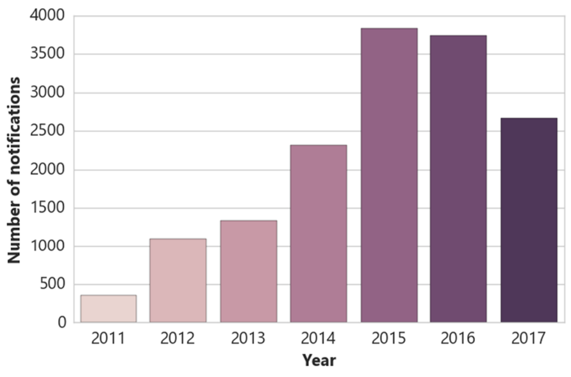


**Fig. 4.** Number of notifications generated in each year.

agency.

With each of the notification generated, the number of details such as detected problem, specific components, and the reason of the problem is stated. Fig. 5 provides an overview of the identified problem during the inspection. Among others, the frequency plot outlines only top ten most frequently reported problems.

Few interesting key aspects noted from data analysis are explained as follows:

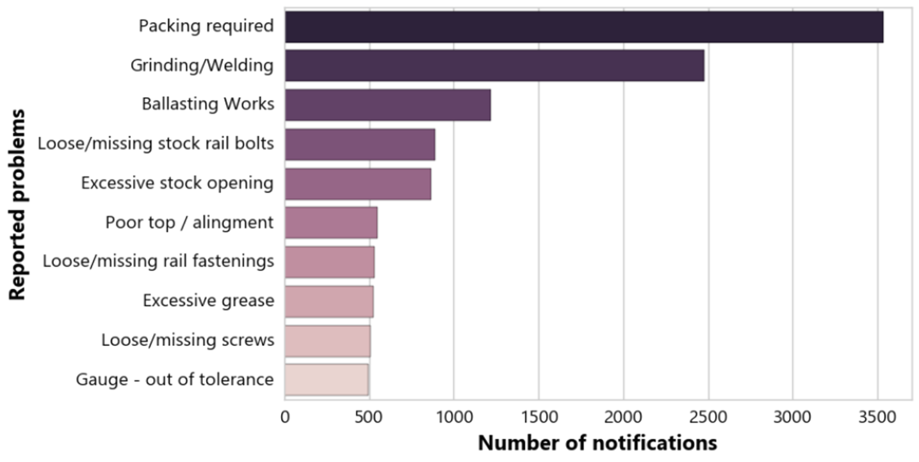


**Fig. 5.** Most frequently identified problems.

- Among the multiple components of a switch e.g. nose, crossing panel, toes, etc., most of the problems are detected at the switch panel, while, the least problematic components are a fish plate and gauge plate.
- Lifecycle deterioration and insufficient maintenance are top two most noted causes of problems reported.
- With the total number of notification created during 2011 and 2017, about 79% resulted in certain maintenance action.
- On average, 19 notifications are created for a single switch over the course of 7 years as a result of visual inspection.
- Among the 18 different maintenance activities noted in work-order, 90% of them belong to *M47* which represents complete switches maintenance. We have selected 6 most frequent occurring maintenance activities that have considerable data representation.

The analysis has provided us with an in-depth insight into the datasets. Considering the number of data attributes from multiple data sources, it is intractable for an infrastructure manager to consider the properties of previous events altogether, to decide about the maintenance need, maintenance activity type or maintenance trigger's status. Thus, the tree-based classification models developed in this paper will use the historical data and provide classification results, which can be used as the basis for the decision-making process.

## 3. Tree-based classification algorithms

Traditionally, a domain expert is required to analyse the data, establish the relationship among the features (or attributes), and derive standard rules. Since people are prone to making mistakes while analysing large amount of data, Machine Learning (ML) algorithms provide more efficient alternative for capturing the knowledge in data, and finding hidden patterns in order to facilitate in data-driven decision-making. The models developed using ML algorithms are successfully being used in web search, spam filtering, ad placement, credit scoring, fault detection and many such applications. There are mainly three types of ML algorithms, supervised, unsupervised and reinforcement learning (Raschka and Mirjalili, 2017, Chapter 1). Supervised learning algorithms require the labelled data to train the model in order to make predictions on future unseen data. The term supervised refers to the input data where the feature set and labelled-pairs are already available. In unsupervised learning, the data is without labels with unknown structure, where the algorithms tries to find the clusters (or relationship) of given data points. In reinforcement learning, an agent is developed which learns by receiving feedback from an environment in the form of some reward function. The self-driving cars and robotics are the main application area of reinforcement learning.

In our case, the data generated from the MRP is labelled, therefore we applied the supervised learning algorithms. Irrespective of type of learning, there are several ML algorithms to choose from for a specific prediction problem. In context of selecting best algorithm, the theorem of 'no free lunch' is popular in ML community. The theorem states that there is no single algorithm that perform best for all the problems and all the datasets. In other words, an algorithm which performs best for a particular problem might perform poor for others. The choice of an optimal algorithm depends on the size and format of the available data. The data format can be structured (tabular data) and of unstructured (e.g. audio, video, images) nature. An interested reader may refer to (Kotsiantis et al., 2007, Table 4) for the detailed explanation of supervised algorithms and their comparison for different predictive problems.

The MRP dataset, used in this study, is of structured nature where the columns represent the different features and rows represent the data instances. During the preliminary analysis, we considered several learning algorithms (such as logistic regression, support vector regression and neural net) and found that the tree-based learning algorithms perform reasonably well for our dataset. These tree-based classifiers have proven to provide optimal prediction results for the structured dataset in academic literature and in many industry challenges (e.g. Kaggle competitions). For instance, Fernandez-Delgado et al. (2014) conducted an extensive review of 179 classifiers of different types (e.g. Bayesian, neural network, support vector machine, boosting, bagging, and decision trees, etc.) on 121 structured datasets to evaluate their performance. The tree-based classifier namely random forest is found to be best with maximum accuracy. Similarly, in a review of supervised ML algorithms, Kotsiantis et al. (2007) concludes that rule-based system (e.g. decision tree) perform best when dealing with discrete (or categorical features), where support vector machine and neural network are better for the prediction of continuous attributes. With the reasonably good results in our preliminary analysis, and validated by the literature, we have selected the tree-based algorithms, namely decision trees, random forest and gradient boosted trees, for the development of maintenance predictive models. These tree-based algorithms are scalable and seamlessly solve both binary and multi-class classification problems.

In addition to ease of interpretability, tree-based models offer multiple advantages. For the small data-sets, tree based models are efficient to train and require minimal hyper-parameters tuning. These models perform internal feature selection by selecting the most prominent features as the root nodes. In contrast to various regression models, tree-based models do not require any feature normalisation, thus enabling the efficient implementation. A brief introduction of each of the considered tree-based model is provided below:

### *3.1. Decision tree*

The concept of Decision Tree (DT) has been implemented using different algorithms e.g. ID3 (Quinlan, 1986), C4.5 (Quinlan, 1993), and CART (Classification and Regression Trees) (Breiman et al., 1984). For DT development, CART and other algorithms work on a simple strategy of divide and conquer by employing the recursive partitioning of the training set. The key idea is to divide the training set into homogeneous subsets, where the resulting sets are pure and belong to the same target class. The algorithm quantifies the homogeneity or purity of multiple split nodes, in order to find a node split with maximum purity. The partitioning of a dataset is continued until the terminal node (i.e. target class) is reached.

Here, we have used the optimised version of CART implementation, which formulate a binary tree based on feature set as well as on thresholds. Let the data at node $m$ be represented by $\mathscr{S}$. For each candidate split $\theta = (j, t_m)$ consisting of a $jth$ feature and threshold $t_m$, partition the data into $\mathscr{S}_{left}(\theta)$ and $\mathscr{S}_{right}(\theta)$ subsets

$$\mathscr{S}_{left}(\theta) = (x, y)|x_j <= t_m \tag{1}$$

$$\mathscr{S}_{right}(\theta) = \mathscr{S} \setminus \mathscr{S}_{left}(\theta) \tag{2}$$

where $x$ is a feature value $f$ and $y$ represents the target class. CART generates a binary tree in which each parent node can have only two child nodes. Once a split based on $\theta$ is defined, the impurity at each node $m$ is computed.

With the target class values as $k \in y$: $k = \{1, 2, \ldots, K\}$ for node $m$, the proportion of class $k$ belong to node $m$ is calculated as follows:

$$p_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \tag{3}$$

where $x_i$ represents the feature value, $y_i$ represents the class label, $N_m$ and $R_m$ represent the total instances available at node $m$ and region of $m$ respectively. By using the $p_{mk}$, the gini index which is a measure of node impurity, is computed as:

$$H(X_m) = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) \tag{4}$$

The best node split by substituting the $H(X_m)$ takes the following form:

$$G(\mathscr{S}, \theta) = \frac{n_{left}}{N_m} H(\mathscr{S}_{left}(\theta)) + \frac{n_{right}}{N_m} H(\mathscr{S}_{right}(\theta)) \tag{5}$$

whereas the value of $\theta$ can be further optimised as $\operatorname{argmin}_{\theta} G(\mathscr{S}, \theta)$ to minimise the $H(X_m)$ impurity.

Having the minimisation of Gini index as an objective function, DT apply the top-down greedy search to determine the best split of nodes. The recursive partitioning algorithm is executed until the maximum allowable length of a decision tree is reached. The mathematical formulation explained above are taken from Pedregosa et al. (2011). In general, the DT model can be seen as a disjunction of conjunctions or as if-then rules, which are useful to aid human interpretability.

### 3.2. Random forest

The Random Forest (RF) is a term for an ensemble approach of DT, which consists of a number of trees. Unlike classical classification techniques which builds a single tree on a complete dataset, RF randomly selects the instances and features to construct multiple trees. Each DT then casts a vote for a particular target class and a class having the majority votes is model's prediction with a certain probability.

The RF learning algorithm can be explained as follows (Breiman, 2001; Guo and Berkhahn, 2016):

1. Get $N$ bootstrap samples from the dataset.
2. For a single sample set $i$ generate a decision tree $T_i$ as explained in Eqs. (1) and (2) of Section 3.1.
3. Output the ensemble of $T$ generated trees.
4. Classify a new test instance as:
   - For binary classification where $k = \{0, 1\}$, aggregate the prediction of $T$ trees with respect to each class, where, a class having majority votes is a model's prediction.
   - For multi-class classification where $k = \{1, 2, .., n\}$, it averages the predictions of $T$ trees as:

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} T_i(x) \tag{6}$$

In comparison with a traditional DT, RF shows good predictive performance even with considerable noise induced by random selection of features and instances (Hastie et al., 2009; Biau, 2012). Moreover, it is able to deal with large dataset having several features with diverse data types e.g. categorical or continuous values.

### 3.3. Gradient boosted tree

Gradient Boosting Tree (GBT) is an alternative ensemble learning technique that consecutively produces weak tree classifiers in a stage-wise fashion as other boosting algorithms do with a different base model (Friedman, 2001). The key idea of boosting methods is to strategically resample and incrementally build multiple models for training instances that are difficult to estimate with previous ones by minimising some arbitrary differentiable loss function e.g. cross entropy or sum of squared errors. This is a major distinguishing factor as compared to RF which produce each tree through random sampling with replacement. Principally, gradient boosting methods perform optimisation in the functional space in an additive form by sequentially adding an estimator $h(x)$ to an overall ensemble function $F_m(x)$ for generating a new model $F_{m+1}(x)$.

Given a squared error loss function, this method employs gradient descent algorithm to select a base-learner $h(x)$ that highly correlates with negative of the gradient as:

$$\begin{aligned} \rho_m &= argmin_p \sum_{i=1}^{N} [\mathscr{L}(y_i, F_{m-1}(x_i) + \rho h(x_i))] \\ F_m(x) &= F_{m-1}(x) + \rho_m h(x) \end{aligned} \tag{7}$$

where $N$ represents the total number of training instances, $\rho$ is the gradient descent step-size, and $\mathscr{L}$ is a loss function. More information on GBT methods can be found in Natekin and Knoll (2013). The GBT builds only one tree at a time, which results in longer training times as compared to RF. On the other hand, the sequential training on difficult examples provides a strong base to deal with unbalanced datasets.

## 4. Maintenance classification

To facilitate the effective and efficient maintenance decision-making, we propose to employ tree-based classification methods for the development of predictive maintenance models. The development of a ML model involves few preparatory tasks such as data pre-processing, class labelling, and definition of evaluation approaches. Data pre-processing aims to eliminate the noise, and prepare the feature sets that support in the classification process. In case the data is not explicitly labelled, class labelling step is performed to analyse the data in order to define the target labels. Finally, the evaluation approaches are defined to gauge the predictive power of the developed models. The following sub-sections provide a detailed explanation of each of these tasks.

### *4.1. Feature extraction*

Feature extraction is one of the most vital steps of ML model development. It involves the domain knowledge to select the most relevant features, combine existing attributes and often create new features from existing data. The focus is to identify the attributes that are most relevant to identify the target class, which will ultimately improve the predictive power of the model.

We initiated by eliminating any duplicating features such as, an age of switch and year of construction, unique identifiers, geo-coordinates of switches, and any features that do not directly contribute to a decision-making process. This selection procedure reduced the total number of features considerably. The selected features mainly include detected problem, switch component, problem reason and cause, functional location of a switch, track type, technical details, and age, etc. We did not perform further data processing, for instance, data normalisation or feature scaling, as the tree-based models are invariant to feature scales (Friedman et al., 2001). Though, the textual features were processed further to eliminate the irrelevant information by performing lower casing, removing English stop words and special characters. Since ML models are unable to process the textual data directly, we calculated term frequency-inverse document frequency (tf-idf) features from free-form text. Tf-idf is an information retrieval technique that assign weights to each word in an documents/attribute. The TF (term frequency) calculates the number of occurrence of each term in an document, where IDF (inverse document frequency) is a measure of how significant a word is to a document. The product of tf and idf form the total weight of a term. The higher is the weight of the term the more relevant it is to a document. Further details to compute the tf-idf is provided in Ramos (2003). The weights of tf-idf serves as a feature value, where we have selected top eighty text terms with highest tf-idf score for further analysis.

### *4.2. Label processing*

The supervised ML models require the labelled data for model training and testing purposes. Often the data instances are not explicitly labelled, therefore manual class labelling has to be performed.

The notification dataset does not contain any feature(s) that shows if the notification leads to a maintenance activity in past. Though, an analysis of notifications showed that a work-order number is attached to few notifications, which represents the type of maintenance performed, whereas other notifications have no associated work-order numbers, meaning no maintenance was performed. This notion leads us to label the notification dataset as follows:

$$l = \begin{cases} 0, & \forall\ m \text{ if } wo \in \varnothing \\ 1, & \forall\ m \text{ if } wo \in \ \mathrm{IR} \end{cases}$$

where $l$ represents data labels, $m$ is a data instance and $wo$ is a work-order number. For all the notifications $m$, where $wo$ belongs to a null set, the data is marked as *0* indicating no maintenance was performed. For those notifications, where work-order $wo$ was present and belonged to a real number, the data is marked as *1* indicating maintenance was performed. The distribution of class labels for the maintenance need prediction problem is provided in Table 2. Regarding the prediction of maintenance activity and trigger's status, the data was already labelled with distinct activity types and statuses, respectively.

Predictive models learn well when each class has at least one-tenth representation in overall training data. Ideally, a balanced

**Table 2**
Class representation for maintenance need prediction.

| Class label | Class name | Data representation |
|---|---|---|
| 1 | Maintenance performed | 79% (10,743) |
| 0 | No maintenance performed | 21% (2805) |

dataset has an approximately equal ratio of respective class instances for model learning. However, in case of an imbalanced dataset, the model tends to be biased towards the class having major representation, therefore performs poorly for minority class(es) (Akosa, 2017). With the class labelling, the dataset was found to be highly imbalanced in nature particularly for the maintenance activity and trigger's status prediction. Our preliminary analysis also presents the high bias of model towards majority classes, which results in poor classification ability. To mitigate data imbalance and to improve the classification precision, the classes having the large data representation must be under-sampled. The simple random sampling approach is used, where each data point has equal probability of selection on independent and identically distributed dataset. By random sampling, 13–15% of the data from majority classes has been selected (He and Garcia, 2008). This has balanced the overall representation of each class; thus reducing the considerable convergence to a single class only.

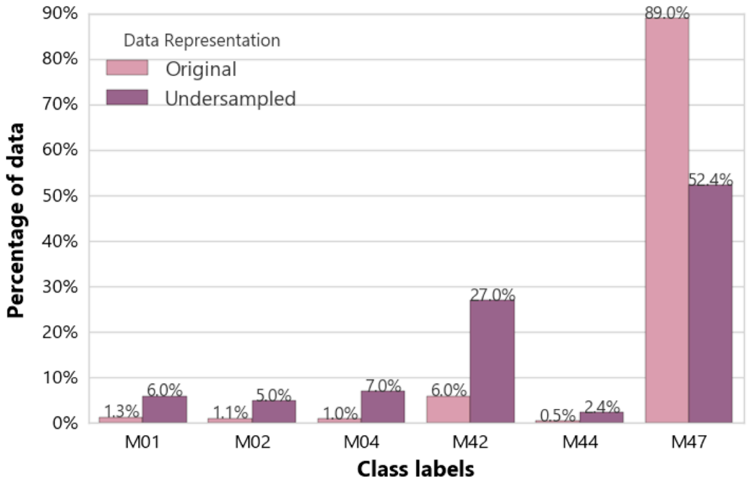
Fig. 6 shows the original and undersampled representation of maintenance activity type classes. *M47* has the highest number of instance for activity type prediction. The undersampling has reduced its representation from 89% to 52%, which consequently increased the data representation of other classes. Note that, the work-order instances presented in Fig. 6 will not add up to the number of instances presented in Table 1. This is because, among 18 different maintenance activities, we have selected top six most frequent occurring activities having sufficient data representation. Fig. 7 presents the original and undersampled class distribution for trigger's status prediction. *COM* has the highest share of data for prediction of maintenance triggers, which is reduced to 53% only. The undersampling of majority classes has resulted in increased representation of other classes in the dataset. The notable difference is for classes *M44* and *M04* (see Fig. 6) and for *ESC* and *OBS* (see Fig. 7).

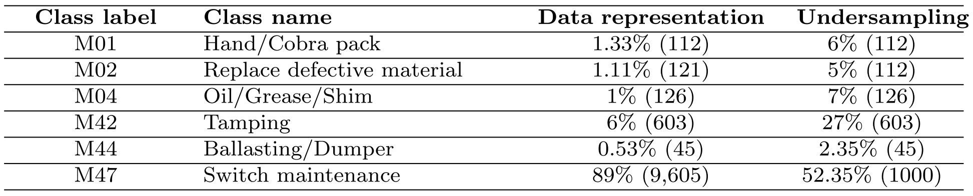
## 4.3. Evaluation approaches

The performance of the predictive models for maintenance classification is assessed through held-out test set and stratified cross validation, which is discussed in detail in this section. Moreover, multiple performance measures to gauge the models' predictive power are introduced here.

### 4.3.1. Held-out test set

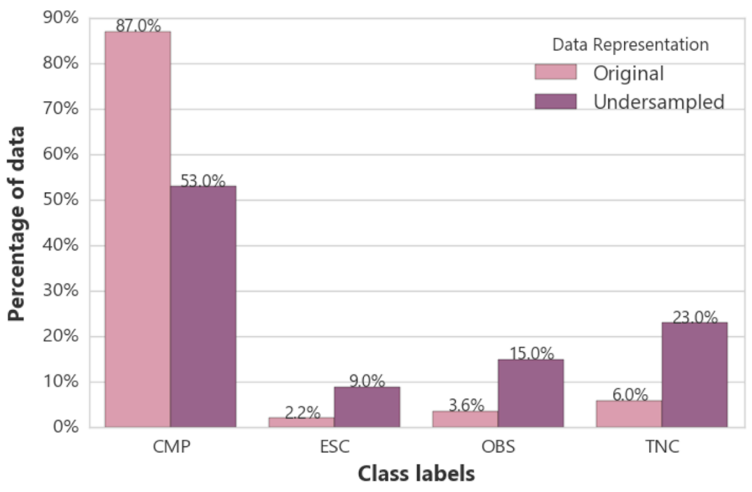
In a held-out method, the complete dataset is split into training and test sets. The training set is used to train the model and the





| Class label | Class name | Data representation | Undersampling |
|---|---|---|---|
| M01 | Hand/Cobra pack | 1.33% (112) | 6% (112) |
| M02 | Replace defective material | 1.11% (121) | 5% (112) |
| M04 | Oil/Grease/Shim | 1% (126) | 7% (126) |
| M42 | Tamping | 6% (603) | 27% (603) |
| M44 | Ballasting/Dumper | 0.53% (45) | 2.35% (45) |
| M47 | Switch maintenance | 89% (9,605) | 52.35% (1000) |

**Fig. 6.** Undersampling the major classes for maintenance activity prediction.



| Class label | Class name | Data representation | Undersampling |
|---|---|---|---|
| CMP | Completed | 87% (11,795) | 53% (1500) |
| ESC | Escalated | 2.16% (244) | 9% (244) |
| OBS | Observation | 3.65% (443) | 15% (443) |
| TNC | Technical non-compliance | 6% (781) | 23% (781) |

**Fig. 7.** Undersampling the major classes for maintenance trigger's status prediction.

test set is used to evaluate the model's performance. One of the most common approach is to randomly select the 70% of data instances for training and rest 30% for test. In our work, we have used a slightly different version of the held-out method, where the train and test split is defined based on the notification year. All the notifications generated before 2016 are used for training, while the notifications generated during and after 2016 are used for testing purposes. The split based on the year will evaluate the model performance more robustly, where the model was trained on only past data and tested-out on future unseen data. This is also in line with the purpose of the model, namely to predict the future maintenance needs. For reference about a number of notifications generated each year, see Fig. 4.

### *4.3.2. Stratified cross validation*

In Stratified Cross Validation (SCV), the whole dataset is randomly split into a number of equally sized units referred as 'folds'. Here, the term stratified represent the equal class distribution for each fold. Having $N$ number of fold, the $N-1$ are used for the training, while the $N_{th}$ fold is used for the model testing. This process is repeated $N$ times until each fold had the opportunity of being used as $N_{th}$ test and training fold (Witten et al., 2016). Finally, the output is averaged across all folds to estimate the performance of the model. This method ensures that every data point is used at least once as training example and once as a test example. The SCV is performed for the completeness of validations and for the evaluation of model's robustness.

### *4.3.3. Performance measures*

Several performance measures are used to compare and evaluate the models' prediction power. In classification problems, we used confusion matrix analysis which represents the models' predicted classes of test data for which the true values are already known. Table 3 shows the confusion matrix for a binary classification presenting positive and negative as class values. The True Negatives (TN) are cases belonging to *negative* class and are correctly classified as *negative* by a model. Similarly, the True Positives (TP) are number of cases belonging to *positive* class and are correctly classified as *positive*. In contrast, False Positives (FP) are *positive* samples that are incorrectly classified as a *negative* class. While False Negative (FN) are *negative* class samples incorrectly classified as positive. A normalised confusion matrix with perfect classification has TN and TP of *one* and FP and FN of *zero*.

With the confusion matrix, a number of performance measures can be calculated (Flach, 2012, Chapter 2). The metrics used in this study for evaluation purposes are explained as follows:

**Table 3**
Confusion matrix for binary classification problem.

| | Classified negative | Classified positive |
|---|---|---|
| Actual negative | True Negatives (TN) | False Positives (FP) |
| Actual positive | False Negatives (FN) | True Positives (TP) |

**Accuracy** is a measure of correct predictions of the model compared to the total data points. It shows how often the model classifies the instances correctly. It is computed as:

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

The accuracy is a good measure where the data is balanced for each class labels. However, in case of imbalanced data set, the accuracy measure without other performance measures can be misleading.

**Misclassification rate** determines how often the model has misclassified an instance compared to the total given instances. For example, a positive instance assigned to a negative class by the model. Misclassification rate is calculated as:

$$\text{Misclassification rate} = 1 - \text{Accuracy} = \frac{FP + FN}{TP + TN + FP + FN}.$$

This measure is mainly used for the binary classification problems, as the calculating the rate of misclassification for multi-class problems is complicated.

**F-Score** is a combination of precision (or positive predictive value) and recall (sensitivity) measures (Sokolova and Lapalme, 2009). The precision determines the exactness of the model. It is a ratio of correctly predicted positive instances (TP) to the total positively predicted instances (TP + FP). Precision is represented as:

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

In contrast, recall provides a measure of model's completeness. It is a ratio of correctly predicted positive instance to the total instance of positive class (TP + FN) in test data. Recall is calculated as:

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

Precision represents the model's performance with respect to false positives, whereas recall represents the performance with respect to false negatives. The F-score convey the balance between precision and recall by taking their weighted harmonic mean. F-score is calculated as follows:

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

Similar to the accuracy, F-score performs well with the fairly balanced dataset. In case of imbalanced dataset, the adjusted F-measure is utilised.

**Kappa** presents an inter-rater agreement between qualitative items, which measure the relative observed agreement ($p_o$) with the hypothetical probability of chance agreement ($p_e$) (Cohen, 1960). The kappa measure does not only calculates the percentage accuracy, but also consider the possibility of agreement between raters (qualitative items) by chance. The value of kappa is calculated as follows:

$$\text{Kappa} = \frac{p_o - p_e}{1 - p_e}$$

In case of imbalanced datasets, the kappa is a robust measure compared to F-score and accuracy. It can be said that kappa determines how well a model performed ($p_o$) as compared to how well it could have been performed by chance ($p_e$), while considering the marginal distribution of a target class.

Given the performance evaluation measures, the idea is to maximise the TP and TN and minimise the FN and FP. However, in real-world applications, there is always a trade-off between true and false positives, and negatives rate (Batista et al., 2004) depending on the business needs and problem context. For instance, in the context of maintenance need prediction the false negatives (model predict no need of maintenance, where it is needed) are more critical compared to the false positives.

## 5. Results

For the development of maintenance classification models, machine learning python library scikit-learn (Pedregosa et al., 2011) is employed. Three distinct models based on DT, RF, and GBT algorithms are trained with scikit-learn default hyper-parameters configurations. Only the number of trees for RF and GBT algorithm are manually tuned within a range of 100–2000 trees. It is found that the model performs optimal given the 1500 trees, where no improvement in performance is noted by increasing number of trees.

It is important to note that each of the predictive model is developed independently, where every model utilises its own set of input features. By the rigorous features selection, we have used only those input features set for modelling of prediction tasks that are available during real-time decision-making. Further, the performance of these trained models are evaluated by considering the DT as a baseline. The results explain which model performs the best for a particular classification problem. The developed predictive models will be able to predict maintenance need, maintenance activity type and maintenance trigger's status as represented in Fig. 1.

**Table 4**
Results of maintenance need prediction with held-out test.

| Model | Accuracy | Misclassification | F-score | Kappa |
|---|---|---|---|---|
| Decision Tree (DT) | 0.844 | 0.151 | 0.847 | 0.540 |
| Random Forest (RF) | 0.857 | 0.142 | 0.853 | 0.548 |
| Gradient Boosted Tree (GBT) | **0.862** | **0.137** | **0.860** | **0.575** |

### 5.1. *Maintenance need prediction*

Three distinct models are developed *to predict if a maintenance will be performed or delayed* based on the respective switch properties and problems reported. Table 4 shows the model evaluation results tested on held-out dataset. All models show a negligible difference in performance. The GBT model performed best (depicted with bold in Table 4) with f-score of 86% with 13.7% misclassification rate.

Note that, for held-out evaluation, the models were trained on a dataset of notifications generated from 2011 to 2015, while tested on data of 2016 and 2017. This warrants that the developed models will be able to perform sufficiently well in real-world settings having the notifications generated in future. Fig. 8 shows the confusion matrix analysis to evaluate the classification accuracy. Confusion matrix analysis provides a summary of correctly and incorrectly classified instances with respect to each class. The confusion matrix provided in Fig. 8a shows that 62% of times the DT model correctly classified a 'no' maintenance as 'no' (true positives), whereas 38% of times decision tree model has incorrectly classified a 'no' maintenance to 'yes' (false positives). Similarly, 91% of times the model correctly predicts 'yes' (true positives), whereas 9% of times it is incorrectly classified as 'no' (false negative). For all three models, most of the misclassification is attributed to false positives (top right value). This means the model predicted that a maintenance should be performed while in reality the decision was not to perform the maintenance. By using the frequency distribution per class label provided in Table 2, the approximate number of (in) correctly classified instances can be calculated. Though RF shows the highest number of false positives (see Fig. 8b), DT has in total highest misclassification rate of 15%. In the context of maintenance, the false negatives are more critical as compared to false positives since in those cases the model suggests not to perform the maintenance; while the maintenance is needed. In addition to the trade-offs between false positives and false negatives, the misclassification by model can be substantially reduced by gathering more training data for minority classes or by applying techniques like SMOTE (Synthetic Minority over-sampling techniques) that generates data points by interpolation from the minority class samples (Chawla et al., 2002). Another solutions is to manually inspect false positives and false negatives by single instance-level interpretability and by relative feature importance. Further details on this are provided in Section 6.

The trained models are also evaluated with 10-fold Stratified Cross Validation (SCV). Table 5 presents the results of SCV, which shows the averaged scores across 10-folds along with standard deviations. Based on SCV, the GBT model still performed better than others. Overall, all the models show considerable performance improvements compared with models of the held-out test set. The reason for this is due to the difference in validation approach (See Sections 4.3.1 and 4.3.2). With SCV the model is trained and tested multiple times on randomly created data folds. The data from a single year can be used in both training and testing folds. Fig. 9 presents the averaged confusion matrices analysis for SCV. With both validations, it can be noted that the models are able to learn well for both true positives and negatives whereas false positives and negatives are critical to learn.

### 5.2. *Maintenance type prediction*

The purpose of these models is *to predict the specific maintenance activity type*. As mentioned earlier, we have selected only 6 most occurring maintenance activities that have sufficient data representation for each class. The description of each selected maintenance type and details of their class representation is provided in Fig. 6. Based on the problem reported, any one of these 6 maintenance activity types can be chosen. Often the selection of maintenance activity type is based on experts' judgement. We have developed three distinct models that learn from the historical data and predict the most suitable maintenance activity type.

Table 6 shows the models classification results as evaluated with held-out dataset. RF model achieved the highest accuracy having

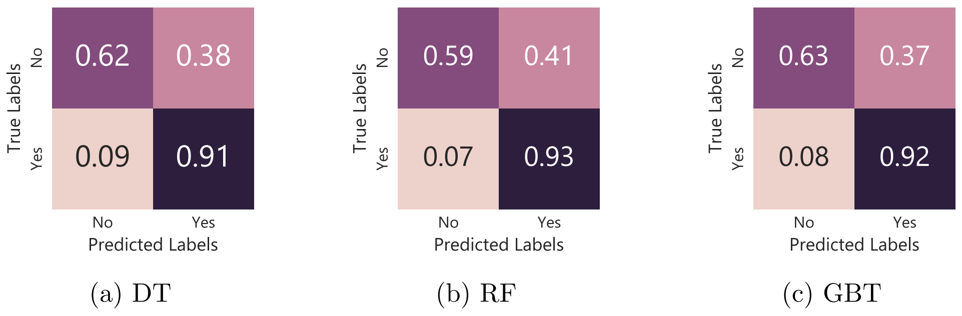
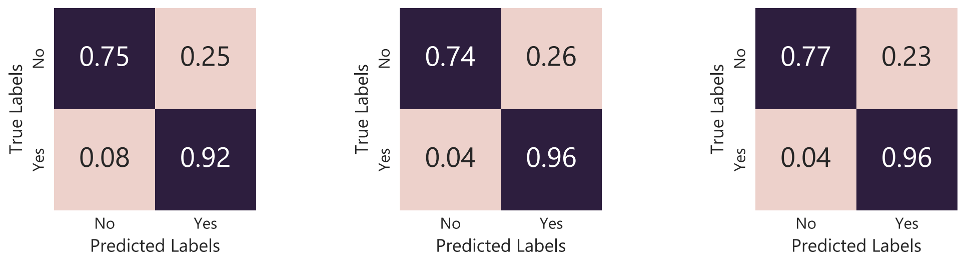


**Fig. 8.** Confusion matrices of maintenance need prediction on held-out test set.

**Table 5**
Results of maintenance need prediction with SCV (Averaged scores and standard deviations).

| Model | Accuracy | Misclassification | F-score | Kappa |
|---|---|---|---|---|
| DT | 0.877 | 0.114 ± 0.006 | 0.886 ± 0.006 | 0.658 ± 0.019 |
| RF | 0.913 | 0.087 ± 0.008 | 0.911 ± 0.008 | 0.725 ± 0.028 |
| GBT | **0.923** | **0.079** ± **0.005** | **0.919** ± **0.006** | **0.752** ± **0.018** |



(a) DT (b) RF (c) GBT

**Fig. 9.** Confusion matrices of maintenance need prediction with SCV.

**Table 6**
Result of maintenance type prediction with held-out test.

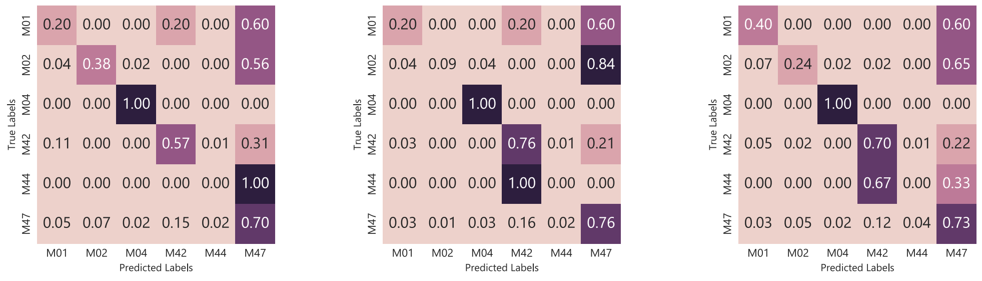
| Model | Accuracy | F-score | Kappa |
|---|---|---|---|
| Decision Tree (DT) | 0.630 | 0.659 | 0.330 |
| Random Forest (RF) | **0.70** | **0.70** | **0.427** |
| Gradient Boosted Tree (GBT) | 0.678 | 0.695 | 0.409 |

the f-score and kappa of 70% and 42.7% respectively. Maintenance type prediction is a multi-class problem, therefore it is not possible to calculate the straightforward misclassification rate. Though, the misclassification of each model for a particular class can be analysed with confusion matrices provided in Fig. 10. For all the models, the classification of maintenance type M01, M02 M42 and M44 are confused as M47 by varying ratios. For instance, M44 is wrongly classified by all the models. Similarly, M02 is poorly classified by all through the accuracy of DT is better than RF and GBT. The reason of M02 and M42 misclassification can be attributed due to the only 5% and 2.35% class representation, respectively (see Fig. 6). In general, the overall misclassification by the models is due to varying class distribution in train and test dataset.

These models are also evaluated with 10-fold SCV and results are represented in Table 7. The GBT model achieved an average f-score of 76.8% and kappa of 68.3%. Likewise, the RF model has further improved the results having average f-score of 78.9% and 69.9% of kappa. Fig. 11 shows the confusion matrices analysis of each model. Overall DT shows the highest number of misclassification rate. But, for classes *M02* and *M44* DT performed better than RF. Similarly, GBT have classified *M01*, *M02* and *M44* better than RF. However, RF shows the higher percentages of accurately predicted classes (see the diagonal of Fig. 11b), which makes it the best classifier for activity type prediction.

### 5.3. Maintenance trigger's status prediction

The status of maintenance trigger (i.e. a notification) is decided based on the problem reported. Instead of maintenance only, the



(a) DT (b) RF (c) GBT

**Fig. 10.** Confusion matrices of maintenance type prediction with held-out test set.

**Table 7**
Results of maintenance type prediction with SCV (Averaged scores and standard deviations).

| Model | Accuracy | F-score | Kappa |
|---|---|---|---|
| Decision Tree (DT) | 0.708 | 0.740 ± 0.018 | 0.602 ± 0.029 |
| Random Forest (RF) | **0.789** | **0.803 ± 0.02** | **0.699 ± 0.038** |
| Gradient Boosted Tree (GBT) | 0.768 | 0.792 ± 0.021 | 0.683 ± 0.032 |

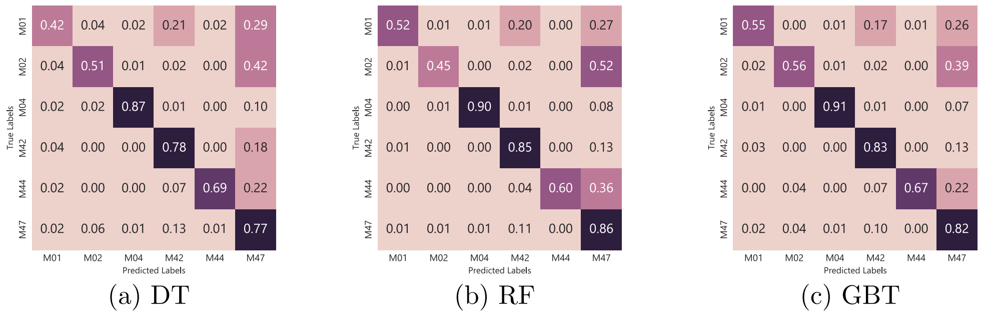


**Fig. 11.** Confusion matrices of maintenance type prediction with SCV.

status of a maintenance trigger defines the further actions, which includes technical non-compliance (TNC), escalated (ESC), observation (OBS) and completed (COM). The details of class distribution is presented in Fig. 7. We have developed three predictive models based on DT, RF and GBT algorithms *to predict the status of a maintenance trigger*.

Table 8 shows the evaluation results of models on held-out test set. The performance of RF model with f-score of 77.7% and kappa of 62.3% is best among all the models. Fig. 12 shows the confusion matrices which presents relatively poor classification, where most of the classes are misclassified as *COM*. This is because of the highest class representation of this class (see Fig. 7). Comparatively, there are fewer misclassification for *TNC* because it has a sufficient number of data samples for the model to be able to learn accurately.

Table 9 shows the models evaluation results with SCV. The RF model performed best with 82.4% f-score and 73.1% kappa. Fig. 13 shows the results of classifications with confusion matrices. All the models have misclassified the *OBS* with a *COM* status with a certain ratio. This is because, the postponing of a maintenance activity leads to an additional monitoring or regular visual inspection, here designated as *OBS*. Since *OBS* is a just a delayed maintenance activity, these cases are most likely to share their properties with the cases of *COM*. The similar properties of these cases may have led to the misclassification by the models.

## 6. Interpretability

For automated decision-aid systems, the transparency and interpretability have crucial importance. The ML models are notorious for being black boxes, which provide little to no explanation of their predictions. Many recent studies have emphasised the importance of interpretable models (Doshi-Velez and Kim, 2017; Lipton, 2016). The focus is towards providing an explanation about which input features positively or negatively impact the model's output (Ribeiro et al., 2016) and to explain the model behaviour locally for a specific instance or a case (Datta et al., 2016). The model interpretability is also very important for domain experts in order to be able to trust and employ predictive models in their daily decision-making scenarios (Andreou et al., 2018). Moreover, the details on features importance and instance details also enables the further improvement of the model by reducing the critical false negatives and possible false positives. This section provides an in-depth analysis of the model behaviour with respect to the features importance and local instance level explanations.

### 6.1. Features interpretebility

Among the number of features that are used for the model training, not all of them equally contribute towards learning a model. With tree-based classification models, the importance of a feature learned by a predictive model can be estimated. The feature

**Table 8**
Result of maintenance trigger's status prediction with held-out test.

| Model | Accuracy | F-score | Kappa |
|---|---|---|---|
| Decision Tree | 0.707 | 0.714 | 0.494 |
| Random Forest | **0.791** | **0.777** | **0.623** |
| Gradient Boosted Tree | 0.737 | 0.733 | 0.532 |

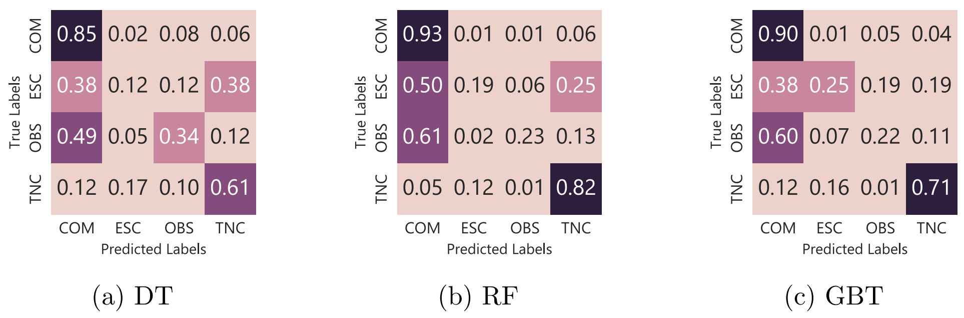


**Fig. 12.** Confusion matrices of maintenance trigger's status prediction with held-out test set.

**Table 9**
Results of maintenance trigger's status prediction with SCV (Averaged scores and standard deviations).

| Model | Accuracy | F-score | Kappa |
|---|---|---|---|
| Decision Tree | 0.783 | 0.768 ± 0.019 | 0.641 ± 0.03 |
| Random Forest | **0.827** | **0.824 ± 0.018** | **0.731 ± 0.03** |
| Gradient Boosted Tree | 0.834 | 0.823 ± 0.020 | 0.727 ± 0.03 |

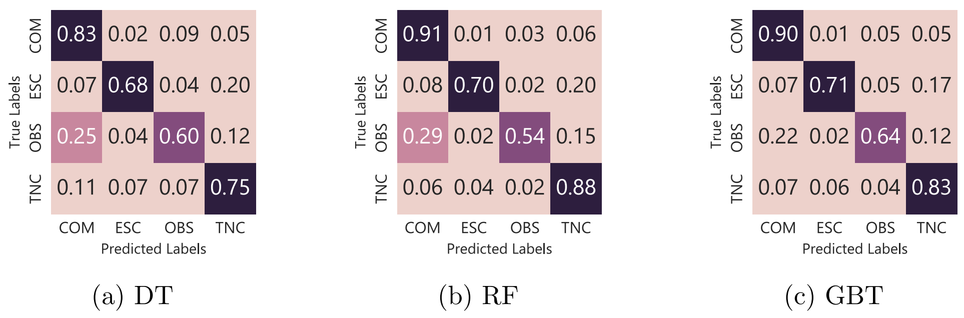


**Fig. 13.** Confusion matrices of maintenance trigger's status prediction with SCV.

importance score quantifies the importance of each attribute in overall prediction of the model. The high importance score of an attribute suggests that the attribute is very important and positively contributes towards the models predicative capability. However, the negative importance score shows that the attribute is negatively influencing the performance of model. This means by eliminating the attributes having negative importance score, the performance of the model is likely to improve. To enable the comparison among multiple attributes, we have calculated the feature importance of only RF models. Along these lines, RF has also performed best for two of the three maintenance predictive models discussed in Section 5. There are multiple methods to compute the feature importance of predictive models, namely, default mean decrease accuracy, permutation importance, and drop-feature importance (Raschka and Mirjalili, 2017, Chapter 4). Drop-feature importance is one of the most accurate method to compute features importance but have a high computation cost. Since we have small dataset, we choose to employ drop-feature importance method, which demands model (re) training multiple times.

The main idea of the drop-feature method is to train a RF model on all feature set, as explained in Section 5, and term its accuracy/error estimation as baseline represented as $ps_b$. Based on the $n$ number of features, the RF model is trained $n$ times wherein each iteration a single feature $i$ is dropped and accuracy of the model is recomputed. The new accuracy score is represented as $ps_n$. The single feature importance of $i$ is then calculated as $\Delta = ps_b - ps_n$, which a difference in baseline and drop in overall accuracy score. This process is repeated for $n$ number of features until the importance of each feature is computed. The higher is the difference between the resulting model's accuracy and baseline the more important a feature is for model's prediction. Note that, these computed values provides a relative importance of features that may not add to one (Strobl et al., 2008).

In order to elaborate the models' results, the feature importance level for each classification model are developed. Fig. 14 provides the feature importance of a maintenance need prediction model. It shows that the *functional location*, *age* and *stretcher type* are main contributing factors when deciding to perform or delay the maintenance. The *problem cause*, *switch component* and *problem reason* are negatively influencing the models' predictions. This is in contrary to general assumptions that the detected problem and notifications details drive the *need of maintenance* decision.

For the prediction of maintenance treatment type, the feature importance graph is presented in Fig. 15. The *detected problem* and *functional location* are most important for maintenance type prediction whereas the *track type* and *switch component* are least

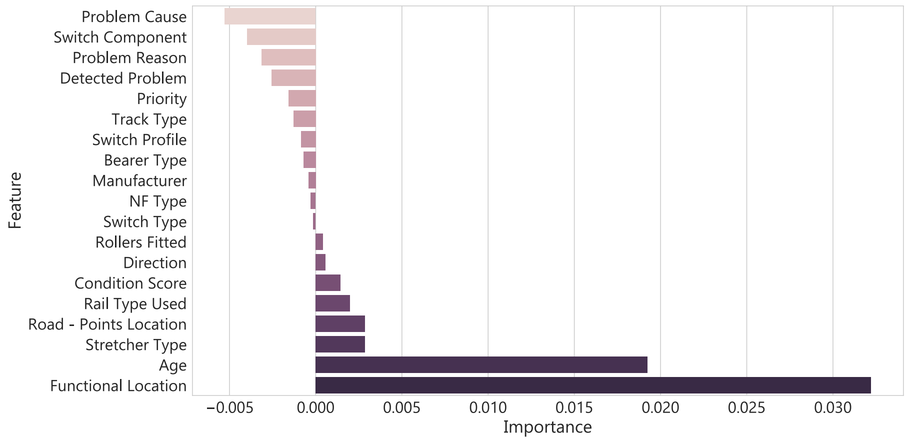


**Fig. 14.** Features importance of a RF model for maintenance need prediction.

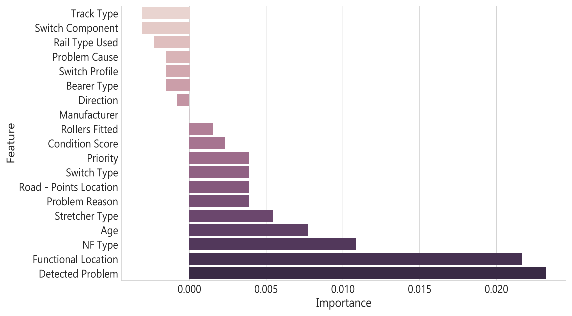


**Fig. 15.** Features importance of a RF model for maintenance type prediction.

important. Notice the *manufacturer* attribute has zero importance, which means if we eliminate this feature from dataset the performance of the model will not likely be affected. Moreover, if the features, such as *direction*, *bearer type*, having the importance level less than zero are eliminated, the overall performance scores of the model is likely to improve.

Fig. 16 shows the feature importance of RF model for the prediction of maintenance triggers' status. As expected *detected problem* followed by *functional location* are relatively most important features to decide on maintenance triggers' status. While, *condition score* and *switch component* features can be eliminated since they have no importance and are not contributing in model's final predictions.

Comparing the feature importance graphs of Figs. 15 and 16, it is interesting to note that given the similar feature set for training each model remarks the different number of features as important for the model's performance. There are multiple benefits of performing feature importance analysis. First, it enables domain experts and decision-makers to gain an in-depth understanding model's results by analysing the features importance. Second, it illustrates how the models' decision pattern diverts from the real decision-making practices as we noted in importance graph of maintenance need prediction (see Fig. 14). Third, it can help in the feature selection and extraction procedure in order to train the models only on positively contributing feature set. Fourth, complying with business rules, it enables us to drop those features that are not directly relevant for the decision-making process.

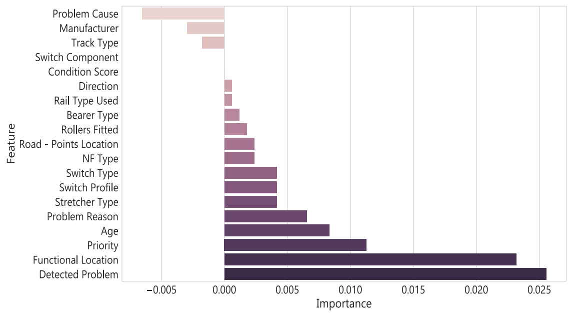


**Fig. 16.** Features importance of a RF model for maintenance triggers' status prediction.

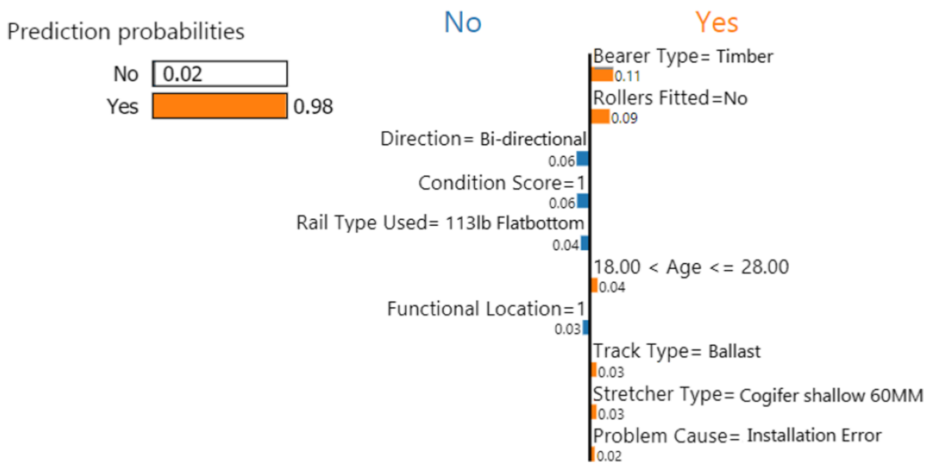


**Fig. 17.** LIME explanation of maintenance need prediction model.

## 6.2. Instance level interpretebility

Feature importance analysis provides the understanding of model's prediction as a whole. However, the need to contemplate why and how a model classifies an unseen (new) instance to a specific label still remains. In order to explain the model's prediction for a single test instance, we have employed Local Interpretable Model-Agnostic Explanations (LIME) framework (Ribeiro et al., 2016). LIME explains the predictions of any model locally in an interpretable manner. This enables the domain experts and decision-makers to understand, interpret and possibly further improve the model's performance.

The LIME explanation of maintenance need prediction model for a single instance is provided in Fig. 17. The considered instance has been classified as *yes* with 98% probability. On the right side of Fig. 17, a detailed explanation of relative features' importance and their contribution to final classification is provided. The explanations can be read as: A switch with *timber* bearer, *no* roller fitted, having a *age between 18 and 28*, with *ballast* track type, and with *shallow* stretcher type is noted to have *installation error*; therefore must be maintained. However, the *bi-directional* switch having *good as new condition* state with *flatbottom* rail type contributes to *no* maintenance needed. In contrast to standard performance measures to validate models, the LIME explanation has provided the feature level contribution and explicitly shows how each of them is relevant to final prediction probabilities.

Figs. 18 and 19 provides the explanation of maintenance type prediction and maintenance trigger's status prediction models, respectively. Both of these models deal with multi-classification problem where a type of maintenance treatment or a maintenance status trigger is predicted. By default, LIME explains the multi-classification models in a binary manner as can be noted with *M02* and *Not M02* in Fig. 18 and as *Escalated* and *Not Escalated* in Fig. 19. The LIME explanation of maintenance treatment type prediction model can be described as a *defective sleeper* particularly a *narrow T- piece* on *sleepers* is detected by *patrolling inspection*. Considering

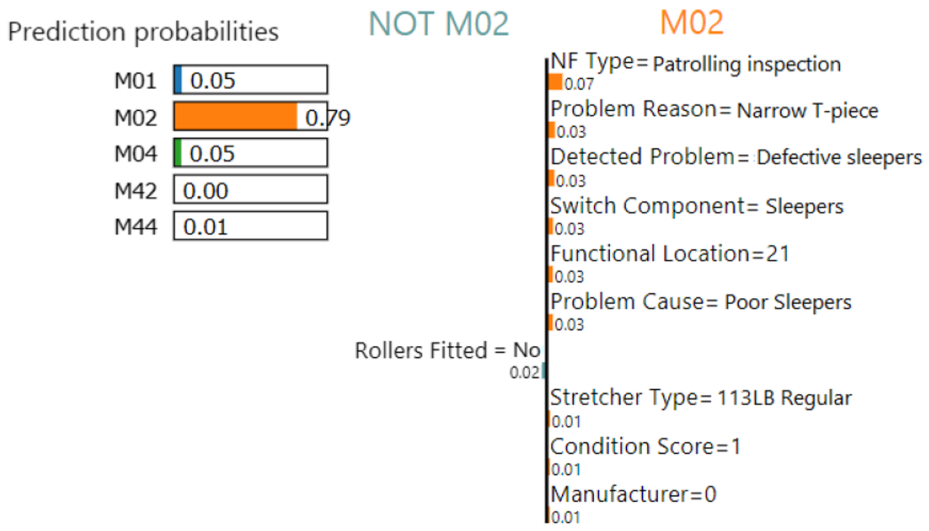


**Fig. 18.** LIME explanation of maintenance treatment type prediction model: M01 = Hand/Cobra pack, M02 = Defective material replace, M04 = Oil/Grease/Shim, M42 = Tamping, M44 = Ballasting/Dumper, M47 = Switch maintenance.

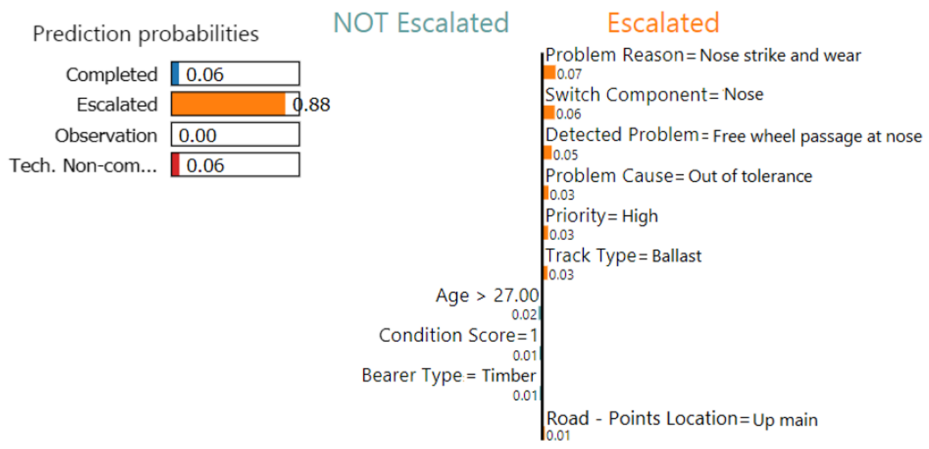


**Fig. 19.** LIME explanation of maintenance triggers prediction model: CMP = Completed, ESC = Escalated, OBS = Observation, TNC = Technical non-compliance.

these features values, the models with 79% certainty predicts that *M02: defective material replacement* should be performed on the switch sleepers.

Similarly, for the maintenance trigger's status prediction model the LIME explanation is outlined in Fig. 19. The explanation shows that a switch is in *very good condition state*, however, a *free wheel passage on a switch nose* is reported due to *nose strike and wear*. This problem reported is termed as *out of tolerance* with *high priority*. Having analysed these properties of all the features, the model with 88% certainty classifies the status of maintenance trigger as escalated to the technical manager for the final decision.

With the ability to better interpret the results of predictive models, domain experts and decision-makers can better interact with and trust the models' prediction. This gives a reliable decision-aid to infrastructure managers to efficiently predict and plan the maintenance activities.

## 7. Discussion

This paper has applied the tree-based classification algorithms of machine learning for the development of maintenance predictive models. Multiple models based on decision tree (DT), random forest (RF) and gradient boosted tree (GBT) were trained and evaluated for the prediction of maintenance need, activity types and trigger's status. Unlike many studies that use additional data collection procedures such as monitoring systems and sensors, our models utilise the data generated from in-use business process of maintenance request from SAP ERP system.

The purpose is to develop such predictive models that can be readily used as an aid in decision tool by infrastructure managers for maintenance planning. Learned from the historical data, the model analyse each maintenance trigger as an individual case and provide its prediction. Moreover, these predictive models bring efficiency particularly for unplanned maintenance planning, as enormous amount of notifications can be labelled with maintenance need, activity type and trigger's status in few seconds. Since the intention is to deploy the predictive models in real decision scenarios, our work especially takes into account the interpretability of the models' outcome. We explained the models' prediction by feature importance analysis as well as by per case details through employing LIME framework.

With the exploratory data analysis, development of maintenance classification models and explanation of their result, few remarks can be noted:

First it is found that the data of notifications generated during 2017 was highly imbalance having only three distinct maintenance activity classes and one new trigger's status label. The data imbalance is the reason that the models show relatively poor predictive power on held-out validation.

Second for the completeness of models' evaluation, stratified cross validation is performed where several data folds (data subsets) are used as training and test set iteratively. The models show overall good predictive performance with stratified cross validation.

Third with feature importance analysis it is noted that functional location, age and detected problems are most important features. This can be very well explained with agency decision-making practices, where decision are mostly based on age and importance of functional location and much less attention is given to other factors namely, condition scores.

Fourth the instance level explanation using LIME shows the varying level of feature contributions for final prediction probabilities compared with feature importance analysis. This is because some feature may have a considerable impact on models' predictions for a single case, which would be visible with instance level graphs. However, the same feature may have been fragmented based on data values deep in a tree and thus attain lower feature importance score globally.

The solution approach of predictive models' development and their results explanation have wider applicability and are generalizable to other assets types and maintenance planning scenarios. This is because we have utilised the data from a business process of plant maintenance module of SAP ERP, which is currently being used in multiple railway agencies. Moreover, the maintenance request process, mentioned in this study, is a standard procedure of triggering unplanned maintenance irrespective of a specific asset type. Therefore, the details of datasets mentioned in Section 2 and structure of maintenance classification problem given in Section 4 can be utilised as instructions to develop predictive models for other type of assets. The sensitivity analysis of models' performance attributed to hyper-parameters tuning is deliberately skipped in this study. This is to illustrate how simple models, providing more than 70% of accuracy and trained on ERP structured data, can prove useful for maintenance prediction tasks. There are many studies in machine learning research that analyses the models performance with respect to hyper-parameters across the different datasets. For an extensive overview, an interested reader may refer to Rijn and Hutter (2018).

From the practical perspective, it is worth noting that the real datasets evolve over time and may have new class labels in the future. To accommodate evolving nature of data while aiming for sound predictions, the models must be retrained periodically.

## 8. Conclusions

This paper employs the machine learning techniques for the development of predictive maintenance models. Multiple models based on decision tree (DT), random forest (RF) and gradient boosted tree (GBT) were trained and evaluated (i) to predict if a maintenance will be performed or delayed as a result of trigger, (ii) to predict the maintenance activity type and (iii) to predict the status of a maintenance trigger (notification). For the prediction of maintenance need, the GBT model performed most optimally as compared to other methods with 86% accuracy. For maintenance activity type and trigger's status prediction the RF model attains an accuracy of 70% and 79% on the held-out test set respectively. By collecting more data, specifically for minority classes, the predictive performance of the models can even be further improved. We show how the existing data readily available at railway agencies can be used for the development of ML models without the need of additional monitoring devices. Since the data is generated from in-use business processes, the models can be directly incorporated for maintenance decision-making. Moreover, we have introduced an approach to provide a detailed explanation of the outputs of ML models by illustrating the features' importance. The feature importance analysis shows which input features positively or negatively impact the model's output. In addition to global explanations, we have employed the local interpretable model-agnostic explanations framework to explain the model behaviour locally for a specific instance. This interpretability of the models also provides a understanding of the decision-making process, and identify the most important data attributes which may help in future data collection procedures.

The future work of this study seeks to develop regression models in order to estimate the best time to perform the maintenance. This will require the data of historical maintenance actions and operational details of the network. Such regression model can be used by a railway agency for the budget planning as well as for the possession planning of the network. Another extension of this work is to implement multi-task learning using deep neural networks in order to develop a unified model for the prediction of related maintenance tasks.

## Acknowledgement

This study has been performed under funding from the European Union's Horizon 2020 research and innovation program with grant agreement No 636285 DESTination Rail.

## References

Akosa, J., 2017. Predictive accuracy: a misleading performance measure for highly imbalanced data. In: Proceedings of the SAS Global Forum.
Al-Douri, Y.K., Tretten, P., Karim, R., 2016. Improvement of railway perfor-mance: a study of swedish railway infrastructure. J. Modern Transport. 24 (1), 22–37. https://doi.org/10.1007/s40534-015-0092-0.
Andreou, A., Venkatadri, G., Goga, O., Gummadi, K.P., Loiseau, P., Mislove, A., 2018. Investigating ad transparency mechanisms in social media: a case study of facebooks explanations. In: The Network and Distributed System Security Symposium (NDSS).
Batista, G.E., Prati, R.C., Monard, M.C., 2004. A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explor. Newsl. 6 (1), 20–29.
Biau, G., 2012. Analysis of a random forests model. J. Mach. Learn. Res. 13 (Apr), 1063–1095.
Bohm, T., 2017. Remaining useful life prediction for railway switch engines using clas-sification techniques. Int. J. Prognos. Health Manage. 59.
Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32. https://doi.org/10.1023/A:1010933404324.
Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classification and Regression Trees, Wadsworth International Group, Belmont, California. Google Scholar.
Cabitza, F., Banfi, G., 2018. Machine learning in laboratory medicine: waiting for the flood? Clin. Chem. Lab. Med. 56 (4), 516–524.
Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.
Chen, J., Liu, Z., Wang, H., Nunez, A., Han, Z., 2018. Automatic defect detection of fasteners on the catenary support device using deep convolutional neural network. IEEE Trans. Instrum. Meas. 67 (2), 257–269.
Chen, T., Guestrin, C., 2016. Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794.
Cohen, J., 1960. A coefficient of agreement for nominal scales. Educ. Psycho-log. Meas. 20 (1), 37–46.
Datta, A., Sen, S., Zick, Y., 2016. Algorithmic transparency via quantitative input influence: theory and experiments with learning systems. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 598–617.
de Bruin, T., Verbert, K., Babuska, R., 2017. Railway track circuit fault diagnosis using recurrent neural networks. IEEE Trans. Neural Networks Learn. Syst. 28 (3), 523–533.
Doshi-Velez, F., Kim, B., 2017. Towards a Rigorous Science of Interpretable Machine Learning. arXiv. < https://arxiv.org/abs/1702.08608 > .
ERF, 2013. Road Asset Management: An ERF Position Paper for Maintaining and Improving a Sustainable and Efficient Road Network (Tech. Rep.). Brussels, Belgium: European Union Road Foundation.

European Railway Agency, 2014. Railway Safety Performance in the European Union (Tech. Rep.). Brussels, Belgium: European Union Railway.
EuroStat, 2016. Railway Passenger Transport Statistics - Quarterly and Annual Data (Tech. Rep.). Brussels, Belgium: European Union Railway.
Fernandez-Delgado, M., Cernadas, E., Barro, S., Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems? J. Mach. Learn. Res. 15 (1), 3133–3181.
Flach, P., 2012. Machine Learning: The Art and Science of Algorithms that Make Sense of Data. Cambridge University Press.
Friedman, J., Hastie, T., Tibshirani, R., 2001. The Elements of Statistical Learning, vol. 1 Springer.
Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Ann. Stat. 1189–1232.
Guo, C., Berkhahn, F., 2016. Entity Embeddings of Categorical Variables. Available from: arXiv preprint arXiv: 1604.06737.
Hastie, T., Tibshirani, R., Friedman, J., 2009. Unsupervised learning. In: The Elements of Statistical Learning. Springer, pp. 485–585.
He, H., Garcia, E.A., 2008. Learning from imbalanced data. IEEE Trans. Knowl. Data Eng. (9), 1263–1284.
Kauschke, S., Fiirnkranz, J., Janssen, F., 2016. Predicting cargo train failures: a machine learning approach for a lightweight prototype. In: International Conference on Discovery Science, pp. 151–166.
Kotsiantis, S.B., Zaharakis, I., Pintelas, P., 2007. Supervised machine learning: a review of classification techniques. Emerg. Artif. Intell. Appl. Comput. Eng. 160, 3–24.
Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., Hampapur, A., 2014. Improving rail network velocity: a machine learning approach to predictive maintenance. Transport. Res. Part C: Emerg. Technol. 45, 17–26.
Lipton, Z.C., 2016. The Mythos of Model Interpretability. CoRR, abs/1606.03490. < http://arxiv.org/abs/1606.03490 > .
Manco, G., Ritacco, E., Rullo, P., Gallucci, L., Astill, W., Kimber, D., Antonelli, M., 2017. Fault detection and explanation through big data analysis on sensor streams. Exp. Syst. Appl. 87, 141–156.
Marquez, A.C., 2007. The Maintenance Management Framework: Models and Methods for Complex Systems Maintenance. Springer Science & Business Media.
Natekin, A., Knoll, A., 2013. Gradient boosting machines, a tutorial. Front. Neurorobot. 7, 21.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Duch-esnay, E., 2011. Scikit-learn: machine learning in python. J. Mach. Learn. Res. 12, 2825–2830.
Quinlan, J.R., 1986. Induction of decision trees. Mach. Learn. 1 (1), 81–106.
Quinlan, J.R., 1993. C4. 5: Programming for Machine Learning, vol. 38 Morgan Kauffmann p. 48.
Ramos, J., et al., 2003. Using tf-idf to determine word relevance in document queries. In: Proceedings of the First Instructional Conference on Machine Learning, vol. 242, pp. 133–142.
Raschka, S., Mirjalili, V., 2017. Python Machine Learning. Packt Publishing Ltd.
Ribeiro, M.T., Singh, S., Guestrin, C., 2016. Why should i trust you?: explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135–1144.
Sokolova, M., Lapalme, G., 2009. A systematic analysis of performance measures for classification tasks. Inform. Process. Manage. 45 (4), 427–437.
Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., Zeileis, A., 2008. Conditional variable importance for random forests. BMC Bioinform. 9 (1), 307.
van Rijn, J.N., Hutter, F., 2018. Hyperparameter importance across datasets. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2367–2376.
Witten, I.H., Frank, E., Hall, M.A., Pal, C.J., 2016. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufman.