Delay Prediction System for Large-Scale Railway Networks based on Big Data Analytics

L. Oneto, E. Fumeo, G. Clerico, R. Canepa, F. Papa,

C. Dambra, N. Mazzino, and D. Anguita

Delay Prediction System for Large-Scale Railway Networks based on Big Data Analytics

Luca Oneto[1]**, Emanuele Fumeo[1], Giorgio Clerico[1], Renzo Canepa[2],
Federico Papa[3], Carlo Dambra[3], Nadia Mazzino[3], and Davide Anguita[1]

DIBRIS - University of Genoa, Via Opera Pia 11A, Genova, I-16145, Italy
{luca.oneto,emanuele.fumeo,g.clerico,davide.anguita}@unige.it
Rete Ferroviaria Italiana S.p.A., Via Don Vincenzo Minetti 6/5, 16126 Genoa, Italy
r.canepa@rfi.it
Ansaldo STS S.p.A., Via Paolo Mantovani 3-5, 16151 Genoa, Italy
{federico.papa,carlo.dambra,nadia.mazzino}@ansaldo-sts.com

Abstract. State-of-the-art train delay prediction systems do not exploit historical train movements data collected by the railway information systems, but they rely on static rules built by expert of the railway infrastructure based on classical univariate statistic. The purpose of this paper is to build a data-driven train delay prediction system for large-scale railway networks which exploits the most recent Big Data technologies and learning algorithms. In particular, we propose a fast learning algorithm for predicting train delays based on the Extreme Learning Machine that fully exploits the recent in-memory large-scale data processing technologies. Our system is able to rapidly extract nontrivial information from the large amount of data available in order to make accurate predictions about different future states of the railway network. Results on real world data coming from the Italian railway network show that our proposal is able to improve the current state-of-the-art train delay prediction systems.

Keywords: Condition–Based Maintenance, Naval Propulsion Plant, Machine Learning, Publicly Distributed Dataset

1 Introduction

Big Data Analytics is one of the current trending research interests in the context of railway transportation systems. Indeed, many aspects of the railway world can greatly benefit from new technologies and methodologies able to collect, store, process, analyze and visualize large amounts of data [34, 37], e.g. condition based maintenance of railway assets [9, 25], alarm detection with wireless sensor networks [20], passenger information systems [23], risk analysis [8], and the like. In particular, this paper focuses on predicting train delays in order to improve traffic management and dispatching using Big Data Analytics, scaling to large railway networks.

** This research has been supported by the European Union through the projects Capacity4Rail (European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement 605650) and In2Rail (European Union's Horizon 2020 research and innovation programme under grant agreement 635900).

Although trains should respect a fixed schedule called “nominal timetable”, delays occur daily because of accidents, repair works, extreme weather conditions, etc., and affect negatively railway operations, causing service disruptions and losses in the worst cases. Rail Traffic Management Systems (TMSs) have been developed to support managing complex rail services and networks and to increase operations efficiency by allowing train dispatching through remote control of signaling systems. By providing accurate train delay predictions to TMSs, it can be possible to greatly improve traffic management and dispatching in terms of passenger information systems and perceived reliability [24], freight tracking systems for improved customers decision-making, timetable planning [5] by detecting recurrent delays, and delay management (rescheduling) [6].

Due to its key role, TMS stores the information about every “train movement”, i.e. every train arrival and departure timestamp and delays at “checkpoints” monitored by signaling systems (e.g. a station, a switch, etc.). Datasets composed of train movements records have been used as fundamental data sources for every work addressing the problem of train delays predictions. For instance, Milinkovic et al. [22] developed a Fuzzy Petri Net (FPN) model to estimate train delays based both on expert knowledge and on historical train movements data. Berger et al. [2] presented a stochastic model for delay propagation and forecasts based on directed acyclic graphs. Goverde, Keckman et al. [11, 12, 17, 18] developed an intensive research in the context of delay prediction and propagation by using process mining techniques based on innovative timed event graphs, on historical train movements data, and on expert knowledge about railway infrastructure. However, their models are based on classical univariate statistics, while our solution integrates multivariate statistical concepts that allow our models to be extended in the future by including other kind of data (e.g. weather forecasts, passenger flows, etc.). Moreover, these models are not especially developed for Big Data technologies, possibly limitating their adoption for large scale networks. Last but not least, S. Pongnumkul et al. [29] worked on data-driven models for train delays predictions, treating the problem as a time series forecast problem. The described system investigates the application of ARIMA and k-NN models over limited train data, making it unsuitable for Big Data.

For these reasons, this paper investigates the problem of predicting train delays for large scale railway networks by treating it as a time series forecast problem where every train movement represents an event in time, and by exploiting Big Data Analytics methodologies. Delay profiles for each train are used to build a set of data-driven models that, working together, make possible to perform a regression analysis on the past delay profiles and consequently to predict the future ones. The data-driven models exploit a well-know Machine Learning algorithm, i.e. the Extreme Learning Machines (ELMs), which has been adapted to exploit typical Big Data parallel architectures. Moreover, the data have been analyzed by using state-of-art Big Data technologies, i.e. Apache Spark on Apache Hadoop, so that it can be used for large scale railway networks. The described approach and the prediction system performance have been validated based on the real historical data provided by Rete Ferroviaria Italiana (RFI), the Italian Infrastructure Manager (IM) that controls all the traffic of the Italian railway network. For this purpose, a set of novel Key Performance Indicators (KPIs) agreed with RFI has been designed and used. Several months of records from the entire Italian railway network have been exploited to show that the new proposed methodology outperforms the current technique used by RFI to predict train delays in terms of overall accuracy.

2 Train Delay Prediction Problem: the Italian Case

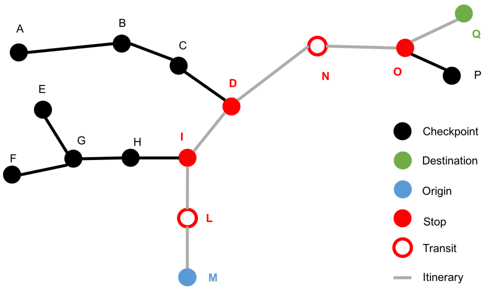

Fig. 1: A railway network depicted as a graph, including a train itinerary from checkpoint M to Q

A railway network can be considered as a graph where nodes represent a series of checkpoints connected one to each other. Any train that runs over the network follows an itinerary defined by a series of n_c checkpoints $\mathscr{C} = \{C_1, C_2, \cdots, C_{n_c}\}$, which is characterized by a station of origin, a station of destination, some stops and some transits (see Figure 1). For any checkpoint C, the train should arrive at time t_A^C e and should depart at time t_D^C, defined in the nominal timetable (with a precision of 30 seconds or 1 minute). The actual arrival and departure times of the train are defined as $\hat{t}_A^C$ and $\hat{t}_D^C$. The differences $(\hat{t}_A^C - t_A^C)$ and $(\hat{t}_D^C - t_D^C)$ are defined as arrival and departure delays respectively, and a train is considered a "delayed train" if its delay is greater than 30 seconds or 1 minute. A dwell time is defined as the difference between the departure time and the arrival time for a fixed checkpoint $(\hat{t}_D^C - \hat{t}_A^C)$, while a running time is defined as the amount of time needed to depart from the first of two subsequent checkpoints and to arrive to the second one $(\hat{t}_A^{C+1} - \hat{t}_D^C)$.

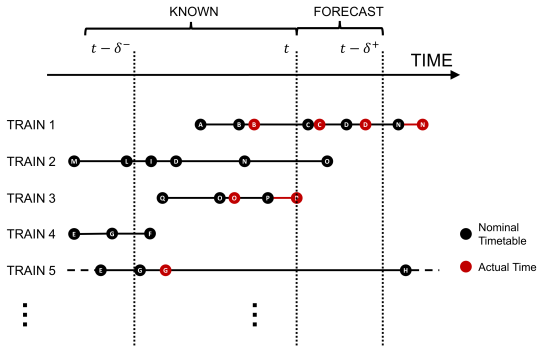

Fig. 2: Data for the train delay forecasting models for the network of Figure 1

In order to tackle the problem of train delays predictions, we propose the following solution. Taking into account the itinerary of a train, for each checkpoint C_i where $i \in \{0, 1, \cdots, n_c\}$, we want to be able to predict the train delays for each subsequent checkpoint C_j with $j \in \{i+1, \cdots, n_c\}$. Note that C_0 represents the train before its departure from the origin station. In this solution, the train delays predictions problem is treated as a time series forecast problem, where a set of predictive models perform a regression analysis over the delay profiles for each train, for each checkpoint C_i of the itineraries of these trains, and for each subsequent checkpoint C_j with $j \in \{i+1, \cdots, n_c\}$. The models are built by exploiting a slightly modified version of the popular ELM algorithm. The historical records about arrivals and departures for each train at each checkpoint, as well as additional derived information such as dwell times and running times, are used as the input of the algorithm. For example, Figure 2 shows the data needed to build forecasting models based on the railway network depicted in Figure 1. Basically, based on the state of the railway network between time $(t-\delta^{-})$ and time t, we want to predict the network state from time t to $(t+\delta^{+})$ and this is nothing but a classical regression problem [28]. To sum up, for each train characterized by a specific itinerary of n_c checkpoints, we need to build n_c models for C_0, (n_c-1) for C_1, and so on. Consequently, the total number of models to be built for each train can be calculated as $n_c+(n_c-1)+\cdots+1={}^{n_c(n_c-1)}/_2$. These models work together in order to make possible to estimate the delays of a particular train during its entire itinerary.

Considering the case of the Italian railway network, RFI controls every day approximately 10 thousand trains traveling along the national railway network. Every train is characterized by an itinerary composed of approximately 12 checkpoints, which means that the number of train movements is greater than or equal to 120 thousands per day. This results in roughly one message per second and more than 10 GB of messages per day to be stored. Note that every time that we retrieve a complete set of messages describing the entire planned itinerary of a particular train for one day, the predictive models associated with that train must be retrained. Since for each train we need to build at least ${}^{n_c(n_c-1)}/_2 \approx 60$ models, the number of models that has to be retrained every day in the Italian case is greater than or equal to 600 thousands.

3 Delay Prediction System for Large-Scale Railway Networks

Let us consider a regression problem [35] where $\mathscr{X} \in \mathbb{R}^d$ is the input space and $\mathscr{Y} \in \mathbb{R}$ the output one. Moreover, let us consider a set of examples of the mapping $\mathscr{D}_n : \{z_1,\cdots,z_n\}$ of cardinality n, where $z_{i\in\{1,\cdots,n\}}=(x_i,y_i)$ with $x\in\mathscr{X}$ and $y\in\mathscr{Y}$. A learning algorithm $\mathscr{A}_{\mathscr{H}}$, characterized by a set of hyperparameters $\mathscr{H}$ that must be tuned, maps $\mathscr{D}_n$ into a function $f:\mathscr{A}_{(\mathscr{D}_n,\mathscr{H})}$ from $\mathscr{X}$ to $\mathscr{Y}$. The accuracy of a function $f:\mathscr{A}_{(\mathscr{D}_n,\mathscr{H})}$ in representing the hidden relationship between input and output space is measured with reference to a loss function $\ell(f,z):\mathscr{F}_{\mathscr{H}}\times(\mathscr{X}\times\mathscr{Y})\to\mathbb{R}$. The quantity which we are interested in is the generalization error [1,35], namely the error that a model will perform on new data generated by μ and previously unseen $L(f)=\mathbb{E}_z\ell(f,z)$. Unfortunately, since μ is unknown, $L(f)$ cannot be computed and, consequently, must be estimated. The most common empirical estimator is the empirical error $\widehat{L}(f)=\frac{1}{n}\sum_{z\in\mathscr{D}_n}\ell(f,z)$.

The Extreme Learning Machines (ELM) algorithm is a state-of-the-art tool for regression problems [3, 13, 15] and was introduced to overcome problems posed by backpropagation training algorithm [32]: potentially slow convergence rates, critical tuning of optimization parameters, and presence of local minima that call for multi-start and re-training strategies. ELM was originally developed for the single-hidden-layer feedforward neural networks. The weight of the hidden layer, contrarily to the backpropagation, are randomly assigned while the weights of the output layer are found via Regularized Least Squares (RLS) [14–16]. More formally a vector of weighted links, $w\in\mathbb{R}^h$, connects the hidden neurons to the output neuron without any bias

$$f(x)=\sum_{i=1}^{h} w_i\varphi\left(W_{i,0}+\sum_{j=1}^{d} W_{i,j}x_j\right). \tag{1}$$

where $\varphi : \mathbb{R} \to \mathbb{R}$ is a nonlinear activation function of the hidden neurons and $W \in \mathbb{R}^{h\times(0,\cdots,d)}$, the weight of the hidden units, are set randomly. Finally the weight of the output layer $w \in \mathbb{R}^h$ are found by solving the following RLS problem

$$w^* = \arg\min_{w} \|Aw - y\|^2 + \lambda \|w\|^2 = (A^T A + \lambda I)^{-1} A^T y, \tag{2}$$

where $A \in \mathbb{R}^{n\times h}$, $A_{u,v} = \varphi\left(W_{v,0} + \sum_{j=1}^{d} W_{v,j} x_{u,j}\right)$. Consequently we can see A as a concatenation of the random projection of vectors $x_{\{i\in\{1,\cdots,n\}\}}$ based on the hidden neuron of the ELM: $A = [\phi(x_1), \cdots, \phi(x_n)]^T$.

```
Algorithm 1: SGD for ELM.
Input: $\mathscr{D}_n$, $\lambda$, $\tau$, $n_{\text{iter}}$
Output: $w$
Read $\mathscr{D}_n$ ;
Compute $A$ ;
$w = 0$ ;
for $t \leftarrow 1$ to $n_{iter}$ do
    $w = w - \frac{\tau}{\sqrt{t}} \frac{\partial}{\partial w} \left[\|Aw - y\|^2 + \lambda \|w\|^2\right]$;
return $(w, b)$;
```

Spark [21, 36] is a state-of-the-art framework for high performance in-memory parallel computing. The main idea behind the Spark technology is that we have to reduce access to the disk as much as possible and make as much computation as possible in memory. Moreover, since Spark is designed to efficiently deal with iterative computational procedures that recursively perform operations over the same data, it may not be efficient to compute the solution in the form of Eq. (2). Consequently, instead of solving the inversion problem of Eq. (2), let us adopt a Stochastic Gradient Descent (SGD) algorithm. The SGD algorithm is a very general optimization algorithm, which is efficiently able to solve a problem in the following form: $\min_{f\in\mathscr{F}_{\mathscr{H}}} \widehat{L}(f) + \lambda R(f)$, where $R(f)$ is a regularizer [21]. λ balances the tradeoff between the over- and underfitting tendency of the algorithm. Based on the choice of $R(f)$ and $\widehat{L}(f)$ we can retrieve different algorithms [21]. If we set $R(f) = \|w\|^2$ and $\widehat{L}(f) = {}^1\!/\!_n \sum_{i=1}^{n} [f(x_i) - y_i]^2$ we get the ELM formulation of Eq. (2). The Stochastic Gradient Descent (SGD) algorithm for ELM is reported in Algorithm 1 [33].

```
Algorithm 2: SGD for ELM on Spark ($d \geq h$)
Input: $\mathscr{D}_n$, $\lambda$, $\tau$, $n_{\text{iter}}$
Output: $w$
Read $\mathscr{D}_n$ ;
Compute $A$ /* Compute the projection $\phi$ */
$w = 0$;
for $t \leftarrow 1$ to $n_{iter}$ do
    $g = (A, y)$.map(Gradient())
    /* Compute the gradient for each sample */
    .reduce(Sum())
    /* Sum all the gradients of each sample */
    $w = w - \frac{\tau}{\sqrt{t}} g$ ;
return $w$;
```

In Algorithm 1 τ and n_{iter} are parameters related with the speed of the optimization algorithms. Therefore, usually τ and n_{iter} are set based on the experience of the user. In any case τ and n_{iter} can be seen as other regularization terms as λ since they are connected with the early stopping regularization technique [4, 30].

Algorithm 1 is well-suited for implementation in Spark and many of these tools are already available in MLlib [21]. Basically, the implementation of Algorithm 1 reported in Algorithm 2 is an application of two functions: a map for the computation of the gradient and a reduction function for the sum of each single gradient.

The main problem of Algorithm 2 is the computation and storage of A. If $h \ll d$ it means that $A \in \mathbb{R}^{n \times h}$ will be much smaller than the dataset which belongs to $\mathbb{R}^{n \times d}$. In this case, it is more appropriate to compute it before the SGD algorithms starts the iterative process and keep it in memory (note that the computation of A is fully parallel). In this way all the data $\mathbb{R}^{n \times d}$ projected by ϕ into to matrix $A \in \mathbb{R}^{n \times h}$ can be largely kept in volatile memory (RAM) instead of reading from the disk. If instead $h \gg d$, employing Algorithm 2 we risk that $A \in \mathbb{R}^{n \times h}$ does not fit into the RAM, consequently making too many accesses to the disk. For this reason, we adopt two different strategies:

- if h is approximately the same magnitude or smaller than d we use Algorithm 2 and we compute the matrix A at the beginning;
- if $h \gg d$ we adopt Algorithm 3 where $\phi(x_i)$ is computed online in order to avoid to read the data from the disk.

Quite obviously, the limit is given by the size of the RAM of each node and the number of nodes. Until the algorithm is able to keep most of the data in memory, it is better to use Algorithm 2. Algorithm 3 allows us to partially reduce the effect of having to access the data on the disk by paying the price of computing $\phi(x_i)$ online. In fact, Algorithm 3 does not precompute $A \in \mathbb{R}^{n \times h}$ at the beginning but it keeps in memory the data $\mathscr{D}_n$ and, at every iteration of the SGD algorithm, it computes online both the projection induced by ϕ and the gradient. Consequently, there is no need to store $A \in \mathbb{R}^{n \times h}$.

Algorithm 3: SGD for ELM on Spark ($d \leq h$).

```
Input: $\mathscr{D}_n$, $\lambda$, $\tau$, $n_{\text{iter}}$
Output: $w$
Read $\mathscr{D}_n$ ;
$w = 0$;
for $t \leftarrow 1$ to $n_{iter}$ do
    $g = \mathscr{D}_n$.map($\phi$&Gradient())
    /* Compute both the projection $\phi$ and the
       gradient for each sample                        */
    .reduce(Sum())
    /* Sum all the gradients of each sample            */
    $w = w - \frac{\tau}{\sqrt{t}} g$;
return $w$;
```

In this context, the selection of the optimal λ and h remains a fundamental problem, which is still the target of current research [1]. Resampling methods such as k–Fold Cross Validation (KCV), the Leave–One–Out, and the Non-parametric Bootstrap (BTS) [7, 19] are favored by practitioners because they work well in many situations and allow the application of simple statistical techniques for estimating the quantities of interest. Unfortunately the computational burden required by KCV and BTS is quite high for this reason the Bag of Little Bootstraps (BLB) will be exploited [27, 26].

4 Results on the Italian Case Study

In order to validate our methodology and to assess the performance of the new prediction system, we exploited real data provided by RFI. For the purpose of this work, RFI gave access to almost one year of data related to two main areas in Italy. The data included more than a thousand trains and two hundred checkpoints. Note that, the information has been anonymized for privacy and security concerns.

Our approach to the experiments consisted in (i) building for each train in the dataset the needed set of models based on the ELM algorithm, (ii) applying the models to the current state of the trains, and finally (iii) validating the models in terms of performance based on what really happens in a future instant. Consequently, we performed simulations for all the trains included in the dataset simulating the online-approach that updates predictive models every day, so to take advantage of new information as soon as they becomes available. We compared the results of our simulations with the results of the current train delay prediction system used by RFI, which is quite similar the one described in [12]. In order to fairly assess the performance of the two systems, a set of novel KPIs agreed with RFI has been designed and exploited. Since the purpose of this work was to build predictive models able to forecast the train delays, these KPIs represent different indicators of the quality of these predictive models. Based on these considerations, three different indicators have been used, which are also proposed in Figure 3 in a graphical fashion:

- *Average Accuracy at the i-th following Checkpoint for train j (AAiCj):* for a particular train j, we average the absolute value of the difference between the predicted delay and its actual delay, at the i-th following Checkpoint with respect to the actual Checkpoint.
- *AAiC:* is the average over the different trains j of AAiCj
- *Average Accuracy at Checkpoint-i for train j (AACij):* for a particular train j, the average of the absolute value of the difference between the predicted delay and its actual delay, at the i-th checkpoint, is computed.
- *AACi:* is the average over the different trains j of AACij
- *Total Average Accuracy for train j (TAAj):* is the average over the different checkpoint i of AACij (or equivalently the average over the index i of AAiCj).
- *TAA:* is the average over the different trains j of TAAj

We ran the experiments by exploiting the Google Compute Engine [10] on the Google Cloud Platform. We employed a four-machines cluster, each one equipped with two cores (machine type n1-highcpu-2), 1.8 GB of RAM and an HDD of 30 GB. We use Spark 1.5.1 running on Hadoop 2.7.1 configured analogously to [31]. The set of possible configurations of hyperparameters is defined as a set $\mathscr{H}$ where $\mathscr{H} = \{(h, \lambda) : h \in \mathscr{G}_h, \lambda \in \mathscr{G}_\lambda\}$ with $\mathscr{G}_h = \{\lfloor 10^{\{1,1.2,\cdots,3.8,4\}} \rfloor\}$ and $\mathscr{G}_\lambda = 10^{\{-6,-5.5,\cdots,3.5,4\}}$. Finally, as suggested by the RFI experts, $t_0 - \delta^-$ is set equal to the time, in the nominal timetable, of the origin of the train.

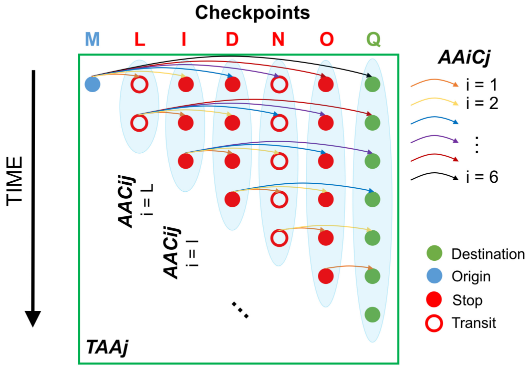

Fig. 3: KPIs for the train and the itinerary depicted in Figure 1

In Table 1 we have reported the KPIs for the proposed prediction system and for the one currently used by RFI. Note that Trains and Checkpoints IDs have been anonymized, and that only part of the results have been included in the paper because of space constraints. Although the RFI system has shown to be robust and accurate during our simulations, our data-driven prediction system managed to outperform it by a factor of ×1.65

for TAA, since it is able to infer a model which takes into account the entire state of the network and not just the local dependencies. The accuracy of the ELM-based method is quite homogeneous with respect to different trains and stations. Moreover, AAiCj grows with *i* as expected (the further is the prediction in the future, the more uncertain is the prediction itself). Finally, note that there are blank spaces in the tables: this means that, for example, for AAiCj some trains have less checkpoints (at least in the dataset that we analysed), or for AACij different trains run over different checkpoints.

As final issue we would like to underline that if just Algorithm 2 is exploited and not the proposal of combining Algorithms 2 and 3 based on *d* and *h* we have that in the first case in order to train all the models of our simulation 15 hours are needed while in the second case approximately one hour is enough. In other words, our optimization strategy is 15 time faster than the naive approach.

Table 1: ELM based and RFI prediction systems KPIs (in minutes).

AAiCj	ELM	RFI	ELM	RFI	ELM	RFI	ELM	RFI	ELM	RFI	
j \ i	1st		2nd		3rd		4th		5th		
1	1.6 ± 0.1	1.8 ± 0.5	1.8 ± 0.2	2.1 ± 0.8	2.1 ± 0.1	2.3 ± 0.3	2.3 ± 0.2	2.5 ± 0.6	2.4 ± 0.1	2.7 ± 0.4	
2	1.8 ± 0.1	3.2 ± 1.7	1.9 ± 0.1	3.4 ± 0.1	2.2 ± 0.1	3.8 ± 3.2	2.4 ± 0.0	4.2 ± 0.6	2.6 ± 0.5	4.6 ± 0.1	
3	1.4 ± 0.1	1.9 ± 0.2	1.6 ± 0.5	2.0 ± 0.2	1.8 ± 0.1	2.3 ± 0.1	1.9 ± 0.4	2.6 ± 1.0	2.0 ± 0.2	2.8 ± 0.4	
4	1.5 ± 0.1	2.0 ± 0.1	1.6 ± 0.0	2.2 ± 1.1	1.9 ± 0.1	2.6 ± 0.8	2.1 ± 0.4	3.0 ± 0.7	2.3 ± 0.4	3.4 ± 0.4	
5	0.9 ± 0.0	1.4 ± 0.2	1.0 ± 0.2	1.7 ± 0.1	1.2 ± 0.2	2.0 ± 0.3	1.4 ± 0.0	2.3 ± 1.1	1.6 ± 0.1	2.6 ± 0.0	
6	1.3 ± 0.1	1.4 ± 0.3	1.5 ± 0.2	1.7 ± 0.3	1.8 ± 0.2	2.0 ± 1.9	2.1 ± 0.0	2.3 ± 0.4	2.3 ± 0.0	2.6 ± 0.9	
7	1.0 ± 0.1	1.3 ± 0.4	1.1 ± 0.1	1.4 ± 0.5	1.3 ± 0.1	1.6 ± 0.5	1.5 ± 0.1	1.8 ± 0.4	1.6 ± 0.1	2.0 ± 0.2	
8	1.0 ± 0.2	1.3 ± 1.1	1.3 ± 0.0	1.6 ± 1.0	1.4 ± 0.1	1.9 ± 0.0	1.6 ± 0.1	2.1 ± 0.3	1.7 ± 0.1	2.3 ± 0.4	
9	0.8 ± 0.0	1.2 ± 0.7	0.9 ± 0.0	1.2 ± 0.7	1.0 ± 0.2	1.4 ± 0.0	1.1 ± 0.0	1.5 ± 1.0	1.2 ± 0.1	1.5 ± 0.3	
10	1.0 ± 0.2	1.5 ± 0.1	1.1 ± 0.1	1.6 ± 0.1	1.3 ± 0.0	2.0 ± 0.3	1.5 ± 0.2	2.3 ± 0.3	1.6 ± 0.0	2.4 ± 0.6	
11	1.2 ± 0.2	1.4 ± 0.1	1.3 ± 0.2	1.5 ± 0.3	1.5 ± 0.1	1.7 ± 0.0	1.6 ± 0.1	1.9 ± 0.2	1.7 ± 0.3	2.1 ± 0.2	
12	1.6 ± 0.0	2.1 ± 0.8	1.9 ± 0.2	2.6 ± 0.8	2.1 ± 0.2	3.1 ± 1.6	2.3 ± 0.3	3.5 ± 0.6	2.6 ± 0.0	3.8 ± 0.1	
13	0.9 ± 0.1	1.2 ± 0.5	1.0 ± 0.2	1.3 ± 0.4	1.1 ± 0.1	1.4 ± 0.5	1.3 ± 0.0	1.6 ± 0.4	1.4 ± 0.1	1.6 ± 0.4	…
14	2.1 ± 0.2	3.1 ± 0.7	2.3 ± 0.3	3.5 ± 0.8	-	-	-	-	-	-	-
15	1.6 ± 0.2	2.2 ± 0.4	1.8 ± 0.1	2.4 ± 0.5	2.0 ± 0.2	2.8 ± 1.1	2.1 ± 0.1	3.1 ± 0.7	2.1 ± 0.0	3.2 ± 1.2	
16	1.8 ± 0.1	2.7 ± 0.3	2.1 ± 0.1	2.9 ± 0.4	2.4 ± 0.0	3.4 ± 0.0	2.6 ± 0.6	3.9 ± 1.6	2.9 ± 0.0	4.1 ± 0.6	
17	1.7 ± 0.1	2.3 ± 1.5	1.9 ± 0.2	2.5 ± 0.8	2.2 ± 0.1	2.9 ± 0.1	2.3 ± 0.3	3.3 ± 1.4	2.4 ± 0.2	3.4 ± 0.4	
18	1.1 ± 0.1	2.8 ± 1.6	1.4 ± 0.0	3.3 ± 0.6	1.6 ± 0.0	4.3 ± 0.7	1.8 ± 0.0	4.7 ± 0.0	2.1 ± 0.2	4.2 ± 1.2	
19	1.7 ± 0.0	2.3 ± 0.6	1.7 ± 0.1	2.4 ± 0.9	1.9 ± 0.0	2.5 ± 1.2	2.0 ± 0.0	2.7 ± 1.8	2.1 ± 0.2	2.7 ± 0.1	
20	2.3 ± 0.5	3.7 ± 1.7	2.5 ± 0.1	4.2 ± 0.8	2.8 ± 0.3	4.7 ± 1.4	3.0 ± 0.2	5.2 ± 0.3	3.3 ± 0.2	5.6 ± 0.1	
21	2.5 ± 0.0	2.5 ± 0.5	2.6 ± 0.1	2.6 ± 0.1	2.8 ± 0.2	2.8 ± 1.3	3.0 ± 0.4	3.0 ± 0.5	3.2 ± 0.2	3.2 ± 0.1	
22	1.9 ± 0.3	3.7 ± 0.5	2.1 ± 0.3	4.0 ± 0.9	2.3 ± 0.0	4.3 ± 1.4	2.5 ± 0.3	4.6 ± 2.1	2.7 ± 0.1	5.0 ± 0.7	
					…						
AAiC	1.6 ± 0.1	3.0 ± 0.8	1.7 ± 0.1	2.9 ± 1.3	2.0 ± 0.4	3.2 ± 1.6	2.2 ± 0.5	3.4 ± 0.0	2.4 ± 0.2	3.4 ± 0.0	

AACij	ELM	RFI	ELM	RFI	ELM	RFI	ELM	RFI	
j \ i	1		2		3		4		
1	2.3 ± 0.4	2.9 ± 0.4	-	-	-	-	-	-	
2	0.1 ± 0.0	0.0 ± 0.0	-	-	-	-	-	-	
3	0.0 ± 0.0	0.2 ± 0.1	-	-	-	-	-	-	
4	1.5 ± 0.0	1.7 ± 0.2	-	-	1.8 ± 0.2	2.3 ± 0.7	-	-	
5	-	-	-	-	1.1 ± 0.0	1.8 ± 0.2	-	-	
6	-	-	-	-	1.5 ± 0.1	1.8 ± 0.8	-	-	
7	-	-	-	-	0.8 ± 0.0	1.1 ± 0.1	-	-	
8	-	-	-	-	1.2 ± 0.0	1.7 ± 0.4	-	-	
9	-	-	-	-	0.7 ± 0.1	0.7 ± 0.1	-	-	
10	-	-	-	-	1.0 ± 0.0	1.3 ± 0.0	-	-	
11	-	-	-	-	-	-	-	-	
12	-	-	1.8 ± 0.2	2.3 ± 0.5	-	-	-	-	
13	-	-	-	-	-	-	-	-	…
14	-	-	-	-	-	-	-	-	
15	-	-	-	-	-	-	-	-	
16	-	-	-	-	-	-	-	-	
17	-	-	-	-	-	-	-	-	
18	-	-	-	-	-	-	-	-	
19	-	-	-	-	-	-	1.3 ± 0.1	1.6 ± 0.1	
20	-	-	-	-	-	-	3.0 ± 0.6	5.1 ± 0.6	
21	-	-	-	-	-	-	2.3 ± 0.0	1.9 ± 1.4	
22	-	-	-	-	-	-	1.7 ± 0.1	2.1 ± 1.0	
					…				
AACi	1.5 ± 0.1	3.3 ± 1.5	1.3 ± 0.1	1.8 ± 0.1	1.1 ± 0.1	1.6 ± 0.1	2.1 ± 0.1	2.9 ± 0.6	

	TAAj	
j	ELM	RFI
1	2.0 ± 0.1	2.2 ± 1.5
2	2.2 ± 0.1	4.3 ± 0.1
3	1.6 ± 0.2	2.3 ± 0.1
4	1.7 ± 0.5	2.4 ± 0.2
5	1.1 ± 0.0	1.7 ± 0.8
6	1.7 ± 0.1	1.9 ± 0.0
7	1.2 ± 0.0	1.5 ± 0.1
8	1.5 ± 0.2	1.9 ± 0.2
9	0.9 ± 0.1	1.4 ± 0.0
10	1.2 ± 0.1	1.8 ± 0.3
11	1.5 ± 0.1	1.8 ± 0.6
12	2.0 ± 0.1	2.8 ± 0.4
13	1.1 ± 0.2	1.4 ± 0.4
14	2.1 ± 0.3	3.1 ± 0.2
15	1.8 ± 0.3	2.6 ± 0.9
16	2.2 ± 0.1	3.1 ± 0.5
17	2.2 ± 0.0	2.7 ± 0.7
18	1.3 ± 0.2	3.3 ± 0.7
19	1.9 ± 0.3	2.5 ± 0.1
20	2.7 ± 0.2	4.3 ± 1.0
21	2.9 ± 0.5	3.0 ± 0.1
22	2.3 ± 0.5	4.8 ± 1.3
	…	
TAA	2.0 ± 0.3	3.3 ± 1.6

Our future works will take into account also exogenous information available from external sources. For example, we will include in the models information about passenger flows by using touristic databases, about weather conditions by using the Italian National Weather Service, about railway assets conditions, or any other source of data which may affect railway dispatching operations.

References

1. Anguita, D., Ghio, A., Oneto, L., Ridella, S.: In-sample and out-of-sample model selection and error estimation for support vector machines. IEEE Transactions on Neural Networks and Learning Systems 23(9), 1390–1406 (2012)
2. Berger, A., Gebhardt, A., Müller-Hannemann, M., Ostrowski, M.: Stochastic delay prediction in large train networks. In: OASIcs-OpenAccess Series in Informatics (2011)
3. Cambria, E., Huang, G.B.: Extreme learning machines. IEEE Intelligent Systems 28(6), 30–59 (2013)
4. Caruana, R., Lawrence, S., Lee, G.: Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In: Neural Information Processing Systems (2001)
5. Cordeau, J.F., Toth, P., Vigo, D.: A survey of optimization models for train routing and scheduling. Transportation science 32(4), 380–404 (1998)
6. Dollevoet, T., Corman, F., D'Ariano, A., Huisman, D.: An iterative optimization framework for delay management and train scheduling. Flexible Services and Manufacturing Journal 26(4), 490–515 (2014)
7. Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall (1993)
8. Figueres-Esteban, M., Hughes, P., Van Gulijk, C.: The role of data visualization in railway big data risk analysis. In: European Safety and Reliability Conference (2015)
9. Fumeo, E., Oneto, L., Anguita, D.: Condition based maintenance in railway transportation systems based on big data streaming analysis. The INNS Big Data conference (2015)
10. Google: Google Compute Engine. `https://cloud.google.com/compute/` (2016), online; accessed 3 May 2016
11. Goverde, R.M.P.: A delay propagation algorithm for large-scale railway traffic networks. Transportation Research Part C: Emerging Technologies 18(3), 269–287 (2010)
12. Hansen, I.A., Goverde, R.M.P., Van Der Meer, D.J.: Online train delay recognition and running time prediction. In: IEEE International Conference on Intelligent Transportation Systems (2010)
13. Huang, G., Huang, G.B., Song, S., You, K.: Trends in extreme learning machines: A review. Neural Networks 61, 32–48 (2015)
14. Huang, G.B., Chen, L., Siew, C.K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Transactions onNeural Networks 17(4), 879–892 (2006)
15. Huang, G.B., Zhou, H., Ding, X., Zhang, R.: Extreme learning machine for regression and multiclass classification. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42(2), 513–529 (2012)
16. Huang, G.B., Zhu, Q.Y., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: IEEE International Joint Conference on Neural Networks (2004)
17. Kecman, P.: Models for predictive railway traffic management (PhD Thesis). TU Delft, Delft University of Technology (2014)
18. Kecman, P., Goverde, R.M.P.: Online data-driven adaptive prediction of train event times. IEEE Transactions on Intelligent Transportation Systems 16(1), 465–474 (2015)

19. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: International Joint Conference on Artificial Intelligence (1995)
20. Li, H., Parikh, D., He, Q., Qian, B., Li, Z., Fang, D., Hampapur, A.: Improving rail network velocity: A machine learning approach to predictive maintenance. Transportation Research Part C: Emerging Technologies 45, 17–26 (2014)
21. Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., Freeman, J., Manish Amde, D.B.T., Owen, S., Xin, D., Xin, R., Franklin, M.J., Zadeh, R., Zaharia, M., Talwalkar, A.: MLlib: Machine Learning in Apache Spark. Journal of Machine Learning Research 17(34), 1–7 (2016)
22. Milinković, S., Marković, M., Vesković, S., Ivić, M., Pavlović, N.: A fuzzy petri net model to estimate train delays. Simulation Modelling Practice and Theory 33, 144–157 (2013)
23. Morris, C., Easton, J., Roberts, C.: Applications of linked data in the rail domain. In: IEEE International Conference on Big Data (2014)
24. Müller-Hannemann, M., Schnee, M.: Efficient timetable information in the presence of delays. In: Robust and Online Large-Scale Optimization (2009)
25. Núñez, A., Hendriks, J., Li, Z., De Schutter, B., Dollevoet, R.: Facilitating maintenance decisions on the dutch railways using big data: The aba case study. In: IEEE International Conference on Big Data (2014)
26. Oneto, L., Orlandi, I., Anguita, D.: Performance assessment and uncertainty quantification of predictive models for smart manufacturing systems. In: IEEE International Conference on Big Data (Big Data) (2015)
27. Oneto, L., Pilarz, B., Ghio, A., D., A.: Model selection for big data: Algorithmic stability and bag of little bootstraps on gpus. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (2015)
28. Packard, N.H., Crutchfield, J.P., Farmer, J.D., Shaw, R.S.: Geometry from a time series. Physical Review Letters 45(9), 712 (1980)
29. Pongnumkul, S., Pechprasarn, T., Kunaseth, N., Chaipah, K.: Improving arrival time prediction of thailand's passenger trains using historical travel times. In: International Joint Conference on Computer Science and Software Engineering (2014)
30. Prechelt, L.: Automatic early stopping using cross validation: quantifying the criteria. Neural Networks 11(4), 761–767 (1998)
31. Reyes-Ortiz, J.L., Oneto, L., Anguita, D.: Big data analytics in the cloud: Spark on hadoop vs mpi/openmp on beowulf. The INNS Big Data conference (2015)
32. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cognitive modeling 5(3), 1 (1988)
33. Shoro, A.G., Soomro, T.R.: Big data analysis: Apache spark perspective. Global Journal of Computer Science and Technology 15(1) (2015)
34. Thaduri, A., Galar, D., Kumar, U.: Railway assets: A potential domain for big data analytics. The INNS Big Data conference (2015)
35. Vapnik, V.N.: An overview of statistical learning theory. IEEE Transactions on Neural Networks 10(5), 988–999 (1999)
36. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: A fault-tolerant abstraction for inmemory cluster computing. In: USENIX Conference on Networked Systems Design and Implementation (2012)
37. Zarembski, A.M.: Some examples of big data in railroad engineering. In: IEEE International Conference on Big Data (2014)