Advanced Analytics for Train Delay Prediction Systems by Including Exogenous Weather Data

Luca Oneto, *Member, IEEE*, Emanuele Fumeo, Giorgio Clerico, Renzo Canepa, Federico Papa, Carlo Dambra, Nadia Mazzino, and Davide Anguita, *Senior Member, IEEE*

***Abstract*— State-of-the-art train delay prediction systems neither exploit historical data about train movements, nor exogenous data about phenomena that can affect railway operations. They rely, instead, on static rules built by experts of the railway infrastructure based on classical univariate statistics. The purpose of this paper is to build a data-driven train delay prediction system that exploits the most recent analytics tools. The train delay prediction problem has been mapped into a multivariate regression problem and the performance of kernel methods, ensemble methods and feed-forward neural networks have been compared. Firstly, it is shown that it is possible to build a reliable and robust data-driven model based only on the historical data about the train movements. Additionally, the model can be further improved by including data coming from exogenous sources, in particular the weather information provided by national weather services. Results on real world data coming from the Italian railway network show that the proposal of this paper is able to remarkably improve the current state-of-the-art train delay prediction systems. Moreover, the performed simulations show that the inclusion of weather data into the model has a significant positive impact on its performance.**

I. Introduction

Current research trends in railway transportation systems have shown an increasing interest in the application of advanced data analytics to sector specific problems, such as condition based maintenance of railway assets [1], [2], automatic visual inspection systems [3], network capacity estimation [4], optimization for energy-efficient railway operations [5], and the like. In particular, this paper focuses on predicting train delays in order to improve traffic management and dispatching using advanced analytics techniques able to integrate heterogeneous data.

Delays can have various causes: disruptions in the operations flow, accidents, malfunctioning or damaged equipment, construction work, repair work, and weather conditions like snow and ice, floods, and landslides, to name just a few. Although trains should respect a fixed schedule called "nominal timetable", delays occur daily and can affect negatively railway operations, causing service disruptions and losses in the worst cases.

Rail Traffic Management Systems (TMSs) [6] have been developed to support managing the inherent complexity of rail services and networks by providing an integrated and holistic view of operational performance, enabling high levels of rail operations efficiency. By providing accurate train delay predictions to TMSs, it is possible to greatly improve traffic management and dispatching in terms of:

- *Passenger information systems*, increasing the perception of the reliability of train passenger services and, in case of service disruptions, providing valid alternatives to passengers looking for the best train connections [7].
- *Freight tracking systems*, estimating goods' time to arrival correctly so to improve customers' decision-making processes.
- *Timetable planning*, providing the possibility of updating the train trip scheduling to cope with recurrent delays [8].
- *Delay management (rescheduling)*, allowing traffic managers to reroute trains so to utilize the railway network in a better way [9].

Due to its key role, the TMS stores the information about every "train movement", i.e. every train arrival and departure timestamp 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 delay prediction. For instance, Milinkovic et al. [10] developed a Fuzzy Petri Net (FPN) model to estimate train delays based both on expert knowledge and on historical data. Berger et al. [11] presented a stochastic model for delay propagation and forecasts based on directed acyclic graphs. S. Pongnumkul et al. [12] worked on data-driven models for train delay predictions, treating the problem as a time series forecast one. Their system was based on ARIMA and k-NN models, although their work reports the application of their models over a limited set of data from a few trains. Last but not least, Goverde, Keckman et al. [13], [14], [15], [16] 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, these models are based on classical univariate statistics, and they only consider train movements data in order to make their predictions. Other factors affecting railway operations (e.g. drivers behaviour, passengers volumes, strikes and holidays, etc.) are indirectly considered (e.g. specific models for weekends), or even not considered, and in some cases they cannot be easily integrated in the models. Instead, using advanced analytics algorithms (like kernel

Luca Oneto, Emanuele Fumeo, Giorgio Clerico, and Davide Anguita are with the DIBRIS, University of Genoa, Via Opera Pia 13, I-16145, Genoa, Italy (email: {luca.oneto,emanuele.fumeo,giorgio.clerico,davide.anguita}@unige.it). Renzo Canepa is with Rete Ferroviaria Italiana S.p.A., Via Don Vincenzo Minetti 6/5, I-16126, Genoa, Italy (email: r.canepa@rfi.it). Federico Papa, Carlo Dambra, and Nadia Mazzino are with Ansaldo STS S.p.A., Via Paolo Mantovani 3-5, I-16151, Genova, Italy (email: {federico.papa, carlo.dambra.ext, nadia.mazzino}@ansaldo-sts.com).

methods, neural networks, ensemble methods, etc.), it is possible to perform a multivariate analysis over data coming from different sources but related to the same phenomena, pursuing the idea that the more information is available for the creation of the model, the better the performance of the model will be.

For these reasons, this paper investigates the problem of predicting train delays by exploiting advanced data analytics techniques based on multivariate statistical concepts that allow data-driven models to include heterogeneous data. The proposed solution considers the problem of train delays as a time series forecast problem, where every train movement represents an event in time. Train movements data identifies for each train a dataset of delay profiles from which it is possible to build a set of data-driven models that, working together, perform a regression analysis on the past delay profiles and consequently predict future ones. Moreover, this solution can be extended by including data about weather conditions related to the itineraries of the considered trains, as an example of the integration of exogenous variables into the forecasting models. Three different algorithms are used to solve the problem, i.e. Extreme Learning Machines (ELM), Kernel Regularized Least Squares (KRLS) and Random Forests (RF), and their performance are compared. Moreover, in order to tune hyperparameters of the aforementioned algorithms, the Nonparametric Bootstrap (BTS) procedure has been used. 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, and on historical data about weather conditions and forecasts, which is publicly available from the Italian weather services. For this purpose, a set of novel Key Performance Indicators (KPIs) agreed with RFI has been designed and used. Several months or train movements records and weather conditions data from the entire Italian railway network have been exploited, showing that the new proposed methodology outperforms the current technique used by RFI to predict train delays in terms of overall accuracy.

II. Train Delay Prediction Problem: the Italian Case

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 composed of n_c checkpoints $\mathcal{C} = \{C_1, C_2, \cdots, C_{n_c}\}$, which is characterized by a station of origin, a station of destination, some stops and some transits at checkpoints in between (see Figure 1). For any checkpoint C, a train should arrive at time t_A^C e and should depart at time t_D^C, defined in the nominal timetable. Usually time references included in the nominal timetable are approximated with a precision of 30 seconds or 1 minute. The actual arrival and departure times of a train are defined as $\hat{t}_A^C$ and $\hat{t}_D^C$. The difference between the time references included in the nominal timetable and the actual times, either of arrival $(\hat{t}_A^C - t_A^C)$ or of departure $(\hat{t}_D^C - t_D^C)$, is defined as delay.

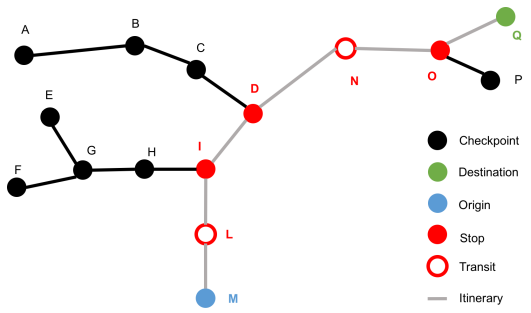

Fig. 1: A railway network depicted as a graph, including a train itinerary from checkpoint M to checkpoint Q

Moreover, if the delay is greater than 30 seconds or 1 minute, then a train is considered as "delayed train". Note that, for the origin station there is no arrival time, while for the destination station there is no departure time. 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 $(\hat{t}_A^{C+1} - \hat{t}_D^C)$.

In order to tackle the problem of train delay predictions, the following solution is proposed. Taking into account the itinerary of a train, the goal is to be able to predict the delays that will affect that specific train for each subsequent checkpoint with respect to the last one in which the train has transited. To make it general, for each checkpoint C_i, where $i \in \{0, 1, \cdots, n_c\}$, the prediction system must be able to predict the train delays for each subsequent checkpoint $\{C_{i+1}, C_{i+2}, \cdots, C_{n_c}\}$. Note that C_0 is a virtual checkpoint that reproduces the condition of a train that still has to depart from its origin. In this solution, the train delay prediction 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 $j \in \{i+1, i+2, \cdots, n_c\}$. 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 network between time $t - \delta^-$ and time t, the proposed system must be able to predict train delays occurring from time t and $t + \delta^+$, and this is nothing but a classical regression problem.

Additionally, this solution takes into account information about weather conditions that could influence the ordinary train operations. For example, weather conditions could influence the passengers flow and consequently the dwell times at each station. Usually, for a particular area, it is possible to access to a big number of weather stations, and for any weather station, it is possible to retrieve the measured values and the forecasted values (for different time horizons) of many variables, such as atmospheric pressure, solar radiation, temperature, humidity, wind and rainfall. Since the granularity of these weather stations is quite fine, it is possible retrieve also the actual and forecasted weather

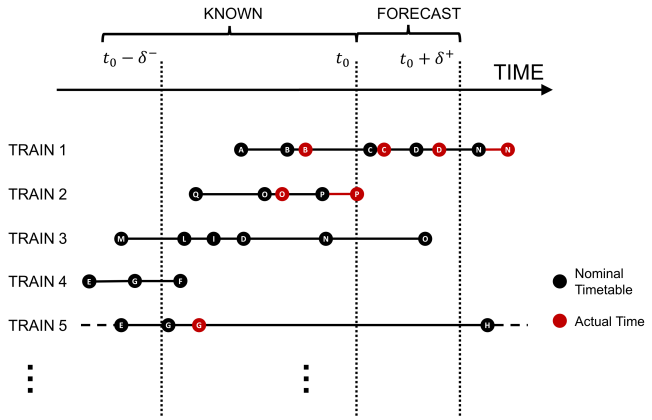

Fig. 2: Data for the train delay forecasting models for the network of Figure 1

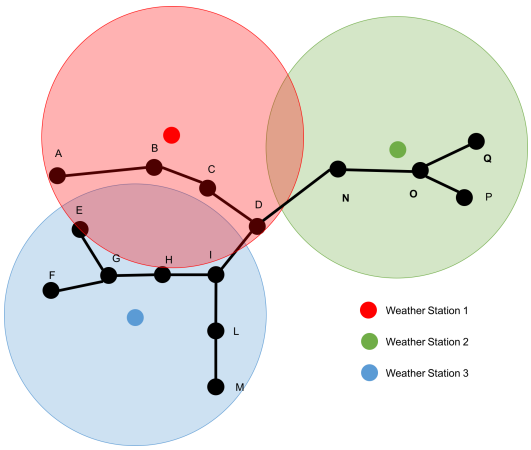

Fig. 3: Weather Informations

conditions for all the checkpoints by looking for the closest one (as depicted in Figure 3). The regression problem can be easily extended to include also historical weather data.

To sum up, for each train characterized by a specific itinerary of n_c checkpoints, n_c models have to be built 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 travelling 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 a complete set of messages describing the entire planned itinerary of a particular train for one day is retrieved, the predictive models associated with that train must be retrained. Since for each train at least ${}^{n_c(n_c-1)}/_2 \approx 60$ models have to be built, the number of models that has to be retrained every day in the Italian case is greater than or equal to 600 thousands. Note that all these training phases can be done during the night when just few trains are flowing through the network. This allows both to have always the best performing models, as required by RFI, which exploit all the data available, and both to deal with the small or big changes in the timetables that occur during the year.

III. Advanced Analytics for Train Delay Prediction Systems

This section deals with the problem of building a data-driven train delay prediction system able to integrate heterogenous data. In particular, focusing on the prediction of the delay of a single train, there are a variable of interest (i.e. the delay profile of a train along its itinerary) and other possible correlated variables (e.g. information about other trains travelling on the network, weather conditions, etc.). The goal is to predict the delay of that train at a particular time in the future $t = t + \delta^+$, i.e. at one of its following checkpoints. For some of the correlated variables (e.g. weather conditions), the forecasted values could be available in addition to historical values, i.e. forecasts at future times made at past times. Given the previous observations, train delay prediction can be attributed into a classical multivariate regression problem [17], [18].

In the conventional regression framework [19], [20] a set of data $\mathcal{D}_n = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, with $\boldsymbol{x}_i \in \mathcal{X} \in \mathbb{R}^d$ and $y_i \in \mathcal{Y} \in \mathbb{R}$, are available from the automation system. The goal of the authors is to identify the unknown model $\mathfrak{S} : \mathcal{X} \to \mathcal{Y}$ through a model $\mathfrak{M} : \mathcal{X} \to \mathcal{Y}$ chosen by an algorithm $\mathscr{A}_{\mathcal{H}}$ defined by its set of hyperparameters $\mathcal{H}$. The accuracy of the model $\mathfrak{M}$ in representing the unknown system $\mathfrak{S}$ can be evaluated with reference to different measures of accuracy [21], [22]. In the case reported by this paper, they have been defined together with RFI experts, and have been reported in Section IV.

In order to map the train delay prediction problem into a multivariate regression model, let's consider the train of interest T_k, which is at checkpoint $C_i^{T_k}$ with $i \in \{0, 1, \cdots, n_c\}$ at time t_0. The goal is to predict the delay at one of its subsequent checkpoints $C_j^{T_k}$, with $j \in \{i+1, i+2, \cdots, n_c\}$. Consequently, the input space $\mathcal{X}$ will be composed by:

- the weather conditions, the delays, the dwell times and the running times for T_k for $t \in [t_0 - \delta^-, t_0]$
- the weather conditions, the delays, the dwell times and the running times for all the other trains T_w with $w \neq k$ which were running over the railway network for $t \in [t_0 - \delta^-, t_0]$
- the values of weather conditions that had been forecasted at past times, for all the subsequent checkpoints of all the trains (including T_k) traveling along the network for $t \in [t_0, t_0 + \delta^+]$

Concerning the output space $\mathcal{Y}$, it is composed by $C_j^{T_k}$ with $j \in \{i+1, i+2, \cdots, n_c\}$ where $t_0 + \delta^+$ is equal to the nominal timetable of T_k for every $C_j^{T_k}$.

Figure 4 shows a graphical representation of the mapping of the the train delay prediction problem into a multivariate regression problem.

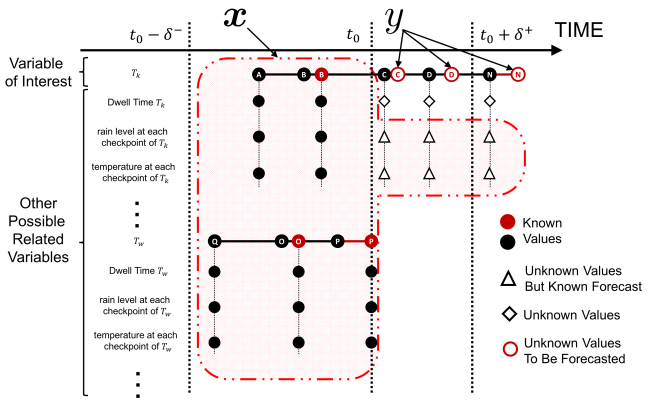

Fig. 4: Mapping of the train delay problem into a multivariate regression problem.

A. *Kernel Methods*

Kernel methods are a family of machine learning algorithms that represent the solution in terms of pairwise similarity between input examples, while they do not work on an explicit representation of the examples [23]. Kernel methods consistently outperformed previous generations of learning techniques because they provide a flexible and expressive learning framework that has been successfully applied to a wide range of real world problems. It is worth to mention that, recently, novel algorithms have increased their competitiveness against them [24], for example Deep Neural Networks [25] and Ensemble Methods [26].

Since the target is a regression problem [19], [27], [28] aiming at building $\mathfrak{M}$, the purpose is to find the best approximating function $h(\boldsymbol{x})$, where $h : \mathbb{R}^d \to \mathbb{R}$, of the system $\mathfrak{S}$. During the training phase, the quality of the regressor $h(\boldsymbol{x})$ is measured according to a loss function $\ell(h(\boldsymbol{x}), y)$ [29], [30], which calculates the discrepancy between the true and the estimated output, respectively y and $\widehat{y} = h(\boldsymbol{x})$. The empirical error then computes the average discrepancy, reported by a model over $\mathcal{D}_n$:

$$\widehat{L}_n(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(\boldsymbol{x}_i), y_i). \tag{1}$$

A simple criterium for selecting the final model during the training phase consists in choosing the approximating function that minimizes the empirical error $\widehat{L}_n(h)$: this approach is known as Empirical Risk Minimization (ERM) [31]. However, ERM is usually avoided in ML as it leads to severely overfitting the model on the training dataset [19], [32], [33], [34]. A more effective approach consists in the minimisation of a cost function where the trade-off between accuracy on the training data and a measure of the complexity of the selected approximating function is implemented [35], [36]:

$$h^* : \quad \min_{h} \quad \widehat{L}_n(h) + \lambda\, \mathcal{C}(h). \tag{2}$$

where $\mathcal{C}(\cdot)$ is a complexity measure which depends on the selected ML approach and λ is a hyperparameter that must be set a priori and regulates the trade-off between the overfitting tendency, related to the minimisation of the empirical error, and the underfitting tendency, related to the minimization of $\mathcal{C}(\cdot)$. This approach is known as Structural Risk Minimization (SRM) [19]. The optimal value for λ is problem-dependent, and tuning this hyperparameter is a non-trivial task [32] and will be faced later in this section.

The Kernelized Regularized Least Squares (KRLS) [37], [38], [39] is the approach here adopted. In KRLS, approximation functions are defined as

$$h(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}), \tag{3}$$

where a non-linear mapping $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^D$, $D \gg d$, is applied so that non-linearity is pursued while still coping with linear models.

For KRLS, Problem (2) is defined as follows. The complexity of the approximation function is measured as

$$\mathcal{C}(h) = \|\boldsymbol{w}\|_2^2 \tag{4}$$

i.e. the Euclidean norm of the set of weights describing the regressor, which is a quite standard complexity measure in ML [35]. Regarding the loss function, the MSE is adopted:

$$\widehat{L}_n(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(h(\boldsymbol{x}_i), y_i) = \frac{1}{n}\sum_{i=1}^{n} \left[h(\boldsymbol{x}_i) - y_i\right]^2. \tag{5}$$

Consequently, Problem (2) can be reformulated as:

$$\boldsymbol{w}^* : \quad \min_{\boldsymbol{w}} \quad \frac{1}{n}\sum_{i=1}^{n} \left[\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) - y_i\right]^2 + \lambda \|\boldsymbol{w}\|_2^2. \tag{6}$$

By exploiting the Representer Theorem [40], the solution h^* of the RLS Problem (6) can be expressed as a linear combination of the samples projected in the space defined by $\boldsymbol{\phi}$:

$$h^*(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x}). \tag{7}$$

It is worth underlining that, according to the kernel trick [41], [27], it is possible to reformulate $h^*(\boldsymbol{x})$ without an explicit knowledge of $\boldsymbol{\phi}$ by using a proper kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x}_i)^T \boldsymbol{\phi}(\boldsymbol{x})$:

$$h^*(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}_i, \boldsymbol{x}). \tag{8}$$

Among the several kernel functions which can be found in literature [41], [27], the Gaussian kernel is often used as it enables learning every possible function [42], [43]:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}, \tag{9}$$

where γ is an hyperparameter which regulates the non-linearity of the solution [43] and must be set a priori, analogously to λ. Small values of γ lead the optimisation to converge to simpler functions $h(x)$ (note that for $\gamma \to 0$ the optimisation converges to a linear regressor), while high values of γ allow higher complexity of $h(x)$.

Finally, the KRLS Problem (6) can be reformulated by exploiting kernels:

$$\boldsymbol{\alpha}^* : \quad \min_{\boldsymbol{\alpha}} \quad \frac{1}{n} \|K\boldsymbol{\alpha} - \boldsymbol{y}\|_2^2 + \lambda \boldsymbol{\alpha}^T K \boldsymbol{\alpha} \tag{10}$$

where $\boldsymbol{y} = [y_1, \cdots, y_n]^T$, $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_n]^T$, K is a matrix such that $K_{i,j} = K_{ji} = K(\boldsymbol{x}_j, \boldsymbol{x}_i)$, and $I \in \mathbb{R}^{n \times n}$ is the Identity matrix.

By setting the derivative with respect to $\boldsymbol{\alpha}$ equal to zero, $\boldsymbol{\alpha}$ can be found by solving the following linear system:

$$(K + n\lambda I)\, \boldsymbol{\alpha}^* = \boldsymbol{y}. \tag{11}$$

Effective solvers have been developed throughout the years, allowing to efficiently solve the problem of Eq. (11) even when very large sets of training data are available [44], [45].

B. Extreme Learning Machine

The ELM approach [46], [47], [48] was introduced to overcome problems posed by back-propagation training algorithm [49], [50]: 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 [51], [52] and then generalized in order to cope with cases where ELM is not neuron alike:

$$h(\boldsymbol{x}) = \sum_{i=1}^{h} w_i g_i(\boldsymbol{x}). \tag{12}$$

where $g_i : \mathbb{R}^d \to \mathbb{R},\ i \in \{1, \cdots, h\}$ is the hidden-layer output corresponding to the input sample $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{w} \in \mathbb{R}^h$ is the output weight vector between the hidden layer and the output layer.

In this case, the input layer has d neurons and connects to the hidden layer (having h neurons) through a set of weights $W \in \mathbb{R}^{h \times (0, \cdots, d)}$ and a nonlinear activation function, $\varphi : \mathbb{R} \to \mathbb{R}$. Thus the i-th neuron response to an input stimulus $\boldsymbol{x}$ is:

$$g_i(\boldsymbol{x}) = \varphi \left(W_{i,0} + \sum_{j=1}^{d} W_{i,j} x_j \right). \tag{13}$$

Note that Eq. (13) can be further generalized to include a wider class of functions [53], [51], [52]; therefore, the response of a neuron to an input stimulus $\boldsymbol{x}$ can be generically represented by any nonlinear piecewise continuous function characterized by a set of parameters. In ELM, the parameters W are set randomly. A vector of weighted links, $\boldsymbol{w} \in \mathbb{R}^h$, connects the hidden neurons to the output neuron without any bias. The overall output function, $f(\boldsymbol{x})$, of the network is:

$$f(\boldsymbol{x}) = \sum_{i=1}^{h} w_i \varphi \left(W_{i,0} + \sum_{j=1}^{d} W_{i,j} x_j \right) = \sum_{i=1}^{h} w_i \varphi_i(\boldsymbol{x}). \tag{14}$$

It is convenient to define an activation matrix, $A \in \mathbb{R}^{n \times h}$, such that the entry $A_{i,j}$ is the activation value of the j-th hidden neuron for the i-th input pattern. The A matrix is:

$$A = \begin{bmatrix} \varphi_1(\boldsymbol{x}_1) & \cdots & \varphi_h(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \varphi_1(\boldsymbol{x}_n) & \cdots & \varphi_h(\boldsymbol{x}_n) \end{bmatrix}. \tag{15}$$

In the ELM model the weights W are set randomly and are not subject to any adjustment, and the quantity $\boldsymbol{w}$ in Eq. (14) is the only degree of freedom. Hence, the training problem reduces to minimization of the convex cost:

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \|A\boldsymbol{w} - \boldsymbol{y}\|^2. \tag{16}$$

A matrix pseudo-inversion yields the unique L_2 solution, as proven in [51], [54]:

$$\boldsymbol{w}^* = A^+ \boldsymbol{y}. \tag{17}$$

The simple, efficient procedure to train an ELM therefore involves the following steps: (I) Randomly generate hidden node parameters (in or case W); (II) Compute the activation matrix A (Eq. (15)); (III) Compute the output weights (Eq. (17)).

Despite the apparent simplicity of the ELM approach, the crucial result is that even random weights in the hidden layer endow a network with notable representation ability. Moreover, the theory derived in [54] proves that regularization strategies can further improve the approach's generalization performance. As a result, the cost function of Eq. (16) is augmented by a regularization factor [54]. A common approach is then to use the L_2 regularizer

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \|A\boldsymbol{w} - \boldsymbol{y}\|^2 + \lambda \|\boldsymbol{w}\|^2, \tag{18}$$

and consequently the vector of weights $\boldsymbol{w}^*$ is then obtained as follows:

$$\boldsymbol{w}^* = (A^T A + \lambda I)^{-1} A^T \boldsymbol{y}, \tag{19}$$

where $I \in \mathbb{R}^{h \times h}$ is an identity matrix. Note that h, the number of hidden neuron, is an hyperparameter that needs to be tuned based on the problem under exam.

C. Ensemble Methods

It is well known that combining the output of several classifiers results in a much better performance than using any one of them alone [26], [55]. In fact, many state-of-the-art algorithms search for a weighted combination of simpler classifiers [56]: Bagging [26], Boosting [57] and Bayesian approaches [58] or even NN [59] and Kernel methods such as SVM [19], [60]. Optimising the generalisation performance of the final model still represents an unsolved problem. How do simple classifiers have to be built? How many simple classifiers have to be combined? How do they have to be combined? Is there any theory which can help in making these choices?

In [26] Breiman tried to give an answer to these questions by proposing the Random Forests (RF) of tree classifiers, one of the state-of-the-art algorithm for classification which has shown to be probably one of the most effective tool in this context [24]. RF combine bagging to random subset feature selection. In bagging, each tree is independently constructed using a bootstrap sample of the dataset [61]. RF add an

additional layer of randomness to bagging. In addition to constructing each tree using a different bootstrap sample of the data, RF change how the classification trees are constructed. In standard trees, each node is split using the best division among all variables. In RF, each node is split using the best among a subset of predictors randomly chosen at that node. Eventually, a simple majority vote for classification tasks or the average response for the regression ones is taken for prediction. In [26] it is shown that the accuracy of the final model depends mainly on three different factors: the number of trees composing the forest, the accuracy of each tree and the correlation between them. The accuracy for RF converges to a limit as the number of trees in the forest increases, while it rises as the accuracy of each tree increases and the correlation between them decreases. RF counterintuitive learning strategy turns out to perform very well compared to many other classifiers, including NN and SVM, and is robust against overfitting [26], [24].

In RF the learning phase of each of the n_t trees composing the forest is quite simple. From $\mathcal{D}_n$, $\lfloor bn \rfloor$ samples are sampled with replacement and $\mathcal{D}'_{\lfloor bn \rfloor}$ is built. A tree is constructed with $\mathcal{D}'_{\lfloor bn \rfloor}$ but the best split is chosen among a subset of n_v predictors over the possible d predictors randomly chosen at each node. The tree is grown until the node contains a maximum of n_l samples. During the classification phase of a previously unseen $\boldsymbol{x}$, each tree classifies $\boldsymbol{x}$ in a class $y_{i\in\{1,\cdots,n_t\}}$, and then the final classification is the $\{p_1,\cdots,p_{n_t}\}$-weighted combination of all the answers of each tree of the RF (note that $\sum_{i=1}^{n_t} p_i = 1$). If $b = 1$, $n_v = \sqrt{n}$, $n_l = 1$, and $p_{i\in\{1,\cdots,n_t\}} = 1/n_t$ the original RF formulation is obtained [26], where n_t is usually chosen to tradeoff accuracy and efficiency [62] or based on the out-of-bag estimate [26] or according to some consistency result [62].

A common misconception about RF is to consider this algorithm as an hyperparameter-free learning algorithm [63], [64]. In fact, there are several hyperparameters which characterize the performance of the final model: the number of trees n_t, the number of samples to extract during the bootstrap procedure b, the depth of each tree n_l, and the number of predictors n_v exploited in each subset during the growth of each tree. Besides b, n_v and n_l, the weights $p_{\{i\in1,\cdots,n_t\}}$ are of paramount importance for the accuracy of an ensemble classifier [56], [55] and for this reason the strategy proposed in [65] and recently further developed in [56], [66] will be used to weight each tree T_i based on its out of bag empirical error $\widehat{L}(T_i)$ [65], [56], [66], [67]:

$$p_i = \frac{e^{-\gamma\widehat{L}(T_i)}}{\sum_{j=1}^{n_t} e^{-\gamma\widehat{L}(T_i)}}, \tag{20}$$

where γ is another hyperparameter to be tuned. A tuning procedure is then needed in order to select the set of hyperparameters [68] which allow to build a RF model characterized by the best generalization performances.

D. Model Selection

Model Selection (MS) deals with the problem of tuning the hyperparameters of each learning algorithm [68]. Several methods exist for MS purpose: resampling methods, like the well-known k-Fold Cross Validation (KCV) [69] or the Nonparametric Bootstrap (BTS) approach [32] approaches, which represents the state-of-the-art model selection approaches when targeting several applications [68]. Resampling methods rely on a simple idea: the original dataset $\mathcal{D}_n$ is resampled once or many (n_r) times, with or without replacement, to build two independent datasets called training, and validation sets, respectively $\mathcal{L}_l^r$ and $\mathcal{V}_v^r$, with $r \in \{1,\cdots,n_r\}$. Note that $\mathcal{L}_l^r \cap \mathcal{V}_v^r = \oslash$, $\mathcal{L}_l^r \cup \mathcal{V}_v^r = \mathcal{D}_n$. Then, in order to select the best set of hyperparameters $\mathcal{H}$ in a set of possible ones $\mathfrak{H} = \{\mathcal{H}_1, \mathcal{H}_2, \cdots\}$ for the algorithm $\mathscr{A}_{\mathcal{H}}$ or, in other words, to perform the MS phase, the following procedure has to be applied:

$$\mathcal{H}^* : \quad \min_{\mathcal{H}\in\mathfrak{H}} \frac{1}{n_r}\sum_{r=1}^{n_r}\frac{1}{v}\sum_{(\boldsymbol{x}_i,y_i)\in\mathcal{V}_v^r} \ell(\mathscr{A}_{\mathcal{H},\mathcal{L}_l^r}, y_i), \tag{21}$$

where $\mathscr{A}_{\mathcal{H},\mathcal{L}_l^r}$ is a model build with the algorithm $\mathscr{A}$ with its set of hyperparameters $\mathcal{H}$ and with the data $\mathcal{L}_l^r$. Since the data in $\mathcal{L}_l^r$ are i.i.d. from the one in $\mathcal{V}_v^r$, the idea is that $\mathcal{H}^*$ should be the set of hyperparameters which allows to achieve a small error on a data set that is independent from the training set.

Note that if $r = 1$, if l, v, and t are aprioristically set such that $n = l + v + t$, and if the the resample procedure is performed without replacement, the hold out method is obtained [32]. For implementing the complete k-fold cross validation, instead, it is needed to set $r \leq \binom{n}{k}\binom{n-\frac{n}{k}}{k}$, $l = (k-2)\frac{n}{k}$, $v = \frac{n}{k}$, and $t = \frac{n}{k}$ and the resampling must be done without replacement [69], [70], [32]. Finally, for implementing the bootstrap, $l = n$ and $\mathcal{L}_l^r$ must be sampled with replacement from $\mathcal{D}_n$, while $\mathcal{V}_v^r$ and $\mathcal{T}_t^r$ are sampled without replacement from the sample of $\mathcal{D}_n$ that have not been sampled in $\mathcal{L}_l^r$ [71], [32]. Note that for the bootstrap procedure $r \leq \binom{2n-1}{n}$. In this paper the BTS is exploited because it represents the state of the art approach [71], [32], [68].

IV. Description of Data and Custom KPIs

In order to validate the proposed methodology and to assess the performance of the new prediction system, a large number of experiments have been performed on the real data provided by RFI. The Italian IM owns records of the train movements from the entire Italian railway network over several years. For the purpose of this work, RFI gave access to more than 6 months of data related to two main areas in Italy, including more than 1000 trains and several checkpoints.

Each record refers to a single train movement, and is composed of the following information: Date, Train ID, Checkpoint ID, Checkpoint Name, Arrival Time, Arrival Delay, Departure Time, Departure Delay and Event Type. The last field, namely "Event Type", refers to the type of event that has been recorded with respect to the train itinerary. For instance, this field can assume different values: Origin (O), Destination (D), Stop (F), Transit (T). The Arrival (Departure) Time field reports the actual time of arrival

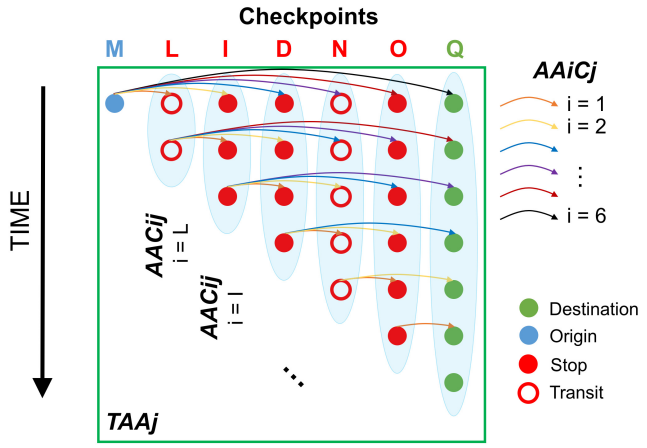

Fig. 5: KPIs for the train and the itinerary of Figure 1

(departure) of a train at a particular checkpoint. Combining this information with the value contained in the Arrival (Departure) Delay field, it is possible to retrieve the scheduled time of arrival (departure). Note that, although IMs usually own proprietary software solutions, this kind of data can be retrieved by any rail TMS, since system of this kind store the same raw information but in different formats. For example, some systems provide the theoretical time and the delay of a train, while others provide the theoretical time and the actual time, making the two information sets exchangeable without loss of information. Finally, note that the information has been anonymized for privacy and security concerns.

Concerning the exogenous variables, the weather conditions data related to the same time period has been retrieved from the regional weather services of the two considered areas, i.e. from [72] and [73]. These data included several actual and forecasted information (i.e. temperature $[^oC]$, relative humidity $[\%]$, wind direction $[^o]$ and intensity $[m/s]$, rain level $[mm]$, pressure $[bar]$ and solar radiation $[W/m^2]$) for every checkpoint included in the train movements dataset.

The approach used to perform the experiments consisted in (i) building for each train in the dataset the needed set of models based on the three algorithms (i.e. KRLS, ELM and RF), (ii) contemporarily tuning the models' hyperparameters through suitable models selection methodologies, (iii) applying the models to the current state of the trains, and finally (iv) validating the models in terms of performance based on what really happens in a future instant. Consequently, simulations have been performed for all the trains included in the dataset adopting an online-approach that updates predictive models every day, so to take advantage of new information as soon as it becomes available.

The results of the simulations have been compared with the results of the current train delay prediction system used by RFI, with and without the additional weather data. The RFI system is quite similar to the one described in [14] from Goverde, although the latter includes process mining refinements which potentially increase its performance.

In order to fairly assess the performance of the proposed prediction system, a set of novel KPIs agreed with RFI has been designed and used. 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. Note that the predictive models should be able to predict, for each train and at each checkpoint of its itinerary, the delay that the train will have in any of the successive checkpoints. Based on this consideration, three different indicators of the quality of predictive models have been used, which are also proposed in Figure 5 in a graphical fashion:

- *Average Accuracy at the i-th following Checkpoint for train j (AAiCj):* for a particular train j, the absolute value of the difference between the predicted delay and its actual delay is averaged, 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-th of AASij (or equivalently the average over the index i of AAiSj).
- *TAA:* is the average over the different trains j of TAAj

V. Results

This section reports the results of the experiments exploiting the approaches described in Section III, benchmarked with the KPIs described in Section IV.

The experiments have been run in two different scenarios:

- NoWI: in this case, only the historical data about train movements provided by RFI have been exploited
- WI: both historical data about train movements and data retrieved from the national weather service have been exploited

The performance of different methods have been compared:

- RFI: the RFI system has been implemented, which is quite similar to the one described in [14]. Note that, the RFI method do not exploit weather informations.
- RF: the Random Forests algorithm has been exploited, where the set of possible configurations of hyperparameters has been defined as $\mathcal{H}_{RF} = \{(b, n_v, n_l, \gamma) : b \in \mathcal{G}_b, n_v \in \mathcal{G}_{n_v}, n_l \in \mathcal{G}_{n_l}, \gamma \in \mathcal{G}_\gamma\}$ with $\mathcal{G}_b = \{0.20, 0.22, \cdots, 1.20\}$, $\mathcal{G}_{n_v} = d^{\{0.00, 0.02, \cdots, 1.00\}}$, $n_l \in n \cdot \{0.00, 0.01, \cdots, 0.50\} + 1$ and $\mathcal{G}_\gamma = 10^{\{-6.0, -5.8, \cdots, 4\}}$ optimized based on the BTS MS procedure with $n_r = 100$. The number of trees is set to $n_t = 500$;
- KM: the kerneled version of RLS with the Gaussian kernel has been exploited, where the set of possible configurations of hyperparameters has been defined as $\mathcal{H}_{KM} = \{(\lambda, \gamma) : \lambda \in \mathcal{G}_\lambda, \gamma \in \mathcal{G}_\gamma\}$ with $\mathcal{G}_\lambda = 10^{\{-6.0, -5.8, \cdots, 4\}}$ and $\mathcal{G}_\gamma = 10^{\{-6.0, -5.8, \cdots, 4\}}$ optimized based on the BTS MS procedure with $n_r = 100$.

- ELM: the Extreme Learning Machine has been exploited, where the set of possible configurations of hyperparameters has been defined as $\mathcal{H}_{ELM} = \{(h, \lambda) : h \in \mathcal{G}_h, \lambda \in \mathcal{G}_\lambda\}$ with $\mathcal{G}_h = \left\{\left\lfloor 10^{\{1,1.2,\cdots,3.8,4\}}\right\rfloor\right\}$ and $\mathcal{G}_\lambda = 10^{\{-6.0,-5.8,\cdots,4\}}$ optimized based on the BTS MS procedure with $n_r = 100$.

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.

In Tables I, II and III the KPIs of the different methods in the two different scenarios have been reported. In particular:

- Table I reports the AAiCj and AAiC. From Table I it is possible to observe that RF method in the WI scenario is the best performing method which improves up to $\times 2$ the current RFI system. When the RF method in the NoWI scenario is exploited, it improves the RFI system by a large amount. Instead, the usage of weather data, with respect to not using it, improves the accuracy of approximately 10%. Also ELM and KM (both in the WI and NoWI scenarios) improve over the RFI system by a large amount. Finally, note that the accuracy decreases as j increases since the forecast refers to an event which is further into the future, and note that some trains have less checkpoint than the others (this is the reason of the symbol '-' for train $j = 14$, which only passes through two checkpoints).
- Table II reports the AACij and the AACi. From Table II it is possible to derive the same observations derived from Table I. In this case it is important to underline that not all the trains run over all the checkpoints, and this is the reason why for some combinations of train j and station i there is a symbol '-'.
- Table III reports the TAAj and the TAA. The latter is more concise and underlines better the advantage, from a final performance perspective, of the RF method in the WI scenario with respect to the RFI prediction system.

Finally, note that the Tables are not complete due to space constraints and that the train and station IDs have been anonymized because of privacy issues.

VI. Conclusions

This paper deals with the problem of building a train delay prediction system based on advance analytics techniques able to grasp the knowledge hidden in historical data about train movements and exogenous weather data. In particular, the proposed solution improves the state of the art methodologies which rely, instead, on static rules built by experts of the railway infrastructure and are based on classical univariate statistic. Results on real world train movements data provided by the Italian Infrastructure Manager (Rete Ferroviaria Italiana - RFI) and weather data retrieved from the national weather services show that advanced analytics approaches can perform up to twice better than current state-of-the-art methodologies. In particular, exploiting only historical data about train movement gives robust models with high performance with respect to the actual train delay prediction system of RFI Moreover, these models can be further improved by taking into account also weather informations. Different state of the art analytics tools have been compared, and Random Forest consistently performed better with respect to the other methodologies exploited.

Future works will take into account other exogenous information available from external sources, such as information about passenger flows by using touristic databases, about railway assets conditions, or any other source of data which may affect railway dispatching operations.

Acknowledgments

This research has been supported by the European Union through the projects C4R (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).

References

[1] E. Fumeo, L. Oneto, and D. Anguita, "Condition based maintenance in railway transportation systems based on big data streaming analysis," *The INNS Big Data conference*, 2015.

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

[3] H. Feng, Z. Jiang, F. Xie, P. Yang, J. Shi, and L. Chen, "Automatic fastener classification and defect detection in vision-based railway inspection systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 4, pp. 877–888, 2014.

[4] S. A. Branishtov, Y. A. Vershinin, D. A. Tumchenok, and A. M. Shirvanyan, "Graph methods for estimation of railway capacity," in *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 525–530.

[5] Y. Bai, T. K. Ho, B. Mao, Y. Ding, and S. Chen, "Energy-efficient locomotive operation for chinese mainline railways by fuzzy predictive control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 938–948, 2014.

[6] E. Davey, "Rail traffic management systems (tms)," in *IET Professional Development Course Railway Signalling and Control Systems*, 2012.

[7] M. Müller-Hannemann and M. Schnee, "Efficient timetable information in the presence of delays," in *Robust and Online Large-Scale Optimization*, 2009.

[8] J. F. Cordeau, P. Toth, and D. Vigo, "A survey of optimization models for train routing and scheduling," *Transportation science*, vol. 32, no. 4, pp. 380–404, 1998.

[9] T. Dollevoet, F. Corman, A. D'Ariano, and D. Huisman, "An iterative optimization framework for delay management and train scheduling," *Flexible Services and Manufacturing Journal*, vol. 26, no. 4, pp. 490–515, 2014.

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

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

[12] S. Pongnumkul, T. Pechprasarn, N. Kunaseth, and K. Chaipah, "Improving arrival time prediction of thailand's passenger trains using historical travel times," in *International Joint Conference on Computer Science and Software Engineering*, 2014.

[13] R. M. P. Goverde, "A delay propagation algorithm for large-scale railway traffic networks," *Transportation Research Part C: Emerging Technologies*, vol. 18, no. 3, pp. 269–287, 2010.

[14] I. A. Hansen, R. M. P. Goverde, and D. J. Van Der Meer, "Online train delay recognition and running time prediction," in *IEEE International Conference on Intelligent Transportation Systems*, 2010.

[15] P. Kecman, *Models for predictive railway traffic management (PhD Thesis)*. TU Delft, Delft University of Technology, 2014.

TABLE I: Data Driven Models and RFI prediction systems AAiCj and AAiC (in minutes).

AAiCj	RFI	NoWI			WI			RFI	NoWI			WI			RFI	NoWI			WI			RFI	NoWI			WI		
j \ i		RF	KM	ELM	RF	KM	ELM		RF	KM	ELM	RF	KM	ELM		RF	KM	ELM	RF	KM	ELM		RF	KM	ELM	RF	KM	ELM
		1st							2nd							3rd							4th					
1	1.8	1.6	1.8	1.6	**1.4**	1.7	1.5	2.1	1.8	1.9	1.8	**1.7**	2.0	1.8	2.3	**2.0**	2.3	2.1	**2.0**	2.2	2.2	2.5	2.3	2.5	2.3	**2.2**	2.5	2.3
2	3.2	1.9	2.0	**1.8**	**1.8**	**1.8**	**1.8**	3.4	**1.9**	2.2	**1.9**	**1.9**	2.1	**1.9**	3.8	2.4	2.6	2.2	**2.1**	2.4	2.2	4.2	2.4	2.6	2.4	**2.3**	2.6	2.4
3	1.9	1.5	1.6	**1.4**	**1.4**	1.6	1.5	2.0	1.6	1.7	1.6	**1.5**	1.6	**1.5**	2.3	**1.7**	2.1	1.8	**1.7**	1.9	1.8	2.6	**1.9**	2.1	**1.9**	**1.9**	2.1	**1.9**
4	2.0	**1.5**	1.7	**1.5**	**1.5**	1.6	**1.5**	2.2	1.6	1.8	1.6	**1.5**	1.6	1.7	2.6	1.9	2.0	1.9	**1.7**	2.0	1.9	3.0	2.1	2.2	2.1	**1.9**	2.2	2.2
5	1.4	**0.8**	1.0	0.9	**0.8**	1.0	**0.8**	1.7	1.1	1.1	**1.0**	**1.0**	1.2	**1.0**	2.0	**1.2**	1.3	**1.2**	**1.2**	1.3	**1.2**	2.3	1.5	1.5	1.4	**1.3**	1.6	1.5
6	1.4	**1.2**	1.4	1.3	1.3	1.4	**1.2**	1.7	1.6	1.6	**1.5**	**1.5**	1.6	**1.5**	2.0	1.9	2.0	1.8	**1.7**	1.8	1.9	2.3	2.1	2.2	2.1	**1.9**	2.2	2.0
7	1.3	**1.0**	1.1	**1.0**	**1.0**	1.1	**1.0**	1.4	1.1	1.3	1.1	**1.0**	1.2	1.1	1.6	1.4	1.4	1.3	**1.2**	1.3	1.4	1.8	1.5	1.6	1.5	1.5	1.6	**1.4**
8	1.3	1.1	1.1	**1.0**	**1.0**	**1.0**	**1.0**	1.6	1.3	1.4	1.3	**1.2**	1.3	1.3	1.9	1.4	1.7	1.4	**1.3**	1.4	1.5	2.1	1.7	1.7	1.6	1.6	1.6	**1.5**
9	1.2	**0.8**	**0.8**	**0.8**	**0.8**	0.9	**0.8**	1.2	0.9	1.0	0.9	**0.8**	0.9	0.9	1.4	1.1	1.1	**1.0**	**1.0**	**1.0**	**1.0**	1.5	1.2	1.2	1.1	**1.0**	1.2	1.1
10	1.5	1.1	**1.0**	**1.0**	**1.0**	**1.0**	1.1	1.6	1.1	1.2	1.1	**1.0**	1.1	1.1	2.0	**1.3**	1.4	**1.3**	**1.3**	**1.3**	**1.3**	2.3	1.5	1.5	1.5	**1.4**	1.5	1.5
11	1.4	1.2	1.3	1.2	**1.1**	1.3	1.2	1.5	1.4	1.5	**1.3**	**1.3**	1.4	1.4	1.7	1.6	1.6	1.5	**1.4**	1.6	**1.4**	1.9	1.7	1.8	**1.6**	**1.6**	1.8	**1.6**
12	2.1	**1.5**	1.8	1.6	**1.5**	1.7	**1.5**	2.6	1.9	2.1	1.9	**1.8**	1.9	**1.8**	3.1	2.2	2.3	**2.1**	**2.1**	**2.1**	2.2	3.5	2.4	2.7	2.3	**2.2**	2.4	2.4
13	1.2	**0.9**	1.0	**0.9**	**0.9**	**0.9**	**0.9**	1.3	**1.0**	1.1	**1.0**	**1.0**	**1.0**	**1.0**	1.4	**1.1**	1.2	**1.1**	**1.1**	1.2	1.2	1.6	1.3	1.4	1.3	1.3	1.3	**1.2**
14	3.1	2.2	2.3	2.1	**2.0**	2.3	2.3	2.2	2.3	2.1	**2.0**	2.3	2.3	2.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	2.0	1.4	1.5	**1.3**	**1.3**	1.4	1.4	2.4	1.5	1.7	1.5	**1.4**	1.6	1.5	2.9	**1.7**	1.9	**1.7**	**1.7**	1.8	**1.7**	3.4	**1.8**	2.0	1.9	**1.8**	2.0	2.0
16	1.7	1.2	1.3	**1.1**	1.2	1.2	1.2	2.0	1.3	1.4	1.3	**1.2**	1.3	1.3	2.4	1.6	1.7	1.5	**1.4**	1.6	1.5	2.8	1.7	1.8	**1.6**	1.7	1.7	1.7
17	1.9	1.3	1.5	1.3	**1.2**	1.4	1.4	2.2	**1.4**	1.6	**1.4**	**1.4**	1.5	1.5	2.7	1.7	1.7	**1.6**	**1.6**	1.8	1.7	3.1	**1.8**	2.1	**1.8**	1.9	2.0	**1.8**
18	1.3	0.4	0.4	0.4	**0.3**	0.4	0.4	1.3	**0.4**	0.5	**0.4**	**0.4**	0.5	**0.4**	1.5	**0.5**	0.6	**0.5**	**0.5**	**0.5**	0.6	1.7	0.7	0.7	**0.6**	**0.6**	0.7	0.7
19	1.5	0.7	0.7	0.7	**0.6**	0.7	0.7	1.6	**0.7**	0.8	**0.7**	**0.7**	**0.7**	**0.7**	1.8	**0.8**	0.9	**0.8**	**0.8**	0.9	0.9	1.9	**0.9**	1.0	**0.9**	**0.9**	1.0	**0.9**
20	1.5	**0.3**	0.4	**0.3**	**0.3**	0.4	**0.3**	1.7	**0.4**	0.5	**0.4**	**0.4**	**0.4**	**0.4**	1.8	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	1.8	**0.6**	0.7	**0.6**	**0.6**	**0.6**	**0.6**
21	1.1	0.6	0.6	**0.5**	**0.5**	**0.5**	**0.5**	1.2	**0.6**	**0.6**	**0.6**	**0.6**	0.7	**0.6**	1.2	**0.7**	0.8	**0.7**	**0.7**	0.8	**0.7**	1.2	0.9	0.9	**0.8**	**0.8**	0.9	0.9
22	1.2	**0.4**	**0.4**	**0.4**	**0.4**	**0.4**	**0.4**	1.2	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	1.3	**0.6**	0.7	**0.6**	**0.6**	0.7	**0.6**	1.3	**0.7**	0.9	**0.7**	**0.7**	0.8	**0.7**
23	1.9	**0.7**	0.8	**0.7**	**0.7**	0.8	**0.7**	2.0	0.9	1.0	**0.8**	**0.8**	0.9	**0.8**	2.4	**1.0**	1.1	**1.0**	**1.0**	**1.0**	**1.0**	2.6	1.1	1.2	1.1	1.1	1.1	**1.0**
24	1.0	**0.4**	0.5	**0.4**	**0.4**	0.5	0.5	1.1	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	1.1	**0.6**	0.7	**0.6**	**0.6**	**0.6**	**0.6**	1.1	**0.7**	0.8	**0.7**	**0.7**	**0.7**	**0.7**
25	1.0	0.4	0.4	0.4	**0.3**	0.4	0.4	1.1	**0.4**	**0.4**	**0.4**	**0.4**	**0.4**	**0.4**	1.2	**0.5**	0.6	**0.5**	**0.5**	0.6	0.6	1.1	**0.6**	0.7	**0.6**	**0.6**	0.7	**0.6**
26	1.9	**0.6**	0.7	0.7	**0.6**	0.7	0.7	2.0	0.8	0.8	0.8	**0.7**	0.8	0.8	2.3	**0.9**	1.0	**0.9**	**0.9**	**0.9**	**0.9**	2.6	**1.0**	1.1	**1.0**	**1.0**	**1.0**	**1.0**
27	1.0	0.4	0.4	0.4	**0.3**	0.4	0.4	1.1	**0.4**	0.5	**0.4**	**0.4**	0.5	**0.4**	1.2	**0.5**	0.6	**0.5**	**0.5**	0.6	**0.5**	1.1	**0.6**	0.7	0.7	0.7	0.7	0.7
28	1.0	**0.4**	0.5	**0.4**	**0.4**	**0.4**	**0.4**	1.0	**0.5**	0.6	**0.5**	**0.5**	0.6	0.6	1.2	**0.6**	0.7	**0.6**	**0.6**	0.7	0.7	1.4	0.8	0.8	**0.7**	**0.7**	0.8	**0.7**
29	1.1	**0.3**	0.4	**0.3**	**0.3**	0.4	**0.3**	1.1	**0.4**	**0.4**	**0.4**	**0.4**	0.5	**0.4**	1.2	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	1.1	**0.6**	0.7	**0.6**	**0.6**	0.7	0.7
30	2.0	**0.6**	0.7	**0.6**	**0.6**	0.7	0.7	2.1	**0.7**	0.8	**0.7**	**0.7**	0.8	0.8	2.4	0.9	0.9	0.9	**0.8**	0.9	0.9	2.7	1.0	1.1	**0.9**	**0.9**	1.1	**0.9**
AACi	3.0	**1.6**	1.8	**1.6**	**1.6**	1.7	**1.6**	2.9	1.7	1.9	1.7	**1.6**	1.8	1.7	3.2	2.0	2.2	2.0	**1.9**	2.1	2.0	3.4	2.3	2.5	2.2	**2.1**	2.4	2.3

TABLE II: Data Driven Models and RFI prediction systems AACij and AACi (in minutes).

AACij	RFI	NoWI			WI			RFI	NoWI			WI			RFI	NoWI			WI			RFI	NoWI			WI		
j \ i		RF	KM	ELM	RF	KM	ELM		RF	KM	ELM	RF	KM	ELM		RF	KM	ELM	RF	KM	ELM		RF	KM	ELM	RF	KM	ELM
		1							2							3							4					
1	2.9	2.4	2.5	**2.3**	**2.3**	2.5	**2.3**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2.2	2.2	2.5	2.2	**2.0**	2.4	2.2
2	**0.0**	0.1	0.1	0.1	0.1	0.1	0.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2.5	1.8	1.9	**1.7**	**1.7**	1.8	1.8
3	0.2	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	-	-	-	-	-	-	-	-	-	-	2.2	1.8	1.8	**1.6**	1.7	1.7	**1.6**
4	1.7	1.5	1.7	1.5	**1.4**	1.6	1.5	2.3	1.8	2.1	1.8	**1.7**	1.9	1.8	2.9	1.7	1.9	1.8	**1.6**	1.9	1.8	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	**1.1**	1.2	1.3	**1.1**	**1.1**	**1.1**	**1.1**	1.1	**0.9**	1.0	**0.9**	**0.9**	1.0	**0.9**	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	**1.2**	1.4	1.5	1.4	1.4	1.5	1.5	1.8	1.8	2.1	1.8	**1.7**	1.9	1.8	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	1.8	1.3	1.3	1.3	1.3	1.3	**1.2**	1.7	**1.5**	1.8	**1.5**	**1.5**	1.6	1.6	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	1.5	1.4	1.6	1.4	**1.3**	1.4	1.4	3.0	2.5	2.9	2.5	**2.4**	2.6	2.7	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	1.1	1.0	1.1	1.0	**0.9**	1.1	1.0	1.2	1.1	1.2	1.1	**1.0**	1.2	1.2	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	1.9	**1.2**	1.3	**1.2**	**1.2**	1.3	**1.2**	1.8	1.3	1.6	1.4	**1.2**	1.5	1.4	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
15	1.3	**1.1**	1.3	**1.1**	**1.1**	1.2	1.2	1.8	1.2	1.3	1.1	**1.0**	1.2	1.2	1.2	1.1	1.2	1.1	1.1	1.1	**1.0**	-	-	-	-	-	-	-
16	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3.9	1.0	1.0	1.0	1.0	1.0	**0.9**
17	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5.8	**2.6**	3.0	2.7	**2.6**	2.9	2.8
18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	6.7	4.6	4.9	4.3	**4.2**	4.6	**4.2**
19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3.8	1.0	1.1	1.0	**0.9**	1.0	1.0
20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	3.7	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**
21	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5.9	2.5	2.7	2.4	**2.3**	2.5	2.5
22	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4.9	**2.2**	2.4	**2.2**	2.3	2.5	2.3
23	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	6.5	3.6	4.1	3.6	**3.5**	4.0	**3.5**
24	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5.1	**2.2**	2.6	2.3	**2.2**	2.4	**2.2**
25	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4.6	1.9	2.0	**1.8**	**1.8**	**1.8**	**1.8**
26	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5.6	**2.9**	3.1	**2.9**	**2.9**	3.1	**2.9**
27	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	6.2	3.0	3.1	2.8	**2.5**	2.8	2.7
28	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	5.5	2.9	3.0	2.8	**2.6**	3.0	2.8
29	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4.2	**1.1**	1.2	**1.1**	**1.1**	1.2	**1.1**
30	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	4.7	1.7	2.0	1.8	**1.6**	2.0	1.8
AAiC	3.3	**1.5**	1.7	**1.5**	**1.5**	1.6	**1.5**	3.1	1.5	1.6	1.5	**1.4**	1.5	1.5	3.3	1.4	1.5	1.4	**1.3**	1.5	1.4	4.2	2.3	2.5	**2.2**	**2.2**	2.4	2.3

[16] P. Kecman and R. M. P. Goverde, "Online data-driven adaptive prediction of train event times," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 465–474, 2015.

[17] F. Takens, *Detecting strange attractors in turbulence*. Springer, 1981.

[18] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," *Physical Review Letters*, vol. 45, no. 9, p. 712, 1980.

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

[20] J. Shawe-Taylor and N. Cristianini, *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[21] E. E. Elattar, J. Goulermas, and Q. H. Wu, "Electric load forecasting based on locally weighted support vector regression," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 40, no. 4, pp. 438–447, 2010.

[22] L. Ghelardoni, A. Ghio, and D. Anguita, "Energy load forecasting using empirical mode decomposition and support vector regression," *IEEE Transactions on Smart Grid*, vol. 4, no. 1, pp. 549–556, 2013.

[23] B. Schölkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[24] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?" *JMLR*, vol. 15, no. 1, pp. 3133–3181, 2014.

[25] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[26] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[27] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[28] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and computing*, vol. 14, no. 3, pp. 199–222, 2004.

[29] W. S. Lee, P. L. Bartlett, and R. C. Williamson, "The importance of convexity in learning with squared loss," *IEEE Transactions on Information Theory*, vol. 44, no. 5, pp. 1974–1980, 1998.

[30] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri, "Are loss functions all the same?" *Neural Computation*, vol. 16, no. 5, pp. 1063–1076, 2004.

[31] G. Lugosi and K. Zeger, "Nonparametric estimation via empirical risk minimization," *IEEE Transactions on Information Theory*, vol. 41, no. 3, pp. 677–687, 1995.

[32] D. Anguita, A. Ghio, L. Oneto, and S. Ridella, "In-sample model selection for support vector machines," in *International Joint Conference on Neural Networks*, 2011.

[33] ——, "Maximal discrepancy vs. rademacher complexity for error estimation." in *ESANN*, 2011.

[34] ——, "A learning machine with a bit-based hypothesis space," in

TABLE III: Data Driven Models and RFI prediction systems TAAj and TAA (in minutes).

j	RFI	TAAj					
		NoWI			WI		
		RF	KM	ELM	RF	KM	ELM
1	2.2	1.9	2.2	2.0	**1.8**	2.1	2.0
2	4.3	2.3	2.4	2.2	**2.1**	2.3	2.2
3	2.3	1.6	1.8	1.6	**1.5**	1.7	1.6
4	2.4	1.8	2.0	**1.7**	**1.7**	1.9	**1.7**
5	1.7	1.1	1.3	1.1	**1.0**	1.1	1.1
6	1.9	1.7	2.0	1.7	**1.6**	1.9	1.7
7	1.5	1.3	1.4	**1.2**	**1.2**	1.3	**1.2**
8	1.9	1.6	1.7	1.5	**1.4**	1.6	1.6
9	1.4	**0.9**	1.1	**0.9**	**0.9**	1.0	1.0
10	1.8	**1.1**	1.2	1.2	1.2	1.2	1.2
11	1.8	1.5	1.6	1.5	**1.4**	1.6	1.5
12	2.8	2.1	2.2	2.0	2.0	2.1	**1.9**
13	1.4	1.1	1.2	1.1	**1.0**	1.2	1.1
14	3.1	2.2	2.3	**2.1**	**2.1**	2.4	2.2
15	1.2	**0.9**	1.0	1.0	**0.9**	1.1	1.0
16	3.9	1.0	1.0	1.0	**0.9**	1.1	1.0
17	5.8	2.8	3.0	2.7	**2.5**	2.8	2.6
18	6.7	4.6	4.9	4.3	**4.1**	4.5	4.6
19	3.8	**1.0**	1.1	**1.0**	**1.0**	**1.0**	**1.0**
20	3.7	**0.9**	1.1	1.0	1.0	1.1	1.0
21	5.9	2.5	2.7	2.4	**2.3**	2.4	2.5
22	4.9	2.3	2.6	**2.2**	2.3	2.4	**2.2**
23	6.5	3.6	4.1	3.6	**3.4**	4.0	3.7
24	5.1	2.4	2.5	2.3	2.3	2.5	**2.2**
25	4.6	1.9	2.1	**1.8**	**1.8**	2.0	**1.8**
26	5.6	2.9	3.1	2.9	**2.8**	3.2	**2.8**
27	6.2	2.9	3.1	**2.8**	**2.8**	3.1	**2.8**
28	5.5	2.9	3.0	2.8	**2.6**	2.9	2.9
29	4.2	**1.0**	1.2	1.1	1.1	1.1	**1.0**
30	4.7	1.9	1.9	1.8	**1.7**	1.9	**1.7**
		...					
TAA	3.3	2.0	2.2	2.0	**1.9**	2.1	2.0

European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2013.

[35] A. Tikhonov and V. Y. Arsenin, *Methods for solving ill-posed problems*. Nauka, Moscow, 1979.

[36] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of inverse problems*. Springer, 1996.

[37] L. Györfi, *A distribution-free theory of nonparametric regression*. Springer, 2002.

[38] D. Pollard, "Empirical processes: theory and applications," in *NSF-CBMS regional conference series in probability and statistics*, 1990.

[39] A. Caponnetto and E. De Vito, "Optimal rates for the regularized least-squares algorithm," *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 331–368, 2007.

[40] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *Computational learning theory*, 2001.

[41] B. Scholkopf, "The kernel trick for distances," in *Neural information processing systems*, 2001.

[42] S. S. Keerthi and C.-J. Lin, "Asymptotic behaviors of support vector machines with gaussian kernel," *Neural computation*, vol. 15, no. 7, pp. 1667–1689, 2003.

[43] L. Oneto, A. Ghio, S. Ridella, and D. Anguita, "Support vector machines and strictly positive definite kernel: The regularization hyperparameter is more important than the kernel hyperparameters," in *International Joint Conference on Neural Networks*, 2015.

[44] D. M. Young, *Iterative solution of large linear systems*. DoverPublications. com, 2003.

[45] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. B. Tsai, M. Amde, S. Owen, D. Xin, D. Xin, M. J. Franklin, R. Z. Matei Zaharia, and A. Talwalkar, "Mllib: Machine learning in apache spark," *arXiv preprint arXiv:1505.06807*, 2015.

[46] E. Cambria and G.-B. Huang *et al.*, "Extreme learning machines," *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 30–59, 2013.

[47] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, pp. 32–48, 2015.

[48] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006.

[49] S. Ridella, S. Rovetta, and R. Zunino, "Circular backpropagation networks for classification," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 84–97, 1997.

[50] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[51] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[52] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in *IEEE International Joint Conference on Neural Networks*, 2004.

[53] F. Bisio, P. Gastaldo, R. Zunino, and E. Cambria, "A learning scheme based on similarity functions for affective common-sense reasoning," in *International Joint Conference on Neural Networks*, 2015.

[54] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 42, no. 2, pp. 513–529, 2012.

[55] P. Germain, A. Lacasse, M. Laviolette, A. ahd Marchand, and R. J. F., "Risk bounds for the majority vote: From a pac-bayesian analysis to a learning algorithm," *JMLR*, vol. 16, no. 4, pp. 787–860, 2015.

[56] G. Lever, F. Laviolette, and F. Shawe-Taylor, "Tighter pac-bayes bounds through distribution-dependent priors," *Theoretical Computer Science*, vol. 473, pp. 4–28, 2013.

[57] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The Annals of Statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[58] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian data analysis*. Taylor & Francis, 2014.

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

[60] D. Anguita, A. Ghio, L. Oneto, and S. Ridella, "Selecting the hypothesis space for improving the generalization ability of support vector machines," in *International Joint Conference on Neural Networks*, 2011, pp. 1169–1176.

[61] B. Efron, "Bootstrap methods: Another look at the jackknife," *The Annals of Statistics*, vol. 7, no. 1, pp. 1–26, 1979.

[62] D. Hernández-Lobato, G. Martínez-Muñoz, and A. Suárez, "How large should ensembles of classifiers be?" *Pattern Recognition*, vol. 46, no. 5, pp. 1323–1336, 2013.

[63] G. Biau, "Analysis of a random forests model," *JMLR*, vol. 13, no. 1, pp. 1063–1095, 2012.

[64] S. Bernard, L. Heutte, and S. Adam, "Influence of hyperparameters on random forest accuracy," in *International Workshop on Multiple Classifier Systems*, 2009.

[65] O. Catoni, *Pac-Bayesian Supervised Classification*. Institute of Mathematical Statistics, 2007.

[66] L. Oneto, S. Ridella, and D. Anguita, "Learning theory; pac bayes; generalisation error," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2016.

[67] O. Ilenia, L. Oneto, and D. Anguita, "Random forests model selection," in *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2016.

[68] D. Anguita, A. Ghio, L. Oneto, and S. Ridella, "In-sample and out-of-sample model selection and error estimation for support vector machines," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 9, pp. 1390–1406, 2012.

[69] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *International Joint Conference on Artificial Intelligence*, 1995.

[70] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics Surveys*, vol. 4, pp. 40–79, 2010.

[71] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Chapman & Hall, 1993.

[72] Regione Liguria, "Weather Data of Regione Liguria," http://www2.arpalombardia.it/siti/arpalombardia/meteo/richiesta-dati-misurati/Pagine/RichiestaDatiMisurati.aspx, 2016, online; accessed 3 May 2016.

[73] Regione Lombardia, "Weather Data of Regione Lombardia," http://www.cartografiarl.regione.liguria.it/SiraQualMeteo/script/PubAccessoDatiMeteo.asp, 2016, online; accessed 3 May 2016.