Research Article

TRR
JOURNAL OF THE TRANSPORTATION RESEARCH BOARD

Transportation Research Record
1–10
© National Academy of Sciences:
Transportation Research Board 2019
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0361198119840339
journals.sagepub.com/home/trr
SAGE

Predicting Near-Term Train Schedule Performance and Delay Using Bi-Level Random Forests

Mohammad Amin Nabian[1], Negin Alemazkoor[1], and Hadi Meidani[1]

Abstract
Accurate near-term passenger train delay prediction is critical for optimal railway management and providing passengers with accurate train arrival times. In this work, a novel bi-level random forest approach is proposed to predict passenger train delays in the Netherlands. The primary level predicts whether a train delay will increase, decrease, or remain unchanged in a specified time frame. The secondary level then estimates the actual delay (in minutes), given the predicted delay category at primary level. For validation purposes, the proposed model has been compared with several alternative statistical and machine-learning approaches. The results show that the proposed model provides the best prediction accuracy compared with other alternatives. Moreover, constructing the proposed bi-level model is computationally cheap, thereby being easily applicable.

Punctuality is one of the major performance measures for passenger trains, which can highly affect the demand for this mode of transportation (*1*). A passenger train is punctual if it arrives at the planned time that was agreed on between passengers and railways operators. This agreement is usually reflected in a timetable. A timetable specifies assignments for each train, including but not limited to the entry and exit times at every station on the train track. To design a timetable, several physical constraints and safety operation regulations, such as maximum allowable train speed and minimum headway between trains, must be considered (*2*). Ideally, train operation should follow the schedule in the timetable. However, actual operation can be affected by several factors, such as weather, facility failure, and drivers' and travelers' behavior. Consequently, deviations from timetables and delays are often unavoidable in railway operation.

A train is delayed if it arrives after the planned time. Train delays can cause significant monetary and nonmonetary losses to passengers and operators. About 22% of trains in the British national railways system were reported to be delayed in 2006/2007, which resulted in a total delay of 14 million minutes (*3*). Considering the monetary value of each minute of delay for all delayed passengers to be £73, delays caused costs of more than £1 billion. To alleviate such a high cost of train delays, it is important to investigate the potential impact of railway system characteristics on train delays. To this end, several works have tried to develop descriptive and predictive models that establish a function between delay and influential characteristics of the railway system. Such functions can be used for short-term prediction of train delays (*4*, *5*) as well as for optimal design of timetables and resource scheduling (*6*–*9*).

Different approaches have been used to model and analyze train delays. One of these approaches that has been widely used is regression. In (*10*), Norwegian train data was studied and a significant correlation between train departure delay, number of passengers, and occupancy ratio was observed. It was concluded that better management of boarding and alighting passengers can significantly reduce the delay. In (*11*), linear regression, along with a combinatorial model, was used to forecast delays at consecutive stations of Swiss Railways, given the information provided by online train monitoring data. In (*12*), robust linear regression was applied to explain delay propagation and evaluate the correlation between feeder train arrival delay and connecting train departure delay.

[1]Department of Civil and Environmental Engineering, University of Illinois at Urbana–Champaign, Urbana, IL

Corresponding Author:
Address correspondence to Mohammad Amin Nabian: mnabia2@illinois.edu

More recently, machine-learning approaches have been used for delay prediction. Similar to statistical approaches, machine-learning approaches aim to estimate/predict a quantity of interest given a set of input/explanatory variables. In (*13*), artificial neural network (ANN) was used to predict passenger train delays for Iranian railways. ANN was found to perform better than decision tree and multinomial logistic regression in its prediction accuracy. In (*14*), shallow and deep extreme leaning algorithms were used to predict delay in Italian railways. It was shown that such advanced analytical approaches can significantly outperform the state-of-art approaches that are currently used for delay prediction in Italian railways. In (*15*), for the first time, support vector regression (SVR) was used to predict passenger train delays. Performance of SVR was compared with ANN, and it was found that SVR results in better prediction accuracy. This shows the importance of comparing different approaches rather than preselecting and training a single model. It is worth mentioning that for a specific rail network, different models can be compared to choose the model with the highest accuracy. However, it should be noted that it is not possible to compare available models and recommend one single model that always achieve the highest accuracy for all railway networks. This is because performance of a model may vary for different rail networks, considering the unique characteristics of the networks that are not captured by the prediction models.

This work aims to predict passenger train delays in the Netherlands, where rail transport is a major mode of transportation that serves more than 1 million passengers on a daily basis (*16*). In particular, one goal is to understand how the delay of each train in service changes in 20 min, that is, whether the current train delay increases, remains unchanged, or decreases. Moreover, another goal is to accurately quantify the change in delay. These two goals are actually motivated by the recommendation of ProRail and NS, which are the main railway system operators in the Netherlands (*16*). From a machine-learning perspective, the former goal is a classification task, and the latter one is a regression task. Therefore, this work is dealing with a coupled classification–regression task. Research studies in the literature mainly deal with a single classification or regression task, and therefore might have shortcomings when used for a coupled classification–regression task. A new method, called "bi-level random forest" in what follows, is therefore proposed which is shown to be superior in prediction accuracy performance, when compared with the performance of other methods commonly used in the literature for near-term train delay prediction. To be precise, it is shown in the Results section that the available methods in the literature, compared with the method proposed here, result in lower prediction accuracy or inconsistent predictions.

Random forest was first introduced in (*17*), and has been widely applied in various fields (*18–20*). In (*21*), random forest was successfully used to predict conflict-free train running and dwell time. This work proposes a bi-level random forest approach for a more general case, in which a conflict-free condition is not a requirement. At primary level, a random forest model is used as a classifier to predict whether the current train delay will increase, decrease, or remain unchanged. At secondary level, random forest regression models are used to quantify the change in delay. To validate this approach, random forest has been compared with several alternative approaches including linear regression, multinomial logistic regression, decision tree, *K*-nearest neighbors, and support vector machine/regression. It can be observed that the proposed bi-level random forest outperforms these alternative approaches in its prediction accuracy for passenger train delay prediction in the Netherlands' railways.

The rest of this paper is organized as follows. The next section contains an overview of the train delay prediction problem in the Netherlands' railways. Next, the proposed methodology is introduced in detail. The Results and Discussion section includes the empirical results of the proposed method, and a discussion on the performance of the proposed method compared with other common methods in the literature. Finally, the last section highlights future research directions and offers concluding remarks.

Problem Description

This section overviews the near-term train schedule performance prediction problem, the raw data set used in this work, and data cleaning and mining steps.

Train Delay Forecast

Passenger train transportation is a key mode of transport in the Netherlands that serves more than a million passengers daily. Netherlands Railways is the largest passenger train operator and operates about 6,000 trains daily. Reliable real-time data is of high importance for smooth, punctual operations. Over the past decade, real-time data for delay in the entire rail network has been recorded and made available to the operators. However, making accurate near-term predictions for the rail network performance is still a challenge. Currently, in the Netherlands, the most commonly used estimate for near-term delays is that the current delay remains unchanged (*16*). In other words, it is assumed that the train neither loses nor makes up time when behind. However, this is usually not the case, thereby necessitating a more accurate model.

In this work, using the raw data provided by the ProRail and NS (*16*), the aim is to develop a data-driven predictive model that, once trained, given the real-time delay of a passenger train, can predict that train's delay category (decrease, equal, or increase) and the actual delay about 20 min later. The developed predictive model can be very effective in constructing accurate passenger information systems such as smartphone applications for real-time train schedules. Additionally, this predictive model can help the decision makers to come up with the optimal management strategies for the system.

Description of Raw Data

The data used in this study are provided by the Netherlands' railway infrastructure managers, ProRail. The raw data are for the period of September 4, 2017 to December 9, 2017, and include the following attributes (*16*):

- **Planned timetable**: The planned timetable consists of information on the train schedule as it was planned in advance. The timetable is the same for Mondays, Tuesdays, Thursdays, and Fridays. On Wednesdays, on a busy part of the network between Eindhoven and Amsterdam, additional trains are operated. The weekend timetable is not included.
- **Actual historical train performance**: This dataset includes the realized timetable, that is, the realized departure and arrival times, in seconds, for each of the operated trains.
- **Crew schedules**: The crew schedule includes all scheduled driver changes. It should be noted that train drivers usually stay on the same train during stops at stations. However, it is sometimes possible to change drivers.
- **Rolling stock circulation**: Usually a train travels in the same rolling stock composition. However, it is possible to change the composition after a stop at a station. This dataset contains scheduled rolling stop composition changes during intermediate stops. Additionally, all scheduled rolling stop connections at the end stations of trains are also given.
- **Infrastructure data**: The distances between consecutive stations are given in this dataset.
- **Weather conditions overview**: This dataset provides a daily overview of the Netherlands' average weather conditions, including maximum wind speed, maximum, minimum, and average temperature, and rain depth. This overview is obtained from the Dutch weather agency (KNMI).

Feature Construction and Data Cleaning

A variety of data cleaning and data transformation tasks are usually necessary for organizing the inputs to any prediction model. The available raw data include approximately 10,000,000 data points for a 13-week period. Data for Wednesdays are excluded as Wednesday's timetable is different from other weekdays. Trips with delays of more than 15 min are also excluded, because delays larger than 15 min may cause special cases and consequently trip cancellation. The goal is to predict the train delay category and actual delay about 20 min after a realization. To construct the training/test data, samples are randomly (without replacement) drawn from the raw data. A sample includes a train ID, location, planned time, and realized time. The location of these sampled trains is referred to as the "origin." Next, the raw data are searched to select the train location with a planned time scheduled for 20 min later. This location is referred to as the "destination." Not all samples have an event precisely 20 min after the origin realization. In this case, the last planned event before 20 min is considered.

Several features summarized in Table 1 are considered as the independent variables for the predictive model. The dependent variable is train delay. Origin delay can affect train delays, because if delay at the origin station is relatively large, train drivers will try to compensate for the delay by, for instance, reducing the time they spend at stations. The extent to which the drivers can compensate for the delay also depends on the distance and the planned time difference between the stations, and therefore, these two factors can also affect the delay. When a train is near the end of its ride, it is possible that the rolling stock of this train has to be used for another train right after arriving, and thus, when the initial train is delayed, a delay propagation is caused. Driver change is also a source of delay propagation, because if a train driver is on a delayed train but the driver needs to switch trains at the next station, it can be reasoned that the departing train will then also be delayed because of having to wait for its driver. Peak hours can also adversely affect train delays, as, for instance, trains sometimes need to have longer stops at stations than usual to serve the in-and-out crowd. Historical train delay statistics can also guide the predictor model to better capture the pattern of historical delays between specific stations. Trains sometimes share a track, so if the front train on a track is delayed, it can delay the following trains as well. Severe weather conditions might also force some trains to ride slower than usual, thus resulting in a delay propagation in the system.

Any absolute delay change between the origin and destination stations of greater than or equal to 2 min is considered a "jump." It was observed that by creating the training/test data using the proposed procedure,

Table 1. Summary of the Features Constructed from the Raw Data

Feature	Notation	Description
Origin delay	d_s	Delay at the origin station
Distance	ℓ	Distance between origin and destination stations
Planned time diff.	δP	Difference between the planned time at origin and destination stations
No. *A–V*	N_{A-V}	Number of train arrivals (*A*) and departures (*V*) between the origin and destination stations with a long (i.e., 5 min) stop
No. KA–KV	$N_{\mathrm{KA-KV}}$	Number of train arrivals (KA) and departures (KV) between the origin and destination stations with a short (i.e., 1 min) stop
Composite change	I_c	Whether the composite has changed during the trip between origin and destination stations (binary variable)
Driver change	I_d	Whether the driver has changed during the trip between origin and destination stations (binary variable)
Peak hour	I_r	Whether the trip is during the peak hour (i.e., 7:00–9:00 a.m. or 4:00–6:00 p.m.) (binary variable)
Delay diff. mean	δd_{avg}	Mean value of the difference between historical delays at origin and destination stations
Delay diff. mode	δd_{mode}	Mode of the difference between historical delays at origin and destination stations
Max delay diff.	$\delta d_{\max}$	Maximum of the difference between historical delays at origin and destination stations
Min delay diff.	$\delta d_{\min}$	Minimum of the difference between historical delays at origin and destination stations
Freq. delay diff. > 1	δd_{f_1}	Frequency of the historical events with difference between delays at origin and destination stations greater than 1
Freq. delay diff. > 4	δd_{f_4}	Frequency of the historical events with difference between delays at origin and destination stations greater than 4
Freq. delay diff. < -1	$\delta d_{f_{-1}}$	Frequency of the historical events with difference between delays at origin and destination stations less than -1
Freq. delay diff. < -4	$\delta d_{f_{-4}}$	Frequency of the historical events with difference between delays at origin and destination stations less than -4
Front train mean delay	δf_{avg}	Mean of the historical front train delays
Front train delay mode	δf_{mode}	Mode of the historical front train delays
Freq. front train delay > 1	δf_{f_1}	Frequency of the historical events with front train delays of greater than 1
Freq. front train delay > 4	δf_{f_4}	Frequency of the historical events with front train delays of greater than 4
Freq. front train delay < -1	$\delta f_{f_{-1}}$	Frequency of the historical events with front train delays of less than -1
Freq. front train delay < -4	$\delta f_{f_{-4}}$	Frequency of the historical events with front train delays of less than -4
Avg. wind speed	W_{avg}	Wind speed daily average
Max. wind speed	$W_{\max}$	Maximum daily wind speed
Avg. temperature	T_{avg}	Temperature daily average
Min. temperature	$T_{\min}$	Minimum daily temperature
Max. temperature	$T_{\max}$	Maximum daily temperature
Rain depth	R	Average daily rain depth (mm)

Note: diff. = difference; no. = number; freq. = frequency; avg. = average; max. = maximum; min. = minimum.

about 80% of the processed data did not include a jump event. Therefore, to balance the training/test data and give the predictor the chance to be exposed to all delay scenarios equally, a sufficiently large pool of processed data was first generated following the proposed procedure, and then 180,000 data points were subsampled from the processed data, out of which one-third of the subsampled data involved no jump, one-third involved a jump in the positive direction (increase in delay), and the rest of the subsampled data involved a jump in the negative direction (decrease in delay). Out of 60,000 data points for each scenario, 10,000 data points were reserved as the test data. Therefore, the training data consist of 150,000 data points, and the test data consist of 30,000 data points.

Methodology

Decision Tree Learning

Decision tree learning is a nonparametric unsupervised predictive modeling approach which is extensively used in machine-learning, data mining, and statistics. Given observations from an item, decision tree learning is utilized to infer information about that item's target value. Decision tree models in which the target variable can take a discrete set of values are called classification trees. In classification trees, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision tree models in which the target variable can take continuous values are called regression trees (*22*).

Decision trees follow a decision rule that can produce a prediction given a set of features. One approach to build the rules is with a sequence of simple tests, in which each test uses the results of all previous tests. This class of rules can be depicted as a tree, on which each test is represented by a node, and the possible outcomes of the test are represented by the edges. To predict a test item with such a tree, features of the test item are fed into the first node, and the outcome of the test performed in that node determines which node it goes to at the next step, and so on, until the test item arrives at a leaf. Finally, the test item outcome is predicted using a voting or averaging scheme at the leaf nodes (*23*).

Assuming the decision function and split threshold are defined, the process starts with the root node, then recursively splits the data at that node, passing the left pool of data to the left and the right pool of data to the right, until the leaf nodes are arrived at (*23*). Splitting involves choosing a decision function to give the best split for a node. The main modeling questions faced are how to choose the best split, and when to stop.

Stopping is relatively straightforward. It is hard to select a decision function with a small number of data points, so splitting should be stopped when there are too few data points at a node. Moreover, constructing a too deep tree tends to result in overfitting, so usually the process does not allow more than a fixed depth of splits. Cross-validation can be used to select the best tree depth.

At each tree node, it is natural to choose a limiting feature and a threshold that is the most informative. An informative split means that the uncertainty about data class/value is reduced significantly if it is known whether it goes left or right (*23*) using that split. To choose the best split, it is necessary to determine how informative the split is.

Given the training samples $\mathbf{x}_i \in \mathcal{R}^k$, $i = 1 \cdots n$, and a label vector $\mathbf{y}_i \in \mathcal{R}^l$, where k and l are the input and output dimensionality respectively and $\mathcal{R}$ is the set of real numbers, a decision tree recursively splits the feature space such that the training samples with similar labels or values are grouped together. Let us denote the data at node j by D_j. For each partitioning candidate $\theta_j = (f_j, t_j)$ with a feature f_j and threshold t_j, the data D_j is partitioned into $D_j^{(l)}(\theta_j)$ and $D_j^{(r)}(\theta_j)$ subsets:

$$D_j^{(l)}(\theta_j) = \left\{(\mathbf{x}, \mathbf{y}) \in D_j : \mathbf{x}_i^{f_j} < = t_j\right\} \tag{1}$$

$$D_j^{(r)}(\theta_j) = \left\{(\mathbf{x}, \mathbf{y}) \in D_j : \mathbf{x}_i^{f_j} > t_j\right\} \tag{2}$$

The impurity at node j is computed using an impurity function, H. The choice of impurity function depends on the task being performed (classification or regression). The overall impurity of split at node j is defined as

$$G(D_j, \theta_j) : = \frac{|D_j^{(l)}(\theta_j)|}{|D_j(\theta_j)|} H(D_j^{(l)}(\theta_j)) + \frac{|D_j^{(r)}(\theta_j)|}{|D_j(\theta_j)|} H(D_j^{(r)}(\theta_j)) \tag{3}$$

where $|\cdot|$ is the cardinality function. The optimal split θ_j^* is obtained by minimizing the impurity

$$\theta_j^* = \underset{\theta_j}{\operatorname{argmin}}\, G(D_j, \theta_j) \tag{4}$$

The above procedure is recursed for subsets $D_j^{(l)}(\theta_j^*)$ and $D_j^{(r)}(\theta_j^*)$ until the maximum allowable depth is reached.

Classification Criteria. If the target is a classification outcome taking values in $1, 2, ..., l$ with n_j observations, the proportion of class m observations is calculated as

$$p_{jm} = \frac{1}{n_j} \sum_{\mathbf{x}_i \in D_j} I(\mathbf{y}_i = m) \tag{5}$$

Common measures of impurity include Gini, H_G, cross-entropy, H_E, and misclassification, H_M, which are, respectively, defined as

$$H_G(D_j) : = \sum_m p_{jm}(1 - p_{jm}) \tag{6}$$

$$H_E(D_j) : = - \sum_m p_{jm} \log(p_{jm}) \tag{7}$$

$$H_M(D_j) : = 1 - \max(p_{jm}) \tag{8}$$

Regression Criteria. If the target is a continuous value, then for node j, with n_j observations, common criteria for determining locations for future splits are mean absolute error (MAE), which minimizes the ℓ_1 error using median values at leaf nodes, and mean squared error (MSE), which minimizes the ℓ_2 error using mean values at leaf nodes. H_{MAE} is defined as

$$H_{\mathrm{MAE}}(D_j) : = \frac{1}{n_j} \sum_{\mathbf{x}_i \in D_j} |y_i - \bar{y}_{jm}| \tag{9}$$

where

$$\bar{y}_{jm} = \frac{1}{n_j} \sum_{\mathbf{x}_i \in D_j} (y_i) \tag{10}$$

Similarly, H_{MSE} is defined as

$$H_{\mathrm{MSE}}(D_j) : = \frac{1}{n_j} \sum_{\mathbf{x}_i \in D_j} (y_i - \bar{y}_{jm})^2 \tag{11}$$

Random Forest

A random forest is a meta estimator that fits several decision trees on various subsamples of the training data and uses voting or averaging to improve the accuracy of predictions and prevent overfitting (*22*, *24*). The size of subsamples can be smaller than or equal to the size of the training data. In case of equal size, samples are drawn with replacement. Because of the random nature of these subsamples, a different tree is obtained each time a tree is trained on a subsample of a training dataset. None of these trees are particularly optimal (they are sometimes called weak learners) (*23*). An effective strategy is to produce many such trees (a random forest), and allow each tree to vote or take part in the final averaging. It is proven that this strategy is extremely effective (*23*, *24*).

Bi-Level Random Forest

As stated earlier, this investigation is interested in understanding how the delay of each train in service changes in 20 min (a classification task) and also the actual delay of that train (a regression task). To perform this coupled classification–regression task, a new predictive model is proposed, referred to as bi-level random forest, which consists of a two-level random forest, for which a classification task is performed at primary level, followed by regression tasks at the other level. The proposed bi-level random forest is formed of a random forest classifier, and l random forest regression models, where l is the number of class labels in the classification task. Therefore, a bi-level random forest consists of $l + 1$ random forests. More precisely, each regression model is trained specifically for data within a class. Different decision rules (e.g., Gini, cross-entropy, or misclassification for the primary level, and mean square or absolute error for the secondary level) may be used to develop each of the random forests. At each node of each decision tree in the bi-level random forest, the data feature which minimizes the selected decision rule is used as the nodal feature for that node. Different decision rules may be used for different random forests in the bi-level random forest. Similarly, the depth of each random forest may be different compared with others, and the optimal depth for each random forest can be easily obtained using cross-validation. This predictive model is useful when a coupled classification–regression task is required, when, because of the nature of the dataset and features, performing a regression task followed by a simple classification will not yield accurate results. Further discussion on the advantages of this method is postponed to the next section. The algorithm for bi-level random forest is expressed as

Algorithm 1 Bi-level Random Forest

```
1. Draw $s$ set of samples (with or without replacement) from the
   training data, each of size $n_s$.
2. For each set of samples, grow a classification tree. At each
   node, choose the best split by solving an optimization problem
   defined in Equation 4. Use cross-validation to find the optimal
   tree depth.
3. Split the training data into $\ell$ subsets based on the predicted
   class label.
4. For each of the $\ell$ subsets of training data, draw $s^{(i)}$,
   $i \in \{1, \cdots, \ell\}$ set of samples (with or without replacement),
   each of size $n_s^{(i)}$.
5. For each set of samples of a subset of training data, grow a
   regression tree. At each node, choose the best split by solving
   an optimization problem defined in Equation 4. Use cross-
   validation to find the optimal tree depth.
6. To predict a new data class label and target value, first
   aggregate the predictions of $s$ classification trees (by voting) to
   evaluate the class label, $m$. Then, aggregate the predictions of
   $s^{(m)}$ regression trees (by averaging) to predict the target value.
```

Results and Discussion

In this section, first, several performance measures are first introduced to evaluate the accuracy of the classification and regression tasks. Next, there is an illustration of a significance test for selecting important data features. Then, results for train delay prediction are presented using the proposed bi-level random forest predictive model, and are compared with the results for alternative predictive models. Finally, a discussion is provided to highlight the effectiveness of the proposed predictive model for train delay prediction when compared with other existing models.

Performance Measures

Each train in the test data will have three predicted values: (*a*) jump, which is binary, with "1" showing a jump in delay, and "0" showing no jump; (*b*) direction, which has three categories to show increase, decrease, or no change in the delay; and (*c*) delay prediction, rounded to the nearest minute. Accuracy of jump and direction prediction is scored using an *F*-score. Let us define the precision, α_P, and recall, α_R, accuracy measures as follows

$$\alpha_P := \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{12}$$

$$\alpha_R := \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{13}$$

where

TP (true positive) is the number of times that a jump was predicted and actually occurred;

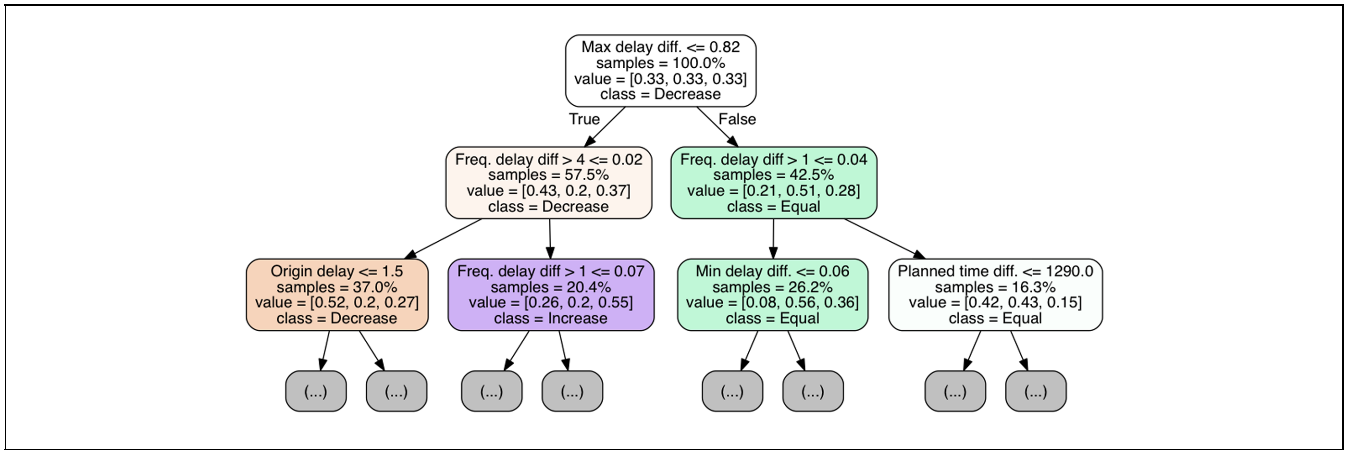
Max delay diff. <= 0.82
samples = 100.0%
value = [0.33, 0.33, 0.33]
class = Decrease
True
False
Freq. delay diff > 4 <= 0.02
samples = 57.5%
value = [0.43, 0.2, 0.37]
class = Decrease
Freq. delay diff > 1 <= 0.04
samples = 42.5%
value = [0.21, 0.51, 0.28]
class = Equal
Origin delay <= 1.5
samples = 37.0%
value = [0.52, 0.2, 0.27]
class = Decrease
Freq. delay diff > 1 <= 0.07
samples = 20.4%
value = [0.26, 0.2, 0.55]
class = Increase
Min delay diff. <= 0.06
samples = 26.2%
value = [0.08, 0.56, 0.36]
class = Equal
Planned time diff. <= 1290.0
samples = 16.3%
value = [0.42, 0.43, 0.15]
class = Equal
(...)

Figure 1. The first three layers in the classifier level of the trained bi-level random forest predictive model.

FP (false positive) is the number of times that a jump was predicted but did not occur in reality;

FN (false negative) is the number of times that a jump occurred, but was not predicted. The F-score is defined as

$$F = \frac{2\alpha_P\alpha_R}{\alpha_P + \alpha_R} \tag{14}$$

The F-score is between 0 and 1, where 1 is the perfect score.

The delay prediction accuracy is scored by calculating the root mean square error α_{RMS} of all prediction values, defined as

$$\alpha_{\text{RMS}} = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{15}$$

where

$\hat{y}$ is the delay prediction,

y is the actual delay, and

n is the number of test data points.

Finally, the overall prediction accuracy score α is defined as

$$\alpha : = 10F_j + 5F_d - \alpha_{\text{RMS}} \tag{16}$$

where F_j and F_d are, respectively, the F-scores for delay jump and delay direction predictions. The definition of delay categories and the weights for each term in the overall accuracy measure are suggested by ProRail (*16*). Delay jump and direction prediction performance terms have relatively larger weights compared with actual delay prediction error, as it is reported in (*16*) that delay jump and direction have the highest importance for ProRail and NS decision makers for their system management purposes.

Results

Using the training data with all the 28 constructed features, a bi-level random forest is trained which consists of a random forest classifier and three random forest regression models (corresponding to "decrease,""equal," and "increase" class labels). The input to the classifier model is the values of these 28 features for a test item, and the output is the delay class labels. The input to the regression models is also the values of these 28 features for a test item, and the output is the actual delay in minutes. In training of these forests, to calculate the best nodal split, the cross-entropy impurity, H_E, and MSE, H_{MSE}, are used for forest classifier and forest regression models, respectively. The best nodal split is then calculated using Equation 4. For each of the four random forests, cross-validation is used to specify the optimal number of trees and depth of trees. It is observed that increasing the size of the training dataset does not result in a considerable change in prediction accuracy and, therefore, it is believed that the training dataset is of appropriate size. The first three layers in the classifier level of the trained bi-level random forest predictive model are depicted in Figure 1. At each node, the first row shows the split (feature and threshold) criteria of that node, the second row is the percentage of the training data that falls into that node, the third row shows the proportion of the training data in that node corresponding to each class label, and the last row shows the class label with the highest proportion in that node. Table 2 shows the performance of the bi-level random forest predictive model on the test data. A comprehensive comparison between the performance of the proposed bi-level random forest model and other machine-learning predictive models is presented in the next section.

Table 2. Performance of the Bi-level Random Forest Predictive Model on the Test Data

P_D	P_E	P_I	F_j	F_d	α_{RMS}	α
0.93	0.67	0.62	0.84	0.77	2.24	10.01

Note: P_D, P_E, and P_I, respectively, stand for the proportion of accurate prediction of "decrease,""equal," and "increase" classes.

Table 3. A Summary of the Performance of a Variety of Classification Models

Rank	Classifier	P_D	P_E	P_I	$10F_j + 5F_d$
1	Random forest	0.93	0.67	0.62	12.25
2	Gradient boosting	0.92	0.67	0.55	12.22
3	Adaboost	0.91	0.65	0.55	12.16
4	SVM	0.91	0.70	0.55	11.95
5	Extra tree	0.90	0.68	0.57	11.93
6	Logistic regression	0.85	0.67	0.55	11.66
7	Decision tree	0.82	0.59	0.57	11.55
8	KNN	0.78	0.55	0.53	10.91
9	Naive Bayes	0.48	0.84	0.40	8.83

Note: P_D, P_E, and P_I stand for percentage of accurate predictions for "decrease,""equal," and "increase" classes, respectively.

Table 4. A Summary of the Performance of a Variety of Regression Models

Regression model	RF	SVR	2nd Order Polynomial	Linear	3rd Order Polynomial (d = 3)
α_{RMS}	2.01	2.34	2.21	2.37	3.21

Note: RF = Random Forest.

Significant Features

The training and test data consist of 28 features. To evaluate the relative importance of a feature with respect to the predictability of the target variable, the relative rank (depth in the tree) of that feature in a node can be used (*22*). Those features that are used in the top parts of the tree contribute to the final prediction decision of a larger fraction of the input data. Therefore, the expected fraction of the samples for which a feature contributes to their predictions can be used to estimate the feature's relative importance. Using the random forest predictor for classification, a score is calculated for each feature based on the expected fraction of samples it contributes to. This score reflects the relative importance of each feature. Considering a threshold of 0.001, it was found that the following features are of significance with respect to the classification task: origin delay (d_s), distance (ℓ), planned time difference (δP), number of train arrivals and departures (N_{A-V}, $N_{\text{KA-KV}}$), composite change (I_c), driver change (I_d), peak hour (I_r), and historical delay statistics (δd_{avg}, δd_{mode}, $\delta d_{\max}$, $\delta d_{\min}$, δd_{f_1}, δd_{f_4}, $\delta d_{f_{-1}}$, and $\delta d_{f_{-4}}$). Historical front train delay statistics are not considered significant because sometimes front trains ride on a different track from trains behind them, and therefore, the front train delay will not affect the delay of following trains. However, the study did not have access to the track ID for each train and assumed that all trains ride on a single track. It is expected that the predictive model will perform better in the presence of information on train track ID. Moreover, the weather condition features could not be considered significant as only the daily average of weather data was available; it is expected that more precise weather data such as hourly average data can improve the model predictions.

Performance Comparison between Different Predictive Models

A variety of predictive models were implemented on the training data in an effort to find the best predictive model for the given problem. For brevity's sake, an introduction to these predictive models has not been provided and, instead, readers are referred to relevant references for gradient boosting (*25*), adaboost (*26*), support vector machine (SVM) (*27*), extra tree (*28*), logistic regression (LR) (*29*), *K*-nearest neighbors (*30*), naive Bayes (*31*), linear regression (*32*), polynomial regression (*33*), and SVR (*34*) models. Table 3 represents a summary of the performance of a variety of classification models. Table 4 also represents a summary of the performance of a variety of regression models.

Discussion

It is evident from Table 3 that the random forest classifier has a superior performance compared with the other classification models considered in this work. In addition, Table 4 shows that the random forest regression model has a superior performance compared with the other regression models. Therefore, random forest classification and regression models provide the best accuracy (among the other models investigated here) for the given problem. However, it is obvious that these two predictive models cannot be used independently or, otherwise, in some cases results might be inconsistent. For example, if these two predictive models are used for a test item, it might be possible that the classifier predicts an increase in delay, whereas the regression model predicts a 1.50 min increase in delay (which corresponds to the "equal" class). The proposed bi-level random forest predictive model inherently avoids this inconsistency by training a unique regression model for each of the class labels. Therefore, the proposed algorithm is suitable for coupled classification–regression tasks.

Another alternative to perform these classification and regression tasks is to first perform a regression task, and then, based on the outcome of the regression, infer about the jump and direction class of the input data. To this end, a single random forest regression model (which has shown a superior accuracy compared with other considered models) has been trained and then the performance of this model tested on the test data. It turns out that the accuracy of this model is as follows: $F_j = 0.80$, $F_d = 0.75$, $\alpha_{\text{RMS}} = 2.01$, and $\alpha = 9.74$. On the one hand, it is observed that the random forest regression model outperforms the proposed bi-level random forest with respect to the α_{RMS} measure. However, this was expected as in the bi-level random forest, the misclassification error from the classification level penetrates the regression level. On the other hand, it is evident that the proposed bi-level random forest model outperforms the random forest regression model with respect to F_j, F_d, and, most importantly, α measures (which is the ultimate measure of model performance), showing the effectiveness of the proposed method for coupled classification–regression tasks.

Constructing the proposed bi-level random forest predictive model is computationally cheap, and is therefore easily applicable. With 150,000 items of training data, it took 1.84 s to train the bi-level random forest (consisting of one classification and three regression forests) on a quad core 2.5 GHz MacBook Pro with 16 gigabytes (GB) of random-access memory (RAM).

Conclusion

The goal of this study was to predict the near-term delay category and the actual delay of the Netherlands' passenger trains. To this end, a bi-level random forest model was proposed. This model consists of a classification forest at the primary level, and several regression forests at the secondary level. Specifically, at primary level the model predicts whether the current delay will decrease, increase, or remain unchanged within the next 20 min from the present time. At secondary level, the model quantifies the amount of delay (in minutes). It was shown that the proposed model was successful in providing accurate predictions for delay category and actual delay, and thus can be regarded as a viable approach in situations for which delay prediction involves a coupled classification–regression task. The proposed model was compared with several potential alternative approaches common in the literature. It was observed that, because of the nature of the problem, which involved a coupled classification–regression task, the proposed bi-level model provided the most accurate predictions for the Netherlands' railway system.

Author Contributions

All authors contributed to all aspects of the study including data preparation, methodology development, analysis and interpretation of results, and manuscript preparation. All authors reviewed the results and approved the final version of the manuscript.

References

1. Nyström, B. *Aspects of Improving Punctuality: From Data to Decision in Railway Maintenance*. PhD thesis. Luleå tekniska universitet, 2008.
2. Lee, W.-H., L.-H. Yen, and C.-M. Chou. A Delay Root Cause Discovery and Timetable Adjustment Model for Enhancing the Punctuality of Railway Services. *Transportation Research Part C: Emerging Technologies*, Vol. 73, 2016, pp. 49–64.
3. Burr, T., S. Merrifield, D. Duffy, J. Griffiths, S. Wright, and G. Barker. *Reducing Passenger Rail Delays by Better Management of Incidents*. National Audit Office for the Office of Rail Regulation, 2008.
4. Berger, A., A. Gebhardt, M. Müller-Hannemann, and M. Ostrowski. Stochastic Delay Prediction in Large Train Networks. In *OASIcs: OpenAccess Series in Informatics*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011, Vol. 20.
5. Kecman, P., and R. M. Goverde. Online Data-Driven Adaptive Prediction of Train Event Times. *IEEE Transactions on Intelligent Transportation Systems*, Vol. 16, No. 1, 2015, pp. 465–474.
6. Medeossi, G., G. Longo, and S. de Fabris. A Method for Using Stochastic Blocking Times to Improve Timetable

Planning. *Journal of Rail Transport Planning & Management*, Vol. 1, No. 1, 2011, pp. 1–13.
7. Andersson, E. V., A. Peterson, and J. T. Krasemann. Quantifying Railway Timetable Robustness in Critical Points. *Journal of Rail Transport Planning & Management*, Vol. 3, No. 3, 2013, pp. 95–110.
8. Andersson, E., A. Peterson, and J. Törnquist Krasemann. Improved Railway Timetable Robustness for Reduced Traffic Delays: A Milp Approach. In *6th International Conference on Railway Operations Modelling and Analysis: Rail Tokyo*, 2015.
9. Jovanović, P., P. Kecman, N. Bojović, and D. Mandić. Optimal Allocation of Buffer Times to Increase Train Schedule Robustness. *European Journal of Operational Research*, Vol. 256, No. 1, 2017, pp. 44–54.
10. Olsson, N. O., and H. Haugland. Influencing Factors on Train Punctuality: Results from Some Norwegian Studies. *Transport Policy*, Vol. 11, No. 4, 2004, pp. 387–397.
11. Flier, H., R. Gelashvili, T. Graffagnino, and M. Nunkesser. Mining Railway Delay Dependencies in Large-Scale Real-World Delay Data. In *Robust and Online Large-Scale Optimization*, Springer, 2009, pp. 354–368.
12. Goverde, R. M. *Punctuality of Railway Operations and Timetable Stability Analysis*. TRAIL Thesis Series T2005/10. PhD thesis. Delft, 2005.
13. Yaghini, M., M. M. Khoshraftar, and M. Seyedabadi. Railway Passenger Train Delay Prediction via Neural Network Model. *Journal of Advanced Transportation*, Vol. 47, No. 3, 2013, pp. 355–368.
14. Oneto, L., E. Fumeo, G. Clerico, R. Canepa, F. Papa, C. Dambra, N. Mazzino, and D. Anguita, Train Delay Prediction Systems: A Big Data Analytics Perspective. *Big Data Research*, Vol. 11, 2018, pp. 54–64.
15. Marković, N., S. Milinković, K. S. Tikhonov, and P. Schonfeld. Analyzing Passenger Train Arrival Delays with Support Vector Regression. *Transportation Research Part C: Emerging Technologies*, Vol. 56, 2015, pp. 251–262.
16. The Institute for Operations Research and the Management Sciences. *2018 RAS Problem Solving Competition: Train Delay Forecasting*. http://connect.informs.org/railway-applications/awards/problem-solving-competition/new-item2. Accessed July 23, 2018.
17. Ho, T. K. Random Decision Forests. In *Proc.,Third International Conference on Document Analysis and Recognition*, IEEE, 1995, Vol. 1, pp. 278–282.
18. Gislason, P. O., J. A. Benediktsson, and J. R. Sveinsson. Random Forests for Land Cover Classification. *Pattern Recognition Letters*, Vol. 27, No. 4, 2006, pp. 294–300.
19. Bosch, A., A. Zisserman, and X. Munoz. Image Classification Using Random Forests and Ferns. In *ICCV 2007. IEEE 11th International Conference on Computer Vision, 2007*. IEEE, 2007, pp. 1–8.
20. Cutler, D. R., T. C. Edwards Jr, K. H. Beard, A. Cutler, K. T. Hess, J. Gibson, and J. J. Lawler, Random Forests for Classification in Ecology. *Ecology*, Vol. 88, No. 11, 2007, pp. 2783–2792.
21. Kecman, P., and R. M. Goverde. Predictive Modelling of Running and Dwell Times in Railway Traffic. *Public Transport*, Vol. 7, No. 3, 2015, pp. 295–319.
22. Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-Learn: Machine Learning in Python. *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2825–2830.
23. Forsyth, D. *Probability and Statistics for Computer Science*. Springer, 2018.
24. Breiman, L. Random Forests. *Machine Learning*, Vol. 45, No. 1, 2001, pp. 5–32.
25. Friedman, J. H. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics*, 2001, pp. 1189–1232.
26. Hastie, T., S. Rosset, J. Zhu, and H. Zou. Multi-Class Adaboost. *Statistics and its Interface*, Vol. 2, No. 3, 2009, pp. 349–360.
27. Wu, T.-F., C.-J. Lin, and R. C. Weng. Probability Estimates for Multi-Class Classification by Pairwise Coupling. *Journal of Machine Learning Research*, Vol. 5, 2004, pp. 975–1005.
28. Geurts, P., D. Ernst, and L. Wehenkel. Extremely Randomized Trees. *Machine Learning*, Vol. 63, No. 1, 2006, pp. 3–42.
29. Peng, C.-Y. J., K. L. Lee, and G. M. Ingersoll. An Introduction to Logistic Regression Analysis and Reporting. *The Journal of Educational Research*, Vol. 96, No. 1, 2002, pp. 3–14.
30. Bhatia, N. Survey of Nearest Neighbor Techniques. *arXiv preprint arXiv:1007.0085*, 2010.
31. Rish, I. An Empirical Study of the Naive Bayes Classifier. In *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*, IBM, New York, 2001, Vol. 3, pp. 41–46.
32. Neter, J., M. H. Kutner, C. J. Nachtsheim, and W. Wasserman. *Applied Linear Statistical Models*, Vol. 4. Irwin Chicago, 1996.
33. Seifert, B., and T. Gasser. Local Polynomial Smoothing. In *Encyclopedia of Statistical Sciences* (Kotz, S., C. B. Read, and D. L. Banks, eds.), Vol. 7, 2004.
34. Basak, D., S. Pal, and D. C. Patranabis. Support Vector Regression. *Neural Information Processing: Letters and Reviews*, Vol. 11, No. 10, 2007, pp. 203–224.

The Standing Committee on Artificial Intelligence and Advanced Computing Applications (ABJ70) peer-reviewed this paper (19-05846).