ARTICLE

Machine Learning Techniques to Predict Cascading Delays on British Railway Network

Panukorn Taleongpong[a], Simon Hu[b], Zhoutong Jiang [c], Chao Wu[d],Sunday Popo-Ola[a], Ke Han[a]

[a]Center for Transport Studies, Department of Civil and Environmental Engineering, Imperial College London,SW7 2BU, UK;
[b]School of Civil Engineering, Zhejiang University/University of Illinois at Urbana-Champaign Institute (ZJU-UIUC Institute), Zhejiang University, Haining, 314400, China;
[c]Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, IL. 61801, USA;
[d]School of public affairs, Zhejiang University, Hangzhou, 310058, China

ARTICLE HISTORY
Compiled November 22, 2019

ABSTRACT
Cascading delays that propagate from a primary source along a railway network is an immediate concern for British railway systems. Complex non-linear interactions between various spatio-temporal variables govern the propagation of these delays which can avalanche throughout railways networks causing further severe disruptions. This paper introduces several machine learning techniques alongside data mining processes to create a framework that forecasts key performance indicators (KPIs). Not only do these frameworks provide great accuracy, they also allow for insight into understanding the mechanism of delay propagation with the railway network. This paper goes on to perform up to ten future steps for KPI predictions through state-of-the-art ML models for trains at a certain checkpoint. Further discussion on the improvements, applicability and scalability of this framework are also provided in this paper.

KEYWORDS
Railway delay prediction, cascading delays, machine learning, deep neural network, gradient boosting

1. Introduction

The British railway, dating back to 1825, is the oldest operational railway network in the world, the majority of which is owned and maintained by the Network Rail. Train operating companies (TOCs) provide transportation services to be run on the Network Rail's infrastructure and are members of the National Rail, whose purpose is to promote these services and provide the public with information on all passenger rail services (National Rail, 2017). The British rail industry is currently experiencing a stagnation in performance affecting a rapidly growing commuter population on a daily basis. RRUKA (2017) predicts the increase of cascading delays, train delays that are a result of its prior delays or the propagation of delay from any other train, from 600,000 minutes annually to 800,000 minutes annually in the last five years. With the number

of passengers travelling on British train networks almost doubling from 1 billion to 1.7 billion in the past two decades (Office Of Rail and Road, 2018), this trend will only continuously worsen unless appropriate measures are put into place. Due to the complex non-linear spatio-temporal variable interactions governing the propagation of these delays, they are inherently difficult to predict.

Many mathematical and statistical models have been established in the literature to predict delays and understand its mechanism of propagation throughout a railway network. Buker and Seybold (2012) propose the utilization of activity graphs for the computation of delay propagation and outlining of distribution functions that describe the random nature of delays. Each node in the activity graph describes delays represented by a cumulative distribution function concerning train status along its journey. The edges of the activity graph represent four activities (drive, stop, inter-linking, route conflict) that may affect the change of delay in the next status. For each node, a delay distribution is computed dependent upon its predecessors. Khadilkar (2016) propose the prediction of delay propagation along a transportation network with stochasticity induced in resource conflict as opposed to waiting for connecting trains. Trains in the network are assigned static priority classes that shall be used to resolve resource conflict. Corman, Goverde, and D'Ariano (2009) propose the modelling of a railway system as a linear system in max-plus algebra with zero-order dynamics that represent delay propagation. Yang, Huang, Peng, LI, and Wen (2019) propose different distributions for eleven delays types ranging from bad weather to fault in tracks and utilized maximum likelihood estimation (MLE) and the Kolmogorov-Smirnov test (K-S) to evaluate each proposed distribution. Sorensen, Landmark, Olsson, and Seim (2017) developed an algorithm that analyses real-world data to identify delays and cascading delays defined according to various conditions, returning a network of dynamic delay propagation. Cule, Goethals, Tassenoy, and Verboven (2011) utilize the closed episode algorithm to mine cascading delays through a Belgian railway network focusing on specific reference points throughout the network. Wang and Work (2015) developed three regression models the first taking into account delay relationships between current and past trips among stations, factoring how delays of closely scheduled trains at various stations may influence the delay of a train in question, the second assuming that the delay of a train at a station is simply equal to the delay of the same train at the previous station and the third, assuming no interaction of delays amongst the interaction of different trains and that the delay of a train at a station is a results of a linear combination of delays of that same train at all of its upstream stations. Wang and Work (2015) yielded an RSME of 17 minutes for the first model, 8.4 minutes for the second and 7.7 minutes for the third. With recent advancements in the field of Machine Learning (ML), many have tried to create ML models to predict delays and understand its mechanism of propagation in the aviation industry, however, only a handful of projects utilize ML models in the same way in the railway industry. In his thesis, Bosscha (2016) proposes the adoption of recurrent neural networks (RNNs) alongside Irish Rail System data with labeled delay types to perform a one station step delay forecast, achieving a mean absolute percentage error (MAPE) of 7.22% for the "reactionary delays" type. Oneto et al. (2018) produced a train delay prediction system, forecasting the time taken for the train to reach its next checkpoint concerning is planned journey up to its termination station, achieving an average absolute difference between predicted and actual delay of 1.4 minutes for a one-step prediction and 3.4 minutes for a 10-step prediction. Wang and Zhang (2019) utilize weather records, historical train delays and train schedule to identify most important delay-inducing factors and utilize gradient-boosted regression trees to predict train delays along the

Beijing Guanzhou line.

The complexity of existing deterministic and stochastic models result in the inability for its utilization in networks other than the network in their respective case study. Whereas existing data-driven methods generally focus on small or single-line networks, hence unable to be utilized in real-world complex networks. This paper shall endeavour to strike a balance between the downfalls of existing methods and through a standard feed-forward deep neural network (DNN) and the extreme gradient boosting (XGBoost) framework, demonstrate the advantages of utilizing a tabular-based input data-structure to predict KPIs allowing for scalability, accuracy and insights in explaining the mechanisms of delay propagation.

2. Problem Formulation

Before delving into the methodology outlined by this paper, the problem formulation shall be explored. For train in question, k, its journey by certain checkpoints $C_i = \{C_0, C_1, C_2, ..., C_N\}$. Checkpoints are either the origin station (C_0), terminating station (C_N) or a stop (any stations in between the origin and destination). At checkpoint C_i, the scheduled arrival time of train k and the actual arrival time are TA_{C_i} and TA'_{C_i} respectively. The scheduled departure time and actual departure time are TD_{C_i} and TD'_{C_i} respectively. The scheduled and actual travel times from checkpoint C to the checkpoint C_{i+1} are $TT_{C_i,C_{i+1}}$ and $TT'_{C_i,C_{i+1}}$ respectively, the scheduled and actual dwell times (the amount of time a train is stationary at a particular train station) are DT_{C_i} and DT'_{C_i} respectively and the actual delay in arrival and departure times are A_{C_i} and D_{C_i} respectively whereby:

$$TT_{C_i,Ci+1} = TA_{C_{i+1}} - TD_{C_i} \text{ and } TT'_{C_i,C_{i+1}} = TA'_{C_i} - TD'_{C_i} \tag{1}$$

$$DT_{C_i} = TD_{C_i} - TA_{C_i} \text{ and } DT'_{C_i} = TD'_{C_i} - TA'_{C_i} \tag{2}$$

$$A_{C_i} = TA_{C_i} - TA'_{C_i} \text{ and } D_{C_i} = TD_{C_i} - TD'_{C_i} \tag{3}$$

At a micro level, the ML models in this paper shall forecast KPIs of all trains at any checkpoint within its journey as illustrated in Figure 1. In this example, Train 1 is at checkpoint C_D. All known information from its origin, C_A, up to its current checkpoint are utilized to predict KPIs at future checkpoint C_{i+1} using a one-step prediction model. Train 2 is at checkpoint C_D. All known information from train 2's origin are utilized to forecast KPIs at checkpoint C_{i+2} using a two-step prediction model. The ML models created in this paper are scalable up to N relative steps within a train's journey as illustrated in the case of train 3.

At a macro level, the overall framework this paper presents is a series of simultaneous regression models to predict the KPIs of a railway network. In this example, trains 1, 2 and 3, at time t, are undertaking their respective journeys from Checkpoints C_G to C_D, C_J to C_A and C_A to C_J. Through data pre-processing, each train journey and their current state in the network is separated individually whereby in this example,

Train 1 is currently at its origin station C_G and the ML models shall predict KPIs, Y, for Train 1's journey throughout checkpoints C_G, C_F, C_C, C_E and C_D. Train 2 is currently at its second checkpoint within its journey from C_J to C_A. Once train 2 departs, the ML models shall predict its KPIs throughout checkpoints C_G, C_F, C_C, C_B and C_A utilizing information amalgamated from its current and previous checkpoints C_H, C_J and C_I in doing so. Train 3 is dwelling at its second checkpoint in its journey. Once Train 3 leaves, the ML models shall predict its KPIs throughout checkpoints C_C, C_F, C_G, C_H, C_I and C_J utilizing information amalgamated from its current and previous checkpoints in doing so. Not only do these models utilize information known at a specific time, the data pre-processing stage also analyzes the network in question pulling out key network-specific information that shall be used alongside dynamic, train-journey-specific information as the input space x for the ML models. These processes shall be explored in greater detail in Section 2.3.

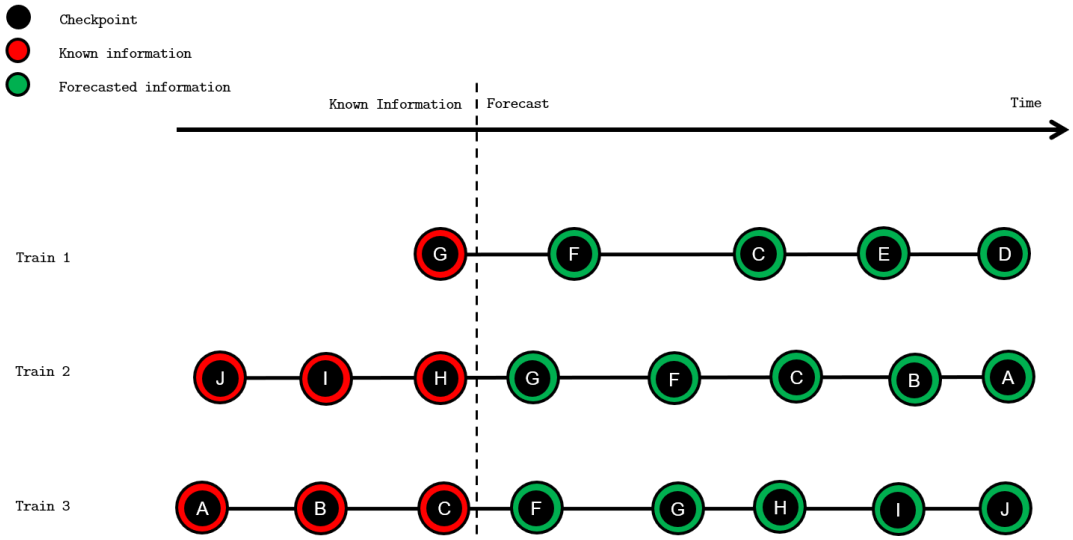


Figure 1.: Forecasting schematic

2.1. The Deep Neural Network Model

A DNN is a hierarchical model where each layer uses the output of the previous layer as its input and is based on unsupervised learning of multiple levels where higher level features are derived from the lower level features. Each layer in the neural network is made up of neurons. The input layer of the DNN shall represent the input matrix, the output layer representing the predictions. Hidden layers are those in between the input and output layer. Iterative gradient methods are used to train the neural network. Repeatedly, weights are assigned to each neuron to progressively make predictions on the subset of the data in a forward pass manner. Each neuron receives weighted input variables from the outputs of the neurons connected to it in the previous layer, and in response, a set of predetermined functions are performed on these values to produce an output to be sent the connecting neurons in the next layer. Upon the completion of the forward pass procedure, the loss function L is calculated and updated weighted values w proportional to dL/dw are formed. This process of forwards and backwards propagation is repeated until a stopping criterion is met.

2.2. The Extreme Gradient Boosting Model

The gradient boosting technique is an ensemble learning method whereby trees are sequentially built, aiming to reduce the errors of the previous tree. For a sample set of $\{(x_1, y_1), ..., (x_N, y_N)\}$ with input variables $\boldsymbol{X}$ and Y output variables, an initial model $F_0(\boldsymbol{X})$ is defined to predict the output variables, $\hat{y}_0$, by minimizing the differentiable convex loss function $l(y, F_0(\boldsymbol{X}))$. The gradient of the loss function, r_0 is then computed and fitted with a new regression tree h_1. The first boosted model, $F_1(\boldsymbol{X})$, is then computed through a combination of weighted regression trees and the previous model. This process is repeated for M iterations until a stopping condition has been reached. The Extreme Gradient boosting framework (XGBoost) is a supervised ensemble learning method based on ensemble trees and follows the gradient boosting principles however utilizes a more regularized model to minimize over-fitting. Having all trees K, the prediction, $\hat{y}_i$ can be represented by the following function:

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} h_k(x_i),\ h_k \in H \tag{4}$$

or similarly, the prediction at the m^{th} step is defined as:

$$\hat{y}_{im} = \phi(x_i)_m = \sum_{k=1}^{m} h_k(x_i),\ h_k \in H \tag{5}$$

where, $\hat{y}_i$ is the predicted values, K is the number of trees, H is all possible regression trees and h_k is the specific regression tree that is learned through minimizing the following cost objective function at step m:

$$L(\theta)_m = \sum_{i=1}^{N} l(y_i, \hat{y}_{im}) + \sum_{i=1}^{m} \Omega(h_i) = \sum_{i=1}^{N} \left[l(y_i, \hat{y}_{i(m-1)}) + h_m(x_i) \right] + \sum_{i=1}^{m} \Omega(h_i) \tag{6}$$

Through utilizing the Taylor expansion, the cost objective function can be approximated to be:

$$L(\theta)_m = \sum_{i=1}^{N} \left[l(y_i, \hat{y}_{i(m-1)}) + g_i h_m(x_i) + \frac{1}{2} h_i h_m^2(x_i) \right] + \sum_{i=1}^{m} \Omega(h_i) \tag{7}$$

where

$$g_i = \partial \hat{y}_{i(m-1)} l(y_i, \hat{y}_{i(m-1)}) \tag{8}$$

$$h_i = \partial^2 \hat{y}_{i(m-1)} l(y_i, \hat{y}_{i(m-1)}) \tag{9}$$

$\Omega(h_i)$ is the regularization term in the form of a penalty that increases with the complexity of the ensemble H in order to avoid over-fitting. For the XGBoost algorithm, the penalty relates to the number of leaves per tree T, regression tree score w_j at each leaf and hyperparameters γ and λ as portrayed below:

$$\Omega(h) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{10}$$

At each step m, the algorithm greedily adds the hypothesis h_m that best minimizes that objective.

2.3. The Data

The data utilized in this paper is provided through Darwin, Great Britain's official railway information engine (Bleakley, Akinola, & Fullard, 2019). Specifically, the application programming interface (API) utilized is the historical service performance (HSP) API (*Darwin Data Feeds*, 2019). This API provides two data sets in Javascript Object Notation (JSON) Format that shall be used in conjunction with each other, Service Metrics —- requiring the origin and destination stations, first departure and final arrival times and start and end dates to be defined as inputs and Service Details —- requiring train IDs provided by the Service metrics API as input. A detailed list of metrics provided by the two APIs are provided in the Tables 1 and 2 below.

Table 1.: Service metrics provided by Darwin's HSP

Key	Description
Origin Location	Computer Reservation System (CRS) Code of Origin
Destination Location	CRS code of destination
gbtt ptd	Public departure time at departure station
gbtt pta	Public arrival time at destination station
TOC code	Code of train operating company
RIDs	Train ID
Matched services	List of all train RIDs
Tolerance Value	Selected tolerance value
Num not tolerance	Number of trains outside the tolerance
Num tolerance	Number of trains within the tolerance

Table 2.: Service details provided by Darwin's HSP

Key	Description
Date of service	Date of service of the specified train RID
TOC Code	Code of train operating company
RID	Inputted RID
Location	CRS code of train location
gbtt ptd	Public departure time
gbtt pta	Public arrival time
Actual td	Actual departure time
Actual ta	Actual arrival time
Late canc reason	Code that specifies late or cancellation reason

All time values provided by Darwin's service details API are accurate to the nearest minute and include all origin-destination trips that pass through the allocated origin and destination stations. For this paper, Didcot Parkway and London Paddington were selected as the terminating stations respectively i.e. all train journeys that include both these stations in their schedule in both directions are to be used to train and test the models. Journeys between these stations are the choice of this paper due to its notoriety in providing prevalent delayed services (Department of Transport, 2016). All train journeys between 06:00 am and 10:00 am from 2016 shall be used in training the ML models while all train journeys between 06:00 am and 10:00 am in 2017 data shall be used to test the ML models. This time range was selected to capture the mechanics of delay propagation amidst morning rush hour and hence, peak usage. The days used to train this model have been specified as non-holiday weekdays. The day, month, order of train departure, origin departure period and current train checkpoint adds temporal dimensions to the model whereas all other variables add spatial dimensions to the model. Initial data provided by Darwin include 10,767 journeys in 2016 and 10,742 journeys in 2017 initiating at various stations, passing through Didcot Parkway and terminating at London Paddington. 9,069 journeys in 2016 and 8,969 journeys in 2017 initiating at London Paddington and passing through Didcot Parkway to their respective destination stations. Figure 2 portrays all the checkpoints considered in this paper.

2.4. Data Preprocessing

The dataset provided by Darwin's HSP API underwent preprocessing and feature engineering in order to produce an input space that best outlines the context of the problem per Section 2, therefore ensuring the best performance of the XGBoost and DNN models outlined in Sections 2.2 and 2.1. The data that shall be used in the models of this paper is processed by looping through Darwin's Service Metrics and Service Details API to achieve the spatial and temporal constraints outlined in section 2.3. Missing information was of primary concern and was dealt with before any features were engineered. All missing information in this paper originate from the 'actual_td' and the 'actual_ta' categories of Darwin's Service Details API, that is the actual departure time and arrival time of a train at a station respectively due to the nature of data collection, in this case through sensors. In order to account for this missing data issue we apply the practice to dismiss any datasets that have more than 10% missing

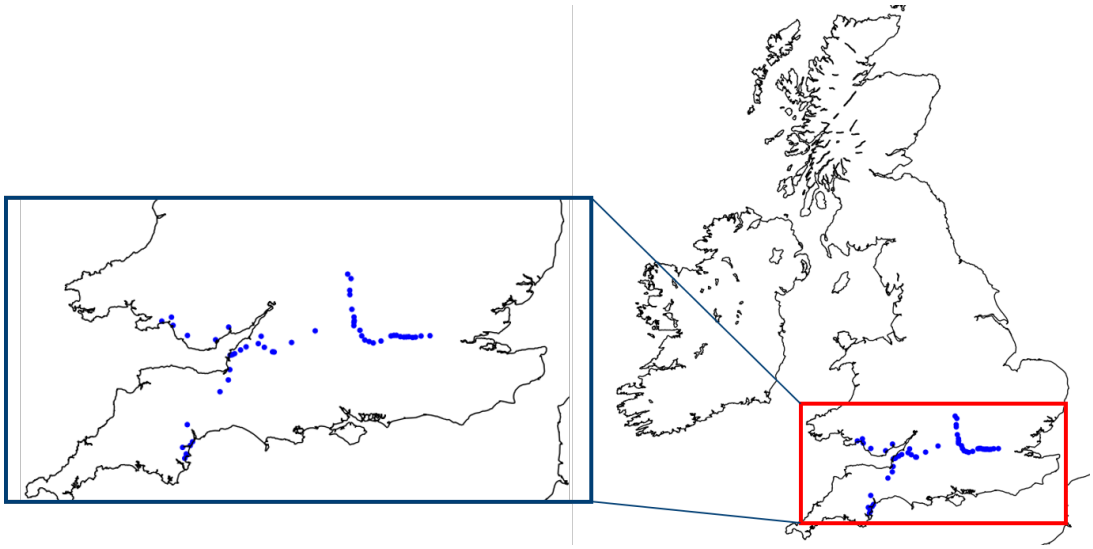


Figure 2.: Figure illustrating all checkpoints considered in this paper

data in relation to a particular data hierarchy. For example, when daily cyclicality is being considered, daily datasets with greater than 10% missing data should be terminated. When applying this logic to this paper, if for any journey, the amount of missing data is greater than 10%, the whole journey shall be discarded. When there is less than 10% missing data, sample imputation shall be used. The average actual and average planned travel times between two unique station pairs and the average dwell time of each unique station are evaluated and utilized to fill in the missing arrival and departure times as shown below in the pseudo-codes outlined in Tables A1 and A2 respectively. Following the missing data imputation process, feature engineering is performed to produce the input space, $\boldsymbol{X} \in \mathbb{R}^{n \times p}$, where n is the number of data points and p is the number of input features for the models in this paper. These features and the processes undertaken to produce them are outlined in Section 2.6.

2.5. Hyper-Parameter Optimization

This section shall explain and justify the architectural configuration of the DNN and XGBoost models used in this paper. Any learning algorithm, A, is made of a set of hyper-parameters, $\boldsymbol{\lambda}$s, the purpose in fine-tuning $\boldsymbol{\lambda}$s is to produce an optimal A that minimizes the expected loss of the model L. Although currently, there is no superior global algorithm to find the optimal set of hyper-parameters, several techniques exist such as the grid searching cross-validation, randomized search cross-validation and genetic algorithms. This paper shall utilize the the randomized search cross-validation technique due to the algorithm's great efficiency in exploring a wide range of $\boldsymbol{\lambda}$s whilst minimizing processing time (Bergstra & Bengio, 2012).

2.5.1. DNN

A two-step process is implemented in fine-tuning the DNN's $\boldsymbol{\lambda}$s; Initially, a coarse range of $\boldsymbol{\lambda}$s are selected through trial and error as well as convention, in the case of the number of hidden layer neurons, values encompassing the mean of the number of input neurons and the output neurons were chosen for further optimization. The DNN

is then optimized through 20 random $\boldsymbol{\lambda}$ combinations to retrieve a generalized optimal model as detailed in 3. After which, a second set of finer $\boldsymbol{\lambda}$ ranges were optimized based on the previous optimal model through 25 random combinations. This is detailed in Table 4, whereby all values that are not refered to in Table 4 are finalized in Table 3.

Table 3.: Summary of initial DNN hyper-parameter fine-tuning

Variable	Values(s)	Optimum	Description
Batch Size	50, 100	50	The number of examples that are utilized in each iteration before the neural network's parameters are updated. A large batch size reduces the training time; however, this may cause accuracy loss (Smith, Kindermans, & Le, 2017).
Epoch	50, 100, 150	100	The number of times the algorithm completes forwards and backwards pass of all training examples. A larger epoch number risks over-fitting but a smaller epoch number risk lower accuracy.
Drop-out	0, 0.2	0.2	The drop-out algorithm is used to combat over-fitting. A value too small may still induce over-fitting whereas a value too large may hinder the model's learning process.
Number of hidden layer neurons	10, 18, 25	10	A value too small fails to convey the information of the input neurons whereas a value too large will increase processing times.
Number of hidden layers	1, 2, 3	3	A value too small fails to convey the information of the input neurons whereas a value too large will increase processing times.
Optimization Algorithm	Adam	-	The optimization algorithm was not varied since it does not have a direct effect in improving the model accuracy as much as how fast the model takes to converge. It has been empirically proven that the optimization algorithm 'Adam' outperforms other algorithms in a neural network (Kingma & Ba, 2014).
Network Weight Initialization	Random Normal	-	The weight initializer is set as random normal (mean of 0 and standard deviation of 0.5) since the input parameters of the model are normalized and scaled relative to each other theoretically meaning faster convergence with this method.
Neuron Activation Function	ReLU	-	This activation function is used as convergence is reached significantly faster than other activation functions (Krizhevsky, Sutskever, & Hinton, 2012)

Table 4.: Summary of the second iteration of DNN hyper-parameter fine-tuning

Variable	Value(s)	Optimum	Description
Batch Size	32, 50, 64	64	-
Neurons	22, 24, 26, 28	28	Although, in the first iteration of the optimization process, the optimal number of neurons in the hidden layer was shown to be 10, once tested with the 2017 data, the performance was worse than the initial model, after some trial and error, it was clear that for this specific problem, hidden layer neuron numbers between 20 and 30 excelled in performance. Hence, this particular range was chosen.
Drop-out rate	0	-	Once again, through trial and error, contrary to the optimal drop-out rate from the initial optimisation process of 0.2, a drop-out rate of 0 actually produced more accurate results.

2.5.2. XGBoost

For the XGBoost model, we applied fine-tuning to the maximum depth of tree and the minimum sum of instance weight needed in a child (minimum child weight), regarded as the two most important hyperparameters dictating model performance as well as the learning rate and the number of trees. The details regarding these parameters are in Table 5.

Table 5.: Table detailing the XGBoost model's variables fine-tuned for the one-step prediction

Variable	Value(s)	Optimum	Description
Learning Rate	0.01, 0.1	0.1	Generally, the learning rate is best kept as 0.1, however, if the learning rate were to be increased, then the number of trees would be reduced and vice versa.
Number of trees	100, 200, 300	300	The default number of trees provided by the XGBoost package is set as 100.
Maximum depth of tree	3, 5, 7, 9	5	By increasing this value, the model's complexity will increase however, over-fitting must be considered.
Minimum sum of instance weight needed in a child	1, 3, 5	1	A larger value results in a more conservative model.

2.6. Feature Engineering

All historical data provided by Darwin's HSP underwent an initial feature engineering process to extract the features summarized in Table 6.

Table 6.: Feature engineering summary

Feature	Input Parameter	Feature Summary
1	Cumulative arrival delay of train journey up to C_i	$\sum_{i=0}^{i} A_{C_i}$
2	Cumulative departure delay of the train journey up to C_i	$\sum_{i=0}^{i} D_{C_i}$
3	Day	Provided by HSP
4	Arrival delay, A_{C_i}, of the train at the current checkpoint	$TA_{C_i} - TA'_{C_i}$
5	Departure delay, D_{C_i}, of the train at the current checkpoint	$TD_{C_i} - TD'_{C_i}$
6	Direction of travel	
7	Dwell time, DT, of the train at the current checkpoint	$TD_{C_i} - TA_{C_i}$
8	Average dwell time of all trains at station C_{i+n}	Deduced from unique station dwell time
9	Whether or not the current station is the origin station	Self explanatory
10	Whether or not the current station is the terminating station	Self explanatory
11	Month	Provided by HSP
12	Number of stops	Provided by HSP
13	Order of departure from the origin station of the train on the day of its journey relative to other trains	All individual journeys in a day are ranked
14	What stop the train is at with respect to its journey	Self explanatory
15	Origin departure period	Split up into four categories, if the train departed from its original station from 06:00 and 07:00, then $\chi_{15} = 0$, if its departure is between 07:00, $\chi_{15} = 2$ and finally, 09:00 and 10:00, $\chi_{15} = 3$
16	Average travel time from the $C_{\text{Prediction}-1}$ to $C_{\text{Prediction}}$	Deduced from average travel time between unique OD pairs
17	Travel time from the previous checkpoint to the current checkpoint TT_{C_{i-1},C_i}	$TA_{C_i} - TD_{C_{i-1}}$

In order to make the engineered data compatible with the ML models, we processed the data further. We flattened all train journeys as provided by Darwin, and following the feature engineering process, results seen in Table B1, we encoded nominal categorical parameters, specifically features 3 - day, 6 - direction of travel, 9 - whether or not the current station is the origin station, 10 - whether or not the current station is the terminating station and 11 - month, this is represented in Table B2. As is the nature of nominal categorical data, their representation must not convey ordinality, therefore, we created dummy variables and omitted one dummy variable as to avoid perfect collinearity or the "dummy variable trap" in the prediction process, this can be seen in Table B3, where dummy variables were created for the day and month input features for the sample portrayed in tables B1 and B2. Finally, we standardized each feature x_i independently through centering and scoring using the mean μ and variance σ as shown in Equation 11, this is shown in Table B4. Table 7 shall hereafter, be used as reference for the input variable. Note that the input space for the models are derived from actual information upon the departure of train k from station C_i i.e. all data engineered for the input space for the ML models correspond to data from accumulated for a train k up and prior to the train departing from station C_i.

$$Z = \frac{x_i - \mu}{\sigma} \tag{11}$$

Table 7.: Final input variable matrix

Feature No.	Feature	Feature No.	Feature
1	Monday	17	December
2	Tuesday	18	Cumulative Arrival Delay
3	Wednesday	19	Cumulative Departure Delay
4	Thursday	20	Arrival Delay at Current Checkpoint
5	Friday	21	Departure Delay at Current Checkpoint
6	January	22	Direction of Travel
7	February	23	Dwell Time at Current Checkpoint
8	March	24	Average Dwell Time at Predicted Station
9	April	25	Origin Station
10	May	26	Terminal Station
11	June	27	Number of Stops
12	July	28	Scheduled Order of Departure Of the Day
13	August	29	Current Checkpoint of the Train Journey
14	September	30	Origin Departure Period
15	October	31	Average Travel Time from the Checkpoint before the predicted to the predicted Checkpoint
16	November	32	Travel Time from previous check point to Current Checkpoint

The actual output space Y^A and forecasted output space Y^F for each journey of each train k at each checkpoint C_i, composes of, actual and forecasted time travelled from the station C_{i+n-1} to station C_{i+n}, i.e. $TT_{C_{i+n-1},C_{i+n}}$ and $\hat{TT}_{C_{i+n},C_{i+n-1}}$ respectively, actual and forecasted dwell time at station C_{i+n}, i.e. $DT_{C_{i+n}}$ and $\hat{DT}_{C_{i+n}}$ respectively, actual and forecasted arrival delay at station C_{i+n}, i.e. AC_{i+n} and $\hat{A}C_{i+n}$ respectively and actual and forecasted departure delay at station C_{i+n}, i.e. $D_{C_{i+n}}$ and $\hat{D}_{C_{i+n}}$. In this

paper, KPIs of up to ten future checkpoints relative to the train's current checkpoint shall be predicted, i.e. $n \in \{1, 2, ..., 10\}$. Hence, ten models shall be trained through the DNN and XGBoost frameworks each. During the training stage, if the future prediction step n is greater than the number of checkpoints left within the train's particular journey the output space is set to be 0. If the number of checkpoints left within the train's particular journey is equal to the number of remaining checkpoints, only the dwell time and departure departure delay predictions are set to be 0.

3. Results and Evaluation

To ensure an unbiased evaluation of the predictive capabilities of the DNN and XGBoost models, careful consideration is required in selecting performance metrics. Due to the nature of the predictions made by the models of this paper encompassing near-zero values, we decided that the standard performance metric, the mean absolute percentage error (MAPE) or the symmetrical mean and absolute percentage error (SMAPE) shall not be utilized to evaluate performance due to these metrics' setbacks in dealing with near zero values. Instead, we propose utilizing the mean arctangent absolute percentage error term (MAAPE) developed by Kim and Kim (2016) that allows for improved symmetry compared to MAPE, but most importantly, the ability to assess near-zero values that render MAPE and SMAPE to infinity. We shall also utilize the root mean square error (RMSE) and the coefficient of determination, r^2 and the average absolute error for each variable of the output space $\boldsymbol{Y}$, ATTAE (average travel time absolute error), AADAE (average arrival delay absolute error), ADDAE (average departure delay absolute error) and ADTAE (average dwell time absolute error) to evaluate our models' performances.These metrics are illustrated in Equations 12 to 18. It should be noted that whilst the models were trained using all data available, we shall omit forecasts of all trains that do not experience a delay in departure from its most recent checkpoint from the evaluation. This is because the aim of this research is to produce models that predict train KPIs following a delay and hence to understand the mechanism of cascading delays.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Y_i^F - Y_i^A)^2}{n}} \tag{12}$$

$$r^2 = \frac{n(\sum_{i=1}^{n} Y_i^F Y_i^A) - (\sum_{i=1}^{n} A_i^F)(\sum_{i=1}^{n} Y_i^A)}{\sqrt{n\sum_{i=1}^{n}(Y_i^F)^2 - (\sum_{i=1}^{n} Y_i^F)^2}} \tag{13}$$

$$MAAPE = \frac{\sum_{i=1}^{n} \arctan\left(\left|\left|\frac{Y_i^A - Y_i^F}{Y_i^A}\right|\right|\right)}{n} \tag{14}$$

$$TTAE = ||\hat{TT}_{C_{i+n},C_{i+n-1}} - TT_{C_{i+n},C_{i+n-1}}|| \qquad ATTAE = \frac{\sum_{i=1}^{n} TTAE_i}{n} \tag{15}$$

$$ADAE = ||\hat{A}_{C_{i+n}} - A_{C_{i+n}}|| \qquad AADAE = \frac{\sum_{i=1}^{n} ADAE_i}{n} \tag{16}$$

$$DDAE = ||\hat{D}_{C_{i+n}} - D_{C_{i+n}}|| \qquad ADDAE = \frac{\sum_{i=1}^{n} DDAE_i}{n} \tag{17}$$

$$DTAE = ||\hat{DT}_{C_{i+n}} - DT_{C_{i+n}}|| \qquad ADTAE = \frac{\sum_{i=1}^{n} DTAE_i}{n} \tag{18}$$

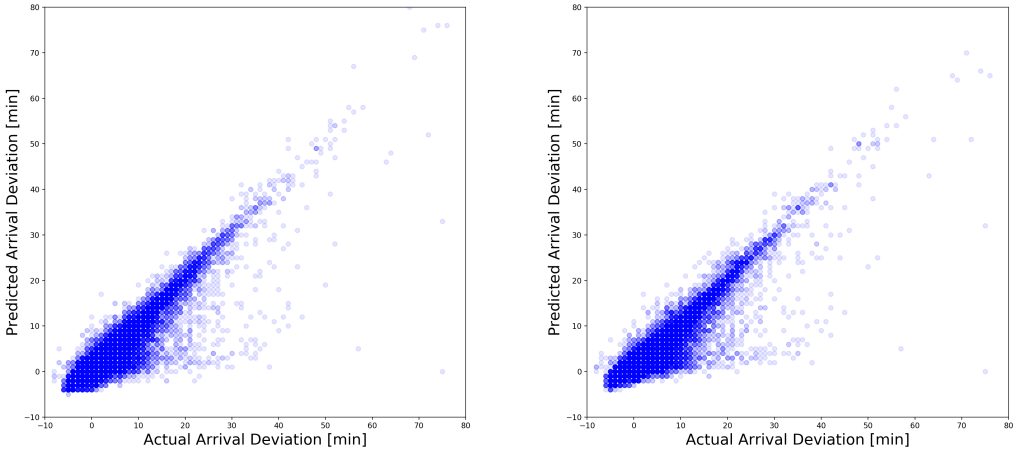
3.1. Model Interpretability

In order to introduce understanding with regards to the propagation of reactionary delays, we propose exploring recently developed tools that seek to explain the process at which ML models come to their predictions. Many popular tools such as the Local Interpretable Model-agnostic Explanations (LIME) algorithm (Ribeiro, Singh, & Guestrin, 2016) or XGBoost feature importance functionality attempt to eradicate the "black-box" connotation with regards to ML models by providing local and global explanation to features and how they contribute toward the ML model's final prediction. We however, decided to utilize the SHAP framework (S. Lundberg & Lee, 2017) to interpret our ML models due to superior performance and interpretabilty. As described by S. M. Lundberg et al. (2019), three desirable properties of attribution methods should consist of local accuracy, consistency, and missingness. Regarding accuracy, for a model f with a set of specific inputs $\boldsymbol{x}$, the sum of all attributed explanation values should be equal to the model output $f(\boldsymbol{x})$. Consistency disallows the explanation attribution of a feature to decrease if the model has been altered to increase the importance of that feature regardless of other features. Missingness states that features that do not have an impact on the set function $f(\boldsymbol{x})$ shall have no assigned attributed explanation value. A solution proposed by S. M. Lundberg et al. (2019) that follows from cooperative game theory, adapts Shapley values (Shapley, 1953) to assign a Shapley Additive Explanation (SHAP) value to each individual feature as well as satisfying the desirable properties mentioned above. The theorem in which this is achieved is as follows:

$$\phi_i(f, x) = \sum_{R \in \mathbb{R}} \frac{1}{M!} \left[f_x(P_i^R \cup i) - f_x(P_i^R) \right] \tag{19}$$

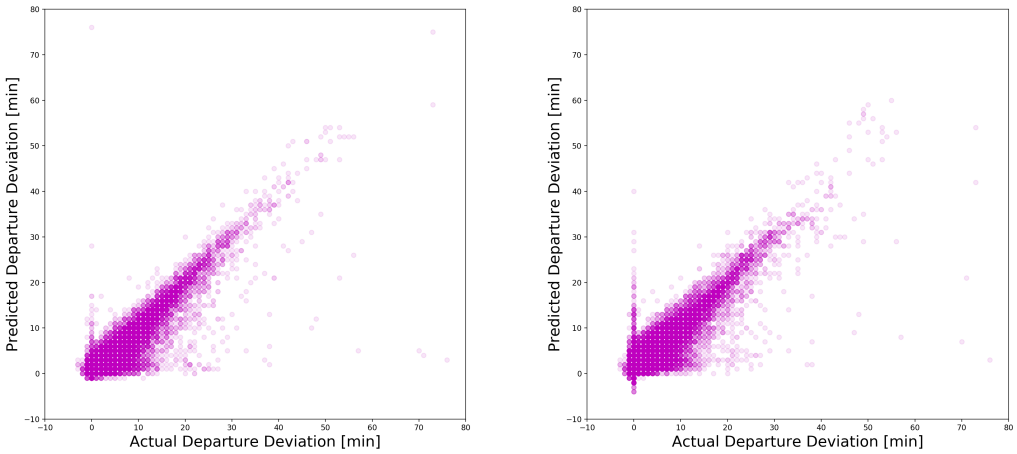
Whereby, ϕ_i is the attribute explanation value of input feature i, M is the number of input features in the model, $\mathbb{R}$ is the set of all input feature orderings, P_i^R is the set of all input features introduced before feature i in the ordering R. The specific SHAP methods utilized in this paper are the Tree Explainer method (S. M. Lundberg et al., 2019) that shall interpret the XGBoost model and the Deep SHAP method (S. Lundberg & Lee, 2017) that shall interpret the DNN model; all visualizations presented as a result, have been developed by S. M. Lundberg, Erion, and Lee (2018).

3.2. Model Results



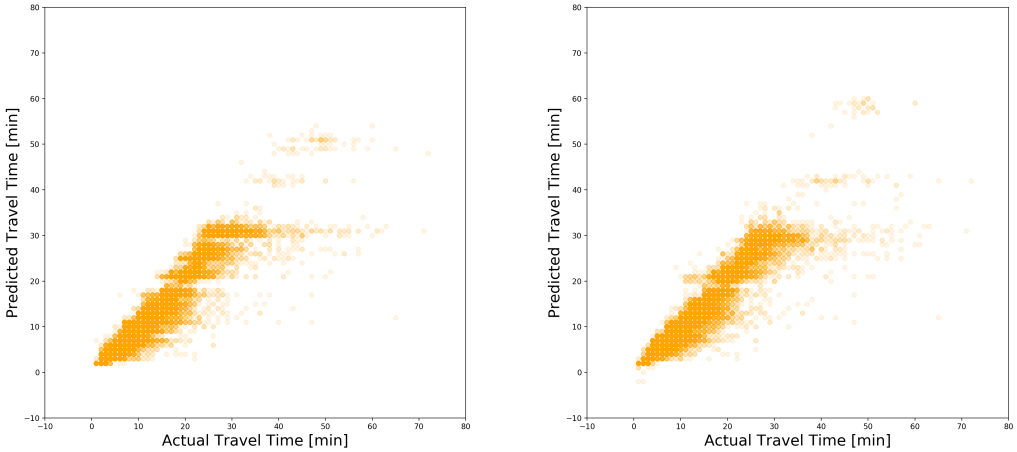
(a) DNN (b) XGBoost

Figure 3.: 1-step predicted vs actual deviation from arrival



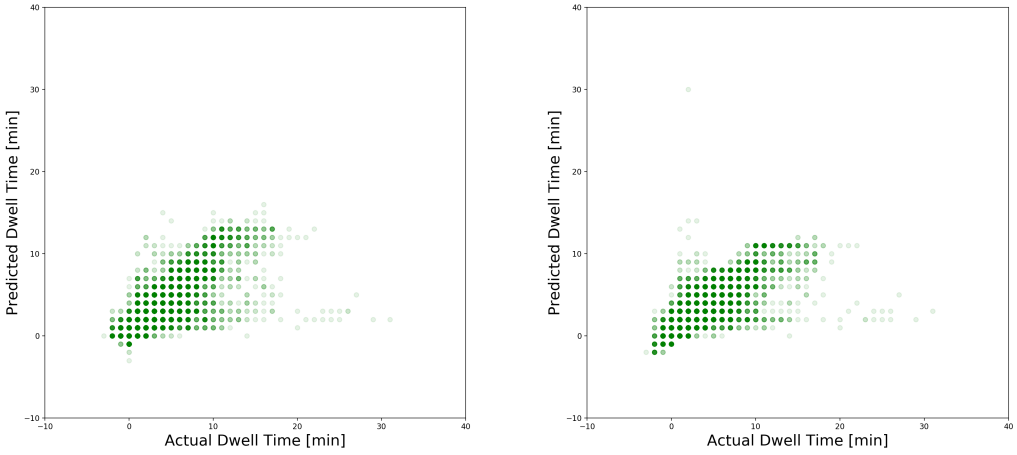
(a) DNN (b) XGBoost

Figure 4.: 1-step predicted vs actual deviation from departure



(a) DNN (b) XGBoost

Figure 5.: 1-step predicted vs actual travel time



(a) DNN (b) XGBoost

Figure 6.: 1-step predicted vs actual dwell time

Table 8.: Model performance for arrival deviation forecasting

	DNN				XGBoost			
Step	RMSE	r^2	AADAE	MAAPE	RMSE	r^2	AADAE	MAAPE
1	1.97	0.84	1.06	1.12	1.96	0.84	1.04	1.08
2	2.47	0.75	1.38	1.14	2.48	0.74	1.37	1.13
3	2.65	0.69	1.42	1.19	2.63	0.69	1.40	1.17
4	2.78	0.62	1.50	1.22	2.74	0.64	1.44	1.22
5	2.79	0.58	1.37	1.29	2.69	0.61	1.35	1.26
6	2.85	0.50	1.41	1.31	2.64	0.57	1.26	1.30
7	2.74	0.49	1.24	1.34	2.62	0.53	1.21	1.33
8	3.07	0.28	1.30	1.36	2.58	0.49	1.14	1.35
9	2.79	0.34	1.12	1.38	2.56	0.45	1.06	1.38
10	2.69	0.34	1.03	1.40	2.48	0.44	0.98	1.40

Table 9.: Model performance for departure deviation forecasting

	DNN				XGBoost			
Step	RMSE	r^2	ADDAE	MAAPE	RMSE	r^2	ADDAE	MAAPE
1	1.80	0.85	0.92	1.11	2.02	0.80	1.10	1.11
2	2.20	0.76	1.27	1.18	2.19	0.76	1.19	1.18
3	2.51	0.67	1.29	1.25	2.39	0.70	1.28	1.24
4	2.43	0.66	1.29	1.29	2.41	0.67	1.26	1.29
5	2.44	0.62	1.19	1.33	2.39	0.64	1.19	1.33
6	2.57	0.53	1.22	1.35	2.31	0.62	1.09	1.35
7	2.46	0.52	1.07	1.37	2.32	0.58	1.06	1.38
8	2.81	0.30	1.12	1.40	2.44	0.47	1.08	1.40
9	2.55	0.35	1.00	1.42	2.26	0.49	0.93	1.42
10	2.30	0.40	0.83	1.44	2.09	0.51	0.82	1.44

Table 10.: Model performance for travel time forecasting

	DNN				XGBoost			
Step	RMSE	r^2	ATTAE	MAAPE	RMSE	r^2	ATTAE	MAAPE
1	1.81	0.92	0.89	1.30	1.79	0.92	0.79	1.31
2	1.77	0.93	0.86	1.33	1.74	0.93	0.72	1.33
3	1.57	0.94	0.68	1.36	1.56	0.94	0.62	1.36
4	1.48	0.94	0.62	1.38	1.47	0.94	0.55	1.38
5	1.35	0.95	0.51	1.40	1.35	0.95	0.46	1.40
6	1.24	0.94	0.42	1.42	1.20	0.95	0.37	1.42
7	1.16	0.95	0.37	1.43	1.14	0.95	0.33	1.43
8	1.12	0.95	0.34	1.45	1.11	0.95	0.32	1.45
9	1.07	0.94	0.31	1.45	1.04	0.95	0.28	1.46
10	1.14	0.94	0.30	1.47	1.15	0.94	0.28	1.47

Table 11.: Model performance for dwell time forecasting

	DNN				XGBoost			
Step	RMSE	r^2	ADTAE	MAAPE	RMSE	r^2	ADTAE	MAAPE
1	1.09	0.64	0.55	0.82	1.07	0.65	0.51	0.81
2	1.18	0.58	0.62	0.81	1.13	0.61	0.55	0.83
3	1.06	0.65	0.52	0.91	1.06	0.65	0.48	0.89
4	0.99	0.68	0.45	0.96	1.03	0.65	0.43	0.98
5	0.96	0.69	0.40	1.03	0.97	0.68	0.40	1.03
6	0.99	0.66	0.37	1.09	0.94	0.70	0.37	1.10
7	0.86	0.74	0.32	1.14	0.91	0.71	0.34	1.15
8	0.84	0.74	0.30	1.20	0.80	0.77	0.29	1.19
9	0.81	0.74	0.29	1.22	0.72	0.80	0.26	1.23
10	0.86	0.70	0.26	1.26	0.78	0.75	0.23	1.25

3.3. Performance Compared to Darwin

From Darwin's push port archive (Mount, 2017), we filtered out Darwin's forecasts from February to May of 2017 to match the train journeys in our case study and associated these forecasts with forecasts made by our DNN and XGBoost models. Unlike our model, Darwin did not forecast all train journeys within this time period, hence the number of analyzed journeys are significantly lower that that of section 3.2. Furthermore, Darwin only forecasts arrival times and departure times hence, only arrival and departure deviations will be computed from Darwin's forecasts and compared to that predicted by the DNN and XGBoost models.

Table 12.: Model performance for arrival deviation forecasting vs Darwin

	DNN				XGBoost				Darwin			
Step	RMSE	r^2	AADAE	MAAPE	RMSE	r^2	AADAE	MAAPE	RSME	r^2	AADAE	MAAPE
1	2.24	0.91	1.10	1.09	2.15	0.92	1.13	1.05	5.03	0.56	2.59	1.19
2	2.08	0.94	1.40	1.10	1.92	0.95	1.09	1.09	5.78	0.57	2.97	1.22
3	2.42	0.91	1.50	1.13	2.11	0.93	1.37	1.12	5.90	0.48	2.93	1.23
4	2.25	0.90	1.57	1.12	1.98	0.92	1.25	1.12	5.63	0.37	3.21	1.22
5	2.58	0.89	1.80	1.16	2.96	0.86	1.75	1.09	6.35	0.36	3.67	1.22
6	3.92	0.69	2.13	1.09	3.04	0.81	1.84	1.07	5.50	0.38	3.14	1.21
7	3.17	0.80	1.95	1.08	2.98	0.82	1.79	1.08	6.01	0.27	3.48	1.23
8	4.22	0.63	2.24	1.07	2.53	0.87	1.59	1.07	5.73	0.32	3.18	1.20
9	3.30	0.54	1.76	1.03	2.47	0.74	1.41	1.00	4.11	0.29	2.06	1.15
10	4.08	0.34	2.04	1.02	2.99	0.64	1.33	1.08	4.83	0.07	2.32	1.20

Table 13.: Model performance for departure deviation forecasting vs Darwin

	DNN				XGBoost				Darwin			
Step	RMSE	r^2	AADAE	MAAPE	RMSE	r^2	AADAE	MAAPE	RSME	r^2	AADAE	MAAPE
1	2.02	0.93	1.02	0.91	2.07	0.93	1.17	0.90	5.77	0.42	3.41	1.11
2	5.34	0.64	1.90	1.03	5.41	0.63	1.61	1.02	6.27	0.50	3.42	1.17
3	7.75	0.06	2.77	1.02	6.50	0.34	2.22	1.01	6.40	0.36	3.63	1.19
4	2.11	0.91	1.30	1.09	2.42	0.88	1.37	1.08	5.89	0.31	3.42	1.24
5	3.83	0.77	2.28	1.11	3.79	0.77	2.21	1.12	6.98	0.23	4.43	1.24
6	3.82	0.67	2.06	1.10	3.38	0.74	1.79	1.10	5.94	0.19	3.66	1.25
7	3.24	0.79	1.96	1.13	3.56	0.75	1.78	1.13	6.53	0.14	4.11	1.25
8	4.58	0.55	2.36	1.18	3.24	0.78	1.84	1.15	6.07	0.22	3.33	1.28
9	4.50	0.16	2.34	1.09	3.30	0.55	1.78	1.11	4.37	0.21	2.20	1.22
10	4.54	0.12	2.27	1.06	3.19	0.57	1.70	1.02	5.10	-0.11	2.88	1.17

3.4. Model Evaluation

Figures 3 to 6 suggest very accurate 1-step predictions for both the DNN and XGBoost models for all four KPIs, with deteriorating performance with an increasing number of steps as portrayed in the Figures of Appendix C. With lower RMSE, absolute error, MAAPE and greater r^2 metric values for most step forecasts for all four KPIs shown in tables 8 to 11, in general, the XGBoost model outperforms the DNN model. Not only is the XGBoost model generally more accurate, it is more computationally efficient during the training stage and the Tree Explainer method fitted to the XGBoost model also proves to be less time consuming compared to the DNN's Deep SHAP method. Although the RMSE and absolute error terms for the arrival and departure deviation

predictions for both models generally show stable performance as the forecast steps increase, the r^2 and MAAPE metrics suggest worsened performance which is to be expected. However, both models' travel and dwell time prediction performance remain stable as the number of forecast steps increase. This supports the fact that these respective KPIs are generally fixed in a railway network. Tables 12 and 13 clearly that our models greatly outperform Darwin. Specifically, the XGBoost model outperforms Darwin in every metric at every step forecast and the DNN model doing so in most cases. As it is clear that the XGBoost model outperforms the DNN model and Darwin, we shall evaluate XGBoost's most accurate prediction's SHAP evaluations (i.e. 1-step forecast for all KPIs).

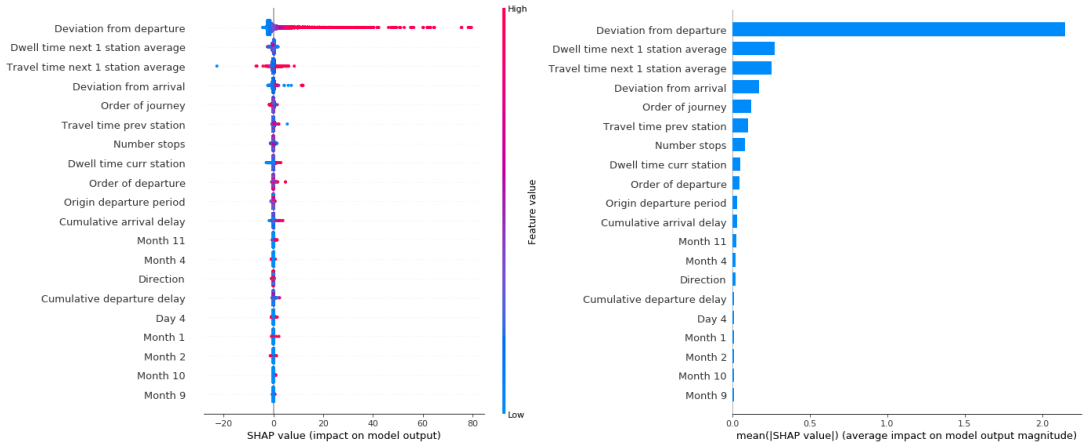


Figure 7.: 1-step XGBoost deviation from arrival SHAP evaluation

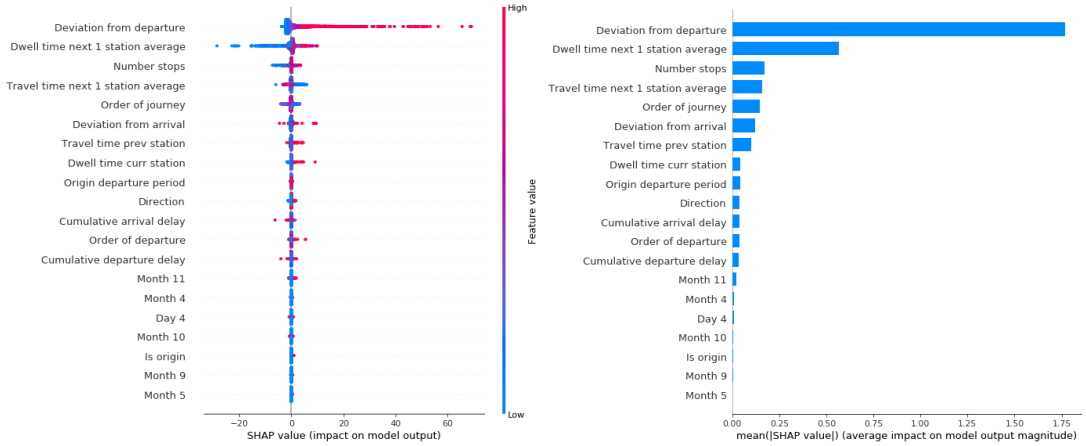


Figure 8.: 1-step XGBoost deviation from departure SHAP evaluation

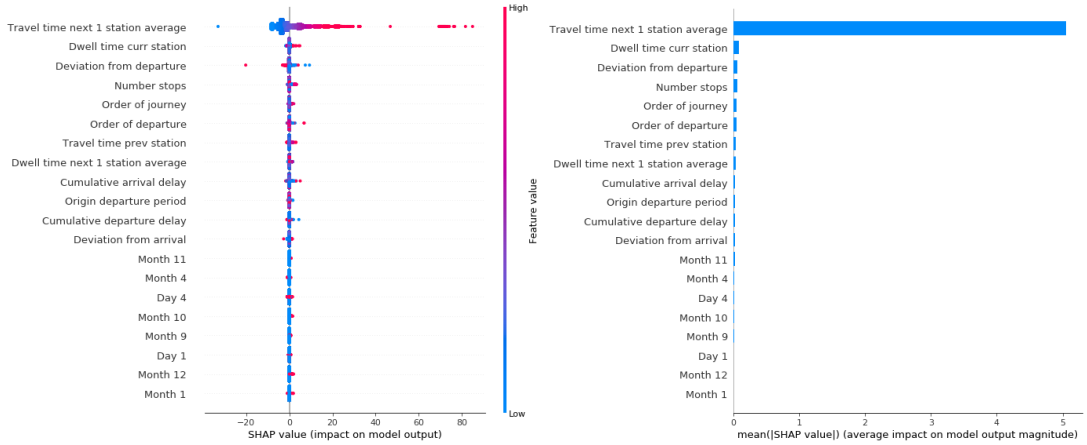


Figure 9.: 1-step XGBoost travel time SHAP evaluation

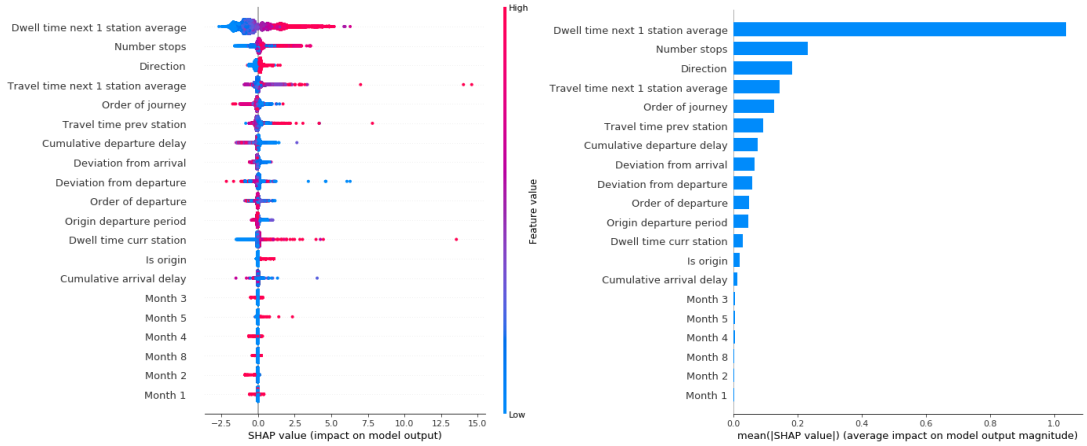


Figure 10.: 1-step XGBoost dwell time SHAP evaluation

As expected, for the arrival and departure deviation predictions, the most important features influencing the predictions are the deviation from departure of the trains previous station. Additionally, the 1-step deviation from departure predictions are highly dependent upon both the deviation from departure from the train's previous station and the average dwell time of the train 1-step station ahead which is intuitive as the dwell time of a train at any station should be fixed and therefore, the deviation from departure time is simply the dwell time in addition to the deviation from arrival. The most important features in the 1-step travel and dwell time predictions are the average 1-step travel time from checkpoint C_i to C_{i+1} and average dwell time at checkpoint C_{i+1} respectively. This is testament to the fixed nature of these KPIs. It should be noted that the SHAP framework's local predictive capabilities explained in section 3.1 are critical in formulating hypothesis for further testing. For example, as a result of the travel time SHAP evaluations, testing whether or not the average travel time from checkpoint C_{i+n-1} to C_{i+n} is sufficient on its own in the travel time prediction model should be undertaken as the global SHAP values are high for this feature and the local predictions of this feature are consistent, i.e. long average travel times from

checkpoints C_{i+n-1} to C_{i+n} have high SHAP values and vice versa. On the other hand, in the 1-step arrival deviation SHAP evaluation, it could be naively deduced that the average travel time feature is significant in the arrival deviation prediction due to a mean SHAP value. However, upon further inspection, there is no clear consistency to the SHAP values associated with each individual prediction i.e. a high average travel time would result in both positive and negative SHAP values, hence, suggesting that the average travel time feature is simply adding noise to the model and no further statistical tests to determine feature significance should be undertaken and should actually be omitted to improve model performance.

4. Concluding Remarks and Future Works

In summary, this paper demonstrates the predictive capabilities and applicability of state-of-the-art ML techniques when coupled with vast data repositories. Specifically, the DNN and XGBoost models produced, if utilised alongside Darwin's Push Port, will be able to provide real-time predictions of arrival delay, departure delay, travel time and dwell time for any train at any point in its journey for as many steps as the user desires. The SHAP framework allowing for the identification of input feature importance provides the ability to choose features to omit, to undertake further statistical tests such as the F-test or the Granger's causality test and provides an initial stepping stone towards further understanding the causes of delays and hence, a clearer idea of how reactionary delays propagate in a network can be formed. As part of this research, it is evident that the planning of data structure that feeds into the models is paramount in dictating the model selection itself as well as its performance. Furthermore, this research proves that simple models can be applied to extremely complex problems as long as the data structure appropriately represents the complexity, in this case, the interconnectivity of train schedules and delays. As we have been able to accurately predict the cascading delay of a train as a result of its previous delays, the next stage of the research is to incorporate the primary delays which will require more external features such as the weather alongside raising the model's capabilities to be able to predict cascading delays of a particular train due to a delay of another train. This added dimensionality in the predictive capabilities will require a more appropriate model to be able represent the movement of trains in a railway network. Graphical neural networks (GNN) would be appropriate in this task where the vertices can represent the stations and the edges can represent movement of trains. Railway infrastructure data such as speed limits and signals can also be embedded into the GNN to provide further meaningful data. A combination of the SHAP framework and statistical tests such as the Granger's causality test should be utilized to optimize model performance through feature pruning and to provide statistical statements on what is the cause of any train delay at any time with what level of certainty. The ability to make and explain these combinations of delays (primary and cascading due to the train's previous delays and cascading due to another train delay) will provide unprecedented understanding of the delay propagation mechanism and will be extremely useful in formulating appropriate counter-measures and maintenance schedules.

References

Bergstra, J., & Bengio, Y. (2012, February). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, *13*(1), 281–305. Retrieved from http://dl.acm.org/citation.cfm?id=2503308.2188395

Bleakley, L., Akinola, A., & Fullard, R. (2019, Oct). *Rdg information feeds developer pack.* Retrieved from https://www.nationalrail.co.uk/46391.aspx

Bosscha, E. (2016). *Big data in railway operations: Using artificial neural networks to predict train delay propagation* (Unpublished doctoral dissertation).

Buker, T., & Seybold, B. (2012). Stochastic modelling of delay propagation in large networks. *Journal of Rail Transport Planning and Management*, *2*(1), 34-50. Retrieved from http://www.sciencedirect.com/science/article/pii/S2210970612000182

Corman, F., Goverde, R. M. P., & D'Ariano, A. (2009). Rescheduling dense train traffic over complex station interlocking areas. In R. K. Ahuja, R. H. Möhring, & C. D. Zaroliagis (Eds.), (p. 369-386). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from https://doi.org/10.1007/978-3-642-05465-5_16 (ID: Corman2009)

Cule, B., Goethals, B., Tassenoy, S., & Verboven, S. (2011). Mining train delays. In J. Gama, E. Bradley, & J. Hollmén (Eds.), (p. 113-124). Berlin, Heidelberg: Springer Berlin Heidelberg. (ID: 10.1007/978-3-642-24800-9˙13)

Darwin data feeds. (2019). Retrieved from https://www.nationalrail.co.uk/100296.aspx

Department of Transport. (2016, Jul 28). *England and wales 'top 10' overcrowded train services: Spring and autumn 2015* (Tech. Rep.).

Khadilkar, H. (2016). Modelling the impact of control strategy on stochastic delay propagation in transportation networks. In (p. 2471-2476). (ID: 1)

Kim, S., & Kim, H. (2016). A new metric of absolute percentage error for intermittent demand forecasts. *International Journal of Forecasting*, *32*(3), 669 - 679. Retrieved from http://www.sciencedirect.com/science/article/pii/S0169207016000121

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, *abs/1412.6980*. Retrieved from http://arxiv.org/abs/1412.6980

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In (pp. 1097–1105). USA: Curran Associates Inc. Retrieved from http://dl.acm.org/citation.cfm?id=2999134.2999257

Lundberg, S., & Lee, S. (2017). A unified approach to interpreting model predictions. In (Vol. abs/1705.07874). Retrieved from http://arxiv.org/abs/1705.07874

Lundberg, S. M., Erion, G., Chen, H., DeGrave, A., Prutkin, J. M., Nair, B., ... Lee, S.-I. (2019). Explainable ai for trees: From local explanations to global understanding. *arXiv preprint arXiv:1905.04610*.

Lundberg, S. M., Erion, G. G., & Lee, S. (2018). Consistent individualized feature attribution for tree ensembles. *CoRR*, *abs/1802.03888*. Retrieved from http://arxiv.org/abs/1802.03888

Mount, P. (2017). *Downloadable index of darwin.* Retrieved from https://cdn.area51.onl/archive/rail/darwin/

National Rail. (2017, 7 Mar). *About national rail enquiries* (Vol. 2018) (No. 1 Apr). Retrieved from http://www.nationalrail.co.uk/46383.aspx

Office Of Rail and Road. (2018, May 24). *Passenger and freight rail performance 2017-18 q4 statistical release* (Tech. Rep.).

Oneto, L., Fumeo, E., Clerico, G., Canepa, R., Papa, F., Dambra, C., ... Anguita, D. (2018). Train delay prediction systems: A big data analytics perspective. *Big Data Research; Selected papers from the 2nd INNS Conference on Big Data: Big Data and Neural Networks*, *11*, 54-64. Retrieved from http://www.sciencedirect.com/science/article/pii/S2214579617300060

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "why should I trust you?": Explaining the predictions of any classifier. *CoRR*, *abs/1602.04938*. Retrieved from http://arxiv.org/abs/1602.04938

RRUKA. (2017, Oct 31,). *Call for research, data sandbox: Improving network performance* (Tech. Rep.).

Shapley, L. S. (1953). 17. a value for n-person games. *Contributions to the Theory of Games (AM-28), Volume II*, 307–318.

Smith, S. L., Kindermans, P., & Le, Q. V. (2017). Don't decay the learning rate, increase the batch size. *CoRR*, *abs/1711.00489*. Retrieved from `http://arxiv.org/abs/1711.00489`

Sorensen, A. O., Landmark, A. D., Olsson, N. O. E., & Seim, A. A. (2017). Method of analysis for delay propagation in a single-track network. *Journal of Rail Transport Planning and Management*, *7*(1), 77-97. Retrieved from `http://dx.doi.org/10.1016/j.jrtpm.2017.04.001`

Wang, P., & Zhang, Q.-p. (2019, 02). Train delay analysis and prediction based on big data fusion. *Transportation Safety and Environment*, *1*(1), 79-88. Retrieved from `https://doi.org/10.1093/tse/tdy001`

Wang, R., & Work, D. B. (2015, Sep.). Data driven approaches for passenger train delay estimation. In *2015 ieee 18th international conference on intelligent transportation systems* (p. 535-540).

Yang, Y., Huang, P., Peng, Q., LI, J., & Wen, C. (2019, Sep 01). Statistical delay distribution analysis on high-speed railway trains. *Journal of Modern Transportation*, *27*(3), 188–197. Retrieved from `https://doi.org/10.1007/s40534-019-0188-z`

5. Appendices

Appendix A. Data Preprocessing Pseudo-code

Table A1.: Pseudo-code for filling in missing actual train arrival time at current station

Data: Darwin HSP
Result: Filled in missing arrival time of current station
while *Number of total null values change* **do**
  for *All Data* **do**
    if $TD_{C_{i-1}} \neq null$ *and* $TD_{C_i} \neq null$ *and* $i \neq N$ **then**
      $TA_{C_i} \leftarrow TD_{C_{i-1}}$ + *Unique travel time between stations* $C_{i\text{-}1}$ *and* C_i
      if $TA_{C_i} > TD_{C_i}$ **then**
        $TA_{C_i} \leftarrow TD_{C_i}$ − *Unique station dwell time of station* C_i
        if $TA_{C_i} < TD_{C_{i-1}}$ **then**
          $TA_{C_i} \leftarrow null$
        end
      end
    end
    if $TD_{C_{i-1}} \neq null$ *and* $TD_{C_i} = null$ *and* $i = N$ **then**
      $TA_{C_i} \leftarrow TD_{C_{i-1}}$ + *Unique travel time between stations* $C_{i\text{-}1}$ *and* C_i
    end
    if $TD_{C_{i-1}} = null$ *and* $TD_{C_i} \neq null$ *and* $i \neq N$ **then**
      $TA_{C_i} \leftarrow TD_{C_i}$ − *Unique station dwell time of station* C_i
    end
  end
  Calculate total null values
end

Table A2.: Pseudo-code for filling in missing actual train departure time at current station

Data: Darwin HSP
Result: Filled in missing departure time of current station
while *Number of total null values change* **do**
  for *All Data* **do**
    if $TA_{C_i} \neq null$ *and* $TA_{C_{i+1}} \neq null$ *and* $i \neq 0$ **then**
      $TD_{C_i} \leftarrow TA_{C_{i+1}}$ − *Unique travel time between stations* C_i *and* C_{i+1}
      if $TD_{C_i} < TA_{C_i}$ **then**
        $TD_{C_i} \leftarrow TA_{C_i}$ + *Unique station dwell time of station* C_i
        if $TD_{C_i} > TA_{C_{i+1}}$ **then**
          $TD_{C_i} \leftarrow null$
        end
      end
    end
    if $(TA_{C_{i+1}} \neq null$ *and* $i = 0)$ *or* $(TA_{C_i} = null$ *and* $TA_{C_{i+1}} \neq null)$ **then**
      $TD_{C_i} \leftarrow TA_{C_{i+1}}$ − *Unique travel time between stations* C_i *and* C_{i+1}
    end
    if $TA_{C_i} \neq null$ *and* $TA_{C_{i+1}} = null$ *and* $i \neq 0$ **then**
      $TD_{C_i} \leftarrow TA_{C_i}$ + *Unique station dwell time of station* C_i
    end
  end
  Calculate total null values
end

Appendix B. Feature Engineering Process

Table B1.: Pre-processed data

Input Feature																
1	2	3	4	5	6	7	8	9	1 0	11	12	13	14	15	16	17
0	-1	Friday	0	-1	To Didcot Parkway	0	0.961219	No	Yes	January	12	1	1	2	22	0
-3	-2	Friday	-3	-1	To Didcot Parkway	2	1.78744	No	No	January	12	1	2	2	21	22
-5	-3	Friday	-2	-1	To Didcot Parkway	3	2.15385	No	No	January	12	1	3	2	9.5	21
-7	-4	Friday	-2	-1	To Didcot Parkway	2	2.61119	No	No	January	12	1	4	2	4.85994	9
-9	-4	Friday	-2	0	To Didcot Parkway	3	2.013	No	No	January	12	1	5	2	9.83636	5
-9	-5	Friday	0	-1	To Didcot Parkway	2	1.48424	No	No	January	12	1	6	2	12.2636	10
-9	-6	Friday	0	-1	To Didcot Parkway	1	2.26357	No	No	January	12	1	7	2	10.8893	12
-10	-6	Friday	-1	0	To Didcot Parkway	2	2.41181	No	No	January	12	1	8	2	14.0913	11
-10	4	Friday	0	10	To Didcot Parkway	11	3.43594	No	No	January	12	1	9	2	17.213	15
1	22	Friday	11	18	To Didcot Parkway	10	3.68206	No	No	January	12	1	10	2	12.3693	18

Table B2.: Encoded pre-processed data

Input Feature																
1	2	3	4	5	6	7	8	9	1 0	11	12	13	14	15	16	17
0	-1	4	0	-1	1	0	0.961219	0	1	1	12	1	1	2	22	0
-3	-2	4	-3	-1	1	2	1.78744	0	0	1	12	1	2	2	21	22
-5	-3	4	-2	-1	1	3	2.15385	0	0	1	12	1	3	2	9.5	21
-7	-4	4	-2	-1	1	2	2.61119	0	0	1	12	1	4	2	4.85994	9
-9	-4	4	-2	0	1	3	2.013	0	0	1	12	1	5	2	9.83636	5
-9	-5	4	0	-1	1	2	1.48424	0	0	1	12	1	6	2	12.2636	10
-9	-6	4	0	-1	1	1	2.26357	0	0	1	12	1	7	2	10.8893	12
-10	-6	4	-1	0	1	2	2.41181	0	0	1	12	1	8	2	14.0913	11
-10	4	4	0	10	1	11	3.43594	0	0	1	12	1	9	2	17.213	15
1	22	4	11	18	1	10	3.68206	0	0	1	12	1	10	2	12.3693	18

Table B3.: Encoded pre-processed data

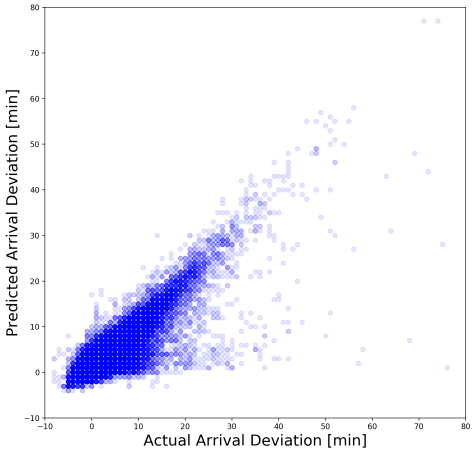
Input Feature																
Mon	Tue	Wed	Thu	Fri	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0

Table B4.: Input matrix

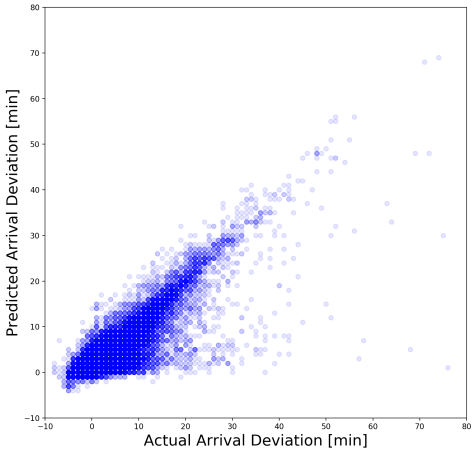
Input Feature															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31
-0.48	-0.50	-0.51	-0.51	1.99	3.32	-0.30	-0.31	-0.30	-0.31	-0.30	-0.30	-0.29	-0.31	-0.30	-0.31

Input Feature															
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
-0.28	-0.36	-0.57	-0.35	-0.78	0.93	-0.77	-0.44	-0.30	3.30	-0.51	-1.62	-1.31	0.38	2.04	-1.09
-0.28	-0.46	-0.60	-1.06	-0.78	0.93	0.24	0.28	-0.30	-0.30	-0.51	-1.62	-1.12	0.38	1.89	1.98
-0.28	-0.54	-0.64	-0.82	-0.78	0.93	0.74	0.59	-0.30	-0.30	-0.51	-1.62	-0.93	0.38	0.24	1.84
-0.28	-0.61	-0.67	-0.82	-0.78	0.93	0.24	0.99	-0.30	-0.30	-0.51	-1.62	-0.74	0.38	-0.43	0.17
-0.28	-0.68	-0.67	-0.82	-0.52	0.93	0.74	0.47	-0.30	-0.30	-0.51	-1.62	-0.55	0.38	0.29	-0.39
-0.28	-0.68	-0.70	-0.35	-0.78	0.93	0.24	0.01	-0.30	-0.30	-0.51	-1.62	-0.35	0.38	0.64	0.31
-0.28	-0.68	-0.74	-0.35	-0.78	0.93	-0.27	0.69	-0.30	-0.30	-0.51	-1.62	-0.16	0.38	0.44	0.59
-0.28	-0.71	-0.74	-0.59	-0.52	0.93	0.24	0.82	-0.30	-0.30	-0.51	-1.62	0.03	0.38	0.90	0.45
-0.28	-0.71	-0.40	-0.35	2.04	0.93	4.77	1.70	-0.30	-0.30	-0.51	-1.62	0.22	0.38	1.35	1.00
-0.28	-0.32	0.20	2.24	4.10	0.93	4.27	1.92	-0.30	-0.30	-0.51	-1.62	0.41	0.38	0.65	1.42

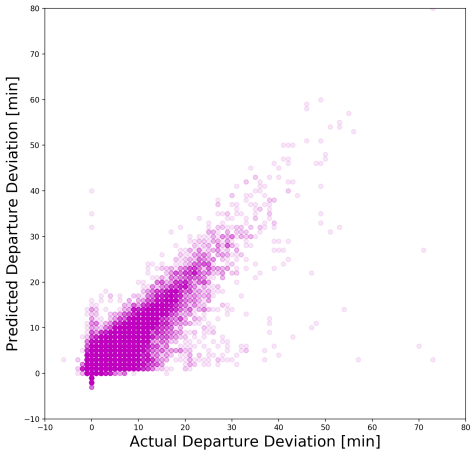
Appendix C. DNN vs XGBoost Results



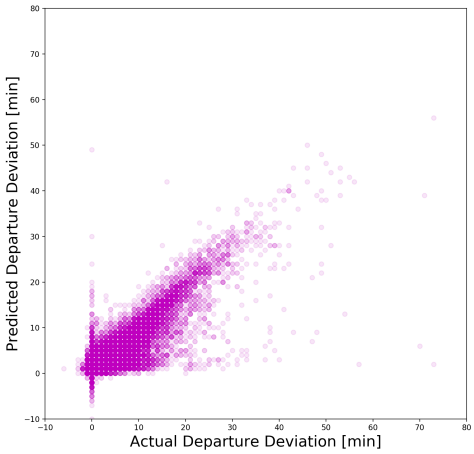
(a) 2-step DNN deviation from arrival



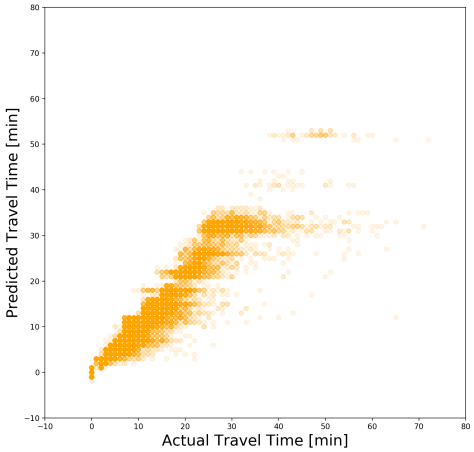
(b) 2-step XGBoost deviation from arrival



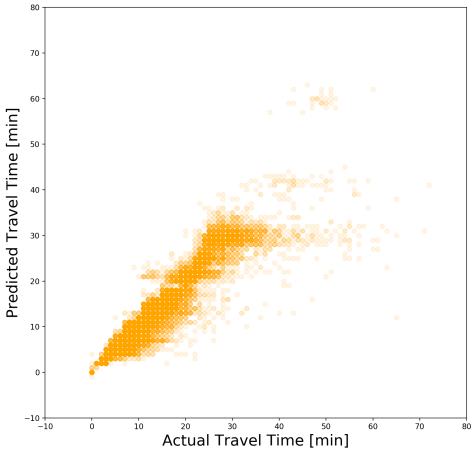
(c) 2-step DNN deviation from departure



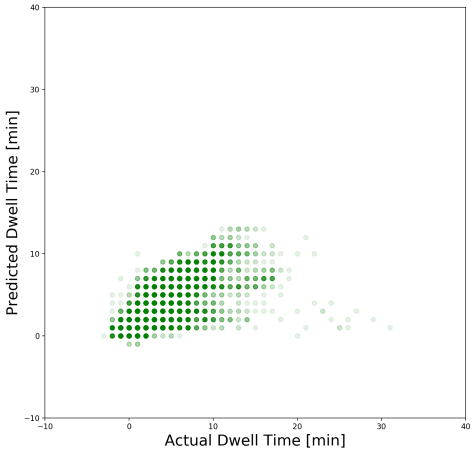
(d) 2-step XGBoost deviation from departure



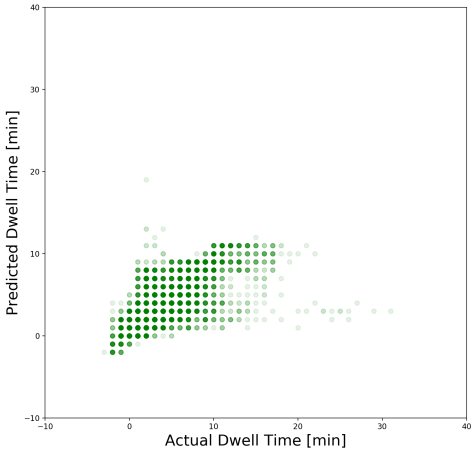
(e) 2-step DNN travel time



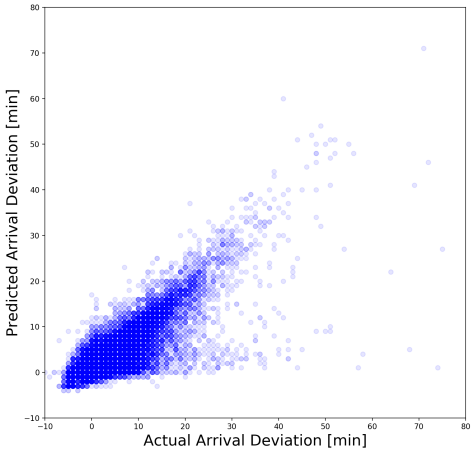
(f) 2-step XGBoost travel time



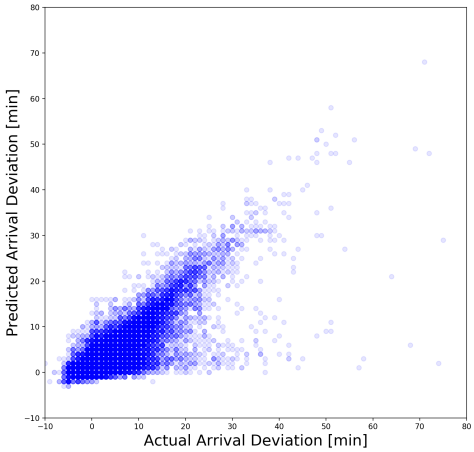
(g) 2-step DNN dwell time



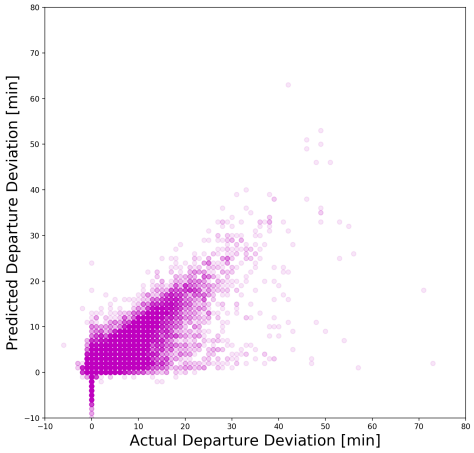
(h) 2-step XGBoost dwell time



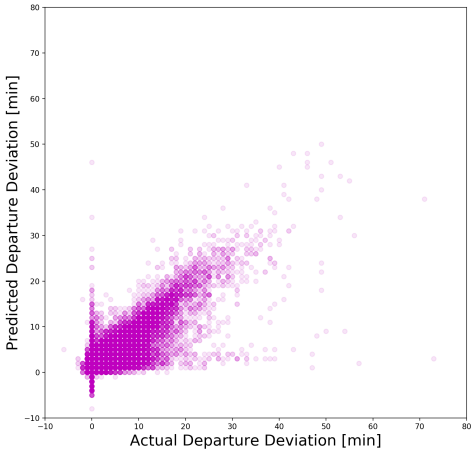
(i) 3-step DNN deviation from arrival



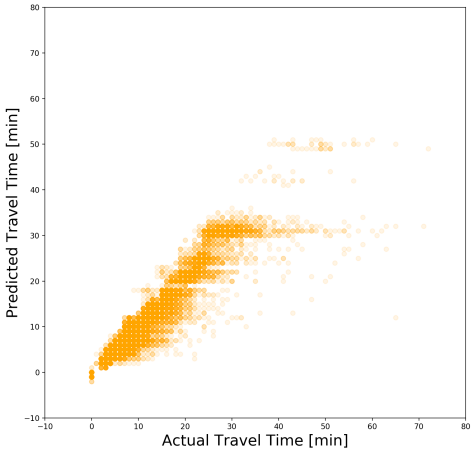
(j) 3-step XGBoost deviation from arrival



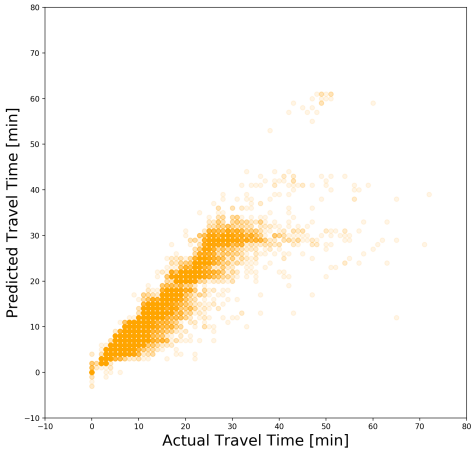
(k) 3-step DNN deviation from departure



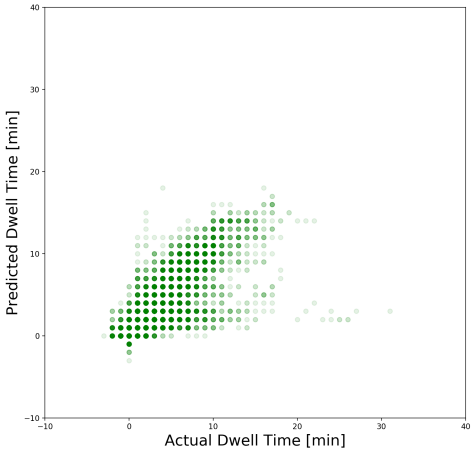
(l) 3-step XGBoost deviation from departure



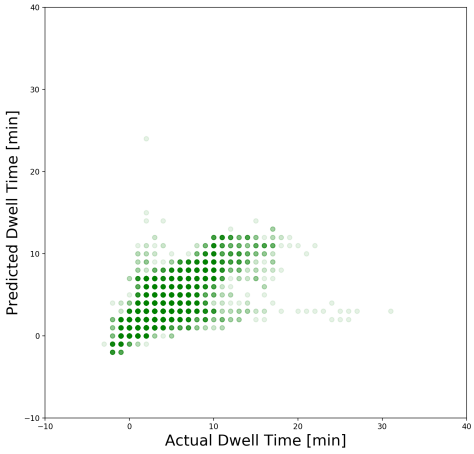
(m) 3-step DNN travel time



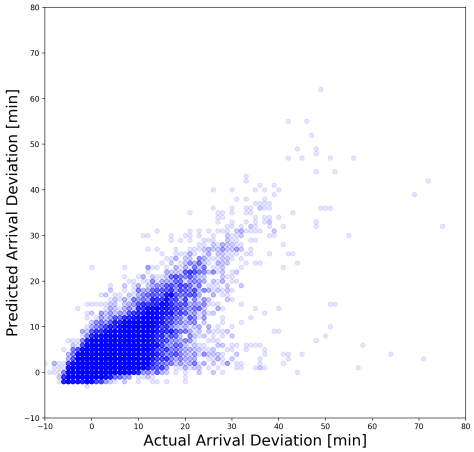
(n) 3-step XGBoost travel time



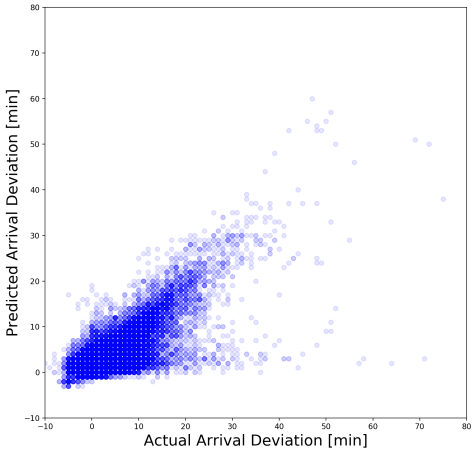
(o) 3-step DNN dwell time



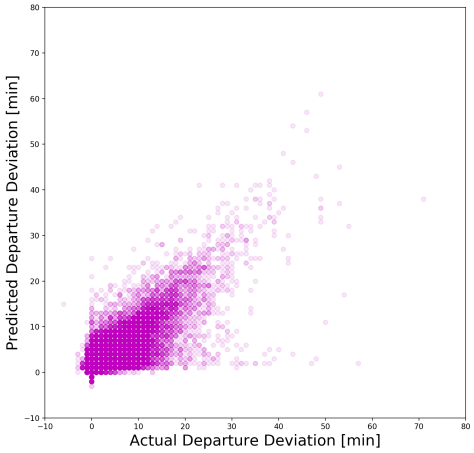
(p) 3-step XGBoost dwell time



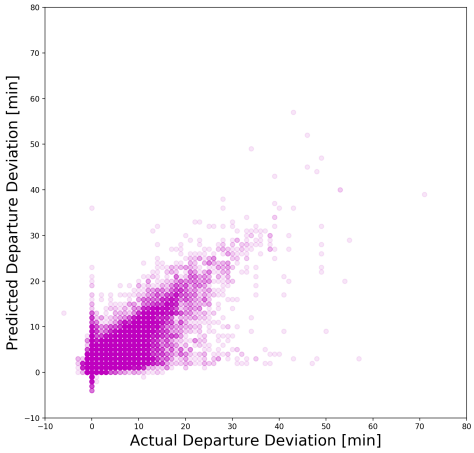
(q) 4-step DNN deviation from arrival



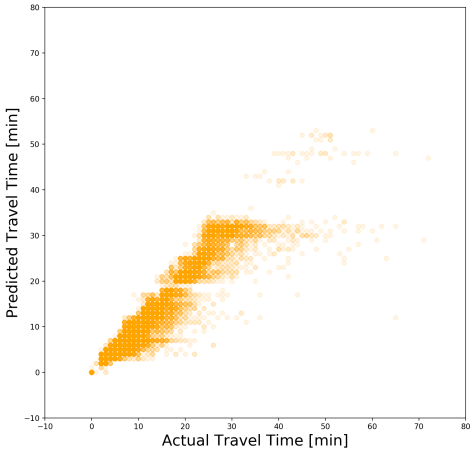
(r) 4-step XGBoost deviation from arrival



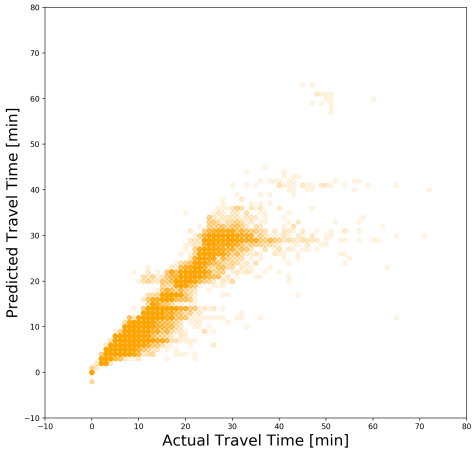
(s) 4-step DNN deviation from departure



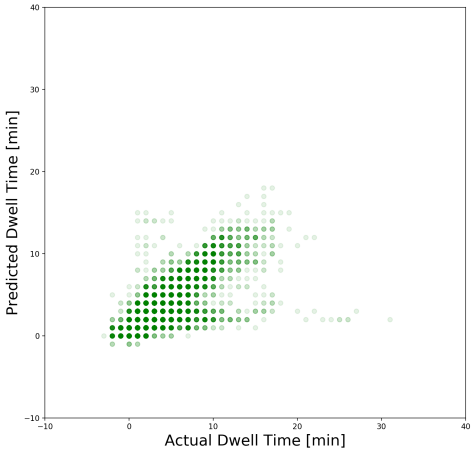
(t) 4-Step XGBoost Deviation from Departure



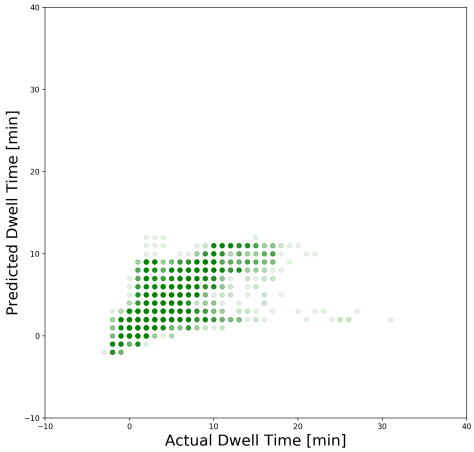
(u) 4-step DNN travel time



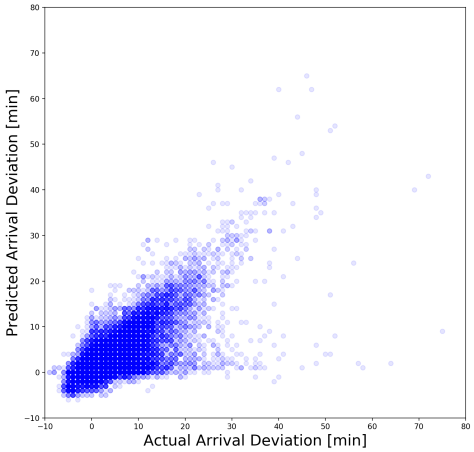
(v) 4-step XGBoost travel time



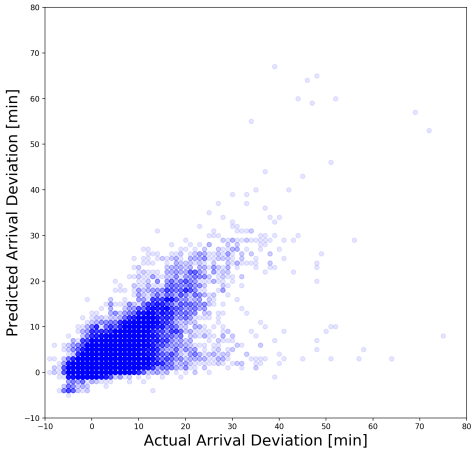
(w) 4-step DNN dwell time



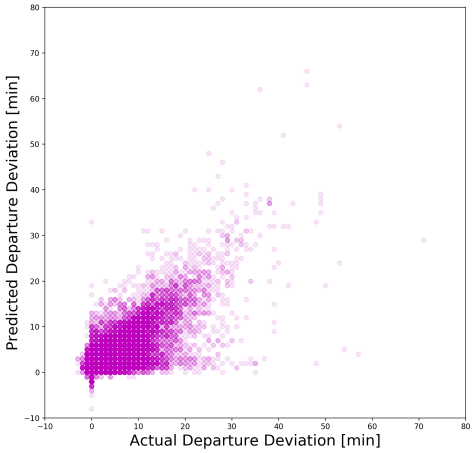
(x) 4-step XGBoost dwell time



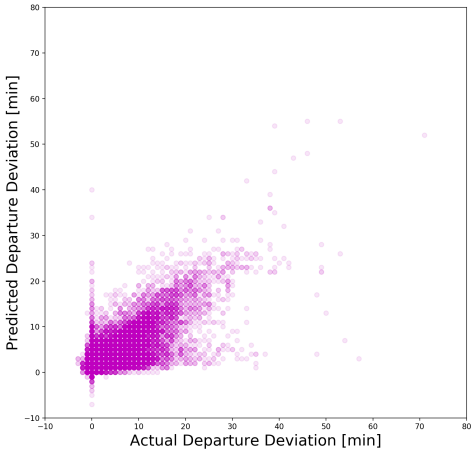
(y) 5-step DNN deviation from arrival



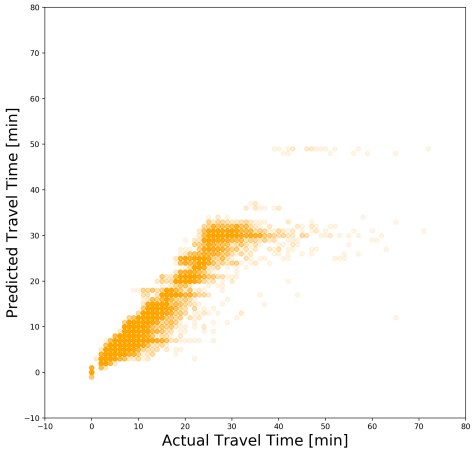
(z) 5-step XGBoost deviation from arrival



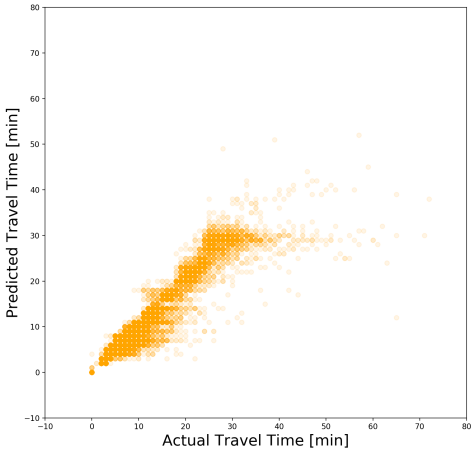
(aa) 5-step DNN deviation from departure



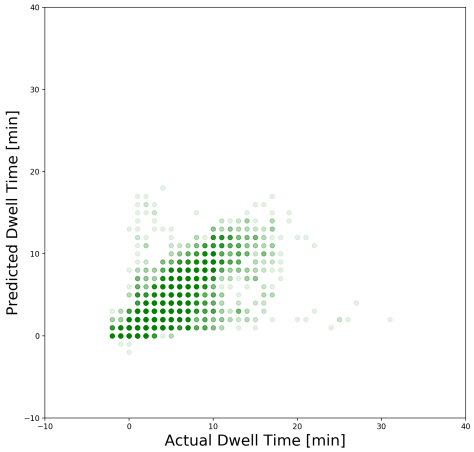
(ab) 5-step XGBoost deviation from departure



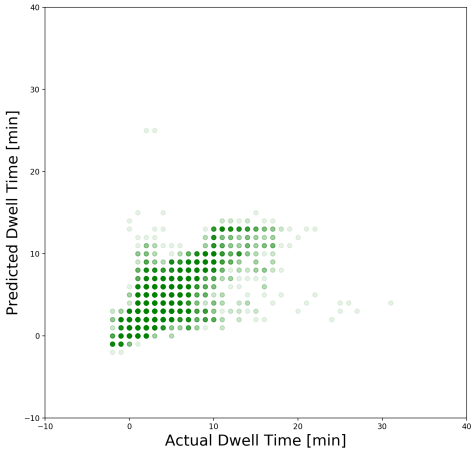
(ac) 5-step DNN travel time



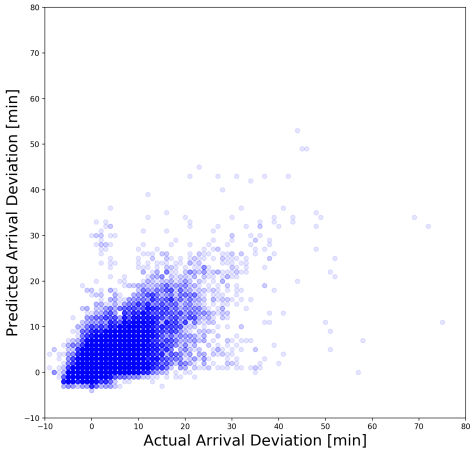
(ad) 5-step XGBoost travel time



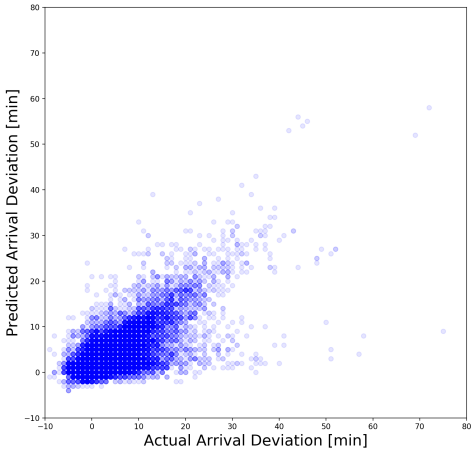
(ae) 5-step DNN dwell time



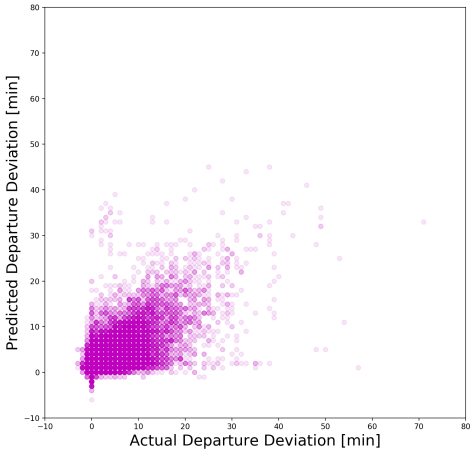
(af) 5-step XGBoost dwell time



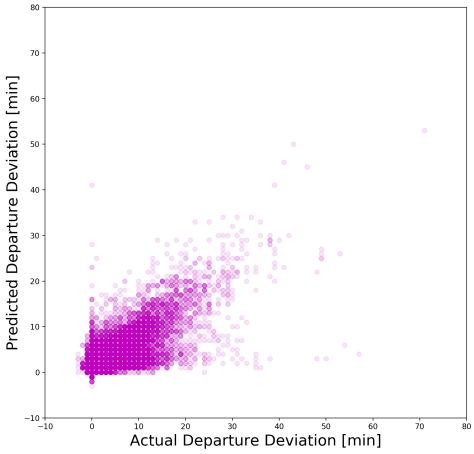
(ag) 6-step DNN deviation from arrival



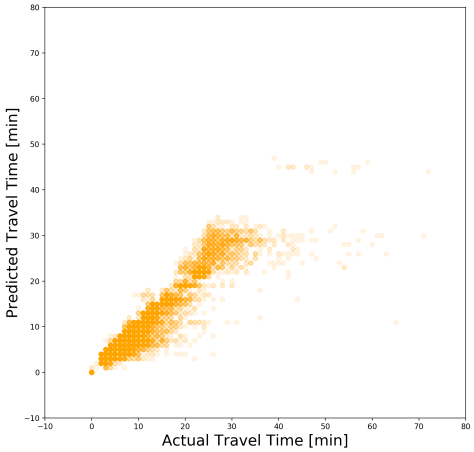
(ah) 6-step XGBoost deviation from arrival



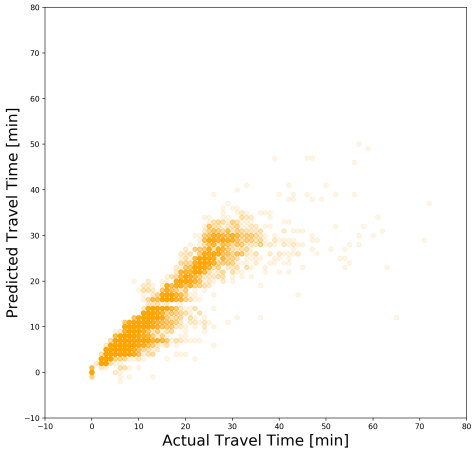
(ai) 6-step DNN deviation from departure



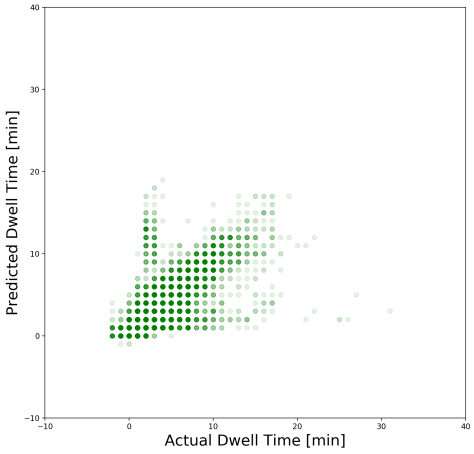
(aj) 6-step XGBoost deviation from departure



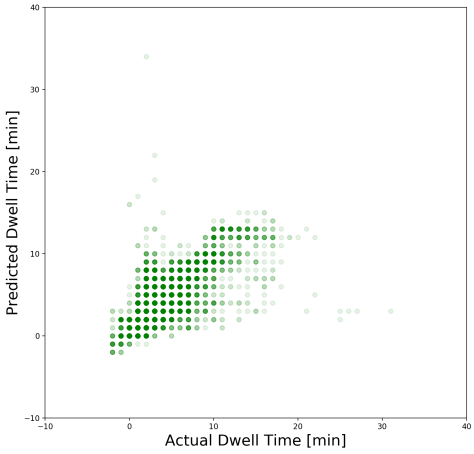
(ak) 6-step DNN travel time



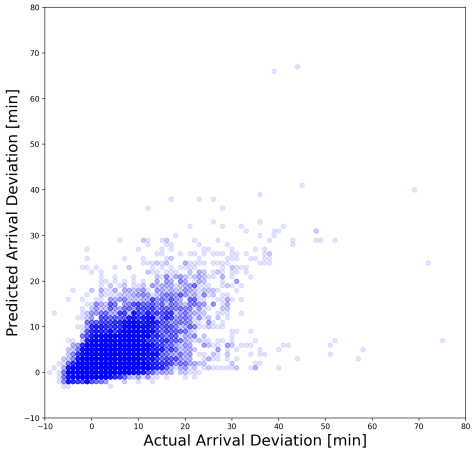
(al) 6-step XGBoost travel time



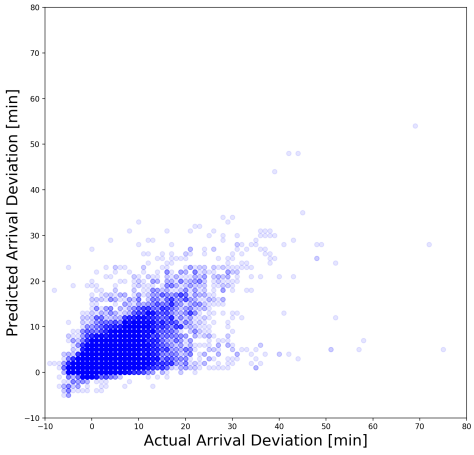
(am) 6-step DNN dwell time



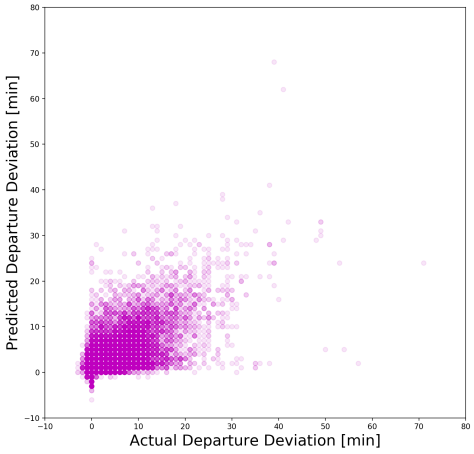
(an) 6-step XGBoost dwell time



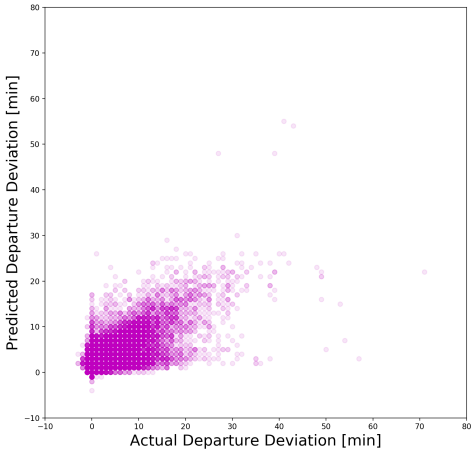
(ao) 7-step DNN deviation from arrival



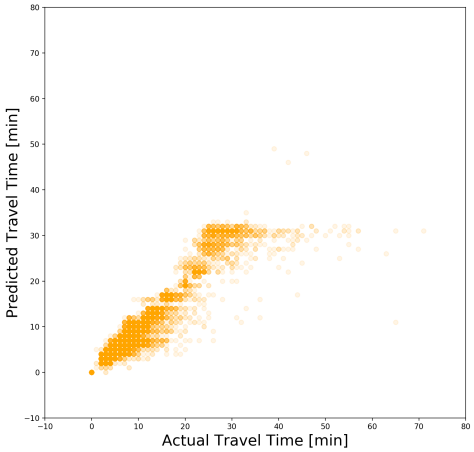
(ap) 7-step XGBoost deviation from arrival



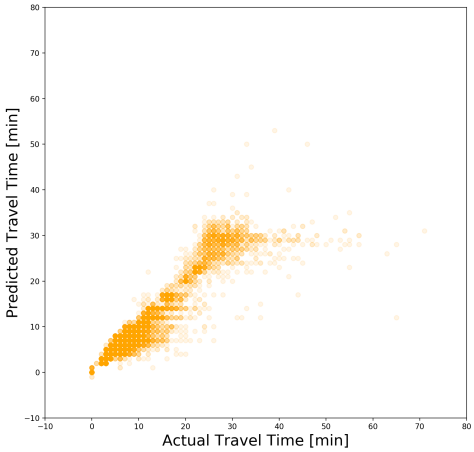
(aq) 7-step DNN deviation from departure



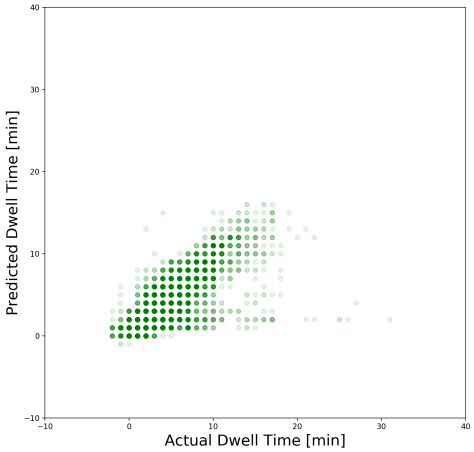
(ar) 7-step XGBoost deviation from departure



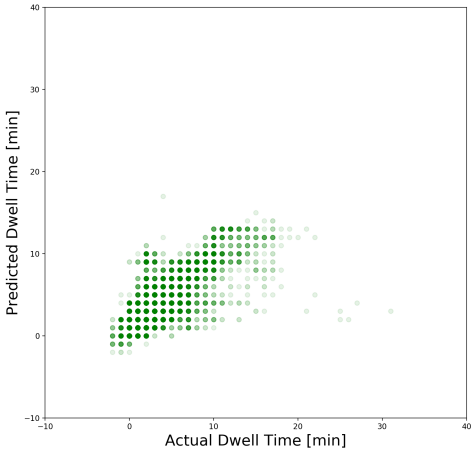
(as) 7-step DNN travel time



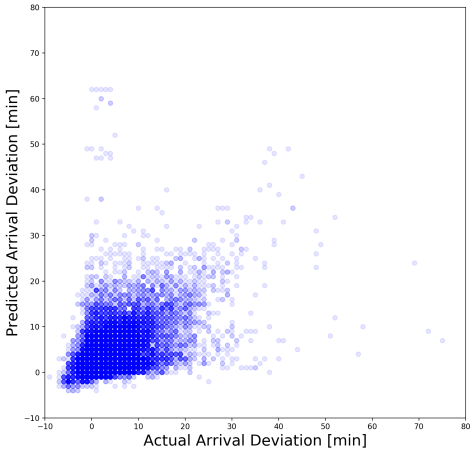
(at) 7-step XGBoost travel time



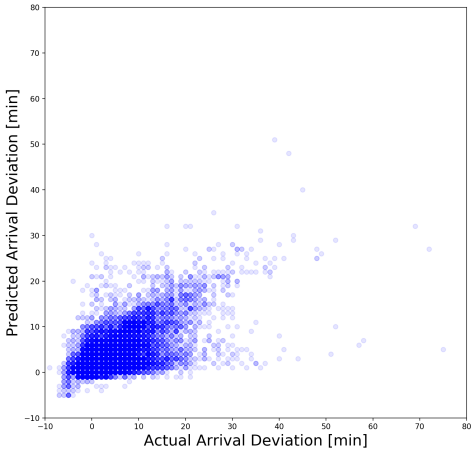
(au) 7-step DNN dwell time



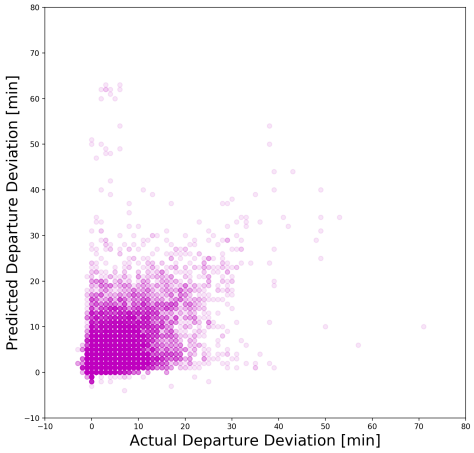
(av) 7-step XGBoost dwell time



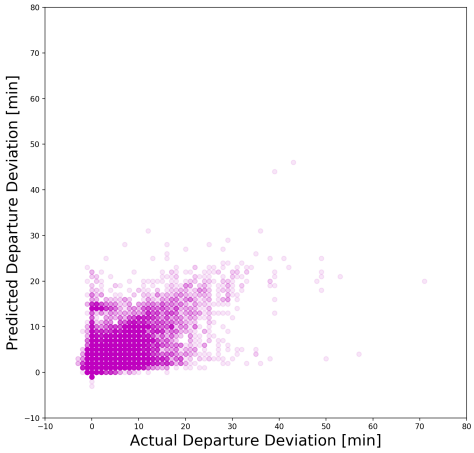
(aw) 8-step DNN deviation from arrival



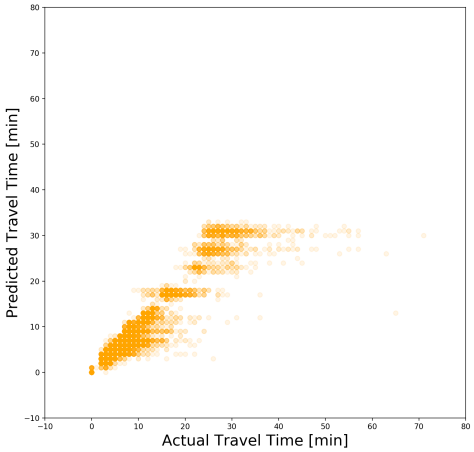
(ax) 8-step XGBoost deviation from arrival



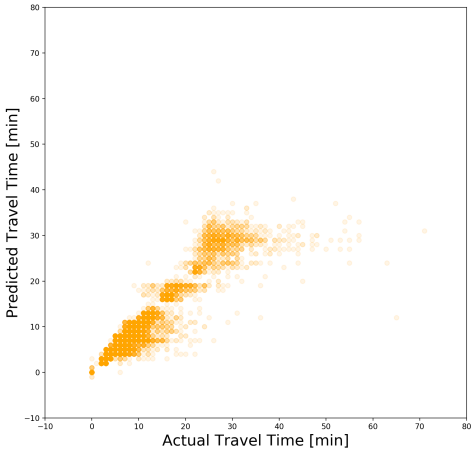
(ay) 8-step DNN deviation from departure



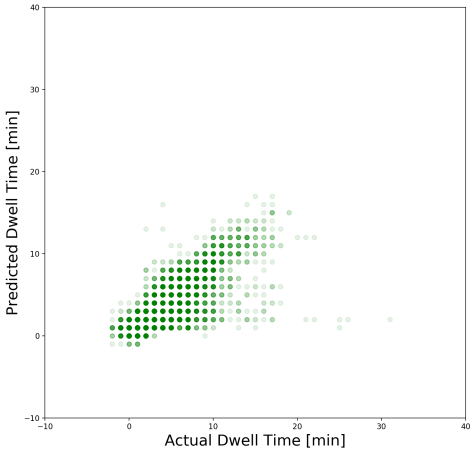
(az) 8-step XGBoost deviation from departure



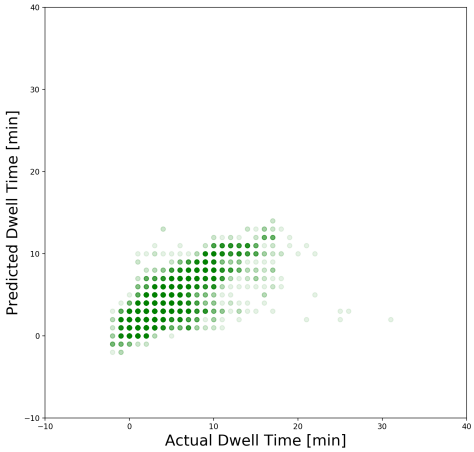
(ba) 8-step DNN travel time



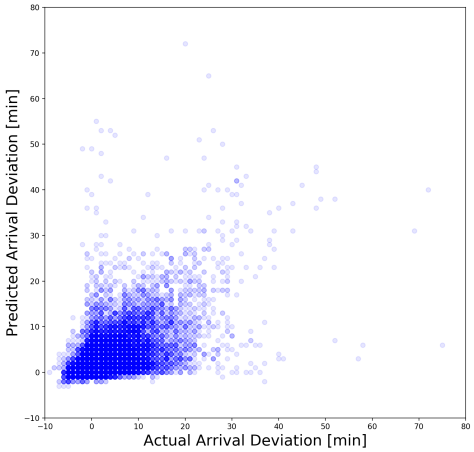
(bb) 8-step XGBoost travel time



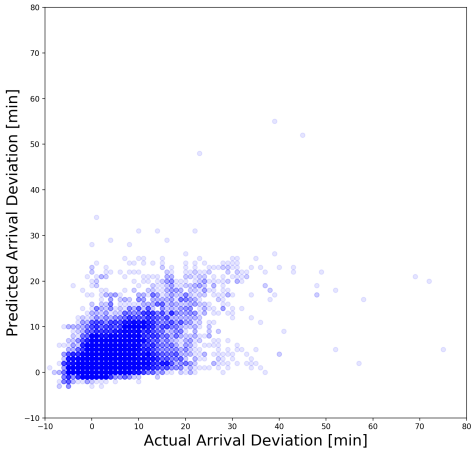
(bc) 8-step DNN dwell time



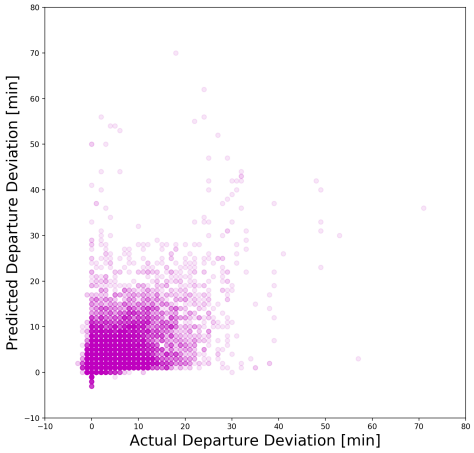
(bd) 8-step XGBoost dwell time



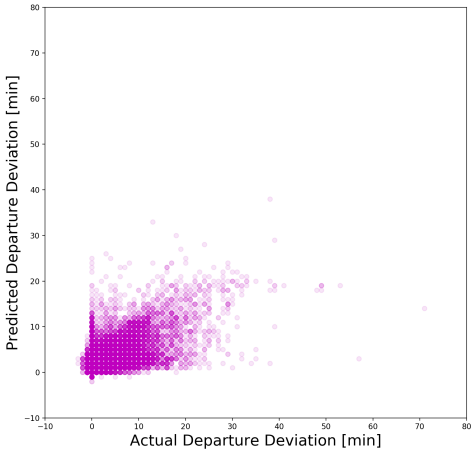
(be) 9-step DNN deviation from arrival



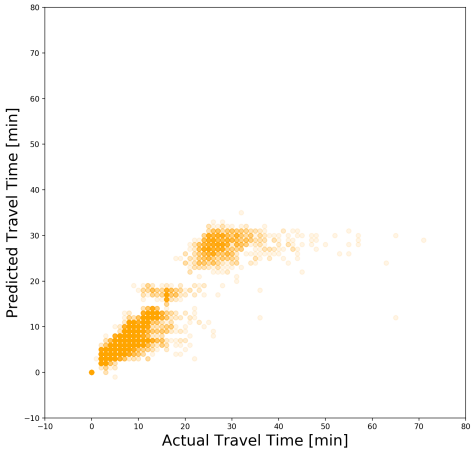
(bf) 9-step XGBoost deviation from arrival



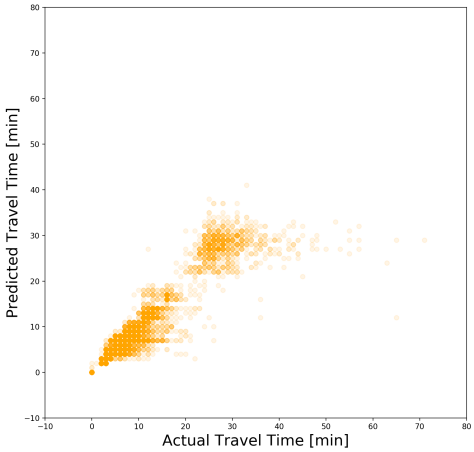
(bg) 9-step DNN deviation from departure



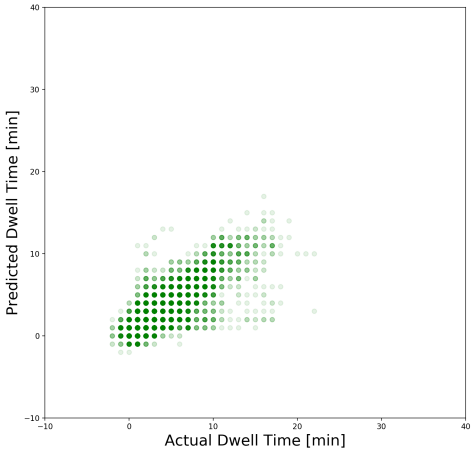
(bh) 9-step XGBoost deviation from departure



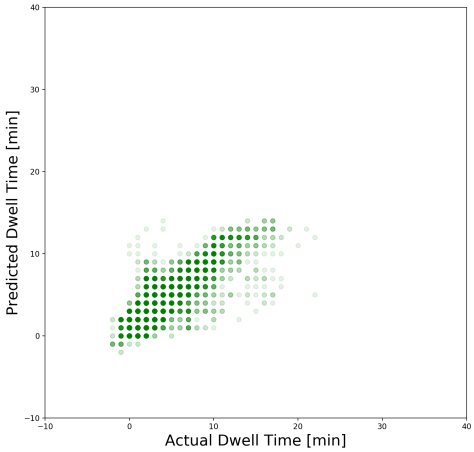
(bi) 9-step DNN travel time



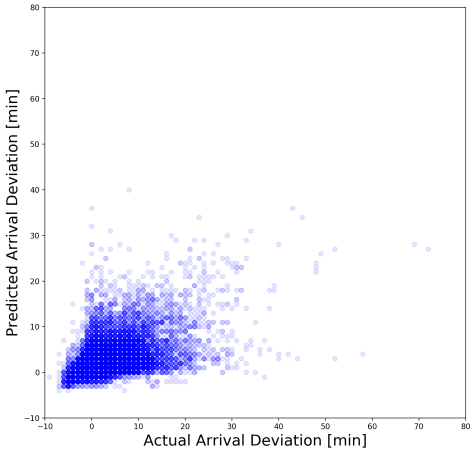
(bj) 9-step XGBoost travel time



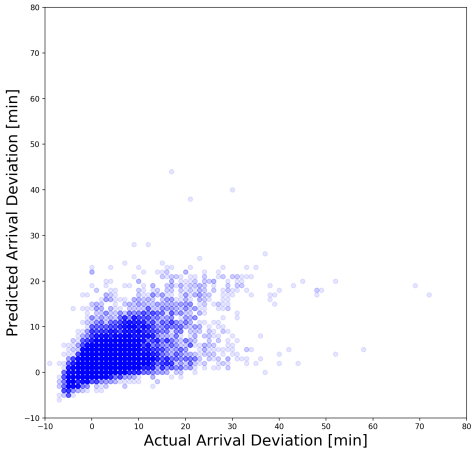
(bk) 9-step DNN dwell time



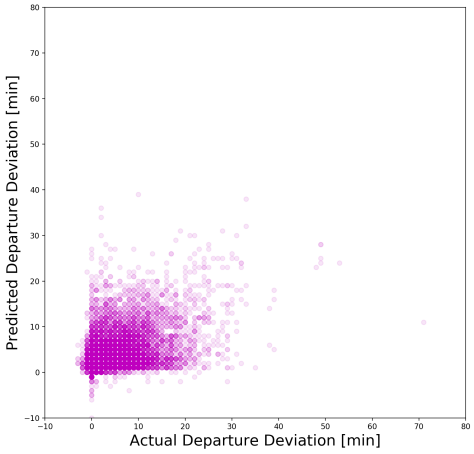
(bl) 9-step XGBoost dwell time



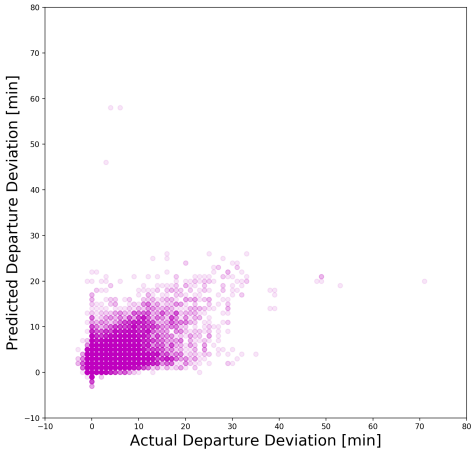
(bm) 10-step DNN deviation from arrival



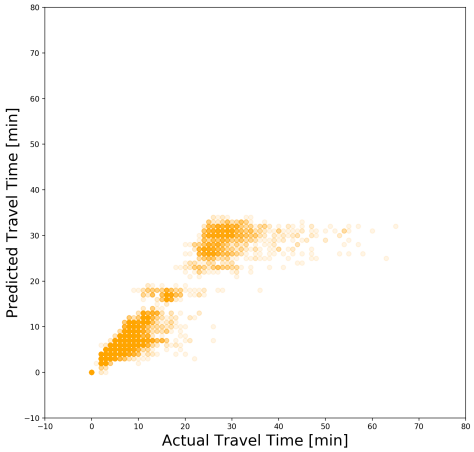
(bn) 10-step XGBoost deviation from arrival



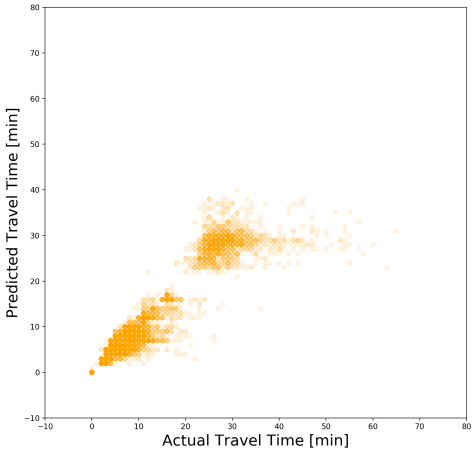
(bo) 10-step DNN deviation from departure



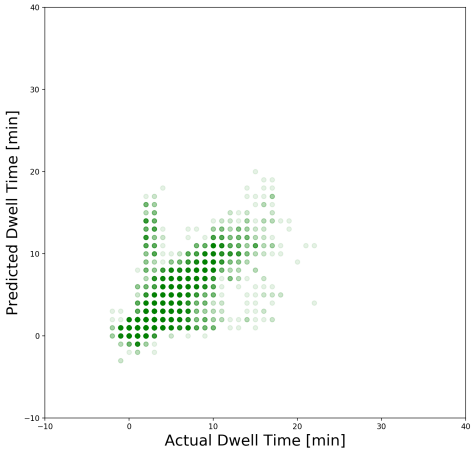
(bp) 10-step XGBoost deviation from departure



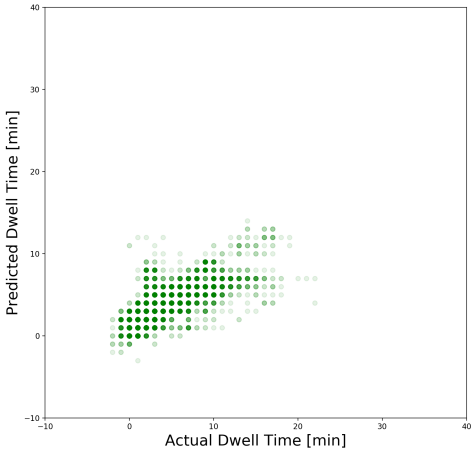
(bq) 10-step DNN travel time



(br) 10-step XGBoost travel time

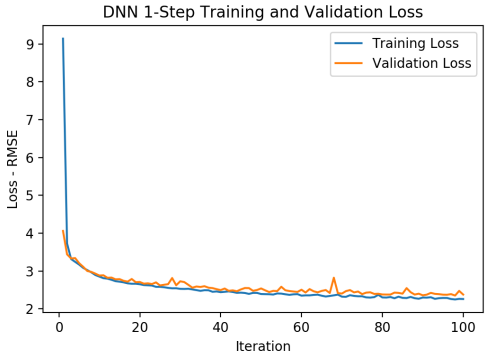


(bs) 10-step DNN dwell time

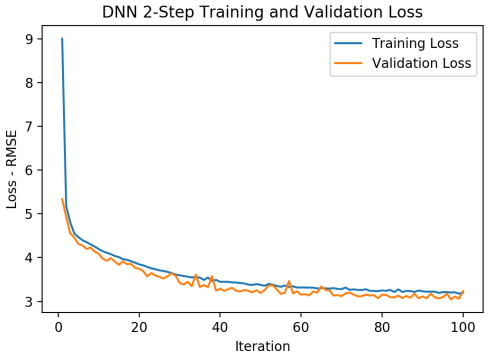


(bt) 10-step XGBoost dwell time

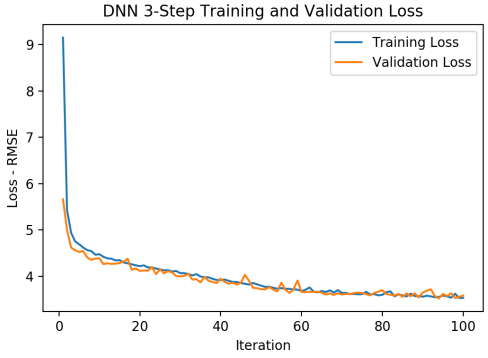
Appendix D. DNN Training and Validation Loss



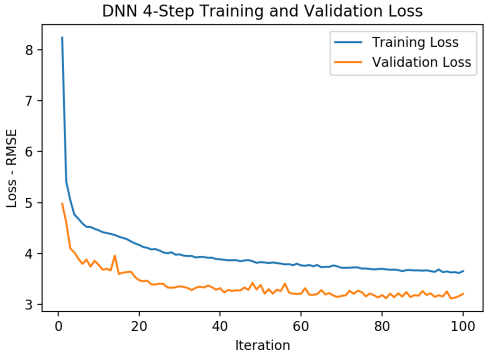
(a) 1-step



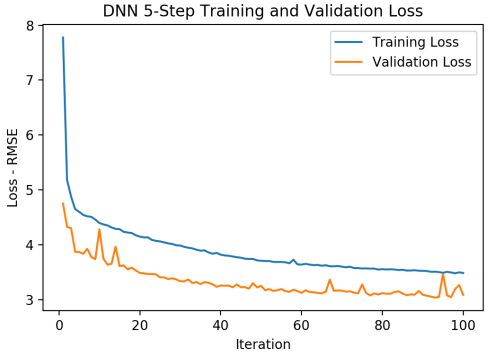
(b) 2-step



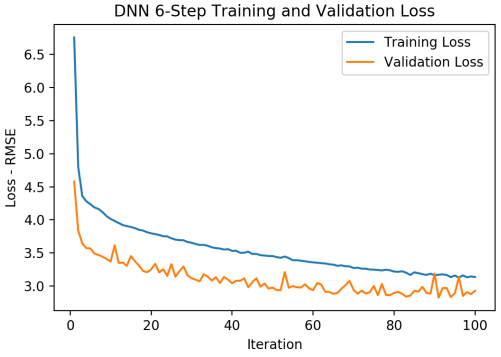
(c) 3-step



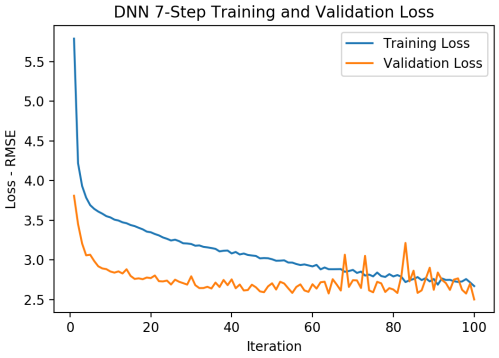
(d) 4-step



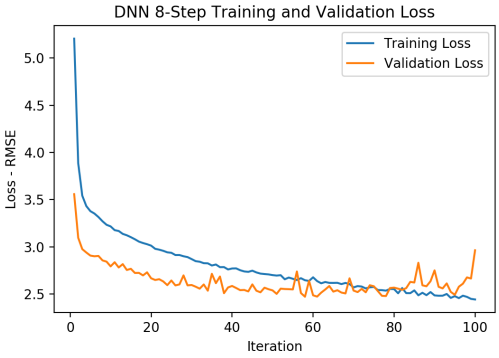
(e) 5-step



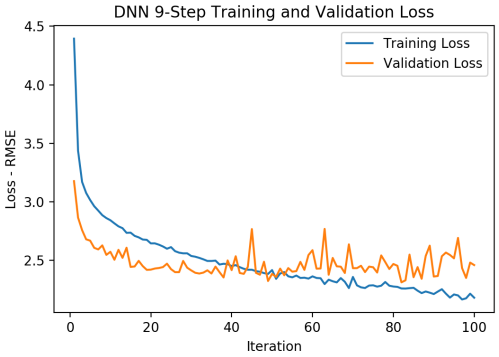
(f) 6-step



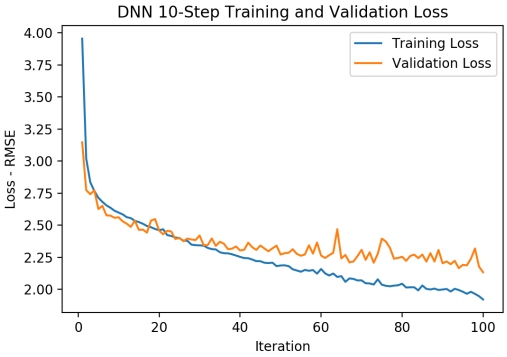
(g) 7-step



(h) 8-step

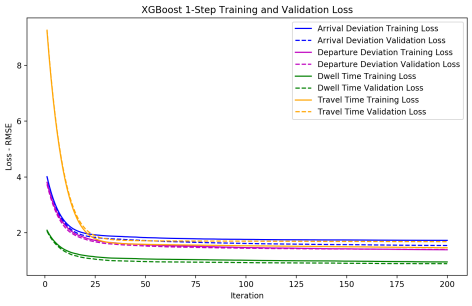


(i) 9-step

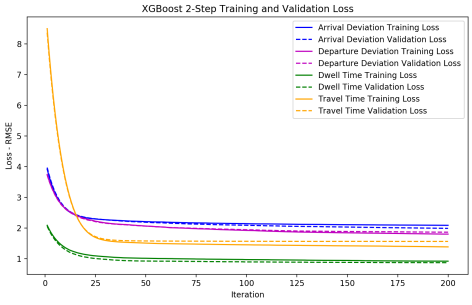


(j) 10-step

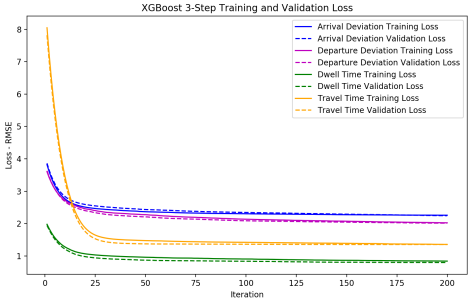
Appendix E. XGBoost Training and Validation Loss



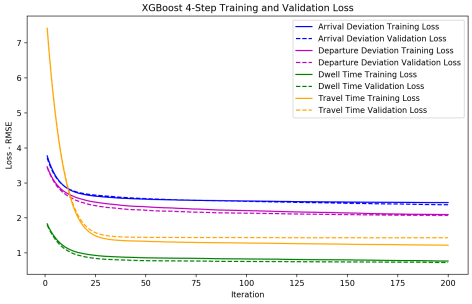
(a) 1-step



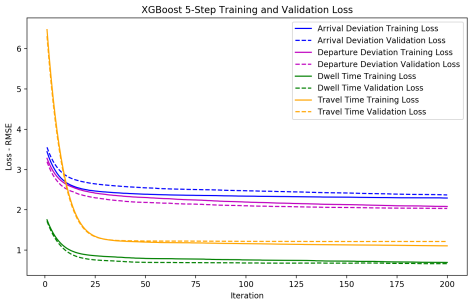
(b) 2-step



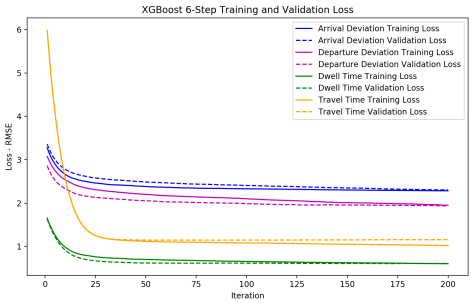
(c) 3-step



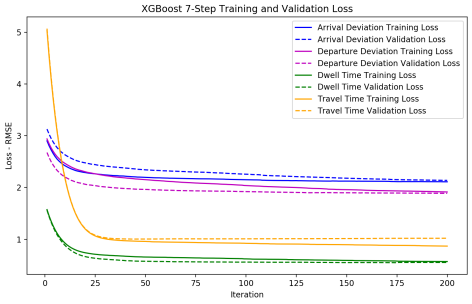
(d) 4-step



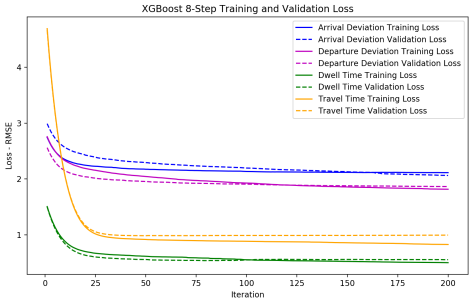
(e) 5-step



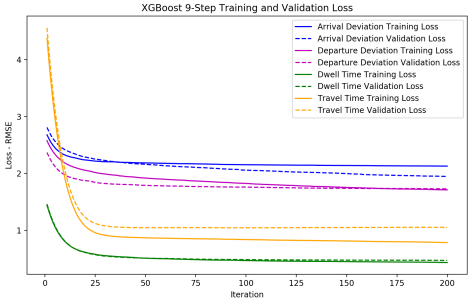
(f) 6-step



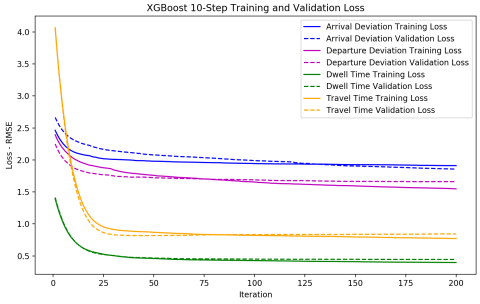
(g) 7-step



(h) 8-step



(i) 9-step



(j) 10-step