Real-Time Prediction System of Train Carriage Load Based on Multi-Stream Fuzzy Learning

Hang Yu, *Member, IEEE*, Jie Lu, *Fellow, IEEE*, Anjin Liu, *Member, IEEE*, Bin Wang, Ruimin Li, and Guangquan Zhang, *Member, IEEE*

***Abstract*— When a train leaves a platform, knowing the carriage load (the number of passengers in each carriage) of this train will support train managers to guide passengers at the next platform to choose carriages to avoid congestion. This capacity has become critical since the onset of the pandemic. However, with the dynamicity of passengers and the speed of trains improved (about 3 minutes travel between stations) as well as the station stop period reduced (60–90 second per station), the real-time prediction is more challenging. This paper presents an intelligent system, which is developed in collaboration with Sydney Trains, for real-time predicting carriage load across a city passenger train network. The system comprises three innovations. First, a fuzzy time-matching method significantly improves prediction accuracy in the uncertain situations and allows noisy historical data to be used for training. Second, the LightGBM model is extended with an incremental learning scheme to make forecasting in real-time possible. Third, a new multi-stream learning strategy that merges data streams with similar concept drift patterns is pioneered to increase the amount of suitable training data while reducing generalization errors. A comprehensive suite of practical tests on real-world datasets demonstrates the merit of these solutions.**

***Index Terms*— Transportation systems, real-time prediction, carriage load, multi-stream, concept drift.**

I. Introduction

CITY trains transport management is an important part of a smart city. One of intelligent train operation management tasks is carriage load prediction, which needs to be real-time. That is, when a train leaves a platform, real-time knowledge of the number of passengers in each train carriage. Carriage load prediction contributes to guiding passengers at the next platform to choose which carriage to take to avoid congestion. It is especially important given COVID-19 because the need to socially distance makes this a vital function for the continued safe-running of any rail transport network. Consequently, many prediction methods in the public transport area have been proposed and classified into a traditional parametric approach such as time-series analysis or a non-parametric method based on multiple regression analysis [3], [4].

The most common parametric method, especially for traffic flow prediction, is auto-regressive integrated moving average (ARIMA) [5]. However, ARIMA models only support one input variable, which limits them to relatively simple prediction tasks. The non-parametric methods include neural networks [7] and support vector regression [8], [9]. Wei and Chen [6] combined the back-propagation network and empirical mode decomposition to predict the passenger flow of short-term subway. Roos *et al.* [1] proposed a dynamic Bayesian network to predict the short-term passenger flow of city rail transit in Paris. Pekel *et al.* [2] combined the algorithm of parliament optimization and artificial neural network. These models are simpler and easier to interpret results, and have a quicker response and lower cost. However, these methods involve manual feature selection, and good results are heavily dependent on which features of the traffic data are selected. More recently, deep learning approaches have become the norm, with convolutional neural networks (CNNs) or long short-term memory (LSTM) as the usual implementation choice [10]–[14]. For example, Lv *et al.* [14] proposed a CNNs-based traffic flow prediction method that inherently considered the spatial and temporal correlations. Yang *et al.* [10] proposed a stacked autoencoder model, which is a type of deep architecture of CNNs approach aiming to improve forecasting accuracy. Ma *et al.* [12] applied LSTM to forecast traffic speed, and demonstrated that LSTM could capture the long-term temporal dependence of traffic data. Liu *et al.* [13] proposed an end-to-end combined CNNs and LSTM, being able to extract the spatial-temporal information from the traffic flow. Most of these mentioned models have good predication accuracy, but have lower response and higher cost, and cannot learn in a nonstationary environment.

The Sydney Trains cooperation also needs an intelligent system to predict the passenger loads in COVID-19. However, the Sydney Train network is a hybrid city-suburban light rail system, so the carriage load are highly dynamic. In addition, the stations are relatively close together, with generally only 3–5 minutes between stops and each train only setting down at a station for around 90 seconds. Hence, this system

Manuscript received May 11, 2021; revised September 1, 2021 and October 27, 2021; accepted December 14, 2021. This work was supported in part by the Carriage Load Prediction Project coordinated by Sydney Trains under Grant PRO20-9756, in part by the Australian Research Council (ARC) under Discovery Grant DP190101733, and in part by the Laureate Project under Grant FL190100149. The Associate Editor for this article was S. Santini. *(Corresponding author: Jie Lu.)*

Hang Yu, Jie Lu, Anjin Liu, Bin Wang, and Guangquan Zhang are with the Decision Systems and e-Service Intelligence Laboratory, Australian Artificial Intelligence Institute, Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW 2007, Australia (e-mail: hang.yu@student.uts.edu.au; jie.lu@uts.edu.au; anjin.liu@uts.edu.au; bin.wang-7@student.uts.edu.au; guangquan.zhang@uts.edu.au).

Ruimin Li is with the Operations Analysis and Modeling Department, Sydney Trains, Haymarket, NSW 2000, Australia (e-mail: ruimin.li2@transport.nsw.gov.au).

Digital Object Identifier 10.1109/TITS.2021.3137446

is a real-time system essentially for predicting train carriage load. As a result, some of the complicating factors include:

1) Inputted data were collected from the Open Data Hub[1] in a stream manner. Hence, all data cannot be obtained at once precluding batch-based prediction; a streaming method is the only solution [18]–[21].
2) During special events or times of disruption, historical carriage load data has a different underlying distribution i.e., concept drift, which means the data are nonstationary [15]–[17].
3) Each train has multiple carriages (usually 4–8) but only some share the same passenger load pattern. Therefore, the system needs to output multiple predictions [32]–[34].
4) In practice, uncertainty is the main issue that existing carriage load prediction models have yet to adequately address [22], [23].

Therefore, to devise a more accurate system for predicting passenger loads, we have added robust real-time forecasting and drift detection to Sydney Train's current prediction system. The main contributions of this paper include:

– the development of a fuzzy time-matching method [24]–[26] to significantly improve prediction accuracy given noisy and uncertain historical data;
– an exploration of the factors that influence model selection and feature engineering that provides valuable insights into train carriage load prediction;
– an extension to the LightGBM model [27] to accommodate incremental learning. This variant is designed to predict train carriage load based on streaming data, while taking noise, drift, and disruption into account;
– a discussion of our findings from this investigation of strategies for multi-stream learning with drift detection.

The rest of this paper is organized as follows. Section 2 details the background of the study. Section 3 describes the specific problem setting and methodology proposed to solve the prediction task. Section 4 presents the results of our evaluation experiments. Section 5 concludes the paper.

II. Background

In collaborating on this research, Sydney Trains was responsible for collecting the training data from the Open Data Hub hosted by Transport for NSW (TfNSW) [28]. The team at UTS was responsible for designing a system that could make the desired predictions in real time.

A. Data Source

As of Sep 2020, the Open Data Hub community numbers more than 30,000 users [28]. Its apps, which relate to everything from information about train running times to the weight of each of a train's carriages have been downloaded millions of times. The data is gathered through sensors on the trains and can be accessed in real time via an API and a HTTP GET request. Although the Open Data Hub has information on many different rail networks across the state, we chose the passenger network running through the suburbs of Sydney as the subject of this paper since it is, by far, the most complex.

[1]The Open Data Hub is a state government initiative where developers, entrepreneurs, and data analysts can come together to create innovative solutions for smart transportation. On this platform, a large amount of transactional data containing very detailed information is made available by transport service providers and other interest groups.

B. Trains & Network

Sydney Trains is a hybrid city-suburban railway comprising 813 kilometres of track, 175 stations, and eight lines. It also includes a small underground system of five stations at the core of the network [29]. The underground section has subway-comparable train frequencies of every three minutes or shorter. Off-peak frequencies average 3 to 5 minutes at most major and inner-city stations, and 15 minutes at most minor stations, with greater frequencies during peak times [29]. In 2018–2019, the network carried roughly 377.1 million passengers.

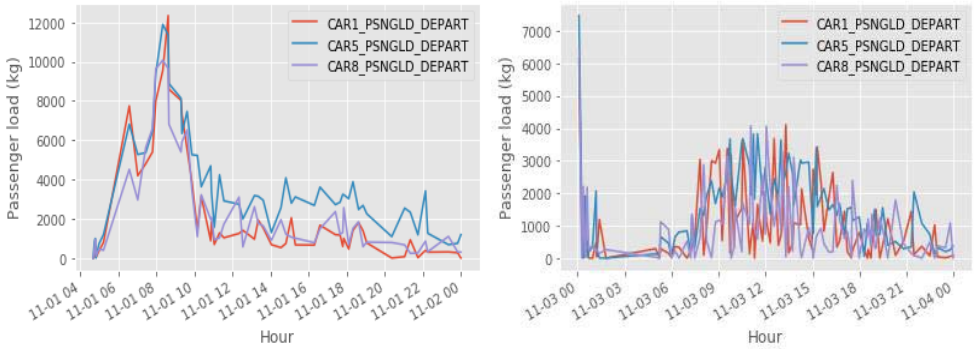
Of all the train models, the Waratah is the most modern and the most commonly used, accounting for 60% of the fleet [30]. Importantly, the Waratah models are the only trains equipped with an occupancy weighing system, so these are the only trains capable of providing real-time information about carriage load.

III. Methodology

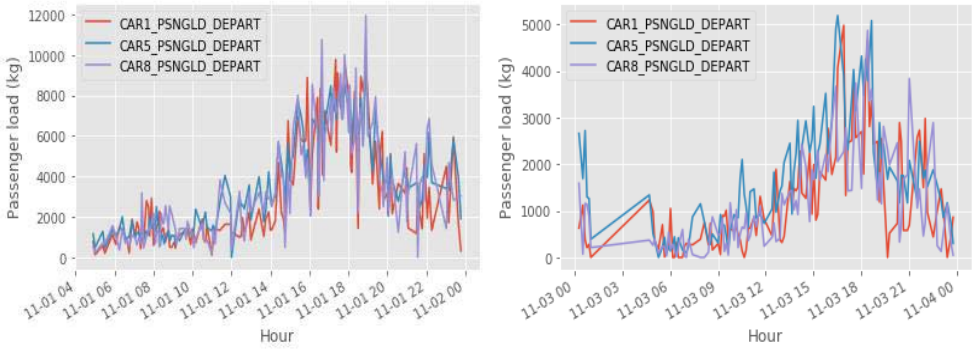
A. Feature Selection

There is detailed information on the Open Data Hub for hundreds of different features pertaining to trains and train networks, so selecting the most useful features for predicting carriage load was an important task. Fig. 1 shows an exemplary distribution of loads patterns during different time periods with different locations. Note that each train has eight carriages, numbered 1 to 8 from head to end. Recordings of both the arrival and departure weights for all carriages are available; we have illustrated the departure weights of Carriages 1, 5, and 8 as examples. The panel on the left in Fig. 1a shows data for trains departing platform RD05 of Redfern station on Fri 1 Nov 2019, while the panel on the right shows the same platform on Sun 3 Nov 2019. Redfern is the first station outside Sydney's central business district for many of the lines in the network. The Friday data has a distinct peak phenomenon. At around 8:00 am, the weight difference between the three carriages is minimal but, by 10:00 am, Carriage 5, near the generally-preferred middle of the train, is consistently carrying more weight than the other carriages. By contrast, the Sunday data has no clear peak, except for a massive passenger influx at midnight, which is explained by laws surrounding liquor licensing in Sydney's inner city. Fig. 1b shows the same cases for Central station, the point of origin for those trains. On the left, the peak time shifts from morning to afternoon, when people are leaving work. Notably, the difference in flow between Friday and Sunday is not very prominent; it is just that the Friday numbers are much higher. From the results of an exploratory data analysis and previous related work [46], [51], we found many factors influence passenger loads:

- Time includes the time of day, day of the week, and month of the year day. Each of these is a determining



(a) Origin station: Central; Destination station: Schofields; Current station: Redfern; Platform: RD05; Date: 2019-11-01 and 2019-11-03.



(b) Origin station: Central; Destination station: Schofields; Current station: Central (the origin station); Platform: CE18; Date: 2019-11-01 and 2019-11-03.

Fig. 1. A demonstration of the passenger load patterns during different time periods with different locations.

factor in peak versus non-peak time. Public holidays and school breaks are also influential.

- Spatial factors pertain to the type of district in which the station is located. For example, the proportion of residential, commercial, and industrial zoning in the district; whether it is a predominantly entertainment or university precinct; etc.
- Transport network factors include permanent variations to the train timetable. For example, in response to local population increases or the development of a new public transport interchange hub, and similar.
- Noisy or ambiguous sensor data may also be an issue. For example, a null value recorded by a sensor that monitors the flow of passengers boarding and alighting a carriage could mean no passengers were present, a missing value, or a sensor malfunction.
- External factors include extreme weather events or special events like music concerts, the football grand finale, New Year's Eve, and such. Collecting this information is arduous and, even when fully represented, will only amount to minority classes in the data. Hence, we opted to ignore these factors.

However, our method computes training data in a stream manner, so there will be very little training data compared to a batch method. The decrease in the number of samples required us to find more effective features to improve the prediction accuracy. Hence, we compared the data on all services in Mar 2019 with a similarity analysis for finding more key features. For each train service, we calculated the similarity between the loads of each of the eight carriages given the same platforms and the same planned arrive time. Our procedure was to systematically work through each day of the month with a one-to-one comparison, i.e., comparing the trains on 1 Mar with 2 Mar, then with 3 Mar, 4 Mar, and so on, followed by all the trains on 2 Mar with those on 3 Mar, 4 Mar, etc. If there was no train for a given platform and planned arrive time on a day, we simply moved on that day. In our comparison, the similarity was measured in terms of the Jensen-Shannon divergence (JS) [35], defined as follows:

$$JS(P_1 \| P_2) = \frac{1}{2} KL\left(P_1 \| \frac{P_1 + P_2}{2}\right) + \frac{1}{2} KL\left(P_2 \| \frac{P_1 + P_2}{2}\right) \tag{1}$$

where KL is the Kullback-Leibler divergence [36] and $P_{\mathrm{i}} = [p_i^1, p_i^2, p_i^3, p_i^4, p_i^5, p_i^6, p_i^7, p_i^8]$. p_i^j was calculated by

$$p_i^j = \frac{W_j}{\sum_{j=1}^{8} W_j} \tag{2}$$

where W_j is the load (weight) of the jth carriage.

In analysis, we find that about 98% of JS value were located in range [0.0–0.2]. This means almost of the train services, which with the same origin-destination (O-D), platform, and planned arrival time, have similar carriage load patterns. As a result, the pattern of train carriage load can be learnt from historical data. Hence, we devised an additional feature:

- *contextual moving average*, which indicates historically an increasing or decreasing of carriage load consistent

with trains running in the same time slice on the same type of day.

Next, for each train service, we calculated the similarity between the loads of each of the eight carriages given the same platforms but different planned arrive times. Two interesting points are worth mentioning: 1) when the time interval between two services is less than 30min, we find that about 95% of the JS value were located in range of [0.0, 0.2]. Especially, when the time interval between two services is less than 10min, we find that about 99% all of the JS value were located in range of [0.0, 0.1]. In contrast, when greater than 30min, the JS value suggests that the carriage load pattern of most two-train services are not similar. 2) However, if two trains are an adjacent service, i.e., one train arrives and must wait until another train leaves, most of the JS value were located in a range of [0.0, 0.2] even for the time-interval of the two train services over 30min. Based on these two findings, we devised two additional features:

- *last stopped*, which reflects the information on carriage load (from a Waratah train) of last service.
- *passenger flow patterns*, which reflects the most recent information on carriage load (from a Waratah train). For this feature, we summed up the carriage load of all trains arriving at a platform in 30-minute time slices throughout the day to create a multiple time series for time-slice, e.g., {10:00–10:30 am Mon, 10:00–10:30 am Tue, . . .} and {10:30–11:00 am Mon, 10:30–11:00 am Tue, . . .}. From a macro perspective, and without considering service disruptions, passenger flows should be very stable across the time slices. That is, passenger flows between 10:00 and 10:30 am should be about the same on any given weekday; the same goes for weekends.

These three features are one of the key differences between our method and other methods.

B. Fuzzy Time Matching

Our two novel features, i.e., contextual moving average and last stop, have the potential to be very powerful, in practical terms, but both are very difficult to calculate. For a start, open data does not paint a precise picture over time. For instance, the same train service may depart Central Station at 10:30 am one morning but at 10:35 am the next. In low-frequency train service, i.e., the time-interval between two services over 15min, we can easily identify they are the same train service. However, in high-frequency train services, i.e., the time-interval between two services is less than 15min, even some only 3min, we cannot identify they are the same train service. This makes it hard to match today's information with the correct historical data.

Therefore, in high-frequency train services, our solution to this problem is a fuzzy time-matching method. Our intuition is therefore to transform a precise time point into a time interval and then match the trip information according to a weighting calculated based on the size of the overlapped time interval. As an example, and using Fig. 2 to demonstrate, assume we have a service that left Central Station at 10:30 am on Thursday, 10.31 am on Monday, 10.34 am on Tuesday, and 10.28 am on Wednesday. Based on the above analysis, we define the fuzzy time interval as 10 mins, so all services departing Central between 10:30 am +/− 5 mins will be treated as strongly correlated trips; the closer to 10:30 am, the stronger the correlation. The *contextual moving average* is then calculated by multiplying the carriage load of each service by a normalized weight according to its degree of membership in the window. Fig. 3 shows how to calculate the normalized weight of one carriage. Assume the ith data sample is obtained. The first step is a triangular fuzzy number [31], which is used to compute the membership $\mu_{i,j}^{ma}$, where j is $\{1, 2, 3, \ldots, 7\}$, based on the overlapping areas between the current and historical trips (previous seven days). The reason we don't use the Type-2 fuzzy model [31], [49] is the type-2 fuzzy model has a process of reduction, which has the weakness of low efficiency, and thus it is difficult to use for online identification and control compared with the type-1 fuzzy model.

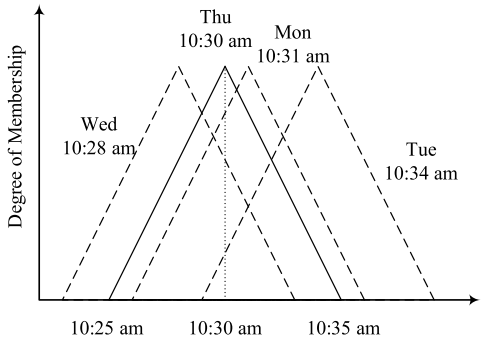


Fig. 2. A demonstration of fuzzy time interval matching for calculating the *contextual moving average*.

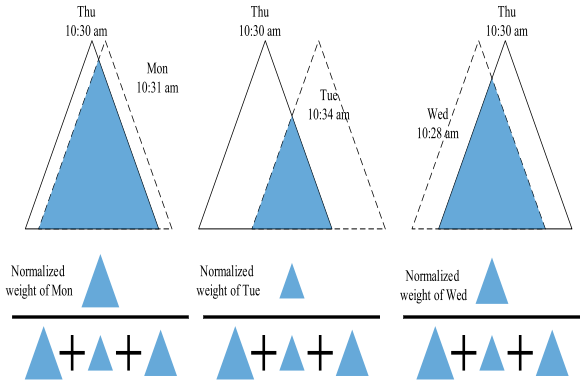


Fig. 3. Calculating the normalized weight.

Then the weights of one carriage are normalized based on the sum of all the overlapping areas:

$$weight_i = \sum_{j=1}^{7} \mu_{i,j}^{cma} \cdot weight_j \tag{3}$$

and

$$\mu_i^{cma} = \bigwedge_{j=1}^{7} \mu_{i,j}^{cma} \tag{4}$$

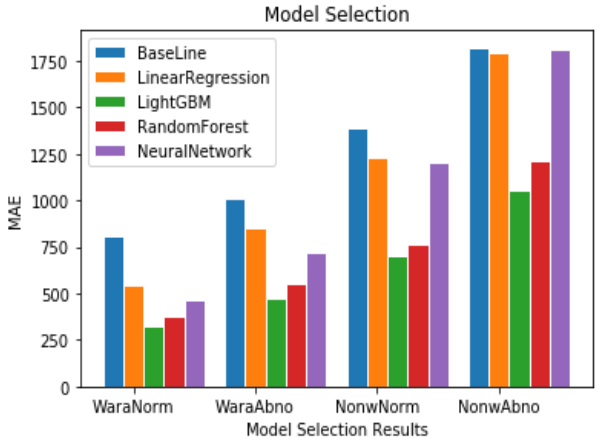


Fig. 4. Model selection results for the four tasks. WaraNorm: Waratah trains operating without disruption; WaraAbno: Waratah trains operating with disruptions; NonwNorm: non-Waratah trains operating without disruption; NonwAbno: non-Waratah trains operate with disruptions.

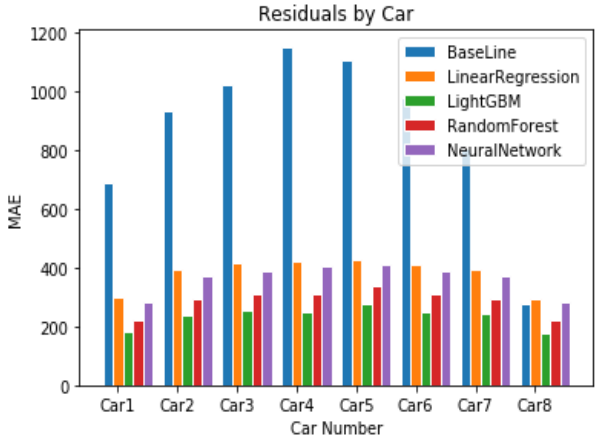


Fig. 5. MAEs for each carriage.

where μ_i^{cma} is the degree of the *contextual moving average* feature.

In the training stage, each data sample has a different importance. The extent of uncertainty associated with each data sample determines how important it is to the training. Hence, the formula for weighting the importance of each sample according to its membership in the fuzzy interval is:

$$\mu_i = \mu_i^{ls} \cap \mu_i^{ma} \tag{5}$$

where μ_i^{ls} is the degree of the *last stopped* feature, which is also calculated by our proposed fuzzy time matching method.

C. Multi-Stream Learning

Because each train has eight carriages, the learning objective is to minimize the generalization error of eight outputs. However, unlike the multiple regression problem [37], the outputs in our scenario do not come from the same input. Rather, each output has its own corresponding input. Hence, our problem is a multi-stream learning problem [38]. In our exploratory data analysis [46], we found that some carriages have similar properties. For example, Carriages 2 and 7 have similar passenger flows, as do 3 and 6, and 4 and 5, as shown

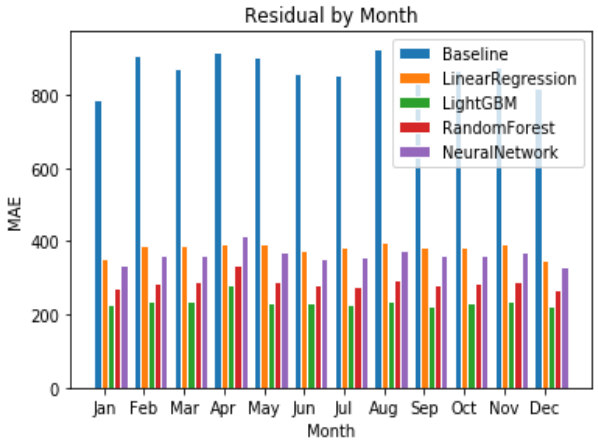


Fig. 6. MAEs by month.

in Fig. 6. It is well-known that the more training data a model fits, the lower its generalization error, and so the basic idea of our multi-stream learning method is to merge the data in similar data streams to increase the size of the training set.

However, data streams inevitably suffer from concept drift. In the case of rail networks, concept drift happens at multiple levels – from one hour to the next, as with peak times; seasonally, as with summer or school holidays; long-term, as regions change with population growth or demographic shifts in a district; or suddenly, like with the completion of a new real estate development. However, the implications of concept drift in multiple regression have not yet been fully considered. In our case, we must consider the possibility that there could be different degrees of drift between each carriage on the same train. For these reasons, we have leveraged the modularity property of data streams to evaluate the similarity between each carriage load in terms of passenger load distribution and degree of drift. Highly similar carriage load are then built as a modularity. For example, assume the passenger load in Carriage 1 of a non-peak hour service follows a Gaussian distribution of $N(\mu, \sigma^2)$, where $N(4000, 90^2)$, and the passenger load of Carriage 8 follows the distribution $N(4300, 80^2)$. Once peak hour arrives, μ increases to 12000 for Carriage 1 and to 13000 for Carriage 8. By these metrics, the passenger load distribution and degree of drift for μ are similar. Therefore, we can safely combine Carriage 1 and Carriage 8 into a modularity {Car1, Car8} and a train one prediction model for both to boost the overall performance. This idea of leveraging modularity is a key advancement over existing multiple regression methods [50].

D. Model Design

As mentioned, most stations in the Sydney rail network are close together and, at Waratah train speeds, the average time between stops is 3–5 minutes. Further, trains only stop at stations to pick up and set down for around 90 seconds. As such, the real-time performance requirements on the algorithm are very strict. This makes a shallow learning model our first choice; a deep leaning model would likely struggle. LightGBM is based on Gradient Boosting Decision Tree (GBDT) [41], like GBDT, it performs well with many prediction tasks,

but it is a much faster and more memory-efficient model. More importantly, extending it to accommodate incremental learning can be done relatively easily, making it well suited for streaming data. We chose the LightGBM decision tree [39], [40] model proposed by Microsoft in 2017 [27] to state the modularity.

Further, although predicting the carriage load is not a multiple regression problem, learning different modularities is, e.g., the modularity of {Car 1, Car8}. However, LightGBM cannot predict multiple outputs at the same time, so this aspect of the model also needs to be modified.

Assume $D = \left\{(X_i, Y_i)_{i=1}^{n}\right\}$ be a dataset of a modularity with n samples, where $X \in \mathbb{R}^m$ is an m dimensional input and $Y \in \mathbb{R}^d$ is a d dimension output. In other words, the modularity consists of d data streams. Let f be the function of a decision tree which maps X into the output space, and obtained the function f according to the main idea of GBDT-MO [48]. Finally, we get the flowchart of our system and prediction model. The data features are extracted and filtered as per the descriptions in Section III.A, and our proposed LightGBM makes the predictions. Due to the open data being updated every few minutes for our case, the specific procedure is as follows:

(1) The open data is downloaded from TfNSW and spliced together to form a batch of training data.
(2) The eigenvalues and weights for each raw sample are derived using our fuzzy time matching method, and deal with samples that are saved into an offline database as training data.
(3) The weighted data is then input into LightGBM and the model is retrained.
(4) Predictions are generated from the updated model.

IV. Experiments

In this section, we have evaluated our carriage load prediction system by analysing the performance of each component. All experiments were conducted in Python v3.7 for Windows 10, running on a PC with an Intel Core i5 processor (2.40 GHz) and 16-GB RAM.

A. Experimental Settings

Being a commercial project and one that involves location details of the general public, only the results of experiments conducted with publicly-available data are provided in this paper. The dataset used comprised the operations records for Sydney Trains services for Nov and Dec 2019. These data were divided into four groups: 116,112 sampled instances of Waratah trains with a normal operating status (WaraNorm); 11,864 Waratah trains with abnormal status (WaraAbno), i.e., delayed, cancelled, or disrupted services; 174,440 for normal non-Waratah trains (Non-WNorm); and 17,552 for abnormal non-Waratah trains (Non-WAbno). About 80 percent are training data and about 20 percent are test data.

For the evaluation metric, we used the mean absolute error (MAE) of the carriage load, calculated as

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{8}\left|y_{ij} - \hat{y}_{ij}\right| \tag{6}$$

where y_{ij} represents the real value of the jth carriage of the ith sample, and $\hat{y}_{ij}$ represents the predicted value of the jth carriage of the ith sample.

B. Feature Selection

Our first set of experiments was designed to evaluate the contribution of the base features and our newly-designed features, *contextual moving average* and *last stopped*. Our assumption is that, if these two indicators are useful, prediction accuracy should improve when they are used. In addition, we are also concerned with the weight given to each feature and how those weights affect the experimental results. LightGBM has a built-in plot function that shows the importance of each feature exactly. Furthermore, although LightGBM has many parameters, only four parameters needed to be optimized to obtain a satisfactory result for our purposes: $Objectiv$m, m_{leaves}, $learning_{rate}$, and $num_{estimators}$,. Hence, in this experiment, we chose the base LightGBM as the prediction model and MAE as the evaluation metric.

Following the guidelines for parameter selection that were proposed in [27], we conducted a grid search [42], which resulted in $Objective \leftarrow$ Poisson; $num_{leaves} \leftarrow 20$, $learning_{rate} \leftarrow 0.05$; $num_{estimators} \leftarrow 1000$.

As for our proposed features, the *contextual moving averages* were calculated based on the average departure loads from the previous week. *Last stopped* values were taken from a sample search based on the timestamp closest to the current train. Table I summarizes the average results with different features given 10-fold cross-validation.

We calculated the feature importance of each feature in terms of *total gain* [43], and selected the top 10 features, as shown in Table II. Furthermore, an interesting finding regarding feature importance is that, the important features patterns of some carriages are very similar. For example, the *contextual moving average* was the most important feature for Carriage 1 and second-most important for Carriage 8.

C. Fuzzy Time Matching

From the above experimental results, we know the *contextual moving average* is useful for improving prediction accuracy. Therefore, if too many contextual moving averages are missing, performance will suffer. Hence, our next series of experiments was designed to analyze how well different matching methods – both exact and fuzzy – offset the impact of this kind of uncertainty. The results were assessed using a miss ratio, calculated using the equation:

$$miss\ ratio = \frac{the\ nubmer\ of\ CMA_{miss}}{N} \tag{7}$$

where CMA_{miss} represents the value of *contextual moving averages* as 0, and N represents the number of training samples.

The four matching methods compared were: 1) Accurate matching (platform) (AMP), where two trains are only matched if they run at exactly the same time and from the same platform; 2) Accurate matching (station) (AMS), where two trains are matched if they run at exactly the same time

TABLE I
MAEs OF LIGHTGBM MODEL WITH DIFFERENT FEATURES

	WaraNorm	WaraAbno	NonwNorm	NonwAbno	Average
BasicFeatures	321.29	469.86	701.64	1057.39	574.54
HistorcialMean	317.12	498.57	601.88	964.48	514.62
PassengerFlowPatterns	311.46	467.31	612.02	998.55	554.68
LastAvailableWaratah	305.75	468.51	624.23	981.62	522.48
Mixture	**299.68**	**489.49**	**550.33**	**882.1**	**475.37**

TABLE II
TOP 10 IMPORTANT FEATURES

Abbreviation	Denotes
CAR1_PSNGLD_DEPART_MEAN	as the mean departure load in the previous 7 days of Carriage 1
CAR8_PSNGLD_DEPART_MEAN	as above for Carriage 8
CAR1_PSNGLD_ARRIVE	as the arrival load of Carriage 1
LAST_CAR7_PSNGLD_DEPART	as the departure load of Carriage 7 of the last Waratah model train to have departed the given station
LAST_CAR2_PSNGLD_DEPART	as above for Carriage 2
LAST_CAR8_PSNGLD_DEPART	as above for Carriage 8
STATION_NAME	as the name of the station the train is departing from
PERIODS	the period the train is running in (as at its departure time), e.g., peak time
CAR7_PSNGLD_DEPART_MEAN	as above for Carriage 7
CAR2_PSNGLD_DEPART_MEAN	as above for Carriage 2

TABLE III
MISSING RATIOS WITH DIFFERENT MATCHING SCHEMES

	Platform	Station
Accurate Matching	28%	22%
Fuzzy Matching	19%	11%

from the same station; 3) Fuzzy Matching (platform) (FMP) where the two trains can run at slightly different times but leave from the same platform; and 4) Fuzzy Matching (station) (FMS), where the trains can run at slightly different times and only need to depart from the same station. The *miss ratio* of each matching method is shown in Table III.

As we can see, the fourth method, where the trains neither ran at exactly the same time nor from the same platform, gave us the lowest miss ratio. This demonstrates that *contextual moving averages* are very easily undermined by uncertainty and noise. Hence, an accurate matching method, which leads to many missing values, will lead to substantially lower accuracy. By contrast, the fuzzy matching methods produced better accuracy without the need to manipulate, impute, or infer values.

Using the same experimental settings and evaluation scheme as in the first set of experiments, we tested the four methods with LightGBM and 10-fold cross-validation, and assessed the results in terms of MAE. Table IV shows the results. Again, accuracy improved significantly with the two fuzzy matching methods, especially when relaxed to a station-only match. However, we still felt that too many instances had a *contextual moving average* of 0, simply because trains do not run at exactly the same time every day. Yet when we combined station-level fuzzy match with our proposed fuzzy weighting method, accuracy considerably improved. This result supports our assumption that a sample with relatively uncertain information is less important to the learning process and can actually significantly negatively impact the accuracy of the model.

To further evaluate the merits of fuzzy learning, we also compared each pre-processing method in terms of mean absolute percentage error (MAPE):

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - y_i'}{y_i}\right| \tag{8}$$

where y_i is the actual value, y_i' is the forest value, and N is the number of training samples.

The comparative results, shown in Table V, confirm that fuzzy matching + fuzzy weighting yields the highest accuracy. Overall, we find that applying fuzzy learning at the

TABLE IV
MAEs FOR THE LIGHTGBM MODEL WITH DIFFERENT MATCHING SCHEMES

Abbrev	Method	WaraNorm	WaraAbno	NonwNorm	NonwAbno	Average
AMP	Accurate Matching (platform)	342.34	586.91	630.18	955.91	628.84
AMS	Accurate Matching (station)	299.68	489.49	550.33	882.10	475.37
FMP	Fuzzy Matching (platform)	292.45	477.67	545.25	863.87	468.48
FMS	Fuzzy Matching (station)	279.22	469.34	541.08	847.16	465.01
FMW	Fuzzy Matching (station) + Fuzzy Weighting	255.75	420.51	526.12	806.70	445.48

TABLE V
MAPE BY EACH PREPROCESSING METHOD

	AMP	AMS	FMP	FMS	FMW
MAPE	68.4	60.2	58.7	50.3	42.8

pre-processing stage significantly reduces the impact of noisy and uncertain samples.

D. Incremental Learning

Our next batch of experiments was designed to evaluate LightGBM with our incremental learning scheme. Here, performance was assessed by MAE in comparison to four other methods. These were:

1. Last Value (baseline): The simplest forecasting method is to forward the last observed load on the train to the next one at the same station. More specifically, this model predicts the load of a train carriage will be the same when it departs as it was when it arrived.
2. Linear Regression: This is the most intuitive regressor solution. It is able to optimize the weight of each feature and link the features to a target variable function. In our case, the features concern the train's status, and the target variable is the carriage load.
3. Random Forest: This regression model produces a prediction model in the form of an ensemble of weak decision trees, where the prediction is given by bagging. This model is like gradient boosting, but it does not iteratively update the cost for each base learner.
4. Neural Network: Our final model is a universal approximator that can capture any form of relationship in data [44], [45]. Neither a recurrent neural network nor a convolutional neural network is applicable to this study. Therefore, we deployed a simple feed-forward neural network to model the relationships between the input and output.

These four methods were chosen because they are the most-used methods for forecasting problems and can easily be extended to incremental learning for streaming data.

The parameters for each of the methods were tuned using search method [42] and were as follows:

- Last Value (baseline): no specific settings. For non-Waratah trains, we used the mean of the training set as the prediction.
- Linear Regression: the SciKit-Learn implementation with intercept fitting and normalization.
- LightGBM: the same as those used in Section B.
- RandomForest: the SciKit-Learn implementation with $num_{estimators} \leftarrow 20$; $\max\_depth \leftarrow 10$.
- NeuralNetwork: the SciKit-Learn MLPRegressor implementation with a $hidden\_layer\_size \leftarrow 32$; a *ReLu* activation function; the $Adam$ optimizer; and an $adaptive$ learning rate.

TABLE VI
RUNNING TIMES OF EACH METHOD

	Linear Regression	Random Forest	Neural Network	LightGBM
Times (seconds)	0.21	13.97	58.73	0.57

The results are shown in Figs. 4–6. From Fig. 4. we can see that LightGBM performed best with all four data groups, i.e., WaraNorm, WaraAbno, NonwNorm, NonwAbno, followed by RandomForest, NeuralNetwork, LinearRegression and then the baseline. Fig. 5 shows the MAEs for the different carriages where, again, we see LightGBM with the best performance every time, particularly for Carriages 1 and 8. From this result, we extended our dataset to include all of 2019, as shown in Fig. 6. The results did not change; LightGBM still delivered the best performance with Random Forest following in second. These results suggest that machine learning models, and, more particularly, ensemble learning approaches such as LightGBM and Random Forest, are better at load forecasting. We believe this is because they use a combination of short- and long-term features that translate influencing factors.

Next, we compared the learning time for each method. The results appear in Table VI. Linear Regression was the fastest method, followed by incremental LightGBM. However, what is lost in speed is made up for in accuracy, presenting a choice for users. We further compared the running time of our incremental LightGBM with the standard batched LightGBM and found that batched LightGBM took 12.5 secs compared to the 0.45 of our incremental variants. This is a remarkable difference that makes LightGBM viable as a real-time forecasting model.

The last point to note is on the deep learning method, which performed very poorly. Given that we only used a very shallow

TABLE VII
MULTI-STREAM LEARNING WITH DIFFERENT SAMPLE SIZES

Sample Size	25000		20000		15000		10000	
Learning Method	MulOutput	MulStream	MulOutput	MulStream	MulOutput	MulStream	MulOutput	MulStream
Car 1-8	386.90	382.14	392.52	386.15	395.32	386.98	397.46	391.44
Car 2-7	492.63	486.58	498.25	489.57	501.72	493.28	510.84	502.23
Car 3-6	513.15	505.01	518.86	510.44	520.38	512.51	531.32	518.40
Car 4-5	487.22	470.69	492.00	471.34	496.92	472.85	507.69	481.03
Average	469.98	461.10	475.41	464.37	478.58	466.40	486.83	473.28
Improvement	8.87		11.03		12.18		13.55	

The results show a higher average improvement when using multi-stream learning than a multi-output regressor with one regressor per target. This is a simple strategy for extending LightGMB to support multi-target regression.

neural network structure, future study on this regressor should be undertaken before passing a final judgment. That said, we are confident in concluding that incremental LightGBM holds the most promise as a final solution for our system.

E. Multi-Stream Learning

As explained in Section III.D, our multi-steam learning strategy was designed to fit as much data as possible while maximizing the generalization ability using modularity. Hence, in this experiment, we tested two things: first, the treatment of the problem with multi-stream learning compared to multi-output learning; and second, the accuracy when combining Carriages 1 and 8 versus not combining them.

Since LightGBM does not support multi-output regression, we adopted the SciKit-Learn local multi-output strategy – fitting one regressor per target and building an independent regressor for each carriage with the same model configuration, as introduced in Section III.D. The experimental settings were slightly different than for the previous experiments, but the evaluation metric was still MAE.

For multi-stream learning, we concatenated the data of Carriages 1 and 8 vertically and removed the duplicate carriage load information. We applied the same multi-stream learning strategy to Carriages 2–7, 3–6, and 4–5 to result in four independent regressors instead of eight but doubled the training data for each. Since the fundamental idea of multi-stream learning is to increase the size of the training set but reduce the generalization errors, we evaluated this strategy at different sample sizes.

In the next experiments, the prediction error is represented by the residual of weights (kg), i.e., the gap between predicted weights of carriages and the real weights of carriages. Table VII shows the results of these experiments. What is clear is that multi-stream learning always outperforms multi-output learning, and the average improvement increases as the size of the training set decreases. With a sample size of 25,000, the improvement of gap was 8.87kg; for 20,000 the improvement of gap was 11.03 kg; 12.18kg with 15,000; and 13.55kg with 10,000. This finding strongly supports our assumption that multi-stream learning can increase accuracy by boosting the amount of training data and at the same time reduce generalization errors.

V. CONCLUSION AND FURTHER STUDY

In this paper, we presented a real-time machine learning application to help Sydney Trains improve the efficiency of its city rail network by forecasting a train's carriage load at the time it departs a platform. The analytics is based on open data, which is becoming more accessible for smart transport solutions in Australia, Europe, and other parts of the world. The first step is to extract useful features from the data and process them with a series of fuzzy learning methods to reduce the impact of noise and uncertainty. Unlike many other algorithms, these mechanisms allow us to fully consider historical data. As new data arrives, small batched data are fed into LightGBM. However, rather than using a traditional batch-based learning schemes, we introduced a novel incremental learning scheme to allow for real-time forecasting. Called multi-stream learning, the strategy is to merge data streams with similar patterns of passenger loading, reducing the number of models that need to be trained. The result is high accuracy in the face of concept drift, and also a high generalization ability.

In future work, we will continue to work with Sydney Trains to extend current prediction systems to a bus replacement situation. We also plan to focus on improving the performance of multi-stream learning by redesigning the merge indicator to allow control over training sample sizes.

REFERENCES

[1] J. Roos, S. Bonnevay, and G. Gavin, "Short-term urban rail passenger flow forecasting: A dynamic Bayesian network approach," in *Proc. IEEE Int. Conf. Mach. Learn. Appl.*, Dec. 2017, pp. 1034–1039.
[2] E. Pekel and S. S. Kara, "Passenger flow prediction based on newly adopted algorithms," *Appl. Artif. Intell.*, vol. 31, no. 1, pp. 64–79, 2017.
[3] H. Yu, J. Lu, and G. Zhang, "MORStreaming: A multi-output regression system for streaming data," *IEEE Trans. Syst. Man, Cybern. Syst.*, early access, Sep. 1, 2020, doi: 10.1109/TSMC.2021.3102978.
[4] H. Yu, A. Liu, B. Wang, R. Li, G. Zhang, and J. Lu, "Real-time decision making for train carriage load prediction via multi-stream learning," in *Advances in Artificial Intelligence*, (Lecture Notes in Computer Science), vol. 12576, M. Gallagher, N. Moustafa, and E. Lakshika, Eds. Cham, Switzerland: Springer, 2020, doi: 10.1007/978-3-030-64984-5_3.
[5] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining Kohonen maps with ARIMA time series models to forecast traffic flow," *Transp. Res. C, Emerg. Technol.*, vol. 4, no. 5, pp. 307–318, Oct. 1996.
[6] Y. Wei and M.-C. Chen, "Forecasting the short-term metro passenger flow with empirical mode decomposition and neural networks," *Transp. Res. C, Emerg. Technol.*, vol. 21, no. 1, pp. 148–162, 2012.
[7] S. Clark, "Traffic prediction using multivariate nonparametric regression," *J. Transp. Eng.*, vol. 129, no. 2, pp. 161–168, Mar. 2003.
[8] J. Tang, X. Chen, Z. Hu, F. Zong, C. Han, and L. Li, "Traffic flow prediction based on combination of support vector machine and data denoising schemes," *Phys. A, Stat. Mech. Appl.*, vol. 534, Nov. 2019, Art. no. 120642.
[9] X. Feng, X. Ling, H. Zheng, Z. Chen, and Y. Xu, "Adaptive multi-kernel SVM with spatial-temporal correlation for short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 6, pp. 2001–2013, Jun. 2019.

[10] H.-F. Yang, T. S. Dillon, and Y.-P. P. Chen, "Optimized structure of the traffic flow forecasting model with a deep learning approach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2371–2381, Oct. 2017.

[11] J. Mackenzie, J. F. Roddick, and R. Zito, "An evaluation of HTM and LSTM for short-term arterial traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 5, pp. 1847–1857, May 2019.

[12] X. Ma, Z. Tao, Y. Wang, H. Yu, and Y. Wang, "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data," *Transp. Res. C, Emerg. Technol.*, vol. 54, pp. 187–197, May 2015.

[13] Y. P. Liu, H. F. Zheng, and X. X. Feng, "Short-term traffic flow prediction with Conv-LSTM," in *Proc. Int. Conf. Wireless Commun. Signal Process.*, 2017, pp. 1–6.

[14] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[15] J. Xuan, J. Lu, and G. Zhang, "Bayesian nonparametric unsupervised concept drift detection for data stream mining," *ACM Trans. Intell. Syst. Technol.*, vol. 12, no. 1, pp. 1–22, Feb. 2021.

[16] A. Liu, J. Lu, and G. Zhang, "Concept drift detection via equal intensity k-means space partitioning," *IEEE Trans. Cybern.*, early access, Apr. 22, 2020, doi: 10.1109/TCYB.2020.2983962.

[17] A. Liu, J. Lu, and G. Zhang, "Diverse instance-weighting ensemble based on region drift disagreement for concept drift adaptation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 1–15, Jan. 2020.

[18] H. Yu, J. Lu, and G. Zhang, "An online robust support vector regression for data streams," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 1–14, Jan. 2020.

[19] H. Yu, J. Lu, and G. Zhang, "Continuous support vector regression for nonstationary streaming data," *IEEE Trans. Cybern.*, early access, Sep. 10, 2020, doi: 10.1109/TCYB.2020.3015266.

[20] M. Shuai, K. Xie, W. Pu, G. Song, and X. Ma, "An online approach based on locally weighted learning for short-term traffic flow prediction," in *Proc. 16th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst. (GIS)*, 2008, pp. 383–386.

[21] H. Yu, J. Lu, and G. Zhang, "Online topology learning by a Gaussian membership-based self-organizing incremental neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 1–15, Nov. 2019.

[22] D. Desiraju, T. Chantem, and K. Heaslip, "Minimizing the disruption of traffic flow of automated vehicles during lane changes," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1249–1258, Jun. 2015.

[23] M. Aslani, M. S. Mesgari, and M. Wiering, "Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events," *Transp. Res. C, Emerg. Technol.*, vol. 85, pp. 732–752, Dec. 2017.

[24] W. Chen *et al.*, "A novel fuzzy deep-learning approach to traffic flow prediction with uncertain spatial–temporal data features," *Future Gener. Comput. Syst.*, vol. 89, pp. 78–88, Dec. 2018.

[25] S. Deng, S. Jia, and J. Chen, "Exploring spatial–temporal relations via deep convolutional neural networks for traffic flow prediction with incomplete data," *Appl. Soft Comput.*, vol. 78, pp. 712–721, May 2019.

[26] Y. Tian, K. Zhang, J. Li, X. Lin, and B. Yang, "LSTM-based traffic flow prediction with missing data," *Neurocomputing*, vol. 318, pp. 297–305, Nov. 2018.

[27] G. Ke *et al.*, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. Adv. Neural Inf. Process. Syst.*, Dec. 2017, pp. 3147–3155.

[28] TfNSW. (Jul. 2020). *Tfnsw Open Data Hub and Developer Portal*. [Online]. Available: https://opendata.transport.nsw.gov.au/user-guide

[29] TrainWiki. *Sydney Trains*. Accessed: Jul. 2020. [Online]. Available: https://nswtrains.fandom.com/wiki/Sydney_Trains

[30] TfNSW. *Waratah Trains*. Accessed: Jul. 2020. [Online]. Available: https://transportnsw.info/travel-info/ways-to-get-around/train/fleet-facilities/waratah-trains

[31] K. K. Yen, S. Ghoshray, and G. Roig, "A linear regression model using triangular fuzzy number coefficients," *Fuzzy Sets Syst.*, vol. 106, no. 2, pp. 167–177, 1999.

[32] H. Borchani, G. Varando, C. Bielza, and P. Larrañaga, "A survey on multi-output regression," *Data Mining Knowl. Discovery*, vol. 5, no. 5, pp. 216–233, 2015.

[33] H. Yu, J. Lu, and G. Zhang, "Topology learning-based fuzzy random neural network for streaming data Regression," *IEEE Trans. Fuzzy Syst.*, early access, Nov. 20, 2020, doi: 10.1109/TFUZZ.2020.3039681.

[34] S. M. Mastelini *et al.*, "Multi-output tree chaining: An interpretative modelling and lightweight multi-target approach," *J. Signal Process. Syst.*, vol. 91, no. 2, pp. 191–215, Feb. 2019.

[35] B. Fuglede and F. Topsoe, "Jensen-Shannon divergence and Hilbert space embedding," in *Proc. Int. Symp. Inf. Theory (ISIT)*, 2004, p. 30.

[36] S. Kullback and R. A. Leibler, "On information and sufficiency," *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.

[37] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: A multi-output streaming framework," *J. Mach. Learn. Res.*, vol. 19, no. 72, pp. 1–5, 2018.

[38] A. G. Helmy, A. R. Hedayat, and N. Al-Dhahir, "Robust weighted sum-rate maximization for the multi-stream MIMO interference channel with sparse equalization," *IEEE Trans. Commun.*, vol. 63, no. 10, pp. 3645–3659, Oct. 2015.

[39] Z. Mei, F. Xiang, and L. Zhen-hui, "Short-term traffic flow prediction based on combination model of xgboost-lightgbm," in *Proc. Int. Conf. Sensor Netw. Signal Process. (SNSP)*, Oct. 2018, pp. 322–327.

[40] Y. Ju, G. Sun, Q. Chen, M. Zhang, H. Zhu, and M. U. Rehman, "A model combining convolutional neural network and LightGBM algorithm for ultra-short-term wind power forecasting," *IEEE Access*, vol. 7, pp. 28309–28318, 2019.

[41] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[42] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2012.

[43] R. L. De Mántaras, "A distance-based attribute selection measure for decision tree induction," *Mach. Learn.*, vol. 6, no. 1, pp. 81–92, 1991.

[44] B. Wang *et al.*, "Deep uncertainty quantification: A machine learning approach for weather forecasting," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 2087–2095.

[45] B. Wang, Z. Yan, J. Lu, G. Zhang, and T. Li, "Deep multi-task learning for air quality prediction," in *Neural Information Processing* (Lecture Notes in Computer Science), vol. 11305, L. Cheng, A. Leung, and S. Ozawa, Eds. Cham, Switzerland: Springer, 2018, doi: 10.1007/978-3-030-04221-9_9.

[46] H.-T. Yu, C.-J. Jiang, R.-D. Xiao, H.-O. Liu, and W. Lv, "Passenger flow prediction for new line using region dividing and fuzzy boundary processing," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 5, pp. 994–1007, May 2019.

[47] H. Yu, J. Lu, and G. Zhang, "Topology learning-based fuzzy random neural network for streaming data regression," *IEEE Trans. Fuzzy Syst.*, early access, Nov. 20, 2020, doi: 10.1109/TFUZZ.2020.3039681.

[48] Z. Zhang and C. Jung, "GBDT-MO: Gradient-boosted decision trees for multiple outputs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 7, pp. 3156–3167, Jul. 2021.

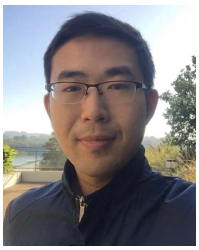
[49] J. M. Mendel, "Type-2 fuzzy sets and systems: An overview," *IEEE Comput. Intell. Mag.*, vol. 2, no. 1, pp. 20–29, Feb. 2007.

[50] J. Lu, H. Zuo, and G. Zhang, "Fuzzy multiple-source transfer learning," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 12, pp. 2371–2381, Nov. 2019.

[51] E. Jenelius, "Data-driven metro train crowding prediction based on real-time load data," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2254–2265, Jun. 2020, doi: 10.1109/TITS.2019.2914729.

Hang Yu (Member, IEEE) received the Ph.D. degree from the University of Technology Sydney, Australia, in 2020. He is currently a member of the Faculty of Engineering and Information Technology, Australian Artificial Intelligence Institute, University of Technology Sydney. He is also a Lecturer with the Faculty of Computer Engineering and Science, Shanghai University. He has published more than ten research articles in IEEE TRANSACTIONS. His research interests include streaming data mining, concept drift, and fuzzy systems.

Jie Lu (Fellow, IEEE) received the Ph.D. degree from Curtin University, Australia, in 2000. She is currently a Distinguished Professor and the Director of the Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. Her main research expertise is in fuzzy transfer learning, decision support systems, concept drift, and recommender systems. She has published six research books and 450 papers in IEEE TRANSACTIONS and other journals and conferences. She has completed over 20 Australian Research Council Discovery Grants and other research projects. She is an IFSA Fellow and an Australian Laureate Fellow. She serves as the Editor-In-Chief for *Knowledge-Based Systems* (Elsevier) and the *International Journal of Computational Intelligence Systems* (Atlantis). She has delivered 25 keynote speeches at international conferences and chaired ten international conferences.



Anjin Liu (Member, IEEE) received the B.I.T. degree (Hons.) from The University of Sydney, Sydney, NSW, Australia, in 2012. He was a Post-Doctoral Research Associate with the Faculty of Engineering and Information Technology, Australian Artificial Intelligence Institute, University of Technology Sydney, Ultimo, NSW, Australia. His research interests include concept drift detection, adaptive data stream learning, multi-stream learning, machine learning, and big data analytics.

Bin Wang is currently pursuing the dual Ph.D. degree with the School of Information Science and Technology, Southwest Jiaotong University, China, and the Faculty of Engineering and Information Technology, University of Technology Sydney, Australia. His current research interests include time series forecasting and deep learning.

Ruimin Li received the Ph.D. degree from Monash University. She is currently a Senior Manager at the Planning Analysis and Modeling Department, Sydney Trains, Australia.

Guangquan Zhang (Member, IEEE) received the Ph.D. degree in applied mathematics from Curtin University, Australia, in 2001. He is currently an Associate Professor and the Director of the Decision Systems and e-Service Intelligent (DeSI) Research Laboratory, Australian Artificial Intelligence Institute, University of Technology Sydney, Australia. He has authored five monographs, five textbooks, and 470 papers, including 225 refereed international journal articles. He has been awarded nine Australian Research Council (ARC) Discovery Project, an ARC QEII Fellowship in 2005, and many other research grants. His research interests include fuzzy modeling in machine learning and data analytics. He has served on editorial boards for several international journals and a guest editor of eight special issues for IEEE TRANSACTIONS and other international journals.