Users' Departure Time Prediction Based on Light Gradient Boosting Decision Tree

Lingyu Zhang[1,2], Zhijie He[2], Xiao Wang[2], Ying Zhang[2], Jian Liang[2], Guobin Wu[2], Ziqiang Yu[3], Penghui Zhang[4], Minghao Ji[4], Pengfei Xu[4], and Yunhai Wang[1](✉)

[1] School of Computer Science and Technology, Shandong University, Qingdao, China
cloudseawang@gmail.com

[2] Didi Chuxing, Beijing, China
{hezhijie_i,yingzhangying,liangjian,wuguobin}@didiglobal.com,
wangxiao@didichuxing.com

[3] Yantai University, Yantai, China
zqyu@ytu.edu.cn

[4] School of Information Science and Technology, Northwest University, Kirkland, USA
pfxu@nwu.edu.cn

Abstract. With the development of urban transportation networks, the flow of people in cities generally shows the characteristics of concentration, periodicity and irregularity, and a typical example is rush hour. For most existing taxi-hailing apps, users frequently queue up for a relatively long time during rush hour and may even fail to get orders taken due to various factors. To solve this problem, we propose a users' departure time prediction model based on Light Gradient Boosting Machine (TP-LightGBM), which will remind users to book taxis before their journeys. As we know, TP-LightGBM may be the first model for departure time prediction. We uncover that travel behavior patterns vary under different external conditions through statistics and analysis of users' historical orders from multiple perspectives. Furthermore, we extract multiple features from these orders and select the favorable features by calculating their information gain as the input of TP-LightGBM to predict users' departure time. Therefore, our model can provide users with the recommendations of the best departure time if they need them. The final experimental results on our datasets indicate that TP-LightGBM has more excellent performance with great stability in predicting user departure time than other baseline models.

Keywords: Departure time prediction · Light gradient boosting machine · Data analysis · Feature engineering · Loss assessment

1 Introduction

The accelerating development of Smart City has put forward new requirements for Intelligent Transportation System and Smart Travel, and the big data on travel provides strong support for related researches. As an essential part of intelligent travel, accurate travel time prediction is crucial. Specifically, user departure time prediction refers to data mining from numerous users' historical travel records to predict where and when users may have travel plans, so as to timely remind them to arrange travel or book taxis in advance. As users cannot grasp the specific information of actual traffic flow, it is difficult to reasonably choose the departure time and travel modes, which is the crux of contradiction and is also one of the main problems to be solved in Smart Travel.

At present, the increasingly severe traffic jam causes great inconvenience to people's daily travel, especially at some specific time such as the rush hour. Due to the diversity of individual travel behavior and the complexity of traffic information, the model of departure time prediction based on historical travel reports cannot always be accurate. At present, there are few works about users' departure time prediction. While, the existing relevant models are passive statistical models, which passively predict the future through the statistics of actual historical data and analyzing their patterns. The insufficient information on future travel rein in the performance of models, which leads to that such models can not consistently maintain high prediction accuracy. However, the time prediction models based on deep learning require a large amount of data and complex calculations. Although they can achieve sound prediction effects, it is difficult for these models to guarantee real-time performance when many users are online.

Therefore, a users' departure time prediction model based on Light Gradient Boosting Machine (TP-LightGBM) is proposed in this paper. TP-LightGBM can be used to remind users to arrange travel and to book taxis in advance within a reasonable time. The prediction results can help users choose their optimal departure time and travel patterns more freely to reduce information delay, and also can avoid congestion and significantly improve the quality and efficiency of users' travel. Of course, whether to provide relevant services is determined according to users' requirements on the recommendations of the best departure time.

2 Related Works

Users' departure time prediction is a necessary function of intelligent transportation and is also an essential part of intelligent city construction. With the rapid development of Intelligent Traffic Systems (ITS), various machine learning algorithms have contributed to traffic data reconstruction, traffic flow prediction, urban traffic pattern mining, and so on.

There are many methods to predict travel time in previous works, most of which focus on the travel time prediction of vehicles on the road to assist traffic control, yet few works are about users' departure time prediction. For example, Chien et al. proposed a prediction model of bus arrival time based on an artificial neural network by using the data of trajectories and bus stops [6]. This model for arrival time and location prediction combines an artificial neural network, and Kalman filter [4]. It estimates the arrival time and updates the real-time locations of vehicles according to the data of automatic passenger counters. In addition, many other works focusing on the prediction of the travel time of vehicles on the road make it more convenient to analyze the traffic flow [1,18,20]. For example, combined with Decision Tree and Linear Regression, future highway travel time can be predicted based on flow and occupancy data [13]. Besides, to solve the problem that the travel time is just a simple addition of link time, the data for modeling is path-based rather than link-based [3]. Meanwhile, the Kalman filter is introduced, and through continuous updating state variables as the new observation variable to predict the traffic on the motorway driving time [7]. As to particular unpredictable events, a Bayesian dynamic linear learning model [10] was proposed, which could adjust the parameter settings and noise level adaptively.

The rising Machine Learning and Deep Learning methods in recent years shed light on a new way to predict travel time. Duan et al. established an LSTM network for each link [9], which verified the prospects of the deep learning model considering the time-series relationship in travel time prediction. In addition, Gradient Boosting Decision Tree (GBDT) is applied to analyzing and modeling the travel time of highway vehicles [21] and discussing the impact of different parameters on the model's performance. Gradient Boosting Tree(GBT) is a boosting method based on the weak learner of tree model [14] pertaining two typical usages as Gradient Boosting Decision Tree (GBDT) and Gradient Boosting Regression Tree (GBRT). GBDT can be applied to the prediction [5,8,19] and classification [17] problems, and it can effectively merge different types of variables and fit complex nonlinear relationships. However, rather low efficiency is always a demerit of GBDT, especially with large-scale features and big data. For this problem, a gradient-based unilateral sampling method [12] is offered using the information gain of samples with larger gradients to estimate the overall information gain to improve efficiency with little compromised accuracy. Meanwhile, a feature selection method based on artificial bee colonies and GBDT was presented in [16], which globally optimized the feature space to enhance the efficiency and quality. Besides, Light Gradient Boosting Machine (LightGBM) [12] downsizes the features by bundling mutual exclusive features and downs sample the data instances by keeping all instances with big gradients and randomly sampling instances with small gradients, which reduces the number without changing the distribution of original data by much.

In summary, to improve the performance of individual travel time prediction, we introduce the LightGBM to predict the user's departure time based on the historical taxi orders of Didi Chuxing's users. More specifically, we have trained a model with individual characteristics for each user based on his/her historical orders of Didi Chuxing, which will remind users to book a taxi in advance before needed. And experiments prove that our model is not sensitive to the independence between diverse features and can correctly fit complex feature relationships.

3 The Analysis of Users' Departure Time and Travel Behaviors

3.1 The Overall Information of Users' Historical Taxi Orders

In this section, we analyze the users' historical taxi orders from multiple perspectives and visualize the results of the data analysis. The dataset used in this paper is the historical taxi orders of users who used Didi Chuxing online taxi-hailing platform, and the information of each sample mainly includes the user ID, order ID, time, locations, date attribute, and so on. Where, time (Notation as T) is the specific time of a day, and has been processed into the form of periods with hourly granularity. The range of T is in [0, 23). Besides, we also transform the original time into a day of the week (Notation as W), so a type of time-series data can be used to mine the regularity of users' travel behaviors, and the range of W is in [0,6]. Each location name is mapped to a corresponding pair of longitude and latitude. The range of the longitude (Notation as Lng) is in $[-180, +180]$, and the range of Latitude (Notation as Lat) is in $[-90, 90]$. The attribute values of the date (Notation as D) can be 0 or 1, and 1 stands for working days while 0 stands for holidays. In addition, all the orders have been anonymized and aggregated, and we correct the latitude and longitude of all locations and delete historical orders with abnormal order status.

3.2 The Analysis of Users' Travel Behaviors

Users' travel time shows noticeable regularity in a certain period. For example, users have a relatively regular commute time on weekdays and leisure time to go out on holidays, and even have regular travel times at several certain workplaces. Therefore, it is suitable for us to estimate the users' departure time using their historical orders. Users' travel behavior also shows strong regularities in the spatial domain. The following discussion is only a starting point, and we can draw similar conclusions in terms of destination. The spatial distribution of historical orders shows strong sparseness and concentration. Most of the starting points are concentrated in certain areas, while some others only appear once.

Moreover, we discover that users' travel time shows strong regularity in the spatial domain. The users' departure time in some places may concentrate on one specific time period. In addition, there are more cases of calling taxis in similar places in a similar time period, although there have been some effects of departure time on different starting points. For example, the users' taxi-hailing locations may be primarily residential areas in the morning, while workplaces, commercial areas, and entertainment venues at night.

The analysis result above is drawn from users' historical orders and reveals some commonalities in users' taxi booking behaviors: (1) The taxi-hailing time distribution of the same user tends to show a concentrated distribution in specific locations and times rather than a uniform distribution. (2) Most users have distinct travel patterns between workdays and holidays, and there are differences in users' departure times when the day type changes. (3) The same user tends to set the same destination in a certain period and rarely book a taxi in other periods. (4) Most users tend to go to a certain place within a fixed time.

3.3 Feature Selection

Feature selection plays a vital role in feature engineering. Due to the limited samples and the sparsity of distribution of users' historical orders, we use as few features as possible to predict users' departure time so as to avoid the high computational complexity and performance degradation of models caused by large-scale features. We list several candidate features which affect users' travel time, as shown in Table 1.

Table 1. Information gain (ratio) of each feature.

Feature	IG	IGR
Origin longitude	1.93	0.47
Origin latitude	1.92	0.47
Destination longitude	1.71	0.43
Destination latitude	1.81	0.33
Date attribute	0.11	0.14
Day of the week	0.55	0.42

Furthermore, we apply the feature selection method based on the Decision Tree. More specifically, we calculate the Information Gain (IG) and Information Gain Ratio (IGR) of each feature (see Table 1), which are respectively used in the module of ID3 [14], and C4.5 [15]. The methods to calculate IG and IGR are stated below:

Assume that the dataset is D, which has the size of $|D|$. The samples in D are divided into K categories $C_k(k = 1, 2, ..., K)$, and there are $|C_k|$ samples in class C_k. Then we assume that feature A can take n different values $a_1, a_2, ..., a_n$, which can divide D into n subsets $D_1, D_2, ..., D_n$, and $|D_i|$ $(i = 1, 2, ..., n)$ is the number of samples in D_i. In addition, let D_{ik} denotes the sample set of category k in subset D_i, and its size is denoted by $|D_{ik}|$. Thus, the information gained can be written as

$$g(D, A) = H(D) - H(D|A) \tag{1}$$

where, $H(D)$ is the empirical entropy of D, and $H(D|A)$ is the empirical conditional entropy of feature A to dataset D, $H(D)$ is calculated by

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} log_2 \frac{|C_k|}{|D|} \tag{2}$$

and $H(D|A)$ is

$$H(D|A) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} log_2 \frac{|D_{ik}|}{|D_i|} \tag{3}$$

The information gain ratio can be calculated by $g_R(D, A) = \frac{g(D,A)}{H_A(D)}$, where $H_A(D)$ is the entropy of the dataset D for feature A:

$$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|} \tag{4}$$

If the feature has a more significant Information Gain (Ratio), it will be more influential for classification and have a stronger ability to classify the samples. From the results in Table 1, IG and IGR of date attributes are both the smallest and should be discarded, while others should be retained in principle. However, the users' destinations are normally unknown in the actual scenario. If we first predict the destination and then the departure time according to the attribute of date, the cost will inevitably arise, and the accuracy cannot be guaranteed. Moreover, time features are weighty in the travel prediction. As we analyzed before, the feature day of the week contains information of regularity. Therefore, we decide to retain the attribute of date and day of the week and then elide the longitude and latitude of the destination. The experiments also demonstrate that the outcome using four features of origin longitude, latitude, day of the week, and date attribute is sounder than the origin longitude and latitude alone.

4 Time Prediction Model for Predicting Users' Departure Time

Through the in-depth analysis of orders of Didi Chuxing users, we convert users' departure time prediction into a multivariate classification problem. Users are classified according to their objective features using the category label of departure time. We use the hourly granularity as the classification standard and divide the users' historical orders into 24 categories.

4.1 Model Description

The probability of a user traveling at a fixed time period can be expressed in the form of conditional probability using Bayes' theorem:

$$P(T = t_i|X) = \frac{P(X|T = t_i)P(T = t_i)}{\sum_{i=1}^{24} P(X|T = t_i)P(T = t_i)} \tag{5}$$

where $X = Lng, Lat, D$. The process of solving the conditional probability $P(X|T = t_i)$ is extremely complicated, but the calculation difficulty will be greatly reduced if the method of conditional independent assumption of features in the naive Bayes algorithm is adopted, i.e. $P(X|T = t_i) = P(Lng|T = t_i)P(LatT = t_i|)P(D|T = t_i)P(W|T = t_i)$.

However, the features extracted from actual data are not as independent as the ideal assumption. Specifically, the latitude Lat and longitude Lng in the users' historical orders always emerge in pairs. For example, if location A often appears in one user's historical orders, then the latitude Lat_A and longitude Lng_A of location A have a highly correlated relationship, which does not meet the premise of conditional independence of each feature in the Naive Bayes algorithm. Moreover, taxi-hailing actions are purely personal behaviors, and regularity and irregularity coexist. The time distribution of taxi rides of a sample user may be evenly distributed throughout a day, which would confound the final prediction. Therefore, Gradient Boosting Decision Tree (GBDT) [11] is a suitable method for users' departure time prediction due to less demanding input features. However, the performance is unsatisfactory when the size of the data balloon. To balance this drawback, we introduce Light Gradient Boosting Machine (LightGBM) [12] which has an excellent performance to deal with a large number of data instances.

4.2 Departure Time Prediction Based on Light Gradient Boosting Machine (TP-LightGBM)

Decision Tree [14] is a primary classification and regression method, and its classification rules can be seen as a grouping of a series of if-then conditional statements or as a conditional probability model defined on features and class space. GBDT is a boosting algorithm based on the Classification and Regression Tree (CART) [2] and is one of the most widely used classification algorithms with high precision. Its main idea is to fit the residual of the previous base learner through the negative gradient of the loss function so that the residual estimation of each round declines. GBDT combines Gradient Boosting and Decision Tree to establish a new decision tree model (weak classifiers) in the gradient direction of the previous model residual reduction at each iteration. Finally, a well-trained GBDT classification model is a linear combination of these weak classifiers with different weights. The conventional implementation of GBDT is scanning all the instances for every feature to locate the optimal split points, which is time-consuming with big data. LightGBM based on GBDT proposes two techniques: Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) to reduce computational complexities. GOSS downsizes the instances by keeping them with large gradients and randomly chooses instances with slight gradients. EFB decreases the features by bundling mutually exclusive features.

The training process of multi-class LightGBM can be viewed as an additive model, as shown in Algorithm 1. In practice, the multi-class LightGBM generates a tree for each category during the training process, i.e., a total of $K \times M$ sub-trees are generated in Algorithm 1, and Softmax obtains the final category result. Specifically, the loss function we choose is log-likelihood, which can be written as

$$L(y, f(x)) = -\sum_{k=1}^{K} y_k log(p_k(x)) \tag{6}$$

where y denotes the actual value of a sample, $f(x)$ is the predictive value, $p_k(x)$ represents the probability that the sample belongs to the category k.

5 The Experimental Results and Analysis

Due to the coexistence of regularity and uncertainty in user travel, we delete orders whose starting point appeared less than five times in a month. Since each user's travel pattern is unique and it is impossible to select every user who travels regularly, we set a threshold τ to filter the prediction results. The result will be output if its probability surpasses the threshold τ. Our purpose is to predict the period for taxi-hailing of the users of Didi Chuxing and does not involve a specific timestamp. Therefore, we take the prediction time as the midpoint to extend one hour as a period for the final result, i.e., so the final result will be expressed in $[t-1, t+1]$ if the output result is t, and the actual label of a test sample is regarded as a correct prediction if falls within the interval $[t-1, t+1]$.

Algorithm 1: The training process of GBDT classifier

```
   Input: iterations (number of weak classifiers) M,number of samples N,
   number of categorise K,loss function $L(y, f(x))$,training set
   $T_{train} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$
   Output: GBDT classifier $\hat{f}(x)$
 1 Initialize:weak classifier $f_0(x) = arg\ \underset{\theta}{min} \sum_{i=1}^{N} L(y_i, \theta)$;
 2 while $m = 1, 2, \ldots, M$ do
 3     for $i = 1, 2, \ldots, N$ do
 4         for $k = 1, 2, \ldots, K$ do
 5             // Calculate the probability of $x_i \subseteq class\ k$
 6             $p_k(x_i) = \left[\frac{exp(f_k(x_i))}{\sum_{k=1}^{K} exp(f_k(x_i))}\right]_{f_k(x)=f_{k,m-1}(x)}$;
 7             // Calculate negatice gradient error
 8             $r_{mik} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f_k(x)=f_{k,m-1}(x)} = y_{ik} - p_k(x_i)$;
 9         end
10     end
11     // Fit the decision tree
12     use $r_{mik}$ to fit the decision tree, which leaf node area is $R_{mjk}$,
13     $j = 1, 2, \ldots, J, k = 1, 2, \ldots, K$;
14     // Estimate the gain of leaf nodes
15     for $j = 1, 2, \ldots, J$ do
16         for $k = 1, 2, \ldots, K$ do
17             $\theta_{mjk} = \frac{K-1}{K} \frac{\sum_{x_i \in R_{mjk}} r_{mjk}}{\sum_{x_i \in R_{mjk}} |r_{mjk}| (1 - |r_{mjk}|)}$;
18         end
19     end
20     // Update classification tree
21     $f_{km}(x) = f_{k,m-1}(x) + \sum_{j=1}^{J} \theta_{mjk} I(x_i \in R_{mjk}), k = 1, 2, \ldots, K$;
22 end
23 // Output GBDT classifier
24 $\hat{f}_k(x) = f_{kM}(x) = \sum_{m=1}^{M} \sum_{j=1}^{J} \theta_{mjk} I(x_i \in R_{mjk}), k = 1, 2, \ldots, K$;
```

5.1 Experimental Setups

We randomly select 80% of the dataset as the training set and the other 20% as the test set, then multiple experiments are conducted using our model under different thresholds, and the results are shown in Fig. 1. We apply two metrics $PDP = N_{out}/N_{test}$ and $AUC = N_{auc}/N_{out}$ to measure the performance of the model. Where N_{test} denotes the number of samples in the test set, N_{out} is the number of samples with the output results, and N_{auc} denotes the number of samples accurately predicted. It can be seen that AUC and PDP are proportional and inversely proportional to the threshold τ respectively, and the growth rate of AUC slows down, but PDP still has a strong downward trend when $\tau > 0.7$. Therefore, we finally adopt the threshold $\tau = 0.7$ to test our model's all-around performance.

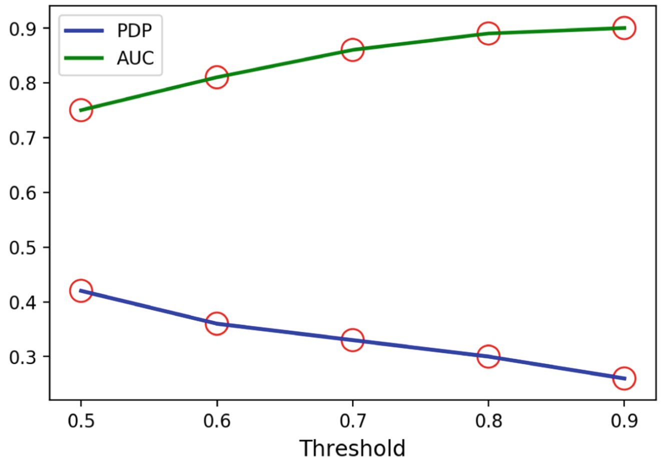

Fig. 1. An example of indirect blocking

5.2 Experimental Results and Comparative Analysis

In order to verify the superior performance of our model, GBDT, XGBoost, MultinomialNB, GaussianNB, BernoulliNB, and ModeMod are used as the comparison models. PDP, AUC, $Kappa$, $Hamming(H)$ and $Time(T)$ in Table 2 are used as the metrics of model performance evaluation. Here, $Hamming$ and $Time$ respectively represent Hamming distance and average time consumption of the models. $Kappa$ (Kappa coefficient) is often used to evaluate prediction accuracy and consistency, and it can be defined as $K_{appa} = (p_o - p_e)/(1 - p_e)$. Where p_o is the sum of the number of samples correctly classified in each category divided by the total number of samples, and p_e is the sum of the products of the actual and predicted sample numbers corresponding to all categories divided by the square of the total number of samples. Therefore, the prediction accuracy is positively correlated with the value of Kappa. Hamming distances measure the distance between the predicted label and the actual label. Thus, the prediction accuracy is negatively correlated with the value of Hamming distance. It needs to be stated that all metrics in Table 2 are the average values obtained from all test samples.

From Table 2, It can be seen that ModeMod has the most promising performance in terms of PDP and $Time$ metrics. ModeMod searches for orders that match the user's current status from the historical taxi-hailing orders and extract the departure times that meet the conditions. Then the departure time with the most occurrences is the predicted value. In this way, ModeMod shows the best stability and the lowest time complexity. In addition, the naive Bayes models (MultinomialNB, GaussianNB, and BernoulliNB) have a higher prediction accuracy than ModeMod. However, as we mentioned before, the conditional independence relationship between each feature is hard to achieve, and

Table 2. Experimental results on different models.

	PDP	AUC	$Kappa$	H	T(s)
TP-LightGBM	0.35	**0.90**	**0.64**	**0.14**	0.22
GBDT	0.33	0.87	0.58	0.16	0.74
XGBoost	0.33	0.88	0.62	0.17	0.51
MultinomialNB	0.84	0.54	0.16	0.44	0.107
GaussianNB	0.61	0.30	0.23	0.70	0.10
BernoulliNB	0.74	0.55	0.24	0.45	0.10
ModeMod	**1.00**	0.37	0.30	0.63	**0.09**

the distribution of features is difficult to determine. In contrast, the evaluation parameters of PDP, AUC, $Kappa$, and $Hamming$ are much better than the set of naive Bayes models, although the set of GBDT models has the highest time consumption, which TP-LightGBM can solve. In conclusion, TP-LightGBM only serves about 35% of orders, but the prediction accuracy has reached 92%. The main idea of TP-LightGBM is that the GBDT algorithm requires multiple iterations to fit data and train different weak classifiers, so it has a higher time consumption, but the average prediction time for each order can still be restrained within one second.

6 Conclusion and Future Work

Users' departure time prediction is an application of Machine Learning to Smart City construction. Accurately predicting users' departure times can remind them to call a taxi in advance and avoid queuing up during the rush hour. In this paper, we conduct a multi-perspective analysis and pattern discovery on the historical taxi orders of Didi Chuxing's users and verify the possibility of using these orders to predict departure time. Moreover, we propose such a model based on LightGBM using the users' current location, the order of the day of the week, and the date attribute. Finally, the experimental results indicate the superior performance of TP-LightGBM in predicting users' departure time. However, our model can only serve about 35% of orders. Thus, improving the prediction probability and expanding the service volume become the focus of our future research.

Acknowledgment. We are grateful to anonymous reviewers for their helpful comments. This work is partially supported by the National Key Research and Development Program of China under Grant No. 2018AAA0101100.

References

1. Van der Aalst, W.M., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Inf. Syst. **36**(2), 450–475 (2011)
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Routledge, Milton Park (2017)
3. Chen, M., Chien, S.I.: Dynamic freeway travel-time prediction with probe vehicle data: link based versus path based. Transp. Res. Rec. **1768**(1), 157–161 (2001)
4. Chen, M., Liu, X., Xia, J., Chien, S.I.: A dynamic bus-arrival time prediction model based on APC data. Comput.-Aided Civil Infrastruct. Eng. **19**(5), 364–376 (2004)
5. Cheng, J., Li, G., Chen, X.: Research on travel time prediction model of freeway based on gradient boosting decision tree. IEEE Access **7**, 7466–7480 (2018)
6. Chien, S.I.J., Ding, Y., Wei, C.: Dynamic bus arrival time prediction with artificial neural networks. J. Transp. Eng. **128**(5), 429–438 (2002)
7. Chien, S.I.J., Kuchipudi, C.M.: Dynamic travel time prediction with real-time and historic data. J. Transp. Eng. **129**(6), 608–616 (2003)
8. Ding, C., Wang, D., Ma, X., Li, H.: Predicting short-term subway ridership and prioritizing its influential factors using gradient boosting decision trees. Sustainability **8**(11), 1100 (2016)
9. Duan, Y., Yisheng, L., Wang, F.Y.: Travel time prediction with LSTM neural network. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pp. 1053–1058. IEEE (2016)
10. Fei, X., Lu, C.C., Liu, K.: A Bayesian dynamic linear model approach for real-time short-term freeway travel time prediction. Transp. Res. Part C: Emerg. Technol. **19**(6), 1306–1318 (2011)
11. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. 1189–1232 (2001)
12. Ke, G., et al.: LightGBM: a highly efficient gradient boosting decision tree. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
13. Kwon, J., Coifman, B., Bickel, P.: Day-to-day travel-time trends and travel-time prediction from loop-detector data. Transp. Res. Rec. **1717**(1), 120–129 (2000)
14. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
15. Quinlan, J.R.: C4. 5: programming for machine learning. Morgan Kauffmann **38**(48), 49 (1993)
16. Rao, H., et al.: Feature selection based on artificial bee colony and gradient boosting decision tree. Appl. Soft Comput. **74**, 634–642 (2019)
17. Sun, R., Wang, G., Zhang, W., Hsu, L.T., Ochieng, W.Y.: A gradient boosting decision tree based GPS signal reception classification algorithm. Appl. Soft Comput. **86**, 105942 (2020)
18. Xu, J., Rahmatizadeh, R., Bölöni, L., Turgut, D.: Real-time prediction of taxi demand using recurrent neural networks. IEEE Trans. Intell. Transp. Syst. **19**(8), 2572–2581 (2018)
19. Yang, L., Zhang, X., Liang, S., Yao, Y., Jia, K., Jia, A.: Estimating surface downward shortwave radiation over china based on the gradient boosting decision tree method. Remote Sens. **10**(2), 185 (2018)
20. Zhang, X., Rice, J.A.: Short-term travel time prediction. Trans. Res. Part C: Emerg. Technol. **11**(3–4), 187–210 (2003)
21. Zhang, Y., Haghani, A.: A gradient boosting method to improve travel time prediction. Transp. Res. Part C: Emerg. Technol. **58**, 308–324 (2015)