Prediction of Passengers Demand for Customized Bus Systems

Saumya Bhatnagar*[1], Rongge Guo[1], Jihui Ma[2], and Mauro Vallati[1]

[1]School of Computing and Engineering, University of Huddersfield, United Kingdom
[2]School of Traffic and Transportation, Beijing Jiaotong University, China

Short summary

The customized bus system is an innovative demand-responsive public transit service with the potential to significantly alleviate congestion and environmental footprint. To fully exploit the flexibility of this approach, it is pivotal to forecast the demand for the service, in order to optimize the use of vehicles and resources. In this paper, with the aim for supporting the use of customized bus systems, we formalize the predictive task and assess the performance of a range of machine learning techniques. We introduce a two-step predictive task aiming at (i) identifying the presence of demand and, if there is actual demand, (ii) estimating the number of passengers to be served. The experimental analysis, based on realistic data from the Beijing area, shed some light into the performance of different classes of approaches.
Keywords: Customized Bus System, Machine Learning, Artificial Intelligence.

1 Introduction

The customized Bus (CB) system is an innovative and extremely flexible public transit (PT) service with the benefits of congestion alleviation and environmental friendliness (Shu & Li, 2022), that is emerging as an alternative to conventional buses and private cars. CB is a class of demand-oriented transit that holds the promise to provide "door-to-door" service to passengers with similar travel requirements in both space and time (Liu & Ceder, 2015).
The vast majority of existing work on CB systems assumes that travel demands are collected in advance from dedicated online booking platforms. The CB system can then produce plans including routes and schedules to satisfy the time- and space-restricted requests accordingly (Huang et al., 2020). In literature, a class of approaches allows to generate optimal solutions for these static demands, but at the cost of reduced flexibility for the service, as dynamic travel requests are ignored and not handled during operation. To tackle this pivotal issue, that reduces the usability of the service, a growing number of approaches are exploring the idea of a two-stage optimization procedure for on-demand CB service, as shown in Fig. 1. The first stage is similar to the traditional CB service design and generates the initial plans (including routes, timetables and schedules) for static requests. Each request contains a paired origin-destination (OD) and preferred departure time. The second stage responds to dynamic requests received by adjusting the initial plans during operation (Wang et al., 2020). However, even this class of approaches is not fully exploiting the flexibility of the CB system. It may be impractical to satisfy all the dynamic requests, that also provide strict constraints on preferred time windows for pickup and drop off, starting from the generated static plan.
During the operation, the dynamic stage is triggered once new requests pop up. The re-planning process activities are implemented to insert each emerging request into the current network. If existing CB routes are able to fully satisfy the spatial-temporal constraints of the request, the service is updated and followed by drivers via real-time communication; otherwise, the request is ignored by the system and defined as unserved. The main reason behind this is that the short response time makes it impossible to rapidly input new requests into the current network while fulfilling travel restrictions of both already assigned and new requests. Therefore, the prediction of the dynamic requests is essential for planning to prevent the unserved requests and minimize uncertainty in operation of the dynamic stage, which can improve the operational efficiency of CBs.
To fully reap the benefits of CB systems, it is crucial to have in advance an estimate of the dynamic requests, to generate plans that are robust with regards to dynamic changes and adjustments. Having approaches that can accurately estimate future passenger demands would help ensuring

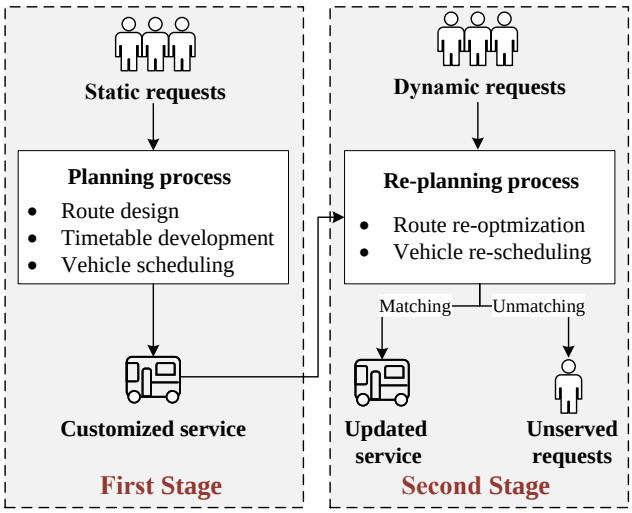

Figure 1: The planning process of the on-demand customized bus service.

Table 1: Data structure of collected SCD information.

Field Name	Description
CardID	Card identification number
LineID	Bus line number
BusID	Vehicle identification number
UpTradeStation	Boarding station number
UpTradeTime	Boarding time
DownTradeStation	Alighting station number
DownTradeTime	Alighting time

a higher level of optimization in CB systems, as well as better service and higher environmental benefits. To this end, in this paper we formalize the predicting task, we describe best practice, and investigate the use of a range of machine learning algorithms. Our analysis is performed considering real-world smartcard data collected by the Automatic Fare Collection system in Beijing over 2 months, and the results demonstrate that machine learning is a promising approach to be used to support CB systems in improving their effectiveness. Further, the proposed analysis provides useful insights into how to preprocess data and to represent complex variables, that can help in fostering the use of machine learning techniques in similar applications.

2 Methodology

A quantitative case-study approach has been adopted to define the predictive task and determine the machine learning technique that best predicts passenger demands for a CB system.

Data Source

In this study we use real-world data extracted from the SCD collected by the Automatic Fare Collection (AFC) system in Beijing, China, which can provide a complete overview of the trips. The SCD contains more than 12 million transactions per day. Each transaction records details of the get on and get off location, and time for the trip information (see Table 1).
For this experimental analysis, we consider passenger spatial-temporal dynamics collected from December 1, 2018 to January 31, 2019. The considered period of time is before the start of the COVID-19 pandemic, so no restrictions were in place at the time. We focus only on working days, as travel requests of CB are expected to be mainly from commuters. For the same reason, we select records from typical residential and working areas during the morning peak hours (7:30-9:00). This is the time when the CB system is expected to be most under stress from dynamic travel demands. Starting from the raw SCD data, we follow the approach proposed by Guo et

al. (2019) to generate corresponding demands for a CB system. In particular, the implemented approach relies on three major stages, namely trip chain generation for non-transfer and transfer trips, station identification, and OD matrix generation. After that, travel demands for each OD and corresponding boarding and alighting timestamps for passenger groups are collected, and used as the input for the prediction system. The distribution of extracted travel demands is given in Fig. 2.

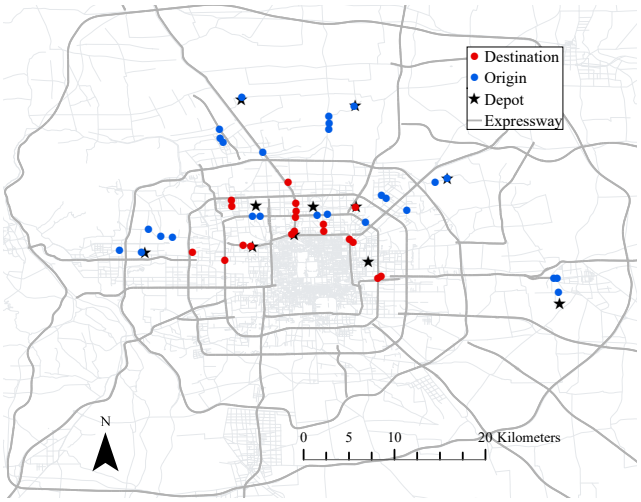

Figure 2: Distribution of extracted travel demands.

The data set includes 5 features, all alphanumeric, referred to as original features in Table 2.

Data Preprocessing and Transformation

To support the predictive task, the limited initial set of available features has been extended as shown in Table 2. From the original date, we extracted two additional features referring to the specific day of the week (Monday, Tuesday, etc.) and the indication of whether the day is a Monday, Friday, or any other day of the week. This is because this 3 classes of days can show different travel patterns. From the original time stamp, 5 time categories have been identified, namely 0700-0730, 0730-0800, 0800-0815, 0815-0830, 0830-0845 and converted to ordinal categorical values. The categorization of starting demand time decreases the complexity of the predicting element, while preserving the usefulness of the information.
The 2 features reporting the IDs of the origin and destination bus stops have been combined into a single variable by concatenating the corresponding values. In order to do that, all the possible combinations have been generated. This lead to some empty demands, that augmented the size of the data set, with a ratio of datapoints showing demand vs without demand to be in the imbalanced ratio of 8/23. Finally, the original numeric demand feature has been divided into 2 different features, namely the size of the passenger group for the demand, and a Boolean feature indicating the presence of the demand for a specific set of features (Origin-Destination pair, Date and Time). In the considered data set, the size of demands ranges between 0 (where there is no demand) and 30.

Table 2: Extended Features

Original features	Extracted features	Type
Date	Date	datetime format
	Week Day	Categorical
	DayType	Categorical
Time Stamp	Time period	Categorical
Origin Destination	Origin-Destination	Categorical
Demand	Size	Integer
	Presence	Boolean

We performed a correlation analysis of the extended variables, that confirmed that there is no

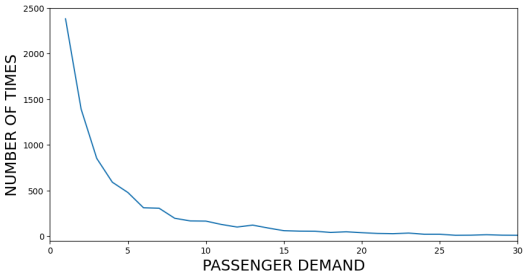

Figure 3: Demand size distribution.

correlation among them, hence the set is suitable to be used for training purposes.
Fig. 3 shows the distribution of the passenger groups sizes in the considered data set. Notably, smaller groups are very common, while larger groups – here the largest considered is of 30 passengers – are much less usual.

Predictive Task

Given the complexity of the predictive task at hand, we decided to define it as two different predicting tasks. First, a classification step that allows to identify the presence of demand (last row, Table 2), then there is a subsequent regression step that aims at predicting the size of the corresponding passengers group. This separation gives also us the opportunity to assess the abilities of a range of well-known machine learning approaches on two different yet crucial tasks.
The following approaches have been considered for this analysis: K-Nearest Neighbors (KNN)(Wu et al., 2008), Decision Tree, Random Forest (Breiman, 2001), AdaBoost (Schapire, 2013), LightGBM (Ogunleye & Wang, 2019), and Multi Layer Perceptron (MLP) (Haykin, 1994). The approaches have been selected based on the range of implemented techniques, and on their performance on well-known benchmarks.
As a baseline algorithm, to better contextualize the achieved results, we use a traditional Linear Regression approach, declined in its Logistic Regression form for the classification task.

Evaluation Metrics

For the binary classification problem of predicting the presence of demand, we relied on the well-known notions of true positive (negative), false positive (negative). We then consider as metrics for assessing the abilities of systems, *accuracy*, *F1 score*, and *Area Under the Curve* (AUC).
When it comes to the metrics to assess the capabilities of the regression models, those selected are *Mean Absolute Error* (MAE), *Mean Squared Error* (MSE), and *Root Mean Squared Error* (RMSE). MAE measures the average of the absolute residuals (the difference between the actual value and the predicted value) in the dataset. MSE measures the variance of the residuals and the RMSE measures the standard deviation of those residuals. For those metrics, the lower the value, the better the performance of the regression model.

Training Settings

For training, testing, and validating purposes we used sklearn (Pedregosa et al., 2011). sklearn is a robust and well-known Python library for performing ML tasks. It natively supports hyper-parameter optimization (Feurer & Hutter, 2019). In the following, we shortly summarize the parameters considered for the optimization of the learning algorithms.

- KNN: The value of K, i.e. the number of nearest neighbors to be considered.
- AdaBoost: we optimized 3 aspects: the base estimators, the number of estimators in the ensemble, and the learning rate. The base estimators were given three choices between Support Vector Machine, Logistic Regression, and Decision Trees.
- LightGBM: the 4 parameters to be optimized include the boosting type (GBDT or DART), the number of maximum leaves in each learner, the boosting learning rate, and the number of boosted trees.
- MLP: we optimized 5 parameters, namely the set of hidden layer sizes, activation functions, solvers, alphas, and learning rates.

Table 3: Classification Results Summary

Classifiers	Accuracy	F1	AUC
Logistic Regression	73.81	2.42	58.75
KNN	82.65	49.15	80.52
MLPClassifier	79.49	45.95	79.88
Decision Tree	78.14	39.50	75.98
AdaBoost	75.88	15.89	76.00
Random Forest	72.59	71.81	82.67
LightGBM - DART	87.17	77.85	94.65

Ensemble Models		RF		DART	
		Predicted			
		Positive	Negative	Positive	Negative
TRUE	Positive	2206	962	1425	193
	Negative	770	2382	618	4084

Figure 4: Confusion matrices of Random Forest and LightGBM - DART.

The learning rate ranges for all the aforementioned hyper-parameter optimizations, have been kept as floating values ranging between 0.05 to 1.
Finally, we also optimized the best Logistic Regression algorithm, selecting among those implemented by the sklearn library.
To reduce noise, random seeds have been fixed.
For the classification task, the hyper-parameters optimization aims at improving two metrics, namely F1 and ROC-AUC. Ties are broken in favor of AUC. For regression, the focus is on minimizing the Mean Absolute Error (MAE).

3 Results

In this section we present the results of our extensive experimental analysis. Some preliminary tests, not shown in this paper, performed by considering the single straightforward regression task of predicting at the same time both presence and size of demand lead to models not usable in practice, due to the extreme error. Therefore, in this section we consider the two steps predictions discussed in Methodology. First, we focus on the classification step that aims at predicting the likeliness of travel demand between two stops, and second we consider the regression task of predicting how many passengers will be part of the travel group.
Table 3 shows the performance of the considered algorithms on the binary classification task. The results indicate that, unsurprisingly, Random Forest is the algorithm that shows the worst accuracy; it has been outperformed even by Logistic Regression and Decision Tree. This is due to the limited number of available features, even in the extended set, that is a known element that can reduce the performance of this class of approaches. On the other hand, Random f=Forest should be the second best choices when evaluating on F1 score and AUC, demonstrating that the unbalanced nature of the data is not strongly negatively affecting the generated models. The optimized LightGBM using DART is delivering outstanding performance according to all metrics; however, we observed that with almost all other hyper-parameter configurations of LightGBM showed significantly worse results (not shown in the table). Finally, it is worth noticing the very low F1 score and AUC of the Logistic Regression approach, suggesting that the approach does not cope well with the unbalanced data set.
To shed some light into the relative performance of Random Forest and LightGBM, Figure 4 shows the corresponding confusion matrices. Random Forest (RF) is better in predicting positive cases, but shows low performance when predicting cases for which no demand is expected.
We now turn our attention to the regression task. Results are presented in Table 4. As a first remark, we do not include KNN results because, due to the way KNN models are generated, it quickly runs out of memory for large data sets – hence providing predictions of very low quality.

Table 4: Regression Results Summary

Regressors	MAE	MSE	RMSE
Linear Regression	1.87	13.49	3.67
MLPRegressor	1.92	13.69	3.70
Decision Tree	1.10	9.30	3.04
AdaBoost	3.62	23.54	4.85
Random Forest	0.82	4.05	2.01
LightGBM - DART	0.88	3.62	1.90

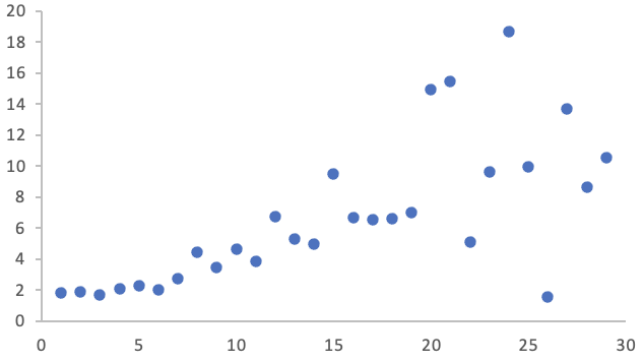

Figure 5: Mean absolute error (y-axis) for each actual demand value (x-axis) when LightGBM is used for the regression task.

With regards to the other considered algorithms, LightGBM is again providing extremely good performance, but in terms of MAE Random Forest is, surprisingly, the approach that shows best results. AdaBoost is instead the worst one, that produces predictions that are not usable in practice for optimizing the routes of CB systems.
Figure 5 shows how the MAE varies according to the actual demand to be predicted, when the best predictor is used. As expected, for ranges of demand that are extremely well represented in the data set, the error is very limited. When demands get larger and less common, the predictions show a higher degree of variability. Pragmatically, this means that the predictions will be more precise for common demands – and the graph shown already provides some information about the reliability of predictions made for different demand sizes.
Finally, a general result can be derived by comparing Random Forest and AdaBoost results on both classification and regression tasks that using Bagging ensembles over Boosting ensembles lead to better results in regression, but the contrary is true in terms of classification. While this behavior may not generalize on different data sets, it can provide an interesting take-home message for this class of applications.

4 Conclusion

With the aim of fostering the use and efficiency of customized bus systems, in this paper we tackled the task of forecasting travel demands to be served. Our analysis provided insights into how data can be processed to provide high quality input for machine learning algorithms, and we showed how a challenging task can be divided into more amenable tasks that are easier to be be dealt with. On this regards, we demonstrated the abilities of a range of predictive techniques on both classification and regression tasks, using realistic data from the Beijing area. Our results indicate that machine learning provides suitable approaches to tackle the challenges of CB systems due to the limited ability to forecast ongoing demand.
Future work will focus on extending the analysis to different metropolitan areas, and to increase the considered features to include aspects such as weather, traffic, etc.

Acknowledgements

This work is supported by the UKRI Future Leaders Fellowship [grant number MR/T041196/1].

References

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In F. Hutter, L. Kotthoff, & J. Vanschoren (Eds.), *Automated machine learning: Methods, systems, challenges* (pp. 3–33). Springer International Publishing.

Guo, R., Guan, W., Huang, A., & Zhang, W. (2019). Exploring potential travel demand of customized bus using smartcard data. In *2019 ieee intelligent transportation systems conference (itsc)* (p. 2645-2650). doi: 10.1109/ITSC.2019.8916843

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

Huang, D., Gu, Y., Wang, S., Liu, Z., & Zhang, W. (2020). A two-phase optimization model for the demand-responsive customized bus network design. *Transportation Research Part C: Emerging Technologies*, *111*, 1–21.

Liu, T., & Ceder, A. A. (2015). Analysis of a new public-transport-service concept: Customized bus in china. *Transport Policy*, *39*, 63–76.

Ogunleye, A., & Wang, Q.-G. (2019). Xgboost model for chronic kidney disease diagnosis. *IEEE/ACM transactions on computational biology and bioinformatics*, *17*(6), 2131–2140.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Schapire, R. E. (2013). Explaining adaboost. In *Empirical inference* (pp. 37–52). Springer.

Shu, W., & Li, Y. (2022). A novel demand-responsive customized bus based on improved ant colony optimization and clustering algorithms. *IEEE Transactions on Intelligent Transportation Systems*.

Wang, C., Ma, C., & Xu, X. D. (2020). Multi-objective optimization of real-time customized bus routes based on two-stage method. *Physica A: Statistical Mechanics and its Applications*, *537*, 122774.

Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., ... others (2008). Top 10 algorithms in data mining. *Knowledge and information systems*, *14*(1), 1–37.