IET Intelligent Transport Systems

Research Article

IET Journals
The Institution of Engineering and Technology

How to pay less: a location-specific approach to predict dynamic prices in ride-on-demand services

ISSN 1751-956X
Received on 29th September 2017
Revised 13th April 2018
Accepted on 8th May 2018
E-First on 29th May 2018
doi: 10.1049/iet-its.2017.0300
www.ietdl.org

Suiming Guo[1,2], Chao Chen[3] ✉, Yaxiao Liu[4], Ke Xu[4], Bin Guo[5], Dah Ming Chiu[2]

[1]*College of Information Science and Technology, Jinan University, Guangzhou, People's Republic of China*
[2]*Department of Information Engineering, the Chinese University of Hong Kong, Shatin, N.T., Hong Kong*
[3]*College of Computer Science, Chongqing University, Chongqing, People's Republic of China*
[4]*Department of Computer Science and Technology, Tsinghua University, Beijing, People's Republic of China*
[5]*School of Computer Science and Technology, Northwestern Polytechnical University, Xi'an, People's Republic of China*
✉ *E-mail: cschaochen@cqu.edu.cn*

Abstract: In emerging ride-on-demand (RoD) services, dynamic pricing plays an important role in regulating supply and demand and improving service efficiency. Despite this, it also makes passenger anxious: whether the current price is low enough, or otherwise, how to get a lower price. It is thus necessary to provide more information to ease the anxiety, and predicting the prices is one possible solution. In this study, the authors predict the dynamic prices to help passengers learn if there is a lower price around. They first use entropy of historical prices to characterize the predictability of prices in different locations and claim that different prediction algorithms should be used to balance between efficiency and accuracy. They present an ensemble learning approach to price prediction and compare it with two baseline predictors, namely a Markov and a neural network predictor. The performance evaluation is based on the real data from a major RoD service provider. Results verify that the two baseline predictors work well in locations with different levels of predictabilities, and that ensemble learning significantly increases the prediction accuracy. Finally, they also evaluate the effects of prediction, i.e., the probability that passengers could benefit from the prediction and get a lower price.

Nomenclature

Δt	length of each time slot
K	total length of price multiplier sequence
$R^{(i,j)}$	(sub)-sequence of price multipliers at cell (i, j): $R^{(i,j)} = \left\{R_k^{(i,j)}, \quad k = 1, 2, \ldots, L, L \le K\right\}$
L	length of the (sub)-sequence $R^{(i,j)}$
$N^{(i,j)}$	number of distinct price multipliers at cell (i, j)
$r_m^{(i,j)}$	distinct price multipliers in $R^{(i,j)}$, $1 \le m \le N^{(i,j)}$
$S_{\text{Shannon}}^{(i,j)}$	Shannon entropy of $R^{(i,j)}$
$S_{\text{real}}^{(i,j)}$	real entropy of $R^{(i,j)}$
$S^{(i,j)}$	set of time-ordered sub-sequences of $R^{(i,j)}$, $S^{(i,j)} = \left\{s^{(i,j)} \mid s^{(i,j)} \subseteq R^{(i,j)}\right\}$
$s_k^{'(i,j)}$	length of the shortest unseen sub-sequence of $R^{(i,j)}$ starting at time k
$\Pi^{\max}$	maximum predictability of any prediction algorithm

1 Introduction

Emerging ride-on-demand (RoD) services such as Uber and Didi have drawn increasing attention recently. They attract passengers by its cleanness, convenience, as well as flexible and affordable prices, and on the other hand, attract drivers who want to make use of their idle cars without applying for licenses. Meanwhile, they also create concerns that sometimes their dynamic prices go so high (as high as five or ten times the normal prices) during a big event or in bad weather.

Dynamic pricing is the core and distinctive feature in RoD service, and it reflects the effort in controlling the supply (the number of cars) and demand (the number of requests) in a particular location so that an equilibrium is approached: a higher price reduces demand and increases supply in a busy area, and a lower price does the opposite in a not-busy area. As a result, the service becomes more responsive for both drivers and passengers.

On the other hand, dynamic pricing exerts mental burden on passengers. In traditional taxi service with fixed pricing, passengers are able to estimate the trip fare based on their personal experience. In emerging RoD services, however, they have an extra task before making decisions – predicting the dynamic prices – based on their estimate of the supply & demand condition nearby, and this is invariably inaccurate for most individual passengers. Without relevant information, passengers may wonder 'Could I get a lower price in neighbouring locations (e.g. within hundreds of metres) or within a short time (e.g. 10 to 20 min)?'. These questions create hesitation and exert mental burden on passengers.

Giving more information to passengers help ease the anxiety, including, for example, by explaining why the current price is high or low, giving a recent history of prices to passengers, predicting the prices in the next time slot in the neighbouring locations etc. Among these ideas, the most direct one is price prediction, and passengers could answer the above questions and make a decision based on the predicted prices, instead of relying on a rough and unreliable estimate of the supply & demand nearby.

Price predictions have not received much attention in RoD services. Chen *et al.* [1] tried to predict the future prices by guessing the relationship between dynamic prices and a combination of supply and demand. Since most RoD services keep their dynamic pricing algorithms as secrets, guessing the relationship from data is not accurate enough to generate a good prediction. Instead, we propose to predict dynamic prices based on the historical data, instead of trying to learn some unknown, internal relationship from outside. This is inspired by the demand prediction work that was common in the studies on taxi service. These studies try to predict the demand for taxi service by studying the historical data, using methods such as time-series analysis, SVM or neural networks (NNs).

In RoD services, dynamic pricing is represented by a price multiplier. The price of a trip is the product of a dynamic price multiplier (dependent on the supply & demand condition) and a fixed normal price (dependent on the estimate distance & time of the trip). The geographical and temporal variation of the price

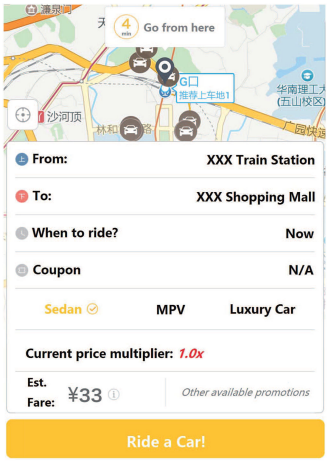

Fig. 1 *User interface of a typical RoD service*

multiplier is the source of anxiety and is also the target of prediction in this study.

The regularity of price multipliers varies among different areas in a city. For example, at the airport terminals or train stations, the price multiplier is always stable during the day, not only because the demand is stable throughout the day, but also because drivers know this and are more inclined to cruise in these areas looking for passengers. In business areas (e.g. the Financial Street in Beijing), contrarily, the price multiplier is more volatile, as there are different kinds of passengers leaving this area, including those in travel, business visit, going/leaving workplaces etc. and there is not a regularity of passengers' behaviour from day to day.

This inspires us to use a metric – *maximum predictability* $\Pi^{\max}$ – to characterise the patterns of price multipliers in different locations. The calculation of predictability is based on the entropy of price multiplier and captures both the randomness and the temporal correlation behind it. This metric expresses both the regularity and randomness of price multipliers. For example, an area with a $\Pi^{\max} = 0.3$ means that for 30% of the time, the price multiplier is regular and predictable and for the rest 70% of the time, the price multiplier appears to be random.

We then use different prediction algorithms to predict price multipliers in areas with different $\Pi^{\max}$. To start with, we first implement two baseline predictors. For an area with high $\Pi^{\max}$, the prediction could be done by easier and faster algorithm, and we use a Markov-chain predictor, considering only temporal correlations between price multipliers. Alternatively, the predictor should consider more factors such as weather condition in an area with lower $\Pi^{\max}$, and we use a NN predictor in this case. We then employ an ensemble learning approach, i.e. building multiple baseline predictors and combining their results to improve the prediction accuracy and implement a boosted NN predictor. In short, in ensemble learning, we try to build every new predictor to focus more on the wrong predictions by the last predictor and finally combine the predictions of all predictors by a weighted summation.

In this study, we collaborate with a major RoD service provider in China and evaluate the performance of different predictors based on real data collected in Beijing. Our evaluation results show that, for the two baseline predictors, while in areas with high $\Pi^{\max}$ the Markov-chain predictor is both faster and more accurate, the NN predictor significantly outperformed the Markov-chain predictor in areas with lower $\Pi^{\max}$. This justifies our claim that different prediction algorithms should be used in different areas. For the ensemble learning approach, the boosted NN predictor further reduces the prediction error by up to about 21% in areas with low $\Pi^{\max}$, and about 9% in areas with high $\Pi^{\max}$. Finally, we also use our evaluation to show the effect of our prediction: if passengers in different functional areas follow our prediction and choose the lowest prices nearby, how many of them could get a lower price.

The remainder of this paper is organised as follows. Section 2 presents some background information and our dataset. In Section 3, we discuss the maximum predictability of price multipliers and use it to characterise city cells and areas. Two baseline predictors, as well as the boosted-NN predictor, are implemented in Section 4 and we also evaluate the performance and effect of our prediction. Section 5 discusses related work and Section 6 concludes the paper.

2 Background and dataset

2.1 RoD services

Taxi service is probably the oldest RoD service: people could request a ride when they wish. The pricing scheme is mostly fixed and is dependent on the time and distance between two locations. In recent years, some new RoD service providers have come into the market, and the major difference between them and taxi service is the use of dynamic pricing scheme. The price of a trip now is the product of a dynamic price multiplier and the normal price (comparative to the price in a taxi). Another difference is that new services rely on global positioning system (GPS)-assisted mobile apps to accurately locate both drivers and passengers.

In this subsection, we explain the user interface of a mobile app of a typical RoD service, shown in Fig. 1, to give a generic explanation of how such a service works. Usually, a passenger first opens the app on his/her mobile phone when she/he wants to travel from a boarding location A to an arriving location B, and types the address of both locations in the app. The passenger could also choose 'when to ride (now or several minutes later)' and 'using coupon'. After the passenger has specified the locations and chosen all available options, the mobile app sends all the information to the service provider, obtains in return (a) the estimated trip fare and (b) the current dynamic price multiplier, and displays them. The price multiplier reflects the current supply and demand condition around the boarding location A. The service provider sets a lower and upper bound on the price multiplier in the service policy. The passenger then chooses either to accept the current price (by pressing 'Ride a Car!' button) or give up the current fare estimation if she/he considers the price multiplier too high.

2.2 Anxiety from dynamic pricing

Passengers' anxiety comes from their uncertainty about the dynamic prices: they do not know whether the current dynamic price multiplier is low enough, and, whether they could get a lower one if they choose to wait for several minutes or to walk away for hundreds of metres.

The possibility of getting a lower price has been validated in [2]. The authors of [2] conclude that during rush hours if a passenger is getting a too high price multiplier, she/he could usually walk away for 1–2 km to get a lower one. Also, based on the analysis of our data used in this study, we find that during morning rush hours (i.e. 7–9 am) the probability of finding a lower average price multiplier within 1 km is about 75.99%. This probability is 76.10 and 34.21% for evening rush hours and non-rush hours.

Passengers' reaction to dynamic prices also indicates their anxiety. Guo *et al*. [3] measured passengers' reaction, i.e. how many times they perform fare estimations before finally giving up or getting on a car. As mentioned in Section 2.1, a passenger may choose to give up or accept the fare estimation. If she/he considers the price multiplier is too high, she/he may choose to wait for a while or walk away and estimate the fare again. Guo *et al*. [3] show that only in 39.77% cases one accepts the price multiplier after only one fare estimation. This justifies that most passengers feel uncertain about the dynamic price multiplier.

2.3 Our dataset

Our data of the RoD service is collected from Shenzhou UCar (http://www.10101111.com/), one of the major RoD service providers in China. By the end of 2015, Shenzhou UCar's service covers more than 50 cities in China, with a fleet of more than 30,000 cars, offering more than 300,000 trips per day [4].

We collect the event-log dataset from Shenzhou UCar's service in Beijing, China. The event-log dataset contains two major events:

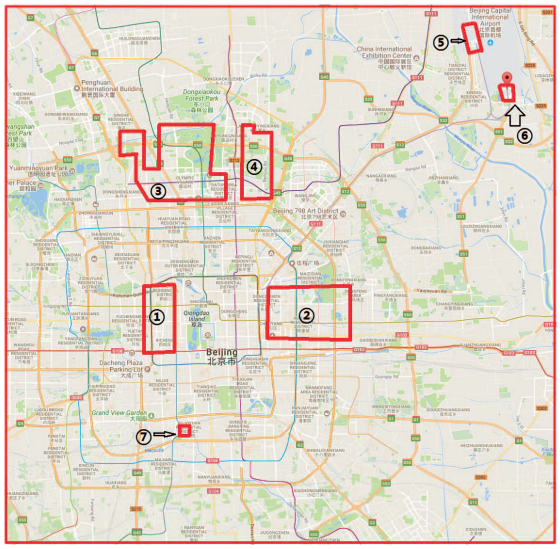

Fig. 2 *Map of Beijing and typical functional areas*

EstimateFee and *CreateOrder* event. The *EstimateFee* event is triggered when the mobile app sends all the information of a passenger's request (including the two locations, the requested group of cars, the time of the request etc.) to the service provider, and returns the current price multiplier and the estimated trip fare. When one performs multiple fare estimations, the same number of *EstimateFee* events are generated. The *CreateOrder* event is triggered only when the passenger is satisfied with the current price multiplier and presses the 'Ride a Car!' button.

The event-log dataset contains the complete record of events in the complete four weeks in October 2016 (from 3 October to 30 October) in Beijing. Each entry includes the time it happened, the event code (i.e. *EstimateFee* or *CreateOrder*), the international mobile equipment identity (IMEI) of the passenger's device, location information, the estimated trip fare and the price multiplier for *EstimateFee* etc. The volume of the dataset is about 5.3 million, and all entries are properly anonymised.

We also find out from the event-log dataset that the service provider sets a lower and upper bound for the price multiplier. The lower bound is $m = 1.0$ and the upper bound is $U = 1.6$.

We also collect climate data for building NN predictor. Specifically, we collect the daily precipitation and hourly temperature data reported at the Beijing Capital International Airport from TuTiempo (https://en.tutiempo.net/records/zbaa). The granularity of the precipitation data may not be small enough: we could not find the hourly precipitation data, and the precipitation data at the airport do not represent the precipitation in other locations of Beijing. The temperature data do not have this problem.

3 Predictability of price multipliers

In this section, we first define city cells and areas of Beijing and present the variation of price multipliers in different city areas. We then define the *maximum predictability* $\Pi^{\max}$ and show the distribution of $\Pi^{\max}$. The difference of $\Pi^{\max}$ in various city cells and areas makes it necessary to use different prediction algorithms.

3.1 City cells and areas

To show the predictability of price multipliers in different locations of a city, we first divide the city map into $N \times N$ rectangular cells of equal sizes, and cell (i, j) denotes the cell on row i, column j $(i, j = 1, 2, \ldots, N)$. In this study, we set $N = 100$, so that each cell is small enough to enable finer granularity of data analysis, and is also large enough to have enough events happening inside each cell. When $N = 100$ each cell is about 420×300 m^2.

In addition to cells, we also consider a larger and also more representative unit – the functional areas of a city. A large city always has a clear partitioning of functional areas including, for example:

- *Business area*: the place for working. Different industries (e.g. financial or IT) may have different areas.
- *Residential area*: the place for living. In China, some large residential areas accommodate $\geq 10,000$ residents.
- *Transportation area*: typical transportation areas include airport terminals, railway stations for inter-city trains etc.

The partitioning of functional areas could be performed either from the city plan of Beijing or by analysing real data. The task of identifying functional areas of a large city such as Beijing is a sophisticated task and has been studied using various techniques based on different sources of data. For example, in our previous work [2], we cluster the boarding and arriving locations of trips from frequent passengers (i.e. those passengers having more trips monthly) in RoD service using the *k*-means algorithm and verify that the clustering result matches the city plan as well as our knowledge of the distribution of functional areas in Beijing.

Fig. 2 shows some typical functional areas found in [2] and validated by the city plan: areas 1 and 2 are typical business areas; areas 3 and 4 are typical residential areas; and areas 5–7 are the airport terminals and a major train station – typical transportation areas. We use these typical functional areas in the remainder of the paper.

3.2 Variation of price multipliers

The target of our study is the price multiplier passengers obtain in *EstimateFee* events. We divide the whole time range of data (i.e. four weeks) into K time slots with equal length Δt and calculate the average price multiplier in all *EstimateFee* events. So for each city cell/area, we obtain a sequence of length K, and the kth element R_k is the average price multiplier in the kth time slot. In practice, we choose $\Delta t = 1$ h, so $K = 24 \times 7 \times 4 = 672$, and we thus focus on the hourly average price multiplier in each city cell/area.

The variation of price multipliers may have different regularities in different locations. In Fig. 3, we give the variation of the hourly average price multiplier in typical business, residential and transportation area during the four-week time range of our data (from 0:00 on Day 0 to 23:59 on Day 27).

The variation in transportation area (Fig. 3*c*) exhibits a strong temporal pattern. The price multiplier is higher than 1.2 from 11 pm to 5 am. During the evening (from about 6 pm to 11 pm), the price multiplier is always 1.1. In other hours of a weekday, the price multiplier is always as low as 1.0. The highest price multiplier appears at the end of Sundays (in most cases it is 1.4).

To the other extreme, the variation in the business area (Fig. 3*a*) seems to be more random. There is no clear temporal pattern in price multiplier, except that during weekends the multiplier is lower (smaller than 1.1× in most cases).

There may be many reasons why the variation of price multipliers have different regularities in different locations, such as different travel patterns, different compositions of passengers' intention etc. In this study, we do not try to find out these reasons; instead, we define the regularity mathematically and study how to take it into account in predicting price multipliers.

3.3 Maximum predictability $\Pi^{\max}$

To characterise the regularity of price multiplier, we try to answer the following problem, and define the maximum predictability $\Pi^{\max}$:

Problem 1.: *Given a sequence of the price multipliers* $R^{(i,j)} = \left\{R_k^{(i,j)}, k = 1, 2, \ldots, L, L \leq K\right\}$ *of cell* (i, j)*, from time slot* $1, 2, \ldots, L$*, considering both the randomness and the temporal correlation of the price multipliers, we want to find out the maximum predictability* $0 \leq \Pi^{\max} \leq 1$ *(the highest potential accuracy of prediction) that any prediction can reach.*

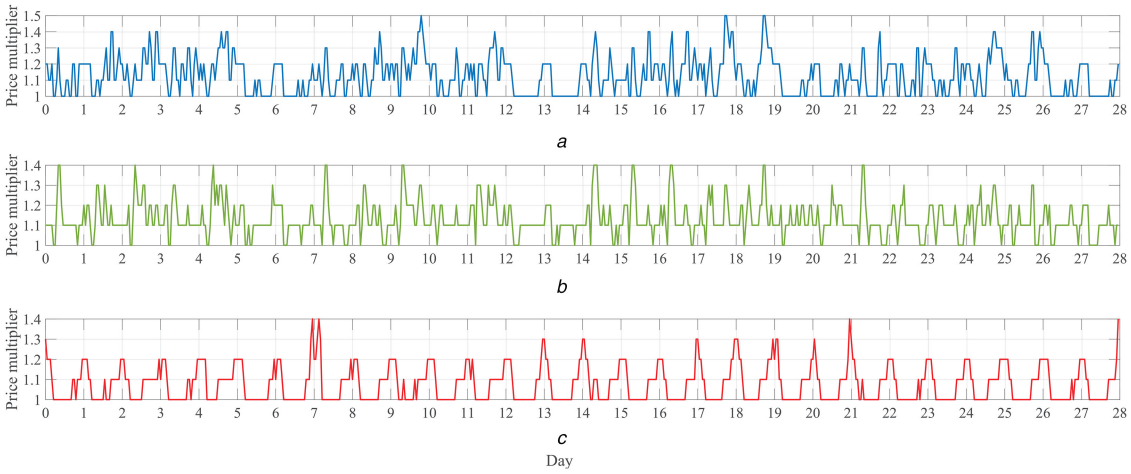

Fig. 3 *Variation of price multipliers in different functional areasway*
(a) Business area, ***(b)*** Residential area, ***(c)*** Transportation area

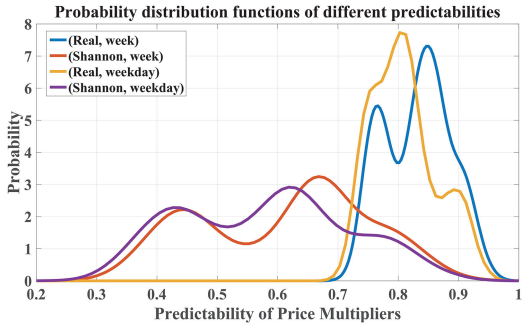

Fig. 4 *PDFs of predictabilities in city cells*

Table 1 Predictability in different functional areas

	Business	Residential	Transportation
Π^{Shannon}	0.4739	0.6677	0.6107
Π^{real}	0.8322	0.8622	0.9554

Note that in the problem definition, we define the predictability on a sub-sequence of price multipliers: the length of the sub-sequence is $L \leq K$. This is because in predicting price multipliers, we need to train the algorithm based on a training set, which is usually a sub-sequence of the original data.

The problem of maximum predictability has been discussed in a number of previous works about the prediction of, for example, taxi demand, human mobility etc. The calculation of $\Pi^{\max}$ is based on the entropy of $R^{(i,j)}$, and using different entropies have a different meaning on the predictability. Here we discuss the Shannon entropy and the real entropy.

The Shannon entropy measures only the uncertainty of price multipliers and does not take into account the temporal correlation. On the other hand, the real entropy [5, 6] tries to consider both. In (1) and (2), $p\left(r_m^{(i,j)}\right)$ is the probability of multiplier $r_m^{(i,j)}$ in cell (i, j), and $p\left(s^{(i,j)}\right)$ is the probability of finding a particular time-ordered sub-sequence $s^{(i,j)}$ in $R^{(i,j)}$

$$S_{\text{Shannon}}^{(i,j)} = -\sum_{m=1}^{N^{(i,j)}} p\left(r_m^{(i,j)}\right)\log_2 p\left(r_m^{(i,j)}\right), \tag{1}$$

$$S_{\text{real}}^{(i,j)} = -\sum_{s^{(i,j)} \in S^{(i,j)}} p\left(s^{(i,j)}\right)\log_2 p\left(s^{(i,j)}\right). \tag{2}$$

We could see that unlike the Shannon entropy, the real entropy considers not only the frequency of different price multipliers but also the order of the temporal patterns of price multipliers.

Based on the two entropy measures for each cell, we could then calculate the corresponding predictability measure: the success rate that the most accurate algorithm could correctly predict the future price multiplier at a particular cell. The predictability measure Π is subject to Fano's inequality and could be calculated by [5]

$$S = -\Pi\log_2(\Pi) - (1-\Pi)\log_2(1-\Pi) + (1-\Pi)\log_2\left(N^{(i,j)} - 1\right). \tag{3}$$

In (3), we obtain Π^{Shannon} and Π^{real} when we let $S = S_{\text{Shannon}}^{(i,j)}$ and $S_{\text{real}}^{(i,j)}$, respectively. It has been proven in [5] that $\Pi^{\text{Shannon}} \leq \Pi^{\text{real}}$ so that the maximum predictability is $\Pi^{\max} = \Pi^{\text{real}}$. This indicates that if we want to have a more accurate prediction, we need to consider the temporal patterns of price multipliers.

Now we show the $\Pi^{\max}$ in different city cells/areas. For the $N \times N$ cells, we count the number of fare estimations in each cell during the four-week period and only consider those with the number higher than the median value. In other words, we exclude the cells without enough passenger activity. After the exclusion, the total number of cells to study is 3760.

In Fig. 4, we show the probability distribution function (PDF) of the predictability measures in all these cells: $\Pi^{\max}\left(=\Pi^{\text{real}}\right)$ and Π^{Shannon} when we consider the whole dataset, and when considering only the weekdays in the four-week period. We also calculate the Π^{Shannon} and Π^{real} for the business, residential and transportation areas and show the results in Table 1.

We have the following observations on the predictability:

i. It is necessary to consider temporal correlation in price multipliers. The mean Π^{real} is 0.8315 and 0.8089 for all the days and for weekdays only; while the counterpart of Π^{Shannon} is only 0.6097 and 0.5789. Similarly, considering temporal correlation also brings a 75.61, 29.13 and 56.44% increase in predictability for business, residential and transportation area, respectively.
ii. It is necessary to consider all the days instead of only weekdays. This is a trade-off between regularity and data size. Price multipliers of only weekdays may have more regularity, but meanwhile, the data size is larger when considering all the days. Fig. 4 shows that considering the whole week provides higher predictability.
iii. Predictability measures could also be used to characterise city areas, and this agrees with our observation in Section 3.2 and Fig. 3. Business area has the lowest $\Pi^{\max}$ at 0.8322 and transportation area has the highest $\Pi^{\max}$ at 0.9554, and there is a significant difference in $\Pi^{\max}$ between these functional areas.

4 Predicting price multipliers

In this section, we first implement two baseline predictors, namely a Markov-chain and a NN predictor, to predict future price multipliers based on historical data. We then use an ensemble learning approach and build a boosted-NN predictor. These predictors are evaluated in different city cells/areas.

4.1 Baseline predictors

4.1.1 Markov-chain predictor: In a high-order Markov-chain predictor, the price multiplier in the next time slot is determined only by the multiplier in the current and a few past time slots. Assuming the order is q and that we already have the historical data $R^{(i,j)} = \left\{R_k^{(i,j)}, 1 \le k \le L\right\}$, an order q Markov-chain predictor predicts the next multiplier $R_{L+1}^{(i,j)}$ based on the current and past price multipliers $R_{L-q+1}^{(i,j)}, \ldots, R_{L-1}^{(i,j)}, R_L^{(i,j)}$. The Markov property assumes that the transition probability $P\left(R_{L+1}^{(i,j)} \mid R_{L-q+1}^{(i,j)}, \ldots, R_{L-1}^{(i,j)}, R_L^{(i,j)}\right)$ is independent of the current time slot L

$$\begin{aligned} &P\left(R_{L+1}^{(i,j)} \mid R_{L-q+1}^{(i,j)}, \ldots, R_{L-1}^{(i,j)}, R_L^{(i,j)}\right) \\ &\quad = P\left(R_t^{(i,j)} \mid R_{t-q}^{(i,j)}, \ldots, R_{t-2}^{(i,j)}, R_{t-1}^{(i,j)}\right), \quad \forall q < t \le L. \end{aligned} \tag{4}$$

The transition probabilities form the transition matrix $\boldsymbol{T}$, and each element of $\boldsymbol{T}$, denoted by $\boldsymbol{T}_{c,c'}$, is

$$\boldsymbol{T}_{c,c'} = P\left(R_t^{(i,j)} = c' \mid R_{t-q}^{(i,j)}, \ldots, R_{t-2}^{(i,j)}, R_{t-1}^{(i,j)} = c\right). \tag{5}$$

Note that in (5), c is a sequence of price multipliers of length q, and c' is a single price multiplier.

The probability transition matrix $\boldsymbol{T}$ is derived from the historical data. We first find out every tuple of length q (i.e. a series of price multipliers in q consecutive time slots) from the historical data of length L $\left(\text{i.e. } R^{(i,j)} = \left\{R_k^{(i,j)}, 1 \le k \le L\right\}\right)$. The set of distinct tuples are the rows of $\boldsymbol{T}$. For each distinct tuple c, we find out all the possible next price multipliers in every occurrence of tuple c. The set of all distinct next price multipliers are the columns of $\boldsymbol{T}$. For each distinct next price multiplier c', we obtain the corresponding element $\boldsymbol{T}_{c,c'}$ of $\boldsymbol{T}$ – the probability of the next price multiplier c' occurring immediately after the presence of a tuple c in $R^{(i,j)}$.

Predicting the price multiplier in the next time slot is straightforward based on the learned matrix $\boldsymbol{T}$ and historical data. In general, it consists of calculating the next price multiplier, appending to the historical data and updating the matrix $\boldsymbol{T}$. Algorithm 1 summarises the Markov-chain predictor.

Algorithm 1*:* The Markov-chain predictor

Input: the Markov-chain order q, historical data of price multipliers $R^{(i,j)} = \left\{R_k^{(i,j)}, 1 \le k \le L\right\}$, transition matrix $\boldsymbol{T}$.

1: Extract the most recent q price multipliers c from $R^{(i,j)}$, i.e. $c = \left(R_{L-q+1}^{(i,j)}, \ldots, R_{L-1}^{(i,j)}, R_L^{(i,j)}\right)$.
2: **if** c exists in $\boldsymbol{T}$'s rows **then**
3: Find in this row the largest value $\boldsymbol{T}_{c,c'}$.
4: return c'.
5: **else**
6: Find the most frequent price multiplier appearing in $R^{(i,j)}$, denoted by c'.
7: return c'.
8: **end if**
9: Append c' to $R^{(i,j)}$ to form the new historical data $R^{(i,j)} = \left\{R_k^{(i,j)}, 1 \le k \le L+1\right\}$.
10: Update the transition matrix $\boldsymbol{T}$ based on $R^{(i,j)}$.

4.1.2 NN predictor: Unlike the Markov-chain predictor that predicts the next multiplier solely based on the temporal correlation, in the NN predictor we are able to include more factors that decide the prediction result. In our implementation, we consider four features. Note that we have already stated that the length of each time slot is 1 h, so we actually collect hourly average price multiplier in each city cell/area. The features are as below:

1. *Hour-of-a-day*: the value ranges from 0 to 23, and it represents the hour of a particular day.
2. *Day-of-a-week*: the value ranges from 0 to 6, and it represents the day of a week (from Monday to Sunday).
3. *Daily precipitation*: we obtain the daily precipitation (in millimetres) in Beijing International Airport from a public data source. Note that this data may not be accurate enough, as the precipitation around the airport may not be the precipitation around a particular city cell or functional area. Also, we only have the daily precipitation instead of hourly data, and this add up to the inaccuracy. However, we consider this enough to catch the essence, and in the future, we will try to get more accurate data to refine our NN predictor.
4. *Hourly temperature*: similarly, we also obtain the hourly temperature (in °C) in the airport from a public data source. This data is better than the precipitation data because (a) this is an hourly data and (b) the temperature around the airport is more representative of the temperature in other city cells/areas.

Our NN predictor uses a two-layer structure: a ReLU activation layer follows the input layer and then a Softmax output layer. The data fed to the input layer is a four-element tuple (i.e. values of the four features above), and the output from the Softmax layer is a categorical value ranging from 1 to 7 (each possible value represents a possible price multiplier, so in our data we have seven categories as the service provider sets the lower and upper bound of price multiplier to be 1.0 and 1.6).

4.2 Ensemble learning and boosted-NN predictor

Ensemble learning is a technique to improve weak learners to become strong learners. Under the setting of our paper, a weak learner is a prediction model that may not have a high accuracy, as long as its performance is better than random guessing. A strong learner, as its name suggests, is a prediction model that has higher accuracy. In ensemble learning, the prediction result is a weighted combination of multiple weak learners. In general, there are three categories of ensemble learning models:

- *Bagging*: In the bagging model, the training set of each weak learner is a subset of the original training set, and is sampled by randomly selecting training data with replacement. So different weak learners use different training sets and the final prediction is based on the majority voting scheme.
- *Boosting*: In the boosting model, predictors are trained sequentially, and the next predictor assigns more weights to the mistakes that the previous predictor made. So those wrongly predicted data become more important in the next prediction.
- *Stacking*: In the stacking model, outputs of the first level of predictors are inputs for the second level.

In this subsection, we use the boosting model and choose the NN baseline predictor as the weak learner to be improved. The reason for not choosing the bagging model is that the number of training data is relatively small (smaller than $L = 672$), so a sampled training set is even smaller and may not be accurate enough. A famous algorithm in the boosting model is AdaBoost [7], and for the multi-class prediction, the stage-wise additive modelling using a multi-class exponential loss function (SAMME) [8] algorithm is one of the most frequently used. Our prediction target is multi-class, as we have seven categories in the output for the price multiplier 1.0–1.6.

SAMME is a general multi-class boosting algorithm. Assuming the number of training data is n, the input feature and output

prediction result are x_k and $y_k (1 \le k \le n)$, respectively, and the number of class is S. It first initialises equal observation weights $\omega_k = (1/n)(1 \le k \le n)$ for these data and then runs the following boosting process for M times. For the mth $(1 \le m \le M)$ boosting process

1. SAMME first fits a predictor $T^{(m)}(x)$ to the training data using weights ω_k.
2. The weighted error rate err^m is calculated

$$\text{err}^m = \frac{\sum_{k=1}^{n} \omega_k I\left(y_k \ne T^{(m)}(x_k)\right)}{\sum_{i=k}^{i=k} \omega_k}. \quad (6)$$

3. With the error rate, the weight of this predictor $T^{(m)}(x)$ is calculated as $\alpha^{(m)} = \log((1 - \text{err}^{(m)})/(\text{err}^{(m)})) + \log(S - 1)$.
4. The weight ω_k of the training data is updated

$$\omega_k \leftarrow \omega_k \exp\left(\alpha^{(m)} I\left(y_k \ne T^{(m)}(x_k)\right)\right). \quad (7)$$

5. After the last step, those wrongly predicted training data have their weights increased. Now we normalise all the weights to make all ω_k sum to 1.

After M times of boosting process, we have M predictors $T^{(m)}, 1 \le m \le M$. Also, the way we combine them together is

$$T(x) = \arg \max_{1 \le s \le S} \sum_{m=1}^{M} \alpha^{(m)} * I\left(T^{(m)}(x) = s\right). \quad (8)$$

In other words, the output of the combined predictor is the class that has the highest weight.

During any of the above boosting process, if the calculated error rate is $\text{err}^m \ge ((S - 1)/(S))$, then the corresponding weight of the predictor α_m will be negative. So, this requires a stopping criterion: if the error rate is greater than $((S - 1)/S)$, we should break the boosting process, and combine the predictors that have already been generated.

In this study, we use a variant of SAMME and build a boosting model based on the NN predictor in Section 4.1.2. Note that in the SAMME algorithm mentioned above, a weight is applied to each training data, but when training a NN, the weight could not be applied to training data. Hence, we modify the original SAMME algorithm: instead of training a predictor using weights on the data, we re-sample the training data with replacement and weights ω_k in every boosting process. We call our predictor as the boosted-NN predictor, and Algorithm 2 summarises the whole training process. Similar to the NN predictor, the input to the boosted-NN predictor is the four-feature vector, and the output is a seven-category value, ranging from 1 to 7, corresponding to price multipliers from 1.0 to 1.6.

***Algorithm 2**:* The boosted-NN predictor

Input: For each cell (i, j), the training set is $R_{\text{train}}^{(i,j)} = \left\{R_k^{(i,j)}, 1 \le k \le L_{\text{train}}\right\}$, and the number of class is 7. The input four-feature vector is denoted by $x_k, (1 \le k \le L_{\text{train}})$.

1: Initialise equal weights $\omega_k = 1/L_{\text{train}}, 1 \le k \le L_{\text{train}}$.

2: **for** $m = 1$ to M **do**

3: Train a NN predictor to the training set. Each data point has a weight ω_k, but these weights are not applied to predictor training.

4: Calculate the weighted error rate

$$\text{err}^m = \frac{\sum_{k=1}^{L_{\text{train}}} \omega_k I\left(R_k^{(i,j)} \ne T^{(m)}(x_k)\right)}{\sum_{k=1}^{L_{\text{train}}} \omega_k}.$$

if $\text{err}^m \ge 6/7$, then stop the boosting process.

5: Calculate the weight of this predictor $T^{(m)}(x)$

$$\alpha^{(m)} = \log\frac{1 - \text{err}^{(m)}}{\text{err}^{(m)}} + \log(6).$$

6: Update the weight of the training set

$$\omega_k \leftarrow \omega_k * \exp\left(\alpha^{(m)} * I\left(R_k^{(i,j)} \ne T^{(m)}(x_k)\right)\right).$$

7: Normalise all ω_k so that $\Sigma_{k=1}^{L_{\text{train}}} \omega_k = 1$.

8: Based on all ω_k, re-sample the training set with replacement and weights ω_k, and form the new training set for the next m.

9: **end for**

10: Combine M predictors $\left\{T^{(m)}(x), 1 \le m \le M\right\}$ to become the boosted-NN predictor $T(x)$, such that

$$T(x) = \arg \max_{1 \le s \le 7} \sum_{m=1}^{M} \alpha^{(m)} * I\left(T^{(m)}(x) = s\right).$$

4.3 Evaluation setup

We discuss how to evaluate the performance of our price multiplier predictors from three perspectives: preparing the training and test set, choosing evaluation metrics, and sampling city cells.

4.3.1 Training and test set: Both the Markov-chain predictor and NN predictor need training and test set. In our dataset, we split the four-week data into four parts – one-week data for each part. We then use any one part as the test set and the remaining three parts as the training set. As a result, the length of the test set, L_{test}, is 168 (=24 × 7) and the length of the training set, L_{test}, is 504 (=3 × 24 × 7). For the Markov-chain predictor, the first three weeks of data are the training set, and the last week of data are the test set. We first generate the transition matrix $\boldsymbol{T}$ based on the training set, and then successively predict the next multiplier until we reach the end of the fourth week, while at the same time updating $\boldsymbol{T}$ in each prediction.

For the NN predictor, we run it for 500 times for each city cell/area, and in each run, we randomly choose the test set (and the corresponding training set) and obtain the evaluation result as an average among multiple runs. After training parameters on the training set, the features corresponding to each hour in the week of the test set are then fed into the network, with the output as the test results to be compared by our ground-truth data.

The procedure for the boosted-NN predictor is similar, except that in each run, we successively train M NN predictors, and combine them together. We test different choices of M and find that $M = 20$ is enough for the NN predictors to converge. In other words, the evaluation metric of the resulting boosted-NN predictor will not change after training 20 NN predictors.

4.3.2 Evaluation metrics: The usual way to evaluate the performance of a prediction algorithm is based on the accuracy measure, i.e. how many of the predicted items are equal to the corresponding ground-truth items. In other words, if we use $R_{\text{test}}^{(i,j)} = \left\{R_{test,k}^{(i,j)}, k = 1, 2, \ldots, L_{\text{test}}\right\}$ to denote the price multiplier sequence in cell (i, j) of the test set and use $R_{\text{pred}}^{(i,j)} = \left\{R_{\text{pred},k}^{(i,j)}, k = 1, 2, \ldots, L_{\text{test}}\right\}$ to denote the predicted price multiplier sequence, then the metric is $\text{Count}_{\text{equal}}^{(i,j)} = \sum_{k=1}^{L_{test}} \delta_k^{(i,j)}$, where $\delta_k^{(i,j)} = 1$ if $R_{\text{pred},k}^{(i,j)} = R_{\text{test},k}^{(i,j)}$, and is 0 otherwise.

In predicting price multipliers, on the other hand, we do not care that much about the accuracy measure. In some cases, even though there is a slight difference between the predicted price multiplier and the ground truth, it is not a problem for passengers. For example, a user faced with a price multiplier 1.3 in a particular location may only want to know if there is any possibility to get a multiplier lower than 1.3 in neighbouring locations or in a short time but does not care about whether it is 1.1 or 1.2.

In our evaluation, we use the symmetric mean absolute percentage error (sMAPE) [9], an accuracy measure based on the relative difference (error)

$$\text{sMAPE}^{(i,j)} = \frac{1}{L_{\text{test}}} \sum_{k=1}^{L_{\text{test}}} \frac{\left| R^{(i,j)}_{\text{pred},k} - R^{(i,j)}_{\text{test},k} \right|}{R^{(i,j)}_{\text{pred},k} + R^{(i,j)}_{\text{test},k}}. \quad (9)$$

To sum up, a higher sMAPE means lower prediction accuracy and a lower $\text{Count}_{\text{equal}}$ means the same.

4.3.3 Sampling of city cells: We evaluate the predictors in both functional areas and city cells. The limited number of functional areas in our study makes it easy to evaluate the predictors, but the 3760 city cells (see Section 3.3) may be a large number to evaluate.

We sample some representative city cells for the evaluation. We first sort all the city cells according to the ascending order of their maximum predictability $\Pi^{\max}$ and then split them into ten groups of equal sizes. In our case, each group contains 376 city cells. For example, the first group (denoted by '0–10%') means the bottom 10% cells according to $\Pi^{\max}$. For each group, we take out the median 10 cells as samples and use the average sMAPE of these 10 cells as the sMAPE of this group of cells. In all, we have 100 sample cells. Table 2 summarises the median $\Pi^{\max}$ of sample cells in each group.

4.4 Performance evaluation of predictors

In the following evaluation, the order used in the Markov-chain predictor (Section 4.1.1) is set to $q = 3$, as we find from running the predictor that increasing q over this value does not significantly increase the accuracy of the prediction. For the NN predictor (Section 4.1.2), the dimension of the input layer is 4, as we have four features; the number of neurons in the ReLU activation layer is 5, and this is the result of our parameter tuning; the number of neurons in the Softmax output layer is 7, as the range of price multiplier is from 1.0 to 1.6.

We first run all predictors on every sampled cell for 500 times and calculate the average sMAPE values. In Fig. 5 we show the sMAPE values for all predictors in the ten sampled groups: group 1 means the '0–10%' cells; group 2 means the '10–20%' cells, and so on.

The functional area is a much larger unit, and we also run all predictors on the three functional areas: business, residential and transportation area. In Table 3, we show the result, including the sMAPE and $\text{Count}_{\text{equal}}$ (only for reference), of running these predictors.

For the baseline predictors, the Markov-chain predictor has a prediction accuracy decreasing quickly when the maximum predictability decreases. For the group with the highest median of $\Pi^{\max} = 0.9196$, the average sMAPE of the prediction is as low as 0.0372, and for the group with the lowest median, the average sMAPE climbs to 0.0702–88.7% higher. Comparatively, the prediction accuracy of the NN predictor is much more stable with different $\Pi^{\max}$.

Comparing the sMAPE of baseline predictors in Fig. 5, when a city cell's $\Pi^{\max} \leq 0.8564$, the NN predictor has higher prediction accuracy; otherwise, the Markov-chain predictor performs better.

Regarding the prediction accuracy in functional areas, we compare the results in business and transportation area. Residential area stands in between, so comparing the two extremes is enough. We have the following observations:

i. For a functional area with high predictability (e.g. the transportation area), while the NN predictor brings a small increase of 15.1% in $\text{Count}_{\text{equal}}$, it results in a sMAPE 40.2% higher, compared with the Markov-chain predictor.
ii. For a functional area with low predictability (e.g. the business area), the NN predictor not only increases the $\text{Count}_{\text{equal}}$ by 90.3% but also results in an 18.2% lower sMAPE.

Considering the fact that the NN predictor needs more computation time (in our case about 20 times that of the Markov-chain predictor) and more feature data (i.e. the weather data in our case), we conclude that only in functional area with low predictability the NN predictor is necessary, and the Markov-chain predictor provides enough prediction accuracy in other functional areas with higher $\Pi^{\max}$. This justifies our earlier claim that we should use different prediction algorithms in different city cells/functional areas.

Table 2 $\Pi^{\max}$ of sample cells in each group

Group, %	Median $\Pi^{\max}$	Group, %	Median $\Pi^{\max}$
0–10	0.7460	50–60	0.8436
10–20	0.7632	60–70	0.8564
20–30	0.7799	70–80	0.8705
30–40	0.8107	80–90	0.8932
40–50	0.8306	90–100	0.9196

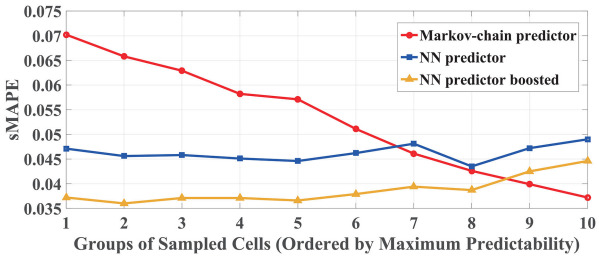

Fig. 5 *SMAPE of all three predictors in city cells*

Table 3 sMAPE of predictors in functional areas

	Business	Residential	Transportation
	Markov-chain predictor		
$\text{Count}_{\text{equal}}$	47.3	54.5	71.6
sMAPE	0.0548	0.0468	0.0366
	NN predictor		
$\text{Count}_{\text{equal}}$	90.2	86.3	82.4
sMAPE	0.0448	0.0457	0.0513
	boosted-NN predictor		
$\text{Count}_{\text{equal}}$	90.5	85.2	82.1
sMAPE	0.0367	0.0366	0.0487

The ensemble learning is used to improve the prediction accuracy based on the NN predictor, which is already better than the Markov-chain predictor in most cases. Regarding the effects of the boosting model, we have the following observations:

i. The boosted-NN predictor always has higher prediction accuracy than the NN predictor, and now for those city cells with $\Pi^{\max} \leq 0.8705$ the boosted-NN predictor performs better than the Markov-chain predictor. In particular, for the '0–10%' group of cells, the boosted-NN predictor generates a sMAPE only 52.28% of that of the Markov-chain predictor.
ii. For the improvement compared with the NN predictor, the boosted-NN predictor reduces the sMAPE by up to 21% when $\Pi^{\max}$ is smaller, and the reduction falls to about 9% in the group with the highest $\Pi^{\max}$. In other words, the improvement is greater for less predictable locations.

To sum up our evaluation results, we have

i. For locations with highly predictable price multipliers (i.e. $\Pi^{\max} > 0.8564$), the Markov-chain predictor is enough.
ii. For locations with less predictable multipliers (i.e. $\Pi^{\max} \leq 0.8564$), the NN predictor gives a more accurate prediction.
iii. For some more unpredictable locations in the city, using the boosted-NN predictor could further reduce the prediction inaccuracy by up to 21% compared with the NN predictor, but may take much longer time to finish.

4.5 Effects of price multiplier prediction

We now return to our initial goal of predicting price multipliers – relieving the anxiety from dynamic pricing. The anxiety comes from passengers' uncertainty about the price multipliers nearby, or within a short time. We envision that our work on price multiplier prediction could reduce this uncertainty and make passengers informed about '*how to get a lower price multiplier*?'. We try to answer:

- *Q1:* If there are indeed chances to get a lower price multiplier in neighbouring cells at the same time, by what probability could our price multiplier prediction find out such a chance?
- *Q2:* With our price multiplier prediction, by what probability a passenger in a particular location could really get a lower price multiplier in neighbouring cells at the same time?

Before answering these questions, we first define 'neighbouring cells': for cell (i, j), its neighbouring cells are $\text{Neighbour}^{(i,j)} = \{(i', j') | i - 1 \leq i' \leq i + 1, j - 1 \leq j' \leq j + 1\}$ ..

We only consider those passengers in business and transportation areas, as these are two representative functional areas with low and high price multiplier predictability, respectively. We use the Markov-chain predictor in the transportation area and the NN predictor in the business area.

For *Q1*, if we indeed find from our data that a passenger standing in a cell in a particular functional area is able to get a lower price multiplier in eight neighbouring cells, our evaluation shows that on average in the transportation area the probability that our predictor could find out such a chance is 89.4%, and in the business area the probability is 82.3%.

For *Q2*, the difference with *Q1* is that we have to consider the fact that in some cells there is not any lower price multiplier in neighbouring cells. Similarly, on average in transportation area the probability a passenger could use the price multiplier prediction to get a lower price in neighbouring cells is 18.7% – because in transportation area the price multiplier is relatively stable and low (see Section 3.2). In the business area, the probability is much higher and reaches 35.4%.

For *Q2*, we also try to focus only on those cells on the edge of a functional area. In practice, passengers in these cells are easier to get lower multipliers, as walking away to a neighbouring cell may enter a different functional area. Our evaluation shows that in the edge cells of transportation area, the corresponding probability rises to 31.2%. The probability in the business area rises to 67.3%.

These results verify that (a) price multiplier prediction is effective in obtaining lower prices and thus relieving passengers' anxiety and (b) passengers standing on the edge of functional areas are easier to find lower prices.

5 Related work

RoD service. Most studies on emerging RoD services are centred on dynamic pricing. Chen *et al*. [1] tried to evaluate Uber's surge pricing mechanism based on the measurement treating Uber as a black-box. In our previous work [2, 3], the demand pattern, the effect of dynamic pricing, and passengers' reaction to dynamic pricing in RoD service have been carefully studied and analysed. Other works focus on economic analysis of the effect and impact of dynamic pricing [10], the supply elasticity [11], consumer surplus [12] etc.

Taxi service. Our work on price multiplier prediction is inspired by previous work on taxi demand prediction. The availability of public taxi dataset leads to a number of related studies. Examples include using a NN to forecast the taxi demand from historical data [13], using SVM to select the most related feature that determines the taxi demand [14], using taxi GPS trajectories to detect anomalous trips [15] etc. Besides, taxi data have also been used in other applications such as crowd-sourcing [16, 17], trip purpose inference [18], traffic data collection [19, 20] and data dissemination [21, 22].

Temporal patterns and predictability. Many studies have found that urban human mobility has strong regularities: people usually go to work, go back home, go shopping/entertaining at specific time and locations. For example, Gonzalez *et al*. [23] found by studying cell phone user's location records and show that human mobility exhibits a high degree of the spatio-temporal pattern. Furthermore, Song *et al*. [5] employed the concept of maximum predictability and uses it to study the temporal pattern of individual human mobility. Similarly, Zhao *et al*. [24] also use this concept to choose different algorithms to predict taxi demand at each building block in New York.

Ensemble learning methods. Ensemble learning is a sophisticated and widely-used technique to improve and combine weak learners to become a strong learner. There are three main categories of methods, including bagging [25], boosting [7] and stacking [26]. In boosting, there are a number of algorithms to handle multi-class boosting, and SAMME [8] is one of the most widely used.

6 Conclusion

We use price multiplier prediction to relieve the anxiety brought by dynamic pricing in RoD services, and help passengers to answer 'Could I get a lower price multiplier within a short time/distance?'.

Based on the data collected from a major service provider in China, we define a metric *maximum predictability* $\Pi^{\max}$ to characterise the predictability of price multipliers in different locations. Data analysis shows that the mean $\Pi^{\max}$ around the city is 0.8315 (i.e. at most 83.15% price multipliers could be predicted) and that considering temporal correlation is necessary for prediction and could increase $\Pi^{\max}$ by up to 75.61% in the business area.

We then implement two baseline predictors, the Markov-chain and NN predictor, and use ensemble learning to build a boosted-NN predictor to further improve the prediction accuracy. Our evaluation shows that the Markov-chain predictor is suitable in highly-predictable locations, whereas the NN predictor performs better in less predictable locations. Also, the boosted-NN predictor could further reduce prediction inaccuracy by up to 21% for more unpredictable locations. Finally, we evaluate the effect of prediction, and results show that our predictors could find out lower price multipliers nearby with a probability as high as 89.4% when there indeed exists lower prices in neighbouring cells. Also, the probability that a passenger in the business area could use price prediction to get lower prices nearby is 35.4% and this rises to 67.3% if considering only edge cells.

7 Acknowledgments

The work was supported by the National Key Research and Development Project of China (no. 2017YFB1002000), the National Science Foundation of China (no. 61602067), the Fundamental Research Funds for the Central Universities (no. 106112017cdjxy180001), the Chongqing Basic and Frontier Research Program (no. cstc2015jcyjA00016), the Open Research Fund Program of Shenzhen Key Laboratory of Spatial Smart Sensing and Services (Shenzhen University).

8 References

[1] Chen, L., Mislove, A., Wilson, C.: 'Peeking beneath the hood of Uber'. Proc. 2015 ACM Conf. on Internet Measurement Conf. (IMC'15), Tokyo, Japan, 2015, pp. 495–508

[2] Guo, S., Liu, Y., Xu, K., *et al.*: 'Understanding ride-on-demand service: demand and dynamic pricing'. 2017 IEEE Int. Conf. on Pervasive Computing and Communication Workshops (PerCom Workshops), Hawaii, HI, USA, 2017, pp. 509–514

[3] Guo, S., Liu, Y., Xu, K., *et al.*: 'Understanding passenger reaction to dynamic prices in ride-on-demand service'. 2017 IEEE Int. Conf. on Pervasive Computing and Communication Workshops (PerCom Workshops), Hawaii, HI, USA, 2017, pp. 42–45

[4] Shenzhou U.Car: 'Annual results for the year ended 31 Dec 2015'. 2015. Available at http://bit.ly/2cFdL6U

[5] Song, C., Qu, Z., Blumm, N., *et al.*: 'Limits of predictability in human mobility', *Science*, 2010, **327**, (5968), pp. 1018–1021

[6] Cover Thomas, M., Thomas Joy, A.: '*Elements of information theory*' (Wiley-Interscience, New York, NY, USA, 2006, 2nd edn.)

[7] Freund, Y., Schapire, R.E.: 'A decision-theoretic generalization of on-line learning and an application to boosting'. Second European Conf. on Computational Learning Theory (EuroCOLT'95), 1995, pp. 23–37
[8] Zhu, J., Zou, H., Rosset, S., *et al.*: 'Multi-class AdaBoost', *Stat. Interface*, 2009, **2**, (3), pp. 349–360
[9] Wikipedia: 'Symmetric mean absolute percentage error', 2017. Available at http://bit.ly/2umuNKT
[10] Hall, J., Kendrick, C., Nosko, C.: 'The effects of Uber's surge pricing: a case study', 2015. Available at http://bit.ly/2kayk9O
[11] Chen, M.K.: 'Dynamic pricing in a labor market: surge pricing and flexible work on the Uber platform'. Proc. 2016 ACM Conf. on Economics and Computation (EC'16), Maastricht, Netherlands, 2016, pp. 455–455
[12] Cohen, P., Hahn, R., Hall, J., *et al.*: '*Using big data to estimate consumer surplus: the case of Uber*' (National Bureau of Economic Research, Cambridge, UK, 2016)
[13] Li, X., Pan, G., Wu, Z., *et al.*: 'Prediction of urban human mobility using large-scale taxi traces and its applications', *Front. Comput. Sci.*, 2012, **6**, (1), pp. 111–121
[14] Li, B., Zhang, D., Chen, C., *et al.*: 'Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset'. 2011 IEEE Int. Conf. on Pervasive Computing and Communication Workshops (PerComWorkshops), Seattle, WA, USA, 2011, pp. 63–68
[15] Chen, C., Zhang, D., Castro, P.S., *et al.*: 'Real-time detection of anomalous taxi trajectories from GPS traces'. 8th Int. Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous 2011), Copenhagen, Denmark, 2011, pp. 63–74
[16] Chen, C., Zhang, D., Ma, X., *et al.*: 'CrowdDeliver: planning city-wide package delivery paths leveraging the crowds of taxis', *IEEE Trans. Intell. Transp. Syst.*, 2016, **18**, (6), pp. 1478–1496
[17] Wang, L., Zhang, D., Wang, Y., *et al.*: 'Sparse mobile crowd sensing: challenges and opportunities', *IEEE Commun. Mag.*, 2016, **54**, (7), pp. 161–167
[18] Chen, C., Jiao, S., Zhang, S., *et al.*: 'TripImputor: real-time imputing taxi trip purpose leveraging multi-sourced urban data', *IEEE Trans. Intell. Transp. Syst.*, 2018, **PP**, (99), pp. 1–13
[19] Guo, B., Han, Q., Chen, H., *et al.*: 'The emergence of visual crowdsensing: challenges and opportunities', *IEEE Commun. Surv. Tutor.*, 2017, **19**, (4), pp. 2526–2543
[20] Guo, B., Chen, H., Han, Q., *et al.*: 'Worker-contributed data utility measurement for visual crowdsensing systems', *IEEE Trans. Mob. Comput.*, 2017, **16**, (8), pp. 2379–2391
[21] Xia, F., Ahmed, A.M., Yang, L.T., *et al.*: 'Exploiting social relationship to enable efficient replica allocation in ad-hoc social networks', *IEEE Trans. Parallel Distrib. Syst.*, 2014, **25**, (12), pp. 3167–3176
[22] Xia, F., Ahmed, A.M., Yang, L.T., *et al.*: 'Community-based event dissemination with optimal load balancing', *IEEE Trans. Comput.*, 2015, **64**, (7), pp. 1857–1869
[23] Gonzalez, M.C., Hidalgo, C.A., Barabasi, A.L.: 'Understanding individual human mobility patterns', *Nature*, 2008, **453**, (7196), pp. 779–782
[24] Zhao, K., Khryashchev, D., Freire, J., *et al.*: 'Predicting taxi demand at high spatial resolution: approaching the limit of predictability'. 2016 IEEE Int. Conf. on Big Data, Washington, DC, USA, 2016, pp. 833–842
[25] Breiman, L.: 'Bagging predictors', *Mach. Learn.*, 1996, **24**, (2), pp. 123–140
[26] Wolpert, D.H.: 'Stacked generalization', *Neural Netw.*, 1992, **5**, pp. 241–259