Customized Regression Model for Airbnb Dynamic Pricing

Peng Ye*
Airbnb Inc.
San Francisco, CA
peng.ye@airbnb.com

Julian Qian*
Ant Financial
San Mateo, CA
j.qian@antfin.com

Jieying Chen
Airbnb Inc.
San Francisco, CA
jieying.chen@airbnb.com

Chen-hung Wu
Airbnb Inc.
San Francisco, CA
chen-hung.wu@airbnb.com

Yitong Zhou
Airbnb Inc.
San Francisco, CA
yitong.zhou@airbnb.com

Spencer De Mars
Impira Inc.
San Francisco, CA
spencer@impira.com

Frank Yang
Airbnb Inc.
San Francisco, CA
frank.yang@airbnb.com

Li Zhang
Airbnb Inc.
San Francisco, CA
li.zhang@airbnb.com

ABSTRACT

This paper describes the pricing strategy model deployed at Airbnb, an online marketplace for sharing home and experience. The goal of price optimization is to help hosts who share their homes on Airbnb set the optimal price for their listings. In contrast to conventional pricing problems, where pricing strategies are applied to a large quantity of identical products, there are no "identical" products on Airbnb, because each listing on our platform offers unique values and experiences to our guests. The unique nature of Airbnb listings makes it very difficult to estimate an accurate demand curve that's required to apply conventional revenue maximization pricing strategies.

Our pricing system consists of three components. First, a binary classification model predicts the booking probability of each listing-night. Second, a regression model predicts the optimal price for each listing-night, in which a customized loss function is used to guide the learning. Finally, we apply additional personalization logic on top of the output from the second model to generate the final price suggestions. In this paper, we focus on describing the regression model in the second stage of our pricing system. We also describe a novel set of metrics for offline evaluation. The proposed pricing strategy has been deployed in production to power the Price Tips and Smart Pricing tool on Airbnb. Online A/B testing results demonstrate the effectiveness of the proposed strategy model.

*These two authors contributed equally. This work was done while the author was a member of the Pricing Modeling team at Airbnb.

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219830

KEYWORDS

Price Optimization, Customized Regression Model, Dynamic Pricing

ACM Reference Format:
Peng Ye, Julian Qian, Jieying Chen, Chen-hung Wu, Yitong Zhou, Spencer De Mars, Frank Yang, and Li Zhang. 2018. Customized Regression Model for Airbnb Dynamic Pricing. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219830

1 INTRODUCTION

Airbnb is an online marketplace for sharing homes and experiences, where guests who seek accommodation are matched to hosts who have spare rooms to share. In this two-sided marketplace, pricing is an important lever to better match supply and demand. As a platform provider, Airbnb does not control how our hosts set prices for their listings, but we do provide various tools to help our hosts set their prices more effectively. For example, we allow hosts to set customized daily prices, weekend prices, discounts for long term stays, etc. We also provide price suggestions to our hosts. Our price suggestions are presented to our hosts in two different ways. First, we have "Price Tips" (shown in Fig. 1), where we color code current prices on the calendar to inform hosts how likely the night is to be booked, and once the host selects a particular night, we show them our price tip in the price setting panel on the right side of the calendar. Additionally, explanations about our tips are also provided. We should note that "Price Tips" obliges hosts to review and, if they are agreeable, adopt our suggestions every day, in order to keep up with the latest tips. To make price tips adoption easier for our hosts, we introduced "Smart Pricing". With "Smart Pricing", hosts can set a min price and a max price between which they would like their prices to fall. Then new price suggestions that are between the min and max prices are automatically adopted for all available nights. Fig. 2 shows the calendar of a listing with "Smart Pricing".

Due to uncertainty about demand in each time period, optimal prices often vary over time. This is a typical dynamic pricing problem, where Airbnb hosts offer "nights" to be sold over a fixed time horizon. Our price suggestions are generated by a machine learning algorithm and are updated every day according to our best understanding about the current status of market dynamics. The objective of this model is to determine a dynamic pricing strategy that helps hosts set the optimal price for the entire selling period. In this paper, we describe the algorithms that underpin the dynamic pricing system used in "Price Tips" and "Smart Pricing".

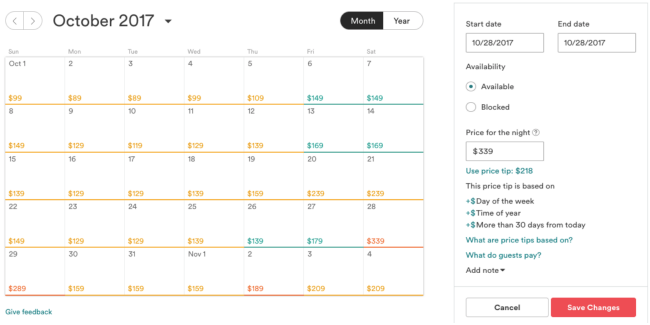

Figure 1: A snapshot of Price Tips.

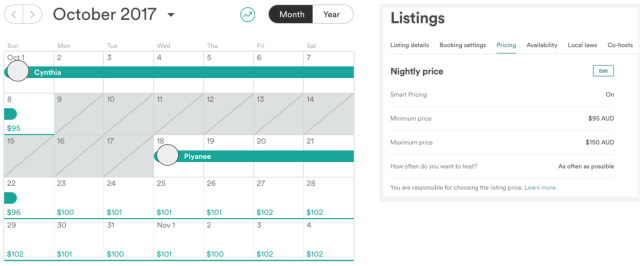

Figure 2: A snapshot of Smart Pricing.

1.1 Related Works

The growth of Airbnb has attracted substantial academic attention. Most research has focused on Airbnb's competitiveness to its counterpart, hotel industries, and its challenges. For example, Zervas *et al.* [18] studied the impact of Airbnb in Texas area on the local hotel industry, quantified the causal impact on prices to be −8% on average and described the non-uniformity of the impact on various hotel ranges, where low-priced hotels were among the most affected. Quattrone *et al.* [13] studied Airbnb listings in London from 2012 to 2015 and proposed an algorithmic policy-making procedure, in which the regulations could be made more responsive to the evolving nature of sharing economies. Ikkala *et al.* studied how monetary reward could incentivize network hospitality in Airbnb. Authors in [8] applied the geographically weighted regression (GWR) approach to identify some factors that are correlated with Airbnb listing prices. Similar efforts have been seen in [9, 11, 16] to identify possible factors that correlate with Airbnb listing prices.

1.2 Dynamic Pricing at Airbnb

In this section, we discuss two unique challenges of the dynamic pricing problem at Airbnb.

Demand Estimation

Studies of dynamic pricing often focus on homogeneous products [6]. By tracking how demand varies with respect to price for a large number of identical products, a demand curve $F(P)$ can be estimated, which determines demand as a function of price P. Then the problem of revenue maximization is to find the price P that yields maximal $P \times F(P)$. The key to the success of this approach is to get an accurate estimation of the demand function $F(P)$. There has been an increasing interest in using machine learning approaches for demand estimation [2, 15]. In the Airbnb pricing problem, the demand function $F(P)$ is not just a function of price, as it also varies across time and listings. Let's denote the demand function as $F(P, t, id)$, where t is time and id is a unique id for a listing on Airbnb. Next, we explain why the demand can vary over time and listing.

Time-varying: There are two main factors that cause the demand function to be time-varying. (1) Seasonality and events: travel activities subject to quite strong seasonal variation. Fig. 3 shows the global Google search trend for "Airbnb" over the past five years. We can see that there is a strong seasonality pattern. During the summertime, more people search "Airbnb" with Google. This is perhaps because more people are doing travel planning during the summer. Of course, for different countries, the seasonality pattern will be different. If we zoom into a particular region, we may observe a stronger seasonality pattern. In addition to seasonality, special events also cause a surge in demand for certain nights in the region where the event is held. (2) Lead time: Let's denote the current date as *ds* and the night that we are interested in estimating demand for as *ds_night*. Then *lead time* is the distance between *ds* and *ds_night*. As lead time reduces, there are less opportunities for this night to be booked, which leads to the change in the demand function.

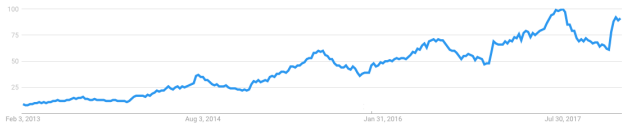

Figure 3: Google trend for Airbnb worldwide over 5 Years [Data source: Google Trends www.google.com/trends]

Listing-varying: Unlike hotels, where all the rooms of the same type are identical, Airbnb listings are all quite different. For example, in addition to standard housing properties, there are also castles, tree houses, boats, etc on Airbnb. The demand for a treehouse can be quite different from the demand for a standard single family house. Even if we only consider single family houses in the same neighborhood with the same person capacity, a listing with 100+ five star reviews will be more popular than a new listing. When demand is sufficiently high for the neighborhood, listings with more good reviews may be able to charge higher prices without

hurting the occupancy rate, while increasing price may hurt the booking probability of a new listing more.

The time-varying and listing-varying nature of demand makes estimating demand function $F(P, t, id)$ extremely hard. We will discuss more about the challenges in Section 2.

Partial Price Adoption

In conventional pricing settings, e.g. hotel, airline, retail pricing, the firm who owns the product has the full control of how to set prices, so that they can experiment and explore different pricing strategies and observe the market responses. The fact that our suggested prices are only "partially" adopted by our hosts introduces additional complexity when designing a feasible pricing strategy. Hosts who use "Price Tips" may select only some of our price suggestions and the adoption is often biased towards higher suggestions. For "Smart Pricing" users, the estimated "optimal price" may be out of the min-max bound set by the host. For example, some hosts may set a very high min-price and as a result, prices for most of their nights are min-bounded.

1.3 Our approach

Due to the difficulties in estimating an accurate demand curve $F(P, t, id)$, directly applying a revenue maximization strategy often fails to maximize revenue for our hosts in practice. To overcome this difficulty, we build a second model that maps $F(P, t, id)$ to price suggestions. We call this model the pricing strategy model. It is a regression model that is trained by minimizing a customized loss function. Previous work on using regression model for estimating market values often assume the market value of each product is linear in the value of product features [1, 4, 5], our model, on the other hand, is highly non-linear. First, we model the demand function using a Gradient Boosting Machine (GBM) [3, 7], which is a non-linear mapping from a large set of raw features to an estimated booking probability. Next, the booking probability is mapped to the suggestion via another non-linear function which we will discuss in more detail in Sec. 4. The customized loss function for training the pricing strategy model is inspired by the ϵ-insensitive loss function used in the Support Vector Regression (SVR) [14]. However, our loss function differs from the ϵ-insensitive loss in that we do not have an accurate target variable and instead of using a constant ϵ for all training samples, ϵ varies for different training samples and its value is determined by a price range into which we believe an optimal price should fall.

Our contributions are two-fold. First, we introduce a set of metrics that can be used to measure the effectiveness of a pricing strategy. Second, we propose a customized regression model to learn a pricing strategy that can help minimize "bad" suggestions defined by the proposed metrics.

The remainder of this paper is organized as follows. First, in Sec. 2, we give an overview of the pricing system, introduce the booking probability model and show how the pricing strategy model fits in the system. In Sec. 3, we propose a set of evaluation metrics. Then, we describe the pricing strategy model in details in Sec. 4. Finally, we present results from both offline and online experiments to show that the proposed strategy model helps to optimize revenue better compared to a direct revenue maximization strategy.

2 PRICING SYSTEM OVERVIEW

The pricing system (shown in Fig. 4) consists of three main components. First, a binary classification model predicts the booking probability of each listing-night. Second, a per-listing regression model predicts the "optimal" price for each listing-night, in which a customized loss function is used to guide the learning. Predicted booking probability from the first model is used as an input feature for the regression model. Finally, we apply additional personalization logic incorporating hosting goals, special events, etc on top of the output from the second model to generate the final price suggestions.

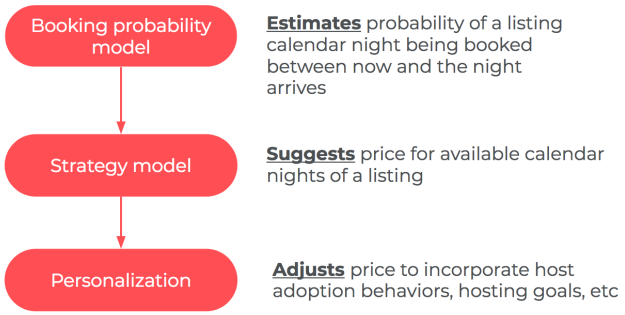

Figure 4: Overview of the pricing system.

2.1 The Booking Probability Model

In order to estimate the demand curve for each-listing night, we build a booking probability model. The objective of learning is to predict whether an available future night of a specific listing will be booked as of the day we make this prediction, which is a standard binary classification problem. Here are examples of features that we used in this model.

- Listing Features: listing price per night, room type, person capacity, the number of bedrooms/bathrooms, amenities, locations, reviews, historical occupancy rate, instant booking enabled, etc.
- Temporal Features: seasonality (the day of the year, day of the week, etc), the calendar availability (e.g. the gap between check in and check out), distance between ds and ds_night, etc.
- Supply and demand dynamics: number of available listings in the neighborhood, listing views, searches/contacts rates, etc.

We used a Gradient Boosting Machine (GBM) [3, 7] to predict the booking probability. Instead of training a global GBM model with a constant sampling rate of training data, we found that a better performance, in terms of global AUC, can be achieved by training a separate GBM model for each market with an adaptive training data sampling rate as illustrated in Figure 5. Markets with high density of listings benefit from the location-based models the most, which we sample at a rate higher than the global constant sampling rate.

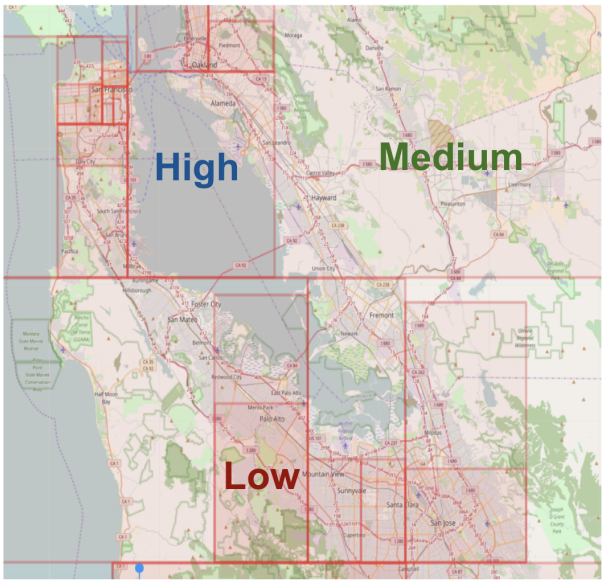

Figure 5: Example of adaptive sampling.

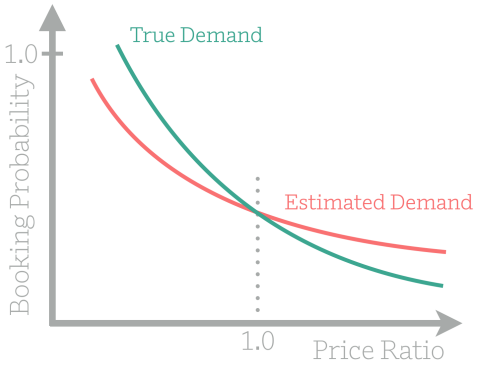

Figure 6: Example of estimated and real demand curve.

The predicted booking probility as a function of price gives us an estimated demand curve. Fig. 6 shows an example of the estimated demand curve. It is derived by scoring the booking probability model at different price points in a range. The x-axis is Fig. 6 represents a normalized price range, where the price ratio 1.0 indicates the listing price. The estimated demand curve may not be accurate, the green curve in Fig. 6 represents the underlying true demand curve. When the price ratio goes to 0.0, i.e. the listing-night is free, we expect the booking probabilities from both curves approach 1.0. On the other hand, when the price ratio approaches ∞, we would expect the booking probabilities to approach 0.0.

The challenges of deriving an accurate elasticity estimation are of three kinds:

- Data sparseness: Most listings on Airbnb do not vary their nightly prices dramatically. For example, a private room at San Francisco that's normally priced at $150 may never be priced lower than $50, or higher than $500. As a result, we don't have observations of the price points that are far away from the listing "base" price, which makes price extrapolation very difficult.
- Sample uniqueness: The uniqueness of listings makes it hard to generalize the learning from one listing to others.
- Feature dependency: Some raw features are price dependent, for example, the searches or listing views are usually negatively correlated with the price feature. Another example is the occupancy rate feature, some hosts may set their listing prices too high so that their occupancy rate is very low. It may require additional models to learn how they are correlated. Therefore, simply scoring the model by varying price features alone cannot derive the true demand curve.

Due to these challenges, getting an accurate demand curve at listing-night level is extremely difficult. We have tried to directly apply revenue maximization strategies based on our estimated demand curve, but online A/B testing results showed that these methods often fail to optimize revenue for our hosts in practice. Therefore, we decide to pursue alternative solutions and rethink about what is *optimal* price point and what would be the right offline evaluation metrics. In the next section, we introduce a set of metrics for offline evaluation.

3 EVALUATING PRICE SUGGESTION

Unlike conventional supervised learning problems, where a label is often available for each training sample, we don't really have a ground-truth of "optimal" price. This makes evaluating price suggestions a non-trivial problem. In this section, we describe a set of evaluation metrics that we use to characterize the quality of our price suggestions.

3.1 Motivation

The proposed metrics are derived from our intuitions about where the "optimal" price should fall and what would be a bad price suggestion. Let's denote the actual price of a listing-night listed on airbnb P, our suggested price as P_{sug} and the "optimal" price (if exists) as P_o. We regard P_{sug} as bad in the following two cases:

- The listing-night was booked and $P_{\text{sug}} < P$: in this case, the listing-night was booked at P, but we are suggesting a lower price. This means if the host had adopted our suggestion, there would be $P - P_{\text{sug}}$ loss in revenue. Therefore, we should have $P_o >= P$.
- The listing-night was not booked and $P_{\text{sug}} >= P$: in this case, the listing-night was not booked at P and we are suggesting higher prices, the probability of booking would be even smaller. Therefore we regard P_{sug} as a bad suggestion and we should have $P_o < P$.

There are also cases where we cannot make a conclusion about whether the suggestion is good or bad.

- The listing-night was booked and $P_{\text{sug}} >= P$: in this case, our suggestion is higher than the booking price. We don't know if the host had adopted our suggestion, the listing-night would still be booked or not.

	Booking	Non-Booking
$P_{\text{sug}} >= P$	a	b
$P_{\text{sug}} < P$	c	d

Table 1: Number of samples

- The listing-night was not booked and $P_{\text{sug}} < P$: in this case, if the host had priced lower, the probability of booking for the listing-night would be higher. But if P_{sug} is too low, e.g. lower than the inherent cost of hosting, it may still be bad for the host even if the night can be booked at P_{sug}.

3.2 Metric Definition

Assuming the number of suggestions in each of the four cases are defined in Table 1, we define a set of metrics as follows:

- **Price Decrease Recall (PDR)**: among all non-booked nights, the percentage of suggestions that are lower than calendar prices.

$$PDR = \frac{d}{b+d} \tag{1}$$

  We illustrate the meaning of PDR in Fig. 7. In this case, 3 of 5 of unbooked nights have suggestions lower than the actual prices. By definition, we have $PDR = 0.6$.
- **Price Decrease Precision (PDP)**: among all cases where $P_{\text{sug}} < P$, the percentage of nights that are non-booked.

$$PDP = \frac{d}{c+d} \tag{2}$$

- **Price Increase Recall (PIR)**: among all booked nights, the percentage of suggestions that are higher than or equal to calendar prices.

$$PIR = \frac{a}{a+c} \tag{3}$$

- **Price Increase Precision (PIP)**: among all cases where $P_{\text{sug}} >= P$, the percentage of nights that are booked.

$$PIP = \frac{a}{a+b} \tag{4}$$

- **Booking Regret (BR)**:

$$BR = median_{bookings}(max(0, \frac{P - P_{\text{sug}}}{P})) \tag{5}$$

  Fig. 8 illustrate our BR metric. In this case, on two occasions our suggestion was at or above the booked price. On two occasions suggested prices were below booked prices, by 14%($15), 5%($10), and 6%($5). Therefore, BR = Median(14, 5, 6, 0, 0) = 5%.

A good set of offline evaluation metrics should be tied to the online business metrics that we are trying to drive. In other words, the improvement on our offline evaluation metrics should yield a lift in our business metrics with high likelihood, so that they can be used to guide our offline strategy hyper-parameter tuning and model development. Among the proposed metrics, we found that PDR and BR are well correlated with the online business metrics. In particular, PDR is correlated with the bookings gain. Intuitively, PDR measures how likely our suggested prices are to be lower than the current listing prices for a non-booked listing-night. Our suggested

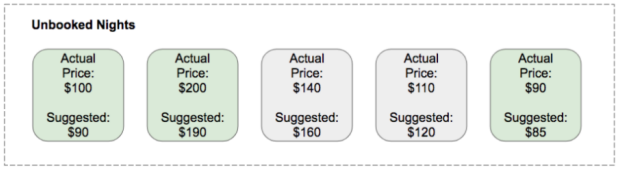

Figure 7: Example of Price Decrease Recall(PDR).

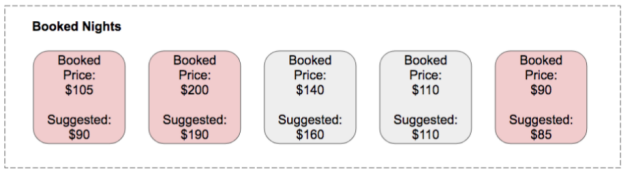

Figure 8: Example of Booking Regret(BR).

prices, in this case, might help improve the competitiveness of those unsuccessful listing-nights. On the other hand, BR measures how close our suggested prices are to those booked prices, where the booked prices, in general, indicate the prices were set successfully and competitive in the markets. Pricing strategies with lower BR usually help to earn more trust from our hosts. The pricing strategy should try to jointly optimize PDR and BR. However, in practice, there is almost always trade-off among these metrics. Usually, a strategy that suggests lower prices can often improve PDR but hurt BR.

4 STRATEGY MODEL

4.1 Objective Function

The objective function of the strategy model training was inspired by the intuition about where the "optimal" price should lie as described in Sec. 3.1. Given N training samples $\{\boldsymbol{x}_i, y_i\}_{i=1}^{N}$, where $\boldsymbol{x}_i$ represents input features associated with the listing-night; y_i indicates booking status: for booked sample $y_i = 1$, and $y_i = 0$ for non-booked sample.

The list of features $\boldsymbol{x}_i$ consists of:

(1) The calendar price P_i set by the host.
(2) The guest booking probability of the listing-night q_i, which is the output of the booking probability model described in Sec. 2.1.
(3) Market demand signals: there might be other demand related signals that are not fully captured by the booking probability model, which we incorporate in the strategy layer. This allows us to quickly respond to market demand.

The suggested price for $\boldsymbol{x}_i$ is denoted as $f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$, where $\boldsymbol{\theta}$ is a set of parameters that need to be learned for the mapping function f. Details about $f_{\boldsymbol{\theta}}$ will be discussed in Sec. 4.2. Our proposed loss function is inspired by the ϵ-insensitive loss used in SVR. Fig. 9 shows a comparison between ϵ-insensitive loss and our customized loss function. In the ϵ-insensitive loss, a golden label is required. In our case, we do not know exactly what the "optimal" price is, but

we can derive a price range from business insight into which we believe the "optimal" price may fall.

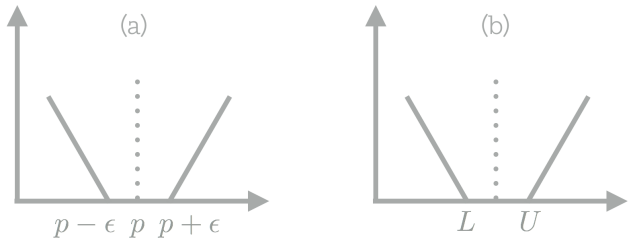

Figure 9: (a) ϵ-insensitive loss. (b) The proposed loss function.

The objective of learning is to minimize the loss $\mathcal{L}$:

$$\mathcal{L} = \arg\min_{\theta} \sum_{i=1}^{N} (L(P_i, y_i) - f_{\theta}(\boldsymbol{x}_i))^+ + (f_{\theta}(\boldsymbol{x}_i) - U(P_i, y_i))^+ \quad (6)$$

where the superscripted "+"' is taken by $\max(0, \cdot)$. We use $L(P_i, y_i)$ and $U(P_i, y_i)$ for the lower bound and the upper bound of the optimal price range, respectively. If suggestions fall between the upper bound and the lower bound, the loss is 0; otherwise, the loss is the distance between the suggestion and the bound. In particular, L and U are defined as follows

$$L(P_i, y_i) = y_i \cdot P_i + (1 - y_i) \cdot c_1 P_i \quad (7)$$

For booked listing-nights, the lower bound $L(P_i, y_i)$ is the booking price P_i, while for the non-booked listing nights, the lower bound is $c_1 P_i$, where c_1 is a constant $\in (0, 1)$.

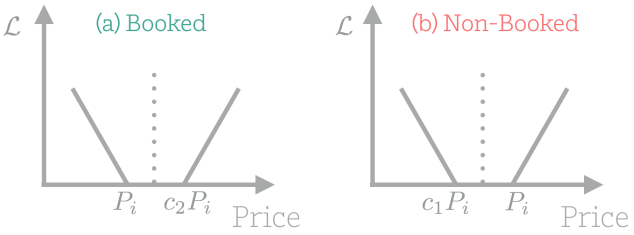

Figure 10: Loss function (a) positive sample. (b) negative sample.

$$U(P_i, y_i) = (1 - y_i) \cdot P_i + y_i \cdot c_2 P_i \quad (8)$$

For non-booked listing-nights, the upper bound $U(P_i, y_i)$ is the calendar price P_i at which the sample was not booked. For booked listing-night, the upper bound is $c_2 P_i$, where $c_2 > 1$ is a constant. The constants c_1 and c_2 are parameters learned via hyper-parameter searches to optimize for the best outcome for hosts and marketplace.

Fig. 10 illustrates the loss function for positive and negative samples. It is important to note that the design of the strategy model objective function is directly tied to offline evaluation metrics described in Sec. 3. For a training sample of a booked listing-night, when the suggested price falls in the range of $(P_i, c_2 P_i)$, the loss $\mathcal{L}$ is reduced to 0, which also leads to booking regret metric reaching a minimum at 0. For a training sample consisting of a non-booked listing-night, when the price suggestion falls in the range of $(c_i P_i, P_i)$, the loss $\mathcal{L}$ is reduced to 0, which also leads to a higher PDR because the suggested price is lower than the calendar price P_i.

In particular, if $c_1 = c_2 = 1$ in Eq. (7)-(8), it gives $L = U$ and the corresponding "optimal price"' will be exactly the calendar price P_i. Therefore, c_1 and c_2 are chosen such that $U - L$ is wider than commonly used ϵ for regression.

4.2 Demand-Enhanced Pricing

In the previous section, we have discussed the training objective of the strategy model. This section describes a specific functional form, mapping input features to price suggestions. There are several assumptions behind this model:

(1) For the same listing, suggested price is positively correlated with the booking probability at current price. This is to ensure that our price suggestions are responsive to changes in booking probability.
(2) Price suggestions are centered around the most representative price that is often set by the host, with learnable increasing/decreasing magnitudes.
(3) Additional demand signals that are not fully captured by the booking probability model should be easily plugged in.

Based on above assumptions, we introduce an asymmetric exponent-exponential form model, which applies price increases/decreases upon the calendar price with magnitude learned from data. The price suggestion P_{sug} is given by

$$P_{\text{sug}} = P \cdot V, \quad (9)$$

where the increase/decrease magnitude V is

$$V = \begin{cases} 1 + \theta_1 (q^{\varphi_H^{-qD}} - \theta_2) & \text{if } D > 0, \\ 1 + \theta_1 (q^{\varphi_L^{-(1-q)D}} - \theta_2) & \text{if } D \le 0; \end{cases} \quad (10)$$

P is the representative calendar price[1] set by the host, q is the booking probability at P estimated by the model described in Sec. 2, and D is a demand score derived from additional demand signals at the cluster level. A cluster is a group of similar listings. For example, we can group listings in the same market to one cluster.

In the above equation, θ_1 controls the ratio of maximal price increasing/decreasing magnitude and θ_2 tweaks when we suggest the original calendar price (i.e. $P_{\text{sug}} = P$). Given θ_1 and θ_2, P_{sug} is monotonically increasing with booking probability q, which ensures that we suggest higher prices for listing nights with higher booking probabilities.

The demand score D is normalized by adjusting cluster level demand signals onto a common Gaussian scale. The higher the value of D, the higher the demand in the corresponding cluster. The demand constants $1 < \varphi_L < \varphi_H < 2$ are chosen to control the extent to which the suggestion curves bend. We do not use the same constants for price increases and decreases since we would like the training system to learn the ratios asymmetrically. In this way, price

[1]There are different ways to determining the representative prices, for example, it can be the median booked price for a listing with booking history

suggestions can reflect the demand sensitivity more thoroughly by taking advantage of the non-linear manner in which markets perceive demand and supply. A simple illustration is shown in Fig. 11.

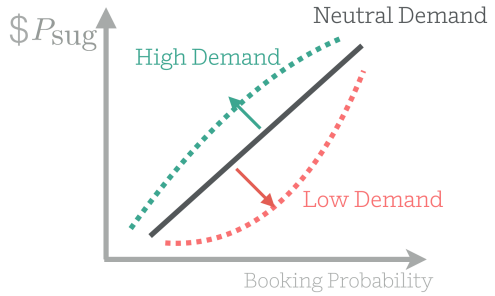

Figure 11: Demand-enhanced pricing strategy illustration.

In (10), θ_1 and θ_2 are parameters trained at the listing level. This formulation provides a good demand-reflective region for the parameters learning. The demand-enhanced strategy gives a straightforward gradient calculation:

$$\frac{\partial V}{\partial \theta_1} = p^{\varphi_H^{-pD}} - \theta_2, \tag{11}$$

$$\frac{\partial V}{\partial \theta_2} = -\theta_1, \tag{12}$$

which yields P_{sug} to this end.

4.3 Training

4.3.1 Listing-Level Training. Airbnb listing prices can be significantly different from each other – even listings that belong to the same type and market. And each listing's optimal strategy is also heavily subject to its own properties and hosting preferences. Therefore unlike the booking probability model which is trained on listing clusters, the strategy model is trained for each individual of the total 4 million+ active listings on Airbnb. We still prepare market-level and global-level fallback parameters in case a listing has insufficient training samples, e.g. new listings with zero bookings or listings with most of the calendar nights blocked.

4.3.2 Parameter Constraints. We introduce several parameter constraints to guarantee pricing results land in heuristically reasonable regions:

$$l_1 <= \theta_1 <= \mu_1 \tag{13}$$

$$l_2 <= \theta_2 <= \mu_2 \tag{14}$$

where (l_1, l_2, μ_1, μ_2) serve as box constraints to limit parameter ranges. For example, in real-life experiments, we often observe an overly high θ_1 leads to unstable suggestions or overly high θ_2 leads to too few price increases. Neither scenario is desired by most hosts.

Optionally for some low-demand listings, we may require that:

$$a\theta_1 + b <= \theta_2 (a > 0) \tag{15}$$

which requires θ_2 to be proportionally large when θ_1 is large, throttling the frequency with which high price suggestions are generated. These constraints are mostly treated as fail-safes and corner-case handlers to avoid unrealistic price suggestions when issues arise during training. Hyper parameters are chosen on a global level or on a few important listing cohorts to avoid over-tuning.

4.3.3 Stochastic Gradient Descent. Spark [17] is used to parallelize the training. We train and update the models on a daily basis. We use Stochastic Gradient Descent (SGD) [10] to optimize the loss function. Mini-batches and a fast-decaying learning rate are applied to regularize the model fitting. The constraints we adopt can be trivially solved via a Euclidean distance projection onto the constrained space as demonstrated in proximal algorithms [12]. This can be combined simply and effectively with SGD by carrying out an unconstrained descent step and then projecting the updated parameters back to the constrained space.

4.3.4 Training Data. Airbnb listings often experience obvious demand changes across seasons. The training strategy is therefore designed to focus more on the latest booking behaviors to better reflect recent seasonal signals. While the booking probability model is trained over the entire past year's booking, the strategy model leverages all future observed booking samples plus past a few weeks' bookings. Hyper parameter search and evaluation are carried out using a pseudo forward-looking dataset where all forward-looking nights have a known booking status but we mask out the "future"' information to see if a certain learning setting can yield to ideal results.

5 EXPERIMENTS

5.1 Offline Evaluation

5.1.1 Data set. To demonstrate the effectiveness of the proposed strategy, we have run extensive offline evaluations. Assuming *ds* is the date on which we made the price suggestions and *ds_eval* is the date on which we collected the booking label, i.e. we consider listing-night which got at least one booking request between *ds* and *ds_eval* as positive samples and listing-night that were available between *ds* and *ds_eval*, and didn't get any booking requests and which *ds_night* <= *ds_eval* as negative samples. Note that for non-booked nights in the future (i.e. *ds_night* > *ds*), we don't include them in the evaluation since these nights may be booked in the future. Fig. 12 shows examples of the three types of evaluation samples.

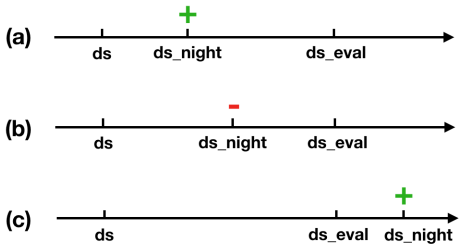

Figure 12: Examples of training data (a) Realized night, positive sample. (b) Realized night, negative sample. (c) Unrealized night, positive sample.

dataset	PDR	BR
(a)	+15.6%	5.74%
(b)	+13.43%	-6.38%
(c)	+11.85%	-7.5%

Table 2: Percentage gains of the strategy model compared to a naive pricing strategy in offline evaluation.

We randomly sample about 10% of all listing-nights to form our evaluation set. We choose the following *ds* and *ds_eval* for evaluation.

(a) *ds=2018-01-15, ds_eval=2018-02-04*: the selected *ds* is in mid-January which is high booking season for Airbnb. The distance between *ds* and *ds_eval* is about three weeks.

(b) *ds=2017-10-20, ds_eval=2017-11-11*: in this set, we consider *ds* in low booking season and the evaluation lead time is about three weeks.

(c) *ds=2017-10-20, ds_eval=2018-01-20*: in this set, we consider a longer *ds* to *ds_eval* window.

5.1.2 Comparison with a Naive Pricing Strategy. In this section, we compare the performance of the proposed strategy model with a naive pricing strategy that suggests prices that maximize the expected revenue based on the demand curve generated by the booking probability model introduced in Sec. 2.1. Table 2 shows the relative performance gain of the strategy model. We can see that the strategy model in general significantly improves both PDR and BR metrics except for BR of dataset (a). The naive pricing strategy likely suffers because it purely depends on the accuracy of demand curves which are hard to estimate due to challenges pointed out in Sec. 2.1, while as our approach leverages the latest booking information on the platform to learn a demand-enhanced price that tries to avoid "bad" suggestions.

5.2 Online Evaluation

The proposed strategy model has been deployed in production for more than 1 year at Airbnb. The launch of the first iteration of the strategy model yielded significant gains on bookings and booking values for hosts who have adopted our suggestions. The proposed set of offline metrics were also greatly improved over the previous production model. Multiple iterations of the strategy model have been experimented and launched into the production to futher improve the quality of our price suggestions.

5.3 Spot Checking Price Suggestions

In addition to the quantitive evaluation, we also did qualitative evaluation to check if our price suggestions are responsive to demand change. In particular, we sampled the final price suggestions generated on 2018-02-08 for future 120 nights. Fig. 13 and 14 show how the average price suggestions vary over time[2] for Tokyo and Tahoe. For both markets, we see that there are strong weekly pattern. Our model tends to suggest higher weekend prices. For the Tokyo market, there is a strong spike from late March to early April, which corresponds to the cherry blossom season. From these two examples, we see that our model can indeed capture the market dynamics in a timely fashion.

[2]Due to the sensitivity of the data, we did not include the absolute price suggestion ranges.

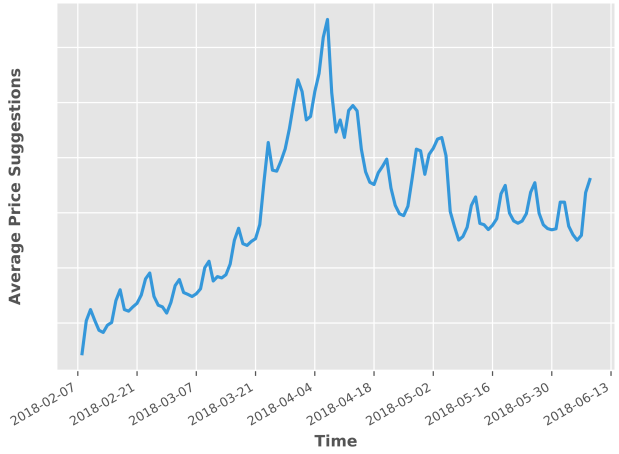

Figure 13: Tokyo

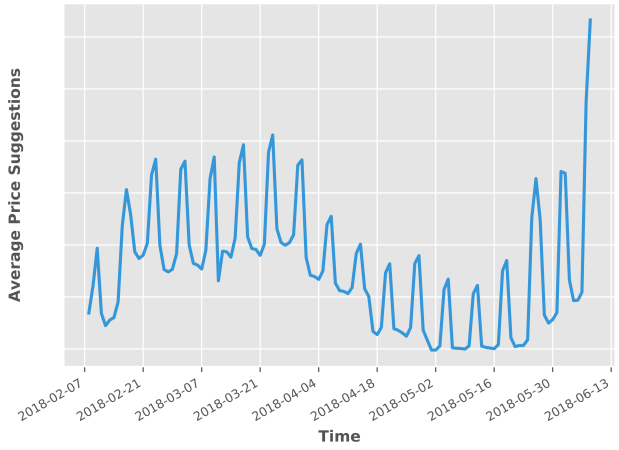

Figure 14: Tahoe, CA

6 CONCLUSION

In this paper, we described challenges of the dynamic pricing problem at Airbnb and presented a pricing strategy model that was deployed in production to help Airbnb hosts set their prices more effectively. Offline and online evaluation results show that the proposed strategy model performs significantly better than a direct max-rev pricing strategy. We are also actively working on improving the demand curve estimation. With more accurate demand curve, we may revisit the direct revenue maximization strategy in the future.

ACKNOWLEDGEMENT

Thanks Thanasis Noulas, Bar Ifrach and all members of the MDX modeling team at Airbnb for valuable suggestions and discussions.

REFERENCES

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. 2011. Improved Algorithms for Linear Stochastic Bandits. In *Proceedings of the 24th International Conference on Neural Information Processing Systems (NIPS'11)*. Curran Associates Inc., USA, 2312–2320.

[2] Patrick Bajari, Denis Nekipelov, Stephen P. Ryan, and Miaoyu Yang. 2015. Demand estimation with machine learning and model combination. *National Bureau of Economic Research* (2015).

[3] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794.

[4] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. 2011. Contextual Bandits with Linear Payoff Functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research)*, Geoffrey Gordon, David Dunson, and Miroslav DudÃŋk (Eds.), Vol. 15. PMLR, Fort Lauderdale, FL, USA, 208–214.

[5] Maxime Cohen, Ilan Lobel, and Renato Paes Leme. 2016. Feature-based Dynamic Pricing. In *Proceedings of the 2016 ACM Conference on Economics and Computation*. https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2737045

[6] Arnoud V. den Boer. 2015. Dynamic pricing and learning: Historical origins, current research, and new directions. *Surveys in Operations Research and Management Science* 20, 1 (2015), 1 – 18.

[7] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Ann. Statist.* 29, 5 (10 2001), 1189–1232. https://doi.org/10.1214/aos/1013203451

[8] hihua Zhang, Rachel J. C. Chen, Lee D. Han, and Lu Yang. 2017. Defining the price of hospitality: networked hospitality exchange via Airbnb. *sustainability* 9, 1635 (2017).

[9] A Ikkala, T.; Lampinen. 2014. Defining the price of hospitality: networked hospitality exchange via Airbnb. In *Proceedings of the Companion Publication of the 17th ACM Conference on Computer Supported Cooperative Work and Social Computing, (Proceedings of Machine Learning Research)*. Baltimore, MD, USA, 73âĂŞ–176.

[10] Bottou L. 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. In *COMPSTAT*. 122 – 186.

[11] Q.; Yang T.; Guo L Li, Y.; Pan. 2016. Reasonable price recommendation on Airbnb using Multi-Scale clustering. In *Proceedings of the 2016 35th Control Conference (CCC),*. Chengdu, China, 7038âĂŞ7041.

[12] Neal Parikh and Stephen Boyd. 2014. Proximal Algorithms. *Foundations and Trends in Optimization archive* (2014).

[13] Giovanni Quattrone, Davide Proserpio, Daniele Quercia, Licia Capra, and Mirco Musolesi. 2016. Who Benefits from the "Sharing" Economy of Airbnb? *(WWW '16)*.

[14] Alex J. Smola and Bernhard Schölkopf. 2004. A tutorial on support vector regression. *Statistics and Computing* 14, 1 (2004), 199 – 222.

[15] H. Varian. 2014. Big data: New tricks for econometrics. *Journal of Economic Perspectives* 28, 2 (2014), 3 – 28.

[16] J.L. Wang, D.; Nicolau. 2017. Price determinants of sharing economy based accommodation rental: A study oflistings from 33 cities on Airbnb.com. *Int. J. Hosp. Manag.* 62 (2017), 120–131.

[17] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. 2016. Apache Spark: A Unified Engine for Big Data Processing. *Commun. ACM* 59, 11 (Oct. 2016), 56–65.

[18] Georgios Zervas, Davide Proserpio, and John W. Byers. 2017. The Rise of the Sharing Economy: Estimating the Impact of Airbnb on the Hotel Industry. *Journal of Marketing Research* 54, 5 (2017), 687–705.