Estimating the Booking Curve

Nazanin Abedini[1], Rahul Dhopeshwar[2], Prashanth Lakshmi Narasimhan[2], Tyron Lardy[3], Khadija Meddouni[4], Muriel Pérez[2]

Abstract

Booking curves are a normalized measure of the proportion of the demand for seats in each flight as a function of the time before its departure. Their estimation is central to the operation and revenue maximization of commercial airlines. This case study addresses this problem as posed by Airfrance-KLM, a European airline with worldwide operations. Currently, Airfrance-KLM uses hard-coded rules based on expert knowledge to estimate the booking curves. These rules decide what the demand will be using features of each flight (such as its origin, destination, and many others) for estimation. This report addresses the challenge of implementing a data-driven approach for the task, as posed by the company members. It is shown that this problem is amenable to multivariate regression techniques, and the performance of several state-of-the-art methods is assessed on sample data. These methods include tree-based methods like random forests and light gradient boosting machines.

Keywords: KLM-Airfrance, booking curve, decision tree.

1 Introduction

Airfrance-KLM is an airline holding that operates flights to roughly 300 destinations, carrying over 70 million passengers yearly (KLM, 2024). Two passengers traveling next to each other may have paid different prices for their seats. This is because the price of a ticket is changed dynamically during the booking window; the time from the moment when seats become available for purchase to the time of departure of the flight. Flight and client-specific data are used to determine a pricing strategy with the goal of revenue maximization. These data – which serve as input features – include, among others, the time of the day, the season of the year, the length of the stay, whether it is a one-way or round-trip ticket, and the purpose of the travel.

Since clients are willing to pay different prices at different moments of the booking window, any dynamic pricing strategy will need accurate temporal forecasts of the demand for

[1]Vrije Universiteit Amsterdam
[2]Eindhoven University of Technology
[3]Leiden University
[4]Radboud University

seats in each plane. For example, business travelers typically make their bookings closer to departure time and are willing to pay more. This means two things: (1) Prices should be higher at the end of the booking window for this type of customer, and (2) prices earlier in the booking window need to be high enough so that not all seats are sold before this final stage. This is clearly a delicate task. As this example shows, these demand forecasts are expected to benefit from using flight and client-specific data, as mentioned above.

The analysts of Airfrance-KLM divide their problem of forecasting the demand for each flight into four separate estimation problems.

1. How many seats will be sold (total demand);
2. Customers' willingness to pay for a seat (price elasticity);
3. How many seats can be sold accounting for possible clients that do not show up to a flight (capacity);
4. When the seats will be sold (booking curve).

The final item, the estimation of the booking curve, is the subject of this study case.

The booking curve represents the course over time of the number of tickets that are sold relative to the total demand over the whole booking window. Since the company effectively intervenes in the demand of each flight with its dynamic pricing strategies, the booking curves are normalized so that they are comparable across flights. Booking curves are decreasing functions of the time before departure, and decrease from its value of 1 at the moment of departure, when 100% of its demand has been satisfied, to 0 at the moment when the booking window begins and no seats have been sold.

The fact that estimation is performed using past data tacitly assumes that past booking curves are informative for future ones. This assumption may be problematic as the shape of booking curves may be non-stationary over time. For example, the shape of booking curves may be affected during special events (for example, a big sports event), or years of altered demand (for example, during the 2020 world pandemic). In such cases, the flights are priced by hand by Airfrance-KLM employees. The non-stationarity of the booking curves is therefore not addressed in this report.

In this report, we address the problem of estimating booking curves as a function of time before departure using historical data. In particular, we show that this problem is amenable to standard multivariate regression methods. In Section 2, we describe the data that was made available by Airfrance-KLM. In Section 3, we describe the methods that we use to tackle the estimation problem. In particular, we show how the problem fits in the framework of multivariate regression and briefly describe the regression models that we fitted and the performance metrics that were considered. These methods include linear regression and tree-based methods like random forests and light gradient boosting machines (LightGBM). In Section 4, we show the results from fitting different models using a random train/test data split and optimizing several standard metrics of performance for regression. Interestingly, since the booking curves resemble cumulative probability distribution functions, distance metrics that can be applied to such functions can also be used to measure the distance between curves. One family of standard metrics between probability distributions is the Wasserstein distance (see Section 3 for more details).

2 Data

The data set made available by Airfrance-KLM consists of $n = 351820$ curves. Each curve describes the historical demand for a flight during thirty time intervals (called time frames) prior to the departure of the flight. Each curve is accompanied by a number of flight and client-specific features (see Table 1). Here, the demand index is a real number that has been normalized in such a way that demands for different flights and passenger types are comparable. From this demand index, the booking curve—our main object of interest—can be constructed by dividing the cumulative demand over time by the total demand over the 30 time intervals, as illustrated in Figure 1.

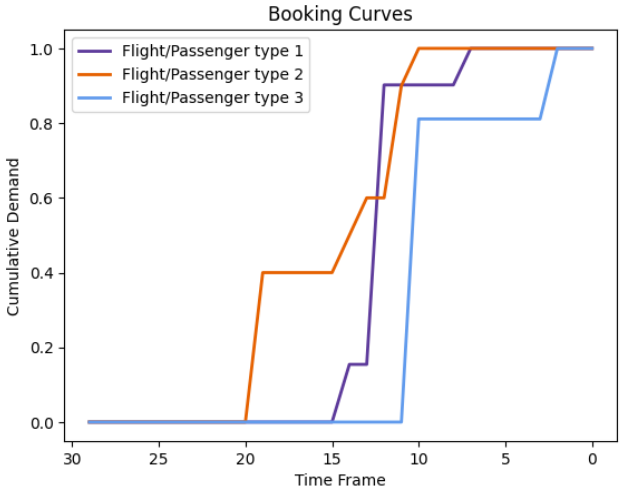

Figure 1: Booking curves for different flights and passenger types.

Feature	Description
Date	Date of the flight (DD-MM-YYYY)
Origin	Codifies the origin airport
Destination	Codifies the destination airport
Sell-up group	One of 18 client-specific features
Connection indicator	Whether the passenger is connecting outside the AFKL network.
Commencement point	Original starting country of the trip of the passenger

Table 1: Features of each demand curve. Each demand curve is given at 30 time intervals before flight departure (see Section 2).

In Table 1, client-specific features are included under the sell-up group features. This sell-up group contains 18 client-specific features including the cabin (first class, economy, etc.); a booking type (whether the seat was acquired with miles, with a regular payment, or other); a return type (whether the curve is for one way, round trip or all passengers); connection type;

type of haul (long or short haul); dominant airline (Airfrance of KLM); and passenger profile (business, leisure or special passengers).

3 Methods

This section describes the methods that were used to tackle the problem described in the previous sections of estimating booking curves. In Section 3.1, we describe the setting of multivariate regression and the method based on a training-test split of the data available to assess the performance of the predictors. In Section

3.1 Fitting multivariate regressors

We formalize the estimation of booking curves as a multivariate regression problem. We are given historical data $\{\vec{x}_i, \vec{y}_i\}_{i=1}^n$, where the $\vec{y}_i \in [0,1]^T$ ($T = 30$) are the booking curves and the $\vec{x}_i$ are k-dimensional vector of covariates (see Section 2 for a more detailed data description). Here, $T = 30$ denotes the number of time frames in which the booking window is separated. The restriction on $\vec{y}_i$ is then that at the earliest time frame ($i = 1$), the value is 0, and at the latest time frame ($i = 30$), it is 1. The space $\mathcal{X}$ where each feature vector $\vec{x}_i$ takes values signifies all possible combinations of features, and is the product of some finite sets of categories and some subsets of the real number line. In Section 3.2, we briefly discuss how we deal with categorical values in practice.

The goal is to use these data to construct an estimator $\hat{f} : \mathcal{X} \to [0,1]^T$. This estimator takes as inputs vectors of features (vectors in $\mathcal{X}$) and outputs booking curves (vectors in $[0,1]^T$). Given a vector of features $\vec{x}$, $\hat{f}(\vec{x})$ is our estimate of the booking curve corresponding to the flight and passenger group encoded in $\vec{x}$. As we will see in the next section, we consider predictors such as linear regressors and decision-based trees.

To evaluate the performance of predictors, we will consider several loss functions $L : [0,1]^T \times [0,1]^T \to \mathbb{R}$, where $L(\hat{f}(\vec{x}), \vec{y})$ is to be understood as the loss that is incurred when one predicts $\hat{f}(\vec{x})$, while the actual booking curve was represented by $\vec{y}$. A standard approach is to regard the available data as a random sample from an unknown distribution, and to look for the prediction method that minimizes the risk, or expected loss, i.e.,

$$\mathbb{E}_{(\vec{x},\vec{y})}[L(\hat{f}(\vec{x}), \vec{y})],$$

over all valid predictors—for example, all linear functions or all tree regressors—and the expectation is taken with respect to the underlying (unknown) distribution that generates the features/booking curves (Vapnik, 1991). Since this distribution of the data is generally unknown and too complicated to be estimated accurately, there is no way to compute exactly the loss in the previous display. Instead, we divide the data into a training and a test and use the test data to estimate the performance of predictors $\hat{f}_{\text{train}}$ trained on the training set. Thus, as a proxy for the expected risk in the previous display, we use empirical risk in the test set

$$\frac{1}{n_{\text{test}}} \sum_{i \in \text{test}} L(\hat{f}_{\text{train}}(\vec{x}_i), \vec{y}_i),$$

where the sum is over the data in the training set and n_{test} is the number of data points in the test set. After training and fitting a number of models $\hat{f}_{\text{train},1}, \ldots, \hat{f}_{\text{train},k}$ on the training set, we deem as best the model that minimizes the empirical risk over the test set.

In the following, we will consider the following loss functions:

1. The mean squared error given by $L_{\text{MSE}}(\vec{y'}, \vec{y}) = \frac{1}{T}\sum_{i=1}^{T}(y'_i - y_i)^2$ for $\vec{y}, \vec{y'} \in [0,1]$, which corresponds to the scaled and squared Euclidean distance.

2. The relative mean squared error given by $L_{\text{ARMSE}}(\vec{y'}, \vec{y}) = \frac{\sum_{i=1}^{T}(y'_i - y_i)}{\sum_{i=1}^{T}(y_i - \bar{y})}$, where $\bar{y} = \frac{1}{T}\sum_{i=1}^{T} y_i$.

3. The Wasserstein loss given by $L_{\text{WAS}}(\vec{y'}, \vec{y}) = W_2(\vec{y'}, \vec{y})$, where W_2 denotes the Wasserstein 2-distance (Villani, 2009).

3.2 Preprocessing

The specific date of each historical flight is not predictive of the booking curves of future flights, but the day of the week and month are expected to be. We extract these from the dates of the historical flights.

The implementation of many regression methods does not accept categorical variables. However, this can be dealt with in a number of standard ways. We convert the categorical variables using a One-Hot-Encoder. This type of encoding transforms each categorical feature with m possible values into m binary features, with one of them 1, and all others 0. For example, the categorical variable 'day of the week' is transformed to seven (one for each day of the week) zero-one-valued categorical variables.

We use smoothing on the booking curves. The data provided presents demand for 30 time frames leading up to flight departure, with each time frame representing a specific number of days. However, this data reflects ticket bookings rather than actual demand, resulting in observing zero demand across several time frames. If the booking curves are interpreted as observations of an underlying true positive 'demand process,' smoothing techniques can be used to estimate the demand in moments of zero demand using the demands of the adjacent time points. Various methods exist for smoothing time-series data, including moving averages, exponential smoothing, and kernel smoothing. The selection of an appropriate smoothing technique depends on understanding the characteristics of the data, and on the time scale of the observations. Exponential smoothing is a simple strategy (with minimal parameter tuning) where recent past observations are given higher weight in the smoothing process. Simple exponential smoothing techniques have been employed within the airline industry to model booking curves (Weatherford, 2016; Shintani and Umeno, 2023). However, given that the time frames are not necessarily spaced equally in time, it remains to see whether smoothing in this time scale is representative of the true demand processes under consideration.

Finally, we randomly perform an 80/20-split of the data in a train and a test set. All of the models discussed in the following are trained on the train set and evaluated on the test set.

3.3 Linear Regression

A machine learning model may have several features, but some features might have a higher impact on the output than others. For example, if a model is predicting flight demands, the weekday might have a higher impact on the output than the airport destination. Hence, we need to come up with the concept of weights.

Each feature is associated with a weight, i.e., the higher the feature has an impact on the output, the larger the weight associated with it. To find the proper weight for each feature, we need to apply the gradient descent method.

Gradient descent is an optimization algorithm that helps to find the optimal weights for the model. It uses a cost function to find the proper weights that fit the model in the best way. The cost function is defined based on the difference between the actual output (demand) and the predicted output. We consider the following cost function:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(X^{(i)}, \theta) - y^{(i)} \right)^2, \tag{1}$$

where m is the number of data points, $h(X^{(i)})$ is the output of our hypothesis for a particular value of i, and $y^{(i)}$ is the value of the exact output which in our example is demand. We consider the following function as a predicted value for demand:

$$h(X) = \beta + \theta X, \tag{2}$$

where β is the bias, X is the vector of features and the vector θ is the weights. The goal is to find a minimum of a cost function. To do this, we use the gradient descent formula:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta} J(\theta). \tag{3}$$

As you can see, the learning rate of gradient descent plays a crucial role in this formula. The large learning rate can give us poor values for β and θ. To understand which value of learning rate is better for data sets, we plot the cost function for 500 iterations and compare the results with different values of learning rate.

3.4 Decision-tree-based Methods

3.4.1 Decision Tree Regression

Decision trees are non-parametric supervised learning algorithms for classification and regression tasks (Breiman, 2017). They aim to predict the value of a target variable by learning simple decision rules (if-then-else) inferred from the data features. Formally, a decision tree recursively partitions the feature space such that the samples with the same labels or similar target values are grouped together. Setting a fixed maximum depth for the tree, at a node k, each candidate split of the feature space is presented by two elements: a feature index $j \in \mathbb{N}$ and a threshold $t_k \in \mathbb{R}$, in order to partition the data into two subsets: $\{(x, y)|x_j \leq t_k\}$ and $\{(x, y)|x_j > t_k\}$. The quality of a candidate split of the node is assessed using an impurity, or

loss, function H, which essentially captures how well the data is grouped together. We opt for the combination (j, t_k) that minimizes the impurity. Once the maximum depth is reached, each leaf node is assigned the average target value of the training data that lands in that leaf node. For a new set of features, the predicted value is equal to the value of the leaf node that it ends up in.

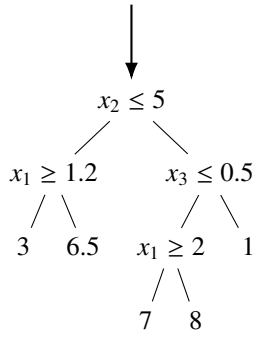

Figure 2: Example of a decision tree with a uni-variate target value.

This procedure can be refined by considering multiple features at each time to split on, which vastly increases the number of possibilities. Similarly, one often does not only want to specify a maximum depth, but also a minimum number of samples that is required to further split the data, as well as the minimum number of samples that is required to be in each leaf node (to avoid overfitting). Together, these parameters determine whether or not the recursive splitting will be terminated at a certain point in the tree. It can, therefore, happen that the tree is deeper in one 'direction' than in another; an example is given in Figure 2. Ideally, all of the parameters should be chosen in a systematic way, e.g., through cross validation. However, due to the time constraints of this project, we have simply set what we deemed 'reasonable' values for all of the parameters.

In our case, we use a decision tree as a regression algorithm to estimate the booking curve over 30 time frames. Since this is a problem with several outputs to predict, we can proceed in one of two ways:

- build a single tree model capable of predicting simultaneously all 30 outputs, i.e. the whole booking curve, with the underlying assumption that the outputs are correlated.

- assume there is no correlation between the outputs and build 30 independent models, one for each time frame, and then use those models to independently predict each one of the 30 outputs.

Since the outputs for different time frames seem uncorrelated, we opt for the latter implementation. The one downside to this approach is that the predicted booking curves are not necessarily strictly increasing, even though this is the case in the data. To compensate for this, we take the running maximum.

3.4.2 Random Forests

While decision trees are highly useful because of their simplicity and interpretability, they suffer a lot from overfitting to the data. One way to overcome this is through the use of random forests. Random forests are a way of averaging multiple decision trees trained on different parts of the data, which reduces the variance of the method (Ho, 1995; Breiman, 2017). This often leads to better performance, but the downside is that the method is less interpretable. The algorithm for training random forests essentially applies the bootstrap and aggregating (bagging) approach with some extra randomization; it is described in Algorithm 1. After training, predictions can be made by averaging the predictions made by the individual trees that make up the random forest. The training procedure for random forests obviously takes longer than for a single decision tree, so we have chosen here to train a single random forest that predicts all 30 outputs at once due to time constraints. Furthermore, as above, we have simply chosen reasonable settings for the involved hyperparameters.

Algorithm 1: Random forest training

Input: Number of trees B, number of training samples K, number of features p

for $b = 1$ **to** B **do**
  Randomly sample K training examples from $\{(\vec{x}_i, \vec{y}_i)\}_{i=1}^{n}$ with replacement;
  Train a decision tree on the bootstrapped data set, using p randomly chosen features at each split;
end

An advantage of the random forest over other methods of aggregating decision trees is that the predictions can be accompanied by a confidence interval (Zhang et al., 2019). This stems from the fact that, for each data point i, there is a sub-forest of roughly the size $\exp(-1)B$ that is trained without using i-th data point. We can evaluate how far off this sub-forest is when predicting $\vec{y}_i$, i.e. consider $\hat{y}_{\text{sub}} - \vec{y}_i$. This can be done for all data points, after which we can consider the α-quantile and the $(1-\alpha)$-quantile of the sub-forest prediction errors and output those as error margins around our estimate. If data are i.i.d., then this is an asymptotically valid $(1-\alpha)$ confidence interval as n, and B go to infinity (see (Zhang et al., 2019, Section 3) for the underlying assumptions). We have confirmed on the test set that the confidence intervals that this procedure generates indeed contain the true value in roughly $(1-\alpha)$ of the cases. For the other methods considered, we have not found a similar procedure.

3.4.3 Light Gradient Boosting Machine

Similar to a random forest, the gradient boosting machine (GBM) (Friedman (2001)) is also an ensemble model of decision trees. But, unlike random forest regressors, the decision trees in a gradient-boosting machine are trained sequentially. Each new decision tree is trained on the residuals of the predictions from the previous ensemble of trees. This improves accuracy but is inefficient for high feature dimensions and large data sizes. The most computationally expensive part of training a decision tree is finding the best-split point. In conventional gradient boosting, this is done by scanning all data to estimate the information gain of all possible

splits for each feature.

Light gradient boosting machine (LightGBM) (Ke et al. (2017)) significantly improves the efficiency of GBM using two key techniques. Firstly, data instances with small residuals are excluded as data instances with larger residuals are more important in calculating information gain. This technique called gradient-based one-side sampling (GOSS), significantly reduces the data size. Secondly, a technique called exclusive feature bundling (EFB) reduces the effective feature size using a greedy algorithm to bundle features based on their exclusivity, i.e., features that rarely take non-zero values simultaneously.

3.4.4 Feature Importance

Besides the interpretability, a big advantage of decision-tree-based methods is that they can be associated with a natural measure of feature importance. Indeed, for each feature, one can consider how much the impurity in the samples decreases on average when this feature is used to split the data. This gives an idea of the quality of the splits in which a feature is involved.

4 Results

4.1 Linear Regression

In Figure 3, we show the predicted value for demand based on the gradient descent method. In this method, we use the numerical values of data. The values used in this figure correspond to columns, 'DAY', 'MONTH', and 'PRICING CAB LVL'. We also illustrated the cumulative demand over time with three predicted curves corresponding to different values of α in Figure 4.

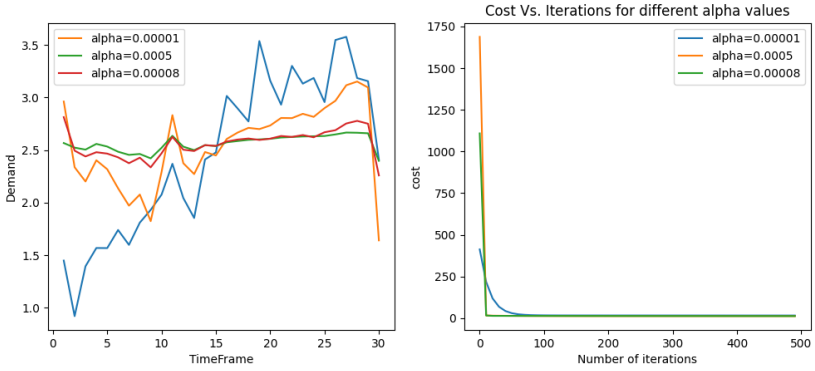

Figure 3: Mean value of demand based on time frame is displayed in the left panel. The right panel displays the cost function of the Gradient descent method based on iterations.

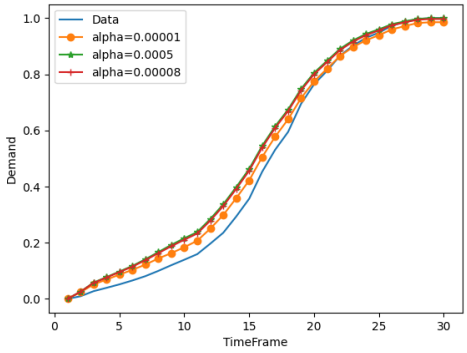

Figure 4: Cumulative demand values over time frame with different values of learning rate α.

4.2 Decision-tree-based Methods

The performance of the different tree-based methods described in Section 3.4 in terms of the four different loss functions described in Section 3 is given in Table 2. Since the LGBM model seems to work slightly better than the other methods, we plot the feature importance (see Section 3.4.4) according to this model in Figure 5. Examples of the predicted curves for three selected elements of the test set are given in Figure 6.

	Wasserstein distance	Mean squared error	Relative mean squared error
Decision tree	0.1648	0.0645	0.3652
Random forest	0.1639	0.0663	0.4560
LightGBM	**0.1565**	**0.0604**	**0.3468**

Table 2: The average loss of the tree-based methods on the test set relative to four different loss functions. The smallest loss is indicated in bold.

5 Discussion

We have considered different methods for predicting the booking curves of future flights. Because of the high computational load of the method described in Section 4.1, the analyses that we have carried out are most comparable among the decision-tree-based methods (see 3.4), so we will mostly restrict our discussion to those. With regard to the loss functions considered here, it is clear that there is very little difference in performance between a simple decision tree, a random forest, and the light gradient boosted machine. However, if one is interested in uncertainty quantification, then a random forest seems the way to go. Nevertheless, the

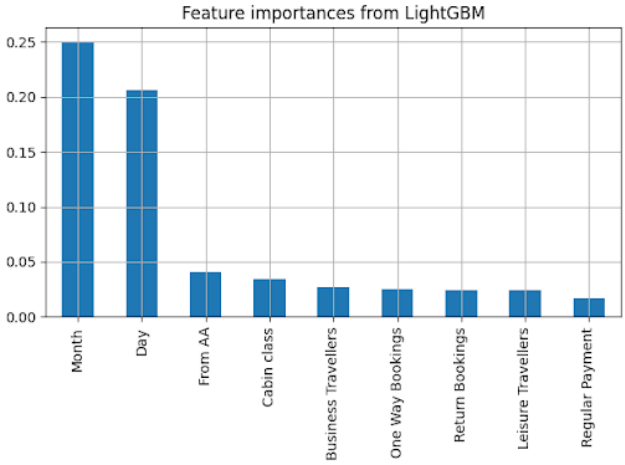

Figure 5: Feature importances according to LGBM.

LGBM seems to have a slight advantage in most cases. The features with the most predictive power seem to be, by far, the month of the flight and the day of the week.

An important point to remember is that both the decision tree and the LGBM were trained to predict each output separately, while the random forest was trained for the whole booking curve. It can be seen from the predicted curves in Figure 6 that this causes the latter to predict smoother curves, ignoring some trends in the data. Since the performance is already so close, it would be an interesting future endeavor to see how this changes when a random forest is trained for each output. Finally, it would be interesting to see how well the method in Section 4.1 does when the features are chosen according to the feature importances that are output by LGBM.

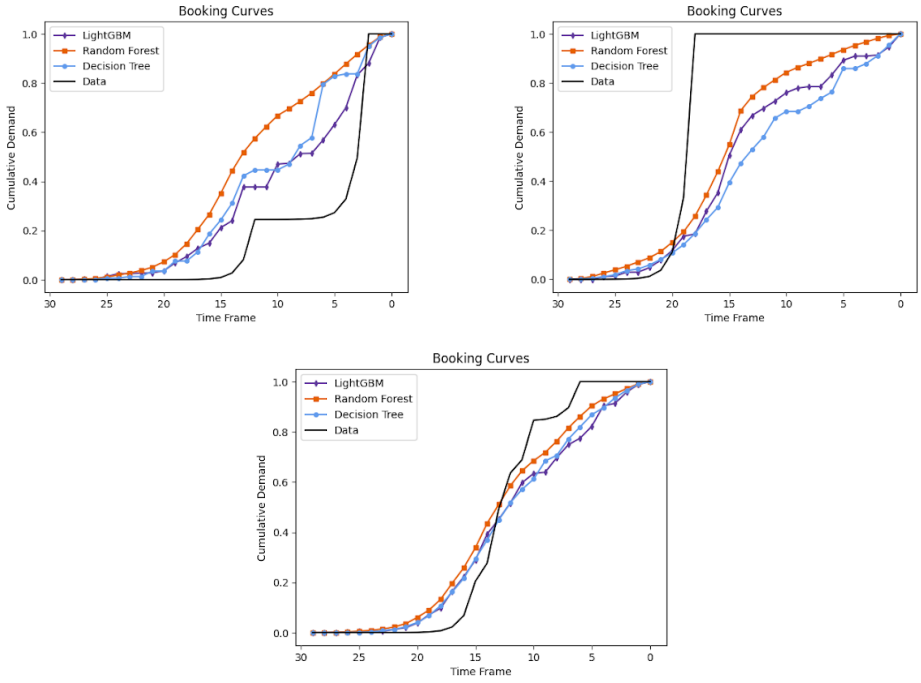

Figure 6: Three observed booking curves in the test set and the corresponding predictions by the tree-based methods.

References

Leo Breiman. *Classification and regression trees*. Routledge, 2017.

Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

Air France KLM. About air france klm, 2024. URL https://www.klm.nl/en/information/corporate/about-air-france-klm.

Masaru Shintani and Ken Umeno. Average booking curves draw exponential functions. *Scientific Reports*, 13(1):15773, 2023.

Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

Cédric Villani. *The Wasserstein distances*, pages 93–111. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-540-71050-9. doi: 10.1007/978-3-540-71050-9_6. URL https://doi.org/10.1007/978-3-540-71050-9_6.

Larry Weatherford. The history of forecasting models in revenue management. *Journal of Revenue and Pricing Management*, 15:212–221, 2016.

Haozhe Zhang, Joshua Zimmerman, Dan Nettleton, and Daniel J Nordman. Random forest prediction intervals. *The American Statistician*, 2019.