An Appendix to
What Makes a Good Image? Airbnb Demand Analytics Leveraging Interpretable Image Features

Table of Contents

I. Classifying Image Quality Using a Deep Learning-based Classification Model

1. Training Set Construction

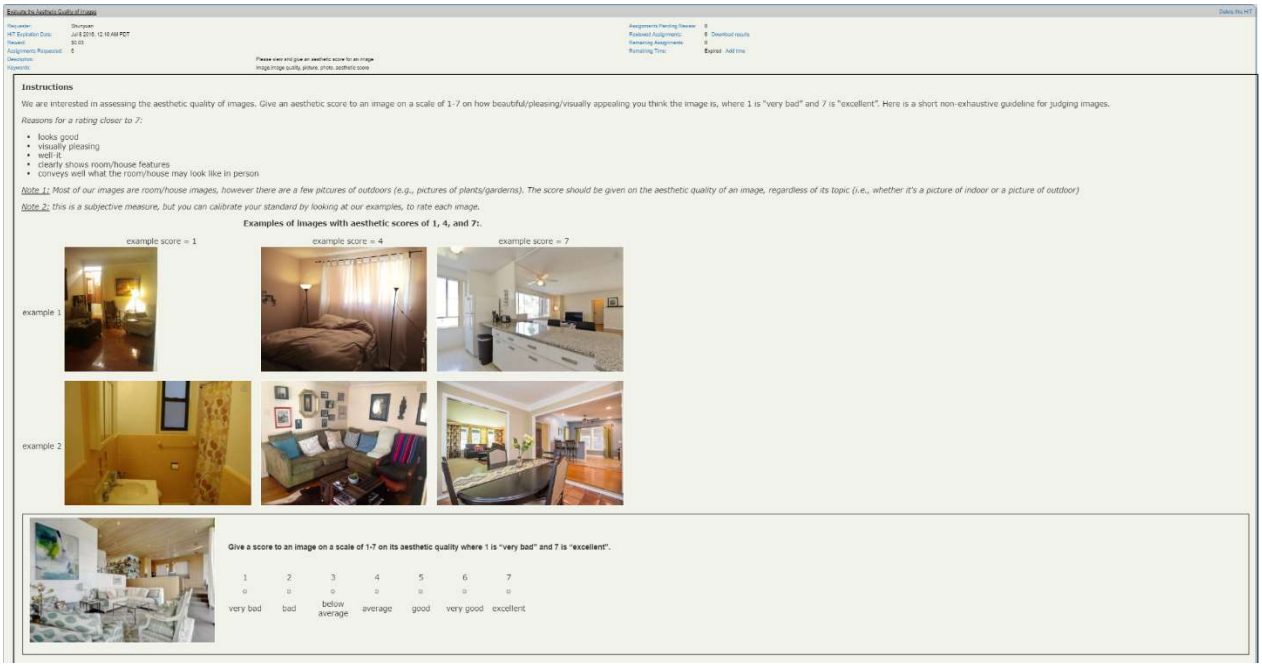
Image Quality Assessment Survey on Amazon Mechanical Turk

We describe the steps for creating the dataset to train our classifier using Amazon Mechanical Turk (AMT). AMT is a platform of Amazon Web Services that enables users to outsource small tasks to a large group of workers at a relatively low cost. It has been widely used for human intelligence tasks, such as data collection and data cleaning. As a crowdsourcing method, AMT has been found to be quite efficient and accurate (Casalboni 2015; Laws et al. 2011).

To construct a labeled training set for supervised learning, we selected 3,000 Airbnb property images from our dataset and used AMT to tag each image based on its quality. In the selection of images for AMT tagging, full random sampling was not optimal, as we did not have information on the distribution of image quality beforehand. We used stratified random sampling to ensure that a sufficient number of images were evaluated and labeled for different categories of image quality. A random sample stratified by a crude metric of quality was necessary, as it ensures that the sample is balanced and random. We randomly selected 500 images from the pool of verified images, as these were guaranteed to be taken by professional photographers and are most likely to be of high quality. Then, from all the unverified images, a human judge chose 8,000 images that *looked bad*, and 500 images from this group were randomly sampled. From the unverified images, we chose 5,000 images that we judged to be in between *excellent* and *very bad*. Of these, 500 images were randomly sampled. Lastly, we randomly sampled 1,500 images from the entire sample. Constructing the AMT data in this way ensured that we would have a sufficient random sample of images from each stratum (i.e., subgroup of images with a certain quality). We also manually reviewed the selected images to ensure that no image was repeatedly sampled.

For the AMT tagging task, we created a survey instrument that asked the Turker to assign a score to a displayed image based on its aesthetic quality. To provide accurate guidelines for image evaluation, we borrowed instructions from professional photography forums, as well as Airbnb's guidelines for shooting good property photos. We also provided example photos. The quality measurement was based on a Likert scale from 1 to 7, in which 1 is *very bad* and 7 is *excellent*. Figure A1 shows an example question on the survey given to the Turkers. To ensure high-quality and consistent responses from the Turkers, we required them to have an approval rate of higher than 95% and to have completed at least 50 approved tasks.

Figure A1. Example of the AMT Aesthetic Quality Assessment Task



Constructing the Training Set from Tagged Images

After the AMT survey, we obtained 3,000 tagged images, each of which was evaluated by five Turkers, to be pre-processed for the construction of our training set. Following previous studies on aesthetic quality (Datta et al. 2006; Datta et al. 2008; Marchesotti et al. 2011), we computed for each image, *i*, the mean aesthetic score, score_i, given by five Turkers. We then set two thresholds: $\theta_1 = \overline{score} + gap/2$ and $\theta_2 = \overline{score} - gap/2$. Next, we annotated image *i* with the term *high quality* if the average $\text{score}_i \geq \theta_1$ and with the term *low quality* if the average $\text{score}_i \leq \theta_2$. The mean score for all 3,000 images was $\overline{score} = 4.5$. Images with an average score of θ_1 and θ_2 were excluded from the training set, leaving an artificial gap between high- and low-quality images. As argued by Datta et al. (2006), the reasons for creating a gap between high- and low-quality images is that close aesthetic scores (e.g., 4.4 and 4.5) are unlikely to reflect that the images differ in aesthetic quality; rather, they indicate noise in the quality measurement process. Increasing the value of the gap makes it easier to distinguish between high- and low-quality images, but it leads to a smaller training set, as images lying within the gap are dropped. To choose an optimal value, we varied the value of the gap and selected the one that resulted in the best performance of the trained classifier on a hold-out set (Datta et al. 2006; Marchesotti et al. 2011). Using this method, we chose a gap of 0.8, which left us with a training set of 2,259 images. As 50% of the training set was selected at random, we repeated the analysis with the selected 50% and found similar results for the image labeling tasks.

2. Image Quality Classifier Training

Architecture of the Convolutional Neural Network (CNN) Framework

After a training set was constructed, the next task was to build an image quality classifier using labeled data. We applied CNN, an emerging deep learning framework that is widely applied in the field of computer vision and has been shown to perform very well for tasks, such as object recognition and image classification (Krizhevsky et al. 2012; Simonyan and Zisserman 2015).

As shown in Figure A2, a CNN model consists of a sequence of layers, each having multiple *neurons*. The number of neurons can vary from one to thousands. These neuron layers perform matrix multiplication based on an input, generating an output to serve as the input for the next layer. Both the input and output take the form of multi-dimensional matrices. The sequence of layers makes the neural network *deep*.

Images serve as the first input for the deep learning framework. In our training task, we resized all the images to 224 × 224 pixels, determined the pixel intensity of each image, and represented the image with a 3D array (matrix) that contains pixel information for the three channels (RGB). This was done to alleviate the computational burden and ensure that the image size aligned with the pre-trained VGG16 model (described below).

The last output layer predicts the binary label for its input, which, after passing through the whole network, is an *N*-dimensional vector extracted from the image. For an image in our training set (represented by IMG^k), the output layer applies a sigmoid function and predicts the label regarding the image's quality:

$$\widehat{\text{Label}}\left(\text{Image Quality}\middle|IMG^k\right) = \begin{cases} 1\ (high\ quality) & if\ \frac{1}{1+\exp\left(-(X^T W_1+W_0)\right)} \geq 0.5 \\ 0\ (low\ quality) & if\ \frac{1}{1+\exp\left(-(X^T W_1+W_0)\right)} < 0.5 \end{cases},$$

where X^T represents the output from the layer preceding the output layer (in our model, this was the FC2 layer, which produces a 4096*1 vector), W_1 represents the weight parameters, W_0 represents the bias (a constant) connecting the preceding layer to the output layer, and $\frac{1}{1+\exp\left(-(X^T W_1+W_0)\right)}$ is the probability that the image is of high quality, given the X^T and $W_1$and W_0 values for the output layer.

Throughout the CNN model, a sequence of weights on each layer define the intermediate extracted vectors from each layer, including X^T. These weights are adjusted during the training process to optimize the model's predictive performance.

Operation of Key Layers in the CNN

A few key layers determine the performance of a CNN: the convolution layer, the zero-padding layer, and the max-pooling layer. We describe these below.

Convolution Layer

The convolution layer is the most important and unique layer in the CNN. It consists of a stack of convolution filters or convolution kernels. For example, the two convolution layers in Layer Block A (shown in Figure A2) consist of 64 and 128 convolution filters, respectively. A convolution filter is a matrix in which each element represents a numeric value. For example, in Layer Block A, the convolution layers have a size of 3 × 3 and hence consist of nine numeric values.[1] This matrix, treating an image or an intermediate input as a matrix, operates dot production by *sliding* through the input. For an input of a relatively large size (e.g., 224 × 224), a 3 × 3 convolution filter operates dot production for every 3 × 3 section. The convolution operation is beneficial because it reduces the dimensionality of parameters and well explores and reserves the (local) spatial relationships of the input. Regarding the second benefit, if a convolution kernel extracts an oriented edge of an object, operating this kernel on every small square (e.g., 3 × 3) of an image would enable the extraction of all edges with that direction from the image. Many kernels that extract edges do so in all directions, potentially constructing the contour of an object. As can be seen in Figure A2, each block consists of a varying number of convolutional filters (e.g., 64, 128, 256, and 512 filters). These kernels extract features from the input data, which represent the extracted features from the preceding layers. Toward the output layer (i.e., layers closer to the output layer) in the CNN, the filters combined together extract higher- and lower-level features. That is, the CNN can extract a hierarchical structure of related features to predict the output labels.

Zero-Padding Layer

The zero-padding layer adds numerical arrays consisting of all 0 values to the edge of an intermediate output from a layer. The size of the zero-padding layer is a hyperparameter in the CNN model. Typically, as was done in our model, the intermediate output is padded with 1*M 0 vectors on each side, causing the width and height to both increase by 1 after the zero-padding value. As zeros do not contribute to the matrix multiplication procedure, the zero-padding layer does not affect the features extracted by the layers. In addition, the zero-padding layer allows us to control the spatial size of the intermediate outputs, and it can prevent the outputs from decreasing too quickly after layers of convolution operations.

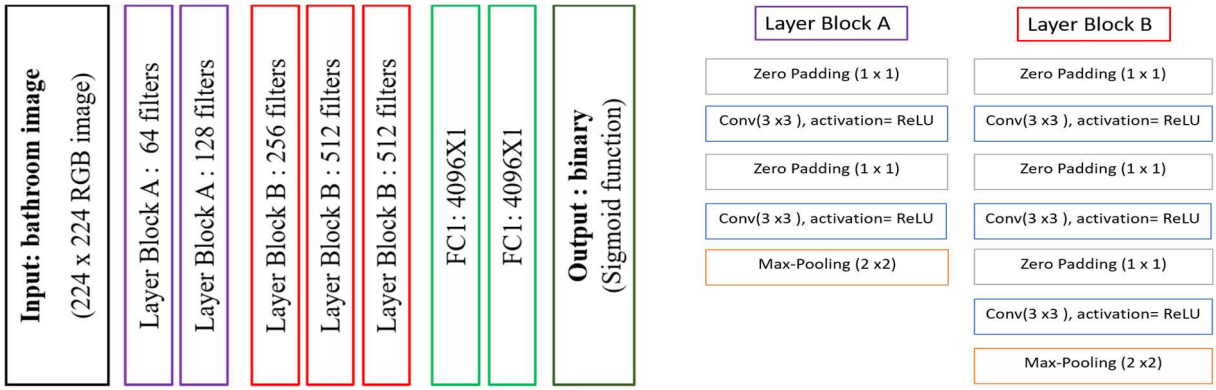
Max-Pooling Layer

Inserting a max-pooling layer between successive convolution layers is common in CNNs. A max-pooling layer is a small square filter (in our model, a 2 × 2 matrix). Similar to the convolution filter, a max-pooling layer applies to every 2 × 2 square patch in input data. Its function is to select and preserve only the maximum value

[1] The size of a convolution layer is a choice of the model architecture. A 3 × 3 or 5 × 5 configuration is common.

in that 2 × 2 square. Adding max-pooling layers can reduce the spatial size of the intermediate features and the dimension of the trained parameters in the model, and it can help to efficiently prevent overfitting.

Figure A2. Description of the Architecture and Description of the Layers of the CNN Classifier



Filters: Indicate the number of convolution windows (i.e., number of feature maps) on each convolution layer.

Zero-padding layer: Pads the input with zeros on the edges to control the spatial size of the output. It has no impact on the predicted output.

Max-pooling: Subsampling method. A 2 × 2 window slides through each feature map (without overlap) at that layer, and then the maximum value in the window is selected as a representation of the window. This reduces computation and ensures translation invariance.

Training the CNN

We randomly split the dataset into training and validation sets, with 80% of the examples forming the training set and the remainder being used as the validation set. To reduce the overfitting problem in the training step, we use data augmentation and implement a real-time (i.e., during training) image transformation for each image in the training sample by randomly (1) flipping the input image horizontally, (2) rescaling the input image within a scale of 1.2, and (3) rotating the image within 20° (images being rotated by a random value between 0° and 20°). This method introduces random variation in the training sample, increases the training set size, and reduces overfitting (Krizhevsky et al. 2012).

To effectively learn features from a relatively small sample size, we apply the idea of *transfer learning*, which involved building our model on top of an existing well-trained CNN model and then fine-tuning it. As the features extracted from images are generic to some extent (e.g., almost all CNNs extract edge information at the first layer), transfer learning is quite common in deep learning and is suggested as an effective approach for dealing with problems caused by limited data (Girshick et al. 2014; Lin et al. 2015; Zhang et al. 2015). In this study, we used VGG16 (Simonyan and Zisserman 2015), as it is a conceptually simple and popular pre-trained

model. We removed the last fully connected layers from the original VGG16 model, as they contain more data-specific features, and then we added the output layer as the last layer. Figure 2A presents the architecture of our image classification model. The parameters were initialized using the pre-trained weights, except for the output layers, for which the parameters were initialized following LeCun's uniform scaled initiation method (LeCun et al. 1998). Then, we fixed the parameters on the first 25 layers and fine-tuned the model. The model was trained on the training set using a NVIDIA K80 GPU, and then performance was tested on the hold-out set at the end of the training. Optimization was performed with an adaptive method of gradient descent (Adadelta optimization; Zeiler 2012) for each mini-batch of 16 examples.

II. Algorithm and Concepts for Image Attribute Computation

In this section, we define the key concepts used in the process of image attribute computation. These key concepts include image saliency, the key/salient region, and the figure-ground (FG). We first discuss each concept's definition and then present the image algorithm used to detect, extract, or compute the concept.

1. Visual (Image) Saliency

The basic unit determining image saliency is visual saliency at the pixel level. The overall saliency score for a local patch of an image can be computed based on the pixel saliency within the local region.

Definition

Saliency describes a concept that originates from visual unpredictability. In images, it is often captured by variations in, for example, boundaries and colors. Studies on cognitive psychology and computer vision have investigated how humans process and pay attention to visual information and have found that we allocate our attention to parts of given information (e.g., the regions of an image) while cognitively ignoring other parts. Visual uniqueness is salient in the sense that it easily attracts the attention of viewers.

Calculation

In general, the models proposed for calculating visual saliency are based on local contrast to surroundings. The contrasts are determined using features, such as color, intensity, edge density, and orientation. A simple example is the *gradient* of pixel intensity. A pixel with great contrast is assigned a high saliency value.

2. Salient Region

Definition

Following the definition of visual saliency, salient regions are defined as the regions of an image that have a high overall saliency score.

Detection

The detection of a salient region involves four steps: (1) segmenting an image into local *patches*, (2) assigning a saliency score to each path, (3) merging similar patches into a *region*, and (4) finding the most salient region. These steps are discussed in greater depth below.

1) *Segmenting an image*: This process generally involves grouping the pixels of an image into multiple parts containing pixels that are similar to one another. Segmentation can be based on edges (detected edges are assumed to define the boundaries of objects), colors, or other factors. A segmentation algorithm

is intended to compare two intensity differences: the difference across the boundary of patches and the difference between two neighboring pixels within the same patch.

2) *Assigning a saliency score to a patch*: Each pixel is assigned a saliency value. The saliency score of a patch is calculated by averaging the scores of all pixels within the patch.
3) *Merging patches into a region*: If neighboring patches have similar colors, then they are merged into a larger region.
4) *Finding the most salient region*: The salient region is found by selecting the region with the highest average salient score.

3. FG

Definition

The *figure* is the foreground of an image, and the *ground* is the background. Only one figure and one ground can be detected for each image. This is different from the detection of salient regions, for which multiple regions can be detected.

Detection

The figure is detected and extracted from an image, and then the ground is defined as the rest of the image. Detection of the foreground is an extension of image segmentation, as a pixel is assigned a value of either 1 (foreground) or 0 (background).

Detailed Algorithm

We used GrabCut, a state-of-the-art model for foreground extraction (Rother et al. 2004). In this model, an image is treated as a graph, with each pixel serving as a node and pixel similarity being defined as an edge. GrabCut implements the expectation–maximization (EM) algorithm and min-cut algorithm to iteratively assign a foreground/background label to pixels and to cut the graph into two subgraphs: one representing the foreground and the other representing the background.

1) Initially, an arbitrary rectangle separates the image into two parts. The pixels in the rectangle are labeled "1" (foreground), and those outside it are labeled "0" (background). The initial position of the rectangle can be arbitrary. Alternatively, one can specify the rectangle's location or hard label some pixels with good prior knowledge of where the foreground might be.
2) A Gaussian mixture model (GMM) is trained with the EM algorithm based on the distribution of pixel color statistics. From the GMM, we can determine the probability that each pixel belongs to a particular mode (or cluster). That is, the GMM labels a pixel as a *probable foreground* or a *probable background*.

3) A graph is built in which each node represents a pixel, and the edge weight between two pixels represents pixel similarity. Similar pixels will be assigned a low edge weight and vice versa. Pixel similarity can be computed based on intensity, color, or texture.
4) Two additional nodes are created on the graph: the source node and the sink node. All pixel nodes labeled as a probable foreground (probable background) are connected to the source node (sink node), with the edge weight between each pixel node and the source node (sink node) representing the probability that the pixel belongs to the foreground (background).
5) Next, the graph is cut into two parts, and the min-cut algorithm is implemented by minimizing the cost function, defined as the sum of the edge weights across all edges that are cut. The min-cut algorithm penalizes a cut if it will cause two similar pixels to be separated into two subgraphs. Intuitively, if two pixels both have a high probability of being in the foreground (background), then it is desirable for them to be labeled "1" ("0") at the end of this iteration.
6) Steps 2–5 are repeated until convergence in the pixel labeling is achieved.

III. Measurement of Interpretable Image Attributes

Using the computer vision algorithm described above, we perform image processing tasks to segment images into patches and detect key/salient regions. After salient regions are detected, subsequent computation is performed to measure image attributes. This section discusses the steps for computing image attribute measurements after image processing.

a. Composition

Four image attributes are categorized during the composition step. How well an image performs concerning a particular attribute is evaluated by distance, such as the distance between two pixels. A smaller distance indicates better performance. For all four composition attributes, we compute the distance metrics and then subtract the metrics from zero.

Diagonal Dominance (Attribute 1): Diagonal dominance captures how close an image's key region is positioned to the two diagonals of the image. For an image, we first identify the key region and then measure the weighted Manhattan distance from the key region to each diagonal (Liu et al. 2010; Wang et al. 2013).[2] The measurement of diagonal dominance is computed by subtracting the minimum weighted distance from zero. A greater diagonal dominance value suggests that the image is more diagonally dominant.

Rule of Thirds **(ROT)** (Attribute 2): We first divide an image into nine equal parts with two (imaginary) equally spaced vertical lines and two (imaginary) equally spaced horizontal lines. Then, we calculate the Euclidean distance from the centroid of the key region to each of the intersections (Wang et al. 2013). If the minimum distance is small, then the image follows the ROT, with its key region close to at least one intersection. The ROT is measured by subtracting the minimum distance from zero. If an image follows the ROT more closely, the ROT value is greater.

Visual Balance Intensity (Attribute 3): In this measurement, we split the image along its vertical central line. On each half of the image, we identify a key region and compute the distance from its centroid to the vertical central line (Liu et al. 2010). A relative distance measure is calculated by subtracting the shorter distance from the longer one and dividing the difference by the longer distance. Then, visual balance intensity is computed by subtracting the relative distance from zero. A greater value for this measure suggests that the image is more (vertically) visually balanced in terms of pixel intensity.

Visual Balance Color (Attribute 4): The color measurement of visual balance compares the left half of an image with the right half based on color. We first calculate the Euclidean distance in terms of color intensity

[2] The Manhattan distance between two points on an image is measured as the number of pixels between them, with only horizontal and vertical paths allowed.

(i.e., RGB channel) between each pixel and its symmetrical pixel (across the vertical line). Then, visual balance color is computed by subtracting the average difference from zero. A greater value suggests that the image is more visually balanced in terms of color around its vertical central line.

b. Color

Five image attributes related to color are computed. The measurements are based on pixel intensity or related values (e.g., hue and saturation).

Warm Hue (Attribute 5): The warm hue measurement captures the warmth of an image, which is defined by the relative proportion of warm hues (e.g., yellow) to cool hues (e.g., green). The measurement is computed in the HSV (hue, saturation, and volume) space. Specifically, we calculate the pixel hues that fall outside the cool range (30–110) on the hue spectrum (Wang et al. 2013). If an image contains more warm hues, such as yellow and orange, it will have a greater warm hue value.

Saturation (Attribute 6): We compute pixel saturation in the HSV space by averaging the saturation values of all pixels in the image. A greater value indicates higher saturation (for example, the image contains more saturated colors).

Brightness (Attribute 7): The brightness of an image is defined as the overall level of illumination. We calculate the intensity of each pixel and then average the intensity values across all pixels in the image. A brighter image has a greater brightness value.

Contrast of Brightness (Attribute 8): Contrast is calculated as the *SD* of pixel intensity in the whole image. A lower contrast of brightness value suggests that the brightness is more evenly distributed across the image.

Image Clarity (Attribute 9): Image clarity captures the portion of hues with sufficient intensity. We measure pixel brightness on a 0–1 scale and then compute the portion of pixels whose brightness is 0.7–1 (Wang et al. 2013). A clear image has a higher image clarity value.

c. FG Relationship

The FG relationship is described as the difference between the figure and its ground in terms of three metrics: size, color, and texture. An image with a good FG relationship has a clearly separable figure and ground (i.e., greater differences).

Size Difference (Attribute 10): The size difference attribute compares the size of the figure with that of the ground. We detect the figure and ground in an image and calculate the size of each in relation to the whole image (Cheng et al. 2011). Then, the size difference is computed by subtracting the ground's size ratio from the

figure's size ratio. A greater size difference value indicates that the image has a figure occupying a relatively larger area, standing out from the ground.

Color Difference (Attribute 11): The color difference attribute captures the difference in colors between the figure and the ground. We compute the Euclidean distance between the mean color of the figure and that of the ground. A high color difference value suggests that the figure and ground contain distinct colors. In such cases, the figure can be easily distinguished from the ground.

Texture Difference (Attribute 12): Texture difference measures the difference between the figure and the ground in terms of texture, which is captured by the edge density within a local region. For the figure and ground, we use the Canny edge detector to detect the edge and then compute edge density. Then, we measure the absolute difference between the two densities. A high texture difference value suggests that the figure and ground have a clear separation based on texture.

IV. Room Type Classification

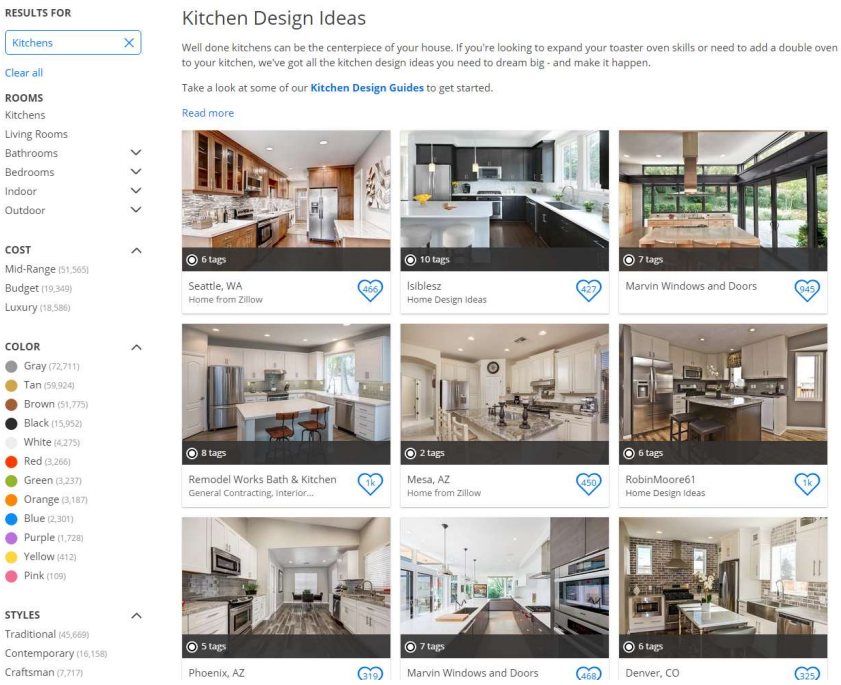
We build a deep learning model to automatically categorize the type of scene photographed. The goal is to compute the distribution of different types of rooms depicted in property images. Controlling the distribution in our demand model helps us address the concern that professional photographers know which types of places appeal more to consumers and thus present these aspects of properties.

We build a deep learning model to automatically classify the room type (bathroom, bedroom, kitchen, living room, or outdoor area) depicted in a given property image. Using transfer learning with a deep learning model that was pre-trained on a large scene classification dataset, Places205 (Zhou et al. 2014), we optimize the classifier for a dataset we collected, which consists of 54,557 images of bathrooms, 59,082 images of bedrooms, 88,030 images of kitchens, 81,819 images of living rooms, and 5,734 images of outdoor areas. The average classification accuracy is 95.05% on a hold-out set across the five categories.

Training Set of Indoor/Outdoor Photos

To train a room type classifier, we need a large number of room images labeled with a room type. We would have preferred to use original images from Airbnb.com, but the images on property web pages are not labeled, and it would incur a high labor cost to have someone (e.g., AMT) manually label images for us. Therefore, we crawled data from real estate–related websites on which a vast number of indoor/outdoor images are classified into categories. Figure A3 shows a portion of a web page displaying 23 images of kitchens at multiple properties.

Figure A3 A Real Estate Web Page Showing Kitchens



From the website, we collected images aligning with the five room types: bathroom, bedroom, living room, kitchen, and outdoor area. We then split the dataset, 80% of which was used as a training set and the remaining 20% used as a hold-out test set.

Training a Room Type Classifier on a Collected Training Set

We used the VGG16 ConvNet model, which was pre-trained on the Places205 dataset (Zhou et al. 2014) and then fine-tuned on our training set.[3] The model was used for a task to classify 205 categories of places. To transfer the pre-trained model to our study, we removed the output layer in the pre-trained model and then added an output layer designed for our specific task. The added output layer was a 5*1 vector in which each element indicated the predicted probability of assigning the corresponding label. To ensure that all the probabilities add up to 1 (as we assume that a room belongs to only one category), the Softmax function was used to calculate the predicted probability (the function ensures that all the predicted probabilities add up to 1). For example, the probability that an image is assigned room type k is computed as follows:

$$\text{Prob}(\text{Room Type} = \text{k}|\text{X}^{\text{T}}) = \frac{X^T W_1^k + W_0^k}{\sum_{j=0}^{4} X^T W_1^j + W_0^j}, \quad (1)$$

[3] The pre-trained VGG16 model (including the architecture and parameters) can be accessed at http://places.csail.mit.edu/downloadCNN.html.

where j = 0,…,4 represents the room type (bathroom, bedroom, kitchen, living room, or outdoor area, respectively), X^T represents the output from FC2 (the second fully connected layer), W_1^j represents the weight connecting FC2 to the j^{th} node on the output layer, and W_0^j represents the bias connecting FC2 to the j^{th} node on the output layer.

V. Robustness Checks, Empirical Extension, and Exclusion of Alternative Explanations

This section reports a series of analyses performed to test the robustness of our main results and/or exclude alternative explanations. We begin by presenting the validation of the propensity score method used in our empirical model, which included a propensity score weighting (PSW) strategy and a sensitivity assessment of unobservables.

1) Validating the Propensity Score Method

To ensure that the propensity score (PS) approach effectively eliminates potential systematic imbalances between the treatment and control groups, one needs to show that the PSs have balanced the covariates for matched or weighted samples.

We implemented a balance check, which compares, in terms of the covariates, the weighted means of the treatment group, $\bar{\boldsymbol{X}}_{\boldsymbol{treatment}} = \frac{\sum_{i \epsilon treatment} \omega_i \boldsymbol{X_i}}{\sum_{i \epsilon treatment} \omega_i}$, and the control group, $\bar{\boldsymbol{X}}_{\boldsymbol{control}} = \frac{\sum_{i \epsilon control} \omega_i \boldsymbol{X_i}}{\sum_{i \epsilon control} \omega_i}$. Here, X_i is a 1*M dimensional vector of the pre-treatment covariates (i.e., the covariates observed before the treatment) of unit i, and ω_i is the sample weight for unit i, computed based on the estimated PSs.

Table A1 presents the weighted group means and tests for the differences in the means for each variable X^m (m = 1, 2, …, M). As shown by the *t*-statistics in the table, the weighted samples are not statistically different at the 95% significance level. That is, the systematic differences in the weighed samples are negligible after performing the PSW method. Therefore, we validated that our PSW method effectively eliminated imbalances in the sample and that our weighted treatment and control groups are comparable in terms of the observed covariates that may affect the treatment selection process.

Table A1. PSW Validation: Covariate Balance Check

	Weighted Means in Groups		***T*-test**	
Variables	**Treated**	**Untreated**	***t***	***p*-value**
REVIEW_COUNT	20.56	19.88	0.27	0.790
IMAGE_QUALITY	0.27	0.25	1.00	0.316
IMAGE_COUNT	14.48	15.1	−0.67	0.506
NIGHTLY_RATE	170.15	191.36	−1.14	0.257
MINIMUM_STAY	2.57	2.57	−0.00	1.000
MAX_GUESTS	3.5	3.67	−0.82	0.410
RESPONSE_RATE	92.25	91.19	0.79	0.431
RESPONSE_TIME (minutes)	225.12	260.98	−1.18	0.238

SUPER_HOST	0.15	0.11	1.05	0.292
INSTANT_BOOK	0.11	0.11	−0.14	0.888
# BLOCKED DAYS	9.51	8.32	1.10	0.271
# RESERVATION DAYS	6.62	6.74	−0.16	0.877
PARKING	0.5	0.49	0.09	0.929
POOL	0.1	0.08	0.76	0.445
BEACH	0.02	0.02	0.34	0.737
INTERNET	0.99	1	−0.58	0.563
TV	0.79	0.81	−0.55	0.579
WASHER	0.6	0.57	0.81	0.419
MICROWAVE	0.15	0.13	0.64	0.523
ELEVATOR	0.2	0.21	−0.22	0.826
GYM	0.11	0.13	−0.69	0.490
FAMILY_FRIENDLY	0.19	0.2	−0.56	0.576
SMOKE_DETECTOR	0.55	0.52	0.62	0.534
SHAMPOO	0.45	0.44	0.09	0.929
BATHROOM_PHOTO_RATIO	0.22	0.21	1.05	0.295
BEDROOM_PHOTO_RATIO	0.29	0.29	−0.26	0.795
KITCHEN_PHOTO_RATIO	0.1	0.1	−0.15	0.880
LIVINGROOM_PHOTO_RATIO	0.18	0.19	−0.52	0.602
OUTDOOR_PHOTO_RATIO	0.2	0.21	−0.13	0.898
SECURITY_DEPOSIT	202.77	225.19	−0.65	0.517
# OPEN DAYS	21.49	22.68	−1.10	0.271
CANCELLATION_STRICT	0.26	0.29	−0.89	0.371
# BEDS	1.81	1.96	−1.17	0.242
APARTMENT	0.6	0.61	−0.27	0.786
ENTIRE_HOME	0.61	0.64	−0.64	0.522

2) Sensitivity Analysis of the Propensity Score Method (Rosenbaum Bounds Test)

In the model for estimating the propensity scores, we included a rich set of variables and their interactions. The inclusion of a complete set of covariates reduced the odds that our main results would be affected by variables that were not accounted for when computing the propensity scores.

Despite the long list of included covariates for propensity scores estimation, some variables that may affect one's decision regarding participation in a professional program were omitted. As is commonly acknowledged,

PSs are computed based on observed variables; therefore, there may be a hidden bias if unobserved variables simultaneously affect the selection process (i.e., the treatment assignment) and the outcome variables. To assess the sensitivity of our estimation to potential hidden bias, we implement a widely adopted approach of sensitive analysis—the Rosenbaum bounds test (Rosenbaum 2002).

The logic of the Rosenbaum bounds analysis is as follows. Suppose the participation probability of unit i is $P_i(treated_i = 1|x_i, u_i) = f(\beta x_i + \gamma u_i)$, where x_i and u_i are the vectors of observed and unobserved variables, respectively, and γ is 0 if there are no unobserved variables affecting the treatment selection process. Units i and j with $x_i = x_j$ have the same probability of receiving the treatment if and only if $\gamma(u_i - u_j) = 0$. Rosenbaum bounds evaluate the degree of minimum change in the odds ratio (OR) of participation due to unobservables would be required to nullify the treatment effect identified with the PS method. The estimation results inspire more confidence if a greater change in the OR caused by unobservables is needed to overturn the estimated treatment effect.

The results of the Rosenbaum bounds test are provided in Table A2. As our main DiD (Difference in Differences) analysis identified that verified photos had a positive effect on property demand, we were more concerned with potential upward (positive) than downward (negative) bias in the DiD estimator. Therefore, in Table A3, we are mostly interested in the sig+ column. The gamma results suggest that even when a hypothetical unobservable increases the OR by 1.4 times, our causal inference of the positive treatment coefficient of professional images on property demand, identified with the PS method, will be robust at the 95% confidence level (and will remain positively significant at the 90% confidence level until the gamma is increased to 1.55). The results suggest that for a positive estimated treatment effect on property demand to be overturned, the potential unobserved factors affecting the treatment assignment process would have to be large enough to increase the OR of participation by at least 60%. Moreover, if we look at Hodges–Lehmann's estimates (Rosenbaum 1993), they suggest a more robust result, as the upper (t-hat+) and lower (t-hat) bounds do not contain 0, at least until 1.6.

The results of our sensitivity analysis reveal a similar increase in gamma to that reported in the extant literature (1.2- to 1.6-fold; DiPrete et al. 2004; Li et al. 2016; Manchanda et al. 2015; Sun and Zhu 2014). Although the treatment effect will be insignificant if the unobservable is large enough to change propensities, this does not mean that the PS method is invalid because Rosenbaum bounds analysis gives us a lower bound of confidence when making a causal inference in the worst-case scenario with hypothetical hidden selection bias (note that hidden bias caused by unobservables does not necessarily exist). For this reason, we are confident that our study is, to some extent, robust to hidden bias caused by hypothetical unobserved factors affecting the selection process.

Table A2. Sensitivity Analysis: Rosenbaum Bounds Test

Gamma	Significance Level		Hodges–Lehmann Point Estimate		95% Confidence Interval	
	sig+	sig-	t-hat+	t-hat	CI+	CI-
1	0.000205	0.000205	13.9456	13.9456	5.50612	24.1935
1.05	0.00052	0.000075	12.9032	15.8307	4.83871	25
1.1	0.00119	0.000027	12.5	17.5824	3.22581	26.4631
1.15	0.002482	9.50E-06	11.2903	18.3333	2.2043	27.7778
1.2	0.004777	3.30E-06	10.2716	19.3548	1.25353	29.6461
1.25	0.008569	1.10E-06	9.05707	20.2419	−4.40E-07	30
1.3	0.014442	3.80E-07	8.06452	21.4286	−4.40E-07	30.8405
1.35	0.02304	1.30E-07	6.45162	22.4194	−4.40E-07	32.0079
1.4	0.035005	4.20E-08	6.45161	23.5526	−4.40E-07	33.1349
1.45	0.050917	1.40E-08	5.50612	24.1935	−4.00E-06	33.8095
1.5	0.071236	4.50E-09	5	24.7878	−1.0326	35.0824
1.55	0.096249	1.50E-09	4.07337	25.8064	−2.01613	35.7692
1.6	0.126037	4.70E-10	3.22581	26.7374	−3.10676	36.7374

Note: Gamma log odds of differential assignment because of unobserved factors; sig + (−): upper (lower) bound of the significance level; t-hat + (−): upper (lower) bound of the Hodges–Lehmann point estimate CI + (−): upper (lower) bound of the confidence interval (a = 0.95).

3) Addressing the Possibility of an Inflated Long-Term Effect

i. Analyses of a Shorter-Period Sample

This is an alternative approach to address the concern about the possible long-term inflation of the treatment effect if Airbnb's search algorithm favors properties with professional photos. To address this concern, we

estimate our main DiD specification on a set of subsamples in which we included a shorter period (8 months) for each treated unit.

For each treated unit, the observation spans exactly four periods per property before and after treatment adoption. For the untreated (control) units, we used the full periods (16 months) for estimation. This is because for untreated units, there is no reference period for which we can define the pre- and post-spans.

Table A3 presents the estimation results for the robustness analysis of the selected subsets. As shown, the estimated coefficients of the key variable, *TREATIND*, remain consistently positive and significant. The consistent estimated treatment effect when shorter periods are applied adds confidence that our main finding was not driven by long-term inflation caused by Airbnb's search ranking algorithm.

Table A3. DiD Robustness Tests: Selecting Shorter Periods of Samples (Limited Eight Periods/Month Spans)

Variables	Subset of four pre- and four post-treatment periods for treated units and a whole period for untreated units	
	Estimates	**Robust S.E.**
TREATIND	10.33***	1.624
log REVIEW_COUNT$_{t-1}$	10.03***	0.768
NIGHTLY_RATE	−0.175***	0.0262
INSTANT_BOOK	4.436***	1.169
CLEANING_FEE	0.0898***	0.0125
MAX_GUESTS	−2.142*	1.034
RESPONSE_RATE	0.0767*	0.0348
RESPONSE_TIME (minutes)	−0.00152	0.00136
MINIMUM_STAY	0.0279	0.0873
SECURITY_DEPOSIT	0.00298*	0.00143
SUPER_HOST	1.989*	1.003
BUSINESS_READY	0.902	0.919
CANCELLATION_STRICT	0.521	1.045
HAS_RATING	6.380	8.310
HAS_RATING × COMMUNICATION	0.204	1.336
HAS_RATING × ACCURACY	0.00741	1.056
HAS_RATING × CLEANLINESS	−2.015	1.035
HAS_RATING × CHECKIN	−1.142	1.398
HAS_RATING × LOCATION	0.805	1.074
HAS_RATING × VALUE	1.552	0.998
Fixed Effect	Property	
Seasonality	City-Year-Month	
Observations	75,406	
R-squared	0.6943	

Note: The number of observations is smaller than that in the main analyses (Tables 2, 3, 4, and 7 in the main paper) because a shorter period was used for the treated units. Specifically, instead of using the whole panel of 16 periods of data, we used up to eight periods for the estimation in this table.
*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$.

ii. Potential Dynamics in the Treatment Effect (Robustness Test of Possible Long-term Inflation)

To address the concern that the estimated effect could be inflated in the long term if Airbnb's ranking algorithm favors properties with professional images, we implemented a relative-time model to break the *after* period into a series of short periods, and we estimated the treatment effect for each month following treatment adoption. We again relied on a relative model to estimate the following:

$$DEMAND_{it} = INTERCEPT + \sum_j \beta_j (PRE_{it}(j) \cdot TREAT_i) + \sum_k \beta_k (POST_{it}(k) \cdot TREAT_i) + \lambda CONTROLS_{it} + SEASONALITY_{cym} + PROPERTY_i + \varepsilon_{it}.$$

The inclusion of the period leads, *POST(k)*, allowed us to examine the possible dynamics in the treatment effect across periods after adopting the treatment. That is, if the treatment had a greater impact a certain number of months after the treatment adoption, we expected that the estimated coefficient associated with that period dummy would be greater.

As shown in Table A4, the treatment effect exhibits dynamics in the post-treatment period. Specifically, the effect size of the treatment increases over time before stabilizing (decreasing). We interpret this as follows: in the month in which the treatment was adopted, (*POST(0) *TREAT*), we may have underestimated the treatment effect, as some bookings may have been made in previous periods. In later periods, the estimated coefficient of the verified photos initially increases before stabilizing. There are two forces that may lead to this change:

(1) All the demand in these periods is due to the verified photos and not the unverified ones.

(2) With an increased demand, Airbnb ranking algorithms may have started showing this property higher in search rankings, leading to more demand.

Comparing the effect size in the month following the treatment month (i.e., the coefficient of *POST(1) *TREAT*) to the treatment effect obtained from the main model, we see that the results are consistent (8.958 versus 9.303). The coefficient of interaction term with the last period dummy, *POST (5) *TREAT*, decreased compared with the estimated coefficient for $\{POST\ (k) * TREAT\}_{k=1}^{4}$. This is likely because we have fewer properties for which more than four post-treatment periods are observed (for example, a property that adopted treatment in February 2017 contributes to the estimation of coefficients for $\{POST\ (k) * TREAT\}_{k=1}^{4}$, with k = 0–2 only).

Note that to conclusively rule out or separate any potential effect of Airbnb's ranking system that could be confounded with verified photo adoption, we would like to know whether Airbnb's ranking system considers property images and, if so, how the inclusion of images affects the ranking algorithm. Unfortunately, we do not have information about whether and how (if at all) Airbnb increases the ranking of treated properties, as the algorithm is proprietary. In addition, if the data allow, we would like to use more granular information for investigating whether Airbnb ranks properties with verified photos higher immediately after the adoption of

verified photos. For example, using the available data, one could examine whether a property received more bookings within the next couple of hours or days. Yet, such an analysis is not feasible because our demand and treatment data are at a monthly level. We knew the month in which a property adopted the treatment condition but did not have more granular information about the treatment time (e.g., time stamp, exact date, or exact week). In addition, our demand data were at a monthly level (e.g., number of days booked in a month). As a result, we could not test whether demand increased within a very short period after the properties adopted verified photos. We acknowledge that more granular data are needed to resolve this issue. The results should therefore be interpreted with this caveat in mind.

Table A4. Falsification Checks of Pre-Treatment Trends: Relative-Time Model

	DiD Model (Relative Time)	
Variables	**Estimates**	**Robust S.E.**
*PRE (−4) *TREAT*	2.674	2.494
*PRE (−3) *TREAT*	−0.687	3.163
*PRE (−2) *TREAT*	1.706	3.102
*PRE (−1) *TREAT* (reference month)	--	--
*POST (0) *TREAT*	6.858*	3.130
*POST (1) *TREAT*	9.303**	2.917
*POST (2) *TREAT*	12.78***	2.932
*POST (3) *TREAT*	12.92***	2.918
*POST (4) *TREAT*	13.00***	2.927
*POST (5) *TREAT*	8.818***	2.609
Property (Non-Photo) Characteristics		
log REVIEW_COUNT$_{t-1}$	9.367***	0.931
NIGHTLY_RATE	−0.145***	0.0319
INSTANT_BOOK	4.059**	1.364
CLEANING_FEE	0.0808***	0.0183
MAX_GUESTS	0.0581	1.098
RESPONSE_RATE	0.0705	0.0433
RESPONSE_TIME (minutes)	−0.000424	0.00162
MINIMUM_STAY	0.132	0.130
SECURITY_DEPOSIT	0.00162	0.00198
SUPER_HOST	3.698*	1.489
BUSINESS_READY	1.809	0.982

CANCELLATION_STRICT	1.068	1.270
HAS_RATING	13.86	12.19
HAS_RATING × *COMMUNICATION*	−0.183	1.425
HAS_RATING × *ACCURACY*	1.028	1.208
HAS_RATING × *CLEANLINESS*	−1.509	1.136
HAS_RATING × *CHECKIN*	−2.039	1.523
HAS_RATING × *LOCATION*	−0.686	1.176
HAS_RATING × *VALUE*	2.057	1.180
Fixed Effect	Property	
Seasonality	City-Year-Month	
Num. Observations	76,901	
R-squared	0.6616	

* $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

4) Adding Interaction Terms with Meaningful Amenities

One concern regarding the self-selection issue is that properties with particular amenities may be more likely to adopt the treatment. For example, if some amenities make the properties more attractive in particular seasons (e.g., a pool or beach in summer), and the hosts adopt the treatment at that time, then some of the increase in demand could be brought about by those attractive amenities. The effects of these amenities on demand (which are fixed characteristics of the property) cannot be fully accounted for by the property's fixed effect terms, as amenities' effects may be time varying (e.g., a pool has a greater effect in summer than in winter).

To address this concern, when estimating the PSs for PSW, we obtained additional data on the properties' amenities and incorporated amenity information (e.g., AC, pool, whether the property is close to a beach) to account for the possible factors that may be correlated with both property demand and the hosts' treatment adoption decision in particular seasons but that cannot be captured by property fixed effects. In addition, in the model specification, we added interaction terms for the dummy *AFTER* and meaningful amenities (e.g., pool, beach, AC) to account for the higher effect of these time-invariant variables after the treatment or in particular seasons.

In Table A5, we present the estimation results. Column (1) presents the results of Equation (2a), which is our main specification, and column (2) presents the results of Equation (2b), in which we incorporated the interaction of *AFTER* with meaningful amenities. The consistency of the estimated results confirms a positive and significant treatment coefficient of more than an 8% increase in the property occupancy rate after controlling for area-specific seasonality, as well as the time-varying effect of particular time-invariant property

amenities. In the panel "Interacting with Meaningful Property Amenities," we present the coefficients of the interaction terms. As can be seen, all coefficients are insignificant.

$$DEMAND_{it} = INTERCEPT + \alpha_3 TREATIND_{it} + \lambda CONTROLS_{it} + SEASONALITY_{cym} + PROPERTY_i + \varepsilon_{it}. \quad (2a)$$

$$DEMAND_{it} = INTERCEPT + \alpha_3 TREATIND_{it} + \lambda CONTROLS_{it} + \vartheta_1 AFTER \times POOL + \vartheta_2 AFTER \times BEACH + \vartheta_3 AFTER \times AC + SEASONALITY_{cym} + PROPERTY_i + \varepsilon_{it}. \quad (2b)$$

Table A5. DiD Model: Regressing Property Demand on Verified Photos

Variables	Main DiD Model (Equation 2a)		Interacting with Amenities (Equation 2b)	
	Estimates	**Robust S.E.**	**Estimates**	**Robust S.E.**
TREATIND (TREAT × AFTER)	**8.985*** **	**1.660**	**8.668*** **	**1.747**
Property (Non-Photo) Characteristics				
log REVIEW_COUNT$_{t-1}$	9.375***	0.930	9.403***	0.933
NIGHTLY_RATE	−0.146***	0.0320	−0.149***	0.0320
INSTANT_BOOK	4.156**	1.361	4.168**	1.362
CLEANING_FEE	0.0808***	0.0184	0.0814***	0.0185
MAX_GUESTS	0.260	1.117	0.305	1.111
RESPONSE_RATE	0.0699	0.0430	0.0693	0.0430
RESPONSE_TIME (minutes)	−0.000477	0.00161	−0.000591	0.00161
MINIMUM_STAY	0.133	0.131	0.136	0.131
SECURITY_DEPOSIT	0.00177	0.00201	0.00171	0.00199
SUPER_HOST	3.801*	1.494	3.781*	1.494
BUSINESS_READY	1.806	0.985	1.791	0.984
CANCELLATION_STRICT	1.016	1.271	1.040	1.271
HAS_RATING	14.32	12.25	14.56	12.24
HAS_RATING × COMMUNICATION	−0.212	1.420	−0.145	1.421
HAS_RATING × ACCURACY	0.878	1.211	0.837	1.211
HAS_RATING × CLEANLINESS	−1.344	1.133	−1.323	1.134

HAS_RATING × CHECKIN	−2.060	1.526	−2.118	1.526
HAS_RATING × LOCATION	−0.757	1.183	−0.767	1.183
HAS_RATING × VALUE	2.141	1.176	2.142	1.173
Interacting with Meaningful Property Amenities				
AFTER × POOL			6.157	4.692
AFTER × BEACH			−10.77	10.83
AFTER × AC			1.034	3.990
Fixed Effect	Property		Property	
Seasonality	City-Year-Month		City-Year-Month	
No. of Observations	76,901		76,901	
R-squared	0.6608		0.6609	

Note: *$p < 0.05$, **$p < 0.01$, ***$p < 0.001$.

5) Testing Changes in the Property or the Host's Unobserved Quality (via Multidimensional Ratings)

It is possible that properties or hosts self-select the adoption of professional Airbnb images when there are substantial changes in the quality of the experience delivered to guests. Such changes could include an upgrade of the house/room or the delivery of more friendly/better services to guests. This would introduce an upward bias in the estimated coefficient, as it could happen at the same time as the treatment adoption. Although we were not able to observe and control for everything, we adopted a few measures to help control for or alleviate this issue.

First, in the demand model, we added the measurement of host responsiveness (host response rate and host response time) to address the concern that hosts are more responsible in the post-treatment period.

Second, in the demand model, we added a complete set of multidimensional guest ratings to capture and account for any potential changes in the stay experience or hosting quality.

Third, as we show below, for the properties for which ratings are available, we compared the average ratings in terms of multidimensional aspects for a few periods before and after the treatment. The goal is to examine whether there are substantial changes in the property characteristics that were unobserved by us but captured in guests' ratings. To implement this robustness test, we estimated the following:

$$Rating_{it} = INTERCEPT + \alpha_3 TREATIND_{it} + \lambda CONTROLS_{it} + SEASONALITY_{cym} + PROPERTY_i + \varepsilon_{it},$$

where the specification is the same as that in our main demand model, and the dependent variable is replaced with one of the following guest ratings capturing, to some extent, how good a stay or host was: communication,

accuracy, cleanliness, and check-in. The metrics for cleanliness capture potential changes in the property, whereas those for communication capture the quality of communication between a host and a guest. The coefficient α_3 captures changes in ratings along a particular dimension after treatment adoption.

As shown in Table A6, the coefficients of *TREATIND* are insignificant across all specifications, suggesting that for the treated units, there was no substantial change in the stay or hosting quality delivered to guests before and after adopting verified photos.

Table A6. Robustness Test: Changes in Multidimensional Guest Ratings after Verified Photo Adoption

	DV: Multidimensional Guest Review Rating			
Variables	**Communication**	**Accuracy**	**Cleanliness**	**Check-in**
TREATIND	−0.0155	0.0175	−0.0261	−0.0245
	(0.0118)	(0.0167)	(0.0193)	(0.0177)
log REVIEW_COUNT$_{t-1}$	−0.0371	0.0277	−0.0582*	0.0261
	(0.0189)	(0.0265)	(0.0286)	(0.0285)
NIGHTLY_RATE	0.00189***	−0.000621	0.00123*	0.000400
	(0.000297)	(0.000344)	(0.000546)	(0.000394)
INSTANT_BOOK	−0.0134	0.0154	−0.0174	0.0162
	(0.0153)	(0.0187)	(0.0197)	(0.0158)
CLEANING_FEE	−0.000325	0.000604	−0.000343	−0.000104
	(0.000208)	(0.000378)	(0.000665)	(0.000318)
MAX_GUESTS	−0.0101	0.00335	0.0449***	0.0106
	(0.00728)	(0.00814)	(0.0125)	(0.00988)
RESPONSE_RATE	−0.000951**	0.000425	−0.000782	0.000130
	(0.000332)	(0.000367)	(0.000504)	(0.000357)
RESPONSE_TIME (minutes)	−0.00000573	0.000000617	−0.00000225	−0.0000180
	(0.0000131)	(0.0000126)	(0.0000163)	(0.0000133)
MINIMUM_STAY	−0.00129**	−0.000554	−0.000550	0.000187
	(0.000395)	(0.000612)	(0.000739)	(0.000565)
SECURITY_DEPOSIT	0.00000253	0.0000174	0.0000104	−0.00000397
	(0.0000107)	(0.0000112)	(0.0000168)	(0.0000101)
SUPER_HOST	−0.0246*	0.0169*	0.0188	0.00510
	(0.0117)	(0.00826)	(0.0154)	(0.00905)

BUSINESS_READY	−0.00618	0.0166	0.0270	0.00700
	(0.00943)	(0.0109)	(0.0140)	(0.00982)
CANCELLATION_STRICT	−0.0281	0.0284	−0.0152	0.0236
	(0.0147)	(0.0164)	(0.0200)	(0.0194)
Fixed Effect	Property	Property	Property	Property
Seasonality	City-Year-Month	City-Year-Month	City-Year-Month	City-Year-Month
Observations	45,386	45,386	45,386	45,386
R-squared	0.8771	0.8931	0.9155	0.8562

Note: The number of observations (45,386) is lower than that in our main analyses (76,901). This is because the set of analyses presented here used a subset of samples that included only properties with guest ratings presented on the property page. Note that Airbnb computes and presents ratings only when a property has more than three guest reviews. As a result, when a property had less than three reviews in a given period, that observation was automatically dropped from the series of analyses presented in this table.
*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$.

6) Alternative Specifications: Logging All Numeric Variables

We performed a log transformation of all numeric variables (security deposit, price, etc.) and reported the results of the estimation of the main DiD regression (Table 2 in the main paper). Table A7 presents the estimation results. As can be seen, the estimated coefficient of the verified photos is consistent with the treatment effect reported in the main paper.

Table A7. Alternative DiD Model Specification: Regressing Property Demand on Verified Photos

Variables	**DiD Model**	
	Estimates	**Robust S.E.**
TREATIND	9.135***	(1.647)
log REVIEW_COUNT$_{t-1}$	9.044***	(0.867)
logNIGHTLY_RATE	−6.182***	(1.145)
INSTANT_BOOK	3.725**	(1.388)
logCLEANING_FEE	1.898***	(0.491)
logMAX_GUESTS	−1.125	(5.412)
logRESPONSE_RATE	5.569***	(1.368)
logRESPONSE_TIME (minutes)	−0.0876	(0.258)
logMINIMUM_STAY	1.480	(1.429)
logSECURITY_DEPOSIT	0.280	(0.318)
SUPER_HOST	3.937**	(1.493)
BUSINESS_READY	1.666	(0.992)

CANCELLATION_STRICT	1.426	(1.298)
HAS_RATING	13.42	(12.26)
HAS_RATING × *COMMUNICATION*	−0.0991	(1.423)
HAS_RATING × *ACCURACY*	0.975	(1.224)
HAS_RATING × *CLEANLINESS*	−1.328	(1.118)
HAS_RATING × *CHECKIN*	−2.237	(1.520)
HAS_RATING × *LOCATION*	−0.666	(1.174)
HAS_RATING × *VALUE*	2.267	(1.174)
Fixed Effect	Property	
Seasonality	City-Year-Month	
Num. Observations	76,901	
R-squared	0.6613	

Note: *$p < 0.05$, **$p < 0.01$, ***$p < 0.001$.

7) Including Pre-treated Units in the Control Group

As a robustness test, we included pre-treated units in the control group (which we excluded from our main analyses for the sake of conceptual clarity, as these pre-treated units had verified photos prior to the start of our observation window). As such, we estimated the change in property demand when the units to be treated (i.e., properties that were untreated prior to our observation window and adopted verified photos during our observation window) used verified photos compared with the pre-treated and untreated units. The logic is that the properties (or hosts) in the pre-treated and to-be-treated groups might be more similar in terms of characteristics in obtaining verified photos. Therefore, including pre-treated units in the comparison group can serve as a falsification check, as the treatment effect should occur only after a property changes from untreated to treated. A similar logic was used in previous work (Manchanda et al. 2015; Narang and Shankar 2019) as a robustness test when self-selection for the treatment was a concern.

Table A8 presents the estimation results. As can be seen, we obtained a consistent estimated treatment effect (the coefficient for *TREATIND* is 7.977 and statistically significant) when we included the pre-treated units in the control group[4].

[4] Note that although that the effect size was 11% lower (compared with the value of 8.985 that we obtained from the main DiD regression), the estimated treatment effect was very close, and it remained at the same level. For example, Narang and Shankar (2019) reported that the estimated effects from their robustness test were 16%–35% lower than their main results.

Table A8. DiD Model: Regressing Property Demand on Verified Photos on Property Demand, Including Pre-treated Properties in the Control Group

Variables	DiD Model	
	Estimates	**Robust S.E.**
TREATIND	7.977***	(1.756)
log REVIEW_COUNT$_{t-1}$	8.082***	(0.882)
NIGHTLY_RATE	−0.126***	(0.0354)
INSTANT_BOOK	4.595***	(1.230)
CLEANING_FEE	0.107***	(0.0237)
MAX_GUESTS	1.385	(1.215)
RESPONSE_RATE	0.0735	(0.0448)
RESPONSE_TIME (minute)	−0.000490	(0.00172)
MINIMUM_STY	0.198	(0.171)
SECURITY_DEPOSIT	0.00313	(0.00223)
SUPER_HOST	3.346*	(1.495)
BUSINESS_READY	0.969	(0.963)
CANCELLATION_STRICT	1.362	(1.305)
HAS_RATING	12.37	(16.05)
HAS_RATING × COMMUNICATION	−0.637	(1.482)
HAS_RATING × ACCURACY	0.712	(1.236)
HAS_RATING × CLEANLINESS	−0.437	(1.299)
HAS_RATING × CHECKIN	−1.162	(1.610)
HAS_RATING × LOCATION	−1.165	(1.376)
HAS_RATING × VALUE	1.632	(1.305)
Fixed Effect	Property	
Seasonality	City-Year-Month	
Observations	123186	
R-squared	0.6611	

Note: The number of observations here is larger than that in the main analyses (76,901). This is because the regression presented in this table was performed on a PSW-matched sample that included pre-treated units in the control group.
*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$

Using Only Pre-treated Units as the Control Group

In Table A8, we present the robustness checks by including the pre-treated units in the control group. In this subsection, we use a stronger control—using the pre-treated units only. Similar to the robustness test in the studies of Manchanda et al. (2015) and Narang and Shankar (2019), this analysis provides a conservative view by comparing the dependent variable among *adopters* only.

Table A9 presents the estimation results. As can be seen, we obtained consistent results (the estimated coefficient of *TREATIND* is 6.77 and significant at the 0.05 significance level) when using only the pre-treated units as the control[5]. This robustness analysis, along with Table A8, suggests that our PSW combined with DiD can mitigate the self-selection issue in the verified photo adoptions.

Table A9 Regressing Property Demand on Verified Photos on Property Demand Using Only Pre-treatment Properties as a Control Group

VARIABLES	**DiD Model**	
	ESTIMATES	**Robust S.E.**
TREATIND	6.769***	(1.011)
log REVIEW_COUNT$_{t-1}$	5.943***	(0.345)
NIGHTLY_RATE	−0.0433***	(0.0102)
INSTANT_BOOK	4.259***	(0.488)
CLEANING_FEE	0.0228*	(0.0107)
MAX_GUESTS	−0.185	(0.297)
RESPONSE_RATE	0.0245	(0.0169)
RESPONSE_TIME (minute)	−0.000291	(0.000650)
MINIMUM_STY	0.00502	(0.0503)
SECURITY_DEPOSIT	−0.0000970	(0.000710)
SUPER_HOST	1.393**	(0.487)
BUSINESS_READY	0.402	(0.354)
CANCELLATION_STRICT	−1.046	(0.844)
HAS_RATING	5.406	(6.542)
HAS_RATING × COMMUNICATION	0.150	(0.617)
HAS_RATING × ACCURACY	−0.118	(0.490)

[5] Note that despite the effect size being 24% lower, the estimated effect was very close and remained at the same level (e.g., Narang and Shankar [2019] reported that the estimated effects from this robustness analysis were 16%–35% lower than their main results across multiple DVs of interest).

HAS_RATING × *CLEANLINESS*	−0.302	(0.424)
HAS_RATING × *CHECKIN*	−0.299	(0.586)
HAS_RATING × *LOCATION*	−0.275	(0.430)
HAS_RATING × *VALUE*	0.709	(0.456)
Fixed Effect	Property	
Seasonality	City-Year-Month	
Observations	51918	
R-squared	0.6814	

Note: The number of observations is different from that in the main analysis (76,901), as this regression was performed on a different sample, in which the 4,932 pre-treated units (i.e., properties that adopted verified photos prior to our observation) formed the control group. The DV is the property monthly occupancy rate.

*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$

8) Additional Analyses

i. Regressing Property Price as the DV

We examined how price changed in the post-treatment period. We replicated our DiD regression but replaced the DV with the property's nightly price to estimate the coefficient of verified photos on price. As shown in Table A10, the insignificant coefficient of *TREATIND* is positive, suggesting an upward change in price. However, the coefficient is insignificant at a significance level of 95%. The results imply that after controlling for the property fixed effects and city-specific seasonality and yearly effects, a property's price does not change the price based on having obtained verified photos.

Note that this result does not mean that a property's price cannot increase or change during the post-treatment period. For example, a property's price may increase when it has accumulated more reviews or the host has gained more experience, or as a response to seasonality effects. Rather, the insignificant coefficient of *TREATIND* in Table A10 should be interpreted as indicating that controlling for all other factors (time-varying property characteristics, city-specific seasonality), a property's price would not increase solely because of the use of verified photos. A possible explanation is that changing photos is not automatically associated with improvements in other aspects of the property and host. Thus, everything else being equal, a host would not increase the price after adopting verified photos. Another explanation is that many Airbnb hosts lack rich information about demand and price, so they rely on Airbnb's smart pricing algorithm to automatically set prices. As a result, it is possible that the price would be higher because of the use of verified photos if the smart pricing algorithm did not include the property pictures as an input. Of course, Airbnb does not disclose the details of its proprietary algorithm. Our understanding is that the algorithm considers many factors, such as

seasonality, market popularity, and review history. In Airbnb's report on its algorithm, property pictures were not included or described as one of the many factors at play in the algorithm.[6]

Table A10. Regressing Property Price on the Treatment Indicator: DiD Model

	Regressing Property Price as the DV	
Variables	**Estimates**	**Robust S.E.**
TREATIND	**4.388**	**(3.047)**
log REVIEW_COUNT$_{t-1}$	−14.29***	(2.245)
INSTANT_BOOK	−6.833**	(2.145)
CLEANING_FEE	0.721***	(0.0514)
MAX_GUESTS	4.898	(2.821)
RESPONSE_RATE	0.241**	(0.0900)
RESPONSE_TIME (minute)	0.000684	(0.00445)
MINIMUM_STY	−0.947**	(0.310)
SECURITY_DEPOSIT	0.0443	(0.0237)
SUPER_HOST	−4.432**	(1.569)
BUSINESS_READY	1.909	(2.344)
CANCELLATION_STRICT	−8.532***	(2.318)
HAS_RATING	−142.2***	(21.38)
HAS_RATING × COMMUNICATION	−3.999	(2.071)
HAS_RATING × ACCURACY	3.298	(2.604)
HAS_RATING × CLEANLINESS	7.547***	(1.775)
HAS_RATING × CHECKIN	3.812	(2.279)
HAS_RATING × LOCATION	11.37***	(1.899)
HAS_RATING × VALUE	−8.340**	(2.677)
Fixed Effect	Property	
Seasonality	City-Year-Month	
Observations	76901	
R-squared	0.8810	

Note: The DiD model was estimated by regressing property price on the same set of variables as the main DiD model, except for instrumented price on the right-hand side, which was excluded. Regression was performed on a dataset including properties that had at least one open day in a month. *$p < 0.05$, **$p < 0.01$, ***$p < 0.001$.

[6] See https://blog.atairbnb.com/smart-pricing/.

ii. Including Review Probability in the PSW and DiD Model

Zhang et al. (2019) found that image quality affects the probability that a customer (after the stay) will write a review for that property. We follow Zhang et al. (2019) and investigate the aspect of review probability as a key variable affecting property demand. To do so, we match properties based on *review probability* to control for this pre-treatment characteristic. In the DiD model, we also control for review probability (the review probability from the previous month). The review probability for a property in each month is computed as the number of received reviews divided by the reservations in that month.

As can be seen in Table A11, the estimated coefficient of *TREATIND* ($b = 7.271$, $p < 0.001$) is consistent with our main results[7]. In addition, the estimated coefficient of *REVIW_PROBABILITY* is insignificant at the 0.05 significance level.

Our interpretation is that although in our main analyses, we did not explicitly include review probability in the PSW and DiD analyses, this variable was indirectly controlled for. This is because review probability can be computed from (as a function of) a set of covariates, including image quality, host response time, price, and other property characteristics (Zhang et al. 2019). These covariates were exactly used in the PSW model and the DiD model. Furthermore, when we thought about how review probability may influence property demand, we found that review probability should not have an impact on the present demand. This is because review probability affects the evolution of *Number of Reviews*, which is a key driver in attracting bookings. Following the arguments and findings of Zhang et al. (2019), review probability affects a property's future demand by affecting the *speed* of accumulating reviews. We believe that review probability alone should not directly affect demand (in fact, Zhang et al. (2019) did not include this variable in their demand model). Although review probability is related to property quality and guest satisfaction, it is unobserved by guests. We can compute review probability because we obtain the # reviews and # bookings in each month. For an average Airbnb guest, however, such information is unknown (a guest observes the total number of reviews but does not know how many bookings a property has received). As a result, guests cannot use review probability as an important variable to aid their booking decisions. It then follows that review probability alone should not have an impact on the present demand for a property.

Table A11 Including Review Writing Probability in Demand Regression

VARIABLES	**DiD Model**	
	ESTIMATES	**Robust S.E.**

[7] Note that this regression is estimated on those observations in which *review probability* can be defined and computed, that is, the observations when a property in a month had at least one booking. By including review probability in the model, we automatically excluded those observations with zero booking (*demand* is 0) in a month, which is more likely for untreated properties than for treated properties. As a result, the coefficient of *TREATIND* in this analysis is a conservative estimate.

TREATIND	7.271***	(2.022)
log REVIEW_COUNT$_{t-1}$	7.362***	(1.249)
NIGHTLY_RATE	−0.149***	(0.0374)
INSTANT_BOOK	5.775***	(1.501)
CLEANING_FEE	0.0779**	(0.0279)
MAX_GUESTS	−0.312	(1.263)
RESPONSE_RATE	0.120	(0.0670)
RESPONSE_TIME (minute)	−0.00152	(0.00245)
MINIMUM_STY	0.227	(0.158)
SECURITY_DEPOSIT	0.00486	(0.00261)
SUPER_HOST	2.094	(1.576)
BUSINESS_READY	0.503	(1.171)
CANCELLATION_STRICT	0.666	(1.557)
HAS_RATING	21.64	(20.22)
HAS_RATING × COMMUNICATION	−1.632	(1.842)
HAS_RATING × ACCURACY	−0.430	(1.588)
HAS_RATING × CLEANLINESS	−0.865	(1.336)
HAS_RATING × CHECKIN	0.885	(1.807)
HAS_RATING × LOCATION	−1.783	(1.620)
HAS_RATING × VALUE	1.932	(1.292)
REVIW_PROBABILITY$_{t-1}$	2.06	(1.507)
Fixed Effect	Property	
Seasonality	City-Year-Month	
Observations	40590	
R-squared	0.5797	

Note: The DV is the property monthly demand. The number of observations in this regression is different from that in the main analysis (76,901). This is because this regression includes review probability in the previous month as a control variable. Therefore, the model was regressed on observations in which a property had at least one booking in the previous month (other: in addition to this, the property had at least one open day in the current month).

*$p < 0.05$, **$p < 0.01$, ***$p < 0.001$

9) Testing Host Behavior – Changes in the Number of Open Days

One self-selection concern is that hosts may change their behavior (in a way that will increase the property booking) along with the treatment adoption. To test this, we used the number of open days to serve as a falsification test for endogeneity. Note that the treatment and control properties differ in the observed variables (the treatment is endogenous in our setting). To address this, we performed a propensity score-based adjustment to make the treatment and control groups comparable on the observed measures. If the propensity score-weighted treatment and the control properties were to exhibit a different number of nights open subsequent to the treatment, then it would indicate that part of the treatment may be driven by hosts' desire to open for more nights.

To check this, we produced the following results. Here, we regressed open nights as a function of the treatment indicator and other controls (the dependent variable was log (#open nights +1)). We found that the treatment had no impact on the number of open nights. Hosts' adoption of treatment did not appear to be driven by the intent to open the property for more days in the future. As shown in Table A12, the insignificant coefficients across the two model specifications suggest that the increase in demand was not introduced by hosts being more likely to open nights after adopting Airbnb professional photos. Note that in the model specification in column (2), we added the number of reservation days in the previous period as an additional control.

Table A12 Robustness Test: Changes in # of Open Days along with Verified Photo Adoption

	DV: log # Open Days in Period *t*	
VARIABLES	**Model (1)**	**Model (2)**
TREATIND	0.0201	−0.0267
	(0.0339)	(0.0317)
log # RESERVATION_DAYS$_{t-1}$		0.194***
		(0.00831)
log REVIEW_COUNT$_{t-1}$	−0.00463	−0.0650***
	(0.0181)	(0.0176)
NIGHTLY_RATE	0.00140*	0.00184***
	(0.000554)	(0.000517)
INSTANT_BOOK	0.0431	0.01000
	(0.0226)	(0.0213)
CLEANING_FEE	−0.000765*	−0.00103**

	(0.000359)	(0.000346)
MAX_GUESTS	0.0112	0.00981
	(0.0232)	(0.0204)
RESPONSE_RATE	0.000359	0.000376
	(0.00106)	(0.000983)
RESPONSE_TIME (minutes)	−0.0000379	−0.0000210
	(0.0000331)	(0.0000307)
MINIMUM_STAY	−0.00824**	−0.00884**
	(0.00296)	(0.00269)
SECURITY_DEPOSIT	−0.0000389	−0.0000309
	(0.0000398)	(0.0000369)
SUPER_HOST	−0.0588	−0.0759*
	(0.0342)	(0.0297)
BUSINESS_READY	0.00305	−0.00470
	(0.0199)	(0.0177)
CANCELLATION_STRICT	−0.0156	−0.0277
	(0.0270)	(0.0254)
HAS_RATING	−0.157	−0.255
	(0.239)	(0.219)
HAS_RATING × COMMUNICATION	−0.114***	−0.100***
	(0.0304)	(0.0276)
HAS_RATING × ACCURACY	0.0581*	0.0439
	(0.0267)	(0.0257)
HAS_RATING × CLEANLINESS	−0.000146	0.0128
	(0.0224)	(0.0207)
HAS_RATING × CHECKIN	0.0426	0.0531
	(0.0317)	(0.0280)
HAS_RATING × LOCATION	0.0242	0.0230
	(0.0243)	(0.0221)
HAS_RATING × VALUE	−0.00595	−0.0193
	(0.0233)	(0.0214)

INTERCEPT	2.913***	2.729***
	(0.144)	(0.134)
Fixed Effect	Property	Property
Seasonality	City-Year-Month	City-Year-Month
Observations	76,901	76,901
R-squared	0.4444	0.5092

Note: The difference between the two specifications is that in Model (2), we control for the number of reservation days in the previous period; robust standard errors are in parentheses; *$p < 0.05$, **$p < 0.01$, ***$p < 0.001$.

VI. Data Description and Sample Construction

We started with over 13,000 unique Airbnb properties in seven US cities, collecting images from the property pages in each month from January 2016 to April 2017. Of the full sample, 4,932 properties were pre-treated (i.e., had verified photos prior to our observation in January 2016). For the analyses presented in this study, we removed these properties from our sample for the sake of conceptual clarity (in an additional robustness test, we included these units in the control group and obtained consistent results; see Section V.7) in this Appendix for detailed results and discussions). The remaining properties totaled 8,858 (13,790 – 4,932). As described in the main paper, we removed missing data points and properties for which there were web page errors during data collection. Errors may have occurred because Airbnb blocked our data collection requests, and/or hosts unlisted their properties from Airbnb temporarily or permanently.

Next, using the valid units, we performed a propensity score analysis to match the treated and untreated units by comparison (matching) based on a set of observed covariates. Of the 7,825 valid properties identified during the observational window, there were 231 treated units and 7,594 untreated units before PSW. Next, we performed a propensity score estimation to find properties in the two groups that were close or matched with each other in terms of the property and host characteristics in the pre-treatment period. We first estimated the propensity score as a logit function of a set of property, host, and neighborhood covariates measured at the start of the first period (i.e., the pre-treatment period). The set of variables used in the propensity score estimation included time-invariant characteristics (e.g., number of bedrooms, property type, property amenities) and time-varying dynamic characteristics (e.g., property price, number of reviews).

We first denoted the set of variables for the units to be matched on as *X*. Following prior literature that applies the propensity score approach, we computed the propensity score as a logit function of *X*. Unit *i* has an estimated propensity score of $ps_i = f(X_i)$. Then, as we described in the paper, we computed a weight for each unit based on its estimated propensity score. Specifically, we calculated weight as $w_i = \frac{1}{ps_i}$ if *i* was a treated unit and as $w_i = \frac{1}{1-ps_i}$ if *i* was an untreated unit. As we described in the paper, this is a widely used PSW approach called inverse probability treatment weighting. Later, we used the sample weights for DiD regressions. Furthermore, to address the concern that a very low estimated probability can result in extremely large (and possibly unstable) weights, we followed common practice in PSW and used trimmed weights, excluding those that were outside 1% and 99% of the distribution (Austin and Stuart 2015; Cole and Hernan 2008). The PSW generated a sample that included 7,211 of the 7,594 untreated units and 212 of the 231 treated units. This

matched sample was used in the main regressions (Tables 2, 3, 4, and 7 in the main paper). The total number of observations used in the analyses was 76,901[8].

[8] Note that the total observations in the regressions was fewer than # units * # periods because for properties that did not have open days in a month, the dependent variable—property demand (occupancy rate)—would be indefinable.

VII. Example of Property Images with 1 SD of Improvement in Key Image Features

As an example, we show what an increase of 1 SD may look like visually in Figure A1. The figure shows the original photos and a 1 SD increase for two example attributes—brightness and saturation.

Figure A1 Examples of Images with a 1 SD Increase in Image Attributes

Attribute	Original Photo	+1 *SD* in Attribute
Brightness		
Saturation