

## 1. ABSTRACT

Airbnb is an online marketplace for hosts to list their properties so that guests can rent to stay for lodging and tourism activities. Even though Airbnb provides general guidance currently there is no free service that will help owners to set the prices. Airbnb hosts are facing difficulty to set the optimal price for their listings. Our goal is to develop a machine learning model that will help hosts to set the prices for their properties using the pricing data for other listings in San Francisco.

## 2. DATA

The dataset for this project is downloaded from [Inside Airbnb website](#). Two csv files are used in the analysis. *listings\_detailed.csv* contains detailed information about Airbnb listings in San Francisco and it is scraped on 03-02-2021. The dataset contains 6883 rows with 73 attributes and loaded as df. *calendar.csv* contains the pricing information at the date of scraping for each listing for the next year. The dataset contains 2512295 rows with 7 attributes and loaded as calendar. The definition of each attributes that is used in analysis can be found in **Appendix** section. The main limitation of the dataset is it lacks the actual price that a guest pays for a listing. The pricing information is the list price and it could be different than the actual paid amount. In addition, the dataset does not contain other fees such as cleaning fee which could vary largely and can make a big difference in the final price paid by the guest.

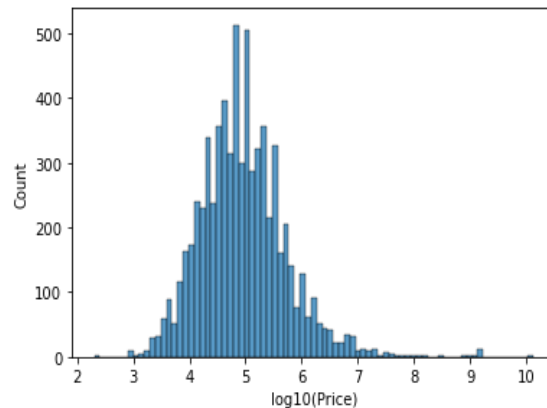
### 2.1 Data Wrangling:

The first step of the data wrangling section is dropping of the attributes that has too many missing values (>50%) and that contain useless information (URL's for host, description for neighborhood or listing etc.) for the machine learning model. Following this initial step, each attribute is investigated one by one and necessary data wrangling steps are applied. These steps can be found in detail in the data wrangling [notebook](#).

## 2.2 Exploratory Data Analysis:

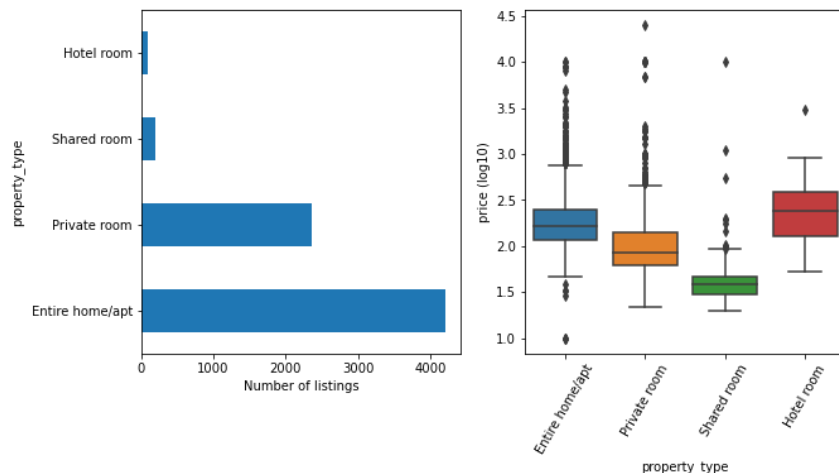
Exploratory data analysis is critical for the any data science work since helps data scientist to look at the data before making any assumptions. For this purpose, I plotted the distribution of target and target's relation with various attributes.

```
count    6882.000000
mean     219.870096
std      603.523080
min       10.000000
2.5%     38.000000
25%      85.000000
50%     135.000000
75%     219.000000
97.5%    799.000000
max     25000.000000
Name: price, dtype: float64
```



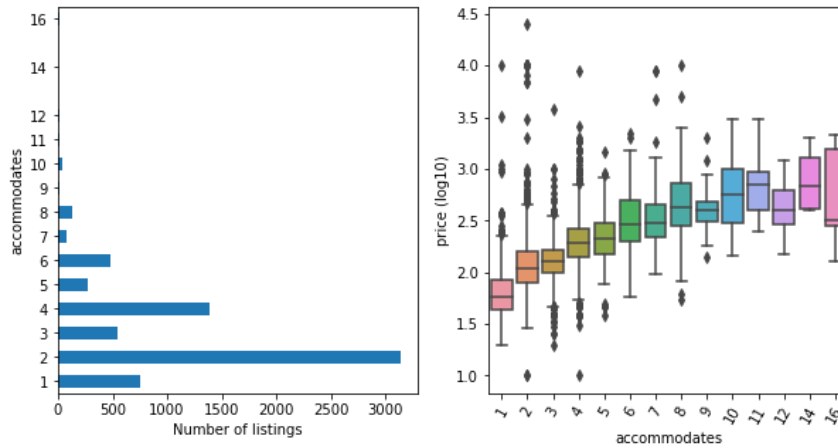
**Figure 1:** (a) Price distributions of Airbnb listings in San Francisco (b) Summary of Airbnb listings price.

Descriptive statistics in Figure 1a revealed price data has extremely low and high values which can cause problems for training of the machine learning model. Price data was log transformed for the visualization purposes (Figure 1b). These listings with very high or low price can be either a rare incidence or data entry error. Based on the initial performance of the model, these extreme values can be filtered.



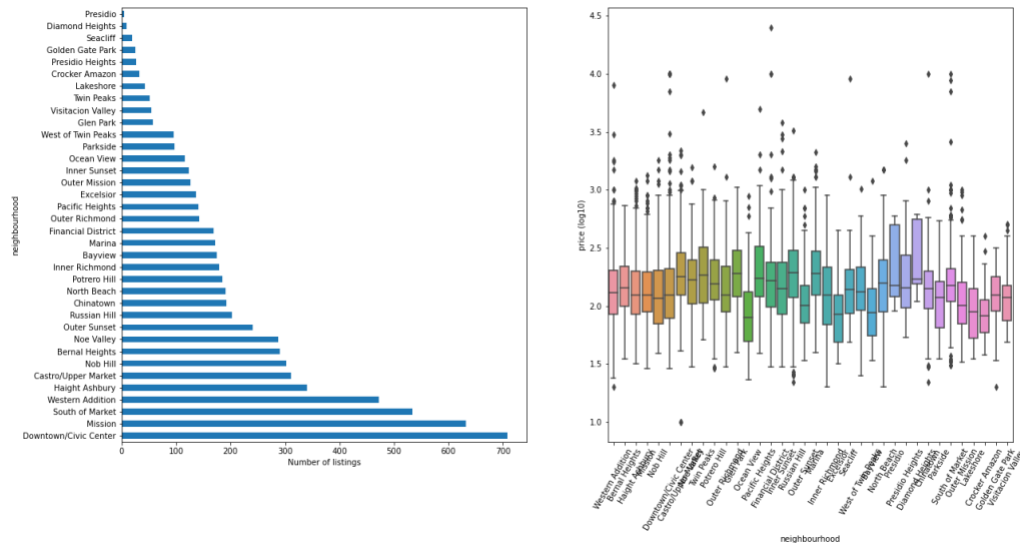
**Figure 2:** Number of listings vs property type (left), Price (log10) vs property type

Majority of the listings in San Francisco are 'entire home or apartments' and the median price of these listings are more than private or shared rooms.



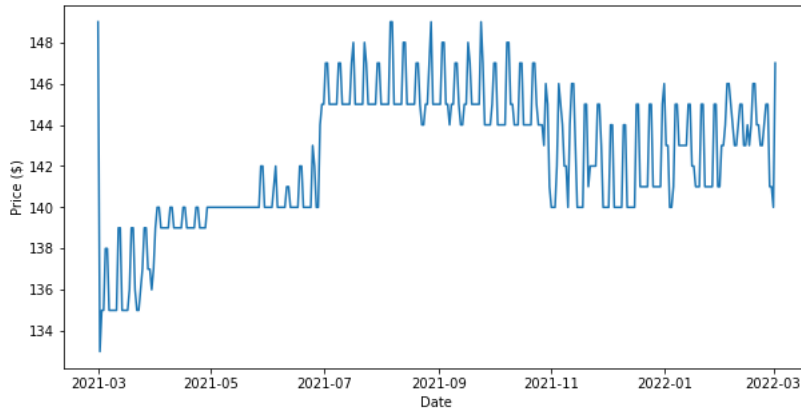
**Figure 3:** Number of listings vs accommodations (left), Price (log10) vs accommodations

The price of listings increases as the number of people it can accommodate. Majority of the listings can accommodate 3 or less guests.



**Figure 4:** Number of listings vs neighbourhood (left), Price (log10) vs neighbourhood

Prices are similar among the neighborhoods in the San Francisco. Downtown/Civic Center has the most of the listings.



**Figure 5:** The median of listed prices between 03-02-2021 and 03-02-2021.

The listing prices have increased in summer months and show spikes that correspond to weekends.

### 2.3 Preprocessing of the Data:

Several attributes have been binned to create categorical variables. All of the categorical variables have been converted to dummy variables. Details can be found in data wrangling [notebook](#). Tree-based models have been used as machine learning models. These models do not require standardization or normalization, however, since linear regression is used as base model both standardization and normalization are done to the data. MinMaxScaler is used as normalization method since it is applied to whole data frame and the data frame consists dummy variables with 0 and 1 entries. Details can be found in model [notebook](#).

## 3. MODELING and RESULTS

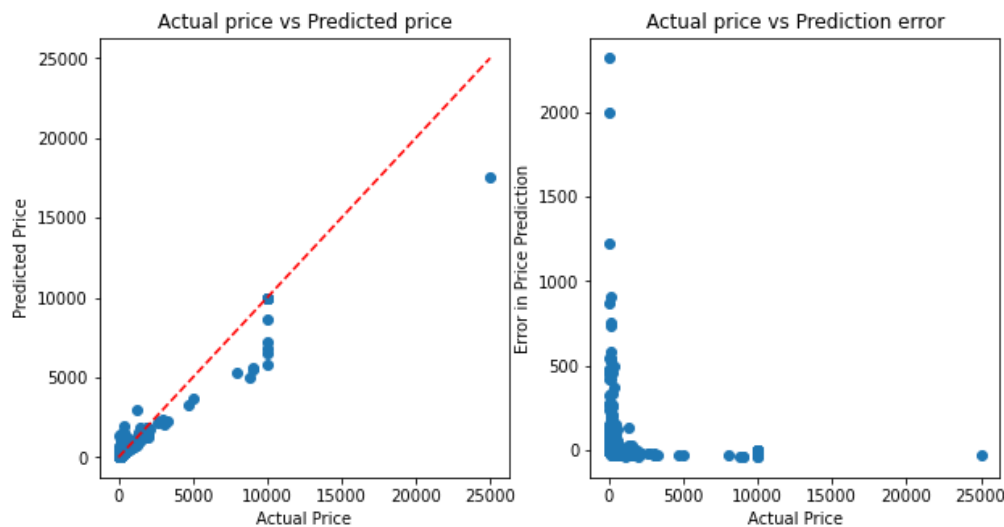
Three models are used for price prediction:

1. Linear Regressor
2. Random Forest Regressor
3. XGBoost Regressor

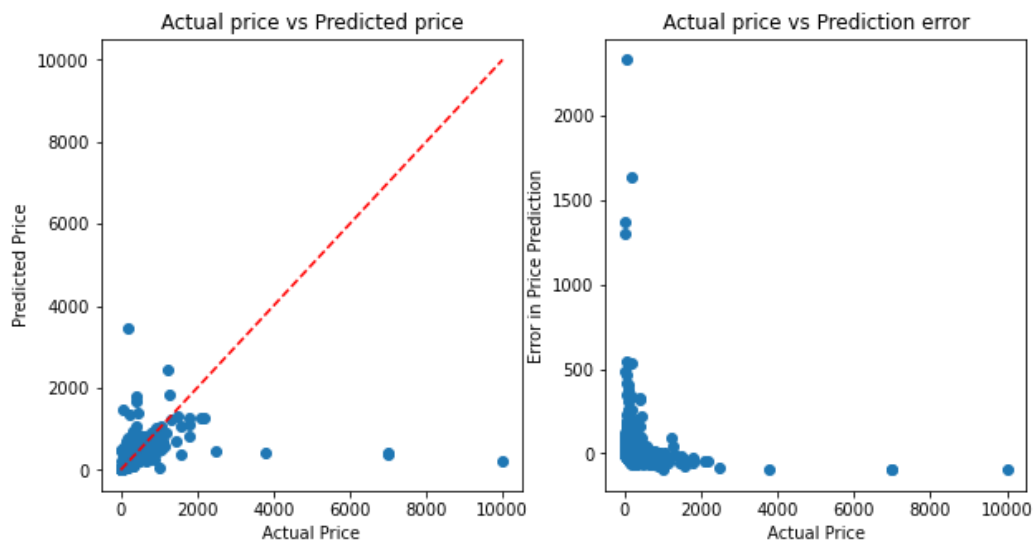
<i>Model</i>	<i>Training Data</i>	<i>Test Data</i>
<i>Linear Regressor</i>	0.11	0.02
<i>XGBoost Regressor</i>	0.99	-0.02
<i>Random Forest Regressor</i>	0.92	0.11

**Table 1:** R-squared scores for 3 different models.

Initial results (Table 1) reveal that all three models have overfit to the training data and perform poorly on the test data. Looking deeper into model performance using 5-fold cross-validation revealed that some folds have really low R-squared values (negative values, data not shown) and others having reasonable R-squared values (0.5-0.7, data not shown). Plotting the Actual Price vs Predicted Price revealed a trend where high price listings tend to be predicted lower and low-price listings tend to be predicted higher (Figure 5). A new variable, % error, is created using actual and predicted price ( $\% \text{ error} = 100 * (\text{predicted price} - \text{actual price}) / (\text{actual price})$ ). Actual price vs % error revealed that, model performs poorly on the low-price range for both training data (Figure 6) and test data (Figure 7).



**Figure 6:** Actual price vs predicted price (left), Error in Price Prediction vs Actual Price (right) for the **training data**.



**Figure 7:** Actual price vs predicted price (left), Error in Price Prediction vs Actual Price (right) for the **test data**.

In the EDA section, descriptive statistics of the 'price' has revealed that some listings had very low price. To further investigate this, price\_per\_accommodate, variable is created. Descriptive statistics of this variable makes it clear that some listings have really low prices which seems as a data entry error.

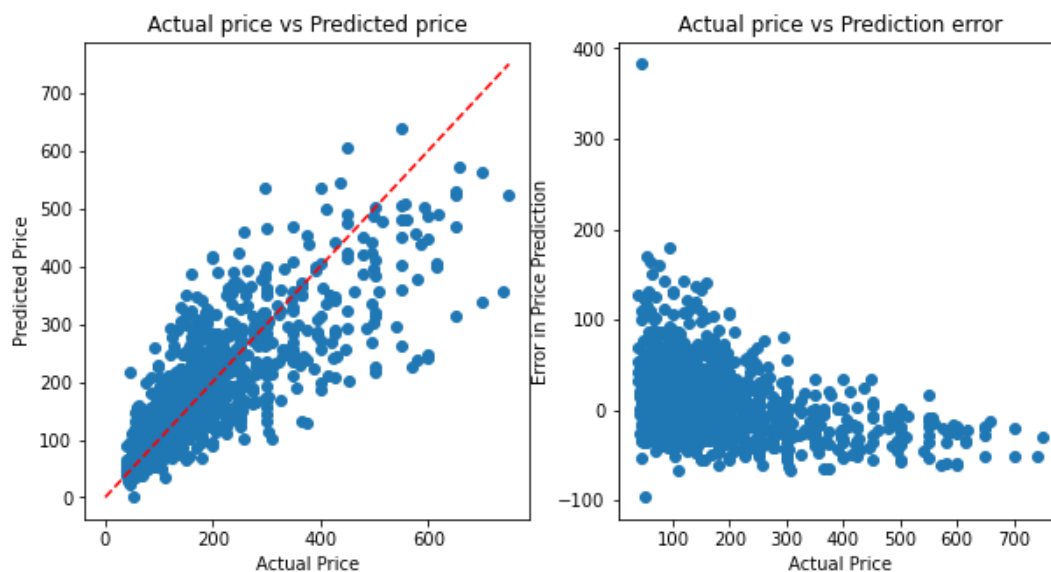
```
count      6882.000000
mean       77.494464
std        289.329700
min         2.500000
2.5%       19.430357
25%        37.250000
50%        50.000000
75%        75.000000
97.5%     187.491667
max       12500.000000
Name: price_per_accommodate, dtype: float64
```

The outliers (lower 2.5% and upper 2.5%) are filtered out and models are trained with the filtered dataset.

<i>Model</i>	<i>Training Data</i>	<i>Test Data</i>
<i>Linear Regressor</i>	0.53	0.53
<i>XGBoost Regressor</i>	0.94	0.66
<i>Random Forest Regressor</i>	0.95	0.65

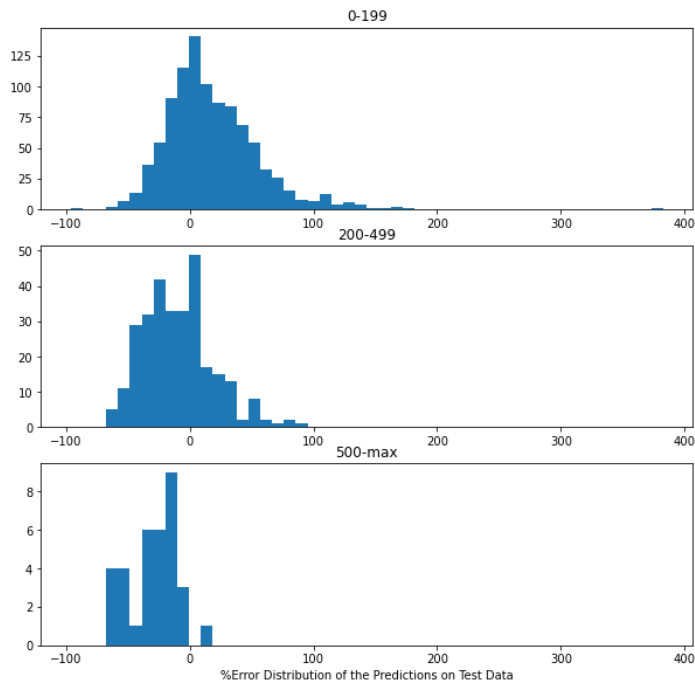
**Table 2:** R-squared scores for 3 different models following removal of outliers.

Removal of the outliers significantly improved the model performance. Hyperparameter tuning is performed for XGBoost Regressor using GridSearchCV. R-squared value for the training data is 0.85 and for the test data is 0.67.



**Figure 8:** Actual price vs predicted price (left), Error in Price Prediction vs Actual Price (right) for the **test data** following removal of outliers and hyperparameter tuning.

Listings with low prices are predicted with large errors even after removal of outliers and hyperparameter tuning. One way to improve the model performance could be dividing the dataset based on price and train each dataset separately. % error distribution for various price ranges are plotted.



**Figure 9:** %Error distribution on price prediction with respect to price range (\$0-199, \$200-499, \$500-max)

The distribution of %error reveal that, %error is mostly positive for prices lower than \$200 and it is negative for prices higher than \$200. The dataset is divided into two based on listing price and hyperparameter tuning was done for XGBoost Regressor model on both datasets.

<i>Dataset</i>	<i>Training Data</i>	<i>Test Data</i>
<i>Price more than \$200</i>	0.78	0.29
<i>Price less than \$200</i>	0.88	0.65
<i>combined</i>	0.85	0.67

**Table 3:** R-squared scores for XGBoost Regressor for 3 different models following hyperparameter tuning.

The likely reason why model has performed poorly on dataset with price more than \$200 is the small size of the dataset. This problem could be overcome by using a larger dataset.

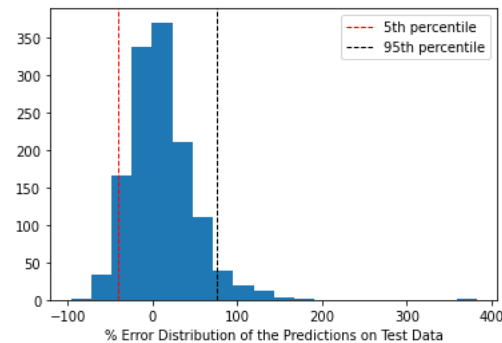
## 4. SUMMARY

The R-squared value of the XGBoost Regressor model for Airbnb price prediction is 0.66, meaning that the model can explain 66% of the variance of the Airbnb listing price.

Two new variables are created to evaluate the model performance:

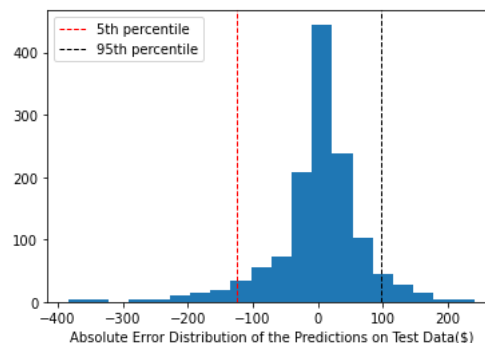
- $\% \text{ Error} = 100 * (\text{predicted price} - \text{actual price}) / \text{actual price}$
- $\text{Absolute Error} = \text{predicted price} - \text{actual price}$

```
count    1305.000000
mean      10.567670
std       38.090377
min       -96.417501
25%      -13.935114
50%       4.046777
75%      30.970241
max      382.739766
Name: 0, dtype: float64
```



**Figure 10:** Descriptive statistics of the % Error (left), distribution of %Error (right)

```
count    1305.000000
mean       0.732301
std       69.274361
min      -384.110199
25%      -19.509892
50%       5.044113
75%      32.347427
max      240.521301
Name: 0, dtype: float64
```



**Figure 11:** Descriptive statistics of the Absolute Error (left), distribution of Absolute Error (right)

To assess the business value of the model, two scenarios are created

### 4.1 Scenario One:

The model utilizes more than 100 attributes to predict the Airbnb listing price. A new host that does not have a data analytics or market research team would tend to list their unit based on 3-4 attributes of the unit: location (compares to listings in close proximity), number of bedrooms, number of beds, number of people a listing can accommodate. We decided to create a new data frame with these attributes and train our model (limited model) and compare the R-squared value and mean



absolute error between limited model and our model using same algorithm (XGBoost Regressor).

<i>Dataset</i>	<i>R-squared</i>	<i>Mean Absolute Error (\$)</i>
<i>Original</i>	0.66	26.9
<i>Limited</i>	0.34	50.8

**Table 4:** R-squared value and Mean Absolute Error (\$) for XGBoost Regressor for 2 different datasets (Original, Limited, contains 3 attributes: bedrooms, beds, accommodate)

Apparently, our model has better performance than the limited model which used a fraction of attributes that an average individual would focus. There is a second level that needs addressing. The dataset that is used in this project is scraped on 2020-03-02 and contains all the listings in San Francisco. Are the prices of the listings fair, in other terms what is the quality of the data?

#### 4.2 Scenario Two:

I hypothesize that there are 3 main types of hosts:

1. Companies/professional: These hosts follow the market trend and optimize the price of the listing to maximize revenue and occupancy.
2. Residents of SF: These hosts live in SF and are willing to list their residence for only above market price or high price so that they could use the revenue for travelling somewhere else.
3. New/Inexperienced: These host do not experience in Airbnb listings, could be listing their unit for low prices which provide high occupancy or high prices which cause low occupancy.

My assumption is to compare the model performance (R-squared, mean absolute error) when model is trained on listings of companies/professionals that has multiple listings (more than 20) vs the remaining listings. The dataset is gone through same cleaning and wrangling and right before model training is filtered based on hosts that have more than 20 listings (>20). Model is trained on both datasets (hosts that have more than 20 listings (>20), hosts that have less than 20 listings (<20) and predictions and evaluations are performed.

<i>Dataset</i>	<i>R-squared</i>	<i>Mean Absolute Error (\$)</i>
<i>Original</i>	0.66	26.9
<i>&gt;20</i>	0.90	6.9
<i>&lt;20</i>	0.63	31.8

**Table 5:** R-squared value and Mean Absolute Error (\$) for XGBoost Regressor for 3 different datasets (Original, >20: listings of hosts that have 20 or more listing, <20: listings of hosts that have 20 or less listing)

The model performed significantly better on the dataset that contains listings of hosts that have 20 or more listings. To assess whether, size of the dataset has any impact on this performance, a new dataset is created by randomly selecting same number of listings with >20 dataset. Model is trained on both datasets predictions and evaluations are performed.

<i>Dataset</i>	<i>R-squared</i>	<i>Mean Absolute Error (\$)</i>
<i>&gt;20</i>	0.90	6.9
<i>random</i>	0.50	34.1

**Table 6:** R-squared value and Mean Absolute Error (\$) for XGBoost Regressor for 2 different datasets (>20: listings of hosts that have 20 or more listing, random: random sampling of the original dataset with same number of listings with >20 dataset)

Our model reduces the mean absolute error of the price assigned by inexperienced hosts from \$50 to \$27. In addition, when the model is trained on data that has better quality, the mean absolute error for price prediction decreases down to \$7.

## 5. APPENDIX

Definitions of the attributes of the listing data frame that are used in the model are listed below.

host\_id: Id of the host  
host\_since: registration date of the host  
host\_response\_time: Response time of the host  
host\_response\_rate: Response rate of the host  
host\_acceptance\_rate: Acceptance rate by the host  
host\_is\_superhost: Is host a super host  
host\_listings\_count: Number of listings that the host owns  
host\_total\_listings\_count: Number of listings that the host owns  
host\_verifications: Methods of verifications that host has done  
host\_has\_profile\_pic: Host has profile picture or not  
host\_identity\_verified: Host has verified identity or not  
neighbourhood: neighborhood information  
neighbourhood\_cleansed: neighborhood information  
neighbourhood\_group\_cleansed: neighborhood information  
latitude: latitude of the listing  
longitude: longitude of the listing  
property\_type: sub-groups of room\_type column  
room\_type: entire home, private room, shared room or hotel  
accommodates: how many guests can a listing accommodate  
bathrooms\_text: number of bathrooms  
bedrooms: number of bedrooms  
beds: number of beds  
amenities: amenities in the listing  
price: price of the listing at the time of scraping  
minimum\_nights: minimum nights a listing available  
maximum\_nights: maximum nights a listing available  
has\_availability: listing is available or not at the time of the scraping  
availability\_30: availability in next 30 days  
availability\_60: availability in next 60 days  
availability\_90: availability in next 90 days  
availability\_365: availability in next 365 days  
number\_of\_reviews  
number\_of\_reviews\_ltm: number of reviews lifetime  
number\_of\_reviews\_l30d: number of reviews last 30 days  
first\_review: date of first review  
last\_review: date of last review  
review\_scores\_rating  
review\_scores\_accuracy  
review\_scores\_cleanliness  
review\_scores\_checkin  
review\_scores\_communication  
review\_scores\_location  
review\_scores\_value  
instant\_bookable: instant bookable or not  
calculated\_host\_listings\_count: Number of listings that the host owns  
calculated\_host\_listings\_count\_entire\_homes: Number of entire homes that the host owns  
calculated\_host\_listings\_count\_private\_rooms: Number of private rooms that the host owns  
calculated\_host\_listings\_count\_shared\_rooms: Number of shared rooms that the host owns  
reviews\_per\_month

Definitions of the attributes of the calendar data frame that are used in the model are listed below.

listing\_id: id of the listing

date: date of the pricing

available: listing available or not

price: price of the listing

adjusted\_price: price of the listing

minimum\_nights: minimum nights a listing available

maximum\_nights: maximum nights a listing available