U.S. Major Cities' Airbnb Listing Price Prediction

Yiyuan Cao 116020006 Jingxin Jiang 117020119 Xuanyi Wang 117010253
Jie Xi 117020302

April 22, 2020

Abstract

In this report, price predictors and suggestions are given to help hosts and guests to determine the base price of the property. After processing and splitting the data, some linear and nonlinear models have been tested and judged by the RMSE performance. The final chosen model is the XGBoost model, which gives the smallest test RMSE. Scalable heatmaps are used to compare the real price and the prediction results. Feature importance graphs show the primary contributors to the price are the property-related and location-related factors. Further requisite explanations of the rank is also included.

Contents

1. Introduction

Airbnb is an online home-sharing platform that connects people who want to rent out their properties with people who are looking for accommodations in that location. Home-owners (hosts) will put their properties (listings) online, so that renters (guests) can pay to stay in them.

Hosts are expected to price their own listings. Although Airbnb has provided some general guidance, there is no free service available that can help hosts to price their property properly according to their locations and conditions.

Third-party paid pricing tools are available, such as PriceLabs and WheelHouse, but still the hosts are required to input their expected nightly price, also known as base price, and then the algorithm will vary nightly price around the base price for each single night to generate a sequence of prices, depending on the seasonality, the day of the week and some other factors.

As a host, a natural and fundamental question to ask is how to determine the base price of the property. It is important to get the base price right, particularly in the major cities where there is a lot of price competition that even small differences in the prices can make a huge difference. It is also a challenging task to do correctly - price too high and no one will book, price too low and you will be missing out of a lot of potential income.

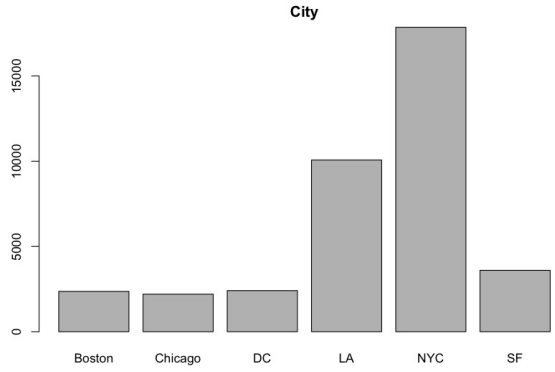
This project aims to solve this problem. By building a model that predicts the base price of the property based on the detailed information of the property, we can generate a tailored price for each property based on their competitiveness on the rental market. Such a model provides a powerful guidance for hosts' reference and also help the guests better compare the listing prices based on their demands. As a result, the properties on the rental market can be fairly priced, and the market efficiency will be improved, making both the hosts and the guests better off.

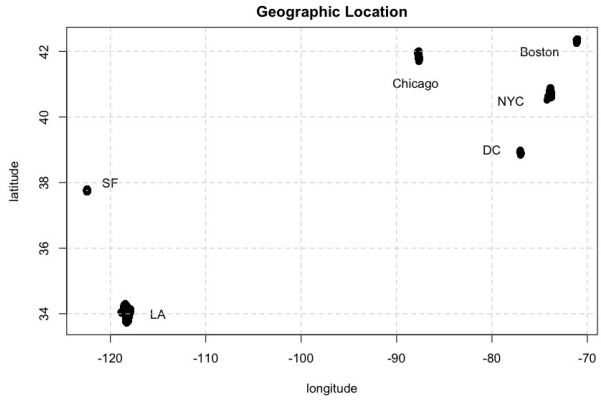
2.1 Data

2.1.1 Data description

This data set is the public Airbnb listing in the U.S.major cities downladed from Kaggle. The data includes 73773 samples and 29 features, including our main research object log_price and the remained predictors which can be divided into 4 parts as location, property, host and review according to their meanings.

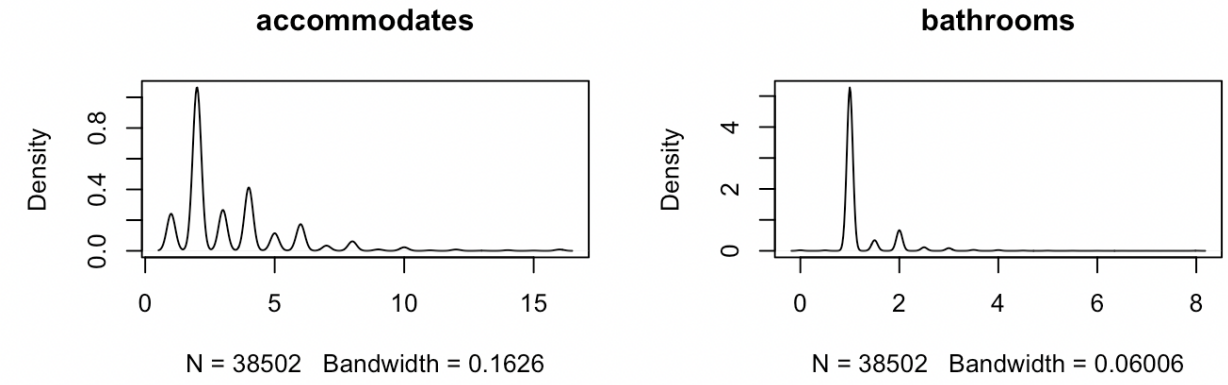
The location variables consists of the geographic location like the longitude and latitude, and administrative divisions like city, zip code and neighborhood.

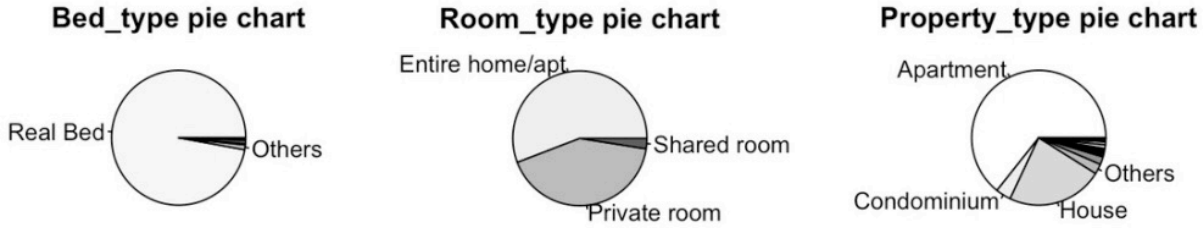




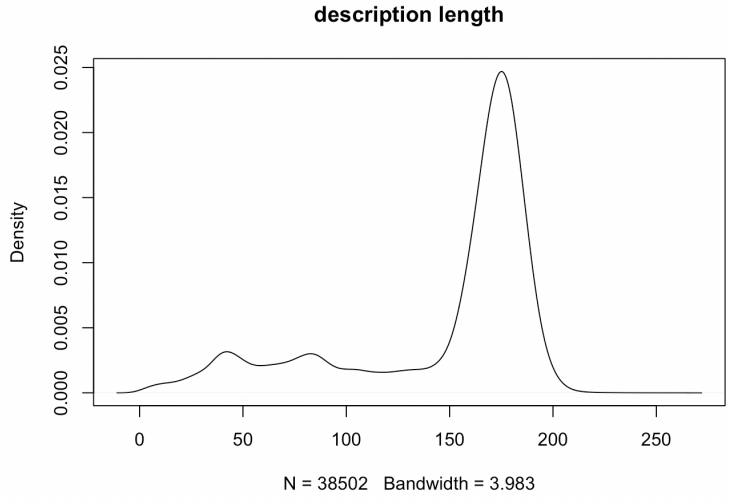
For property, it contains bathrooms, bedrooms, accomodates, bed type, property type, amenities, and room type. It can be divided into dummy variables like the property type (whether it's an apartment or a house) and numeric variables like the number of bedrooms. The amenities is the list of the name of the furnitures and electrical appliances in the house. The bathrooms, bedrooms indicate the number of corresponding

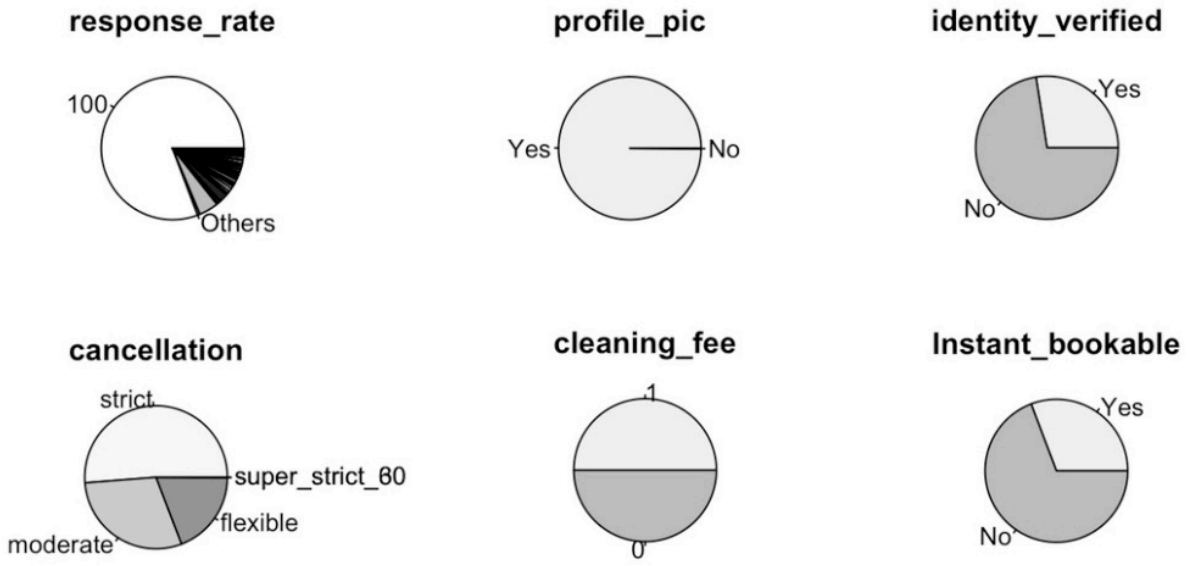
rooms, while the accomodates is the number of guests the house can live. And the room type is to indicate that whether the guest needs to share the house with the host or not.



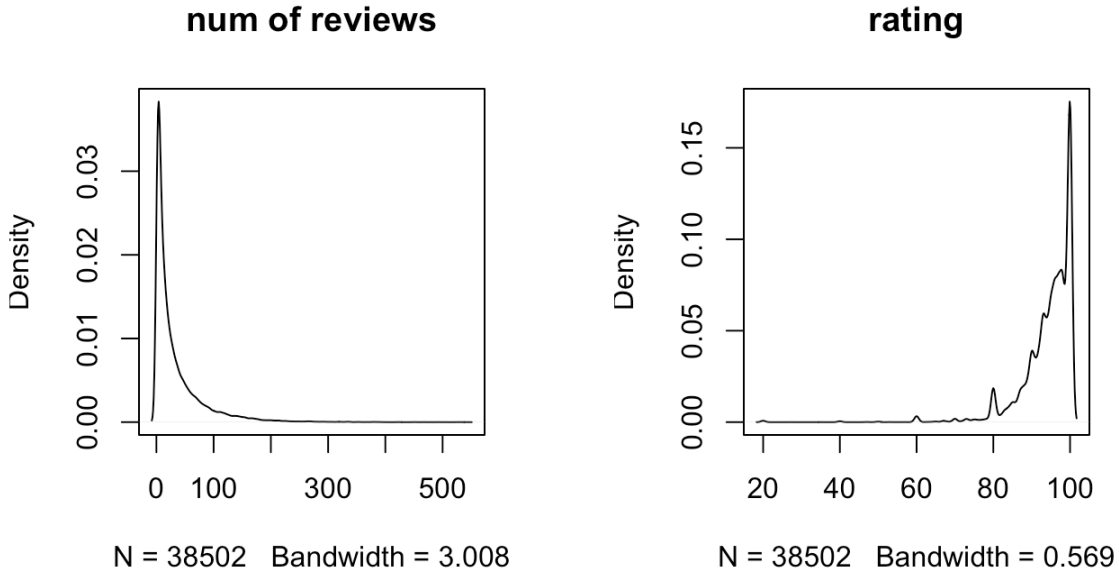


For host, it has host_since,host_response_rate, has_profile_pic, identity_verified, Instant bookable, Cancellation policy, has_cleaning fee and Description, which are related to the host and the listing information on the Airbnb website. And the description of the property is a short paragraph about the situation of the listing. The host_since is the starting operating time of the listing. The host response rate is the rate that the host response to guest's questions in the app. And for the host, the profile picture and the identity variables show the information of the host. The instant bookable, cancellation and cleaning fee policy are the setting of the host, which contains mainly the degree of the policy and yes or no information.





The last type is the review data from guests like the number of reviews and the review scores rating. Besides, there are also two datetime variables-first_review and last_review, which are the time of the first review and last review.



2.1.2 Data cleaning and transformation

In the data processing, we first omit NAs, and then convert the type of the data.Then we process the text data as well as the time data. For the text data, for the amenitiesis - the list of the name of the furnitures and electrical appliances in the house which is the set form, it was converted to a list of dummy variables using one-hot encoding. For another short paragraph-the description, it is represented by the length of the description paragraph, indicating that the longer the paragraph, the deeper the influence of it to the price. Besides the text data,there are two datetime variables-first_review and last_review which are the time of the first review and last review, are converted into corresponding time length.

During our analysis, we can get the relative importance of those 4 different parts of variables, and answering

the question that how can our hosts and guests utilize this information for their benefit.

```
train = df

train['amenities_n'] = train.apply(lambda x : len(x['amenities'].split(",")), axis = 1)
train['description_n'] = train.apply(lambda x : len(x['description'].split(" ")), axis = 1)
train['name_n'] = train.apply(lambda x:len(x['name'].split(" ")), axis = 1)
train['first_review_sum'] = train.apply(lambda x : datetime.strptime(x['first_review'],"%Y-%m-%d"), axis
train['last_review_sum'] = train.apply(lambda x : datetime.strptime(x['last_review'],"%Y-%m-%d"), axis =
train['host_since_sum'] = train.apply(lambda x : datetime.strptime(x['host_since'],"%Y-%m-%d"), axis = 1
train['time_reduce'] = ((train['last_review_sum'] -  train['first_review_sum']) / timedelta(days = 1)).a
train['first_review_sum'] = train['first_review_sum'] - min(train['first_review_sum'])
train['first_review_day'] = (train['first_review_sum'] / timedelta(days = 1)) / (12 * 30)
train['last_review_sum'] = train['last_review_sum'] - min(train['last_review_sum'])
train['last_review_day'] = (train['last_review_sum'] / timedelta(days = 1)) / (12 * 30)
train['host_since_sum'] = train['host_since_sum'] - min(train['host_since_sum'])
train['host_since_day'] = (train['host_since_sum'] / timedelta(days = 1)) / (12 * 30)
train['host_response_rate'] = train.apply(lambda x : int(str(x['host_response_rate'])[:-1]), axis = 1)
train['cleaning_fee']  = train['cleaning_fee'].astype(int)
train = train.drop(['first_review_sum','last_review_sum','host_since_sum','amenities','description','fi



train_property_dum = pd.get_dummies(train.property_type)

train_room_dum = pd.get_dummies(train.room_type)

train_bed_dum = pd.get_dummies(train.bed_type)

train_cancel_dum = pd.get_dummies(train.cancellation_policy)

train_profile_dum = pd.get_dummies(train.host_has_profile_pic)

train_neighbor_dum = pd.get_dummies(train.neighbourhood)

train_verified_dum = pd.get_dummies(train.host_identity_verified)

train_bookable_dum = pd.get_dummies(train.instant_bookable)
```

2.2 Models

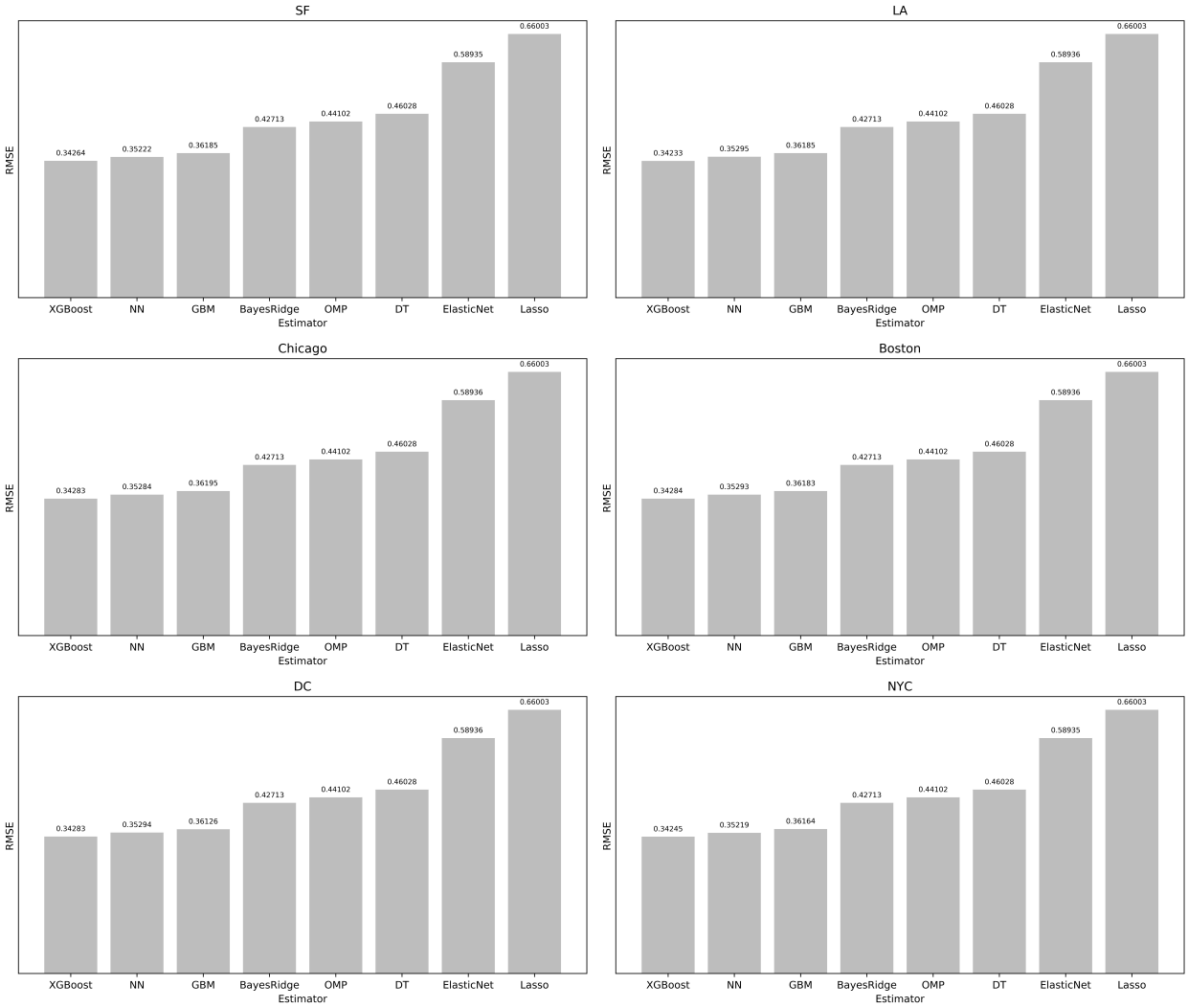
2.2.1 Model selection

We divide the data set into two parts. 80% of the data is used to train the model, and the other part is used to test. Moreover, in order to make the results more practical, we also split the training set and test set by city to train the city model separately. Training set is used to train model and test data to evalute models' performance. We considered a variety of linear and nonlinear models. For example, OMP, Linear Regression, LASSO, Ridge, NN, etc. The model is trained on the training set according to the default parameters, and the score of each model is calculated using RMSE as the standard of the model selection.

```
ests = [linear_model.Lasso(), linear_model.ElasticNet(),
        linear_model.BayesianRidge(),linear_model.OrthogonalMatchingPursuit(),
        DecisionTreeRegressor(),XGBClassifier(booster = "gbtree",eval_metric="rmse"),
        GradientBoostingClassifier(n_estimators = 100),
        MLPClassifier(activation='relu', solver='adam', alpha=0.0001)]
```

```
ests_labels = np.array(['LASSO','ElasticNet', 'BayesRidge', 'OMP','DT','XGBoost','GBM','NN'])
```

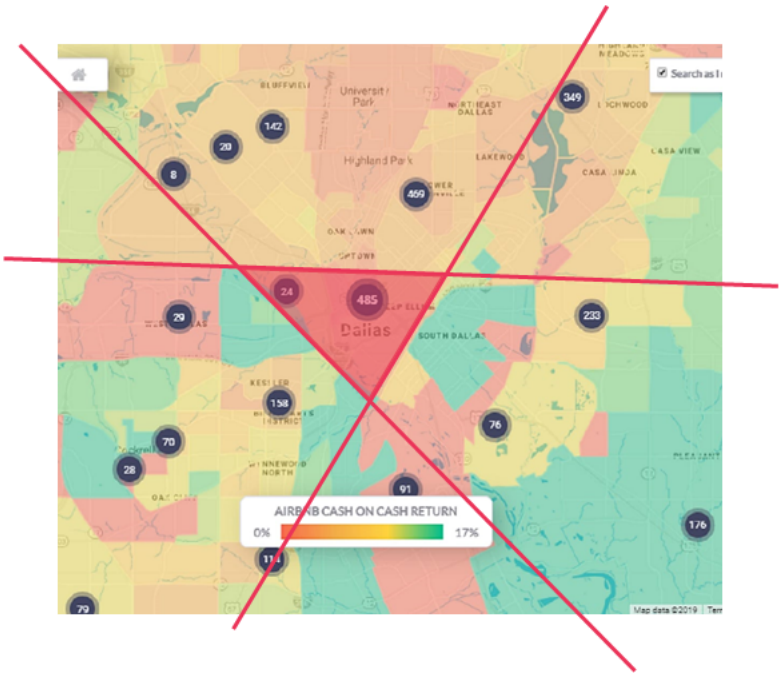
The linear models have larger RMSE, exceeding other models, so the linear model is not our optimal choice. They as a whole perform weaker than non-linear models. Especially the neural network and XGBoost, RMSE are less than 0.374.



We use in-depth processed full data sets and sub-city data sets to train various models, and find that the performance of the models is roughly the same. In the horizontal comparison, the overall performance of the nonlinear model is better than the linear model. Among them, XGBoost performed best. It can be said that the XGBoost algorithm best meets our task requirements.

2.2.2 XGBoost

After picking the XGBoost, potential explanations can be found to justify the performance of this model. Linear model performs badly probably because there are a lot of dummies variables inside the predictors, so nonlinear model is preferred. The property holders would typically price their properties and would refer to the prices of their neighborhoods to help determine their own base price, so the price should be very related to the locations. In the light of this statement, the tree model can perfectly split the data points in to regions according to their locations. For example, this line can be represented by the branch splitting criterion that a longitute+b latitute>c . XGBoost has made many improvement compared to a single tree model as it is an ensemble tree model that ensembles sums the prediction of multiple trees together.



The basic training process of the model is as follows:

1. Boosting:

$$
\begin{aligned}
0) \quad & \widehat{y}_i^{(0)} = 0 \\
1) \quad & \widehat{y}_i^{(1)} = f_1(x_i) = \widehat{y}_i^{(0)} + f_1(x_i) \\
2) \quad & \widehat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \widehat{y}_i^{(1)} + f_2(x_i) \\
& \dots \\
t) \quad & \widehat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \widehat{y}_i^{(t-1)} + f_t(x_i)
\end{aligned}
$$

2. Optimization: The objective function to be optimized is given by $obj = \sum_{i=1}^{n} l(y_i, \widehat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i), f_k \in \mathcal{F}$. It consists of two parts, training loss $L(\theta) = \sum_i (y_i - \widehat{y}_i)^2$ and regularization term $\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} \omega_j^2$

$$
\begin{aligned}
obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \Omega(f_t) \\
&= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \\
&\simeq \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t), g_i \triangleq \partial_{\widehat{y}^{(t-1)}} l(y_i, \widehat{y}^{(t-1)}), h_i \triangleq \partial^2_{\widehat{y}^{(t-1)}} l(\hat{y}_i, \widehat{y}^{(t-1)}) \\
&= \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w^2_{q(x_i)} \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2 \\
&= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T, I_j = \{i | q(x_i) = j\} \\
&= \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T
\end{aligned}
$$

The optimization problem has an analytic solution:

$$
\omega_j^* = -\frac{G_j}{H_j + \lambda} \ \& \ obj^* = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T
$$

3. Spliting: The tree will grow if certain spliting creteria met.

$$
Gain = \frac{1}{2} [\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L^2 + G_R^2)}{H_L + H_R + \lambda}] - \gamma
$$

2.2.3 Parameter tuning

The model will have a slight impact on performance due to different parameter selection, but it is not easy to determine the optimal parameters. In order to find the most suitable parameters for the urban housing price data set, we adopted the GridSearch algorithm to determine the optimal solution of the parameters by enumerating the parameter combinations.

```
set.seed(0)
all = expand.grid(max.depth = seq(1, 10),
                  eta = seq(0, 1, length.out = 10),
                  subsample = seq(0.5, 0.9, length.out = 5),
                  min_child_weight=seq(10, 50, length.out = 5),
                  gamma = seq(10, 30, length.out = 10),
                  nround = seq(500,2000,length.out = 20))
```

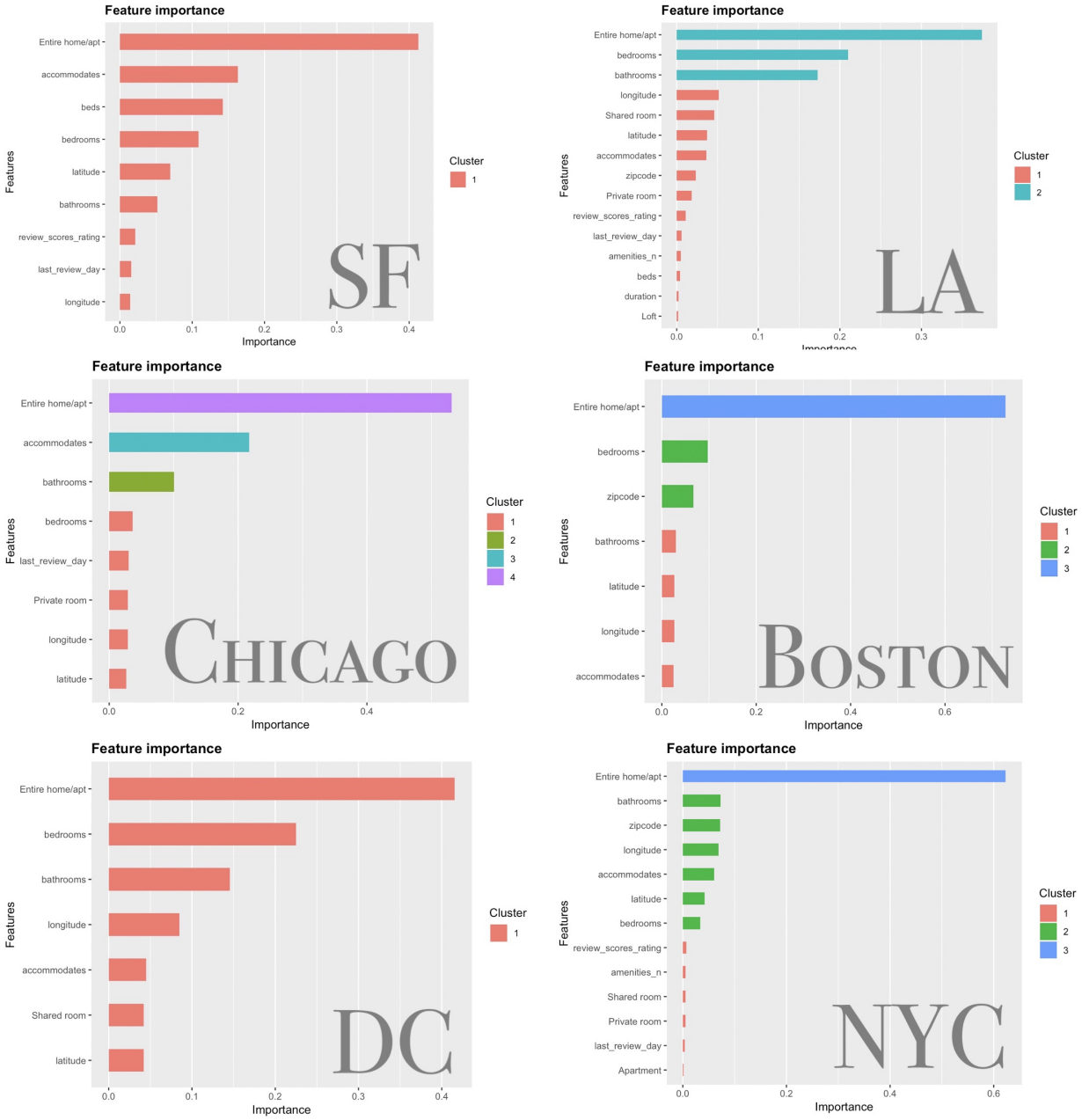
```
xgb_fit = xgboost(data = data.matrix(train[,-1]),
                  label = as.numeric(train$log_price),
                  booster = "gbtree",
                  max.depth = 10,
                  eta = 0.4444,
                  subsample=0.8,
                  colsample_bytree=0.8,
                  min_child_weight=10,
                  gamma = 10,
                  nround = 500,
                  objective = "reg:linear",
```

```
                eval_metric="rmse",
                verbose = 0)
pre_xgboost <- predict(xgb_fit ,data.matrix(test[,-1]))
rmse.predict = RMSE(pre_xgboost, test$log_price)
names <- colnames(train[,-1])
importance_matrix <- xgb.importance(names, model = xgb_fit)
xgb.ggplot.importance(importance_matrix)
```

3.1 Results

3.1.1 Feature Importance

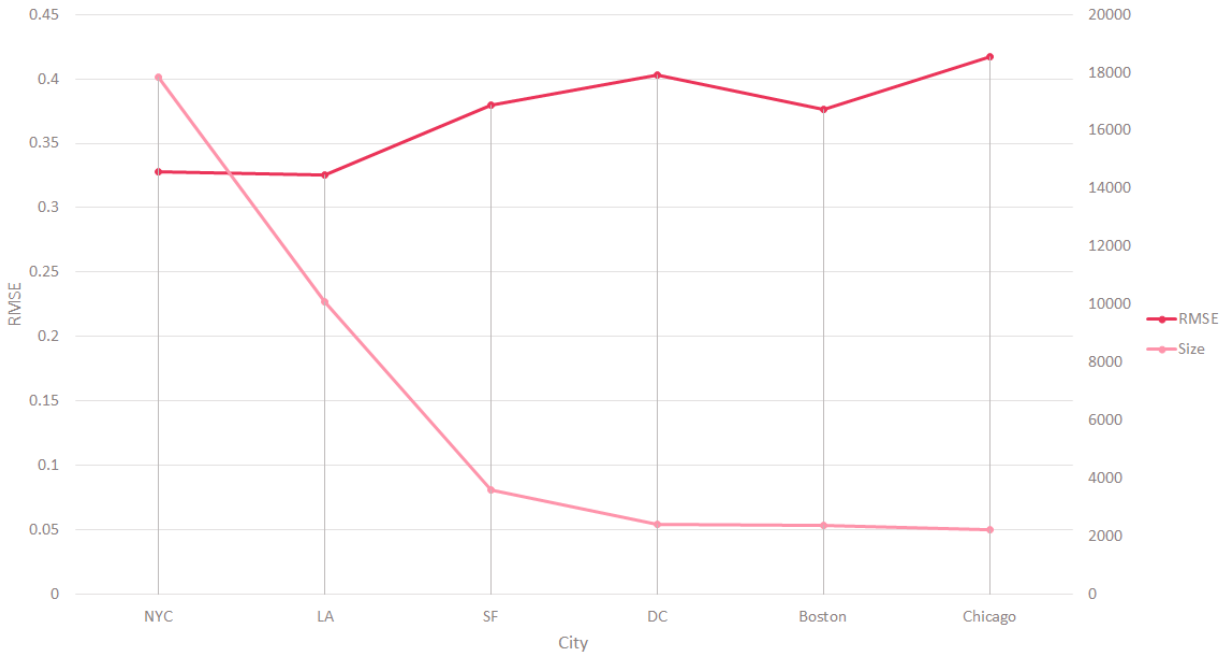
Feature importance are basically similar among all these cities despite some particular exceptions due to the specific conditions of each city. All features are first divided into four types: Location, Property, Host and Review. Location includes all features with geographic meanings. Property indicates all factors concerning the housing condition. Host comes from all information about the house owners. Review reflects the comments given by the previous tenants. The most important predictor is property_type (Property), which is intuitive that the entire house should price hight than a single apartment. Location also determines the price a lot through some factions such as zipcode, longtitude, latitude and so on. Factors like bedrooms and bathrooms can also be essential since these are Property-related features which cannot be easily altered.



3.1.2 Accuracy

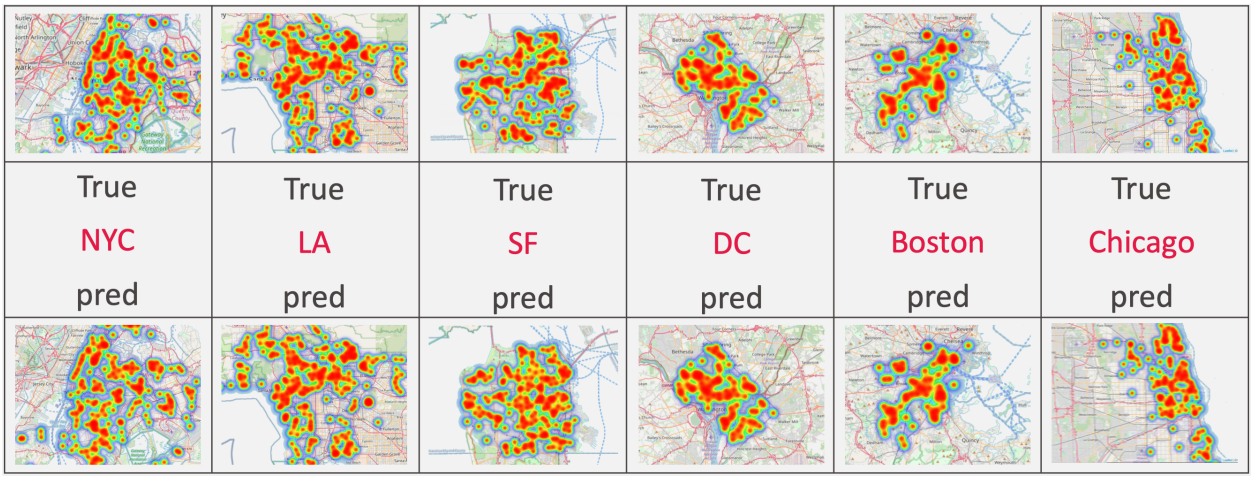
RMSE vs size

The RMSEs of each city, though are roughly the same, tend to perform better when the size of the samples is large(i.e., NYC and LA who have the largest sample size are of the best RMSE).



RMSE in Heatmap

To have a better sense of the accuracy of our model, heatmap is used as a visualized criteria of the RMSE. The warmer the color is, the higher the price is. Two heatmaps, one is plotted using the actual price and the other is plotted using the predicted price, are almost identical overall. If one take a look at detail, tiny differences can be found. The heatmaps show that the XGBoost model generally performs well instead of some minor deviation.




```
output = cbind.data.frame(test,pre_xgboost)
data_NYC = output
##NYC
leaflet(data_NYC) %>%
  addTiles() %>%
  # setView( 178, -20, 5 ) %>%
  addHeatmap(lng = ~longitude, lat = ~latitude, intensity = ~log_price,
             blur = 20, max = 0.05, radius = 15)
```

```
leaflet(data_NYC) %>%
  addTiles() %>%
  # setView( 178, -20, 5 ) %>%
  addHeatmap(lng = ~longitude, lat = ~latitude, intensity = ~pre_xgboost,
             blur = 20, max = 0.05, radius = 15)
```

3.2 Discussion

The results show that, our model gives an accurate prediction for the base price of the listings, where the property-related attributes contribute to the most predictive power of the model and location-related attributes play the second important role. It sheds light on the formation of prices on the market. Common sense may put too much weight on the location of the property when pricing, our results suggests that in addition to referring to prices of their neighborhoods, hosts need to be aware that the condition of the property should also be priced in. For hosts who has no ideas on property pricing and are reluctant to let house agents run the operations of their own properties, they now have a powerful tool to help them determine the base price of their properties.

By far we have tackled the problem we've come up at the beginning on determining the base price. With the base price determined, hosts can now dynamically price their properties with the assistance of the third-party tools. However, the situation could be that different properties may have different competitiveness at different time periods. For example, mountain resorts is likely to be more popular during the summertime and more sensitive to the seasonality, while apartment in the city centre is less sensitive. Thus in terms of the market efficiency, it's far from enough to determine only the base price itself and generates the same price sequence for properties with the same base price. Further studies can be focused on the time-series prediction that generate a dynamic price based on not only the physical attributes but also the time attributes of the property.

4. Conclusion

Though the accuracy of the prediction varies among six cities due to some inherent factors like the sample size of the data, the XGBoost model gives a relatively satisfactory prediction results of the price measured by the RMSE. Digging into the feature importance, the property-related and location-related factors dominate the price, which matches with the guess before the analysis. Property_type overwhelms all other factors to rank first in feature importance of most cities, is however, not expected before the experiment. The effect of location, though intuitively essential, has been dispersed by many collaborative elements to act on price. Generally speaking, this model gives a specific blueprint to the host and the guest to predict the price of listings in both numeric and graphical ways.

5. Reference

Airbnb price prediction, kaggle, https://www.kaggle.com/stevezhenghp/airbnb-price-prediction
T. T.Cai and L. Wang, "Orthogonal Matching Pursuit for Sparse Signal Recovery With Noise," in IEEE Transactions on Information Theory, vol. 57, no. 7, pp. 4680-4688, July 2011, doi: 10.1109/TIT.2011.2146090.
Murphy, Kp. Machine Learning : A Probabilistic Perspective /. Cambridge, Mass. :: MIT, 2012. Web.
Chen, Tianqi, and Carlos Guestrin. "XGBoost." Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge
Discovery and Data Mining (2016): n. pag. Crossref. Web. Bishop,
Christopher M. . "Neural Networks for Pattern Recognition." (1995).
T. Windeatt, "Accuracy/Diversity and Ensemble MLP Classifier Design," in IEEE Transactions on Neural Networks, vol. 17, no. 5, pp. 1194-1211, Sept. 2006, doi: 10.1109/TNN.2006.875979.