Use XGBOOST to Predict the Rental Based on Airbnb Open Data

Suhang Yao

School of International College, Beijing University of Posts and Telecommunications, Beijing 100876, China
ricardo@bupt.edu.cn

Abstract

XGBOOST is a considerably effective method for machine learning, which performs well in all kinds of competition. Using a dataset from Airbnb Open data, the paper examines extent of factors affecting the rental and what is the future trend of rental by applying XGBOOST and other machine learning methods. The various features of data are visualized for further procession, and knowledge transfers between the statistic of hyperparameters of XGBOOST and test error has been cognized, which are illustrates by the merging and collapsing of different hyperparameters. Results show that the data pre-processing significantly influence on the precision of prediction. Also, variation of distinct hyperparameters impacts test error on this dataset in varying degrees, and the most appropriate figure of them has been confirmed by making comparisons recursively. Finally, general ideas of data preprocessing and hyperparameters tuning are proposed.

Keywords

Machine Learning, XGBOOST, Data Preprocessing, Visualization, Prediction on Rental

1. Introduction

Since 2008, guests and hosts have used global home stay short rent and reservation platform Airbnb (AirBed and Breakfast) to enjoy the home-like customized living experience and offer international housing resources for investment. There are quantities of factors () influencing the rental in Airbnb, which has been a great concern of the users and investors. XGBOOST is a useful tool for massively parallel booted trees [1-3]. It is the fastest and best open source booted tree toolkit at the moment. Therefore, the research is developed to analyze and predict the rental base on the XGBOOST technologyand the dataset of the listing activity and metrics in NYC, NY for 2019.

The project refers number of papers, and the main sources of them come from various areas, including xgboost, K-fold cross validation, Pandas, Robust covariance and Scatter matrix, Numpy, and IPython, Boosting, Decision tree and Heatmap. What's more, I come to kaggle to learn other's code and imagination, finally make my project down. In this report, I absorb these paper's and website's thought, but I don't use their concrete sentence.

This report is intended for executing XGBOOST algorithm to predict the rental in Airbnb. The second and third paragraph are written to introduce the Airbnb dataset and XGBOOST algorithms respectively. The fourth paragraph presents the process of preprocessing the data set, EDA, choosing and tuning hyperparameter, etc. The conclusion and the list of references will be shown on the last paragraph.

This data file includes comprehensive and diggable information about hosts, geographical availability, necessary metrics to make predictions and draw conclusions. It has 16 columns and 48577 rows. Table 1 shows the concrete features of dataset:

Table 1
features of dataset

Id	listing ID
Name	name of the listing
host_id	host ID
host_name	name of the host
neighbourhood_group	city

Neighbourhood	area
Latitude	latitude coordinates
Longitude	longitude coordinates
room_type	listing space type
Price	price in dollars
minimum_nights	amount of nights minimum
number_of_reviews	number of reviews
last_review	latest review
reviews_per_month	Num of reviews per month
calculated_host_listings_count	amount of listing per host
availability_365	number of days when listing is available for booking

2. Analysis base on XGBOOST Algorithm

XGBOOST algorithm is the used tool for the article. The algorithm I use is XGBOOST, and below is my acknowledgement about XGBOOST. Above all, there are two significant concepts need to be acknowledged before this. The first one is decision tree, which is a common method for classification and regression. Decision tree algorithm is the process of classifying instances according to the input features, that is, through the tree model, judge the eigenvalues at each layer of the tree, and then reach the leaf node of the decision tree, which completes the classification process. I will conclude that XGBOOST is the integration of many cart regression trees. In fact, the essence of classification and regression is the same. They are both the mapping between feature and label, while the result of classification is discrete value and regression is continuous. Therefore, the output of the regression tree is in the form of numerical value as well. For example, the housing loan to someone may be a continuous value, which can be any value between 0 and 0.5 million dollars. The second is boosting ensemble learning, which refers to joint decision-making by multiple decision trees with correlation. What's for correlation? For instance, there is a sample is [(2, 4, 5) -> 4] ([data -> label]), and the first decision tree's output trained with this sample predicts 3.3. Then the input of the second decision tree to be trained will be [(2, 4, 5) - > 0.7], and 0.7=4-3.3, which means the input sample of the next decision tree will be related to the training and prediction of the previous decision tree.

On the previous basis, I will conclude the goal of XGBOOST [3] are to establish K regression trees to make the predicted value close to the real value as far as possible and have the best ability of generalization. Therefore, XGBOOST built a model like this:

$$L(\phi) = \sum l(\omega - y_i) + \gamma + \frac{1}{2}\lambda\|\omega\|^2 \quad (1)$$

It can be seen that if the loss function $l(\omega - y_i)$here uses square error, and the above function can be simplified into a quadratic function to find the minimum, that is, quadratic optimization. If the loss function is not a square error, we can approximate it into a square error by some mathematical methods like Taylor expansion. Therefore, the idea of XGBOOST algorithm is to select a feature to split, calculate the minimum value of loss function, and iterate until a stop condition. There are 3 condition for stopping: a. when the gain caused by the introduced splitting is less than a threshold; b. set a super parameter max_ depth, stop when the tree reaches the maximum depth c. set the minimum sample weight 'min_ child_ weight', when the sample weight sum is less than the set threshold, XGBOOST will stop.

Every machine learning algorithm has multiple hyperparameters, XGBOOST is no exception as well. I'm going to introduce some parameters for tree booster. There are the lists: 1.eta, control the weight reduction of each learning, shrinks the feature weights to make the boosting process more conservative. 2.gamma, minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be. 3.max_depth, the maximum depth of the tree. Increasing this value will make the model more complex and more likely to overfit. 4.min_child_weight, minimum sum of instance weight (hessian) needed in a child, if it is set to a positive value, it can help making the update step more conservative. 5.subsample, subsample ratio of the training instances. Setting it to 0.5 means that XGBOOST would randomly sample half of the

training data prior to growing trees. and this will prevent overfitting. Subsampling will occur once in every boosting iteration. 6.colsample_bytree, the subsample ratio of columns when constructing each tree. Subsampling occurs once for every tree constructed. 7.alpha, L1 regularization term on weights. Increasing this value will make model more conservative. Above these is also the hyperparameters I tuned in my project.

3. Preprocessing and Visualization

I'm going to describe this preprocessing with seven parts, which contains all main features. Moreover, visualization are made on each features I concerned by using sklearn and pandas as tools [4-6].

3.1. Preliminary exploratory data analysis on the data set.

The purpose of preliminary exploratory data analysis for the precision and accuracy on the further analysis, which can be achieved by missing value analysis, useless features deletion and filling the empty of significant features. Visualization of the remaining dataset is also applied for explore.

Firstly, count the concrete number of each feature in the dataset and analyze the missing values. The statistic result of counting is shown as figure 1. It can be observed that majority of the columns has 48895 columns, while three rows (reviews_per_month, last_review, name and host_name) with fewer not-null amount are identified to have missing values. These rows with missing value are dropped on the purpose of eliminate influence on further analysis.

```
Data columns (total 16 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              48895 non-null  int64
 1   name                            48879 non-null  object
 2   host_id                         48895 non-null  int64
 3   host_name                       48874 non-null  object
 4   neighbourhood_group             48895 non-null  object
 5   neighbourhood                   48895 non-null  object
 6   latitude                        48895 non-null  float64
 7   longitude                       48895 non-null  float64
 8   room_type                       48895 non-null  object
 9   price                           48895 non-null  int64
 10  minimum_nights                  48895 non-null  int64
 11  number_of_reviews               48895 non-null  int64
 12  last_review                     38843 non-null  object
 13  reviews_per_month               38843 non-null  float64
 14  calculated_host_listings_count  48895 non-null  int64
 15  availability_365                48895 non-null  int64
```

Figure 1: number of each feature

Secondly, there are relatively few missing in name and host_name. Considering they make little contribution to data modeling; these two columns are deleted.

Third, two features with generous missing values (reviews_per_month, last_review) are not deleted despite the account for 20% of the total samples, in the consideration to reserve the diggable information in the dataset. Due to the reason that review_per_month and last_review would maintain empty if there is no review, the action is taken to fill reviews_ per_ month with 0 and fill in last_review with the earliest comment time of the whole data set.

As for deeper analysis, I visualize the distribution of some features and performed correlation analysis. The amount for host_id accounting for 76% of the total number of samples in the data set, and the host_ ID belongs

to discrete data which has little contribution or even introduce the negative contribution of modelling. Therefore, consider deleting this feature. The feature id is deleted for the same purpose.

3.2. The procession and visualization of each particular feature

The purpose of inner procession of particular feature of the remaining data is to filter out the edge/not common value and make classical analysis.

3.2.1. Neighborhood group

The data of neighborhood group are counted as output. The analysis only reserves three of five neighborhood groups because there are limited rooms distributed in Staten Island and the Bronx. After this step, remaining data visualization is applied as figure 2.

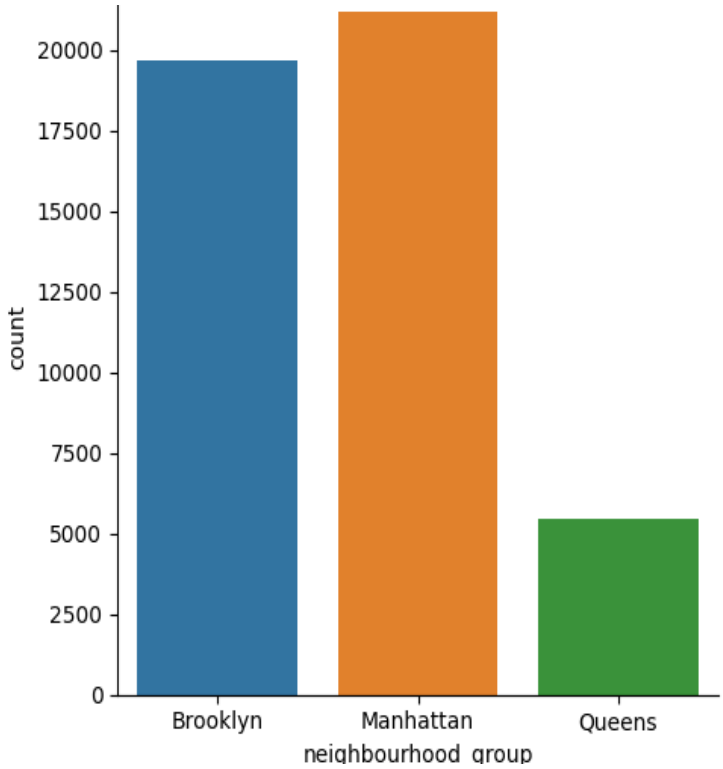

Figure 2: statistics in each city after processing

3.2.2. Longitude and latitude distribution

Longitude and latitude are somewhat correlated with each other. This is because the locations of properties tend to come from clusters. Which can be observed in the figure 3 and figure 4 is that house with the feature that longitude close to - 73.9 ~ - 74.0 and latitude close to 40.7 are more welcomed by tourists.

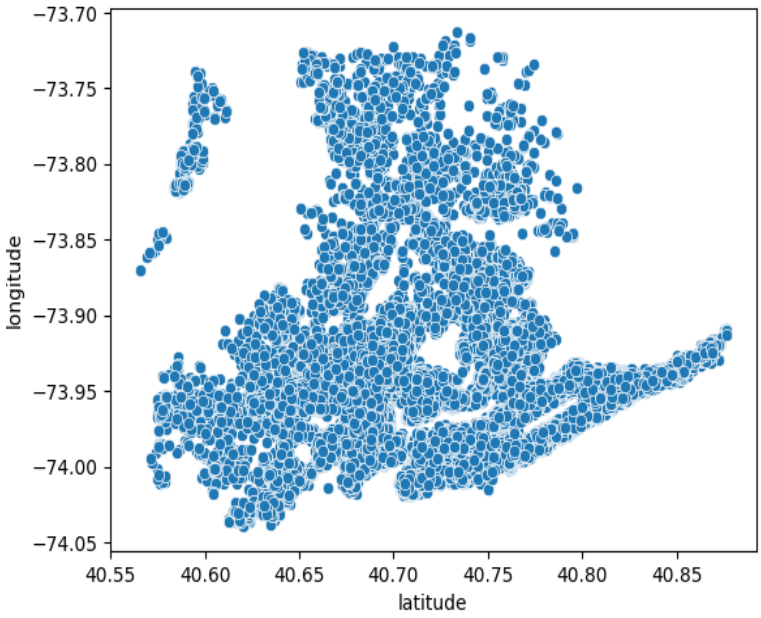

Figure 3: correlation between longitude and latitude

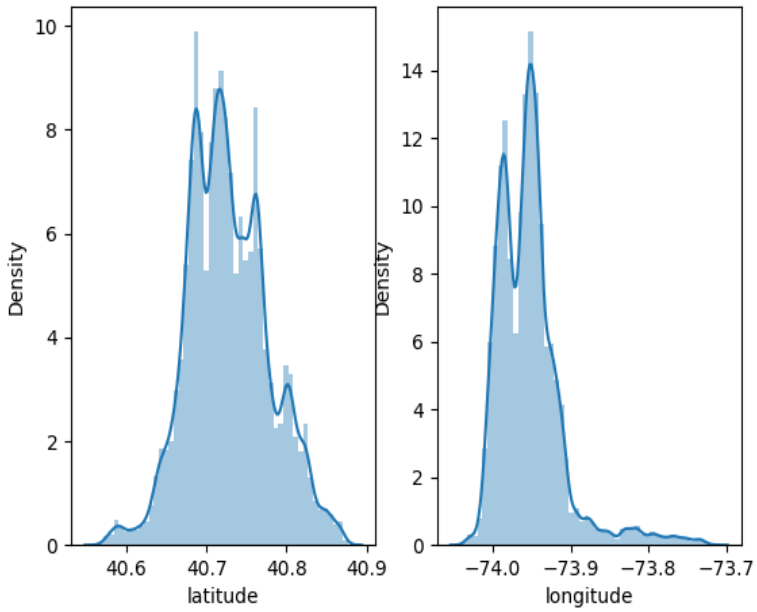

Figure 4: relation between density and latitude or longitude

3.2.3. room_type

There are three categories of the room source after exploring the statistics of room type. The data of shared room could be outlier due to the limited amount, so it is cancelled. The following figure 5 shows the count of private room and entire room respectively.

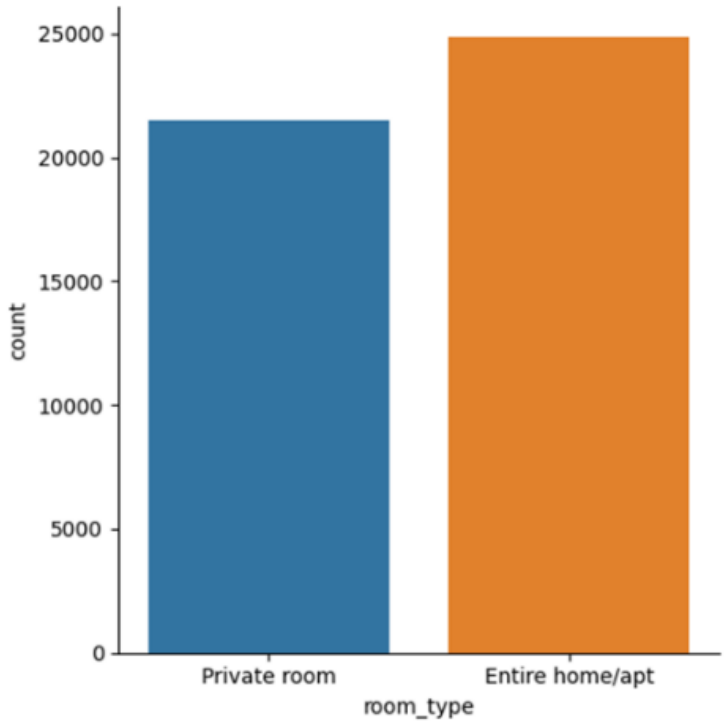

Figure 5: data of different type room

3.2.4. minimum_nights

As figure 6 shows, the data presents a very severe skewed distribution, and the minimum day value required for most rooms is very small. For severely skewed data, log1p() function in numpy can be used to process it (figure 7).

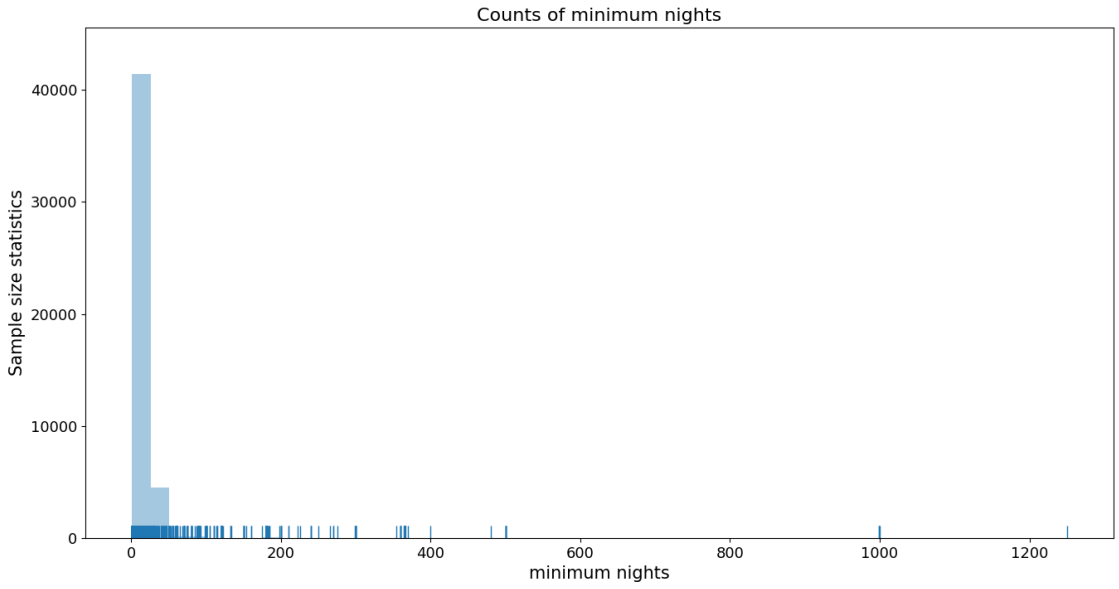

Figure 6: distribution of minimum nights

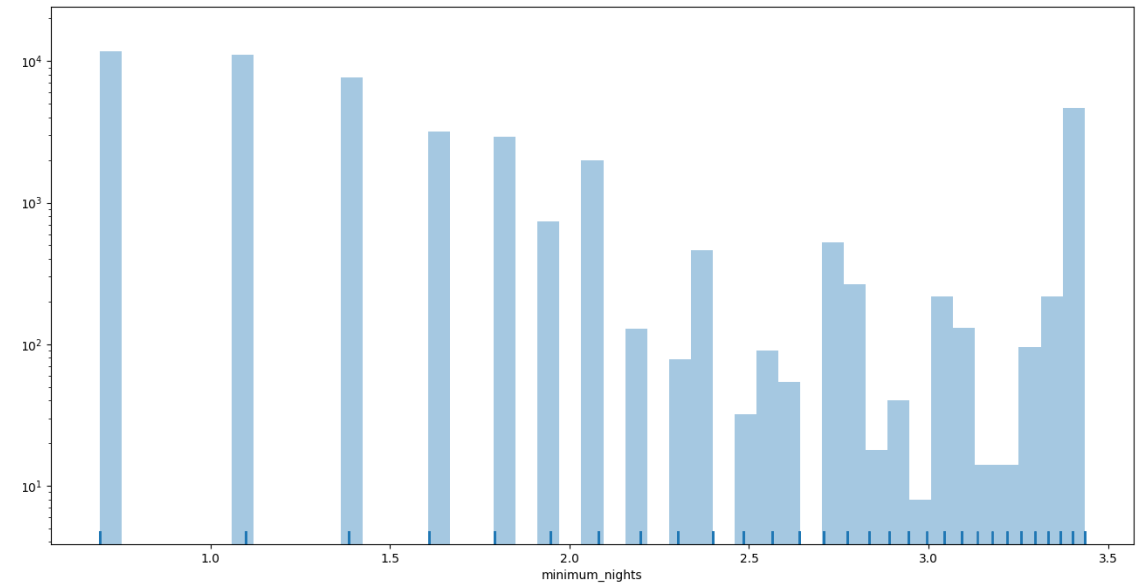

Figure 7: distribution of minimum nights after logarithmic transformation

3.2.5. Ability_365

The statistics of bookable days of multiple samples in dataset shows a larger probability to be booked within 15 days, but the distribution does not show intermittent skewness

Meanwhile, the majority of Ability_365 of the sample within 15 days, and the number of other days is small, but the distribution does not show intermittent skewness. (figure8)

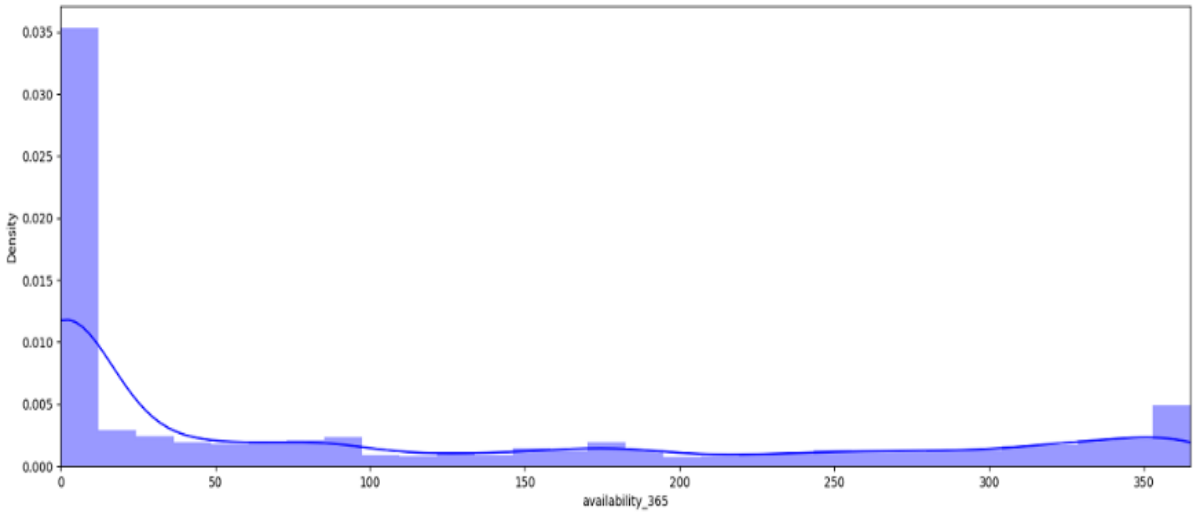

Figure 8: distribution of available days in one year

3.2.6. Price

The figure 9 of price shows a very significant skew distribution, which is quite similar comparing to minimum nights. The distribution of room price data can be transformed into an approximate Gaussian distribution by using np.log1p function, and it is presented in figure 10.

Due to the reason that outlier of the importance as a super parameter and the serious skew distribution of price, this hyperparameter is focused on for deeper analysis. By checking the percentage distribution

of the price, we can get the price value and extreme value corresponding to several important percentages. 95% of price is less than 369.00, 99% of price is less than 799.00, and the maximum value is 10000.00.

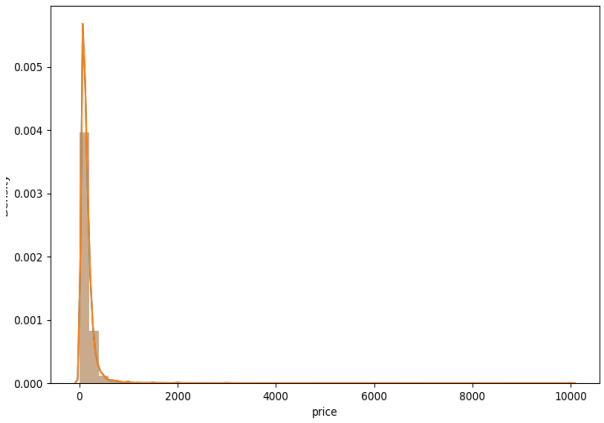

Figure 9: distribution of price

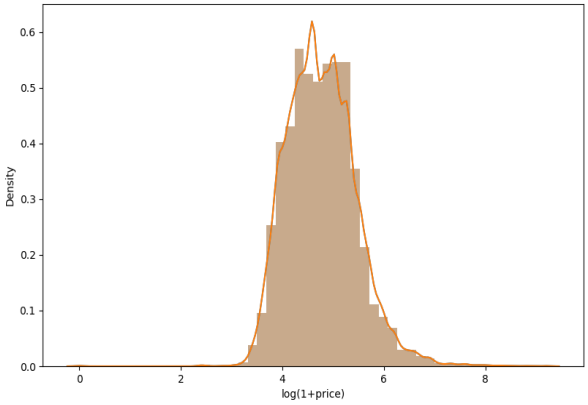

Figure 10: distribution of price after logarithmic transformation

4. Correlation Analysis

This part makes the correlation analysis of the main features based on the Heatmap and scatter matrix.

Figure 11 is Heatmap [7] of main features, which is also called Pearson correlation matrix. The figure shows the number of reviews per month is fair (40%) correlated with the total number of reviews and the total number of reviews is correlated (30%) with the availability of the property. Both of these correlations make sense. It's also interesting that the longitude is anticorrelated (20%) with the price. That also makes sense - property in the Bronx and in Queens is cheaper than Manhattan and Brooklyn.

Pearson correlation coefficient can only reflect the linear correlation degree between the two features. Therefore, figure 12 as the scatter matrix [8] more intuitively explore the correlation between the features. It is used to visualize the features between the two. The scatter matrix of price with longitude and latitude and it with number_ of_ reviews are separately here.

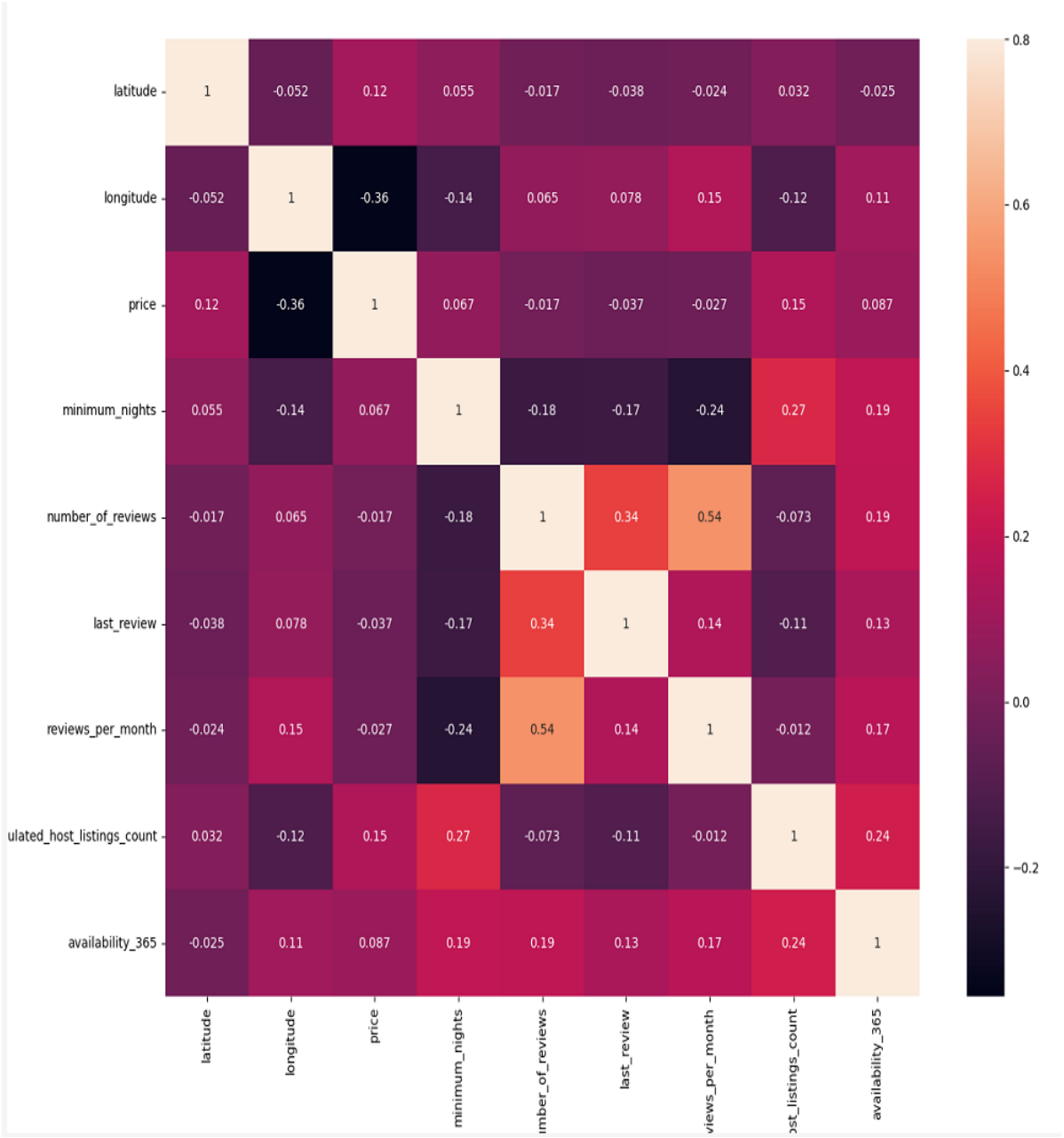

Figure 11: heatmap of main features

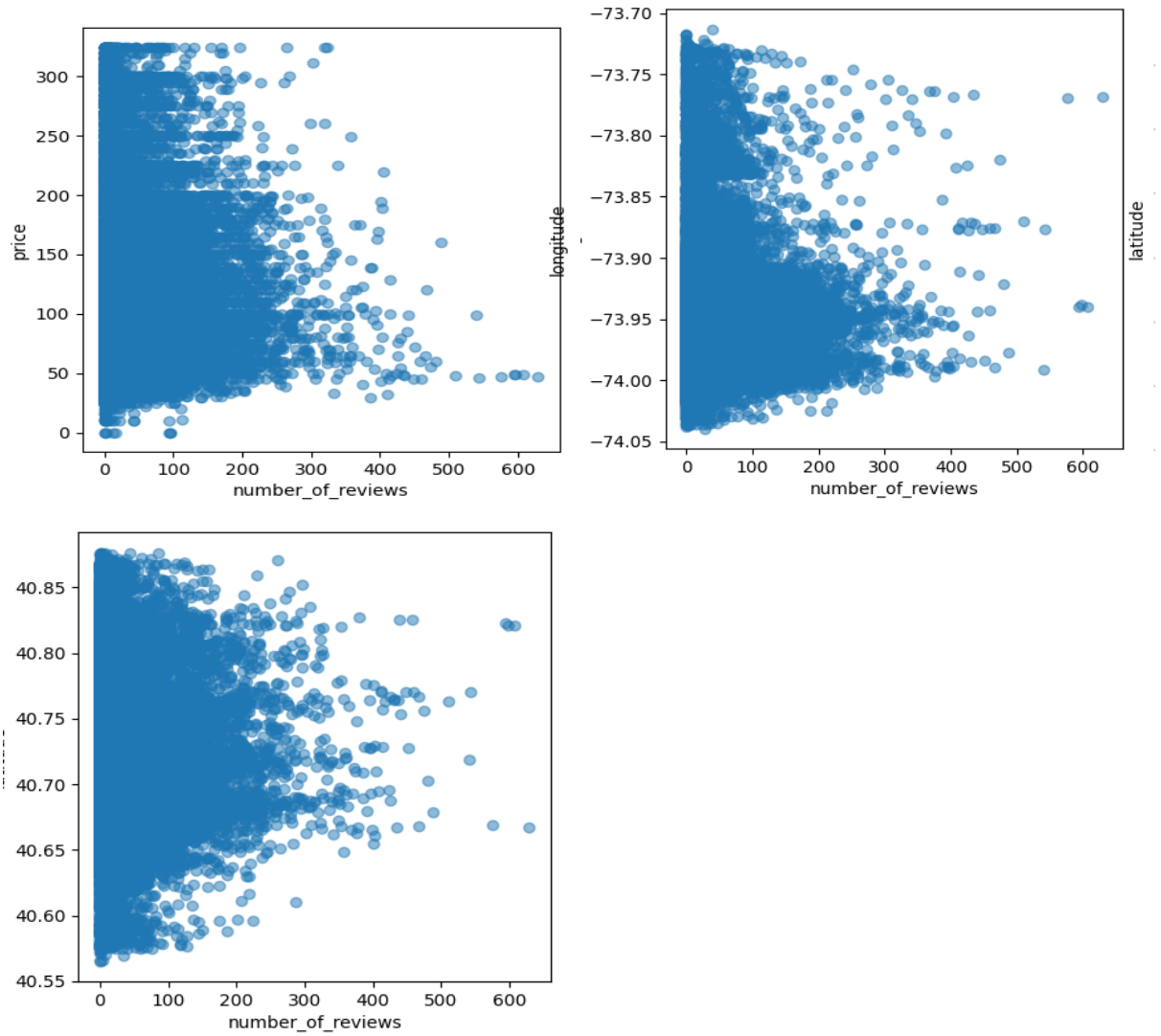

Figure 12: three type'scatter matrix

As for the final preprocessing before data modeling, the First stage is encoding the categorical features. Before modeling, the data type and the classified data needs to be processed with dummy variables first. Get using pandas_ Dummies module can quickly encode all classification variables by one-hot.

Secondly, splice the dummy variables in the first step with all numerical data in the original data set. The data is split into a test set and a training set, where training set compose 80% and test set compose 20%. Next, hold out the test set until the very end and use the error on those data as an unbiased estimate of how my models did. Finally, I now scale the design matrix with RobustScaler() in sklearn[4] so that each predictor has zero mean and unit variance. This helps the convergence of machine learning algorithms such as linear regression. Above these, consequently, I had the final data modeling and used it for tuning hyperparameters.

In terms of the hyperparameters, I mainly considered seven ones, separately are learning rate, gamma, alpha, subsample, colsample_bytree, max_depth and min_child_weight. I tuned these hyperparameters through the values of mean crossover error, crossover error standard deviation and training error.

5. Evaluation and DATA mODELING

After modification and evaluation, the for best predicted model is shown in table 2.

Table 2
error of best predicted model

Algorithm	CV error	CV std	
XGBRegressor	0.122506	0.001864	
training error	test error	training_score	test_score
0.116814	0.127254	0.646903	0.614792

5.1. Training error and test error analysis

Mean squared error of test data is used as the final performance evaluation of the model, which is called training error here. It can be seen that the value of test error is 0.127254, which is at a relatively low level, indicating that this model has high prediction ability. However, the value of training error is lower because some of them participate in model training as validation sets. Therefore, as the untrained data, the mean square error of test set will increase to a certain extent.

5.2. Data cleaning

The main factor of test error is data cleaning in this predicting model, followed by tuning hyperparameter. As for data cleaning, the maximum 5% price is treated as the outsider and apply the delete operation. If the outsider of price is not cleaned, the test error will be very high, reaching about 19%, indicating that the prediction model doesn't perform well. Deleting a greater proportion of prices (8% or 10% of the total amount) is also an attempt, the predicting model of which will produce even a lower test error at 11%. However, in this circumstance, the deleted sample size is extremely large, which is not applicable to the real situation. Under the above considerations, I chose a price of 5% as the outsider, which can satisfy the condition of both ensuring a lower test error and better conforming to the actual situation simultaneously.

5.3. Evaluation

After finishing data cleaning, I compared the distribution of actual prices and predicted values by means of scatter matrix. It can be seen that the predicted values fit the real house prices better under most conditions, but large error can still be observed when predicting the edge prices. Meanwhile, due to the deletion of outlier, overpriced houses do not exist in this data set and cannot be predicted.

5.4. K-fold cross validation

K-fold cross validation [9, 10] can robustly evaluate the performance of the model on unknown data, but larger computational complexity, longer time and more computational resources are consumed due to the large amount of data in this data set. I divide my previous training set into k parts, of which k-1 is used as the new training set and the remaining 1 is used as the verification set. In this way, I execute K rounds to obtain mean cross validation error as the evaluation of the overall performance of hyperparameter. The specific process is to verify the super parameters on the training set and verification set, and select the super parameters with the minimum average error. After selecting the appropriate super parameters, the training set and verification set can be combined, and the model can be trained again to obtain the final model.

5.5. Predicted result

In the last stage, table 3 shows my final price prediction results. It can be seen that when the selected data concentrates on the center, the prediction results of the model are more accurate with smaller error, which I pointed out before. This is because there are few data distributions at the edge, which is not able to generate a decisive impact on the data modeling; On the contrary, data modeling mostly depends

on the data located in most intervals, which is the reason for the high prediction accuracy of this model for most data.

Table 3
comparison between actual rental and the prediction

	actual price	prediction
count	8805	8805
mean	125.3752	117.2988
std	72.49449	51.49348
min	0	34.97095
50%	100	111.1701
75%	165	158.1578
80%	185	168.8609
85%	200	178.501
90%	230	189.0144
95%	275	205.3155
max	369	287.4212

6. Conclusion

In this paper, I use XGBOOST to finish the prediction of rental in Airbnb. Result has exposed best performed prediction and the conditions for approaching it. Furthermore, the work proposes a general idea for the preprocess of dataset and for further improvement of this model, a better result may be produced by repeatedly using distinct random grouping data for cross validation and reducing the variance of a single cross validation scheme to increase the prediction accuracy outside the sample.

7. References

[1] Myles A J, Feudale R N, Liu Y, et al. An introduction to decision tree modeling[J]. Journal of Chemometrics: A Journal of the Chemometrics Society, 2004, 18(6): 275-285.

[2] Svetnik V, Wang T, Tong C, et al. Boosting: An ensemble learning tool for compound classification and QSAR modeling[J]. Journal of chemical information and modeling, 2005, 45(3): 786-799.

[3] Chen T, He T, Benesty M, et al. Xgboost: extreme gradient boosting[J]. R package version 0.4-2, 2015, 1(4): 1-4.

[4] Feurer M, Klein A, Eggensperger K, et al. Auto-sklearn: efficient and robust automated machine learning[M]//Automated Machine Learning. Springer, Cham, 2019: 113-134. MLA

[5] McKinney W. pandas: a foundational Python library for data analysis and statistics[J]. Python for high performance and scientific computing, 2011, 14(9): 1-9.

[6] McKinney W. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython[M]. " O'Reilly Media, Inc.", 2012.

[7] Metsalu T, Vilo J. ClustVis: a web tool for visualizing clustering of multivariate data using Principal Component Analysis and heatmap[J]. Nucleic acids research, 2015, 43(W1): W566-W570.

[8] Chen M, Gao C, Ren Z. Robust covariance and scatter matrix estimation under Huber's contamination model[J]. The Annals of Statistics, 2018, 46(5): 1932-1960.

[9] Fushiki T. Estimation of prediction error by using K-fold cross-validation[J]. Statistics and Computing, 2011, 21(2): 137-146.

[10] Refaeilzadeh P, Tang L, Liu H. Cross-validation[J]. Encyclopedia of database systems, 2009, 5: 532-538