IEOR 142 Final Project
Using Machine Learning Models to Predict Airbnb Listing Prices

Yuchen Yao
Dec 16th, 2022

Link For Code:
https://drive.google.com/drive/u/1/folders/1L5HsGtmojW0X5l1peLVh8MjbQTZs_p9F

Source of Data:
http://insideairbnb.com/get-the-data/

PART I: Motivation

Airbnb is one of the most popular internet marketplace for short-term housing rentals. This internet unicorn has more than 200 million active users with over 6 million listings on its website. While hundreds of thousands of listings give users a great flexibility of choosing beautiful homes for all price ranges, one biggest challenge that customers face is to accurately determine whether the money they pay is worth the conditions of the accommodation. On the other hand, the Airbnb hosts would like to determine their optimal price for their listing to maximize their profits.

Although the current Airbnb algorithm provides their hosts a general guidance on pricing their homes, it's still impossible for hosts to accurately set the "best" price. Some third-party softwares such as *Beyond* do provide paid service to help users determine the right price for housing, but people shy away from using it for its hefty price. Therefore, our aim is to design an algorithm using various machine-learning methodologies and a number of listing features to predict the prices of listings. Our prediction would give an Airbnb host an optimal price they should charge for their listing. On the consumer side, this will help travelers determine whether or not the listing price they see is overpriced or underpriced.

PART II: DATA Cleaning & Pre-Processing

For the source, we are using data from insideairbnb.com, a website that combines data about Airbnb listings from all over the world in order to provide advocacy on the impact of short-term rentals on residential communities. Originally, we wanted to combine data from three major cities (San Francisco, Los Angeles, and San Diego) in California. However, this dataset is too large to perform any cross-validation, therefore we decided to focus on San Francisco only. Each row of the dataset represents one listing on the Airbnb website on Sep 7, 2022. The dataset contains informations on the per night price, the host (eg. host name, whether the host is super host), the housing (eg. amenities), and reviews (eg. score on its location, score on its cleanliness)

Drop rows

Except for the 4527 listings in San Francisco, the original dataset also contains 166 listings in Daly City and 3 listings in Tiburon, constituting 3.7% of the total observations. We excluded those 169 listings for our project, as they are unrelated to the goal of our project (predicting the price in San Francisco) and may negatively affect our model performance.

Drop columns

Among those features, we first deleted features that aren't giving information about the houses' property, like "id", "url", " name", or features that have long texts as value and are hard to be processed into categories, like "description" , "neighborhood_overview" etc.

Process "amenities" and "property_type"

Given that the "amenities" feature contains various information in the form of texts for each housing, we converted this feature into various dummy features. The original value for amenities feature is a text depicting all the amenities in each apartment. To have a better view of what this feature contains, we first converted each original value (a text depicting all the amenities that this apartment offers) to an array and took out each element (each amenity that this apartment offers) from arrays then calculated how many times they have appeared in the data set. We excluded amenities that occurred too frequently(more than 5000 times) or barely occurred(less than 10 times) to avoid redundant features, then chose keywords for the rest of the amenities that we are going to include in our final dataset to construct new columns, indicating whether this housing has this amenity. For example, one of the new features is named "tv", if a housing's original "amenities" contains the keyword "tv", then it will has 1 as this new feature's value, otherwise it will has 0. For the "property_type" feature which also uses text as value, we applied the same method by looking through all the categories in this feature and their occurrences. Then we extracted some keywords as the names of the new features.

The housing will have this feature's value as 1 if its "property_type" contains this keyword, otherwise, it will be 0. Those features will present what property_type the housing is.

Convert data type

Given that values of some features are originally provided as strings, we concerted the data type of these features to float for the convenience of model training. Specifically, for [host_since], depicting the time the host first started listing, we extracted the year and converted to float. For [host_response_rate] and [host_acceptance_rate], we converted strings to float. For [host_is_super_host], [host_has_profile_pic], [host_identity_verified], [has_availability], [instant_bookable], which are all dummy variables, we converted original values in strings, 't' and 'f' to float, 1 and 0. For [bathrooms_text], we replaced it with two features [bathroom_count] and [bathroom_type] by splitting the original text , such as "1 shared both" into 1 (number of bathroom) and "shared bath" (type of bathroom). For [bathroom_type], we further cleaned up to four types so that "shared baths" and "shared bath" are both categorized into "shared bath". For [price], we converted strings to float.

Normalization

Considering our data has very varying scales, we normalized the dataset for the models' better performance and easier analysis and interpretation.

PART III: Analytics Models

Since this is a prediction problem and the dependent variable is continuous, we used the following machine learning methods to build prediction models on our data. To improve model performance, we performed feature engineering to select independent variables and utilize cross-validation to select the optimal technical parameters. By looking at the data using these potential methods, we experimented with different approaches and figured out the optimal way to analyze the data. We also evaluated our model performances using different metrics such as RMSE, MAE, and R^2 (please refer to appendix 1)

Linear Regression

For the linear regression model, we first regress price on all the other variables. However, we realized that there are a lot of multicolinearity issues among the variables. Therefore, we then wrote an algorithm to automate the variable removing process by first evaluating the VIF of each variable, and then removed variables that has the highest VIF one by one until no feature had a VIF greater than 10. After that, we fitted an OSR model on the standardized data with scikit-learn. The resulting out-of-sample R-squared value is 0.4195, MAE is 0.4818, and RMSE is 0.7506.

Decision Tree Regressor (CART)

For the decision tree regressor model, we fitted the data through using a grid search to find the optimum model performance. We used a n_splits = 5 K-Fold and did the cross validation analysis on the Regressor model. Then, the data is tested and evaluated based on its R2 score performance. The best performance model is hence producing a resulting OSR square of 0.3447. The MAE and RMSE for the decision tree model are 0.5399 and 0.6407 respectively.

Random Forest

For random forest, we started by creating the grid values used for grid search cross validation process for the optimum performing model. The grid search is conducted based on selecting the best performing R2 score. Moreover, the model is re-evaluated using grid search for the optimal performance. This resulted in an OSR-square value of 0.5684, with MAE and RMSE to be 0.3725 and 0.6407 respectively. The random forest model performs better than CART as it is a collection of decision trees whose results are aggregated into one. In the random forest model, each tree fits the data well and they are averaged to reduce variance. Therefore, it has better ability to limit the problem of overfitting without substantially increasing errors due to bias.

Bagging

We also implemented a vanilla bagging method by setting the max_feature parameter to equal 20. This method achieved an out-of-sample R-squared value of 0.5839, MAE of 0.6354, and RMSE of 0.3656. The bagging method has a better performance than CART as it allows many weak learners to combine efforts to outdo a single strong learner. However, we also lost some interpretability of the model compared to CART.

Boosting

In addition to bagging, we also implemented boosting to combine a bunch of "weak" models sequentially. The model parameters that we used are n_estimators=2000, learning_rate= 0.001, max_leaf_nodes=3, max_depth=10, and min_samples_leaf=10. The resulting OSR2 is 0.3513, MAE is 0.5401, and RMSE is 0.7934. The performance of boosting is a lot worse compared to random forest and bagging, and is only a bit better than CART. We believe the reason is that the parameter for boosting is very hard to turn, and we didn't perform cross-validation to optimize the parameter for boosting due to the limitation of computing power.

Neural Network

We also implemented a more complicated neural network mode that comprised of a node layers, an input layer, many hidden layers, and an output layer. We used "relu" as our activation function and "lbfgs" as our solver. Then we tried to optimize the OSR2 by manipulating the hidden layer size. We first set the hidden layer size to be 10. However, this relatively deep neural network would easily overfit the training set if we have too many hidden layers. Therefore, we reduced the number of hidden layers to 3 to produce the best OSR2. The resulting OSR2 for neural network is 0.4850, with MAE and RMSE equal 0.4467 and 0.7069 respectively. Here, the performance of this more complex mode is better than linear regression and CART, but not as good as random forest and bagging. Therefore, we think using the more complex model might not be necessary in this case.

KNN

We implemented a KNN (K nearest neighbor) model by first finding the distances between a query and all the samples in the data, and then selecting the number of samples closest to the query, and then averaging the labels. We set n_neighbors equals 5, and that results the OSR2 of 0.3925, with MAE and RMSE equals 0.4616 and 0.7678 respectively.

Model Performance Results

We created 500 bootsrtap sample from the test set for each of the six models built above. The MSE is computed and recorded in the dictionary with the model as the key. After plotting the MSE value under each bootstrap sample, we noticed that bagging and random forest model perform better than other models, with lower mean squared error and less variability. (Please refer to Appendix 2). In addition, after applying several different types of data analysis techniques on the model, we are not extremely confident in the model as many of these data shows great room for improvement. The largest out-of-sample R squared value of 0.5839 for bagging and 0.5770 for random forest model. These numbers indicate that the model is able to explain approximately 57% of the variance in the data, and predict approximately 57% of the differences in Airbnb prices based on the input features that were used to train the model.

There are several limitations to this analysis that could impact the performance of the model and the interpretation of the R squared value. One limitation is that the model may be overfitted to the training data, meaning that it performs well on the data that it was trained on but may not generalize well to new, unseen data. Another limitation is that the model may not have been trained on a diverse enough dataset, which could lead to biases in the predictions. Additionally, the R squared value does not take into account the absolute errors in the predictions, only the relative errors. This means that the model could have large errors in its predictions but still have a high R squared value if the errors are consistent across all predictions. Finally, the R squared value only represents the variance in the data that is explained by the input features that were used to train the model. There may be other

factors that influence Airbnb prices that were not included in the model, which could limit the overall accuracy of the predictions.

Future directions

If we were to extend the analysis in the future, we would aim for improving the performances of the analysis through exploring more methods and looking at more ways we can better interpret these types of data and more accurately. We would also incorporate data from before and after COVID period data to see if the housing price has returned to normal and if the influence of the pandemic is still visible on the housing market. This would make our data analysis model more comprehensive and takes into consideration more factors that plays a part in the housing market.

PART IV: Impact

By building this model, we were able to better understand the Airbnb platform and see how property owners can maximize their profit through finding the optimal listing price for their houses. Consumers can also have a better understanding of the overall housing market by being able to identify if the house is listed with an appropriate price and whether or not it would be a good decision to rent this unit compared to others. Our model can also help the Airbnb platform to better predict listing prices, hence helping the platform to choose which units should be displayed at top of the search results to attract more attention and build machine learning results into the platform in order to attract more users and improve user retention rate performance. While our model is trained on data from San Francisco, our methodologies can also apply to other cities around the world.

Hence, the scope of our research can be expanded to better improve the scope of the research. To enhance the effectiveness of machine learning in determining Airbnb rental prices in San Francisco, it would be useful to incorporate more data, such as details about local events and attractions, weather patterns, and past pricing. This would enable the machine learning model to make more informed pricing decisions that better reflect the demand for rentals in different areas and seasons. The model's performance could also be improved by training it on a larger dataset. Another option would be to incorporate the model into a system that enables dynamic pricing, which adjusts prices based on changes in demand and other factors in real-time. By continuously adapting to new data and market conditions, the machine learning model could have a greater influence on Airbnb rental pricing strategies in San Francisco.

The impact of machine learning on Airbnb prices in San Francisco may vary across different subpopulations of interest for a number of reasons. One factor could be differences in demand for rentals in different areas of the city. For example, rentals in popular tourist areas or neighborhoods with high demand may be able to command higher prices, while rentals in less desirable locations may have lower prices. Additionally, the availability of alternative housing options, such as hotels or traditional rentals, may affect the pricing of Airbnb rentals in different areas. Other factors that could impact the impact of machine learning on Airbnb prices in San Francisco include demographic factors, such as the age, income, and education level of potential renters. For example, Airbnb rentals in areas with a high concentration of young, educated, and high-income individuals may be able to command higher prices than rentals in areas with a lower concentration of these characteristics. Finally, external events or circumstances, such as natural disasters, major sporting events, or political conventions, can also affect the demand for and pricing of Airbnb rentals in different areas of the city. Machine learning models that are able to incorporate data on these types of events and circumstances may be able to better reflect the demand for rentals in different areas and adjust pricing accordingly.

Part V: Appendix

	Model	R2	MAE	RMSE
0	Linear Regression	0.419450	0.481840	0.750606
1	CART	0.344662	0.539960	0.797490
2	Random Forest	0.576988	0.372511	0.640721
3	Bagging	0.583957	0.365593	0.635421
4	Boosting	0.351294	0.540061	0.793445
5	Neural Networ	0.485031	0.446664	0.706931
6	KNN	0.392481	0.461617	0.767832