DS 4400: Machine Learning and Data Mining I

Spring 2022

Project Title: New York Airbnb Price predicted in 2019

TA: Xuyang Li

Team Member: Taoli Zhao

Recording of presentation:

https://drive.google.com/file/d/1NAKxvC3MSJfTrUKCLH2ax4SHz44f97L3/view?usp=sharing

Dataset:

DS4400/AB_NYC_2019.csv at main · ztl01/DS4400 (github.com)

The original page: https://www.kaggle.com/dgomonov/new-york-city-airbnb-open-data

Code:

https://drive.google.com/file/d/1uqyXIrMK2UxDrk1-acUqTgxmbf0fYBIX/view?usp=sharing

Problem Description

Along with the development of globalization, more people tend to design a trip during their vacation. According to the number disclosed by the tourism-marketing agency of New York City, the number of visitors to the city was 65.2 million in 2018. In the past nine years, the number was continuously increasing. In other words, tourists were an important income resource for New York City. The statistics informed the public that the house renting industry has a huge marketing size in New York because we know that every visitor needs a place to stay at night.

Nowadays, because of the development of the Internet, people are more likely to make a house appointment before arriving. Based on this phenomenon, many Apps appeared on the market such as Agoda, Vrbo, and HomeAway. Among all the choices, Airbnb is one of the most popular picks by the customers. According to the statistics from the website, Business of Apps, Airbnb had 150 million users, even without updating, in 2018. Moreover, even during the COVID-19 pandemic, there still were 300 million bookings on Airbnb in 2021.

Therefore, the previous data could convey that creating a predictive model based on the Airbnb dataset of New York City in 2019 is a significant topic for people to research. It is possible to argue that working on the dataset before the pandemic is more meaningful than the dataset during the COVID-19 since the pandemic will eliminate in one day. Subsequently, the dataset in 2019 will be a better fit for future situations. The goal of this project is to create a predictive model that could become a helpful tool for both visitors and renters. Visitors could use the model to know the possible prices of the house based on their conditions. Conversely, the renters could set a fair price for their houses with the help of the predictive model. A sensible price would play an essential role in attracting customers.

Dataset and Exploratory Data Analysis

The original dataset includes 16 features, which are the id of house, name of the house, the id of the host, name of the host, neighborhood group, the specific neighborhood, latitude, longitude, room type, minimum nights, number of reviews, date of the last review, review times per month, calculated host listings count, and number of days available in 365 days. Nevertheless, not all of the features are necessary for people to analyze. Before moving to the OneHot Encoder, I dropped the id and name of the house and the host, the date of the last review, and calculated the host listings count. Afterward, the dataset provides eleven possible meaningful features to model to analyze in the further step.

Correlation elucidates the relationship between two variables. If two variables have a high correlation, it means that they are strongly linked together. In machine learning, a higher correlation between two features could cause a problem of an unstable model because a higher correlation means that there are more possibilities to fit the model. As shown in figure 1, the correlation heatmap among features, the highest correlation, 0.55, appears between the total number of reviews and the number of reviews per month. In other words, the heatmap displays the character of the dataset that there is no strong linear relationship among all the possible pairs of the feature, which means that the model could have a stable performance. Furthermore, we also could find that the target of the model, price, does not show strong relationships with all the continuous features at this time since the average absolute correlation value is 0.064. However, since we have not converted categorical features, neighborhood group, detailed neighborhood, and room type, to 0 and 1 format, we could expect to find stronger linear relationships after applying the OneHot encoder.

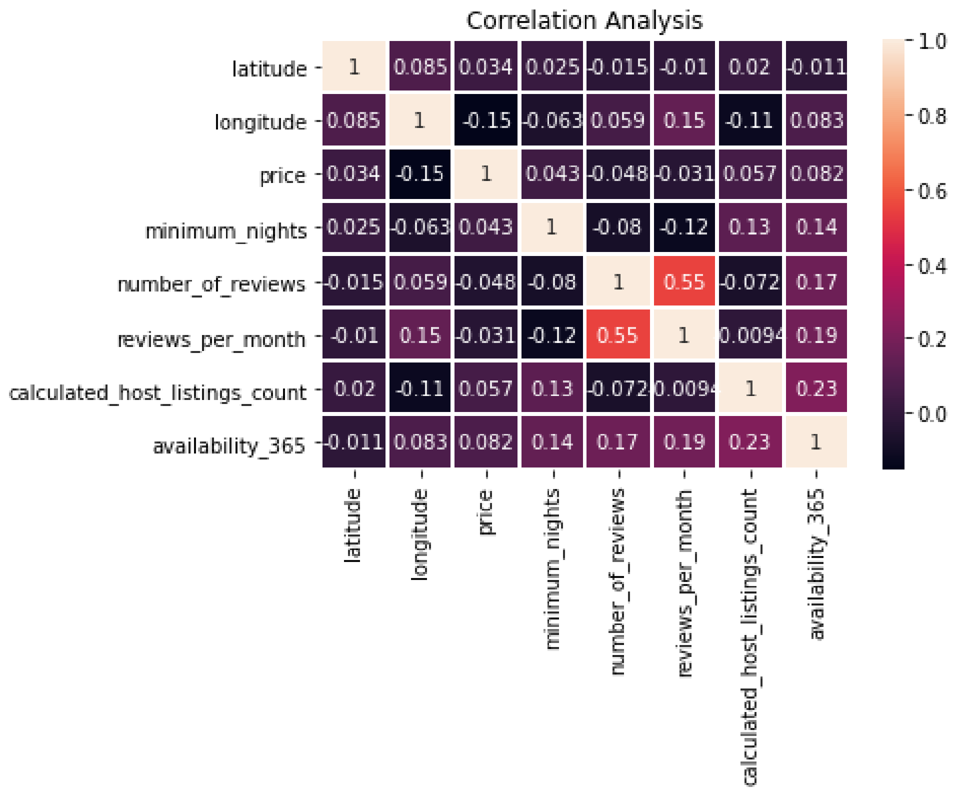


Figure 1: Correlation Heatmap among features before OneHot Encoder

The OneHot Encoder allows people to convert string format categorical data to 0 and 1 format, which helps us to satisfy the requirements of creating regression models. An easier way to understand it is considering1 as yes and 0 as no. Take a feature, neighborhood group, as an example, it turns into five different features after applying the OneHot Encoder and each of them indicates whether the house locates in the neighborhood group. We also applied the same method to neighborhood and room types. Then, the dataset has 224 more features since it has 221 specific neighborhoods and 3 room types. The last change in the dataset is updating the none data of review per month to 0. Consequently, the shape of the final cleaned dataset is 48,895 rows multiplied by 236 columns.

Because the dataset has too many features after applying the OneHot Encoder, it is inefficient to create a heatmap to analyze the correlation as previously. However, we still could find some observations from the data frame function directly. Among five neighborhood groups, the Manhattan group has the highest correlation value with price but still not a strong linear relationship. The visualization in figure 2 that the Manhattan group has the largest number of Airbnb houses could be evidence for this observation. Most of the specific neighborhoods portray a negative linear relationship. The coefficient correlations portray that the entire home/apt and private room, which are two features extracted from room types, could be important features for the regression model since the correlation with the price is much higher than other features.

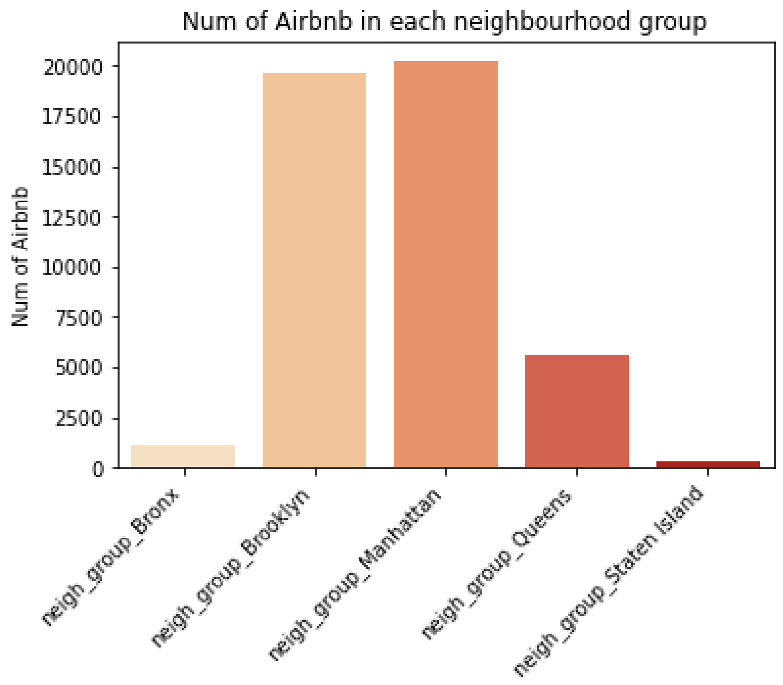


Figure 2: Number of Airbnb in Each Neighborhood Group

The target data, which is the price in this project, is equally salient as the features since it is the goal. Nevertheless, the distribution of the target is salient to consider because it could tell us whether the unnecessary data exists. We should get rid of some extreme cases since they could cause the overfitting of the model. As presented in figure 3, distribution of price with original cleaned data, it is reasonable to submit that some extreme data are unnecessary to be included in the model because their frequency is very close to 0. For this reason, we should only analyze the data of the house whose price was under a specific range. Besides, the standard deviation of the price of the original cleaned data is 240.154 with a 152.72 mean. Hige standard deviation highlights that the data are spreading out. We can roughly see that the meaningful data is in the range of 400 to 600. Since we cannot gather more detailed information from figure 3, we could pick 400 on the first try.

Figure 4 displays the distribution of price data after setting the range as under 400. Even though we still can find some ups and downs in the line, it still could argue that it is a much better distribution than figure 3. This decision also leads the standard deviation down to 76.007, which means that the dataset becomes more centralized. Therefore, we can conclude that 400 is a sensible limitation and should drop the rows whose price is greater than 400 since the visualization is in a nice shape and the meaningful data appears at every point of the range. Afterward, the data size jumps from 48,895 to 46,907.

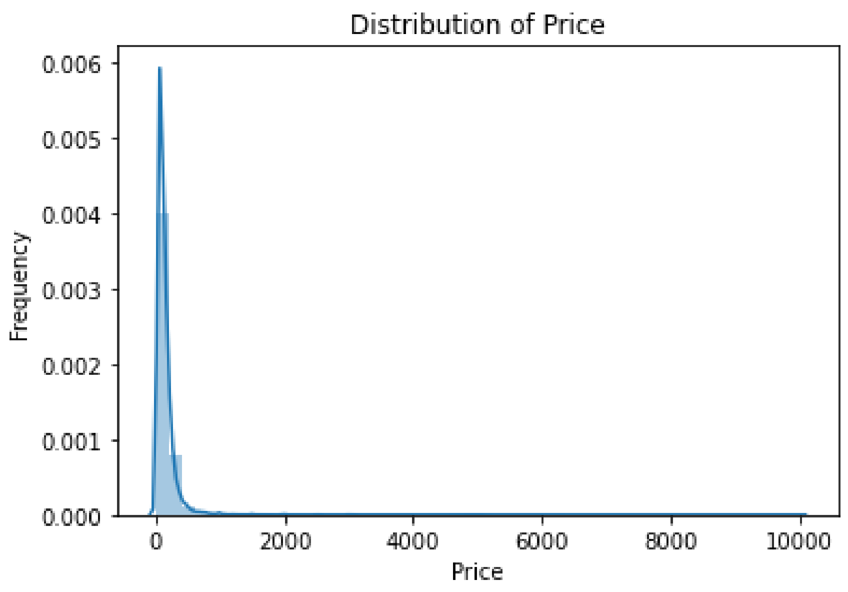


Figure 3: The Distribution of Price with original cleaned data

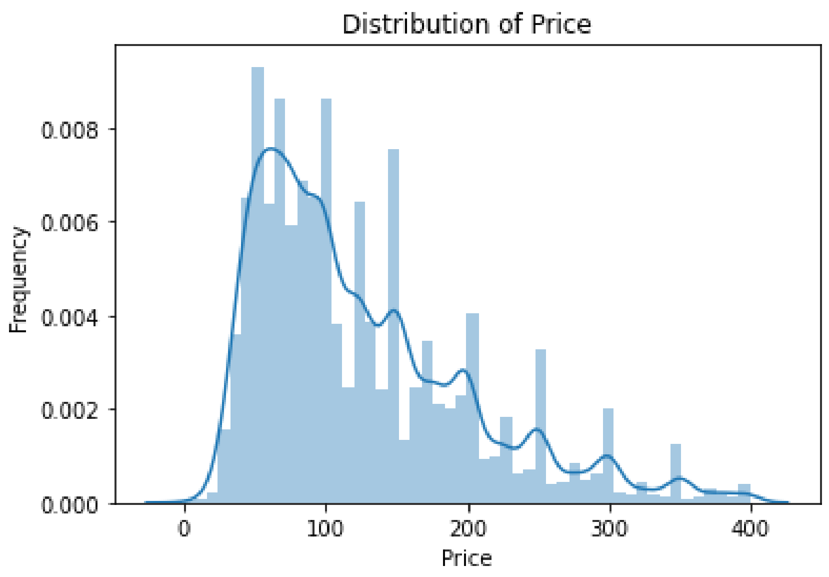


Figure 4: The Distribution of Price with Price Range Under 400

Approach and Methodology

The features of the model are all the columns except price. The predictive target is price. The data was split by train_test_split into 75% and 25%. Moreover, the x_train and x_test are scaled by Standard Scaler. Those are the base for the following machine learning models.

A. Linear Regression, Ridge, and Lasso

The first comparison exists among linear regression, ridge, and lasso. As shown in the following, the R^2 score of the linear regression with the whole cleaned data is around -7, which indicates that some of the data are negatively influencing the learning of the model. In other words, we are supposed to select some important features for the model to reduce the interference.

```
For training data of Linear Regression:
Mean Squared Error: 2876.068358805312
R2 square Score: 0.5045784437506105

For Testing data of Linear Regression:
Mean Squared Error: 4.216428084733843e+25
R2 square Score: -7.395932196765145e+21
```

Before moving into the feature selection, we should compare the results of linear regression with ridge and lasso at first because they have made some adjustments in the base of linear regression. The ridge regression performs linear regression with the regularization with L2. And the lasso regression already utilized the feature selection by setting some coefficients to zero. After tunning the models among the alpha values 0.05, 0.1, 1, 10, and 100, the best performance of the ridge appears when alpha equals 100. For the lasso, the model performs best when alpha equals 0.05. The following data depicts that both the ridge and lasso regressions are better than the original linear regression.

For Ridge Regression

```
The model performance for the training set (alpha = 100)
--------------------------------------
MSE is 2860.3682519420313
R2 is 0.5072828896834323

The model performance for the testing set (alpha = 100)
--------------------------------------
MSE is 2924.0816626957467
R2 is 0.48709406918611853
```

For Lasso Regression

```
The model performance for the training set (alpha = 0.05)
--------------------------------------
MSE is 2861.391958214025
R2 is 0.5071065495930234

The model performance for the testing set (alpha = 0.05)
--------------------------------------
MSE is 2919.503219573257
R2 is 0.4878971625680092
```

The R^2 scores are very close between the lasso and ridge regression model. The score of the lasso of the testing set is slightly higher than the ridge. Besides, the results corroborate the necessity of selecting features on the linear regression model. To achieve the goal, we choose to use the SelectKBest to evaluate the importance of each feature. SelectKBest is an existing function in the sklearn package that selects features according to the k highest scores. We set the parameter k as ‘all’ to go over the score of all the features. After applying the function and transforming the data, one of the observations is that only a few of the features are significant to the model. When we found the importance of each feature, we trained the linear regression model three more times with different lowest importance restrictions, 1, 5, and 10. Afterward, the condition of the best performance of linear regression with selected features is only training the model with features whose importance scores are higher than 1. In other words, the feature selection process helps us to decide 192 most important features. Therefore, even though the score of linear regression still is slightly lower than the ridge and lasso regressions, the performance is much better than the original model, which proves that the previous selection is sufficient.

```
When only using the features whose importance is greater than 1

For training data of Linear Regression:
Mean Squared Error: 2867.014756142649
R2 square Score: 0.5061379859315422

For Testing data of Linear Regression:
Mean Squared Error: 2924.9039400899014
R2 square Score: 0.48694983554257265
```

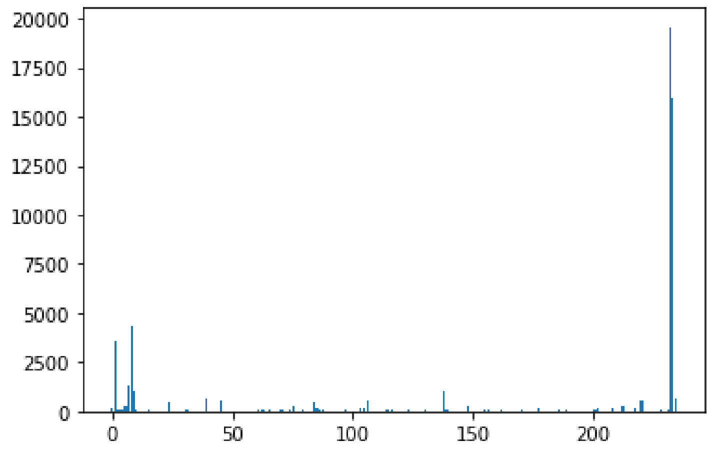


Figure 5: The Importance of Features of the Linear Regression Model

B. Decision Tree Regressor and Random Forest Regressor

The reason the compare the decision tree regressor and random forest regressor is that the decision tree regressor is the base regressor of the random forest regressor. Moreover, we could suspect that the random forest regressor will be a more powerful model than the decision tree since it is an ensemble learning technique. Because of the character of the decision tree regressor, the original model is overfitting. The same issue equally happens in the random forest tree model. The large gap in the R^2 score between the training and testing set contends that we need to limit the max depth of the model and select pertinent features.

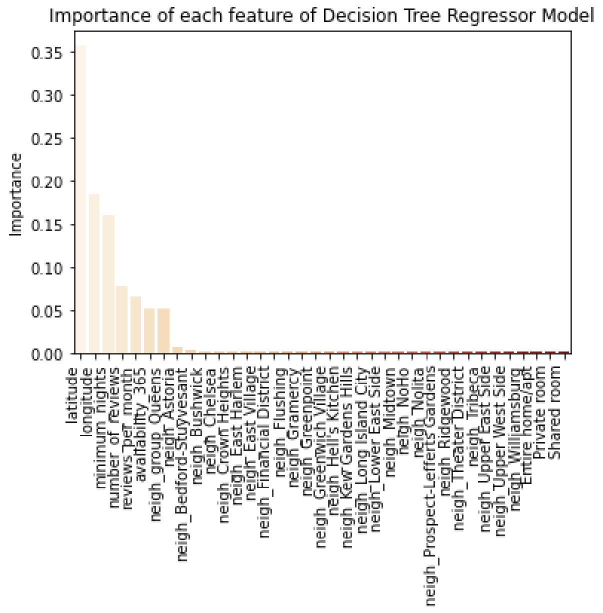
The original Decision Tree Model Performance

```
For training data of Decision Tree Regressor:
Mean Squared Error: 0.0
R2 square Score: 1.0

For Testing data of Decision Tree Regressor:
Mean Squared Error: 5084.565191438561
R2 square Score: 0.11242669591703314
```

The original Random Forest Regressor Model Performance

```
For Training set of Random Forest Regressor:
Mean Squared Error: 366.2656790535716
R2 square Score: 0.9359750972791295

For Testing set of Random Forest Regressor:
Mean Squared Error: 2752.2905121902972
R2 square Score: 0.5378199677275428
```

After tunning the model, the optimal value of the max_depth parameter of both models is 7. The random forest regressor also needs to tune the n_estimator parameter and the best value is 150. Figures 6 and 7 displays the feature importance of the decision tree regressor and the random forest regressor. We can find that both models have the same top 7 features. Specific neighborhoods are more important in the decision tree regressor than the rest features. Nevertheless, neighborhood groups are more essential for the random forest regressor model.



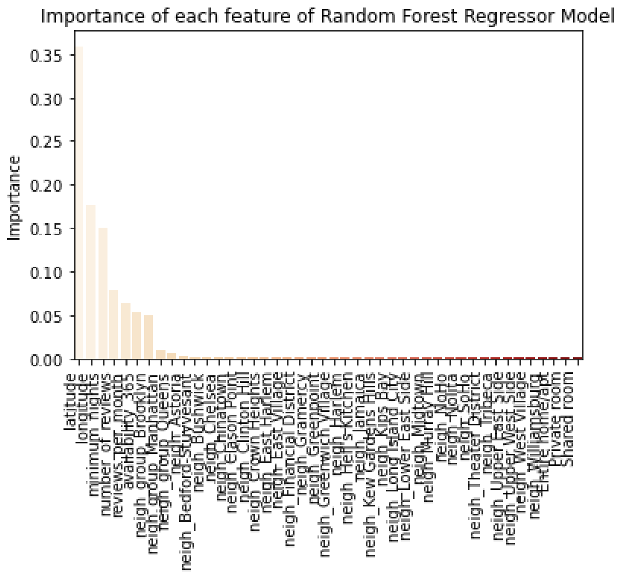


Figure 6: Feature Importance of Decision Tree Regressor (Left)

Figure 7: Feature Importance of Random Forest Regressor (Right)

Compared to the original decision tree regressor and the random forest regressor, the optimized models conveys the results with less overfitting. The R^2 and MSE scores contend that the random forest regressor is the better model compared to the decision tree regressor.

Optimized Decision Tree Regressor

```
For training data of Decision Tree Regressor:
Mean Squared Error: 2690.074729206404
R2 square Score: 0.5335694209635925

For Testing data of Decision Tree Regressor:
Mean Squared Error: 2835.8183084467064
```

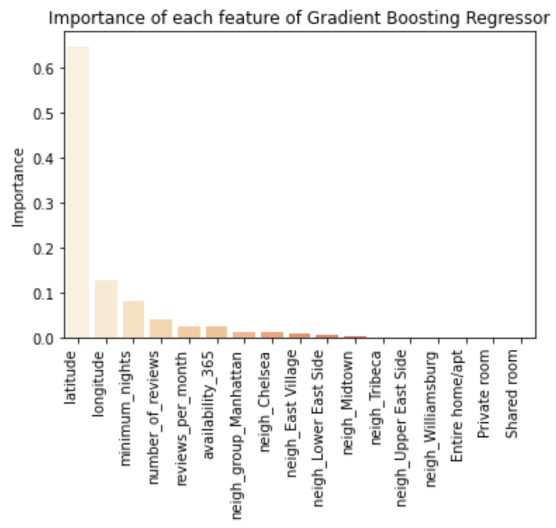
```
R2 square Score: 0.5123237788033089
```

Optimized Random Forest Regressor

```
For training data of Random Forest Regressor:
Mean Squared Error: 2520.337976420043
R2 square Score: 0.5594334850566032

For Testing data of Random Forest Regressor:
Mean Squared Error: 2853.0184838091727
R2 square Score: 0.5209051627796805
```

C. Gradient Boosting Regressor

The Gradient Boosting Regressor is another ensemble learning technique. The exploring process of this model elucidates that the best combination of learning rate and n_estimators is 0.1 and 100. Besides, there is no need to adjust the max_depth parameter. Furthermore, figure 8 contends that the gradient boosting regressor has the fewest important features. It only has 17 selected features whose importance is greater than 0.



```
When learning_rate = 0.1 ,
n_estimators = 100
For training data of Gradient Boosting
Regressor:
Mean Squared Error: 2603.6113505668714
R2 square Score: 0.5466646184017651

For Testing data of Gradient Boosting R
egressor:
Mean Squared Error: 2711.316094553206
R2 square Score: 0.5394602407729334
```

Figure 8:Feature Importance of Gradient Boosting Regressor

D. MLP Regressor – Neural Network

According to the official explanation, the MLP regressor “optimizes the squared error using LBFGS or stochastic gradient descent”. The original performance of this model also displays a few overfitting but not as serious as the decision tree regressor and the random forest regressor. The tunning process of the MLP regressor happens in the parameters of hidden layer size and learning rate. Following scores highlight that the model performs great but with the highest overfitting level compared to other optimized models.

```
When hidden_layer_size = 100 learning_rate = 0.05
For training data of MLP:
Mean Squared Error: 2669.437977470269
R2 square Score: 0.5401718762836032
```

```
For Testing data of MLP:
Mean Squared Error: 2806.7548129812453
R2 square Score: 0.5076740816495181
```

Discussion and Result Interpretation

Compared to other regression models, the entire home/apt, private room, and shared room, which are the room types in the original dataset are the most significant features of the linear regression model. On the contrary, the continuous features are more salient in other regression models. The rank of neighborhoods and neighborhood groups are various in every model. We cannot decide the rank of feature importance of the MLP regressor until applying an appropriate method.

The exploring process corroborates that there are serval possible reasons for the models making errors. Firstly, the noisy features could lead to errors in the model. As demonstrated by the previous scores, all the scores with selected features are better than the original scores. Secondly, the value of parameters could essentially influence the performance of models. In other words, tunning the models and finding the appropriate parameters is an indispensable step in creating an accurate model. Besides, the first and the second points also decrease the overfitting level of models. Overfitting is a common but paramount problem that happens during machine learning. One way to evaluate whether the model is overfitting is by calculating the difference in the metrics between the training and testing sets. Thus, we should adjust the model based on the metrics, which are R^2 and MSE in this project.

Even though the project has explored many aspects of the dataset, this project still is a challenging task. As shown above, the highest R^2 score is 0.5395 for the gradient boosting regressor, which indicates that the model has many possibilities to enhance its performance. Besides, it is difficult to analyze some of the results since the models have too many features after applying the OneHot Encoder. For example, people cannot analyze the heatmap of the cleaned dataset because there are over 400,000 rectangles in the visualization. Consequently, these reasons make this project more challenging.

Conclusion

Overall, the best-fitted model for this project is Gradient Boosting Regressor. Even though the R^2 scores of the testing set among the decision tree regressor, random forest regressor, and gradient boosting regressor are in a similar range, the gradient boosting regressor has the smallest R^2 score gap between the training and testing set. However, it cannot deny that the project needs much further work. Firstly, as discussed above, both correlations and heatmap depict that current features do not show a strong linear relationship with price. To solve this issue, we need to gather more information about the data and find more meaningful features for the prediction. Secondly, the main reason for having too many features is there are too many specific neighborhoods after applying the OneHot Encoder. In the further step, we should come up with a solution to reduce the number of features like the neighborhoods that only have a few Airbnb houses. Lastly, the project will play a pertinent role in real-life situations when we could address all the mentioned issues because the topic closely relates to the daily life of the public.

Team member contribution

The code and report are written by Taoli Zhao individually with the help of the teaching assistant, Xuyang Li, and the professor, Alina Oprea.

References

1. N.Y. Draws a Record 65 Million Tourists (in Spite of Trump's Trade War, Many Were Chinese) - The New York Times (nytimes.com)
2. Airbnb Revenue and Usage Statistics (2022) - Business of Apps
3. How to Perform Feature Selection for Regression Data (machinelearningmastery.com)
4. Linear Regression vs Ridge Regression vs Lasso Regression | by Carla Martins | MLearning.ai | Medium
5. sklearn.feature_selection.SelectKBest — scikit-learn 1.0.2 documentation
6. sklearn.neural_network.MLPRegressor — scikit-learn 1.0.2 documentation