University of Toronto

Faculty of Applied Science and Engineering

MIE 368 Final Report

Prediction of Post-Covid-19 Airbnb Rental Prices in Toronto

Team 1

Zhen Yang	1003866364
Yifei Liu	1003864231
Haolin Wu	1003189934
Zijian Wang	1004154960

1. Introduction

From June 4, as the Ontario government lifted restrictions and reopened the short-term rentals, many Airbnb hosts noticed a post-Covid-19 uptick in bookings, and most of them hope guests can book their listings this summer [1]. However, it's difficult for Airbnb hosts to decide on a reasonable price for their properties. Under the impact of Covid-19, the Airbnb market is not stable, self-researching of similar properties nearby is time-consuming, and every host wants to set the price as high as possible to cover their loss. For Airbnb hosts who decide the price on their own, the price could be biased and subjective. To help the hosts set the prices more objectively and effectively, the team builds a reliable tool that predicts a reasonable price under the impact of Covid-19.

2. Data

The team collects data from two public data sources. One is the Inside Airbnb [2], containing 12 months of data from October 2019 to September 2020. There are 74 attributes, including listed housing's physical location (i.e. 'latitude', 'longitude', 'neighborhood'), house types (i.e. 'accommodates', 'room type', 'number of beds'), and reviews of listings (i.e. 'number of reviews', 'average ratings', 'comments'). The other data source is the City of Toronto COVID-19 Data Dashboard [3]. The data includes detailed cumulative and monthly cases for Toronto and daily cases by neighborhood. By matching the neighborhood name, the team constructs our raw dataset with 76 columns and 83344 rows.

Data Cleaning

The team first removes irrelevant columns (i.e. 'website URL', 'host ID', 'listing ID', 'host name') and those cannot be converted into numerical or categorical format. In addition, we discover and delete 17348 rows that have one or more missing values. Since the missing feature is 'Accommodation' which has a large range and varies for every listing, we decide to not apply the imputation method here.

COVID-19 Neighborhood Cases Data

Since the Airbnb Listing data we select ranges from October 2019 to September 2020, our team first tried to find the neighbourhood COVID-19 cases during this period. However, the Toronto COVID-19 Data Dashboard only keeps the latest information. To accommodate this, the team creates a new feature called Neighbourhood Cases Ratio (NCR). We calculate NCR for each neighbourhood using Sep 20th data.

NCR formula:

$$Neighbourhood\ Case\ Ratio\ = \frac{Neighbourhood\ Cases\ Number\ as\ of\ Sep\ 20th}{City\ of\ Toronto\ Cases\ Number\ as\ of\ Sep\ 20th}$$

Feature Selection

Exploratory Data Analysis (EDA) is applied on the dataset to find the most important features. We first check the correlation between the remaining features and find a few groups are highly correlated. For example, 'the number of beds', 'number of bedrooms' and 'accommodation' that are highly correlated. We choose to keep the 'accommodation' column and delete the other two as 'accommodation' is the most accurate representation.

Data Normalization

The team plots all the numerical features in a histogram and discovers that most of the distributions are skewed (see Appendix A for details). This might cause overfitting on the model, so the team uses log transformation on right-skewed features and square root transformation on left-skewed features. Figure 2.1 and 2.2 shows the Price feature after normalization.

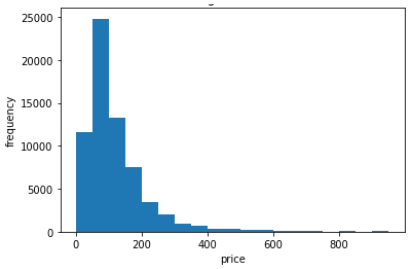

Figure 2.1 Distribution of Price

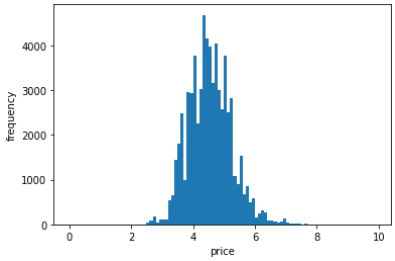

Figure 2.2 Distribution of Price (log-transformed)

One-hot Encoding

In the dataset, there are two categorical features, 'property_type', and 'room_type'. In order to input them into the model, we decide to transform them into numerical values. Among the methods dealing with categorical variables, we choose to use dummy variables for different classes. Integer Encoding is not chosen since it will order the features alphabetically while these two do not have rankings. As a result, we create one dummy variable for all the unique types and use binary values to indicate which type that row falls into.

For example, the room type for the first listing is "Entire home/apt", and room type for the second listing is "Private room". For both the first and second listing, the property type is not Villa.

property_type_Villa	room_type_Entire home/apt	room_type_Hotel room	room_type_Private room
0	1	0	0
0	0	0	1
0	0	0	1
0	0	0	1

Figure 2-3: Sample of dataset after one-hot encoding

In the end, we obtained a clean Airbnb listing dataset including 65996 rows and 49 columns. The final features are 'latitude', 'longitude', 'number of accommodation', 'minimum_nights', 'maximum_nights', 'availablility_365', 'number_of_reviews', 'review_scores_ratings', 'room_type', 'property_type', 'price', and 'number of neighbourhood period cases of the listings'.

3. Methods

Figure 3-1 shows our methodology's process flow from data processing to model comparison and selection. Method 1 is a regression model and generates a predicted price.

In addition, the team develops Method 2. Since the target variable "price" has a large variance from $12 to $13,000. We first use a classification model to classify the input into price bins to present three Airbnb categories: luxury, comfort and economical. Then we run a regression model in each price bin to predict an exact price. The team hypothesizes that the regression model with price bins can predict the prices more accurately given a smaller price range.

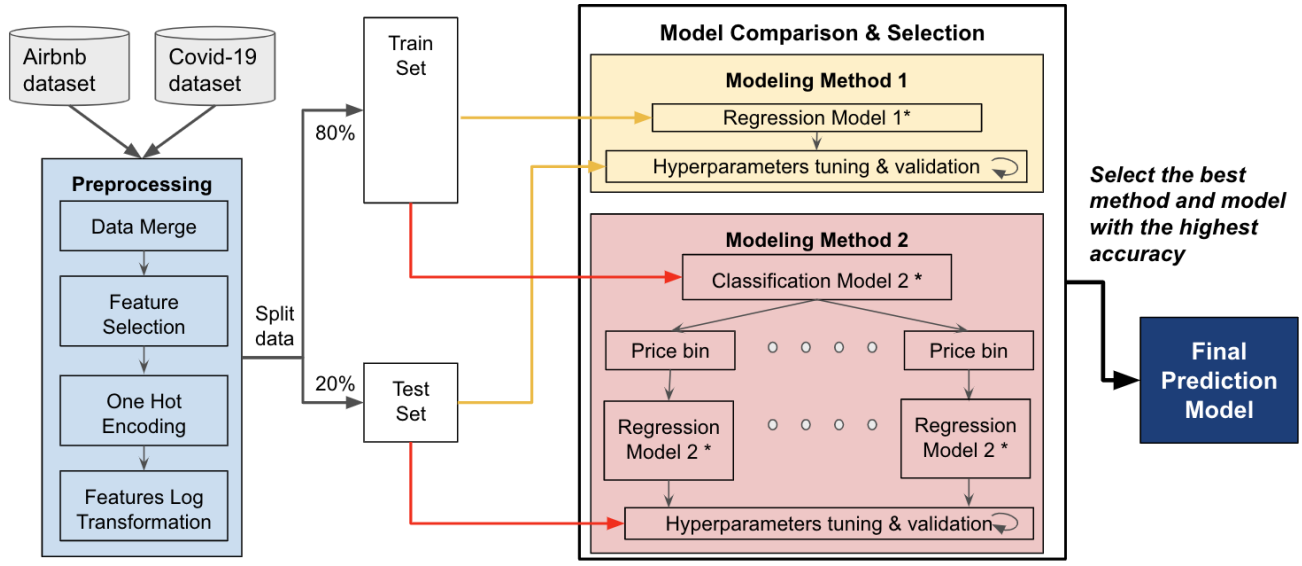

Figure 3-1. Flow Chart of Method

The team defines price bins and outliers as shown in Table 3-2. Only 1.7% of data has a price above $500. Thus, we consider them as outliers and exclude them from the model. For the remaining 98.3% of data with prices between $0 and $500, we separate them into three price bins of $0-80, $80-130 and $130-500 where each price bin contains around 30% of the data. It ensures that there is enough data for each price bin to train a regression model.

Table 3-2. Details of Price Bins and Outliers

Price Bin #	Price Range in Dollars	Number of Listings	Percentage of Data
1	[0, 80)	21506	32.7%
2	[80, 130)	24021	36.4%
3	[130, 500)	19401	29.4%
Outliers	[500, 13000]	1068	1.7%

For both methods, the team compares different classification models (Logistic, Random Forest, Vote), and regression models (Linear, Lasso, Ridge, and XGB), then chooses the best-performing one.

The XGB regression stands for eXtreme Gradient Boosting that is designed to work with numeric values only, such as prices. The team chooses XGB Regression due to its superior performance, as observed in Kaggle competitions. XGB regression uses an ensemble technique where new models are created and predict the errors of prior models and then added together to make the final prediction [4]. It is likely to provide good accuracy and fast computational speed.

After applying a regression model in Method 1, the team realizes a common issue of overfitting. It is because too many variables are included in the model, and the model appears to fit well to the train set (train score = 0.934) and fit poorly to the test set (test score = 0.608). To address the overfitting, the team uses the *.feature_importances_* function to check how each feature affects the model. The outputs of the first 22 features and their importance scores after sorting are shown in Figure 3-3. The higher the score, the more important the feature. The team decides to only keep the top 14 features (circled with the red square in Figure 3-3) with the importance score at or higher than 0.01. After the second round of feature selection, the overfitting issue is mitigated, and the test score increases by around 34%.

```
Variable: host_total_listings_count Importance: 0.18
Variable: latitude             Importance: 0.17
Variable: longitude            Importance: 0.15
Variable: number_of_reviews    Importance: 0.14
Variable: maximum_nights       Importance: 0.07
Variable: Cumulative cases per neighbourhood Importance: 0.05
Variable: availability_365     Importance: 0.05
Variable: minimum_nights       Importance: 0.04
Variable: date                 Importance: 0.03
Variable: accommodates         Importance: 0.03
Variable: property_type_Villa  Importance: 0.03
Variable: room_type_Shared room Importance: 0.02
Variable: property_type_Castle Importance: 0.01
Variable: property_type_Hotel  Importance: 0.01
Variable: review_scores_rating Importance: 0.0
Variable: property_type_Aparthotel Importance: 0.0
Variable: property_type_Apartment Importance: 0.0
Variable: property_type_Barn   Importance: 0.0
Variable: property_type_Bed and breakfast Importance: 0.0
Variable: property_type_Boutique hotel Importance: 0.0
Variable: property_type_Bungalow Importance: 0.0
Variable: property_type_Cabin  Importance: 0.0
```

Figure 3-3. Feature Importances Outputs

Finally, we apply the grid search on each model for parameter tuning. The team tests seven parameter values for both Lasso and Ridge models and two-parameter values for XGB regression's five parameters. The modified parameters can be found in Appendix B.

4. Results

The results of different models used in Method 1 and Method 2 are summarized in the following table. The Root Mean Square Error (RMSE) is used to evaluate the regression models. The test accuracy is used to evaluate the classification model. It is noteworthy that, in Method 2, after the team determines the best-performing classifier as Random Forest, then we run different regression models including the predicted price bin information.

Regarding the regression models, in both Method 1 and 2, the team found that the Linear, Lasso and Ridge regression performs similarly, and they are much better than XGB regression's performance. Indeed, Method 2 with price bins performs generally better than Method 1. The models' RMSEs using

Method 1 are above 1 while the model's RMSEs using Method 2 are below 0.5. It proves the team's hypothesis that using a classification model prior to a regression model can effectively lower the prediction error.

	Train RMSE	Test RMSE
Method 1: Regression		
Linear Regression	1.096	1.103
Lasso Regression	1.096	1.098
Ridge Regression	1.096	1.097
XGB Regressor	1.424	1.425

Figure 4-1. Results of Method 1

	Train RMSE	Test RMSE	Test Accuracy (Classification Only)
Method 2 Part 1: Classification Model			
Logistics Regression	N/A	N/A	0.67
Random Forest Classifier	N/A	N/A	0.74
Vote Classifier	N/A	N/A	0.70
Method 2 Part 2: Regression model for each price bin			
Linear Regression	Bin1: 0.076 Bin2: 0.213 Bin3: 0.441	Bin1: 0.081 Bin2: 0.263 Bin3: 0.435	N/A
Lasso Regression	Bin1: 0.083 Bin2: 0.109 Bin3: 0.457	Bin1: 0.092 Bin2: 0.137 Bin3: 0.459	N/A
Ridge Regression	Bin1: 0.076 Bin2: 0.213 Bin3: 0.441	Bin1: 0.0814 Bin2: 0.263 Bin3: 0.445	N/A
XGB Regression	Bin1: 2.057 Bin2: 1.774 Bin3: 1.366	Bin1: 2.051 Bin2: 1.751 Bin3: 1.390	N/A

Figure 4-2. Results of Method 2

5. Discussion

5.1 The Result

The team is satisfied with our model's test accuracy, but we still want to validate the model's performance on new data. We do so by inputting ten sets of listings on October 9 and comparing the outcome prices from our prediction model and the actual listing prices on Airbnb for clear comparison. Four out of the ten results are shown in Table 5-1.

Table 5-1. Evaluation of the Model on New Inputs

Date	Neighbourhood	...	Predicted	Actual	Difference (Predicted - actual)	Error (difference /actual)
2020-10-09	Kensington-Chinatown	...	$65	$59	$6	10.2%
2020-10-09	Palmerston-Little Italy	...	$763	$599	$164	27.4%
2020-10-09	South Riverdale	...	$221	$240	$19	7.9%
2020-10-09	Trinity-Bellwoods	...	$82	$79	$3	3.8%

5.2 The Error

As shown in Table 5-1, our model can precisely predict the rental price below $500, but it cannot accurately portray the rental prices over $500 due to the lack of a corresponding price bin.

For listings with prices under $500, the average error is around 10%. One underlying reason may come from the Neighbourhood Case Ratio we manually created. As we discussed in section 3.0, we used NCR to represent monthly cases by neighbourhoods since accurate data cannot be found.

To justify the NCR's effectiveness, we generate the Partial Dependent Plot (PDP) for the predicted price and the neighbour covid case number to see if the model's prediction matches the reality. The PDP plot shows the marginal effect of cumulative cases on listing price. As the cumulative cases number increases, its effect on listing price becomes greater. And It matches the actual correlation.

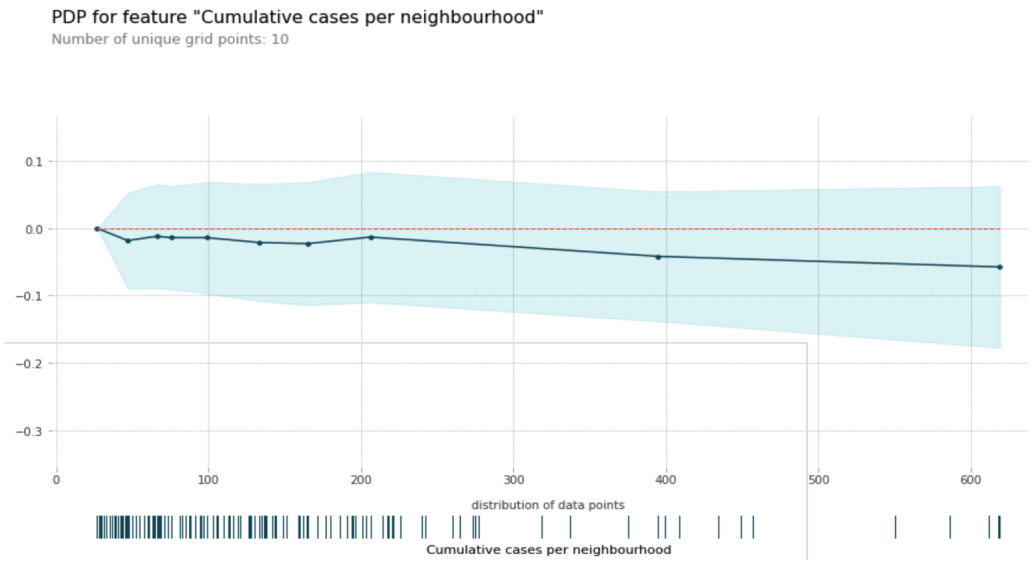

Figure 5-2. Partial Dependence Plot for feature "Cumulative cases per neighbourhood"

Interestingly, we find a report showing that the City of Toronto Neighbourhood Cases Dashboard did not actually include all the cases since some patients didn't input their locations [6]. Therefore, this could be one of the reasons that causes error for the prediction.

5.3 The Method

From a method perspective, we tried Ridge, Linear, Lasso and XGB regression for the project. Here, we explore the potential reasons that may cause the performance difference.

Both Ridge and Lasso regression are variants of Linear regression. Ridge regression performs L2 regularization and does not necessarily push the coefficients to zero. Therefore, Ridge is useful in reducing or preventing overfitting but may not help with feature selection. In comparison, Lasso performs L1 regularization and can push the coefficients to zero. Ridge is computationally faster than Lasso, but Lasso has the advantage of ultimately reducing unnecessary parameters in the model.

Lasso tends to do well if there are a small number of significant parameters and the others are close to zero. Ridge works well if there are many large parameters of about the same value [7]. In our case, we have already done two rounds of feature selection in the EDA so that most features are significant. Therefore, Ridge regression outperforms in our case.

Comparing the Ridge and XGB regression, although XGB regression performs very well in Kaggle competitions, it does not stand out in our project. The reason could be that XGB regression's strength to handle missing values cannot fit our model. The XGB regression adds the missing values in a branch in which it could minimize the loss. However, in our EDA process, the data with missing values have already been discarded. Therefore, XGB regression loses its advantage.

6. Conclusion

Due to COVID-19 pandemic, the Airbnb rental industry has been affected significantly, making it harder for the hosts to decide a fair price. We set out to develop a price prediction tool for the hosts to solve this problem, given their listing information.

For model implementation, we tried regression and regression combined with classification price bin methods. From the result, we conclude that the latter performs much better.

The final proposed tool takes the 'number of total listings', 'location of listing', 'maximum nights', 'availability in a year', 'minimum nights', 'accommodations' and 'property type' as input resources by users and returns a recommended price. Inside the tool, a classification model is first used to classify the input property into one of the price bins: economical, comfortable or luxury properties. Then the regression models are applied in each price bin to predict the price. We find that the Random Forest classifier and a Ridge regression model give the best result. The model provides a final price prediction result within $10 error for over 80% of listings.

7. Future Direction

After identifying the model drawbacks that we discussed in section 5.0, the next step for this tool would be to add more Airbnb property datasets to cover the listing price over $500. For the COVID-19 related missing data, we plan to contact the City of Toronto for the official historical neighbourhood data. Furthermore, we would also like to expand the dataset by including other city's listing information.

Another limitation of our tool is that the model does not take into account rent pricing strategies. For instance, new Airbnb hosts usually tend to offer a lower price to attract guests [8]. In the future, relevant algorithms will be researched and added. Finally, we could implement a front-end interface for our model. We will have current hosts to validate our price suggestions and determine the additional functionality that would be ideal for commercial use.

8. Reference

[1]L. Lane, "How Bad Are Covid-19 Pandemic Effects On Airbnb Guests, Hosts?", *Forbes*, 2020. [Online]. Available: https://www.forbes.com/sites/lealane/2020/06/09/how-bad-are-covid-19-pandemic-effects-on-airbnb-guests-hosts/?sh=1783e6727432. [Accessed: 25- Nov- 2020].

[2] “Inside Airbnb. Adding data to the debate.,” Inside Airbnb. [Online]. Available: http://insideairbnb.com/index.html. [Accessed: 08-Oct-2020].

[3]C. Toronto, C. News and C. Toronto, "COVID-19: Status of Cases in Toronto", *City of Toronto*, 2020. [Online]. Available: https://www.toronto.ca/home/covid-19/covid-19-latest-city-of-toronto-news/covid-19-status-of-cases-in-toronto/. [Accessed: 08- Dec- 2020].

[4]J. Brownlee, "A Gentle Introduction to XGBoost for Applied Machine Learning", *Machine Learning Mastery*, 2020. [Online]. Available: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/. [Accessed: 08- Dec- 2020].

[5]"Voting Classifier", *Medium*, 2020. [Online]. Available: https://medium.com/@sanchitamangale12/voting-classifier-1be10db6d7a5#:~:text=A%20voting%20classifier%20can%20be,winning%20one%20during%20a%20prediction. [Accessed: 08- Dec- 2020].

[6]"Toronto COVID-19 hot spots map missing 2,000 confirmed cases | CBC News", *CBC*, 2020. [Online]. Available: https://www.cbc.ca/news/canada/toronto/covid-19-data-toronto-public-health-hot-spots-1.5598844. [Accessed: 08- Dec- 2020].

[7]“Ridge-lasso-elastic-net”. [Online]. Available: https://www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net. [Accessed: 08- Dec- 2020].

[8]"A Step-by-Step Guide to Pricing Your Place on Airbnb - Padlifter", *Padlifter*, 2020. [Online]. Available: https://padlifter.com/free-tips-and-resources/pricing/a-step-by-step-guide-to-pricing-your-place-on-airbnb/. [Accessed: 04- Dec- 2020].

Appendix A. Original feature before transformation

	Features	Feature Type	Data Range	Distribution skewness	Data range that more than 80% falls in
1	Price	Numerical	[12, 13000]	Right-skewed	[12, 200]
2	Host-total-listings-count	Numerical	[0, 272]	Right-skewed	[0, 25]
3	Latitude	Numerical	[43.58761, 43.83468]	Slightly right-skewed	[43.65, 43.75]
4	Longitude	Numerical	[-79.63042, -79,12781]	Equally distributed	[-79.5, -79.3]
7	Accommodates	Numerical	[1, 18]	Right-skewed	[1, 6]
8	Minimum-nights	Numerical	[1, 365]	Right-skewed	[1,10]
9	Maximum-nights	Numerical	[1, 365]	Right-skewed	[1, 100]
10	Number-of-days-available-within-365-days	Numerical	[1, 365]	Right-skewed	[1, 100]
11	Number-of-reviews	Numerical	[0, 823]	Right-skewed	[0, 50]
12	Overall-review-score	Numerical	[0, 100]	Left-skewed	[80,100]
13	Cumulative-cases-per-neighborhood	Numerical	[27, 619]	Right-skewed and	[27, 200]
14	Neighbourhood-monthly-new-cases	Numerical	[0, 143]	Right-skewed	[0, 25]
15	Property-type	Categorical	29 categories	Unevenly distributed	[‘condominium’, ‘apartment’, ‘house’]
16	Room-type	Categorical	4 categories	Unevenly distributed	[‘entire home/apt’, ‘private room’]

Appendix B. Code for Grid Search

```
if model == "xgb":
  params_to_search = {
      'max_depth': [0,1],
      'n_estimators': [1,5],
      'min_child_weight': [1],
      'reg_alpha': [0.1],
      'random_state' : [1,2]
      }
  mdl = xgboost.XGBRegressor()
  optimized = GridSearchCV(mdl,
                  param_grid= params_to_search,
                  verbose=False)
elif model == "LinearRegression":
    # LinearRegression No parameter
    optimized = LinearRegression()
elif model == "Lasso":
    alpha = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
    param_grid = dict(alpha=alpha)
    mdl = Lasso(alpha=1)
    optimized = GridSearchCV(mdl, param_grid=param_grid, scoring='r2', verbose=1, n_jobs=-1)
elif model == "Ridge":
    alpha = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
    param_grid = dict(alpha=alpha)
    mdl = Ridge(alpha=1)
    optimized = GridSearchCV(mdl, param_grid=param_grid, scoring='r2', verbose=1, n_jobs=-1)
```