COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

COMS4995 SEC-013: Applied Machine Learning, Fall 2022

PROJECT REPORT

AirBnb Price Prediction

Group 6
Swati Bararia (sb4700)
Xindi Deng (xd2287)
Morris Hsieh (ch3698)
Vijit Singh (vs2817)
Peiyu You (py2277)

Instructed by
Dr. Sunder Pappu

Submission Date
December 5, 2022

1 Introduction

The number of active rentals listed was about 31,260 in Quarter 2, 2022, which has a 2% quarterly growth compared with the last period. In order to benefit both customers and AirBnb house owners in New York City, we build models to help customers look for the best deals by predicting the price of a house based on different criteria. Also, these models are used to determine what price is reasonable for their listing in order for owners to get better reviews and earn maximum profit.

2 Data Exploration and Visualization

2.1 Dataset

The dataset of our project is Inside AirBnB: NYC from Kaggle[1]. The main dataset *listings2* contains 37,410 rows of samples and 74 columns of features, which includes two kinds of information: the housing part (e.g., room type, etc) and the host part (e.g., response time, etc). In view of data type, the dataset includes categorical features, numerical variables as well as descriptive variables.

2.2 Insights from Data Exploration

From Figure 1, we found that the distributions of some variables are skewed, so we need to do the data transformation in data preprocessing. From Figure 2, we found that there exist many outliers and we need to remove outliers to make results become statistically significant. From the distribution of the target according to different categorical variables, we found that *Manhattan* and *Hotel rooms* have the highest range of prices. The dataset has missing values and high-correlation features, so we need to handle these problems in the data preprocessing step.

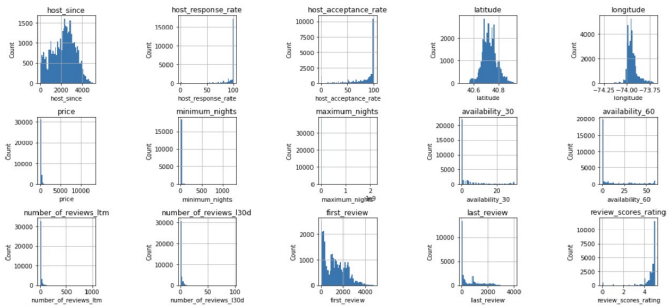

Figure 1: Histograms of Numerical Variables

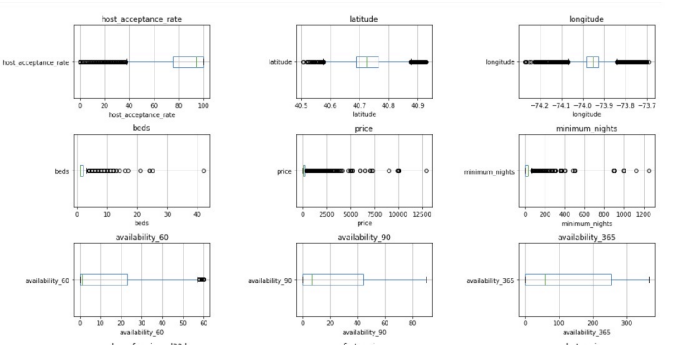

Figure 2: Boxplots of Categorical Variables

[1]https://kaggle.com/datasets/dominoweir/inside-airbnb-nyc

3 Data Preprocessing

3.1 Handling Missing Values

From the initial exploration of our dataset, we found that 8 numerical features and 6 categorical features had missing values. For most of the numerical features, we used KNN imputer to impute the missing values by their neighbors. For others, we imputed zero, for those missing values were collected because of the inexistence of those samples. For categorical features, *neighborhood* had over 40% missing values, so we dropped that column because it was bad to let the model learn such a heavily imputed feature. For other categorical features, since the percentage of missing values was very low, we simply impute with the mode.

3.2 Removing Outliers

We used IQR (Interquartile Range) to detect outliers. The sample with the price on either side of $1.5 \cdot \text{IQR}$ was treated as the outlier. We removed outliers, which can help to build a more generalized and robust model.

3.3 Scaling Data

The data distribution of most features is skewed, thus we used a standard scaler to normalize the numerical features. Since the distribution of price is highly right-skewed, we took log transformation on it to improve the prediction performance.

3.4 Feature Encoding

For 8 features with boolean values, we used ordinary encoding since it only has 2 categories. For 1 feature with 3 categories, we used one-hot encoding since the categories have no order. For the remaining 4 features, we used target encoding, because the categories have no order and the number of categories is over 20.

4 Feature Engineering

4.1 Extracting Features

In the dataset, we also had descriptive features. To extract the information from the descriptive feature *amenities*, we calculated the number of amenities in each sample and took it as a new feature.

4.2 Dropping Features

We dropped features in different stages with different methods. After importing the dataset, we first selected 49 features from 74 features according to common sense. At the stage of correlation exploration, Figure 3, the correlation heatmap, illustrated how numerical features were related to each other. To reduce the dimensionality of the dataset, we dropped 2 features with ≥ 0.09 correlation with each other and dropped 9 features with ≤ 0.02 correlation with the target. At the stage of handling missing values, we dropped 1 feature with over 40% missing values for it was bad to let the model learn such a heavily imputed feature.

5 Model

In the model section, we have three main approaches including regularization regression models, bagging and boosting models, and neural networks models. For regularization regression models, we have fine-tuned ElasticNet which also includes both L1 (Lasso) and L2 (Ridge) regularization. For bagging and boosting models, we start with a simple regression tree with pruning. Then, we fine-tuned Random Forest, AdaBoost, and XGBoosting. Last, we build a neural network model to predict house pricing. In total, we utilize seven models to experiment with which

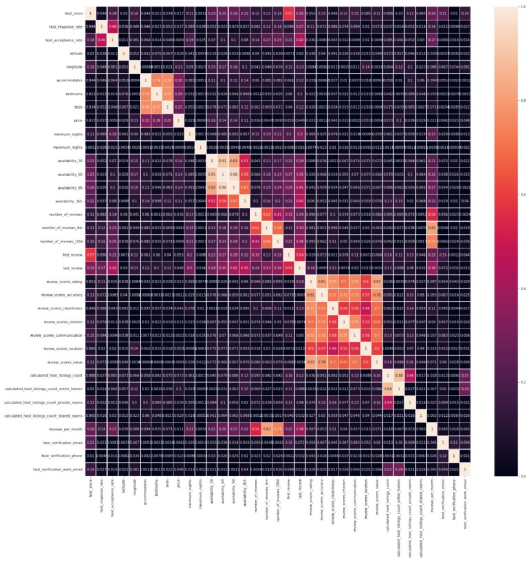

Figure 3: Correlation Heatmap of features

model best fits the pricing problem. For each trained model, except neural networks, we used the cross-validation technique (cv = 5) and the grid search to find the optimal hyperparameters for our models. Last, the metrics for model comparison is Root Mean Square Log Loss, abbreviated as RMSLE, which is calculated on testing set with original price level rather than logarithm scale.

5.1 Model Evaluation

5.1.1 Linear Regression

We used a default linear regression model as our baseline. The RMSLE is 0.460 on testing set.

5.1.2 Elastic Net

For the elastic net model, we chose to tune hyperparameters: L1 ratio and alpha. The best values for these hyperparameters were 0.0 and 0.0 and 0.0215, respectively. The RMSLE is 0.459 on testing set which is slightly better than the default linear regression model. Figure 4 differs from baseline linear regression and can be a better model to select because it gives achieves better feature selection.

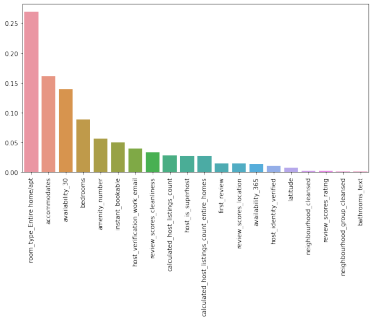

Figure 4: Elastic Net - Feature Importance

5.1.3 Decision Tree

We used a decision tree model as our baseline. The hyperparameters tuned were max_depth, max_leaf_nodes, and max_features. The best hyperparameters obtained were max depth = 12, max features = 17, max leaf nodes = 100. The RMSLE is 0.466 after minimum cost-complexity pruning. Figure 5 shows us that this model gives maximum weightage to the feature *room_type_entire home/apt*, which makes sense since an entire home would have a higher price than an apartment.

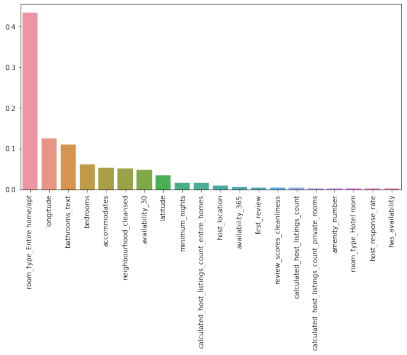

Figure 5: Decision Tree - Feature Importance

5.1.4 Bagging: Random Forest

For a random forest regressor that uses the bagging technique, the hyperparameters used were min sample split, n_estimators, max depth, and max features. The best hyperparameters obtained for random forest are n_estimators = 150, max depth = 86, max features = 10, and min sample split = 2. To ensure the random forest regressor does not overfit, we used early stopping. The RMSLE was 0.392 on testing set. From Figure 6, we can verify that because of bagging many features contribute to the final prediction.

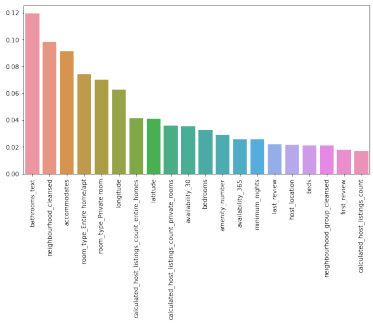

Figure 6: Random Forest - Feature Importance

5.1.5 Boosting: Adaboost

For an Adaboost regressor that uses the adaptive boosting technique. The hyperparameters used were n_estimators, and base_estimator. For the base_estimator we used the decision tree with best hyperparameters we obtained before and the best value of n_estimators was 50. The RMSLE was 0.446 on testing set. Figure 7 shows us that this model gives high weightage to the features *accomodates*, *neighborhood_cleansed*, *latitude*, and *longitude* among others which indicates that it could be a good model.

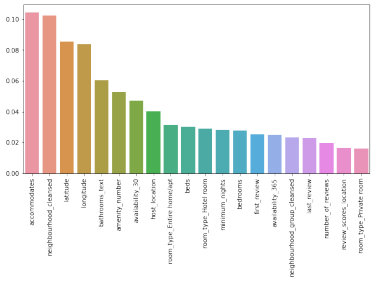

Figure 7: AdaBoost - Feature Importance

5.1.6 Boosting: XGBoost

For an XGBoost regressor that uses the boosting technique, the hyperparameters used were min sample split, n_estimators, max depth, and learning rate. The best hyperparameters obtained for the XGBoost regressor are: n_estimators = 150, max depth = 11, learning rate = 0.004642. The RMSLE was 0.383 on testing set. Hence, XGBoost performs better than Random forest. Figure 8 exhibits the distribution of Predicted v/s actual data. We can see that the overall trend is followed in our prediction.

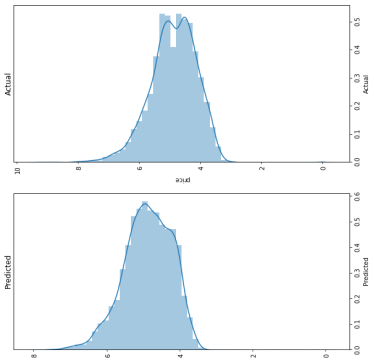

Figure 8: XGBoost - Feature Importance

5.1.7 Neural Network

For the neural network model, we have set five neural structures and fine-tuned on hyperparameters. The number of epoch is 100, the batch sizes are 8 across five models, and optimizers are Adam with no weight decay and learning rates equal to 10^{-3}. The five models are updated by MSE losses. Figure 9 exhibits the model configurations including the number of hidden layers, hidden nodes, activation functions, dropout and normalization, and finally, the real model RMSLE losses.

Layers	Hidden Nodes	Activation function	Dropout	Norm	Loss
4	128, 64, 32, 1	ReLU	-	-	0.838
	32, 16, 8, 1	Leaky ReLU	0.5, 0.5, 0.5	Layer	0.479
	24, 16, 8, 1	ReLU, GeLU, Sigmoid	0.5, 0.5, 0.5	Batch	0.472
5	**128, 64, 32, 16, 1**	**Leaky ReLU**	**-**	**-**	**0.414**
	512, 128, 128, 8, 1	ReLU	0.75, 0.5, 0.5, 0.1	Batch	0.815

Figure 9: Neural Network - Model Configuration and MSLE Loss

The best neural model is highlighted in bold text, which has 5 hidden layers with 128, 64, 32, 16, and 1 nodes, respectively. Leaky ReLU is used as activation function and with no dropout and normalization layers. It achieves the RMSLE loss of 0.414. The training/validation curves, which records the MSE of log prices, from the 5th epoch to the 100th epochs are depicted in Figure 10.

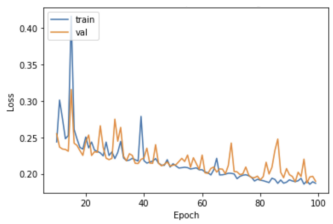

Figure 10: Neural Network - Training Curves

5.2 Model Selection

The model selection is based on the RMSLE of each model performance on the testing set which is illustrated by Table 1. The best model of the seven is **XGBoost** with RMSLE 0.383.

Model	RMSLE on testing set
Linear Regression	0.460
Elastic Net	0.459
Descision Tree	0.480
Random Forest	0.392
AdaBoost	0.446
XGBoost	**0.383**
Neural Network	0.414

Table 1: RMSLE on test set of different models

5.3 Model Comparison

As we can see from the experiment, for regularization, it does not help significantly from simple regression. For bagging, random forest do help a lot comparing to pruned tree and even rank as the second lowest loss. For boosting, both AdaBoosting and XGBoost perform better than previous approaches. However, much contrast to our original expectation, neural network did not perform far better than all other models, and in fact ranks as the third. In Caruana and Niculescu-Mizil (2006) empirical studies, they pointed out that although neural networks generally have best result among all other approaches, boosted trees and random forest still outperform it in many tasks. Also, Jiang et al. (2020) concludes his experiments that Gradient boosting algorithm is optimal in small dataset while neural network is optimal in larger one. Hence, the fact that our neural network does not outperform XGBoost and random forest is reasonable since that we work on a small dataset with nearly 20K data after cleaning.

6 Conclusion

In the Airbnb housing problem, we first do the data exploration and data visualization to have an overview of the dataset. After that, we do the data preprocessing and feature engineering to prepare for the modeling part. Followed by the ML techniques in the lectures, we have started from regularization to bagging and boosting, and finally the neural network. After selecting based on RMSLE, the final model is XGBoosting. Among all models, except linear regression and neural networks, the important features are accommodates, neighborhood cleanse, location (longitude and latitude), bathrooms, room types, and amenities. Those features ranks highly in at least three of the models which suggest that they shuold contribute to the housing pricing significantly. In the period of high inflation and shock in labor market, every agent in the economy is trying hard to bid up the prices for higher income. Our report aims to figuring out what those real underlying features do consumer care most about when it comes to finding Airbnb housing. The proposed model can assist in agents to determine their prices is reasonable or not and further inform the householders which aspects should pay more attention to.

References

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006.

Jian Jiang, Rui Wang, Menglun Wang, Kaifu Gao, Duc Duy Nguyen, and Guo-Wei Wei. Boosting tree-assisted multitask deep learning for small scientific datasets. *Journal of chemical information and modeling*, 60(3):1235–1244, 2020.