Comparative Analysis of Machine Learning Models for Short-Term Rental Price Prediction

Yihua Hu
M.S. in Computer Science
UNI : yh3485

Rajwinder Mahal
M.S. in Computer Science
UNI: rsm2207

Mouwei Lin
M.S. in Data Science
UNI : lm3756

Zhaomeng Wang
M.S. in Computer Engineering
UNI : zw2801

Lihan Zhou
M.A. in Mathematics
UNI: lz2765

I. Introduction

Short-term rentals are gaining popularity in recent years, with Airbnb leading the change. Airbnb has enabled anyone to quickly put up a listing on its platform to make extra money by renting out unused space in their homes and apartments. When listing properties, one of the biggest challenges is finding the perfect listing price to attract potential customers. If you have multiple listings or have been listing your property for some time, you can easily use historical data to find competitive listing pricing. However, when you are just starting out, you might not be able to predict the best possible pricing. At the same time, it is possible that things are changing in the market based on seasonal demand and the overall economy. So, in this case, simply using the average prices from the past might not be the best option to find a competitive listing price. So, we are going to create a prediction model that can be used by hosts to find the most competitive price for their listings.

We use the Airbnb Listings Dataset as the data source to train our machine learning model. This dataset has a total number of $494,954$ records of Airbnb rental information with 89 features. Features include categorical features such as city, country, and Room type, numerical features like the number of bedrooms, beds, and baths, and descriptive text features like neighborhood overview and Summary. The label of the data is the daily rental price.

We are applying the following ML techniques to solve the problem.

- **Data cleaning, exploratory data analysis, and feature engineering.** Records from the raw dataset are not able to be directly fed into the model. We need first to process the data and do visualizations to better prepare it for the machine learning models.
- **NLP techniques to transform descriptive text features into categorical features.** Descriptive text features like neighborhood overview are hardly learnable for models. NLP helps us to transform these kinds of features into ones that are easier to interpret.
- **Supervised machine learning.** We are planning to build simple regression models and move toward more complex ones. Potential Models will be ensembled decision tree-based models: bagging models like Random Forest and boosting models such as CatBoost and XGBoost, and Deep Learning models like ANN under the spatial consideration of the price being non-linear.

II. Our solution

In this section, we show our detailed solution to building the model based on the Airbnb Listings dataset to predict the best labeling.

A. Initial data exploration

The $494,954$ records from the Airbnb dataset are from more than 20 countries, most of which are concentrated in four English-speaking countries, the United States, the United Kingdom, Canada, and Australia. So we filter out records that are from other countries and focus on data from these four countries. Moreover, after scanning through all provided features, we leave out features carrying no useful information like the *Listing Url* and *Picture Url*.

As a result, we have a total of $267,066$ records with 47 features after the initial exploration of the dataset.

B. Data preprocessing

Our work in data preprocessing includes three parts: missing value preprocessing, data encoding, and text feature extracting.

1) Missing value processing: There are columns that have a high percentage (over 25%) of missing values. We handle this problem as below.

- Neighbourhood is important but has about 27% missing values. We use *Zipcode* to group neighborhoods and then drop the *Neighbourhood* column.
- Different Review Score features like *Review Scores Location* and *Review Scores Accuracy* have a missing rate of over 26%. We impute these features by using the average of them.
- We drop the rest features with their missing value accounting for more than 30% of the records.

2) Data encoding: We combine various encoding methods to extract information from features of different types.

- We use ordinal encoding to encode categorical features of *Host Response Time*, *Room Type*, *Bed Type*, and *Cancellation Policy*.
- We use target encoding to encode categorical features of *City*, *State*, *Zipcode*, *Country*, and *Property Type*.
- For features *Amenities* and *Features* that consist of a list of keywords separated by commas, we split each value

into keywords and use one-hot encoding to encode these two features.

Finally, we apply the standard scaler to rescale numerical features.

3) Text feature extracting: We use VADER Sentiment Analysis for text feature extraction on descriptive text features like Description, Transport, and Notes.

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically tuned to sentiments expressed in social media. It uses a combination of A sentiment lexicon, a list of lexical features generally labeled according to their semantic orientation as either positive or negative with a specific value ranging from -1 to 1. Therefore, applying VADER can give us a quantification of the texts' emotions, which is helpful for the model since the texts' emotions are influential for the pricing for the fact that users are willing to spend more money on a positively described room.

After all the above three preprocessing steps, the dataset comes to $256,298$ rows with 165 features. We use random splitting to split the whole data into a 60% training set, a 20% validation set, and a 20% test set with *random_state = 42*.

C. Machine Learning techniques

We propose the following models to solve the problem.

- **Linear Regression model.** A common starting point to predict a numerical label with encoded features is to use a linear regression model. We use this model as the baseline to evaluate the performance boost of other models.
- **Tree-based models.** Tree-based models are also able to make numerical predictions. Since features like the number of rooms and beds are a good fit as splitting criteria for tree nodes, we apply models including CatBoost and XGBoost to see if this kind of model can perform well.
- **Deep learning models.** Deep learning models have been widely used in various predicting works for their outstanding ability to retrieve information from features over conventional models. Our work includes deep learning models of ANN, and we expect they can give an ideal solution to the problem.

III. Performance

In this section, we report the details about the application as well as the performance of each model. We use both the R2 score and mean absolute error(MAE) as the metric for all models. All following models use mean absolute error as the loss function.

A. Linear Regression

The Linear Regression model uses the basic idea of fitting the price label as a linear function of all features and is the baseline of all other models.

1) parameters: We choose $sklearn.linear\_model.Ridge$ as our LR model with *random_state=42* and all other default parameters.

2) performance: This model achieves an MAE of 50.32 and an R2 score of 0.57 on the test data, both reflecting that this is not a good model. The most important 5 features in this model are *Zipcode*, *Accommodates*, *Bedrooms*, *Room Type*, and *Bathrooms*, ordered by the importance.

B. XGBoost

The XGBoost model is one of the most popular realizations of gradient boosting and can support training on the GPU, which provides an efficient way to handle large datasets.

1) parameters: We utilize $xgboost.XGBRegressor$ API to build our XGBoost model. In terms of model parameters, we applied grid searches on a total number of 12 parameters where the best parameters are as in Figure 1.

```
XGBRegressor(tree_method='gpu_hist',predictor='gpu_predictor',max_depth=7,gamma=1.0,
             min_child_weight=5,random_state=42,subsample=0.96,
             colsample_bytree=0.6,n_jobs=-1,reg_alpha=100,
             learning_rate=0.08,n_estimators=600)
```

Fig. 1. XGBoost model with best parameters.

2) performance: This model achieves an MAE of 36.18 and an R2 score of 0.73 on the test data, which shows a considerable performance boost compared with the LR model. The most important 5 features in this model are *Room Type*, *Bedrooms*, *Bathrooms*, *Accommodates*, and *Zipcode* ordered by the importance.

We graphed the prediction of the model on the first 100 test data with the true label as in Figure 2 to visualize the performance details.

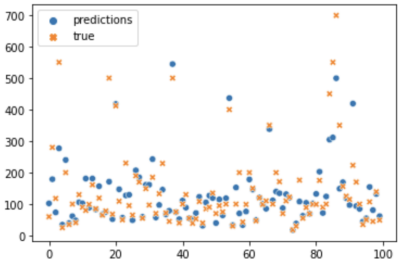

Fig. 2. XGBoost model predictions on the test data.

We can observe from the figure that our model can make a good prediction when the true label is within $300\$$. Data points with too high label values have a high probability of being over-underestimated by the model.

C. CatBoost

The CatBoost model is also a tree-based model as XGBoost but utilizes ordered boosting instead of gradient boosting. We also take advantage of its ability to train on the GPU to reduce training time.

1) parameters: We use $catboost.CatBoostRegressor$ API for the construction of our CatBoost model. Grid search is applied on four parameters, which are *bootstrap_type*, *depth*, *l2_leaf_reg*, and *learning_rate*. The best parameters are as below.

{*'bootstrap_type': 'Bernoulli', 'depth': 10, 'l2_leaf_reg': 3, 'learning_rate': 0.095*}

2) performance: This model achieves an MAE of 35.12 and an R2 score of 0.73 on the test data, which is very close to the performance of the XGBoost model. The most important 5 features in this model are *Zipcode*, *Accommodates*, *Bedrooms*, *Room Type*, and *Bathrooms*, ordered by the importance.

D. ANN-1

The ANN model is the basic version of deep learning models that implements biological neuron systems to simulate the human learning process.

1) parameters: We use $keras.models.Sequential$ API to build our ANN model. The specific model structure is as in Figure 3.

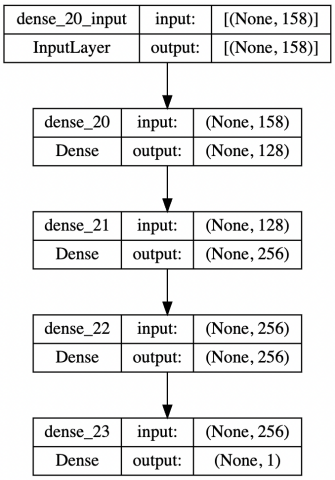

Fig. 3. ANN model structure

We compile this model with *adam* as the optimizer and train the model with *epoch = 100* and *batch_size = 256*.

2) performance: This model achieves an MAE of 45.11 and an R2 score of 0.56 on the test data. We apply SHAP to figure out the feature importance of this model. The most important 5 features in this model are *Room Type*, *Accommodates*, *Zipcode*, *Extra people*, and *City*, ordered by the importance.

E. ANN-2

To fully exploit the potential of the deep learning model, we construct another ANN model with additional layers and use grid search to tune parameters.

1) parameters: We use $keras.models.Sequential$ API to build our ANN-2 model. We apply gird search on five parameters, which are *learning_rate*, *dropout*, *layers*, *batch_size*, and *epoches*. The best model is compiled with *adam* as the optimizer and is trained with *epoch = 30* and *batch_size = 256*.

2) performance: This model achieves an MAE of 40.21 and an R2 score of 0.64 on the test data. Applying SHAP, the most important 5 features in this model are *Room Type*, *Zipcode*, *Accomodates*, *Bedroooms*, and *City*, ordered by the importance.

IV. Conclusion

Based on the observation of the model performance, we draw the following conclusions.

First, all models outperform the linear regression model, where the CatBoost model attains the best performance with an MAE of 35.12 and an R2 score of 0.73. The detailed performance of all models is shown in the table below. This result is not ideal for real-life scenarios, so we propose ways to help optimize it in the following Section V.

TABLE I
PERFOMANCE SUMMARY OF ALL MODEL

	LR	CatBoost	XGBoost	ANN-1	ANN-2
MAE	50.32	35.12	36.18	45.11	40.21
R2	0.57	0.73	0.73	0.56	0.64

Second, by comparing the performances of tree-based models and deep-learning models, we can conclude that tree-based models are more fit for this problem with rather structured data. In terms of training and tuning time, both tree-based models spend less time compared with the ANN-2 model, again proving their superiority for this problem.

Third, all models give very similar results for the most 5 important features. To conclude, *Zipcode*, *Accommodates*, and *Room Type* are the most frequently nominated features, which is in line with the human thinking style.

V. Future Optimizations

Based on the observation of the performances of our models, we propose the following methods that may give a better solution to the problem.

- **Apply advanced NLP learning methods.** In our work, we utilize a primitive NLP method, VADER, to quantify the emotion of the descriptive text to a number ranging from -1 to 1. However, text features such as *Access* carry more specific information like the time to the bus stops or the subways, which is not proper to be evaluated just on its semantic orientation. Thus, more advanced NLP methods are in need to better derive information from complex text features.
- **Build disparate models for records from different countries.** On exploration into the dataset, we find out that the feature *Prices* is of varied currencies according to the country, which may influence the overall accuracy of model predictions on the label. Therefore, training models focusing on only one country may help to learn region-based feature patterns and thus improve the model performance.
- **Add a branch model to learn features from the room pictures.** From visualization of the model predictions in subsection III-B, we find that the model performs badly at high prices. Recalling the process of customers viewing Airbnb records to estimate rental prices, especially when the rent is high, the picture of the room will be a very important element to establish an initial impression. So adding picture learning to the model may contribute to the price estimation.