# Price Prediction Model for Airbnb in Seoul

Jangwoo Park
jpark278@jh.edu

Jundong Hwang
jun0914@korea.ac.kr

Nahyun Park
nadianhpark@korea.ac.kr

## Abstract

This paper introduces several ensemble models which predicts the price of rooms listed on Airbnb. Our team focused on finding various correlations between each feature and prices. We then built regression models for each room type (Entire home/apt, private room, shared room), each noted with their respective $R^2$ scores. After experimenting various machine learning models, including deep neural nets, our team was able to find models of satisfactory performance, but some still need further improvements.

## 1. Introduction

Airbnb is one of the largest accommodation share services in the world. Airbnb, while serving only as a broker without owning any property listed on its platform, currently lists lodgings in more than 190 countries as well as tourist experiences offered and guided by locals. Airbnb in February 2019 had more than fifteen-thousand accommodation listings only in Seoul.

Unlike many other countries where Airbnb operates, South Korea has a special policy on accommodation rentals that prohibits national citizens from renting units on Airbnb. The Government, however, since 2019 has gradually been lifting said regulation. We expect that Airbnb's marketplace will only expand when said regulations are removed. Given the Government's recent deregulation that facilitates Airbnb's operation in South Korea, we believe Airbnb's market share in the vacation rental industry will expand in the coming years. For reasons listed above, processing Airbnb data could help many possible hosts by offering them guidelines on what they have to consider when renting out accommodations via Airbnb.

We planned to find the correlation among rental prices and other factors and build a model that predicts the expected revenue of a listing. Our future application of this final product will include an interactive aspect where users can receive the prediction of our model, in the form of expected revenue, based on the information of user input.

## 2. Datasets

The dataset of our choice has been collected by Tom Slee (https://tomslee.net/category/air-bnbdata). These datasets contain a temporal sequence of data of Airbnb in Seoul from May 2016 to July 2017. Each data contains huge number of features, including 'room_id', 'survey_id', 'host_id', 'room_type', 'country', 'city', 'borough', 'neighborhood', 'reviews', 'overall_satisfaction', 'accommodates', 'bedrooms', 'bathrooms', 'price', 'minstay', 'last_modified', 'latitude', 'longitude', 'location'.

## 3. Methods

In section 3.1 we describe our data preprocessing pathway. We also explain how we analyzed the distribution of Airbnb listings in Seoul and selected relative features. In section 3.2 we describe the process of building regression models and selecting its hyper-parameters.

### 3.1 Preprocessing

Among all the Seoul Airbnb datasets provided by Tom Slee, we opted for the largest and most recent dataset, composed of data collected in July, 2019.

#### 3.1.1 Feature selection

Above all features, our team excluded features 'room_id', 'survey_id', 'host_id' as they are ID inputs and therefore do not affect the price. We also excluded features 'country' and 'city' because our dataset of choice has been limited only to Airbnb data in Seoul. Features 'borough', 'bathrooms', and 'minstay' were excluded as they only carry 'NaN' values. The feature 'last_modified' includes the date and time in which each listing entry was read/scraped from the Airbnb website. For the feature 'overall_satisfaction', the majority of the data were 0.0 and the rest mostly 4.5 or 5.0. As the gap among instances is large, our team concluded that this could possibly lead the model to have wrong bias. Plus, the difference in satisfaction did not affect the price much. Features 'location', and 'neighborhood' proved to be rather problematic to process. 'Neighborhood' data had been categorically named after their adjacent towns, hence the 438 unique categories. The vast variety of categories, our team conjectured, would make encoding with one-hot encoder have too many 0 values, which would eventually lead to a large error of the model. Also, we thought 'latitude' and 'longitude' data would be enough to represent location data. The 'review' feature has also been deleted as it is not a prerequisite information when registering rooms for Airbnb.

In conclusion, our features include 'room_type', 'accommodates', 'bedrooms', 'latitude', 'longitude' data.

Then, we visualized the heatmap distribution of Airbnb on the Seoul map(figure1). Distribution of rooms were concentrated to some famous areas (Hong-dae, Seoul station/

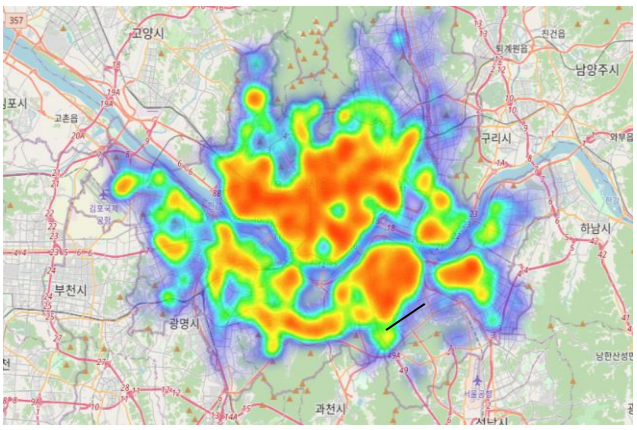



Figure 1: Heatmap of Airbnb in Seoul

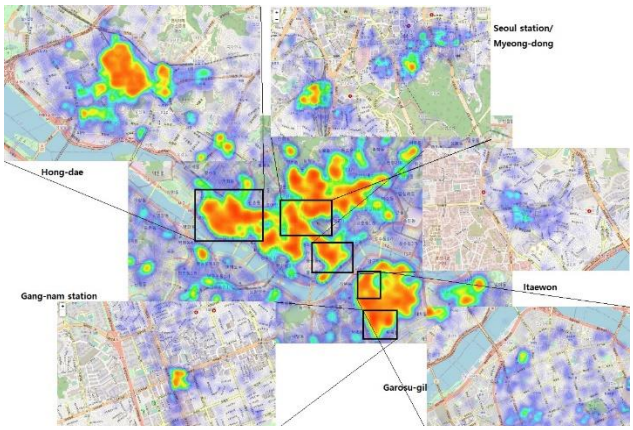



Figure 2: Locally concentrated distribution

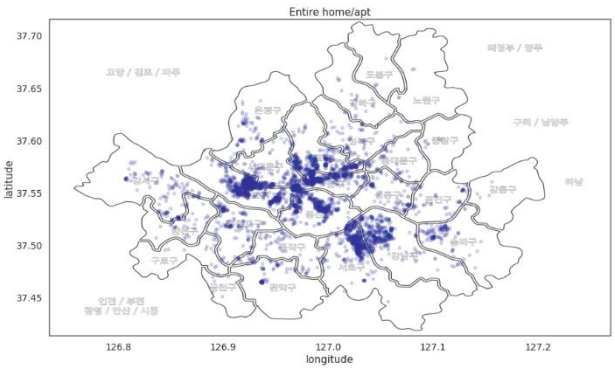
Myeong-dong, Itaewon, Gang-nam station, Garosu-gil) in Seoul, rather than equally spread (figure2).

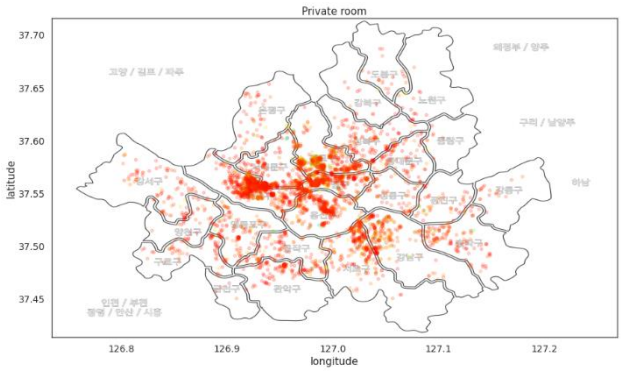
Then, we divided our dataset according to each room type (Entire home/apt, private room, shared room). Room types are what we consider first and the most when we look for a room on Airbnb. Room types don't only affect room types themselves; it affects everything besides room types as well. Facilities and the extent to which guests are allowed to use them differ greatly based on the room type. Having access to the kitchen and a certain number of bathrooms in a shared-property setting, for instance, does not necessarily mean that guests will always have access to such facilities; for shared properties, namely private room and shared room listings, the availability of facilities may differ based on other guests. When one guest is using the bathroom or the kitchen in a shared property, then other guests cannot, whereas in an "entire home" setting, guests are guaranteed access to all of the facilities 100% of the time. Therefore, we decided to build models separately according to room types.

| Room types | Number of subjects |
|---|---|
| Entire home/apt | 6172 |
| Private room | 5176 |
| Shared room | 1058 |

Table 1 : Number of rooms per each room types

There seems not much differences of distribution between each room types (figure 3). Most listings are concentrated to several popular areas, regardless of room types.



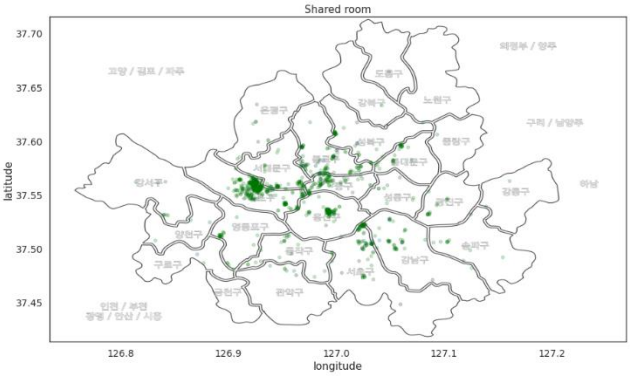
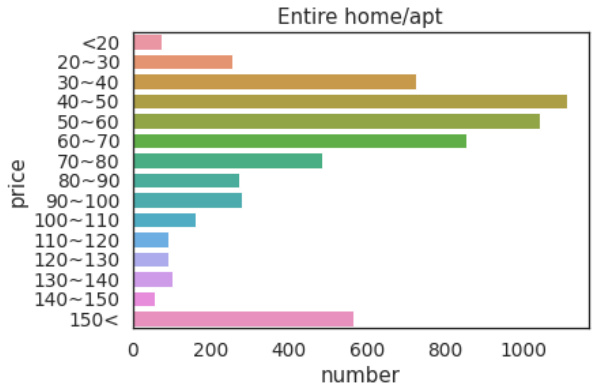
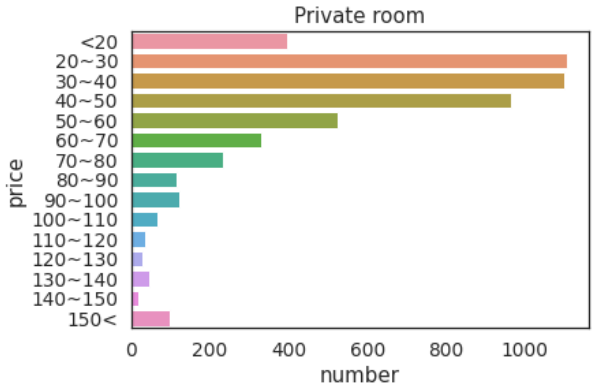




Figure 3: Distribution of each room types.

However, the overall price distribution of each room type showed obvious differences (figure 4).



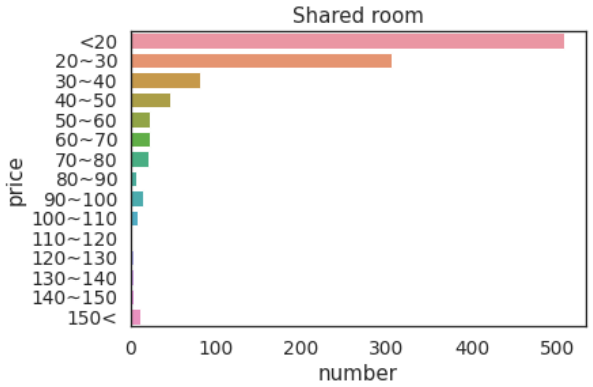


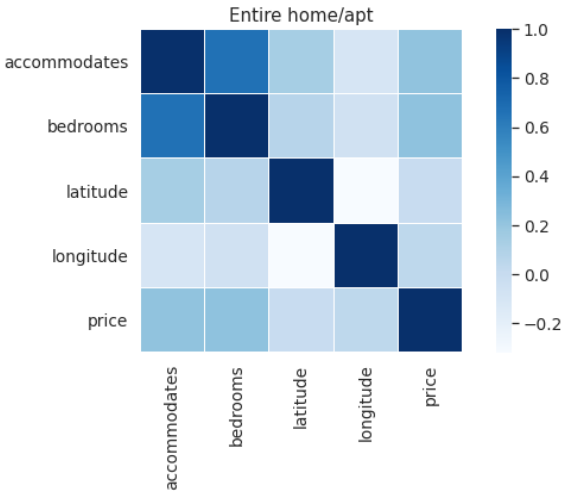

Figure 4: Distribution of prices per each room types

For the entire home/apt, the mean price is 80.69, and the standard deviation (Std) is about 157.55, which is very high. The price of "private rooms" and "shared rooms" shows much less standard deviation compared to "entire home/apt."

| Room types | Mean price | Std |
|---|---|---|
| Entire home/apt | 80.69 | ±157.55 |
| Private room | 47.73 | ±36.68 |
| Shared room | 29.37 | ±32.25 |

Table 2: Mean/std of price per each room types

Next, we checked the correlations among features. Correlation between location (latitude, longitude) and price could not be computed due to the non-linear relation between the two. We therefore checked to see the correlation between price and other two features (accommodates, bedrooms). For "entire home/apt," "accommodates" showed high correlation with "bedrooms" (66%), and both showed some correlation with price (21%, 22% each). Within the private room type,



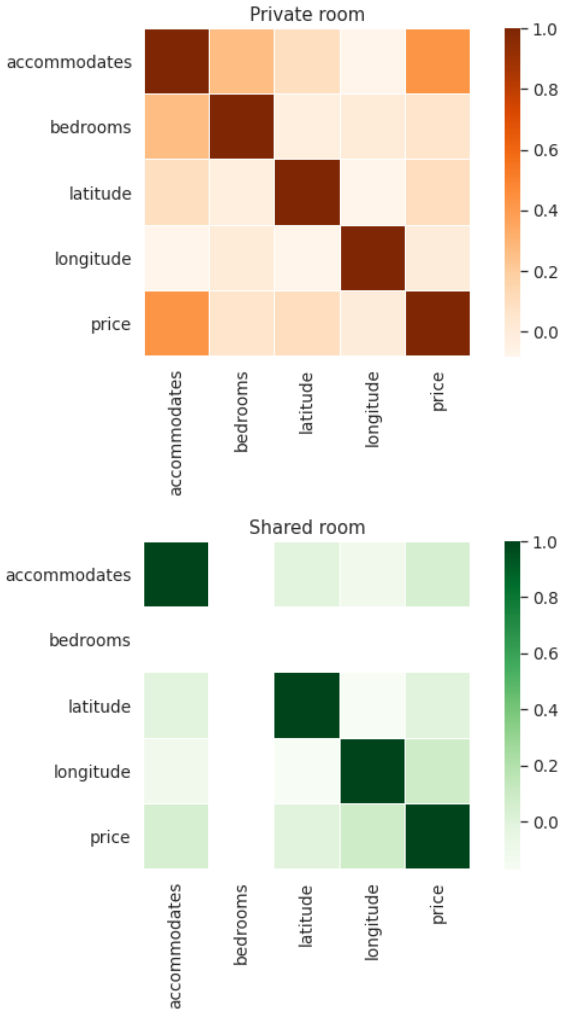




Figure 5: Correlation plot per each room types

"accommodates" showed greatest correlation with price (43%), and some correlation with bedrooms (26%). Lastly in shared rooms, neither "accommodates" nor "bedrooms" proved to have correlation with price (figure 5).

Next, we checked the histogram of price values to sort out the outliers of extreme prices. In Figure 4, especially in "entire home/apt," there are too many rooms priced over 150. We also confirmed some serious outliers, priced at over 1,500 USD. Also, for private room types, most listings were priced between 20 and 50 USD, while the mean price was around 50. Similarly, among shared room listings, most had prices under 20, while few entries still were priced at over 150. To seek for better accuracy by dropping some of the outliers, we decided to exclude the top 10 percent of the highest-priced rooms. Our team also concluded that features

'accommodates', 'bedrooms', 'latitude', and 'longitude' provided insufficient evidence and reasoning for those higher-end listed prices of our dataset. After cutting of top 10 percent of each data, highest prices dropped down to 140, 80, 55, respectively for each room type.

| Room types | Number of subjects (after dropout top 10%) |
|---|---|
| Entire home/apt | 5555 |
| Private room | 4658 |
| Shared room | 952 |

Table 3: Number of rooms per each room types after dropping out top 10% of prices

### 3.1.2 Data splitting

For validation and testing, we divided our data into three parts. First, we split our data into train and test sets with a 9:1 ratio. We then split our train data into train and validation sets to an 8:2 division. The data consequently ended up being divided into 3 parts, each taking 72%, 18%, 10% of the entire dataset.


Figure 6: Size ratio of train, validation, test dataset

### 3.1.3 Data scaling

Standard scaler is one of the most frequently used scalers in machine learning. Our team went with scikit-learn's version of a standard scaler.

## 3.2 Model selection

We searched for different kinds of regressors, tested with parts of our data, and found four candidates that seemed to perform well. In the training part, we concentrated on tuning the parameters, and find the best hyperparameters which shows best $R^2$ score. We then validated our models using 5-fold cross validation, validating with both $R^2$ score and root mean squared error (RMSE). And finally, we get those scores using test set, and decide what is the best model. As our team's objective of this project is predicting Airbnb price with minimum error, we set low RMSE as our greatest priority and a high $R^2$ score as second greatest when choosing best regression model among four distinct regressors. To compare each model visually, we graphed scatterplots of predicted price/real price.

### 3.2.1 XGB Regressor

First, we experimented with a XGB regressor, a machine learning boosting algorithm. We found optimal hyperparameters using grid search. The modulated hyperparameters were 'max_depth', 'n_estimators', and 'learning-_rate'. Detailed code implementation is shown in figure 7. Through a 5-fold cross validation with scoring with $R^2$, we decided best parameters of each room types.

```
import xgboost
from sklearn.model_selection import GridSearchCV

param_range = np.arange(0, 200, 10)
param_grid = [{'n_estimators': param_range,
               'max_depth' : [1,2,3,4,5],
               "learning_rate": [0.1, 0.2, 0.3, 0.4, 0.5]}]
xgb = xgboost.XGBRegressor()
print("========== START GRID SEARCH with 5-Fold CV ==========")
gs = GridSearchCV(estimator=xgb,
                  param_grid=param_grid, cv=5,
                  n_jobs=-1, scoring='r2')
gs = gs.fit(train_features, train_labels)
print("========== FINISH GRID SEARCH with 5-Fold CV ==========")
```

Figure 7] XGB regressor gird search code

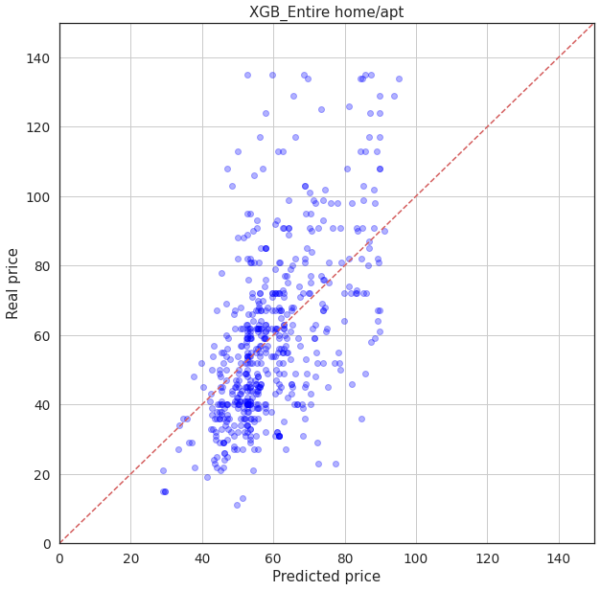



Figure 8] XGB regressor: Entire home/apt

For ‘Entire home/apt’ room type, the optimal hyperparameters were ‘max_depth’: 5, ‘learning_rate’: 0.1, ‘n_estimators’: 80. Through 5-fold CV, this model showed 0.34 $R^2$score and 20.38 RMSE. Also, they showed similar results with the test set ($R^2$:0.34, RMSE:20.07). The graph of linearity between predicted price and actual price of test set is shown in figure 8.

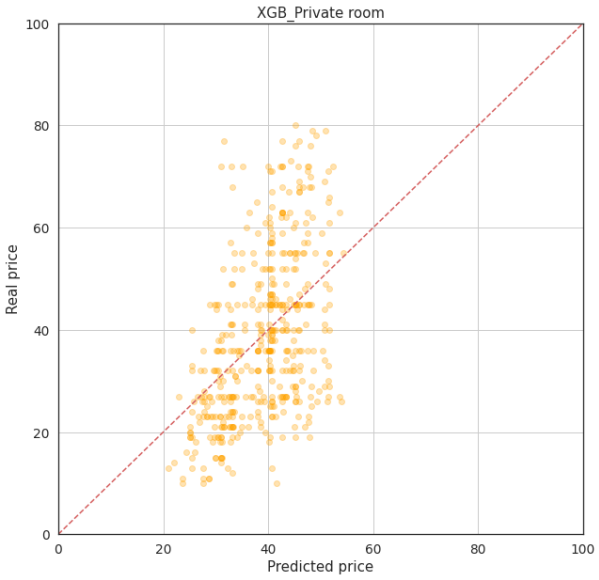



Figure 9] XGB regressor: Private room

For ‘Private room’ room type, best hyperparameters were ‘max_depth’: 5, ‘learning_rate’: 0.1, ‘n_estimators’: 60, showing an $R^2$ score of 0.26 and an RMSE of 13.41. With the test set, we ended up with a 0.24 $R^2$ score and a RMSE of 14.11. Since the price range of private room is smaller than entire home/apt, RMSE of private room was much lower while $R^2$ score is worse than that of entire home/apt.

Lastly for the ‘shared room’ type, best hyperparameters were ‘max_depth’: 5, ‘learning_rate’: 0.1, ‘n_estimators’: 60, with 25.69 $R^2$score and 8.18 RMSE. Similarly, the test set shows an $R^2$ score of 25.37 and RMSE of 8.05.

| Room types | Dataset | $R^2$score | RMSE |
|---|---|---|---|
| Entire home/apt | Train | 0.34 | 20.38 |
| | Test | 0.34 | 20.07 |
| Private room | Train | 0.26 | 13.41 |
| | Test | 0.24 | 14.11 |
| Shared room | Train | 0.26 | 8.18 |
| | Test | 0.25 | 8.05 |

Table 4: $R^2$score/RMSE of XGB Regressor

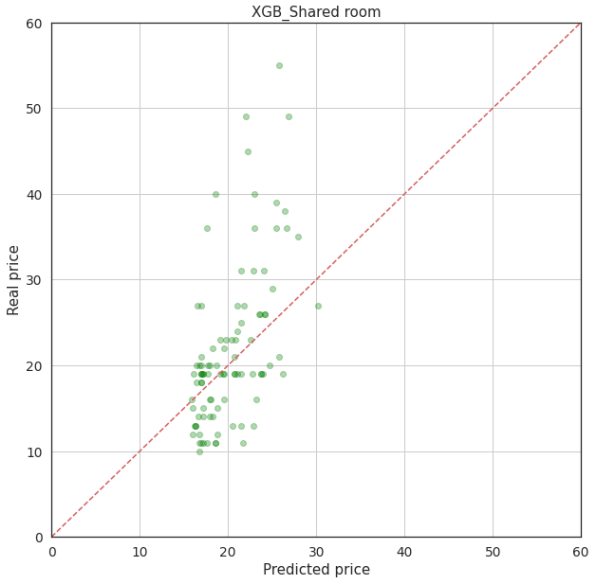



Figure 10] XGB regressor: Shared room

Overall, RMSE of this XGB regressor seems quite small, but we need higher $R^2$ score to use this model to predict prices.

### 3.2.2 Random Forest Regressor

Random Forest (RF) Regressor is an ensemble model with several decision trees. Because of that, this RF regressor is well known to show quite good results for such house price regression problems. So, we decided to go with a RF regressor to our dataset.

We applied grid search to find best parameters. Since this RF regressor took so long time to fit each hyperparameter to our datasets, we did 3-fold CV instead of 5-fold CV.

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV

param_range = np.arange(0, 200, 10)
param_grid = [{'n_estimators': param_range,
               'max_depth' : param_range,
               "min_samples_leaf": [1, 2, 3, 4, 5],
               "min_samples_split" : [1, 2, 3]}]
regr = RandomForestRegressor()
print("========== START GRID SEARCH with 3-Fold CV ==========")
gs = GridSearchCV(estimator=regr, param_grid=param_grid,
                  cv=3, n_jobs=-1, scoring='r2')
gs = gs.fit(train_features, train_labels)
print("========== FINISH GRID SEARCH with 3-Fold CV ==========")
```

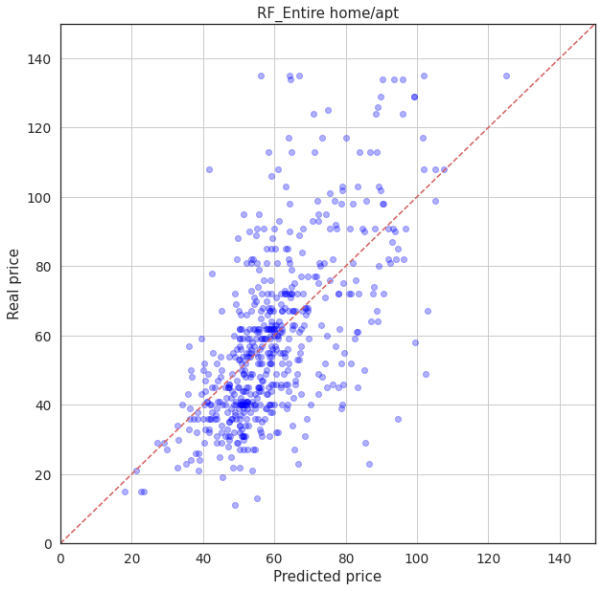
Figure 11] Random Forest regressor gird search code

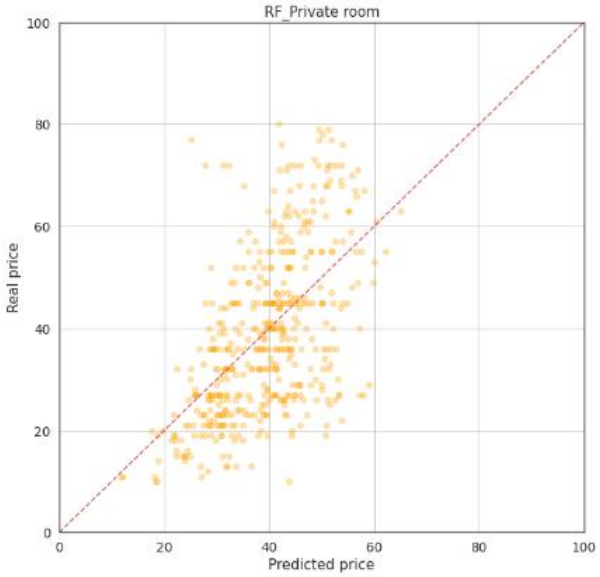
For the ‘entire home/apt’ room type, the optimal parameters were ‘max_depth’: 11, ‘min_sam-ples_leaf’: 2, ‘min_samples_split’:

2, ‘n_esti-mators’: 91. With these parameters, RF regres-sor showed 0.34 $R^2$score and 19.97 RMSE with train set, using 5-fold CV. In test set, this model showed an $R^2$ score of 0.41 and an RMSE of 19.10. One strange thing was the higher score of the test $R^2$ compared to the train $R^2$.

Using the same gird search to ‘private room’ type, and got best hyperparameters of ‘max_depth’: 11, ‘min_samples_leaf’: 1, ‘min_samples_split’: 3, ‘n_estimators’: 171. This RF regressor showed a 0.27 $R^2$score and a 13.26 RMSE. Similarly, this model showed an $R^2$ score of 0.30 and an RMSE of 13.54, using the test set. Just like ‘entire home/apt’, even in the ‘private room’ room type, the test $R^2$ score is higher than the train $R^2$ score. These results will be discussed later part of this paper.

For ‘shared room’ type, best parameters were ‘max_depth’: 160, ‘min_samples_leaf’: 2, ‘min_samples_split’: 3, ‘n_estimators’: 20. Compared to other room types, max depth has increased dramatically and the number of estimators is decreased. With this parameter, this model showed a 0.33 $R^2$score and a 7.44 RMSE. However, in the test set, it showed a 0.53 $R^2$ score and a 6.39 RMSE. Test $R^2$ score over 0.5 is optimistic, but the gap between train/test set result has to be narrowed.



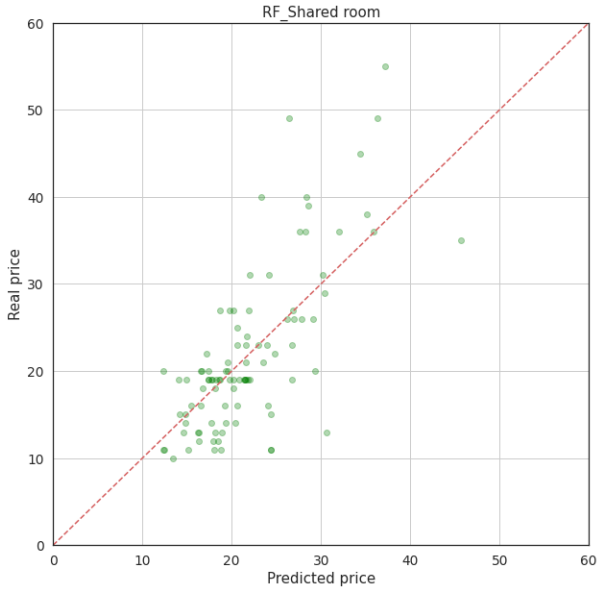




Figure 12] RF regressor: prediction/real price result

Except for ‘Shared room’ room type, the RMSE and $R^2$score of this RF regressor are not much different compared to that of XGB regressor. For the ‘Shared room’ room type however, with little improvement of RMSE, $R^2$score seems to overwhelm XGB regressor $R^2$score. The difference between $R^2$score of train/test data in ‘Shared room’ room type will be discussed in later.

| Room types | Dataset | $R^2$score | RMSE |
|---|---|---|---|
| Entire | Train | 0.34 | 19.97 |
| home/apt | Test | 0.41 | 19.10 |
| Private room | Train | 0.27 | 13.26 |
| | Test | 0.30 | 13.54 |

| | | | |
|---|---|---|---|
| Shared room | Train | 0.33 | 7.44 |
| | Test | 0.53 | 6.39 |

Table 5: $R^2$score/RMSE of RF Regressor

### 3.3.3 K Neighbors Regressor

Our next candidate is K Neighbors Regressor (KNR). KNR finds given number of nearest subjects of such input, and predicts labels accordingly. Since we though Airbnb price is related to location of the room, this KNR seems to fit our purpose. We used grid search with a 5-fold CV and scored with $R^2$. To avoid under-fitting, we limit the range of 'n_neighbors' under 100. The number of neighbors used in predicting price was between 30-60.

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import GridSearchCV

param_range = np.arange(0, 100, 1)
param_grid = [{'n_neighbors': param_range,
               "p": [1, 2, 3]}]
KNR = KNeighborsRegressor(n_jobs=-1,
                          weights = 'distance')
print("===== START GRID SEARCH with 5-Fold CV =====")
gs = GridSearchCV(estimator=KNR, param_grid=param_grid,
                  cv=5, n_jobs=-1, scoring='r2')
gs = gs.fit(train_features, train_labels)
print("===== FINISH GRID SEARCH with 5-Fold CV =====")
```

Figure 13] K Neighbors regressor gird search code

In the 'Entire home/apt' room type, KNR model used 43 neighbors, and p was 1. Using those hyperparameters, 0.36 $R^2$score and 19.75 RMSE was calculated through a 5-fold CV with train set. Moreover, when using the test set, this model showed 0.43 $R^2$score and 18.78 RMSE, which is the best result of all three models given above (XGB, RF, KNR).

KNR also showed greatest performance for 'private room' room type. Fitted parameter was 'n_neighbors': 34, 'p': 1. In training set, this model showed 0.30 $R^2$score and 13.02 RMSE. Considering that other two models (XGB, RF) had under 0.3 $R^2$score in training set, this KNR seems better than other two models. Also, in the test set, this model showed 0.32 $R^2$score and 13.35 RMSE. Like in 'Entire home/apt', this KNR model shows best performance above all three models in this 'Private room' room type.

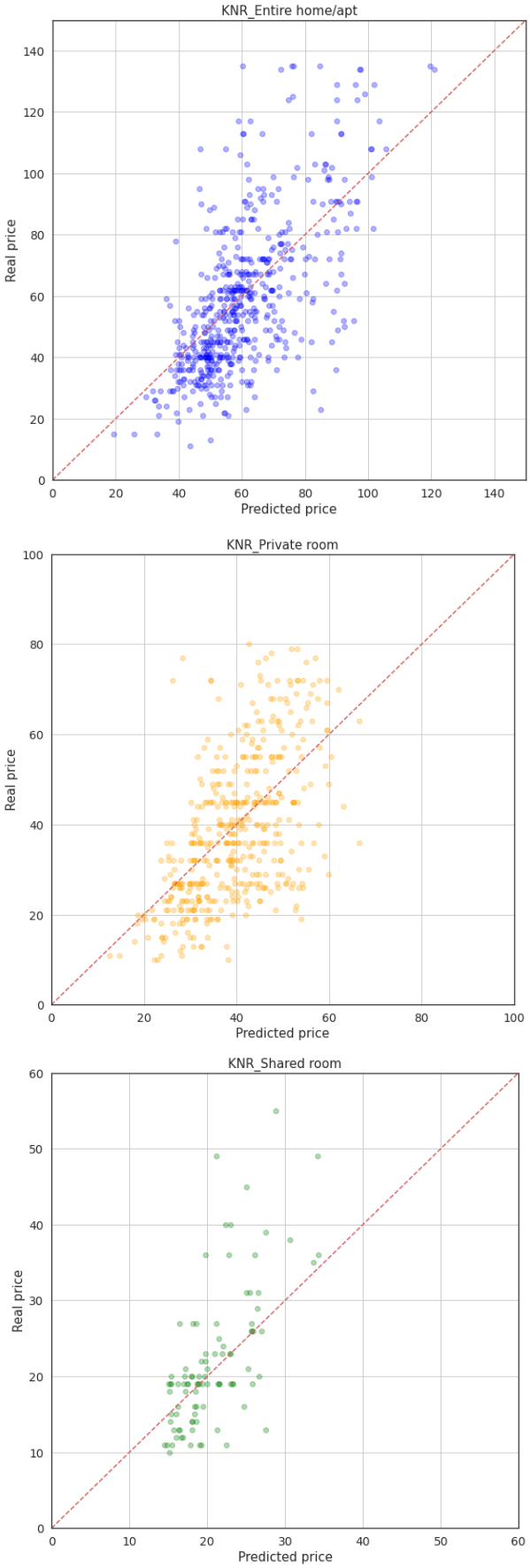



Figure 14] KNR: prediction/real price result

Lastly in ‘Shared room’ room type, best parameters were ‘n_neighbors’: 51, and ‘p’: 1. Like other two room types, this model showed good, showing 0.31 $R^2$score and 7.5 RMSE in the training set. In the test set, it resulted with a 0.39 $R^2$ score and a 7.29 RMSE. However, the result of the RF regressor is better than the KNR regressor in the ‘Shared room’ room type.

| Room types | Dataset | $R^2$score | RMSE |
|---|---|---|---|
| Entire | Train | 0.36 | 19.75 |
| home/apt | Test | 0.43 | 18.78 |
| Private room | Train | 0.30 | 13.02 |
| | Test | 0.32 | 13.35 |
| Shared room | Train | 0.31 | 7.5 |
| | Test | 0.39 | 7.29 |

Table 6: $R^2$score/RMSE of KNR Regressor

So, this KNR showed great performance in both ‘Entire home/apt’ and ‘Private room’ room type, showing highest $R^2$score and lowest RMSE above tested models. However, in ‘Shared room’, Random Forest seems to work better than KNR.

### 3.3.4 Support Vector Regressor

Our last candidate was Support Vector Regressor (SVR). We applied grid search with just parameter ‘C’, with a range of 10-1000. Since this support vector model is for linear data, we assumed this model does not fit our data, but applied it at least to compare with other models.

```
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVR

parameters=[
            {'C':np.arange(10,1000,10)}
]
svr = SVR()
grid_search = GridSearchCV(svr, parameters, cv=5,
                           scoring='r2', return_train_score=True)
grid_search.fit(train_features, train_labels)
```

Figure 15] Support Vector regressor gird search code

Using 5-fold CV, the best hyperparameter of ‘Entire home/apt’ room type was ‘C’: 10. With this parameter, this SVR showed 0.31 $R^2$score and 20.49 RMSE in the training set. Also, this score has not been changed dramatically in the test set, scoring 0.35 $R^2$score and 20.00 RMSE.

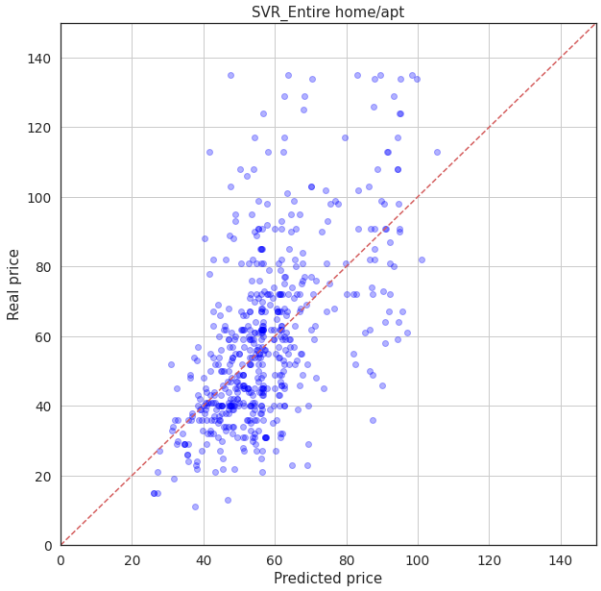



Figure 16] Support Vector regressor: Entire home/apt

SVR also didn’t work out well in ‘Private room’ room type. With their best hyperparameter, which is ‘C’: 20, they showed 0.20 $R^2$score and 13.89 RMSE in the training set, and 0.23 $R^2$score and 14.21 RMSE in the test set.

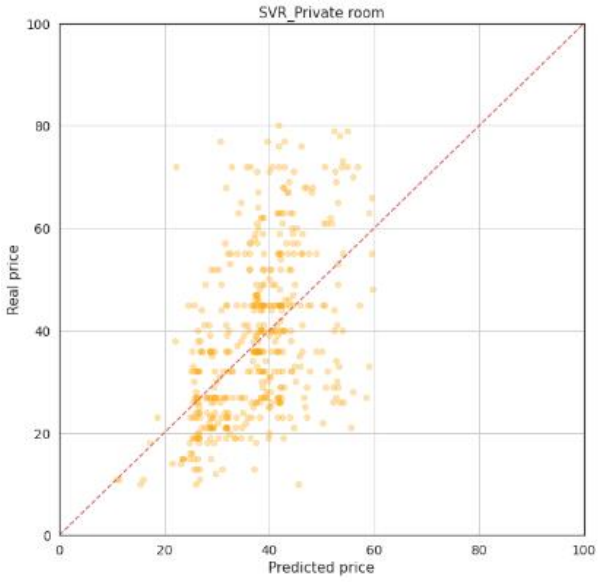



Figure 17] Support Vector regressor: Private room

In the ‘Shared room’ room type, the best hyperparameter was ‘C’: 30. With this, SVR showed 0.12 $R^2$score and 8.49 RMSE in the training set. They showed better results in the test set, scoring a 0.25 $R^2$score and an 8.07 RMSE. However, compared to the first three models, the SVR model doesn’t seem to be working.

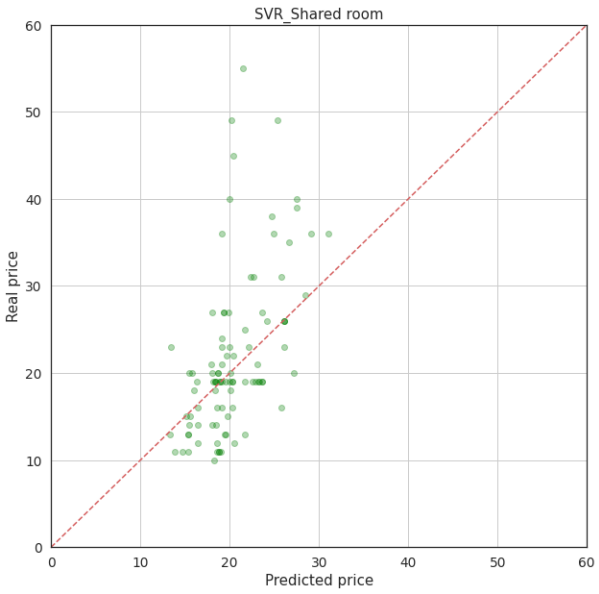


Figure 18] Support Vector regressor: Shared room

Generally, this SVR doesn't seem to fit for predicting price, showing lower $R^2$score and high RMSE than any other models we have tested.

| Room types | Dataset | $R^2$score | RMSE |
|---|---|---|---|
| Entire home/apt | Train | 0.31 | 20.49 |
| | Test | 0.35 | 20.00 |
| Private room | Train | 0.20 | 13.89 |
| | Test | 0.23 | 14.21 |
| Shared room | Train | 0.12 | 8.49 |
| | Test | 0.25 | 8.07 |

Table 7: $R^2$score/RMSE of Support Vector Regressor

## 3.3 Result

Above all models, K Neighbors regressor is selected as a best model for 'Entire home/apt' and 'Private room', while Random Forest regressor is selected as a best model for 'Shared room'. Details of models and hyperparameters are shown below.

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=34, p=1,
                    weights='distance')
```

```
KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=43, p=1,
                    weights='distance')
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=160, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=2,
                      min_samples_split=3, min_weight_fraction_leaf=0.0,
                      n_estimators=20, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

Figure 19] Best model & hyperparameter per each room types (Entire home/apt, private room, shared room)

In 'Entire home/apt' and 'Private room', K Neighbors regressor scored about 0.2-0.3 lower RMSE and 0.02-0.03 higher $R^2$score than 2nd highest model, which is Random Forest regressor. In 'Shared room', Random Forest regressor showed about 0.5-1 lower RMSE and 0.02 higher $R^2$score. Moreover, $R^2$score using RF in test set is about 0.14 higher than that of KNR.

## 3.4 Discussion

We found best models predicting price per each room type, but the result is still not optimal. To get an RMSE even lower and a higher $R^2$-score, we definitely need more features that are related to price, such as size of the room, how many people can stay, the number of beds…etc. For now, the amount and breadth of the data we have are too limited. By using more features, we think we can build models more precisely.

In almost all models that we tested, the test set $R^2$score is generally higher than that of train set. Possible reason of this gap between train/test $R^2$score is bias between those datasets. We, therefore, checked our train/test data, searched for bias in location, accommodations, bedrooms but could not confirm any bias discovered in our data. As R-squared scores represent the variance explained by each feature and label thereof, we assume that the greater the difference between the number of subjects in each set, the lower the resulting R-squared scores. We divided our train/test set in 9:1 ratio. $R^2$score shows linearity of predicted/real price, so as the number of subjects increases, the $R^2$ score also increases accordingly. We think this is the reason that the $R^2$ score of almost every model shows higher scores in test set than scores of train set.

## 3.5 Future works

Our next work would be getting more features. With that, we're thinking about building new models. Our team have come up

with three improvement we could make to better the performance. First is adding additional features that carry and could demonstrate additional corrleation with price, such as the size of an accommodation or check-in/check-out times. Another could be finding other ways to ignore outliers more efficiently by, for instance, incorporating a larger dataset, etc. Last is attempting a price recommendation based on the average price of closely located listings of the same room type and then considering other features. This concept would be based on KNR, with increased dimension to 3D instead of original 2D.

## 3.6 References


[1] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

[2] Cherkassky, V., & Ma, Y. (2003). Comparison of model selection for regression. *Neural computation*, *15*(7), 1691-1714.

[3] Gao, G., Bao, Z., Cao, J., Qin, A. K., Sellis, T., & Wu, Z. (2019). Location-centered house price prediction: a multi-task learning approach. *arXiv preprint arXiv:1901.01774*.

[4] Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. *R news*, *2*(3), 18-22.

[5] Kramer, O. (2011, December). Dimensionality reduction by unsupervised k-nearest neighbor regression. In *2011 10th International Conference on Machine Learning and Applications and Workshops* (Vol. 1, pp. 275-278). IEEE.

[6] Varma, A., Sarma, A., Doshi, S., & Nair, R. (2018, April). House Price Prediction Using Machine Learning and Neural Networks. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)* (pp. 1936-1939). IEEE.