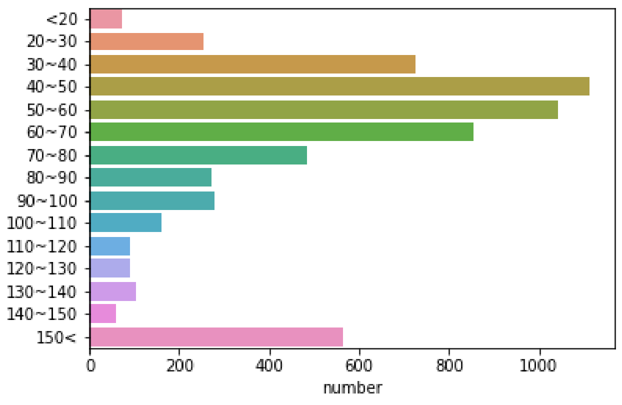
Bi-weekly Report (11/29) - Group 9

1. Model - Entire home/apt



We can see a long tail in the graph of the price of an entire home/apt accommodation. This type of accommodation has the highest price among the whole dataset. The price data has mean value of 59 and median of 81, but it has a maximum value of 10,376. The unbalanced feature of price can be the main reason for errors.

The correlation coefficient with the price data is calculated as follows. Bedrooms and accommodates have positive correlation. This is obvious since more bedrooms and more accommodates make the price go higher. The relation of latitude and longitude with price is not distinct. This is because the location of accommodation and the price do not have linear correlation(we have observed that the accommodations near to specific locations have high prices). Unexpectedly, the number of reviews have slightly negative correlation with the price. So, we decided not to use the 'reviews' column.

```
corr_matrix = data.corr()
corr_matrix['price'].sort_values()
```

```
reviews         -0.069806
latitude        -0.001475
longitude        0.042912
accommodates     0.218491
bedrooms         0.226635
price            1.000000
Name: price, dtype: float64
```

1) RandomForest Regressor

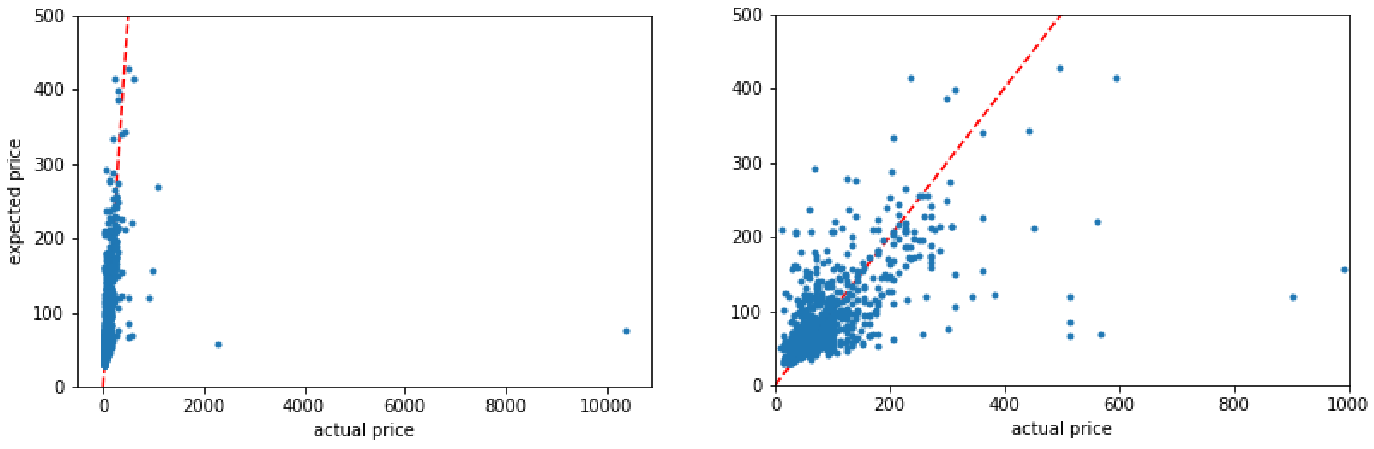
To increase the non-linear property of the model, we first tried a random forest regressor, rather than a decision tree model. Especially, the latitude and longitude data have non-linear correlation with the price, so increasing the number of trees can show better performance.

The scale between accommodation data (accommodates, bedrooms) and location data (latitude, longitude) are different, so we scaled the features with a standard scaler. (We have tried scaling through a robust scaler to decrease the impact of the outlier, but the performance was worse than the standard scaler.)

Using grid search with 10 cross-validation and scored with mean squared error(MSE).

The mean validation RMSE with the model was 71.75, with parameter min_samples_leaf of 5 and n_estimators of 2000. (This is higher than the decision tree model with RMSE of 73.17 ('min_samples_leaf':50).) Considering the mean price value 59, the model does not seem to perform well.

```
71.74828555852123 {'min_samples_leaf': 5, 'n_estimators': 2000, 'n_jobs': -1}
```



Above is a plot between the actual price and the expected price for the test set. First graph shows the whole plot, and the second shows values with actual prices under 1000. The red dash line includes points with the same expected and actual price.

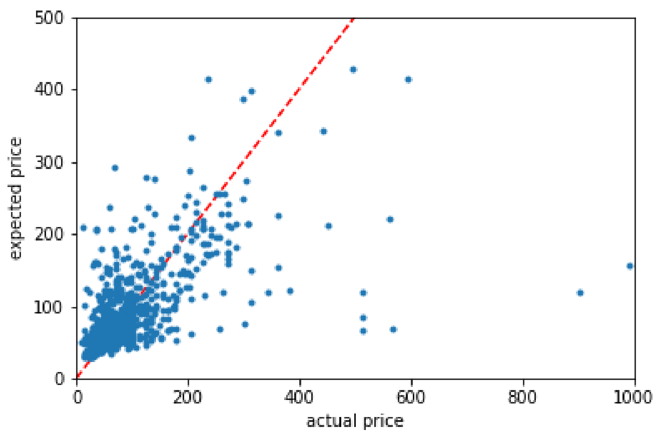
According to the second graph, there are some errors, but the points are headed upper right, so considering the noise, the model seems to present the data to some degree. Also we can observe that instances with high price are predicted too small. As the price data has long tail in the high-price region, the model was not trained properly in the region. So the predicted value has a massive difference with actual value, leading to high error.

2) SVM Regressor

The SVM model uses a kernel to increase non-linear properties. We predicted that location data will be presented well with the rbf kernel. Same as above, we used scaled data with 10 cross-validation using MSE scoring.

The mean validation RMSE of the SVR model (rbf kernel, C=500) was 71.75, which was almost the same as the random forest model. (The polynomial kernel show higher RMSE, 78.0)

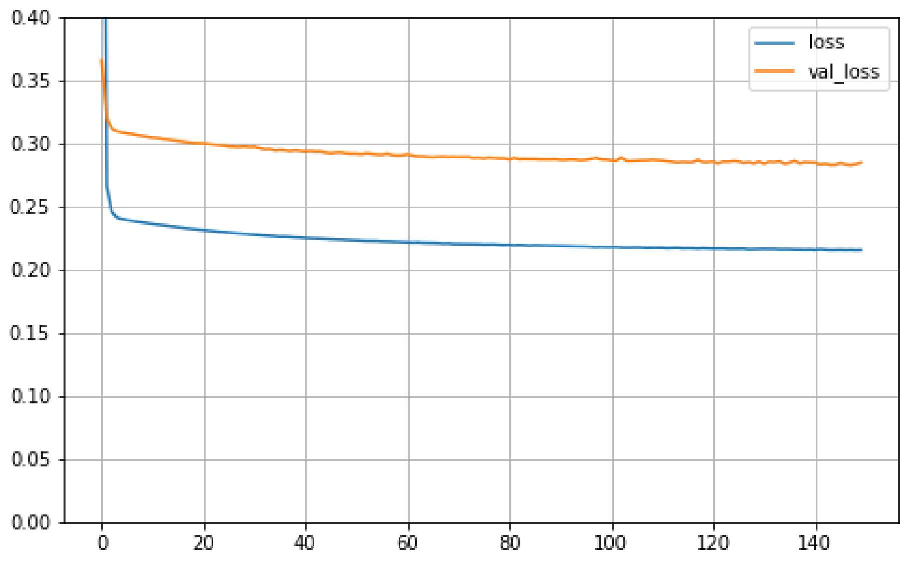
```
73.76368781765639 {'C': 10}
71.74530321209915 {'C': 500}
```



The plot between actual and expected price shows a similar aspect with a random forest model. Therefore we can conclude that the high error was due to few instances with high prices.

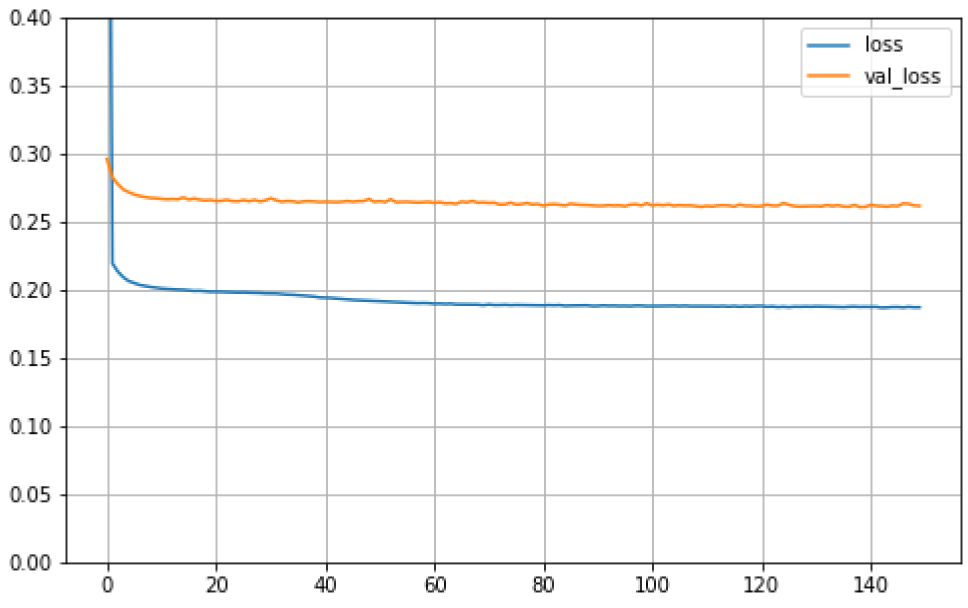
3) MLP regressor

We used keras to implement MLP regressor. First, we built MLP with a single hidden layer with 30 neurons to prevent overfitting, since the dataset has many noises. The hidden layer uses l2 regularization, and we chose relu for activation function and sgd for optimizer. Because of the large error, we used mean squared logarithmic error.

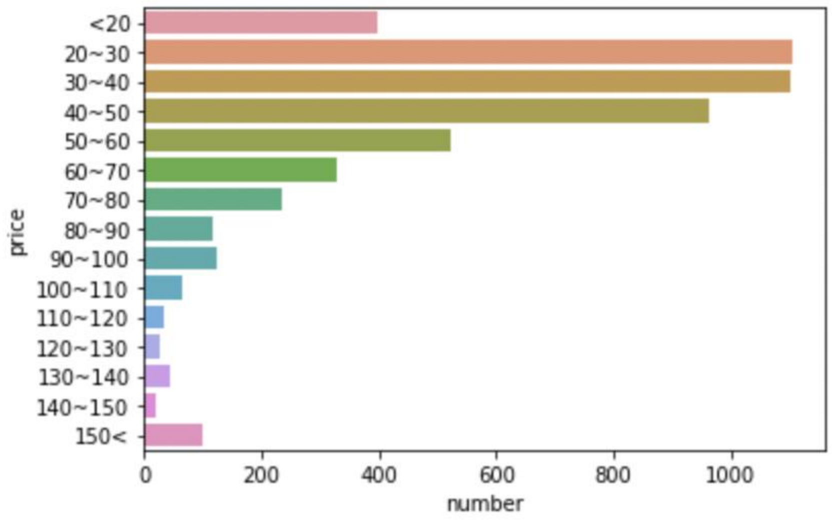


As the epochs increased, both the train and validation loss decreased. Therefore the model is not overfitted, so we need to train the model more. However the validation loss was over 0.28, which is much higher than models before if converted. So we stopped training and tried to find better structure.

We used randomized search with the number of hidden layers(1, 2, 3), number of neurons in each layer(10, 50, 100), and learning rate(1e-4, 1e-3, 1e-2, 1e-1). The best model had three hidden layers with 10 neurons each, and learning rate of 0.1. The validation error for this model was 0.26 (MSLE), still higher than the models above.

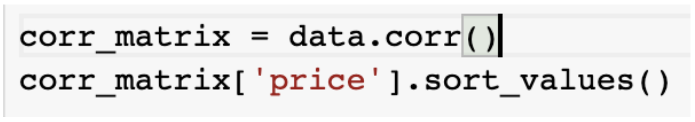


2. Model – Private room

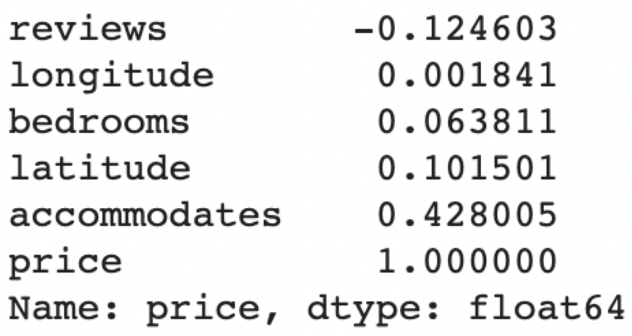


The graph above describes the price distribution of AirBnB listings that fall under the private room category. As described in the above graph, most listings of private room type fall under the 20-50 USD price range. Conceding with the commonsensical prediction, the average price of private room listings is less than that of entire home/apt properties but is greater than shared room listings.

The correlation between price and the rest of the data features (i.e. number of bedrooms, latitude) follows the calculations attached below.

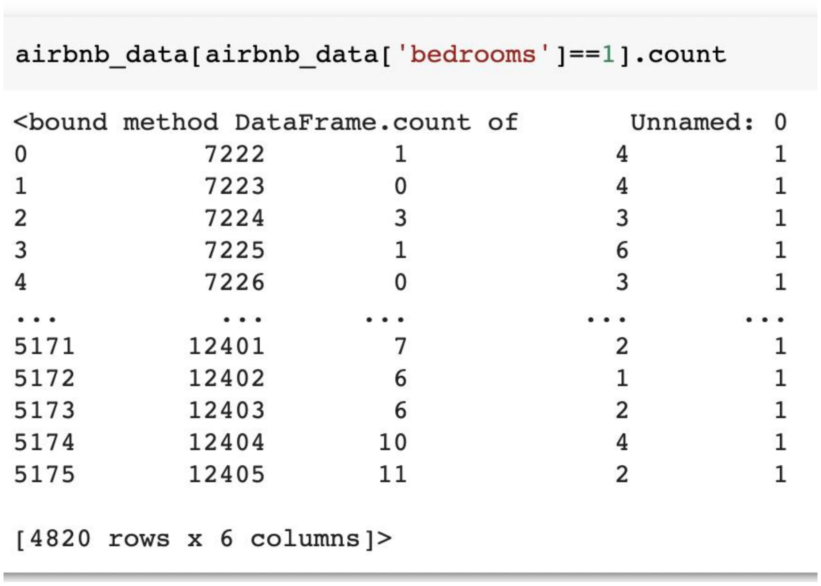


```
corr_matrix = data.corr()
corr_matrix['price'].sort_values()
```



```
reviews        -0.124603
longitude       0.001841
bedrooms        0.063811
latitude        0.101501
accommodates    0.428005
price           1.000000
Name: price, dtype: float64
```

Among all features in the data of private room listings in Seoul, longitude seems to be the least correlative to price, whereas the accommodation capacity seems to be the most correlative feature.

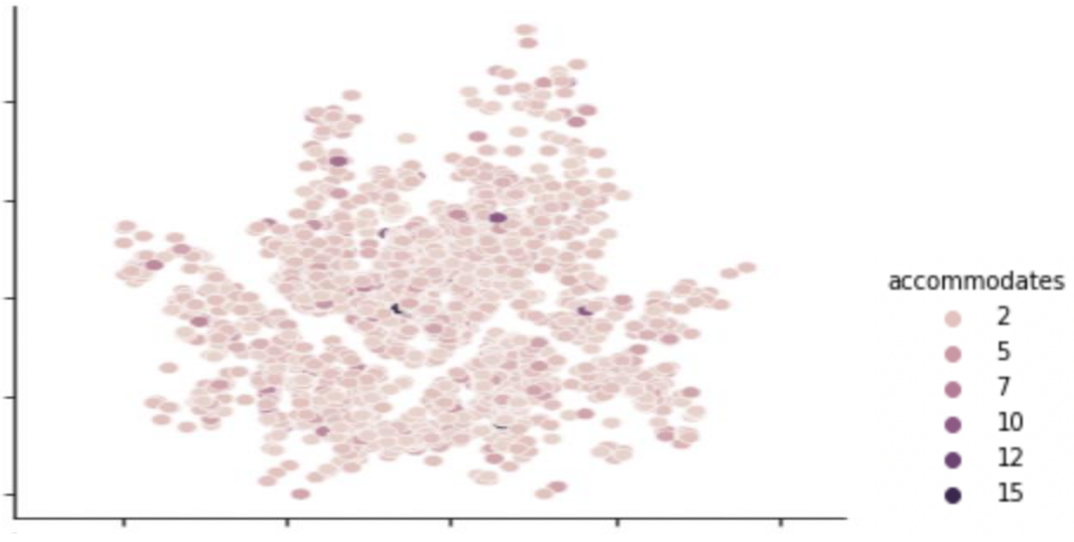


```
airbnb_data[airbnb_data['bedrooms']==1].count
```

```
<bound method DataFrame.count of        Unnamed: 0
0            7222         1             4          1
1            7223         0             4          1
2            7224         3             3          1
3            7225         1             6          1
4            7226         0             3          1
...           ...       ...           ...        ...
5171        12401         7             2          1
5172        12402         6             1          1
5173        12403         6             2          1
5174        12404        10             4          1
5175        12405        11             2          1

[4820 rows x 6 columns]>
```

One noteworthy characteristic of the private room data was that, among all 5176 private room entries, 4820 listing entries (approx. 93%) were of single-room types, the number of rooms of which equals one.

As data distribution is non-linear, our team concluded that linear regression might not be the most suitable modeling strategy to go for for our private room data. Instead of linear regression, our team decided to approach data analysis with Decision Tree Regressor and Random Forest Regressor.

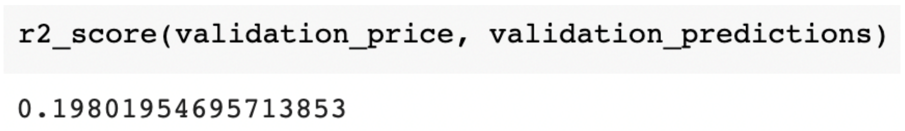


The price distribution based on longitude and latitude is described as above. This diagram is one of the obvious proofs that back the non-linear relation among price and other features.

For the splitting of data for training and test dataset, our team used scikit-learn's train/test split method. The ratio of the data volume between training set and test set was 4 to 1. Also, we used standard scaler to training/test features.

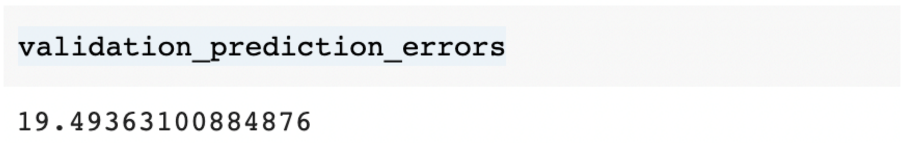
1) DecisionTree Regressor

Our first approach towards the private room data was using DecisionTreeRegressor.



```
r2_score(validation_price, validation_predictions)
```

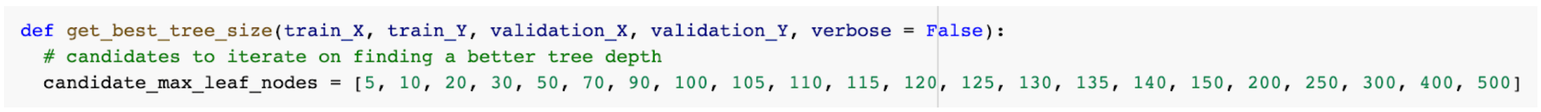
```
0.19801954695713853
```



```
validation_prediction_errors
```

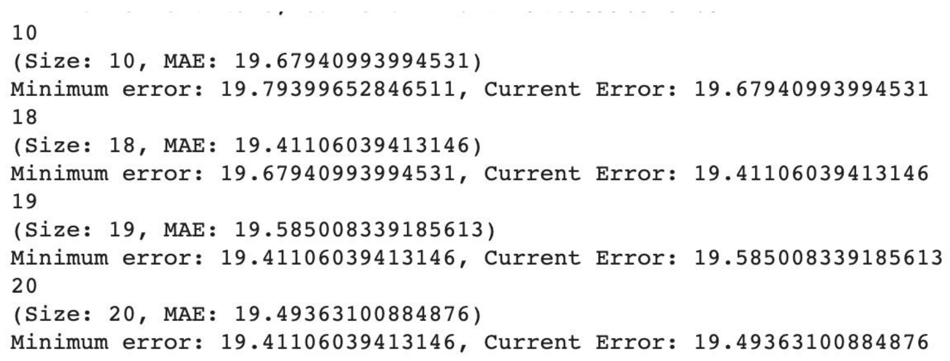
```
19.49363100884876
```

The original MAE (mean absolute error) of the DecisionTreeRegressor-using model was approximately 19.49, and the r2 score thereof was roughly 0.198. The low r2 score was easily predictable as the price of listings did not have a linear correlation with most of the other features. Our guess for the reason behind such high accuracy in our first try is due to the greater amount of data for private room type listings.



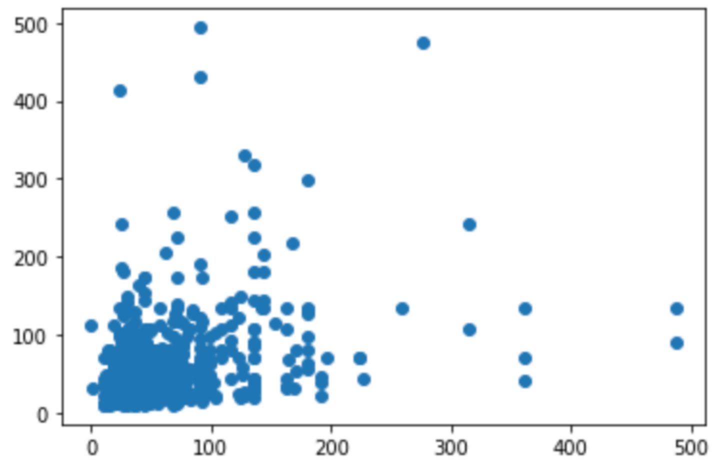
```
def get_best_tree_size(train_X, train_Y, validation_X, validation_Y, verbose = False):
  # candidates to iterate on finding a better tree depth
  candidate_max_leaf_nodes = [5, 10, 20, 30, 50, 70, 90, 100, 105, 110, 115, 120, 125, 130, 135, 140, 150, 200, 250, 300, 400, 500]
```

To find the best maximum number of leaf nodes for our RandomForestRegressor-using model building, our team iterated through all entries listed above in the candidate_max_leaf_nodes.



```
10
(Size: 10, MAE: 19.67940993994531)
Minimum error: 19.79399652846511, Current Error: 19.67940993994531
18
(Size: 18, MAE: 19.41106039413146)
Minimum error: 19.67940993994531, Current Error: 19.41106039413146
19
(Size: 19, MAE: 19.585008339185613)
Minimum error: 19.41106039413146, Current Error: 19.585008339185613
20
(Size: 20, MAE: 19.49363100884876)
Minimum error: 19.41106039413146, Current Error: 19.49363100884876
```

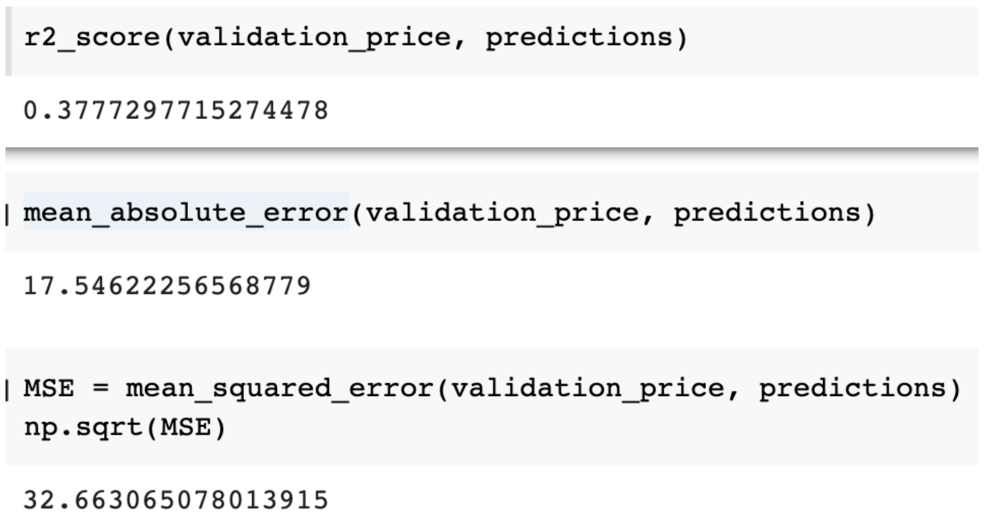
18 seemed to be the optimal number of maximum leaf nodes for our random forest regression. With such a number of maximum leaf nodes, the mean absolute error of the decision tree regression model decreased by a small amount, by less than one decimal point, from 19.493 to 19.411.



Attached graph is the scatter plot composed of validation data (actual price data) in the x-axis and prediction data (predicted price data) in the y-axis.

2) RandomForest Regressor

To maximize the non-linear property of the model, we next moved onto our second approach of using a random forest regressor, rather than a decision tree model. As explained in the above diagrams, the latitude and longitude data clearly does not have a linear correlation with the price, so our team concluded that increasing the number of trees can result in better accuracy.



```
r2_score(validation_price, predictions)
```

```
0.3777297715274478
```

```
mean_absolute_error(validation_price, predictions)
```

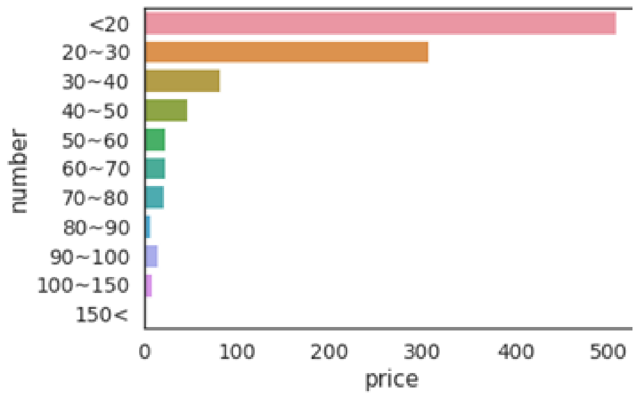
```
17.54622256568779
```

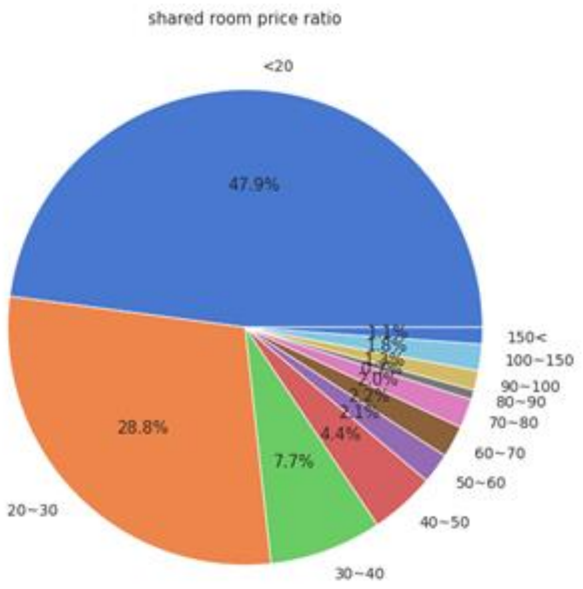
```
MSE = mean_squared_error(validation_price, predictions)
np.sqrt(MSE)
```

```
32.663065078013915
```

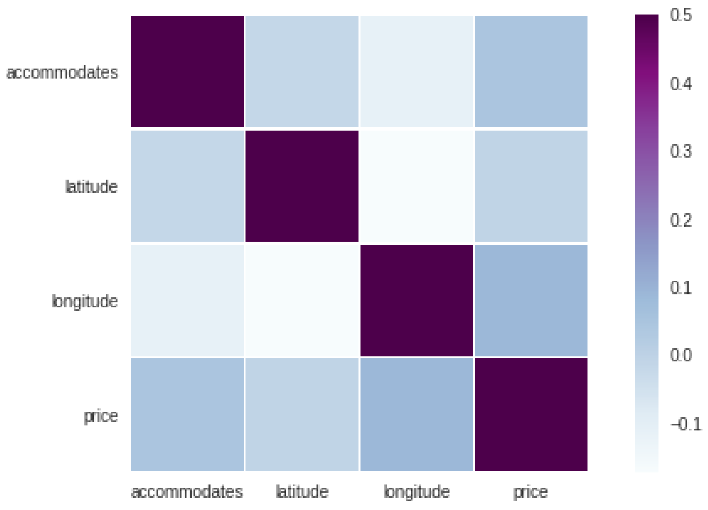
The random forest regressor, the maximum number of trees of which set at 500, resulted in a mean absolute error of approximately 17.546 and a RMSE of 32.66. The r2 score thereof was recorded to be 0.3778.

3. Model – Shared room



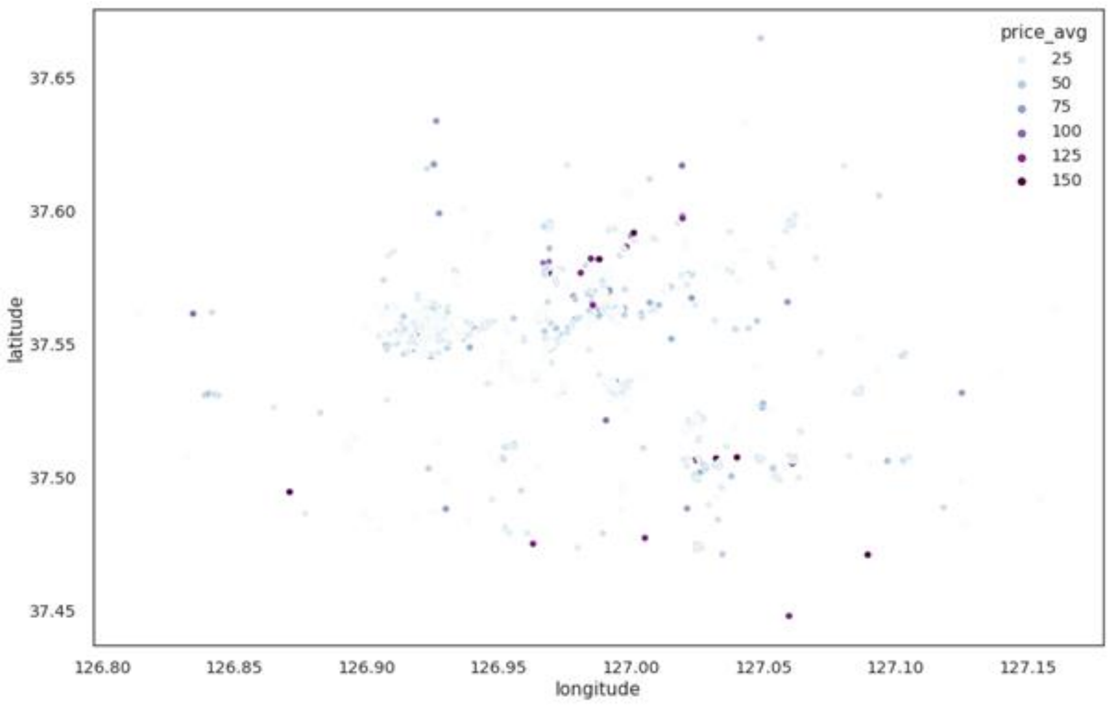


Most of the prices of Airbnb with shared room types are under 30(812 of 1058 subjects). Generally, compared to other two types, it seems that shared room type is cheaper than others.



The number of bedrooms of shared room type are always 1, which is useless for training models. Therefore, we had to build a model with just 3 inputs, which is accommodations, latitude and longitude.

As we already know, data distribution is not linear, so we thought linear regression might not work. Consequently, we started with Random Forest Regressor. For all models, we scored with both 'root mean squared error' and 'r^2', because all models showed little differences between each other.



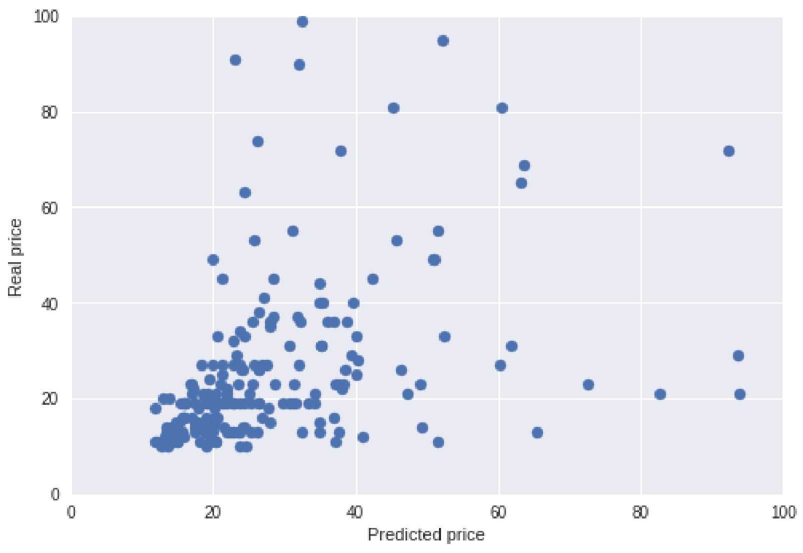
1) RandomForest Regressor

Tried both Grid search/randomized search to find best parameter. Searched hyperparameter was 'n_estimators','max_depth','min_samples_leaf','min_samples_split'. Above all, the best parameter is

max_depth:170, min_samples_leaf:1, min_samples_split:2, n_estimators:140

with 10fold cv. R2score was 0.120, and root mean squared error of test set/prediction was 35.84.

To be more clear, I checked the scatterplot of test/prediction.

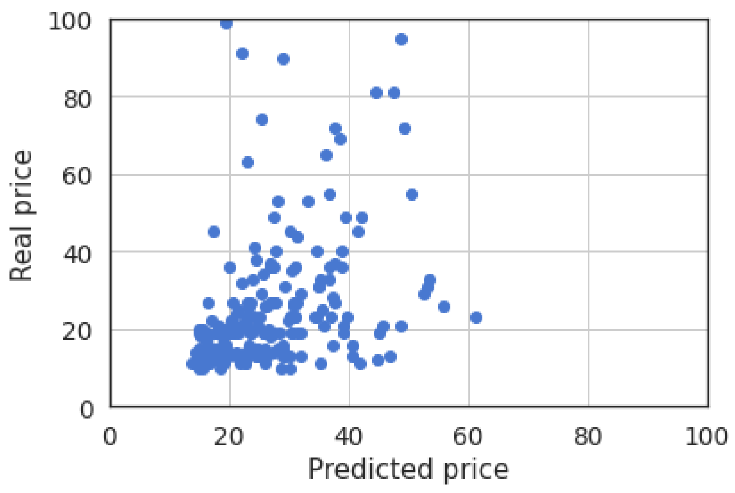


The error goes higher when the price goes high. Maybe it's because most of the prices of shared rooms are about 20~30. So, was the data of higher prices not enough to train this model?

2) KNeighbor Regressor

For KNeighbor Regressor, I did grid search with 'n_neighbors','weights','p'. With 10 fold cv, the best hyperparameter was

'n_neighbor':40, 'p':1, 'weights':'distance'



R2score was about 0.097, which is worse than RandomForest regressor. However, the root mean squared error of the test/prediction was about 27.5. So, I checked scatterplot again.

It seems a little bit more converging than RF regressor. Still, the error rate is high and not enough to use for prediction.

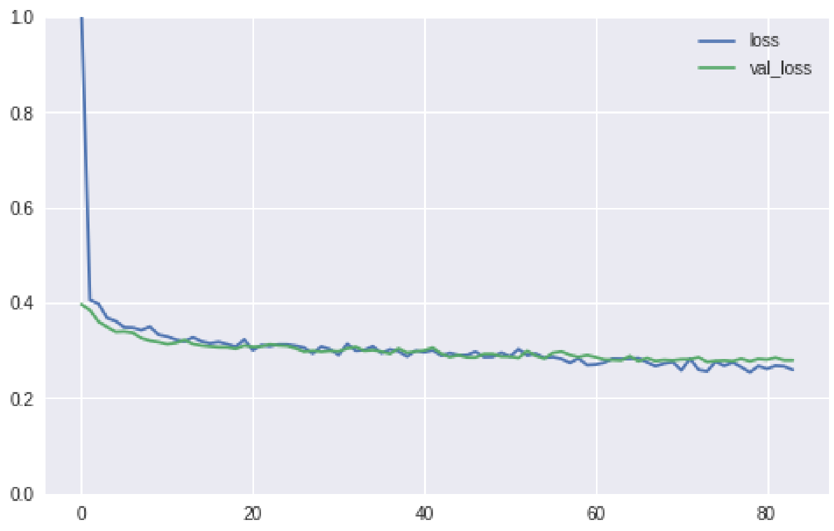
3) MLP Regressor

For MultiLayerPerceptron regression, I just have to decide the activation function and solver. Above all, the best activation function was 'relu', and the solver was 'adam'. R2score was 0.032 and root mean squared error was 29.7. Nothing special, nothing new.

4) DNN

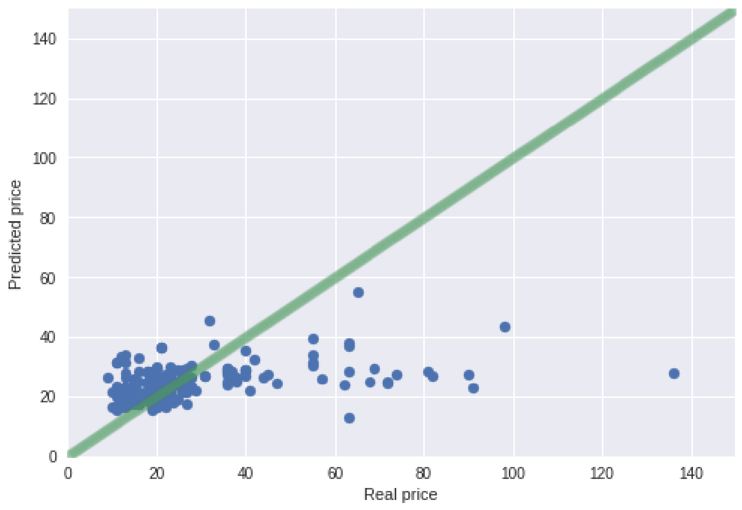
So I decided to build a DNN model.

```
def build_model():

  model = keras.Sequential([

    layers.Dense(320, activation='relu', kernel_initializer = 'he_uniform',
input_shape=X_train.shape[1:]),

    layers.Dropout(rate=0.2),

    layers.Dense(320, activation='relu', kernel_initializer = 'he_uniform'),

    layers.Dropout(rate=0.2),

    layers.Dense(128, activation='elu',kernel_initializer = 'he_normal'),

    layers.Dropout(rate=0.2),

    layers.Dense(32, activation='selu',kernel_initializer = 'lecun_normal'),

    layers.Dropout(rate=0.2),



    layers.Dense(1, activation = 'relu', kernel_initializer='he_uniform')

  ])



  model.compile(loss ='mean_squared_logarithmic_error', optimizer = 'adam')



  return model
```



The DNN model was a Sequential mode with 4 hidden layers, using a dropout of 0.2 rate.
Compiled with 'mean squared logarithmic error' loss function, and used 'adam' optimizer. Tried with different loss functions and optimizer, but logarithmic error and adam was best all above.

The training loss and validation loss were about 0.3. Calculated rmse was about 80.



I also tried with a functional API, but the result was the same or worse than the Sequential model. Overally, the DNN model underestimates prices. Compared to other ML models, I think predicting price lower than it is is better than predicting higher than real price, because price is what we care most about when looking for a shared room type of airbnb. To make a more specific model, we need more features related to prices.

4. What to do next

In all three categories, all regression models r2 score is too low and error is too high. Overall, the gap between predicted price/real price gets larger when the real price goes higher. We assume that there is something more to consider when deciding price.

According to that, we have to find commonalities of high-priced rooms to make new feature columns to strengthen our models.