Airbnb Price Prediction: Analysis of Multiple Ensemble and Non-Ensemble Regression Models

Lilly Sharples lillysharples@uga.edu
Department of Computer Science, University of Georgia
Kimberly Nguyen Kimberly.Nguyen@uga.edu
Department of Computer Science, University of Georgia

Abstract

The objective of this research is to predict the nightly price of an airbnb. Our focus is a dataset of 74,111 airbnbs in New York City, Los Angeles, San Francisco, Washington DC, Boston, and Chicago. We wanted to see if the general price per night could be predicted, based on different features of the given airbnb. Some of the features we experimented with include the number of amenities, property type, number of people that can be accommodated, number and type of beds, and the city. This predicted price will not account for factors out of our control, such as seasonal price increases or added cleaning/service/damage fees.

To accomplish this goal, we created and analyzed seven regression models: linear regression, random forest, decision tree, ADAboost bagging with decision tree as base estimator, ADAboost bagging with linear regrgession as base estimator, bagging with decision tree as base estimator, and bagging with linear regression as base estimator. We experimented with different scaling and feature selection methods with these models. Overall, our best model was a Random Forest Regressor with a feature selection variance threshold of 0.0, with a MAE of 0.2979.

1. Introduction

Airbnb is an online and mobile rental marketplace founded in 2008, where users can rent properties across the world. Many of these properties are privately owned houses for both long and short-term rentals, though there are unique rentals available as well- castles, islands, tents, and boats, just to name a few. The wide variety of properties results in a large range of pricing. In our data set, there were multiple properties for less than \$100 a night, with the most expensive being \$1,998/night for a 10 person/4 bed/4bath house in LA.

Our research objective was to compare multiple regression models to see how well they could calculate the nightly price of an airbnb given multiple features about the property. Specifically, we created a linear regression, random forest, decision tree, ADAboost bagging (one with linear regression and one with decision tree as base estimator), and bagging(one with linear regression and one with decision tree as base estimator). To evaluate the accuracy of these models, we used the mean absolute error (MAE). The MAE of a model is calculated by first subtracting the predicted value from the actual value, then taking the absolute value to get the absolute error. After calculating the absolute error for all entries, the average value of all absolute errors is the MAE [8].

Our original dataset came from Kaggle user Steve Zheng [14]. This dataset contained 74,111 entries, which we later split into a training set of 51,877 entries, and a testing set of 22,234 entries. Before training our models on the data, we took multiple steps of preprocessing the data. There were 29 columns in this dataset, and we initially dropped 8 columns that were irrelevant/contained a large amount of null values. As we are predicitng the price, we were also able to drop the 'log_price' column and assign it to our y-value.

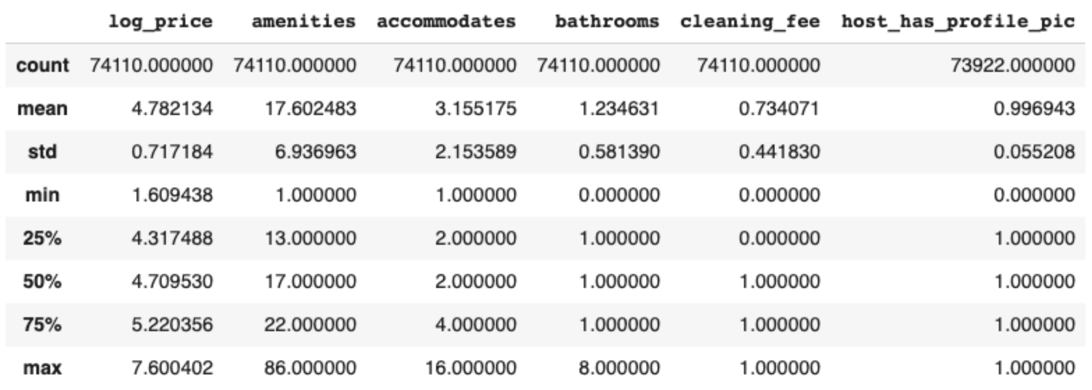
It was first necessary to transform the 'amenities' column from a comma separated list of amenities into a numerical value showing the number of amenities in the airbnb. We then dropped columns with large numbers of null values, as well as columns that were irrelevant to our model.

The next focus of preprocessing our data involved the categorical columns. Columns with true and false values ('host_has_profile_pic', 'host_identity_verified', 'cleaning_fee', and 'instant_bookable') had to be changed to 0's and 1's. 'property_type', 'room_type', 'bed_type', 'zipcode', 'neighbourhood', and 'cancellation_policy' contained strings. Using the Pandas .cat.codes method [13] allowed us to represent these categories as numerical values in the dataframe.

After all our data was in the correct form, we had to go back and deal with the remaining null values. The columns 'bathrooms', 'bedrooms', 'beds', 'zipcode_cat', and 'neighbourhood_cat' were filled with the median value of each respective column. 'host_has_profile_pic' and 'host_identity_verified' both had 188 null values, and we replaced those with the most common occurrence, which was true (numerically 1) for both. Section 3, Data-Preprocessing, goes into depth on the specifics of our preprocessing steps.

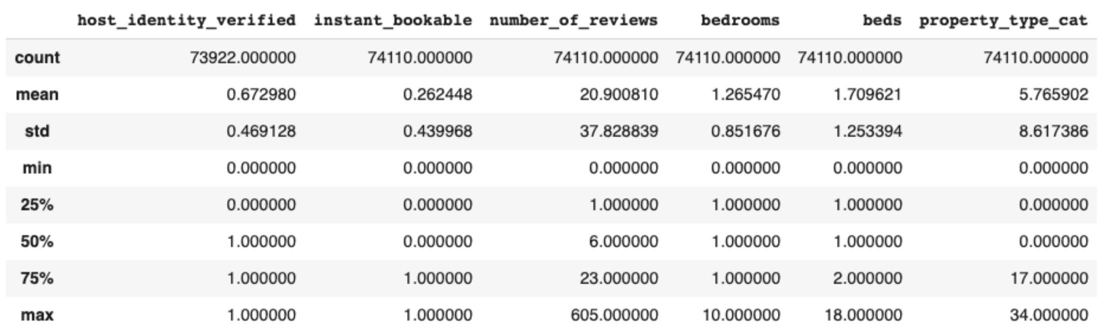
Once our preprocessing was complete, we created five regression models: linear regression, random forest, ADA bagging, decision tree, and another bagging model that utilizes decision trees as the base estimator. Within these models, we experimented with variance threshold and recursive feature selection. Variance feature selection uses the sklearn.feature_selection.VarianceThreshold() method [9] with a threshold of 0.025 to remove features with a variance lower than the threshold. Recursive feature selection selects the 5 most important features for our linear regression, random forest, and decision tree regressors. Section 3.1, Feature Selection, goes more into depth for this process.

Our best performing model was the Random Forest regression model after using variance threshold for feature selection with a threshold of 0.0. This model had an MAE of 0.2979. The bagging regression with decision tree as base estimator was our next best performing with an MAE of 0.3062, followed by the Linear Regression,Bagging with linear regression as base estimator, ADABoost models, then decision tree. More detailed information about each of these models as well as specific experimentation steps can be found in section 4, Experiments.



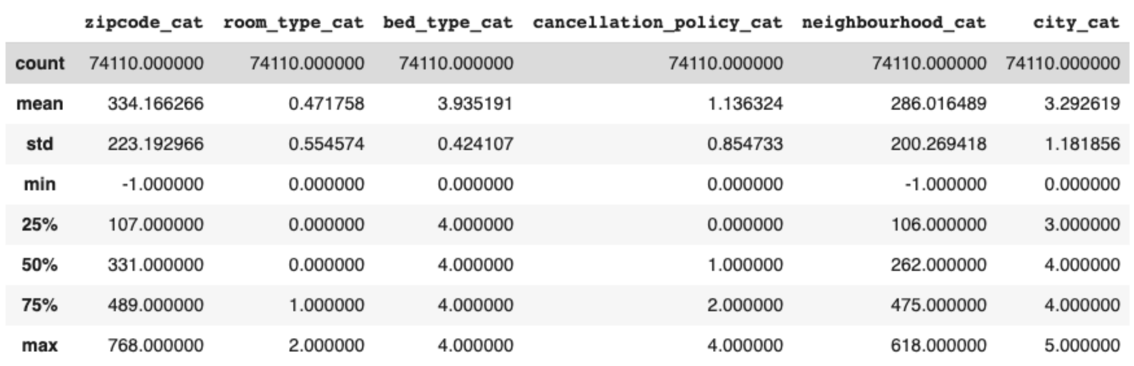
	log_price	amenities	accommodates	bathrooms	cleaning_fee	host_has_profile_pic
count	74110.000000	74110.000000	74110.000000	74110.000000	74110.000000	73922.000000
mean	4.782134	17.602483	3.155175	1.234631	0.734071	0.996943
std	0.717184	6.936963	2.153589	0.581390	0.441830	0.055208
min	1.609438	1.000000	1.000000	0.000000	0.000000	0.000000
25%	4.317488	13.000000	2.000000	1.000000	0.000000	1.000000
50%	4.709530	17.000000	2.000000	1.000000	1.000000	1.000000
75%	5.220356	22.000000	4.000000	1.000000	1.000000	1.000000
max	7.600402	86.000000	16.000000	8.000000	1.000000	1.000000

Figure 1: Describe Data



	host_identity_verified	instant_bookable	number_of_reviews	bedrooms	beds	property_type_cat
count	73922.000000	74110.000000	74110.000000	74110.000000	74110.000000	74110.000000
mean	0.672980	0.262448	20.900810	1.265470	1.709621	5.765902
std	0.469128	0.439968	37.828839	0.851676	1.253394	8.617386
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	1.000000	1.000000	0.000000
50%	1.000000	0.000000	6.000000	1.000000	1.000000	0.000000
75%	1.000000	1.000000	23.000000	1.000000	2.000000	17.000000
max	1.000000	1.000000	605.000000	10.000000	18.000000	34.000000

Figure 2: Describe Data Cont.



	zipcode_cat	room_type_cat	bed_type_cat	cancellation_policy_cat	neighbourhood_cat	city_cat
count	74110.000000	74110.000000	74110.000000	74110.000000	74110.000000	74110.000000
mean	334.166266	0.471758	3.935191	1.136324	286.016489	3.292619
std	223.192966	0.554574	0.424107	0.854733	200.269418	1.181856
min	-1.000000	0.000000	0.000000	0.000000	-1.000000	0.000000
25%	107.000000	0.000000	4.000000	0.000000	106.000000	3.000000
50%	331.000000	0.000000	4.000000	1.000000	262.000000	4.000000
75%	489.000000	1.000000	4.000000	2.000000	475.000000	4.000000
max	768.000000	2.000000	4.000000	4.000000	618.000000	5.000000

Figure 3: Describe Data Cont.

2. Related Work

Meitar Goldenberg [4] is a BI Developer at Investing.com. For his work, he used the same dataset as us. However, he only used airbnb data from NYC. Goldenberg trained and tested three linear regression models, two decision tree models, and three KNN models. The first linear regression model did not include any adjustments to the data, and gave an RMSE of 0.3444. The second linear regression model removed anomalous dots, by only including values between the (mean -3.6*std deviation) and (mean + 3.6*std deviation), with an RMSE of 0.3444. The third linear regression model utilized stepwise selection, performing a "forward-backward feature selection based on p-value from statsmodels.api.OLS" [4]. This model had an RMSE of 0.3440. The first decision tree regressor also did not include any adjustments to the data, with an RMSE of 0.357. The second tree regressor removed anomalous dots in the same way as the second linear regression model, also checking for/improving hyper parameters. This model gave an RMSE of 0.364. Again, the first KNN model did not adjust the data, and had an RMSE of 0.372. The second KNN model removed anomalous dots in the same way as previously, as well as choosing the best k-value (number of neighbours) by looking through k values 1 through 12, giving an RMSE of 0.351. The last KNN model changed weight to distance (Default=Uniform) and used k = 10, giving am RMSE of 0.349. While analyzing the NYC data is important, we took a broader approach and targeted airbnb data from all six cities. However, we did take a similar approach by training linear regression models.

Max B [2] is Kaggle user from Ukraine who also used the same dataset, using all six cities as we did. Max first trained a linear regression, with parameters n_jobs = -1 and normalize=True. This model had an MAE of 0.3, MSE of 0.16, RMSE of 0.41, and r^2 of 0.68. He also trained a random forest regressor, using parameters max_depth=10, n_jobs=-1, random_state=101, and n_estimators=700. This model also had an MAE of 0.3 and MSE of 0.16, with an RMSE of 0.4 and r^2 of 0.73. The next model was a gradient boosting regressor with random_state=101, also giving an MAE of 0.3, MSE of 0.16, RMSE of 0.41, and r^2 of 0.68. For the dense neural model, the data was first scaled using a Min Max Scaler. A sequential model was trained on the data, and resulted in an MAE of 0.32, MSE of 0.18, RMSE of 0.43, and r^2 of 0.69. Additionally, Max used textual data from each airbnb's description, removed the english stopwords, used a tfidf transformer, and trained the text using a random forest regressor. This model had an MAE of 0.47, MSE of 0.38, RMSE of 0.61, and r^2 of 0.35. Max's work is very relevant to our own, as we used the same data in a similar way. It was interesting to see the results of models different than ours, specifically the dense neural model. I also appreciated the random forest regressor on the text description, as I was initially curious to see how relevant an airbnb's description is to it's nightly price.

Laura Lewis [7] is a Oxford PhD currently working as a Data Scientist at Amazon. She has previously worked on data in startups, academia and government. Her work initially differs from ours in that she used a dataset of 80,000 London airbnb's from April 2019. The features in her dataset were similar, with 17 columns of features such as number of people accommodated, number of bathrooms, property and room type, and amenities. There were also features different than in our dataset, such as minimum/maximum nights to stay, number of available days, and whether the host is considered a superhost or not. She trained an XGBoost Regressor model. Before tuning the data, her original model had an MSE of 0.159 and r^2 of 0.727 when ran on her validation data. She next trained a sequential neural network model with four layers, L1 regularization, and an Adam optimizer, getting an MSE of 0.171 and r^2 of 0.711 on her validation data. Overall, the XGBoost model performed slightly better and was less computationally expensive, making it the preferred model for this data. Lewis also provided her predictions for what would make a better performing model- image analysis. She believes that more expensive airbnbs will have higher quality furnishings, shown in high quality photographs. She says that potential future work could be to "incorporate image quality into the model, e.g. by using the output of a convolutional neural network to assess image quality as an input into the pricing model" [7].

Sujith Kumar [6] is a data scientist at Tech Mahindra. He used the same dataset as us, using the features 'property_type, 'room_type', 'accommodates', 'bathrooms', 'bed_type', 'cancellation_policy', 'cleaning_fee', 'city', 'host_identity_verified', 'instant_bookable', 'neighbourhood', 'bedrooms', and 'beds'. Additionally, he created a new dataframe for the amenities by splitting the original entries by comma, creating a dictionary, using the .astype('category') method, then concatenating the new amenity dataframe to the other features. This was a different approach, as we decided to focus on the number of amenities compared to the individual types. He then created a tabular learner deep learning model from the fast.ai library. The RMSE and r^2 were given for each epoch, with the best coming from epoch 9, with an RMSE of 0.410 and r^2 of 0.657.

Raj Sankhe [11] is a Kaggle user who also used the same dataset as us for his work. He created a linear regression model, dropping the 'id', 'number_of_reviews', 'review_scores_rating', 'latitude', and 'longitude' columns after analyzing the correlation matrix. After seeing that 'bathrooms', 'bedrooms', 'beds', and 'accommodates' had a high correlation, he decided to only keep the 'accommodate' column. The original linear regression had a RMSE of 0.472. He continued to build two linear regression models with trees, a random forest and GradientBoosting. The linear regression with random forest had an MAE of 0.37, MSE of 0.239, and RMSE of 0.489. The linear regression with boosting had an MAE of 0.39, MSE of 0.267, and RMSE of 0.517.

Gracelia Carrillo, a data scientist at Candidates.ai and PhD Researcher at University of Glasgow, worked on predicting Airbnb housing prices based on several features, most importantly, the "property's proximity to certain venues." She utilized two models: Spatial Hedonic Price Model with LinearRegression from Scikit-Learn library and Gradient Boosting Model with XGBRegressor from XGBoost library. For the Spatial Hedonic Price Model, RMSE was 0.2661 and R_sq was 0.5108 (validation). For the Gradient Boosting Model, RMSE was 0.1881 and R_sq was 0.6557. The Gradient Boosting Model had more explained variability since it "automatically [provides] estimates of feature importance from a trained predictive model." The study concluded that the ratings of the location had a greater impact on price than the accessibility to interest points. [3]

Kaggle user spunchalski looked at short term rentals, which is less than a 31-day rental, indicating a tourist market to see if Airbnb housing prices can be estimated. He first cleaned the data by replacing any missing values, such as reviews with a 0 and names or host names with "None." From the analysis, it was observed that price had a positive correlation with longitude and availability, while having a negative correlation with number of reviews and reviews per month. [10]

A Kaggle House Price Prediction Competition was conducted to search for the best algorithm to predict housing prices. The models that were used were XGBoost Regressor Model (RMSE=0.0211), RidgeRegressor Model, and LinearRegression Model. For the XGBoost Regressor Model, the RMSE=0.0211. For the RidgeRegressor Model, the RMSE=0.0183. For the LinearRegression Model, the RMSE=0.01721. [12]

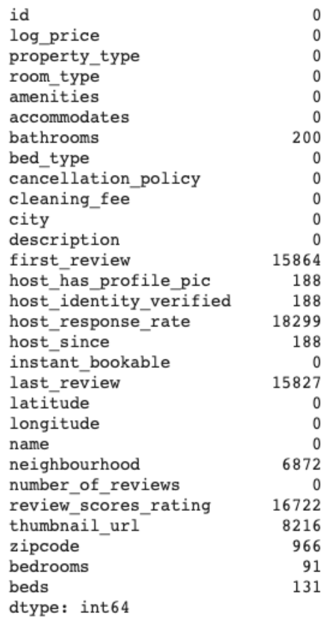
Yohan Jeong, a business analyst at Samsung Electronics America, analyzed housing factors in Melbourne to predict the city's housing prices. Missing values for "Bathroom" and "Car" were calculated using medians in "Rooms." Any missing values in Price and any feature with missing values are removed. Cross validation and grid search were performed. Linear Regression has an R_sq of 0.614591 and RMSE of 264465, Ridge Regression has an R_sq of 0.613263 and RMSE of 264920, KNN has an R_sq of 0.705321 and RMSE or 231250, and Decision Tree has an R_sq of 0.691989 and RMSE of 236423. [5]

John Ade-Ojo, a person who participated in Kaggle's advanced regression techniques competition, utilized Decision Tree, Random Forest, and Gradient Boosting Machines to predict house prices. Duplicates and missing values were removed, replaced N/A values , categorized date features, and decoded variables. Cross validation and grid search were performed. Decision Tree had an NRMSE=-0.205, Random Forst had an NRMSE of -0.144, and Gradient boosting Machine had an NRMSE=-0.126. [1]

3. Data – Preprocessing

The original dataset comes from Kaggle, created by Steve Zheng in 2018. There are 74,111 airbnb entries (rows) and 29 columns total. The first change we had to make dealt with the amenities column. The amenities each airbnb had were comma separated lists. Instead of dealing with encoding hundreds of different amenities, we changed this column to contain the number of amenities- calculated by counting the number of commas and adding one.

It was next necessary to drop irrelevant columns, as well as those with a large number of categorical missing(null) values. While initially analyzing our data, we found one property with a log_price value of 0.0, so we dropped that row as well. We can assume that this means the owner must be contacted for the exact price, which is not uncommon when using airbnb.



```
id                            0
log_price                     0
property_type                 0
room_type                     0
amenities                     0
accommodates                  0
bathrooms                   200
bed_type                      0
cancellation_policy           0
cleaning_fee                  0
city                          0
description                   0
first_review              15864
host_has_profile_pic        188
host_identity_verified      188
host_response_rate        18299
host_since                  188
instant_bookable              0
last_review               15827
latitude                      0
longitude                     0
name                          0
neighbourhood              6872
number_of_reviews             0
review_scores_rating      16722
thumbnail_url              8216
zipcode                     966
bedrooms                     91
beds                        131
dtype: int64
```

Figure 4: Data Null Counts

It is possible to deal with null values in numerical columns by assigning the null value to the mean (for example). Replacing unique columns with many null values such as the 'thumbnail_url' for a property is not possible. With the example of 'thumbnail_url', replacing the url with a different url would not make sense or benefit our model, so it made more sense to drop it. Therefore, we initially dropped the columns below, using the .drop() method[13]:

'id', 'first_review', 'host_response_rate', 'last_review', 'review_scores_rating', 'thumbnail_url', 'latitude', 'longitude'

It was next necessary to encode the true and false values into 1 and 0. This was accomplished using a dictionary to transform the 't' to 1 and 'f' to 0 in the columns 'host_has_profile_pic', 'host_identity_verified', and 'instant_bookable', then replacing all columns containing 't' and 'f' to the numerical values defined in the dictionary. The 'cleaning_fee' column did not have 't' and 'f' but instead had boolean True and False values, so we called the .astype(int) method on this column to transform the booleans to 0 and 1.

The next step to preprocessing the data was to deal with the categorical data. 'property_type', 'room_type', 'bed_type', and 'cancellation_policy' originally contained the following categorical values:

Property type: 'Apartment', 'House', 'Condominium', 'Loft', 'Townhouse', 'Hostel', 'Guest suite', 'Bed Breakfast', 'Bungalow', 'Guesthouse', 'Dorm', 'Other', 'Camper/RV', 'Villa', 'Boutique hotel', 'Timeshare', 'In-law', 'Boat', 'Serviced apartment', 'Castle', 'Cabin', 'Treehouse', 'Tipi', 'Vacation home', 'Tent', 'Hut', 'Casa particular', 'Chalet', 'Yurt', 'Earth House', 'Parking Space', 'Train', 'Cave', 'Lighthouse', 'Island'

Room Type: 'Entire home/apt', 'Private room', 'Shared room'

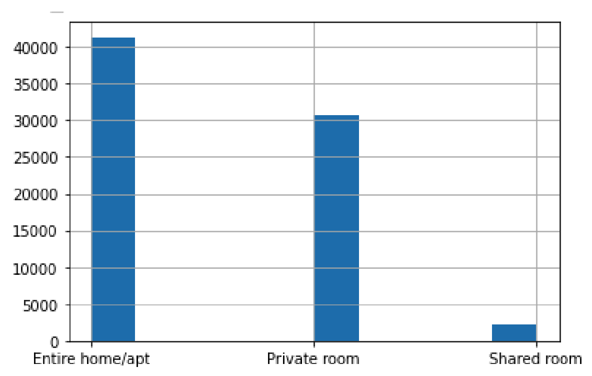


Figure 5: Room Types

Bed Type: 'Real Bed', 'Futon', 'Pull-out Sofa', 'Couch', 'Airbed'

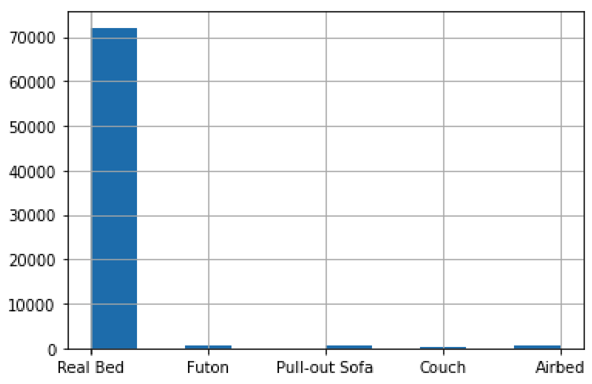


Figure 6: Bed Types

Cancellation Policy: 'strict', 'moderate', 'flexible', 'super_strict_30', 'super_strict_60'

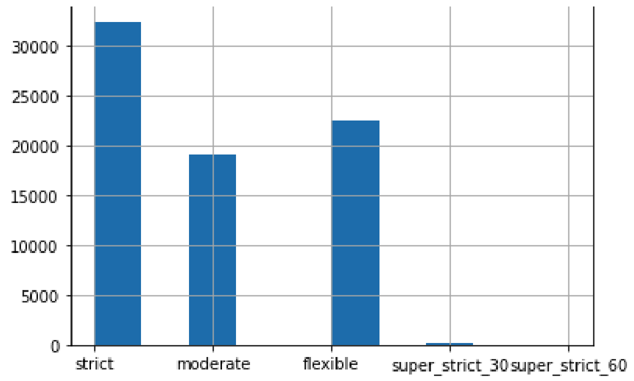


Figure 7: Cancellation Policies

City: ‘NYC’, ‘SF’, ‘DC’, ‘LA’, ‘Chicago’, ‘Boston’

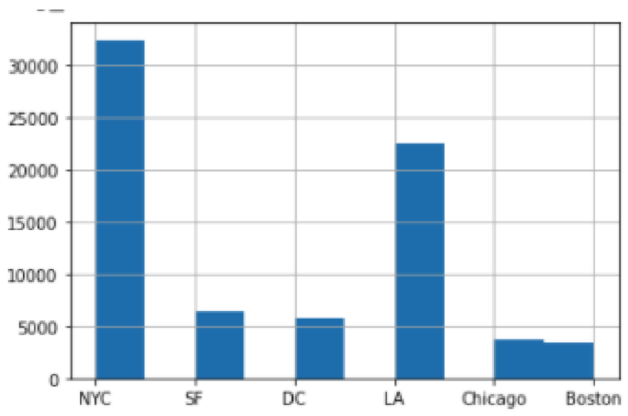


Figure 8: Cities

The ‘zipcode’ column contained 770 unique zipcodes, and the ‘neighbourhood’ column contained 620 unique neighbourhoods. Both of these columns were represented as strings, so it was necessary to change them to numerical categories as well.

We used the .cat.codes method [13] to change each string to a representative number, made a new column of the numerical categories (adding _cat to the column name), and dropped the categorical columns.

Though the categories with large numbers of null values were already dropped, it was necessary to replace the null values in ‘bathrooms’(200 null values), ‘bedrooms’(91 null values), and ‘beds’(131 null values). The number of bathrooms ranges from 0 to 7.5 with a mean of 1.23, number of bedrooms from 0 to 10 with a mean of 1.27, and number of beds from 0 to 18 with a mean of 1.71. The median and most frequent count for these three categories was 1.0, so we decided to replace the null values with the median value, using the .fillna() method [13]. 'zipcode_cat' had 966 null values, and 'neighbourhood_cat' had 6,872 null values. As these were reasonably important categories, we also filled the null values with the median.

'host_has_profile_pic' and 'host_identity_verified' both had 188 null values. Since both of these categories were true/false, we found the most common occurrence and replaced the null values with it. 'host_has_profile_pic' had 73,696 true and 226 false instances, while 'host_identity_verified' had 49,748 true and 24,174 false instances, so we replaced the nulls in both of these columns with true.

3.1 Feature Selection

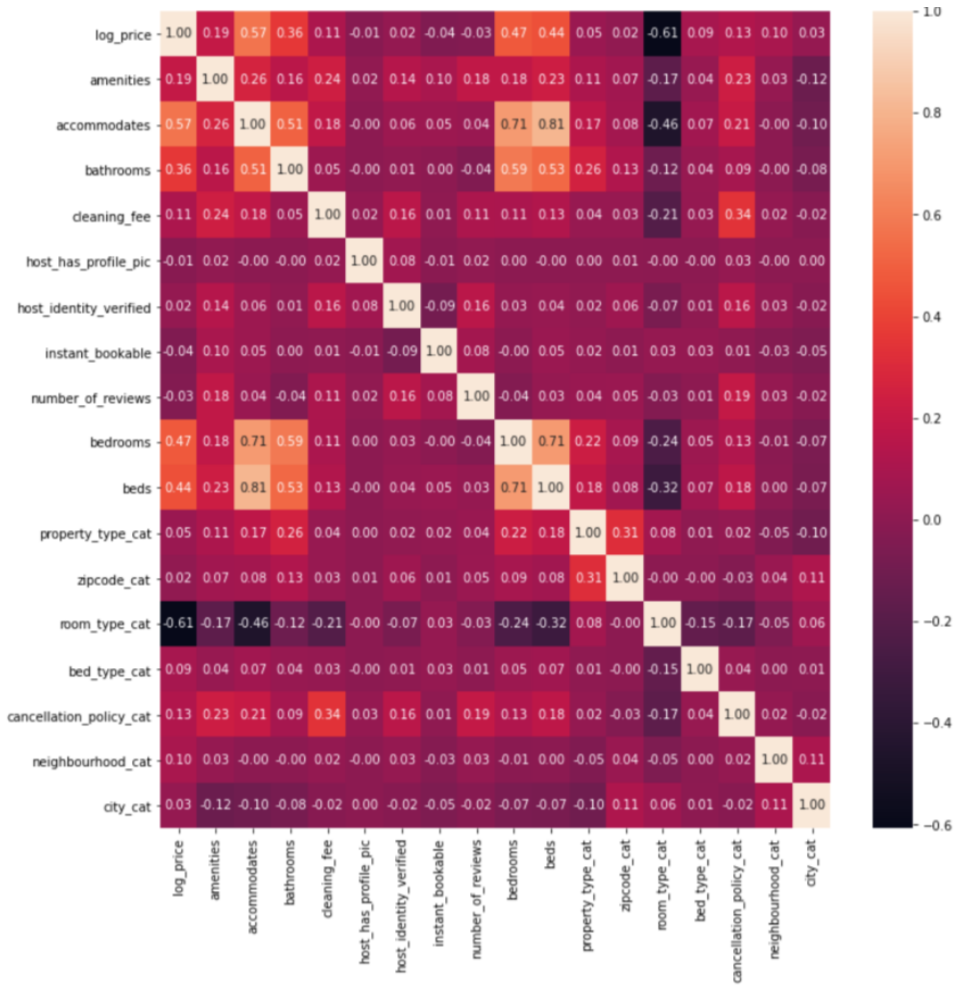
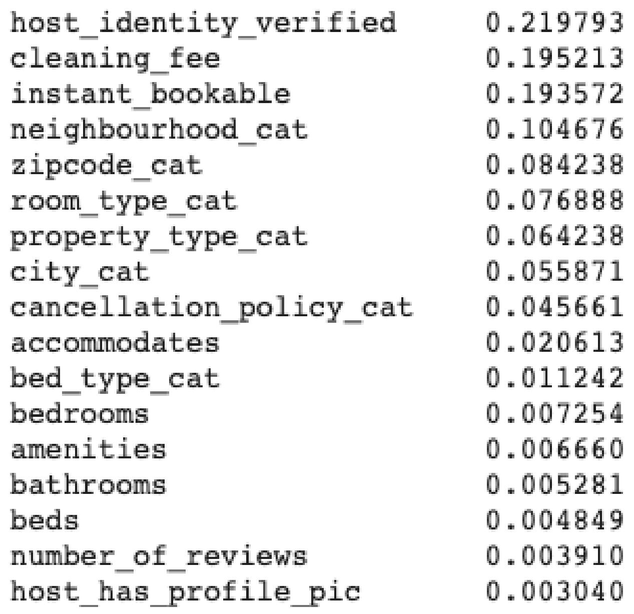


Figure 9: Variable Correlation Matrix

The first method of feature selection we experimented with was Variance Threshold Feature Selection. We used the MinMaxScaler from sklearn's preprocessing package [9] to calculate the variance of each feature, then used a threshold of 0.025 to determine which features should be selected: 'cleaning_fee','host_identity_verified', 'instant_bookable', 'property_type_cat', 'room_type_cat','zipcode_cat', 'cancellation_policy_cat', 'city_cat' and 'neighbourhood_cat'.



```
host_identity_verified     0.219793
cleaning_fee               0.195213
instant_bookable           0.193572
neighbourhood_cat          0.104676
zipcode_cat                0.084238
room_type_cat              0.076888
property_type_cat          0.064238
city_cat                   0.055871
cancellation_policy_cat    0.045661
accommodates               0.020613
bed_type_cat               0.011242
bedrooms                   0.007254
amenities                  0.006660
bathrooms                  0.005281
beds                       0.004849
number_of_reviews          0.003910
host_has_profile_pic       0.003040
```

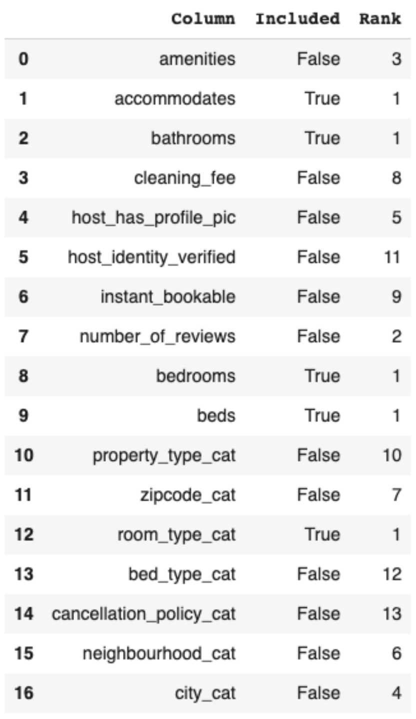
Figure 10: Variance Threshold Feature Selection Values

Recursive feature elimination (RFE) selects the most important n features "by recursively considering smaller and smaller sets of features"[9], until the desired number of features are left.

The results of RFE on the Linear Regression model with n=5 are below(Figure 11):

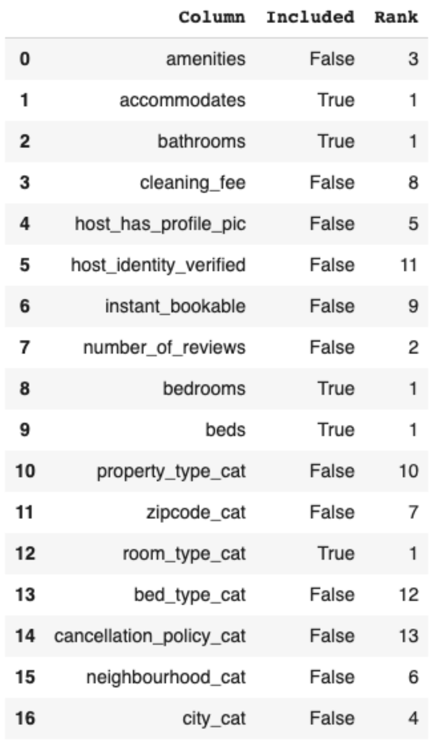
The results of RFE on the Random Forest model with n=5 are below(Figure 12):

The results of RFE on the Decision Tree model with n=5 are below(Figure 13):



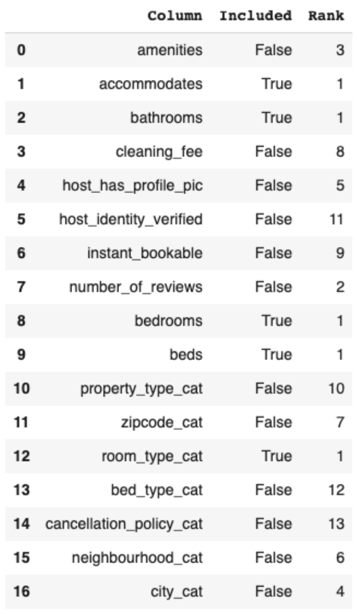
	Column	Included	Rank
0	amenities	False	3
1	accommodates	True	1
2	bathrooms	True	1
3	cleaning_fee	False	8
4	host_has_profile_pic	False	5
5	host_identity_verified	False	11
6	instant_bookable	False	9
7	number_of_reviews	False	2
8	bedrooms	True	1
9	beds	True	1
10	property_type_cat	False	10
11	zipcode_cat	False	7
12	room_type_cat	True	1
13	bed_type_cat	False	12
14	cancellation_policy_cat	False	13
15	neighbourhood_cat	False	6
16	city_cat	False	4

Figure 11: RFE (Linear Regression Model)



	Column	Included	Rank
0	amenities	False	3
1	accommodates	True	1
2	bathrooms	True	1
3	cleaning_fee	False	8
4	host_has_profile_pic	False	5
5	host_identity_verified	False	11
6	instant_bookable	False	9
7	number_of_reviews	False	2
8	bedrooms	True	1
9	beds	True	1
10	property_type_cat	False	10
11	zipcode_cat	False	7
12	room_type_cat	True	1
13	bed_type_cat	False	12
14	cancellation_policy_cat	False	13
15	neighbourhood_cat	False	6
16	city_cat	False	4

Figure 12: RFE (Random Forest Model)



	Column	Included	Rank
0	amenities	False	3
1	accommodates	True	1
2	bathrooms	True	1
3	cleaning_fee	False	8
4	host_has_profile_pic	False	5
5	host_identity_verified	False	11
6	instant_bookable	False	9
7	number_of_reviews	False	2
8	bedrooms	True	1
9	beds	True	1
10	property_type_cat	False	10
11	zipcode_cat	False	7
12	room_type_cat	True	1
13	bed_type_cat	False	12
14	cancellation_policy_cat	False	13
15	neighbourhood_cat	False	6
16	city_cat	False	4

Figure 13: RFE (Decision Tree Model)

3.2 Scaling

As previously mentioned, we used the MinMaxScaler [9] to scale the data for Variance Threshold Feature Selection. The MinMaxScaler scales each value individually, so that it will be between the feature_range, defaulted to zero and one:

X_std = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
X_scaled = X_std * (max - min) + min [9]
where min, max = feature_range.

We also experimented with the StandardScaler, also from the sklearn preprocessing package [9]. The standard score of a value is calculated with:

z = (x - u) / s
where u is the mean, and s is the standard deviation.

The data is split into training and test sets using train_test_split, an sklearn.model_selection method [9]. We used a test_size parameter of 0.3, resulting in a training set with 51,877 entries and a testing set with 22,234 entries.

4. Experiments

Seven regression models were created to predict AirBnB housing costs ($ in log): Linear Regression, Random Forest Regressor, Decision Tree Regressor, Adaboost Regressor that utilizes Linear Regression as the base estimator, Adaboost Regressor that utilizes Decision Tree Regressor as the base estimator, Bagging Regressor that utilizes Linear Regression as the base estimator, and Bagging Regressor that utilizes Decision Tree Regressor as the base estimator.

All of the models were first trained and tested using scaled data. For more information on how the data was scaled, please refer to Section 3, Preprocessing.
The Adaboost Regressor model contains the following parameters:

base_estimator=tree.DecisionTreeRegressor(), n_estimators=25, learning_rate=0.05

To summarize these inputs, the decision tree regressor is used as a base estimator to fit the data, in which the total amount of base estimators is 25. A weight of '0.05' is "applied to each classifier at each boosting iteration." [9].

base_estimator=linear_model.LinearRegression() The Adaboost Regressor model was also tested using linear regression as an estimate.

The Bagging Regressor model contains the following parameters:

base_estimator=tree.DecisionTreeRegressor(), n_estimators=25, max_samples=1.0, max_features=1.0

To summarize these inputs, the decision tree regressor is used as a base estimator to fit the data, in which the total amount of base estimators is 25. The number of samples and features to draw from X to train each base estimator is limited to 1.

base_estimator=linear_model.LinearRegression()

The bagging regressor model was also tested using linear regression as an estimate.

The Linear Regression, Random Forest Regressor, and Decision Tree Regressor models were additionally tested using only the output features from the Variance Threshold Feature Selection and the Recursive Feature Selection methods.

Only input features gathered from the variance threshold feature selection method were used to train and test the models. The target is Airbnb housing price ($ in log). Please refer to the Preprocessing section to obtain the list of features used.

VarianceThreshold(threshold=0.025)

The threshold was set to 0.025, so any feature in the training set with a variance lower than 0.025 will not be used in training the models.

The models were also tested with a threshold of 0. Finally, only input features gathered from the recursive feature selection method were used to train and test the models. The target is Airbnb housing price ($ in log). Please refer to the Preprocessing section to obtain the list of features used.

RFE(linearReg, n_features_to_select=5, step=1)
RFE(randomForest, n_features_to_select=5, step=1)
RFE(decisionTree, n_features_to_select=5, step=1)

For each instance, the estimators used to "provide information about feature importance" are LinearRegression(), RandomForestRegressor(), and DecisionTreeRegressor(). After each iteration, 1 feature is removed. After all the iterations have been performed, 5 features are selected.

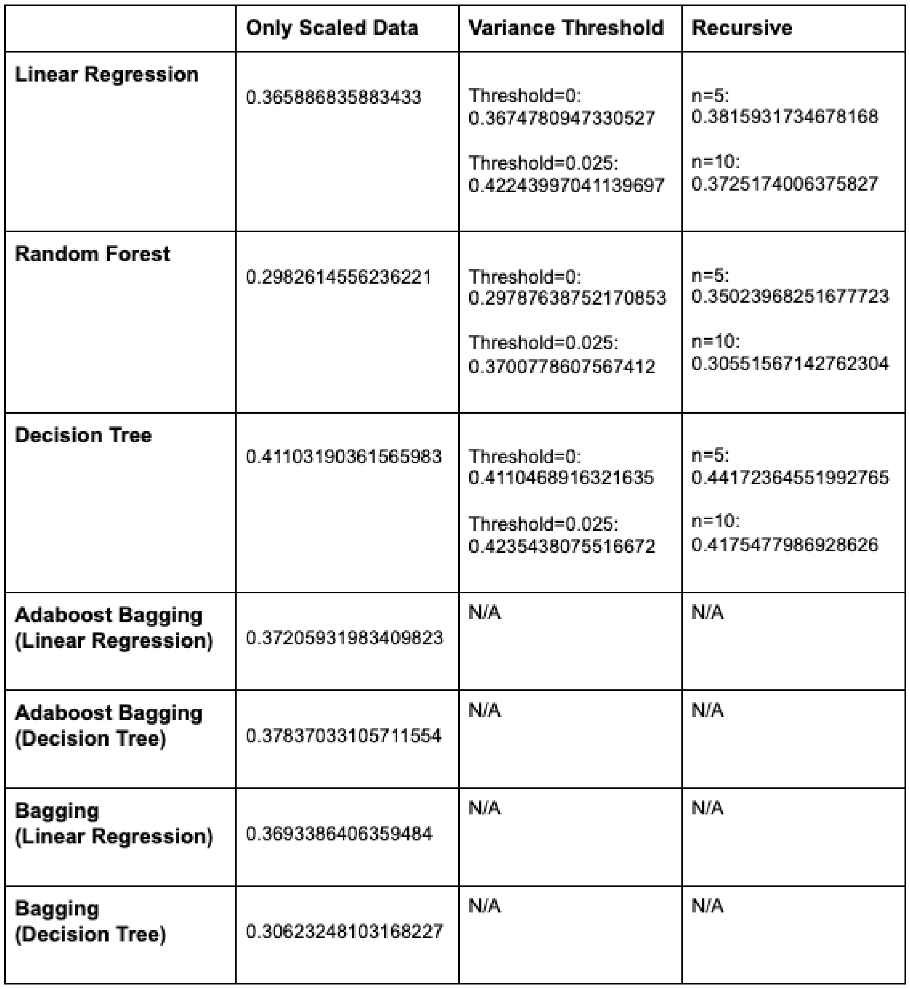
The models were also tested using 10 selected features after all of the iterations were finished.

All the regression models were tested using a 2015 MacBook Pro with 2.9 GHz Dual-Core Intel Core i5 processor in version 11.2.2 of macOS Big Sur and a 2018 MacBook Air with 1.6 GHz Dual-Core Intel Core i5 processor in version 10.15.4 of macOS Catalina. They were coded in Colaboratory, a notebook owned by Google that can run executable code and contain text. Git version control software, namely Github, was also used to remotely hold the dataset.

None of the ML frameworks required significantly more resources or memory than the others. However, the Recursive Feature Selection method (specifically when run on the Random Forest Model) took significantly more time (approximately 7 minutes) than the Variance Threshold Feature Selection method. No noticeable problems were encountered when running the experiments.

5. Analysis

The mean absolute error (MAE) was calculated for each model. For model comparison and analysis, the mean absolute error was used since it takes into consideration the error's magnitude in relation to other errors (standardized). Please note that the target feature 'Price ($)' was in log form.
These were the results for MAE Calculations:



	Only Scaled Data	**Variance Threshold**	**Recursive**

Linear Regression	0.365886835883433	Threshold=0: 0.3674780947330527
		
		Threshold=0.025: 0.42243997041139697
			n=5: 0.3815931734678168
			
			n=10: 0.3725174006375827

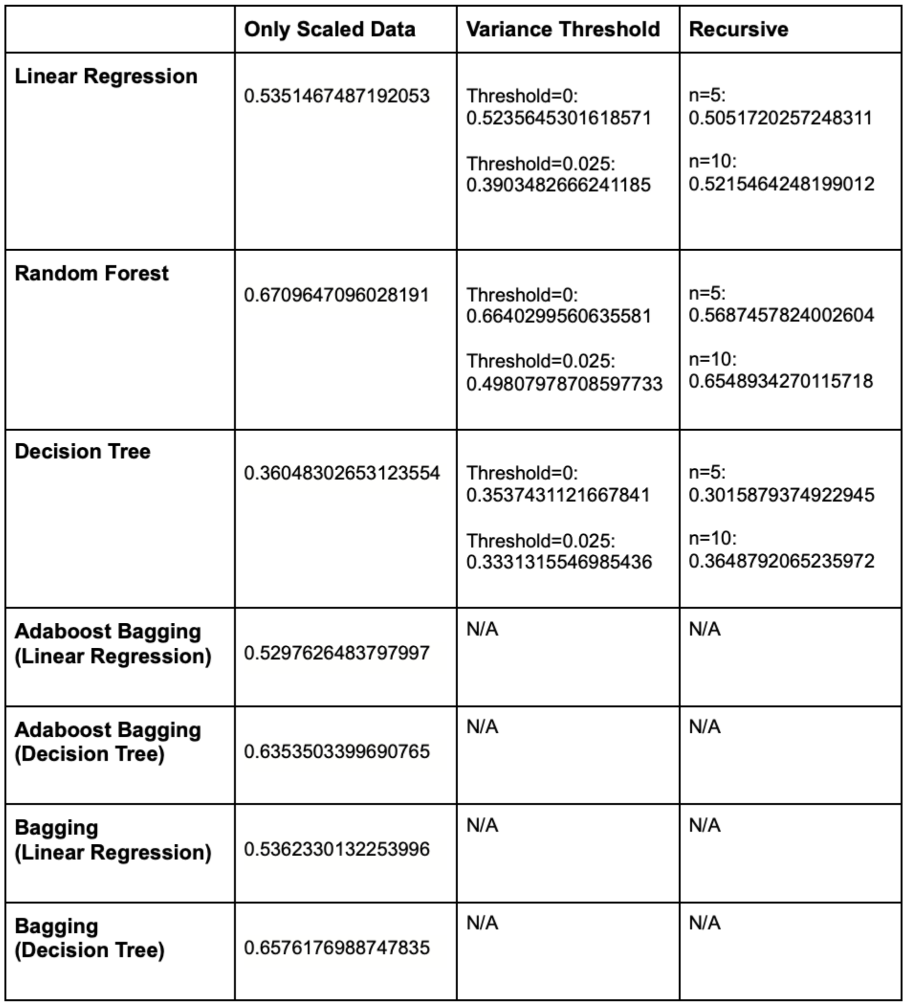
Random Forest	0.2982614556236221	Threshold=0: 0.29787638752170853
		
		Threshold=0.025: 0.3700778607567412
			n=5: 0.35023968251677723
			
			n=10: 0.30551567142762304

Decision Tree	0.41103190361565983	Threshold=0: 0.4110468916321635
		
		Threshold=0.025: 0.4235438075516672
			n=5: 0.44172364551992765
			
			n=10: 0.4175477986928626

Adaboost Bagging (Linear Regression)	0.37205931983409823	N/A	N/A
Adaboost Bagging (Decision Tree)	0.37837033105711554	N/A	N/A
Bagging (Linear Regression)	0.3693386406359484	N/A	N/A
Bagging (Decision Tree)	0.30623248103168227	N/A	N/A

Figure 14: Results(NOTE: All MAE's have a variation of +/- 0.1.)

and R_sq Calculations:



	Only Scaled Data	**Variance Threshold**	**Recursive**

Linear Regression	0.5351467487192053	Threshold=0:
		0.5235645301618571
		
		Threshold=0.025:
		0.3903482666241185
			n=5:
			0.5051720257248311
			
			n=10:
			0.5215464248199012

Random Forest	0.6709647096028191	Threshold=0:
		0.6640299560635581
		
		Threshold=0.025:
		0.49807978708597733
			n=5:
			0.5687457824002604
			
			n=10:
			0.6548934270115718

Decision Tree	0.36048302653123554	Threshold=0:
		0.3537431121667841
		
		Threshold=0.025:
		0.3331315546985436
			n=5:
			0.3015879374922945
			
			n=10:
			0.3648792065235972

Adaboost Bagging (Linear Regression)	0.5297626483797997	N/A	N/A
Adaboost Bagging (Decision Tree)	0.6353503399690765	N/A	N/A
Bagging (Linear Regression)	0.5362330132253996	N/A	N/A
Bagging (Decision Tree)	0.6576176988747835	N/A	N/A

Figure 15: R_sq calculates the model's explained variability. For example, 53.51% of the variation in Airbnb housing price predictions is explained by linear regression. It correlates with MAE. The lower the MAE, the higher the R_sq.

However, for model comparison and analysis, we will refer to the MAE. The order of the model that only used scaled data with the best performance (lowest MAE) to the worst performance (highest MAE) is as follows: Random Forest Regressor, Bagging using the base estimate Decision Tree Regressor, Linear Regression, Bagging using the base estimate Linear Regression, Adaboost Bagging using the base estimate Linear Regression, Adaboost

Bagging using the base estimate Decision Tree, and Decision Tree.

The order model that used selected features from the Variance Threshold Feature Selection with the best performance (lowest MAE) to the worst performance (highest MAE) is as follows: Random Forest Regressor with threshold=0, Linear Regression with threshold=0, Random Forest Regressor with threshold=0.025, Decision Tree Regressor with threshold=0, Linear Regression with threshold=0.025, and Decision Tree Regressor with threshold=0.025.

The order model that used selected features from the Recursive Feature Selection with the best performance (lowest MAE) to the worst performance (highest MAE) is as follows: Random Forest Regressor with n=10, Random Forest Regressor with n=5, Linear Regression with n=10, Linear Regression with n=5, Decision Tree Regressor with n=10, and Decision Tree Regressor with n=5.

Although the estimates were not significantly different between the machine learning schemas, the Random Forest Regressor models tend to perform the best, followed by the Bagging models, Linear Regression models, Adaboost Bagging models, and the Decision Tree Regressor tends to perform the worst. Also, lower MAE is possibly due to:

-low variance threshold and/or
-high number of features selected

The results are significant, because only the most statistically important feature inputs were utilized to train the models. Also, the feature inputs were scaled.

6. Conclusion

Overall, the order of the models from best (lowest MAE) to worst (highest MAE) is Random Forest Regressor models, Bagging models, Linear Regression models, Adaboost Bagging models, and Decision Tree models. The current work can be extended to predicting the location or neighborhood of the Airbnb housing based on price, amenities, property type, etc.
Other possible future works include testing the models with different parameters, such as changing the base estimators for the ensemble models. As mentioned in Laura Lewis'[7] related work, analyzing images of airbnb's would be an interesting extension. It would be assumed that the higher quality of images and quality of furniture within each property should have a correlation with a higher price. Analyzing the textual data (specifically description) such as in Max B's [2] related work could be another place of exploration. He used a Random Forest Regressor on the description text with an MAE of 0.47, but it would be interesting to experiment with other models, such as Naive Bayes.

References

[1] John Ade-Ojo. *Predicting Housing Prices using Cross Validation and Grid Search in Regression Models.* URL: `https://towardsdatascience.com/predicting-house-prices-with-machine-learning-62d5bcd0d68f`.

[2] Max B. *Airbnb: EDA and price prediction.* Dec. 2020. URL: `https://www.kaggle.com/max398434434/airbnb-eda-and-price-prediction#Dense-neural-models`.

[3] Graciela Carrillo. *Predicting Airbnb prices with machine learning and location data.* URL: `https://towardsdatascience.com/predicting-airbnb-prices-with-machine-learning-and-location-data-5c1e033d0a5a`.

[4] Meitar Goldenberg. *NYC Airbnb (Regression Project).* May 2020. URL: `https://www.kaggle.com/meitargoldenberg/nyc-airbnb-regression-project#4.-Machine-Learning`.

[5] Yohan Jeong. *Predicting Housing Prices using Cross Validation and Grid Search in Regression Models.* URL: `https://yjeong5126.medium.com/predicting-housing-prices-in-melbourne-e3d5f49abf20`.

[6] Sujith Kumar. *sujith$_a$irbnb.* Apr. 2020. URL: `https://www.kaggle.com/sujithkumar/sujith-airbnb`.

[7] Laura Lewis. *Predicting Airbnb prices with machine learning and deep learning.* May 2019. URL: `https://towardsdatascience.com/predicting-airbnb-prices-with-machine-learning-and-deep-learning-f46d44afb8a6`.

[8] Ewurama Minka. *Mean Absolute Error MAE [Machine Learning(ML)].* Feb. 2018. URL: `https://medium.com/@ewuramaminka/mean-absolute-error-mae-machine-learning-ml-b9b4afc63077`.

[9] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[10] *Predicting Price of Airbnb Listings in NYC - Part 1.* URL: `https://www.kaggle.com/spuchalski/predicting-price-of-airbnb-listings-in-nyc`.

[11] Raj Sankhe. *Airbnb$_A$nalysis.* Dec. 2018. URL: `https://www.kaggle.com/rajsankhe03/airbnb-analysis/execution`.

[12] Subham Sarkar. *Predicting House prices using Classical Machine Learning and Deep Learning techniques.* URL: `https://medium.com/analytics-vidhya/predicting-house-prices-using-classical-machine-learning-and-deep-learning-techniques-ad4e55945e2d`.

[13] The pandas development team. *pandas-dev/pandas: Pandas.* Version latest. Feb. 2020. DOI: `10.5281/zenodo.3509134`. URL: `https://doi.org/10.5281/zenodo.3509134`.

[14] Steve Zheng. *Airbnb price prediction.* Feb. 2018. URL: `https://www.kaggle.com/stevezhenghp/airbnb-price-prediction`.