Predicting Airbnb Prices

Capstone Project
By Malak Mosly

Problem Statement

- Airbnb is a very popular American company that helps people find houses to rent.
- It is important for a company like Airbnb to offer housing with fair prices in order to remain reputable and competitive.
- Accordingly, we built a Random Forest Regression model that can predict Airbnb house prices.
- The model can help Airbnb regulate prices on its platform.

Dataset Information

- The dataset was obtained from Kaggle and contained 226,030 rows and 17 columns of Airbnb prices in 2020.
- The columns are as follows:
    - **Id** - The listing id or house id
    - **Name** - The name of the listing or house
    - **Host_id** - the id number of the person listing the house
    - **Host_name** - the name of the person listing the house
    - **Neighbourhood_group** - the neighbourhood group the house is in
    - **Neighbourhood** - the neighbourhood the house is in
    - **Latitude** - coordinates of the house
    - **Longitude** - coordinates of the house
    - **Room_type** - the type of room being offered in the listing
    - **Price** - the price of the listing
    - **Minimum_nights** - the minimum amount of nights the house can be rented
    - **Number_of_reviews** - the number of reviews on the house from previous renters
    - **Last_review** - the date of the last review on the house
    - **Reviews_per_month** - how many reviews a particular house gets each month
    - **Calculated_host_listings_count** - how many listings the host has on Airbnb
    - **Availability_365** - the availability of a listing throughout the year
    - **City** - the city the house or listing is located in

Features Used for Prediction

- There were 7 features used by the model to predict prices:
  - room_type
  - minimum_nights
  - availability_365
  - city
  - calculated_host_listings_count
  - reviews_per_month
  - number_of_reviews

Project Steps

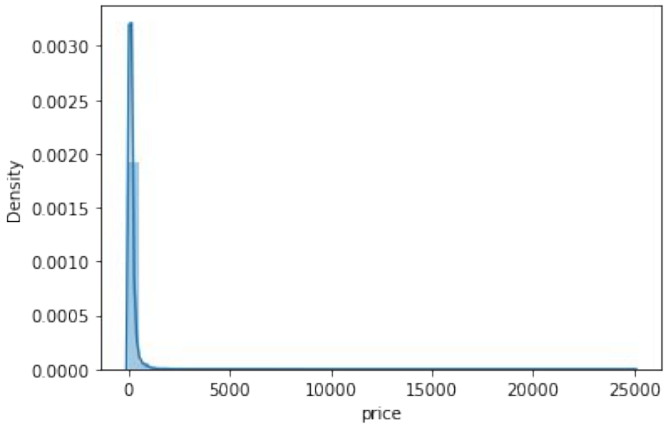
1. Data Wrangling and Cleaning
2. Exploratory Data Analysis
3. Preprocessing and Training
4. Modeling

Data Wrangling and Cleaning

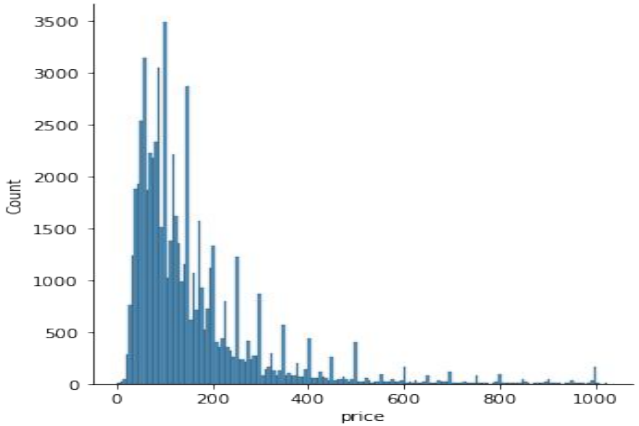
- The original dataset contained 226,030 rows and 17 columns.
- The dataset had a lot of missing values in certain columns.
- The first step of this project was to remove all missing values from the dataset, which left us with 85,144 rows and 17 columns of data.
- The data also had a lot of outliers. Outliers were removed from every column, leaving us with 62,808 rows and 17 columns.

Distribution of prices

Before outlier detection and removal, the distribution of prices looked like this:



After outlier detection and removal, the distribution of prices looks like this:



Exploratory Data Analysis (I)

- The distributions of each column was explored using seaborn's .distplot() function and pandas plt.hist() function.
- Most columns had a skewed distribution even after outlier removal.
- A heatmap of the correlations between columns was explored using seaborn's .heatmap() function.
- It was found that the latitude and longitude were highly correlated with each other.
- Reviews_per_month and number_of_reviews were also highly correlated.

Exploratory Data Analysis (II)

- Columns with a high correlation with one another are bad for our model because they can cause bias when predicting the price of a listing.
- Latitude and longitude had the highest correlation with each other, and they are also least likely to be useful when predicting price. They were therefore removed from our DataFrame.
- Id, host_id, name, and host_name, were also dropped because host information is not likely to be a helpful feature in price prediction.
- We were left with 62,808 rows and 11 columns to work with.

Preprocessing and Training

This step of the project involved:

- Splitting the data up into categorical and numerical variables.
- Creating dummy variables for categorical variables of interest (room_type and city).
- Scaling numerical variables using sklearn's StandardScaler() function.
- Combining scaled numerical and categorical variables into a single DataFrame and making that our X variable.
- Isolating price as the target feature, making it our y variable.
- Splitting the data up into a train/test split (30% test, 70% train split)
- Our X variable had 14 columns and 62,808 rows.
- Our y variable had 1 column (price) and 62,808 rows.

Modeling (I)

- Three models were used in this project:
  - Linear Regression Model
  - Random Forest Model
  - K Nearest Neighbor Regression Model
- All three models were cross-validated, and had hyperparameter tuning done with GridSearchCV. Performance assessment was done for each model and results were compared to determine the best performing model.

Modeling (II)

- After hyperparameter tuning with GridSearchCV for each model, cross validation, and performance assessment, it was determined that the best performing models were the K Nearest Neighbor and Random Forest Regression models.
- Performance assessment included calculation of R2 scores and Mean Absolute Error.
- KNN and Random Forest Regression Models had the lowest Mean Absolute Error, with KNN having a slightly lower MAE.
- The Random Forest Regression Model had the lowest standard deviation of the MAE out of all three models.

Modeling (III)

Below is the table that assesses the mean absolute error and mean absolute error standard deviation of each model.

Model Name	Mean absolute error	Standard Deviation
KNN Model	75.2	1.59
Random Forest Model	75.44	1.18
Linear Regression Model	77.61	1.4

The KNN and RF Models perform better than the Linear Regression Model. Both have a lower MAE by about $2.

Model Predictions Comparison

- Below is a snippet of price prediction comparisons between the RF Model and the K Neighbors Regressor Model.
- Comparing the first three data points, we can see that the first price prediction by the RF Model (ypred_rf) is $128.55, when the real price (y) is $124. The KNN model predicts $121.11. The KNN model's prediction is more accurate for the first data point. The RF Model is more accurate in its second and third price predictions however.

```
1 ypred_rf = rf_best.predict(X_test)
2 ypred_rf
```

```
array([128.549     , 203.819     ,  84.44528532,
       201.865     , 213.718     ])
```

```
1 ypred_knn = model_best.predict(X_test)
2 ypred_knn
```

```
array([121.11111111, 273.11111111,  62.22222222,
       310.77777778, 183.        ])
```

```
1 y
```

```
0    124
1    239
2    120
```

Model Selection

- The accuracy in price prediction between the KNN and RF models were very close; only $0.22 difference.
- However, the RF model had a noticeably lower standard deviation of its MAE, as seen in the previous table.
- If no further tuning was to be done on these models, the recommendation would be to use the RF Model over the KNN Model because it has a similar level of accuracy but lower variability in its predictions.

Model Usage Suggestions

- The model can be used by Airbnb to regulate listed prices.
- The model can be used by individuals listing their own homes, outside of Airbnb, to get a good price estimate.
- The model can be used by other companies that wish to compete with Airbnb.

Future recommendations and improvements

- The model should be expanded to include every city in the United States.
- An ensemble model could be created to expand to every city and make price predictions.
- The model can also be expanded to include more years than 2020 alone. It could be expanded to about 5 or 10 years. This would allow us to analyze the trends in rental prices over time.