Predicting Airbnb Listing Price | Milestone Report

Claire Miles

Problem Statement:

Today's world is more connected than ever. With increasingly long nonstop flights, spontaneous translation apps, and social media friendships spawning worldwide, it's easier than ever to be a global citizen. One company catalyzing connections is Airbnb, which provides a service that allows property owners to rent their houses, apartments, or rooms in their hometowns, offering tourists a unique way to lodge in a new city - one that is arguably more authentic than lounging at a luxury resort.

By creating these home-away-from-homes, Airbnb has skyrocketed in popularity for those looking to live like locals and be more culturally connected to their destination. It's in Airbnb's best interest for its listings to be the best they can be, not only to crete an optimal user experience, but also to maximize the prices of their listings. Therefore, it's very useful to know what goes into a successful listing, and to be able to predict the price of a listing from its attributes. This will help Airbnb increase its revenue and help hosts optimize the value of their listings by predicting their success.

Project Inquiry:

How can we predict an Airbnb listing's price based on other information about that listing?

Dataset Description:

I am using data from Inside Airbnb (http://insideairbnb.com/get-the-data.html), an independent, non-commercial project that collects public data from the travel and accomodations company Airbnb. Data is collected monthly from major cities and metropolitan regions around the world, and includes information about that city's listings, reviews, and calendar data. I am focusing on data about Airbnb listings and the reviews for those listings. The listings data contains almost 100 columns of attributes for each listing, including number of bedrooms and bathroom, square footage, amenities provided, host information, etc. The reviews data includes the text of the reviews left for each listing. The information between the two datasets can be linked using provided ID numbers.

Data Wrangling:

Data Importing:

Since I am using data sourced from many files found on the same webpage, I decided to write a script that scraped that page for the relevant URLs to my project.

To complete this task, I used the requests and BeautifulSoup packages. First, I used requests to capture the response from the url where all of the csv files are hosted: http://insideairbnb.com/get-the-data.html. After catching that response in a variable, I read the text into another variable. I turned the text variable into a BeautifulSoup object using the BeautifulSoup function. From there, I could extract the text from the HTML anchor tags, where

the csv links are found in the 'href' attribute of the tag. I stored the names of the csv urls in a list. Since the links led to gzipped (.csv.gz) files, I called the list 'zipped_links'.

Next, I wrote the information from the online gzipped csv files to local files. Using a context manager, I looped through zipped_links to create custom filenames for each list item and write to that file. The filename includes the city name, the date the data was collected, and the information category (listings or reviews).

Data Combination:

After the raw files were scraped, created, and stored, the data was further consolidated into the listings and reviews categories in the form of the pandas dataframe. Since the scraped data represented data collected monthly from hundred of cities and was incredibly large, I decided it would be a better idea to focus my initial analysis on just one city: Los Angeles.

Because the data from the website is collected monthly, there were several raw files to process with hundreds of thousands of lines of data. Therefore, it helped to create functions that did the heavy lifting:

- **consolidate_data** checks if the csv file for either listings or reviews data has been created for the designated city. If the file has not been created, run the combine_listings or combine_reviews function for that city, and then create the csv file for that city.
- **combine_listings, combine_reviews** goes through files in the directory and checks for the designated city listings or reviews files. The it appends the names of the listings or reviews files of that city to a list, and passes the list and the directory name to the concat_files function.
- **concat_files** creates a pandas dataframe for each file name in the list of files, then adds the date recorded as a column in that dataframe (taken from the file name). Then it appends the dataframe to a list of dataframes. After all files in the list have been converted to pandas dataframes, it concatenates the dataframes together, drops duplicate rows, and resets the dataframe index.
- **export_csv** checks if the desired csv file does not exist in the current working directory, then converts the dataframe to a csv file and moves the the desired folder in the destination directory.

Even though I am starting by analyzing one city, having functions will also allow me to scale my analysis to multiple cities in the future.

After I created the functions, I defined the directory (the pathway where the raw data is stored) and destination folder (where I wanted to store the concatenated data on the computer) and ran the functions. The outcome was a single listings file and single reviews file for Los Angeles.

Data Cleaning:

After combining the data into single csv files for Los Angeles, I still needed to clean and explore the data.

Datatypes:

First, I worked with the listings data, which initially had 97 different columns of float and object datatypes. From an initial inspection of the data, there were many columns that could be converted to more suitable datatypes. For example, price data was stored in strings along with

the ‘$’ character - this data was converted to numeric datatypes with the dollar sign being added to the column name. Additionally, string data that only had a few unique values, like in the ‘property_type’ column, was converted into categorical data. To make the datatype conversion process easier, I created a function **change_datatypes**. After running the function, I was able to reduce the size of the dataframe from 4.9GB to 3.8GB.

Which column to predict?:
In the proposal for this project, I was still deciding whether to create an algorithm to predict Airbnb listing prices or average review scores. I ultimately decided to choose the metric that contained the least missing data. After calculating the percentage of data missing for both, I found that the price data is only missing 2% of data, while review scores is missing 23%. Therefore, I decided to move forward with predicting price data.

Inspecting Price Data:
Since it would not help to train an algorithm on data without the target metric present, I removed rows from the dataframe without price data. Then, I ran the **describe()** function on the data to get some summary statistics. With a relatively low mean and very high maximum value, it was already apparent that the price data was skewed, and that I should inspect further for outliers. After running the **value_counts()** function, I found that 207 listings had a price of $0 and 36 listings had prices above $90,000, with the next highest price being all the way down at $999. These high and low values immediately seemed suspicious.

Luckily, the data included URLs for each listing. Looking at listings whose prices were listed at over $90,000, I found that the URLs led to legitimate listings, but the data entry seemed off in the dataframe. I decided to attribute these 36 erroneous values with data entry error and removed them from the dataset. For listings that were $0, I found that only some of them existed, while the others were probably also a data entry fluke. I decided to remove this small amount of data as well.

After removing outliers, I created a histogram and boxplot of the cleaned price data. From the visualization, it's obvious that the data is very skewed. However, the previous analysis showed that the outliers are likely real listings. Expensive listings are likely more rare for a service like Airbnb, where most customers are generally looking for affordable prices.

Tidying Data:
Several columns in the dataset presented data cleaning challenges, mostly due to data entry errors like misspellings and inconsistent capitalizations of words. Categorical data is the most likely candidates for these types of errors, so I corrected for false and inconsistent data in the zipcode, property type, room type, bed type, cancellation type, city, and state columns.

I also converted columns with more than one type of information into multiple tidy columns. For example, the ‘amenities’ column made the dataset untidy because it contained lists of different features in a single column. Using a function I created called **has_feature()**, I separated unique values of these lists into new boolean columns that displayed whether a listing either has or doesn’t have that amenity. I then repeated that process with a column with a similar structure, ‘host_verifications’.

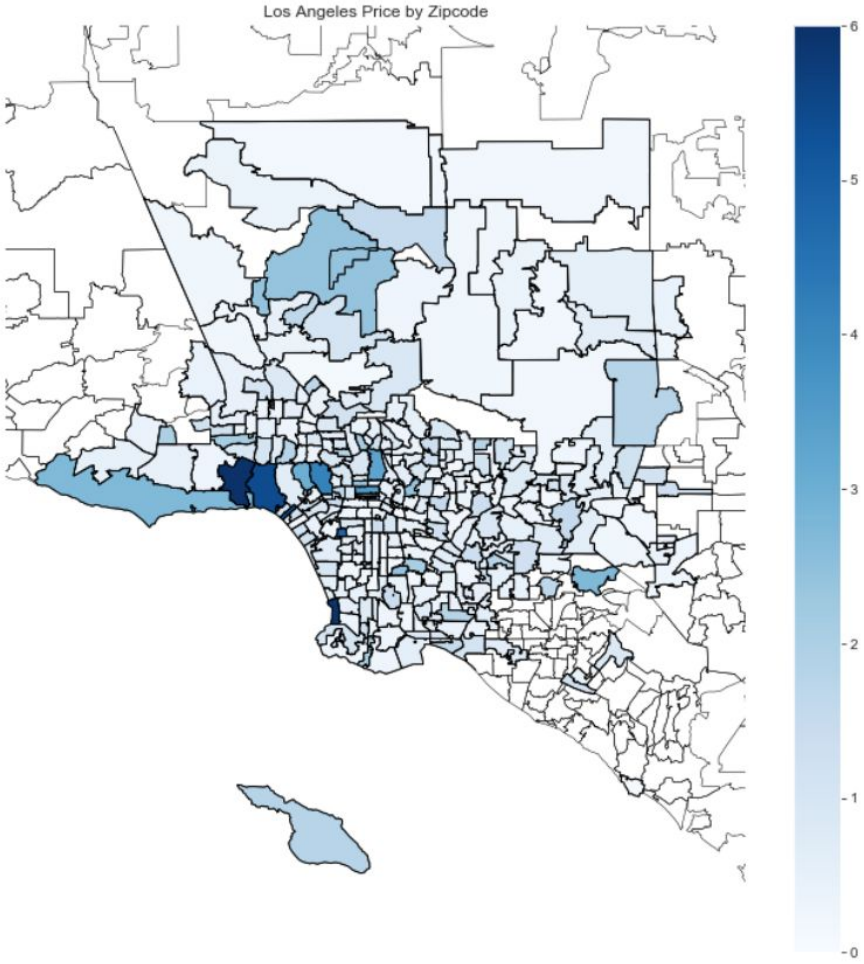
I finished the analysis by saving the data as a csv file to my computer to preserve the changes I made during the data cleaning process.

Reviews Data:
The reviews data only has 8 columns, the most important being the ‘comments’ column that contains the text of the review. I plan to use the text to create a score that indicates the average positivity/negativity of the reviews for a particular listing. Since most NLP algorithms are based on the English language, it’s outside the scope of this project to do sentiment analysis all the languages present in the dataset. Therefore, assuming that the majority of the reviews are in English, I removed the non-English rows in the dataframe using the langdetect python package. I also converted the ‘date’ row to the more suitable datetime datatype before saving the dataframe to a csv file on my computer.

Exploratory Data Analysis:

In the initial part of the data analysis, I created a series of visualizations to see how different features are distributed across the Los Angeles landscape.

Price distributed by zip code



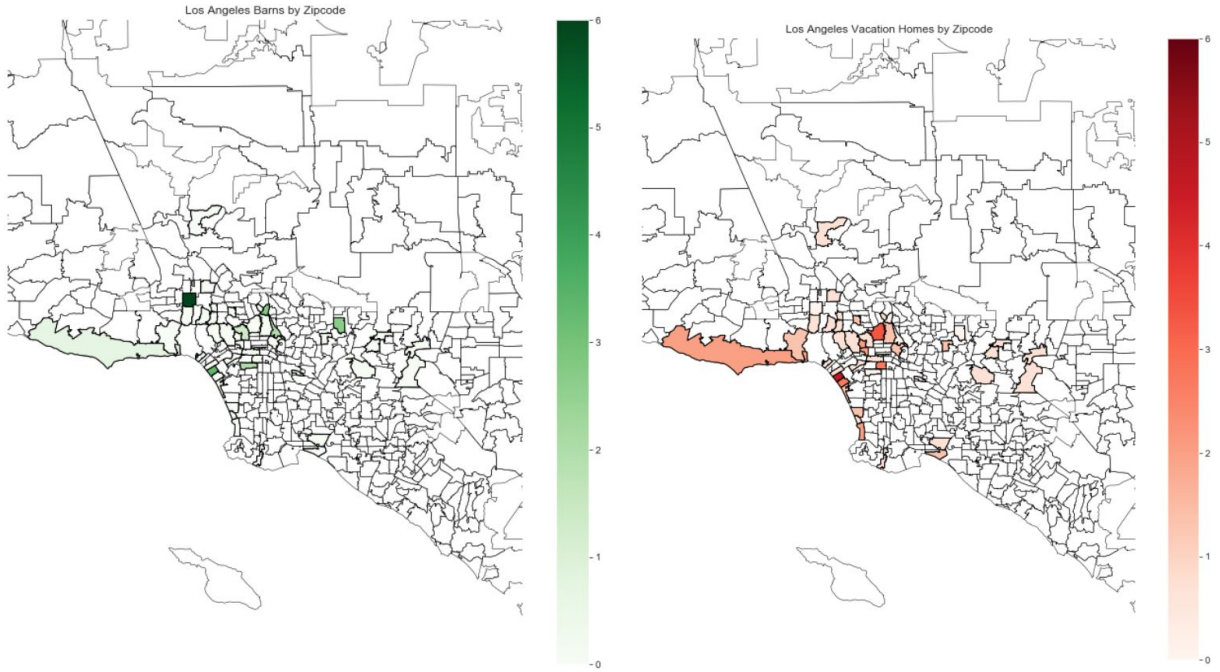
The above map displays how expensive or inexpensive Airbnb listings tend to be in a particular zipcode of Los Angeles, with higher values on the colorbar corresponding to more expensive

listings. It looks like 90290, 90272, and 90263 have very expensive listings - this makes sense because these zip codes corresponds to Calabasas, Pacific Palisades, and Malibu, some of the richest, most luxurious areas in the city.

Square footage vs. price by property type

There are so many different property types that it's difficult for the human eye to distinguish, so we will focus on the property types that are most represented in the dataset (listed above). From the graph, it looks like houses and apartments are the most diverse in price and square footage range, while guest houses and suites tend to be smaller and cheaper. Townhouses are in the mid-range of square footage and price.

Barns and Vacation Homes by Zip Code



It looks like barns show up in the most in the zip code 91335, which corresponds to the neighborhood of Reseda, which (according to Wikipedia) was devoted to agriculture for many years. Vacation homes are more frequently found in 90068, 90291, and 90292. These zip codes correspond to the Hollywood Hills Venice, and Marina Del Rey, all of which upscale communities and tourist destinations in Los Angeles.`

The trends and relationships uncovered in this notebook led to a few conclusions/ideas for further exploration:

1. There is a lot of interesting geographic variability in prices, and this may influence price more than other aspects of the data. On the other end of the spectrum, attributes that I thought would have more influence, like number of bathrooms and square footage, are not as strong as expected.
2. Listings of certain property types tend to have similar pricing and size, with the exception of a few (houses, apartments).
3. It will be interesting to see if there are any NLP techniques that might be applied to the text data that describes the listings, in addition to using NLP on each listings reviews. This could add more detail to the parts of the listings that are not as quantifiable.

From this, I concluded that an Airbnb listing's nightly price will be most affected by its geographic location and its property type.

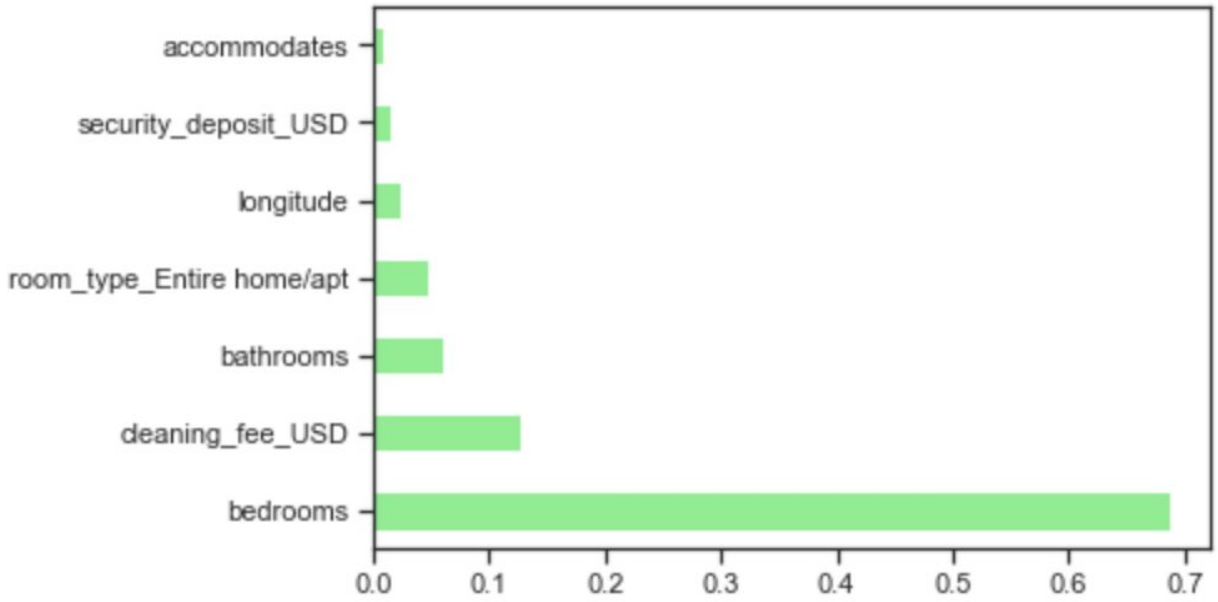
After cleaning the data and starting to craft the beginning of my data story, it was time to enrich my exploration with further analysis. My exploratory data analysis included the following steps:

1. Identify important features with a first-pass random forests model.
2. Look at the distributions of the most important features.
3. Re-evaluate the original project question based on the results.

Once again, I focused on the listings dataset in this notebook, as the listing data hosts all numerical features that the dataset currently has. The reviews data, which contains the review text for different listings, will be interpreted using NLP later on in the project process, as will other text data in the listings file.

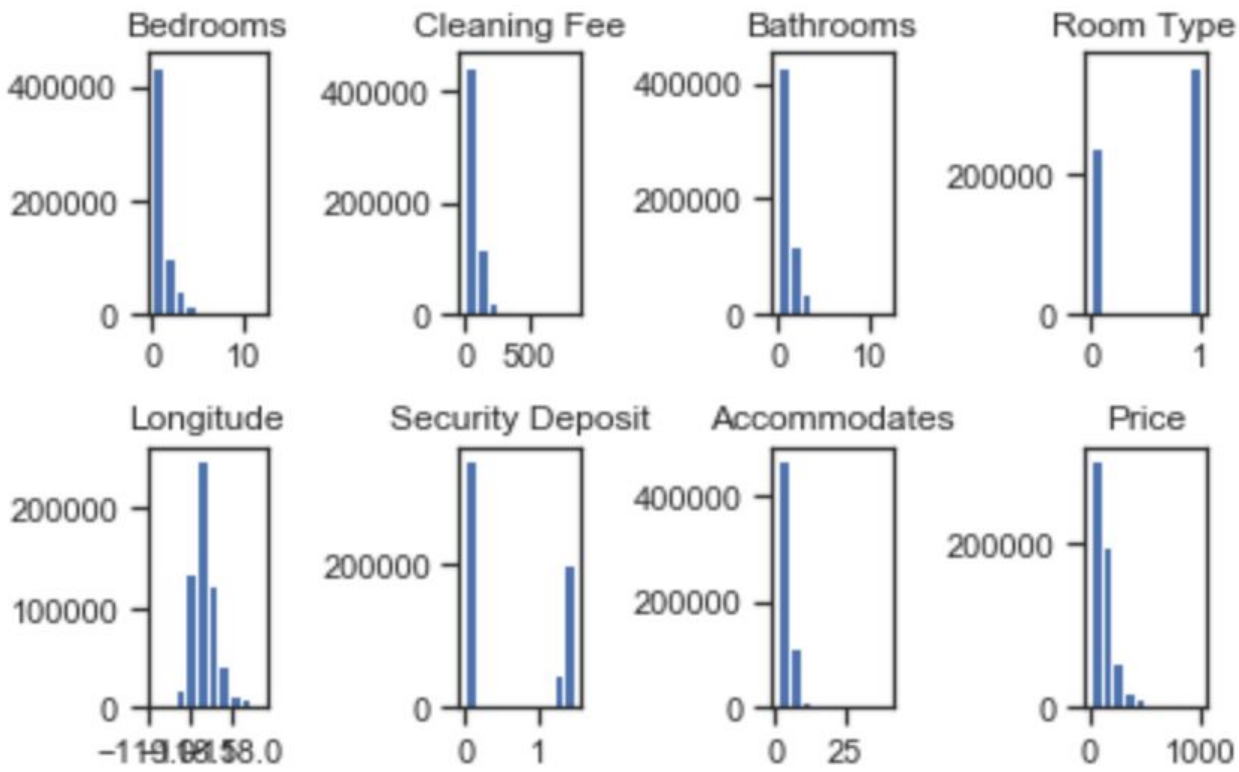
Random Forests:
I briefly prepared the data for the random forests training by converting certain attributes of the dataset to categorical data type, turning booleans into 0/1 values, and selecting only the numeric, datetime, and categorical data types in the dataframe. To get rid of NaN values, I used pandas interpolate function to fill in the values linearly (I'll give more thought to interpolation in the training of the final model, but here I'm just looking for a crude first pass). Then, I trained a RandomForestRegressor classifier on one-third of the data and plotted the features by their importances.

Random Forest Feature Importances



The random forest found the most important features to be bedrooms, cleaning fee, bathrooms, room type-entire home/apartment, longitude, security deposit, and accommodates. It's important to note, however, that this was a very simplistic first-pass at the data, and that the feature importance algorithm for random forests shows some bias towards continuous features or high-cardinality categorical variables.

Next, I plotted histograms of the features and the target variable to get a better picture of their distributions:



It appears that longitude is the only normally distributed feature, with bedrooms, bathrooms, cleaning fee, accommodates, and price being right skewed. This makes sense since Airbnb listings are spread throughout the entire city, with some cities being a bit more popular than others. Also, most Airbnb listings are small, affordable, and for hosting just a few people - the smaller amount of larger listings creates the skew of the histogram. Room type is a categorical variable that has been transformed in the one hot encoding process, therefore the histogram just shows that there are more entire homes/apartments than not. For security deposit, it looks like most listings do not have one, but those that do may give us important insights into a listing's price.

Re-evaluation of project question:
The biggest finding of my statistical analysis is that I've identified several of the most important features in the dataset and looked at their distributions. This will inform which features to pay the most attention to in the next stage of the project.

Moving forward, the machine learning analysis will likely offer more complex insights than this statistical analysis, especially as I incorporate information from the reviews data. It may be the case that some features that are not significant in this analysis play a larger role down the line.

This analysis does not alter the initial project question of trying to decipher a listing's price from its other features.