Sourcing The Data

I was able to acquire the data for my project from Inside Airbnb, which is an independant, non-commercial set of tools that allows you to explore how Airbnb is being used in various cities around the globe. They scrape publicly available data from Airbnb which includes listing descriptions, reviews, calendar data and much more. Inside Airbnb has pre-prepared .csv files as well as geo-spatial data in the form of geo-json files, which can be easily downloaded for free directly from their website. There is also another site, AirDNA**,** which scrapes higher quality data for a fee.

Data Cleaning and Preprocessing

I downloaded the Austin, TX airbnb dataset directly from Inside Airbnb, which includes updated information up to February 19th, 2020. This dataset consisted of 11,151 listings with 78 distinct features. I was easily able to load the data into a Pandas dataframe using Jupyter notebooks. I also dropped 31 initial features that seemed futile for the goal of the project.

Missing Data

After dropping my initial features, I wrote a simple “for” loop to print out the feature names and percentage of missing data for those features. Luckily it was evident that most of the features had either no missing data points or less than 25% of the data was missing. There were however a few features such as `monthly_price` and `square_feet` that were more than 90% missing, so I decided to drop those features. I also wrote a custom function that accepted a single feature name as an argument and returned the following: percent of data missing, the name and count of each unique value, and the number of unique values for that feature. I went through each feature and called this function to provide insight on how to intimately handle the cleaning process of each feature. Overall there were 6 features that had missing data points that I explicitly address. For the `beds`” (number of beds in the property) feature, rather than dropping any rows, I decided to fill in the missing data points with the median value because this is a rather important feature that will surely influence the pricing model. There were 4 categorical features that had Pandas default “NaN” values, which I decided to fill with an “unknown” string. Lastly, I filled in the missing data in the `security_deposit` feature with “0” since this was most likely the correct value.

Binary Features

Upon initial examination, I noticed that there were several columns that had either “t” or “f” strings to denote the binary condition for that particular feature. One of the first steps that I took was to convert these strings into numeric representations (‘t’ : 1 , ‘f ’: 0) so they could easily be used in machine learning models. I was able to accomplish this by leveraging the Pandas `df.replace()` function with a dictionary that correctly mapped these values to each other.

Date-Time Features

I converted all of the date features that were default strings into DateTime objects so that they could be easily manipulated in data analysis. Since I did this, I was able to manipulate two datetime features to create a new feature that represents the amount of days that the listing has been posted on Airbnb.

Categorical Features

There are a lot of categorical features within this dataset. I used “binning” techniques to cut down on the number of unique items within a lot of the features for easier implementation of machine learning models. One example of this was the `property_type` feature. There were about 31 unique descriptions that defined the type of property that the listing was. I cut down these unique features into the following 4 categories: House, Apartment/Condo, Other and Hotel. I was able to accomplish this by creating a dictionary that mapped the appropriate descriptions to these 4 new ones, and then I was able to use the pandas `str.replace()` method.

Numerical Features

I used some “binning” operations with a lot of the numerical features as well. For example, I decided to bin the `days_since_last_review` column into 5 distinct groups; ['0-6 months', '6-12 months', '1-2 years', '2-3 years', '4+ years']. This was also done in an effort to use a regression model later on.

Outliers

One of the most notable challenges in this dataset were the outliers within the `price` feature. The price per night feature ranged from $0 up to $3000+, and after viewing the histogram distribution, it became evident that most of the data resided in the $0-$250 range. I decided to only use the data within this range for further analysis since the

outlying more expensive listings would probably not be a consistent representation of the patterns within the data. ONce the data was trimmed down, I decided to bin the remaining data into 5 distinct buckets using the Pandas `pd.cut()` method.