Capstone 1: Data Wrangling - AirBnB Price Prediction

Alexandra Michel

In order to begin exploring the data set, there were a few steps I took to clean things up and get to know the features I had to work with. I first obtained the data set from InsideAirBnB.com and loaded the csv file for the Los Angeles, CA listings into a Pandas dataframe which I will call the "DF" for the sake of this report. Each row in DF is an available listing on the AirBnB site in Los Angeles, CA, and it has features related to listing criteria (bedrooms, bathrooms, beds, number of guests), as well as reviews, location, amenities, min/max number of nights. DF had many free-text features which I dropped for this initial attempt, but I would include them in the future as I learn to implement NLP techniques.

```
In [4]: cols_to_drop = ['scrape_id', 'last_scraped', 'name', 'summary', 'space', 'description', 'neighborh
        ood_overview', 'notes', 'transit', 'access', 'interaction', 'house_rules', 'thumbnail_url', 'mediu
        m_url', 'picture_url', 'xl_picture_url', 'host_id', 'host_url', 'host_name', 'host_location', 'hos
        t_about', 'host_thumbnail_url', 'host_picture_url', 'host_neighbourhood', 'host_verifications', 'c
        alendar_last_scraped']
        df = df.drop(cols_to_drop, axis=1)
```

I also dropped a handful of columns which have mostly NaN values.

```
In [7]: print(df['license'].isna().sum(),
              df['square_feet'].isna().sum(),
              df['host_acceptance_rate'].isna().sum(),
              df['jurisdiction_names'].isna().sum(),
              df['weekly_price'].isna().sum(),
              df['monthly_price'].isna().sum()
             )
        29412 38568 5837 12074 35096 35335
```

There were also columns in DF that were closely related to each other (having to do with the total listings by that same host) which I dropped and just left the `host_listings_count` column to account for that information.

```
In [9]: df.drop(['host_total_listings_count', 'calculated_host_listings_count',
                 'calculated_host_listings_count_entire_homes',
                 'calculated_host_listings_count_private_rooms',
                 'calculated_host_listings_count_shared_rooms'], axis=1, inplace=True)
```

After these manipulations, I was already down from 106 columns to 60, reducing the dimension of DF by about 40%.

There were also sparse categories within `property_type` which I grouped together into `House`, `Apartment`, and `Other`. I was going to create a group called `Hotel` to include `Hotel`, `Boutique Hotel`, and `Bed and Breakfast`, but there would have only been 468 listings out

of the total ~$38,000 listings so I decided it wasn't significant enough to keep.

```
In [37]: #We can group together the categories that are all classifications of house/apartment
         df.property_type.replace({
             'Townhouse': 'House',
             'Bungalow': 'House',
             'Cottage': 'House',
             'Villa': 'House',
             'Tiny house': 'House',
             'Earth house': 'House',
             'Chalet': 'House',
             'Guesthouse' : 'House',
             'Serviced apartment': 'Apartment',
             'Loft': 'Apartment',
             'Aparthotel' : 'Apartment',
             'Condominium' : 'Apartment',
             }, inplace=True)

         df.loc[~df.property_type.isin(['House', 'Apartment']), 'property_type'] = 'Other'
```

```
In [38]: df.property_type.value_counts()

Out[38]: House        19418
         Apartment    16791
         Other         2642
         Name: property_type, dtype: int64
```

The dollar-value columns `price, security_deposit, cleaning_fee` and the column `extra_people` needed to be changed from `string` to `int` values as well, stripping the dollar signs where needed and converting the `NaN` values to zeros.

```
In [11]: ##change prices to float values
         df.price = df.price.str[1:-3]
         df.price = df.price.str.replace(",", "")
         df.price = df.price.astype('int64')
         df.price.head()

Out[11]: id
         109     122
         344     168
         2708     79
         2732    140
         2864     80
         Name: price, dtype: int64
```

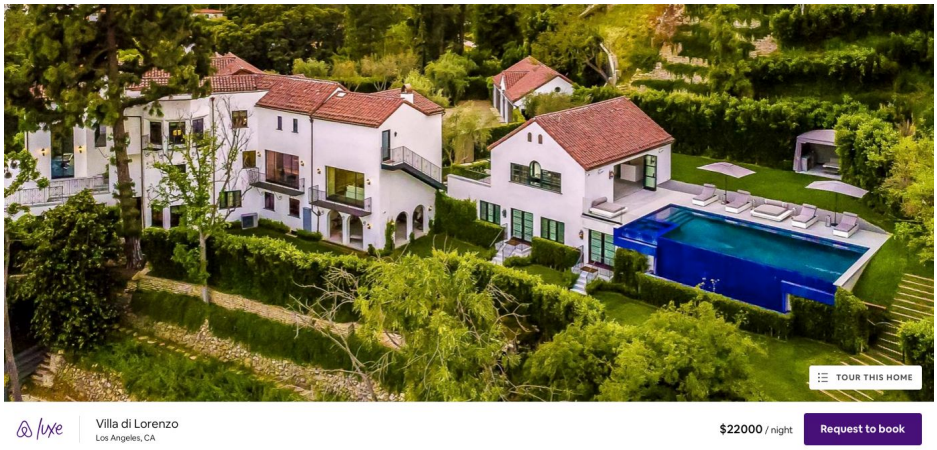
Once I had changed the `price` column to an `int`, I began exploring just the basic range of price values in DF.

```
In [12]: df.price.max()

Out[12]: 22000
```

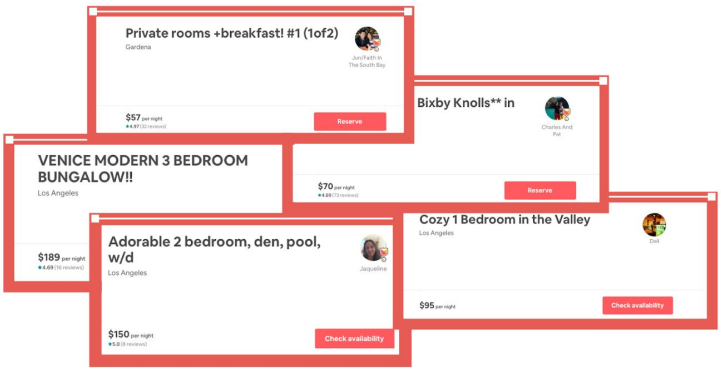
```
In [14]: df.price.min()

Out[14]: 0
```

I came across a few listings with $0 for the nightly price which I thought was odd, and the highest nightly price was $22,000. I found the specific listings at this high price by using the listing URL (one of the features of DF) and was able to validate that this is not an error.

There are two luxury retreat listings at $22,000/night! They are gorgeous, by the way, here's one:



I believe the $0/night listings were mistakes as I didn't see any similarities between these listings, and I was able to verify the actual prices again by using the listing URLs. I know in general I would really just drop those entries because we have so many entries in this data set already, but because there were only 5 of them and I honestly need the practice, I verified the prices posted as of 03-13-20 and manually inputted them into DF.




```
In [16]: df.loc[df.index == 18506601, 'price'] = 70.0
         df.loc[df.index == 18506601, 'price']

Out[16]: id
         18506601    70.0
         Name: price, dtype: float64

In [17]: df.loc[df.index == 18562002, 'price'] = 189.0
         df.loc[df.index == 21099870, 'price'] = 57.0
         df.loc[df.index == 21187671, 'price'] = 95.0
         df.loc[df.index == 29421142, 'price'] = 150.0
```

I began looking through the columns that I had kept to see how many categories each one had, and how much data was in each category. I dropped a few more columns that only seemed to have one category where most/all of the data fell into, so it wouldn't do much to distinguish that attribute in the model. For property_type, I had to weed down how many categories there were, so I narrowed it down to `House, Apartment` and `Other`.

```
In [43]: df.property_type.value_counts()

Out[43]: House         19418
         Apartment     16791
         Other          2642
         Name: property_type, dtype: int64
```

When I got down to pre-processing I realized I had more columns I needed to deal with in order to take care of the one hot encoding correctly. I ended up separating the cancellation policies into 3 main categories, and review attributes into binned ratings ranges and NaN values.

I also dropped the amenities feature because I couldn't figure out how to read in the large strings and encode the separate amenities, but this could be an area of further exploration to see if amenities help the model at all. For location I just kept the latitude/longitude coordinates and neighborhood name, and dropped the rest of the information for simplicity. I was going to keep the city/state to maybe incorporate multiple cities into the model, but simplified it for this approach.