How do neighborhood, host and listing characteristics affect the price of Airbnb listings in New York City?

Project One

Introduction

With the rapid rise of Airbnb, it is important for stakeholders to understand the factors affecting pricing of listings. Such research has numerous applications: In improving the efficiency of Airbnb and short term rental markets; for policy makers to better understand and develop regulations around the market, and for other participants such as tourist services and investors to better understand the platform. New York City, a major financial center and economic powerhouse with a bustling tourism sector, diverse population, rich culture- and an unfolding housing crisis, presents an interesting case study for our question.

Analyzing varying neighborhood characteristics, such as location, safety, landmarks, and local amenities, may uncover their role in shaping Airbnb prices. These factors not only affect the desirability and demand for listings but also reflect the socio-economic status and demographic composition of neighborhoods. Additionally, listing characteristics, including property type, size, amenities, reviews, directly influences the prices chosen by hosts. This research posits that a detailed analysis of these characteristics will provide insights into the economic principles governing the sharing economy's impact on urban housing markets.

Previous studies into this line of research have uncovered interesting results. A study examining Airbnb prices in Bristol found that the room and property type were the biggest drivers of price, and that while 'house' type properties were more expensive the closer they were to the city centre, non house type properties commanded a premium the further away they were (Voltes-Dorta & Sánchez-Medina, 2020). Another study examining Metro Nashville found that the distance to landsmarks, conventions centres and highways were positive drivers in price, but also unusually found a significant negative correlation between the number of reviews and rating with listing price (Zhang et al., 2017). Suggesting that a higher number of reviews and rating correlated with a lower price. Another study found that an Airbnb's proximity to tourist areas had a postive correlation on price (Perez-Sanchez et al., 2018). One more interesting study focusing on Barcelona also finds that Airbnb listings are concentrated in the city centre and the beach (Gutiérrez et al., 2017).

Another comprehensive study looking into price determinants found that a listing's price depended both on the listing's characteristics and reputation, and the host's involvement (measured by various platform activities) and experience. They found that the host's involvement and listing characteristics were by far the two biggest price determinants. A higher review score and higher distance to the city center also had a negative impact on price, while stricter rental policies resulted in higher prices (Toader et al. 2020). Though other studies have found the opposite and that a higher review score is associated with higher prices (Gibbs et al, 2017). Another study examining both listings features, host characteristics, and neighborhood characteristics found that after controlling for all renter visible factors such as listing features, reviews, and

area demographics, that the race of the host was a significant price differential, with Hispanic and Asian hosts charging an average 8-10% lower price relative to white hosts (Voelz et al. 2017).

We will contribute to the literature by performing a comprehensive analysis examining how neighborhood, host, and listings characteristics make up the determinants of Airbnb price. In terms of novel additions we will also scrape data for tourist attractions in New York City. This improves upon other methods of examining how location affects price, which are typically limited to calculated straight line distances to only a handful of standardized landmarks such as city Hall's and city centers (Teubner et al., 2017; Wang & Nicolau, 2017).

Our first step in analyzing the pricing is by examining Airbnb's own dataset. We'll naturally observe price to be our dependent variable, and look at it's relation with the variables 'reviews_per_month', 'room_type', 'minimum_nights', 'neighbourhood_group', and 'number_of_reviews'.

1. reviews_per_month: Reviews per month represents the average number of reviews that a listing gets per month. We could speculate that higher reviews would results in a higher listing price because of supply and demand, however we know this may not be true and uncovering the reason why may give us interesting findings.

2. room_type: This represents the type of property that the listing is and varies between Entire Home/Apartments, private rooms and shared rooms. We can speculate that this will be a strong determinant of price.

3. minimum_nights: This represents the minimum amount of nights that an airbnb customer must stay at the listing for. We can speculate this will be a determinant of price, and that rooms with higher minimum nights may command lower prices because of quantity discounts and pricing strategies.

4. neighbourhood_group: This represents the one of the five New York City borough's that the listing belongs to. We can speculate that different boroughs may command different price and a strong price determinant. We can also look at other factors in the neighborhood to uncover further charateristics which are price determinants.

5. number_of_reviews: The number of reviews are the total number of reviews that a listing has. As opposed to reviews per month, we may view this as more of a 'reputational' score.

6. calculated_host_listings_count: The calculated amount of listings a host owns. We can speculate that hosts who own more listings may act more 'economically' and have different pricing strategies

Data Loading and Cleaning

We will first load our data and import relevant libraries.

In [1]:
```
# import libraries
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import numpy as np # linear algebra
import os # accessing directory structure
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import geopandas as gpd
import stargazer
```

```
import statsmodels.api as sm
from statsmodels.iolib.summary2 import summary_col
from stargazer.stargazer import Stargazer
from IPython.core.display import HTML
from sklearn.preprocessing import StandardScaler

import warnings
warnings.filterwarnings('ignore')
np.random.seed(0)
```

In [14]:
```
# Load in data
df = pd.read_csv("../Data/AB_NYC_2019.csv")

column_names = df.columns
```

We can check for missing values with the code below. It looks good though as missing last reviews and reviews_per_month are not unusual. In fact, the fact that both columns have equal missing values is a good sign suggesting no discrepancies. Missing names are not a concern since we can use host_id should we want to use other airbnb data. I also decided not to drop any rows since we might need them in the future, and we can always slice our existing dataframe.

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
id                                  0
name                               16
host_id                             0
host_name                          21
neighbourhood_group                 0
neighbourhood                       0
latitude                            0
longitude                           0
room_type                           0
price                               0
minimum_nights                      0
number_of_reviews                   0
last_review                     10052
reviews_per_month               10052
calculated_host_listings_count      0
availability_365                    0
dtype: int64
```

Summary Statistics

We will now create our summary statistics table for our dependent and independent variables. We can observe from the table that there 48895 listings.

The average number of reviews per month is about 1.37, with a wide range of variability (standard deviation of 1.68), indicating an interesting disparity in guest visitation or listing popularity. The most common room type is "Entire home/apt," making up a significant portion of the listings (25,409 out of 48,895). This dominance suggests that travelers to NYC have a strong preference for entire homes or apartments, or that there is a large amount of units available to rent out. On average, listings require a 7-night stay, but the standard deviation is quite high (20.51). This indicates a significant variation in how hosts set minimum stay requirements, with some listings being more flexible and others aiming for longer-term stays. The mean listing has 24 total reviews with a high standard deviation, again indicating there is a disparity in activity

between listings. We can note the most popular neighborhood group to be manhattan and further examining what drives the high number of listings there may tell us more about what drives price.

Furthermore looking at calculated_host_listings_count, we can observe that it has a mean of 7, indicating that many hosts own many listings, however the large standard deviation and quartiles tell us that the majority of listings are hosters only property, with only the top 25% owning more than 2 listings.

In [219…
```
selected_summary = df[['reviews_per_month', 'minimum_nights', 'number_of_reviews', 'calculated_h

selected_summary
```

Out[219…

	count	mean	std	min	25%	50%	75%	max
reviews_per_month	38843.0	1.373221	1.680442	0.01	0.19	0.72	2.02	58.5
minimum_nights	48895.0	7.029962	20.510550	1.00	1.00	3.00	5.00	1250.0
number_of_reviews	48895.0	23.274466	44.550582	0.00	1.00	5.00	24.00	629.0
calculated_host_listings_count	48895.0	7.143982	32.952519	1.00	1.00	1.00	2.00	327.0
price	48895.0	152.720687	240.154170	0.00	69.00	106.00	175.00	10000.0

In [220…
```
selected_summary = df[['room_type','neighbourhood_group']].describe(include='all').T

selected_summary
```

Out[220…

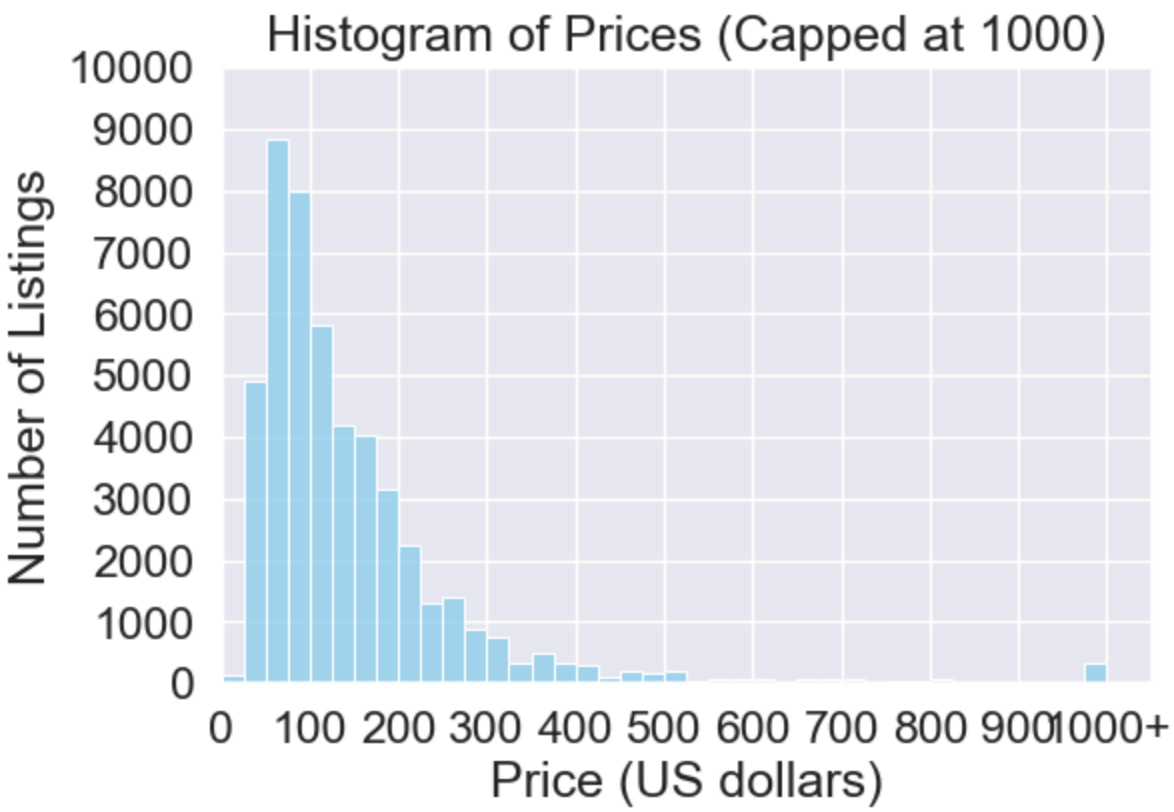
	count	unique	top	freq
room_type	48895	3	Entire home/apt	25409
neighbourhood_group	48895	5	Manhattan	21661

Plots, Histograms, Figures

We now plot some relevant graphs in order to gain a more intuitive understanding of the data and potentially glean some interesting economic results. The first thing worth plotting is a histogram of prices to get a better idea of the airbnb landscape in New York City. We can note that there are many units above our average price of $157. This suggests that renting at higher prices are economically viable, and suggests that New York Tourist's may have low elasticities of demand.

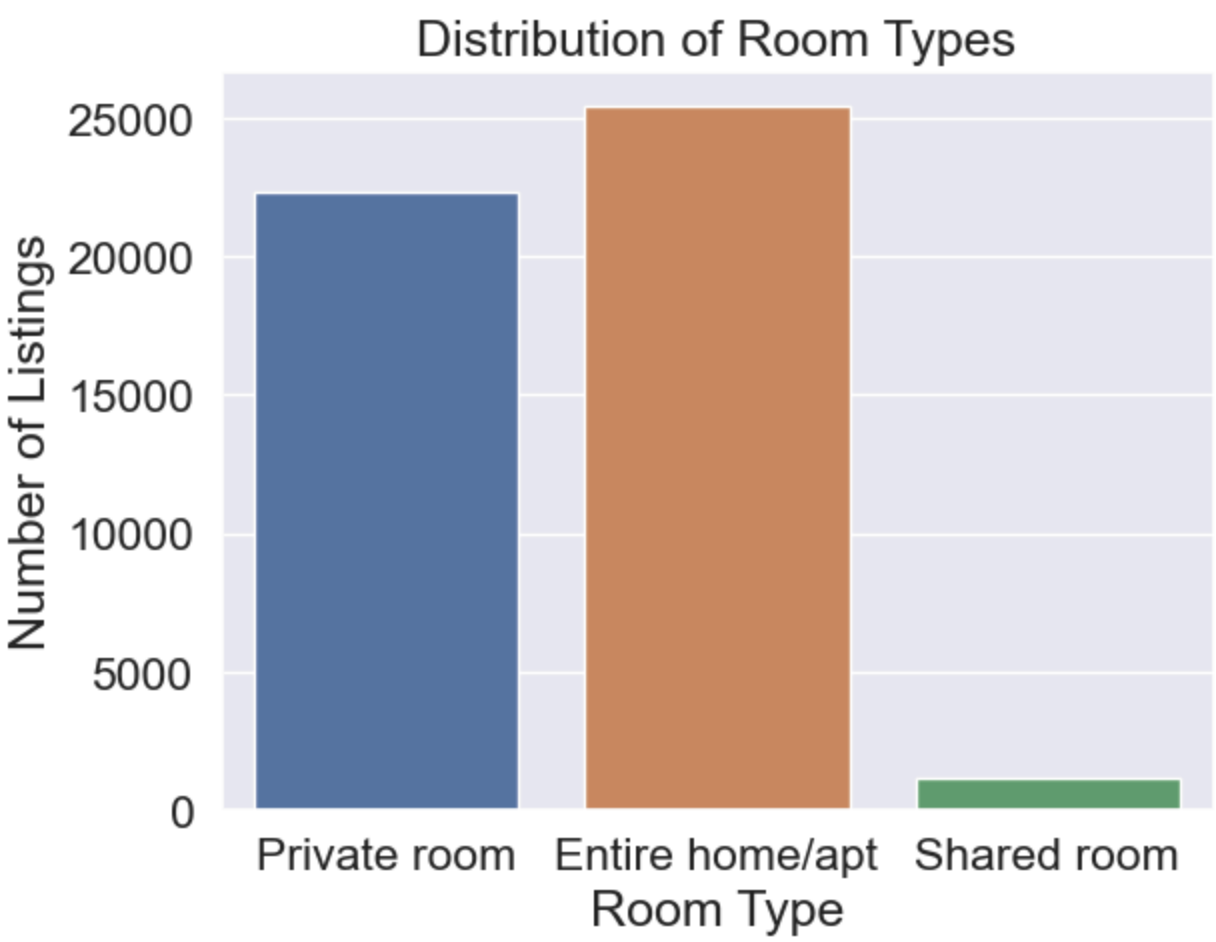
In [178…
```
# In order to produce a more appealing graph we cap the x axis to 1000, note that there are 239 l
sns.set(font_scale=1.5)

df_copy = df.copy()
df_copy['price_capped'] = df_copy['price'].apply(lambda x: min(x, 1000))

plt.figure(figsize=(6, 4))
sns.histplot(df_copy['price_capped'], bins=range(0, 1025, 25), color='skyblue')

plt.title('Histogram of Prices (Capped at 1000)')
plt.xlabel('Price (US dollars)')
plt.ylabel('Number of Listings')

xticks = range(0, 1025, 100)
xtick_labels = [str(x) for x in xticks[:-1]] + ['1000+']
```

```
plt.xticks(xticks, xtick_labels)
y_ticks = range(0, 10000 + 1, 1000)
plt.yticks(y_ticks)

plt.xlim(left=0)
plt.ylim(bottom=0)

plt.show()
```



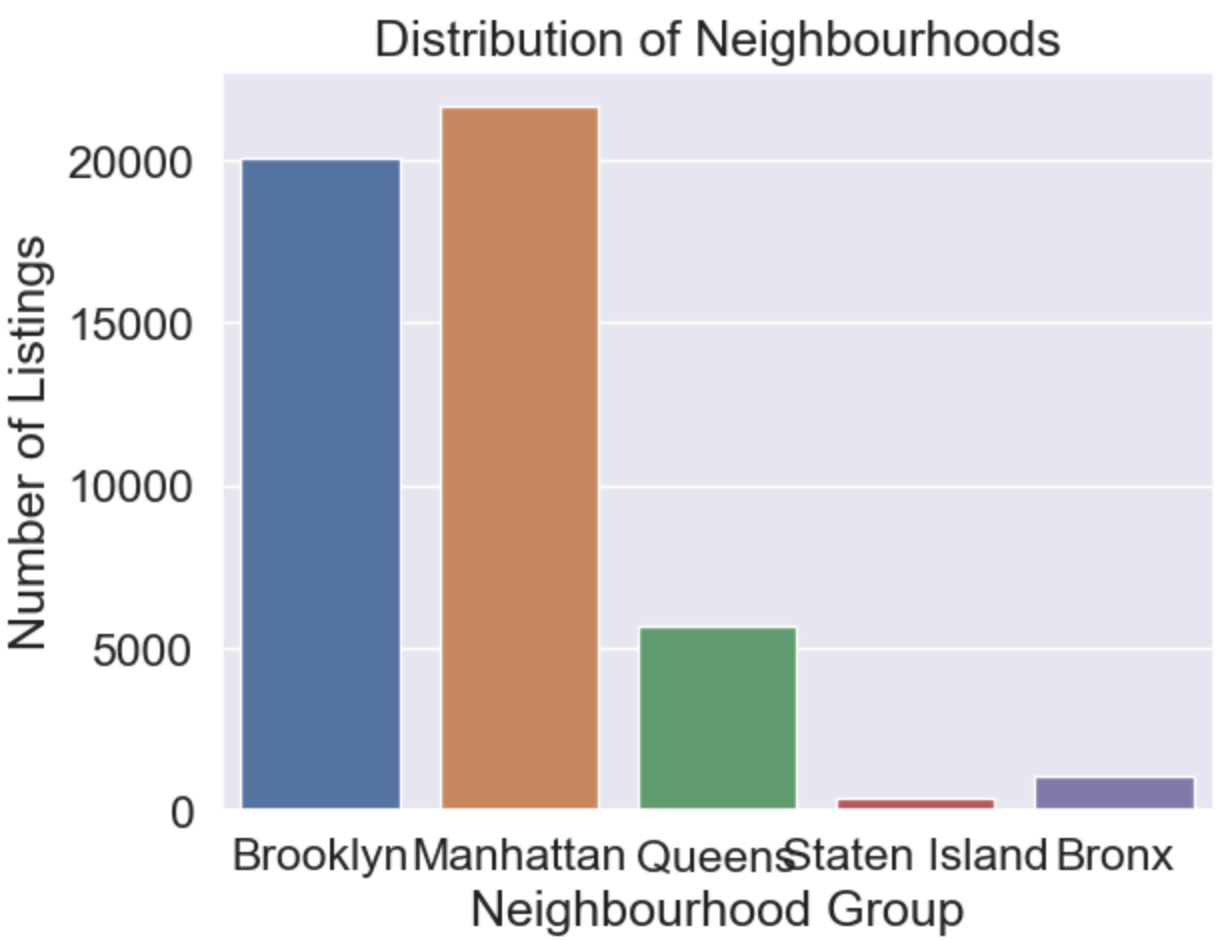
We can observe that the distribution of room types heavily favour private room's and entire home/apartments. Even though we knew entire home's/apartments were the most popular the plot reveal that private rooms are not so far behind. There are somewhat unsurprisingly not many shared rooms, as we would expect those rooms to be preferred by consumers for rentals rather than short term stays. The fact that there are so many private and entire home/apartments being rented out could suggest that they are more economically viable on airbnb than shared rooms. Furthermore, we can speculate private rooms are more likely to be owned by hosts who only own one property and may thus be motivated by a desire to take advantage of favorable economic conditions. While entire home/apartments may also be explained by frictional housing units which are temporarily unoccuped.

In [7]:
```
sns.countplot(x='room_type',data=df)
plt.title("Distribution of Room Types")
plt.xlabel('Room Type')
plt.ylabel('Number of Listings')
plt.show()
```



The plot below shows the distributions of listings across neighborhoods. A clear preference for Brooklyn, Manhattan, and to a lesser extent Queens are observed, while Staten Island and the Bronx have far fewer listings. Uncovering the reasoning between the disparity will help answer our research question.

In [8]:
```
sns.countplot(x='neighbourhood_group',data=df)
plt.title("Distribution of Neighbourhoods")
plt.xlabel('Neighbourhood Group')
plt.ylabel('Number of Listings')
plt.show()
```

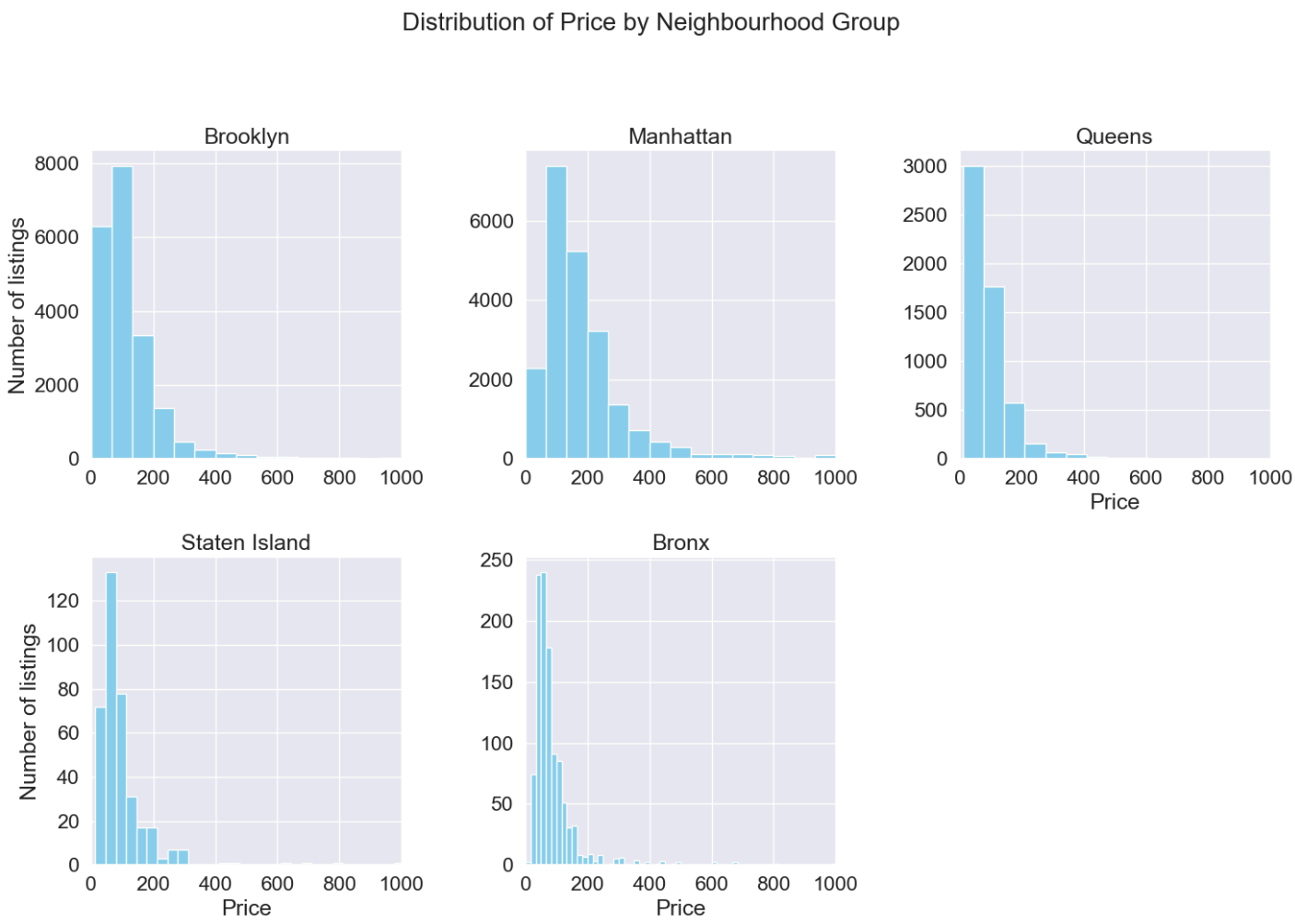


Plotting the distribution of price by each neighbourhood group we can see that Manhattan very convingly has a higher share of units which command higher prices. Interestingly even though Brooklyn and Manhattan have similar number of listings, we can observe from the peak and skew of the graphs that Manhattan has higher prices. This further suggests we should look at distinguishing features that Manhattan alone contains, such as landmarks and tourist attractions.

This also suggests that borough is a determinant of price.

In [64]:
```
current_context = sns.plotting_context()

# Set larger font scale for better readability
sns.set(font_scale=1.5)

# Create the FacetGrid with 3 columns, so it will create 2 rows
g = sns.FacetGrid(df, col='neighbourhood_group', sharex=False, sharey=False, col_wrap=3, height=

# Map the plot type and specify the color and bins
g.map(plt.hist, 'price', bins=150, color='skyblue')

# Set the axis labels
g.set_axis_labels('Price', 'Number of listings')

# Set the titles for the plots
g.set_titles('{col_name}')

# Adjust title positions and increase title font size
for ax in g.axes.flat:
    ax.title.set_position([0.5, 1.05])
if g.fig._suptitle is not None:
    g.fig._suptitle.set_visible(False)
```

```
# Set the main title for the figure
g.fig.suptitle('Distribution of Price by Neighbourhood Group', fontsize=20, y=1.05)

# Set the x limit for better visibility
g.set(xlim=(0, 1000))

# Adjust layout for better fit and to prevent label cutoff
g.tight_layout(w_pad=2, h_pad=2)

# Show the plot
plt.show()


sns.set_context(current_context)
```

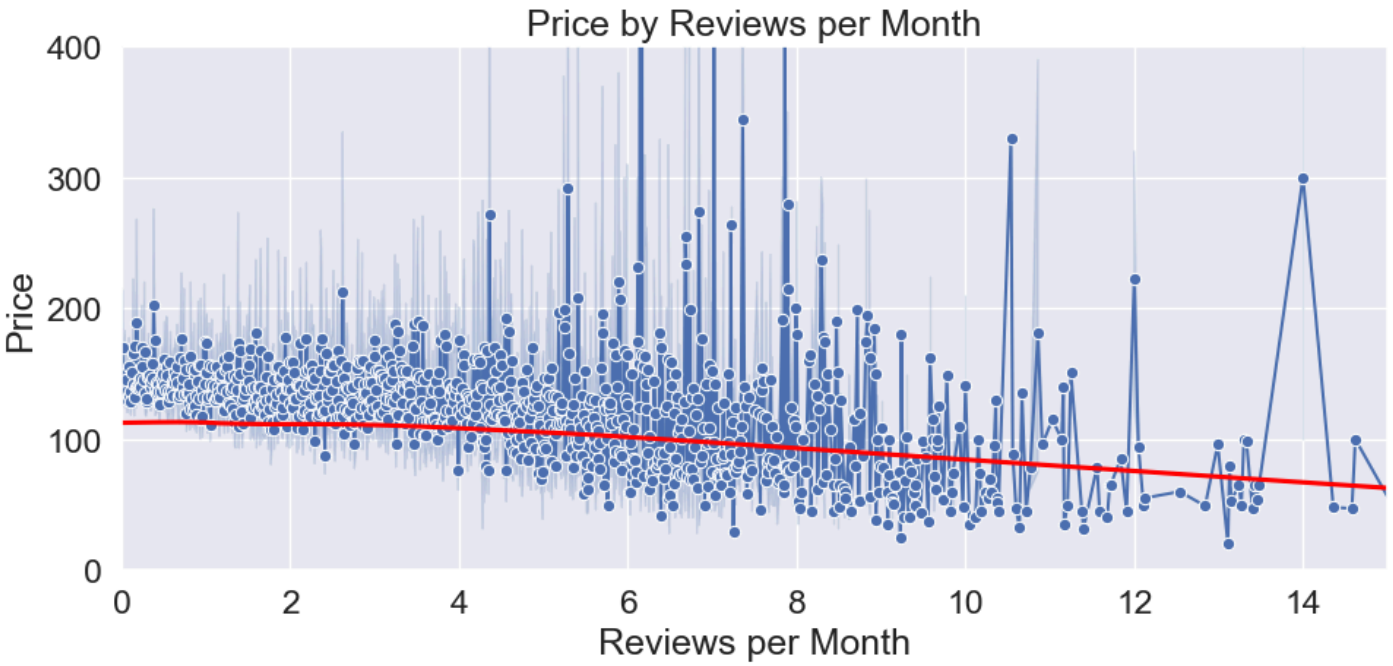


We can speculate that units with higher reviews per month and total reviews may have a higher price because of higher demand or reputation. However, very interestingly the reviews per month graphs below appear to show that units with lower prices have more reviews. This supports the findings from Zhang et al.'s paper. This is also very important because it suggests that reputation of a listing is not as an important of a price determinant as other factors, and may be a negative price determinant. On the other hand, total reviews has almost no correlation with price at any level, suggesting that customers are indifferent to the number of total reviews.

In [182…

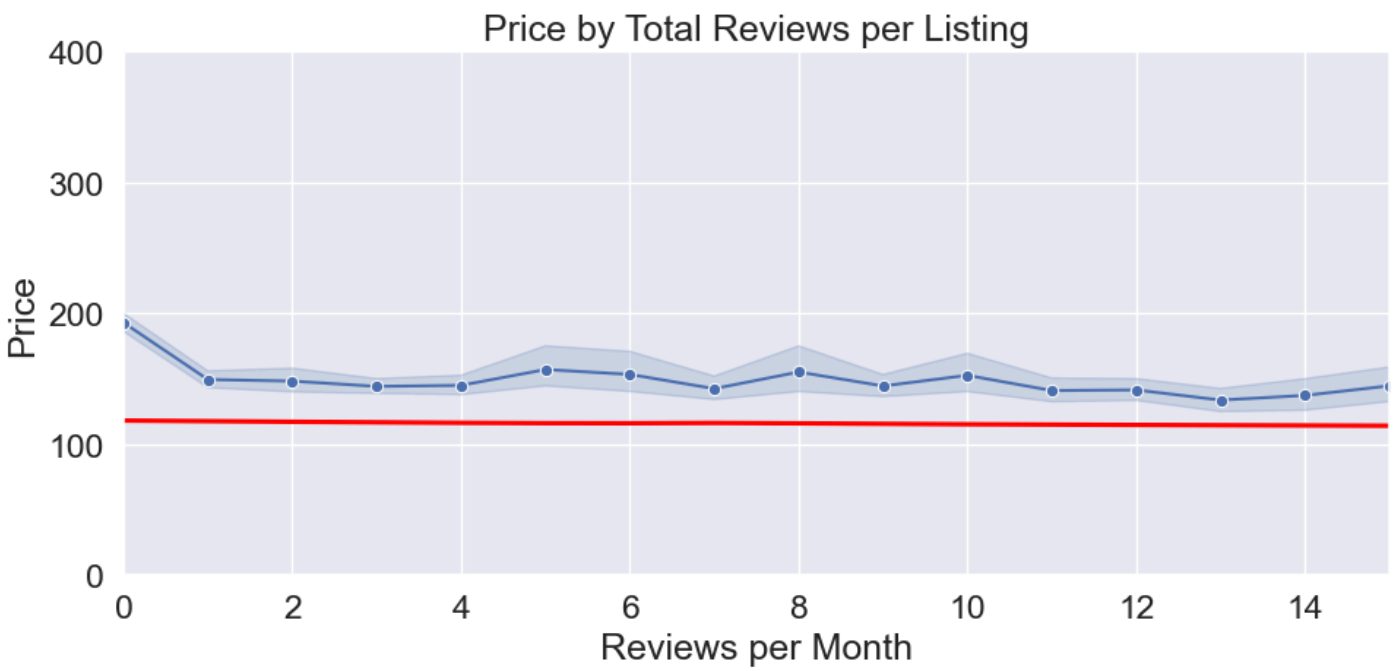
```
plt.figure(figsize=(10, 5))
sns.lineplot(data=df, x='reviews_per_month', y='price', marker='o')
sns.regplot(data=df, x='reviews_per_month', y='price', scatter=False, color='red', lowess=True)

# Set the title and labels
plt.title('Price by Reviews per Month')
```

```
plt.xlabel('Reviews per Month')
plt.ylabel('Price')

# Set the x and y axis limits to focus on the specified region
plt.xlim(0, 15)
plt.ylim(0, 400)

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```



In [183…

```
plt.figure(figsize=(10, 5))
sns.lineplot(data=df, x='number_of_reviews', y='price', marker='o')
sns.regplot(data=df, x='number_of_reviews', y='price', scatter=False, color='red', lowess=True)

# Set the title and labels
plt.title('Price by Total Reviews per Listing')
plt.xlabel('Reviews per Month')
plt.ylabel('Price')

# Set the x and y axis limits to focus on the specified region
plt.xlim(0, 15)
plt.ylim(0, 400)

# Adjust layout and display the plot
plt.tight_layout()
plt.show()
```



Observing our grouped plot of calculated host listings count against price, we can observe a seemingly non linear relationship. Hosts with a few amount of listings will charge lower and lower prices until hosts with about 10+ listings diverge and charge far higher prices with higher variance. The likely explanation for this is that hosts who own more listings also maintain nicer properties, have better reputation, better services, and/or better location and are thus able to get away with charging a premium. Regardless less of the specific explanation what we can take away from the plot is that host characteristic do have some significant impact on price.

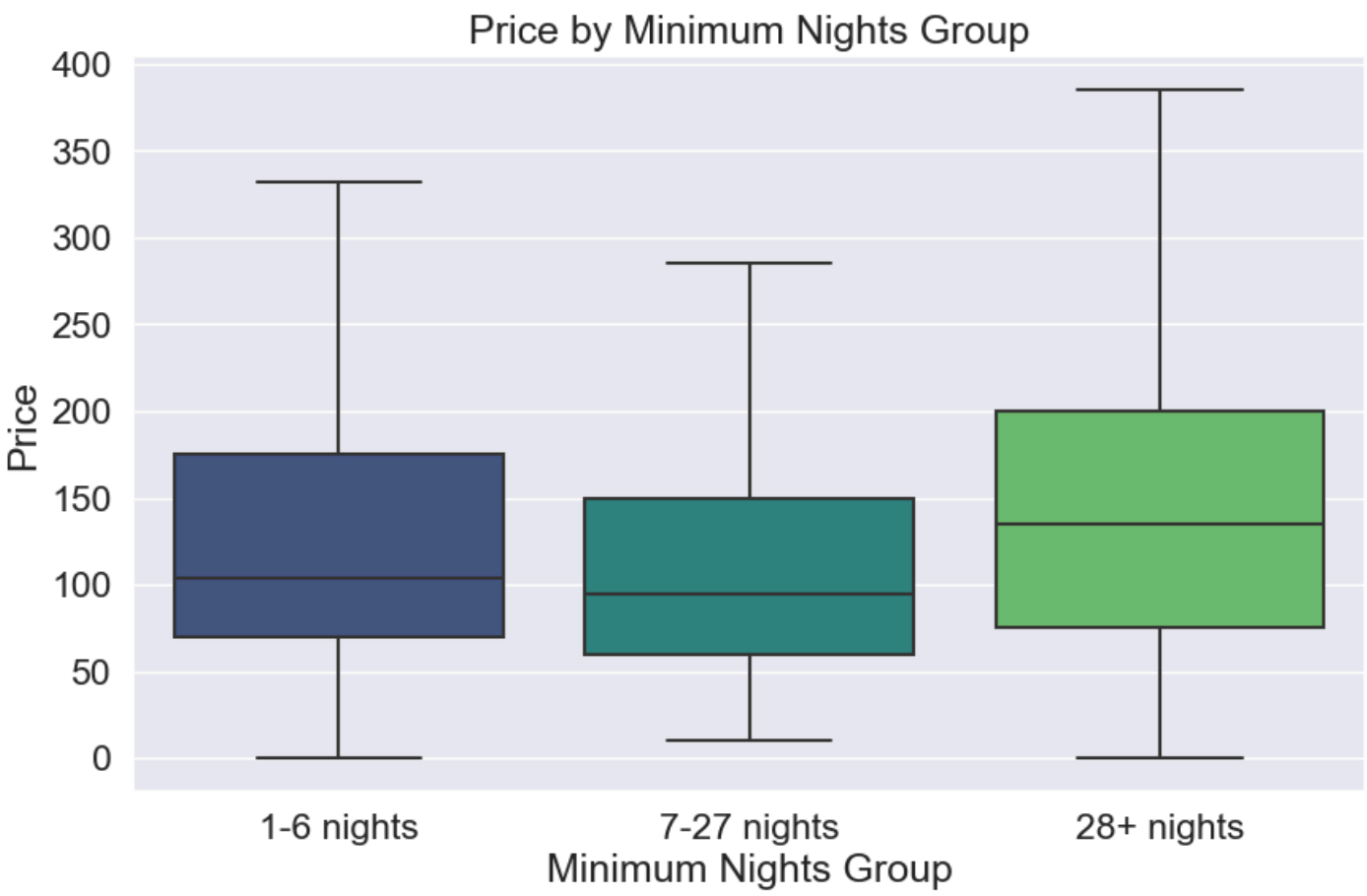
In [186…

```
def listings_group(listings_count):
    if listings_count == 1:
        return '1 listing'
    elif listings_count == 2:
        return '2 listings'
    elif 3 <= listings_count <= 10:
        return '3-10 listings'
    else:
        return '10+ listings'

df['Listings Group'] = df['calculated_host_listings_count'].apply(listings_group)
plt.figure(figsize=(10, 6))
sns.boxplot(x='Listings Group', y='price', data=df, palette='viridis', order=['1 listing', '2 li
plt.title('Price by Number of Listings Group')
plt.xlabel('Number of Listings Group')
plt.ylabel('Price')
plt.show()
```

The plot below and suggests that minimum nights likely has some impact on price. The results can be explained economically quite well. A higher price for short stays (1-6 nights) can be explained by the fact that listings with short minimum stays may attract guests looking for brief stays, possibly willing to pay a premium for flexibility. Hosts might set higher prices to maximize revenue per night to offset the higher turnover costs, such as cleaning and administration, that come with short-term rentals. Mid length Sstays (7-27 nights) might be priced slightly lower as they attract a different segment of the market—guests seeking accommodations for longer durations like extended vacations or business trips. The slightly lower prices could be a strategy to make these properties more attractive and ensure occupancy, given that guests staying longer are less likely to pay a premium for each night. Additionally, the reduced turnover for such properties can lead to lower operational costs per booking, allowing hosts to offer more competitive rates. Lastly long stays (28+ nights) could be priced higher due to their appeal to guests looking for long-term accommodations, but who are not able to secure tenancy in New York.

In [188…

```
def nights_group(nights):
    if 1 <= nights <= 6:
        return '1-6 nights'
    elif 7 <= nights <= 27:
        return '7-27 nights'
    else:
        return '28+ nights'
df['Nights Group'] = df['minimum_nights'].apply(nights_group)
df['Nights Group'] = df['minimum_nights'].apply(nights_group)
plt.figure(figsize=(10, 6))
sns.boxplot(x='Nights Group', y='price', data=df, palette='viridis', order=['1-6 nights', '7-27 
plt.title('Price by Minimum Nights Group')
plt.xlabel('Minimum Nights Group')
plt.ylabel('Price')
plt.show()
```



Project Two

The Message

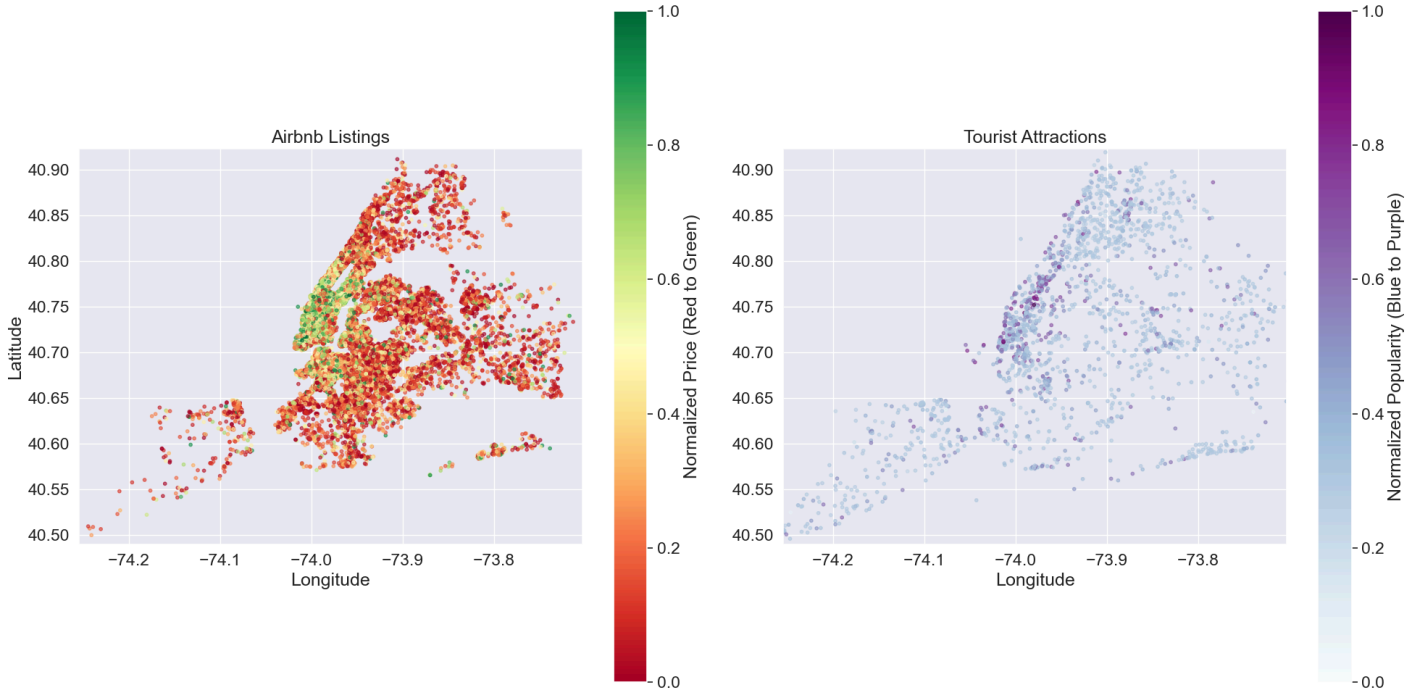
Tourism attractions and their popularity is the biggest driver of the price of Airbnb listing in New York City.

The reasoning behind this message will become clear in later sections, but finding that tourism proximity has the biggest effect on price throughout our later sections is the biggest takeaway from our project and represents a novel contribution to the literature. Observing our map shows the very high correlation between the presence of popular tourist attractions and the price of listings.

In [193…

```
xdf = pd.read_csv('final_df.csv')
anp = pd.read_csv('attractions_wnp.csv')
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))
xdf['price_normalized'] = (xdf['price'] - xdf['price'].min()) / (xdf['price'].max() - xdf['price

# Defining the axis limits to the geographic extent of the points
lon_min, lon_max = xdf['longitude'].min() - 0.01, xdf['longitude'].max() + 0.01
lat_min, lat_max = xdf['latitude'].min() - 0.01, xdf['latitude'].max() + 0.01

# Airbnb listings map with equal aspect ratio and adjusted plot limits
ax1.scatter(xdf['longitude'], xdf['latitude'],
            c=xdf['price_normalized'], cmap='RdYlGn', s=10, alpha=0.6)
ax1.set_title('Airbnb Listings')
ax1.set_xlabel('Longitude')
ax1.set_ylabel('Latitude')
ax1.set_xlim(lon_min, lon_max)
ax1.set_ylim(lat_min, lat_max)
```

```
ax1.set_aspect('equal', adjustable='box')

# Adding color bar for Airbnb listings
sm_airbnb = plt.cm.ScalarMappable(cmap='RdYlGn', norm=plt.Normalize(vmin=0, vmax=1))
sm_airbnb.set_array([])
fig.colorbar(sm_airbnb, ax=ax1, orientation='vertical', label='Normalized Price (Red to Green)')

# Tourist attractions map with equal aspect ratio and adjusted plot limits
ax2.scatter(anp['Longitude'], anp['Latitude'],
            c=anp['Normalized Popularity'], cmap='BuPu', s=10, alpha=0.6)
ax2.set_title('Tourist Attractions')
ax2.set_xlabel('Longitude')
ax2.set_xlim(lon_min, lon_max)
ax2.set_ylim(lat_min, lat_max)
ax2.set_aspect('equal', adjustable='box')

# Adding color bar for tourist attractions
sm_attractions = plt.cm.ScalarMappable(cmap='BuPu', norm=plt.Normalize(vmin=0, vmax=1))
sm_attractions.set_array([])
fig.colorbar(sm_attractions, ax=ax2, orientation='vertical', label='Normalized Popularity (Blue

# Adjust layout
plt.tight_layout()
plt.show()
```



Observing the graph below we can note there are fundamentally two to three different groups. The majority of listings are short term and below 28 days (the 90th quantile) and there several other groups of listings with very high minimum nights, indicating longer term rentals.

Merging with a New Dataset

In order to glean more interesting results I decided to merge in the custom Department of City Planning 2020 US census data for New York City (NYC DCP, 2020). Some columns are custom made by the DCP. The P suffix after an identifier denotes the percentage change from the 2010 census.

The first 6 rows are aggregate data for each of the five boroughs + NYC aggregate, and the rest are a mix of the official 59 community districts/community boards, neighborhood names, parks, and other geographical

features (Airports, ports, etc.). It is worth noting that the 'neighbourhoods' that airbnb uses are informal and somewhat loosely defined districts (particulary around Staten Island) and not official divisions (Buchanan et al., 2023). Not every neighbourhood in the airbnb dataset is defined the same way in the census data. If we wanted to created a finer/granular map we would have to map every airbnb neighbourhood to the census districts, however in the interest of accuracy and time we'll just use the boroughs as our divisions and drop the remaining districts.

Looking at a snapshot everything looks good. We wont eliminate any columns in case we want to use them in the future. For now we'll use this data to create some maps.

In [117…

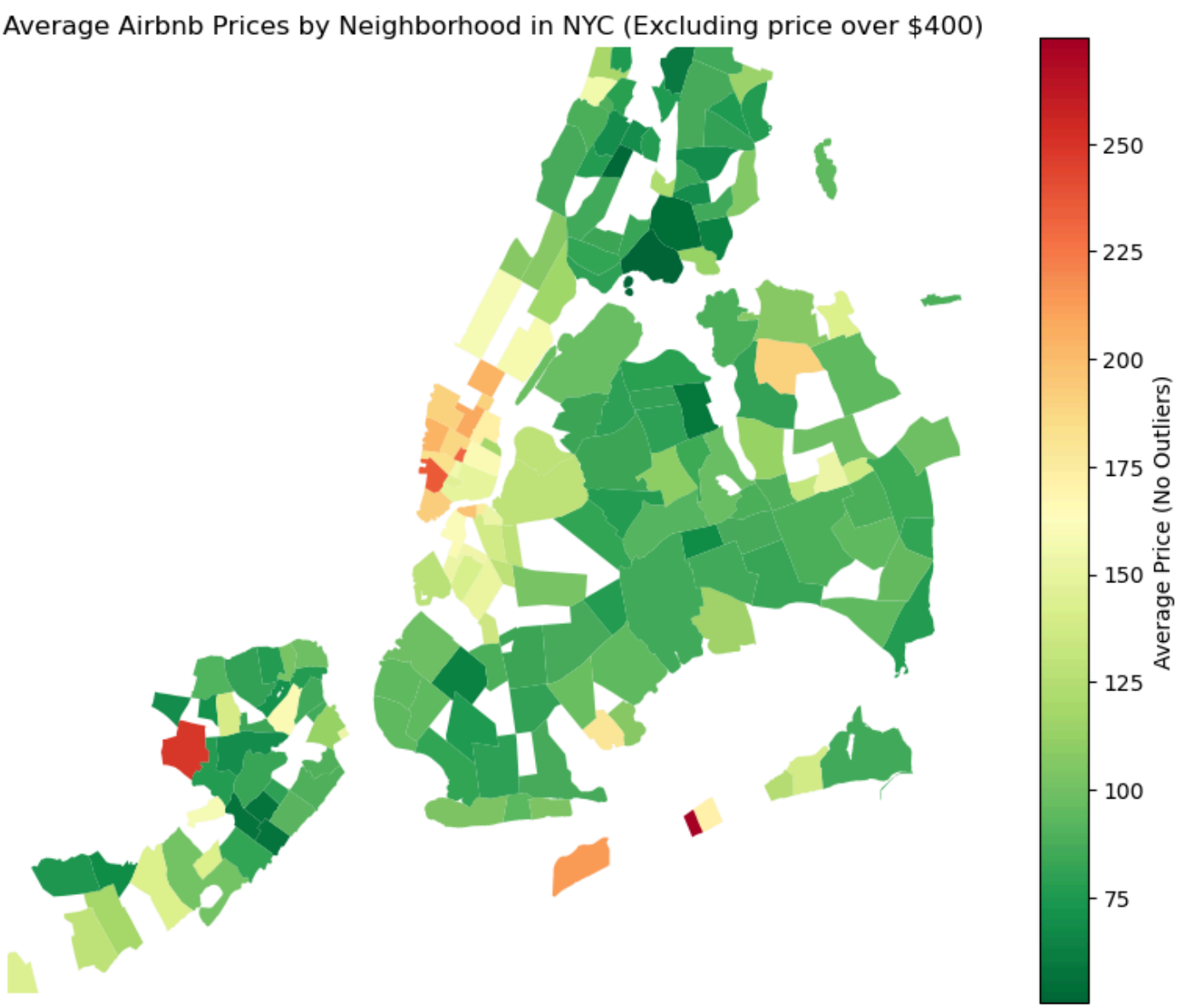
```
# Load the Excel file
file_path = '../Data/nyc_core_censusdata.xlsx'
cn_df = pd.read_excel(file_path)
```

Maps and Interpretation

I put explanations/interpretations below each map.

In [7]:

```
# Load the datasets and copy Airbnb dataset
gdf = gpd.read_file('../Data/map/ZillowNeighborhoods-NY.shp')
airbnb_df = df

# Filter out listings with prices above $400
airbnb_filtered = airbnb_df[airbnb_df['price'] <= 400]

# Calculate the average price per neighborhood
average_prices_filtered = airbnb_filtered.groupby('neighbourhood')['price'].mean().reset_index()

# Merge this data with the geographic data
gdf_merged_filtered = gdf.merge(average_prices_filtered, how='left', left_on='Name', right_on='n
gdf_merged_filtered.dropna(subset=['price'], inplace=True)

# Plot the map with the average Airbnb prices per neighborhood
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
gdf_merged_filtered.plot(column='price', ax=ax, legend=True,
                         legend_kwds={'label': "Average Price (No Outliers)"},
                         cmap='RdYlGn_r', missing_kwds={"color": "lightgrey"})

ax.set_xlim(-74.25, -73.7)
ax.set_ylim(40.5, 40.9)

plt.title('Average Airbnb Prices by Neighborhood in NYC (Excluding price over $400)')
ax.set_axis_off()
plt.show()
```



Our map of average airbnb prices across shows the same relations that we found in our previous plots. We can indeed observe that Manhattan and part of Brooklyn are far higher priced than the rest of New York City. We can note some outliers such as the neighborhood in red on Staten Island which is odd as that area is quite barren compared to the rest of New York. The smaller islands of New York are also price outliers which is not surprising as we might imagine their is little supply, however it may be interesting to investigate that further. Examining the outlier's in lower Brooklyn and upper Queens may also uncover something.

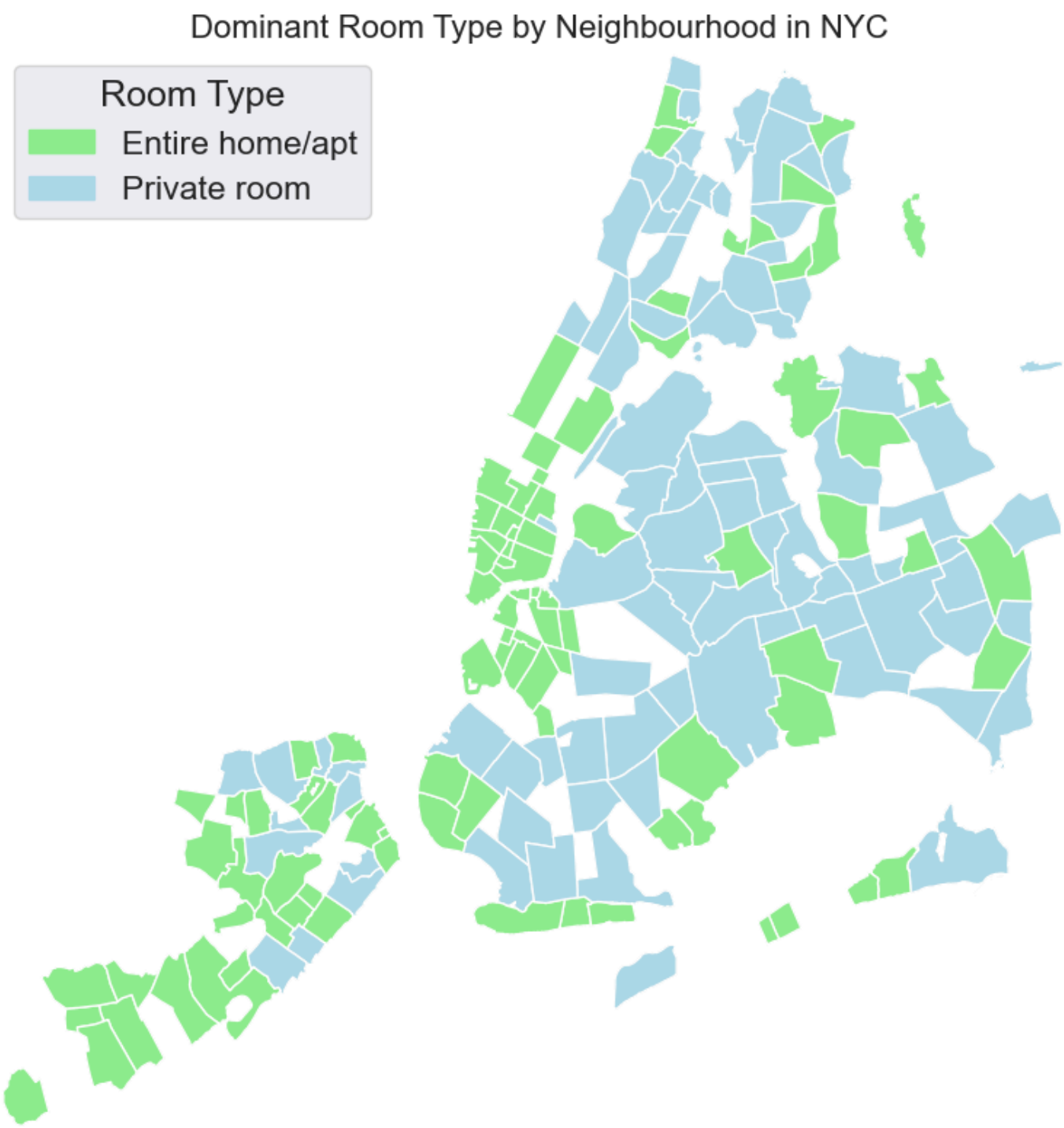
In general though just seeing how prices change across neighborhood's, even when not as extreme as the outliers is interesting and suggests that location base factors influence price.

Another interesting observation is that even though previous literature had indicated that location to water or parks increased airbnb prices, we can not see this same result exhibited in New York City, as prices are more distribution across location.

In [109…

```
import matplotlib.patches as mpatches
from matplotlib.colors import ListedColormap

airbnb_listings = pd.read_csv('../Data/AB_NYC_2019.csv')
nyc_neighbourhoods = gdf

custom_colors = ['lightgreen', 'lightblue']
custom_cmap = ListedColormap(custom_colors)
```

```
color_dict = {
    'Entire home/apt': 'lightgreen',
    'Private room': 'lightblue'
}

# Aggregate Airbnb data to find dominant room type per neighbourhood
room_type_counts = airbnb_listings.groupby(['neighbourhood', 'room_type']).size().reset_index(na
dominant_room_type = room_type_counts.loc[room_type_counts.groupby(['neighbourhood'])['count'].i

# Visualization with Matplotlib
fig, ax = plt.subplots(figsize=(16, 10))
joined_data.plot(ax=ax, column='room_type', cmap=custom_cmap, legend=True)

# Add a title to the plot
ax.set_title('Dominant Room Type by Neighbourhood in NYC', fontdict={'fontsize': '16', 'fontweig

legend_entries = [mpatches.Patch(color=color, label=room_type) for room_type, color in color_dic

# Add the custom legend to the plot
ax.legend(handles=legend_entries, title='Room Type', loc='upper left')

ax.set_xlim(-74.259090, -73.700272)
ax.set_ylim(40.477399, 40.917577)

ax.axis('off')

plt.show()
```

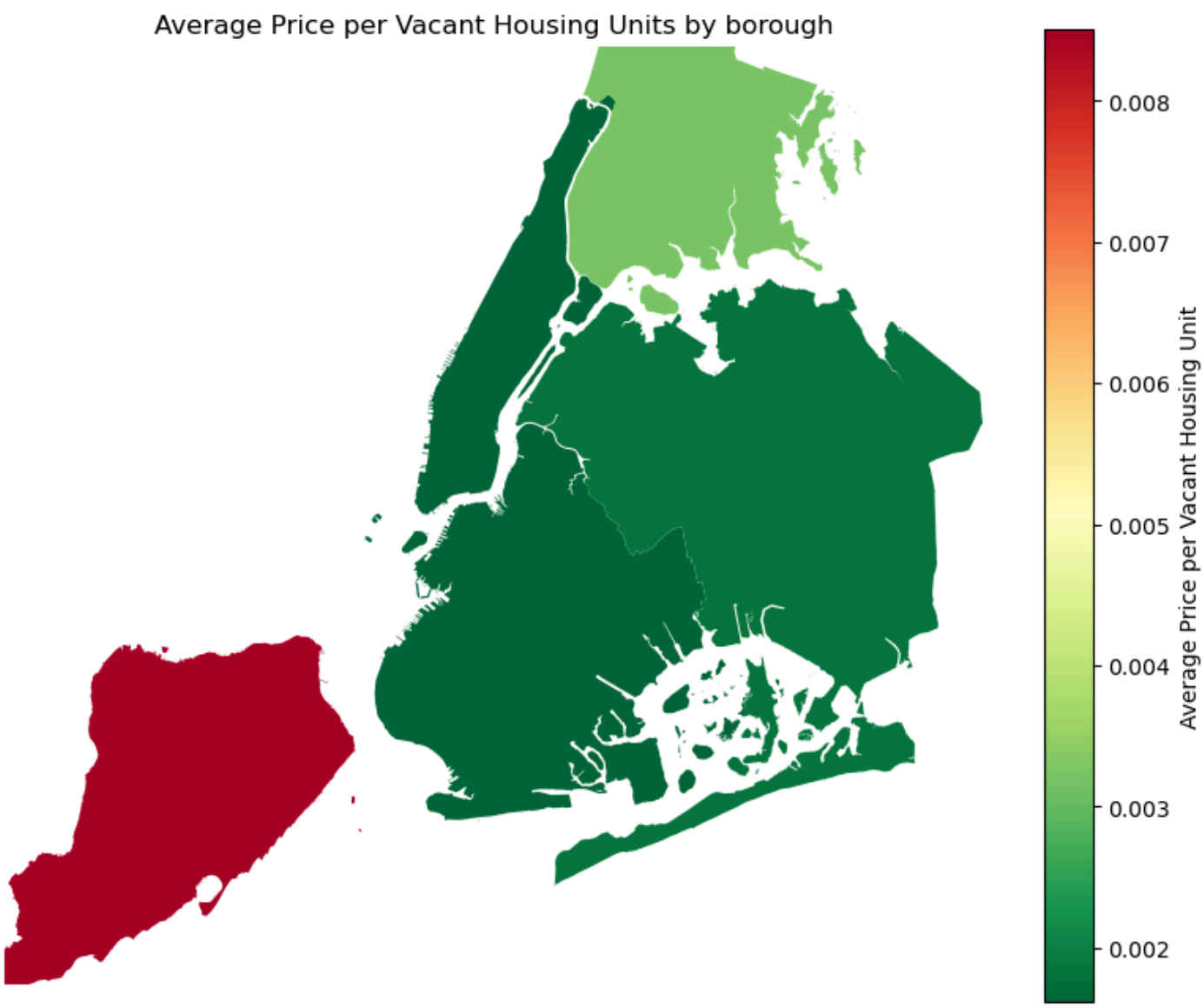


Here I plotted the most dominant room type in each neighborhood. This map supports our previous findings from project one as well. We can see that 'entire home/apt' and 'private rooms' dominate Airbnb listings. Even though the specifics of the number of room types are not plotted, the fact that the most dominant listing type on all of Staten Island and Manhattan are entire home/apt's, while most listings in the Bronx, Brooklyn, and Queens are private rooms is interesting.

If we are to further compare our two maps of room types and price what we can see is that higher priced areas have more entire home/apt listings, save for Staten Island. The reasoning for this becomes apparent below though, as Staten Island has a relatively high vacant unit count. What this tells us is that it more economically viable to list entire home/apt units in certain areas, such as south Manhattan, upper Brooklyn, and east Queens (Staten Island is the exception as it already contains a vacancy rate). The reasons behind this will become clear.

In [8]:
```
# Filter out listings with prices above $400
airbnb_filtered = airbnb_df[airbnb_df['price'] <= 400]

# Calculate the average price per neighborhood
average_prices_filtered = airbnb_filtered.groupby('neighbourhood_group')['price'].mean().reset_i
```

In [17]:
```
# Load the datasets and copy Airbnb dataset
gdf = gpd.read_file('../Data/five_boroughs/fb.shp')
cn_df2 = cn_df #census dataframe
airbnb_df = df #airbnb dataframe)

average_prices_filtered.head(5)

cn_df2 = cn_df2.merge(average_prices_filtered, how='left', left_on='Name', right_on='neighbourho
cn_df2["Price per VacHUs"] = cn_df2["price"]/cn_df2["VacHUs"]


# Merge this data with the geographic data
gdf_merged = gdf.merge(cn_df2, how='left', left_on='boro_name', right_on='Name')

# Plot the map with the 'VacHUs' column.
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
gdf_merged.plot(column='Price per VacHUs', ax=ax, legend=True,
                legend_kwds={'label': "Average Price per Vacant Housing Unit"},
                cmap='RdYlGn_r', missing_kwds={"color": "lightgrey"})

ax.set_xlim(-74.25, -73.7)
ax.set_ylim(40.5, 40.9)

plt.title('Average Price per Vacant Housing Units by borough')

ax.set_axis_off()
plt.show()
```



One variable that stuck out to me that might be interesting from the DCP data was 'vacant housing units' for 2020 (VacHUs). Vacant housing units are a neighborhood charateristics that we might speculate would influence price based off our knowledge of supply and demand, i.e a higher amount of vacan housing units may result in lower priced listings. However dividing vacant housing units against average price for each borough we see the opposite. Boroughs with a lower average price per vacant housing unit (the dark green) also happen to be the more in demand areas which command higher prices and boroughs with a lower value are the less in demand areas. This is contrary to what we might expect and suggests that available supply is a near total non factor on airbnb prices, or that airbnb users have a very low elasticity of demand in New York City.

Vacant housing units are of course an important neighborhood characteristic, which relates to our research question in seeing how neighborhood and listing characteristics affect price.

Project Three

Potential Data To Scrape

Throughout the project and various data explorations three key facts have been revealed so far: that the number of reviews and reviews per month (potentially indicative of a positive review score and reputation)

have a negative to neutral affect on price; that both property type and listing neighborhood have a strong affect on price; and that the price of listings is unaffected by the underlying housing supply in the neighborhood.

This last fact in particular suggests that airbnb users may have a low elasticity of demand, which aligns with our expectation that most airbnb user's are perhaps richer tourists visiting the city. Furthermore while the 'vacant housing units' variable was useful from our census dataset, the literature suggests that neighborhood specific (though not host) socioeconomic characteristics have little to no impact on price. This and the other facts leads us to suspect that the proximity of a listing to tourist destinations may be a major price determinant. In the interest of pursuing this hypothesis scraping tourist destinations as well as their popularity/activity level or at least reviews and review score would be ideal. Merging in this sort of data would allow us to have a more comprehensive answer to our research question.

One place to gather this information from would be google maps. From my research they have the most comprehensive database of both tourist locations, reviews, as well as other data. In order to get this data we can use the google Places API, and more specifically the googlemaps python library api wrapper. In order to make use of this queried data we can aggregate the total destinations divided by total review score by neighborhood- this acts as a de facto measure of tourism popularity in a neighborhood. This would allow us to clearly see how tourist destinations in a neighborhood affects the listing price's in the neighborhood.

Potential Challenges

One potential challenge would be the pricing of the places api, which operates with a 'pay as you go' pricing model. While all users are provided $200 (USD) in free credits, staying under this limit may present a challenge. Should we hit this limit additional webscraping may become a challenge as google is not friendly towards webscrapers. We may be forced to become more strategic in our scraping, including actually specifying our user-agent and using realistic headers, staggering our requests to mimic human behaviour, and mimic 'natural' web navigation. Or use other webscraping solutions.

Another challenge is just in learning how to use the places API, and perhaps other google API's we may use, as these will take time to understand how to use. In terms of time constraints, I expect it to take just a few minutes to tens of minutes to gather all the data, as we are not constrained by explicit rate limits (beyond 150,000 request a day) but by money, also because of the paginization of results, and max results per request being limited to 20, we need to wait a few seconds between requests in order the next page token to work. An additional challenge I found is that google will always limit the results of any query to 60 results to prevent scraping, disregarding the 20 result page limit, to get around this we must make more granular searches.

With all that being said I believe it is doable and will proceed to get the data.

Scraping Data from a Website

After setting up our project and api's via the google cloud console we can authorize ourselves with the API key through the googlemaps python google API's wrapper.

In [16]:
```
import googlemaps
import time
```

```
gmaps = googlemaps.Client(key='AIzaSyAYYplhoBJNoaZyZee3O4yW6xWcG8dIq3g')
```

We will now collect data by iterating over every borough, going through every 'page', putting the resulting relevant json tags into a dataframe, dropping possible duplicates, and classifying each tourist attraction into its appropriate neighborhood by longitude and latitude.

In order to get around the hard query limit of 60, we will perform a search for each individual neighborhood. This will result in duplicates as google will never return empty results (even if there are no tourist attractions in a neighborhood), but we can drop them afterwards. This will also significantly increase our scraping time. If we wanted to we could also iterate over additional tourist attraction related keywords, and make our search locations even smaller to get an even more comprehensive dataset, however this would take exponentially longer and may cost a significant amount.

In [ ]:
```
MAX_REQUEST_TRIES = 2500
current_tries = 0

file_path = '../Data/neighbourhoods.csv'
neighborhoods_df = pd.read_csv(file_path)

keyword = "tourist attractions"
attractions = []

for index, row in neighborhoods_df.iterrows():
    borough = row['neighbourhood_group']
    neighborhood = row['neighbourhood']

    if borough == "Bronx":
        borough = "The Bronx"

    query = f"{keyword} in {neighborhood}, {borough}, New York, NY, USA"
    print(f"Fetching attractions for: {query}")

    response = gmaps.places(query=query)
    current_tries += 1

    while current_tries < MAX_REQUEST_TRIES:
        if 'results' in response:
            for result in response['results']:
                attraction = {
                    'Borough': borough,
                    'Neighborhood': neighborhood,
                    'Name': result['name'],
                    'Address': result.get('formatted_address', ''),
                    'Latitude': result['geometry']['location']['lat'],
                    'Longitude': result['geometry']['location']['lng'],
                    'Rating': result.get('rating', None),
                    'Total Ratings': result.get('user_ratings_total', None),
                    'Place ID': result.get('place_id', None),
                    'Types': ', '.join(result.get('types', []))
                }
                attractions.append(attraction)

        # Check for a next_page_token and make a subsequent request
        next_page_token = response.get('next_page_token')
        if next_page_token:
            time.sleep(2)  # Delay needed before the next page token becomes valid
            response = gmaps.places(query=query, page_token=next_page_token)
            current_tries += 1
```

```
        else:
            break
```

The code took 16 minutes and 43 seconds to execute which is pretty good. We now have a dataset with 9914 rows, which is better than my intial attempt scraping by borough with 251 results. However we now encounter another potential problem in that the length of the dataframe is the same as the length of the dataframe with duplicates dropped, whereas we were expecting numerous duplicates.

In [ ]:
```
attractions_df = pd.DataFrame(attractions)
attractions_dfn = attractions_df.drop_duplicates()
print(len(attractions_df), len(attractions_dfn))
```

After some omitted investigation I found that some tourist destinations appeared in multiple searches, even in non relevant neighborhoods, resulting in numerous rows with 'unique' neighborhoods. We can fix the problem by dropping both borough and neighborhood, as we can see it is inaccurate, and were intending to classify the locations otherwise from the start, then drop duplicates. We are left with 2054 unique tourist attractions.

In [ ]:
```
attractions_df_cleaned = attractions_df.drop(columns=['Borough', 'Neighborhood'])

# Now drop duplicates based on the 'Name' column in the cleaned DataFrame
attractions_df_cleaned = attractions_df_cleaned.drop_duplicates(subset='Name')

attractions_df_cleaned = attractions_df_cleaned.reset_index(drop=True)
len(attractions_df_cleaned)
```

Since the results are not accurate to the location bounds, we will now classify each attraction to a neighborhood based off the longitude and latitude, using the Inside Airbnb geojson map and shapely library.

In [ ]:
```
from shapely.geometry import Point

neighborhoods_gdf = gpd.read_file('../Data/neighbourhoods.geojson')

# Convert the attractions DataFrame to a GeoDataFrame
attractions_gdf = gpd.GeoDataFrame(
    attractions_df_cleaned,
    geometry=gpd.points_from_xy(attractions_df_cleaned.Longitude, attractions_df_cleaned.Latitud
    crs="EPSG:4326"  # Set the coordinate reference system to WGS84
)

# Perform a spatial join to assign neighborhoods to attractions
attractions_with_neighborhoods = gpd.sjoin(attractions_gdf, neighborhoods_gdf, how="left", op='w

# The result is a new GeoDataFrame with attractions and their corresponding neighborhood names a
# Drop the index_right column which comes from the neighborhoods_gdf
attractions_with_neighborhoods.drop(columns=['index_right'], inplace=True)
```

We can note there are 358 attractions with 'null' neighborhoods. Observing these rows shows that most of them are either parks or inside parks. A handful of attractions are technically outside New York City, but we will keep them in because they are still close enough to affect airbnb prices.

In [75]:
```
attractions_with_neighborhoods.isnull().sum()
```

Out[75]:
```
Name                   0
Address                0
Latitude               0
Longitude              0
Rating                67
Total Ratings         67
Place ID               0
Types                  0
geometry               0
neighbourhood        358
neighbourhood_group  358
dtype: int64
```

Visualizing the Scraped Dataset

We can now visualize our scraped dataset. One wise plot to show would be a map of all the airbnb listings, tourist attractions, along with price, and popularity score. We can achieve this by creating a popularity score consisting of the normalized ratings times total ratings, and normalized price with outliers dropped. While ideally we would plot a single map with complementary colors, I found it was extremely hard to see anything with both variables on one map so we will plot two different maps instead.

We normalize price after dropping outliers outside the 10th and 90th percentile.

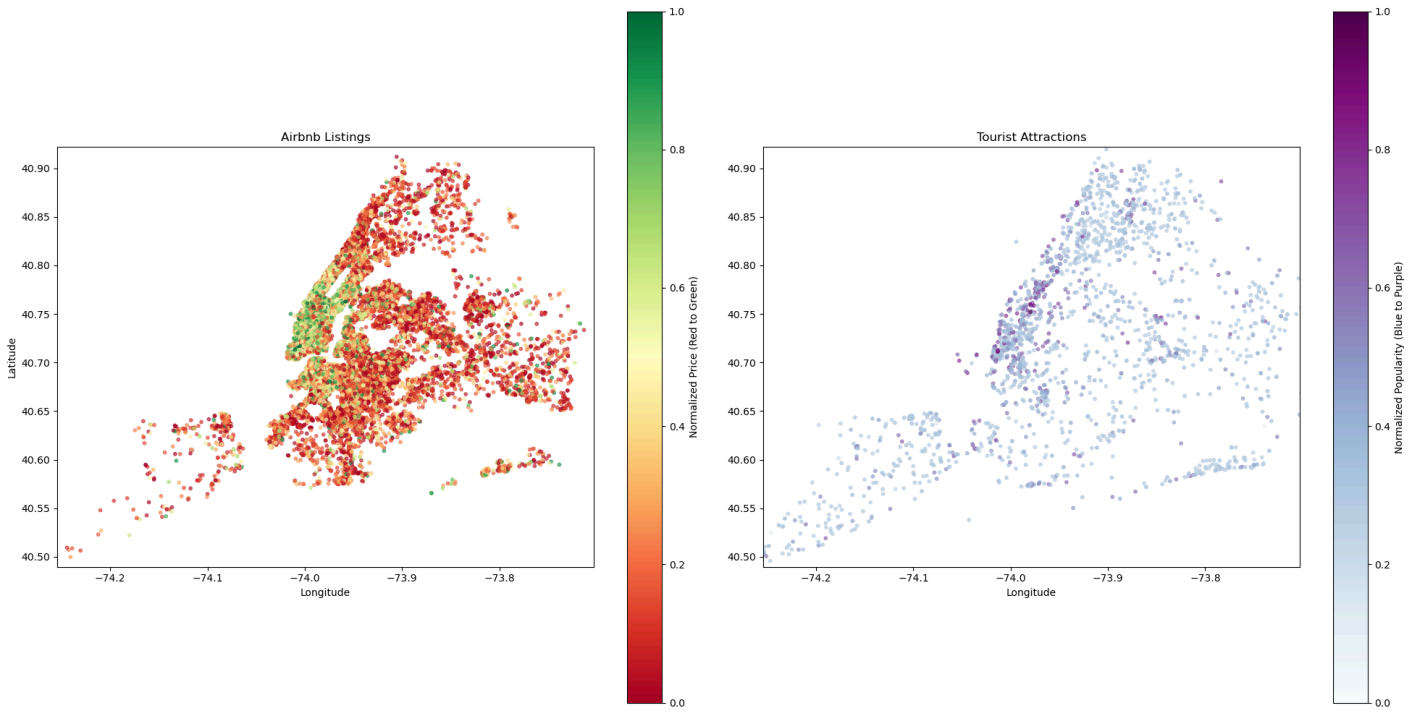
In [189…
```
# Calculate the 10th and 90th percentiles
q10 = df['price'].quantile(0.1)
q90 = df['price'].quantile(0.9)

# Filter the dataset to remove outliers outside these quantiles
filtered_df = df[(df['price'] >= q10) & (df['price'] <= q90)]
filtered_df['price_normalized'] = (filtered_df['price'] - filtered_df['price'].min()) / (filtere
```

Normalize weighted rating of tourist attractions.

In [ ]:
```
C = attractions_with_neighborhoods['Rating'].mean()
m = attractions_with_neighborhoods['Total Ratings'].quantile(0.75)

def weighted_rating(x, m=m, C=C):
    v = x['Total Ratings']
    R = x['Rating']
    return (v/(v+m) * R) + (m/(m+v) * C)

# Apply the function to each row
attractions_wnp = attractions_with_neighborhoods
attractions_wnp['Popularity'] = attractions_with_neighborhoods.apply(weighted_rating, axis=1)

min_popularity = attractions_wnp['Popularity'].min()
max_popularity = attractions_wnp['Popularity'].max()

# Apply the Min-Max scaling to normalize the values
attractions_wnp['Normalized Popularity'] = (attractions_wnp['Popularity'] - min_popularity) / (m
```

In [191…
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 10))

# Defining the axis limits to the geographic extent of the points
lon_min, lon_max = filtered_df['longitude'].min() - 0.01, filtered_df['longitude'].max() + 0.01
lat_min, lat_max = filtered_df['latitude'].min() - 0.01, filtered_df['latitude'].max() + 0.01
```

```
# Airbnb listings map with equal aspect ratio and adjusted plot limits
ax1.scatter(filtered_df['longitude'], filtered_df['latitude'],
            c=filtered_df['price_normalized'], cmap='RdYlGn', s=10, alpha=0.6)
ax1.set_title('Airbnb Listings')
ax1.set_xlabel('Longitude')
ax1.set_ylabel('Latitude')
ax1.set_xlim(lon_min, lon_max)
ax1.set_ylim(lat_min, lat_max)
ax1.set_aspect('equal', adjustable='box')

# Adding color bar for Airbnb listings
sm_airbnb = plt.cm.ScalarMappable(cmap='RdYlGn', norm=plt.Normalize(vmin=0, vmax=1))
sm_airbnb.set_array([])
fig.colorbar(sm_airbnb, ax=ax1, orientation='vertical', label='Normalized Price (Red to Green)')

# Tourist attractions map with equal aspect ratio and adjusted plot limits
ax2.scatter(attractions_wnp['Longitude'], attractions_wnp['Latitude'],
            c=attractions_wnp['Normalized Popularity'], cmap='BuPu', s=10, alpha=0.6)
ax2.set_title('Tourist Attractions')
ax2.set_xlabel('Longitude')
ax2.set_xlim(lon_min, lon_max)
ax2.set_ylim(lat_min, lat_max)
ax2.set_aspect('equal', adjustable='box')

# Adding color bar for tourist attractions
sm_attractions = plt.cm.ScalarMappable(cmap='BuPu', norm=plt.Normalize(vmin=0, vmax=1))
sm_attractions.set_array([])
fig.colorbar(sm_attractions, ax=ax2, orientation='vertical', label='Normalized Popularity (Blue

# Adjust layout
plt.tight_layout()
plt.show()
```



Looking at the map it appears that price is strongly correlated with tourist attractions. We can see that popular tourist destinations pretty much everywhere in NYC are almost always surrounded by higher priced listings. Locations with more tourist attractions also appear to have more airbnbs, while locations that are sparse of tourist destinations also have blank spots in listings. Economically, this makes sense, as we would expect that in areas with more attractions there would be a higher economic incentive to host an airbnb.

The result is of major benefit to our research question in that we now understand that more tourist locations in neighborhoods results in higher listing prices, however I believe we can show this result a little more quantitatively by creating a new variable in our airbnb dataset measuring each listings proximity to all tourist attractions weighted by the popularity score.

We will proceed with this approach.

In [196…
```
attractions_wnp = pd.read_csv('attractions_wnp.csv')
```

In [197…
```
filtered_dfx = filtered_df.dropna(subset=['latitude', 'longitude'])
attractions_wnpx = attractions_wnp.dropna(subset=['Latitude', 'Longitude', 'Normalized Popularity
```

In [25]:
```
import numpy as np

# Convert attractions DataFrame to a list of tuples for faster access
attractions_list = [tuple(x) for x in attractions_wnpx[['Latitude', 'Longitude', 'Normalized Popu

def calculate_distance(lat1, lon1, lat2, lon2):
    # Calculate the Euclidean distance between two points
    return np.sqrt((lat2 - lat1) ** 2 + (lon2 - lon1) ** 2)

# Calculate weighted tourism proximity for each listing
proximity_scores = []
for listing in filtered_dfx.itertuples(index=False):
    weights = []
    for attraction in attractions_list:
        # Check if any value is NaN before proceeding with distance calculation
        if np.isnan(listing.latitude) or np.isnan(listing.longitude) or np.isnan(attraction[0]) 
            print("NaN found in coordinates")
            continue  # Skip this iteration

        distance = calculate_distance(listing.latitude, listing.longitude, attraction[0], attrac
        if np.isnan(distance) or distance <= 0:
            print("Invalid distance computed or division by zero")
            # Handle invalid distance; maybe continue or set a default weight
            continue
        else:
            weight = attraction[2] / distance
            weights.append(weight)

    # Aggregate weights, e.g., by summing them
    proximity_score = sum(weights)
    proximity_scores.append(proximity_score)

# Add the new column to the listings DataFrame
filtered_dfx['Tourism Proximity'] = proximity_scores
```

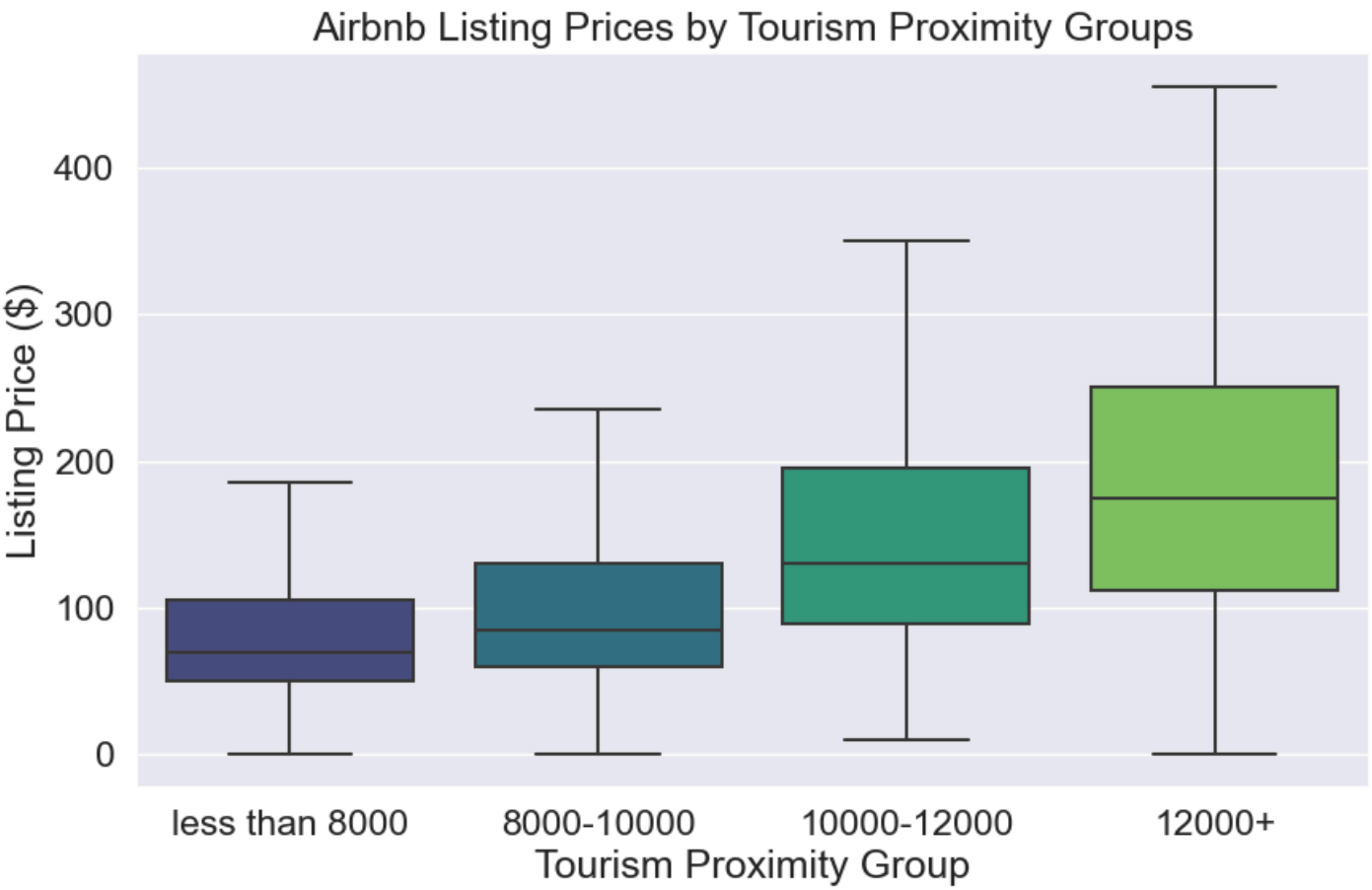
Now that we have this data, we can create a plot to observe the correlation between tourism proximity and a listing's prices. A boxplot provided the most comprehensive, yet least messiest way to display all our data. Observing the box plot we can observe that there does appear to be a linear correlation between price and tourism proximity, and the correlation coefficient of about 0.39 is not insignificant and suggests correlation. This is a strong relationship and what we expected to find and I believe if we look at the more active listings and further interaction terms we will find a stronger relationship.

In terms of our research question we are now able to say that an airbnb listings's proximity to popular tourist locations is a positive determinant to its price, and that neighborhood characteristics like proximity to

tourist destinations is an important factor for a listing's price. This of course is another crucial finding to help our answer our research question.

In [203…

```
# Define the function to categorize tourism proximity
def tourism_proximity_label(proximity):
    if proximity < 8000:
        return 'less than 8000'
    elif 8000 <= proximity <= 10000:
        return '8000-10000'
    elif 10000 <= proximity <= 12000:
        return '10000-12000'
    else:
        return '12000+'

# Apply the function to the 'tourism_proximity' column
adf['Tourism Proximity Group'] = adf['Tourism Proximity'].apply(tourism_proximity_label)

# Create the box plot
plt.figure(figsize=(10, 6))
sns.boxplot(x='Tourism Proximity Group', y='price', data=adf, palette='viridis',
            order=['less than 8000', '8000-10000', '10000-12000', '12000+'], showfliers=False)
plt.title('Airbnb Listing Prices by Tourism Proximity Groups')
plt.xlabel('Tourism Proximity Group')
plt.ylabel('Listing Price ($)')
# If the 'price' column is actually 'log_price' or otherwise named, adjust the 'y' parameter acc
plt.show()
```



In [132…

```
correlation = filtered_dfx['Tourism Proximity'].corr(filtered_dfx['price'])
print(f"Correlation coefficient between Airbnb price and tourism proximity score: {correlation}"
```

```
Correlation coefficient between Airbnb price and tourism proximity score: 0.3891828844855738
```

Adding a New Dataset

The American Community Survey can provide a wealth of neighborhood characteristics for our research, and includes data such as demographic, social, economic, and housing characteristics for each neighborhood. We can see how factors like median income levels, population density, age distribution, etc, impact airbnb prices. Even though previous literature has indicated that the socioeconomic characteristics of a neighborhood have little impact on price, it would be great to confirm this finding for our locality. I'll merge in data on income with our airbnb data so that we may identify patterns that affect listing prices. Neighborhood income stats are of course relevant neighborhood characteristics, and therefore relevant to our research question.

We should note that the ACS does note contain data on the geographic level of 'neighborhoods' as in the airbnb dataset, the closest approximation would be 'Public Use Microdata Areas' (PUMA), however to avoid mapping NYC's various geographic levels and complicating our dataset we will settle for borough (or county) level data. Lets merge in this dataset and begin exploring the connections between various neighborhood characteristics and airbnb prices.

In [116…
```
# Loading the cleaned ACS median income data
income_data_path = '../Data/cleaned_acs_median_income.csv'
income_data_df = pd.read_csv(income_data_path)
```

In [70]:
```
income_data_df.head()
```

Out[70]:

	Borough	Median Household Income
0	Bronx	41432
1	Brooklyn	66937
2	Manhattan	93651
3	Queens	72561
4	Staten Island	89821

In [ ]:
```
# Merging the income data with the Airbnb data
airbnb_merged = pd.merge(filtered_dfx, income_data_df, left_on='neighbourhood_group', right_on='
```

Now that we merged, we would naturally like to see what affect median income might have on a listing's price. What can we observe is a clear relationship between a neighborhoods income and increased airbnb prices. This shows that neighborhood median income is a neighborhood characteristic which has a positive affect on prices, and furthers an answer to our research question.

The prices may reflect the economic profile of these neighborhoods, including higher real estate costs and greater purchasing power of potential guests. However, the significant overlap in price ranges across all income brackets indicates that a variety of factors beyond neighborhood income levels influence Airbnb pricing, such as the individual characteristics of each listing, local demand and supply dynamics, and the diverse preferences of travelers. This helps us further understand how neighborhood level characteristics impact airbnb prices.

In [175…
```
def income_bracket_label(income):
    if income < 60000:
```

```
        return '< $60,000'
    elif 60000 <= income <= 80000:
        return '$60,000 - $80,000'
    else:
        return '> $80,000'

airbnb_with_income_cleaned = airbnb_merged
airbnb_with_income_cleaned['Income Bracket'] = airbnb_with_income_cleaned['Median Household Incor

# Creating the box plot with specified income ranges
plt.figure(figsize=(10, 6))
sns.boxplot(x='Income Bracket', y='price', data=airbnb_with_income_cleaned, palette='viridis', or
plt.title('Airbnb Listing Prices by Specified Income Brackets')
plt.xlabel('Income Bracket')
plt.ylabel('Listing Price ($)')
plt.ylim(0, 300)
plt.show()
```

Final Project

Throughout the project and our study of listing price determinants we have come to see a mostly linear relationship betwen many of our independent variables and price. We observed linear correlations (both postitive and negative) between tourism proximity, reviews, housing supply, and neighborhood characteristics such as income and vacant housing units. We have also observed non linear relationship with calculated host listings count and price.

In order to gain a more comprehensive understanding of airbnb pricing and have a better answer to our research question, we will split our various independent variables into subgroups, and in the interest of interpretability and isolating effects I chose to break the regressions into subgroups consisting of

neighborhood, listing, and host variables, as well as our final comprehensive regressions. As we go through each regression we will include further explanations behind our variable choices.

Note that we drop listings with minimum nights outside the 90th quantile (28 days). While we would prefer to include all listing leaving those values in likely effectively misspecifies our models (which are looking at short term rentals) and results in issues with extremely high impossible coefficients and results on certain regressions. We should also note there is very limited data for listings with minimum nights over 28 in our dropped dummy variables (shared room for room type and the bronx for borough). For reasons which I was unable to conclusively determine, 28 days was the exact boundary before which my test regressions began to experience extreme anomalies, though its likely because listings with minimum nights over 28 days are subject to additional legal regulations. Dropping this data takes us from 48895 to 44048 listings. We could go further and drop listings with zero number of reviews, however the pricing of these listings is still relevant to our research question.

We proceed to setup our dataframe.

In [2]:
```
adf = pd.read_csv("final_df.csv")
quantile_90 = adf['minimum_nights'].quantile(0.90)
adf = adf[adf['minimum_nights'] <= quantile_90]
len(adf)
```

Out[2]:
```
44048
```

In [ ]:
```
# Merge all datasets into airbnb dataset
adf = pd.merge(df, income_data_df, left_on='neighbourhood_group', right_on='Borough', how='left'
adf = adf.merge(cn_df[['Name', 'VacHUs']], how='left', left_on='neighbourhood_group', right_on='
adf['const'] = 1
```

In [14]:
```
# Redo Tourism Proximity calculation for entire dataframe
import numpy as np

# Convert attractions DataFrame to a list of tuples for faster access
attractions_list = [tuple(x) for x in attractions_wnpx[['Latitude', 'Longitude', 'Normalized Popu

def calculate_distance(lat1, lon1, lat2, lon2):
    # Calculate the Euclidean distance between two points
    return np.sqrt((lat2 - lat1) ** 2 + (lon2 - lon1) ** 2)

# Calculate weighted tourism proximity for each listing
proximity_scores = []
for listing in adf.itertuples(index=False):
    weights = []
    for attraction in attractions_list:
        # Check if any value is NaN before proceeding with distance calculation
        if np.isnan(listing.latitude) or np.isnan(listing.longitude) or np.isnan(attraction[0]) 
            print("NaN found in coordinates")
            continue  # Skip this iteration

        distance = calculate_distance(listing.latitude, listing.longitude, attraction[0], attrac
        if np.isnan(distance) or distance <= 0:
            print("Invalid distance computed or division by zero")
            # Handle invalid distance; maybe continue or set a default weight
            continue
        else:
            weight = attraction[2] / distance
            weights.append(weight)
```

```
    # Aggregate weights, e.g., by summing them
    proximity_score = sum(weights)
    proximity_scores.append(proximity_score)

# Add the new column to the listings DataFrame
adf['Tourism Proximity'] = proximity_scores
```

In [15]:
```
# Create log price column
adf['price'] = adf['price'].replace(0, 0.01)
adf['log_price'] = np.log(adf['price'])
```

In [16]:
```
# Create room type dummy variables
adf['private_room'] = 0
adf['entire_homeapt'] = 0

adf.loc[adf['room_type'] == 'Private room', 'private_room'] = 1
adf.loc[adf['room_type'] == 'Entire home/apt', 'entire_homeapt'] = 1
```

In [114…
```
# Create squared variables
continuous_vars = ['Median Household Income', 'reviews_per_month', 'minimum_nights', 'calculated_

# Create squared terms for these variables
for var in continuous_vars:
    adf[f'{var} Squared'] = adf[var] ** 2
```

In [ ]:
```
# Create borough dummy variables
borough_dummies = pd.get_dummies(adf['Borough'], drop_first=True, prefix='borough')
adf = adf.join(borough_dummies)

adf['borough_Brooklyn'] = adf['borough_Brooklyn'].astype(int)
adf['borough_Manhattan'] = adf['borough_Manhattan'].astype(int)
adf['borough_Queens'] = adf['borough_Queens'].astype(int)
adf['borough_Staten Island'] = adf['borough_Staten Island'].astype(int)
```

In [144…
```
# Create normalized variables
adf['Normalized_Tourism_Proximity'] = (adf['Tourism Proximity'] - adf['Tourism Proximity'].min()
adf['Normalized_Tourism_Proximity_Squared'] = adf['Normalized_Tourism_Proximity']**2
adf['Tourism Proximity Squared'] = adf['Tourism Proximity']**2
adf['Normalized_Median_Income'] = (adf['Median Household Income'] - adf['Median Household Income
adf['Normalized_VacHUs'] = (adf['VacHUs'] - adf['VacHUs'].min()) / (adf['VacHUs'].max() - adf['V
```

Seeing how neighborhood characteristics affect price is one of the fundamental questions of our research question, encapsulating location value. Tourism proximity serves to determine a location's attractiveness to tourists; who we speculated are the primary users of airbnb in New York City. Tourism proximity squared captures non-linear effects; we include this term both because we observed a possible exponential relation with price in our graphs, and also because we can hypothesize that locations with an extremely high tourism score may actually be less preferred- reflecting the phenomenon of alternative tourism. Median neighborhood income is a proxy for local economic health; areas with higher incomes may have more expensive properties and thus command a higher airbnb price. Neighborhood income may also capture other neighborhood characteristics related to income such as cleanliness, crime rates, and other socio-economic indicators. Available housing units indicate supply levels; more vacant units might reduce prices due to increased competition among sellers, whereas fewer vacancies could lead to higher prices.

We will now run our regressions with our neighborhood variables, including median household income, vacant housing units, tourism proximity, and dummy variables for our boroughs.

In [252…

```
# Run regressions 1 to 3
X1 = ['const', 'Normalized_Tourism_Proximity', 'Normalized_Tourism_Proximity_Squared','Normalize
X2 = ['const', 'Normalized_Tourism_Proximity', 'Normalized_Tourism_Proximity_Squared','Normalize
      'borough_Brooklyn','borough_Manhattan','borough_Queens','borough_Staten Island']
X3 = ['const', 'borough_Brooklyn','borough_Manhattan','borough_Queens','borough_Staten Island']

reg1 = sm.OLS(endog=adf['log_price'], exog=adf[X1],
              missing = 'drop')
reg2 = sm.OLS(endog=adf['log_price'], exog=adf[X2],
              missing = 'drop')
reg3 = sm.OLS(endog=adf['log_price'], exog=adf[X3],
              missing = 'drop')
```

The results table shows that between the first two regressions with our neighborhood characteristics, and with and without the neighborhood dummy variables; the R^2 increases only by 0.03. This result combined with the relatively high coeffiecient of tourism proximity in the first regression and the highrt coefficients in our second regression with the neighborhood dummy variables suggests multicollinearity between Tourism Proximity and other location variables. We can confirm this results by taking the VIF scores, which return a VIF of around 1.99 for tourism proximity, which is normal, a VIF of over 1170 for VacHUs which is very high, and an infinite score for median household income and our dummy borough variables- suggesting near perfect multicollinearity. We would expect a new theoretical variable to add multicollinearity so the values themselves are not problematic, but since the VIF score for tourism proximity is relatively low what this confirms is that tourism proximity has extremely high explanatory power over location variables, and in fact explains away nearly all high level changes and some more. Which as we observed in the mapping of price to neighborhoods- borough level changes was a bigger differentiator of price than more granular neighborhood zones, which further confirms that tourism proximity tells us more than just location based characteristics. Comparing our first and third model proves this, as our first model has a higher R^2 than the third (0.196 vs 0.116) and explains more than just the changes in borough. This result is very significant for our research question in telling us that Tourism Proximity is the main driver of location's effect on price.

We can also note that the tourism proximity squared is negative as we expected. This tells us that at some point (roughly at a 0.6 normalized tourism proximity score), an even larger tourism proximity score will result in a negative effect on price. This suggests that our hypothesis might be correct but we cannot definitively say so. Looking at Vacant Housing units it does not appear to have a particulary large or very statistically significant effect in our first regression, but the coefficient increases once we add neighborhood dummy variables. This seems strange and could be attributed to either bias or perhaps adding in the neighborhood variables acted as a control variable. In either case we can hypothesize the positive coefficient of vacant housing units economically as higher desirability neighborhoods having higher vacant housing units. This does run counter to what we might expect but reflects what we observed in our graphs.

Furthermore our results are statistically significant with good p-values and f-stats.

In [253…

```
# Create stargazer table
stargazer = Stargazer([reg1.fit(), reg2.fit(),reg3.fit()])

HTML(stargazer.render_html())
```

Out[253…

	Dependent variable: log_price		
	(1)	(2)	(3)
Normalized_Median_Income	0.154***	0.188***	
	(0.021)	(0.020)	
Normalized_Tourism_Proximity	6.018***	6.293***	
	(0.190)	(0.192)	
Normalized_Tourism_Proximity_Squared	-4.965***	-5.228***	
	(0.409)	(0.409)	
Normalized_VacHUs	0.028	0.356***	
	(0.027)	(0.011)	
borough_Brooklyn		-0.005	0.327***
		(0.011)	(0.021)
borough_Manhattan		-0.177***	0.735***
		(0.007)	(0.021)
borough_Queens		0.028**	0.151***
		(0.013)	(0.022)
borough_Staten Island		0.379***	0.133***
		(0.022)	(0.040)
const	3.657***	3.412***	4.242***
	(0.018)	(0.027)	(0.020)
Observations	44048	44048	44048
R^2	0.196	0.199	0.116
Adjusted R^2	0.196	0.199	0.116
Residual Std. Error	0.628 (df=44043)	0.627 (df=44041)	0.659 (df=44043)
F Statistic	2691.507*** (df=4; 44043)	1822.630*** (df=6; 44041)	1439.035*** (df=4; 44043)
Note:			*p<0.1; **p<0.05; ***p<0.01

In [181…

```
# Calculate VIF for neighborhood variables
from statsmodels.stats.outliers_influence import variance_inflation_factor

# First, we need to create a new DataFrame for the independent variables, dropping the constant
X_independent = adf[['Normalized_Tourism_Proximity',
                     'Normalized_Median_Income', 'Normalized_VacHUs',
                     'borough_Brooklyn', 'borough_Manhattan', 'borough_Queens',
```

```
                    'borough_Staten Island']]

# Calculate VIF for each independent variable
vif_data = pd.DataFrame()
vif_data["Feature"] = X_independent.columns
vif_data["VIF"] = [variance_inflation_factor(X_independent.values, i) for i in range(X_independe
vif_data
```

Out[181…

	Feature	VIF
0	Normalized_Tourism_Proximity	1.986584
1	Normalized_Median_Income	inf
2	Normalized_VacHUs	1170.243356
3	borough_Brooklyn	inf
4	borough_Manhattan	inf
5	borough_Queens	inf
6	borough_Staten Island	inf

We would next like to see the effect that user observable listing characteristics have on price. This represents the other half of our research question. We will use dummy variables for our room type's dropping shared room. Room types reflect the product differentiation in the housing market. We expect entire homes/apartments to command higher prices due to greater privacy and space, while singular (private) rooms are likely cheaper. Reviews_per_month and number_of_reviews serve as signals of listing quality and popularity, potentially correlating with price as they provide a measure of consumer satisfaction and frequency of rental—listings with higher review counts and frequency can often charge premium prices. We can speculate that reviews per month and number of reviews may also internalize other listing characteristics for which we dont have direct data on, such as amenties, features, and host interactions. We can also speculate that a higher minimum nights requirement will correspond to a lower price, based off our knowledge of supply and demand, and segmenting and quantity discounts.

The interaction term tests for differential valuation of consumer feedback across product categories; a private room might see a different price impact from reviews compared to an entire home, as customer expectations and experiences can differ based on room type. We would also like to confirm our previous visual findings that number of reviews have no impact on price, and monthly reviews have a negative impact. We will now run three regressions, first seeing how our afforementioned listing variables alone affect price, how interaction terms affect price, and how our neighborhood characteristics by room type affects price.

In [6]:
```
# Normalize more terms
# Create normalized variables
adf['Normalized_reviews_per_month'] = (adf['reviews_per_month'] - adf['reviews_per_month'].min()
adf['Normalized_number_of_reviews'] = (adf['number_of_reviews'] - adf['number_of_reviews'].min()
adf['Normalized_minimum_nights'] = (adf['minimum_nights'] - adf['minimum_nights'].min()) / (adf[
adf['Normalized_minimum_nights_squared'] = adf['Normalized_minimum_nights']**2
```

In [7]:
```
# Create listing interaction terms
# Create interaction terms for Normalized Median Income with room types
adf['income_x_private_room'] = adf['Normalized_Median_Income'] * adf['private_room']
adf['income_x_entire_homeapt'] = adf['Normalized_Median_Income'] * adf['entire_homeapt']
```

```
# Create interaction terms for Tourism Proximity with room types
adf['tourism_x_private_room'] = adf['Normalized_Tourism_Proximity'] * adf['private_room']
adf['tourism_x_entire_homeapt'] = adf['Normalized_Tourism_Proximity'] * adf['entire_homeapt']

# Create interaction terms for Minimum Nights with room types
adf['min_nights_x_private_room'] = adf['Normalized_minimum_nights'] * adf['private_room']
adf['min_nights_x_entire_homeapt'] = adf['Normalized_minimum_nights'] * adf['entire_homeapt']

# Create interaction terms for Reviews per Month with room types
adf['reviews_per_month_x_private_room'] = adf['Normalized_reviews_per_month'] * adf['private_roo
adf['reviews_per_month_x_entire_homeapt'] = adf['Normalized_reviews_per_month'] * adf['entire_ho
```

In [64]:
```
# Run regression 4 to 6
reg4 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'private_room', 'entire_homeapt', 'Norma
                                                'Normalized_minimum_nights_squared']],
              missing = 'drop')
reg5 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'private_room', 'entire_homeapt', 'Norma
                                                'Normalized_minimum_nights','reviews_per_month_x
                                                  'min_nights_x_entire_homeapt']],
              missing = 'drop')
reg6 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'private_room', 'entire_homeapt', 'Norma
                                                'Normalized_minimum_nights','reviews_per_month_x
              missing = 'drop')
```

The first thing we can take away from our regressions is that entire home/apt command a significant premium to private room's, who also command a price premium compared to our baseline shared rooms. This result is not surprising but confirms our visual findings.

We can also see a strong negative relationship between minimum nights and price through all three regressions. This relationship is somewhat unexpected but can be economically explained by a discounting strategy; where hosts might may lower prices to attract bookings that guarantee occupancy for a more extended period, reducing the operational hassles and costs associated with frequent turnovers, such as cleaning, maintenance, and time spent communicating with potential guests. More unusually is that the coefficient of minimum nights squared is positive, indicating that at some point hosts will charge a premium for higher minimum nights. This could be economically explained by hosts targeting relatively inelastic longer term tourists, i.e people who need to stay in an airbnb for longer than a few days or weeks, but for whatever reason might not be able to secure an actual rental unit or other accommodation. Though this result may become more clear in our next regressions.

We can see that number of reviews and reviews per month both have a negative relationship on price, however the interaction terms between reviews per month and room type have positive coefficents. The positive coefficients of the review interaction terms is somewhat unusally high and may suggest a potential bias, however this can just be a side effect of the normalization. If we are to interpret them at face value then we can say that reviews per month can have a very positive effect on price for room types which are non shared rooms. This can potentially reflect the fact that entire home apt's and private rooms are far more popular, and that there may be a select few listings which are very popular. The lack of change in R^2 between the first and second term suggest that listing interaction terms have little effect on the variance of price, thus we eliminate them from the third regression to see what effects might occur.

Looking at our last regression where we attempt to see how tourism proximity differs across room type we can see that the effect of tourism proximity is much more pronouced for private rooms than entire home/apartments. There are many possible reasons why this might be the case but we can speculate that it might be because hosts living in areas with more tourist attractions are more incentivized to rent out rooms

in their home, while hosts renting out entire home/apartments may not have the same flexibility in choosing where exactly to rent out. This hypothesis would be difficult to prove without detailed info about hosts however.

Lastly we should note that the standard error and t-stat, as well as p-values and f-stat for all the regressions are high, showing that there is statistical significance for our variables.

In [133…

```
# Create stargazer table for regressions 4-6
stargazer = Stargazer([reg4.fit(), reg5.fit(),reg6.fit()])

HTML(stargazer.render_html())
```

Out[133…

			Dependent variable: log_price
	(1)	(2)	(3)
Normalized_Tourism_Proximity			3.076***
			(0.224)
Normalized_minimum_nights	-0.468***	-1.141***	-0.585***
	(0.052)	(0.191)	(0.047)
Normalized_minimum_nights_squared	0.252***	0.233***	0.363***
	(0.069)	(0.070)	(0.063)
Normalized_number_of_reviews	-0.139***	-0.142***	-0.185***
	(0.042)	(0.042)	(0.038)
Normalized_reviews_per_month	-0.638***	-2.409***	-2.795***
	(0.115)	(0.685)	(0.610)
const	3.959***	4.037***	3.481***
	(0.019)	(0.029)	(0.044)
entire_homeapt	1.203***	1.123***	1.070***
	(0.019)	(0.029)	(0.046)
min_nights_x_entire_homeapt		0.753***	
		(0.191)	
min_nights_x_private_room		0.637***	
		(0.191)	
private_room	0.370***	0.289***	0.232***
	(0.019)	(0.029)	(0.046)
reviews_per_month_x_entire_homeapt		1.552**	2.945***
		(0.699)	(0.622)
reviews_per_month_x_private_room		2.029***	2.794***
		(0.696)	(0.620)
tourism_x_entire_homeapt			-0.037
			(0.230)
tourism_x_private_room			0.487**
			(0.231)
Observations	35973	35973	35973

R^2	0.398	0.399	0.509
Adjusted R^2	0.398	0.399	0.509
Residual Std. Error	0.524 (df=35966)	0.523 (df=35962)	0.473 (df=35961)
F Statistic	3971.093*** (df=6; 35966)	2387.872*** (df=10; 35962)	3389.259*** (df=11; 35961)

Note: *p<0.1; **p<0.05; ***p<0.01

Host characteristics represent another angle to listing characteristics. We can speculate that hosts operating a higher number of listings may operate more economically as skilled hosts. This reflects the idea that these hosts are professional and are likely to have greater market experience, which could translate into more economically driven and standardized pricing practices. Thus"Calculated Host Listings Count" is a proxy for the level of professionalism and market experience of a host. A higher count might indicate a host's transition from an individual sharing their personal space to a business-like operation managing multiple properties.

While for the most part reviews have been of minimal importance we can speculate more economical hosts may incorporate a listing's perceived worth better into its price. Hence we will test such an interaction term against various variables. Furthermore we can speculate that more hosts with more listings will incorporate more variables into their pricing decisions, such as minimum nights and tourism proximity. Lastly seeing how calculated host listings interacts with availability 365 and price can give us a glimpse into determining if more economical hosts are more succesful in the market, by proxy of hosts with more listings having lower availability 365. We include the minimum night's variable in these regressions as well because they are technically a host decided characteristic as well.

In [4]:
```
# Normalize more terms
adf['Normalized_availability_365'] = (adf['availability_365'] - adf['availability_365'].min()) /
adf['Normalized_reviews_per_month'] = (adf['reviews_per_month'] - adf['reviews_per_month'].min()
adf['Normalized_host_listings_count'] = (adf['calculated_host_listings_count'] - adf['calculated_
```

In [8]:
```
# Create host interaction terms
adf['listings_count_x_min_nights'] = adf['Normalized_host_listings_count'] * adf['Normalized_min
adf['listings_count_x_monthly_reviews'] = adf['Normalized_host_listings_count'] * adf['Normalized
adf['listings_count_x_availability_365'] = adf['Normalized_host_listings_count'] * adf['Normaliz

adf['Normalized_host_listings_count_squared'] = adf['Normalized_host_listings_count']**2
adf['Normalized_host_listings_count_cubed'] = adf['Normalized_host_listings_count']**3

adf['listings_count_squared_x_monthly_reviews'] = adf['Normalized_host_listings_count_squared'] *
adf['listings_count_squared_x_availability_365'] = adf['Normalized_host_listings_count_squared']
```

In [85]:
```
# Run regressions 7-9
reg7 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'Normalized_host_listings_count', 'Norma
                                                'Normalized_reviews_per_month']],
              missing = 'drop')

reg8 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'Normalized_host_listings_count', 'Norma
                                                'Normalized_reviews_per_month','Normalized_host_
              missing = 'drop')

reg9 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'Normalized_host_listings_count','Normal
                                               'Normalized_minimum_nights','Normalized_availabi
```

```
                                                'listings_count_x_min_nights', 'listings_count_x_
                                                'listings_count_squared_x_monthly_reviews', 'lis
            missing = 'drop')
```

Looking at our first regression shows that host listings count does have some overall positive effect on price, but we know from our graphs that the relationship is more complicated so we add a squared term for it in the next regressions and the results are as expected. Host's with a lower host listings count charge relatively less than those with higher host listings count. We observe similar results with the interaction between calculated host listings and monthly reviews. Economically, a negative coefficient for the interaction of listings count and monthly reviews suggests that initially, as hosts accumulate more listings and reviews, there might be a diminishing return in terms of price. However, the positive coefficient for the squared term of this interaction implies that beyond a certain level, hosts with a large number of listings can leverage their scale and high volume of reviews to command higher prices.

[maybe include this in next paragraph] We should note that overall, the effects of our host characteristics and interaction terms explain very little of the variance in log price, as even in our third expanded regression we get an R^2 of 0.035. The results however are statistically significant, but economically not so much. We should note that this result for host characteristics is less economically significant than we might have expected based off the results of previous research; who found host characteristics to be on par with room characteristics. However we do not have access to the same level of host characteristics as they do, and attempts to recreate measures of different hosts characteristics via interaction terms introduce a lot of complexity.

In [86]:
```
# Stargazer table for regressions 7-9
stargazer = Stargazer([reg7.fit(), reg8.fit(),reg9.fit()])

HTML(stargazer.render_html())
```

Out[86]:

	Dependent variable: log_price		
	(1)	(2)	(3)
Normalized_availability_365	0.149***	0.184***	0.178***
	(0.011)	(0.011)	(0.011)
Normalized_host_listings_count	0.766***	-5.262***	-4.839***
	(0.049)	(0.399)	(1.008)
Normalized_host_listings_count_squared		6.093***	5.485***
		(0.400)	(1.021)
Normalized_minimum_nights	0.825***	0.779***	0.770***
	(0.066)	(0.066)	(0.066)
Normalized_minimum_nights_squared	-1.267***	-1.187***	-1.282***
	(0.088)	(0.088)	(0.088)
Normalized_reviews_per_month	-1.062***	-1.063***	-0.157
	(0.128)	(0.128)	(0.145)
const	4.627***	4.636***	4.628***
	(0.007)	(0.007)	(0.007)
listings_count_squared_x_availability_365			-2.779**
			(1.281)
listings_count_squared_x_monthly_reviews			202.946***
			(16.208)
listings_count_x_availability_365			2.250*
			(1.257)
listings_count_x_min_nights			12.282***
			(1.415)
listings_count_x_monthly_reviews			-199.405***
			(15.929)
Observations	35973	35973	35973
R^2	0.020	0.026	0.035
Adjusted R^2	0.020	0.026	0.034
Residual Std. Error	0.668 (df=35967)	0.666 (df=35966)	0.663 (df=35961)
F Statistic	147.089*** (df=5; 35967)	161.943*** (df=6; 35966)	117.157*** (df=11; 35961)

Note: $^{*}p<0.1$; $^{**}p<0.05$; $^{***}p<0.01$

We will now run a final set of comprehensive regressions picking the most important variables using the results we have found from our previous regressions looking at neighborhood, listings, and indirect listing 'host' characteristics. The first regression will be our model specification and in terms of normalized variables, while our second regression will be the same but use the original units in order to take away a few final results in terms of single unit increases where possible.

From the first set of regressions we ran we took away that tourism proximity, median neighborhood and VacHUs were two of the biggest neighborhood factors affecting price, and explained more than just location. In regression 9 and 10 we can see that the effect of median neighborhood income is almost entirely explained by vacant housing, thus we omit it from our final specification (regression 12). We also confirmed that room type was a similarly important determinant of price, and minimum nights also had a mild effect on price. Listing availability, reviews per month and the calculated host listing amount were also host characteristics that had a significant effect on price. Further interaction terms related to room type and the listing count also had mild, yet statistically significant and economically explainable effects on price. We should note that our hypothesis that listings count interaction with availability 365 served as proxy for host success was mostly true, and to simplify the model we will use that as our sole interaction term for host characteristics.

In [151…

```
# Create final ols regression
reg10 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'Normalized_Tourism_Proximity', 'Norma
                                                 'private_room', 'entire_homeapt',
                                                 'Normalized_availability_365', 'Normalized_host_
                                                 'Normalized_host_listings_count_squared',
                                                 'Normalized_minimum_nights', 'Normalized_reviews_
                                                  'listings_count_squared_x_availability_365']],
               missing = 'drop')
reg11 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'Normalized_Tourism_Proximity', 'Norma
                                                 'private_room', 'entire_homeapt',
                                                 'Normalized_availability_365', 'Normalized_host_
                                                 'Normalized_host_listings_count_squared',
                                                 'Normalized_minimum_nights', 'Normalized_reviews_
                                                  'listings_count_squared_x_availability_365']],
               missing = 'drop')
reg12 = sm.OLS(endog=adf['log_price'], exog=adf[['const', 'Normalized_Tourism_Proximity', 'Norma
                                                 'private_room', 'entire_homeapt',
                                                 'Normalized_availability_365', 'Normalized_host_
                                                 'Normalized_host_listings_count_squared',
                                                 'Normalized_minimum_nights', 'Normalized_reviews_
                                                  'listings_count_squared_x_availability_365']],
               missing = 'drop')
```

Regression 9 and 10 prove that our hypothesis that vacant housing units explains most of median neighborhood income's effet on price is true. Our final regression (12) and specification is outputted below. All the variables for regression 12 are statistically and economically significant, have t-stats over 10 and statistically significant p-values. We can note that we have an R^2 of 0.53, indicating that we have explained 53.8% of pricing variance within our test group. This result is quite good considering that from an economic standpoint there will likely be quite a bit of irrationality in airbnb pricing decisions since most host's are amateurs. This R^2 is in line with other comprehensive airbnb research into pricing as well such as the Barcelona study (Toader et al, 2020) who had R^2 of 0.488 and the Canadian airbnb study which explained between 48.8% to 68.8% of the variance in various cities (Gibbs et al, 2018). This is our final OLS preferred

specification. We can also note the adjusted R^2 to be about the same, providing some indication that our variable choice was appropriate.

The mean squared error of the regression is approximately 0.21 and root mean squared error is about 0.447. Considering that log price has an average of 4.710729 and standard deviation of about 0.7, this mse is decent and about in line with our R^2. The mse suggest the model's predictions are about 46% off from the actual price when considering smaller errors. Again this is somewhat large but not unexpected considering the variance in price and our variables.

We can write the preferred specification's regression function as such:

$$\begin{aligned}\log(\text{price}) = \beta_0 &+ \beta_1(\text{Normalized Tourism Proximity}) + \beta_2(\text{Normalized VacHUs}) \\ &+ \beta_3(\text{private room}) + \beta_4(\text{entire homeapt}) \\ &+ \beta_5(\text{Normalized availability 365}) + \beta_6(\text{Normalized\_host listings count}) \\ &+ \beta_7(\text{Normalized host listings count squared}) + \beta_8(\text{Normalized minimum\_nights}) \\ &+ \beta_9(\text{Normalized reviews per month}) + \beta_{10}(\text{listings count x availability 365}) \\ &+ \beta_{11}(\text{listings count squared x availability 365}) + \epsilon\end{aligned}$$

In [152…

```
# Stargazer table for final regression
stargazer = Stargazer([reg10.fit(),reg11.fit(),reg12.fit()])

HTML(stargazer.render_html())
```

Out[152…

	Dependent variable: log_price		
	(1)	(2)	(3)
Normalized_Median_Income	0.179***	0.030*	
	(0.012)	(0.017)	
Normalized_Tourism_Proximity	3.020***	2.825***	2.837***
	(0.045)	(0.048)	(0.047)
Normalized_VacHUs		0.253***	0.280***
		(0.021)	(0.014)
Normalized_availability_365	0.330***	0.339***	0.340***
	(0.008)	(0.008)	(0.008)
Normalized_host_listings_count	8.202***	8.221***	8.215***
	(0.646)	(0.645)	(0.645)
Normalized_host_listings_count_squared	-7.611***	-7.634***	-7.625***
	(0.662)	(0.660)	(0.660)
Normalized_minimum_nights	-0.360***	-0.369***	-0.370***
	(0.020)	(0.020)	(0.020)
Normalized_reviews_per_month	-1.144***	-1.087***	-1.078***
	(0.088)	(0.088)	(0.087)
const	3.139***	3.086***	3.085***
	(0.019)	(0.019)	(0.019)
entire_homeapt	1.201***	1.198***	1.198***
	(0.017)	(0.017)	(0.017)
listings_count_squared_x_availability_365	0.000***	0.000***	0.000***
	(0.000)	(0.000)	(0.000)
listings_count_x_availability_365	-9.450***	-9.498***	-9.484***
	(0.876)	(0.875)	(0.875)
private_room	0.443***	0.441***	0.440***
	(0.017)	(0.017)	(0.017)
Observations	35973	35973	35973
R^2	0.537	0.539	0.539
Adjusted R^2	0.537	0.538	0.538

Residual Std. Error	0.460 (df=35961)	0.459 (df=35960)	0.459 (df=35961)
F Statistic	3786.988*** (df=11; 35961)	3498.069*** (df=12; 35960)	3815.568*** (df=11; 35961)

Note: *p<0.1; **p<0.05; ***p<0.01

In [153…
```
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error

adf_clean = adf.dropna()
y_clean = adf_clean['log_price']
X_clean = adf_clean[['const', 'Normalized_Tourism_Proximity', 'Normalized_VacHUs',
                     'private_room', 'entire_homeapt',
                     'Normalized_availability_365', 'Normalized_host_listings_count',
                     'Normalized_host_listings_count_squared',
                     'Normalized_minimum_nights', 'Normalized_reviews_per_month',
                     'listings_count_x_min_nights', 'listings_count_x_availability_365',
                     'listings_count_x_monthly_reviews', 'listings_count_squared_x_monthly_revie
                     'listings_count_squared_x_availability_365']]

model_fitted = sm.OLS(endog=y_clean, exog=X_clean).fit()
predictions = model_fitted.predict(X_clean)
mse = mean_squared_error(y_clean, predictions)
print(f'Mean Squared Error: {mse}')
```

```
Mean Squared Error: 0.20996822411311292
```

We will now run regression trees to hopefully expand our economic understanding of the problem. Our objective function is the same as our previous preferred specification regression function. We have several regularization parameters to choose from which are effectively tuning knobs that help control the complexity of the model. These parameters include the maximum depth of the tree, which limits how many splits it can make; the minimum number of samples required to allow a split, which prevents the model from reacting to noise in the data; and the maximum number of leaves, which caps the granularity of the model. Adjusting these parameters affects the balance between the model's bias and variance—too much regularization can oversimplify the model (high bias), while too little can lead to a model that captures noise (high variance).

In [66]:
```
from sklearn import (
    linear_model, metrics, pipeline, model_selection
)
from sklearn import tree
```

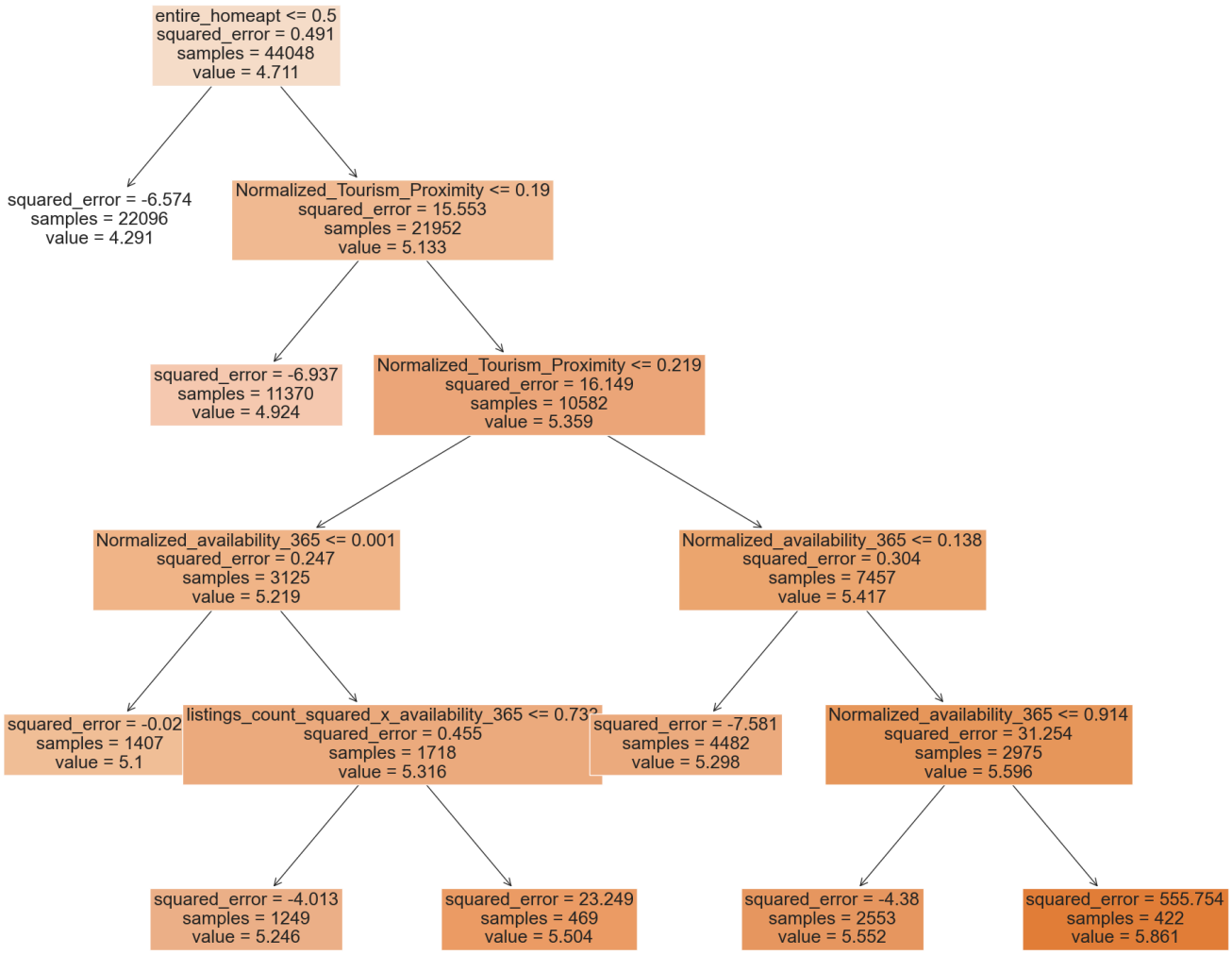
Observing our regression tree it first splits on entire_homeapt, then normalized tourism proximity for the subsequent nodes. In fact it decides other variables are so unimportant it prunes its left side branches. These results are not surprising as we have already room type and tourism proximity to be the biggest predictors of price. We get a mean squared of error of approximately 0.29, which is higher than from our linear regressions. I should note that in omitted regression tree tests using different parameters and values was unable to decrease the MSE significantly (by more than 0.01), and the tree consistently splits on entire_homeapt and tourism proximity first and second.

We can note that the tree further splits on availability 365, what this tells us is that availability 365 partitions price quite well. While we found relatively little signifiance between availbility 365 and price, this tell us that there does in fact exist some non linear relationship between the two.

In [211…

```
X = adf[['Normalized_Tourism_Proximity', 'Normalized_VacHUs',
         'private_room', 'entire_homeapt',
         'Normalized_availability_365', 'Normalized_host_listings_count',
         'Normalized_host_listings_count_squared',
         'Normalized_minimum_nights', 'Normalized_reviews_per_month',
         'listings_count_x_availability_365', 'listings_count_squared_x_availability_365']]
y = adf['log_price']
sqft_tree = tree.DecisionTreeRegressor(max_depth=5, random_state=0).fit(X,y)
y_pred_tree = sqft_tree.predict(X)
print('Mean Squared Error:', metrics.mean_squared_error(y, y_pred_tree))
sqrf_fig = plt.figure(figsize=(25,20))
sqrf_fig = tree.plot_tree(sqft_tree, feature_names=X.columns.tolist(), filled=True, fontsize=19)
```

```
Mean Squared Error: 0.2823122412381323
```



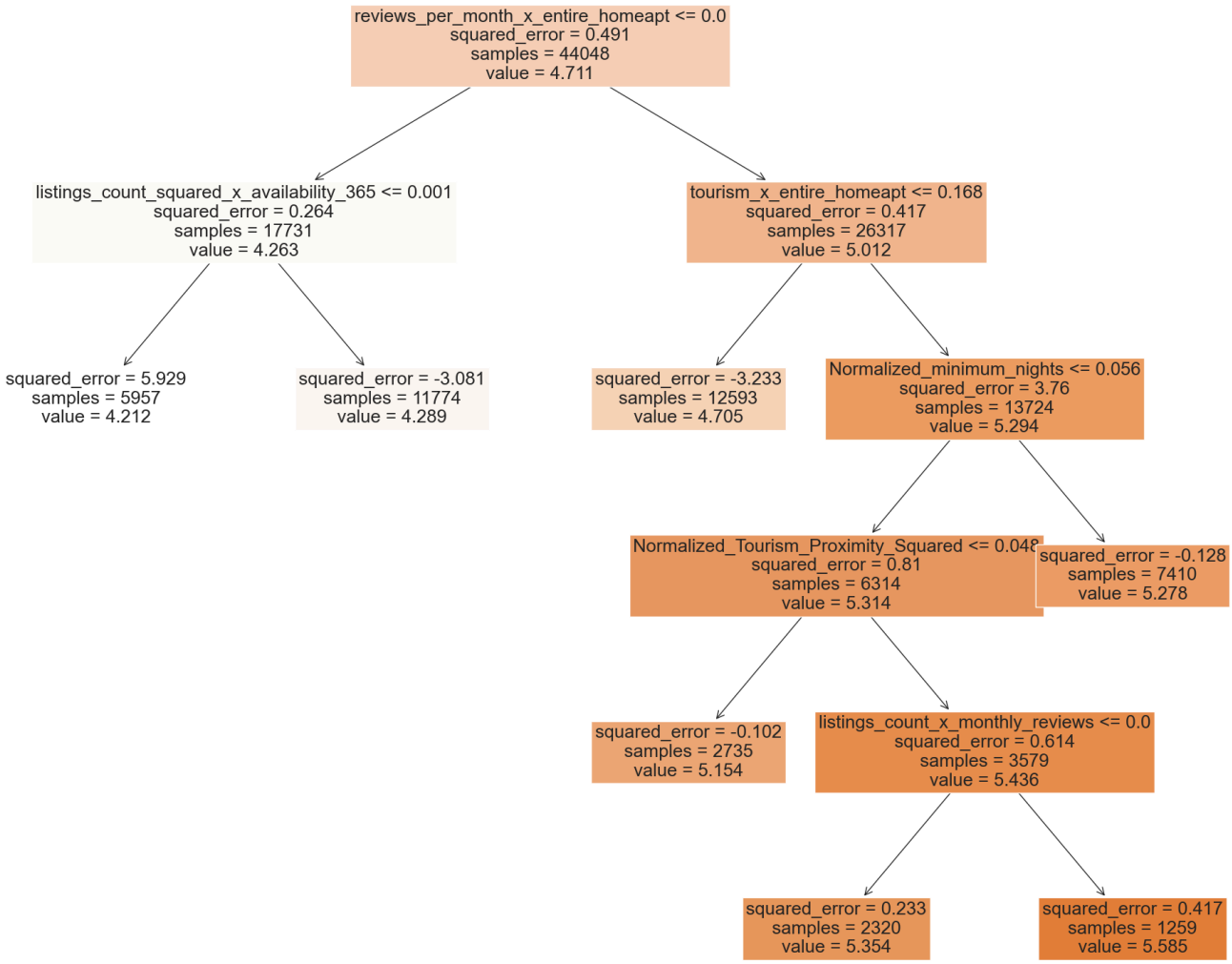
We will now run one more regression tree including all our non-price numerical variables. Note this includes normalized and non normalized variables though it shouldn't matter. The first thing we can notice is that our MSE has once again increased to about 0.3, this is even worse in the context of the log price. Omitted attempts changing various parameters did not bring the MSE down by any significant amount (0.01). We can note that the decision tree chose to split on reviews per month x entire_homeapt, however the likely explanation is that it wanted to split on entire home/apt but the interaction with reviews per month provided a slightly better partition. The explanation is likely the same for further interaction terms.

Comparing our results from our OLS regressions against the decision trees, the OLS regressions were not only statisically more significant and had less error, but were easier to interpret. In this case the OLS

regression was superior indicating that linear models was a better fit for our data.

In [209…
```
columns_to_drop = ['log_price', 'price']
X = adf.select_dtypes(include=[np.number]).drop(columns_to_drop, axis=1)

if 'log_price' in X:
    X = X.drop('log_price', axis=1)
y = adf['log_price']
sqft_tree = tree.DecisionTreeRegressor(max_depth=5, random_state=0).fit(X,y)
y_pred_tree = sqft_tree.predict(X)
print('Mean Squared Error:', metrics.mean_squared_error(y, y_pred_tree))
sqrf_fig = plt.figure(figsize=(25,20))
sqrf_fig = tree.plot_tree(sqft_tree, feature_names=X.columns.tolist(), filled=True, fontsize=19)
```

```
Mean Squared Error: 0.29982747328248377
```



We will now proceed to run random forest models to hopefully uncover additional economic explanations for our resesearch question.

In [104…
```
from sklearn.ensemble import BaggingClassifier, RandomForestClassifier, BaggingRegressor, Random
from sklearn.metrics import mean_squared_error,confusion_matrix, classification_report
```

In [160…
```
X = adf[['Normalized_Tourism_Proximity', 'Normalized_VacHUs',
         'private_room', 'entire_homeapt',
         'Normalized_availability_365', 'Normalized_host_listings_count',
         'Normalized_host_listings_count_squared',
         'Normalized_minimum_nights', 'Normalized_reviews_per_month',
```

```
         'listings_count_x_availability_365', 'listings_count_squared_x_availability_365']]

X = X.fillna(X.mean())
# define and fit
regr_RF = RandomForestRegressor(max_features=5, random_state=1).fit(X, y)

#predict
pred = regr_RF.predict(X)

#calculate MSE
mean_squared_error(y, pred)
```

Out[160… 0.031124729201748035

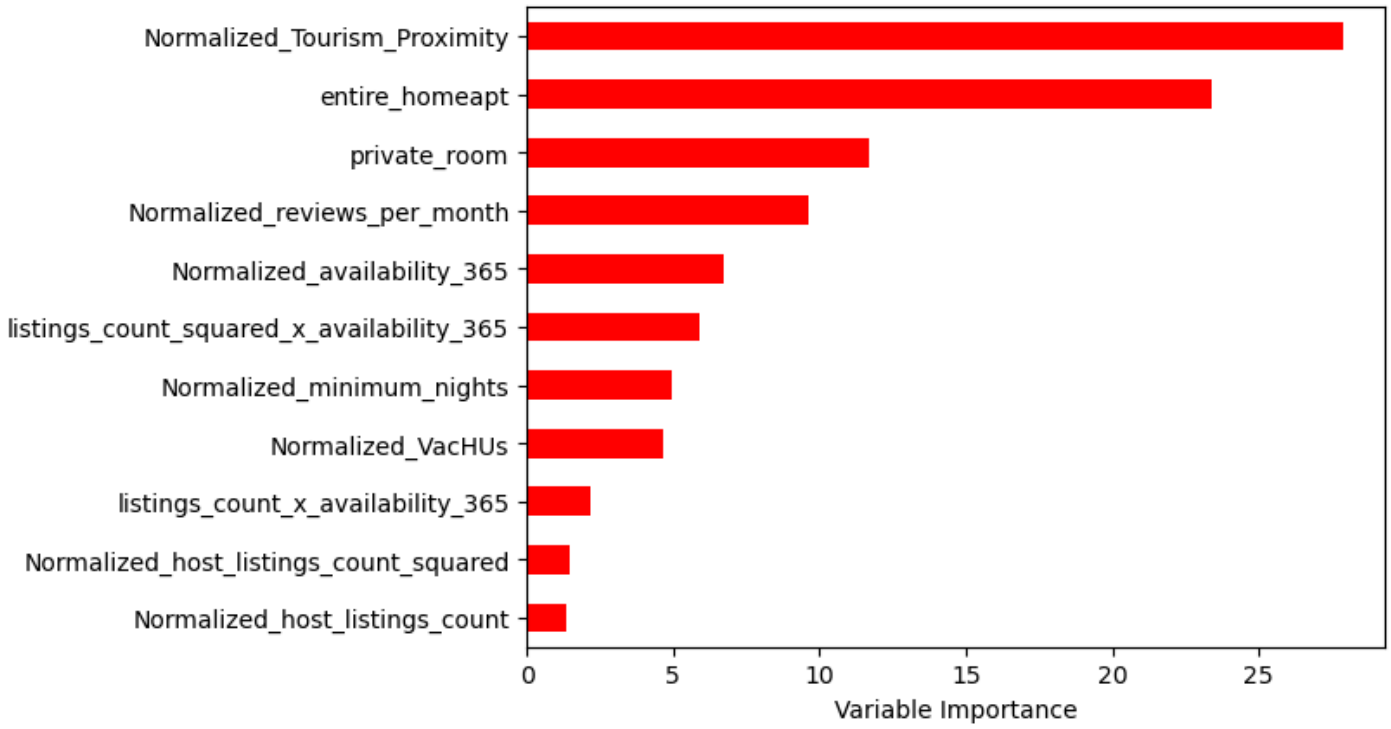
Interpreting the importance matrix we can say that tourism proximity has the strongest effect on price with about 27%, and entire home/apartments with about 23%. This confirms our results from our previous regressions. The other variables have about the same effect and relative magnitude that we expected from our regressions, the only anomaly being that reviews per month has a stronger effect on price than we had seen in our previous graphs or speculated on, though this result was shown in the regressions, particulary our comprehensive regression.

The mean squared error of the random forest model is about 0.03, this is the lowest mse we have gotten so far and smaller than we would have expected. Considering the variance of log price (mean of 4.710729, std of 0.700714), this result is quite impressive and we can interpret this very low mse economically to say that the variables we have chosen have high explanatory power over the prices in our dataset. It also suggests the result of the importance matrix to be quite accurate.

In [161…

```
plt.figure(figsize=(10, 15))
Importance = pd.DataFrame({'Importance':regr_RF.feature_importances_*100}, index=X.columns)
Importance.sort_values('Importance', axis=0, ascending=True).plot(kind='barh', color='r', )
plt.xlabel('Variable Importance')
plt.gca().legend_ = None
```

```
<Figure size 1000x1500 with 0 Axes>
```



Conclusion

The results of our study have confirmed several findings from previous research as well as uncovered some novel findings of our own. We have confirmed previous research that listings characteristics such as room type, and minimum nights have significant effects on price, notably we have confirmed the unusual finding from some research that a greater amount of active reviews for a listing is correlated with a lower listing price.

We have also found that the number of listings a host owns- a likely proxy for experience, also have a significant positive effect on price through interaction terms, and that neighborhood characteristics such as median income and vacant housing units have a positive effect on price as well. Interstingly, we noted that vacant housing units were a bigger explainer than median neighborhood income, suggesting that the forces of supply and demand are at play for airbnb's in New York City. While previous research examining location based factors on price has limited their spatial analysis to using straight line distance's to a handful of notable attraction such as beaches, city halls, and city centers. Our approach to using tourist attractions data and their associated reviews to calculate popularity and tourism proximity scores for each listing is novel and far more comprehensive and gave us our most significant result; that a listing's proximity to a larger number of popular tourist destinations is the biggest determinant of price. Additionally, our method of collecting tourist attractions could be improved upon as suggested in our data section by expanding our search terms or securing alternative forms of tourism data. We were also forced to drop listings with minimum nights over 28 days due to our limited data on them over our dummy variables, which while only dropping 4811 listings was undesirable. Future studies may wish to make use of more data or find other ways to combat numerical errors introduced by the limited sample size.

Nevertheless all our findings were statisically significant and our OLS regressions explained about 53.8% of variance with relatively little error, which is in line with similar research into Airbnb pricing determinants (Toader et al., 2020; Gibbs et al,. 2017; Deboosere et al., 2019).

The story that emerges from our findings is that the primary users of airbnb in New York City are short term tourists with low elasticities of demand and that a high portion of airbnbs are managed by economical hosts. The consistent findings that reviews per month has a negative correlation with price suggests a bidirectional relationship and/or more economical hosts seeking to take advantage of long run strategies to build reputation in an incipient market. Though our findings only weakly supported this theory, a time series causal analysis would uncover the exact reasoning behind it. Furthermore, an additional limitation of the study was lacking detailed host characteristics which could've allowed us to potentially identify economical hosts better and provided additional insight into our data. Establishing causal relationships between our different variables and price would have furthered our understanding of pricing determinants as well.

In conclusion our study was comprehensive and provided important insight to support existing research, while our findings for tourism attractions effect on price expanded the understanding of Airbnb pricing determinants. A future study may seek to determine if the effects of tourism attractions and their popularity are generalizable across different municipalities.

References

Voltes-Dorta, A., & Sánchez-Medina, A. (2020, September 11). Drivers of airbnb prices according to property/room type, season and location: A regression approach. Journal of Hospitality and Tourism

Management. https://www.sciencedirect.com/science/article/pii/S1447677020302023#bib15

Zhang, Z., Chen, R. J. C., Han, L. D., & Yang, L. (2017, September 14). Key factors affecting the price of Airbnb listings: A geographically weighted approach. MDPI. https://www.mdpi.com/2071-1050/9/9/1635

Perez-Sanchez, V. R., Serrano-Estrada, L., Marti, P., & Mora-Garcia, R.-T. (2018, December 5). The what, where, and why of airbnb price determinants. MDPI. https://www.mdpi.com/2071-1050/10/12/4596

Gutiérrez, J., García-Palomares, J. C., Romanillos, G., & Salas-Olmedo, M. H. (2017, May 12). The eruption of airbnb in tourist cities: Comparing spatial patterns of hotels and peer-to-peer accommodation in Barcelona. Tourism Management. https://www.sciencedirect.com/science/article/pii/S0261517717301036

2020 census. 2020 Census -DCP. (n.d.). https://www.nyc.gov/site/planning/planning-level/nyc-population/2020-census.page

United States Census Bureau (2019). https://data.census.gov/

Teubner, T., Hawlitschek, F., & Dann, D. (2017). Price determinants on Airbnb: How reputation pays off in the sharing economy. Journal of Self-Governance & Management Economics, 5(4), 53–80.

Wang, D., & Nicolau, J. L. (2017). Price determinants of sharing economy based accommodation rental: A study of listings from 33 cities on Airbnb.com. International Journal of Hospitality Management, 62, 120–131.

Buchanan, L., Katz, J., Washington, E., & Taylor, R. (2023, October 30). An extremely detailed map of New York City neighborhoods. The New York Times. https://www.nytimes.com/interactive/2023/upshot/extremely-detailed-nyc-neighborhood-map.html

Valentin Toader, Adina Letiţia Negruşa, Oana Ruxandra Bode & Rozalia Veronica Rus (2022) Analysis of price determinants in the case of Airbnb listings, Economic Research-Ekonomska Istraživanja, 35:1, 2493-2509, DOI: 10.1080/1331677X.2021.1962380

Kakar, V., Voelz, J., Wu, J., & Woo, J. (2017, August 25). The visible host: Does race guide airbnb rental rates in San Francisco?. Journal of Housing Economics. https://www.sciencedirect.com/science/article/pii/S1051137716301309#sec0006

Gibbs, C., Guttentag, D., Gretzel, U., Morton, J., & Goodwill, A. (2018). Pricing in the sharing economy: a hedonic pricing model applied to Airbnb listings. Journal of Travel & Tourism Marketing, 35(1), 46–56. https://doi.org/10.1080/10548408.2017.1308292

Deboosere, R., Kerrigan, D. J., Wachsmuth, D., & El-Geneidy, A. (2019). Location, location and professionalization: a multilevel hedonic analysis of Airbnb listing prices and revenue. Regional Studies, Regional Science, 6(1), 143–156. https://doi.org/10.1080/21681376.2019.1592699