Berliner Hochschule für Technik

VISUALIZE NEWYORK CITY AIRBNB DATA

TANIA SULTANA
931031
s87431@bht-berlin.de

&

AHMED DIDER RAHAT
916146
s40183@bht-berlin.de

Submitted on: 10th August 2022

Table of Contents

1. Introduction 2

2. The data set 3

3. Visualizing the data of Airbnb data sets 4

3.1 Basic statistics of the dataset: 4

3.2 Univariate Analysis: 6

3.3 Bivariate analysis: 15

4. Final Discussion 32

5. References 33

1. Introduction

Data visualization is the process to develop an understanding of row data. By using this, we can get insights that are not visible in the first place. It is one of the core processes of generating information from data. In our project, we visualize the Airbnb data for New York City. We try to make some graphical representations of the features and also demonstrate the relationship between features.

Airbnb stands for Air Bed and Breakfast. According to business model navigator “Aribnb is an American company that operates an online marketplace and hospitality service for people to lease or rent short-term lodging including holiday cottages, apartments, homestays, hostel beds, or hotel rooms, to participate in or facilitate experiences related to tourism such as walking tours, and to make reservations at restaurants. The company does not own any real estate or conduct tours; it is a broker which receives percentage service fees in conjunction with every booking. Like all hospitality services, Airbnb is an example of collaborative consumption and sharing. The company has over 4 million lodging listings in 65,000 cities and 191 countries and has facilitated over 260 million check-ins.”

2. The data set

The data set is collected from kaggle.com. According to Kaggle, "Since 2008, guests and hosts have used Airbnb to expand on traveling possibilities and present a more unique, personalized way of experiencing the world. This dataset describes the listing activity and metrics in NYC, NY for 2019. This data file includes all needed information to find out more about hosts, geographical availability, and necessary metrics to make predictions and draw conclusions".

Features of the Dataset:

1. **id:** The id assigned to each airbnb to identify them uniquely.
2. **name:** The name assigned to each airbnb.
3. **host_id:** The id assigned to each host to identify them uniquely.
4. **host_name:** The name assigned to each host.
5. **neighbourhood_group:** The 5 boroughs that New York City is divided into: Manhattan, Queens, Brooklyn, Staten Island and Bronx.
6. **neighbourhood:** The neighborhood where the airbnb is located within the boroughs.
7. **latitude:** The latitude of the location where the airbnb is situated.
8. **longitude:** The longitude of the location where the airbnb is situated.
9. **room_type:** The type of airbnb which is divided into two: Entire home/Apartment, Private room and Shared Room.
10. **price:** The rent of the airbnb per night.
11. **minimum_nights:** The minimum number of nights the airbnb can be rented for.
12. **number_of_reviews:** Total number of reviews posted by customers.
13. **last_review:** Date of the last review posted by a customer.
14. **reviews_per_month:** Monthly total of reviews posted by customers.
15. **calculated_host_listings_count:** Number of total listings by a host.
16. **availability_365:** Yearly number of days the airbnb is available for rent.

3. Visualizing the data of Airbnb data sets

3.1 Basic statistics of the dataset: In this portion we load the data set and see some basic statistics of different features.

3.1.1 Data load in Pandas: At first we load the data from csv file to pandas data-frame.

```
# load dataset
df = pd.read_csv('../data/AB_NYC_2019.csv')
```

Figure 01- data loading

3.1.2 Observe the data type: In this section we analyze the data type of different features.

```
# see the data types of each columns
print(df.dtypes)
```

```
id                                  int64
name                               object
host_id                             int64
host_name                          object
neighbourhood_group                object
neighbourhood                      object
latitude                          float64
longitude                         float64
room_type                          object
price                               int64
minimum_nights                      int64
number_of_reviews                   int64
last_review                        object
reviews_per_month                 float64
calculated_host_listings_count      int64
availability_365                    int64
dtype: object
```

Figure 02- get the data type for each columns

3.1.3 Statistical Analysis of Numerical Data: Here, we have analyzed the statistical properties (count, mean, standard deviation, min, max, and percentile) of the numeric data.

```
# analysise the numeric features of the dataset
display(df.describe().transpose())
```

	count	mean	std	min	25%	50%	75%	max
id	48895.0	1.901714e+07	1.098311e+07	2539.00000	9.471945e+06	1.967728e+07	2.915218e+07	3.648724e+07
host_id	48895.0	6.762001e+07	7.861097e+07	2438.00000	7.822033e+06	3.079382e+07	1.074344e+08	2.743213e+08
latitude	48895.0	4.072895e+01	5.453008e-02	40.49979	4.069010e+01	4.072307e+01	4.076311e+01	4.091306e+01
longitude	48895.0	-7.395217e+01	4.615674e-02	-74.24442	-7.398307e+01	-7.395568e+01	-7.393627e+01	-7.371299e+01
price	48895.0	1.527207e+02	2.401542e+02	0.00000	6.900000e+01	1.060000e+02	1.750000e+02	1.000000e+04
minimum_nights	48895.0	7.029962e+00	2.051055e+01	1.00000	1.000000e+00	3.000000e+00	5.000000e+00	1.250000e+03
number_of_reviews	48895.0	2.327447e+01	4.455058e+01	0.00000	1.000000e+00	5.000000e+00	2.400000e+01	6.290000e+02
reviews_per_month	38843.0	1.373221e+00	1.680442e+00	0.01000	1.900000e-01	7.200000e-01	2.020000e+00	5.850000e+01
calculated_host_listings_count	48895.0	7.143982e+00	3.295252e+01	1.00000	1.000000e+00	1.000000e+00	2.000000e+00	3.270000e+02
availability_365	48895.0	1.127813e+02	1.316223e+02	0.00000	0.000000e+00	4.500000e+01	2.270000e+02	3.650000e+02

Figure 03- Analysis Numeric features

3.1.4 Null Value Analysis: At this stage, we have counted exactly how many null values are in each column. At the end of the count, we see that the highest number of null values are in the review-related features. There are ten thousand and fifty-two (10052) null values in each of this feature. These features have the highest number of null values as customers do not give reviews.

```
# null value count
print(df.isnull().sum())
```

```
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
```

Figure 04- Null value analysis

3.2 Univariate Analysis: In this portion we analyze the univariate features of the dataset. For these type of analysis, at first we get the numbers and then plot them in different figures.

3.2.1 Count neighborhood group: We calculated the number of Airbnb in each group. After getting the count we plot the data into a bar-plot.

```
nh_group = df.groupby(['neighbourhood_group'])['neighbourhood_group'].count().reset_index(name='counts')

groups = list(nh_group.neighbourhood_group)
counts = list(nh_group.counts)

nh_group
```

	neighbourhood_group	counts
0	Bronx	1091
1	Brooklyn	20104
2	Manhattan	21661
3	Queens	5666
4	Staten Island	373

```
def addlabels(x, y, z):
    for i in range(len(x)):
        plt.text(i, y[i], z[i], ha = 'center')

ax = nh_group.plot.bar(x='neighbourhood_group', y='counts', rot=0, \
                       title='Counts of Airbnb in different neighbourhood group')

cnt_percentage = [f'{str(round(d*100/sum(counts),1))}%' for d in counts]
addlabels(groups, counts, cnt_percentage)

plt.savefig('../report/fig/plot_1.png',  bbox_inches='tight')
plt.show()
```

Figure 05- Code and output of neighborhood group data

Analysis: In the bar chart, we can clearly see that Manhattan has the highest number of Airbnb with the percentage of 44.3%. Staten Island has the lowest number of Airbnb, and the count is only 373 (0.6%). The Bronx has about 2.2% of Airbnb which is 2nd lowest in NYC. And lastly, 41.1% and 11.6% of Airbnb are situated Brooklyn and Queens respectively.

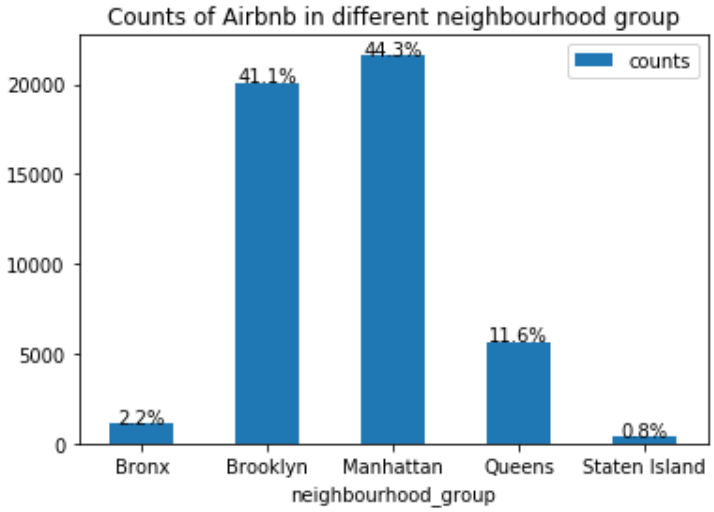


Figure 06- Analysis of the number of Airbnb within different neighborhood group

3.2.2 **Analyze the counts of Airbnb in different neighborhood:** Here, we analyze the volume of Airbnb in each neighborhood. For performing the analysis, we did some interrelated steps.

Step 1: We get the number of the unique neighborhood in NYC and we got 221 as our output.

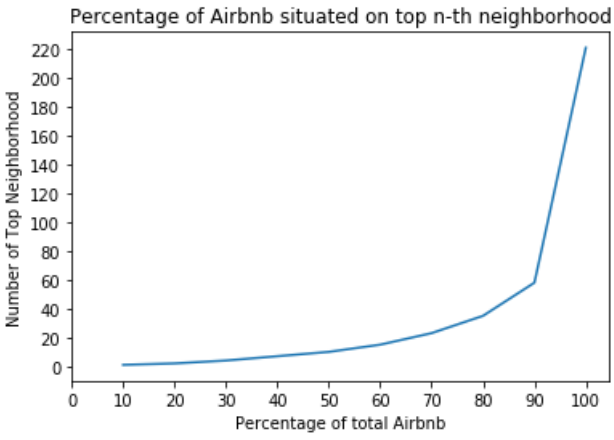
```
#Neighbourhood analysis

mpn_df = pd.DataFrame(df.neighbourhood.value_counts())
mpn_df.reset_index(inplace=True)

mpn_df.rename(columns={'index':'Neighbourhood', 'neighbourhood':'P_Count'}, inplace=True)

print(f'Total Number of Neighborhood is: {len(mpn_df)}\n')
```

```
Total Number of Neighborhood is: 221
```

Figure 07- Code and output of unique neighborhood counting

Step 2: Then we start investigating how much percentage of Airbnb situated in the top-nth neighborhood. So, we sort them based on their Airbnb counts and calculate their percentage. Later we plot a diagram that contains the percentage of total Airbnbs on the X-axis and the number of neighborhoods on the Y-axis. From the graph, we can conclude that the top 20 neighborhoods hold almost 60% of total Airbnbs. Lastly, we saw that 65.87% of total Airbnb situated on only top 20 neighborhood.

```
cm_percnt = []
sum_percent = 0
for index, row in mpn_df.iterrows():
    cnt = row['P_Count']

    if index == len(mpn_df)-1:
        cm_percnt.append(100.00)
        break

    percent = cnt * 100 / sum(mpn_df.P_Count)
    sum_percent += percent

    cm_percnt.append(sum_percent)

_high = list(range(10,101,10))

cnt_per = [len([d for d in cm_percnt if d<=h]) for h in _high]

plt.plot(_high, cnt_per)
plt.xticks(range(0, 101, 10))
plt.yticks(range(0, max(cnt_per)+10, 20))
plt.xlabel('Percentage of total Airbnb')
plt.ylabel('Number of Top Neighborhood')

plt.title(f'Percentage of Airbnb situated on top n-th neighborhood')
plt.savefig('../report/fig/plot_2.png',  bbox_inches='tight')

plt.show()

per_20 = sum(list(mpn_df.P_Count)[:20]) * 100 / sum(mpn_df.P_Count)
print(f'\nTop 20 Neighborhood contains {round(per_20, 2)}% of total Airbnb')
```




```
Top 20 Neighborhood contains 65.87% of total Airbnb
```

Figure 08- Code and Visualization of Top nth Neighborhood and their percentage

Step 3: Then we only focus on the top 20 neighborhoods. In this step, we show them in a table and make a bar plot with their percentages. **From the count and graph,** we make some decisions. Firstly, Williamsburg and Bedford-Stuyvesant have the maximum number of Airbnbs whose volumes are respectively 8.0% and 7.6%. Secondly, Clinton Hill has a minimum ratio of 1.2% in this top-20 list. Finally, all the other neighborhoods have roughly 5% to 1.5% Airbnb in their location.

```
# top 20 neighbor
top_n_count = 20
mpn_df.head(top_n_count)
```

	Neighbourhood	P_Count
0	Williamsburg	3920
1	Bedford-Stuyvesant	3714
2	Harlem	2658
3	Bushwick	2465
4	Upper West Side	1971
5	Hell's Kitchen	1958
6	East Village	1853
7	Upper East Side	1798
8	Crown Heights	1564
9	Midtown	1545
10	East Harlem	1117
11	Greenpoint	1115
12	Chelsea	1113
13	Lower East Side	911
14	Astoria	900
15	Washington Heights	899
16	West Village	768
17	Financial District	744
18	Flatbush	621
19	Clinton Hill	572

```
#Plot neighbourhood
def addlabels(x, y, z):
    for i in range(len(x)):
        plt.text(i, y[i], z[i], ha = 'center')


groups = list(mpn_df.head(top_n_count).Neighbourhood)
counts = list(mpn_df.head(top_n_count).P_Count)

ax = mpn_df.head(top_n_count).plot.bar(x='Neighbourhood', y='P_Count', rot=75, \
                title='Counts of Airbnb in different neighbourhood', figsize=(12,8))

ax.set_xticklabels(groups, fontsize = 10)

cnt_percentage = [f'{str(round(d*100/sum(list(mpn_df.P_Count)),1))}%' for d in counts]
addlabels(groups, counts, cnt_percentage)

plt.savefig('../report/fig/plot_3.png',  bbox_inches='tight')
plt.show()
```

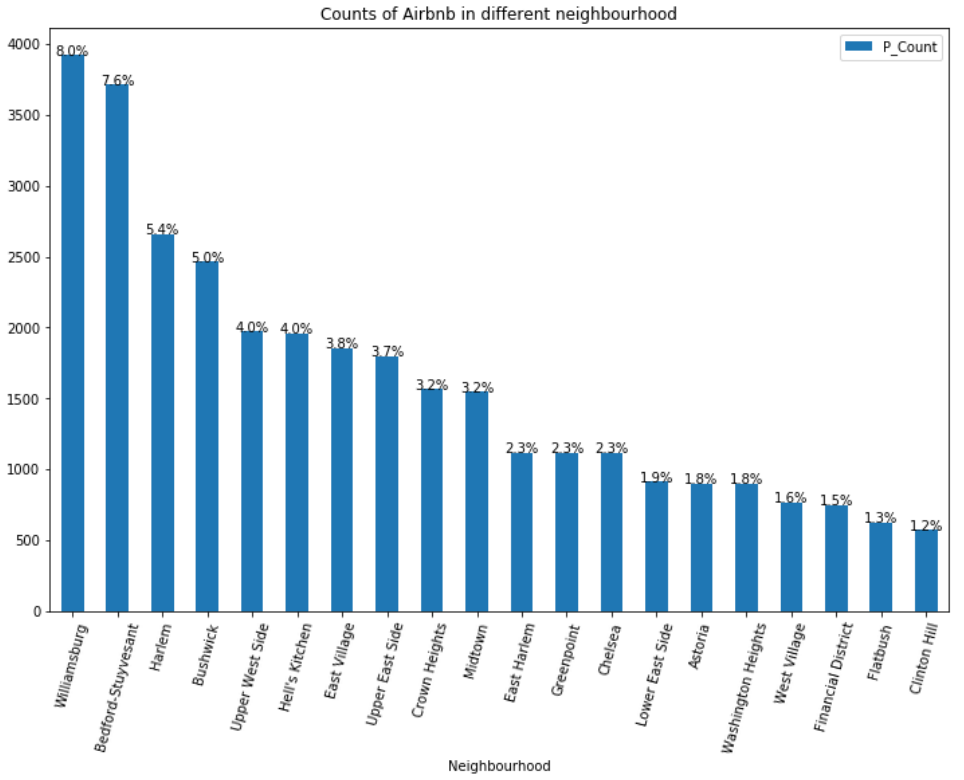


Figure 09- Implementation and analysis of top 20 neighborhoods based on their Airbnb counts

Step 4: Here, we draw the top 20 neighborhoods on the NYC map. We also set a color scale to see the volume of the Airbnb count. **From the map,** we extracted some pieces of information. Firstly, all the 20 neighborhoods are situated in pretty much the same geographical region. Secondly, the top 2 neighborhoods named Williamsburg and Bedford-Stuyvesant are next to each other. Lastly, compared with the whole area of the city, most of the (65.87%) Airbnbs are densely located in a small area (only 20 neighborhoods out of 221).

```
def get_count(_neighborhood):
    tmp = list(mpn_df.P_Count[mpn_df.Neighbourhood==_neighborhood])
    return tmp[0]

nhc_df = df[['latitude', 'longitude', 'neighbourhood']]
nhc_df['P_Count'] = nhc_df['neighbourhood'].apply(get_count)
```

```
import urllib
plt.figure(figsize=(12,8))

url = 'https://upload.wikimedia.org/wikipedia/commons/e/ec/Neighbourhoods_New_York_City_Map.PNG'
nyc_img=plt.imread(urllib.request.urlopen(url))

plt.imshow(nyc_img, zorder=0, extent=[-74.258, -73.7, 40.49, 40.92])
ax = plt.gca()

cnt_list = list(mpn_df.P_Count)
sub_df = nhc_df[nhc_df.P_Count > cnt_list[20]]

sub_df.plot(kind='scatter', x='longitude', y='latitude', label='P_Count',
            c='P_Count', ax=ax, cmap=plt.get_cmap('jet'), colorbar=True, alpha=0.4, zorder=5)

plt.savefig('../report/fig/plot_4.png',  bbox_inches='tight')
plt.show()
```

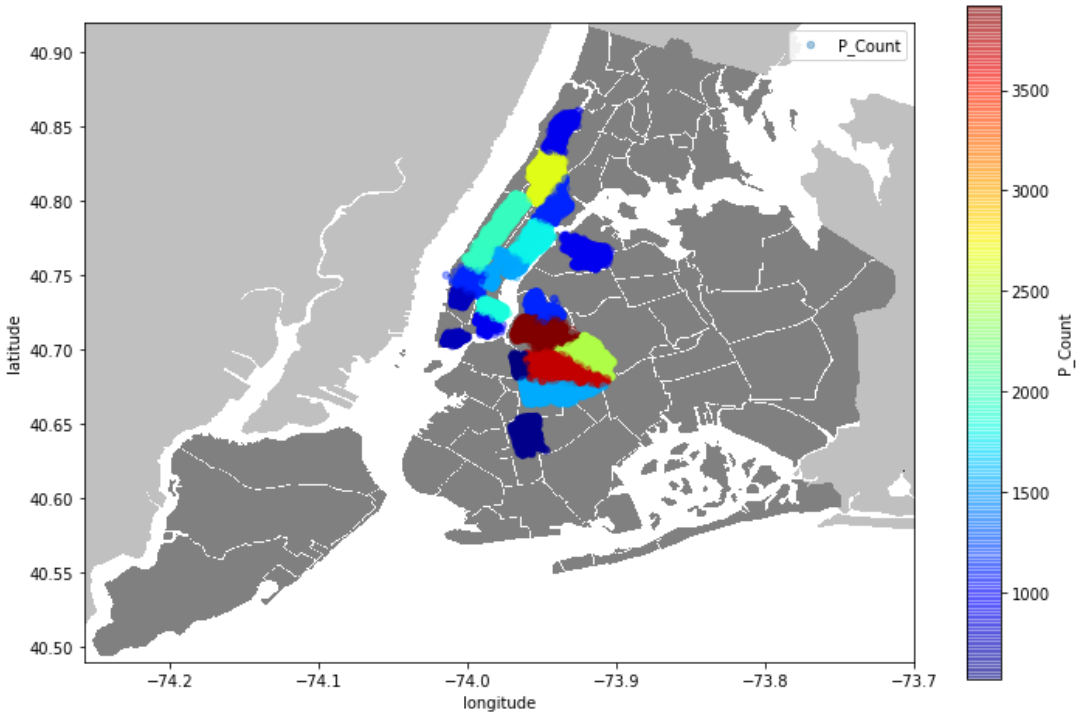


Figure 10- Extract information from NYC map

3.2.3 Count of Apartment type: There are only three types of Airbnb in the dataset. So we fetched the count of each group and plot them in a pie-chart.

```
arbnb_type = df.groupby(['room_type'])['room_type'].count().reset_index(name='counts')
types = list(arbnb_type.room_type)
counts = list(arbnb_type.counts)

arbnb_type
```

	room_type	counts
0	Entire home/apt	25409
1	Private room	22326
2	Shared room	1160

```
explode = (0.05, 0.05, 0.1)

fig1, ax1 = plt.subplots()
ax1.pie(counts, explode=explode, labels=types, autopct='%1.1f%%', shadow=True, \
        colors=['#FEAE65', '#7CDDDD', '#AADEA7'])
ax1.axis('equal')

plt.title('Percentage of Different \ntypes of Airbnb\n')
plt.savefig('../report/fig/plot_3.png',  bbox_inches='tight')

plt.show()
```

Figure 11- Implementation of Airbnb type with counts

Analysis: The pie chart shows that the number of "entire homes/apartments" is higher than the other two types, and the percentage is about 52.0%. The number of "shared room" has minimum counts with 2.4% of apartments. The "private room" type Airbnb has 45.7% of the apartments.

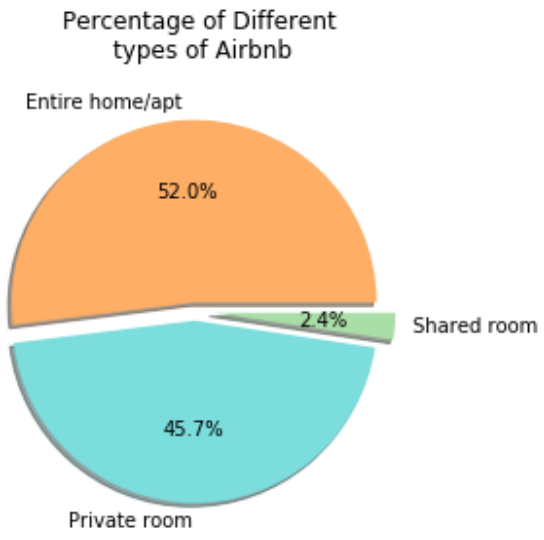


Figure 12- Pie-chart of apartment type

3.2.4 Counts of minimum night stay feature: To analyze this figure we used a box-plot. And implementation, graph, and interpretations are given bellow:

```
mn_df = df.minimum_nights.fillna(0)

print(f'Minimum: {mn_df.min()} and Maximum: {mn_df.max()}')

plt.boxplot(mn_df, vert=False)
plt.title('Minimum night stay in Airbnb dataset')
plt.savefig('../report/fig/plot_4.png',  bbox_inches='tight')
plt.show()
```

```
Minimum: 1 and Maximum: 1250
```

Figure 13- Implementation of minimum night stay feature

Analysis: The minimum and maximum night's stay in the dataset are 1 and 1250 respectively. The numbers between 1-100 are more densely populated in the graph, which means a huge percentage of the apartment has a minimum night stay value in this range. There also have some values from 100-300 (approximately). The rest of the numbers are some sort of outliers.

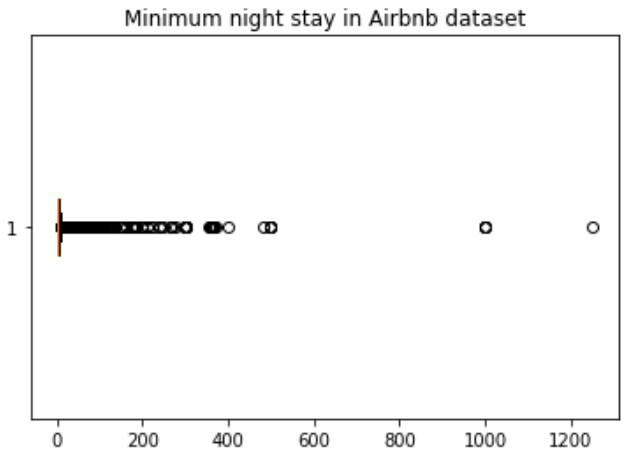


Figure 14- Analysis of minimum night stay feature

3.2.5 Analyze the calculated host listing feature: This feature represents the number of apartments listed by a host. We fetched the data and created a box plot of this feature.

```
chl_df = df.calculated_host_listings_count.fillna(0)
plt.boxplot(chl_df, vert=False)
plt.title('Host listing count of Airbnb data')

print(f'Minimum: {chl_df.min()} and Maximum: {chl_df.max()}')

plt.savefig('../report/fig/plot_5.png',  bbox_inches='tight')
plt.show()
```

```
Minimum: 1 and Maximum: 327
```

Figure 15- Implement host listing feature

Analysis: According to the figure, we can see that a single host listed 327 apartments, which is pretty high compared with other hosts. Most of the hosts listed 1-50 apartments as the graph has much dense population is this section.

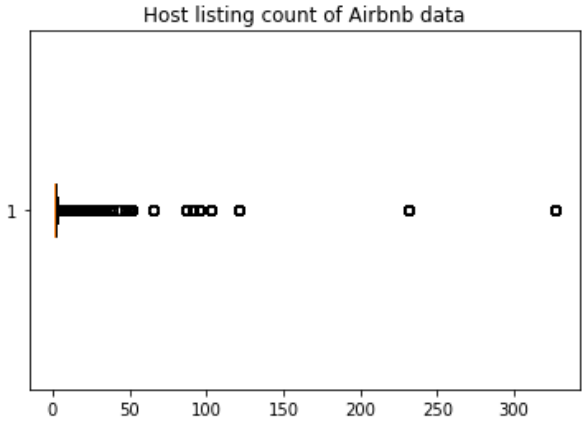


Figure 16- Analysis of host listing in the data set

3.2.6 Get top most listed host: In this part, we have calculated which hosts have been listed with the most apartments. And we've listed the top ten hosts (host IDs) that have been rented more frequently.

```
# get top-most host
top_host = df.host_id.value_counts().head(10)

top_host_df = pd.DataFrame(top_host)
top_host_df.reset_index(inplace=True)
top_host_df.rename(columns={'index':'Host_ID', 'host_id':'P_Count'}, inplace=True)
top_host_df
```

	Host_ID	P_Count
0	219517861	327
1	107434423	232
2	30283594	121
3	137358866	103
4	12243051	96
5	16098958	96
6	61391963	91
7	22541573	87
8	200380610	65
9	7503643	52

```
top_host_plot=sns.barplot(x="Host_ID", y="P_Count", data=top_host_df, palette='Blues_d')
top_host_plot.set_title('Hosts with the most listings Airbnb in NYC')
top_host_plot.set_ylabel('Count of listings')
top_host_plot.set_xlabel('Host IDs')
top_host_plot.set_xticklabels(top_host_plot.get_xticklabels(), rotation=45)

plt.savefig('../report/fig/plot_6.png',  bbox_inches='tight')
plt.show()
```

Figure 17- Code implementation of most top host

Analysis: Looking at the drawn bar chart, it can be seen that this host id 219517861 has been used the most, that is, more than 300 times. Host 7503643 has been used a little more than about fifty times, and this host has been used the least number of times in the top 10 hosts.

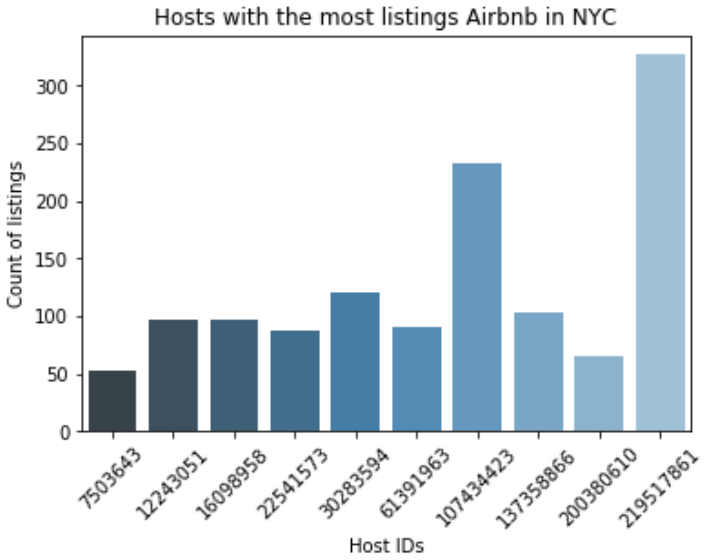


Figure 18- Bar chat and insight of host id feature

3.2.7 Analysis of the 'number of reviews' feature: To analyze this column, we fill all the empty cells with zero. Then we plot all the data into a box plot.

```
review_df = df.number_of_reviews.fillna(0)
plt.boxplot(review_df, vert=False)
plt.title('Number of review in the Airbnb dataset')

print(f'Minimum: {review_df.min()} and Maximum: {review_df.max()}')

plt.savefig('../report/fig/plot_7.png',  bbox_inches='tight')
plt.show()
```

```
Minimum: 0 and Maximum: 629
```

Figure 19- Code implementation number of reviews

Analysis: It is clear from the figure that the most number of reviews of the dataset is distributed into 100-400. Also, some apartments have more than 400 reviews.

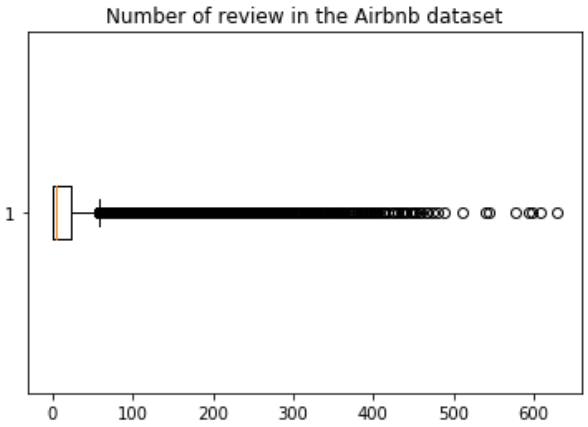


Figure 20- Box-plot of 'number of reviews' and the Insight of the graph

3.2.8 Analysis the price of Airbnb: To analyze the price, we draw two different type plot i.e. the box-plot and scatter plot.

```
df.price.plot(kind='box', vert=False, figsize=(8,5))

plt.title('Boxplot of the price of Airbnb')
plt.xticks(list(range(0, int(max(df.price)*1.1), 1000)))

plt.savefig('../report/fig/plot_8.png',  bbox_inches='tight')
plt.show()
```

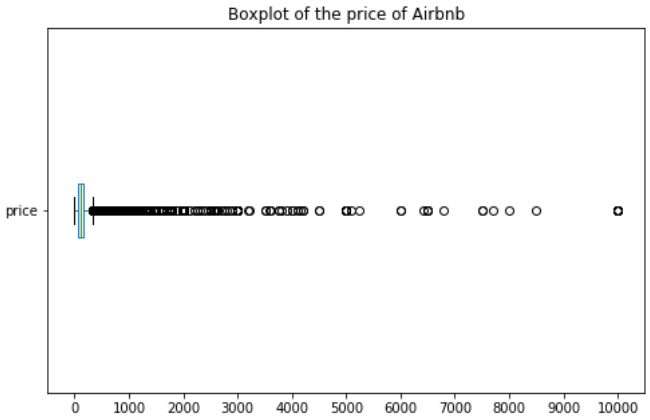


Figure 21- Implementation and Box-plot of feature price

```
plt.scatter(df.price.index, df.price)
plt.title('Scatter plot of the price of Airbnb')
plt.xlabel('index of Airbnb')
plt.ylabel('price')
plt.yticks(list(range(0, int(max(df.price)*1.1), 1000)))

plt.savefig('../report/fig/plot_9.png',  bbox_inches='tight')
plt.show()
```

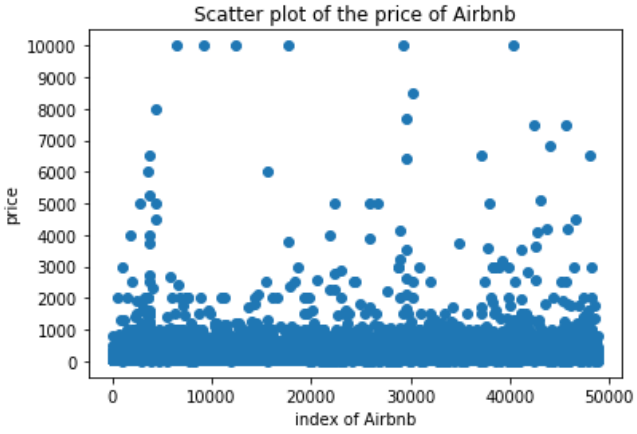


Figure 22- Implementation and Scatter-plot of feature price

Analysis: To analyze the box-plot and scatter-plot we can decide that, most of rent of the apartment are in between 100-1000. Also there are some costly apartment are visible in the graph.

3.3 Bivariate analysis: In this portion we do bivariate features analysis.

3.3.1 Drawing Pair plot: To analyze the numeric features we make a new dataframe. Then we plot the data into a pair plot.

```
p_plot = sns.pairplot(cor_df, kind='scatter', )
p_plot.fig.suptitle('Pair plot of all neumaric features', y=1.05)
plt.savefig('../report/fig/plot_10.png')
plt.show()
```

Figure 23- Implementation of pair plot

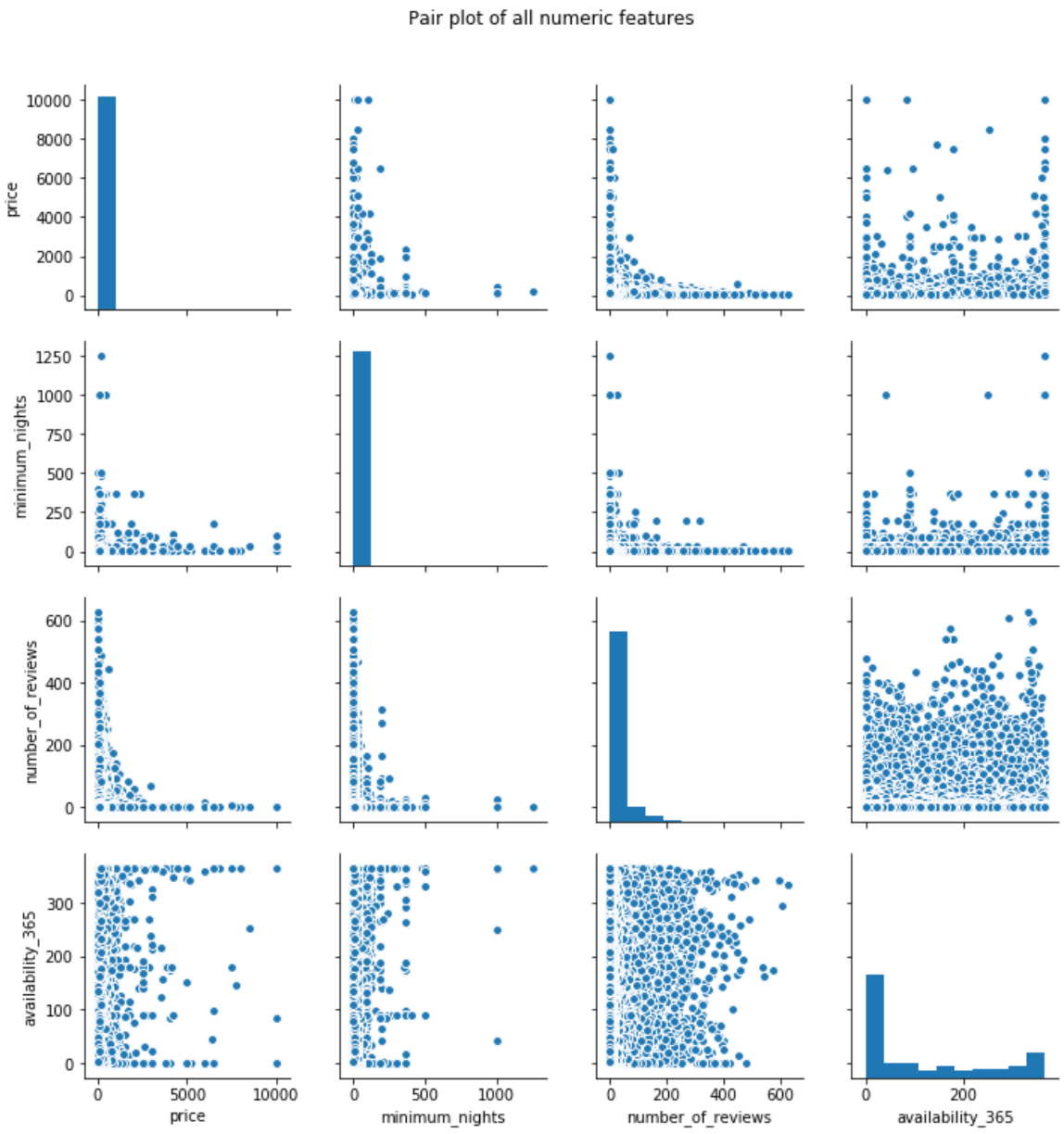


Figure 24- Pair plot of numeric feature

Analysis: From the above graph, we can see a correlation plot of the numerical variables of our data set. A Correlation Plot represents the strength of linear relation among the numeric variables. We assumed at least some linear relationship between the numeric variables initially (At least the price and the popularity of the housing). Here, the number_of_reviews variable is the way to measure the popularity of any Airbnb. Though there is no direct linear relationship between the variables, we observed some lightweight relationships for some of them.

Firstly, the relationship between price and the number of reviews. There is an exponential relationship between them where the number of reviews increases as the price decrease, which means the costly Airbnbs are less acceptable in NYC.

Secondly, we notice that a similar kind of relationship exists in minimum nights and number of reviews. The Airbnbs that have lower minimum night stay are most popular in NYC.

Lastly, we also observed a similar trend in the relationship between price and minimum night stay, with some slight exceptions. Here, most of the expensive Airbnbs have a high value of minimum night stay.

3.3.2 Correlation matrix of the numeric data: In these section we calculate the correlation matrix and also draw them in a heat-map.

```
#calculate correlation matrix
corr_matrix = cor_df.corr()
corr_matrix
```

	price	minimum_nights	number_of_reviews	availability_365
price	1.000000	0.042799	-0.047954	0.081829
minimum_nights	0.042799	1.000000	-0.080116	0.144303
number_of_reviews	-0.047954	-0.080116	1.000000	0.172028
availability_365	0.081829	0.144303	0.172028	1.000000

```
# plot correlation matrix
plt.figure(figsize=(9, 9))

vmax = (sorted(list(set(np.reshape(corr_matrix.values, \
        (1, len(corr_matrix)*len(corr_matrix)))[0])))[-2]+0.05

vmin = min(corr_matrix.min())

corr_hm = sns.heatmap(corr_matrix, annot=True, vmin=vmin, vmax=vmax, cmap="YlGnBu")

corr_hm.set(xlabel = 'Airbnb NYC features', ylabel = 'Airbnb NYC features', \
             title = "Correlation matrix of Airbnb dataset\n")

corr_hm.set_xticklabels(corr_matrix.index, rotation=45)

plt.savefig('../report/fig/plot_15.png',  bbox_inches='tight')
plt.show()
```

Figure 25- Implementation of correlation matrix

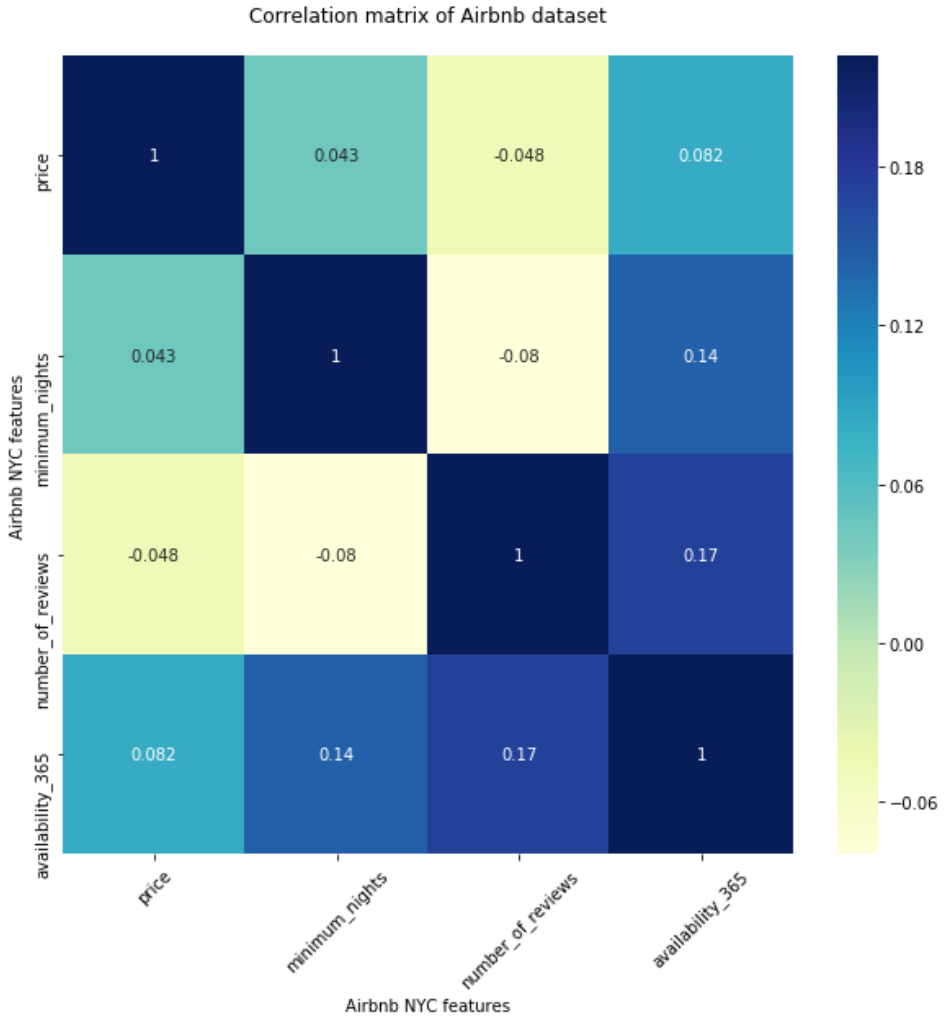


Figure 26- Heat map of correlation matrix

Analysis: From the plotted correlation figure, we clearly state that the correlation values are too small to make a linear relationship. The bluish-colored values represent a positive correlation, and the yellowish-colored values represent negative correlations. The highest positive correlation value is 0.17, between the number of reviews and availability_365. The highest negative correlation between the two features is -0.08 (minimum night stay and number of reviews).

3.3.3 Price VS Neighborhood group: In this portion, we have calculated the mean and median prices for each neighborhood group in NYC. We converted this stat into a bar chart.

```
nhg_df = df[['neighbourhood_group', 'price']]
nhg_mean = nhg_df.groupby('neighbourhood_group').mean()
nhg_median = nhg_df.groupby('neighbourhood_group').median()

nhg_mean.rename(columns={'price':'Mean Price'}, inplace=True)
nhg_median.rename(columns={'price':'Median Price'}, inplace=True)

nhg_mean_meadian = nhg_mean.join(nhg_median)
nhg_mean_meadian.insert(loc=0, column='Neighbour Group', value=list(nhg_mean_meadian.index))
nhg_mean_meadian.reset_index(inplace=True, drop='index')
```

```
labels = list(nhg_mean_meadian['Neighbour Group'])

_means = list(nhg_mean_meadian['Mean Price'])
_medians = list(nhg_mean_meadian['Median Price'])

width = 0.35

fig, ax = plt.subplots(figsize=(10,6))

x = np.arange(len(nhg_mean_meadian))

ax.set_ylabel('Airbnb price (per night)',  fontsize = 12)
ax.set_title('Airbnb price with Neighbor group', fontsize = 16)
ax.set_xlabel('Neighbor group', fontsize = 12)

ax.set_xticks(np.arange(len(labels)))
ax.set_xticklabels(labels, rotation = 0, fontsize = 11)

rects1 = ax.bar(x - width/2, _means, width, label='Mean')
rects2 = ax.bar(x + width/2, _medians, width, label='Median')

ax.legend(bbox_to_anchor=(1, 1),fancybox=True, shadow=True, ncol=1)

def addlabels(x, m, w):
    for i in range(len(x)):
        plt.text(i, m[i], f'{str(round(m[i], 1))}', ha = 'right', fontsize = 10)
        plt.text(i, w[i], f'{str(round(w[i], 1))}', ha = 'left', fontsize = 10)

addlabels(labels, _means, _medians)

plt.show()
```

Figure 27- Implementation of the relationship between price and neighborhood group

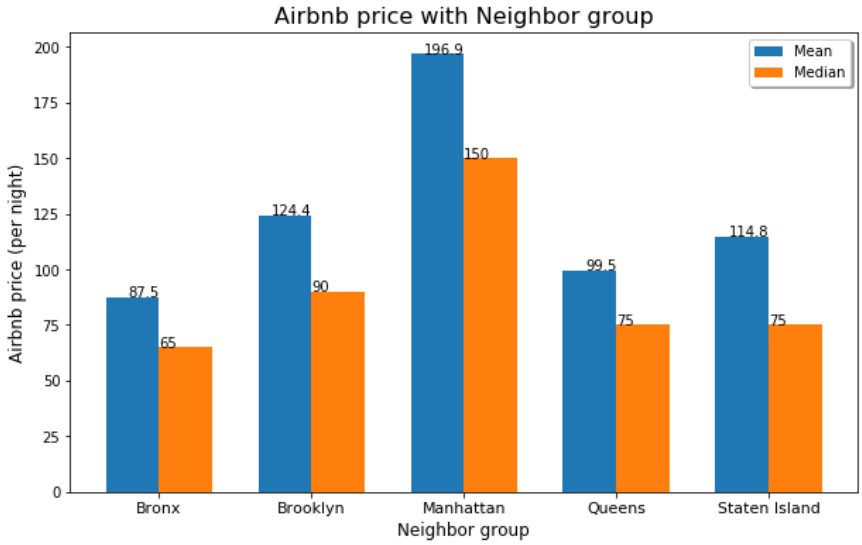


Figure 28- Bar plot of the relationship between price and neighborhood group

Analysis: The bar chart shows that both the mean and median prices of the Manhattan group's Airbnb are higher than the other groups. The mean price of this group is around 200, and the median price is approximately 150. The lowest cost Airbnb is available in the Bronx. The other three groups that have Airbnbs have an almost similar price. In all the cases, the mean price is much higher than the median price because of some extreme values.

3.3.4 Price VS Neighborhood: At this stage, we try to pick the top-10 neighborhoods based on their price. For calculating the price, we calculate the mean and median value. As a result, we got two different sets with some common neighborhoods.

```
# most costly neighbourhood
nh_df = df[['neighbourhood', 'price']]

nh_df = nh_df.groupby('neighbourhood').describe().transpose()

nh_df = nh_df.reset_index(level=[0,1])

del nh_df['level_0']

nh_df.head()

nh_df.rename(columns={'level_1':'Neighbourhood'}, inplace=True)

nh_df = nh_df.transpose()
nh_df.columns = nh_df.iloc[0]
nh_df = nh_df[1:]

nh_df = nh_df.rename_axis(None)
neighborhoods = list(nh_df.index)
nh_df = nh_df.reset_index(drop=True)
nh_df = nh_df.rename_axis(None, axis="columns")

nh_df.insert(0, "neighborhood", neighborhoods)

nh_df.head()
```

```
nh_df["mean"] = pd.to_numeric(nh_df["mean"])

mcn_df1 = nh_df[['neighborhood', 'count', 'mean', '50%', \
                 'min', 'max', 'std']].nlargest(10, 'mean')
mcn_df1 = mcn_df1.fillna(0)

mcn_df1.reset_index(drop=True, inplace=True)

mcn_df1
```

	neighborhood	count	mean	50%	min	max	std
0	Fort Wadsworth	1.0	800.000000	800.0	800.0	800.0	0.000000
1	Woodrow	1.0	700.000000	700.0	700.0	700.0	0.000000
2	Tribeca	177.0	490.638418	295.0	60.0	8500.0	856.341720
3	Sea Gate	7.0	487.857143	125.0	71.0	1485.0	626.900159
4	Riverdale	11.0	442.090909	150.0	49.0	2500.0	724.395535
5	Prince's Bay	4.0	409.500000	151.5	85.0	1250.0	561.875728
6	Battery Park City	70.0	367.557143	195.0	55.0	7500.0	974.671043
7	Flatiron District	80.0	341.925000	225.0	65.0	2000.0	345.657831
8	Randall Manor	19.0	336.000000	79.0	13.0	5000.0	1130.121282
9	NoHo	78.0	295.717949	250.0	75.0	1795.0	218.199593

```
nh_df["50%"] = pd.to_numeric(nh_df["50%"])

mcn_df2 = nh_df[['neighborhood', 'count', 'mean', '50%', \
                 'min', 'max', 'std']].nlargest(10, '50%')
mcn_df2 = mcn_df2.fillna(0)

mcn_df2.reset_index(drop=True, inplace=True)

mcn_df2
```

	neighborhood	count	mean	50%	min	max	std
0	Fort Wadsworth	1.0	800.000000	800.0	800.0	800.0	0.000000
1	Woodrow	1.0	700.000000	700.0	700.0	700.0	0.000000
2	Tribeca	177.0	490.638418	295.0	60.0	8500.0	856.341720
3	Neponsit	3.0	274.666667	274.0	200.0	350.0	75.002222
4	NoHo	78.0	295.717949	250.0	75.0	1795.0	218.199593
5	Willowbrook	1.0	249.000000	249.0	249.0	249.0	0.000000
6	Flatiron District	80.0	341.925000	225.0	65.0	2000.0	345.657831
7	Midtown	1545.0	282.719094	210.0	30.0	5100.0	255.327718
8	Financial District	744.0	225.490591	200.0	12.0	3000.0	168.425509
9	West Village	768.0	267.682292	200.0	50.0	4000.0	275.837779

```
common_nh = set(mcn_df1.neighborhood).intersection(set(mcn_df2.neighborhood))
print(common_nh)
```

```
{'Fort Wadsworth', 'Flatiron District', 'Woodrow', 'NoHo', 'Tribeca'}
```

```python
def neighborhood_price(in_df, fig_name='test_fig'):
    labels = list(in_df['neighborhood'].unique())

    _means = list(in_df['mean'])
    _medians = list(in_df['50%'])

    width = 0.45

    fig, ax = plt.subplots(figsize=(12,6))

    ax.set_ylabel('Airbnb price (per night)',  fontsize = 12)
    ax.set_title('Airbnb price with Neighborhoods', fontsize = 16)
    ax.set_xlabel('Neighborhoods', fontsize = 12)

    ax.set_xticks(np.arange(len(labels)))
    ax.set_xticklabels(labels, rotation = 45, fontsize = 11)

    rects1 = ax.bar(x - width/2, _means, width, label='Mean')
    rects2 = ax.bar(x + width/2, _medians, width, label='Median')

    ax.legend(bbox_to_anchor=(1, 1),fancybox=True, shadow=True, ncol=1)

    def addlabels(x, m, w):
        for i in range(len(x)):
            plt.text(i-.06, (m[i]*101/100), f'{str(round(m[i], 1))}', ha = 'right', fontsize = 10)
            plt.text(i+.06, (w[i]*101/100), f'{str(round(w[i], 1))}', ha = 'left', fontsize = 10)

    addlabels(labels, _means, _medians)

    plt.savefig('../report/fig/'+fig_name,  bbox_inches='tight')

    return plt.show()
```

Figure 29- Implementation of relationship between price and neighborhood group

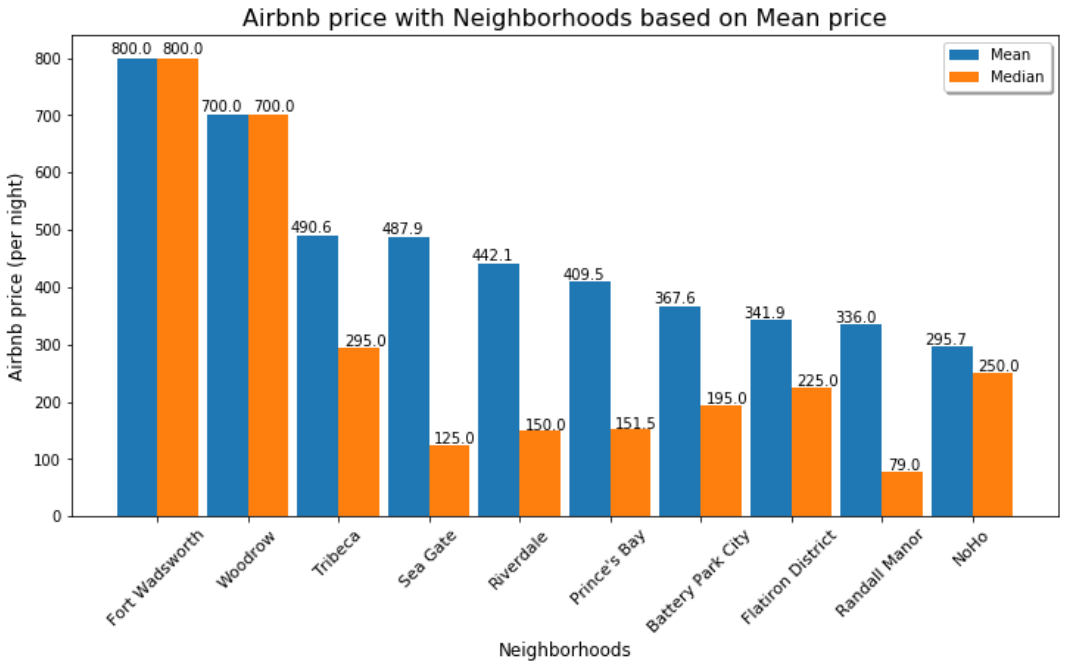


Figure 30- Top 10 neighborhoods based on mean price

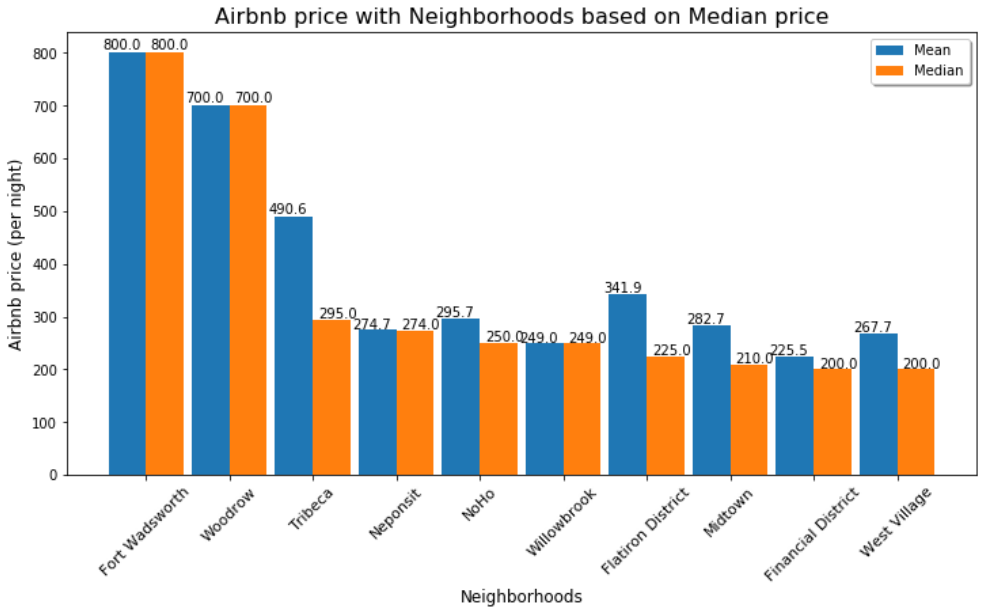


Figure 31- Top 10 neighborhoods based on median price

From the above graph, we made some interpretations.

Firstly, it is clear that using both parameters Fort Wadsworth, Woodrow, and Tribeca are the most expensive neighborhoods.

Secondly, only five out of ten neighborhoods are common in the two price group. Including the above-mentioned three neighborhoods, Flatiron District and NoHo are also common in both sets.

Thirdly, we further investigate why there are two difference sets using mean and median.

```
in_df = pd.concat([mcn_df1, mcn_df2]).drop_duplicates()

cmn_lab = list(common_nh)
mean_lab = [d for d in mcn_df1['neighborhood'] if d not in common_nh]
med_lab = [d for d in mcn_df2['neighborhood'] if d not in common_nh]

cmn_std = []
men_std = []
med_std = []

for ind, row in in_df.iterrows():
    if row['neighborhood'] in mean_lab:
        men_std.append(row['std'])
    elif row['neighborhood'] in med_lab:
        med_std.append(row['std'])
    else:
        cmn_std.append(row['std'])

labels = cmn_lab + med_lab + mean_lab
values = cmn_std + med_std + men_std

r4 = list(range(1, len(in_df)+1))
r1 = r4[:len(cmn_lab)]
r2 = r4[len(r1):len(cmn_lab)+len(r1)]
r3 = r4[len(r1+r2):len(cmn_lab)+len(r1+r2)]
```

```
barWidth = 0.95

fig, ax = plt.subplots(figsize=(12,8))
ax.set_title('Std. deviation of the price of top-15 neighborhoods', fontsize = 16)

# Create barplot
plt.bar(r1, cmn_std, width = barWidth, color = (0.3,0.1,0.4,0.6), label='std. of common neighborhood')
plt.bar(r2, med_std, width = barWidth, color = (0.3,0.5,0.4,0.6), label='std. of median neighborhood')
plt.bar(r3, men_std, width = barWidth, color = (0.3,0.9,0.4,0.6), label='std. of mean neighborhood')

# Create legend
plt.legend()

plt.xticks([r + barWidth for r in range(len(r4))], labels, rotation=90)

for i in range(len(r4)):
    plt.text(x = r4[i], y = (values[i]*101/100), s=(round(values[i],1)), ha='center', size = 12)

plt.savefig('../report/fig/plot_19.png',  bbox_inches='tight')
plt.show()
```

Figure 32- Investigation of the two different set of neighborhoods

Figure 33- Bar of the std. deviation of different neighborhoods

Analysis: Using this figure, we could answer the question for two different sets of neighborhoods. Here it is clear that the mean set is different from the median only because of the high variance in the price. Using this figure, we could answer the question for two different sets of neighborhoods. Here it is clear that the mean set is different from the median only because of the high variance in the price. We can get much variety price apartments in the neighborhoods like Sea Gate, Riverdale, Prince's Bay, Battery Park City, and Randall Manor.

3.3.5 Price VS Airbnb type: In this section, we have calculated the mean and median prices for all three types of Airbnb. Then, we plotted the stat using a bar chart.

```
apt_df = df[['room_type', 'price']]
apt_mean = apt_df.groupby('room_type').mean()
apt_median = apt_df.groupby('room_type').median()

apt_mean.rename(columns={'price':'Mean Price'}, inplace=True)
apt_median.rename(columns={'price':'Median Price'}, inplace=True)

apt_mean_meadian = apt_mean.join(apt_median)
apt_mean_meadian.insert(loc=0, column='Room Type', value=list(apt_mean_meadian.index))
apt_mean_meadian.reset_index(inplace=True, drop='index')

apt_mean_meadian
```

	Room Type	Mean Price	Median Price
0	Entire home/apt	211.794246	160
1	Private room	89.780973	70
2	Shared room	70.127586	45

```
labels = list(apt_mean_meadian['Room Type'])

_means = list(apt_mean_meadian['Mean Price'])
_medians = list(apt_mean_meadian['Median Price'])

width = 0.35

fig, ax = plt.subplots(figsize=(10,6))

x = np.arange(len(apt_mean_meadian))

ax.set_ylabel('Airbnb price (per night)')
ax.set_title('Airbnb price with apartment type')

ax.set_xlabel('apartment type')

ax.set_xticks(np.arange(len(labels)))
ax.set_xticklabels(labels, rotation = 0)

rects1 = ax.bar(x - width/2, _means, width, label='Mean')
rects2 = ax.bar(x + width/2, _medians, width, label='Median', color='#f0bc85')

ax.legend(bbox_to_anchor=(1, 1),fancybox=True, shadow=True, ncol=1)

def addlabels(x, m, w):
    for i in range(len(x)):
        plt.text(i, m[i], f'{str(round(m[i], 1))}', ha = 'right', fontsize = 10)
        plt.text(i, w[i], f'{str(round(w[i], 1))}', ha = 'left', fontsize = 10)

addlabels(labels, _means, _medians)

plt.show()
```

Figure 28- Implementation of the relationship between price and Airbnb type

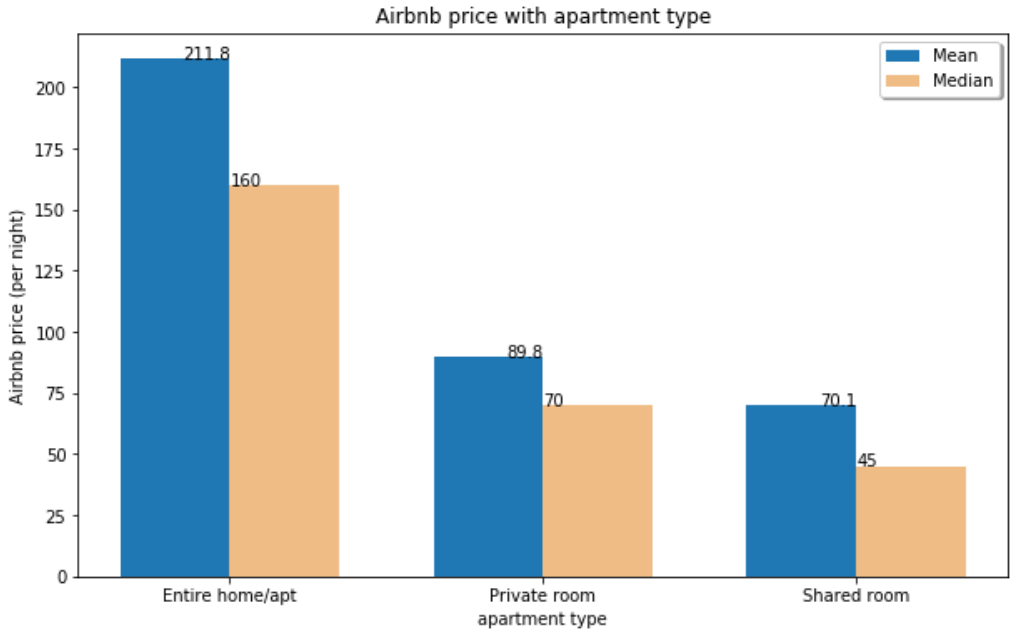


Figure 29- Plot of the relationship between price and Airbnb type

Analysis: The bar chart shows that both prices (mean and median) for the entire home/apartment type are the highest. The lowest price is the shared room. Though both types of parameters private-room are in the middle position, the values are much lower than the entire home/apartment.

3.3.6 Room type VS Neighborhood group: At this stage, we have counted how many Airbnb there are in each neighborhood group. Then, we plotted the data from the list in the form of graphs.

```
# neighbourhood_group vs apt.type

apt_nhg_df = df[['id', 'room_type', 'neighbourhood_group']]
apt_nhg_df = apt_nhg_df.groupby(['room_type','neighbourhood_group'], as_index=False)['id'].count()

apt_nhg_df.rename(columns={'id':'ID_Counts','neighbourhood_group': 'Neighbourhood_Group', \
                           'room_type': 'Room_Type' }, inplace=True)
apt_nhg_df
```

	Room_Type	Neighbourhood_Group	ID_Counts
0	Entire home/apt	Bronx	379
1	Entire home/apt	Brooklyn	9559
2	Entire home/apt	Manhattan	13199
3	Entire home/apt	Queens	2096
4	Entire home/apt	Staten Island	176
5	Private room	Bronx	652
6	Private room	Brooklyn	10132
7	Private room	Manhattan	7982
8	Private room	Queens	3372
9	Private room	Staten Island	188
10	Shared room	Bronx	60
11	Shared room	Brooklyn	413
12	Shared room	Manhattan	480
13	Shared room	Queens	198
14	Shared room	Staten Island	9

```
labels = list(apt_nhg_df['Neighbourhood_Group'].unique())

entire_home = list(apt_nhg_df['ID_Counts'][apt_nhg_df['Room_Type'] == 'Entire home/apt'])
private_apt = list(apt_nhg_df['ID_Counts'][apt_nhg_df['Room_Type'] == 'Private room'])
shared_apt = list(apt_nhg_df['ID_Counts'][apt_nhg_df['Room_Type'] == 'Shared room'])

width = 0.25

fig, ax = plt.subplots(figsize=(12,6))

ax.set_title('Neighborhood group wise counts of different types of Airnbs')

x = np.arange(len(labels))

ax.set_xticks(np.arange(len(labels)))
ax.set_xticklabels(labels, rotation = 0)

rects1 = ax.bar(x-width, entire_home, width, label='Entire Apt', color='#003f5c')
rects2 = ax.bar(x, private_apt, width, label='Private Apt', color='#f0bc85')
rects3 = ax.bar(x + width, shared_apt, width, label='Shared Apt', color='#75a67c')

ax.legend(bbox_to_anchor=(1, 1),fancybox=True, shadow=True, ncol=1)

def addlabels(x, e, p, s):
    for i in range(len(x)):
        plt.text(i+.25, s[i], f'{str(round(s[i], 1))}', ha = 'center', fontsize = 11)
        plt.text(i, p[i], f'{str(round(p[i], 1))}', ha = 'center', fontsize = 11)
        plt.text(i-.25, e[i], f'{str(round(e[i], 1))}', ha = 'center', fontsize = 11)


addlabels(labels, entire_home, private_apt, shared_apt)

plt.savefig('../report/fig/plot_15.png',  bbox_inches='tight')
plt.show()
```

Figure 30- Implementation of the relationship between neighborhood group and Airbnb type

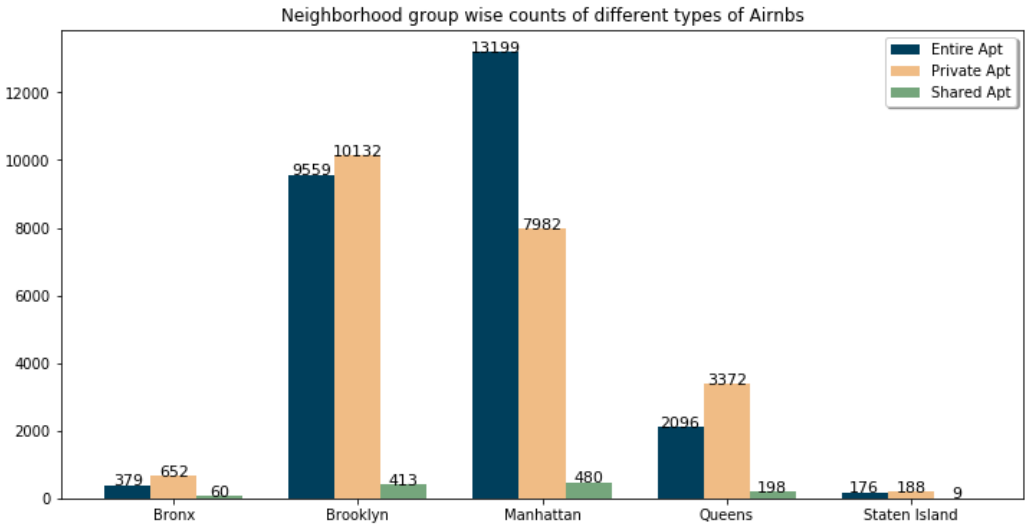


Figure 31- Bar plot of the relationship between neighborhood group and Airbnb type

Analysis: From the graph, we can get some assumption. Firstly, largest number of entire home/apt, and shared apartment are situated in Manhattan. Secondly, Brooklyn has maximum number of private apartment. Lastly, Staten Island has less Airbnb in all three types.

3.3.7 Get top reviewed neighborhoods: Here, we reveal the percentage of reviews in different neighborhood groups. As we don't have any information about the meaning of the review (positive and negative review), we could not analyze the sentiment of the reviews. From the given figure, we only calculate the percentage of the reviews. If we calculate the percentage of reviews based on the review number, it would be highly biased on the count of Airbnbs in those neighborhoods. As a result, we see Brooklyn and Manhattan hold most of the reviews. So, we decided to calculate the review per Airbnb.

```
# Neighborhood group Vs Number of review

nhg_rev_df = df[['number_of_reviews', 'neighbourhood_group']].groupby('neighbourhood_group', as_index=False).sum()
nhg_rev_df
```

	neighbourhood_group	number_of_reviews
0	Bronx	28371
1	Brooklyn	486574
2	Manhattan	454569
3	Queens	156950
4	Staten Island	11541

```
fig, ax = plt.subplots(figsize=(7,7))
ax.set_title('Review percentage in different Neighborhood groups')

ax.pie(nhg_rev_df.number_of_reviews,labels=nhg_rev_df.neighbourhood_group, autopct='%.0f%%',shadow=True)

ax.add_artist(plt.Circle((0,0),0.8,color='white'))
plt.savefig('../report/fig/plot_22.png',  bbox_inches='tight')

plt.show()
```

Figure 32- Implementation of Neighborhood group VS number of review

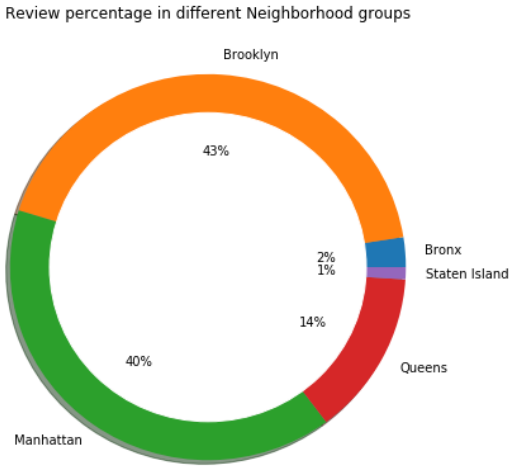


Figure 33- Pie-chart of Neighborhood group VS number of review

Analysis: Here, we reveal the percentage of reviews in different neighborhood groups. As we don't have any information about the meaning of the review, we could not analyze the sentiment of the reviews. From the given figure, we only calculate the percentage of the reviews. If we calculate the percentage of reviews based on the review number, it would be highly biased on the count of Airbnbs in those neighborhoods. As a result, we see Brooklyn and Manhattan hold most of the reviews. So, we decided to calculate the review per Airbnb.

```
reviews = list(nhg_rev_df.number_of_reviews)
airbnb_counts = list(nh_group.counts)

review_per_airbnbns = [reviews[i]/aribnb_counts[i] for i in range(len(reviews))]

nhg_rev_df.insert(2, 'review_per_airbnbn', review_per_airbnbns)

nhg_rev_df
```

	neighbourhood_group	number_of_reviews	review_per_airbnbn
0	Bronx	28371	26.004583
1	Brooklyn	486574	24.202845
2	Manhattan	454569	20.985596
3	Queens	156950	27.700318
4	Staten Island	11541	30.941019

Figure 34- Implementation of Neighborhood group VS review per Airbnb

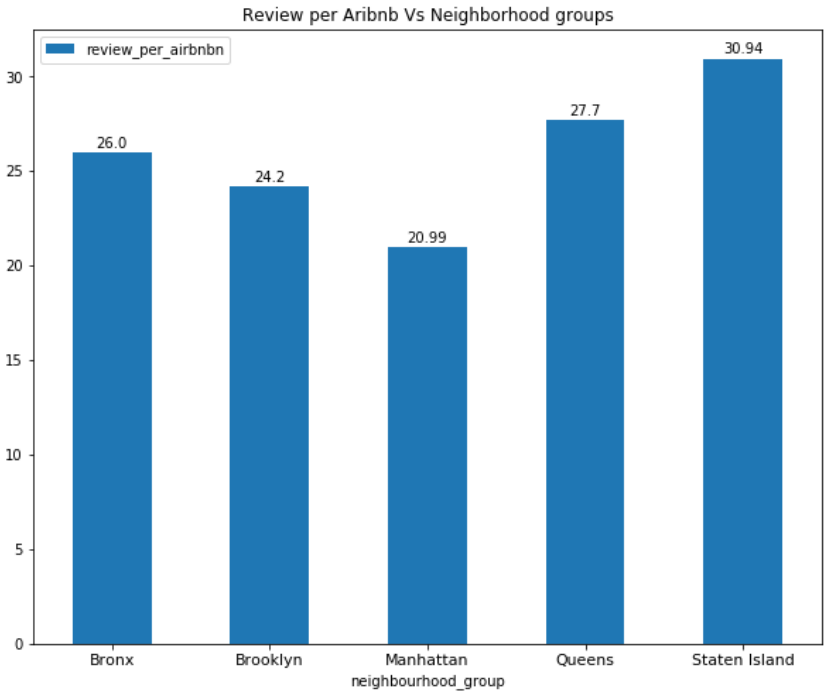


Figure 35- Bar plot of Neighborhood group VS review per Airbnb

Analysis: According to this analysis, The Airbnbs on Staten Island has the maximum average number of reviews, Although it contains only 1% of the total number of reviews. Queens has the second highest number of reviews per Airbnbs (27.7 reviews per Airbnb). More than 80% of the overall reviews are listed on the neighborhood groups Brooklyn and Manhattan, but their average counts are the lowest among entire neighborhoods.

3.3.8 Get top reviewed Airbnb types: Here, we reveal the percentage of reviews in different neighborhood groups. As we don't have any information about the meaning of the review (positive and negative review), we could not analyze the sentiment of the reviews. From the given figure, we only calculate the percentage of the reviews. If we calculate the percentage of reviews based on the review number, it would be highly biased on the count of Airbnbs in those neighborhoods. As a result, we see Brooklyn and Manhattan hold most of the reviews. So, we decided to calculate the review per Airbnb.

```
# Airbnb type Vs Number of review

type_rev_df = df[['number_of_reviews', 'room_type']].groupby('room_type', as_index=False).sum()
type_rev_df
```

	room_type	number_of_reviews
0	Entire home/apt	580403
1	Private room	538346
2	Shared room	19256

```
fig, ax = plt.subplots(figsize=(6,6))
ax.set_title('Review percentage in different Airbnb types')

ax.pie(type_rev_df.number_of_reviews, labels=type_rev_df.room_type, autopct='%.0f%%',shadow=True)
ax.add_artist(plt.Circle((0,0),0.8,color='white'))
plt.savefig('../report/fig/plot_24.png',  bbox_inches='tight')

plt.show()
```

Figure 36- Implementation of Airbnb type VS number of review

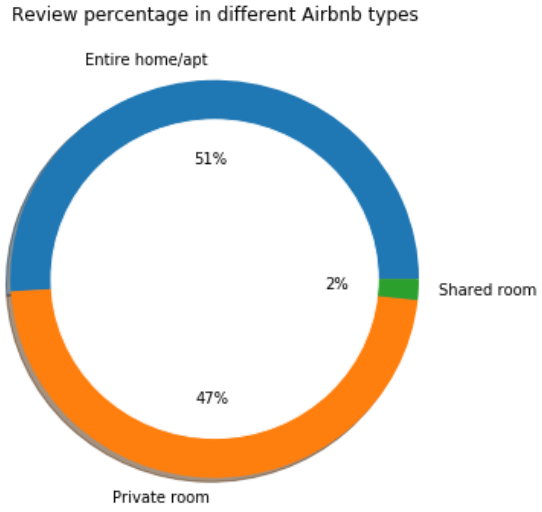


Figure 37- Pie-chart of Airbnb type VS number of review

```
reviews = list(type_rev_df.number_of_reviews)
airbnb_counts = list(arbnb_type.counts)

review_per_airbnbns = [reviews[i]/airbnb_counts[i] for i in range(len(reviews))]

type_rev_df.insert(2, 'review_per_airbnbn', review_per_airbnbns)

type_rev_df.rename(columns={'room_type':'Room Type', \
                            'review_per_airbnbn':'Reviews Per Airbnbs'}, inplace=True)

type_rev_df
```

	Room Type	number_of_reviews	Reviews Per Airbnbs
0	Entire home/apt	580403	22.842418
1	Private room	538346	24.112962
2	Shared room	19256	16.600000

```
def addlabels(x, y):
    for i in range(len(x)):
        plt.text(i, (y[i]*101/100), round(y[i],2), ha = 'center')

ax = type_rev_df.plot.bar(x='Room Type', y='Reviews Per Airbnbs', \
                          title='Review per Aribnb Vs Airbnb Type', figsize=(8,6))

ax.set_xticks(np.arange(len(type_rev_df)))
ax.set_xticklabels(type_rev_df['Room Type'],  rotation = 0, fontsize = 11)

addlabels(list(type_rev_df['Room Type']), list(type_rev_df['Reviews Per Airbnbs']))

plt.savefig('../report/fig/plot_25.png',  bbox_inches='tight')
plt.show()
```

Figure 38- Implementation of Airbnb type VS avg. Airbnb reviews

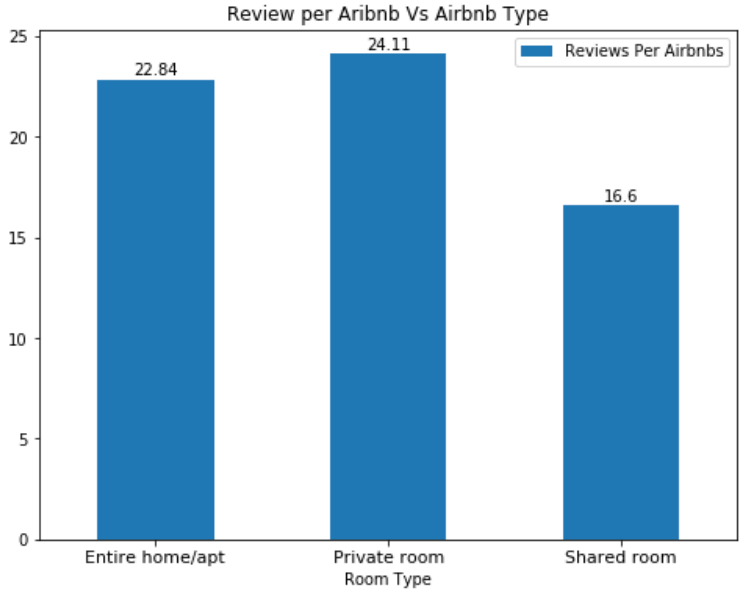


Figure 38- Bar plot of Airbnb type VS number of review (per Airbnb)

Analysis: According to the analysis, we interpret some pieces of information.

Firstly, the entire home/apt type has almost half of the total reviews. The average number of reviews per Airbnb is 22.84 in this category.

Secondly, There are 47% of total reviews are given to this category. On average 24.11 reviews are given to each Airbnb of this category.

Thirdly, As shared apartments are most uncommon in NYC, only 2% of total reviews are owned by this type.

4. Final Discussion

In this section we will summarize our finding according to the dataset and our analysis.

1. Almost all the data set is null free except two review related columns.
2. In NYC the Airbnb are situated on all 5 neighborhood group and 221 neighbor.
3. There is not linear relationship among the numeric feature of the dataset.
4. Renting entire apt/home is most popular type as more than 50% of the data situated on this category also it is most expensive one.
5. Least costly Airbnb type is shared room also it is least popular.
6. Most Expensive Airbnb(s) are situated in Manhattan and also become most popular (based on Airbnb count) among other group.
7. Airbnb with lowest number of minimum night are most popular in NYC.

5. References

1. https://blog.datawrapper.de/beautifulcolors/
2. https://businessmodelnavigator.com/case-firm?id=4
3. https://www.schemecolor.com/flat-orange-blue-green-pie-chart.php
4. https://writing.wisc.edu/handbook/assignments/planresearchpaper/
5. https://www.kaggle.com/datasets/dgomonov/new-york-city-airbnb-open-data