Visualize New York City Airbnb Data

6th July 2022

Ahmed Dider Rahat (916146)
Tania Sultana (931031)

BHT
Berliner Hochschule für Technik

Outlines

Introduction

Data Visualization:

Data visualization is a graphical representation of data that expresses its significance.

Airbnb:

- Airbnb is an American company which operates an online marketplace and hospitality service for people to lease or rent short-term apartments, homestays, hostel beds, or hotel rooms.
- The company has over 4 million lodging listings in 65,000 cities and 191 countries and has facilitated over 260 million check-ins.

The Dataset

- The Data is collected from Kaggle.
- This dataset describes the listing activity and metrics in NYC, NY for 2019.
- There are total 16 features and 48,895 data points.

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.
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 3 categories.
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.

Visualizing the Airbnb dataset

We visualize the data using three broad categories.

1. Analyze basic properties of the features.
2. Univariate feature analysis.
3. Bivariate feature analysis.

Basic statistics of the Dataset

1. Data load in pandas:

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

2. Observe the data type:

```
# 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
```

3. Statistical Analysis for Numerical Data:

```
print(df.describe())
```

```
                 id       host_id      latitude     longitude         price  \
count  4.889500e+04  4.889500e+04  48895.000000  48895.000000  48895.000000
mean   1.901714e+07  6.762001e+07     40.728949    -73.952170    152.720687
std    1.098311e+07  7.861097e+07      0.054530      0.046157    240.154170
min    2.539000e+03  2.438000e+03     40.499790    -74.244420      0.000000
25%    9.471945e+06  7.822033e+06     40.690100    -73.983070     69.000000
50%    1.967728e+07  3.079382e+07     40.723070    -73.955680    106.000000
75%    2.915218e+07  1.074344e+08     40.763115    -73.936275    175.000000
max    3.648724e+07  2.743213e+08     40.913060    -73.712990  10000.000000

       minimum_nights  number_of_reviews  reviews_per_month  \
count    48895.000000       48895.000000       38843.000000
mean         7.029962          23.274466           1.373221
std         20.510550          44.550582           1.680442
min          1.000000           0.000000           0.010000
25%          1.000000           1.000000           0.190000
50%          3.000000           5.000000           0.720000
75%          5.000000          24.000000           2.020000
max       1250.000000         629.000000          58.500000

       calculated_host_listings_count  availability_365
count                    48895.000000     48895.000000
mean                         7.143982       112.781327
std                         32.952519       131.622289
min                          1.000000         0.000000
25%                          1.000000         0.000000
50%                          1.000000        45.000000
75%                          2.000000       227.000000
max                        327.000000       365.000000
```

4. Categorical Feature Analysis:

We have total 3 categorical features in our data set. These are neighbourhood_group, neighborhood, and room_type.

4.1. Neighbourhood_group:

```
ngh_group = list(df['neighbourhood_group'].unique())
print(f'Total unique neighbourhood group: {len(ngh_group)}')
print(f'Neighbourhoods group are: {ngh_group}')
```

```
Total unique neighbourhood group: 5
Neighbourhoods group are: ['Brooklyn', 'Manhattan', 'Queens', 'Staten Island', 'Bronx']
```

4.2. Neighbourhood:

```
neighbourhood = list(df['neighbourhood'].unique())
print(f'Total unique neighbourhood: {len(neighbourhood)}')

mpn_list = list((df.neighbourhood.value_counts().head(10)).index)

print(mpn_list)
```

```
Total unique neighbourhood: 221
['Williamsburg', 'Bedford-Stuyvesant', 'Harlem', 'Bushwick', 'Upper West Side', "Hell's Kitchen", 'East Village', 'Upper East S
ide', 'Crown Heights', 'Midtown']
```

4. Categorical Feature Analysis (cont.):

4.3. room_type:

```
room_type = list(df['room_type'].unique())
print(f'Total unique room type: {len(room_type)}')
print(f'Room types are: {room_type}')
```

```
Total unique room type: 3
Room types are: ['Private room', 'Entire home/apt', 'Shared room']
```

5. Null Value Analysis:

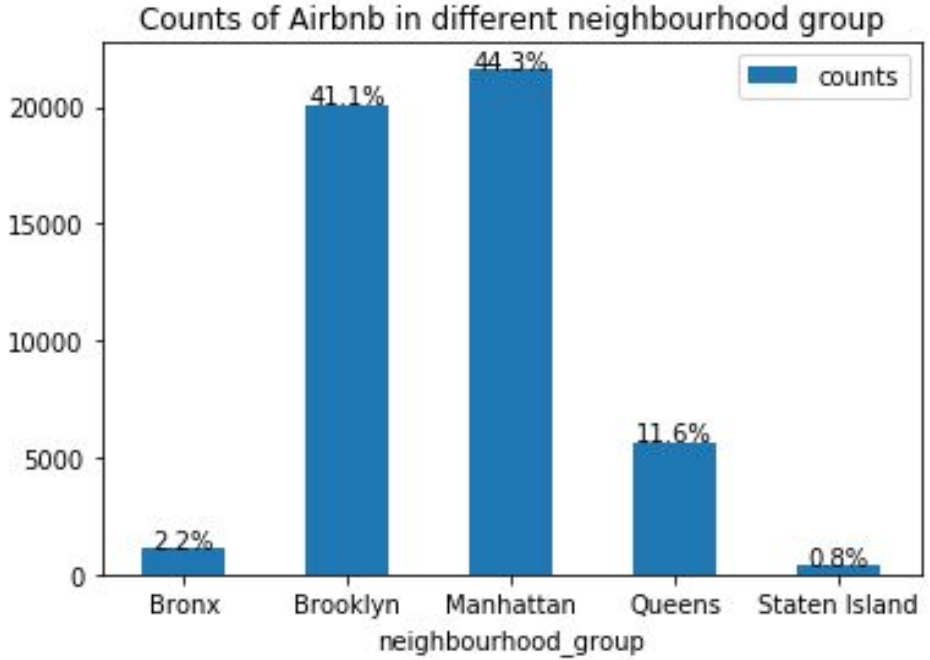
```
# null value count
print(df.isnull().sum())
```

```
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
```

Univariate Feature Analysis

In this section, we analyze the univariate features.

1. Discussion on Neighbourhood Group:



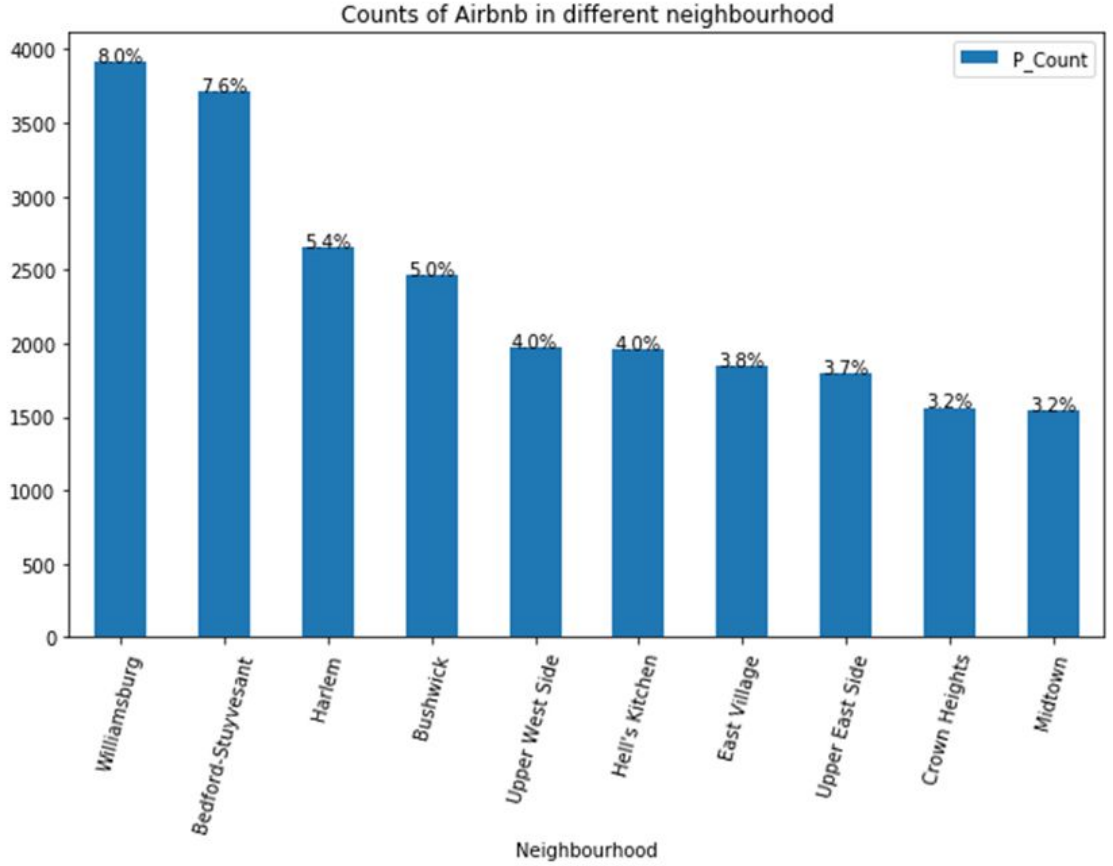
Implementation

```
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()
```

2. Discussion on Neighbourhood:



Implementation

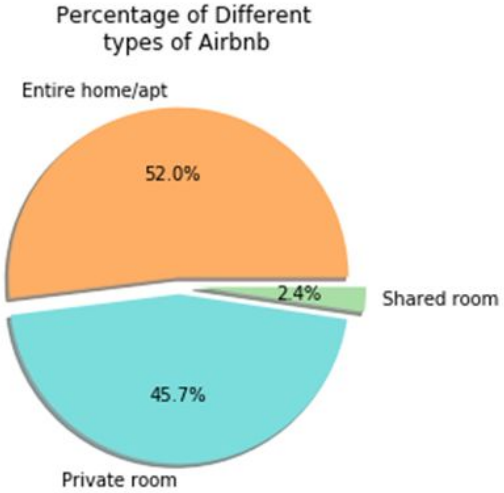
```
mpn_df = pd.DataFrame(df.neighbourhood.value_counts())
mpn_df.reset_index(inplace=True)

mpn_df.rename(columns={'index':'Neighbourhood', 'neighbourhood':'P_Count'}, inplace=True)
mpn_df.head(10)
```

	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

```
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(10).Neighbourhood)
counts = list(mpn_df.head(10).P_Count)
print(f'Total number of percentage for top 10 neighborhood is: \
     {round((sum(counts) * 100 / sum(list(mpn_df.P_Count))),2)}')

ax = mpn_df.head(10).plot.bar(x='Neighbourhood', y='P_Count', rot=75, \
               title='Counts of Airbnb in different neighbourhood', figsize=(10,6))

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_2.png',  bbox_inches='tight')
plt.show()
```

```
Total number of percentage for top 10 neighborhood is:      47.95
```

3. Discussion on Apartment type:

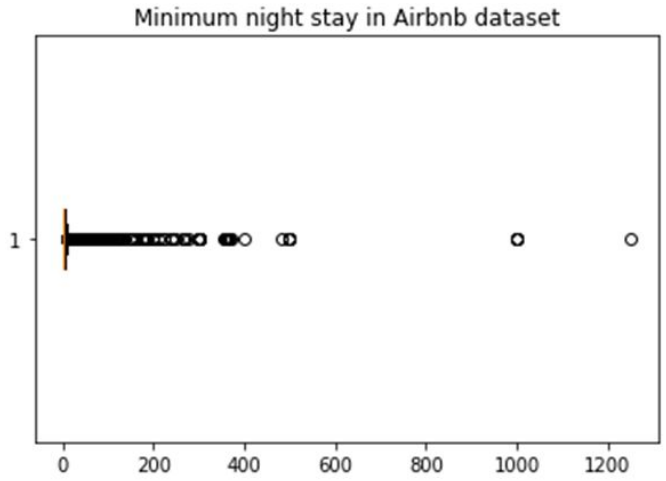



```
arbnb_type = df.groupby(['room_type'])['room_type'].count().reset_index(name='counts')
types = list(arbnb_type.room_type)
counts = list(arbnb_type.counts)

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()
```

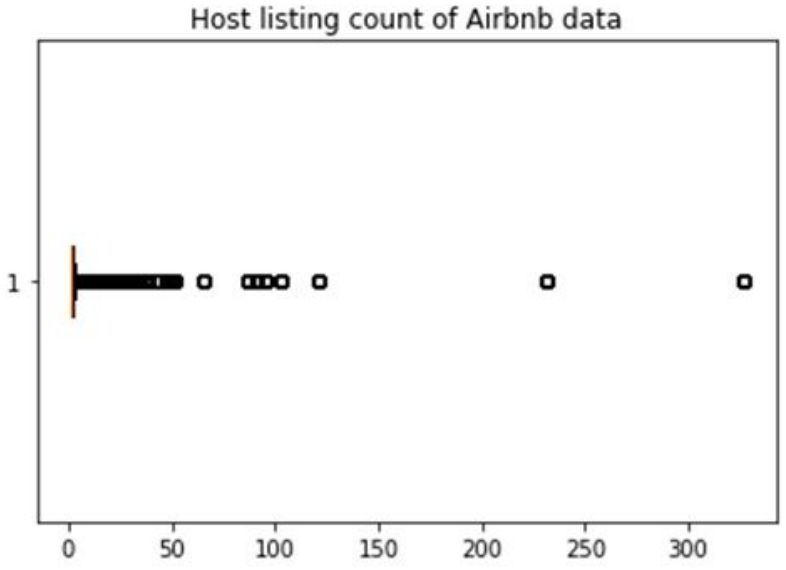
4. Discussion on Apartment type:




```
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
```

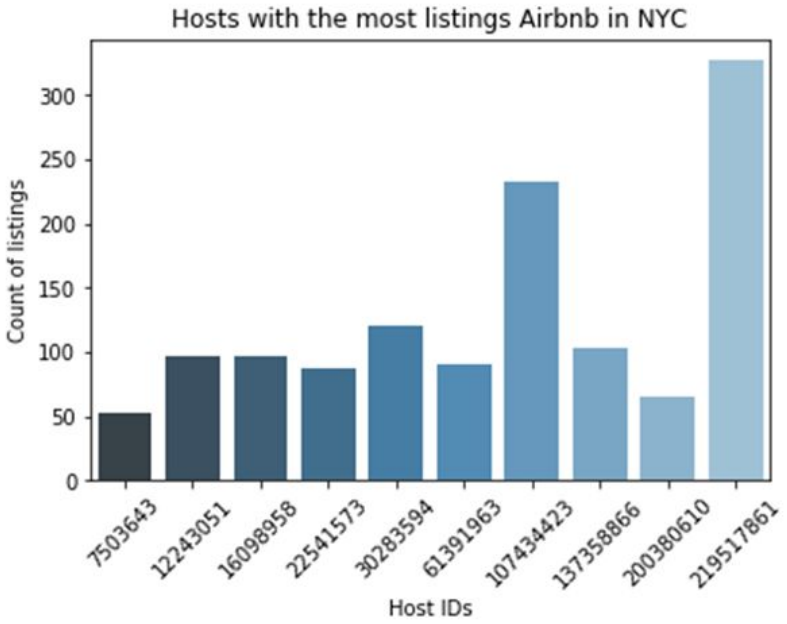
5. Discussion on Host Listing:




```
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
```

6. Discussion on Top Most Host:

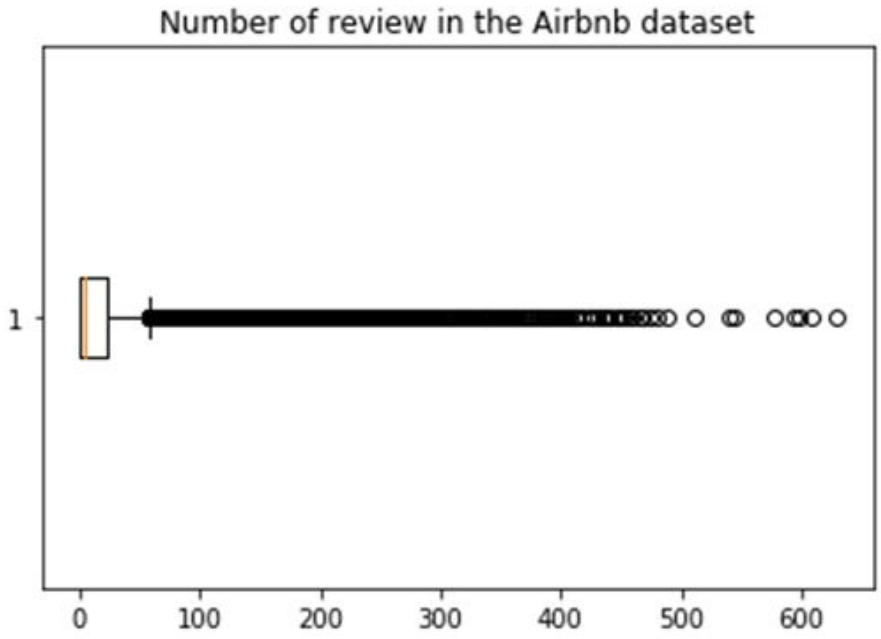



```
# get top-most host
top_host = df.host_id.value_counts().head(10)

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()
```

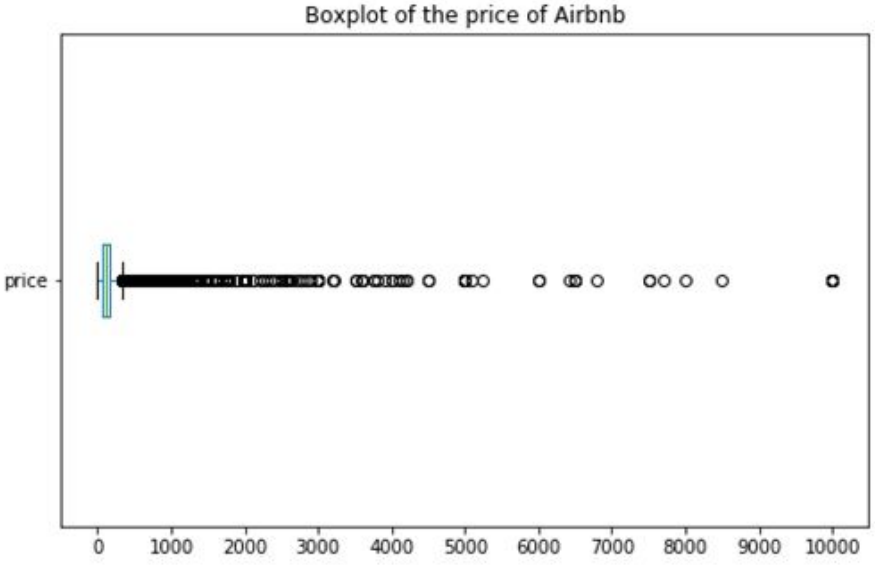
7. Discussion on Number of Reviews:




```
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()
```

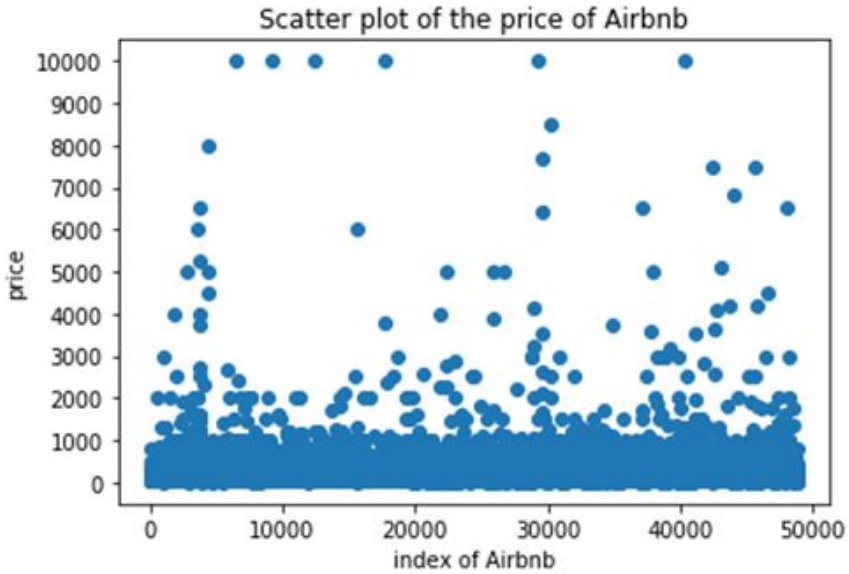
```
Minimum: 0 and Maximum: 629
```

8. Price Analysis of Airbnb (Box 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()
```

9. Price Analysis of Airbnb (Scatter Plot):




```
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()
```

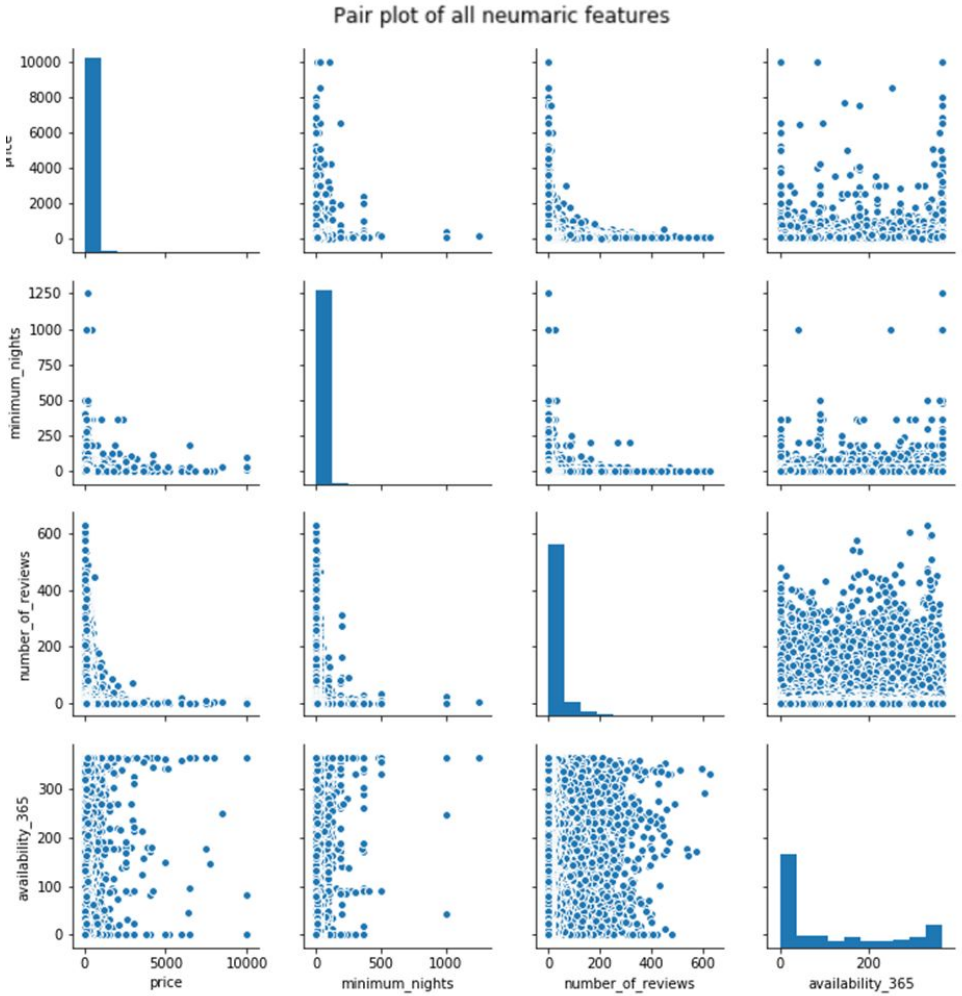
Bivariate Feature Analysis

In this portion, we do bivariate features analysis.

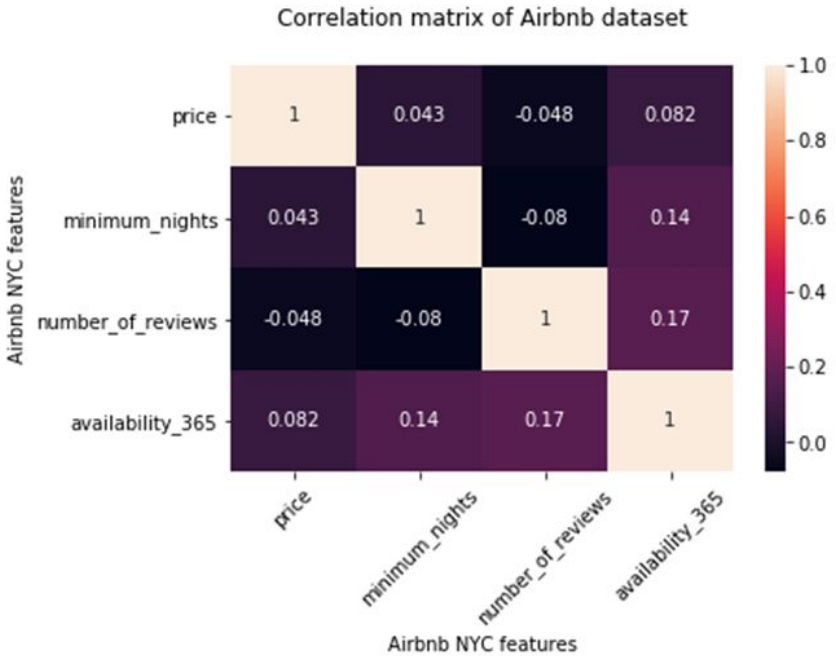
1. **Analysing Numerical Features: (Implementation)**

```
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()
```

Analysing Numerical Features: (pair-plot)



2. Correlation Matrix:

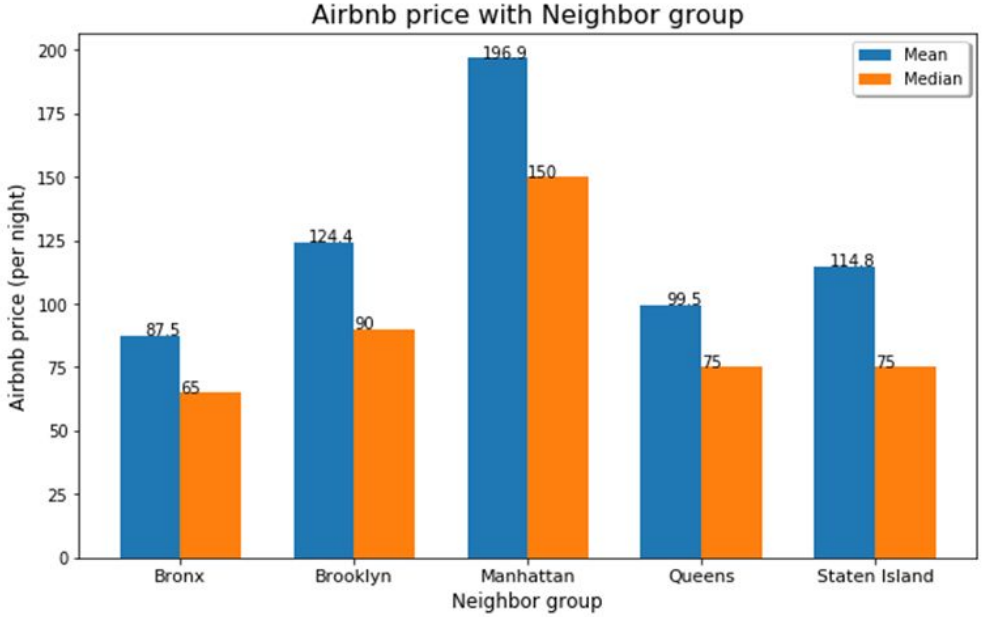



```
corr_matrix = cor_df.corr()
corr_matrix
```

	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

```
corr_hm = sns.heatmap(corr_matrix, annot = True)

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_11.png',  bbox_inches='tight')
plt.show()
```

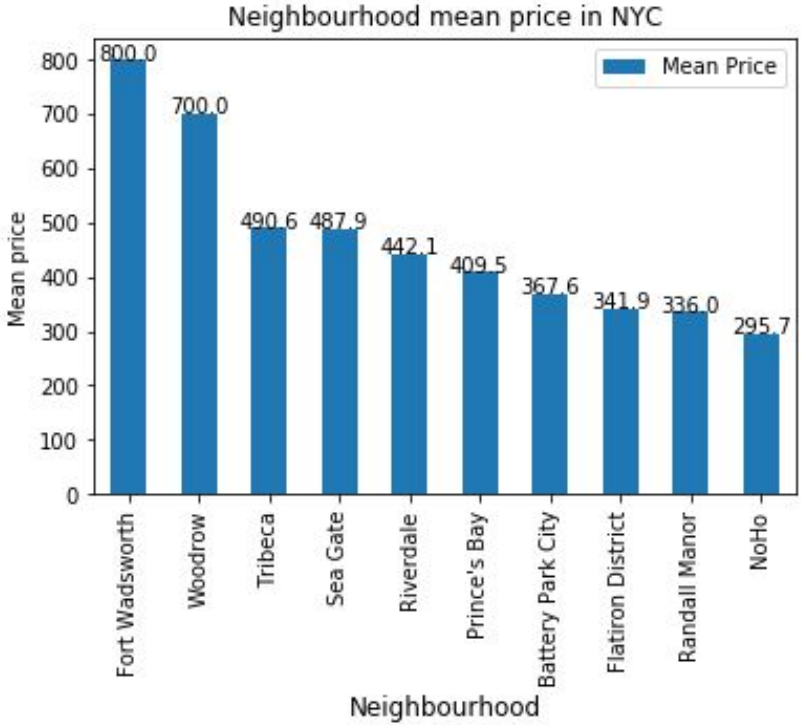
3. Price Vs Neighbourhood Group:

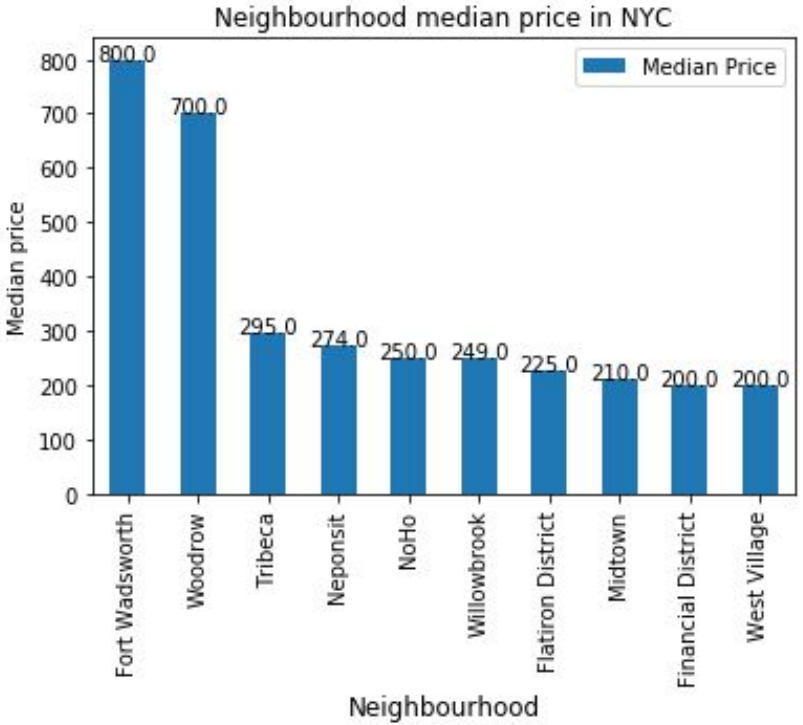


```
nhg_df = df[['neighbourhood_group', 'price']]
nhg_mean = nhg_df.groupby('neighbourhood_group').mean()
nhg_median = nhg_df.groupby('neighbourhood_group').median()

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()
```

4. Price Vs Neighbourhood: To analyze the price column we calculate mean and median of each group.





Implementation

```
# most costly neighbourhood

nh_df = df[['neighbourhood', 'price']]
nh_mean = nh_df.groupby('neighbourhood').mean()
nh_median = nh_df.groupby('neighbourhood').median()

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

nh_mean_meadian = nh_mean.join(nh_median)
nh_mean_meadian.insert(loc=0, column='Neighbour', value=list(nh_mean_meadian.index))
nh_mean_meadian.reset_index(inplace=True, drop='index')

mcn_df1 = nh_mean_meadian[['Neighbour', 'Mean Price']].nlargest(10, 'Mean Price')
mcn_df2 = nh_mean_meadian[['Neighbour', 'Median Price']].nlargest(10, 'Median Price')

nh_mean_meadian.head()
```

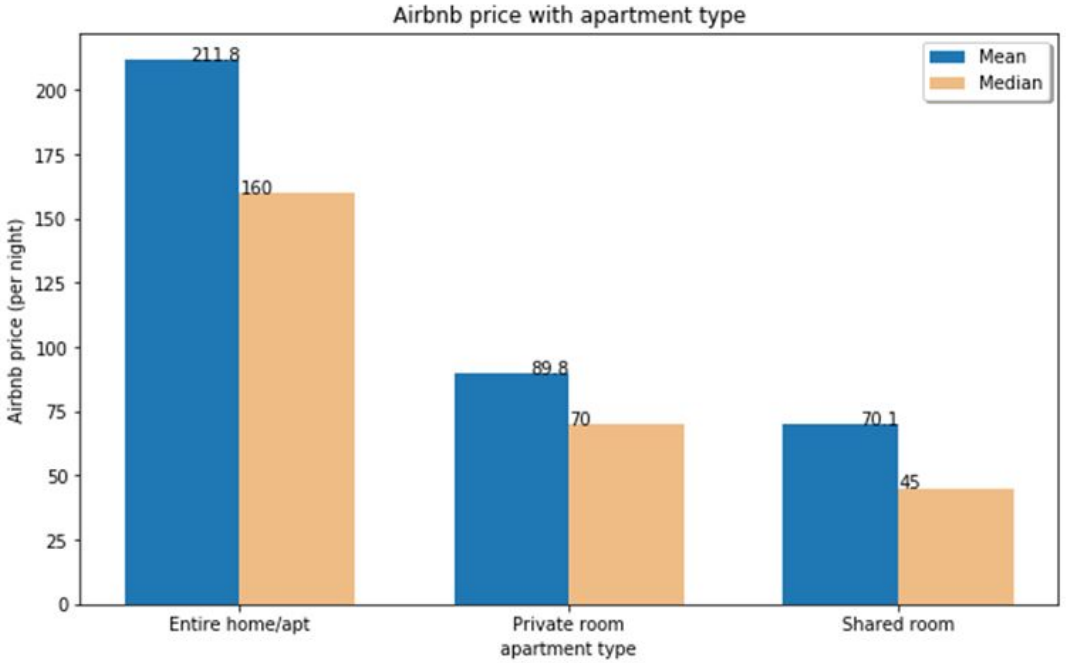
	Neighbour	Mean Price	Median Price
0	Allerton	87.595238	66.5
1	Arden Heights	67.250000	72.5
2	Arrochar	115.000000	65.0
3	Arverne	171.779221	125.0
4	Astoria	117.187778	85.0

Implementation

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

ax = mcn_df1.plot.bar(x='Neighbour', y='Mean Price', rot=90, title='Neighbourhood mean price in NYC')

ax.set_xlabel('Neighbour', fontsize=12)
ax.set_ylabel('Mean price')

addlabels(list(mcn_df1['Neighbour']), list(mcn_df1['Mean Price']))

plt.show()
```

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

ax = mcn_df2.plot.bar(x='Neighbour', y='Median Price', rot=90, title='Neighbourhood median price in NYC')

ax.set_xlabel('Neighbour', fontsize=12)
ax.set_ylabel('Median price')

addlabels(list(mcn_df2['Neighbour']), list(mcn_df2['Median Price']))

plt.show()
```

5. Price Vs Airbnb type: To analyze the price column we calculate mean and median of each group.



Implementation

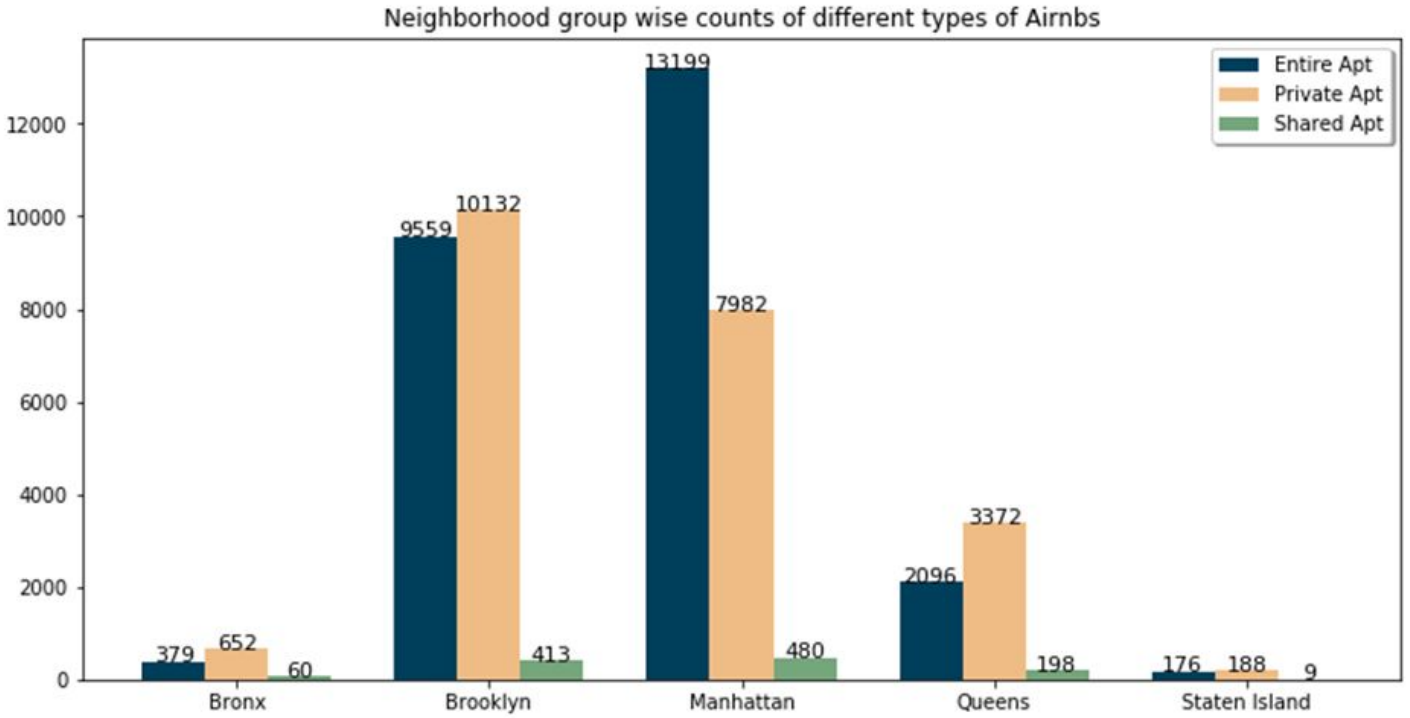
```
apt_df = df[['room_type', 'price']]
apt_mean = apt_df.groupby('room_type').mean()
apt_median = apt_df.groupby('room_type').median()

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

Implementation

```
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()
```

6. Airbnb type Vs Neighbourhood group:



Implementation

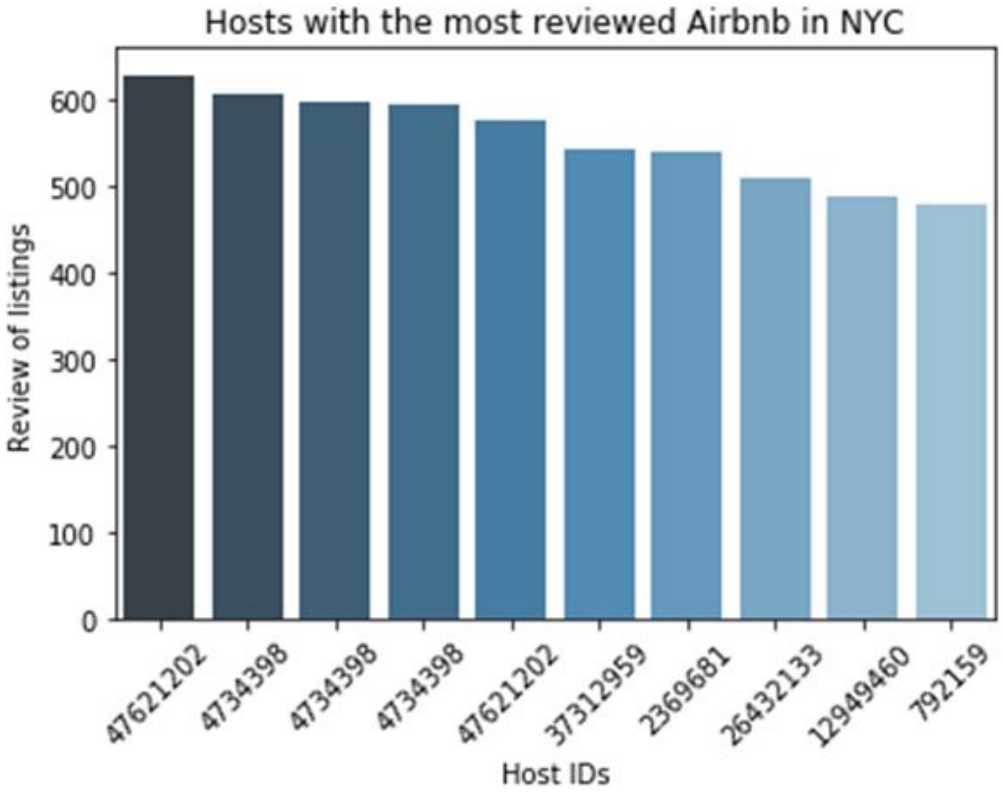
```
apt_nhg_df = df[['id', 'room_type', 'neighbourhood_group']]
apt_nhg_df = apt_nhg_df.groupby(['room_type','neighbourhood_group'], as_index=False)['id'].count()

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

Implementation

```
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()
```

7. Analyze most reviewed host:



Implementation

```
# get top-reviewd host
top_review_df = df[['host_id', 'number_of_reviews']].nlargest(10, 'number_of_reviews')

unique_host = set(list(top_review_df['host_id']))

top_review_df.reset_index(inplace=True, drop='index')
top_review_df.rename(columns={'host_id':'Host_ID', 'number_of_reviews':'R_Count'}, inplace=True)

top_review_df
```

	Host_ID	R_Count
0	47621202	629
1	4734398	607
2	4734398	597
3	4734398	594
4	47621202	576
5	37312959	543
6	2369681	540
7	26432133	510
8	12949460	488
9	792159	480

Implementation

```
top_reviewed_host_plot = sns.barplot(x = top_review_df.index, y = "R_Count", \
                                     data = top_review_df, palette='Blues_d')

top_reviewed_host_plot.set_title('Hosts with the most reviewed Airbnb in NYC')
top_reviewed_host_plot.set_ylabel('Review of listings')
top_reviewed_host_plot.set_xlabel('Host IDs')
top_reviewed_host_plot.set_xticklabels(top_review_df.Host_ID, rotation=45)

plt.savefig('../report/fig/plot_16.png',  bbox_inches='tight')
plt.show()
```

Final Discussions

1. Almost all of the data set is null-free except for two review-related columns.
2. In NYC, Airbnb is situated in all 5 neighborhood groups and 221 neighborhoods.
3. There is no linear relationship among the numeric feature of the dataset.
4. Renting an entire apt/home is the most popular type as more than 50% of the data is located in this category also it is the most expensive one.
5. The least costly Airbnb type is shared room also it is negligibly popular.
6. Most Expensive Airbnb(s) are situated in Manhattan and also become most popular among other groups.

References

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

Questions?

THANK YOU