Airbnb Market Analysis and Forecasting: Insights from an Exploration of Trends.

Aishat Abdulgafar

Table of Contents

PROJECT TITLE

Airbnb Market Analysis and Forecasting: Insights from an Exploration of Trends.

DATASET

The Airbnb dataset for New York City was obtained from insideairbnb.com and contains information on listings, reviews, and calendar availability for properties available on the Airbnb platform in New York City. The listing data includes information such as the property's location, price, number of bedrooms and bathrooms, and amenities offered. The review data includes information such as the date of the review, the reviewer's name, and their comments. The calendar availability data includes information such as the dates when the property is available for booking and the price for those dates. The dataset is typically scraped from the Airbnb website every month, so it's always updated. The dataset is also cleaned and preprocessed to ensure that it is in a usable format for analysis.

Step 1

Frame the problem: This scientific research intends to discover insights and explore the characteristics of the listings and their relationship with the price. Then, use the insights gained and the information provided by the dataset to build a predictive model that can accurately predict the price of a listing based on its characteristics such as location, number of rooms, type of property, and other relevant features. The goal is to provide valuable insights and information for both hosts and guests of Airbnb in NYC, as well as assist hosts in pricing their listings competitively and guests in finding affordable accommodations. Additionally, the model should be able to predict the price of a listing with high accuracy to help hosts and guests make informed decisions.

Objectives

- To build a supervised machine learning model for forecasting the price of an Airbnb based on multiple attributes.
- To provide graphical comparisons to provide an insight on the NYC Airbnb dataset.

What is your solution?

I plan to conduct an in-depth analysis of Airbnb market trends in a specific location, using historical data, advanced analytical techniques, and machine learning model predictions to identify patterns and forecast future market price performance.

Step 2

Get the data.

```
pip install worldcloud
```

```
Note: you may need to restart the kernel to use updated packages.
```

```
ERROR: Could not find a version that satisfies the requirement worldcloud (from versions: none)
ERROR: No matching distribution found for worldcloud
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from IPython.display import HTML, display
import seaborn as sns; sns.set()
from wordcloud import WordCloud
```

Get the Data

```python
df=pd.read_csv('listings.csv')
df.head()
```

```
C:\Users\Aishat\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: DtypeWarning: Columns (17) have mixed types.
Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_review
0	5136	Spacious Brooklyn Duplex, Patio + Garden	7378	Rebecca	Brooklyn	Sunset Park	40.66265	-73.99454	Entire home/apt	275	21	[illegible]
1	5203	Cozy Clean Guest Room - Family Apt	7490	MaryEllen	Manhattan	Upper West Side	40.80380	-73.96751	Private room	75	2	11
2	5121	BlissArtsSpace!	7356	Garon	Brooklyn	Bedford-Stuyvesant	40.68535	-73.95512	Private room	60	30	5
3	5178	Large Furnished Room Near B'way	8967	Shunichi	Manhattan	Midtown	40.76457	-73.98317	Private room	68	2	55
4	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown	40.75356	-73.98559	Entire home/apt	175	30	4

The first code shows the list of packages that were imported to aid this research. To get the data for this analysis, the above code was used to load the CSV file into a Pandas dataframe and the code below it is used to display the first 5 rows for previewing the structure and sample data of the dataset before analyzing it.

Step 3

Explore the data to gain insights.

Understanding the dataset

```
df.shape
```

```
(41533, 18)
```

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41533 entries, 0 to 41532
Data columns (total 18 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   id                              41533 non-null  int64
 1   name                            41520 non-null  object
 2   host_id                         41533 non-null  int64
 3   host_name                       41528 non-null  object
 4   neighbourhood_group             41533 non-null  object
 5   neighbourhood                   41533 non-null  object
 6   latitude                        41533 non-null  float64
 7   longitude                       41533 non-null  float64
 8   room_type                       41533 non-null  object
 9   price                           41533 non-null  int64
 10  minimum_nights                  41533 non-null  int64
 11  number_of_reviews               41533 non-null  int64
 12  last_review                     32140 non-null  object
 13  reviews_per_month               32140 non-null  float64
 14  calculated_host_listings_count  41533 non-null  int64
```

```
df.isnull().sum()
```

```
id                                    0
name                                 13
host_id                               0
host_name                             5
neighbourhood_group                   0
neighbourhood                         0
latitude                              0
longitude                             0
room_type                             0
price                                 0
minimum_nights                        0
number_of_reviews                     0
last_review                        9393
reviews_per_month                  9393
calculated_host_listings_count        0
availability_365                      0
number_of_reviews_ltm                 0
license                           41532
dtype: int64
```

```
df.describe()
```

	id	host_id	latitude	longitude	price	minimum_nights	number_of_reviews	reviews_per_month	calculated_host_listings
count	4.153300e+04	4.153300e+04	41533.000000	41533.000000	41533.000000	41533.000000	41533.000000	32140.000000	41533
mean	1.728318e+17	1.400636e+08	40.728292	-73.944526	221.978282	18.592204	26.204994	1.279287	20
std	2.974371e+17	1.526932e+08	0.057145	0.055965	919.502236	30.699921	56.178847	1.935098	68
min	2.595000e+03	2.438000e+03	40.500314	-74.249840	0.000000	1.000000	0.000000	0.010000	1
25%	1.835861e+07	1.491162e+07	40.687750	-73.982410	80.000000	2.000000	1.000000	0.140000	1
50%	4.117861e+07	6.561181e+07	40.723830	-73.953156	131.000000	10.000000	5.000000	0.580000	1
75%	5.477978e+17	2.418897e+08	40.762200	-73.924990	220.000000	30.000000	25.000000	1.880000	4
max	7.741268e+17	4.899967e+08	40.911380	-73.710870	98159.000000	1250.000000	1866.000000	102.980000	487

```
print("Neighbourhood Groups:", df['neighbourhood_group'].unique().tolist())
print("Room Types:", df['room_type'].unique().tolist())
```

```
Neighbourhood Groups: ['Brooklyn', 'Manhattan', 'Queens', 'Bronx', 'Staten Island']
Room Types: ['Entire home/apt', 'Private room', 'Hotel room', 'Shared room']
```

```
print(df['price'].describe(percentiles=[.25, .50, .75, .95]))
```

```
count    41533.000000
mean       221.978282
std        919.502236
min          0.000000
25%         80.000000
50%        131.000000
75%        220.000000
95%        581.000000
max      98159.000000
Name: price, dtype: float64
```

I utilized the shape () and info() functions to thoroughly examine the structure and characteristics of the Airbnb dataset. The shape () function displayed the dimensions of the data frame, offering valuable information about its size. The info () function, on the other hand, provided a concise summary of the data frame, including the number of non-null values, the number of rows, and the data types of columns. This information was essential in detecting any missing or null values, data type inconsistencies, and memory usage issues. With these tools, I was able to gain a comprehensive understanding of the dataset and make informed decisions when constructing my predictive model.

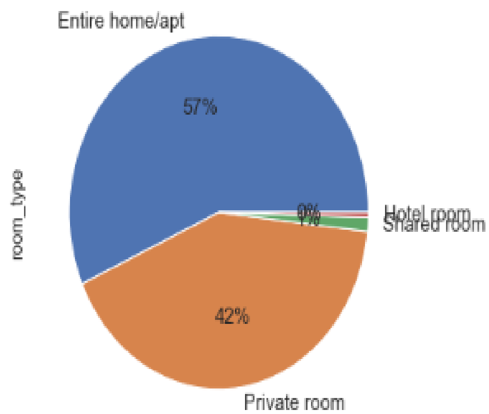
Data Exploration

```
# we noted that the room_type is only of 3 particular types.
df['room_type'].value_counts()
```

```
Entire home/apt    23526
Private room       17287
Shared room          532
Hotel room           188
Name: room_type, dtype: int64
```

```
fig = plt.figure(figsize=(5,5), dpi=80)
df['room_type'].value_counts().plot(kind='pie',  autopct='%1.0f%%', startangle=360, fontsize=13)
```

```
<AxesSubplot:ylabel='room_type'>
```




```
# There are 5 particular neighbourhood_group, which means 5 unique locations.
df['neighbourhood_group'].value_counts()
```

```
Manhattan        17334
Brooklyn         15688
Queens            6519
Bronx             1587
Staten Island      405
Name: neighbourhood_group, dtype: int64
```

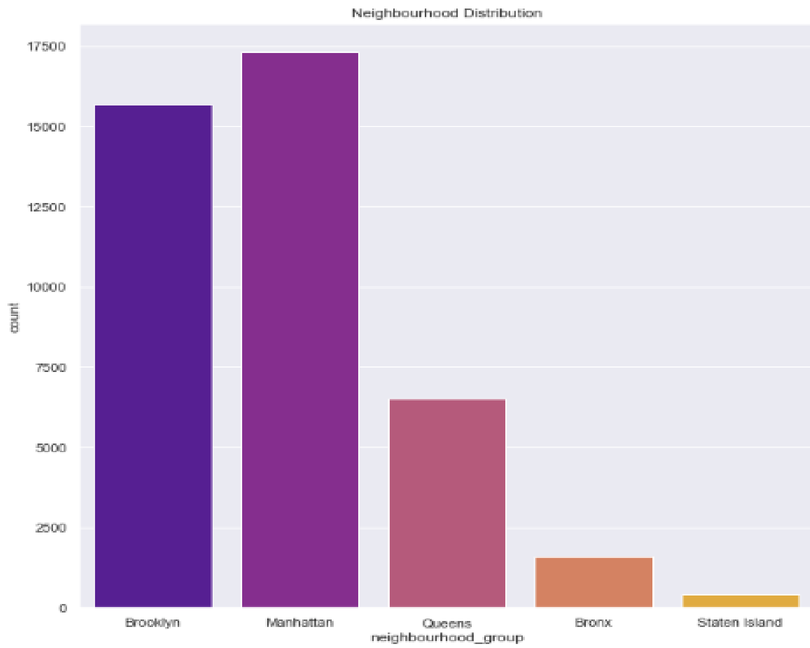
```
df['neighbourhood'].value_counts().iloc[:5]
```

```
Bedford-Stuyvesant    2936
Williamsburg          2570
Harlem                1949
Midtown               1918
Bushwick              1752
Name: neighbourhood, dtype: int64
```

```
sns.countplot(df['neighbourhood_group'], palette="plasma")
fig = plt.gcf()
fig.set_size_inches(10,10)
plt.title('Neighbourhood Distribution')
```

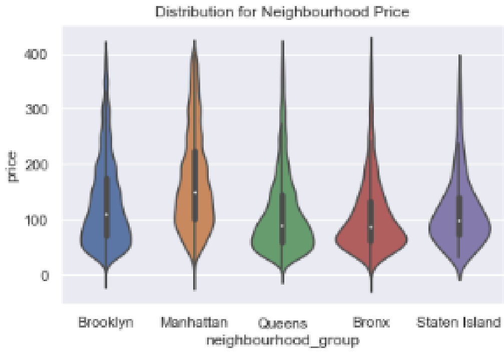
```
C:\Users\Aishat\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword
arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```

```
Text(0.5, 1.0, 'Neighbourhood Distribution')
```



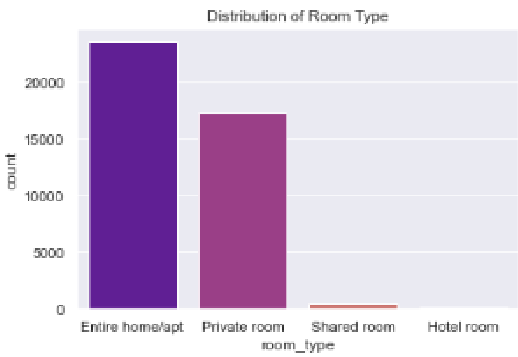
The data reveals that Manhattan and Brooklyn possess the highest number of listings, both hovering just above 15,000. This phenomenon can be attributed to the abundance of tourist hotspots in these neighborhoods, attracting individuals who desire proximity to their desired attractions.

```
price_df=df[df.price < 400]
viz_2=sns.violinplot(data=price_df, x='neighbourhood_group', y='price')
viz_2.set_title('Distribution for Neighbourhood Price')
plt.show()
```



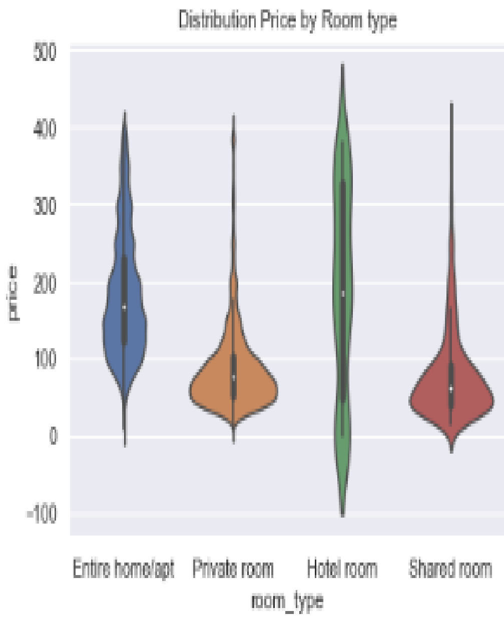

```
ax = sns.countplot('room_type',data=df,order=df['room_type'].value_counts().index, palette="plasma")
ax.set_title('Distribution of Room Type')
plt.show()
```

```
C:\Users\Aishat\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword
arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit k
eyword will result in an error or misinterpretation.
  warnings.warn(
```



The pricing of properties is closely tied to their neighborhood group. Manhattan presents a greater proportion of premium properties, while the Bronx, Staten Island, and Queens offer comparatively more budget-friendly options than Brooklyn and Manhattan. Moreover, all price distributions exhibit a positive skewness.

```
viz_2=sns.violinplot(data=price_df, x='room_type', y='price')
viz_2.set_title('Distribution Price by Room type')
plt.show()
```

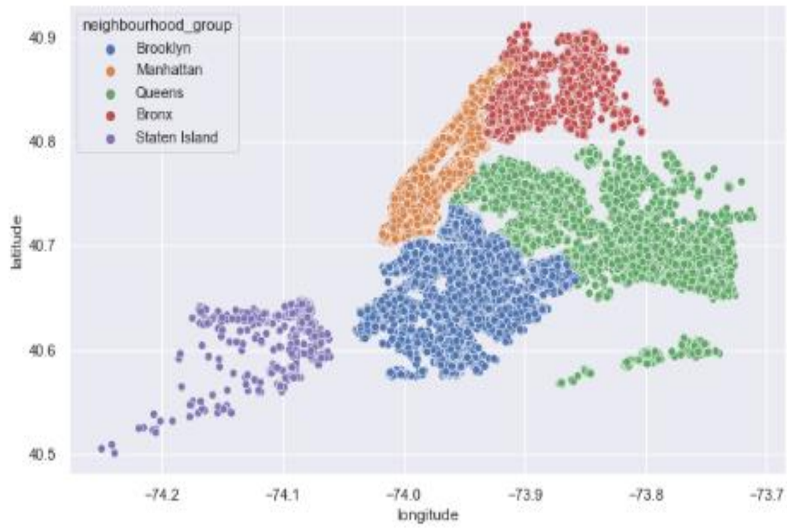


As anticipated, hotel rooms have the mean price at the lowest end of the spectrum, while entire homes command the highest mean price. The spread of prices for all room types appears to be consistent, with private rooms and shared rooms exhibiting greater centrality around their respective means. In contrast, entire homes exhibit a wider range of prices. To provide a clearer illustration of our findings, we can utilize Plotly to map the distribution of Airbnb prices in New York City. This visualization enables us to observe the geographical distribution of premium and budget-friendly Airbnb options in the city.

```
plt.figure(figsize=(10,6))
sns.scatterplot(df.longitude,df.latitude,hue=df.neighbourhood_group)
plt.ioff()
```

```
C:\Users\Aishat\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

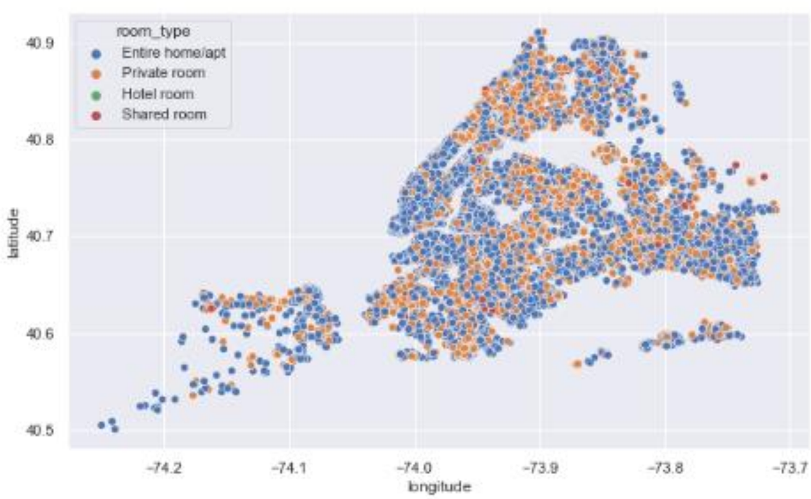
```
<matplotlib.pyplot._IoffContext at 0x18dc67c6940>
```




```
plt.figure(figsize=(10,6))
sns.scatterplot(df.longitude,df.latitude,hue=df.room_type)
plt.ioff()
```

```
C:\Users\Aishat\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```

```
<matplotlib.pyplot._IoffContext at 0x18dc6834a00>
```

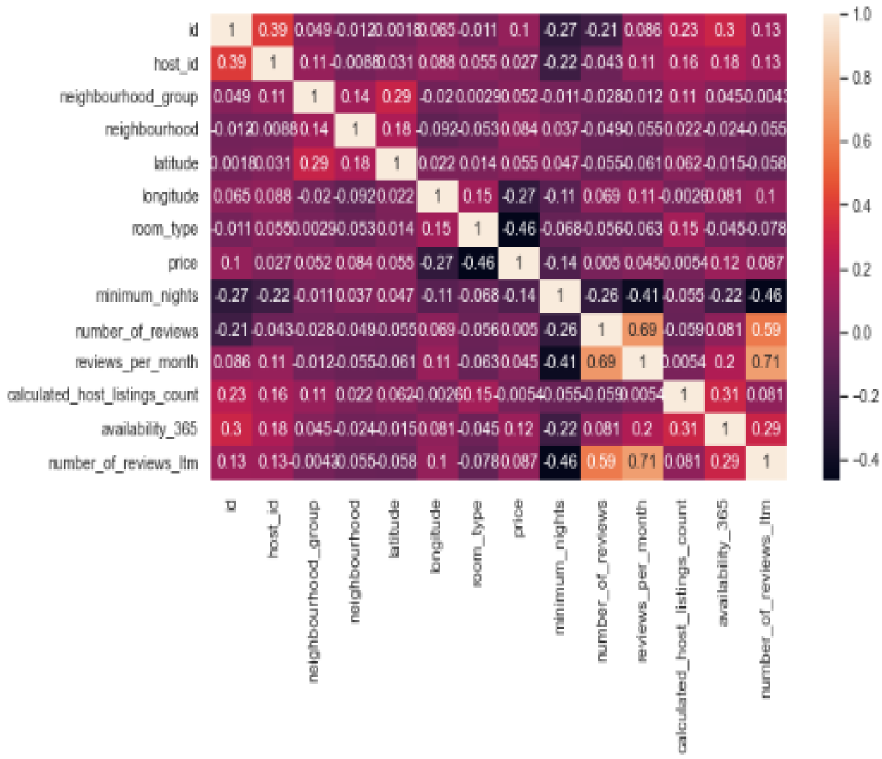


```
df['neighbourhood'].value_counts().iloc[:5]
```

```
Bedford-Stuyvesant    2936
Williamsburg          2570
Harlem                1949
Midtown               1918
Bushwick              1752
Name: neighbourhood, dtype: int64
```

```
corr = df.corr(method='kendall')
plt.figure(figsize=(10,5))
sns.heatmap(corr, annot=True)
df.columns
```

```
Index(['id', 'name', 'host_id', 'host_name', 'neighbourhood_group',
       'neighbourhood', 'latitude', 'longitude', 'room_type', 'price',
       'minimum_nights', 'number_of_reviews', 'last_review',
       'reviews_per_month', 'calculated_host_listings_count',
       'availability_365', 'number_of_reviews_ltm', 'license'],
      dtype='object')
```



```python
plt.subplots(figsize=(25,15))
wordcloud = WordCloud(
                          background_color='white',
                          width=1920,
                          height=1080
                         ).generate(" ".join(df.neighbourhood))
plt.imshow(wordcloud)
plt.axis('off')
plt.savefig('neighbourhood.png')
plt.show()
```


```python
df.price.describe()
```

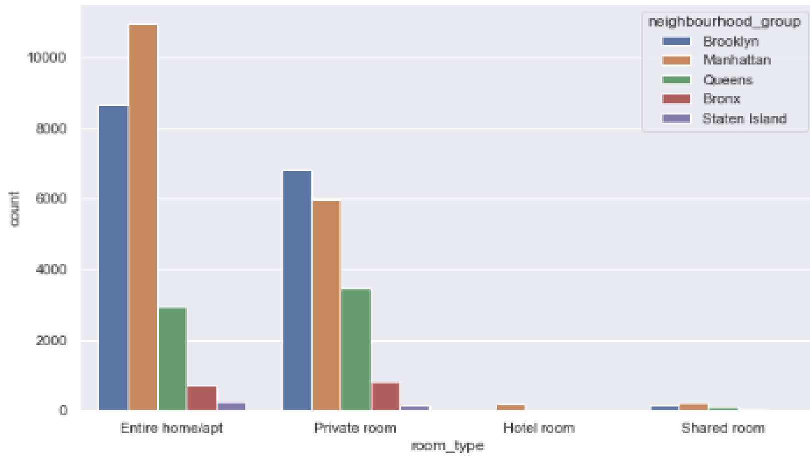
```
count    41533.000000
mean       221.978282
std        919.502236
min          0.000000
25%         80.000000
50%        131.000000
75%        220.000000
max      98159.000000
Name: price, dtype: float64
```

```python
df['minimum_nights'].value_counts()
```

```
30     17625
1       7202
2       5329
3       3780
5       1492
       ...
153        1
999        1
53         1
19         1
88         1
Name: minimum_nights, Length: 126, dtype: int64
```

```
plt.figure(figsize=(10,6))
sns.countplot(data = df, x = 'room_type', hue = 'neighbourhood_group')
```

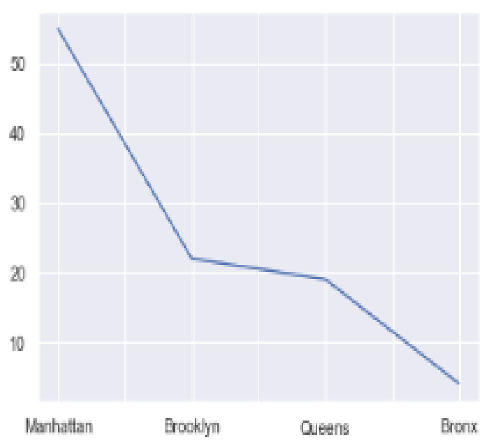
<AxesSubplot:xlabel='room_type', ylabel='count'>



The graph above shows the distribution of room type, and it shows that entire homes/apt had the highest in Manhattan and Brooklyn. The hotel room had the least room type occupied with the highest in Manhattan.

```
# Rooms with top 100 minimum_nights  by neighbourhood
dfr=df.sort_values(by=['minimum_nights'],ascending=False).head(100)
dfr['neighbourhood_group'].value_counts().plot()
```

<AxesSubplot:>



The line graph above shows that Manhattan has the highest Airbnb minimum nights followed by Brooklyn, then Queens, and lastly Bronx.

Step 4

Data Preparation

Prepare the data to better expose the underlying data patterns to Machine Learning algorithms.

Preparing the Data

```
import sklearn
from sklearn import preprocessing
from sklearn import metrics
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# from sklearn.ensemble import RandomForestRegressor,  GradientBoostingRegressor

```

```
df['reviews_per_month']=df['reviews_per_month'].replace(np.nan, 0)
```

```
'''Encode labels with value between 0 and n_classes-1.'''
le = preprocessing.LabelEncoder() # Fit label encoder
le.fit(df['neighbourhood_group'])
df['neighbourhood_group']=le.transform(df['neighbourhood_group']) # Transform labels to normalized encoding.
```

```
le = preprocessing.LabelEncoder()
le.fit(df['neighbourhood'])
df['neighbourhood']=le.transform(df['neighbourhood'])
```

```
le = preprocessing.LabelEncoder()
le.fit(df['room_type'])
df['room_type']=le.transform(df['room_type'])
```

```
df.sort_values(by='price',ascending=True,inplace=True)

df.head()
```

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_c
22006	43247386	6 Columbus Central Park Hotel	335072254	Six Columbus Central Park Hotel	2	97	40.767560	-73.983120	1	0	1	
21148	42065543	Broadway Plaza Hotel	307634016	Broadway Plaza	2	130	40.744440	-73.989200	1	0	1	
21167	42065563	Opera House Hotel	309772430	Opera House Hotel	0	137	40.815130	-73.916020	1	0	30	
21010	41740615	The James New York - NoMad	268417148	The James NoMad	2	130	40.744590	-73.985740	1	0	1	
24785	48417136	HGU New York	390810530	Marcos	2	130	40.746836	-73.982699	1	0	1	

To enhance the interpretability of the data for machine learning algorithms, I convert the categorical variables (neighbourhood_group, neighborhood, room type) into numerical labels using the LabelEncoder function from sci-kit-learn.

Step 5

Explore many different models and shortlist the best ones.

Training the Linear Regression Model

```
lm = LinearRegression()

X = df[['neighbourhood_group','neighbourhood','latitude','longitude','room_type','minimum_nights','number_of_reviews','revie
y = df['price']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)

lm.fit(X_train,y_train)
```

```
LinearRegression()
```

```
'''Get Predictions & Print Metrics'''
predicts = lm.predict(X_test)

print("""
        Mean Absolute Error: {}
        Root Mean Squared Error: {}
        R2 Score: {}
    """.format(
        mean_absolute_error(y_test,predicts),
        np.sqrt(metrics.mean_squared_error(y_test, predicts)),
        r2_score(y_test,predicts),
        ))
```

```
Mean Absolute Error: 155.2150720887138
Root Mean Squared Error: 824.0292262366524
R2 Score: 0.015463947121196364
```

The categorical variables (neighbourhood_group, neighborhood, room type) were encoded using LabelEncoder from sci-kit-learn to facilitate the interpretability of the data for machine learning algorithms. The regression model's predictions for the price were then compared to the actual prices through visualization. Evaluation of the model's performance was conducted using the metrics Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and R2 Score. The results showed an MAE of 155.22, an RMSE of 824.03, and an R2 Score of 0.01.

Step 6

Fined-tuned model.
Fine-tune your models and combine them into a great solution.

Finding a Better Model

Diagnostic Plots

```
# Residuals vs. Fitted
def r_v_fit(m):
    ax = sns.residplot(m.fittedvalues, m.resid)
    plt.title("Residuals vs. Fitted")
    plt.ylabel("Residuals")
    plt.xlabel("Fitted Values")
    plt.show()

# Residuals vs. Order
def r_v_order(m):
    ax = plt.scatter(m.resid.index, m.resid)
    plt.title("Residuals vs. Order")
    plt.ylabel("Residuals")
    plt.xlabel("Order")
    plt.show()

# Histogram
def r_hist(m, binwidth):
    resid = m.resid
    plt.hist(m.resid, bins=np.arange(min(resid), max(resid) + binwidth, binwidth))
    plt.title("Histogram of Residuals")
    plt.show()
```

```
# Get separate dataframe for statsmodels analysis
sm_df = pd.read_csv('listings.csv')

# Split data for training and testing
sm_df['logprice'] = np.log(1 + sm_df['price'])
train_data, test_data = train_test_split(sm_df, test_size=0.2)
```

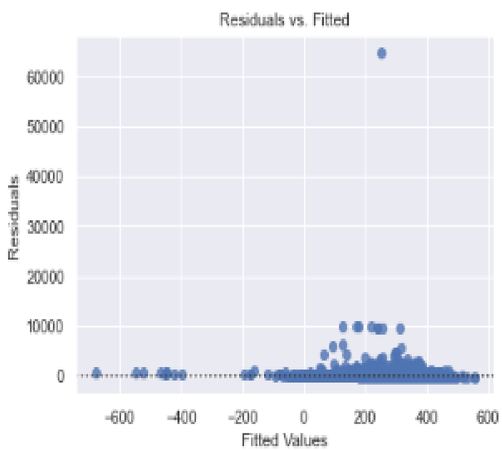
```
C:\Users\Aishat\anaconda3\lib\site-packages\IPython\core\interactiveshell.py:3444: DtypeWarning: Columns (17) have mixed types.
Specify dtype option on import or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

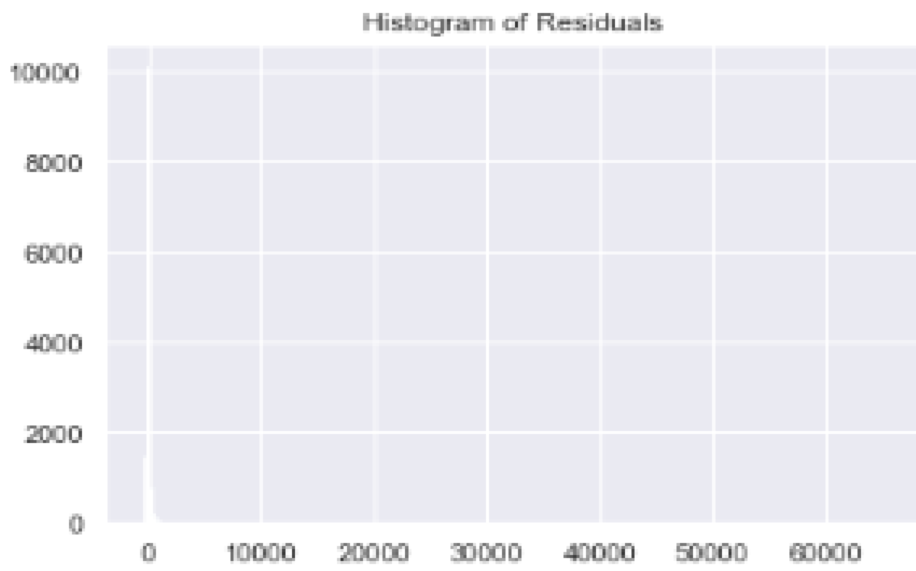
[39]:

```python
import statsmodels.formula.api as smf

# Create the model
model = smf.ols(
    'price ~ neighbourhood_group + latitude + longitude \
     + room_type + minimum_nights + number_of_reviews + reviews_per_month \
     + calculated_host_listings_count + availability_365',
    data=train_data).fit()

print("P-Value:\t{}".format(model.pvalues[0]))
print("R_Squared:\t{}".format(model.rsquared))
print("R_Squared Adj:\t{}".format(model.rsquared_adj))

# Diagnostic Plots for model
r_v_fit(model)
r_v_order(model)
r_hist(model, 100)
```

```
P-Value:        2.3983957336108874e-07
R_Squared:      0.029348150348848745
R_Squared Adj:  0.028818111894401532
```

```
C:\Users\Aishat\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```



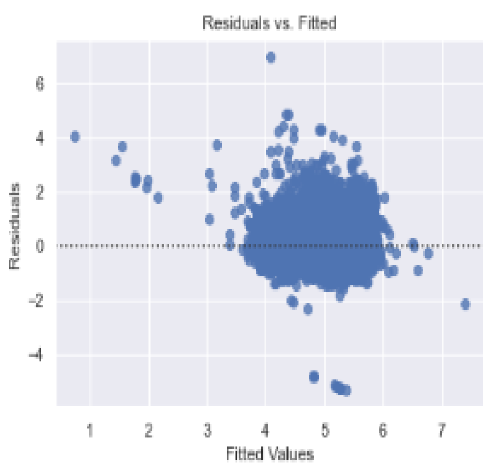


The residual plots highlight several shortcomings in the current model. The residuals vs. fitted plot display mostly positive residuals and one outlier. The residuals vs. Order plot similarly showcases a few outliers and predominantly positive values. Furthermore, the model's R-Squared value, as indicated above in the residual plots, is notably low.

```
# Fitting a new model with a log-transformed price
log_model = smf.ols(
    'logprice ~ neighbourhood_group + latitude + longitude \
     + room_type + minimum_nights + number_of_reviews + reviews_per_month \
     + calculated_host_listings_count + availability_365',
    data=train_data).fit()

print("P-Value:\t{}".format(log_model.pvalues[0]))
print("R_Squared:\t{}".format(log_model.rsquared))
print("R_Squared Adj:\t{}".format(log_model.rsquared_adj))

# Diagnostic Plots for new, transformed model
r_v_fit(log_model)
r_v_order(log_model)
r_hist(log_model, 0.1)
```

```
P-Value:        2.415923452386532e-75
R_Squared:      0.41215083547032016
R_Squared Adj:  0.4118298319480713
```

```
C:\Users\Aishat\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variables as keyword a
rgs: x, y. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explici
t keyword will result in an error or misinterpretation.
  warnings.warn(
```



s

The transformed model shows significant improvement compared to the original. The three diagnostic plots meet the assumptions required for linear regression and display a better distribution of residuals. Residuals vs. fitted plots have reduced the cone shape, while residuals vs. order have a better spread around the x-axis indicating independence of data. The histogram of residuals also demonstrates a more normal distribution. The R-Squared value, which measures the explanatory power of the model, is significantly higher in the transformed model. This implies that the transformed model provides a better explanation of the response variable (price).

The interpretation of model coefficients also changes with the new nonlinear relationship. A 1-unit increase in X corresponds to an increase of log(Y) by the coefficient value m, implying Y increases by a factor of e^m.

Steps 7 and 8

Present your solution, Launch, monitor, and maintain your system.

Reducing the Model

```
1 print(model.pvalues)
```

```
Intercept                               2.398396e-07
neighbourhood_group[T.Brooklyn]         1.499166e-02
neighbourhood_group[T.Manhattan]        2.747765e-02
neighbourhood_group[T.Queens]           9.193475e-01
neighbourhood_group[T.Staten Island]    6.240240e-01
room_type[T.Hotel room]                 1.002419e-02
room_type[T.Private room]               1.858923e-48
room_type[T.Shared room]                2.143726e-03
latitude                                2.953318e-04
longitude                               1.153103e-15
minimum_nights                          4.254730e-10
number_of_reviews                       4.760304e-03
reviews_per_month                       1.354570e-01
calculated_host_listings_count          3.357492e-02
availability_365                        5.049735e-15
dtype: float64
```

Since we now have a better model, I decided to examine it to see if any predictors may be removed from the model. I used the Statsmodels library to look at the p-value of each of the predictors to see how significant they were which is shown above.

we can see from the above output that predictors like neighbourhood_group[T.Manhattan], room_type[T.Private room], room_type[T.Shared room], latitude, longitude, minimum_nights, number_of_reviews, and availability_365 have p-values that are significant predictors of price. On the other hand, predictors like neighbourhood_group[T.Brooklyn], neighbourhood_group[T.Queens], neighbourhood_group[T.Staten Island], room_type[T.Hotel room], reviews_per_month, and calculated_host_listings_count have p-values that are higher than 0.05, indicating they are not statistically significant in predictors of price.

```python
log_model_1 = smf.ols(
    'logprice ~ neighbourhood_group + latitude + longitude \
     + room_type + minimum_nights + reviews_per_month \
     + calculated_host_listings_count + availability_365',
    data=train_data).fit()

print("P-Value:\t{}".format(log_model_1.pvalues[0]))
print("R_Squared:\t{}".format(log_model_1.rsquared))
print("R_Squared Adj:\t{}".format(log_model_1.rsquared_adj))
```

```
P-Value:        1.760025933426194e-73
R_Squared:      0.4102543016518351
R_Squared Adj:  0.4099552769598219
```

```python
model.params
```

```
Intercept                               -40225.814826
neighbourhood_group[T.Brooklyn]            -54.396291
neighbourhood_group[T.Manhattan]            42.603814
neighbourhood_group[T.Queens]               -2.080467
neighbourhood_group[T.Staten Island]       -20.835865
room_type[T.Hotel room]                    119.326826
room_type[T.Private room]                  -93.080603
room_type[T.Shared room]                   -86.719475
latitude                                  -317.243355
longitude                                 -721.752335
minimum_nights                              -0.678888
number_of_reviews                           -0.165764
reviews_per_month                            3.035335
calculated_host_listings_count              -0.169229
availability_365                             0.179562
dtype: float64
```

The result of the code is the estimated coefficients of the linear regression model, which represent the average change in the logarithm of the response (price) for a one-unit change in each predictor, holding all other predictors constant. The coefficient of each predictor can be interpreted as its effect on the price, with positive values indicating an increase in price and negative values indicating a decrease in price. The magnitude of the coefficients can also be used to compare the relative importance of each predictor in explaining the response.

The most sensitive coefficients are longitude, latitude, room_type[T.Private room], room_type[T.Shared room], and neighbourhood_group[T.Brooklyn]. Coincidentally, they are all negatively correlated with price, which intuitively makes sense.

The features most positively correlated with price are room_type[T.Hotel room], neighbourhood_group[T.Manhattan], reviews_per_month.

Link to my code.