Machine Learning, Development Team Project Group 1 -

1. Introduction

Airbnb is an online marketplace that allows individuals to either host a place for travelers to stay, or book a place to stay with a host. Over the past several years, millions of tourists have used the service (Guttentag et al., 2017). But what exactly influences a traveler's choice in where they stay? Knowing why a traveler books a listing is important for hosts to optimize their airbnb rental unit, leading to greater revenue for Airbnb and the host, as well as increased satisfaction for all sides: the traveler, the host, and Airbnb themselves. It may also encourage potential hosts that have rental units that fit the ideal standards for travelers to increase the availability of the unit, leading to more traffic for Airbnb's services. Using data from the listing activity and metrics in New York City for 2019, this report attempts to answer the following: how do various factors influence the likelihood of a listing being booked?

2. Exploratory Data Analysis & Data Pre-processing

Raw experimental data is highly susceptible to noise, missing values, and inconsistencies, making initial data cleaning a critical step before any meaningful analysis can occur (Calabrese 2018). Our first task involved exploring and cleaning the dataset to ensure it was suitable for addressing our business question. As illustrated in Figure 1 of the code, we began by identifying the variables and rows within the dataset. Utilizing the pandas library, we found that the dataset comprised 16 variables and 48,895 entries. To delve deeper into the variables, we employed the describe command to generate summary statistics (Figure 2), providing us with an overview of the data's central tendencies and variability. Additionally, we used the 'isnull' command to pinpoint missing values, discovering that four variables contained null entries. To address these missing values, we either removed the affected rows or replaced the missing values with 0, as shown in Figure 3. This thorough data cleaning process was indispensable for ensuring the accuracy and reliability of our subsequent analysis and findings.

Figures 4 through 7 present various visualizations that provide insights into the dataset's composition. These figures illustrate the count of different room types, the distribution of listings across neighborhoods and neighborhood groups, and the number of room reviews. Notably, 'entire home/apartment' emerges as the most frequently listed room type, with over 600,000 reviews. Manhattan stands out as the area with the highest number of listings, totaling 20,000, with Harlem contributing 2,500 of these listings. Furthermore, Figure 8 highlights the median price analysis, revealing that Manhattan is the most expensive neighborhood group, with an average booking cost exceeding $140. These visualizations not only help in understanding the dataset's structure but also provide a foundation for more detailed data analysis and strategic decision-making based on the identified trends and patterns.

3. Machine learning

Machine learning (ML) is an integral part of modern data science, allowing meaningful insights and predictions to be made from vast amounts of data. In order to create a ML algorithm, first the normalizer from the sklearn.preprocessing module was used to scale the features by adjusting each feature vector to have a length of one. This ensured that each data point contributed equally to the model and that features with larger magnitudes didn't have larger

impacts on the future model and analysis (Verma (2023). After preprocessing, the data was split into training and testing sets. This split allows the model to be trained on one subset of the data and evaluated on another, ensuring that the model's performance can be generalized to new, unseen data (Bishop, 2006). Using this process, we reassured that the model does not overfit the training data, increasing the reliability of its predictions.

In the context of clustering - an unsupervised learning technique - K-Means clustering is applied using KMeans from sklearn.cluster. This algorithm partitions the data into k clusters, where each data point belongs to the cluster with the nearest mean (MacQueen, 1967). Figures 13 and 14 display the output of the initial K-Means clustering algorithm, where the groups show the booking likelihood based on the location of the listing. After the algorithm was created, a test was done to verify which K-value, or the number of groups that can be created without overfitting, would be best (Figure 15). After updating the algorithm to reflect the new K-value, the group differences were more pronounced, with one group having a much higher booking likelihood than the other groups (Figures 16 and 17). Evaluating the quality of the clusters is done using the silhouette score, with higher silhouette scores indicating better-defined clusters (Rousseeuw, 1987). In this case, the silhouette score is 0.563, suggesting that the clusters are well-defined, distinct and the chosen number of clusters is appropriate for the data. K-Means was a good choice for our analysis due to its ability for simple implementation as well as being computationally efficient. By grouping the similar listings together, it allowed us to analyze the complex data and derive actionable insights. Other clustering methods such as Hierarchical clustering were considered but were opted against due to their impracticality with larger datasets, and their being computationally more expensive and less efficient.

4. Conclusion

Our analysis determined that several key factors significantly influence the likelihood of an Airbnb listing being booked. By examining data, we identified that price, number of reviews, location, and property type are crucial determinants of booking likelihood. Specifically, 'entire home/apartment' emerged as the most popular room type, and Manhattan was found to be both the most listed and the most expensive neighborhood group. For Airbnb, understanding these factors is critical for developing strategies that attract more hosts and guests, thereby boosting overall platform engagement and revenue. The insights from our analysis can guide Airbnb in tailoring their services and marketing efforts to align with market trends and traveler preferences. Future Airbnb hosts can leverage these insights to optimize their listings by strategically setting competitive prices, improving the quality and appeal of their property types, and accumulating positive reviews to enhance their booking likelihood. For travelers, the analysis provides a clearer understanding of what influences booking likelihood, helping them make better-informed decisions when choosing accommodations.

References

Guttentag, D.A., Smith, S.L.J., Potwarka, L.R., & Havitz, M.E. (2017) 'Why tourists choose Airbnb: A motivation-based segmentation study underpinned by innovation concepts', Journal of Travel Research, 56(3), pp. 396-409.

Calabrese, B. (2018) 'Data cleaning', in Ranganathan, S., Gribskov, M., Nakai, K., & Schönbach, C. (eds.) Encyclopedia of bioinformatics and computational biology: ABC of bioinformatics. 1st edn. Oxford: Elsevier, pp. 472-481.

Bishop, C.M. (2006) Pattern recognition and machine learning. New York: Springer.

MacQueen, J.B. (1967) 'Some methods for classification and analysis of multivariate observations', in Le Cam, L.M. & Neyman, J. (eds.) Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Berkeley: University of California Press, Vol. 1, pp. 281-297.

Rousseeuw, P.J. (1987) 'Silhouettes: A graphical aid to the interpretation and validation of cluster analysis', Journal of Computational and Applied Mathematics, 20, pp. 53-65.

Verma (2023) https://www.digitalocean.com/community/tutorials/normalize-data-in-python

Appendices

Notebook file, 'Project_Group_1 NB.ipynb' will be provided for easier reading.

Machine Learning - Project Report, Air B&B The assignment involves conducting a business analysis for Airbnb using the 2019 NYC dataset from Kaggle. The dataset includes detailed information about hosts, geographic availability, and various metrics necessary for making predictions and drawing conclusions. The goal is to assess the ability to pose meaningful business questions, process the data through key steps of big data analytics (such as data pre-processing, exploratory data analysis, statistical analysis, data visualization, and using unsupervised machine learning algorithms), and prepare a concise analytical report for Airbnb's executive board.

Business Question = How do various factors (e.g., price, number of reviews, location, property type) influence the likelihood of a listing being booked?

```
#Importing relevant libraries
import pandas as pd
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

#read the csv from GitHub into the variable df
path = "https://cfinal15.github.io/Machine%20Learning/AB_NYC_2019.csv"
df = pd.read_csv(path)

#Viewing Dating
print("Figure 1 - Describing the initial data")
print("Number of columns in data: ", len(df.columns))
print("Number of entries in data: ", len(df))
df.head()
```

Figure 1 - Describing the initial data
Number of columns in data: 16
Number of entries in data: 48895

	id	name	host_id	host_name	neighbourhood_group	neighbourhood
0	2539	Clean & quiet apt home by the park	2787	John	Brooklyn	Kensington
1	2595	Skylit Midtown Castle	2845	Jennifer	Manhattan	Midtown
2	3647	THE VILLAGE OF HARLEM....NEW YORK !	4632	Elisabeth	Manhattan	Harlem
3	3831	Cozy Entire Floor of Brownstone	4869	LisaRoxanne	Brooklyn	Clinton Hill
4	5022	Entire Apt: Spacious Studio/Loft by central park	7192	Laura	Manhattan	East Harlem

```
#Describing the variables and statistical outputs
print("Figure 2 - Summarising data")
print(df.describe())
```

```
Figure 2 - Summarising data
                 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
```

```
df.isnull().sum() #lets check the null values in each column
```

```
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
```

```
#Lets remove the null data in name, last review. Replace reviews per month with 0 if null
df = df[pd.notnull(df['name'])]
df = df[pd.notnull(df['last_review'])]
df = df[pd.notnull(df['host_name'])]
df.fillna({'reviews_per_month':0}, inplace=True)
#Figure 3 - Replacing Null values
print("Figure 3 - Replacing Null values")
df.isnull().sum()
```

```
Figure 3 - Replacing Null values
id                              0
name                            0
host_id                         0
host_name                       0
neighbourhood_group             0
neighbourhood                   0
latitude                        0
longitude                       0
room_type                       0
price                           0
minimum_nights                  0
number_of_reviews               0
last review                     0
```

```
#bookings per room type : Entire homes/apt has the most number of bookings
room_types = df.room_type.value_counts()
room_types

#create function to be called later
def plot_room_types(ax, room_types):
    sns.barplot(x=room_types.index, y=room_types.values, ax=ax)
    ax.set_title('Figure 4 - Count of Room Types')
    ax.set_xlabel('Room Type')
    ax.set_ylabel('Count')
```

```
#which neighbourhood has the most hosts
NB_Count=df.groupby(by=['neighbourhood']).neighbourhood.count()
NB_Count = NB_Count.sort_values(ascending=False)
print(NB_Count) #Williamsburg with the most

# Select top 10 neighborhoods
top_10_NB_Count = NB_Count.head(10)

#create function to be called later
def plot_neighbourhood_counts(ax, NB_Count):
    sns.barplot(x=NB_Count.index, y=NB_Count.values, ax=ax)
    ax.set_title('Figure 5 - Count of Listings by Neighbourhood Top 10')
    ax.set_xlabel('Neighbourhood')
    ax.set_ylabel('Count')
    ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

```
neighbourhood
Williamsburg          3163
Bedford-Stuyvesant    3141
Harlem                2204
Bushwick              1942
Hell's Kitchen        1528
```

```
#Which neighbourhood_group is the biggest one?
NB_Group_Count=df.groupby(by=['neighbourhood_group']).neighbourhood_group.count()
NB_Group_Count = NB_Group_Count.sort_values(ascending=False)
print(NB_Group_Count) #Manhattan contains the most

#create function to be called later
def plot_neighbourhood_group_counts(ax, NB_Group_Count):
    sns.barplot(x=NB_Group_Count.index, y=NB_Group_Count.values, ax=ax)
    ax.set_title('Figure 7 - Count of Listings by Neighbourhood Groups')
    ax.set_xlabel('Neighbourhood Group')
    ax.set_ylabel('Count')
    ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

```
neighbourhood_group
Manhattan        16621
Brooklyn         16439
Queens            4572
Bronx              875
Staten Island      314
Name: neighbourhood_group, dtype: int64
```

```
#Which room_type have more reviews?
Room_Reviews=df.groupby(by=['room_type']).number_of_reviews.sum()
Room_Reviews = Room_Reviews.sort_values(ascending=False)
print(Room_Reviews)

#create function to be called later
def plot_room_reviews(ax, Room_Reviews):
    sns.barplot(x=Room_Reviews.index, y=Room_Reviews.values, ax=ax)
    ax.set_title('Figure 6 - Count of Room Reviews')
    ax.set_xlabel('Type of room')
    ax.set_ylabel('Count')
    ax.set_xticklabels(ax.get_xticklabels(), rotation=90)
```

```
room_type
Entire home/apt    579856
Private room       537965
Shared room         19256
Name: number_of_reviews, dtype: int64
```

```
# Create a figure and a 2x2 grid of subplots
fig, axes = plt.subplots(2, 2, figsize=(14, 14))

# Plot the graphs in the grid layout
plot_room_types(axes[0, 0], room_types)
plot_neighbourhood_counts(axes[0, 1], top_10_NB_Count)
plot_room_reviews(axes[1, 0], Room_Reviews)
plot_neighbourhood_group_counts(axes[1, 1], NB_Group_Count)

# Adjust layout to prevent overlap
plt.tight_layout()
plt.show()
```

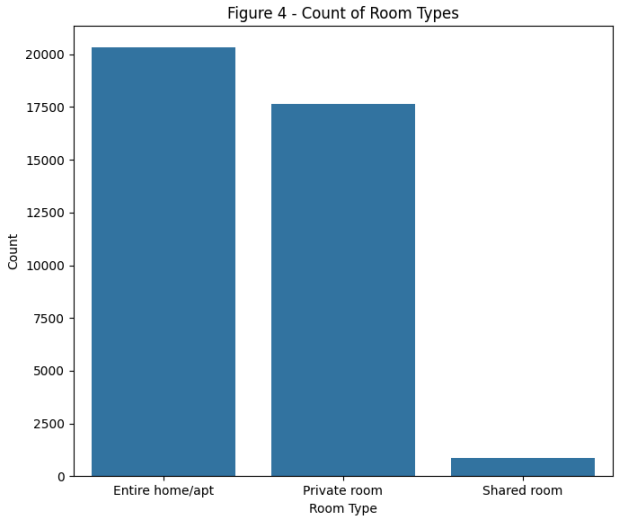


Figure 4 - Count of Room Types
Count
0
2500
5000
7500
10000
12500
15000
17500
20000
Entire home/apt
Private room
Shared room
Room Type

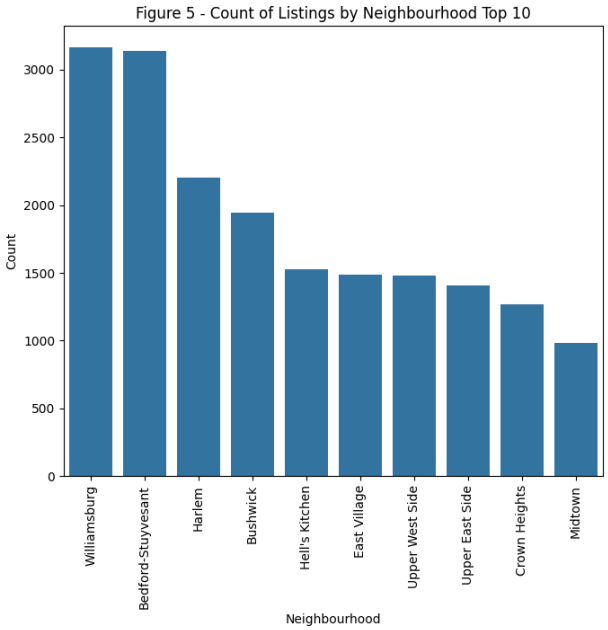


Figure 5 - Count of Listings by Neighbourhood Top 10
Count
0
500
1000
1500
2000
2500
3000
Williamsburg
Bedford-Stuyvesant
Harlem
Bushwick
Hell's Kitchen
East Village
Upper West Side
Upper East Side
Crown Heights
Midtown
Neighbourhood

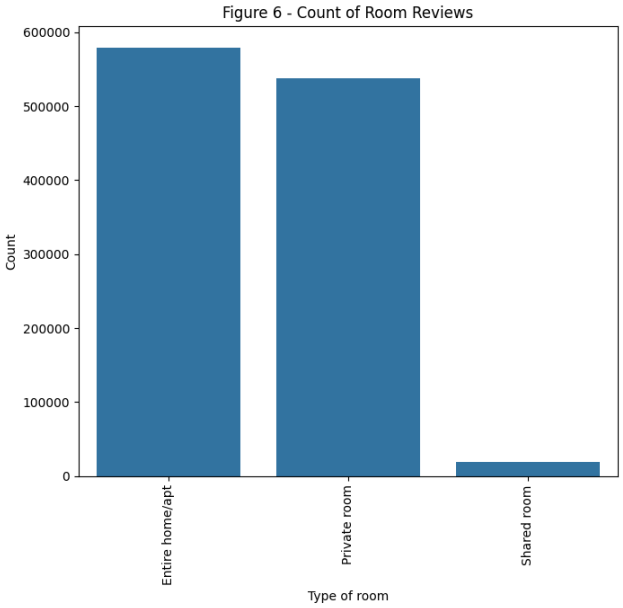


Figure 6 - Count of Room Reviews
Count
0
100000
200000
300000
400000
500000
600000
Entire home/apt
Private room
Shared room
Type of room

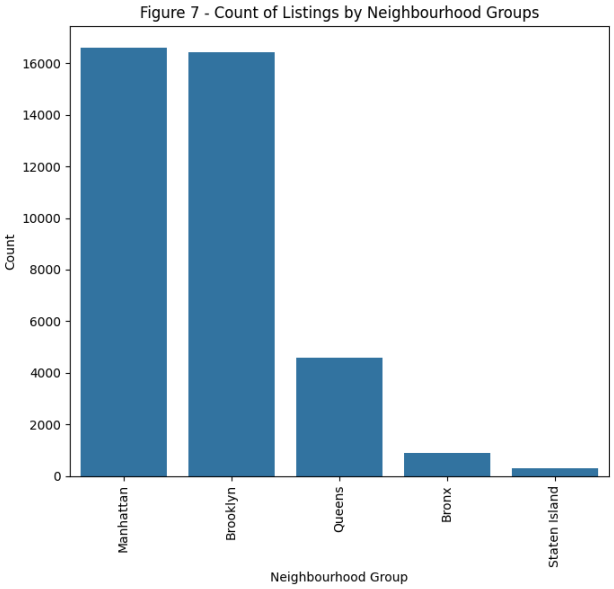
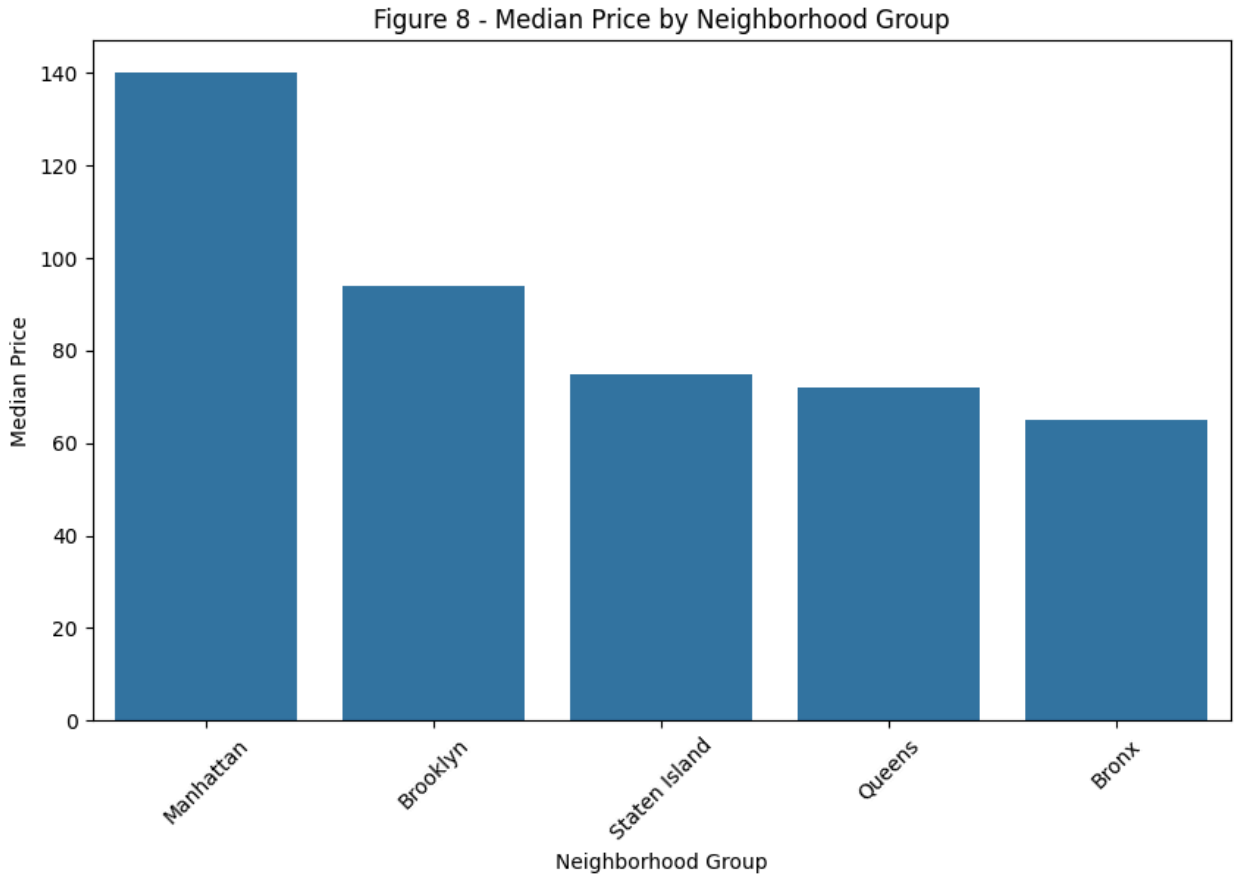


Figure 7 - Count of Listings by Neighbourhood Groups
Count
0
2000
4000
6000
8000
10000
12000
14000
16000
Manhattan
Brooklyn
Queens
Bronx
Staten Island
Neighbourhood Group

```
#Which neighbourhood_group is the most expensive?
#using median over mean due price not being normally distributed
NB_Group_Price=df.groupby(by=['neighbourhood_group']).price.median()
NB_Group_Price = NB_Group_Price.sort_values(ascending=False)
print(NB_Group_Price) #Manhattan most expensive

#Which neighborhood has the most expensive?
NB_Price=df.groupby(by=['neighbourhood']).price.median()
NB_Price = NB_Price.sort_values(ascending=False)
print(NB_Price) #Tribecca most expensvie

#Plotting most median prices of neighbourhood groups
plt.figure(figsize=(10, 6))
sns.barplot(x=NB_Group_Price.index, y=NB_Group_Price.values)
plt.title('Figure 8 - Median Price by Neighborhood Group')
plt.xlabel('Neighborhood Group')
plt.ylabel('Median Price')
plt.xticks(rotation=45)  # Rotate x-axis labels if needed
plt.show()
```

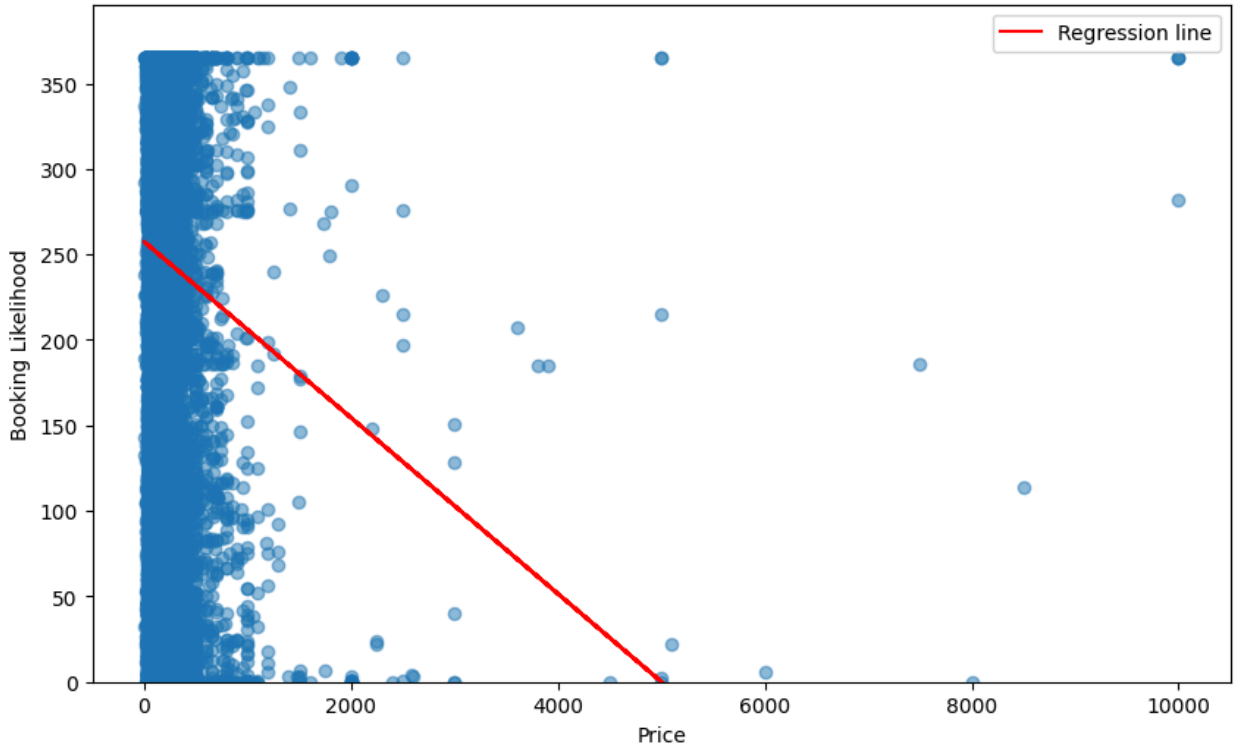


```
# Create 'booking_likelihood' feature -  This can be represented by the availability of the listing.
# Listings with lower availability are more likely to be frequently booked

df['booking_likelihood'] = 365 - df['availability_365']

# Display the first few rows to verify the new feature
df[['availability_365', 'booking_likelihood']].head()
```

	availability_365	booking_likelihood
0	365	0
1	355	10
3	194	171
4	0	365
5	129	236

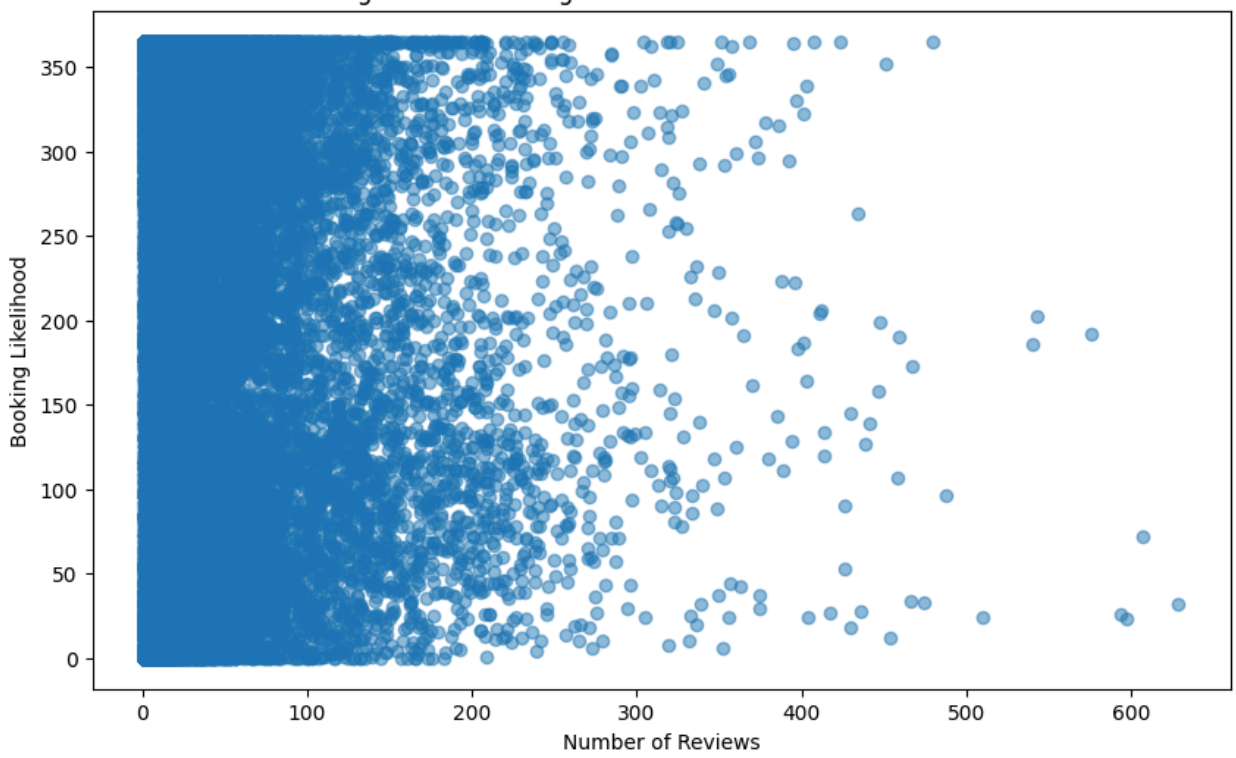
```
# Relationship between booking_likelihood and price.
from sklearn.linear_model import LinearRegression
X = df['price'].values.reshape(-1, 1)
y = df['booking_likelihood'].values
model = LinearRegression()
model.fit(X, y)

# Predict values for the regression line
y_pred = model.predict(X)

plt.figure(figsize=(10, 6))
plt.scatter(df['price'], df['booking_likelihood'], alpha=0.5)
plt.plot(df['price'], y_pred, color='red', label='Regression line')
plt.title('Figure 9 - Booking Likelihood vs Price')
plt.xlabel('Price')
plt.ylabel('Booking Likelihood')
plt.legend()
plt.ylim(bottom=0)  # Ensure the y-axis only shows positive values
plt.show()

print(f"Regression Coefficient: {model.coef_[0]}")
print(f"Regression Intercept: {model.intercept_}")
```

Figure 9 - Booking Likelihood vs Price

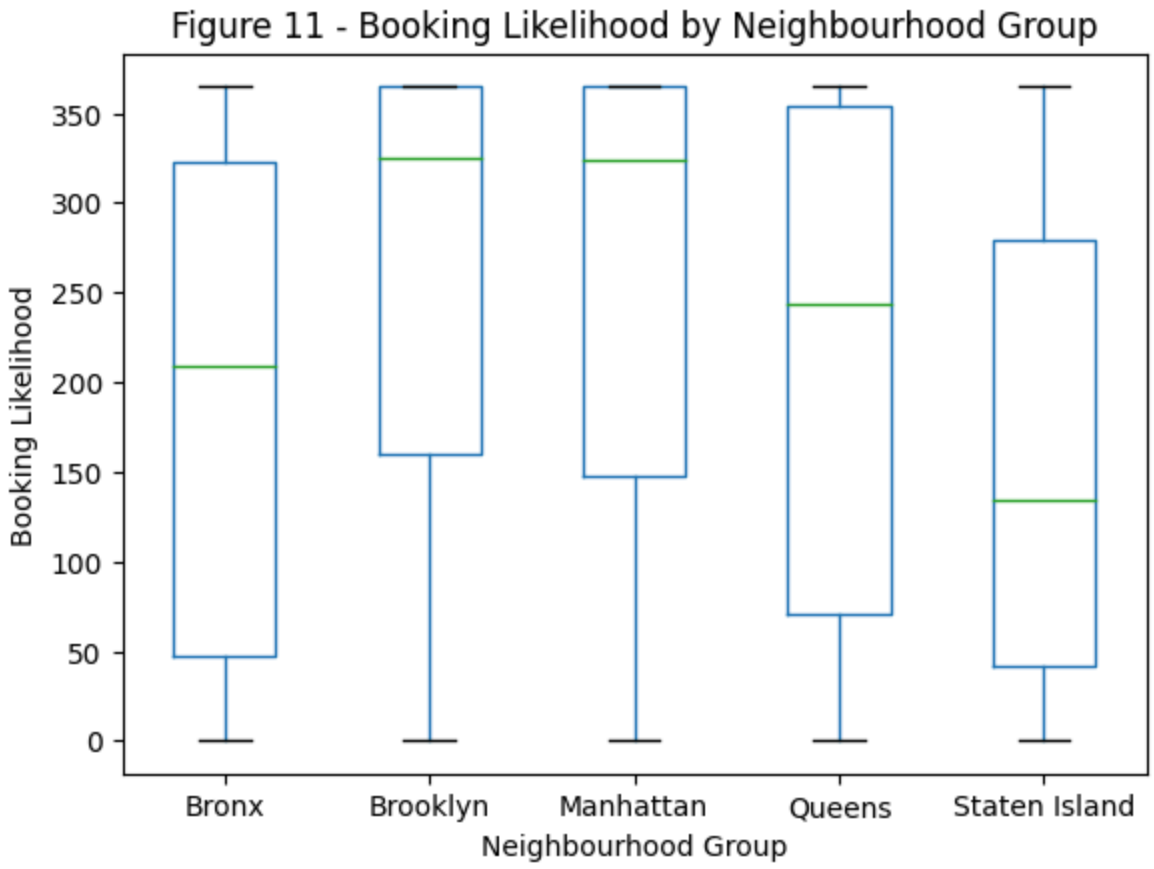


```
# Scatter plot of 'booking_likelihood' vs 'number_of_reviews'

plt.figure(figsize=(10, 6))
plt.scatter(df['number_of_reviews'], df['booking_likelihood'], alpha=0.5)
plt.title('Figure 10 - Booking Likelihood vs Number of Reviews')
plt.xlabel('Number of Reviews')
plt.ylabel('Booking Likelihood')
plt.show()
```

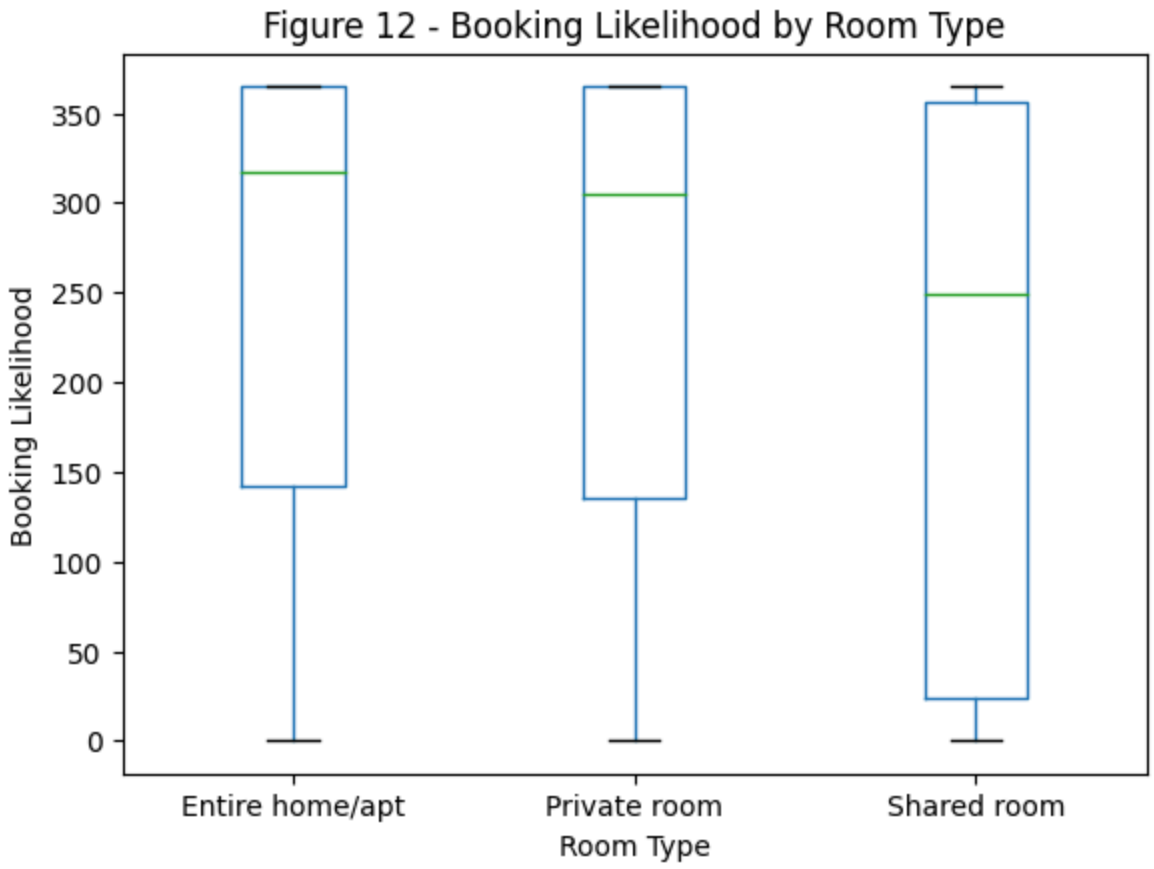
Figure 10 - Booking Likelihood vs Number of Reviews



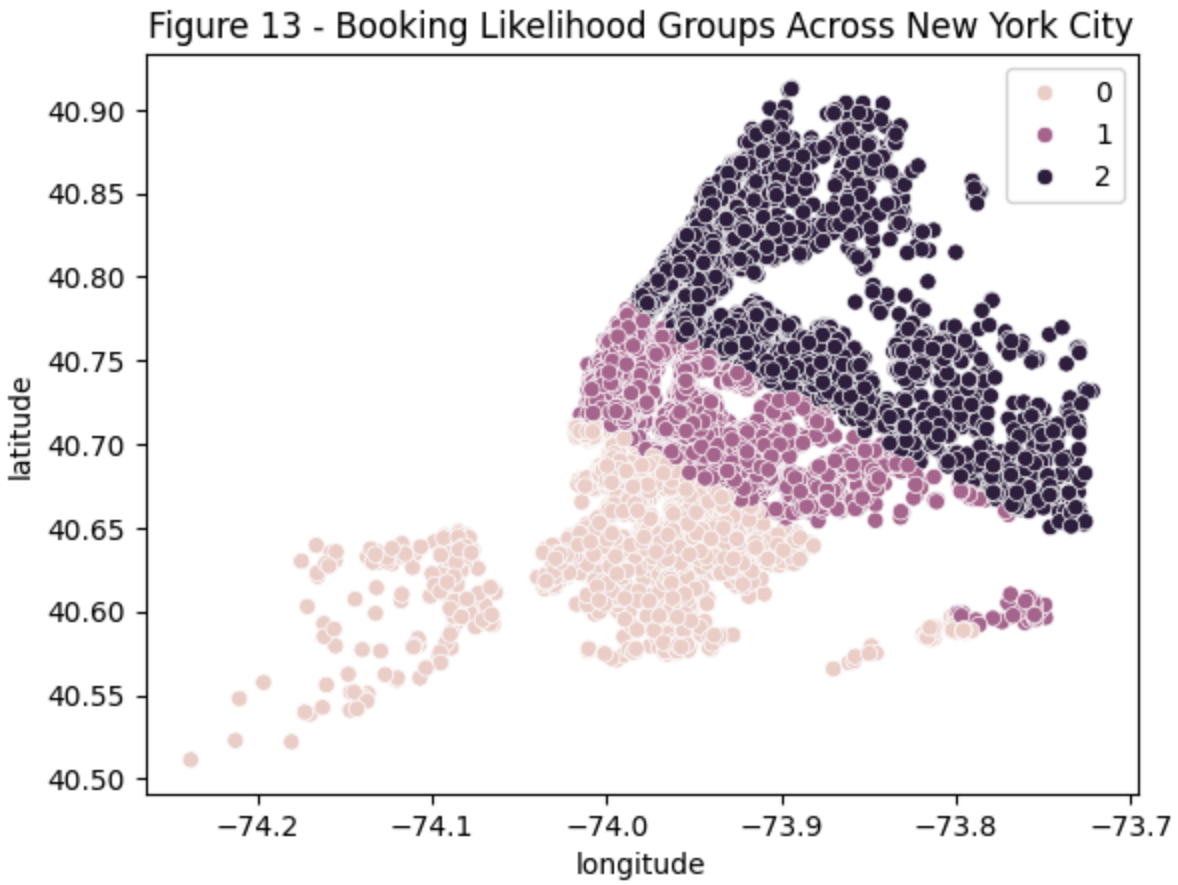

```
# Box plot of 'booking_likelihood' by 'neighbourhood_group'
plt.figure(figsize=(12, 6))
df.boxplot(column='booking_likelihood', by='neighbourhood_group', grid=False)
plt.title('Figure 11 - Booking Likelihood by Neighbourhood Group')
plt.suptitle('')  # Suppress the default title to keep it clean
plt.xlabel('Neighbourhood Group')
plt.ylabel('Booking Likelihood')
plt.show()
```




```
# Box plot of 'booking_likelihood' by 'room_type'
plt.figure(figsize=(10, 6))
df.boxplot(column='booking_likelihood', by='room_type', grid=False)
plt.title('Figure 12 - Booking Likelihood by Room Type')
plt.suptitle('')  # Suppress the default title to keep it clean
plt.xlabel('Room Type')
plt.ylabel('Booking Likelihood')
plt.show()
```




```
# First, we set up training and test splits

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df[['longitude', 'latitude']], df[['booking_likelihood']], test_size=0.33, random_state=0)

# Normalize the training and test data

from sklearn import preprocessing

x_train_norm = preprocessing.normalize(x_train)
x_test_norm = preprocessing.normalize(x_test)

# Fitting and evaluating the model

from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters = 3, random_state = 0, n_init='auto')
kmeans.fit(x_train_norm)

# Visualizing the data we fit

import seaborn as sns

sns.scatterplot(data = x_train, x = 'longitude', y = 'latitude', hue = kmeans.labels_).set_title('Figure 13 - Booking Likelihood Groups Across New York City')
```




```
# Further visualizing the data

sns.boxplot(x = kmeans.labels_, y = y_train['booking_likelihood']).set_title('Figure 14 - Booking Likelihood by Group')
```

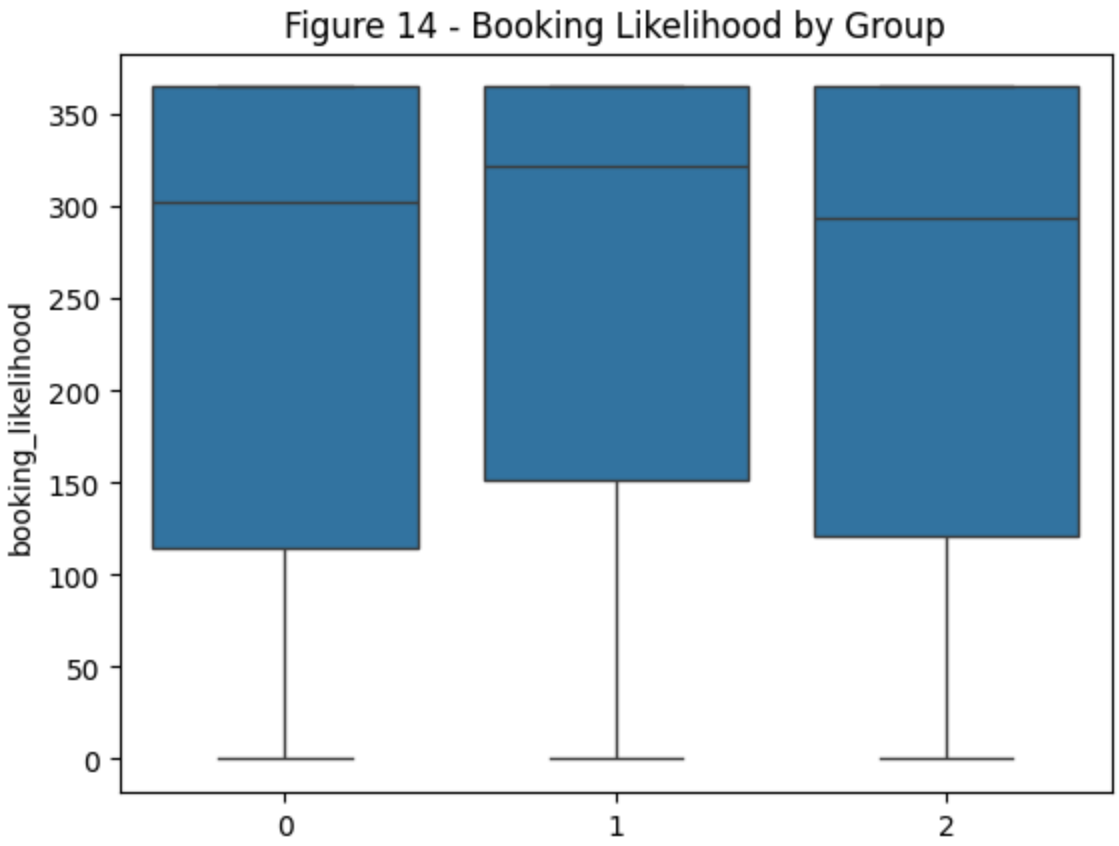


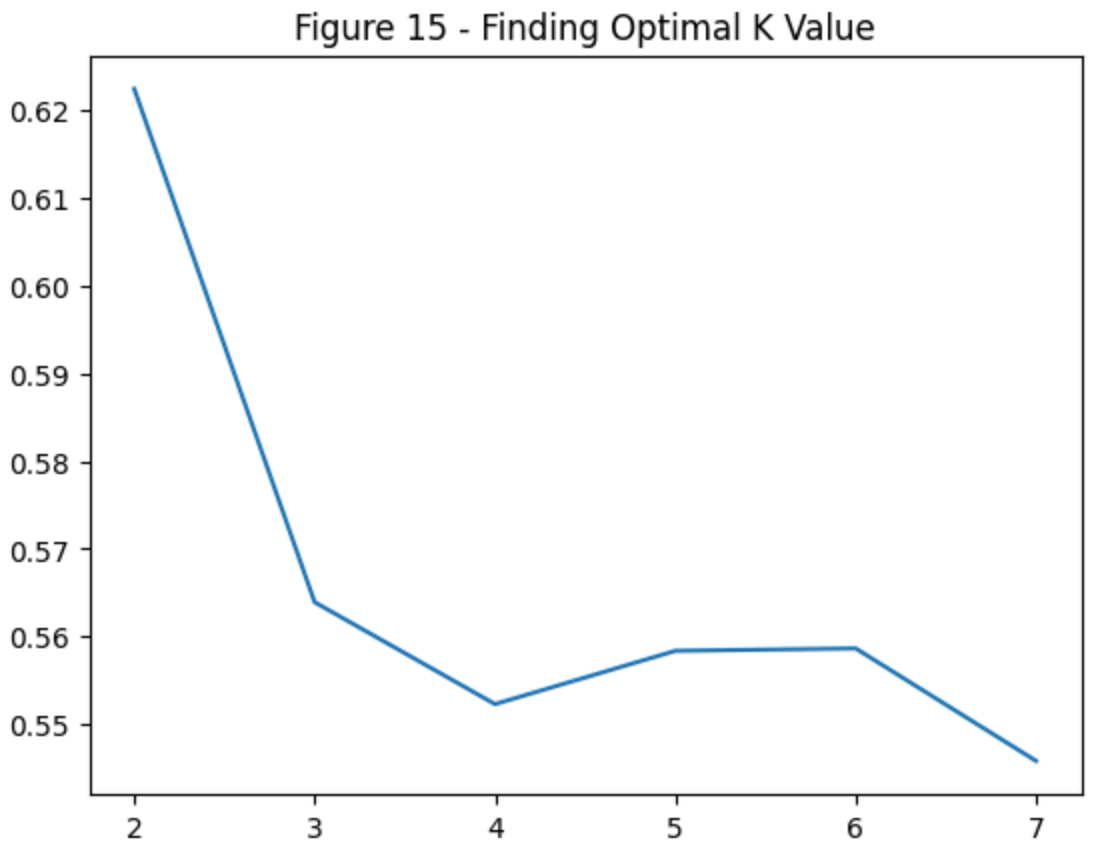
Figure 14 - Booking Likelihood by Group
booking_likelihood
350
300
250
200
150
100
50
0
0
1
2

```
# Evaluating the performance of the clustering algorithm

from sklearn.metrics import silhouette_score

silhouette_score(x_train_norm, kmeans.labels_, metric='euclidean')
```

```
0.5639260093491838
```

```
# Choosing the best number of clusters, as 3 might not be optimal!

K = range(2, 8)
fits = []
score = []


for k in K:
    # train the model for current value of k on training data
    model = KMeans(n_clusters = k, random_state = 0, n_init='auto').fit(x_train_norm)

    # append the model to fits
    fits.append(model)

    # Append the silhouette score to scores
    score.append(silhouette_score(x_train_norm, model.labels_, metric='euclidean'))
```

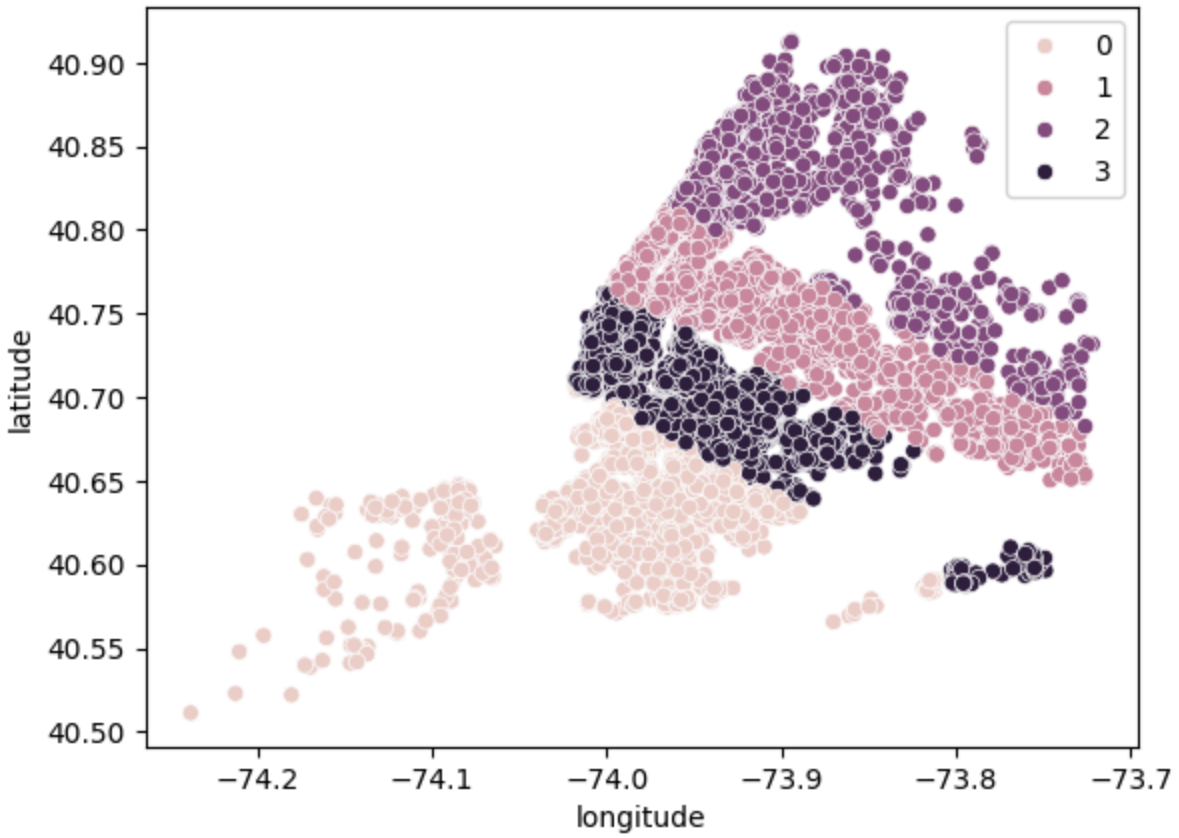
```
# Testing which value of k is best

sns.lineplot(x = K, y = score).set_title('Figure 15 - Finding Optimal K Value')
```



k = 4 is probably the best we can do without overfitting.

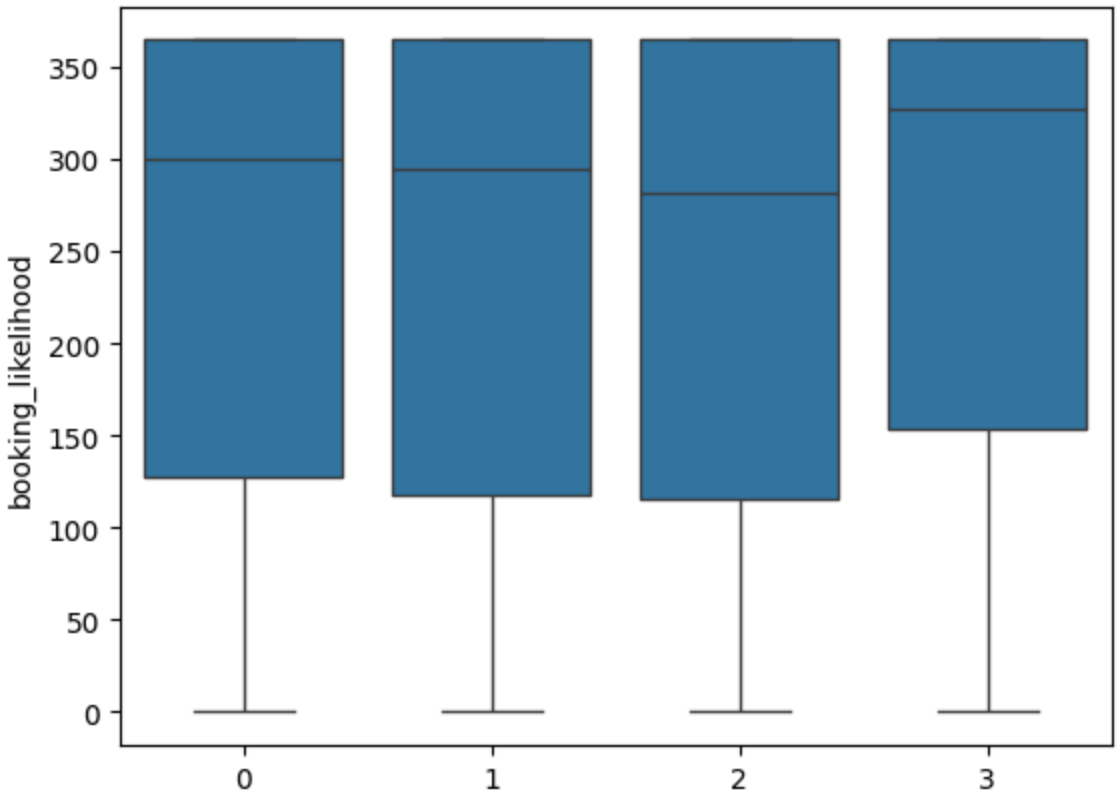
```
[ ]  # Visualizing k = 4

     sns.scatterplot(data = x_train, x = 'longitude', y = 'latitude', hue = fits[2].labels_).set_title('Figure 16 - Updated Booking Likelihood Groups Across New York
```

Figure 16 - Updated Booking Likelihood Groups Across New York City



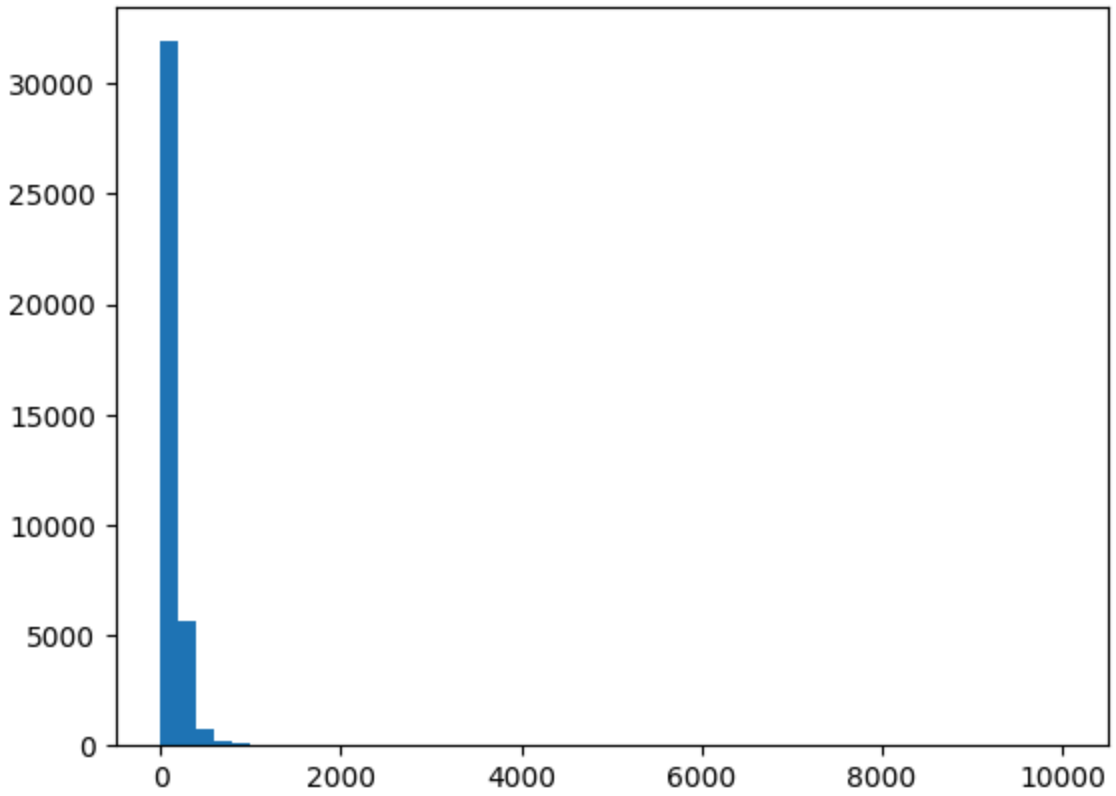

```
# Boxplot for k = 4

sns.boxplot(x = fits[2].labels_, y = y_train['booking_likelihood']).set_title('Figure 17 - Updated Booking Likelihood by Group')
```

Figure 17 - Updated Booking Likelihood by Group




```
#histogram of price to show non-normal distribution of data
plt.hist(df.price, bins=50)
```

```
(array([3.1907e+04, 5.6540e+03, 7.6400e+02, 2.4600e+02, 1.1300e+02,
        5.1000e+01, 1.4000e+01, 1.6000e+01, 3.0000e+00, 3.0000e+00,
        1.5000e+01, 4.0000e+00, 7.0000e+00, 1.0000e+00, 2.0000e+00,
        3.0000e+00, 0.0000e+00, 0.0000e+00, 1.0000e+00, 2.0000e+00,
        0.0000e+00, 0.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00,
        6.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        1.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00,
        1.0000e+00, 0.0000e+00, 1.0000e+00, 0.0000e+00, 0.0000e+00,
        0.0000e+00, 0.0000e+00, 0.0000e+00, 0.0000e+00, 4.0000e+00]),
 array([    0.,   200.,   400.,   600.,   800.,  1000.,  1200.,  1400.,
         1600.,  1800.,  2000.,  2200.,  2400.,  2600.,  2800.,  3000.,
         3200.,  3400.,  3600.,  3800.,  4000.,  4200.,  4400.,  4600.,
         4800.,  5000.,  5200.,  5400.,  5600.,  5800.,  6000.,  6200.,
         6400.,  6600.,  6800.,  7000.,  7200.,  7400.,  7600.,  7800.,
         8000.,  8200.,  8400.,  8600.,  8800.,  9000.,  9200.,  9400.,
         9600.,  9800., 10000.]),
 <BarContainer object of 50 artists>)
```




```
# Drop unnecessary columns
df.drop(['id', 'name', 'host_id', 'host_name'], axis=1, inplace=True)

# One-hot encode categorical variables
df = pd.get_dummies(df, columns=['neighbourhood_group', 'neighbourhood', 'room_type'], drop_first=True)

# Removing outliers in price
Q1 = df['price'].quantile(0.25)
Q3 = df['price'].quantile(0.75)
IQR = Q3 - Q1

# Upper and lower bound for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filtering out the outliers
df = df[(df['price'] >= lower_bound) & (df['price'] <= upper_bound)]

# Convert 'last_review' to numeric
df['last_review'] = pd.to_datetime(df['last_review'], errors='coerce').dt.year.fillna(0).astype(int)

df.head()
```

	latitude	longitude	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availability_365	booking_likel
0	40.64749	-73.97237	149	1	9	2018	0.21	6	365	
1	40.75362	-73.98377	225	1	45	2019	0.38	2	355	
3	40.68514	-73.95976	89	1	270	2019	4.64	1	194	
4	40.79851	-73.94399	80	10	9	2018	0.10	1	0	
5	40.74767	-73.97500	200	3	74	2019	0.59	1	129	

5 rows × 233 columns

```
# Fill missing values with median
df.fillna(df.median(), inplace=True)
```

```
from sklearn.model_selection import train_test_split

# Define features and target variable
X = df.drop('booking_likelihood', axis=1)
y = df['booking_likelihood']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Initialize the model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
rf_model.fit(X_train, y_train)

# Predict on the test set
y_pred = rf_model.predict(X_test)

# Evaluate the model
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))
```

```
Confusion Matrix:
[[ 249    0    0 ...    0    0    0]
 [  15   46    1 ...    0    0    6]
 [   7    1    4 ...    0    0    2]
 ...
 [   0    0    0 ...   43    3   13]
 [   0    0    0 ...    0   95   15]
 [   0    0    0 ...    0    0 4091]]

Classification Report:
              precision    recall  f1-score   support

           0       0.69      1.00      0.81       249
           1       0.52      0.52      0.52        88
           2       0.11      0.10      0.10        41
           3       0.11      0.11      0.11        28
           4       0.18      0.09      0.12        32
           5       0.09      0.07      0.08        29
           6       0.15      0.11      0.13        35
           7       0.06      0.04      0.05        45
           8       0.00      0.00      0.00        20
           9       0.19      0.15      0.17        20
          10       0.04      0.04      0.04        23
          11       0.08      0.11      0.09        19
```