Analytical Report

Introduction

Since its launch in 2008, Airbnb has allowed travellers and hosts to offer more individualised, distinctive travel experiences.

For this assignment, we developed a business analytics question based on the Airbnb dataset and performed Data Analysis to find the answer.

What key factors influence the price of Airbnb listings in New York City?

With this report we aim to understand the primary determinants of listing prices, which can help Airbnb optimise its pricing strategies, identify opportunities for price adjustments, and provide insights for setting competitive prices.

Data Analysis

Data Analysis consisted of two tasks:

- Exploratory Data Analysis
- Machine Learning Modelling

Exploratory Data Analysis (EDA)

Exploratory data analysis is crucial in understanding any dataset, providing insights into patterns, trends, and anomalies, and testing a hypothesis to make accurate, informed decisions (IBM, 2021).

Pre-processing

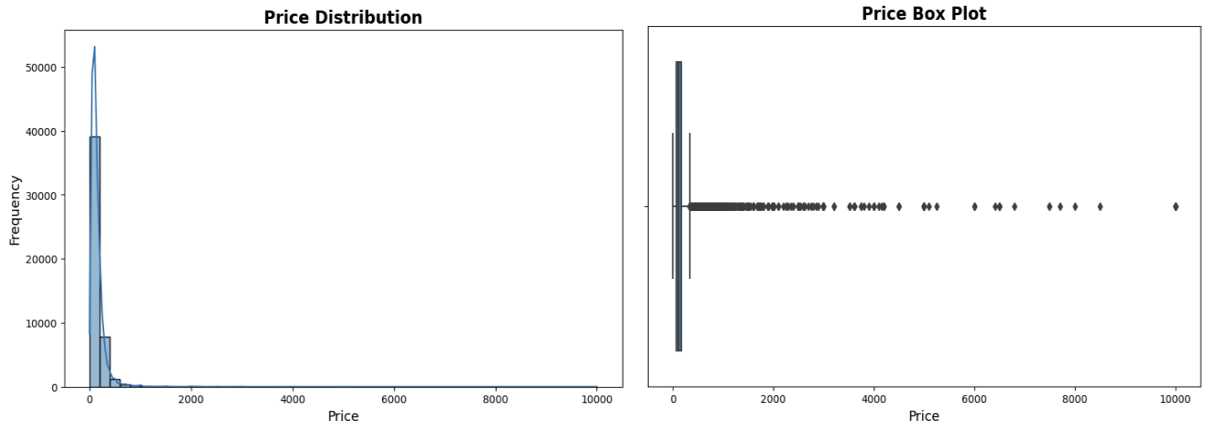
In the data preprocessing stage, we addressed missing values through imputation. We replaced missing values with *'Unknown'* for fields with minor missing data: name and `host_name`. The blanks for review-related columns `last_review` and `reviews_per_month`, which had around 20% missing data, were identified as corresponding to listings with zero reviews. These gaps were filled with *'No Reviews'* and 0, respectively, ensuring data consistency while preserving the dataset's integrity.

Statistical Analysis

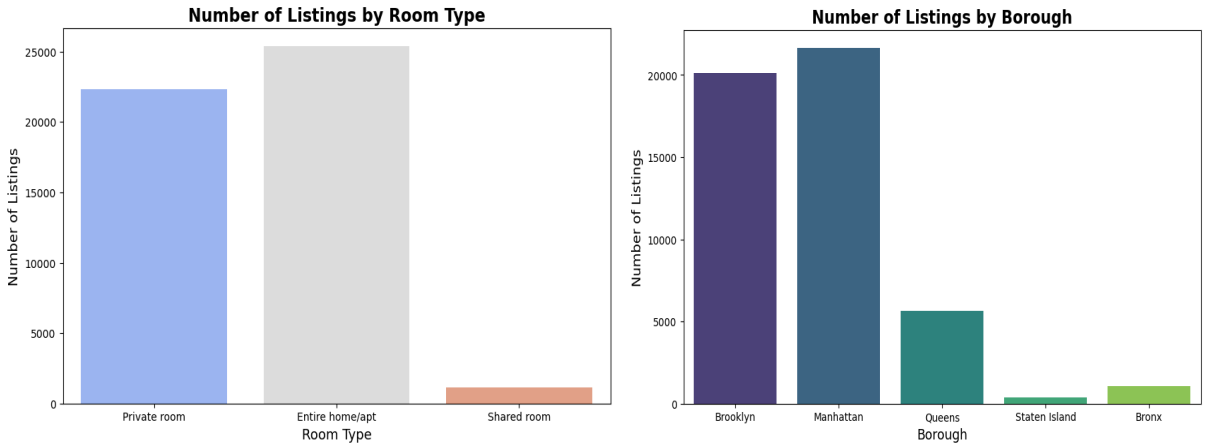
Statistics	latitude	longitude	price	minimum_n	number_of_	reviews_per_month	calculated_host_listings_count	availability_365
mean	390.22	-759.94	152.72	7.03	23.27	1.37	7.14	112.78
std	3754.75	7085.93	240.15	20.51	44.55	1.68	32.95	131.62
min	40.50	-74142	0	1	0	0.01	1	0
25%	40.69	-73.98	69	1	1	0.19	1	0
50%	40.72	-73.96	106	3	5	0.72	1	45
75%	40.76	-73.94	175	5	24	2.02	2	227
max	40896	-73.71	10000	1250	629	58.5	327	365

The summary statistics table highlights notable variability in the dataset, especially regarding geographic coordinates and listing prices. The average latitude and longitude values (390.22 and 759.94) appear incorrect, likely due to data errors, with a high standard deviation further emphasising this inconsistency. Listing prices vary widely, from $0 to $10,000, suggesting a mix of affordable and luxury properties. Listings priced at $0 raise concerns about data accuracy, potentially caused by errors or promotions. Most listings (75%) are priced at or below $175, indicating general affordability. The minimum stay requirement ranges from 1 to 1250 nights, reflecting diverse hosting strategies. While the average number of reviews per listing is 23.27, some listings receive up to 58.5 reviews per month, showing varying popularity. Host listings also range widely, from individual hosts to professionals managing as many as 327 properties, with availability varying from occasional to year-round rentals.

Univariate Analysis

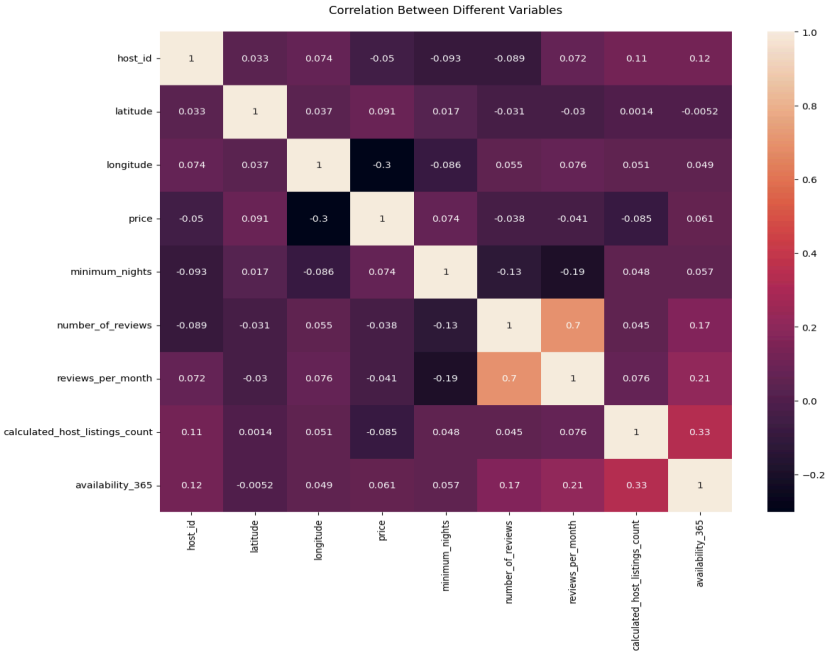


A deeper univariate analysis of the price variable shows a highly skewed distribution, with most listings priced low and a few extremely high-priced listings creating a long tail. The skewness is evident in the box plot, which highlights several high-priced outliers that raise the mean price.



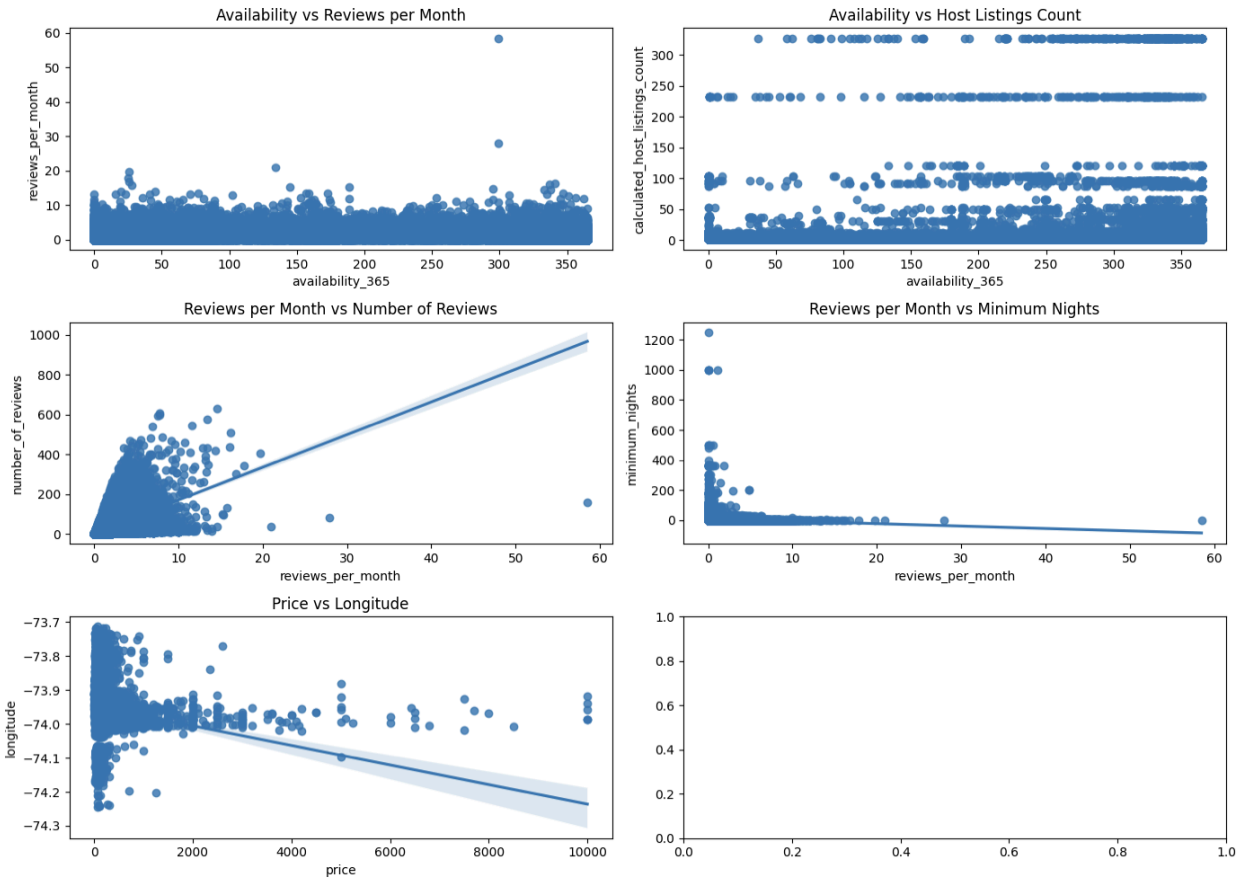
Furthermore, the neighbourhood group and room type are likely significant factors influencing price. As illustrated in the above graphs, Manhattan and Brooklyn have the highest concentration of Airbnb properties, reflecting their popularity and centrality in New York City. Private rooms and entire apartments are the most common offerings on the platform.

Bivariate Analysis



Most correlations in the heatmap were pretty weak, except for the one between `reviews_per_month` and `reviews_per_month`. To get a clearer picture, correlations were filtered using thresholds of -0.2 and 0.2, and a scatter map was created.

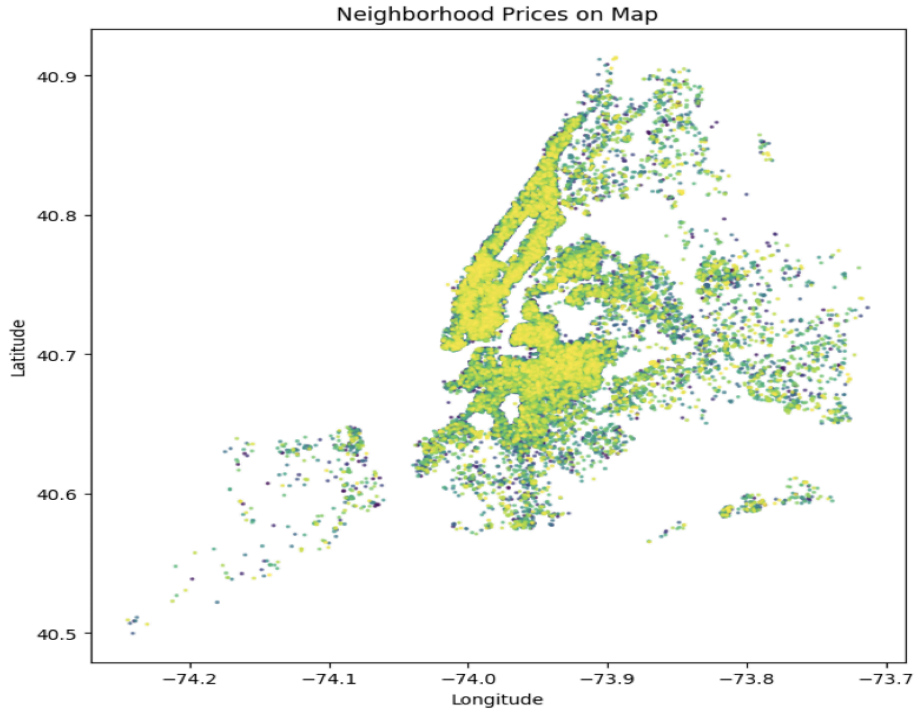
Here's what was found:



- `availability_365` **and** `reviews_per_month`**:** there's no strong correlation here. Reviews are pretty evenly spread across different availability levels, with most listings getting fewer than 20 reviews per month, no matter their availability.
- `availability_365` **and** `calculated_host_listings_count`**:** there isn't a clear correlation. Host listing counts vary across all availability levels. However, there are noticeable horizontal lines, suggesting that some hosts have a typical number of listings (like around 100, 230, and 330).

- `reviews_per_month` **and** `number_of_reviews`**:** there's a strong positive correlation. More reviews per month usually mean a higher total number of reviews, which makes sense - more frequent reviews add up over time.
- `reviews_per_month` **and** `minimum_nights`: there's a slight negative correlation. Listings with more reviews per month often have shorter minimum night requirements, suggesting that places with shorter stay options might get booked and reviewed more often.
- `price` **and** `longitude`**:** a negative correlation is evident. As prices go up, longitude values decrease (meaning they move westward). This indicates that, on average, western parts of New York City tend to have higher-priced listings.

Since the correlation between price and longitude showed the clearest and most interesting trend, further analysis was conducted to test the initial hypothesis.



As anticipated, the plotted map confirmed that prices increased in the western parts of New York City.

ML modelling

Machine Learning modelling assesses data and finds insights to make decisions that enhance company outcomes (Oracle, 2023).

We used clustering to model the Airbnb dataset. This unsupervised machine-learning technique helps to group related objects by recognising patterns and displaying data.

The process required three steps: K-prototype for clustering, Mutual Information for feature selection, and tSNE for high-dimensional visualisation.

By applying Mutual Information, we found factors influencing price and how much information each of them provided about price.

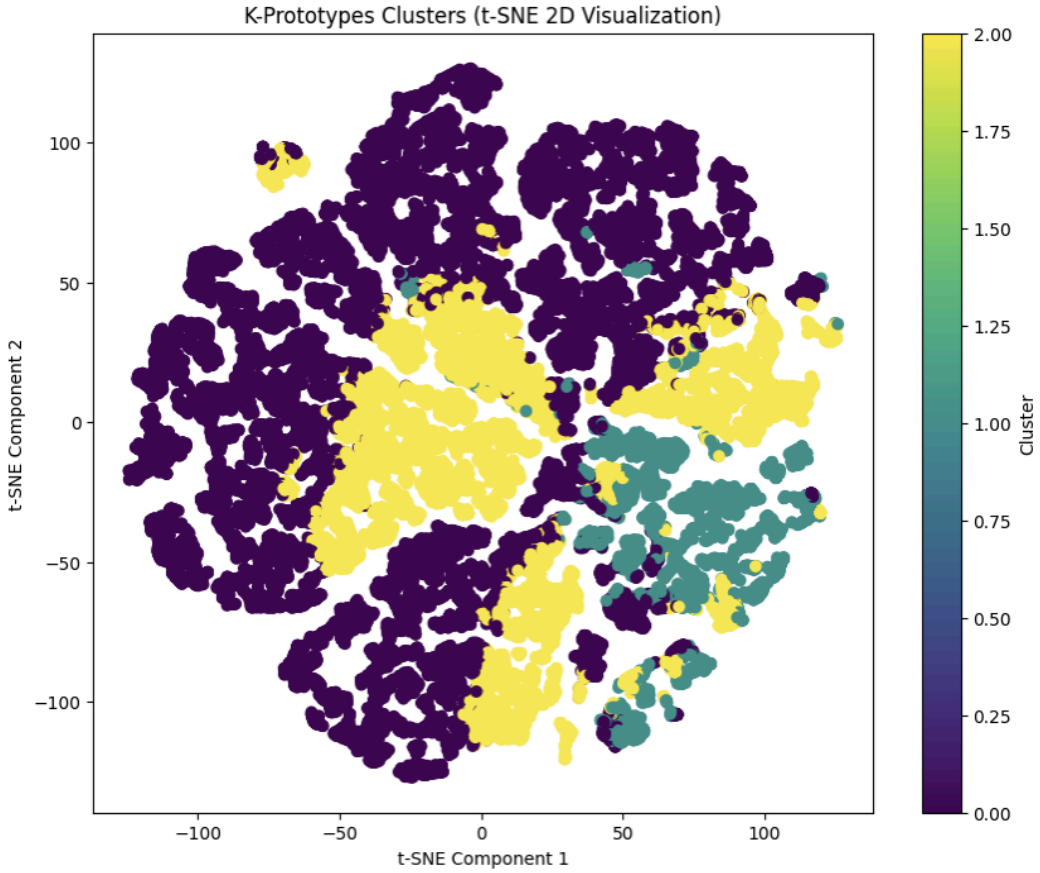
Key Factors Influencing Price:

	Feature	Mutual Information
9	room_type_Entire home/apt	0.293475
10	room_type_Private room	0.250996
2	longitude	0.163658
1	latitude	0.128921
6	neighbourhood_group_Manhattan	0.074281
3	availability_365	0.036106
5	neighbourhood_group_Brooklyn	0.025187
11	room_type_Shared room	0.024140
0	number_of_reviews	0.023117
7	neighbourhood_group_Queens	0.017675
4	neighbourhood_group_Bronx	0.006992
8	neighbourhood_group_Staten Island	0.000000

A high MI score indicates a strong relationship between the feature and the target variable

We selected the K-prototype algorithm to manage the dataset's numerical and categorical attributes.

Our analysis grouped 48000 listings into three distinct clusters. The features that greatly influenced these clusters included `room_type`, `longitude`, `neighbourhood_group`, and `availability_365`.



Budget listings with greater availability are found in less crowded neighbourhoods, represented in the teal cluster. The yellow cluster shows expensive, premium listings with restricted availability in well-known regions such as Manhattan. The purple cluster depicts mid-range listings, maybe private or shared, with a range of availability, a high number of reviews, and a modest price.

Clustering research identifies various market segments within NYC's Airbnb listings, providing useful information for strategic decision-making. Using these insights can boost customer happiness and optimise revenue streams.

Conclusion

This report presented the examination of key factors influencing the price of Airbnb listings in New York by means of Exploratory Data Analysis and Machine Learning modelling. It provides useful information for maximising pricing strategies by differentiating market clusters according to availability, room type, and location. Better client targeting and more revenue creation can result from these results.

References

IBM (2021) What is exploratory data analysis? Available from: https://www.ibm.com/topics/exploratory-data-analysis [Accessed 25 August 2024].

Oracle (2023) What is Machine Learning for Analytics? Available from: https://www.oracle.com/in/business-analytics/what-is-machine-learning-for-analytics/ [Accessed 25 August 2024].

Appendix

EDA:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import geopandas as gpd

from sklearn.preprocessing import StandardScaler, LabelEncoder
from tabulate import tabulate
from shapely.geometry import Point

# Importing dataset and
df = pd.read_excel('AB_NYC_2019.xlsx')

# Getting overview of dataset
df.head()
df.info()

# Detecting and Handling with missing values
missing_values = df.isnull().sum()
print(missing_values)

missing_percentage = df.isnull().mean() * 100
missing_percentage.sort_values(ascending=False)

df['name'].fillna('Unknown', inplace=True)
df['host_name'].fillna('Unknown', inplace=True)

missing_values = df.isnull().sum()
print(missing_values)

df.loc[df['number_of_reviews']      ==      0,      'last_review']      =
df.loc[df['number_of_reviews'] == 0, 'last_review'].fillna('No Reviews')

df.loc[df['number_of_reviews']     ==     0,     'reviews_per_month']     =
df.loc[df['number_of_reviews'] == 0, 'reviews_per_month'].fillna(0)

print(df.isnull().sum())
df.head()
df.shape
```

```
# Calculating summary statistics
df.describe()

summary = df.describe()
summary.to_excel('summary_statistics.xlsx', sheet_name='Summary')

# Plotting variables for univariate analysis
plt.figure(figsize=(10, 6))
sns.histplot(df['price'], bins=50, kde=True)
plt.title('Price Distribution')
plt.xlabel('Price')
plt.ylabel('Frequency')
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='neighbourhood_group', data=df, palette='viridis')
plt.title('Number of Listings by Borough',fontdict={'fontsize': 16,
'fontweight': 'bold'})
plt.xlabel('Borough',fontdict={'fontsize': 13})
plt.ylabel('Number of Listings', fontdict={'fontsize': 13})
plt.show()

plt.figure(figsize=(10, 6))
sns.histplot(df['price'], bins=50, kde=True)
plt.title('Price Distribution', fontdict={'fontsize': 16, 'fontweight':
'bold'})
plt.xlabel('Price', fontdict={'fontsize': 13})
plt.ylabel('Frequency', fontdict={'fontsize': 13})
plt.show()

plt.figure(figsize=(10, 6))
sns.boxplot(x=df['price'])
plt.title('Price Box Plot',fontdict={'fontsize': 16, 'fontweight':
'bold'})
plt.xlabel('Price', fontdict={'fontsize': 13})
plt.show()

plt.figure(figsize=(10, 6))
sns.countplot(x='room_type', data=df, palette='coolwarm')
plt.title('Number of Listings by Room Type',fontdict={'fontsize': 16,
'fontweight': 'bold'})
plt.xlabel('Room Type', fontdict={'fontsize': 13})
plt.ylabel('Number of Listings', fontdict={'fontsize': 13})
```

```
plt.show()

# Correlation
numerical_df = df.select_dtypes(include=['float', 'int'])
numerical_df = numerical_df.drop(columns=['id'])

corr = numerical_df.corr(method='kendall')
plt.figure(figsize=(13,10))
plt.title("Correlation Between Different Variables\n")
sns.heatmap(corr, annot=True)
plt.show()

positive_threshold = 0.2
negative_threshold = -0.2

positive_correlations = set()
negative_correlations = set()

for i in range(len(corr.columns)):
    for j in range(i):
        corr_value = corr.iloc[i, j]
        colname1 = corr.columns[i]
        colname2 = corr.columns[j]

        if corr_value > positive_threshold:
            positive_correlations.add((colname1, colname2))

        elif corr_value < negative_threshold:
            negative_correlations.add((colname1, colname2))

print("Sets of positively correlated columns:")
for pair in positive_correlations:
    print(pair)

print("\nSets of negatively correlated columns:")
for pair in negative_correlations:
    print(pair)

# Scatter plots
fig, ((ax1, ax2), (ax3, ax4), (ax5, ax6)) = plt.subplots(nrows=3,
ncols=2, figsize=(14,10))


# Plot 1: 'availability_365' vs. 'reviews_per_month'
```

```
plot1_data = pd.concat([df['availability_365'], df['reviews_per_month']],
axis=1)
sns.regplot(x='availability_365', y='reviews_per_month', data=plot1_data,
scatter=True, fit_reg=True, ax=ax1)
ax1.set_title('Availability vs Reviews per Month')

# Plot 2: 'availability_365' vs. 'calculated_host_listings_count'
plot2_data              =              pd.concat([df['availability_365'],
df['calculated_host_listings_count']], axis=1)
sns.regplot(x='availability_365',     y='calculated_host_listings_count',
data=plot2_data, scatter=True, fit_reg=True, ax=ax2)
ax2.set_title('Availability vs Host Listings Count')

# Plot 3: 'reviews_per_month' vs. 'number_of_reviews'
plot3_data             =             pd.concat([df['reviews_per_month'],
df['number_of_reviews']], axis=1)
sns.regplot(x='reviews_per_month',                 y='number_of_reviews',
data=plot3_data, scatter=True, fit_reg=True, ax=ax3)
ax3.set_title('Reviews per Month vs Number of Reviews')

# Plot 4: 'reviews_per_month' vs. 'minimum_nights'
plot4_data  =  pd.concat([df['reviews_per_month'],  df['minimum_nights']],
axis=1)
sns.regplot(x='reviews_per_month',  y='minimum_nights',  data=plot4_data,
scatter=True, fit_reg=True, ax=ax4)
ax4.set_title('Reviews per Month vs Minimum Nights')

# Plot 5: 'price' vs. 'longitude'
plot5_data = pd.concat([df['price'], df['longitude']], axis=1)
sns.regplot(x='price',   y='longitude',   data=plot5_data,   scatter=True,
fit_reg=True, ax=ax5)
ax5.set_title('Price vs Longitude')

plt.tight_layout()
plt.show()


gdf = gpd.GeoDataFrame(
   df, geometry=gpd.points_from_xy(df.longitude, df.latitude),
   crs="EPSG:4326"
)

fig, ax = plt.subplots(figsize=(12, 8))
```

```
gdf.plot(ax=ax, markersize=2, c='price', cmap='viridis', alpha=0.6,
legend=True)

ax.set_title('Neighborhood Prices on Map')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')

plt.show()
```

ML modelling:

```
!pip install kmodes
```

```
import pandas as pd
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from kmodes.kprototypes import KPrototypes
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from sklearn.feature_selection import mutual_info_regression

# Load data
df = pd.read_csv('AB_NYC_2019.csv')

#Replace NaN values with 0 for feature selection
df = df.fillna(0)

# Define features and target
features = ['neighbourhood_group', 'room_type', 'number_of_reviews', 'latitude', 'longitude', 'availability_365']
target = 'price'

# Calculate Mutual Information
df_encoded = pd.get_dummies(df[features])
mi = mutual_info_regression(df_encoded, df[target])
mi_df = pd.DataFrame({'Feature': df_encoded.columns, 'Mutual Information': mi})

# Print Mutual Information results
print("\nKey Factors Influencing Price:\n", mi_df.sort_values(by='Mutual Information', ascending=False))

# Select top 5 features based on Mutual Information
top_features = mi_df.sort_values(by='Mutual Information', ascending=False).head(5)['Feature'].tolist()

# Preprocess data
numerical_features = [feat for feat in features if feat not in ['neighbourhood_group', 'room_type']]
categorical_features = ['neighbourhood_group', 'room_type']
```

```
numerical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value=0)),
    ('scaler', StandardScaler())
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='constant', fill_value='missing')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

preprocessor = ColumnTransformer(transformers=[
    ('num', numerical_transformer, numerical_features),
    ('cat', categorical_transformer, categorical_features)
])

X = preprocessor.fit_transform(df)

X_top_features = X[:, [top_features.index(feat) for feat in top_features]]

# K-Prototypes clustering
kproto = KPrototypes(n_clusters=3, verbose=2, max_iter=20)
clusters = kproto.fit_predict(X, categorical=[0, 1])

# t-SNE visualization
tsne = TSNE(n_components=2, random_state=42)
tsne_results = tsne.fit_transform(X)
tsne_df = pd.DataFrame(tsne_results, columns=['Component 1', 'Component 2'])
tsne_df['Cluster'] = clusters
```

```
    plt.figure(figsize=(10, 8))
    scatter = plt.scatter(tsne_results[:, 0], tsne_results[:, 1], c=clusters, cmap='viridis', marker='o')
    plt.title('K-Prototypes Clusters (t-SNE 2D Visualization)')
    plt.xlabel('t-SNE Component 1')
    plt.ylabel('t-SNE Component 2')
    plt.colorbar(scatter, label='Cluster')
    plt.savefig('cluster_visualization.png', dpi=300, bbox_inches='tight')
    plt.show()
```

```
[ ] print(top_features)
```

```
['room_type_Entire home/apt', 'room_type_Private room', 'longitude', 'latitude', 'neighbourhood_group_Manhattan']
```

```
tsne_df
```

	Component 1	Component 2	Cluster
0	112.672935	12.158177	2
1	-26.869822	-17.843817	2
2	19.099230	-100.193253	2
3	26.906172	-1.795045	2
4	-84.750000	-43.105465	0
...	...	...	...
48890	54.715645	89.642624	0
48891	46.880047	72.906052	0
48892	-68.891045	-53.205215	0
48893	41.335316	-78.752571	0
48894	-39.039574	-68.216675	0

48895 rows × 3 columns

```python
purple_cluster_data = df[clusters == 0]

# Step 2: Descriptive statistics for numerical features
geographical_distribution = purple_cluster_data[['latitude', 'longitude']].describe()
price_stats = purple_cluster_data['price'].describe()
availability_stats = purple_cluster_data['availability_365'].describe()
review_stats = purple_cluster_data['number_of_reviews'].describe()

# Step 3: Summary for categorical features
room_type_distribution = purple_cluster_data['room_type'].value_counts()

# Print the results
print("Geographical Distribution (Latitude and Longitude):")
print(geographical_distribution)
print("\nPrice Range Statistics:")
print(price_stats)
print("\nAvailability Statistics:")
print(availability_stats)
print("\nNumber of Reviews Statistics:")
print(review_stats)
print("\nRoom Type Distribution:")
print(room_type_distribution)
```

```
Geographical Distribution (Latitude and Longitude):
           latitude     longitude
count  29888.000000  29888.000000
mean      40.728278    -73.961889
std        0.052266      0.028025
min       40.522110    -74.212380
25%       40.690500    -73.983770
50%       40.721005    -73.958450
75%       40.763280    -73.943050
max       40.907340    -73.878510

Price Range Statistics:
count    29888.000000
mean       143.498160
std        216.571804
min          0.000000
25%         70.000000
50%        105.000000
75%        170.000000
max      10000.000000
Name: price, dtype: float64

Availability Statistics:
count    29888.000000
mean        23.524692
std         38.998238
min          0.000000
25%          0.000000
50%          0.000000
75%         35.000000
max        165.000000
Name: availability_365, dtype: float64

Number of Reviews Statistics:
count    29888.000000
mean        17.234576
std         35.161582
min          0.000000
25%          1.000000
50%          4.000000
75%         16.000000
max        480.000000
Name: number_of_reviews, dtype: float64

Room Type Distribution:
room_type
Entire home/apt    15735
Private room       13606
Shared room          547
Name: count, dtype: int64
```

```
yellow_cluster_data = df[clusters == 1]

# Step 2: Descriptive statistics for numerical features
geographical_distribution = yellow_cluster_data[['latitude', 'longitude']].describe()
price_stats = yellow_cluster_data['price'].describe()
availability_stats = yellow_cluster_data['availability_365'].describe()
review_stats = yellow_cluster_data['number_of_reviews'].describe()

# Step 3: Summary for categorical features
room_type_distribution = yellow_cluster_data['room_type'].value_counts()

# Print the results
print("Geographical Distribution (Latitude and Longitude):")
print(geographical_distribution)
print("\nPrice Range Statistics:")
print(price_stats)
print("\nAvailability Statistics:")
print(availability_stats)
print("\nNumber of Reviews Statistics:")
print(review_stats)
print("\nRoom Type Distribution:")
print(room_type_distribution)
```

```
Geographical Distribution (Latitude and Longitude):
           latitude     longitude
count  14249.000000  14249.000000
mean      40.728029    -73.965235
std        0.055503      0.033524
min       40.499790    -74.244420
25%       40.688050    -73.987270
50%       40.726050    -73.962010
75%       40.763310    -73.942760
max       40.911690    -73.885410

Price Range Statistics:
count    14249.000000
mean       192.957471
std        304.963432
min          0.000000
25%         80.000000
50%        133.000000
75%        215.000000
max       9999.000000
Name: price, dtype: float64

Availability Statistics:
count    14249.000000
mean       282.128079
std         67.488190
min        128.000000
25%        225.000000
50%        297.000000
75%        342.000000
max        365.000000
Name: availability_365, dtype: float64

Number of Reviews Statistics:
count    14249.000000
mean        33.539617
std         55.147798
min          0.000000
25%          1.000000
50%          9.000000
75%         41.000000
max        607.000000
Name: number_of_reviews, dtype: float64

Room Type Distribution:
room_type
Entire home/apt    8169
Private room       5650
Shared room         430
Name: count, dtype: int64
```