In [137]:
```
# importing the libraries
import pandas as pd  # for reading dataset
```

In [138]:
```
df = pd.read_csv("listings.csv") # loading the dataset and reading it
```

```
C:\Users\Sneha\AppData\Local\Temp\ipykernel_27908\2821553885.py:1: DtypeWarning: Columns (68) have mixed types. Speci
fy dtype option on import or set low_memory=False.
  df = pd.read_csv("listings.csv") # loading the dataset and reading it
```

In [139]:
```
df.head() # checking the first 5 rows and cloumns
```

Out[139]:

	id	listing_url	scrape_id	last_scraped	source	name	description	neighborhood_overview	
0	92644	https://www.airbnb.com/rooms/92644	20230906022807	2023-09-06	city scrape	Rental unit in Earlsfield · ★4.57 · 1 bedroom ...	<b>The space</b> <br />Hi everyone! I have 2 ro...	NaN	https://a(
1	93015	https://www.airbnb.com/rooms/93015	20230906022807	2023-09-06	city scrape	Rental unit in Hammersmith · ★4.82 · 2 bedroom...	Gorgeous 2 bed ground floor apartment with per...	A bit of history about the W14 area: <br />Com...	https://a(
2	13913	https://www.airbnb.com/rooms/13913	20230906022807	2023-09-06	city scrape	Rental unit in Islington · ★4.80 · 1 bedroom ·...	My bright double bedroom with a large window h...	Finsbury Park is a friendly melting pot commun...	https://
					city	Rental unit in London ·	Lots of windows and		

In [129]:
```
# checking columns with unique
df.columns.unique()
```

Out[129]:
```
Index(['id', 'listing_url', 'scrape_id', 'last_scraped', 'source', 'name',
       'description', 'neighborhood_overview', 'picture_url', 'host_id',
       'host_url', 'host_name', 'host_since', 'host_location', 'host_about',
       'host_response_time', 'host_response_rate', 'host_acceptance_rate',
       'host_is_superhost', 'host_thumbnail_url', 'host_picture_url',
       'host_neighbourhood', 'host_listings_count',
       'host_total_listings_count', 'host_verifications',
       'host_has_profile_pic', 'host_identity_verified', 'neighbourhood',
       'neighbourhood_cleansed', 'neighbourhood_group_cleansed', 'latitude',
       'longitude', 'property_type', 'room_type', 'accommodates', 'bathrooms',
       'bathrooms_text', 'bedrooms', 'beds', 'amenities', 'price',
       'minimum_nights', 'maximum_nights', 'minimum_minimum_nights',
       'maximum_minimum_nights', 'minimum_maximum_nights',
       'maximum_maximum_nights', 'minimum_nights_avg_ntm',
       'maximum_nights_avg_ntm', 'calendar_updated', 'has_availability',
       'availability_30', 'availability_60', 'availability_90',
       'availability_365', 'calendar_last_scraped', 'number_of_reviews',
       'number_of_reviews_ltm', 'number_of_reviews_l30d', 'first_review',
       'last_review', 'review_scores_rating', 'review_scores_accuracy',
       'review_scores_cleanliness', 'review_scores_checkin',
       'review_scores_communication', 'review_scores_location',
       'review_scores_value', 'license', 'instant_bookable',
       'calculated_host_listings_count',
       'calculated_host_listings_count_entire_homes',
       'calculated_host_listings_count_private_rooms',
       'calculated_host_listings_count_shared_rooms', 'reviews_per_month'],
      dtype='object')
```

To Keep only necessary columns

In [130]:
```
#importing the libraries
import pandas as pd

# selecting columns which are only required
selecting_col = [
    'host_since', 'host_response_time', 'host_response_rate',
    'host_acceptance_rate', 'host_is_superhost', 'host_listings_count',
    'property_type', 'room_type', 'accommodates', 'bathrooms',
    'bedrooms', 'beds', 'amenities', 'price', 'number_of_reviews',
    'latitude', 'longitude', 'neighbourhood_cleansed',
    'review_scores_rating', 'reviews_per_month'
]

# keeping required column only
df1 = df[selecting_col]

#checking the size of the dataset
df1.shape
```

Out[130]:
```
(87946, 20)
```

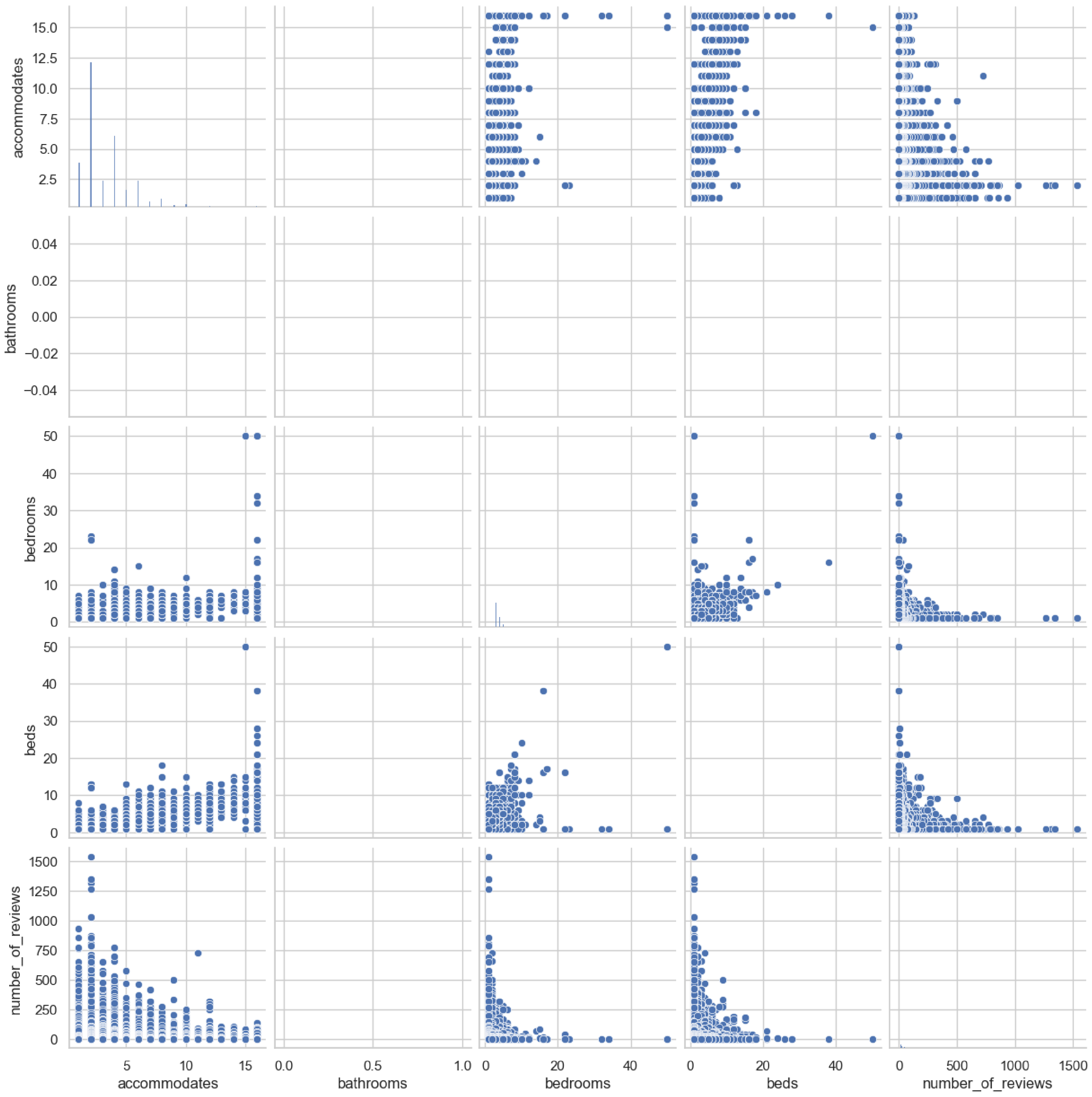
In [131]:
```
df1.head() #Checking first 5 columns
```

Out[131]:

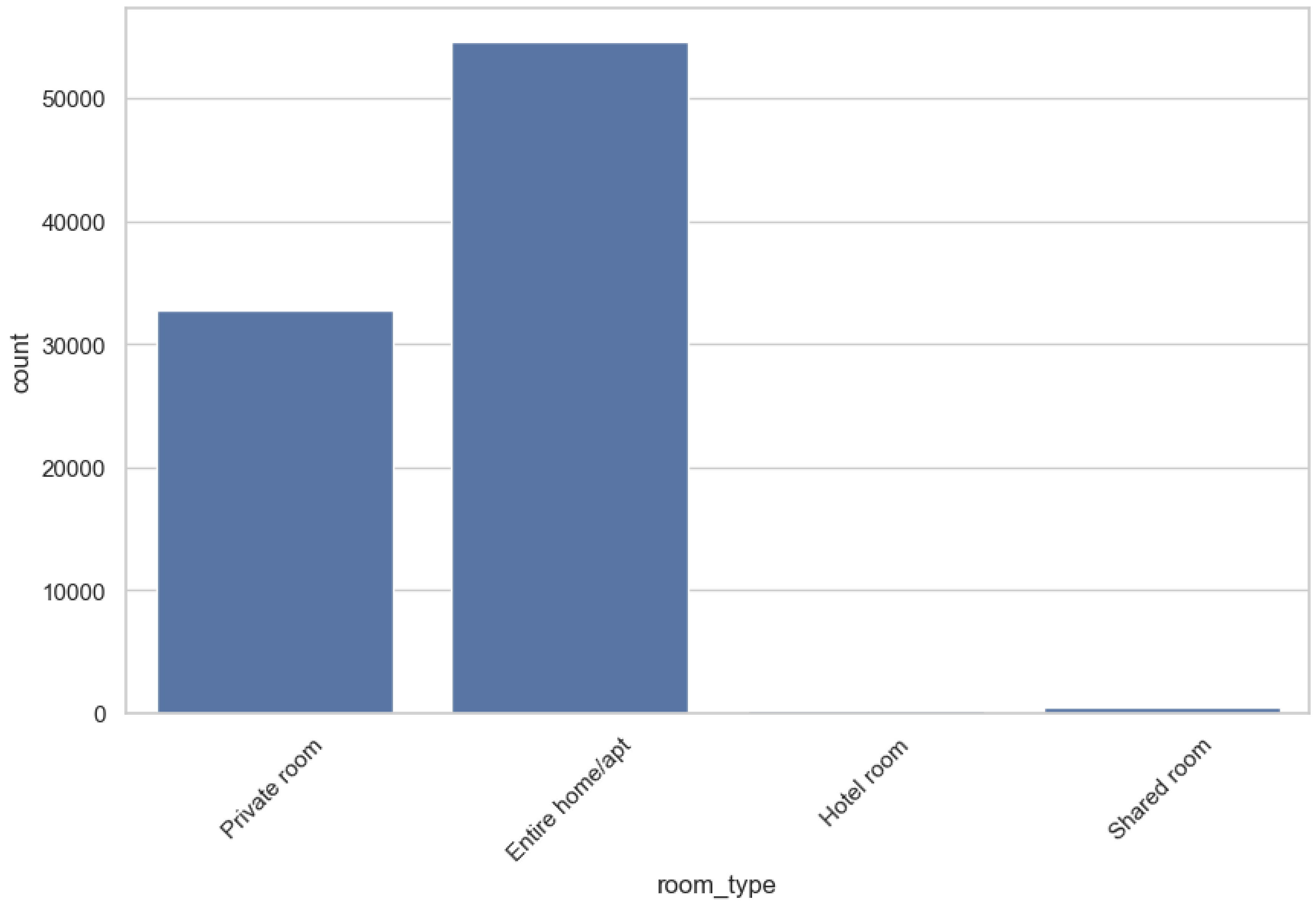
	host_since	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost	host_listings_count	property_type	room_type	a
0	2011-04-10	NaN	NaN	100%	f	1.0	Private room in rental unit	Private room	
1	2011-04-11	within a few hours	100%	25%	f	1.0	Entire rental unit	Entire home/apt	
2	2009-11-16	within a few hours	100%	88%	f	3.0	Private room in rental unit	Private room	
3	2009-12-05	within a day	100%	41%	f	1.0	Entire rental unit	Entire home/apt	
4	2011-04-10	within a few hours	90%	75%	t	1.0	Private room in condo	Private room	

Data visualisation

In [132]:
```python
import seaborn as sns
import matplotlib.pyplot as plt

# Set the style of seaborn
sns.set(style="whitegrid")

# Pairplot to visualize relationships between numerical columns
sns.pairplot(df1[['accommodates', 'bathrooms', 'bedrooms', 'beds', 'price', 'number_of_reviews']])
plt.show()
```



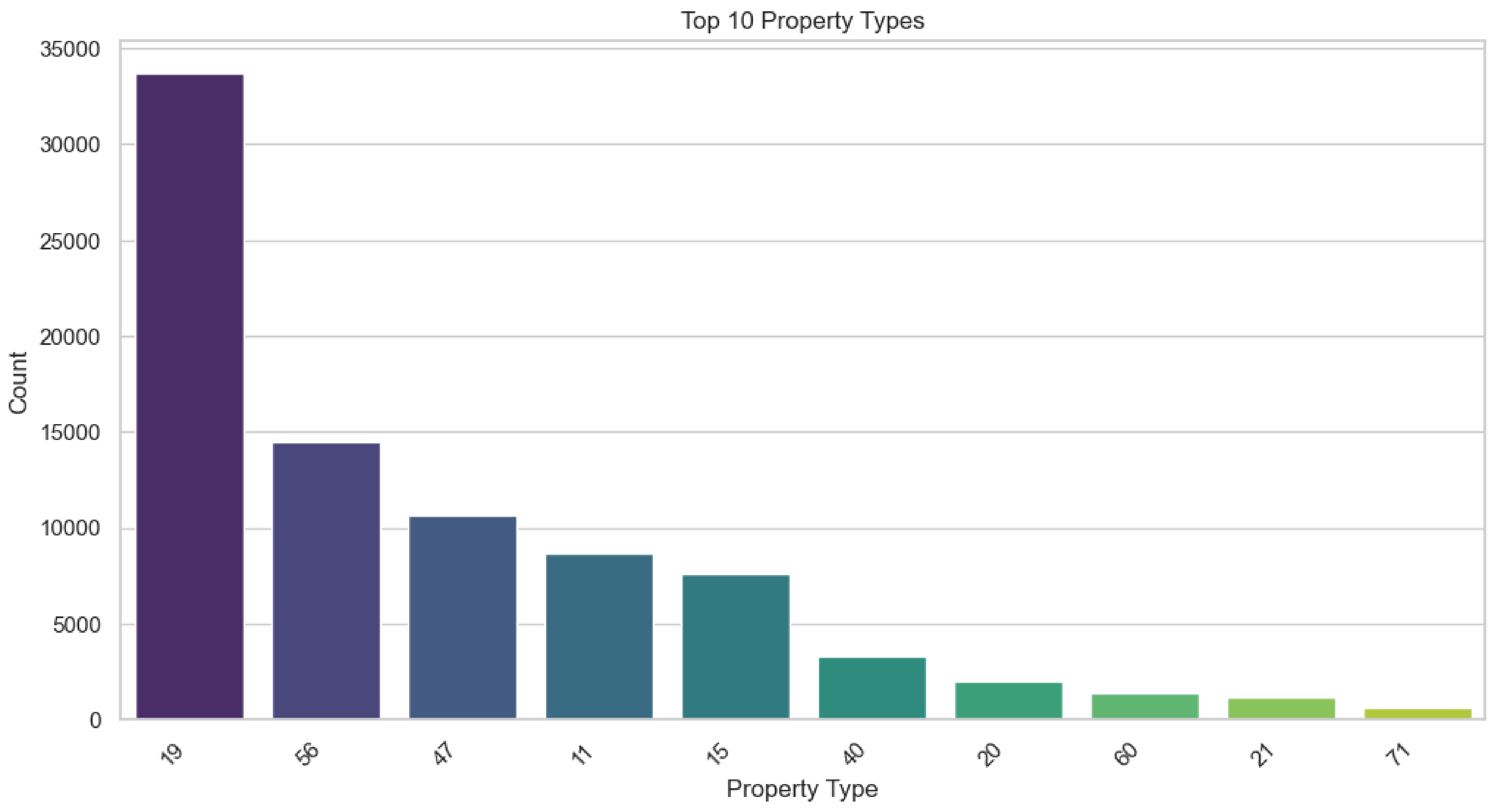
```
In [ ]: # Countplot of room categories
        plt.figure(figsize=(10, 6))
        sns.countplot(x='room_type', data=df1)
        plt.xticks(rotation=45)
        plt.show()
```



In [155]:
```python
#Matplotlib is an extremely sophisticated Python charting package.
#Seaborn is a high-level interface for providing statistical visuals that is built on Matplotlib.
import matplotlib.pyplot as plt
import seaborn as sns


property_types10 = df1['property_type'].value_counts().nlargest(10).index

# Filtering  the dataset for the top 10 property types
df_10 = df1[df1['property_type'].isin(property_types10)]

# Chart for  top 10 property type
plt.figure(figsize=(12, 6))
sns.countplot(x='property_type', data=df_10, order=property_types10, palette='viridis')
plt.title('Top 10 Property Types')
plt.xlabel('Property Type')
plt.ylabel('Count')
plt.xticks(rotation=45, ha='right')
```

```
C:\Users\Sneha\AppData\Local\Temp\ipykernel_27908\194507423.py:14: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `h
ue` and set `legend=False` for the same effect.

  sns.countplot(x='property_type', data=df_10, order=property_types10, palette='viridis')
```

Out[155]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9],
 [Text(0, 0, '19'),
  Text(1, 0, '56'),
  Text(2, 0, '47'),
  Text(3, 0, '11'),
  Text(4, 0, '15'),
  Text(5, 0, '40'),
  Text(6, 0, '20'),
  Text(7, 0, '60'),
  Text(8, 0, '21'),
  Text(9, 0, '71')])
```



EDA

Step 1: Checking the datatypes

In [140]:
```
# Check data types of each column
d_type = df1.dtypes


# Separating data into  numeric and categorical columns
numeric_columns = d_type[d_type != 'object'].index
categorical_columns = d_type[d_type == 'object'].index

# Print the results
print("Numeric Columns:")
print(numeric_columns)

print("\nCategorical Columns:")
print(categorical_columns)
```

```
Numeric Columns:
Index(['host_listings_count', 'accommodates', 'bathrooms', 'bedrooms', 'beds',
       'number_of_reviews', 'latitude', 'longitude', 'review_scores_rating',
       'reviews_per_month'],
      dtype='object')

Categorical Columns:
Index(['host_since', 'host_response_time', 'host_response_rate',
       'host_acceptance_rate', 'host_is_superhost', 'property_type',
       'room_type', 'amenities', 'price', 'neighbourhood_cleansed'],
      dtype='object')
```

In [141]:
```
from sklearn.preprocessing import LabelEncoder

# Categorical columns to convert
categorical_columns = ['host_since', 'host_response_rate', 'host_acceptance_rate', 'host_is_superhost', 'amenities', '

# Initialinsing LabelEncoder
label_encoder = LabelEncoder()

# Apply label encoding to each categorical column using .loc
for column in categorical_columns:
    df1.loc[:, column] = label_encoder.fit_transform(df1[column])

# Display the updated DataFrame
(df1.head())
```

Out[141]:

	host_since	host_response_time	host_response_rate	host_acceptance_rate	host_is_superhost	host_listings_count	property_type	room_type	a
0	286	NaN	95	3	0	1.0	56	2	
1	287	within a few hours	2	18	0	1.0	19	0	
2	35	within a few hours	2	87	0	3.0	56	2	
3	40	within a day	2	36	0	1.0	19	0	
4	286	within a few hours	85	73	1	1.0	40	2	

Checking for Missing values using isnull().sum() it checks for all columns missing values and taking percentage of missing values

In [142]:
```
# Check for missing values in the DataFrame
missing_values = df1.isnull().sum()

# Displaying the columns with missing values, if any
Col_missing_val = missing_values[missing_values > 0]
print("Columns with missing values:")
print(Col_missing_val)

# Percentage of missing values in each column
missing_percentage = (missing_values / len(df1)) * 100
print("\nPercentage of missing values in each column:")
print(missing_percentage)
```

```
Columns with missing values:
host_response_time      28918
host_listings_count         5
bathrooms               87946
bedrooms                32774
beds                     1134
review_scores_rating    22158
reviews_per_month       22158
dtype: int64

Percentage of missing values in each column:
host_since                  0.000000
host_response_time         32.881541
host_response_rate          0.000000
host_acceptance_rate        0.000000
host_is_superhost           0.000000
host_listings_count         0.005685
property_type               0.000000
room_type                   0.000000
accommodates                0.000000
bathrooms                 100.000000
bedrooms                   37.266050
beds                        1.289428
amenities                   0.000000
price                       0.000000
number_of_reviews           0.000000
latitude                    0.000000
longitude                   0.000000
neighbourhood_cleansed      0.000000
review_scores_rating       25.195006
reviews_per_month          25.195006
dtype: float64
```

In [143]:
```
# Dropping the bathroom column since it's showing 100 percent missing values using drop function
df1= df1.drop(['bathrooms'], axis=1)
```

In [144]:
```
# The function guarantees that missing values in df1 are replaced with mode values generated from the original Dataset
#df columns. This is a popular approach for dealing with missing data by substituting the most frequent values from ea

df1.loc[:, :] = df1.fillna(df.mode().iloc[0])
df1.isnull().any().sum() # now again checking if there is null values or not
```

Out[144]:
```
0
```

Feature Engineering

In [145]:
```
# Check summary statistics for columns number_of_reviews,review_scores_rating
print(df1[['number_of_reviews', 'review_scores_rating']].describe())
```

```
       number_of_reviews  review_scores_rating
count       87946.000000          87946.000000
mean           17.977236              4.698192
std            42.834975              0.672555
min             0.000000              0.000000
25%             0.000000              4.670000
50%             4.000000              4.960000
75%            16.000000              5.000000
max          1536.000000              5.000000
```

To provide descriptive statistics for a DataFrame or Series, utilize Pandas' describe function. The distribution, central tendency, and dispersion of the values in the corresponding columns are all shown by these statistics.

In [146]:
```
# Calculating  the mean of 'number_of_reviews'
review_threshold = df1['number_of_reviews'].mean()

# Calculating  the mean of 'review_scores_rating'
rating_threshold = df1['review_scores_rating'].mean()

# Creating a new column 'Success' based on no of reviews and review score rating
# Updating the 'Success' to 'Bad' using .loc
df1.loc[:, 'Success'] = 'Bad'


# Updating  'Success' to 'Good' based on conditions
condition_reviews = df1['number_of_reviews'] >= review_threshold
condition_rating = df1['review_scores_rating'] >= rating_threshold

# specifying the condition of Good
df1.loc[condition_reviews & condition_rating, 'Success'] = 'Good'

#  the updated Dataset
print(df1[['number_of_reviews', 'review_scores_rating', 'Success']])
```

```
       number_of_reviews  review_scores_rating Success
0                    216                  4.57     Bad
1                     38                  4.82    Good
2                     41                  4.80    Good
3                     94                  4.80    Good
4                    180                  4.62     Bad
...                  ...                   ...     ...
87941                  0                  5.00     Bad
87942                  0                  5.00     Bad
87943                  0                  5.00     Bad
87944                  0                  5.00     Bad
87945                  0                  5.00     Bad

[87946 rows x 3 columns]
```

- Determine the average of the columns "number_of_reviews" and "review_scores_rating" in the Dataset df1.
- Assign the value "Bad" to the new column "Success" for every row in df1.
- Set up criteria to determine if the number of reviews ('number_of_reviews') and the review scores ('review_scores_rating') are more than or equal to the computed mean ('reviews_threshold').
- Rows that meet both requirements should have their "Success" column updated to "Good."
- Show the chosen columns of df1 ('number_of_reviews','review_scores_rating', 'Success').

Listings are categorized as "Good" or "Bad" by this code according to factors including the quantity and quality of reviews. The listing is marked as "Good" if both the number of reviews and the review score are more than or equal to the mean; if not, it is marked as "Bad."

In [147]:
```
#Checking count of success using value_counts()
success_counts = df1['Success'].value_counts()

# Display the counts
print(success_counts)
```

```
Success
Bad     74298
Good    13648
Name: count, dtype: int64
```

In [148]:
```
df1.columns # checking dataset is updated with Sucess
```

Out[148]:
```
Index(['host_since', 'host_response_time', 'host_response_rate',
       'host_acceptance_rate', 'host_is_superhost', 'host_listings_count',
       'property_type', 'room_type', 'accommodates', 'bedrooms', 'beds',
       'amenities', 'price', 'number_of_reviews', 'latitude', 'longitude',
       'neighbourhood_cleansed', 'review_scores_rating', 'reviews_per_month',
       'Success'],
      dtype='object')
```

In [149]:

```
#importing  all library required
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler

#Defining the Independent and dependent variable and target variable is Sucess
X = df1.drop('Success', axis=1)
y = df1['Success']

# Initialising the LabelEncoder
label_encoder = LabelEncoder()

# Applying  Label Encoding to categorical columns
for col in ['host_response_time', 'property_type', 'room_type', 'neighbourhood_cleansed']:
    df1[col] = label_encoder.fit_transform(df1[col])

# Create features (X) and target variable (y)
X = df1.drop(['Success'], axis=1)
y = df1['Success']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


#Scailing the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_imputed)


# Initialize Logistic Regression model
logreg_model = LogisticRegression(random_state=42)

# Fit the model on the training data
logreg_model.fit(X_train_scaled, y_train)

# Predictions on the test set
y_pred = logreg_model.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", classification_rep)
```

```
Accuracy: 0.8987492893689596
Confusion Matrix:
 [[14586   259]
 [ 1522  1223]]
Classification Report:
               precision    recall  f1-score   support

         Bad       0.91      0.98      0.94     14845
        Good       0.83      0.45      0.58      2745

    accuracy                           0.90     17590
   macro avg       0.87      0.71      0.76     17590
weighted avg       0.89      0.90      0.89     17590
```

- Accuaracy of the model is 89.87% .This shows how accurate the model is overall in predicting both classifications (Good and Bad).
- The model's predictions are broken out in great depth in the confusion matrix.The estimated classes are represented by columns, and the actual classifications ('Bad' and 'Good') are represented by rows.
This model showed that 14586 cases were properly predicted as "Bad," 1223 instances as "Good," 259 instances as "Good" (false positive), and 1522 instances as "Bad" (false negative) were wrongly anticipated.

Percsion :The fraction of accurately predicted 'Good' cases among all instances anticipated as 'Good'.

F1-score: The F1-score is the harmonic mean of accuracy and recall. It strikes a balance between accuracy and recall. Support: The number of real occurrences of each class in the dataset. Recall: The recall for 'Good' is 0.45, which is the proportion of properly predicted 'Good' cases out of all actual 'Good' instances.

In [151]:
```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix




# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Decision Tree Model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Predictions on the test set
y_pred = dt_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 1.0
Confusion Matrix:
[[14845     0]
 [    0  2745]]
Classification Report:
              precision    recall  f1-score   support

         Bad       1.00      1.00      1.00     14845
        Good       1.00      1.00      1.00      2745

    accuracy                           1.00     17590
   macro avg       1.00      1.00      1.00     17590
weighted avg       1.00      1.00      1.00     17590
```

```
<li> The Accuracy of the model is 1 which means 100 .</li>

Since the Ratio of Bad and good list is not 50 : 50 ratio.While it may seem ideal to achieve perfect accuracy and
performance metrics, doing so increases the risk of overfitting.

<li> Reason for overfitting </li>
Missing data and noise in data ,lack of reguarlisation



<li> Solution </li>
Methods of Regularization:
To penalize large weights in the model, introduce regularization techniques like L1 or L2 regularization. By doing
this, the model is kept from fitting the training set too closely.

Cross- validation
Utilizing this methods such as k-fold cross-validation to evaluate the model's performance across several data
subsets. This may offer a more thorough assessment of the model's capacity for generalization.

Engineering Features:
Evaluate and improve the features you have. Eliminate any superfluous or pointless features that could be making the
model overfit the data's noise.
```

The Importance of Features used while creating the target variable

The significance of this process lies in the creation of a binary classification ('Good' or 'Bad') based on a combination of two important metrics: the number of reviews and the rating of the review scores. This classification can be useful for identifying successful or popular items, services, or entities within the dataset, among other things. It allows you to divide instances into two groups based on whether they have more or fewer reviews and ratings than the average. This type of categorization can help to simplify analysis and decision-making processes in a variety of applications, including customer satisfaction, product evaluation, and service quality evaluation.

The review rating and review score are significant because they provide concise and quantifiable measures of user satisfaction and product or service quality, which influence consumer decisions and business success. High ratings indicate positive experiences, which attract more customers, whereas low ratings may indicate issues that need to be addressed, affecting reputation and profitability. These metrics provide valuable feedback, guiding businesses to improve customer satisfaction and make informed decisions in order to remain competitive in the market.

Dicussion

we were given the airnb dataset with 75 columns so First i started removing the uncessary columns and selecting which columns are most important . Then comes data visualisation and seeing which room type has highest value and then comes checking data types and missing values and handling it which plays cricual and important Pre Processing step

Next important step was feature enginerring which means create a new column with existing one so i took no of review and review score rating compare there threshold and specify some condition telling it is good or bad

After getting the target variable and i did define dependent and independent variable for model ,scale the features ,applied the logistic and Decision mpdel and predictive the values

so for logistic Regression i got 89 % accuary and Decision tree 100% and in decision tree it tells that model is overfit