Airbnb Revenue Analysis

December 2, 2024

1 Airbnb Revenue Analysis

By: Jay Chang, Jeffrey Fernandez, Jack Keaveny

There has been a recent influx of clients interested in entering the short-term rental market, specifically through touristic Airbnb properties. Some motivations for entering this new market is as follows:

- People spent $4.72 trillion on leisure tourism in 2019.
- Considering a booking value of $38 billion in 2019 for Airbnb, the market share was only 0.8% of that market.
- As of 2024, Airbnb has over 8 million active listings spanning 220 countries.
- Since its launch, hosts have earned over $250 billion through Airbnb.
- Airbnb has more than 200 million active users worldwide.
- In 2023, the average host in the U.S. earned $14,000 from Airbnb.
- The United States has the highest number of Airbnb listings, featuring around 2.25 million active properties.

Considering the overall success in the U.S, our clients have tasked our consulting agency with providing actionable investment recommendations specifically for Spain, a popular tourist destination. Primarily, we will identify which key property attributes maximize profitability, guiding our clients in making informed decisions about where and how to invest. We will focus on utilizing data from 2024.

Our action plan is as follows:

1. Determine the best city to invest in within Spain
2. Identify the best property types to invest in
3. Determine what property attributes influence revenue the most
4. Create models that predict which variables influence revenue the most

The final deliverable will be a clear, data-driven presentation outlining the most lucrative opportunities and strategic insights for navigating the competitive landscape of short-term rentals.

---

2 Loading, Cleaning, and Filtering the Data

Before being able to answer the request, we must implement all the necessary preparations and work.

This includes loading the datasets and simultaneously cleaning and filtering the data to account for any errors, inconsistencies, or bias.

2.1 Loading the Data

[1]:
```
#Mounting Drive to be able to access Files
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
```

[2]:
```
import os
import pandas as pd
import numpy as np

#Linking folder to variable
Airbnb_dir = '/content/drive/MyDrive/Final_Project/Spain'
dataframes = {} #creating empty dictionary



# loop itterates through folder of cities in apsin with their Airbnb data␣
 ↪making each city a key pointing to a
# list of 3 data frames: listings, reviews, and calendar

for foldername in os.listdir(Airbnb_dir):
  folder_path = os.path.join(Airbnb_dir, foldername)
  key = foldername
  dataframes[key] = []
  if not os.path.exists(folder_path):
    print(f"Error: Directory '{folder_path}' not found.")
  else:
    for filename in os.listdir(folder_path):
      if filename.endswith(".csv.gz"):
        filepath = os.path.join(folder_path, filename)
        try:
          df = pd.read_csv(filepath)
          print(f"DataFrame created for: {foldername}, {filename}")

          # add dataframe to dict
          dataframes[key].append(df)
```

```
        except pd.errors.ParserError as e:
          print(f"Error parsing {filename}: {e}")
        except Exception as e:
          print(f"An error occurred while processing {filename}: {e}")
      else:
        print(f"Skipping non-CSV.gz file: {filename}")

    print("-" * 20)
```

```
DataFrame created for: Madrid, listings.csv.gz

DataFrame created for: Madrid, calendar.csv.gz
Skipping non-CSV.gz file: listings.csv

DataFrame created for: Madrid, reviews.csv.gz
Skipping non-CSV.gz file: reviews.csv
Skipping non-CSV.gz file: neighbourhoods.csv
Skipping non-CSV.gz file: neighbourhoods.geojson
--------------------

DataFrame created for: Mallorca, listings.csv.gz

/tmp/ipython-input-3991234951.py:25: DtypeWarning: Columns (4) have mixed types.
Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(filepath)

DataFrame created for: Mallorca, calendar.csv.gz
Skipping non-CSV.gz file: listings.csv

DataFrame created for: Mallorca, reviews.csv.gz
Skipping non-CSV.gz file: reviews.csv
Skipping non-CSV.gz file: neighbourhoods.csv
Skipping non-CSV.gz file: neighbourhoods.geojson
--------------------

DataFrame created for: Barcelona, listings.csv.gz

/tmp/ipython-input-3991234951.py:25: DtypeWarning: Columns (4) have mixed types.
Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(filepath)

DataFrame created for: Barcelona, calendar.csv.gz
Skipping non-CSV.gz file: listings.csv

DataFrame created for: Barcelona, reviews.csv.gz
Skipping non-CSV.gz file: reviews.csv
Skipping non-CSV.gz file: neighbourhoods.csv
Skipping non-CSV.gz file: neighbourhoods.geojson
--------------------

DataFrame created for: Menorca, listings.csv.gz
```

```
/tmp/ipython-input-3991234951.py:25: DtypeWarning: Columns (4) have mixed types.
Specify dtype option on import or set low_memory=False.
  df = pd.read_csv(filepath)

DataFrame created for: Menorca, calendar.csv.gz

DataFrame created for: Menorca, reviews.csv.gz
Skipping non-CSV.gz file: listings.csv
Skipping non-CSV.gz file: reviews.csv
Skipping non-CSV.gz file: neighbourhoods.csv
Skipping non-CSV.gz file: neighbourhoods.geojson
--------------------

DataFrame created for: Malaga, listings.csv.gz

DataFrame created for: Malaga, calendar.csv.gz

DataFrame created for: Malaga, reviews.csv.gz
Skipping non-CSV.gz file: listings.csv
Skipping non-CSV.gz file: reviews.csv
Skipping non-CSV.gz file: neighbourhoods.csv
Skipping non-CSV.gz file: neighbourhoods.geojson
--------------------
```

[3]:
```
#showing off Dictionary Keys
dataframes.keys()
```

[3]:
```
dict_keys(['Madrid', 'Mallorca', 'Barcelona', 'Menorca', 'Malaga'])
```

[4]:
```
import pandas as pd

listings = pd.DataFrame()
calendar = pd.DataFrame()
reviews = pd.DataFrame()

for key, value in dataframes.items():
  if value and isinstance(value[0], pd.DataFrame):  # Check if the list is not␣
 ↪empty and contains a DataFrame
    df = value[0].copy()
    df['city'] = key  #add a column to tell which key the row came from
    listings = pd.concat([listings, df], ignore_index=True)

for key, value in dataframes.items():
  if value and isinstance(value[1], pd.DataFrame):  # Check if the list is not␣
 ↪empty and contains a DataFrame
    df = value[1].copy()
    df['city'] = key  #add a column to tell which key the row came from
    calendar = pd.concat([calendar, df], ignore_index=True)
```

```
for key, value in dataframes.items():
  if value and isinstance(value[2], pd.DataFrame):  # Check if the list is not␣
 ↪empty and contains a DataFrame
    df = value[2].copy()
    df['city'] = key  #add a column to tell which key the row came from
    reviews = pd.concat([reviews, df], ignore_index=True)
```

2.2 Cleaning and Filtering

Now that our data is loaded and the respective dataframes are created, we can start cleaning and filtering our data.

[5]:
```
#list of columns deemed unnecessary in Listings dataframe
useless_col = [
               'source',
               'listing_url',
               'scrape_id',
               'last_scraped',
               'name',
               'picture_url',
               'host_id',
               'host_url',
               'host_name',
               'host_since',
               'host_location',
               'host_about',
               'host_thumbnail_url',
               'host_picture_url' ,
               'host_neighbourhood',
               'host_listings_count' ,
               'host_total_listings_count',
               'host_has_profile_pic',
               'host_identity_verified',
               'latitude',
               'longitude',
               'minimum_minimum_nights',
               'maximum_minimum_nights',
               'minimum_maximum_nights',
               'maximum_maximum_nights',
               'minimum_nights_avg_ntm',
               'maximum_nights_avg_ntm',
               'calendar_updated',
               'calendar_last_scraped',
               'availability_30',
               'availability_60',
               'availability_90',
               'first_review',
```

```
                'license',
                'instant_bookable',
                'neighbourhood', #the cleansed is better
                'neighborhood_overview',
                'neighbourhood_group_cleansed', #removed cause Mallorca,Menorca,␣
 ↪and Malaga have no values in this column
                'has_availability'
]


#dropping useless columns:
listings2 = listings.drop(columns=useless_col, errors='ignore')

#remove rows with na values from listings 2
listings2 = listings2.dropna()

# Convert 'price' column to numeric, handling errors
listings2['price'] = listings2['price'].astype(str).str.replace(r'[$,]', '',␣
 ↪regex=True).astype(float)

# Convert percentage strings to decimal floats for specified columns
for col in ['host_response_rate', 'host_acceptance_rate']:
    if col in listings2.columns:
        listings2[col] = pd.to_numeric(listings2[col].str.rstrip('%'),␣
 ↪errors='coerce') / 100

# Filter listings2 based on specified criteria
listings2 = listings2[
    (listings2['price'] >= 5) &
    (listings2['price'] <= 9999) &
    (listings2['reviews_per_month'] <= 40) &
    (listings2['minimum_nights'] <= 365) &
    (listings2['beds'] > 0)
]

# Reset the index of the filtered DataFrame
listings2 = listings2.reset_index(drop=True)
```

Here, we have simply filtered out unnecesary and redundant columns, as well as removing rows with "na" values from our dataframe "listings2". Additionally, we have set boundaries for specific variables

[6]:
```
# doing the same for Reviews and Calendar(no real filtering or extensive␣
 ↪cleaning needed)

rev_useless = ['id', 'reviewer_id', 'reviewer_name']
reviews2 = reviews.drop(columns=rev_useless, errors='ignore')
```

```
cal_useless = ['minimum_nights', 'maximum_nights', 'adjusted_price' ]
calendar2 = calendar.drop(columns=cal_useless, errors='ignore')
```

[7]:
```
listings2.describe(include=[np.number])
```

[7]:

	id	host_response_rate	host_acceptance_rate	accommodates
count	4.640500e+04	46405.000000	46405.000000	46405.000000
mean	4.411750e+17	0.961762	0.915048	4.217369
std	4.757052e+17	0.124220	0.178492	2.416641
min	1.867400e+04	0.000000	0.000000	1.000000
25%	2.888433e+07	0.980000	0.940000	2.000000
50%	5.339035e+07	1.000000	0.990000	4.000000
75%	9.159319e+17	1.000000	1.000000	6.000000
max	1.240331e+18	1.000000	1.000000	16.000000

	bathrooms	bedrooms	beds	price	minimum_nights
count	46405.000000	46405.000000	46405.000000	46405.000000	46405.000000
mean	1.574604	1.953022	2.926236	201.751234	5.854929
std	0.955221	1.291442	2.091047	454.597958	12.360311
min	0.000000	0.000000	1.000000	9.000000	1.000000
25%	1.000000	1.000000	1.000000	86.000000	1.000000
50%	1.000000	2.000000	2.000000	140.000000	2.000000
75%	2.000000	3.000000	4.000000	222.000000	4.000000
max	19.000000	25.000000	40.000000	9999.000000	365.000000

	maximum_nights	…	review_scores_cleanliness	review_scores_checkin
count	46405.000000	…	46405.000000	46405.000000
mean	590.056309	…	4.656717	4.749452
std	445.715928	…	0.445765	0.410848
min	1.000000	…	1.000000	1.000000
25%	330.000000	…	4.540000	4.690000
50%	365.000000	…	4.780000	4.870000
75%	1125.000000	…	4.940000	5.000000
max	1125.000000	…	5.000000	5.000000

	review_scores_communication	review_scores_location
count	46405.000000	46405.000000
mean	4.747081	4.729160
std	0.431972	0.362606
min	1.000000	1.000000
25%	4.690000	4.640000
50%	4.880000	4.830000
75%	5.000000	4.960000
max	5.000000	5.000000

	review_scores_value	calculated_host_listings_count

```
count         46405.000000                    46405.000000
mean              4.526485                       46.427497
std               0.492442                      116.418354
min               1.000000                        1.000000
25%               4.400000                        2.000000
50%               4.640000                        6.000000
75%               4.810000                       37.000000
max               5.000000                      838.000000

       calculated_host_listings_count_entire_homes  \
count                                 46405.000000
mean                                     44.040039
std                                     115.498868
min                                       0.000000
25%                                       1.000000
50%                                       4.000000
75%                                      35.000000
max                                     835.000000

       calculated_host_listings_count_private_rooms  \
count                                  46405.000000
mean                                       2.289969
std                                       16.551036
min                                        0.000000
25%                                        0.000000
50%                                        0.000000
75%                                        1.000000
max                                      356.000000

       calculated_host_listings_count_shared_rooms  reviews_per_month
count                                 46405.000000       46405.000000
mean                                      0.076134           1.513234
std                                       1.087084           1.766894
min                                       0.000000           0.010000
25%                                       0.000000           0.330000
50%                                       0.000000           0.910000
75%                                       0.000000           2.120000
max                                      26.000000          36.970000

[8 rows x 26 columns]
```

[8]:
```
listings2.describe(include=['object', 'category'])
```

[8]:
```
                                        description host_response_time  \
count                                         46405              46405
unique                                        41706                  4
top     Enjoy the simplicity of this quiet, central home.     within an hour
```

```
freq                                                  116              37107

       host_is_superhost  host_verifications neighbourhood_cleansed  \
count              46405               46405                 46405
unique                 2                   6                   268
top                    f  ['email', 'phone']                Centro
freq               32625               35987                  4233

             property_type        room_type bathrooms_text  \
count                46405            46405          46405
unique                  79                4             46
top     Entire rental unit  Entire home/apt         1 bath
freq                 23460            37261          21474

                                                amenities last_review    city
count                                               46405       46405   46405
unique                                              43294        1518       5
top     ["Heating", "Hair dryer", "Toaster", "TV", "Ho…  2024-09-08  Madrid
freq                                                   67         955   14902
```

[9]:
```
#Encoding for certain categorical variables to numerical

from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import OrdinalEncoder

# Ordinal Encoding on host_is_super_host and host_response_time
categories = [
    ["t", "f"],
    ['within an hour', 'within a few hours', 'within a day', 'a few days or␣
 ↪more']
]

encoder = OrdinalEncoder(categories=categories)

listings2[['host_is_superhost', 'host_response_time']] = encoder.fit_transform(
    listings2[['host_is_superhost', 'host_response_time']]
              )

# One-Hot encoding on Nominal Data
listings2 = pd.get_dummies(listings2, columns=['room_type'],
                           prefix=['room_type'],
                          drop_first=True)

# turning true and falses in encoded room types to 0 and 1
for col in listings2.filter(like='room_type_').columns:
    listings2[col] = listings2[col].astype(int)
```

[10]:
```
#one hot encode 'city' in listings2 as type int

# One-hot encode 'city' column in listings2
listings2 = pd.get_dummies(listings2, columns=['city'], prefix='city',␣
 ↪drop_first=True)

# Convert one-hot encoded columns to integers
for col in listings2.filter(like='city_').columns:
    listings2[col] = listings2[col].astype(int)
```

[11]:
```
#list all variables in listings2 and types

import pandas as pd
import numpy as np

# Assuming 'listings2' is already defined as in your provided code.

# Get a list of all variables in listings2 and their types.
variable_types = listings2.dtypes

# Print the variables and their types.
for variable, var_type in variable_types.items():
  print(f"Variable: {variable}, Type: {var_type}")
```

```
Variable: id, Type: int64
Variable: description, Type: object
Variable: host_response_time, Type: float64
Variable: host_response_rate, Type: float64
Variable: host_acceptance_rate, Type: float64
Variable: host_is_superhost, Type: float64
Variable: host_verifications, Type: object
Variable: neighbourhood_cleansed, Type: object
Variable: property_type, Type: object
Variable: accommodates, Type: int64
Variable: bathrooms, Type: float64
Variable: bathrooms_text, Type: object
Variable: bedrooms, Type: float64
Variable: beds, Type: float64
Variable: amenities, Type: object
Variable: price, Type: float64
Variable: minimum_nights, Type: int64
Variable: maximum_nights, Type: int64
Variable: availability_365, Type: int64
Variable: number_of_reviews, Type: int64
Variable: number_of_reviews_ltm, Type: int64
Variable: number_of_reviews_l30d, Type: int64
Variable: last_review, Type: object
```

```
Variable: review_scores_rating, Type: float64
Variable: review_scores_accuracy, Type: float64
Variable: review_scores_cleanliness, Type: float64
Variable: review_scores_checkin, Type: float64
Variable: review_scores_communication, Type: float64
Variable: review_scores_location, Type: float64
Variable: review_scores_value, Type: float64
Variable: calculated_host_listings_count, Type: int64
Variable: calculated_host_listings_count_entire_homes, Type: int64
Variable: calculated_host_listings_count_private_rooms, Type: int64
Variable: calculated_host_listings_count_shared_rooms, Type: int64
Variable: reviews_per_month, Type: float64
Variable: room_type_Hotel room, Type: int64
Variable: room_type_Private room, Type: int64
Variable: room_type_Shared room, Type: int64
Variable: city_Madrid, Type: int64
Variable: city_Malaga, Type: int64
Variable: city_Mallorca, Type: int64
Variable: city_Menorca, Type: int64
```

[12]:
```
calendar2['date'] = pd.to_datetime(calendar2['date'])
calendar2['year'] = calendar2['date'].dt.year
calendar2['month'] = calendar2['date'].dt.month
calendar2['day'] = calendar2['date'].dt.day
```

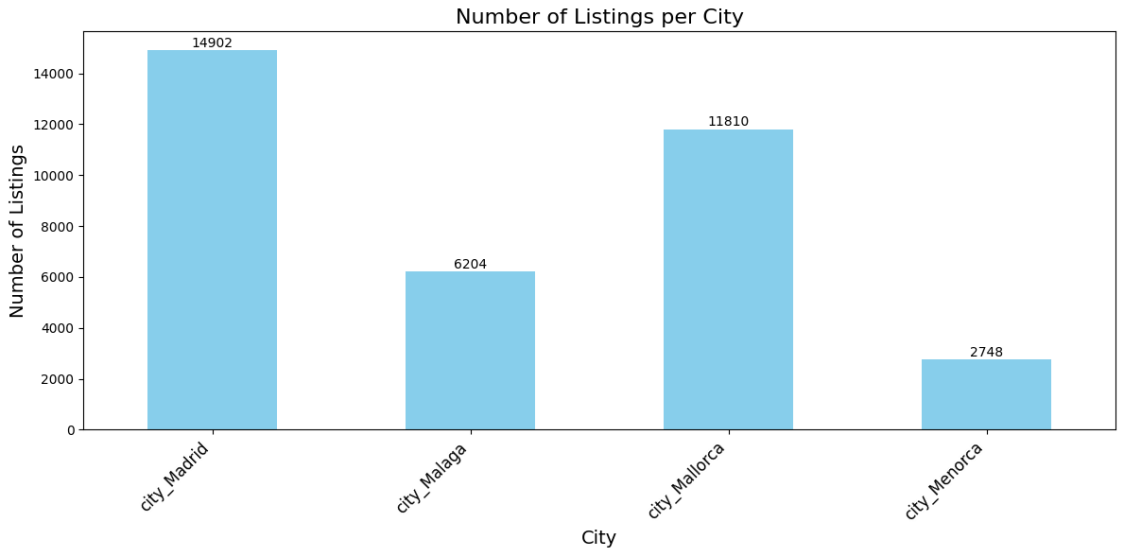
3 Visualizing the Data

With our data loaded, cleaned, and filtered, we can now visualize our data to have a better sense of how certain variables correlate with each other.

[13]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

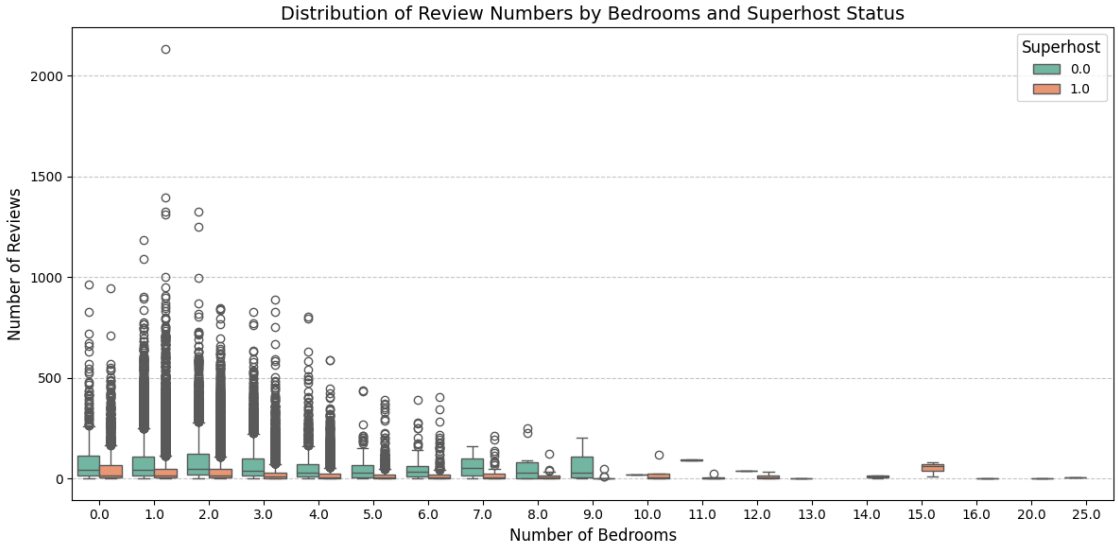
[14]:
```
# Create a bar graph for the city dummy variables
city_columns = listings2.filter(like='city_').columns
city_counts = listings2[city_columns].sum()

plt.figure(figsize=(12, 6))
bars = city_counts.plot(kind='bar', color='skyblue')
plt.title('Number of Listings per City', fontsize=16)
plt.xlabel('City', fontsize=14)
plt.ylabel('Number of Listings', fontsize=14)
plt.xticks(rotation=45, ha='right', fontsize=12)
plt.tight_layout()

# Annotate the bars with their values
for i, v in enumerate(city_counts):
```

```
    plt.text(i, v + 10, str(int(v)), ha='center', va='bottom', fontsize=10)

plt.show()
```



Here, we have created a bar graph of the number of listings per city, which can help us determine which city in Spain is the best to invest in. Although it won't tell us exactly which one's are more profitable than others, it does shed light on which cities have the most properties, and in turn, which cities are the most "popular" to visit, possibly translating to profitability. However, we will explore this further.

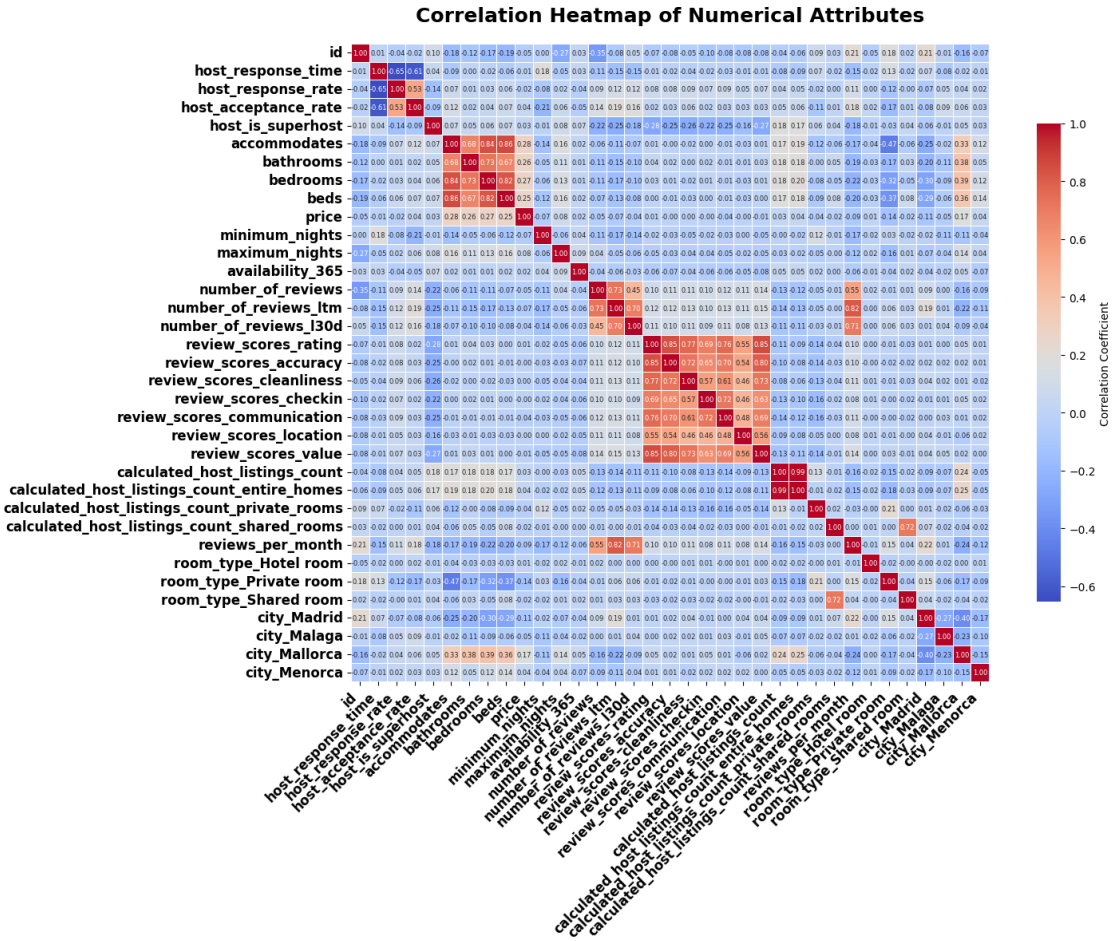
[15]:
```
#Box Plot: Reviews by Bedrooms and Superhost Status
plt.figure(figsize=(12, 6))
sns.boxplot(data=listings2, x='bedrooms', y='number_of_reviews',␣
 ↪hue='host_is_superhost', palette='Set2')
plt.title('Distribution of Review Numbers by Bedrooms and Superhost Status',␣
 ↪fontsize=14)
plt.xlabel('Number of Bedrooms', fontsize=12)
plt.ylabel('Number of Reviews', fontsize=12)
plt.legend(title='Superhost', title_fontsize=12, fontsize=10, loc='upper right')
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```



Now we have created a box plot of the number of reviews based on the number of bedrooms, taking into account of the Superhost status. At first glance, the data looks right skewed where there are a lot more reviews for properties with lower amounts of bedrooms, and a lot fewer reviews for properties with a higher number of bedrooms. Although this could point to properties with a lower number of bedrooms having more reviews (and in turn, possibly being more profitable and "popular" for consumers), the skewed data could be attributed to the popularity and prevalence of properties with a lower number of bedrooms, rather than signs of profitability. However, we will examine this further in our analysis and predictive model.

[16]:
```
#Correlation matrix

# Calculate the correlation matrix for numerical attributes
numerical_attributes = listings2.select_dtypes(include=np.number)
correlation_matrix = numerical_attributes.corr()



plt.figure(figsize=(16, 12))
sns.heatmap(
    correlation_matrix,
    annot=True,
    fmt=".2f",
    cmap="coolwarm",  # Reverted to the original color scheme
    linewidths=0.5,
    linecolor='white',
    annot_kws={"size": 6},  # Further reduced annotation font size
    cbar_kws={"shrink": 0.75, "label": "Correlation Coefficient"},
    square=True
)
```

```
plt.title('Correlation Heatmap of Numerical Attributes', fontsize=18,␣
 ↪fontweight='bold', pad=20)
plt.xticks(fontsize=12, rotation=45, ha='right', fontweight='bold')
plt.yticks(fontsize=12, fontweight='bold')
plt.tight_layout()
plt.show()
```



4 Variable Creation: Calculating Revenue

With our visualizations done, we must now calculate the revenue, an integral part of our property attribute analysis. To do this, we will multiply the price by the number of bookings (or rather number of days NOT available in availability_365) per property in 2024. Although the values of revenue that we calculate don't factor in hidden costs (% that AirBnB takes, maintenance, etc.), it is still a viable estimate of the revenue and a strong indicator of profitability.

[17]:
```
#2024 revenue calculations for unique listings
data_2024_unfiltered = calendar2[calendar2['year'] == 2024]
```

```
data_2024 = data_2024_unfiltered[data_2024_unfiltered['listing_id'].
 ↪isin(listings2['id'])]
data_2024['price'] = data_2024['price'].replace({'\$': '', ',': ''},␣
 ↪regex=True).astype(float)
data_2024['available'] = data_2024['available'].map({'f': False, 't': True})
booked_days = data_2024[data_2024['available'] == False]
revenue_by_property = booked_days.groupby('listing_id')['price'].sum().
 ↪reset_index(name='2024_revenue')
```

```
/tmp/ipython-input-3184644723.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_2024['price'] = data_2024['price'].replace({'\$': '', ',': ''},
regex=True).astype(float)
/tmp/ipython-input-3184644723.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data_2024['available'] = data_2024['available'].map({'f': False, 't': True})
```

[18]:
```
# Check if every unique listing_id in revenue_by_property exists in␣
 ↪listings2['id']
missing_listings = set(revenue_by_property['listing_id']) - set(listings2['id'])

if missing_listings:
    print("The following listing IDs from revenue_by_property are not found in␣
 ↪listings2:")
    print(missing_listings)
else:
    print("All listing IDs in revenue_by_property are present in listings2.")
```

```
All listing IDs in revenue_by_property are present in listings2.
```

[19]:
```
# Merge the revenue data with listings2
listings2 = pd.merge(listings2, revenue_by_property, left_on='id',␣
 ↪right_on='listing_id', how='left')

# Drop the extra 'listing_id' column
#listings2 = listings2.drop('listing_id', axis=1)
```

[20]:
```
listings2.select_dtypes(include=['object', 'category']).columns.tolist()
```

[20]:
```
['description',
 'host_verifications',
 'neighbourhood_cleansed',
 'property_type',
 'bathrooms_text',
 'amenities',
 'last_review']
```

[21]:
```
for col in listings2.select_dtypes(include=['object', 'category']).columns.
↪tolist():
 print(f"Unique values for {col}:")
 print(listings2[col].unique())
 print("-" * 20)
```

```
Unique values for description:
["Charming 40m2 flat in the central neighbourhood of Malasaña. Comfortable and
cosy, it can accommodate up to 4 people in its 2 independent bedrooms with
double beds. With excellent natural light as it is exterior. It has a full
bathroom with shower. Includes wifi connection, coffee machine, bed linen and
towels. For more details, please see the full description below, don't hesitate
to ask!"
 'Enjoy a luxury experience in this central accommodation *DE 29 MTS2.'
 'Malasaña is a dynamic student area with cafes, bakeries and vintage clothing
stores. The gastronomic offer of the neighborhood is very complete and varied.
We can find numerous classic taverns where you can enjoy a refreshing beer
accompanied by your usual tapa. The terraces are another of the protagonists of
this area. In the evening, live bands and DJs perform in trendy dance and rock
music venues.'
 …
 "A beautiful apartment very close to the beach, just a 2 minute walk, quiet and
with green areas.<br />It consists of double bed, living room-kitchen and a
bathroom. It's fully stocked with cookware, towels, and sheets.<br /><br />Come
enjoy the Costa Malagueña and  taste the best dishes on the wonderful Paseo
Maritimo el Pedregal.<br />Great variety of restaurants where you can taste the
best dishes facing the Mediterranean Sea throughout the year with an enviable
climate."
 'Brand new downtown apartment. <br /><br />It is located a 12-minute walk from
Plaza de la Merced and the Roman theater, ideal for families with children or a
group of friends. <br /><br />Its central location, but far enough from the
noise of the historic center is perfect for you to enjoy an unforgettable
vacation! <br /><br />It has a taxi and bus stop at the door of the house in
addition to a free parking area for cars at the door of the house.'
 'Break with your day to day in this room located next to a large park, less
than five minutes away from the beach, 10 minutes from the train stations
"Málaga María Zambrano" and Malaga buses and about 20 minutes away from the city
center, with bus and metro stops to go to said very close city center and also
close stops from both the train and the bus that connects the airport with the
```

```
city.']
--------------------
Unique values for host_verifications:
["['email', 'phone']" "['email', 'phone', 'work_email']" "['phone']"
 "['phone', 'work_email']" "['email']" '[]']
--------------------
Unique values for neighbourhood_cleansed:
['Universidad' 'Justicia' 'Argüelles' 'Palos de Moguer' 'Comillas'
 'Trafalgar' 'Palacio' 'Embajadores' 'Cármenes' 'Puerta del Angel'
 'Casa de Campo' 'Castilla' 'Quintana' 'Los Rosales' 'Pacífico' 'Cortes'
 'Ventas' 'Opañel' 'Orcasur' 'Guindalera' 'Puerta Bonita' 'Cuatro Caminos'
 'Nueva España' 'Almendrales' 'San Pascual' 'San Isidro' 'Valdeacederas'
 'Acacias' 'Sol' 'Prosperidad' 'Castillejos' 'Bellas Vistas' 'Abrantes'
 'Estrella' 'Atocha' 'Pilar' 'San Diego' 'Lista' 'Almenara' 'Campamento'
 'Moscardó' 'Peñagrande' 'Casco Histórico de Vicálvaro' 'Recoletos' 'Goya'
 'Berruguete' 'Hispanoamérica' 'Adelfas' 'San Fermín' 'Concepción'
 'Castellana' 'Vista Alegre' 'Palomeras Bajas' 'Arapiles' 'Aluche'
 'Numancia' 'Ciudad Jardín' 'Gaztambide' 'Valdezarza' 'El Viso' 'Rejas'
 'Valdemarín' 'Pueblo Nuevo' 'San Andrés' 'Jerónimos'
 'Casco Histórico de Barajas' 'Vallehermoso' 'Orcasitas' 'Imperial'
 'Lucero' 'Ciudad Universitaria' 'Delicias' 'Aravaca' 'Pinar del Rey'
 'Corralejos' 'Niño Jesús' 'Valdefuentes' 'Buenavista'
 'Casco Histórico de Vallecas' 'Fuente del Berro' 'Canillas' 'Rios Rosas'
 'Simancas' 'Los Angeles' 'Timón' 'Pradolongo' 'San Juan Bautista'
 'Portazgo' 'Entrevías' 'Legazpi' 'Ibiza' 'Chopera' 'Marroquina' 'Hellín'
 'Salvador' 'Canillejas' 'Zofío' 'Alameda de Osuna' 'Arcos' 'Costillares'
 'Piovera' 'Mirasierra' 'Pavones' 'San Cristobal' 'Almagro'
 'Santa Eugenia' 'Vinateros' 'Rosas' 'Amposta' 'Valverde' 'La Paz'
 'Palomeras Sureste' 'Colina' 'Aguilas' 'Cuatro Vientos' 'Ambroz'
 'Apostol Santiago' 'Fontarrón' 'El Goloso' 'Aeropuerto' 'Media Legua'
 'Butarque' 'Palomas' 'Horcajo' 'El Pardo' 'Fuentelareina' 'El Plantío'
 'Palma de Mallorca' 'Selva' 'Sa Pobla' 'Sóller' 'Deyá'
 'Lloret de Vistalegre' 'Pollença' 'Alcúdia' 'Santanyí' 'Búger'
 'Santa Margalida' 'Felanitx' 'Esporles' 'Calvià' 'Campos' 'Manacor'
 'Llubí' 'Montuïri' 'Llucmajor' 'Porreres' 'Mancor de la Vall' 'Lloseta'
 'Ses Salines' 'Petra' 'Fornalutx' 'Algaida' 'Campanet' 'Andratx'
 'Valldemossa' 'Capdepera' 'Sant Llorenç des Cardassar' 'Muro' 'Inca'
 'Consell' 'Puigpunyent' 'Artà' 'Bunyola' 'Sencelles' 'Costitx'
 'Banyalbufar' 'Sineu' 'Vilafranc de Bonany' 'Son Servera' 'Alaró'
 'Santa María del Camí' 'Estellencs' 'Binissalem' 'Marratxí' 'Escorca'
 'Sant Joan' 'Ariany' 'Maria de la Salut' 'Santa Eugènia'
 'la Sagrada Família' 'el Besòs i el Maresme'
 "el Camp d'en Grassot i Gràcia Nova" 'el Barri Gòtic'
 'Sant Pere, Santa Caterina i la Ribera' "la Dreta de l'Eixample"
 "el Camp de l'Arpa del Clot" 'el Poblenou' 'les Corts'
 'la Vila de Gràcia' 'el Raval' 'el Parc i la Llacuna del Poblenou'
 'el Fort Pienc' "la Nova Esquerra de l'Eixample" 'el Clot'
 'Diagonal Mar i el Front Marítim del Poblenou' 'Sant Antoni' 'Sants'
```

```
 'Pedralbes' 'el Poble Sec' 'el Guinardó'
 "l'Antiga Esquerra de l'Eixample" 'Sant Gervasi - Galvany'
 'el Baix Guinardó' 'el Congrés i els Indians'
 'la Vila Olímpica del Poblenou' 'la Bordeta' 'Sant Martí de Provençals'
 'la Barceloneta' 'el Putxet i el Farró' 'la Maternitat i Sant Ramon'
 'Sarrià' 'Provençals del Poblenou' "la Font d'en Fargues" 'el Carmel'
 'Vallcarca i els Penitents' 'Navas' 'Sants - Badal' 'Hostafrancs'
 'la Salut' 'les Tres Torres' 'la Teixonera' 'Can Baró'
 'Sant Gervasi - la Bonanova' 'la Font de la Guatlla'
 'Vilapicina i la Torre Llobeta' 'la Marina del Prat Vermell' 'el Coll'
 'la Verneda i la Pau' 'la Marina de Port' 'la Sagrera' 'Sant Andreu'
 'la Guineueta' 'Vallvidrera, el Tibidabo i les Planes' "la Vall d'Hebron"
 'Horta' 'la Prosperitat' 'les Roquetes' 'Porta' 'el Turó de la Peira'
 'Verdun' 'Sant Genís dels Agudells' 'la Trinitat Vella' 'la Clota'
 'Montbau' 'el Bon Pastor' 'Canyelles' 'la Trinitat Nova' 'Baró de Viver'
 'Es Mercadal' 'Ciutadella de Menorca' 'Alaior' 'Mahón' 'Sant Lluís'
 'Es Castell' 'Ferreries' 'Es Migjorn Gran' 'Centro'
 'Teatinos-Universidad' 'Este' 'Cruz De Humilladero' 'Carretera de Cadiz'
 'Bailen-Miraflores' 'Palma-Palmilla' 'Ciudad Jardin' 'Campanillas'
 'Puerto de la Torre' 'Churriana']
--------------------
Unique values for property_type:
['Entire rental unit' 'Entire loft' 'Private room in rental unit'
 'Entire home' 'Entire condo' 'Private room in bed and breakfast'
 'Private room in condo' 'Private room in casa particular'
 'Entire guest suite' 'Private room in home'
 'Private room in serviced apartment' 'Shared room in rental unit'
 'Shared room in bed and breakfast' 'Entire serviced apartment'
 'Room in hotel' 'Private room in hostel' 'Private room in guesthouse'
 'Shared room in home' 'Shared room in hostel' 'Tiny home'
 'Casa particular' 'Entire vacation home' 'Private room in guest suite'
 'Room in boutique hotel' 'Entire townhouse' 'Private room in loft'
 'Camper/RV' 'Room in aparthotel' 'Private room'
 'Private room in vacation home' 'Room in hostel' 'Entire chalet'
 'Entire place' 'Entire guesthouse' 'Private room in chalet'
 'Private room in townhouse' 'Private room in dome' 'Entire bungalow'
 'Entire villa' 'Private room in cave' 'Yurt' 'Room in casa particular'
 'Private room in villa' 'Floor' 'Religious building'
 'Shared room in aparthotel' 'Private room in floor' 'Hut'
 'Private room in tiny home' 'Shared room in hotel' 'Shared room in condo'
 'Private room in earthen home' 'Private room in minsu' 'Entire cottage'
 'Entire cabin' 'Castle' 'Farm stay' 'Room in bed and breakfast'
 'Earthen home' 'Private room in farm stay' 'Treehouse'
 'Private room in cottage' 'Boat' 'Room in nature lodge'
 'Private room in cabin' 'Room in serviced apartment' 'Island'
 'Holiday park' 'Entire home/apt' 'Room in heritage hotel' 'Tent'
 'Campsite' 'Private room in island' 'Shipping container' 'Barn'
 'Shared room in boutique hotel' 'Private room in tent'
```

```
 'Private room in castle' 'Private room in yurt']
--------------------
Unique values for bathrooms_text:
['1 bath' '2 baths' '1 shared bath' '3 baths' '1 private bath' '1.5 baths'
 '4 baths' '2.5 baths' '7 baths' '6 baths' 'Half-bath' '1.5 shared baths'
 '2 shared baths' 'Private half-bath' '4 shared baths' '5 baths' '8 baths'
 '8 shared baths' '6 shared baths' '3.5 baths' '0 baths' '5 shared baths'
 '4.5 shared baths' '0 shared baths' '2.5 shared baths' '5.5 shared baths'
 '4.5 baths' '6.5 baths' '3 shared baths' 'Shared half-bath' '5.5 baths'
 '7 shared baths' '10 baths' '7.5 baths' '9.5 baths' '12.5 baths'
 '8.5 baths' '9 baths' '11 baths' '19 baths' '3.5 shared baths' '12 baths'
 '11.5 baths' '10.5 baths' '17.5 baths' '15 baths']
--------------------
Unique values for amenities:
['["Air conditioning", "Wifi", "TV", "Washer", "Kitchen"]'
 '["Dining table", "Hair dryer", "Sun loungers", "Extra pillows and blankets",
"Toaster", "Window guards", "Wine glasses", "Exterior security cameras on
property", "Hot water kettle", "Shower gel", "Cleaning products", "Kitchen",
"Dishwasher", "Wifi", "Coffee maker: drip coffee maker", "Hangers", "Cooking
basics", "TEKA stainless steel single oven", "Private backyard", "Host greets
you", "Microwave", "Safe", "Room-darkening shades", "Clothing storage: closet",
"Smoking allowed", "Dishes and silverware", "Refrigerator", "Free dryer \\u2013
In unit", "Free washer \\u2013 In unit", "Heating - split type ductless system",
"Laundromat nearby", "Essentials", "Hot water", "Bed linens", "Leroy Merlyn
Bluetooth sound system", "Paid parking lot off premises", "43 inch HDTV with
premium cable, Netflix", "Bread maker", "AC - split type ductless system", "Fire
extinguisher", "Luggage dropoff allowed", "Courtyard view", "Outdoor furniture",
"Children\\u2019s dinnerware", "Freezer"]'
 '["Heating", "Hair dryer", "Toaster", "TV", "Hot water kettle", "Patio or
balcony", "Kitchen", "Dishwasher", "Wifi", "Microwave", "Smoking allowed",
"Stove", "Air conditioning", "Dryer", "Refrigerator", "Dishes and silverware",
"Washer", "Essentials", "Hot water", "Bed linens", "Coffee maker", "Oven",
"Elevator", "Iron", "Freezer"]'
 …
 '["Room-darkening shades", "Single level home", "Lockbox", "Clothing storage",
"Extra pillows and blankets", "Mosquito net", "Kitchen", "Mini fridge", "Hot
water", "Cooking basics", "Laundromat nearby", "Essentials", "Washer", "Coffee
maker: pour-over coffee", "Cleaning products", "Freezer", "Dedicated workspace",
"HDTV", "Iron", "Shower gel", "Microwave", "Dishes and silverware", "Body soap",
"Portable fans", "Waterfront", "Self check-in", "Drying rack for clothing",
"Private entrance", "Wifi", "Bed linens", "Shampoo", "Hangers", "Toaster"]'
 '["Hair dryer", "Wine glasses", "Room-darkening shades", "TV", "Free street
parking", "Clothing storage", "Single level home", "Lockbox", "Kitchen", "Mini
fridge", "Hot water", "Essentials", "Washer", "Cleaning products", "Freezer",
"Long term stays allowed", "Oven", "First aid kit", "Iron", "Refrigerator",
"Shower gel", "Microwave", "Dishes and silverware", "Hot water kettle", "Body
soap", "Elevator", "Self check-in", "Private entrance", "Dining table",
"Dishwasher", "Wifi", "Fire extinguisher", "Bed linens", "Shampoo", "Heating",
```

```
"Smoke alarm", "Coffee maker", "Air conditioning", "Hangers", "Toaster",
"Stove"]'
 '["Hair dryer", "Washer", "Wifi", "TV", "Lock on bedroom door", "Shampoo",
"Kitchen", "Microwave", "Dishes and silverware", "Portable fans", "First aid
kit", "Hot water", "Dining table"]']
--------------------
Unique values for last_review:
['2024-09-08' '2024-08-25' '2024-09-04' … '2022-06-08' '2021-12-01'
 '2019-10-14']
--------------------
```

[22]:
```
# Display observations in listings2 with NA for 2024_revenue
listings2[listings2['2024_revenue'].isna()]
```

[22]:

	id	description	\
34	1086253821003841243	GREAT GARDEN AND POOL FLOOR! It has radia…	
62	1173774357588030568	Your family will have everything just a stone'…	
105	5897565	Gran vía is the oldest y avenue in Madrid.<br …	
116	729645395352577944	Enjoy the simplicity of this quiet and central…	
118	1071609701258762364	Tour the most popular shops and restaurants fr…	
…	…	…	
46240	42238940	Spacious duplex in main floor kitchen, living …	
46254	50837184	New and decorated studios, fully equipped and …	
46349	41918863	Apartment located in a very family-friendly ar…	
46391	41547027	Modern 1 bedroom apartment with swimming pool …	
46396	41361385	Modern 2-bedroom apartment with swimming pool …	

	host_response_time	host_response_rate	host_acceptance_rate	\
34	1.0	0.94	0.55	
62	0.0	1.00	0.83	
105	0.0	1.00	0.86	
116	0.0	1.00	0.56	
118	0.0	1.00	0.79	
…	…	…	…	
46240	0.0	0.98	1.00	
46254	0.0	0.98	1.00	
46349	0.0	0.98	1.00	
46391	0.0	0.98	1.00	
46396	0.0	0.98	1.00	

	host_is_superhost	host_verifications	\
34	1.0	['email', 'phone']	
62	1.0	['email', 'phone']	
105	1.0	['email', 'phone']	
116	1.0	['email', 'phone', 'work_email']	
118	1.0	['email', 'phone']	
…	…	…	

```
46240               1.0  ['email', 'phone', 'work_email']
46254               1.0  ['email', 'phone', 'work_email']
46349               1.0  ['email', 'phone', 'work_email']
46391               1.0  ['email', 'phone', 'work_email']
46396               1.0  ['email', 'phone', 'work_email']

      neighbourhood_cleansed                     property_type  accommodates  \
34                San Pascual                 Entire rental unit             2
62                 Guindalera                 Entire rental unit             4
105                    Cortes        Private room in rental unit             6
116                 Trafalgar        Private room in rental unit             2
118                   Acacias  Shared room in bed and breakfast             1
…                         …                                 …           …
46240                 Centro                 Entire rental unit             2
46254                 Centro                 Room in aparthotel             4
46349    Cruz De Humilladero                 Entire rental unit             6
46391    Cruz De Humilladero                 Entire rental unit             4
46396    Cruz De Humilladero                 Entire rental unit             5

       …  reviews_per_month room_type_Hotel room  room_type_Private room  \
34     …                0.18                    0                       0
62     …                1.34                    0                       0
105    …                0.15                    0                       1
116    …                0.51                    0                       1
118    …                1.29                    0                       0
…    …                   …                    …                       …
46240  …                0.27                    0                       0
46254  …                2.68                    0                       1
46349  …                0.57                    0                       0
46391  …                0.52                    0                       0
46396  …                0.28                    0                       0

       room_type_Shared room city_Madrid  city_Malaga  city_Mallorca  \
34                         0           1            0              0
62                         0           1            0              0
105                        0           1            0              0
116                        0           1            0              0
118                        1           1            0              0
…                        …         …          …            …
46240                      0           0            1              0
46254                      0           0            1              0
46349                      0           0            1              0
46391                      0           0            1              0
46396                      0           0            1              0

       city_Menorca  listing_id  2024_revenue
34                0         NaN           NaN
```

```
62                0          NaN           NaN
105               0          NaN           NaN
116               0          NaN           NaN
118               0          NaN           NaN
…               …          …             …
46240             0          NaN           NaN
46254             0          NaN           NaN
46349             0          NaN           NaN
46391             0          NaN           NaN
46396             0          NaN           NaN

[615 rows x 44 columns]
```

[23]:
```
# Count the number of observations with 2024_revenue = NA
na_revenue_count = revenue_by_property['2024_revenue'].isna().sum()
print(f"Number of observations with 2024_revenue = NA: {na_revenue_count}")
```

```
Number of observations with 2024_revenue = NA: 0
```

[24]:
```
# Count unique 'id' in listings2
unique_listings2_id = listings2['id'].nunique()
print(f"Number of unique 'id' in listings2: {unique_listings2_id}")

# Count unique 'listing_id' in revenue_by_property
unique_revenue_listing_id = revenue_by_property['listing_id'].nunique()
print(f"Number of unique 'listing_id' in revenue_by_property:␣
 ↪{unique_revenue_listing_id}")
```

```
Number of unique 'id' in listings2: 46405
Number of unique 'listing_id' in revenue_by_property: 45790
```

[25]:
```
na_count = listings2['2024_revenue'].isna().sum()
print(f"Number of NA values in '2024_revenue': {na_count}")
```

```
Number of NA values in '2024_revenue': 615
```

We've figured out that there are more observations within our listings2 than in our calendar data, leaving us with 615 NA values for our target variable, 2024_revenue. We've decided to get rid of all observations with NA values, since they will simply not be useful to us in the modeling process. We could also change all of these values to 0, but this would skew lots of our statistics in a way that we don't want.

[26]:
```
# Drop rows with NA values in the '2024_revenue' column
listings2 = listings2.dropna(subset=['2024_revenue'])

# Reset the index after dropping rows
listings2 = listings2.reset_index(drop=True)
```

[27]:
```
import numpy as np

# Assuming 'listings2' DataFrame is already defined as in your provided code.

# Log transform the '2024_revenue' column to account for the large amount of␣
 ↪variance
listings2['log_2024_revenue'] = np.log1p(listings2['2024_revenue'])

# Display the first few rows to verify the transformation
print(listings2[['2024_revenue', 'log_2024_revenue']].head())
```

```
   2024_revenue  log_2024_revenue
0       14687.0          9.594786
1        9248.0          9.132271
2        5412.0          8.596559
3        8140.0          9.004668
4        1160.0          7.057037
```

5 Model Creation

Now, with all the data in order and all the necessary variables created, we can start with our model creation. Our decision for which models to use will be explained below.

[28]:
```
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error


# Define features (X) and target (y)
X = listings2.select_dtypes(include=np.number).drop(['id', '2024_revenue',␣
 ↪'listing_id', 'number_of_reviews', 'number_of_reviews_l30d',
                                                      ␣
 ↪'calculated_host_listings_count_entire_homes', 'reviews_per_month',␣
 ↪'log_2024_revenue', 'price', 'availability_365'], axis=1, errors='ignore')
y = listings2['log_2024_revenue']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,␣
 ↪random_state=42)

# Decision Tree Regression
dt_model = DecisionTreeRegressor(random_state=42)
dt_model.fit(X_train, y_train)
dt_predictions = dt_model.predict(X_test)

# Random Forest Regression
```

```
rf_model = RandomForestRegressor(random_state=42)
rf_model.fit(X_train, y_train)
rf_predictions = rf_model.predict(X_test)


def print_metrics(model_name, y_true, y_pred):
  mse = mean_squared_error(y_true, y_pred)
  rmse = np.sqrt(mse)
  r2 = r2_score(y_true, y_pred)
  mae = mean_absolute_error(y_true, y_pred)

  print(f"{model_name} Metrics:")
  print(f"Mean Squared Error: {mse:.2f}")
  print(f"Root Mean Squared Error: {rmse:.2f}")
  print(f"R-squared: {r2:.2f}")
  print(f"Mean Absolute Error: {mae:.2f}")
  print("-" * 20)


print_metrics("Decision Tree", y_test, dt_predictions)
print_metrics("Random Forest", y_test, rf_predictions)
```

```
Decision Tree Metrics:
Mean Squared Error: 1.63
Root Mean Squared Error: 1.28
R-squared: 0.01
Mean Absolute Error: 0.91
--------------------
Random Forest Metrics:
Mean Squared Error: 0.78
Root Mean Squared Error: 0.88
R-squared: 0.52
Mean Absolute Error: 0.63
--------------------
```

5.1 Model Comparison and Analysis

Random Forest vs Decision Tree

Based on the provided evaluation metrics, the Random Forest Regression model outperforms the Decision Tree Regression model in predicting Airbnb revenue.The Random Forest demonstrates significantly higher R-squared, lower Mean Squared Error, and lower Mean Absolute Error, indicating better predictive accuracy and a stronger ability to capture the underlying patterns in the data.

However, the Decision Tree model offers simplicity and interpretability. Thus, we prioritize the Decision Tree model for feature analysis due to its enhanced interpretability. Decision Trees provide a more transparent view of how individual features contribute to the final prediction, enabling a clearer understanding of their relative importance in determining Airbnb revenue. This insight is

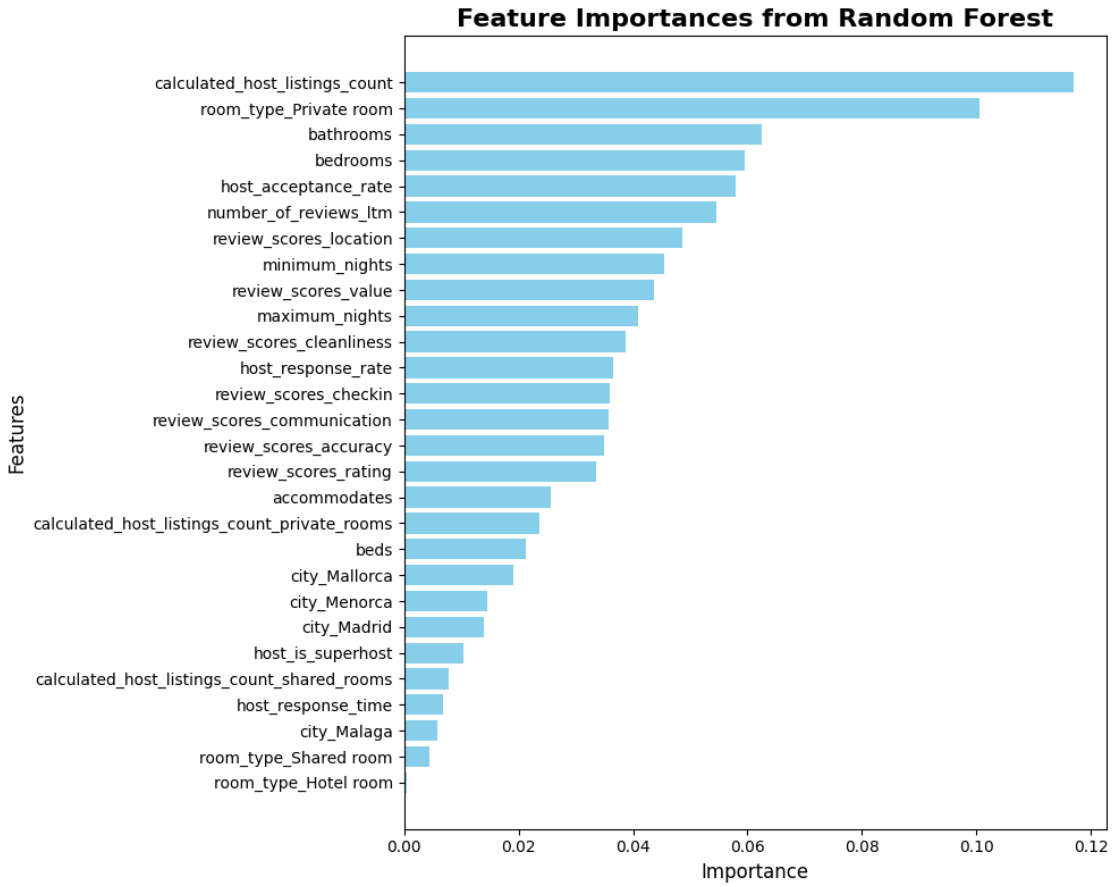
invaluable for identifying key factors that influence revenue and informing targeted strategies for optimization.

We chose to exclude `price` and `availability_365` as features in the model because their inclusion significantly impacted our ability to analyze the relationship between other features and revenue in 2024. Both `price` and `availability_365` are directly involved in the calculation of revenue (`revenue = price * booked days`), which makes them highly correlated with the target variable. Their dominant influence on the model confounded the effects of other important predictors, such as property characteristics, host behaviors, and review metrics. By removing these features, we ensured a more balanced evaluation of the remaining variables. This allowed the model to better assess the characteristics affecting revenue without these two factors skewing the results. This approach provided clearer insights into how other features, like the number of bedrooms, bathrooms, and location, contribute to the financial performance of Airbnb properties.

[29]:
```
import pandas as pd

# Extract feature importances
importances = rf_model.feature_importances_  # Assuming rf_model is your Random␣
 ↪Forest model
feature_names = X_test.columns  # Assuming X_test is your feature set

# Create a DataFrame for feature importances
feature_importances = pd.DataFrame({
    'Feature': feature_names,
    'Importance': importances
})

# Sort the features by importance
feature_importances = feature_importances.sort_values(by='Importance',␣
 ↪ascending=False)

# Display the most important features
print("Most Important Features:")
print(feature_importances)

# Plotting the feature importances
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 8))
plt.barh(feature_importances['Feature'], feature_importances['Importance'],␣
 ↪color='skyblue')
plt.xlabel('Importance', fontsize=12)
plt.ylabel('Features', fontsize=12)
plt.title('Feature Importances from Random Forest', fontsize=16, weight='bold')
plt.gca().invert_yaxis()  # Invert y-axis to have the most important feature on␣
 ↪top
plt.tight_layout()
plt.show()
```

Most Important Features:

	Feature	Importance
18	calculated_host_listings_count	0.116888
22	room_type_Private room	0.100459
5	bathrooms	0.062410
6	bedrooms	0.059465
2	host_acceptance_rate	0.057918
10	number_of_reviews_ltm	0.054585
16	review_scores_location	0.048687
8	minimum_nights	0.045484
17	review_scores_value	0.043660
9	maximum_nights	0.040854
13	review_scores_cleanliness	0.038743
1	host_response_rate	0.036629
14	review_scores_checkin	0.035947
15	review_scores_communication	0.035771
12	review_scores_accuracy	0.035016
11	review_scores_rating	0.033554
4	accommodates	0.025691
19	calculated_host_listings_count_private_rooms	0.023700
7	beds	0.021360
26	city_Mallorca	0.019189
27	city_Menorca	0.014469
24	city_Madrid	0.013930
3	host_is_superhost	0.010339
20	calculated_host_listings_count_shared_rooms	0.007748
0	host_response_time	0.006834
25	city_Malaga	0.005815
23	room_type_Shared room	0.004362
21	room_type_Hotel room	0.000491

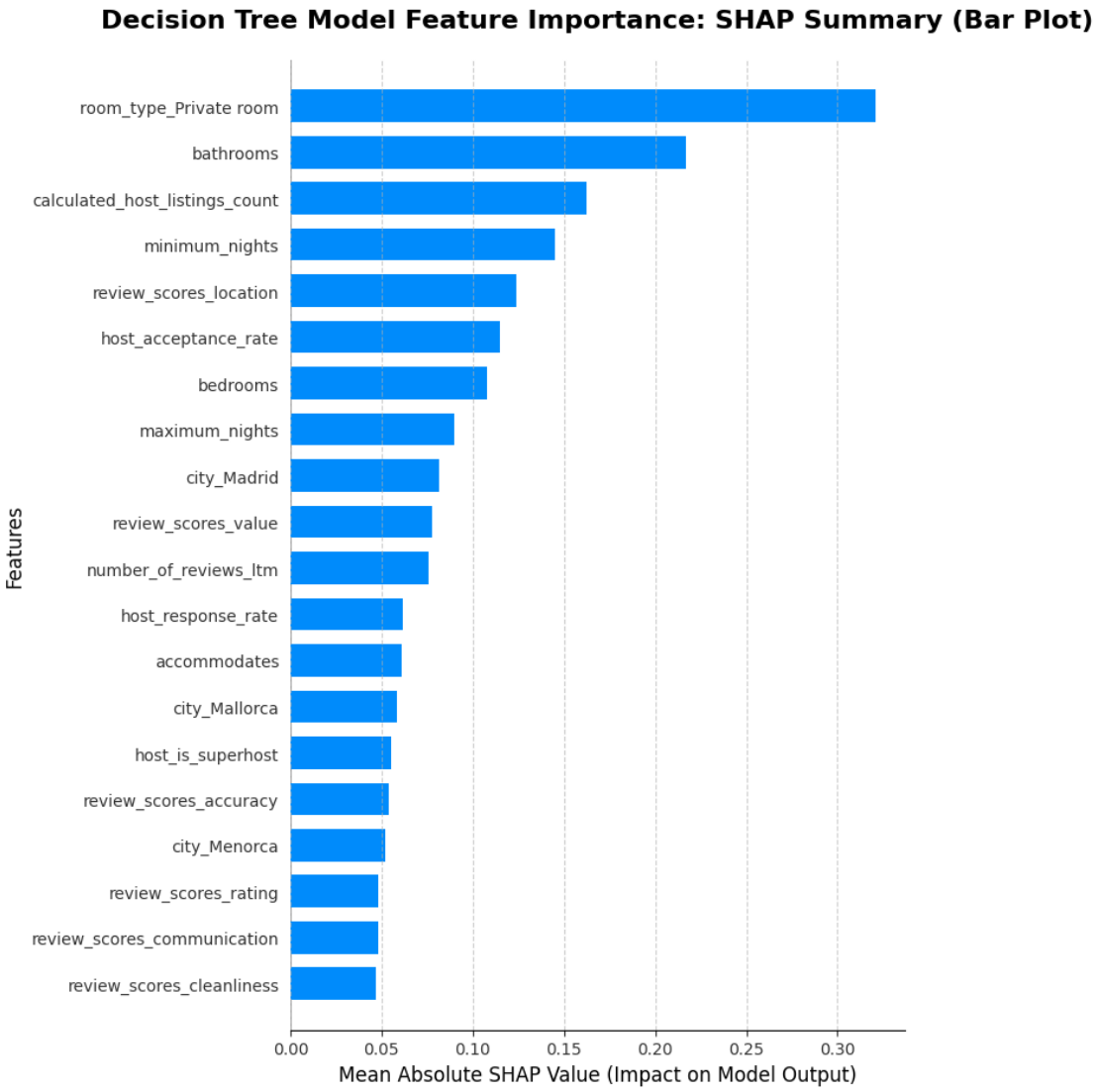


The analysis of feature importance reveals several key drivers of revenue for Airbnb properties. The top predictor, `calculated_host_listings_count`, highlights the advantage of managing multiple listings, suggesting that hosts with larger portfolios can generate higher revenues, likely due to economies of scale and increased booking opportunities. The second most important feature, `room_type_Private room`, emphasizes the importance of offering private accommodations, which are highly desirable among guests. Amenities such as `bathrooms` and `bedrooms` are also critical, underscoring the role of property size and comfort in driving bookings. Lastly, `host_acceptance_rate` indicates that proactive and responsive hosts tend to perform better, suggesting that guest satisfaction and engagement are vital for revenue generation. These insights collectively suggest that investors should focus on scalable portfolios, prioritize private accommodations with enhanced amenities, and encourage host responsiveness to maximize their returns.

Based on the random forest regression feature importance, we can pick out the top 5 most heavily weighted factors in a property's revenue. These are: "calculated_host_listings_count", "room_type_Private room", "bathrooms", "bedrooms", "host_acceptance_rate"

[30]:
```
import shap

# Explain the model's predictions using SHAP values
explainer = shap.TreeExplainer(dt_model)
```

```
shap_values = explainer.shap_values(X_test)

# Visualize the first prediction's explanation (using a force plot)
shap.force_plot(explainer.expected_value, shap_values[0,:], X_test.iloc[0,:])

# Summarize the effects of all the features with enhancements
plt.figure(figsize=(10, 8))
shap.summary_plot(shap_values, X_test, plot_type="bar", show=False)  # Suppress␣
 ↪automatic show to customize

# Adding title and labels
plt.title("Decision Tree Model Feature Importance: SHAP Summary (Bar Plot)",␣
 ↪fontsize=16, weight='bold', pad=20)
plt.xlabel("Mean Absolute SHAP Value (Impact on Model Output)", fontsize=12)
plt.ylabel("Features", fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(axis="x", linestyle="--", alpha=0.6)
plt.tight_layout()
plt.show()
```



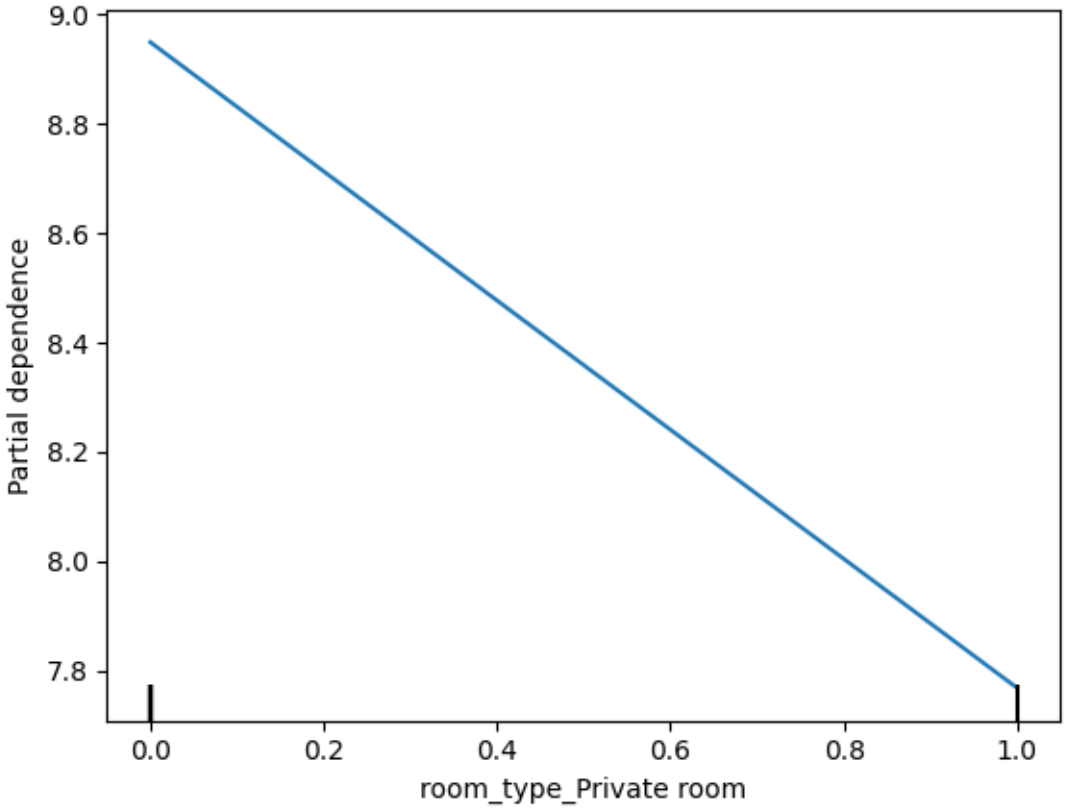
Analysis of the SHAP plot reveals that property characteristics, particularly room type and bathroom count, are the most influential factors in determining Airbnb revenue. Positive guest reviews, especially for location and value, also play a significant role. Host factors like acceptance rate have some impact, while location-specific effects are evident, with properties in Madrid showing higher revenue potential. These findings suggest that optimizing property listings, setting competitive pricing, and maintaining high guest satisfaction are crucial strategies for maximizing revenue.

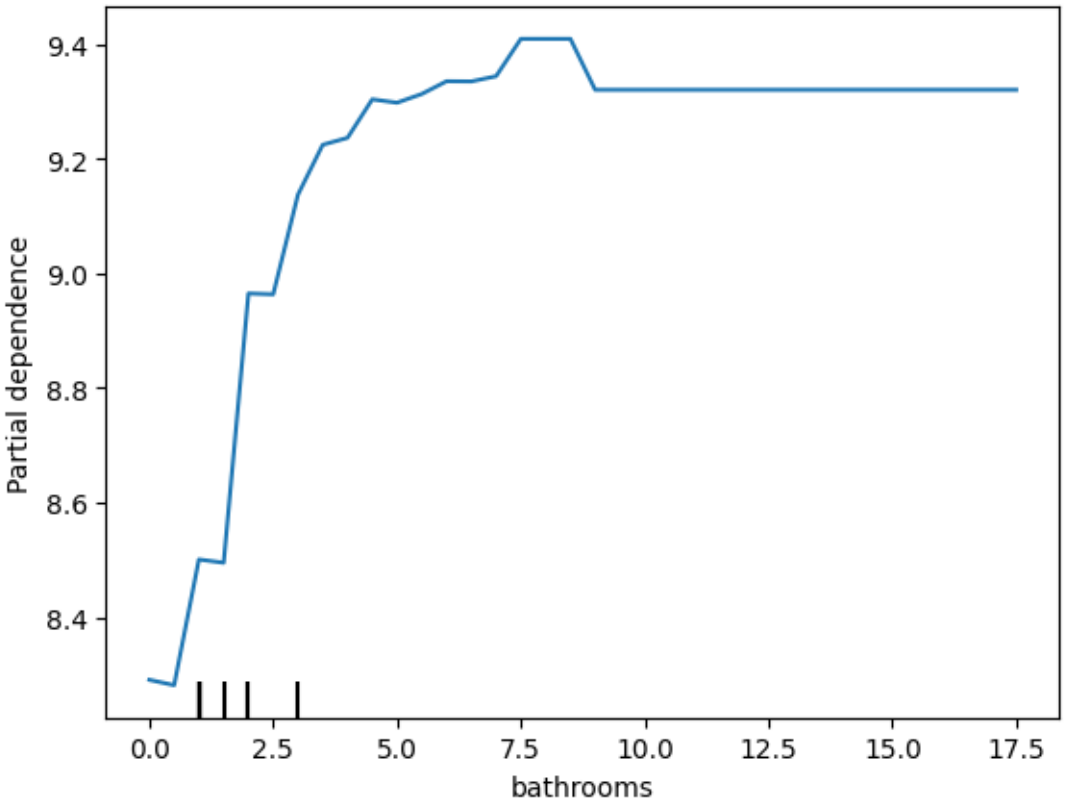
Based on the SHAP summary plot, we can pick out additional heavily weighted factors in a property's revenue. These are: "minimum_nights" and "review_scores_location"

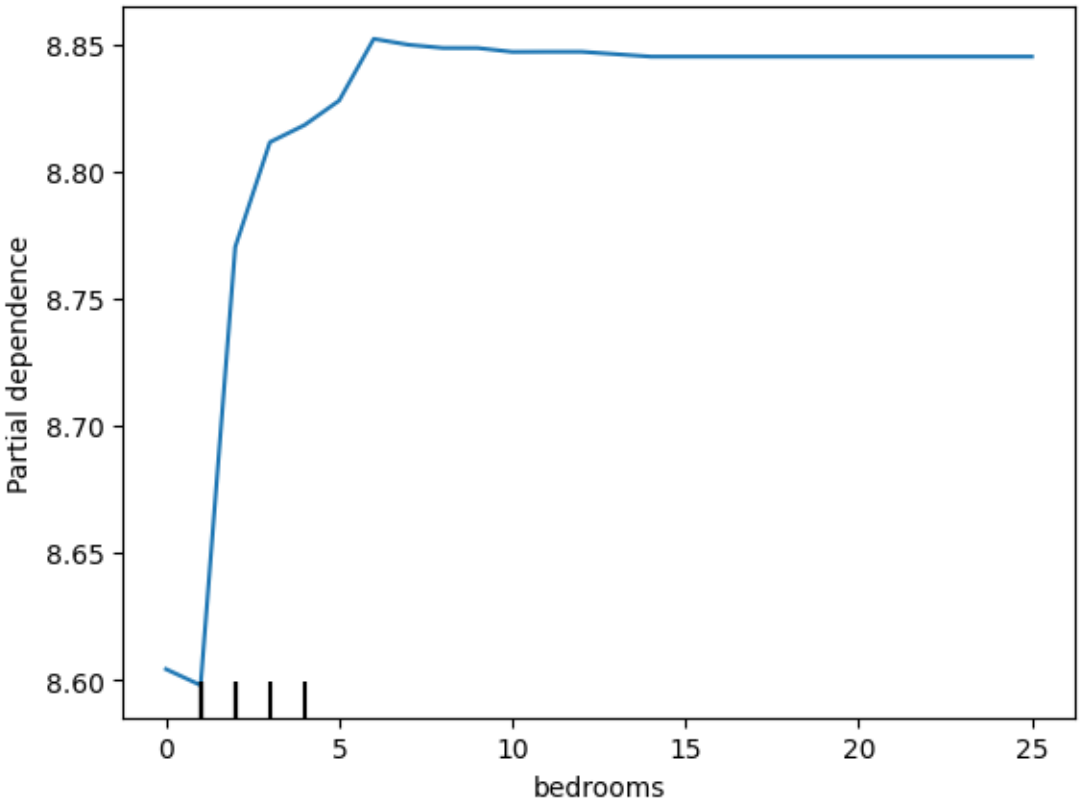
Partial Dependence Plots (PDPs) show how the model's prediction changes as the value of a specific feature is varied while holding other features constant. In this Airbnb revenue analysis, we use PDPs to gain insights into how key property characteristics, such as number of bathrooms, minimum stay requirements, and guest review scores, influence the predicted revenue. This helps us identify which features have the strongest impact and understand the nature of their relationship with the target

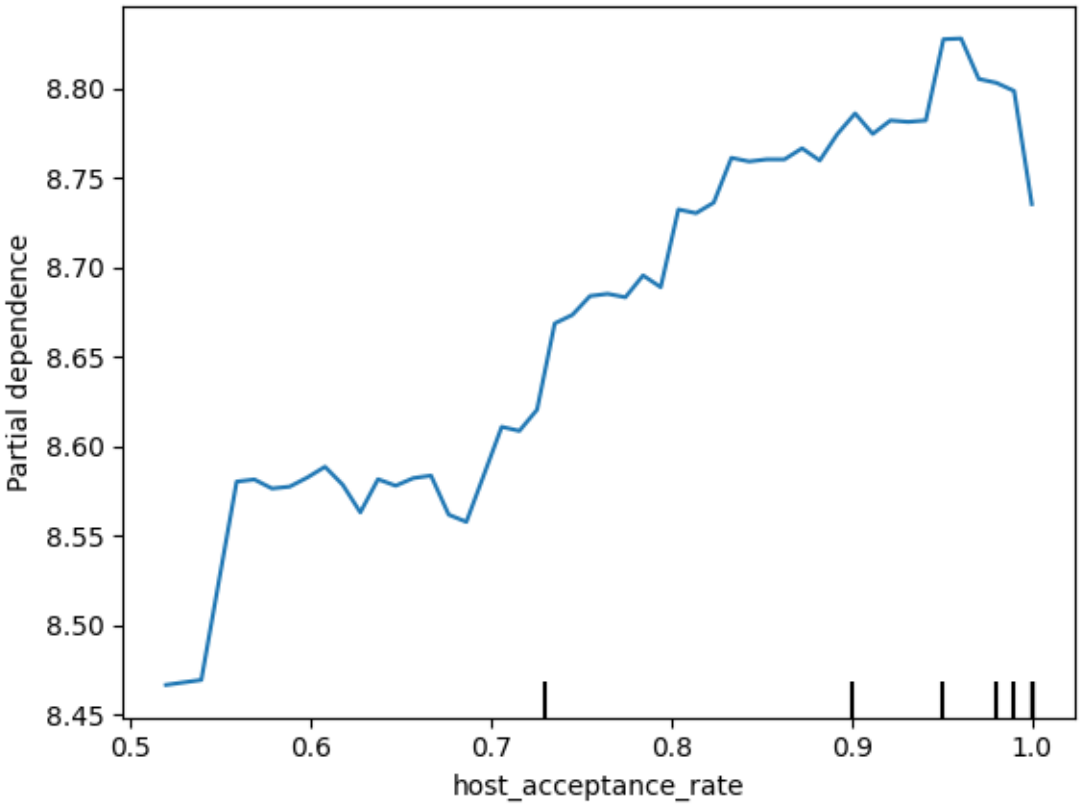
variable.

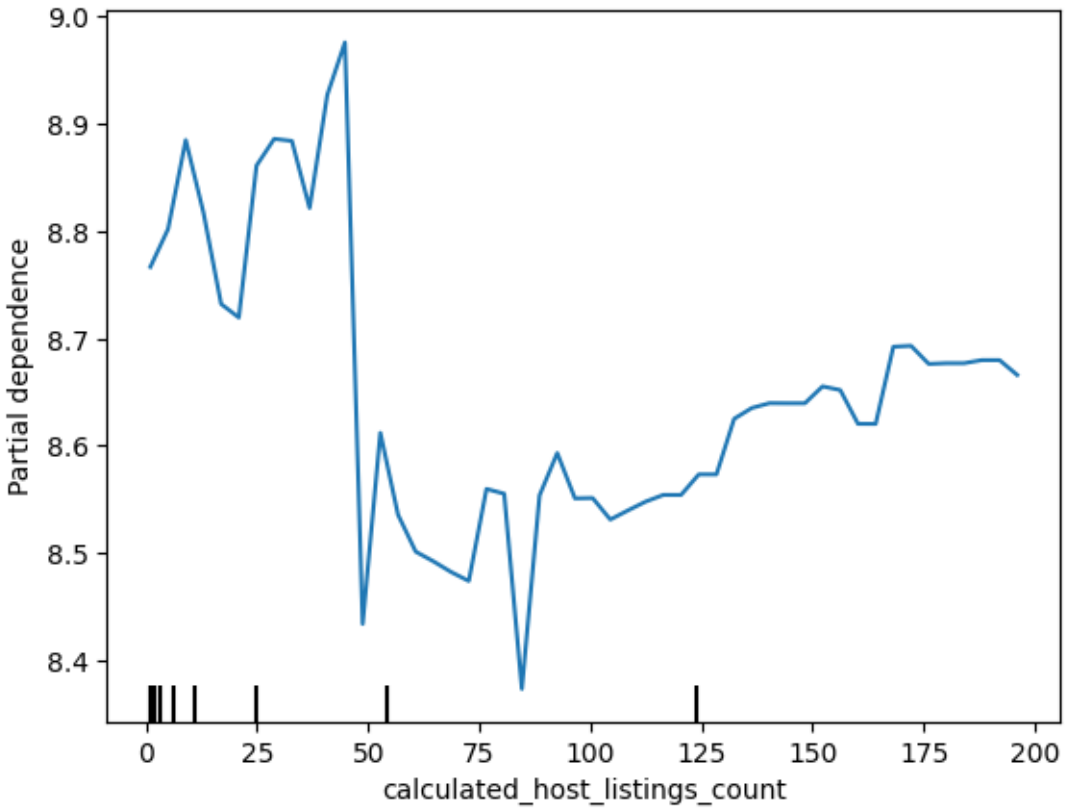
[31]:
```
from sklearn.inspection import PartialDependenceDisplay
top_5_features = ["room_type_Private room", "bathrooms", "bedrooms",␣
 ↪"host_acceptance_rate", "calculated_host_listings_count", "minimum_nights",␣
 ↪"review_scores_location"]
for feature in top_5_features:
    PartialDependenceDisplay.from_estimator(dt_model, X_train, [feature],␣
 ↪grid_resolution=50, feature_names=X_train.columns)
```

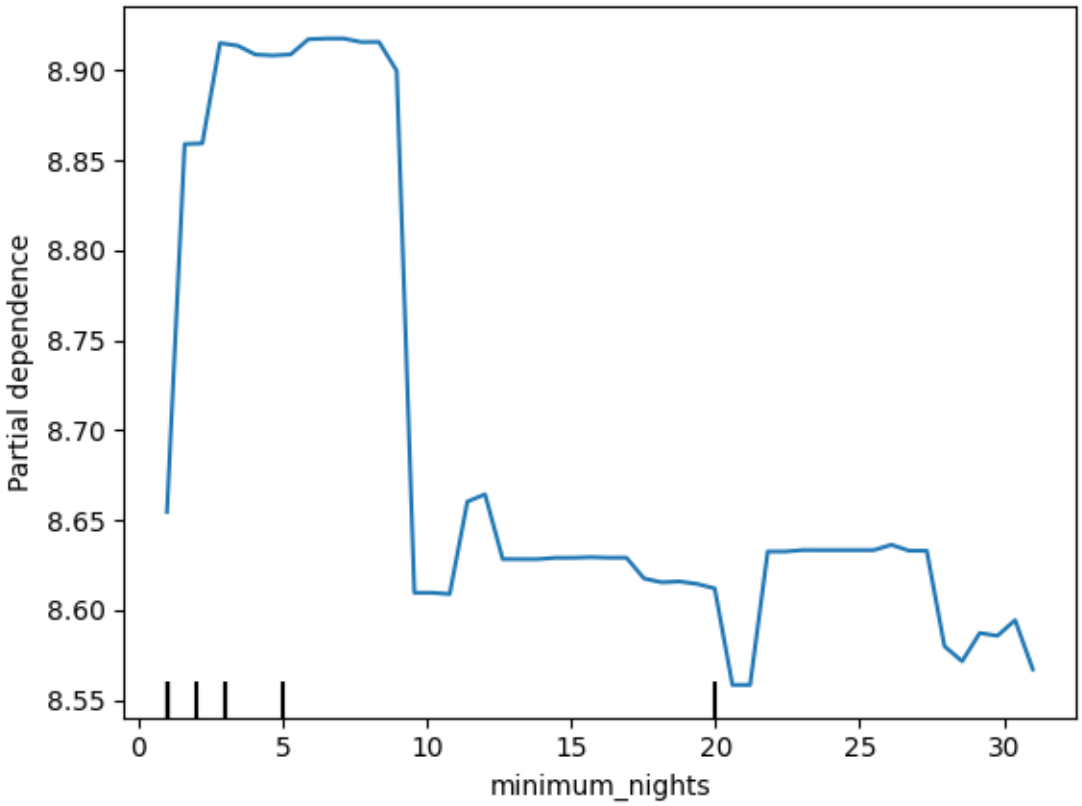


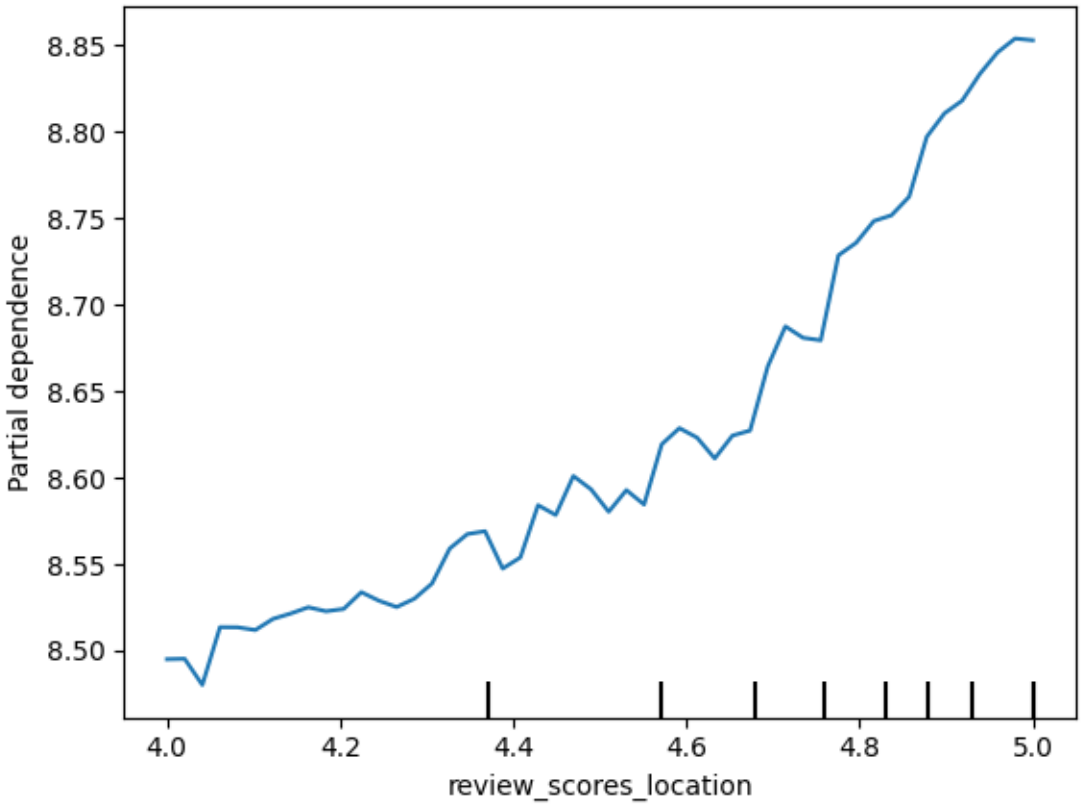

9.4
9.2
9.0
8.8
8.6
8.4
Partial dependence
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
bathrooms


8.85
8.80
8.75
8.70
8.65
8.60
Partial dependence
0
5
10
15
20
25
bedrooms


Partial dependence
8.80
8.75
8.70
8.65
8.60
8.55
8.50
8.45
0.5
0.6
0.7
0.8
0.9
1.0
host_acceptance_rate


9.0
8.9
8.8
8.7
8.6
8.5
8.4
Partial dependence
0
25
50
75
100
125
150
175
200
calculated_host_listings_count


Partial dependence
8.90
8.85
8.80
8.75
8.70
8.65
8.60
8.55
0
5
10
15
20
25
30
minimum_nights



#Insights

calculated_host_listings_count

The PDP for "calculated_host_listings_count" reveals a complex non-linear relationship with the predicted outcome. While a moderate number of listings might be associated with higher revenue, likely due to factors like Superhost status and efficient management, an excessive number of listings can lead to fluctuations and even a decline in predicted revenue. This suggests that managing a large portfolio requires careful resource allocation and may face challenges related to maintaining high-quality service and managing competition in saturated markets.

room_type_Private room

The provided PDP for "room_type_Private room" suggests a negative association between this feature and the predicted outcome. On Airbnb, a "private room" refers to a room in a shared house or apartment where guests have their own space, but share common areas with the host or other guests. The PDP indicates that, based on the Decision Tree model, properties listed as "private rooms" might be associated with lower revenue or less favorable outcomes compared to other room types (eg, entire homes).

bathrooms

The PDP for "bathrooms" reveals a strong positive correlation with the predicted outcome, suggesting that properties with more bathrooms tend to have higher predicted revenue. This relationship

exhibits diminishing returns, with the initial increase in revenue being more pronounced for the first few bathrooms. This finding aligns with the expectation that guests value convenience and privacy, and that properties with multiple bathrooms are likely to be more competitive and command higher prices in the market.

bedrooms

The partial dependence plot for `bedrooms` reveals that revenue increases sharply as the number of bedrooms rises from 0 to 5. This suggests that properties with more bedrooms are significantly more appealing to guests and command higher revenue. However, beyond 5 bedrooms, the curve flattens, indicating diminishing returns for additional bedrooms. Investors should prioritize properties with 2-5 bedrooms, as this range maximizes revenue potential without unnecessary overinvestment in excessively large properties.

host_acceptance_rate

The partial dependence plot for `host_acceptance_rate` shows a clear positive relationship with revenue. Revenue steadily increases as the acceptance rate rises, with notable improvements observed beyond a 70% acceptance rate. This indicates that hosts who are more responsive and accommodating to booking requests tend to generate higher revenue. The steep climb near the upper range (90-100%) suggests that maintaining a high acceptance rate is a critical factor in optimizing property performance. Investors should prioritize working with hosts who exhibit strong engagement and responsiveness, as these behaviors are strongly associated with higher profitability.

minimum_nights

The PDP for "minimum_nights" shows a complex non-linear relationship with the predicted outcome. While a moderate minimum stay requirement might be associated with higher revenue, very strict minimums (eg, exceeding 20 nights) appear to negatively impact the predicted outcome. This suggests that a balance is needed between ensuring longer stays and maintaining flexibility for guests.

review_scores_location

The PDP for "review_scores_location" reveals a strong positive correlation with the predicted outcome. This indicates that properties with higher location scores tend to have higher predicted revenue. This aligns with the expectation that guests value well-located properties and are willing to pay a premium for convenient access to amenities and attractions.

[32]:
```
import plotly.express as px

fig = px.box(listings2, x="room_type_Private room", y="bathrooms",
             labels={"room_type_Private room": "Private Room (0: No, 1: Yes)"})
fig.show()
```

The box plot reveals that private rooms generally have a lower average number of bathrooms compared to non-private rooms. The distribution of bathrooms for non-private rooms is also more variable, suggesting a wider range of property sizes and configurations within this category. These insights can inform pricing strategies, property management decisions, and an understanding of guest expectations for different room types.

#Conclusions Based on the analysis conducted in this project, here are some key recommendations

for Airbnb investors:

1. **Leverage Host Portfolio Size**: The number of properties a host manages (`calculated_host_listings_count`) is the strongest predictor of revenue. Hosts with larger portfolios tend to achieve higher revenues due to economies of scale. Investors should consider working with or becoming hosts who can effectively scale their property portfolios.
2. **Prioritize Properties with Multiple Bathrooms**: The number of bathrooms in a property remains a strong indicator of potential revenue. Investing in properties with more bathrooms, especially for non-private rooms, can significantly enhance revenue potential.
3. **Invest in Properties with 2-5 Bedrooms**: The number of bedrooms is another critical factor. Properties with 2-5 bedrooms offer the best balance between guest appeal and cost-effectiveness. Beyond five bedrooms, returns diminish, making mid-sized properties a more profitable choice.
4. **Encourage High Host Acceptance Rates**: Host responsiveness and willingness to accept bookings (`host_acceptance_rate`) strongly influence revenue. Investors should prioritize properties managed by highly engaged hosts with high acceptance rates or invest in host training to enhance engagement.
5. **Balance Private Rooms and Amenities**: While private rooms generally generate lower revenue compared to entire properties, adding features like multiple bathrooms or other amenities can increase their profitability. Investors should carefully consider this trade-off when making investment decisions.
6. **Optimize Minimum Stay Requirements**: Setting a moderate minimum stay requirement can positively impact revenue by attracting longer stays without deterring potential guests. Overly strict minimum stay policies may reduce booking rates.
7. **Focus on Location**: Properties located near popular amenities and attractions (`review_scores_location`) tend to generate higher revenue. Investors should prioritize acquiring or improving properties in prime locations to maximize returns.
8. **Support Host Engagement**: Hosts with moderate portfolios and strong engagement metrics, such as high response rates, perform better. Investors should provide tools, training, and support to improve host responsiveness and engagement to enhance property performance.

These insights provide a roadmap for Airbnb investors to optimize their strategies, focusing on scalable portfolios, well-located and well-amenitized properties, and host engagement for maximum returns.