Data Science

Project Protocol

By Almog Cohen, Shiry Strauss, Naor Elimelech, Matan Ben David, Anna Katz

Introduction:

AirBnB is an online marketplace for lodging and vacation rentals (BnB is a typical short for Bed and Breakfast). The company founded by 3 young roommates in San Francisco: Brian Chesky, Nathan Blecharczyk and Joe Gebbia.

In 2007, an annual convention venue took place in their neighborhood in San Francisco; the attendants occupied every hotel in San Francisco and as a result all accommodations were fully booked, while some visitors had no solution.

The founders, who were struggling for income at the time, decided to offer accommodation in their apartment for affordable rent fee. One year later, these 3 roommates founded AirBnB. The company has been evolving over the years; And today this online marketplace is a hub, matching travelers with hosts.

Objectives:

What are we trying to find out?
We are trying to predict the nightly price of Airbnb listings in Berlin based on data from 2018. This includes analyzing key features like location, property type, and guest ratings. By identifying patterns in this data, we can develop a model that predicts the price of new listings.

What do we already know?
We have access to a dataset from Kaggle that includes information such as location, property type, guest reviews, and host ratings. The dataset also contains booking history and availability details for each listing between 2011 and 2019. However, we haven't yet explored how these variables correlate with the price per night.

What are we aiming to achieve?
We aim to create an accurate predictive model for the nightly price of Airbnb listings in Berlin. The model will help identify which factors most influence pricing. Success will be measured by the model's accuracy and its ability to provide insights into pricing trends.

What factors affect our results?
Key factors influencing the price include location (e.g., proximity to tourist attractions), property type (e.g., apartment, house), and guest ratings. Host experience, amenities offered, and the number of reviews also play a significant role in pricing. Seasonality may also impact prices, with demand being higher during peak travel times.

Is there something new we can use?
We could explore advanced machine learning algorithms such as ensemble methods or neural networks to improve prediction accuracy. Additionally, new feature engineering techniques could extract more meaningful patterns from the data. External data, such as local event schedules or real estate trends, might also enhance the model's performance.

Data Preparation:

We used one dataset sourced from Kaggle, specifically the 'Berlin Airbnb Ratings' dataset. This dataset contains information on Airbnb listings in Berlin, including reviewer ratings and comments.

The dataset formed as a CSV document containing:

- Review details like review ID, review date, overall rating of the review etc.
- Listing details like listing ID, Neighborhood, coordinates, property type etc.

The original dataset consists of 456961 rows and 47 columns.

Our first steps were filtering the dataset to the latest year in it (2018), and deleting the following irrelevant columns:

Column Name	Reason of Deletion
Country, Country Code, Business Travel Ready	All values on each column are the same.
review_year	We already filtered out dataset to review_year = 2018
City	We only review Berlin
Listing URL, Host URL	In both columns the values are web address consist of listing/host ID – we already have columns for each
Postal Code, Neighborhood	In our dataset there are 3 location related columns, and we decided to focus on the neighborhood group point of view
index	Doesn't add any value to our research
Host Name, Reviewer ID, Review ID, Reviewer Name	These columns focus on the customer aspect, which is not contributing to predict listings' price
Square Feet	There are 5765 out of 157014 rows – 96% of the column's values are empty

In addition, the following columns have been altered:

- Is Superhost, Is Exact Location, Instant Bookable values been changed:
  - 't' = 1
  - 'f' = 0
- we created a new column called 'review_mnt', which is a result of extracting the month out of 'review_date'.

Eventually, the dataset consists of 157014 rows and 34 columns.

Data Types:

Checking the initial datatypes (df.info()) shows there are 16 'object' type columns that need to be changed. These are the changes of datatype that have been made to the 'object' columns:

- String Columns: 'Comments', 'Listing Name', 'Host Response Time', 'Neighborhood Group', 'Property Type', 'Room Type'
- Date Columns: 'review_date', 'Host Since', 'First Review', 'Last Review'
- Boolean Columns: 'Is Superhost', 'Is Exact Location', 'Instant Bookable'

- Float Columns: 'Price', 'Host Response Rate' (converted from % to float)
- Integer Columns: 'review_mnt'

You can see the mentioned changes below:

Before:

```
Data columns (total 34 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   review_date            157014 non-null  object
 1   Comments               156911 non-null  object
 2   Listing ID             157014 non-null  int64
 3   Listing Name           156983 non-null  object
 4   Host ID                157014 non-null  int64
 5   Host Since             157014 non-null  object
 6   Host Response Time     141106 non-null  object
 7   Host Response Rate     141106 non-null  object
 8   Is Superhost           157014 non-null  object
 9   Neighborhood Group     157014 non-null  object
 10  Latitude               157014 non-null  float64
 11  Longitude              157014 non-null  float64
 12  Is Exact Location      157014 non-null  object
 13  Property Type          157014 non-null  object
 14  Room Type              157014 non-null  object
 15  Accomodates            157014 non-null  int64
 16  Bathrooms              156879 non-null  float64
 17  Bedrooms               156833 non-null  float64
 18  Beds                   156989 non-null  float64
 19  Price                  157014 non-null  object
 20  Guests Included        157014 non-null  int64
 21  Min Nights             157014 non-null  int64
 22  Reviews                157014 non-null  int64
 23  First Review           157014 non-null  object
 24  Last Review            157014 non-null  object
 25  Overall Rating         156849 non-null  float64
 26  Accuracy Rating        156850 non-null  float64
 27  Cleanliness Rating     156850 non-null  float64
 28  Checkin Rating         156847 non-null  float64
 29  Communication Rating   156848 non-null  float64
 30  Location Rating        156847 non-null  float64
 31  Value Rating           156847 non-null  float64
 32  Instant Bookable       157014 non-null  object
 33  review_mnt             157014 non-null  object
dtypes: float64(12), int64(6), object(16)
```

After:

```
Data columns (total 34 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   review_date            157014 non-null  datetime64[ns]
 1   Comments               156911 non-null  string
 2   Listing ID             157014 non-null  int64
 3   Listing Name           156983 non-null  string
 4   Host ID                157014 non-null  int64
 5   Host Since             157014 non-null  datetime64[ns]
 6   Host Response Time     141106 non-null  string
 7   Host Response Rate     141106 non-null  float64
 8   Is Superhost           157014 non-null  bool
 9   Neighborhood Group     157014 non-null  string
 10  Latitude               157014 non-null  float64
 11  Longitude              157014 non-null  float64
 12  Is Exact Location      157014 non-null  bool
 13  Property Type          157014 non-null  string
 14  Room Type              157014 non-null  string
 15  Accomodates            157014 non-null  int64
 16  Bathrooms              156879 non-null  float64
 17  Bedrooms               156833 non-null  float64
 18  Beds                   156989 non-null  float64
 19  Price                  156963 non-null  float64
 20  Guests Included        157014 non-null  int64
 21  Min Nights             157014 non-null  int64
 22  Reviews                157014 non-null  int64
 23  First Review           157014 non-null  datetime64[ns]
 24  Last Review            157014 non-null  datetime64[ns]
 25  Overall Rating         156849 non-null  float64
 26  Accuracy Rating        156850 non-null  float64
 27  Cleanliness Rating     156850 non-null  float64
 28  Checkin Rating         156847 non-null  float64
 29  Communication Rating   156848 non-null  float64
 30  Location Rating        156847 non-null  float64
 31  Value Rating           156847 non-null  float64
 32  Instant Bookable       157014 non-null  bool
 33  review_mnt             157014 non-null  Int64
dtypes: Int64(1), bool(3), datetime64[ns](4), float64(14), int64(6), string(6)
```

The last change we made was creating a new column – 'Host tenure' which indicates how long the host is an AIRBNB partner (data type = float). We based our calculation on 'Host Since' column and dropped it after that.

This is the general description of our finalized data frame before heading to the EDA process:

```
                         review_date   Listing ID      Host ID
count                         157014  1.570140e+05  1.570140e+05
mean   2018-07-19 18:42:19.218159616  1.498746e+07  5.400128e+07
min              2018-01-01 00:00:00  2.695000e+03  2.217000e+03
25%              2018-05-06 00:00:00  7.603871e+06  6.169270e+06
50%              2018-07-29 00:00:00  1.663339e+07  2.763790e+07
75%              2018-10-07 00:00:00  2.195616e+07  9.018662e+07
max              2018-12-31 00:00:00  3.120638e+07  2.329172e+08
std                              NaN  8.571267e+06  5.973191e+07

                          Host Since  Host Response Rate       Latitude
count                         157014       141106.000000  157014.000000
mean   2015-01-04 14:31:07.339218176            0.959958      52.512652
min              2008-08-18 00:00:00            0.000000      52.376410
25%              2013-05-26 00:00:00            1.000000      52.493650
50%              2015-03-02 00:00:00            1.000000      52.513010
75%              2016-09-01 00:00:00            1.000000      52.532510
max              2018-12-28 00:00:00            1.000000      52.641500
std                              NaN            0.121714       0.029470

           Longitude    Accomodates      Bathrooms       Bedrooms  ...
count  157014.000000  157014.000000  156879.000000  156833.000000  ...
mean       13.402375       3.117945       1.100683       1.206328  ...
min        13.116320       1.000000       0.000000       0.000000  ...
25%        13.373790       2.000000       1.000000       1.000000  ...
50%        13.410650       2.000000       1.000000       1.000000  ...
75%        13.434410       4.000000       1.000000       1.000000  ...
max        13.721670      16.000000       8.500000      10.000000  ...
std         0.057319       1.954406       0.332590       0.732505  ...
```

```
           Longitude    Accomodates      Bathrooms       Bedrooms  ...  \
count  157014.000000  157014.000000  156879.000000  156833.000000  ...
mean       13.402375       3.117945       1.100683       1.206328  ...
min        13.116320       1.000000       0.000000       0.000000  ...
25%        13.373790       2.000000       1.000000       1.000000  ...
50%        13.410650       2.000000       1.000000       1.000000  ...
75%        13.434410       4.000000       1.000000       1.000000  ...
max        13.721670      16.000000       8.500000      10.000000  ...
std         0.057319       1.954406       0.332590       0.732505  ...

                        First Review                    Last Review  \
count                         157014                         157014
mean   2016-09-26 10:39:52.846497792  2019-03-21 21:28:45.507279104
min              2009-06-20 00:00:00            2018-01-01 00:00:00
25%              2015-11-05 00:00:00            2019-04-07 00:00:00
50%              2017-03-31 00:00:00            2019-04-28 00:00:00
75%              2018-01-17 00:00:00            2019-05-06 00:00:00
max              2018-12-31 00:00:00            2019-05-14 00:00:00
std                              NaN                            NaN

       Overall Rating  Accuracy Rating  Cleanliness Rating Checkin Rating
count   156849.000000    156850.000000       156850.000000  156847.000000
mean        94.836652         9.781377            9.511807       9.813793
min         20.000000         2.000000            2.000000       2.000000
25%         93.000000        10.000000            9.000000      10.000000
50%         96.000000        10.000000           10.000000      10.000000
75%         98.000000        10.000000           10.000000      10.000000
max        100.000000        10.000000           10.000000      10.000000
std          4.323682         0.459535            0.665757       0.424109

       Communication Rating  Location Rating  Value Rating  review_mnt
count         156848.000000    156847.000000 156847.000000    157014.0
mean               9.813539         9.647402      9.441124    7.115633
min                2.000000         2.000000      2.000000         1.0
25%               10.000000         9.000000      9.000000         5.0
50%               10.000000        10.000000      9.000000         7.0
75%               10.000000        10.000000     10.000000        10.0
max               10.000000        10.000000     10.000000        12.0
std                0.432570         0.524059      0.570820    3.152709
```

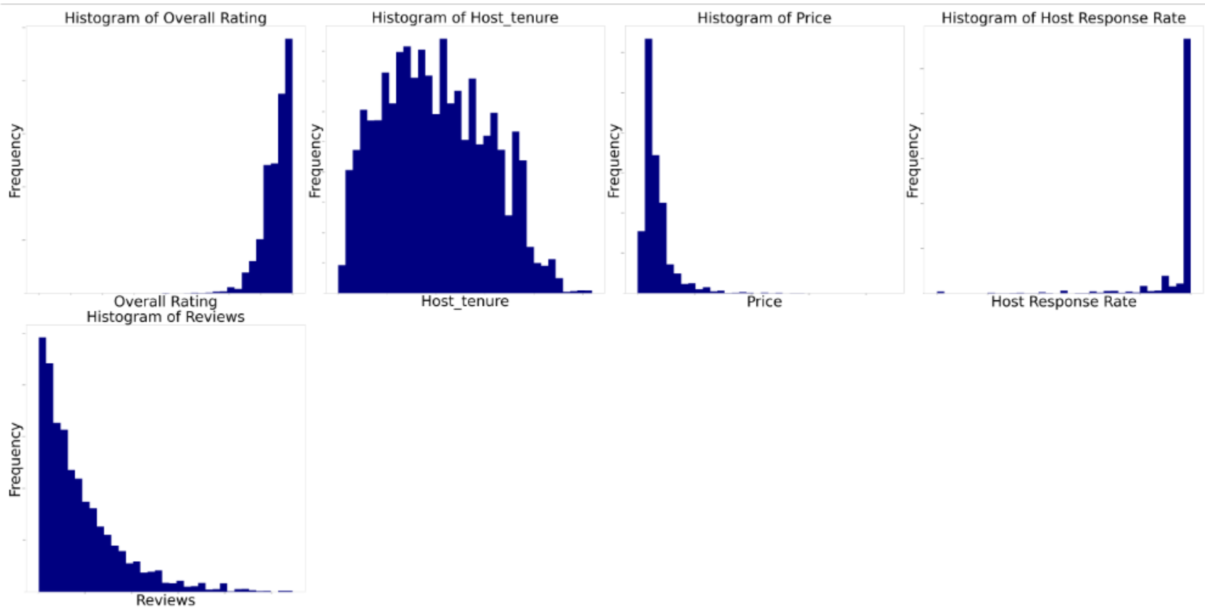
```
Data columns (total 34 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   review_date           157014 non-null  datetime64[ns]
 1   Comments              156911 non-null  string
 2   Listing ID            157014 non-null  int64
 3   Listing Name          156983 non-null  string
 4   Host ID               157014 non-null  int64
 5   Host Response Time    141106 non-null  string
 6   Host Response Rate    141106 non-null  float64
 7   Is Superhost          157014 non-null  bool
 8   Neighborhood Group    157014 non-null  string
 9   Latitude              157014 non-null  float64
 10  Longitude             157014 non-null  float64
 11  Is Exact Location     157014 non-null  bool
 12  Property Type         157014 non-null  string
 13  Room Type             157014 non-null  string
 14  Accomodates           157014 non-null  int64
 15  Bathrooms             156879 non-null  float64
 16  Bedrooms              156833 non-null  float64
 17  Beds                  156989 non-null  float64
 18  Price                 156963 non-null  float64
 19  Guests Included       157014 non-null  int64
 20  Min Nights            157014 non-null  int64
 21  Reviews               157014 non-null  int64
 22  First Review          157014 non-null  datetime64[ns]
 23  Last Review           157014 non-null  datetime64[ns]
 24  Overall Rating        156849 non-null  float64
 25  Accuracy Rating       156850 non-null  float64
 26  Cleanliness Rating    156850 non-null  float64
 27  Checkin Rating        156847 non-null  float64
 28  Communication Rating  156848 non-null  float64
 29  Location Rating       156847 non-null  float64
 30  Value Rating          156847 non-null  float64
 31  Instant Bookable      157014 non-null  bool
 32  review_mnt            157014 non-null  Int64
 33  Host_tenure           157014 non-null  float64
dtypes: Int64(1), bool(3), datetime64[ns](3), float64(15), int64(6), string(6)
```

EDA:

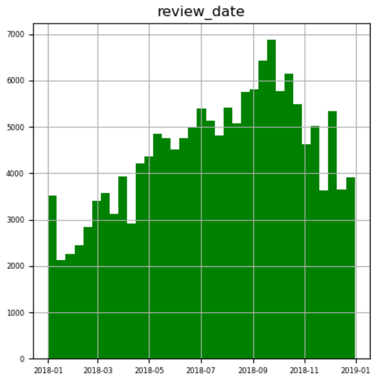
We started by examining the distribution of numeric, categorical, and dummy features using visualizations, in order to understand the general characteristics and spread of the data:

- **Numeric Features:**

'Overall Rating', 'Host_tenure', 'Price', 'Host Response Rate', 'Reviews'

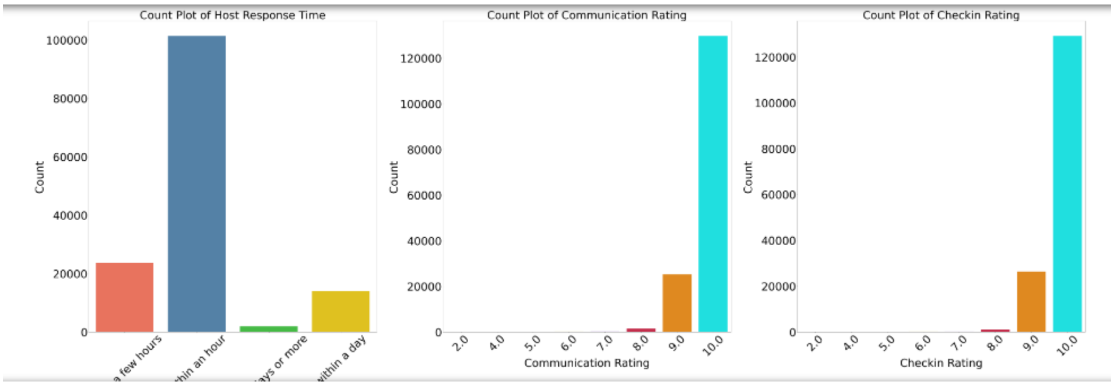


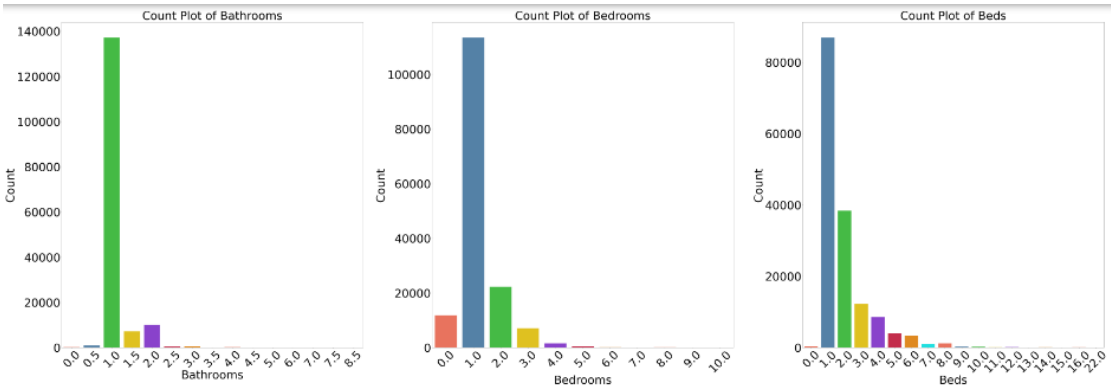
'Review Date'

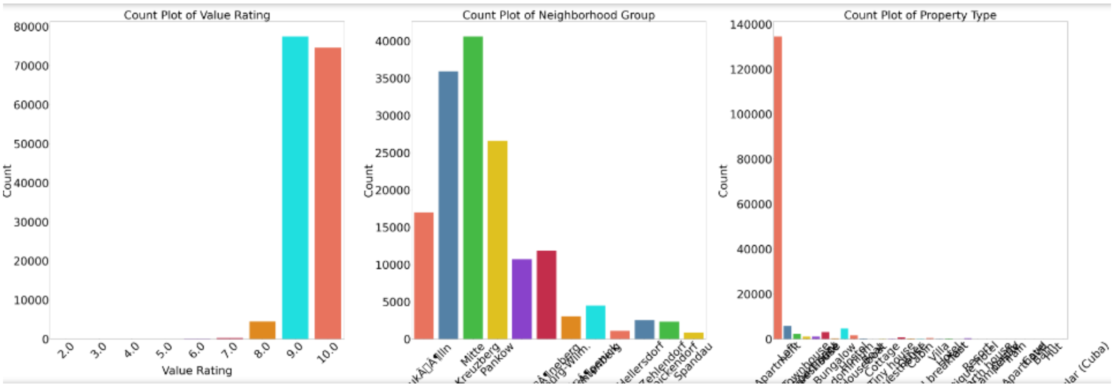


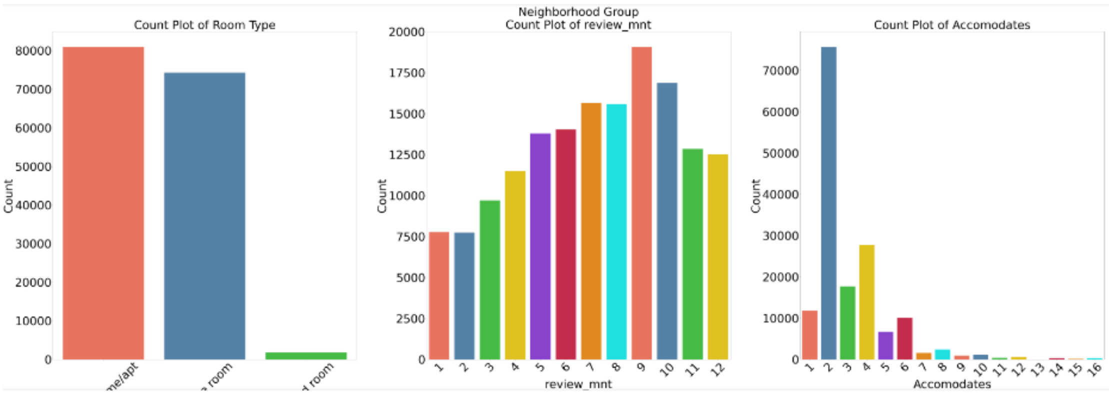
- **Categorial Features:**

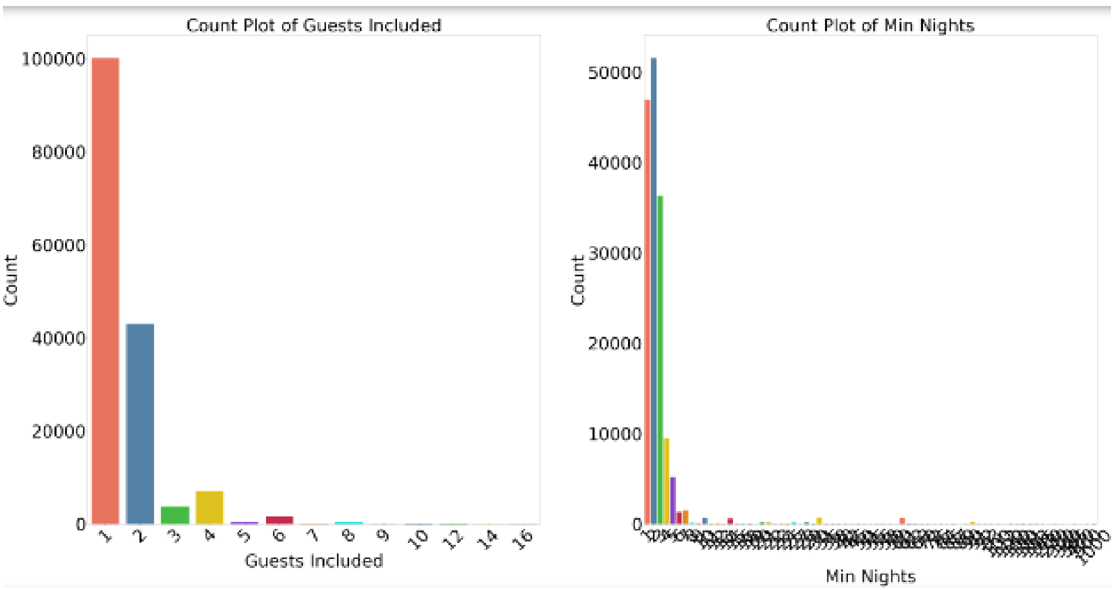
'Host Response Time', 'Communication Rating', 'Checkin Rating', 'Cleanliness Rating', 'Location Rating', 'Accuracy Rating', 'Bathrooms', 'Bedrooms', 'Beds', 'Value Rating', 'Neighborhood Group' ,'Property Type', 'Room Type', 'review_mnt', 'Accomodates', 'Guests Included', 'Min Nights'.




Count Plot of Bathrooms
Count
Bathrooms
Count Plot of Bedrooms
Count
Bedrooms
Count Plot of Beds
Count
Beds

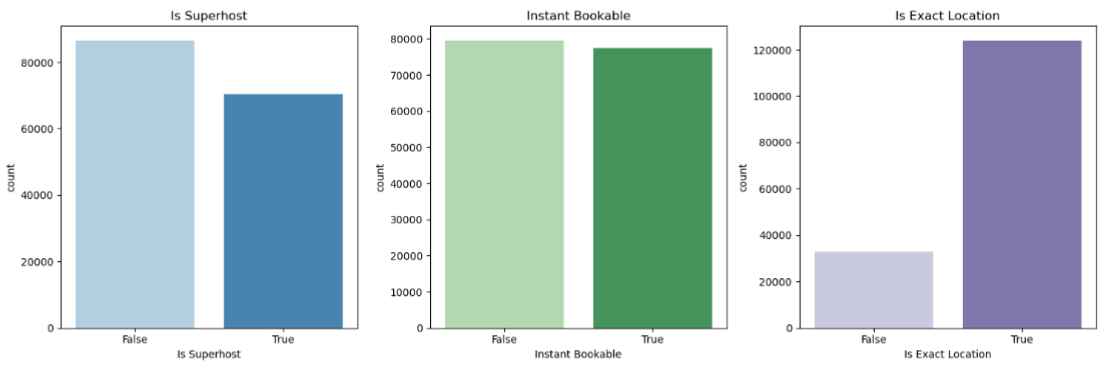

Count Plot of Value Rating
Count
Value Rating
Count Plot of Neighborhood Group
Count
Count Plot of Property Type
Count


Count Plot of Room Type
Count
Neighborhood Group
Count Plot of review_mnt
Count
review_mnt
Count Plot of Accomodates
Count
Accomodates


Count Plot of Guests Included
Count
Guests Included
Count Plot of Min Nights
Count
Min Nights

- **Dummy Features:**

'Is Superhost', 'Is Exact Location', 'Instant Bookable'



Our initial assumptions from the visualizations:

- Most of our numeric feature's distribution is not normal
- There are too many categories in some categorial features, and we might unite them later.
- There is concern that column 'Is Exact Location' is represented by unbalanced data. The issue will be examined as part of the feature engineering

<u>Skewness Analysis:</u>

We conducted a skewness analysis on our numeric features to assess the distribution of the data and determine whether any transformations might be necessary for modeling.

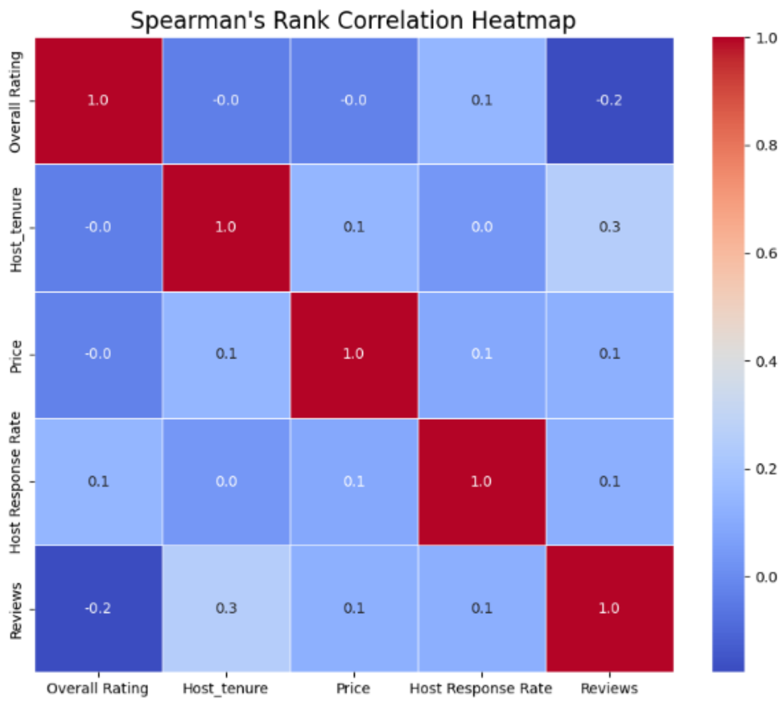
The skewness analysis of our numeric features revealed varying degrees of skewness across the dataset:

```
Overall Rating - Skewness: -1.97 => The data is highly skewed.
Host_tenure - Skewness: 0.23 => The data is approximately normal.
Price - Skewness: 3.17 => The data is highly skewed.
Host Response Rate - Skewness: -4.74 => The data is highly skewed.
Reviews - Skewness: 1.69 => The data is highly skewed.

Summary:
Total Normal Features: 1
Total Non-Normal Features: 4
```

our dataset has 4 non-normal features and 1 normal feature. Hence, our correlation test will be based on Spearman for the numerical features. Also we will perform Chi-Square test for he categorial (includes dummy) features.

<u>Spearman test:</u>

Evaluating the monotonic relationship between variables without assuming linearity. Calculates the Spearman's Rank Correlation matrix for the numerical columns in a DataFrame and visualizes it as a heatmap. The heatmap uses color gradients and annotated values to show the strength and direction of correlations between pairs of columns.



Chi-Square tests:

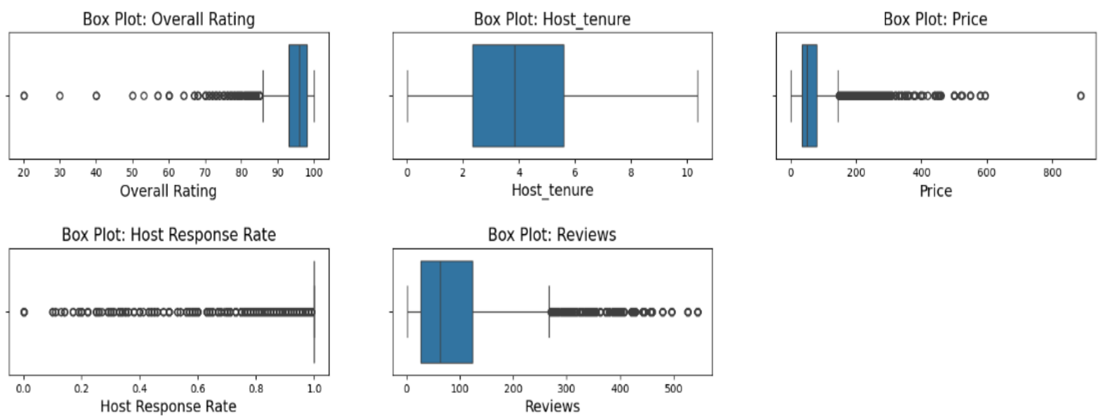
Assessing the independence between categorical variables by comparing observed and expected frequencies.

- Almost all of the relationships tested are highly significant (p-value = 0.0000), which means the variables (ratings, property details, booking features) we tested are closely related to each other.
- The only exception is the review_mnt variable, where the association with Communication Rating is weak but still statistically significant.

```
--------------------------------------------------
Chi-Square Test between Communication Rating and review_mnt:
Chi-Square Statistic: 95.07, p-value: 0.0795
--------------------------------------------------
```

Outlier Detection and Treatment:

To detect potential outliers in our numeric features, we utilized boxplots for each relevant column:



After visually inspecting the boxplots, we decided to retain the outliers in the following cases:

- **Overall Rating & Host Response Rate**: These columns contain ratings influenced by guest satisfaction. Outliers in these features typically represent low ratings, which are less frequent compared to higher ratings. Since these lower ratings provide valuable information about user dissatisfaction, we decided that retaining them is important for a more accurate understanding of customer feedback.

- **Price**: The price feature includes values that may appear as outliers, typically representing higher-priced properties compared to the average. However, these high prices are reflective of certain properties that are truly more expensive due to their premium nature (e.g., luxury listings). Removing these outliers would distort the true variation in pricing, so we chose to retain them as they represent a valid portion of the dataset.

- **Reviews:** This feature shows the number of reviews each listing has received. Outliers in this case may represent properties that have accumulated a significantly higher number of reviews compared to the average. These high values often indicate highly popular listings with frequent guest turnover, which are an important part of the dataset. Removing these outliers would eliminate valuable information regarding the most active properties, so we decided to retain them.

<u>Data Imputation using MICE:</u>

Our data frame had some features containing few hundreds of missing values per feature. As a result we imputed them following the steps below:

- **Identifying Missing Data:** We first identified which features (columns) had missing values. This is important because missing data can impact the accuracy of our models.

- **Filling in Missing Values:** For each feature with missing data, we filled in the gaps usinf MICE. This method ensures that the imputed values are realistic and consistent with the overall data distribution.

- **Ensuring Complete Data:** After filling the missing values, we checked the dataset to confirm that all gaps were successfully filled and no missing data remained.

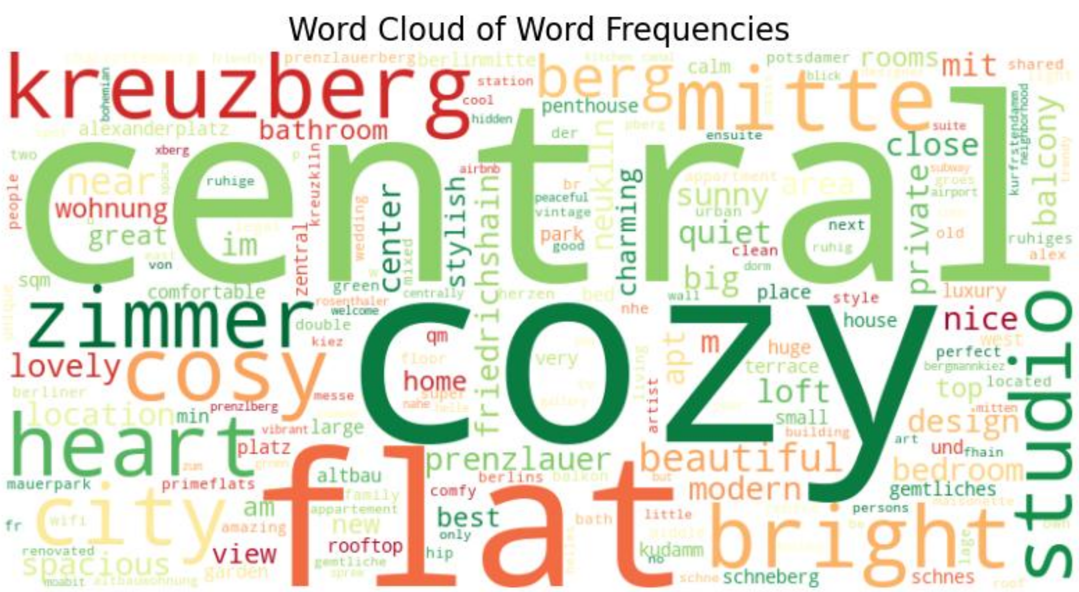
Column	Before:	After:
review_date	0	0
Comments	0	0
Listing ID	0	0
Listing Name	0	0
Host ID	0	0
Host Response Time	0	0
Host Response Rate	0	0
Is Superhost	0	0
Neighborhood Group	0	0
Latitude	0	0
Longitude	0	0
Is Exact Location	0	0
Property Type	0	0
Room Type	0	0
Accomodates	0	0
Bathrooms	135	0
Bedrooms	181	0
Beds	25	0
Price	0	0
Guests Included	0	0
Min Nights	0	0
Reviews	0	0
First Review	0	0
Last Review	0	0
Overall Rating	0	0
Accuracy Rating	164	0
Cleanliness Rating	164	0
Checkin Rating	167	0
Communication Rating	166	0
Location Rating	167	0
Value Rating	167	0
Instant Bookable	0	0
review_mnt	0	0
Host_tenure	0	0

With the data now cleaned and imputed, we have a solid foundation for the next phase of the project: Feature Engineering, where we will enhance the dataset by altering our features and preparing our Data Frame for feature selection.

Feature engineering:

In the Feature Engineering phase, we focused on transforming and enhancing the dataset to better capture the patterns relevant for predicting property prices. Here's a breakdown of the steps we took:

Word Cloud:



To gain insights into the most frequently mentioned keywords in the property descriptions, we created a word cloud. This visualization highlights the most common words, giving us a quick overview of key themes and features associated with the listings.

Sentiment Analysis:

To enhance the prediction model for property prices on Airbnb in Berlin, we applied sentiment analysis on the review comments from 2018 to capture the emotional tone of guest feedback. This was done using Natural Language Processing (NLP) techniques.

- **Language Detection:**

  We used a function to identify whether a comment is written in English. Since the dataset might include comments in different languages, it's crucial to focus only on the English ones to ensure more accurate analysis.

  We filtered out non-English comments to ensure the sentiment analysis (discussed next) is based on relevant, English-language feedback, providing a more accurate understanding of guest opinions and experiences.

- **Sentiment Analysis:**

  We used the TextBlob library to analyze the sentiment of each comment. TextBlob calculates a polarity score between -1 (negative) and +1 (positive). Based on the polarity score:

  - Positive sentiment: Polarity > 0
  - Negative sentiment: Polarity < 0
  - Neutral sentiment: Polarity = 0

  Outcome: A new column called Sentiment was created in the dataset, which assigns a value of 1 (positive), -1 (negative), or 0 (neutral) to each review.

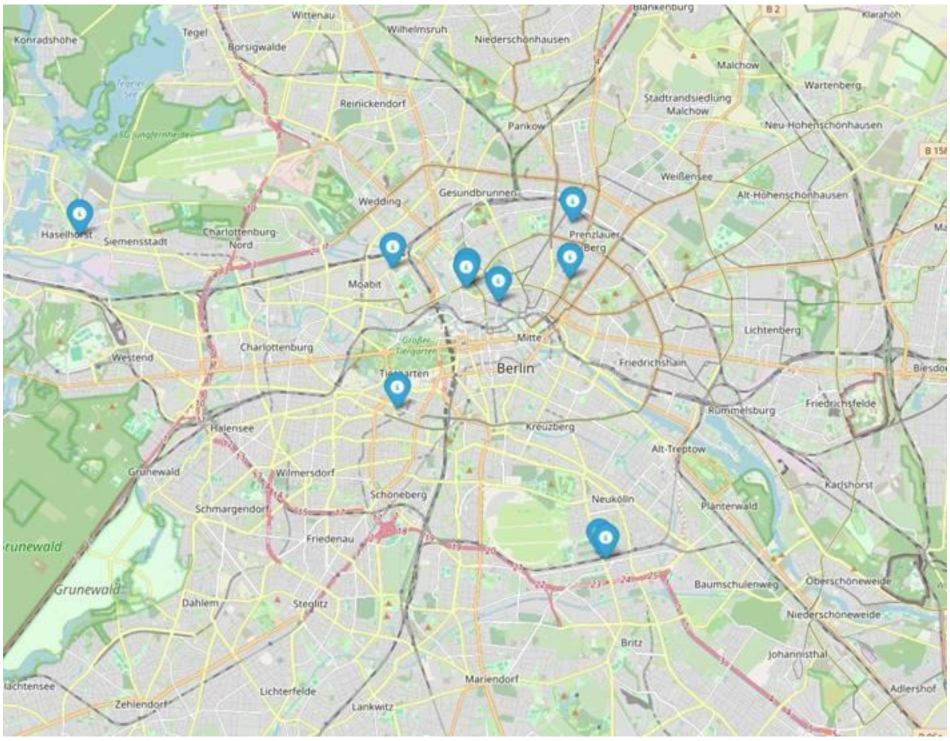
```
                                                 Comments    Review ID  \
3       Fritz has a really nice flat in a cosy neighbo...  269857683.0
4       Carlos place is amazing, nicely decorated, goo...  329973125.0
5       Nice and confortable place, perfect with kids,...  248757642.0
7       Great apartment.  About a 10 min walk from the...  223471676.0
8       Cosy and well situated apartment, exatly as de...  225091235.0
...                                                   ...          ...
157008                     Everything went fine and easy.  357458819.0
157009  Great location; comfortable, clean & quiet acc...  358649474.0
157010  Great location and great host. Martin provides...  359590777.0
157011  This place is awesome! it has everything from ...  361695756.0
157012  Martin's home is great, the location is perfec...  362784618.0

        Listing ID  Is_English  Sentiment
3         10012243        True          1
4         10012834        True          1
5         10029891        True          1
7         10031729        True          1
8         10031729        True          1
...            ...         ...        ...
157008     9994644        True          1
157009     9994644        True          1
157010     9994644        True          1
157011     9994644        True          1
157012     9994644        True          1
```

Our comments divide into:

- 98826 positive comments (95.4%)
- 3604 natural comments (3.5%)
- 1112 negative comments (1.1%)

Eventually we saved this output as CSV and we will aggregate it to the data frame later.

Dynamic Map of Berlin:



We created a dynamic map to visually explore the geographical distribution of the listings across Berlin. This map displayed the locations of several listings from our dataset, helping us to identify potential trends and patterns based on geographic factors. This visualization is useful for understanding how location might influence pricing and other factors in future analyses.

Changing Data Perspective (Listing ID View):

Previously, our dataset was organized at the review level, with multiple rows for each listing due to repeated reviews. However, to prevent model bias towards properties with a high number of reviews, we shifted the dataset focus from the review level to the listing level. This transformation helped maintain a balanced representation across all listings, resulting a row reduction from approximately 157,014 to 12,217 records. This adjustment made the dataset more manageable and better aligned with our analysis objectives.

- Execution:
    - Define the Grouping and Aggregation Columns: We first identified the columns that required aggregation and specified how each column should be processed.

```
# Define the groupby column and lists
groupby = ['Listing ID']

mean = ['Host Response Rate', 'Accomodates', 'Bathrooms', 'Bedrooms', 'Beds', 'Price', 'Guests Included',
        'Min Nights', 'Reviews', 'Overall Rating', 'Accuracy Rating', 'Cleanliness Rating', 'Checkin Rating',
        'Communication Rating', 'Location Rating', 'Value Rating', 'review_mnt', 'Host_tenure']

dele = ['Host ID', 'Comments', 'Listing Name', 'Latitude', 'Longitude','First Review', 'Last Review']

first = ['Host Response Time', 'Is Superhost', 'Neighborhood Group', 'Is Exact Location', 'Property Type',
         'Room Type',  'Instant Bookable']

count = ['review_date']  # New list for count aggregation
```

- With our plan in place, we used pandas to group the data by Listing ID. We applied different aggregation functions to each group of columns:
  - **Mean** for numerical features,
  - **First** for categorical features,
  - **Count** for reviews,
  - **Sum** for reviews if needed.

  Next, we cleaned up the DataFrame by removing unnecessary columns and renamed the review_date column to Total Bookings to reflect the count of reviews per listing.

```python
# Group by 'Listing ID' and aggregate according to the provided lists
grouped = df.groupby(groupby).agg({
    **{col: 'mean' for col in mean},  # Apply mean aggregation to columns in 'mean'
    **{col: 'first' for col in first},  # Apply first aggregation to columns in 'first'
    **{col: 'sum' for col in ['Reviews']},  # Sum reviews separately (if necessary)
    **{col: 'count' for col in count}  # Apply count aggregation to columns in 'count'
}).reset_index()

# Optionally, drop unwanted columns (from the 'dele' list)
df_grouped = grouped.drop(columns=dele, errors='ignore')  # 'errors=ignore' ensures no error if some columns are missing
#df_grouped['Total Bookings'] = df_grouped['review_date']
df_grouped.rename(columns={'review_date': 'Total Bookings'}, inplace=True)

# Print the grouped DataFrame
df_grouped
```

Creating Binary Sentiment Features (from NLP output):

- Binary Sentiment Columns: We created separate columns for neutral, negative, and positive sentiments to allow more detailed analysis of the feedback.
- Sentiment Aggregation: By counting the number of sentiments per listing, we aggregated sentiment data by listing to understand overall feedback volume and sentiment distribution for each property.
  - We aggregated the neutral, negative, and positive sentiment counts by listing to provide an overall sentiment summary for each property.
  - We calculated the percentage of each sentiment type to understand the relative distribution of feedback for each listing
  - We removed the raw sentiment counts, leaving the more insightful aggregated data (total sentiment count and sentiment percentages) for each listing.

Category Consolidation:

We consolidated certain categories to simplify the dataset and make it more meaningful for the model.

- **Rating features** like 'Communication Rating', 'Checkin Rating', 'Cleanliness Rating', 'Location Rating', 'Accuracy Rating', and 'Value Rating'.
  Ratings above 8 were kept as is, while ratings 8 or below were grouped into a single category ('8-'). This helped reduce the number of categories, making the data cleaner and more interpretable.
- **Bathrooms** were grouped into '1+' if there was more than one bathroom.
- **Bedrooms** were grouped into '3+' if there were three or more bedrooms.
- **Beds** were grouped into '7+' if there were seven or more beds.

This consolidation helped simplify the feature space while preserving meaningful distinctions in the data.

Category Encoding:

We encoded categorical features to prepare them for modeling, since most machine learning algorithms require numerical inputs. For this, we used categorical encoding, where each category in a column was assigned a unique numeric code.

We encoded the following columns: Accuracy Rating, Cleanliness Rating, Checkin Rating, Communication Rating, Neighborhood Group, Location Rating, Value Rating, Bathrooms, Bedrooms, Beds, Host Response Time, Property Type, and Room Type.

This encoding process allowed us to convert categorical data into a numeric format, which is necessary for applying machine learning algorithms.

Our Final Dataset heading to Feature Selection:

```
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Listing ID                     12217 non-null  int64
 1   Host Response Rate             12217 non-null  float64
 2   Accomodates                    12217 non-null  float64
 3   Price                          12217 non-null  float64
 4   Guests Included                12217 non-null  float64
 5   Min Nights                     12217 non-null  float64
 6   Reviews                        12217 non-null  float64
 7   Overall Rating                 12217 non-null  float64
 8   review_mnt                     12217 non-null  Float64
 9   Host_tenure                    12217 non-null  float64
 10  Is Superhost                   12217 non-null  bool
 11  Is Exact Location              12217 non-null  bool
 12  Instant Bookable               12217 non-null  bool
 13  Total Bookings                 12217 non-null  int64
 14  Sentiment_All                  12217 non-null  float64
 15  Sentiment_Nut_per              12217 non-null  float64
 16  Sentiment_Neg_per              12217 non-null  float64   FROM NLP
 17  Sentiment_Pos_per              12217 non-null  float64
 18  Accuracy Rating_encoded        12217 non-null  int8
 19  Cleanliness Rating_encoded     12217 non-null  int8
 20  Checkin Rating_encoded         12217 non-null  int8
 21  Communication Rating_encoded   12217 non-null  int8
 22  Neighborhood Group_encoded     12217 non-null  int8
 23  Location Rating_encoded        12217 non-null  int8
 24  Value Rating_encoded           12217 non-null  int8    Encoded
 25  Bathrooms_encoded              12217 non-null  int8    features
 26  Bedrooms_encoded               12217 non-null  int8
 27  Beds_encoded                   12217 non-null  int8
 28  Host Response Time_encoded     12217 non-null  int8
 29  Property Type_encoded          12217 non-null  int8
 30  Room Type_encoded              12217 non-null  int8
dtypes: Float64(1), bool(3), float64(12), int64(2), int8(13)
```

Feature Selection:

The goal of this step is to identify the most important features for predicting the price of properties listed on Airbnb in Berlin.

Methodology: We employed several machine learning models to assess the importance of each feature. The models used were Lasso Regression, Gradient Boosting Regressor, Random Forest Regressor, Ridge Regression, and XGBoost Regressor. Each model provided a different perspective on feature importance, and we combined their results to make informed decisions.

Process:

- Define Feature Set and Target Variable: We defined our feature set X by excluding the target variable Price from the DataFrame. The target variable y was set as the Price column.

- Implement Models and Select Features:
    - **Lasso Regression:** Applied L1 regularization to shrink less important feature coefficients to zero. Features with non-zero coefficients were selected.
    - **Gradient Boosting Regressor:** Used decision trees to provide feature importances. Features with positive importances were selected.
    - **Random Forest Regressor:** Aggregated results of multiple decision trees to provide feature importances. Features with positive importances were selected.
    - **Ridge Regression:** Applied L2 regularization, similar to Lasso but with no sparsity. Features with non-zero coefficients were selected.
    - **XGBoost Regressor:** Used gradient boosting to provide feature importances. Features with positive importances were selected.

Results:

We summarized the models' score for each feature in a table:

	Feature	Lasso	GradientBoost	RandomForest	Ridge	XGBoost	Sum
0	Listing ID	1	1	1	1	1	5
1	Host Response Rate	1	1	1	1	1	5
2	Accomodates	1	1	1	1	1	5
3	Guests Included	1	1	1	1	1	5
4	Min Nights	1	1	1	1	1	5
5	Reviews	1	1	1	1	1	5
6	Overall Rating	1	1	1	1	1	5
7	review_mnt	1	1	1	1	1	5
8	Host_tenure	1	1	1	1	1	5
9	Is Superhost	1	1	1	1	1	5
10	Is Exact Location	1	0	1	1	1	4
11	Instant Bookable	1	1	1	1	1	5
12	Total Bookings	1	1	1	1	1	5
13	Sentiment_All	1	1	1	1	1	5
14	Sentiment_Nut_per	1	1	1	1	1	5
15	Sentiment_Neg_per	0	1	1	1	1	4
16	Sentiment_Pos_per	1	1	1	1	1	5
17	Accuracy Rating_encoded	1	0	1	1	1	4
18	Cleanliness Rating_encoded	1	1	1	1	1	5
19	Checkin Rating_encoded	1	0	1	1	1	4
20	Communication Rating_encoded	1	1	1	1	1	5
21	Neighborhood Group_encoded	1	1	1	1	1	5
22	Location Rating_encoded	1	1	1	1	1	5
23	Value Rating_encoded	1	1	1	1	1	5
24	Bathrooms_encoded	1	1	1	1	1	5
25	Bedrooms_encoded	1	1	1	1	1	5
26	Beds_encoded	1	1	1	1	1	5
27	Host Response Time_encoded	1	1	1	1	1	5
28	Property Type_encoded	1	1	1	1	1	5
29	Room Type_encoded	1	1	1	1	1	5

The features with the lowest score (4/5) that we decided to remove before the Model Selection phase are:

	Feature	Lasso	GradientBoost	RandomForest	Ridge	XGBoost	Sum
10	Is Exact Location	1	0	1	1	1	4
15	Sentiment_Neg_per	0	1	1	1	1	4
17	Accuracy Rating_encoded	1	0	1	1	1	4
19	Checkin Rating_encoded	1	0	1	1	1	4

Model Selection:

With a range of machine learning algorithms available, each with its own strengths and weaknesses, we carefully considered several options to find the best fit for the data. This step involved testing a variety of models, from simpler ones like Linear Regression to more complex ensemble methods, to evaluate how well they could generalize and deliver accurate price predictions. After evaluating their performance across multiple metrics, we were able to identify the most reliable model for this task.

TRAIN/TEST

the models considered included:

- Linear Regression
- Decision Tree
- Random Forest
- ADABoost
- Gradient Boosting Machine (GBM)
- Support Vector Machine (SVM)
- XGBoost

These models represent a mix of simple and complex algorithms, ranging from linear approaches (Linear Regression) to ensemble methods (Random Forest, XGBoost), each chosen to assess different modeling strategies and capture the data's underlying patterns.

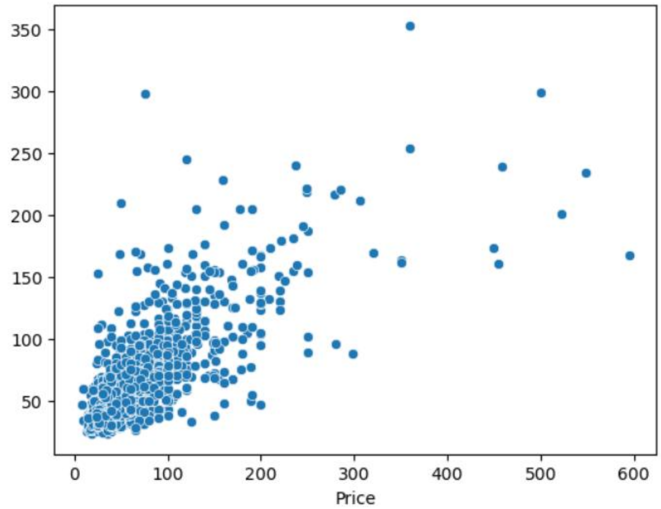
To assess and compare model performance, four key regression metrics were used the following evaluation metrics:

- Mean Squared Error (**MSE**): Tells us how far off the model's predictions were from the actual prices, with lower values being better.
- Root Mean Squared Error (**RMSE**): Gives us an idea of the average prediction error, in the same units as the property prices, making it easier to understand.
- Mean Absolute Error (**MAE**): Measures the average size of the errors in the predictions, helping us understand how much the model's predictions are off, on average.
- Root Mean Squared Logarithmic Error (**RMSLE**): This metric helps to handle large price differences by penalizing under-predictions more, which is useful for real-world property price data.

The results of each model summed up in a comparison table, order by MAE:

	model	MSE	RMSE	MAE	RMSLE
2	RandomForest	1135.687989	33.699970	19.056473	0.385448
4	GBM	1262.956631	35.538101	19.381132	0.398192
6	XGB	1250.342624	35.360184	19.934412	0.405870
0	Linear Regression	1316.639827	36.285532	20.669397	0.443405
1	Decision Tree	1549.950824	39.369415	23.232589	0.457893
5	SVM	2780.709857	52.732436	29.119132	0.582994
3	ADABoost	2337.537473	48.348087	38.195332	0.706123

In summary, the **Random Forest** model was clearly the best-performing model for predicting property prices, as it achieved the lowest error rates across all evaluation metrics.



Fine-Tuning the Random Forest Model:

After selecting the Random Forest model, we moved on to fine-tuning its parameters to further improve its performance. This was done by using GridSearchCV, a technique that automatically searches for the best combination of hyperparameters.

- Max Depth: The maximum number of levels the decision tree can have.
- Min Samples Split and Leaf: The minimum number of samples required to split an internal node or to form a leaf node.
- Max Features: The number of features to consider when looking for the best split.
- Criterion: The function to measure the quality of a split.

Despite performing hyperparameter tuning using Grid Search, we did not observe significant improvements in the model's performance, suggesting that the initial hyperparameters may already be well-suited for the dataset.

	model	MSE	RMSE	MAE	RMSLE
2	RandomForest	1135.687989	33.699970	19.056473	0.385448
4	GBM	1262.956631	35.538101	19.381132	0.398192
6	XGB	1250.342624	35.360184	19.934412	0.405870
0	Linear Regression	1316.639827	36.285532	20.669397	0.443405
7	RandomForest_FT	1191.343225	34.515840	20.939623	0.398375
1	Decision Tree	1549.950824	39.369415	23.232589	0.457893
5	SVM	2780.709857	52.732436	29.119132	0.582994
3	ADABoost	2337.537473	48.348087	38.195332	0.706123

Conclusion:

In summary, the **Random Forest** model proved to be the best at predicting property prices for Airbnb listings in Berlin. With the lowest error rates across all evaluation metrics (MSE, RMSE, MAE, and RMSLE), it demonstrated strong predictive power,Its price predictions were off by approximately 19 EUR (MAE).

Thank you