BOSTON UNIVERSITY

DATA ANALYSIS

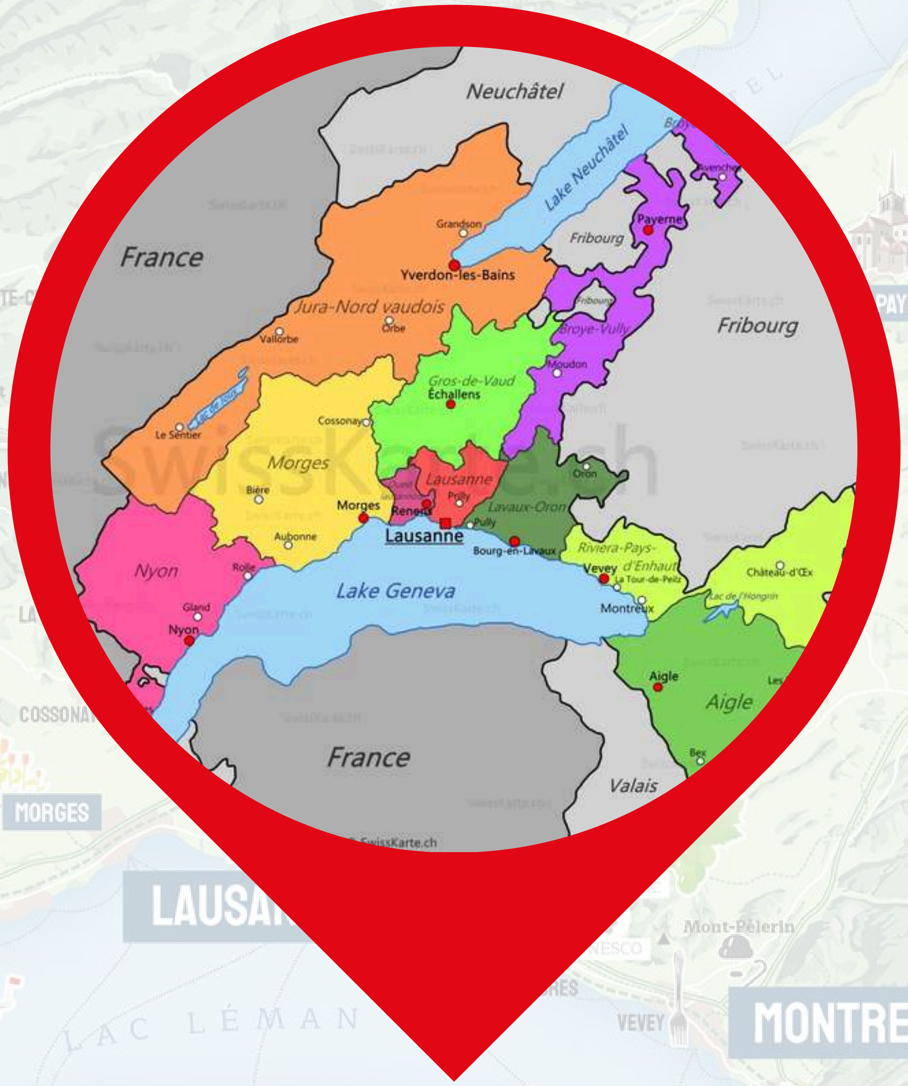
VAUD'S AIRBNB RENTALS MARKET

Presentation by:

Prabu, Moiz, Jitvan

Data Mining - Spring'25

LOCATION



CANTON OF VAUD
Switzerland

Why Vaud?

- We chose Vaud for our Airbnb analysis because it offers a diverse mix of urban, suburban, and rural listings, making it ideal for uncovering meaningful patterns in rental behavior.
- The region includes popular tourist destinations like Lausanne and Montreux, which contribute to variability in price, room types, and demand essential factors for effective clustering and modeling.
- Additionally, Vaud has a rich dataset with 75 columns and 5286 rows which gives sufficient volume and distinct seasonal trends, enabling more robust analysis and insightful comparisons.

DATA CLEANING

No.	STEP	PURPOSE	IMPACT
1.	Dropped columns with >40% missing values	Sparse columns like license, host_about, and neighbourhood lack reliable data for modeling or exploration.	Reduced noise and focused the dataset on useful variables.
2.	Removed rows missing price	price is our target variable; we need it for prediction and evaluation.	Ensures our modeling efforts are based on complete, valid target data.
3.	Filled numeric columns (beds, bedrooms, bathrooms) with median	Median is less sensitive to outliers and better represents central tendency in skewed data.	Preserved useful records while minimizing the influence of extreme values.
4.	Filled review-related columns with 0	Missing values in review columns likely indicate no reviews at all.	0 serves as a meaningful placeholder, enabling fair treatment in models.
5.	Converted and filled % columns	Converted host_response_rate and host_acceptance_rate to numeric and filled with median.	Prevented data type errors and standardized important host metrics.
6.	Imputed categorical fields with mode or placeholders	Used most common values or descriptive placeholders like "Unknown" or "No description provided."	Maintained row integrity while preserving interpretability.
7.	Dropped rows with no review dates	First/last review dates are important for time-based analysis.	Ensured consistency for trend analysis and time-based features.

EDA

SUMMARY STATISTICS

Median Price by Neighborhood

Ollon and Ormont-Dessus are the most expensive neighborhoods with median prices of $237.5 and $214 respectively, while Lausanne has the most affordable median price at $99. This suggests Ollon and Ormont-Dessus cater to higher-end travelers or offer larger accommodations.

```
Median Price by Neighbourhood:

 neighbourhood_cleansed
Ollon              237.5
Ormont-Dessus      214.0
Château-d'Oex      194.0
Gryon              193.0
Montreux           150.0
Leysin             137.0
Lutry              130.0
Nyon               114.0
Morges             107.0
Lausanne            99.0
Name: price, dtype: float64
```

Average Bedrooms by Neighborhood

Neighborhoods like Château-d'Oex, Gryon, Ollon, and Ormont-Dessus average 2 bedrooms per listing, indicating larger properties. In contrast, the rest—including Lausanne, Nyon, and Morges—typically offer 1-bedroom accommodations, which may appeal more to solo travelers or couples.

```
Average Bedrooms by Neighbourhood:

 neighbourhood_cleansed
Château-d'Oex     2
Ormont-Dessus     2
Gryon             2
Ollon             2
Leysin            1
Montreux          1
Lutry             1
Nyon              1
Lausanne          1
Morges            1
Name: bedrooms, dtype: int64
```

Average Review Scores by Neighborhood

Ormont-Dessus tops the list with a high average review score of 4.88, followed closely by Château-d'Oex (4.85) and Leysin (4.84). This reflects consistently positive guest experiences in these areas, suggesting strong host engagement and overall satisfaction.

```
Average Review Score by Neighbourhood:

 neighbourhood_cleansed
Ormont-Dessus     4.877976
Château-d'Oex     4.845588
Leysin            4.837568
Gryon             4.768480
Ollon             4.763556
Nyon              4.706154
Montreux          4.685040
Lausanne          4.683962
Lutry             4.670217
Morges            4.581667
Name: review_scores_rating, dtype: float64
```

neighbourhood_cleansed	Château-d'Oex	Gryon	Lausanne	Leysin	Lutry	Montreux	Morges	Nyon	Ollon	Ormont-Dessus
property_type										
Casa particular	0	0	1	0	0	0	0	0	0	0
Castle	0	0	0	0	0	7	0	0	0	0
Entire cabin	1	0	0	0	0	0	0	0	0	1
Entire chalet	18	38	1	15	0	5	0	0	28	26
Entire condo	7	11	54	17	3	19	0	2	29	11
Entire guest suite	0	0	0	0	1	2	0	0	0	0
Entire guesthouse	0	0	1	1	1	0	0	0	0	0
Entire home	2	7	1	3	5	12	0	1	22	3
Entire loft	0	3	6	1	0	6	0	0	3	0
Entire place	1	0	0	0	1	0	0	0	0	0
Entire rental unit	34	56	362	65	21	126	46	23	179	33
Entire serviced apartment	0	0	1	2	0	9	3	0	3	0
Entire townhouse	0	0	1	0	0	0	0	0	0	0
Entire vacation home	0	2	0	5	0	1	0	0	3	1
Entire villa	0	0	0	0	2	4	0	0	0	0
Private room	1	0	0	0	0	0	0	0	0	0
Private room in bed and breakfast	1	8	18	0	4	5	0	7	4	2
Private room in casa particular	0	0	5	0	0	1	0	0	0	0
Private room in chalet	0	0	0	0	0	3	0	0	2	2
Private room in condo	1	0	5	0	0	3	1	0	0	0
Private room in farm stay	0	0	0	0	0	0	0	3	0	0
Private room in guest suite	0	0	1	0	0	1	0	0	0	0
Private room in guesthouse	0	0	1	0	0	0	0	0	0	0
Private room in home	1	0	8	1	4	4	2	4	5	0
Private room in hostel	0	0	2	0	0	1	0	0	0	0
Private room in nature lodge	0	0	0	0	0	0	0	0	3	0
Private room in rental unit	1	0	128	0	1	40	6	8	2	0
Private room in townhouse	0	0	2	0	0	0	0	0	0	0
Private room in villa	0	0	9	0	3	3	0	1	0	0
Room in boutique hotel	0	0	0	0	0	0	0	0	0	3
Room in heritage hotel	0	0	0	0	0	0	0	0	0	1
Room in hotel	0	0	10	0	0	0	2	0	0	1
Room in serviced apartment	0	0	1	0	0	0	0	0	0	0
Shared room in bed and breakfast	0	0	3	0	0	0	0	3	0	0
Shared room in rental unit	0	0	3	0	0	0	0	0	0	0
Tiny home	0	0	1	1	0	0	0	0	1	0
Yurt	0	0	1	0	0	0	0	0	0	0

Count of Listings by Property Type per Neighbourhood

Dominant Property Types

Entire rental units are the most frequently listed property type across all neighborhoods, especially in Lausanne (362 listings) and Ollon (179). This indicates a strong preference among hosts to rent out fully self-contained accommodations, likely catering to travelers seeking privacy and flexibility.

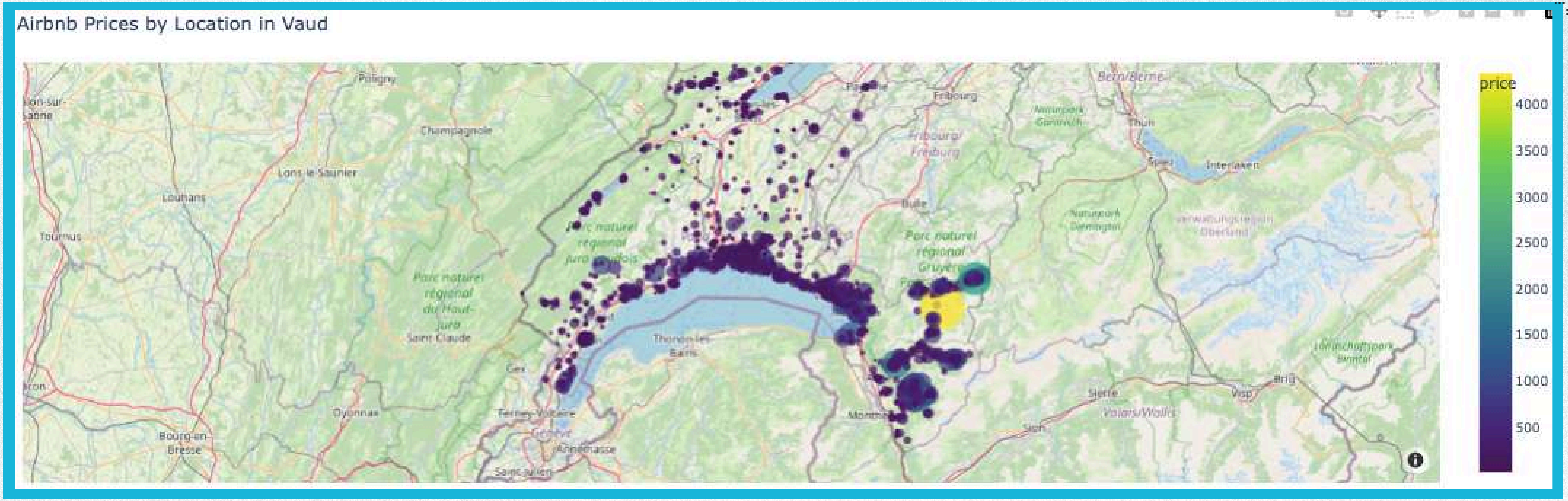
```
Median Availability (days per year) by Neighbourhood:

 neighbourhood_cleansed
Ormont-Dessus    287
Ollon            273
Gryon            238
Leysin           234
Château-d'Oex    218
Lausanne         195
Lutry            193
Montreux         179
Nyon             161
Morges            82
Name: availability_365, dtype: int64
```

Median Availability by Neighborhood

Ormont-Dessus and Ollon have the highest availability at 287 and 273 days/year, indicating year-round access and likely full-time listings. In contrast, Morges has the lowest median availability at just 82 days/year, pointing toward more seasonal or part-time use.

DATA VISUALIZATION

CHOROPLETH - MAP (PRICE BY LOCATION)



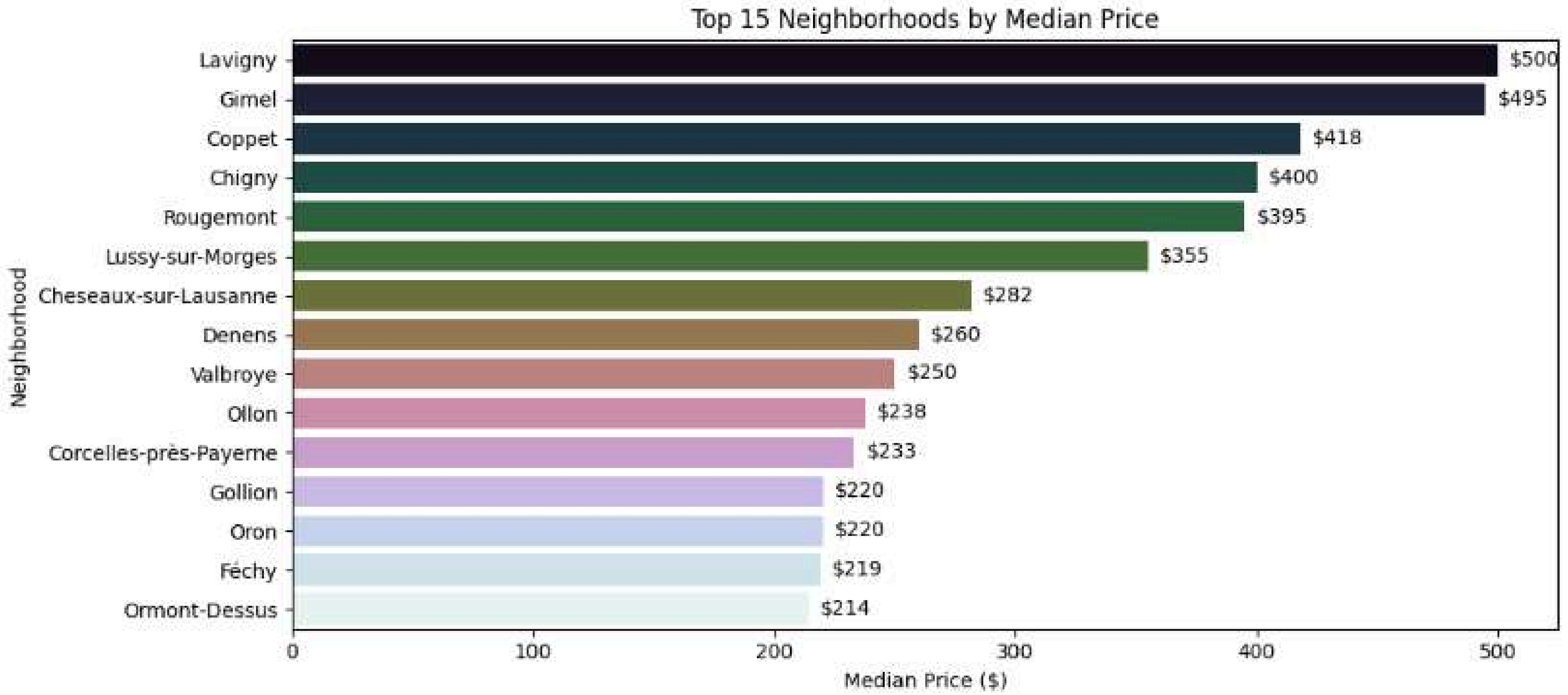
Luxury hotspots in Les Diablerets and Château-d'Oex show high prices, likely due to ski resorts and alpine retreats.

Moderate pricing ($100–$300) dominates the Lausanne–Montreux corridor, reflecting urban tourism and strong demand.

Affordable stays (<$150) cluster in rural and northern areas, appealing to long-term or local travelers.

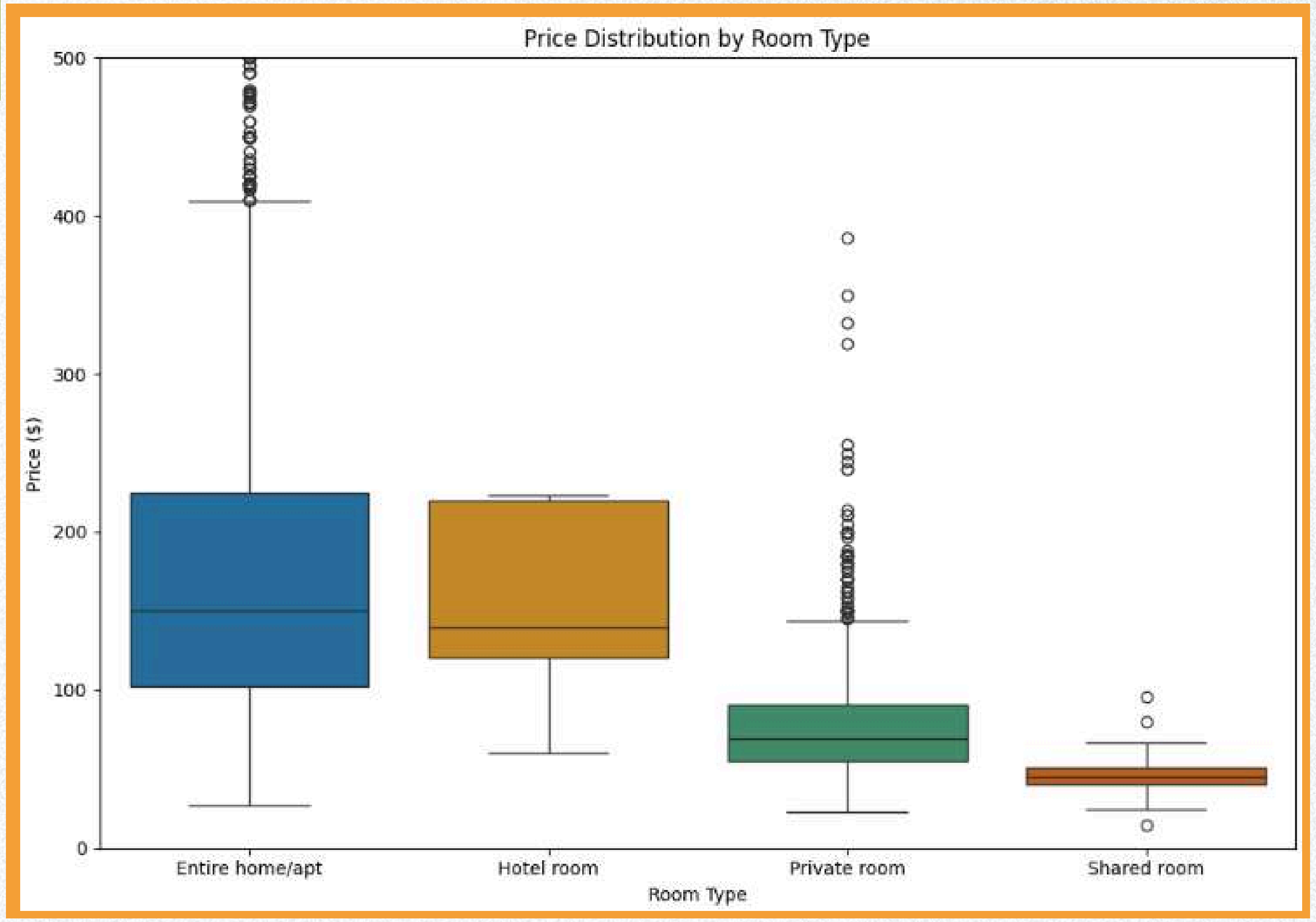
Sparse listings in remote zones still show premium prices, driven by exclusivity or seasonal demand.

BAR PLOT OF MEDIAN PRICE BY NEIGHBORHOOD



- Lavigny and Gimel have the highest median prices (~$500), likely due to luxury stays or scenic value.
- Coppet, Chigny, and Rougemont follow with prices over $400, indicating affluent or touristic appeal.
- Valbroye to Ormont-Dessus show moderate prices ($214–$250), suggesting simpler or more residential listings.

BOX PLOT OF PRICE DISTRIBUTIONS BY ROOM TYPE



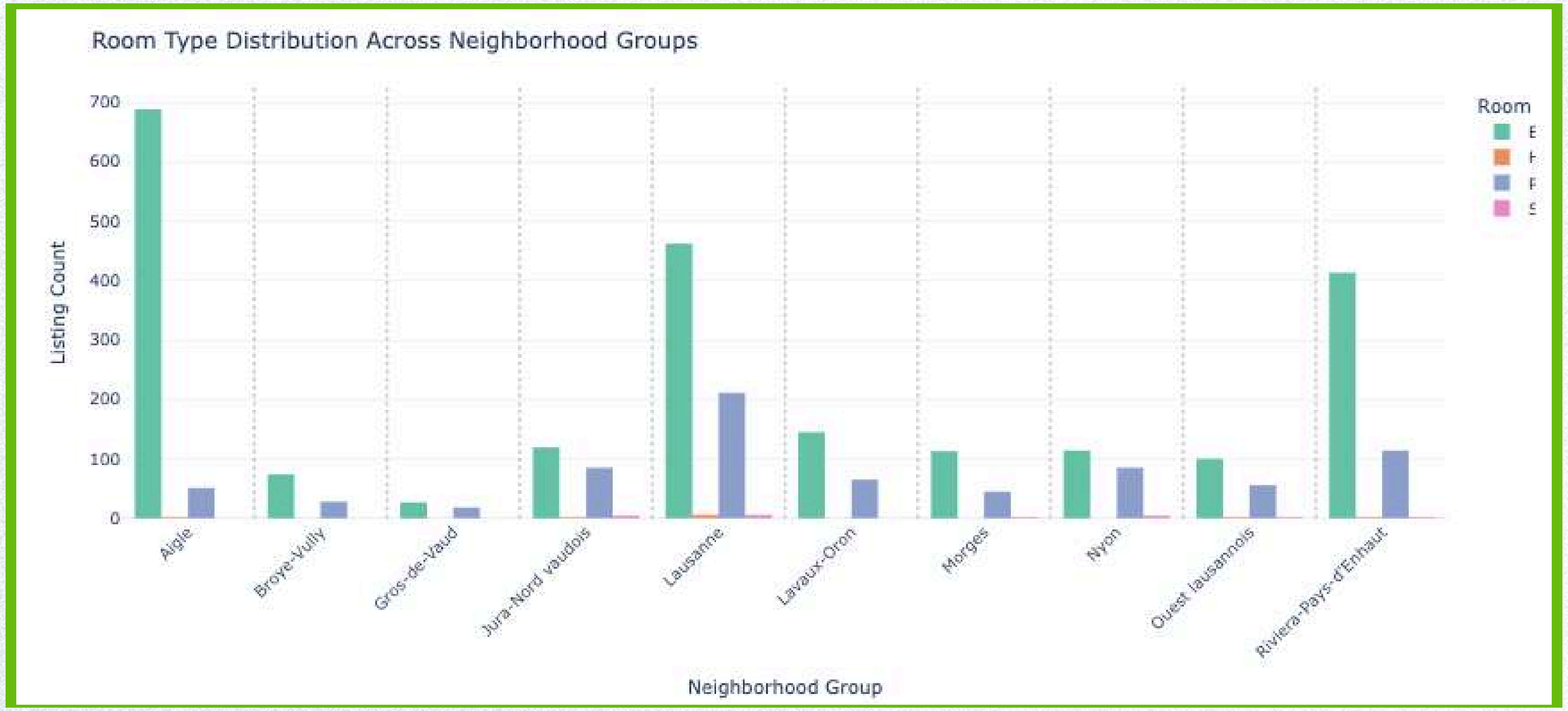
Entire home/apt
- Highest median price and widest range (~$30–$400+)
- Frequent outliers suggest luxury or scenic properties

Hotel room
- Moderately high and consistent prices (~$60–$225)
- Occasional outliers may reflect suites or seasonal spikes

Private room
- Affordable and stable pricing (~$25–$120)
- Outliers likely tied to premium locations or features

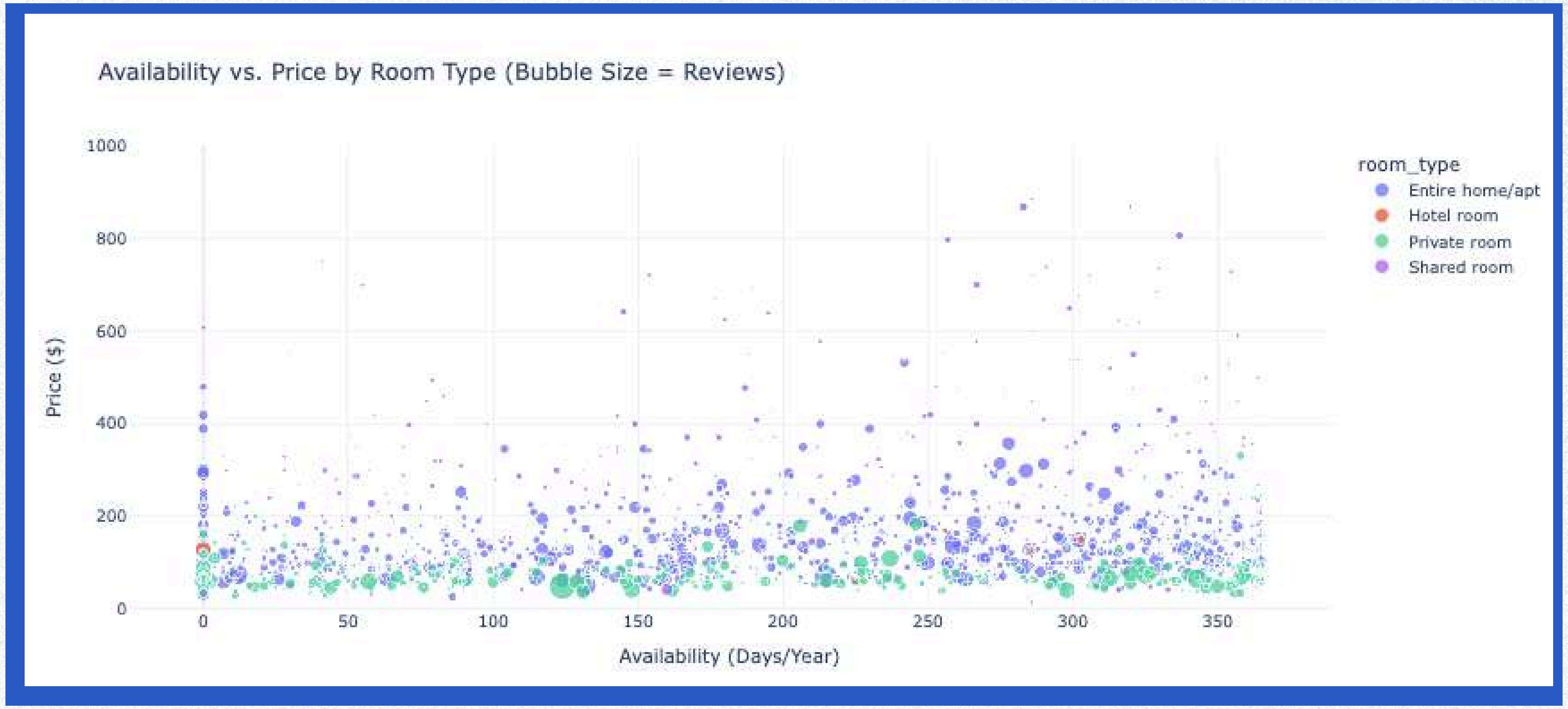
Shared room
- Lowest and most stable pricing (~$20–$70)
- Minimal variation, ideal for budget travelers

COUNT PLOT OF LISTINGS PER ROOM TYPE PER NEIGHBORHOOD GROUP



- Entire home/apartments dominate across all neighborhoods, especially in Aigle (688) and Lausanne (462).
- Private rooms are the next most common, with high counts in Lausanne (211) and Jura-Nord vaudois (85).
- Hotel rooms and shared rooms are rare, suggesting Airbnb in Vaud focuses on full-property and private rentals.

BUBBLE CHART: AVAILABILITY VS. PRICE BY ROOM TYPE (BUBBLE SIZE = REVIEWS)



Price & Availability by Room Type

- Entire homes span the widest price range, including luxury listings over $1000.
- These are available year-round or seasonally, appealing to flexible demand.
- Private rooms stay mostly under $150 and are widely available—popular among budget or long-term travelers.
- Hotel and shared rooms are few, priced lower, suggesting niche use.

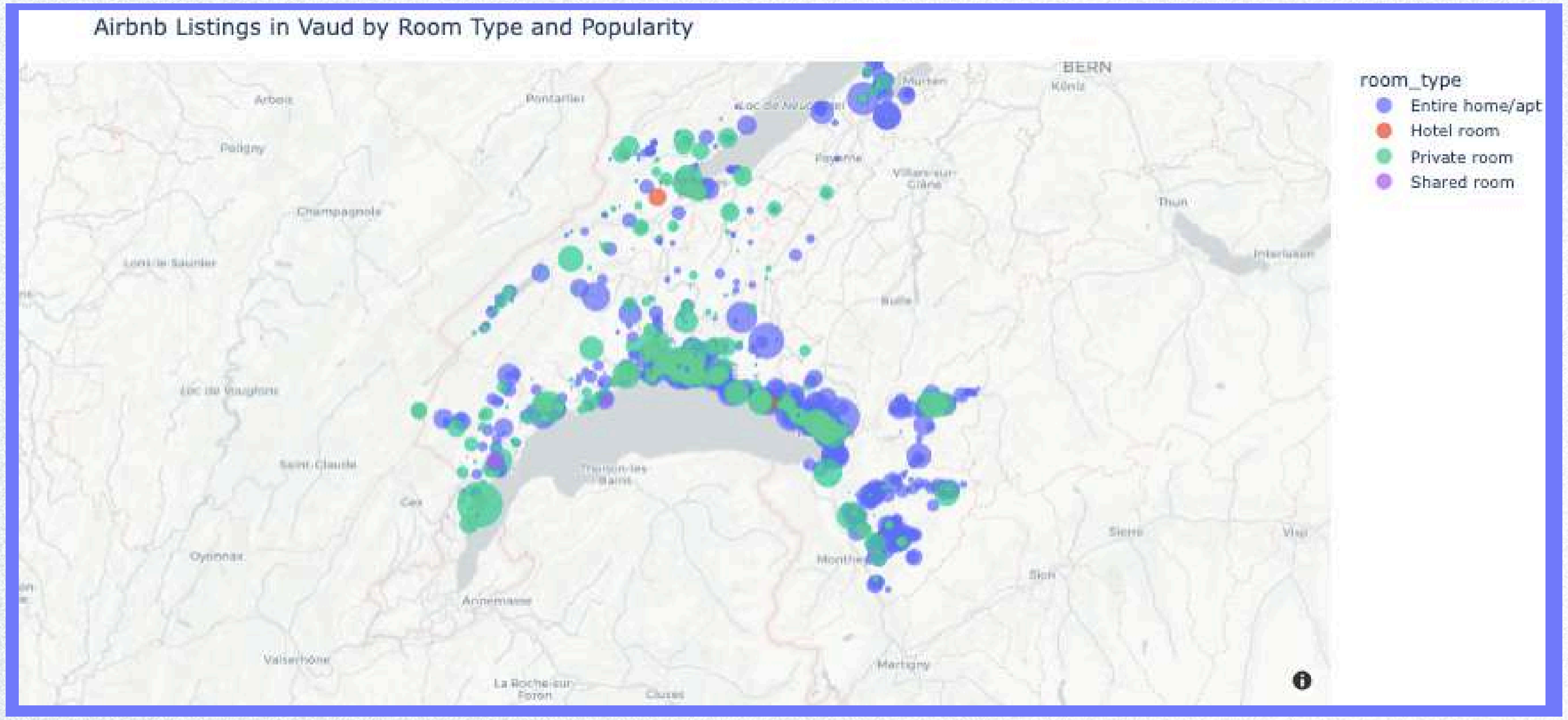
Reviews & Popularity

- Bubble size = popularity (based on review count).
- Most popular listings are affordable ($50–$200) and highly available (>200 days/year).
- These are mostly private rooms and entire homes.
- High-price listings have fewer reviews, indicating lower booking volume.

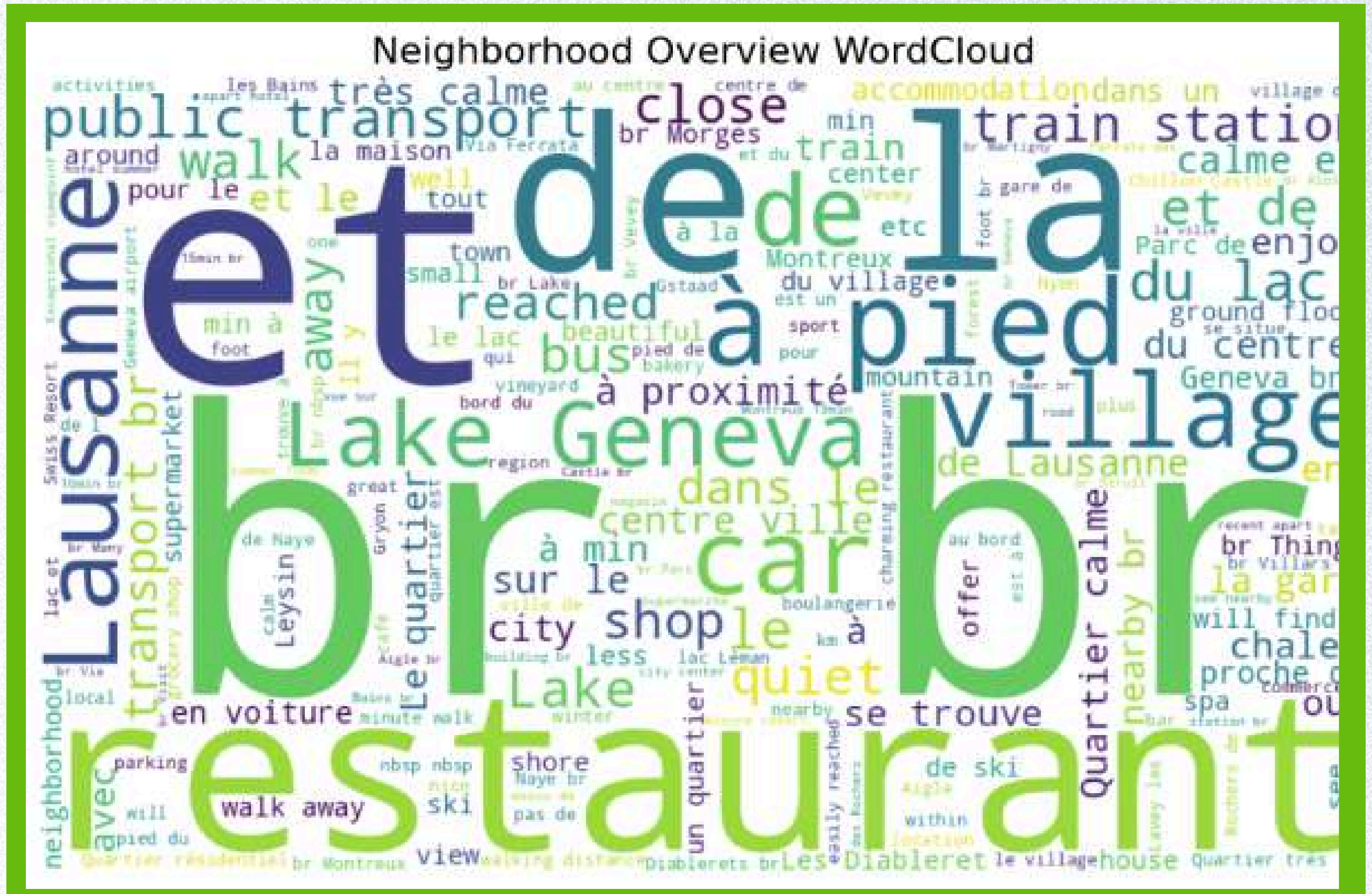
Key Takeaway

- Success = Reasonable pricing + High availability
- Listings in the mid-price range, especially those available year-round, attract the most engagement.

MAPPING



WORDCLOUD



Regional Clusters & Tourist Hubs

- Listings are densely clustered along Lake Geneva, especially in Lausanne, Montreux, and Vevey.
- These areas attract tourists due to scenic views, vineyards, and cultural attractions.

Room Type Distribution

- Entire homes/apartments dominate, especially in lakeside towns, reflecting demand for private stays.
- Private rooms are common in mid-sized areas, catering to solo or budget travelers.
- Hotel and shared rooms are limited across the region.

Popularity & Accessibility

- Listings with the most reviews are near tourist sites and public transport.
- Strategic location appears to influence listing popularity more than price alone

- **Walkability & Transport**: Frequent terms like "walk", "train station", and "public transport" reflect strong transit access and walkable neighborhoods.
- **Scenic & Quiet Atmosphere**: Words like "lake", "quiet", "village", and "mountain" point to peaceful, nature-rich settings—ideal for relaxation.
- **Tourist-Friendly Amenities**: Listings emphasize access to “restaurants,” “shops,” and “city center,” highlighting appeal for short-term visitors.

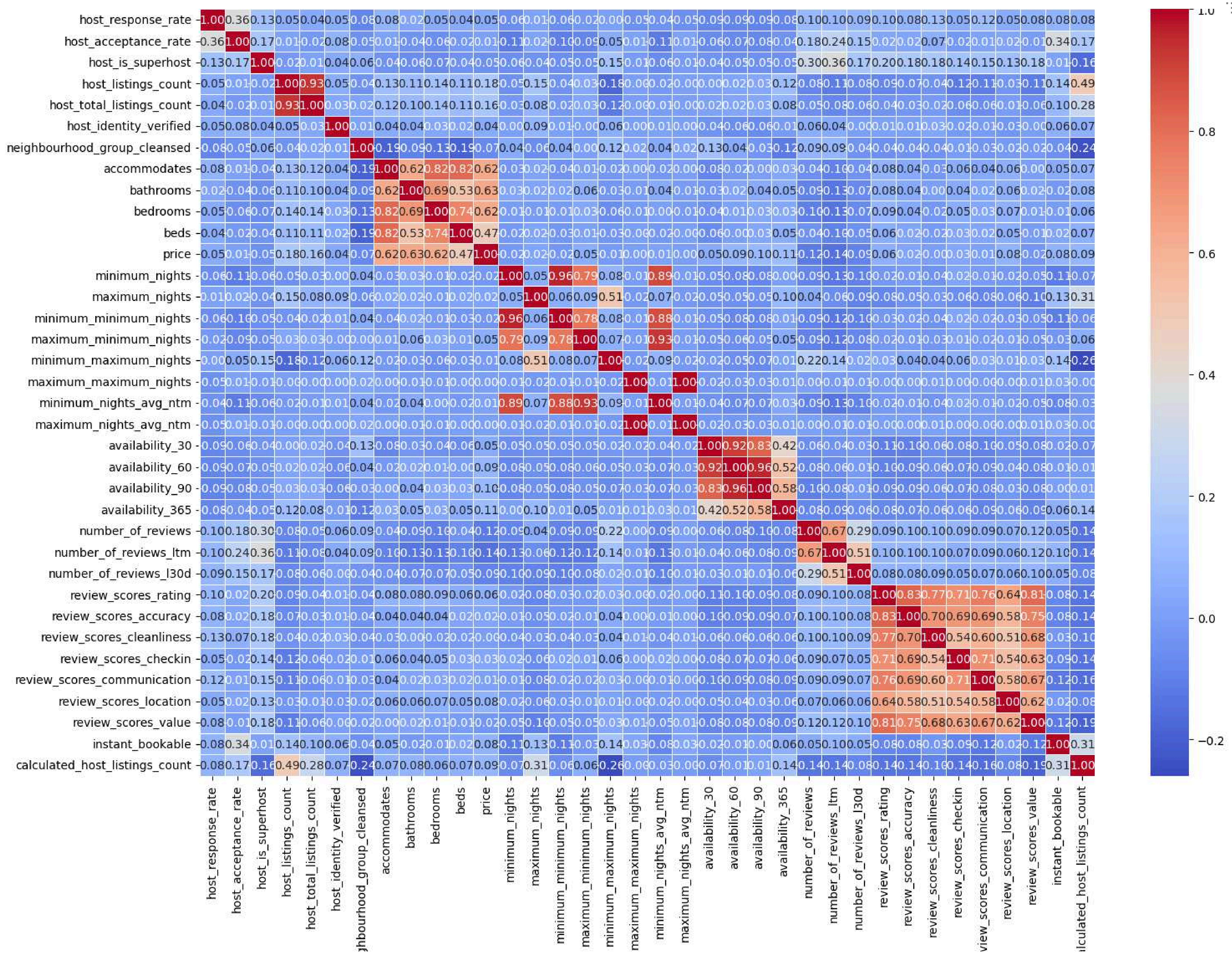
PREDICTION

OBJECTIVE – PREDICTING AIRBNB LISTING PRICE USING MLR

To build a Multiple Linear Regression (MLR) model to predict the price of Airbnb listings in Vaud, Switzerland, based on host attributes, property features, and review scores.

CORRELATION ANALYSIS OF AIRBNB LISTING VARIABLES

```
Correlation with Price (Score):
price                              1.000000
bathrooms                          0.633960
accommodates                       0.621484
bedrooms                           0.620820
beds                               0.474475
host_listings_count                0.175140
host_total_listings_count          0.164158
availability_365                   0.114099
availability_90                    0.102586
calculated_host_listings_count     0.092630
availability_60                    0.090546
instant_bookable                   0.078863
review_scores_location             0.075372
review_scores_rating               0.064519
maximum_minimum_nights             0.054202
availability_30                    0.047615
host_response_rate                 0.047323
host_identity_verified             0.043162
review_scores_checkin              0.025449
review_scores_accuracy             0.021787
maximum_nights                     0.019981
minimum_nights_avg_ntm             0.013586
review_scores_communication        0.009059
host_acceptance_rate               0.007875
maximum_maximum_nights             0.000006
maximum_nights_avg_ntm             0.000006
review_scores_cleanliness         -0.003149
minimum_maximum_nights            -0.014357
minimum_nights                    -0.022918
review_scores_value               -0.024731
minimum_minimum_nights            -0.024882
host_is_superhost                 -0.052959
neighbourhood_group_cleansed      -0.070403
number_of_reviews_l30d            -0.090482
number_of_reviews                 -0.117388
number_of_reviews_ltm             -0.143513
Name: price, dtype: float64
```



DIVERSE VARIABLE SELECTION FOR MLR MODEL

Host Attributes

- host_response_rate (N)
- host_acceptance_rate (N)
- host_is_superhost (C)
- host_identity_verified (C)
- calculated_host_listings_count (N)

Listing Characteristics

- accommodates (N)
- bathrooms (N)
- bedrooms (N)
- beds (N)
- minimum_nights (N)
- availability_365 (N)
- instant_bookable (C)
- neighbourhood_group_cleansed (C)

Review Metrics

- number_of_reviews_ltm (N)
- review_scores_rating (N)
- review_scores_cleanliness (N)

🎯 Target Variable

price (N)

MLR PRICE PREDICTION PIPELINE OVERVIEW

Selected key host attributes, listing characteristics, and review features.

Filtered the dataset to relevant columns.

One-hot encoded categorical variables

Log-transformed the price column.

Split data into features and target.

Created train-test split (80/20).

Scaled numeric features with StandardScaler.

Trained Linear Regression model.

Predicted and inverse-transformed log prices.

Evaluated with RMSE and R^2 metrics.

FEATURE SIGNIFICANCE FROM OLS REGRESSION

```
                            OLS Regression Results
==============================================================================
Dep. Variable:              log_price   R-squared:                       0.622
Model:                            OLS   Adj. R-squared:                  0.619
Method:                 Least Squares   F-statistic:                     172.4
Date:                Fri, 02 May 2025   Prob (F-statistic):               0.00
Time:                        17:42:09   Log-Likelihood:                -1226.5
No. Observations:                2432   AIC:                             2501.
Df Residuals:                    2408   BIC:                             2640.
Df Model:                          23
Covariance Type:            nonrobust
====================================================================================================================
                                                         coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------------------------------------------
const                                                  4.8826      0.008    597.989      0.000       4.867       4.899
host_response_rate                                     0.0097      0.009      1.080      0.280      -0.008       0.027
host_acceptance_rate                                   0.0233      0.010      2.414      0.016       0.004       0.042
accommodates                                           0.3590      0.018     20.288      0.000       0.324       0.394
bathrooms                                              0.1145      0.012      9.943      0.000       0.092       0.137
bedrooms                                               0.1337      0.016      8.142      0.000       0.102       0.166
beds                                                  -0.1433      0.015     -9.733      0.000      -0.172      -0.114
minimum_nights                                        -0.0376      0.008     -4.508      0.000      -0.054      -0.021
availability_365                                       0.0648      0.008      7.734      0.000       0.048       0.081
number_of_reviews_ltm                                 -0.0804      0.009     -8.707      0.000      -0.098      -0.062
review_scores_rating                                   0.0265      0.009      3.080      0.002       0.010       0.043
calculated_host_listings_count                         0.0183      0.009      1.943      0.052      -0.000       0.037
host_is_superhost_t                                    0.0070      0.009      0.759      0.448      -0.011       0.025
host_identity_verified_t                               0.0122      0.008      1.480      0.139      -0.004       0.028
neighbourhood_group_cleansed_Broye-Vully              -0.0258      0.009     -2.946      0.003      -0.043      -0.009
neighbourhood_group_cleansed_Gros-de-Vaud             -0.0320      0.009     -3.749      0.000      -0.049      -0.015
neighbourhood_group_cleansed_Jura-Nord vaudois        -0.0806      0.009     -8.579      0.000      -0.099      -0.062
neighbourhood_group_cleansed_Lausanne                 -0.0181      0.011     -1.581      0.114      -0.041       0.004
neighbourhood_group_cleansed_Lavaux-Oron               0.0082      0.009      0.868      0.385      -0.010       0.027
neighbourhood_group_cleansed_Morges                   -0.0153      0.009     -1.692      0.091      -0.033       0.002
neighbourhood_group_cleansed_Nyon                     -0.0224      0.009     -2.368      0.018      -0.041      -0.004
neighbourhood_group_cleansed_Ouest lausannois         -0.0291      0.009     -3.149      0.002      -0.047      -0.011
neighbourhood_group_cleansed_Riviera-Pays-d'Enhaut     0.0491      0.010      4.746      0.000       0.029       0.069
instant_bookable_t                                     0.0565      0.009      6.178      0.000       0.039       0.074
==============================================================================
Omnibus:                      142.898   Durbin-Watson:                   2.064
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              593.217
Skew:                          -0.041   Prob(JB):                    1.53e-129
Kurtosis:                       5.418   Cond. No.                         4.82
==============================================================================
```

⚠ Not Statistically Significant ($p \geq 0.05$)

These features do not show a significant effect:

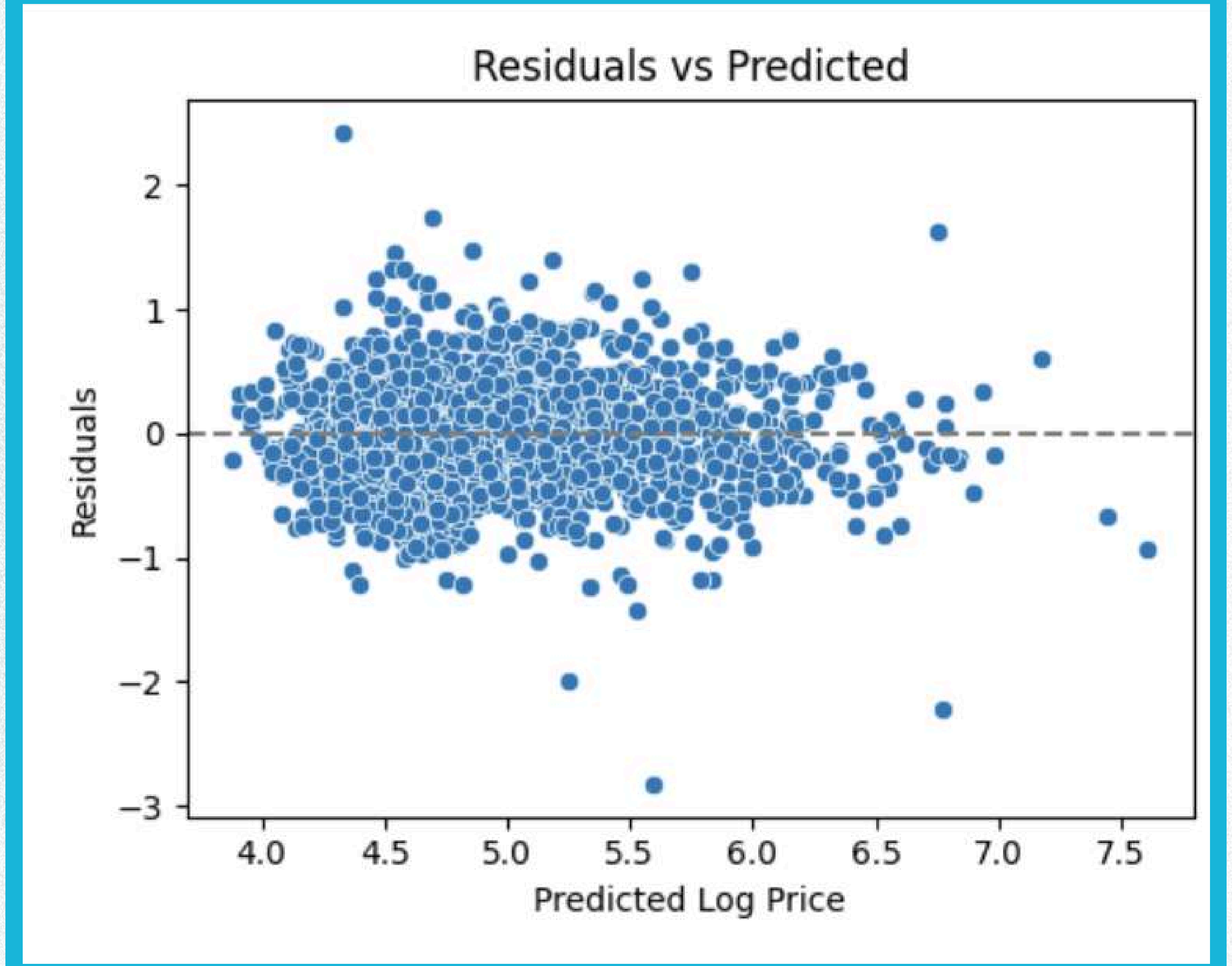
- host_response_rate (0.280)
- host_is_superhost_t (0.448)
- host_identity_verified_t (0.139)
- neighbourhood_group_cleansed_Lausanne (0.114)
- neighbourhood_group_cleansed_Lavaux-Oron (0.385)

VARIANCE INFLATION FACTOR (VIF) FOR MULTICOLLINEARITY

	Feature	VIF
9	review_scores_rating	51.442667
0	host_response_rate	37.820972
12	host_identity_verified_t	22.218171
1	host_acceptance_rate	17.250440
2	accommodates	16.539114
4	bedrooms	12.071745
3	bathrooms	9.695935
5	beds	7.077991
7	availability_365	3.814699
16	neighbourhood_group_cleansed_Lausanne	2.351070
21	neighbourhood_group_cleansed_Riviera-Pays-d'En...	1.847039
8	number_of_reviews_ltm	1.823827
22	instant_bookable_t	1.800671
11	host_is_superhost_t	1.685026
10	calculated_host_listings_count	1.543603
19	neighbourhood_group_cleansed_Nyon	1.394853
17	neighbourhood_group_cleansed_Lavaux-Oron	1.387988
15	neighbourhood_group_cleansed_Jura-Nord vaudois	1.384499
20	neighbourhood_group_cleansed_Ouest lausannois	1.303694
18	neighbourhood_group_cleansed_Morges	1.266171
6	minimum_nights	1.238041
13	neighbourhood_group_cleansed_Broye-Vully	1.188076
14	neighbourhood_group_cleansed_Gros-de-Vaud	1.092468

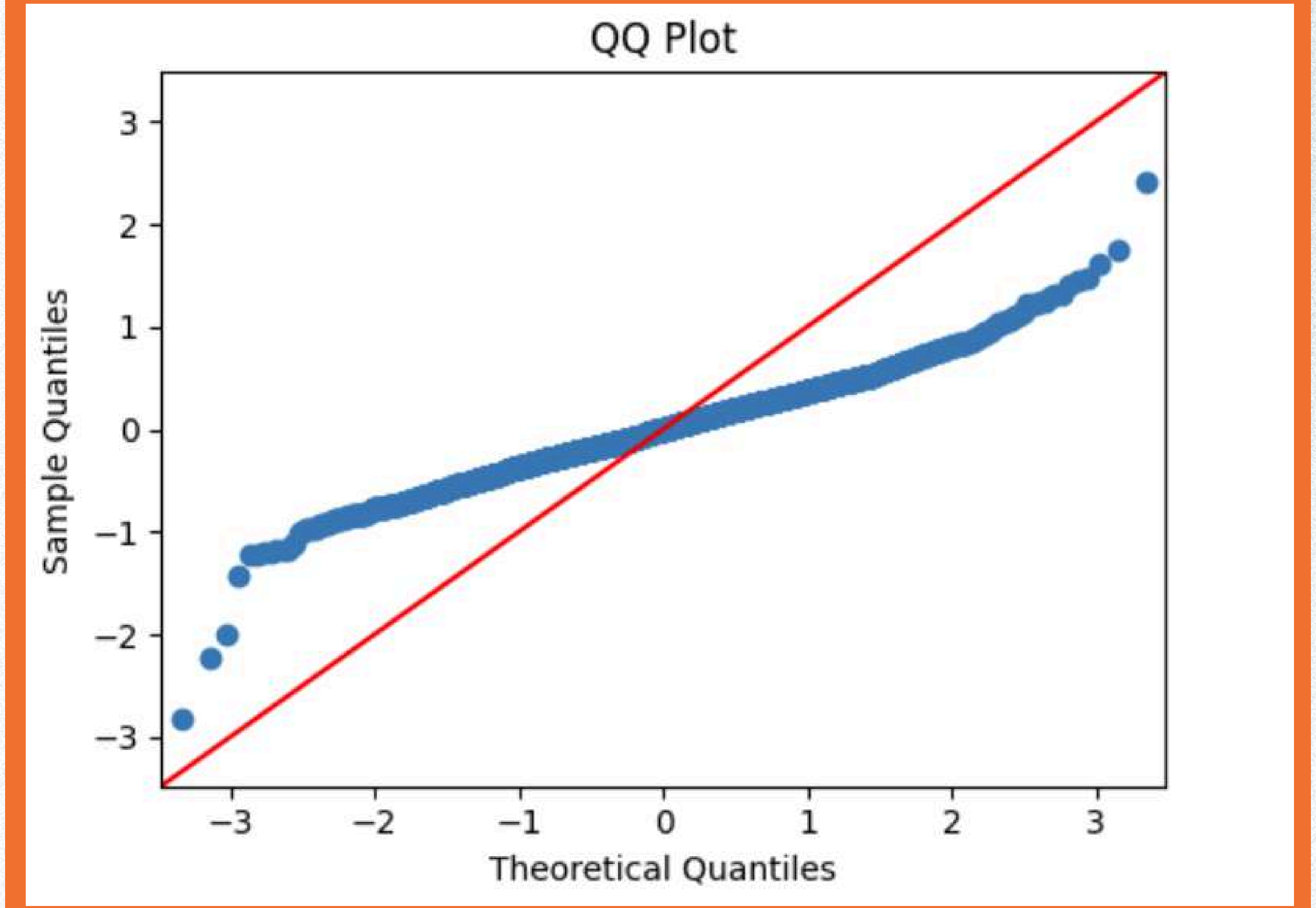
REGRESSION DIAGNOSTICS: RESIDUAL PATTERNS

Checks for linearity and homoscedasticity



- Residuals are centered around zero → supports linearity.
- Funnel shape → indicates heteroscedasticity (non-constant variance).
- Non-uniform scatter → potential violations of regression assumptions.

Tests normality of residuals by comparing the distribution of residuals with a theoretical normal distribution.



- Deviations at tails → residuals not perfectly normally distributed.
- Heavy tails suggest outliers or skewed errors.
- Normality assumption mildly violated → impacts inference accuracy.

BREUSCH-PAGAN TEST FOR HOMOSCEDASTICITY

```
Lagrange Multiplier Statistic: 109.2341
p-value: 0.0000
F-statistic: 4.9236
F-test p-value: 0.0000
```

- Higher value (109.23) suggests stronger evidence against homoscedasticity.
- P-value = 0.0000 → Rejects the null hypothesis of constant variance.
- Indicates heteroscedasticity — residual variance is not constant.
- Suggests model assumptions are violated; standard errors are unreliable.

SAMPLE PREDICTION

SAMPLE PREDICTION

```python
# Step 1: Input for new listing
input_data = pd.DataFrame([{
    'host_response_rate': 95,
    'host_acceptance_rate': 90,
    'accommodates': 4,
    'bathrooms': 2,
    'bedrooms': 2,
    'beds': 2,
    'minimum_nights': 1,
    'availability_365': 100,
    'number_of_reviews_ltm': 5,
    'review_scores_rating': 4.8,
    'review_scores_cleanliness': 4.9,
    'calculated_host_listings_count': 2,
    'host_is_superhost_t': 1,
    'host_identity_verified_t': 1,
    'neighbourhood_group_cleansed_Broye-Vully': 0,
    'neighbourhood_group_cleansed_Gros-de-Vaud': 0,
    'neighbourhood_group_cleansed_Jura-Nord vaudois': 0,
    'neighbourhood_group_cleansed_Lausanne': 0,
    'neighbourhood_group_cleansed_Lavaux-Oron': 0,
    'neighbourhood_group_cleansed_Morges': 0,
    'neighbourhood_group_cleansed_Nyon': 0,
    'neighbourhood_group_cleansed_Ouest lausannois': 1,
    "neighbourhood_group_cleansed_Riviera-Pays-d'Enhaut": 0,
    'instant_bookable_t': 1
}])

# Step 2: Align with training feature columns
X_train_columns = X_train.columns.tolist()

# Add any missing columns to input_data
missing_cols = set(X_train_columns) - set(input_data.columns)
for col in missing_cols:
    input_data[col] = 0

# Ensure correct order
input_data = input_data[X_train_columns]

# Step 3: Scale input using training scaler
input_scaled = scaler.transform(input_data)

# Step 4: Apply coefficients (from your trained model)
coefficients = np.array([
    0.0097, 0.0233, 0.3590, 0.1145, 0.1337, -0.1433, -0.0376, 0.0648,
   -0.0804, 0.0265, 0.0183, 0.0070, 0.0122, -0.0258, -0.0320, -0.0809,
   -0.0189, 0.0079, -0.0153, -0.0224, -0.0293, 0.0491, 0.0565
])
intercept = 4.8826

# Step 5: Predict and inverse log
log_price = intercept + np.dot(input_scaled, coefficients)
predicted_price = np.expm1(log_price)

# Step 6: Output
print(f"Predicted log_price: {log_price[0]:.4f}")
print(f"Predicted Price (CHF): {predicted_price[0]:,.2f}")
```

```
Predicted log_price: 5.1052
Predicted Price (CHF): 163.88
```

```
Model Performance (Train):
RMSE: 117.94
MAE : 53.68
R²  : 0.522

Model Performance (Test):
RMSE: 113.12
MAE : 58.40
R²  : 0.591
```

Classification I
KNN Classification

🔍 K-NEAREST NEIGHBORS (KNN) FOR AMENITY PREDICTION

Goal
Predict whether a listing has a Kitchen based on numerical features

Target Variable
Kitchen_present
(1 = has kitchen, 0 = no kitchen)

Predictors Used
30+ numerical columns (e.g. beds, price, availability, review scores)

Model
KNeighborsClassifier with k=7

Preprocessing Steps
- **Cleaned** amenities column → converted to list
- Created **binary** column for Kitchen presence
- **Standardized** numeric features using StandardScaler
- Split data: **80%** training / **20%** test

```
# Step 1: Clean amenities (remove brackets and split properly)
vaud_cleaned['amenities'] = vaud_cleaned['amenities'].str.replace(r'[\[\]"]', '', regex=True)
vaud_cleaned['amenities_list'] = vaud_cleaned['amenities'].str.split(',')
vaud_cleaned['amenities_list'] = vaud_cleaned['amenities_list'].apply(lambda x: [item.strip() for item in x])
```

```
vaud_cleaned['Kitchen_present'] = vaud_cleaned['amenities_list'].apply(lambda x: 1 if 'Kitchen' in x else 0)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Step 5: Train-test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=218)
```

📊 MODEL EVALUATION

PERFORMANCE METRICS

```
Confusion Matrix:
[[ 12  39]
 [ 16 542]]

Classification Report:
              precision    recall  f1-score   support

           0       0.43      0.24      0.30        51
           1       0.93      0.97      0.95       558

    accuracy                           0.91       609
   macro avg       0.68      0.60      0.63       609
weighted avg       0.89      0.91      0.90       609

Overall Accuracy: 0.91
```

	Predicted No Kitchen	Predicted Kitchen
Actual No Kitchen	12	39
Actual Kitchen	16	542

Accuracy

92%

Confusion Matrix

- True Positives (Kitchen = 1): 542
- False Positives: 39
- False Negatives: 16
- True Negatives (Kitchen = 0): 12

Class Imbalance

90% listings had a kitchen

Precision/ Recall

- Class 1 (Kitchen): Precision 93%, Recall 97%
- Class 0 (No Kitchen): Precision 43%, Recall 24%

Summary

Good accuracy, but weaker performance on minority class due to imbalance

Classification II
Naive Bayes

UNDERSTANDING NAIVE BAYES & OUR CLASSIFICATION GOAL

Goal

Predict value perception (Low, Medium, High) of a rental listing.

Target Variable

review_scores_value
(binned into 3 categories).

Predictors Used

Only numerical predictors were used (excluding all review scores to avoid leakage).

Model

Naive Bayes Classifier – a simple, fast probabilistic algorithm.

Why Naive Bayes

- Works well with categorical data
- Assumes feature independence (Naive assumption)
- Suitable for text, classification, or probability-based decisions

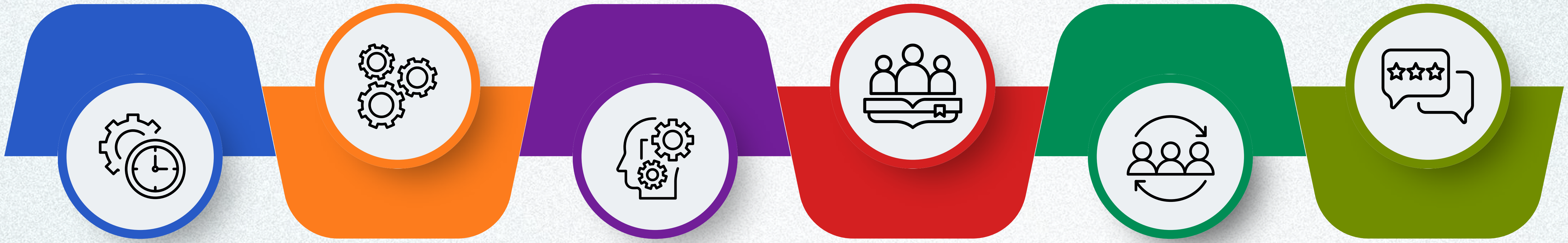
```
# Step 1: Equal Frequency Binning of review_scores_value
# Drop missing values in review_scores_value
vaud_cleaned2 = vaud_cleaned2.dropna(subset=['review_scores_value'])

# Bin into 3 categories (low, medium, high value perception)
kbins = KBinsDiscretizer(n_bins=3, encode='ordinal', strategy='quantile')
vaud_cleaned2['value_binned'] = kbins.fit_transform(vaud_cleaned2[['review_scores_value']]).astype(int)

# Step 2: Select predictors (numerical columns excluding any review_scores_ variables)
numerical_cols = vaud_cleaned2.select_dtypes(include=['float64', 'int64']).columns
predictor_cols = [col for col in numerical_cols if not col.startswith('review_scores') and col != 'value_binned']

# Step 3: Prepare X and y
X = vaud_cleaned2[predictor_cols]
y = vaud_cleaned2['value_binned']

# Step 4: Train-test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=218)

# Step 5: (Optional) Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 6: Train Naive Bayes Model
nb_model = GaussianNB()
nb_model.fit(X_train_scaled, y_train)

# Step 7: Predict
y_pred = nb_model.predict(X_test_scaled)
```

DATA PREPROCESSING & TRAINING THE MODEL

Cleaned missing data in review_scores_value.

Used KBinsDiscretizer to bin values into 3 balanced categories.

Selected 30 numerical predictors excluding all review_scores.



Scaled all predictors using StandardScaler.

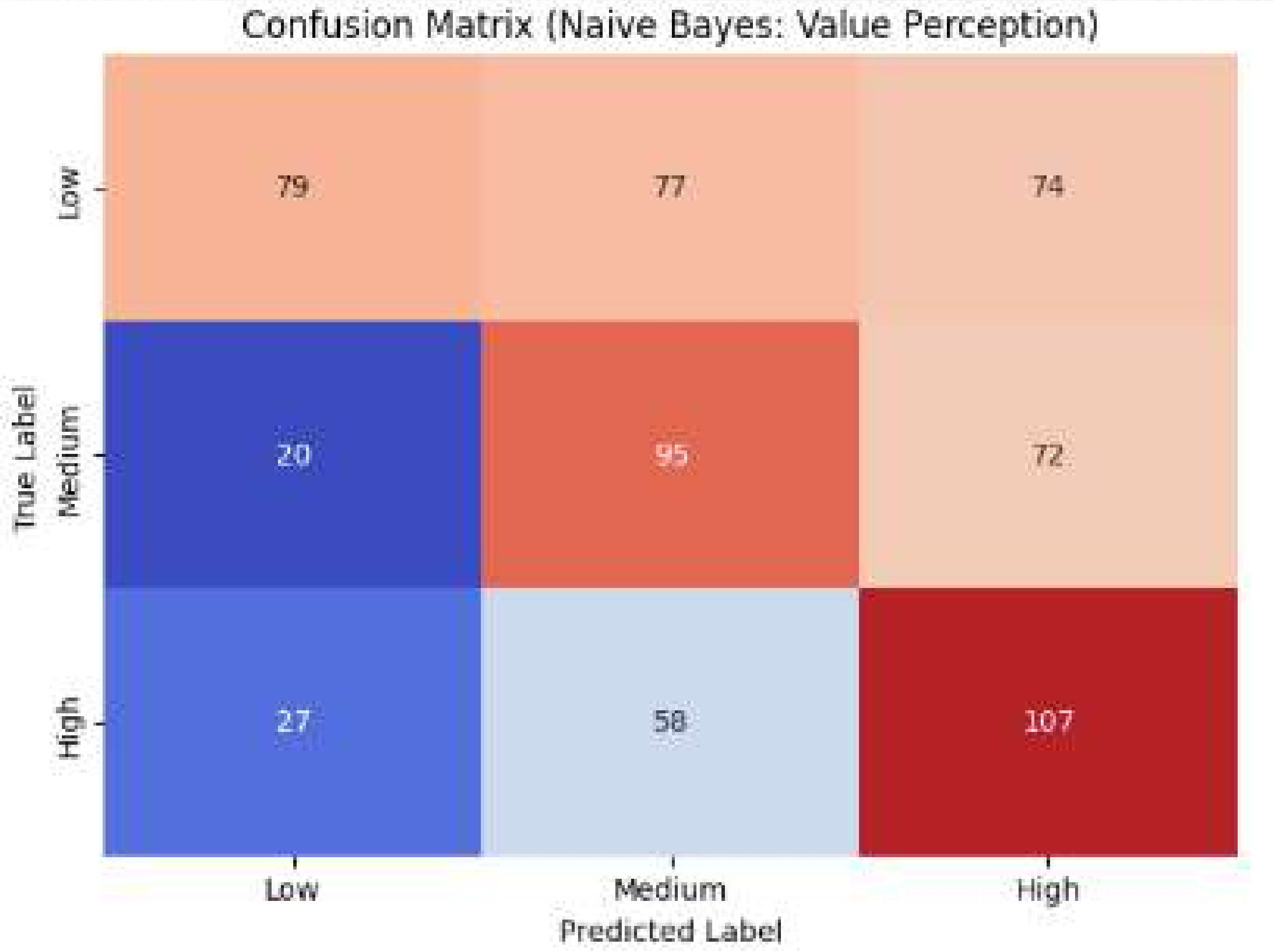
Used 80/20 train-test split.

Trained the model using Gaussian Naive Bayes.

```
# Step 3: Prepare X and y
X = vaud_cleaned2[predictor_cols]
y = vaud_cleaned2['value_binned']

# Step 4: Train-test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=218)

# Step 5: (Optional) Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Step 6: Train Naive Bayes Model
nb_model = GaussianNB()
nb_model.fit(X_train_scaled, y_train)
```

MODEL PERFORMANCE:
ACCURACY & CONFUSION MATRIX

```
Confusion Matrix:
[[ 79  77  74]
 [ 20  95  72]
 [ 27  58 107]]

Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.34      0.44       230
           1       0.41      0.51      0.46       187
           2       0.42      0.56      0.48       192

    accuracy                           0.46       609
   macro avg       0.49      0.47      0.46       609
weighted avg       0.50      0.46      0.46       609

Overall Accuracy: 0.46
```



Accuracy

46%

Predictions

Best predictions for 'High' value class, but struggles between Low & Medium.

Confusion Matrix

- High number of misclassifications across adjacent classes.
- Low recall for "Low" category (many missed).

Better?

Still better than random (33%), but not ideal.

Summary

These findings suggest the model was picking up on some patterns but had room for improvement, especially for edge cases.

TESTING WITH A FICTIONAL SCENARIO

```
# Step 1: Creating a fictional rental
fictional_rental_data = {
    'latitude': 46.6,
    'longitude': 6.5,
    'accommodates': 2,
    'bathrooms': 1.0,
    'bedrooms': 1.0,
    'beds': 1.0,
    'price': 100,
    'minimum_nights': 2,
    'maximum_nights': 30,
    'minimum_minimum_nights': 1,
    'maximum_minimum_nights': 5,
    'minimum_maximum_nights': 25,
    'maximum_maximum_nights': 50,
    'availability_30': 28,
    'availability_60': 50,
    'availability_90': 75,
    'availability_365': 300,
    'number_of_reviews': 35,
    'number_of_reviews_ltm': 12,
    'number_of_reviews_l30d': 2,
    'calculated_host_listings_count': 3,
    'calculated_host_listings_count_entire_homes': 2,
    'calculated_host_listings_count_private_rooms': 1,
    'calculated_host_listings_count_shared_rooms': 0,
    'reviews_per_month': 1.5,
    # Filling all amenities used during model building
    'BBQ_grill_present': 1,
    'Bed linens_present': 1,
    'Cooking basics_present': 1,
    'Dedicated workspace_present': 1,
    'Dishes and silverware_present': 1,
    'Essentials_present': 1,
    'Hair dryer_present': 1,
    'Hangers_present': 1,
    'Hot water_present': 1,
    'Iron_present': 1,
    'Kitchen_present': 1,
    'Microwave_present': 1,
    'Oven_present': 1,
    'Private entrance_present': 1,
    'Refrigerator_present': 1,
    'Self check-in_present': 1,
    'Shampoo_present': 1,
    'TV_present': 1,
    'Washer_present': 1,
    'Wifi_present': 1
}
```

Created a fictional rental with:

- 1 bed, 1 bath, basic amenities
- Priced at 100 CHF/night
- Located near Lake Geneva

```
# Rebuild fictional_rental based on predictor_cols to guarantee order
fictional_rental = pd.DataFrame([{col: fictional_rental_data.get(col, 0) for col in predictor_cols}])

# Step 2: Scale the fictional rental
fictional_rental_scaled = scaler.transform(fictional_rental)

# Step 3: Predict the bin
predicted_bin = nb.predict(fictional_rental_scaled)

# Step 4: Map bin numbers to labels
bin_labels = {0: 'Low Value Perception', 1: 'Medium Value Perception', 2: 'High Value Perception'}
predicted_label = bin_labels[predicted_bin[0]]

# Step 5: Print the result
print(f"The fictional rental is predicted to fall into the '{predicted_label}' bin.")
```

```
The fictional rental is predicted to fall into the 'Low Value Perception' bin.
```

Model predicted this listing as ‘Low Value Perception’

Highlights model’s sensitivity to price vs amenities

Despite good features, price likely influenced the prediction

Classification III
Decision Tree

OBJECTIVE

Predict the host response time category using decision tree.

Understand the key drivers influencing fast vs. delayed host responses.

Compare model interpretability and performance across imbalanced response classes.

Identify the best-suited model for practical deployment based on fairness and accuracy.

TREE CLASSIFIER PIPELINE

STEP 1

Dropped clearly unnecessary columns (IDs, URLs, text descriptions, etc.)

STEP 2

Removed rows with missing host_response_time

STEP 3

Defined host_response_time as the target variable

STEP 4

Dropped columns with unencodable list-type data

STEP 5

One-hot encoded categorical variables

STEP 6

Split the data into training and test sets (80/20 split)

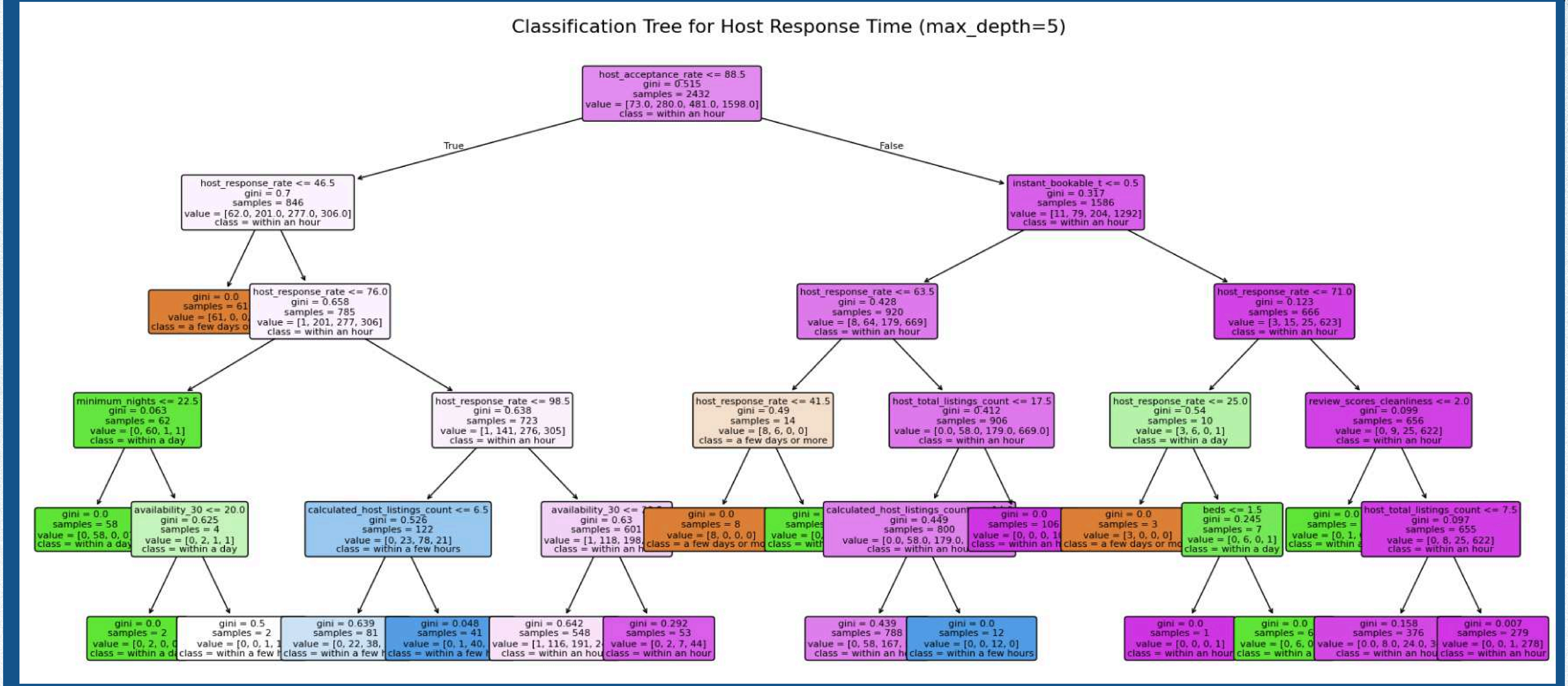
STEP 7

Performed cross-validation with GridSearchCV to find optimal tree depth

STEP 8

Trained final DecisionTreeClassifier using the best max_depth

CLASSIFICATION TREE



CLASSIFICATION TREE INTERPRETATION

- **Root split:** host_acceptance_rate is the most informative feature at the top.
- **Internal nodes:** Uses features like host_response_rate, instant_bookable, availability_30, and minimum_nights.
- **Leaf nodes:** Predict the class label based on majority of samples in that path.
- **Gini score:** Indicates node purity; lower is better.

INSIGHTS

Faster responses are linked to:

- High host_response_rate
- High host_acceptance_rate
- Low minimum_nights
- Few listings per host
- High availability (availability_30)
- Allowing instant booking
- Good cleanliness scores.

Slower responses are linked to:

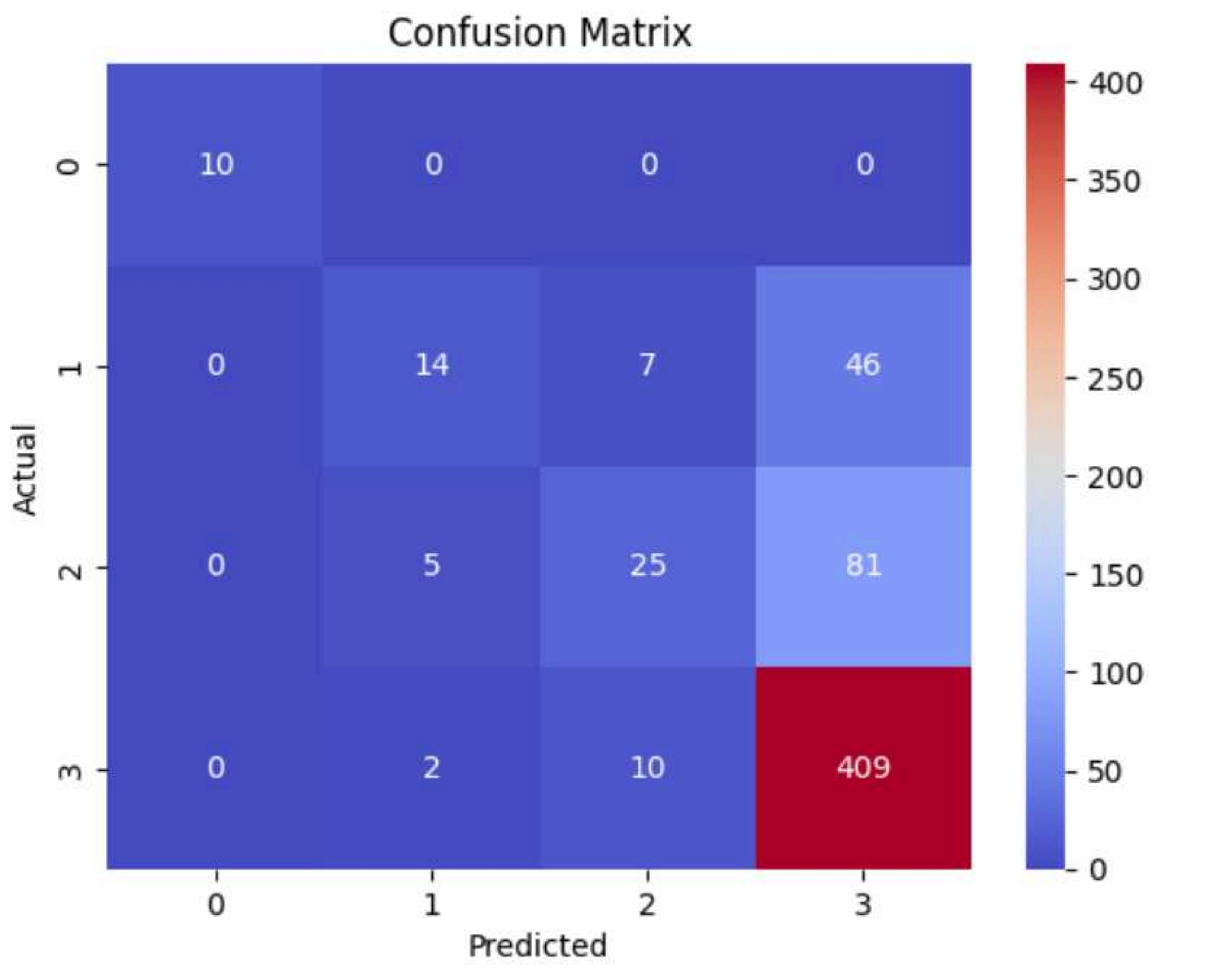
- Low acceptance or response rates
- Restrictive booking conditions (e.g., no instant booking)
- Many listings or low availability

CLASSIFICATION TREE MODEL EVALUATION

```
Classification Tree Report:
                    precision    recall  f1-score   support

a few days or more       1.00      1.00      1.00        10
      within a day       0.67      0.21      0.32        67
within a few hours       0.60      0.23      0.33       111
    within an hour       0.76      0.97      0.85       421

          accuracy                           0.75       609
         macro avg       0.76      0.60      0.62       609
      weighted avg       0.73      0.75      0.70       609
```

host_response_time	count
within an hour	2019
within a few hours	592
within a day	347
a few days or more	83

```
Tree Accuracy: 0.7520525451559934
```



The model is biased toward predicting "within an hour", the most common class.

XGBOOST
CLASSIFICATION PIPELINE

STEP 1

Dropped unnecessary columns irrelevant to prediction.

STEP 2

Removed rows with missing target values (host_response_time

STEP 3

Encoded the target variable using LabelEncoder (string → integer).

STEP 4

Dropped columns containing unencodable list objects.

STEP 5

Applied one-hot encoding to categorical features.

STEP 6

Split the data into training and test sets.

STEP 7

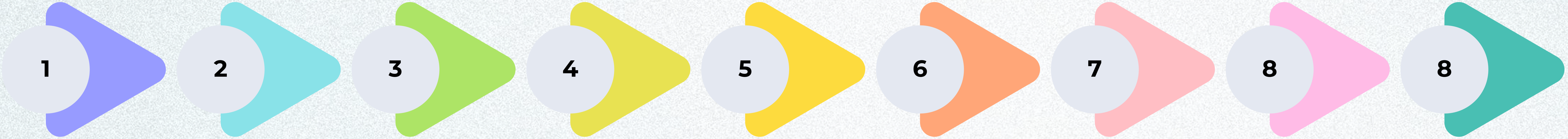
Manually calculated the class weights and passed them to the training sample

STEP 8

GridSearchCV was used for the hyperparameter tuning of the XGBClassifier.

STEP 9

Fit the best model, made predictions, and evaluated performance.



XGB MODEL EVALUATION

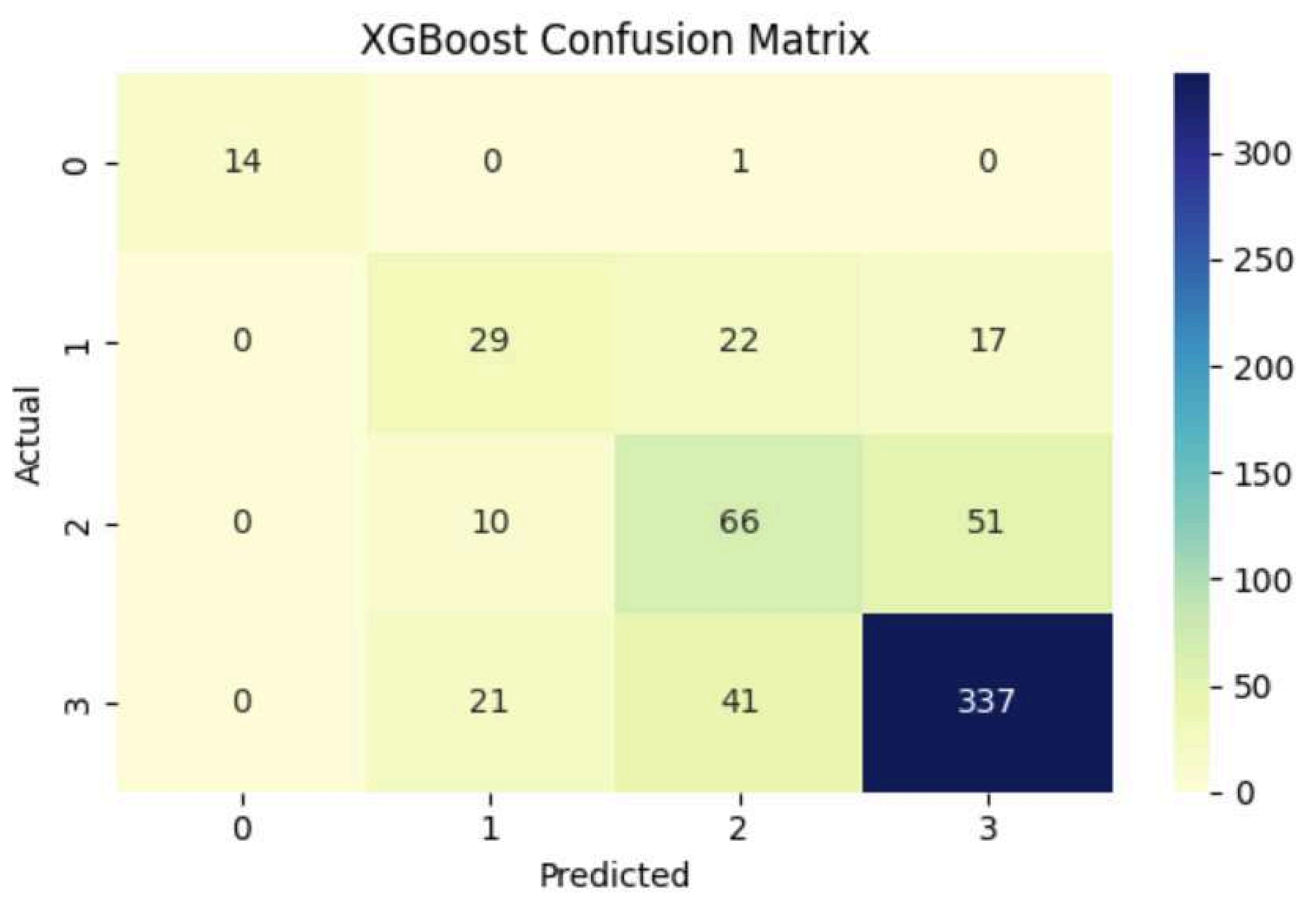
```
Fitting 5 folds for each of 32 candidates, totalling 160 fits
Best Xgb Parameters: {'colsample_bytree': 0.8, 'learning_rate': 0.1, 'max_depth': 6, 'n_estimators': 200, 'subsample': 1.0}
```

Classification Xgb Report:

	precision	recall	f1-score	support
a few days or more	1.00	0.93	0.97	15
within a day	0.48	0.43	0.45	68
within a few hours	0.51	0.52	0.51	127
within an hour	0.83	0.84	0.84	399
accuracy			0.73	609
macro avg	0.71	0.68	0.69	609
weighted avg	0.73	0.73	0.73	609

```
Xgb Accuracy Score: 0.7323481116584565
```

XGBoost performs consistently across classes



MODEL COMPARISON

CLASSIFICATION TREE

Classification Tree Report:

	precision	recall	f1-score	support
a few days or more	1.00	1.00	1.00	10
within a day	0.67	0.21	0.32	67
within a few hours	0.60	0.23	0.33	111
within an hour	0.76	0.97	0.85	421
accuracy			0.75	609
macro avg	0.76	0.60	0.62	609
weighted avg	0.73	0.75	0.70	609

Tree Accuracy: 0.7520525451559934

The model is biased toward predicting "within an hour", the most common class.

XG BOOST

Classification Xgb Report:

	precision	recall	f1-score	support
a few days or more	1.00	0.93	0.97	15
within a day	0.48	0.43	0.45	68
within a few hours	0.51	0.52	0.51	127
within an hour	0.83	0.84	0.84	399
accuracy			0.73	609
macro avg	0.71	0.68	0.69	609
weighted avg	0.73	0.73	0.73	609

Xgb Accuracy Score: 0.7323481116584565

XGBoost performs consistently across classes

KEY TAKEAWAYS

CLASSIFICATION TREE

- The classification tree model effectively reveals how decision rules involving review activity, listing availability, and booking behavior influence host responsiveness.
- Its intuitive structure makes it highly interpretable, providing quick insights into what drives fast versus delayed responses from hosts.

XGBOOST

- For a more imbalanced classification problem like this, XGBoost proves to be a stronger candidate.
- By incorporating class weights and hyperparameter tuning, it improves generalization and ensures fairer predictions across all response time categories.

TRADEOFF

- While the tree model is better for transparency, XGBoost offers a performance boost and is better suited for handling class imbalance and complex patterns in the data.

CLUSTERING
MODEL

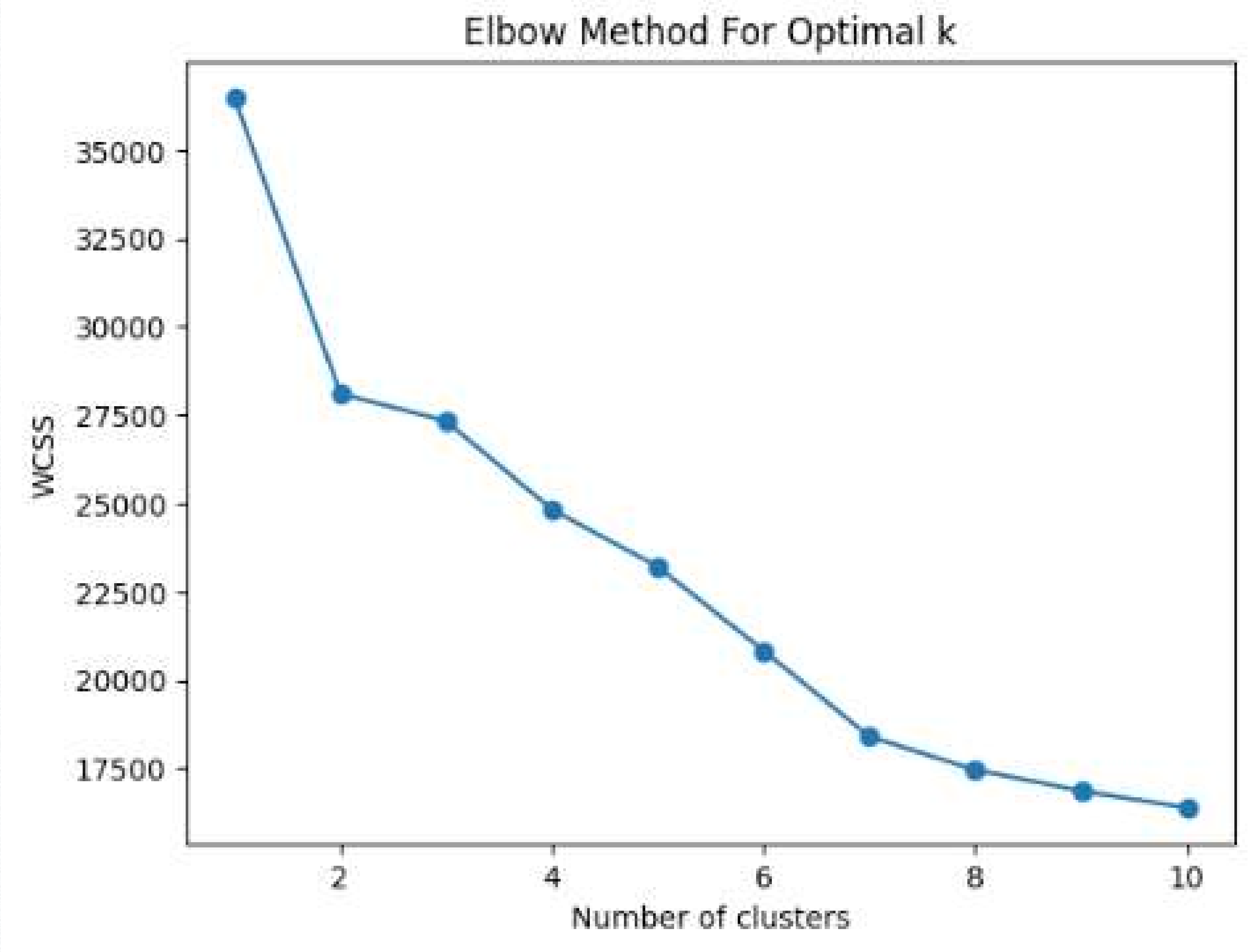
CLUSTERING AIRBNB RENTALS: FEATURE SELECTION, SCALING & ELBOW

Objective: Group rental listings into clusters based on similarity.

Engineered new feature: price_per_person

Selected numerical features: price, availability, beds, reviews, etc.

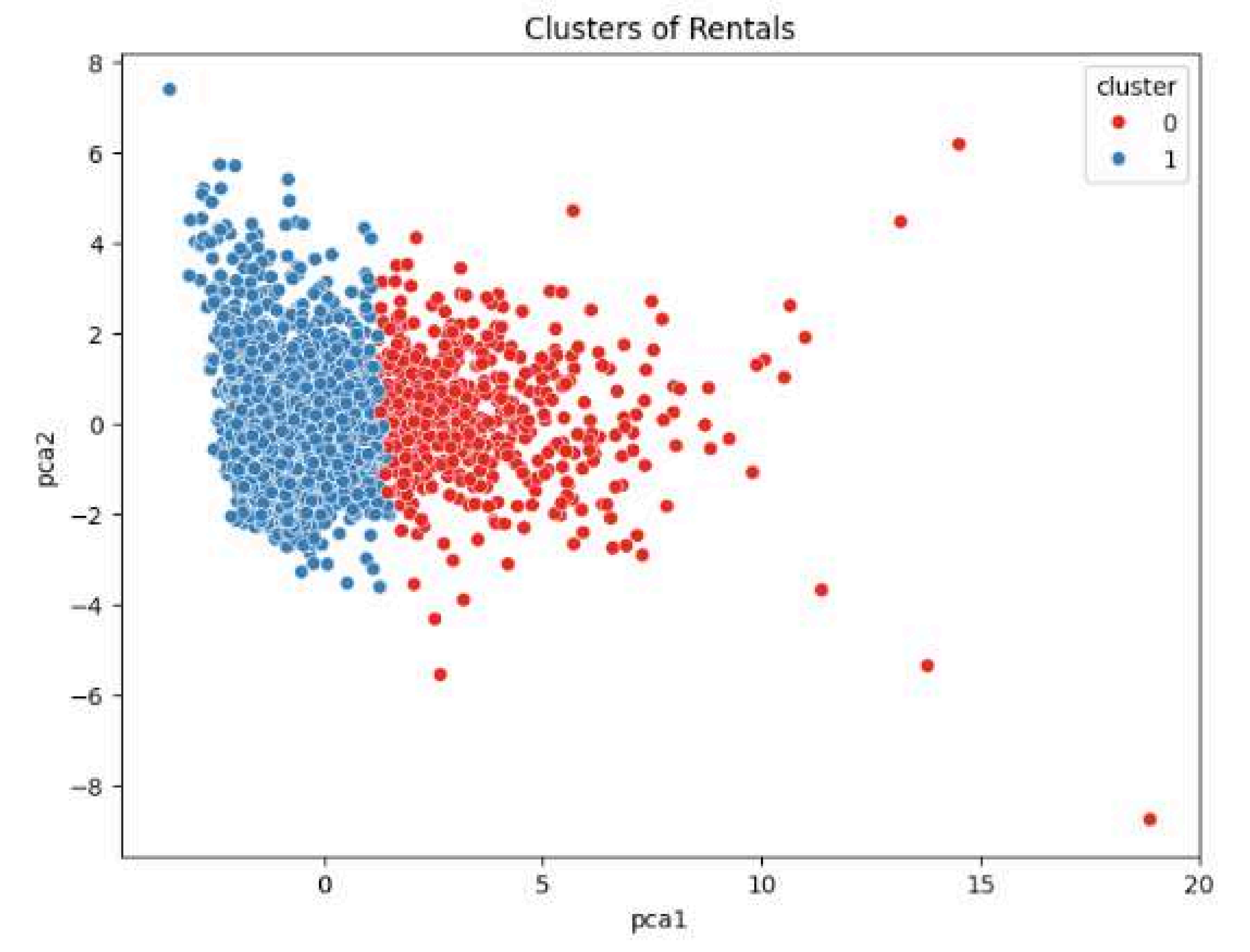
Applied StandardScaler for uniform contribution of variables.



PCA VISUALIZATION

Why PCA?

Used Principal Component Analysis to reduce the data to two dimensions for visualization.



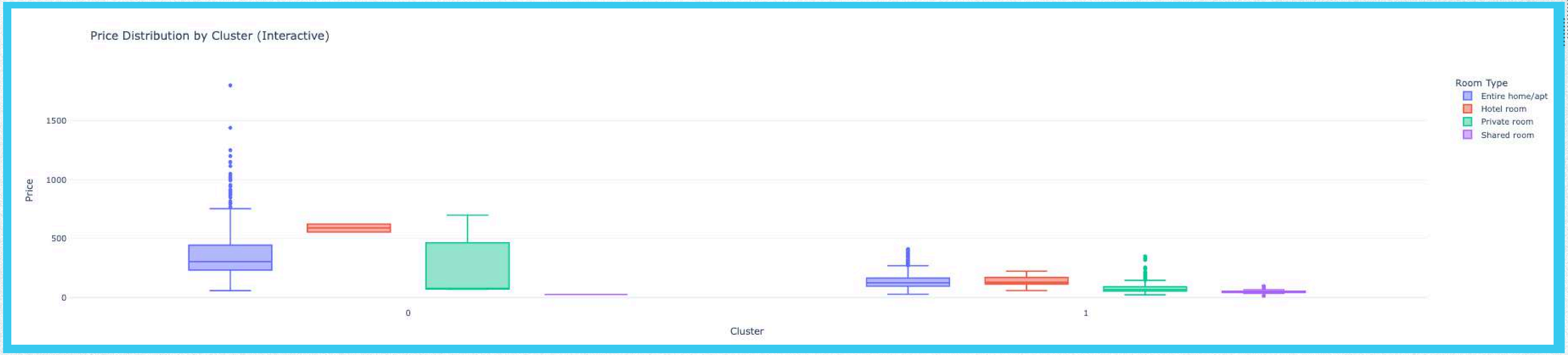
Cluster - 0

Diverse Premium Rentals — high price variation, upscale.

Cluster - 1

Standard Economy Rentals — consistent, affordable listings.

BOXPLOT — PRICE DISTRIBUTION BY CLUSTER



Cluster 0 shows significantly higher prices and greater variability, indicating premium or luxury listings.

Cluster 1 shows lower and more consistent pricing, typical of budget-friendly or economy rentals.

Outliers in Cluster 0 reach beyond $1500, especially for "Entire home/apt".

Room types influence price spread—Hotel and Entire homes skew higher in Cluster 0.

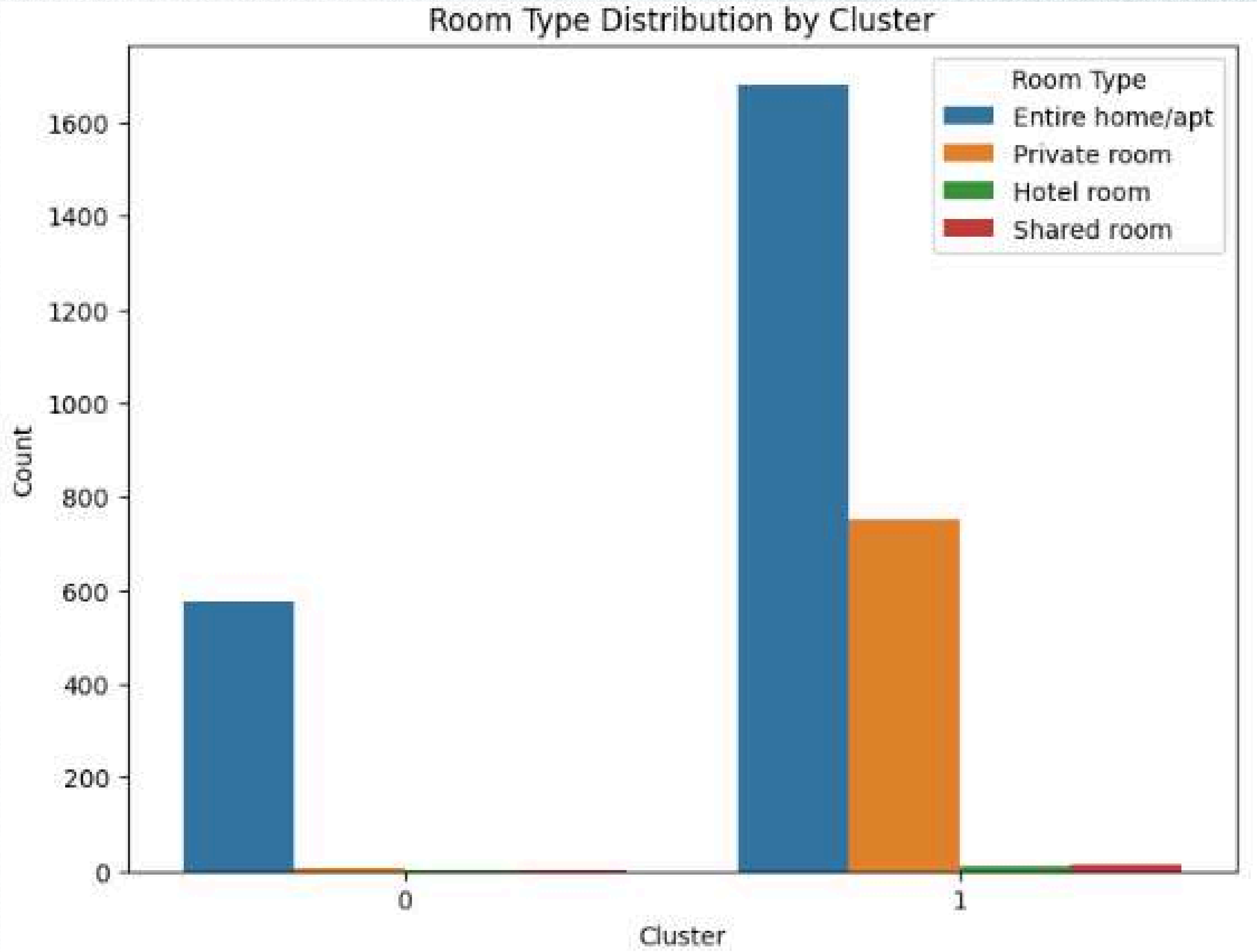
BAR CHART — ROOM TYPE DISTRIBUTION BY CLUSTER

Cluster 0 is dominated by **Entire home/apt,** aligning with its premium nature.

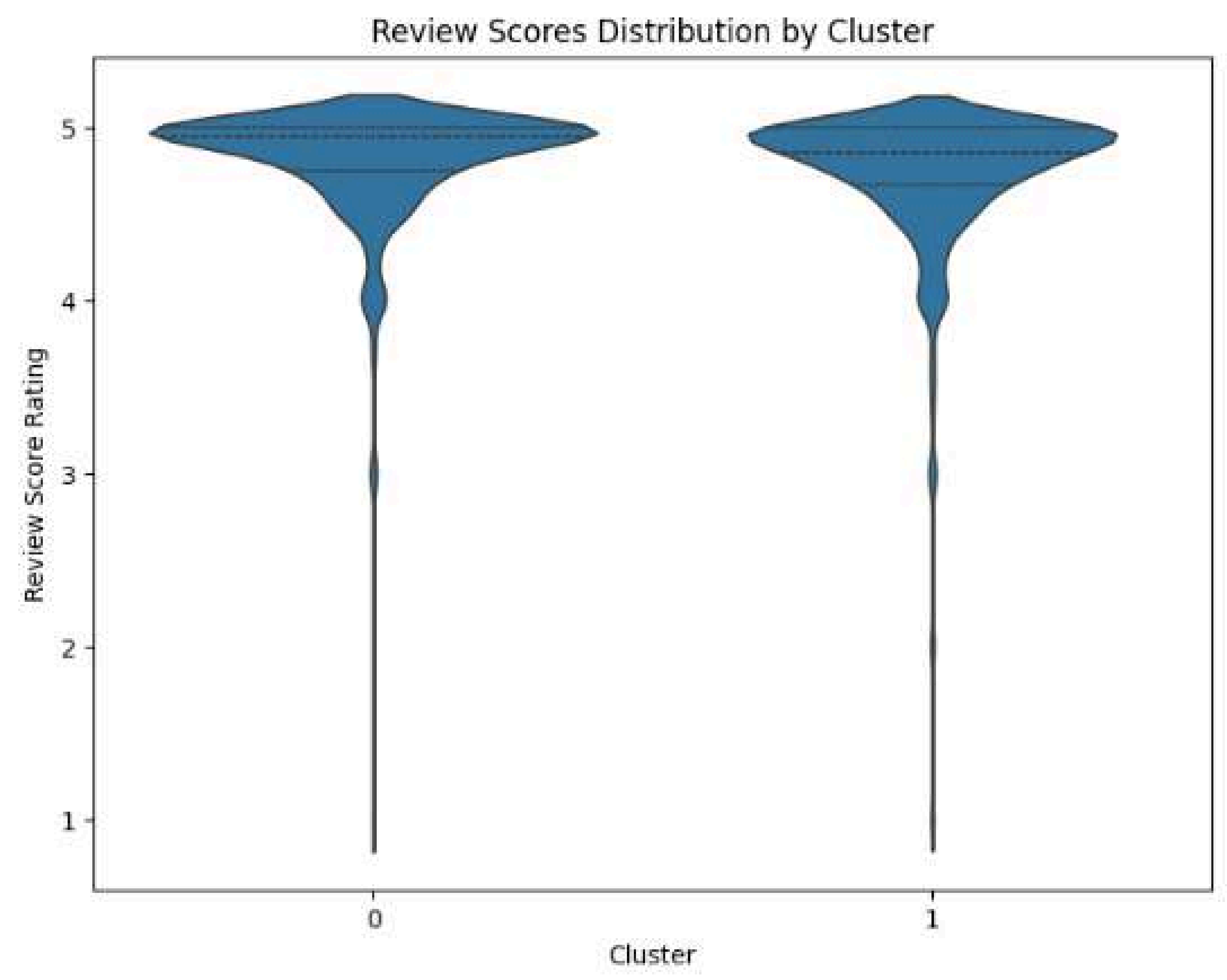
Cluster 1 includes a **broader mix** of room types, especially Private and Shared rooms.

Minimal presence of Hotel rooms in Cluster 1 reinforces its **economy nature.**

Room type distribution **supports pricing segmentation** across clusters.



VIOLIN PLOT — REVIEW SCORE DISTRIBUTION BY CLUSTER



Both clusters have **high average review scores (4.8–5.0)**, suggesting strong guest satisfaction.

Slightly more **variation below 4.5 in Cluster 1** indicates occasional guest dissatisfaction.

Despite pricing and room type differences, **both clusters maintain high quality.**

Confirms that economy listings **don't compromise on guest experience.**

PROJECT SUMMARY

Linear Regression

Predicted price based on features like beds, baths, and availability. Helps hosts **optimize revenue.**

KNN

Classified amenities (e.g., Wi-Fi, parking). Useful for **benchmarking listings.**

Naive Bayes

Predicted review score category. **Supports customer satisfaction insights.**

Classification Tree

Modeled host response time using review and booking activity. **Helps improve host performance**

Clustering

Grouped listings by price, location, availability. **Enables targeted marketing strategies.**

Overall

A multi-model approach delivering data-driven insights for pricing, operations, and customer engagement.


Thank You!