1. Exploratory Data Analysis (EDA):

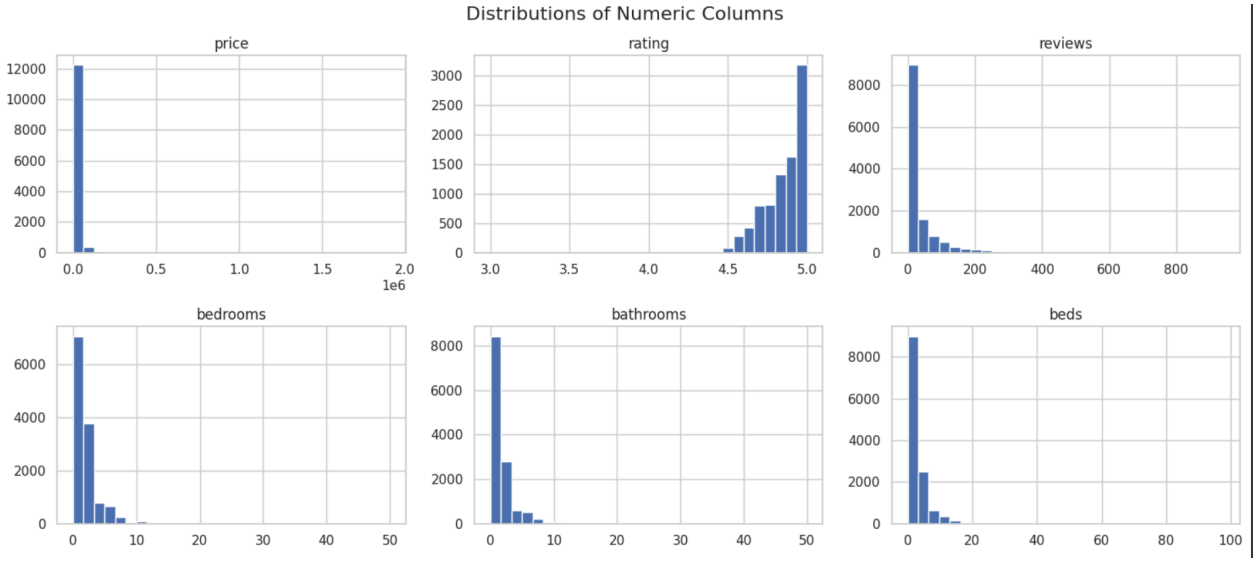
1.1 Understand the dataset,

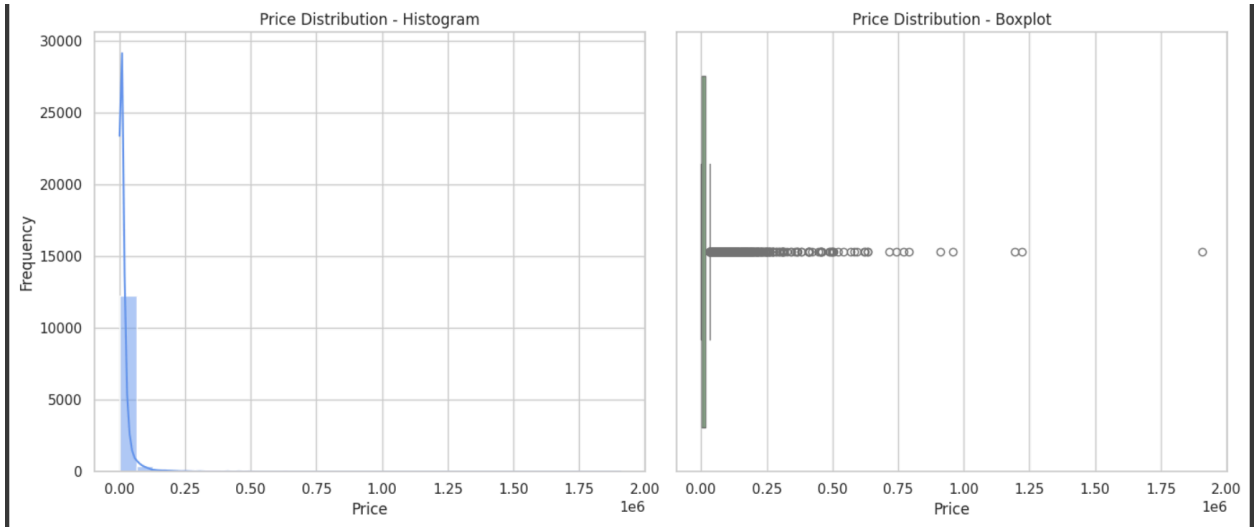
- The dataset has **12,805 rows** and **23 columns**.
- Data types include:
    a. Numerical: int64, float64 (e.g., price, beds, host_id)
    b. Categorical/Text: object (e.g., name, address, rating, checkin)
- Properties vary in capacity from 2 to 4 guests and include features like mountain view, WiFi, and kitchen.
- Rating includes both numeric values like 4.71 and text like “New”.
- Only 3 columns have missing values:
    a. host_name
    b. checkin
    c. checkout

1.2 Check Data Distribution,

The distribution analysis of key numerical features in the dataset reveals significant skewness and potential outliers across multiple columns.
The target variable, price, is highly right-skewed, with the majority of listings priced at lower values while a few outliers reach extremely high prices, even into the millions. This indicates the presence of luxury or possibly erroneous entries that may require further cleaning or transformation.

- Prices: Mostly affordable, but a few extreme outliers (up to millions).
- Ratings: Mostly 4.5-5.0—are low ratings being removed?
- Reviews: Most listings have few reviews, but some have hundreds.
- Bedroom/Bath/Beds: All are right-skewed — most listings offer 1-3 of each.





1.3 Correlation Analysis

- Strongest Correlations:

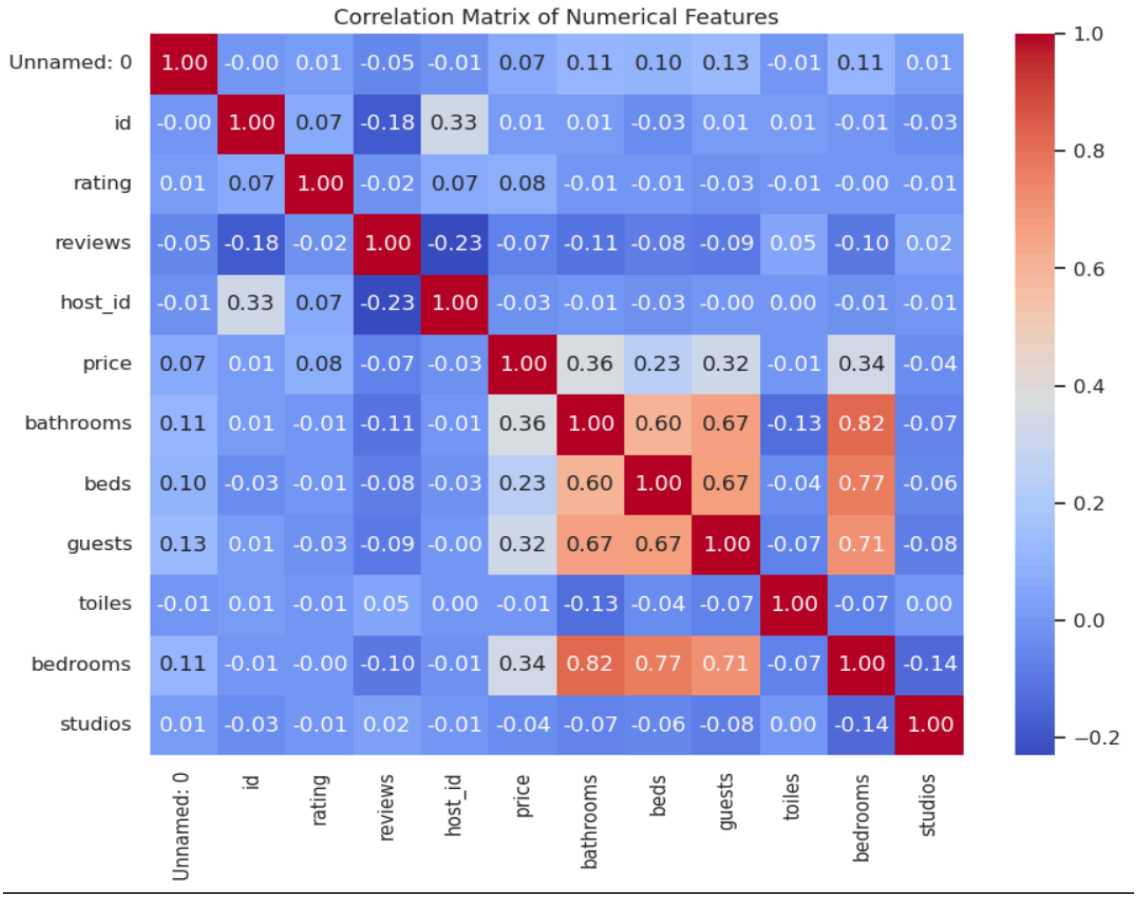
  Property Features,

  a. Bedrooms and Bathrooms (0.82) – Larger properties tend to have more of both.
  b. Bedrooms and Beds (0.77) – More bedrooms usually mean more beds.
  c. Bathrooms and Beds (0.60) – Expected, as bigger accommodations have more facilities.

  Price Features,

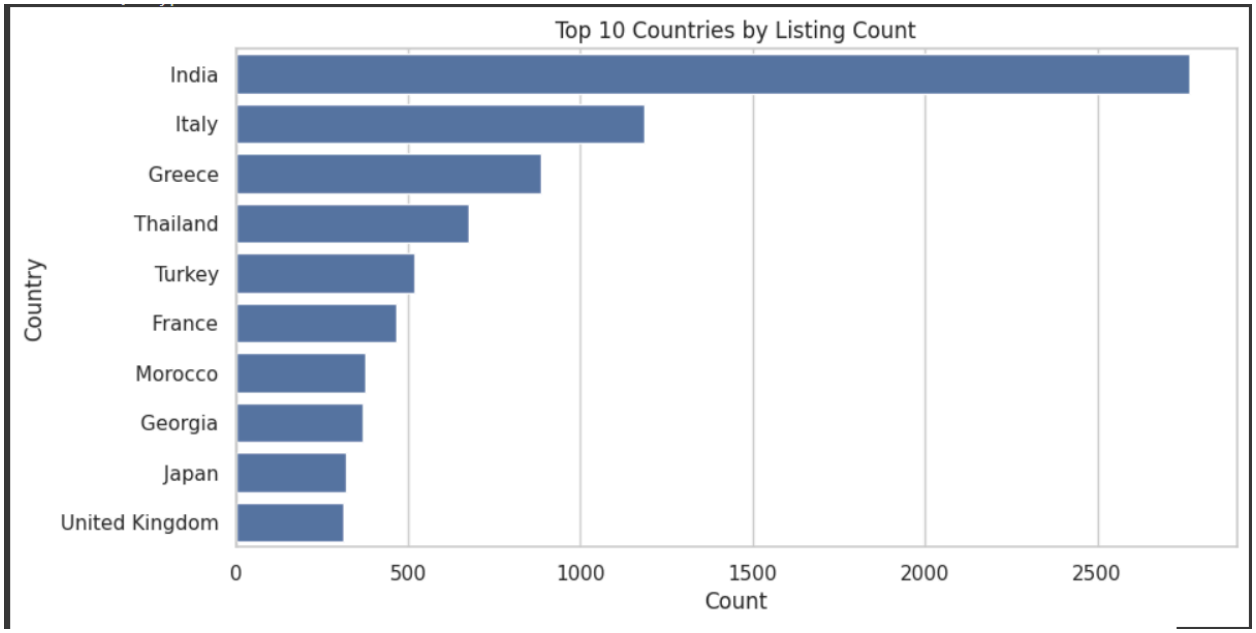
a. Price and Bathrooms (0.36) – More bathrooms slightly increase price.
b. Price and Bedrooms (0.34) – More bedrooms correlate with higher prices.
c. Price and Beds (0.23) – Weak but positive link between beds and price.

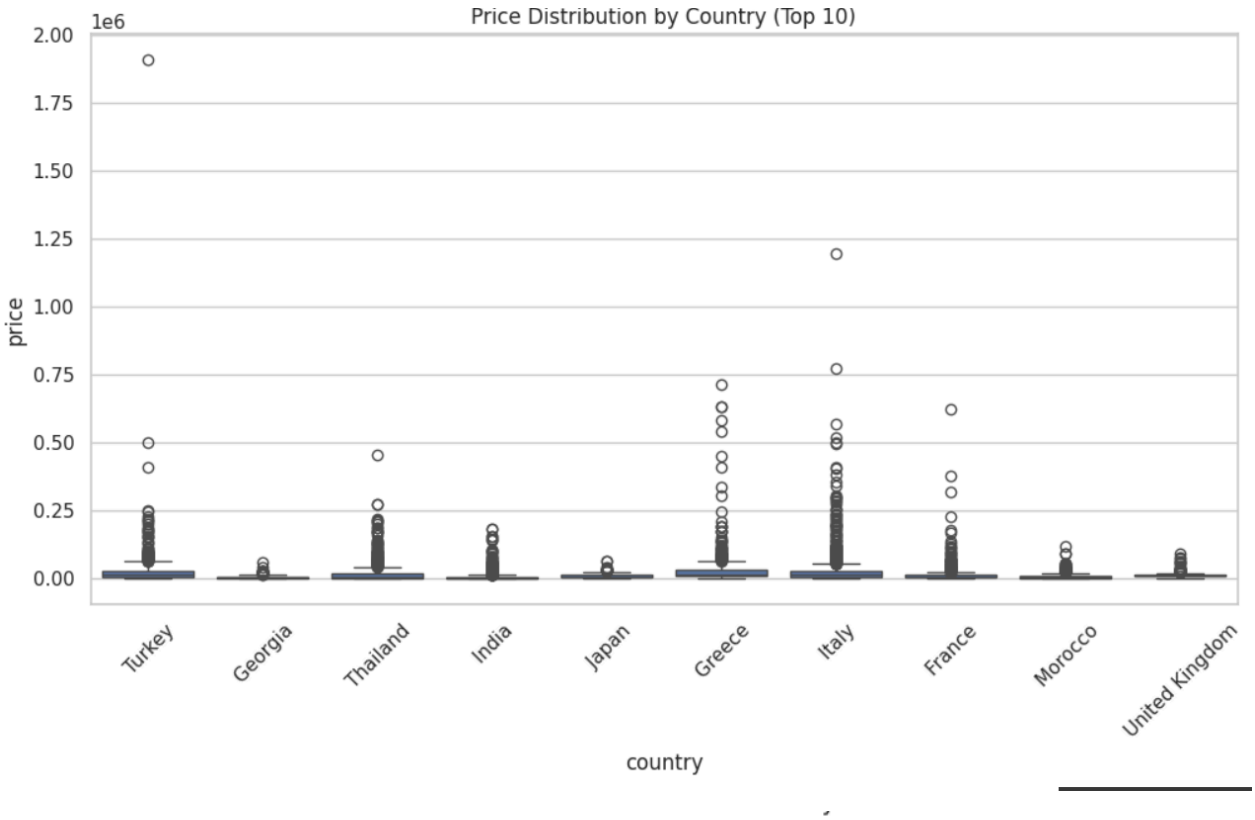
- Weak Correlation:
  a. Rating vs. Price (0.08) – Higher ratings don't strongly predict higher prices.
  b. Reviews vs. Price (-0.07) – More reviews don't mean higher prices (may indicate older/more affordable listings).
  c. Host ID vs. Price (-0.03) – Host identity has almost no pricing impact.

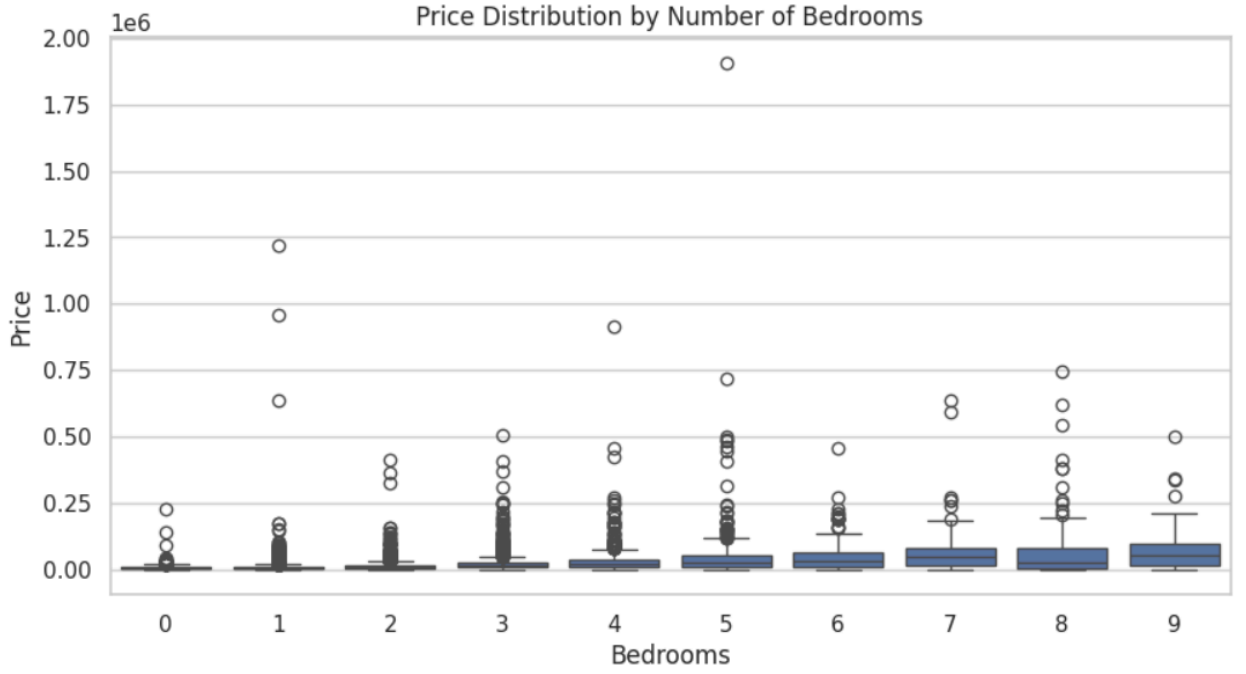


1.4 Analyze Categorical Feature

The categorical feature analysis reveals evident price tendencies fueled by property size, host behavior, and location. India and Thailand are cheaper markets with lower median prices, while Georgia and Turkey are premium markets with higher median prices, which is in line with their high volumes of listings. Hosts like Maria and Onda are top performers with long property listings, which points to potential professional hosting routines that require closer examination of their pricing behavior. Bedroom size has a dramatic impact on pricing, with the bigger the facility, the more it costs, though high-end 1-2 bedroom exceptions indicate niche market segments. Such results indicate tailored geographic pricing strategies, dynamic bedroom-based pricing segments, and additional host-level yield optimization. In order to better act, future work must standardize data labels and analyze outlier drivers (e.g., high-end pricing of Turkey). In general, results provide an actionable foundation for competitive pricing and inventory positioning.







2. Data Cleaning:

2.1 Handle Missing Values:

To ensure data integrity and prepare the dataset for further analysis, missing values were identified and addressed accordingly. Some missing values in the host_name column were deemed important for host information; hence, rows with missing host_name values were eliminated altogether. In checkin and checkout columns, due to categorical nature and additional missing values, the mode (most frequent value) was used to replace missing entries.

2.2 Outlier Detection and Removal:
Outlier removal was conducted on the target variable price to improve data quality and prevent biased modeling results. The Interquartile Range (IQR) method was applied, which identifies values that are significantly below the first quartile (Q1) or significantly above the third quartile (Q3). Specifically, any price values below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR were identified as outliers and subsequently removed from the dataset. This method gives a more robust statistical estimation of the pricing data, removing the impact of outliers while training the model and improving general model performance. A histogram was also plotted after removal in order to visually confirm the normalization of the distribution of prices.
2.3 Convert Categorical Feature:
The ‘country’ column was transformed using one-hot encoding. This technique converts categorical values into binary columns, allowing models to interpret non-numeric data. The drop_first=True parameter was used to avoid multicollinearity by dropping one of the resulting dummy variables.
2.4 Feature Engineering: To make the model more predictive, new features were created from the provided data. A new feature price_per_rating was initially constructed by dividing the listing price by its rating (plus a constant to avoid division by zero). This feature represents the relative cost of a listing in terms of how good it is, providing more fine-grained value. Furthermore, binary features checkin_flex and checkout_flex were incorporated to capture check-in and check-out flexibility. These were created by searching for whether the word "flex" or related terms existed in the corresponding text fields. Surprisingly, when these features were included, the performance of the model greatly improved, demonstrating the advantages of good feature engineering in predictive modeling.

2.5 Train-Test Split and Feature Scaling:

The dataset was divided into training and testing subsets using an 80-20 split. This ensures that model performance can be fairly evaluated on unseen data, preventing overfitting. After that, feature scaling was applied using StandardScaler, but only fit on the training set to avoid data leakage. The same scaling transformation was then applied to the test set to maintain consistency in feature distribution. To ensure no invalid values remained before model training, the scaled training data was converted back to a DataFrame for easier inspection and handling. A check for any remaining missing values in X_train revealed a small number of NaN entries, which were then imputed using the mean value of each column.

3. Model Creation:

To begin the predictive modeling task, an initial model was created using a feedforward neural network implemented with TensorFlow and Keras. The aim was to establish a baseline model to predict property prices using cleaned and scaled feature data.

3.1 Model Architecture:

The architecture of the neural network included the following layers:

- An input layer with 128 neurons and ReLU activation, matching the number of input features.
- A Dropout layer with a 30% drop rate to prevent overfitting.
- A hidden layer with 64 neurons and ReLU activation, followed by another Dropout layer with a 20% drop rate
- An additional hidden layer with 32 neurons and ReLU activation.
- An output layer with a single neuron for regression prediction

The model was compiled using the Adam optimizer and Mean Squared Error (MSE) as the loss function, with Mean Absolute Error (MAE) tracked as an evaluation metric. The model was trained for a maximum of 100 epochs, with

early stopping (patience = 10) applied to reduce overfitting by restoring the best weights based on validation loss.

3.2 Performance:

After training, the neural network achieved the following results,

- Training Loss (MSE): 36,223,528
- Training MAE: 4,748.13
- Test Set MAE: 4,734.37

These results, particularly the high MAE, were predictive of the model failing to generalize to new data. The large amount of error underlined the importance of employing a more solid modeling methodology.

4. Accuracy Improvement:

4.1 Cross Validation:

To improve the predictive performance and ensure generalization on unseen data, 5-fold cross-validation was applied to the initial neural network model. This strategy helped evaluate the model's performance across multiple train-test splits and reduce the risk of overfitting to a single validation set.

Approach-

- The log-transformed target (log1p(price)) was used to stabilize variance and reduce skewness.
- The same feedforward neural network architecture from the initial model was used.
- After training, predictions on the validation set were reverse-transformed using exmp1() to obtain results in the original price scale.
- Mean Absolute Error (MAE) was calculated between the predicted and actual prices for each fold.

Performance-

- Fold 1–5 MAE (average): 4,738.24

This validated that the model's poor generalization was consistent across different data splits. While cross-validation improved the reliability of performance estimation, the error remained significantly high.

4.2 Random Forest Regression:

Following the neural network and cross-validation approach, a Random Forest Regressor was employed as a non-linear ensemble model to improve the model's performance and reduce the mean absolute error (MAE).

Approach-

A 5-fold cross-validation framework was maintained for consistency and robust performance evaluation. In each fold:

- The target variable (price) was log-transformed using np.log1p to reduce skewness and improve model learning.
- The training set was used to fit a Random Forest Regressor with 100 decision trees (n_estimators=100) and a fixed random_state for reproducibility.
- Mean Absolute Error (MAE) was computed on the original scale for each fold.

Performance-

- Fold-wise MAE Average: 4,188.75

This result indicated a notable reduction in error (~11.6% improvement) compared to the neural network with cross-validation (MAE: 4,738.24). The Random Forest model's ability to handle non-linearities and interactions between features contributed to this improvement.

4.3 XGBoost Regression:

After observing improvements with Random Forest, the next step involved using XGBoost (Extreme Gradient Boosting)—a powerful gradient boosting framework known for its speed and performance in structured data tasks.

Approach-

Using the same 5-fold cross-validation strategy and log-transformed target variable (log1p(price)), an XGBRegressor was trained with default parameters:

- For each fold, The model was trained on training data and evaluated on the validation set.
- Predictions were reverse-transformed using np.expm1.
- MAE was calculated in the original price scale.

Performance-

- Fold-wise MAE Average: 4046

The results showed a modest improvement over Random Forest, confirming that boosting was more effective in capturing complex patterns in the data.

4.4 XGBoost Tuning:

To further refine performance, RandomizedSearchCV was used to tune hyperparameters for the XGBoost model. The tuning was performed using 30 combinations across a wide parameter space with 5-fold cross-validation. The search targeted minimizing MAE (using a custom scorer with inverse log transform).

Result-

- Mean MAE (5-fold CV): ~4006

4.5 Feature Engineering:

After completing model selection and hyperparameter tuning, feature engineering method was used to enhance the quality of input data and extract more

meaningful patterns. This step was critical in drastically improving the model's accuracy.

Result-

- Mean MAE (5-fold CV with feature engineering): 208.72.
- Test Set MAE: 192.70

After incorporating the two features and retraining the tuned XGBoost model, the performance improved significantly, reducing the Mean Absolute Error (MAE) from ~4006 to ~208.72. This spectacular improvement indicates how well-crafted features can outperform even heavy model tuning by itself.

5. Conclusion:

This project demonstrates a systematic process of building and fine-tuning a predictive model to estimate listing prices. Starting with a baseline neural network model, incremental refinement was obtained through diligent evaluation methods such as cross-validation, progressing further to more effective ensemble methods such as Random Forest and XGBoost, to careful hyperparameter tuning. Final and most powerful improvement came about through strategic feature engineering, reducing the model prediction error by several orders of magnitude.

Step	Model	Description	Mean MAE
1. Baseline	ANN	Initial model using a simple neural net	~4738
2.Improvement	Random Forest	Non-linear ensemble method with decision trees	~4188

3. Improvement	XGBoost(Default)	Gradient boosting with default settings	~4046
4. Improvement	XGBoost(Tuning)	Hyperparameter-tuned XGBoost model	~4006
5. Improvement	XGBoost (Tuned + FE)	Added price_per_rating, checkin_flex, checkout_flex	~208.72

The mean absolute error (MAE) was brought down from approximately 4738 to just 208.72, marking a substantial performance improvement. This highlights the importance of not only model selection and tuning but also domain-informed feature creation in achieving high predictive accuracy.