1. Load Libraries and Data

- **What**: Imported libraries (pandas, sklearn, xgboost) and loaded airbnb.csv (12,805 rows, 23 columns).
- **Why**: Libraries help process data and build models. The dataset has Airbnb details like price, rating, and beds.
- **How it helps**: We can analyze features to predict prices.

2. Clean Data

- **What**: Checked data with df.info(). Converted reviews and rating to numbers. Removed rows with missing price, rating, etc. Replaced "New" with Average rating in 'rating' column.
- **Why**: Models need numeric data and no missing values to work properly.
- **How it helps**: Clean data prevents errors and ensures accurate predictions.

3. Feature Engineering

- **What**: Created amenities_count by counting amenities. Log-transformed price to log_price. Selected features: rating, reviews, amenities_count, country_encoded, bathrooms, beds, guests, toilets, bedrooms, studios.
- **Why**: amenities_count simplifies text data. Log-transformation reduces price skewness. Selected features are relevant to price.
- **How it helps**: Better features improve model performance.

4. Preprocess Data

- **What**: Filled missing feature values with 0. Replaced infinite values with 0. Scaled features with StandardScaler.
- **Why**: Models can't handle missing or infinite values. Scaling ensures all features contribute equally.
- **How it helps**: Preprocessed data is ready for modeling.

5. Cross-Validation with XGBoost

- **What**: Used 5-fold cross-validation (KFold). Built XGBoost model (200 trees, learning rate 0.1, max depth 6). Made predictions, converted back to normal prices, and calculated MAE, RMSE, R^2.
- **Why**: Cross-validation tests model on different data splits for reliable performance. XGBoost handles complex patterns. Metrics show accuracy.
- **Results**: MAE: ~9,651, RMSE: ~38,369, R^2: ~0.295.
- **How it helps**: Improved MAE from ~11,747 and R^2 from ~0.17, showing better predictions.

6. Hyperparameter Tuning

- **What**: Used RandomizedSearchCV to test 50 combinations of XGBoost parameters (e.g., n_estimators, max_depth, learning_rate). Trained best model on full data and calculated metrics.
- **Why**: Tuning finds optimal settings for better accuracy. Testing on full data shows final performance.
- **Results**: Best parameters: subsample=0.8, n_estimators=200, max_depth=5, etc. MAE: ~8,591, RMSE: ~35,218, R^2: ~0.406.
- **How it helps**: Reduced MAE by ~26% and increased R^2 to 0.406, explaining 40.6% of price variation.

7. Final Model

- **How**: The final model is XGBoost with tuned parameters. It uses selected features (preprocessed) and predicts log_price, converted to normal prices.
- **Why**: XGBoost is powerful for tabular data. Tuning and cross-validation ensure accuracy.
- **Performance**: MAE: ~8,591, RMSE: ~35,218, R^2: ~0.406.

Summary

Stage	**MAE**	**RMSE**	**R^2**
Baseline	11,747	37854	0.17
Cross-Validation	9,651	38,369	0.295
After Tuning	8,591	35,218	0.406