# LA Airbnb Pricing Prediction

Matthew Dall'Asen

Data Science Institute, Brown University

Github Repository: https://github.com/mdallasen/Airbnb-LA-Pricing

# Introduction

Airbnb hosts continually face the challenge of accurately pricing their listings in line with property and location characteristics. Achieving a representative price remains a difficult task, yet doing so will prevent lost earnings from mispricing and improve the customer experience. To determine what an "appropriate" price is for an Airbnb listing, this report will aim to provide a price per night prediction based on several property, review, and market characteristics.

An LA Airbnb Pricing dataset was web scraped from the Airbnb website for property listings [1], providing ~44,000 property listings with over 70+ pricing, market, property, and review features. The target variable selected from this dataset was price per night, which ranged from 6 to 99999 with an average price of 317.20. Due to the nature of how the data was acquired, there were several issues that needed to be solved prior to model development. Namely, over 45% of property listings contained missing values with various imbalanced and skewed features.

Recent studies in Airbnb pricing have achieved relatively high levels of accuracy. For instance, Kalehbasti et al. [2] used machine learning techniques combined with sentiment analysis to predict prices, while Camatti et al. [3] and Alharbi [4] focus on sustainable models for price prediction in various cities. However, recent research hasn't focused on the Los Angeles region, nor does it deploy techniques to deal with non-IID data. This report, therefore, aims to address this gap by considering spatial and market-specific features unique to LA listings.

# Exploratory Data Analysis

## Price

The distribution of price per night is right-skewed (Figure 1), with most listings priced below $250. As prices increase, the frequency of listings decreases sharply, with very few properties priced above $1,000. Consequently, prices have been log transformed.

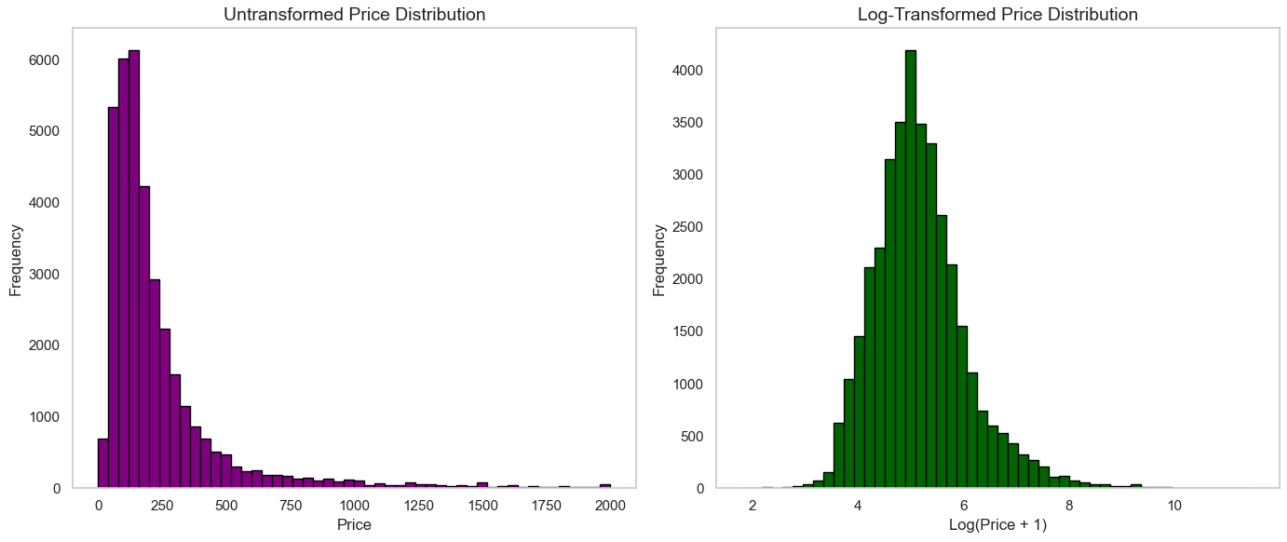


**Figure 1:** Comparison of Price & Log Transformed Price

## Price Vs Property Type

Property type is an important factor in determining prices, given that customers are often more willing to pay for a property that's larger or more luxurious. As seen in Figure 2, Entire villas, vacation homes, and serviced apartments exhibit higher median prices and larger interquartile ranges (IQRs), suggesting greater price variability. In contrast, shared rooms in homes and rental units typically have lower median prices and tighter IQRs.

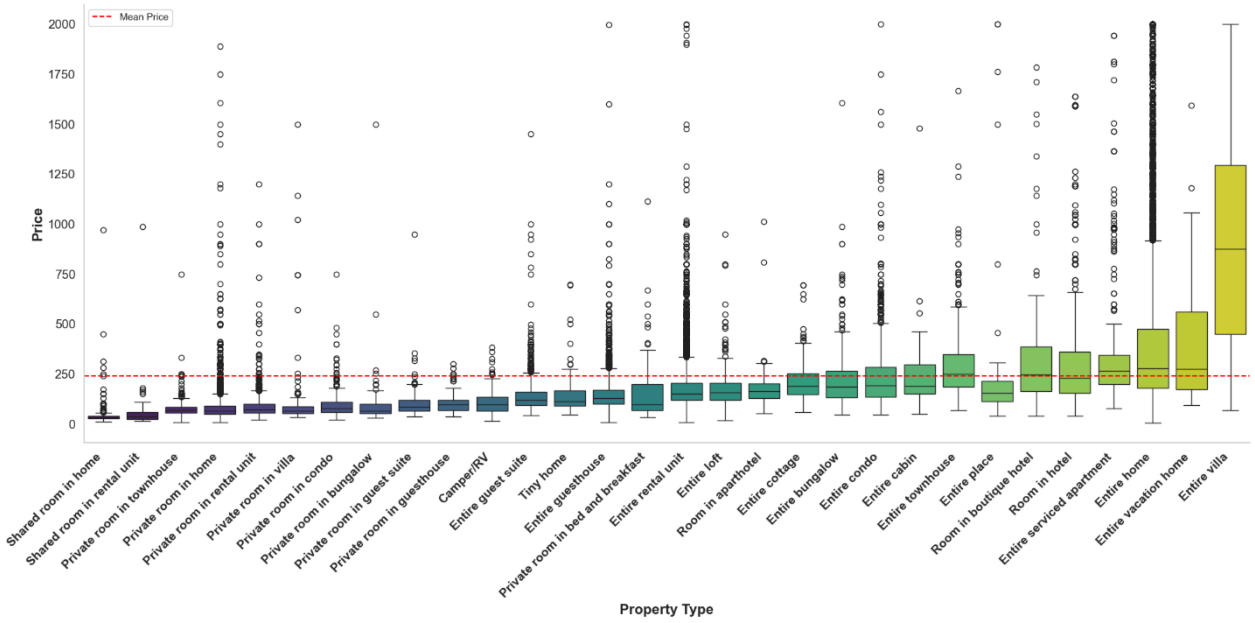


**Figure 2:** Box Plot Comparing Price & Property Type

## Price Vs Neighborhoods

Neighborhood highlights the dependency between values based on their geographical location to one another. Describing the relationship between location and price, Figure 3 demonstrates that the most expensive listings (>$300, black dots) are concentrated in central and coastal regions like Beverly Hills and Santa Monica, known for luxury and tourism. Mid-range prices ($150–$250, orange and red dots) cluster in popular neighborhoods, showing varied pricing in high-demand areas.

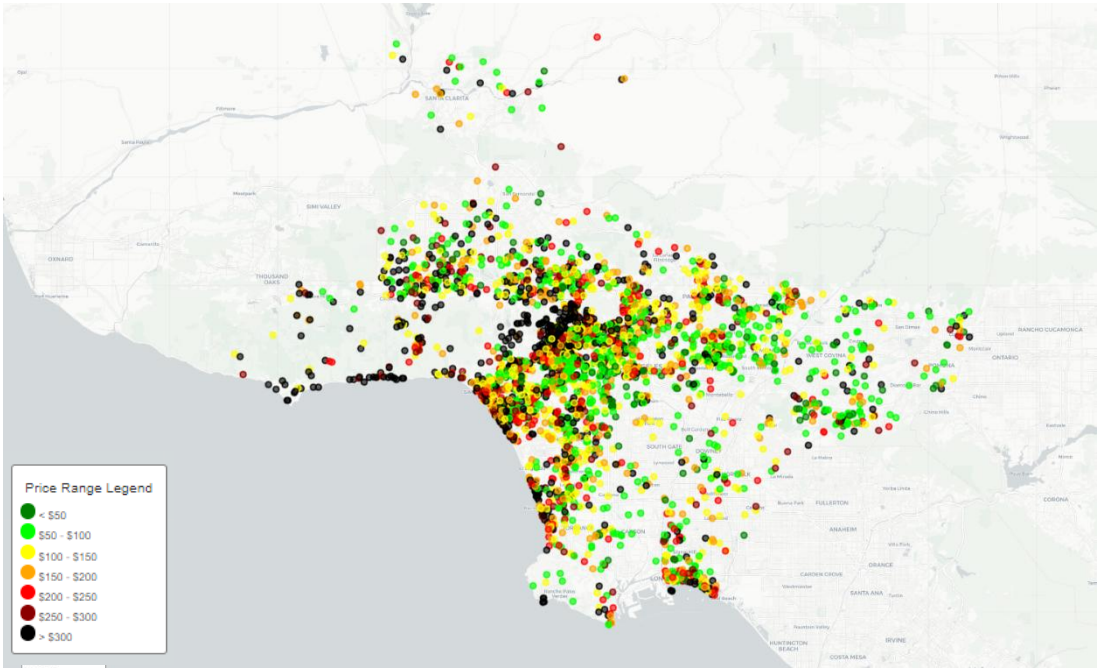


**Figure 3:** Map of LA with Airbnb Listing Prices

## Price Vs Review Ratings

There is little to suggest that customer ratings influence price, at least in a linear context. Most review scores are clustered around 5.0 (Figure 4), regardless of price, indicating high ratings across the market. As prices rise, listing density decreases, though some premium listings maintain excellent ratings. Properties with review scores below 4.5 are rare and distributed across various prices, showing that lower ratings are uncommon.

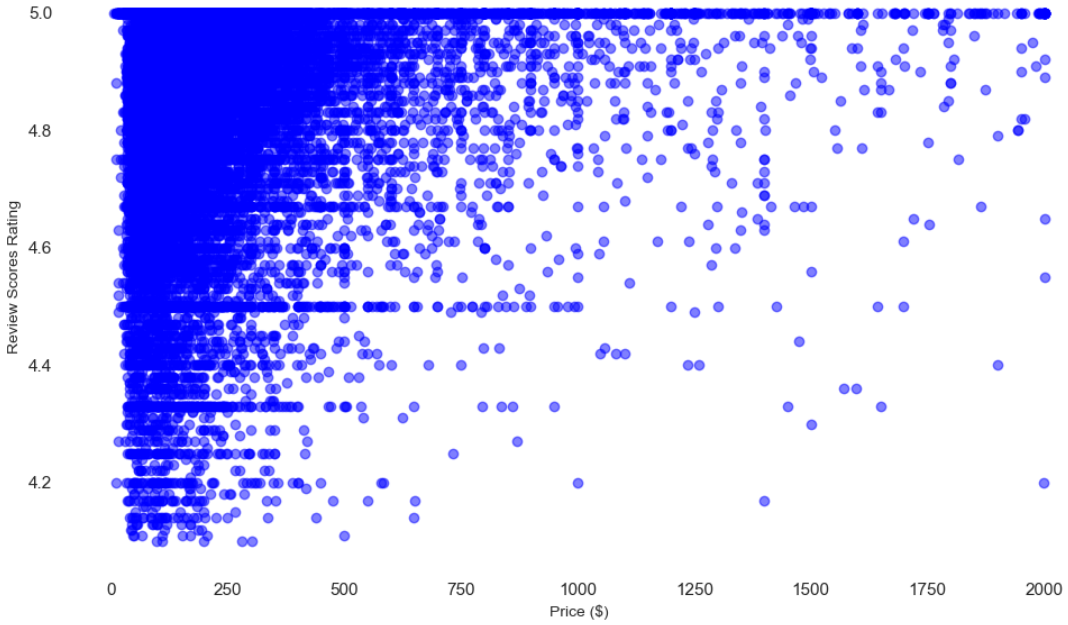


**Figure 4:** Scatter Plot of Price & Overall Review Ratings

# Methods

## Preprocessing

Prior to model deployment, the LA Airbnb Pricing dataset was preprocessed to address missing values, scale features, and encode categorical data. Missing values in continuous features were imputed using a Bayesian Ridge IterativeImputer, which provides a probabilistic, low computational method for capturing relationships in high-dimensional datasets whilst reducing the possibility of overfitting [5]. This approach is useful given the feature interactions present with the LA Airbnb Pricing dataset, as it outperforms simpler imputation methods that might overlook such patterns [6]. Continuous features are subsequently scaled using StandardScaler or MinMaxScaler depending on whether that feature is confined to set ranges or not (e.g. Age vs Availability), ensuring the optimization of models can be conducted appropriately. Categorical features are processed with a SimpleImputer, replacing missing values with a constant placeholder, and then encoded using OneHotEncoder. Features that weren't deemed critical (e.g. ID, URL, scrape_ID etc.) were dropped. The final dataset for development consisted of roughly ~400 features and 44,000 rows.

## Splitting, Cross Validation & Hyperparameter Selection

This dataset violates the non-IID assumption, where individual data points in proximity can hint at each other's price, given they are influenced by the same geographical features and characteristics (e.g. popular tourist attractions) [7]. To determine these groups, KKN clustering was employed based on the longitude and latitude. The number of clusters was treated as a hyperparameter, where 100 was determined as the best tradeoff between minimizing data leakage and capturing a group small enough to not misrepresent pricing. Neighborhoods could have been selected, but it was determined this didn't sufficiently capture proximity given that geographical characteristics can span multiple regions.

GroupShuffleSplit was used to divide the dataset into 80% train / validation and 20% test sets, whilst ensuring that all clusters remain within either the training or testing set. Within the train / validation set, GroupKFold was employed to create five folds that maintained group integrity within each. Each fold serves as the validation set while the remaining folds are used for training. This approach prevents data leakage between sets and ensures that the model's performance is evaluated on entirely unseen groups during each iteration, simulating real-world deployment scenarios where predictions must generalize new group patterns [8].

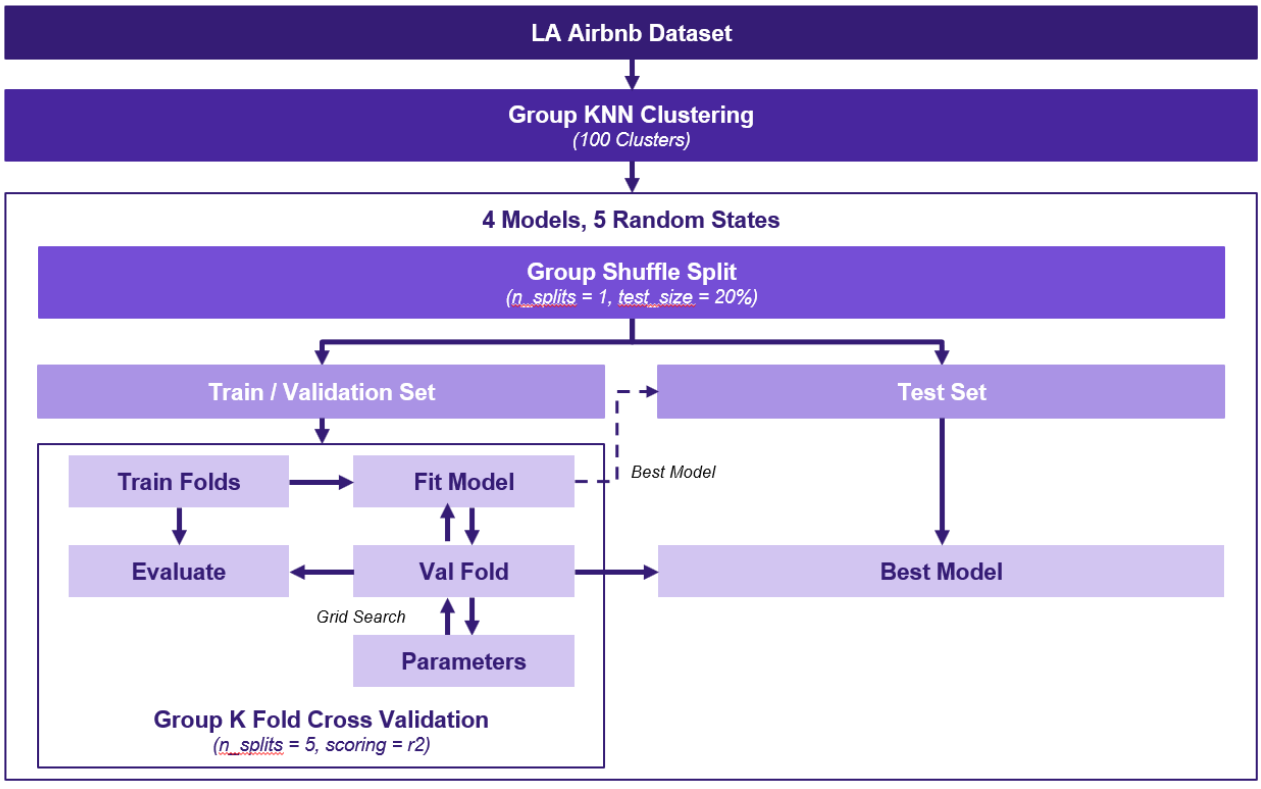


**Figure 5:** Diagram of ML Pipeline

To identify the best hyperparameters for each model, GridSearchCV was applied using the splits generated by GroupKFold. The grid search optimizes the R2, by systematically testing the different parameter combinations. This process was repeated across 5 random states, to minimize uncertainty and maintain robustness throughout testing. For each new random state, a new group aware train test split is generated and cross validation is performed on the training set. To ensure deterministic properties, random states have been fixed across all models.

## Evaluation Metric

The evaluation metric used during deployment was R-squared (R2), representing the 'goodness-of-fit' for regression models [9]. R2 quantifies the proportion of the variance in the dependent variable that can be explained by the independent variables [9]. Unlike error-based metrics, R2 focuses on the relationship between the predicted and actual values, emphasizing how well the mode captures the underlying patterns in the data [9]. In addition to it being highly interpretable, R2 is suitable for this report, given the aim of determining which model accounts for a larger share of the variation in price.

## ML Algorithm & Parameter Selection

This report seeks to determine the "best" model for predicting LA Airbnb Pricing. As seen in Table 1, four models have been selected with a focus on interpretability and minimization of computation requirements. Again, the hyperparameters have been selected with consideration of those that have the highest predictive power, particularly for the decision tree like models (with the exception XGBoost), where max depth and features were shown to have the biggest impact on performance. For XGBoost, the learning rate was more predominate given that this algorithm leverages gradient descent to construct it's trees.

| | Ridge Regression | Decision Tree | Random Forest | XGBoost |
|---|---|---|---|---|
| **Linear / Non Linear** | Linear | Non-Linear | Non-Linear | Non-Linear |
| **Description** | A linear regression model with L2 regularization that penalizes large coefficients | A tree-based model that splits data into subsets based on feature conditions to minimize prediction error | An ensemble learning method that builds multiple decision trees and averages their outputs for regression tasks | An ensemble of gradient boosted decision trees that uses a collection of weak learner trees to create a strong learner |
| **Parameters Tuned** *(Based on Predictive Influence)* | **L2 Regularization:** Adjusts the level of L2 norm regularization present within the MSE loss function | **Max Depth:** Determines the maximum depth of the tree (e.g. splits)<br>**Max Features**: Limits the number of features to consider when looking for the best split at each node. | **Max Depth:** Determines the maximum depth of the tree (e.g. splits)<br>**Max Features**: Limits the number of features to consider when looking for the best split at each node. | **Max Depth:** Determines the maximum depth of the tree (e.g. splits)<br>**Max Features**: Controls how much each tree contributes to the final prediction |
| **Parameters Range** | **L2 Regularization:** [8, 10, 12] | **Max Depth:** [5, 6, 7]<br>**Max Features:** [320, 330, 340] | **Max Depth:** [20, 30, 40]<br>**Max Features:** [100, 125, 150] | **Max Depth:** [2, 7, 10]<br>**Learning Rate:** [0.2, 0.25, 0.3] |
| **Rationale For Selection** | Provides a solid baseline for modeling linear relationships between features (e.g., number of bedrooms, bathrooms, location metrics) and prices. | Captures non-linear relationships effectively and provides interpretable decision paths | Captures interactions between features (e.g., host features and neighborhood amenities) without requiring extensive feature engineering | Well-suited for high-dimensional structured datasets, where interactions and complex patterns (e.g., seasonal price fluctuations or feature interactions) are crucial |

**Table 1:** ML Models & Their Hyperparameters [2, 3, 4]

# Results

## ML Algorithm Comparison

After deploying the models referenced (Figure 6), this report finds that XGBoost achieves the highest mean R2 score when compared to Ridge Regression, Decision Trees, and Random Forest. This is a result of XGBoost's ability to effectively capture complex, nonlinear relationships through gradient boosting and regularization techniques, which minimizes overfitting and allows the model to generalize on unseen data [10]. The small standard deviation produced by this model suggests that it consistently achieves this performance across different random states. Random Forest performs well but trails behind XGBoost. By averaging predictions from multiple decision trees, it reduces variance and handles non-linearity, but its lack of boosting limits its accuracy compared to XGBoost [11]. Decision Tree is the worst performing model, where Decision Trees often overfit training data and lack the ensemble techniques used by Random Forest and XGBoost [12]. Interestingly, Ridge Regression demonstrates a performance relatively equal to Random Forests, where this dataset clearly benefits from higher generalization when applying L2 norm to the loss function. However, Ridge Regression doesn't perform as consistently as the others, with a higher standard deviation [13].

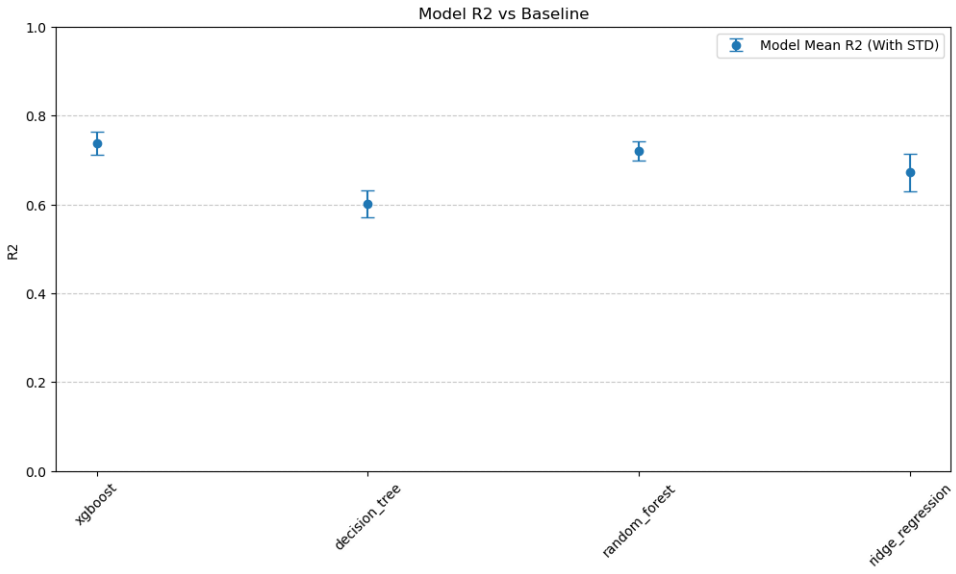


**Figure 6:** Mean & STD R2 for Each Model

XGBoost, as the most predictive model, shows a strong correlation between predicted and actual values. The scatter plot below (Figure 7) demonstrates consistent accuracy, with minimal deviations around the perfect prediction line. Configured with a max depth of 7 and learning rate of 0.2, the model balances complexity and generalization, effectively capturing trends while avoiding overfitting. Significant deviations at the extremes may indicate noise, underfitting, or cases where the model struggles to generalize. Further optimization or data exploration could address outliers, improving accuracy for extreme cases and enhancing XGBoost's reliability. Incorporating additional features, such as property age, recent renovations, unique amenities, or location advantages, could also provide better context for higher-priced listings, helping the model capture nuances and generalize more effectively.

## Feature Importance

To study which features are most important in the best-performing model, different metrics were explored (Figure 8). According to permutation importance, accommodates, neighborhood, bathrooms, room type, and property type emerged as the most critical predictors of property prices. These characteristics define the "size" and functionality of a property, influencing its value. Security also stood out as a significant factor, with hosts charging premium prices for properties equipped with additional security features, reflecting its perceived importance among renters. Furthermore, availability

plays a crucial role in pricing strategies. Properties available within the next 30 days are highly desirable and tend to command higher prices, while those with higher minimum night requirements cater to longer-term stays, shaping their pricing accordingly.

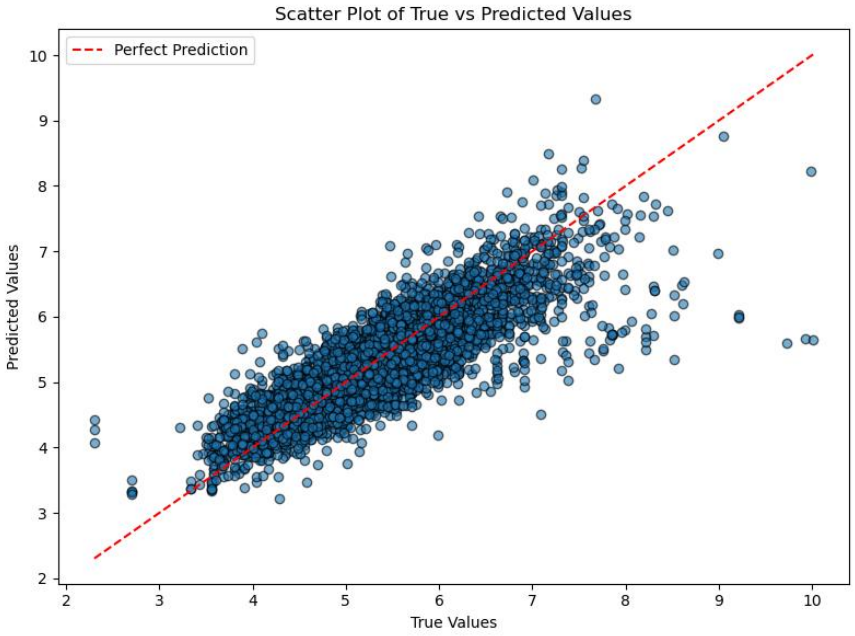


**Figure 7:** XGBoost Predicted vs True Values

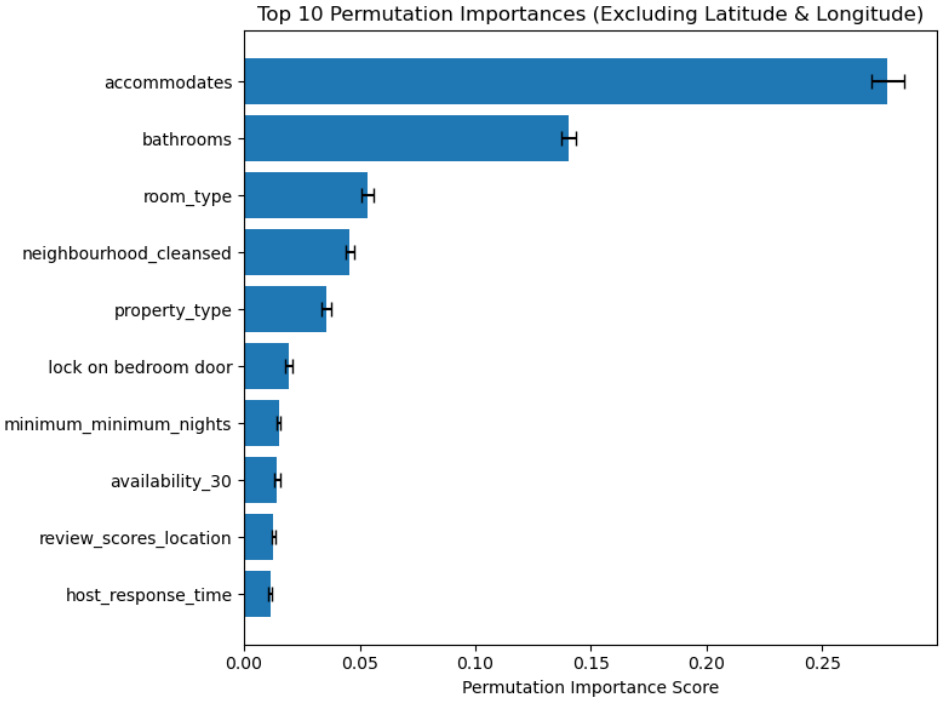


**Figure 8:** XGBoost Permutation Feature Importance

In addition to computing permutation importance, gain and weight metrics were calculated to understand the relative importance of features in the model (Figure 9, 10). The weight chart highlights the top 10 features by their frequency of use

in splits, with key features such as the number of reviews, timing of reviews (first_review and last_review), and host tenure (host_since) emerging as critical indicators. Features like accommodates and availability metrics also contribute, reflecting property capacity and availability. The gain chart, on the other hand, focuses on the improvement in model performance brought by each feature. Again, accommodates and bathrooms are key factors, alongside room type and specific amenities (e.g., lock on bedroom door). More specific than permutation importance, gain highlights Malibu and Pasadena as key areas, highlighting preference for desirable locations that have geographical significance.

To determine the influence of features on a local level (Figure 11), SHAP values were analyzed across two randomly selected predictions. The largest contributing features, such as accommodates (negative impact) and features like bathrooms or amenities_count (positive impact), dominate the prediction. Both graphs indicate the cumulative contributions of several top features, with the remaining features summarized collectively as 461 other features

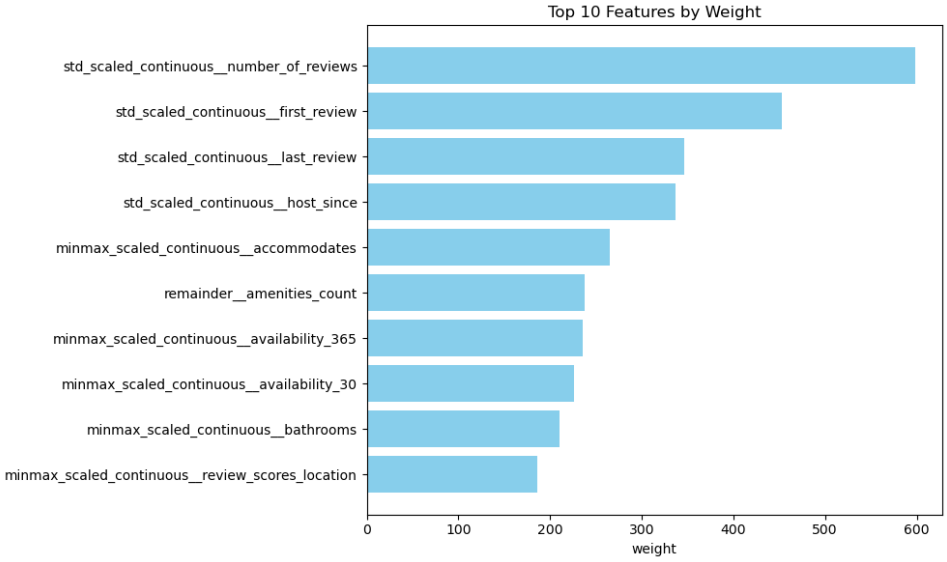


**Figure 9:** XGBoost Weight Feature Importance

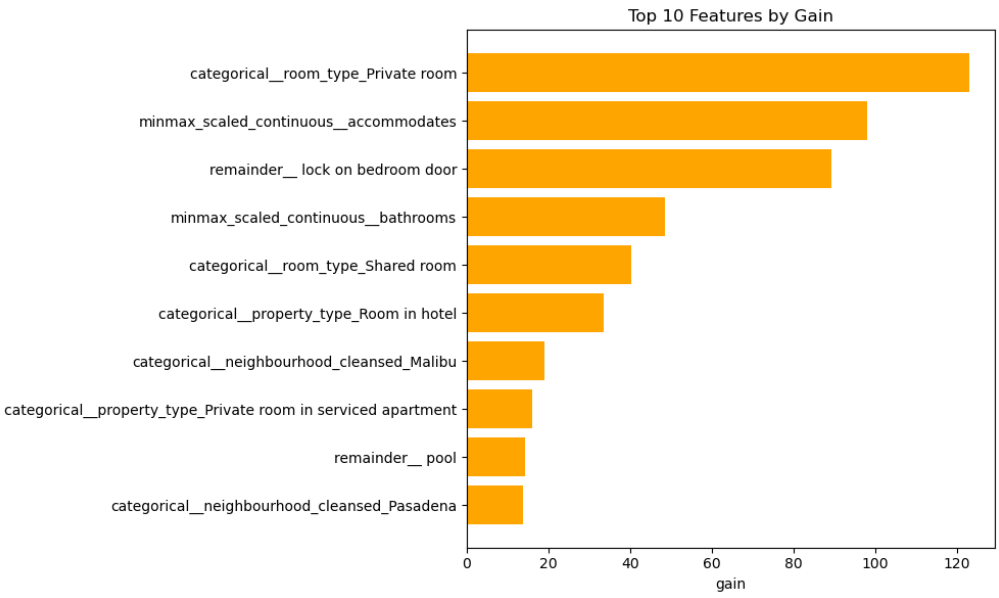


**Figure 10:** XGBoost Gain Feature Importance

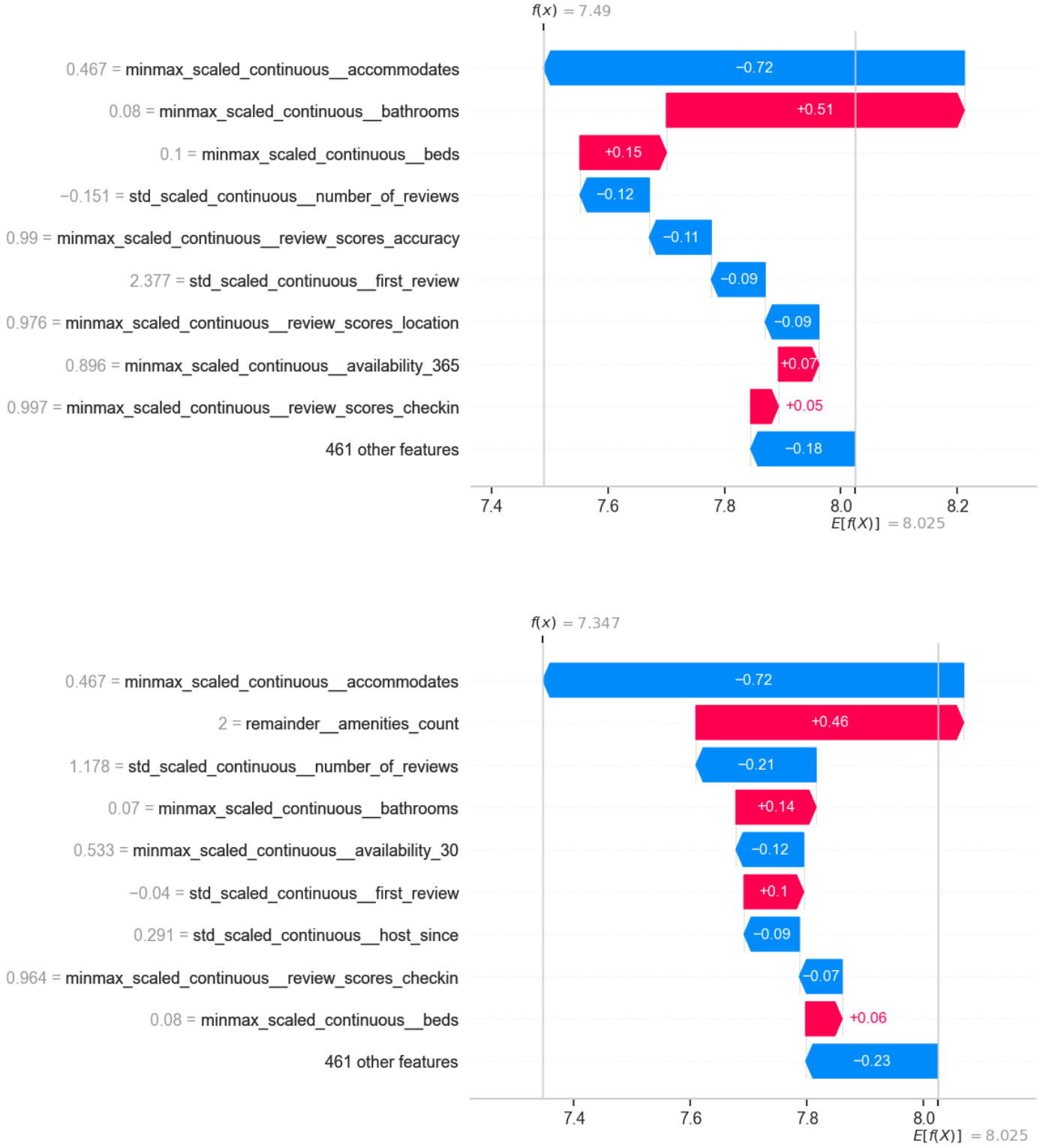


**Figure 11:** Local Feature Importance with SHAP Waterfall

## Outlook

There are several areas for future work to improve performance and accuracy. Namely, more complex model development, such as other ensemble methods and neural networks, to better capture non-linear pricing patterns. Expanding the number of hyperparameters within the current models can also optimize performance. Another crucial step is capturing additional pricing-related data which can provide additional insight into variances amongst higher priced properties. Additionally, reducing the feature dimensionality (e.g. PCA) can simplify the model while identifying the most significant contributors to price variance. Finally, exploring different missing value imputation techniques, such as KNN or pattern-based methods, and comparing them against the imputation approach, will help improve the model's handling of incomplete or missing data.

# References


[1] Inside Airbnb (2024) *Los Angeles: Get the data*. Available at: http://insideairbnb.com/get-the-data/ (Accessed: 15 December 2024).

[2] Rezazadeh Kalehbasti, P., Nikolenko, L. and Rezaei, H. (2021) 'Airbnb price prediction using machine learning and sentiment analysis', in *Machine Learning and Knowledge Extraction*. Lecture Notes in Computer Science, vol 12844. Springer, Cham, pp. 173–184. Available at: https://link.springer.com/chapter/10.1007/978-3-030-84060-0_11 (Accessed: 15 December 2024).

[3] Camatti, N., di Tollo, G., Filograsso, G. and Ghilardi, S. (2024) 'Predicting Airbnb pricing: a comparative analysis of artificial intelligence and traditional approaches', *Computational Management Science*, 21(30). Available at: https://link.springer.com/article/10.1007/s10287-024-00511-4 (Accessed: 15 December 2024).

[4] Alharbi, Z.H. (2023) 'A sustainable price prediction model for Airbnb listings using machine learning and sentiment analysis', *Sustainability*, 15(17), p. 13159. Available at: https://www.mdpi.com/2071-1050/15/17/13159 (Accessed: 15 December 2024).

[5] *Scikit-learn Developers* (2024) *IterativeImputer: Probabilistic imputation with Bayesian Ridge*. Available at: https://scikit-learn.org/stable/modules/impute.html#iterative-imputer (Accessed: 15 December 2024).

[6] Van Buuren, S. and Groothuis-Oudshoorn, K. (2011) 'mice: Multivariate imputation by chained equations in R', *Journal of Statistical Software*, 45(3), pp. 1–67.

[7] Dubé, J.P., Sudhir, K. and Chintagunta, P.K. (2005) 'Estimation of price elasticities from household survey data', *RAND Journal of Economics*, 36(3), pp. 482–502.

[8] Bischl, B., Lang, M., Kotthoff, L., Schiffner, J., Richter, J., Studerus, E., Casalicchio, G. and Jones, Z. (2021) 'mlr3: A modern object-oriented machine learning framework in R', *Journal of Open Source Software*, 6(51), p. 2878.

[9] Draper, N.R. and Smith, H. (1998) *Applied regression analysis*. 3rd edn. New York: Wiley.

[10] Chen, T. and Guestrin, C. (2016) 'XGBoost: A scalable tree boosting system', in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco: ACM, pp. 785–794.

[11] Breiman, L. (2001) 'Random forests', *Machine Learning*, 45(1), pp. 5–32.

[12] Quinlan, J.R. (1986) 'Induction of decision trees', *Machine Learning*, 1(1), pp. 81–106.

[13] Hoerl, A.E. and Kennard, R.W. (1970) 'Ridge regression: Biased estimation for nonorthogonal problems', *Technometrics*, 12(1), pp. 55–67.