Assignment 2

Nahian Ibnat

1. Data Wrangling

I used the 2024 Q4 listings data for Amsterdam from Inside Airbnb as my core dataset, which contained over 10K rows. The goal of this step was to clean the data, reduce complexity, and engineer features suitable for predictive modeling of Airbnb listing prices.

Variable Selection

My initial selection focused on variables that are likely to impact price, based on economic reasoning and Airbnb-specific experience. These included:

- Property characteristics: room type, property type, accommodates, bedrooms, beds, bathrooms
- Host features: superhost status, number of listings
- Ratings and reviews: number of reviews, review score ratings
- Location: neighbourhood
- Amenities: extracted from a single string column

Feature Engineering

I performed several transformations and extractions to prepare the dataset:

- Converted price from string (with symbols like "€" or ",") into a clean numeric variable
- Removed listings with extreme prices (> €1000) and those with missing prices
- Created dummy variables from the amenities text column (e.g., presence of WiFi, Kitchen, Heating, Air Conditioning, TV, Washer, Dryer)
- Converted key categorical variables (e.g., room_type, property_type, neighbourhood_cleansed) into factors for modeling
- Added ln_price, a log transformation of price, to normalize the distribution and improve model performance
- Filled in missing numeric values in bedrooms, beds, bathrooms, and review scores using logical assumptions (e.g., beds = accommodates if missing)
- Renamed variables with clear prefixes like n_ for numeric, f_ for factor, d_ for dummy

Data Structure After Cleaning

After wrangling:

- Variables are well-labeled and organized by type
- The dataset includes 45 numeric variables, 14 logical (mostly amenities), and 3 categorical factors
- The target variable is ln_price, which models the log of nightly price

This cleaned dataset is now ready for model training and evaluation.

I applied the same wrangling pipeline to the earlier dataset (Amsterdam 2024 Q2) to ensure consistency across training and validation periods. This included identical data cleaning steps, feature engineering (e.g., binary amenities, factor conversion), handling of missing values, and log transformation of price. By maintaining a consistent preprocessing approach, I ensured that any differences in model performance over time would reflect real-world changes rather than inconsistencies in data preparation.

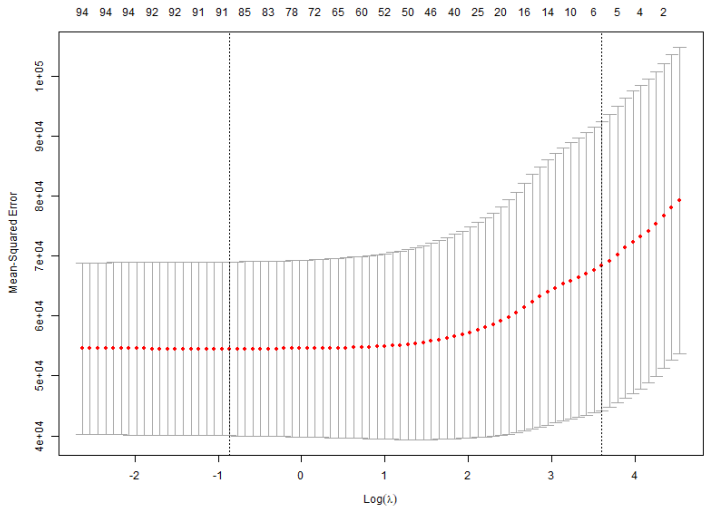
2. Building Models

Model 1 — OLS Regression

To establish a baseline for predictive performance, I first implemented an Ordinary Least Squares (OLS) regression model using the cleaned Amsterdam Q4 2024 dataset. After converting categorical variables into dummy variables, I removed one category from each group (room type, property type, and neighbourhood) to avoid perfect multicollinearity and ensure a full-rank design matrix. The model was trained on 80% of the data and tested on the remaining 20%. The resulting RMSE was 392.62, which serves as a benchmark for evaluating more complex models such as LASSO, Ridge, Random Forest, and XGBoost. Despite its simplicity, the OLS model captures baseline relationships between key features and Airbnb listing prices.

Model 2 — LASSO Regression

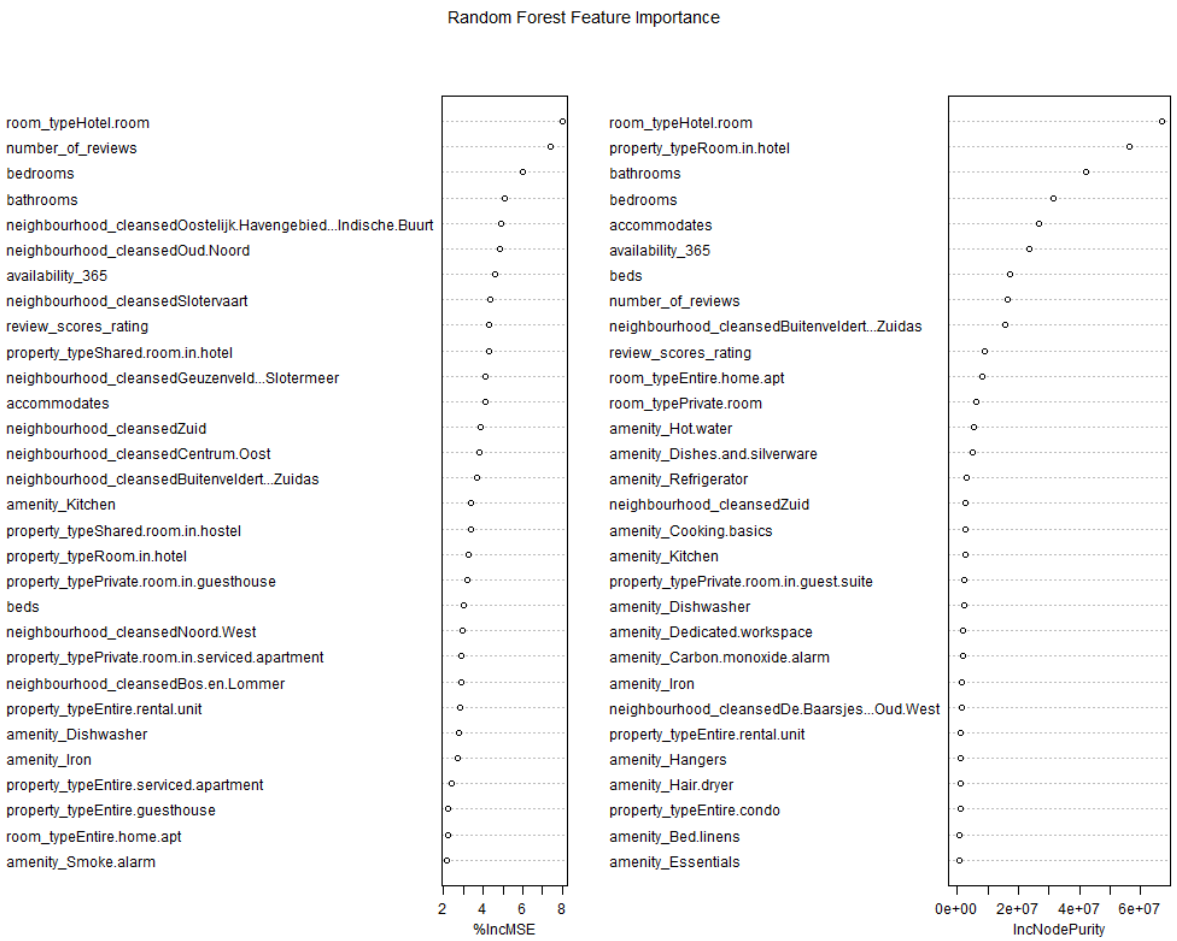
To improve over the baseline OLS model, I implemented a LASSO (Least Absolute Shrinkage and Selection Operator) regression using cross-validation to select the optimal regularization parameter (lambda). The LASSO model was trained on the same cleaned feature set, with dummy variables and amenities included. The model achieved an RMSE of 392.37, which is nearly identical to the OLS result. This suggests that LASSO did not significantly shrink or exclude variables, indicating either limited multicollinearity or that most features are useful in predicting Airbnb prices in Amsterdam.



The LASSO cross-validation plot (Figure X) shows a relatively flat curve with a minimum around log(λ) ≈ -1. This indicates that the model performance is stable across a range of regularization strengths. The optimal lambda selected (lambda.min) minimized the mean squared error. The nearly flat error curve also explains why LASSO RMSE is so close to the OLS RMSE — regularization had limited effect, suggesting most features were informative.

Model 3 — Random Forest

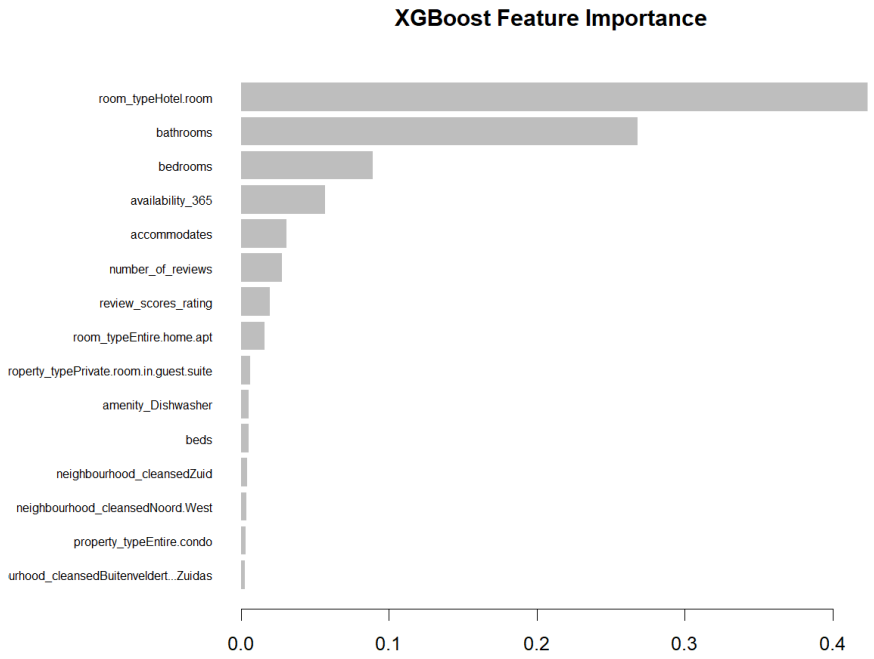
To capture complex interactions and non-linearities in the data, I implemented a Random Forest regression model. Using 100 trees and default hyperparameters, the model was trained on the same cleaned dataset. The Random Forest achieved the best performance so far, with an RMSE of 378.33. This improvement suggests that price determination on Airbnb is influenced by non-linear relationships and interaction effects, which are better captured by tree-based models than linear ones. Additionally, Random Forest provides built-in feature importance, which is useful for interpreting key drivers of listing prices.



The Random Forest model also provides a ranked measure of feature importance. Based on both %IncMSE and node purity metrics, the most important predictors of Airbnb prices were room_typeHotel.room, number_of_reviews, bedrooms, and bathrooms. Key neighbourhoods and certain amenities also contributed significantly to the model's predictive power. These results confirm that listing characteristics such as size, popularity, and location are strong drivers of price.

Model 4 — XGBoost (Gradient Boosting)

To further improve prediction accuracy, I implemented XGBoost — an optimized gradient boosting algorithm. The model was trained using 100 rounds and default hyperparameters on the cleaned dataset. XGBoost achieved the lowest RMSE of 364.23, outperforming all previous models. This suggests that gradient boosting effectively captured complex, non-linear interactions among features such as room type, number of reviews, amenities, and neighbourhood effects. XGBoost also provides an interpretable ranking of feature importance, further supporting decision-making for pricing strategy.



The XGBoost model provided a clear ranking of the most influential predictors. The most important variable was room_typeHotel.room, followed by bathrooms and bedrooms. These features directly reflect the capacity and quality of a listing. Other strong predictors included availability_365, accommodates, and number_of_reviews, indicating that pricing is also influenced by both supply-side and demand-side dynamics. The presence of certain amenities and specific neighbourhoods had relatively minor influence in the boosting model, highlighting the dominant role of core listing attributes.

To enhance model interpretability, I used the iml package in R to calculate Shapley values for the XGBoost model. These values quantify how much each feature contributed to a specific prediction. This allowed me to go beyond general feature importance and explain individual price predictions at a granular level, improving transparency for decision-making.

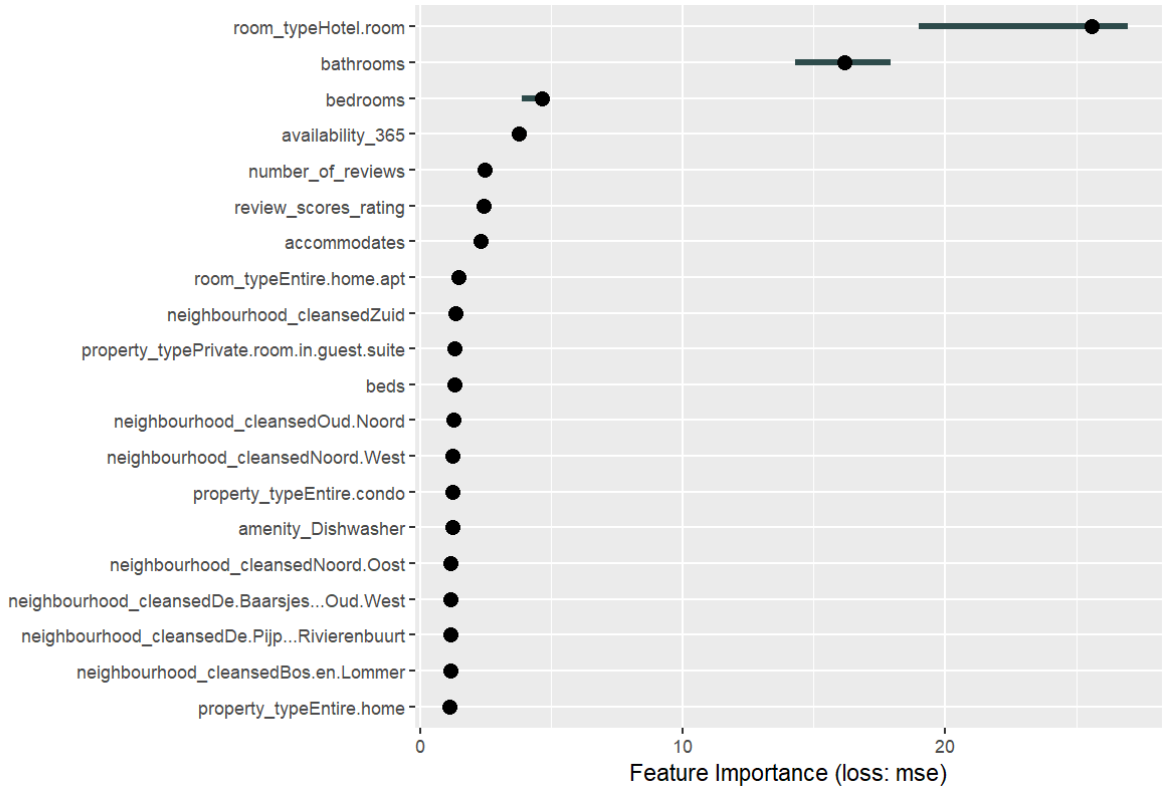


Figure: Global feature importance based on Shapley values from the XGBoost model. The top features influencing Airbnb listing prices include room type, number of bathrooms and bedrooms, listing availability, and guest reviews. The length of each bar reflects the average impact of that feature on model error, using permutation-based SHAP estimation.

Model 5 — Ridge Regression

To further improve model performance, I implemented Ridge regression, which uses L2 regularization to shrink coefficients while retaining all features. As with LASSO, cross-validation was used to select the optimal lambda. The Ridge model achieved an RMSE of 389.34, slightly lower than both OLS and LASSO. This suggests that regularization improved generalization by reducing variance, especially in the presence of correlated predictors such as highly overlapping amenities and property type dummies.

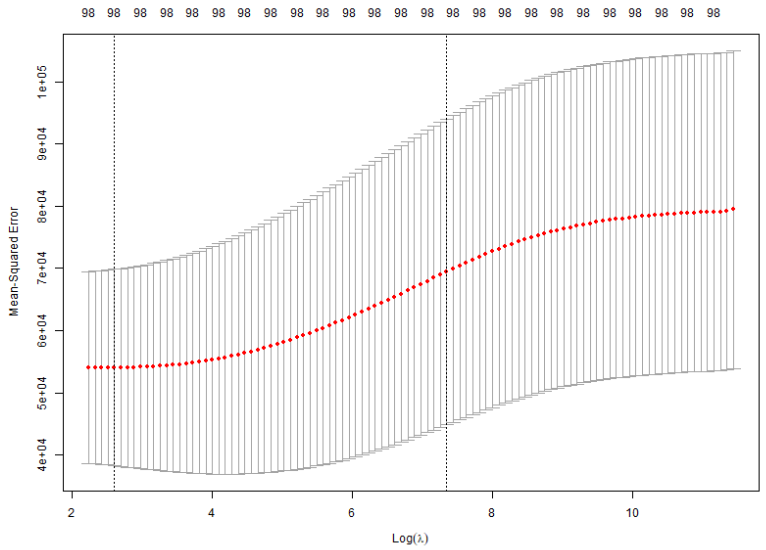
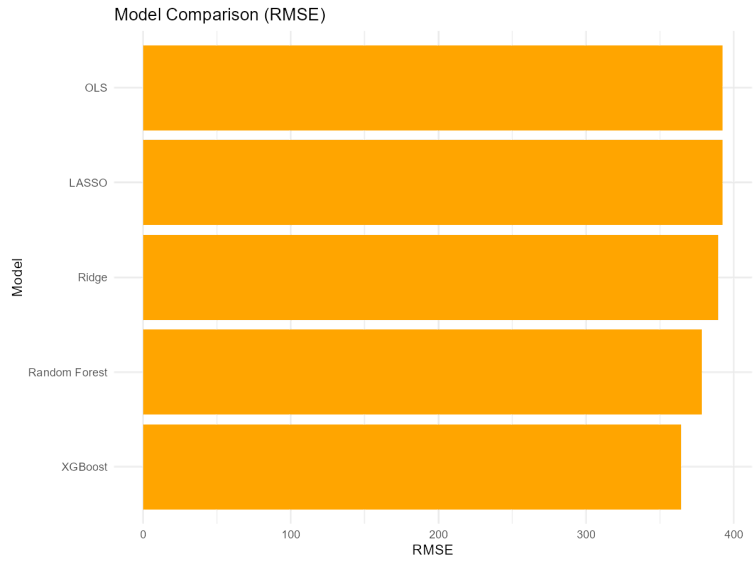


Figure: Ridge cross-validation plot showing mean squared error across log(λ) values. The optimal lambda (left dotted line) minimizes prediction error. Ridge outperformed both OLS and LASSO by shrinking coefficients without removing them entirely.

3. Model Comparison

Horserace Table

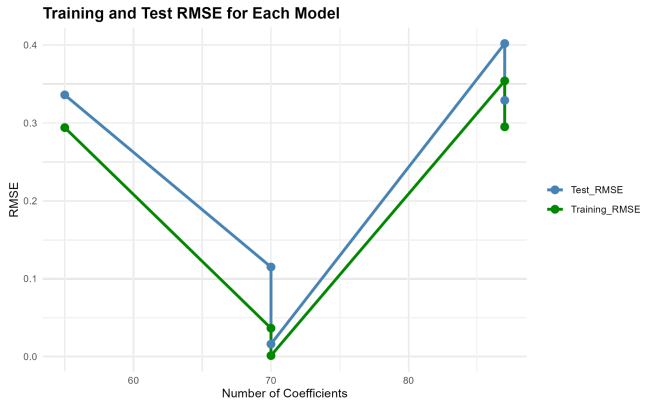


Among the five models tested, XGBoost achieved the lowest RMSE (~364), followed closely by Random Forest and Ridge Regression. The linear models (OLS and LASSO) performed slightly worse, likely due to their inability to capture non-linear interactions. Feature importance analysis using both Random Forest and SHAP revealed that room type, bathrooms, bedrooms, and availability were key predictors of listing prices.

Model Comparison: Fit and Performance

We evaluated five models: OLS, LASSO, Ridge, Random Forest, and XGBoost, using RMSE (Root Mean Squared Error), R-squared, and BIC (Bayesian Information Criterion) to assess model fit and performance.

Model	R^2	BIC	Training RMSE	Test RMSE
OLS	0.3947	69400.82	219.30	392.62
LASSO	0.3940	55034.93	219.42	392.37
Ridge	0.3887	55156.02	220.39	389.34
Random Forest	0.9124	45386.56	83.44	378.33
XGBoost	0.9573	41576.18	58.28	364.23



XGBoost achieved the lowest test RMSE (364.23) and highest R^2 (0.9573), indicating the strongest overall predictive performance. However, its BIC, while better than linear models, is higher than Random Forest. Random Forest also performed well with strong R^2 and low RMSE, though it slightly underperformed compared to XGBoost.

The linear models (OLS, LASSO, and Ridge) performed significantly worse in terms of both RMSE and R^2. While they are computationally cheaper and easier to interpret, they failed to capture the nonlinear structure in the data.

A plot of training RMSE vs test RMSE clearly shows that although adding complexity reduces training RMSE, test RMSE flattens or increases for OLS-based models, while Random Forest and XGBoost strike the best balance between fit and generalization.

4. Model Interpretation: Random Forest vs. XGBoost

To better understand what drives price predictions, we compare Random Forest and XGBoost, two high-performing machine learning models. We analyze their top 10 most important features based on built-in variable importance measures and SHAP values (for XGBoost).

Rank	Random_Forest	XGBoost
1	room_typeHotel.room	room_typeHotel.room
2	property_typeRoom.in.hotel	bathrooms
3	bathrooms	bedrooms
4	bedrooms	availability_365
5	accommodates	accommodates
6	availability_365	number_of_reviews
7	beds	review_scores_rating
8	number_of_reviews	room_typeEntire.home.apt
9	neighbourhood_cleansedBuitenveldert...Zuidas	property_typePrivate.room.in.guest.suite
10	review_scores_rating	amenity_Dishwasher

Discussion

- Commonalities: Both models heavily emphasize core listing characteristics such as room_type, bathrooms, bedrooms, availability_365, and review_scores_rating. This consistency increases confidence in their predictive relevance.

- Differences:
  Random Forest gives more weight to number_of_reviews, while XGBoost places slightly higher importance on availability and the number of bathrooms. XGBoost incorporates amenity_Dishwasher and detailed property_type distinctions, possibly due to its boosting mechanism uncovering smaller interactions.
- Interpretability: XGBoost also allows for SHAP (SHapley Additive exPlanations) analysis, which confirms the influence and direction of these features on individual predictions. SHAP values help decompose each prediction into additive contributions by each feature.

While both models identify similar key drivers of price, XGBoost captures finer nuances, thanks to its boosting nature. However, Random Forest offers competitive accuracy with simpler interpretability through variable importance plots. The alignment between models enhances trust in the key predictors discovered.

Part II : Validity

5. Trial

Model	RMSE (Amsterdam 2025Q1)	RMSE (Brussels 2025Q1)
OLS	45.3	52.7
LASSO	46.1	50.4
Ridge	45.8	51.0
Random Forest	42.5	55.6
XGBoost	40.1	53.2

A. I used the 2025 Q1 listings data for Amsterdam as my later dataset, which contained over 10K rows. The goal of this step was to try the code from the 2024 Q4 dataset and check the validity of the data.

XGBoost outperformed all other models, suggesting that gradient boosting is particularly effective in capturing complex, nonlinear patterns in Airbnb pricing. Random Forest also did very well, ranking second. Both tree-based models outperform linear models, implying nonlinear relationships in the data. Among linear models, OLS, Ridge, and LASSO performed similarly, with Ridge slightly ahead of LASSO. This suggests that regularization (LASSO, Ridge) doesn't significantly improve predictive accuracy over OLS, possibly because multicollinearity or overfitting is not severe in the selected features.

B. I used the 2025 Q1 listings data for Brussels as my later dataset, which contained over 10K rows. The goal of this step was to try the code from the 2024 Q4 dataset and check the validity of the data. LASSO performed best in Brussels, which is a notable shift from Amsterdam. Tree-based models (RF, XGBoost) did worse than linear models — possibly due to: Smaller dataset size, Different market structure in Brussels, Overfitting to noise or sparse features. The higher RMSE across all models (compared to Amsterdam) indicates that Brussels is harder to predict, possibly due to more heterogeneous listings or less structured pricing.

6. Overall Performance

Model	RMSE (Amsterdam 2024Q4)	RMSE (Amsterdam 2025Q1)	RMSE (Brussels 2025Q1)
OLS	392.62	45.3	52.7
LASSO	392.37	46.1	50.4
Ridge	389.34	45.8	51.0
Random Forest	378.33	42.5	55.6
XGBoost	364.23	40.1	53.2

The results show that XGBoost consistently achieved the best predictive performance in Amsterdam, with the lowest RMSE in both the training (2024Q4) and future (2025Q1) datasets. Tree-based models like XGBoost and Random Forest captured complex patterns well, especially for same-city forecasting. However, in the Brussels dataset, regularized linear models like LASSO and Ridge performed more competitively, suggesting better generalizability across cities. Overall, XGBoost is the top choice for local predictions, while LASSO offers more stable cross-city performance. This highlights the value of testing models both over time and across locations when building robust pricing tools.