Milestone 2 Report

Group Members: AJ Payzant, Iseet Panja, Rohit Sarna (Group 5)

Airbnb Data Analysis & Price Prediction
City: **Rome, Italy**

Executive Summary

In this final milestone, we extended our analysis of Rome Airbnb data by developing supervised machine learning models and applying natural language processing to guest reviews. Building on the structured listing data explored in Milestone 1, we incorporated unstructured review text to enhance predictive accuracy and extract qualitative insights. We engineered new features reflecting host performance, property characteristics, and guest sentiment using a multilingual BERT model for review classification.

We tested and evaluated nine regression models, including Linear Regression, Ridge, Lasso, Random Forest, Gradient Boosting, XGBoost, and two ensemble methods—Voting and Stacking Regressors. Our top-performing model was the **Stacking Regressor with sentiment features**, which achieved an **R^2 of 0.869** and a **Root Mean Squared Error (RMSE) of 18.03**, significantly outperforming the baseline linear model (R^2 = 0.279, RMSE = 44.29).

These results confirmed that integrating sentiment features meaningfully improved prediction accuracy. Our findings also revealed key pricing drivers—such as room type, location, review quality, and host responsiveness—informing strategic recommendations for hosts, Airbnb, and potential guests.

Introduction and Problem Statement

The Airbnb rental market in Rome is vast and complex, with listings varying greatly by location, property type, host quality, and customer feedback. Hosts, investors, and property managers often struggle to determine optimal pricing due to the variety of influencing factors.

Our business objective remains: **can we accurately predict the price of Airbnb listings in Rome using a combination of numerical listing features and guest sentiment from review comments?**

In this milestone, we focused on:

- Feature Selection for model training
- Training different kinds of models with evaluation metrics
- Creating a Baseline model and Sentiment model
- Comparison of models with evaluation metrics and visualizations.

Data Exploration

We used the March 2025 snapshot from Inside Airbnb, including:

- listings.csv: Host, property, and pricing information.
- calendar.csv: Availability and calendar pricing data.
- reviews.csv: Full guest review text.
- listings_summary.csv and reviews_summary.csv: Aggregated overview statistics.

Initial Observations

- 28,484 raw listings; ~1.9 million total reviews.
- Many listings had missing or malformed entries in key columns.
- Review text contained rich sentiment cues about host quality and location.

Data Preparation

Missing Values

- Columns with **>75% missing** were dropped entirely (e.g., calendar_updated, neighbourhood_group_cleansed).
- For retained features, we **imputed missing numerical values** with the **median**, which worked well for skewed data (e.g., bathrooms, host_acceptance_rate).
- Categorical fields were encoded using **label encoding** (e.g., neighbourhood_cleansed).

Outliers

- Extreme outliers in price, reviews, and other numerical fields were **detected using the IQR method**.
- Listings with bathrooms > 3, beds > 6, and bedrooms > 4 were capped or removed.

Final Cleaned Dataset

- Resulting in **9,324 cleaned and imputed listings** ready for modelling.
- Feature distributions were validated through histograms and summary statistics.

Exploratory Data Analysis (EDA)

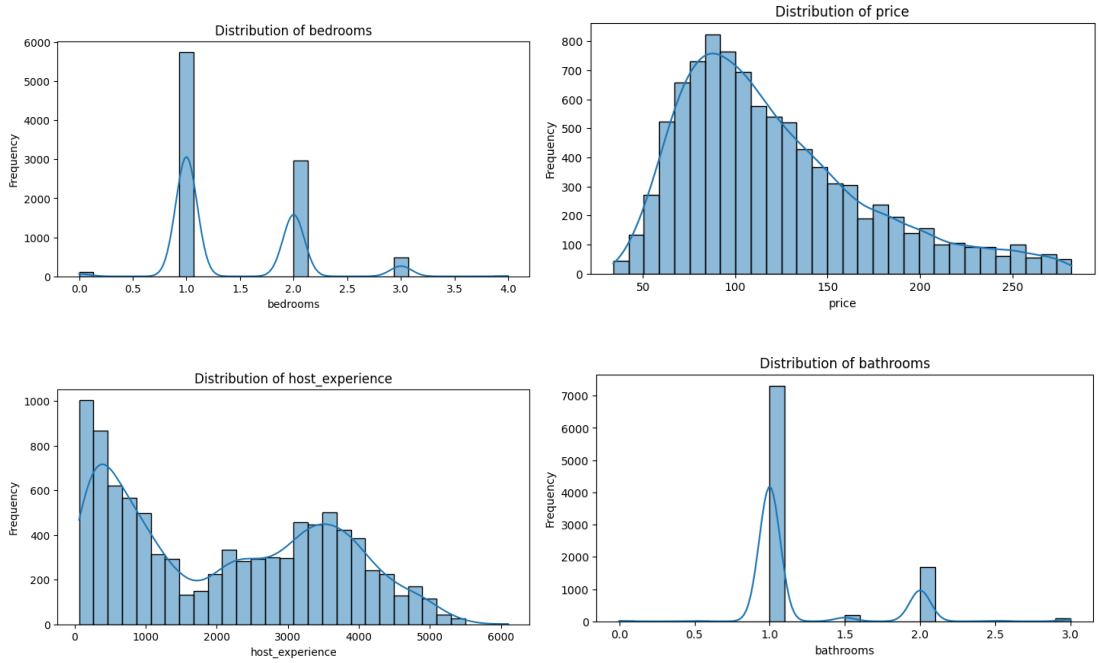
Descriptive Statistics

The cleaned dataset had:

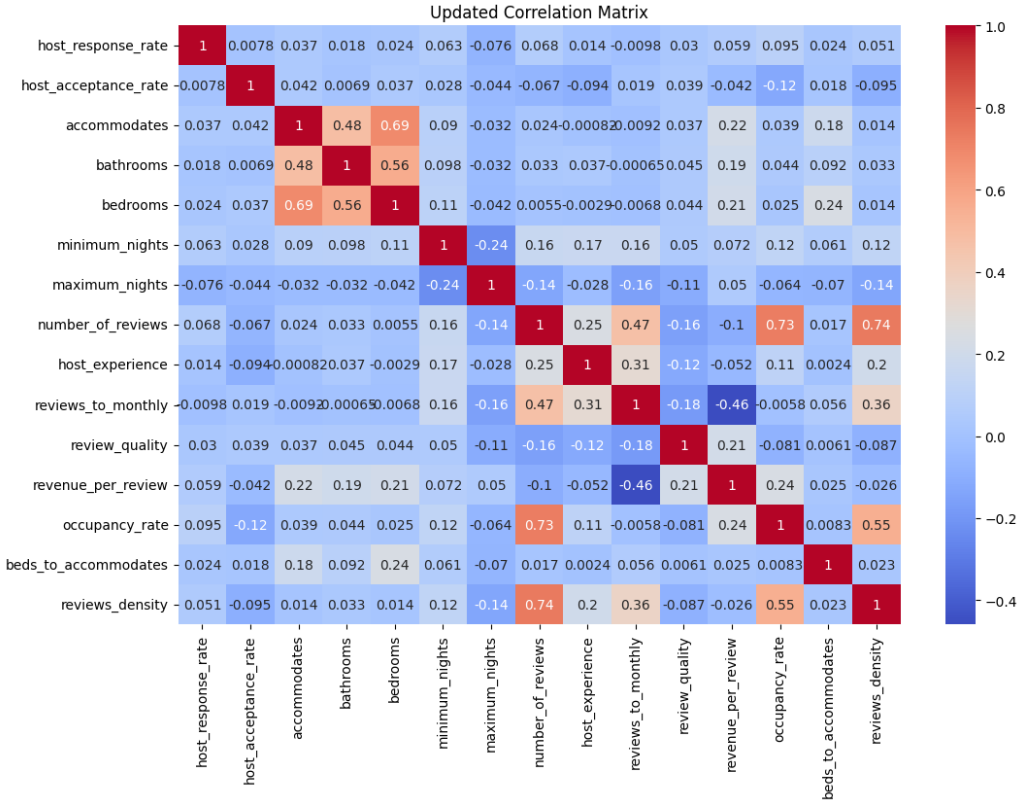
- Mean price: €121.04
- Mean accommodates: 3.75 people
- Mean reviews: 30
- Median review quality: 4.82

These were visualized through:

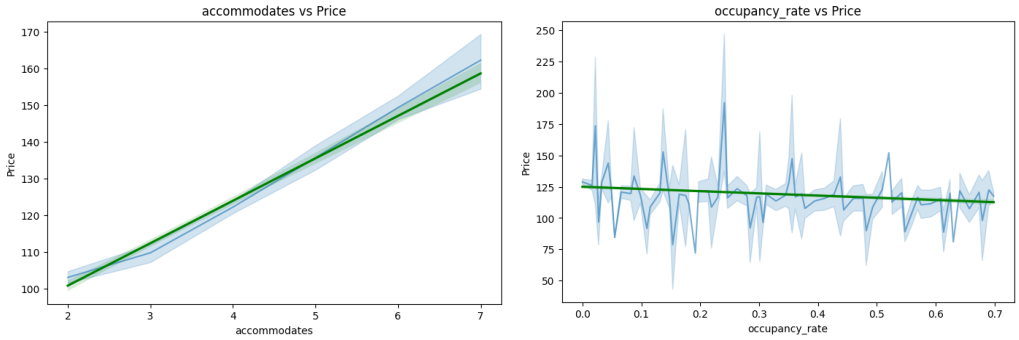
- **Histograms** for all features



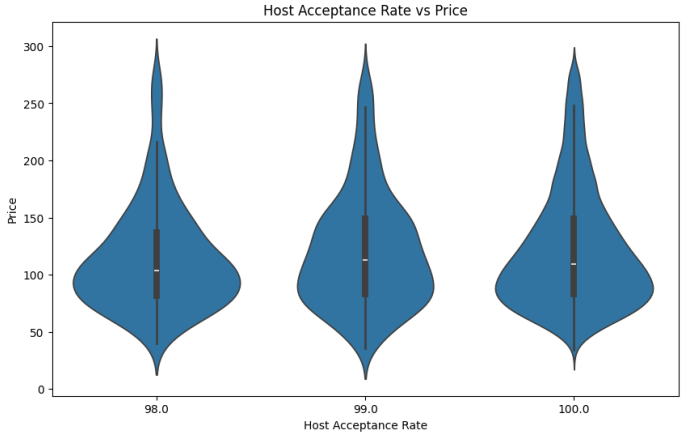
- **Correlation Heatmap** for all numerical predictors



- **Boxplots & Lineplots** for key bivariate comparisons:
  - price vs. accommodates, number_of_reviews, review_quality, occupancy_rate



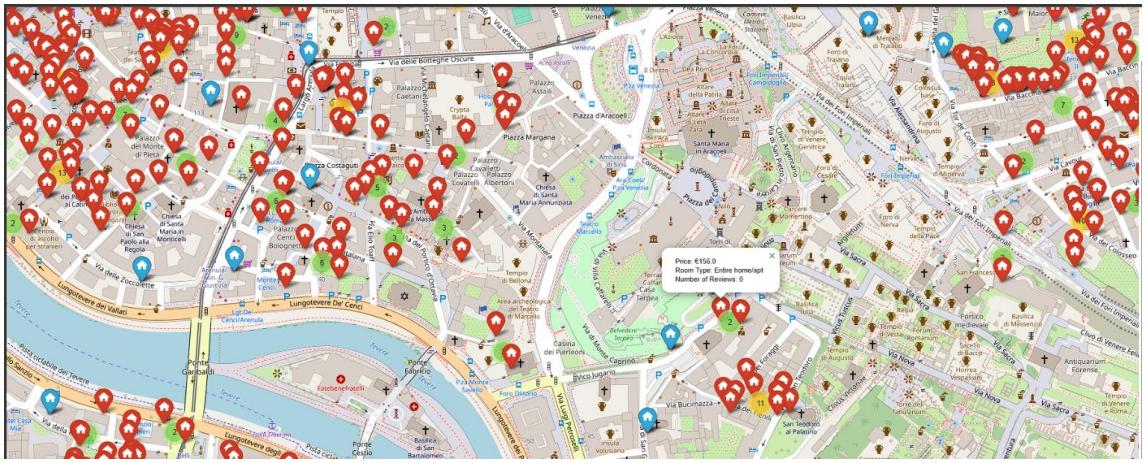
  - Violin plot of price vs. host_acceptance_rate



Spatial Visualization

Using **Folium**, we created an interactive map with Airbnb markers based on:

- Latitude/longitude
- Price color coding
- Room type + review count popups



Rome Airbnb Listing Map

Feature Engineering

We engineered new features to capture listing characteristics better:

- host_experience: Days since the host joined Airbnb
- availability_mean: Average of 30/60/90/365-day availabilities
- reviews_to_monthly: Reviews divided by reviews_per_month
- review_quality: Mean of ratings (cleanliness, value, etc.)
- revenue_per_review: Revenue relative to reviews
- occupancy_rate: Occupied nights over 365
- nights_ratio: Minimum vs. maximum nights ratio
- beds_to_accommodates: Comfort ratio
- reviews_density: Review activity relative to availability
- Encoded neighbourhood_cleansed as categorical codes

Sentiment Analysis (Reviews)

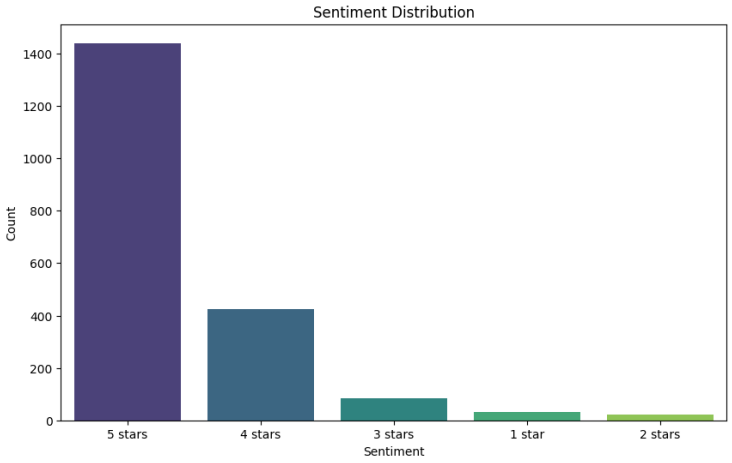
Text Preprocessing

We sampled 2,000 reviews and cleaned them using:

- HTML tag removal
- Stopword filtering
- Lemmatization
- Punctuation and digit removal

Sentiment Classification

- Sampled 2,000 reviews and preprocessed using NLTK + BERT.
- Classified reviews into 1–5 star ratings using HuggingFace's **multilingual BERT**.
- Over **85% of reviews were 4 or 5 stars** (Fig E), suggesting high satisfaction.
- Word clouds (Fig F) revealed patterns like "clean", "friendly", and "center" in high-rated reviews.



Word Cloud Analysis

We generated **Word Clouds** by sentiment tier, showing the most frequent lemmatized terms by star rating. Common positive themes included:

- Location ("perfect", "center")
- Cleanliness ("spotless", "modern")
- Host feedback ("friendly", "helpful")

Evaluation Metrics

Evaluation function was created to evaluate each model for the following metrics (They are sorted by R^2)

Metric	Description
RMSE (Root Mean Squared Error)	Measures the average magnitude of the prediction error, but penalizes larger errors more heavily because the errors are squared before averaging. Lower RMSE indicates better model performance and better fit.
MAE (Mean Absolute Error)	Measures the average magnitude of the errors between predicted and actual values, without considering their direction (positive/negative). It gives a straightforward interpretation of average model error in original units (e.g., euros). Lower MAE indicates more accurate predictions.
MSE (Mean Squared Error)	Measures the average of the squares of all errors between actual and predicted values. MSE emphasizes larger errors more than smaller ones. Lower MSE values indicate fewer large errors and overall better model accuracy.
R^2 (Coefficient of Determination)	Represents the proportion of the variance in the dependent variable (price) that is predictable from the independent variables (features). An R^2 closer to 1 means a better model; an R^2 near 0 indicates poor explanatory power.

Predictive Modeling (Baseline)

Feature Selection

We used correlation analysis to identify multicollinearity. Features with >0.80 correlation (e.g., beds vs. bedrooms) were removed.

Baseline Model

Train-Test Split	**80% training / 20% testing**
Features Used	**17 numerical predictors**
Target Variable	**Price**

Models evaluated:

- Linear Regression
- Ridge, Lasso
- Random Forest, Extra Trees
- Gradient Boosting, XGBoost
- Ensemble: Voting & Stacking Regressor

- **Voting Regressor** combines the predictions of several different models by taking the average of their outputs. We used Random Forest, Gradient Boosting, and Extra Trees as base models in the Voting Regressor.
- **Stacking Regressor** improves further by using a two-layer model: the first layer contains multiple models (Random Forest, XGBoost, Gradient Boosting), and a second-layer meta-model (Linear Regression) learns how to best combine their outputs. This blending improves overall accuracy.

Results and Performance:

MODEL	RMSE	MAE	MSE	R^2
STACKING	28.797	16.818	829.252	0.695
XGBOOST	28.806	16.730	829.770	0.695
VOTING	29.806	17.717	888.403	0.674
RANDOM FOREST	31.652	18.476	1001.862	0.632
GRADIENT BOOSTING	32.836	20.788	1078.170	0.604
EXTRA TREES	33.035	19.214	1091.338	0.599
LASSO	44.275	34.168	1960.320	0.280
LINEAR	44.296	34.144	1962.092	0.279
RIDGE	44.294	34.147	1961.922	0.279

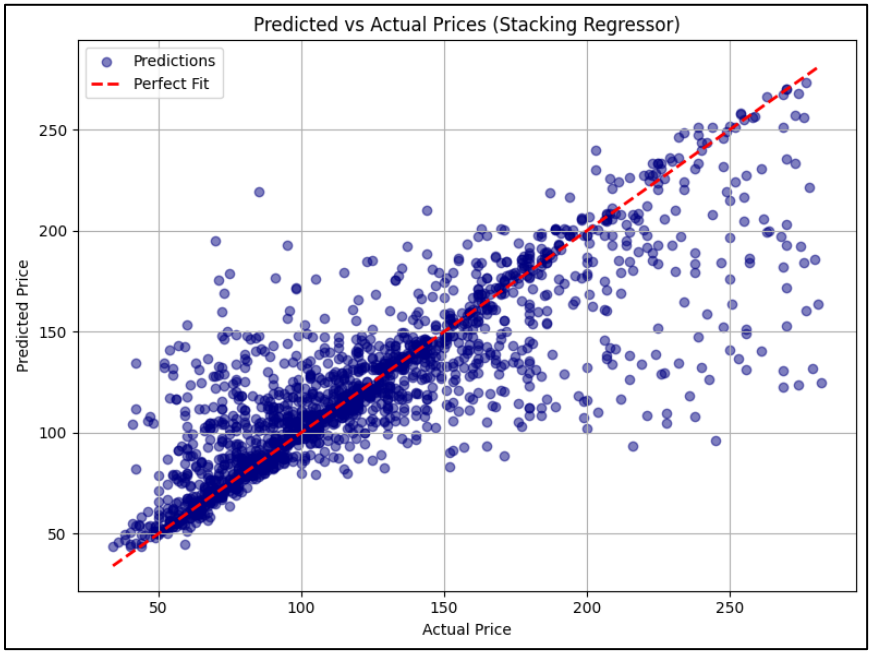
From the results table, we observed that tree-based ensemble methods significantly outperformed linear models.

Stacking achieved the best R^2 value (0.695) and lowest RMSE (28.797), suggesting it captured complex relationships in the data better.

Linear models (Linear, Ridge, Lasso) had much higher error rates and low R^2 (~0.28), indicating they were unable to model the underlying patterns effectively.

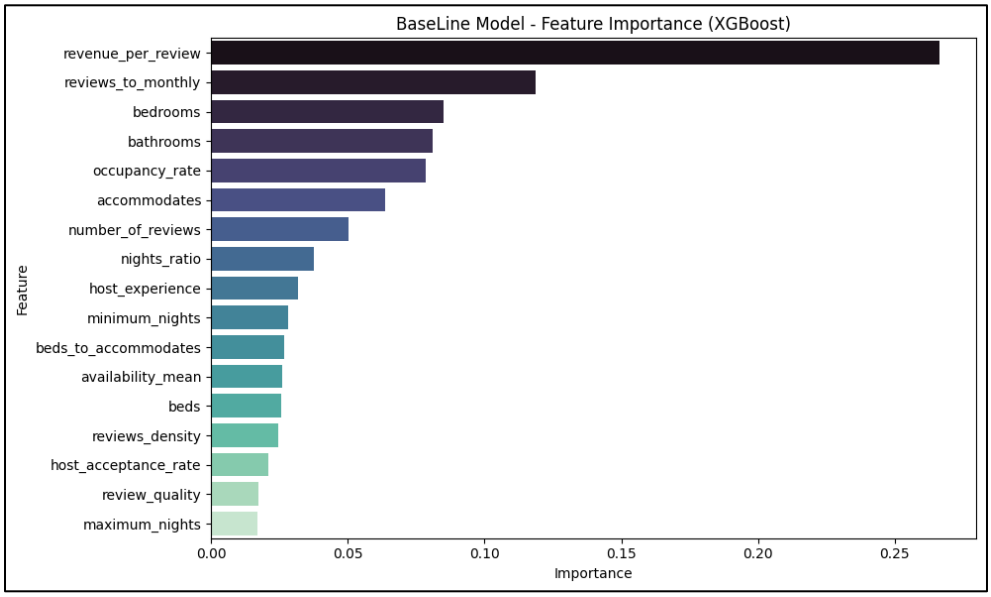
Visualizations

- **Actual vs. Predicted Scatter Plot**



The scatter plot of actual vs. predicted prices showed that ensemble models (especially Stacking and XGBoost) had predictions closely aligned to the actual prices, forming a dense cluster along the 45-degree perfect fit line. Linear models had wider spread and more systematic bias.

- **Feature Importance**



- **Revenue per Review** is by far the most influential feature, indicating that listings generating more income per review tend to have higher prices.
- Reviews to **Monthly Ratio, Bedrooms, and Bathrooms** also play strong roles, showing that size and review frequency impact pricing.
- **Occupancy Rate and Accommodates** contribute moderately, reflecting demand and guest capacity relevance.
- Features like **Review Quality, Host Acceptance Rate**, and **Maximum Nights** had a minimal impact in the baseline model.

Sentiment Analysis Modeling

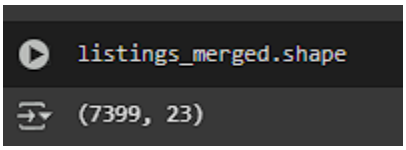
Sentiment Training

We trained the reviews dataset again but with a different and faster model called XLM Roberta from hugging face for 3 sentiments for each review and listing ID.

Negative	Neutral	Positive	listing_id
0.268325	0.319054	0.412621	**2737**
0.109466	0.168116	0.722419	**3079**
0.228036	0.275977	0.495987	**11834**
0.334289	0.368385	0.297326	**12398**
0.263797	0.426646	0.309557	**19965**

Dataset Merge

The preprocessed listings dataset and sentiment dataset were merged with an inner join based on ID and listings ID.



We have a merged dataset with 7399 rows which include engineered features with their respective Sentiments.

Sentiment Model

After incorporating sentiment features into the model (positive, neutral, negative sentiment scores), we retrained the models. These additional features provided deeper customer satisfaction signals, improving the models' ability to predict prices more accurately.

Train-Test Split	**80% training / 20% testing**
Features Used	**20 numerical predictors (3 sentiments)**
Target Variable	**Price**

Models evaluated:

- Linear Regression
- Ridge, Lasso
- Random Forest, Extra Trees
- Gradient Boosting, XGBoost
- Ensemble: Voting & Stacking Regressor

- **Voting Regressor** combines the predictions of several different models by taking the average of their outputs. We used Random Forest, Gradient Boosting, and Extra Trees as base models in the Voting Regressor.
- **Stacking Regressor** improves further by using a two-layer model: the first layer contains multiple models (Random Forest, XGBoost, Gradient Boosting), and a second-layer meta-model (Linear Regression) learns how to best combine their outputs. This blending improves overall accuracy.

Results

MODEL	RMSE	MAE	MSE	R^2
STACKING	18.034	9.786	325.224	0.869
XGBOOST	18.043	9.560	325.566	0.869
VOTING	20.146	11.220	405.853	0.837
RANDOMFOREST	23.062	12.660	531.843	0.786
EXTRATREES	23.413	13.327	548.183	0.780
GRADIENTBOOSTING	23.641	14.393	558.883	0.775
LINEAR	36.289	26.165	1316.867	0.471
RIDGE	36.293	26.167	1317.169	0.471
LASSO	36.387	26.239	1324.030	0.468

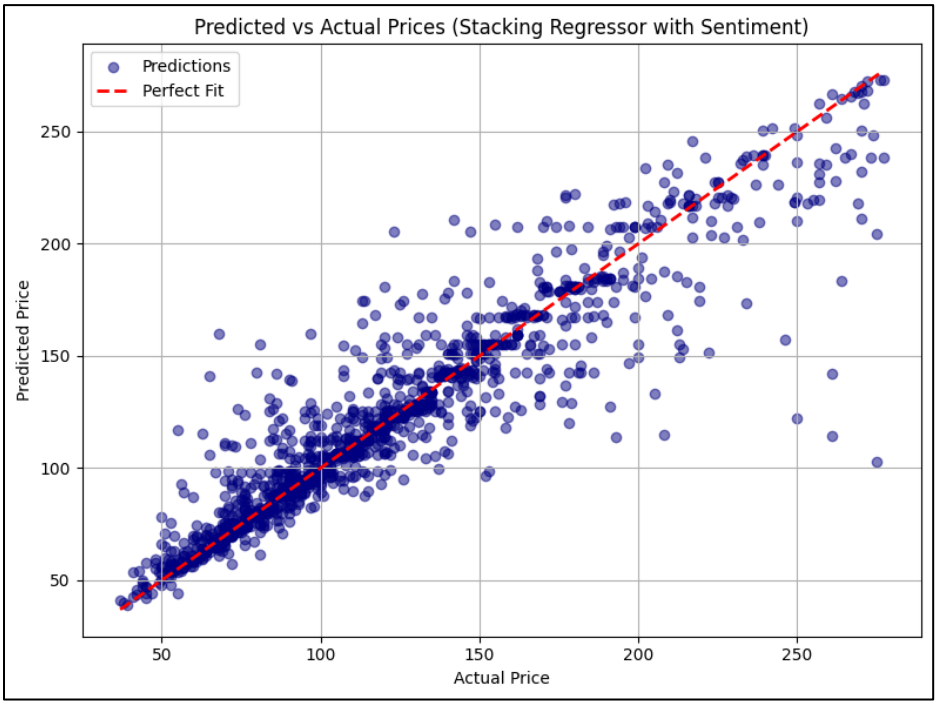
From the results table, we observed that tree-based ensemble methods significantly outperformed linear models.

Stacking achieved the best R^2 value (0.695) and lowest RMSE (28.797), suggesting it captured complex relationships in the data better.

Linear models (Linear, Ridge, Lasso) had much higher error rates and low R^2 ($\sim$0.28), indicating they were unable to model the underlying patterns effectively.

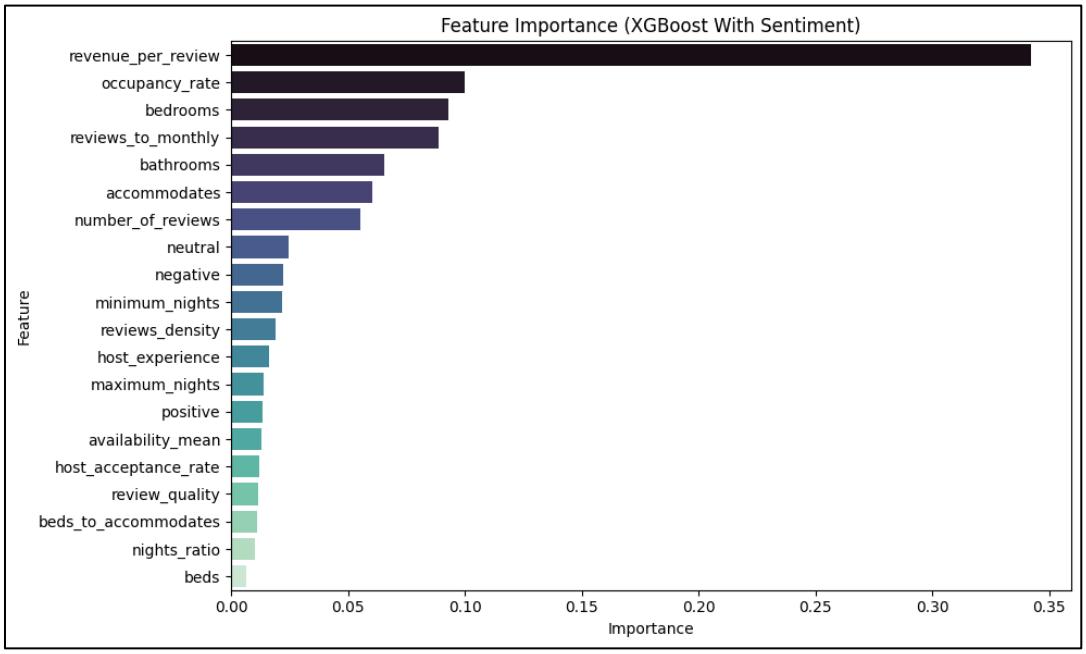
Visualizations

- **Actual vs. Predicted Scatter Plot**



The actual vs. predicted scatter plot for the sentiment model showed a tighter clustering along the ideal line compared to the baseline model, validating the improvement in predictive performance with sentiment data.

- **Feature Importance**

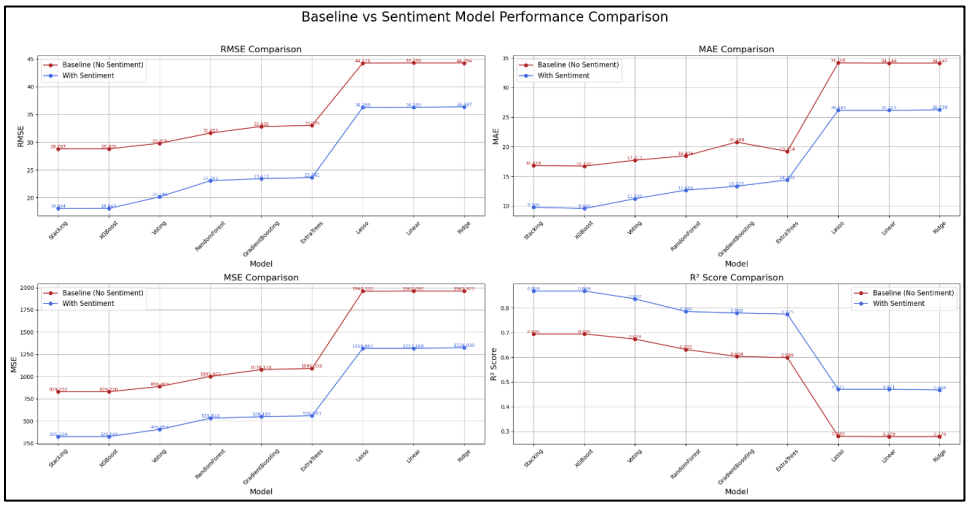


- **Revenue per Review** remains the top predictor, even more dominant than in the baseline model.
- Occupancy Rate, Bedrooms, and Reviews to Monthly follow, showing continued importance of property size and demand-related metrics.
- Sentiment features **(Neutral, Negative, and Positive) appear mid-ranking** — with **Neutral and Negative sentiment contributing more** to price prediction than Positive.
- Features like Beds, Nights Ratio, and Review Quality contributed minimally in the sentiment-enhanced model.

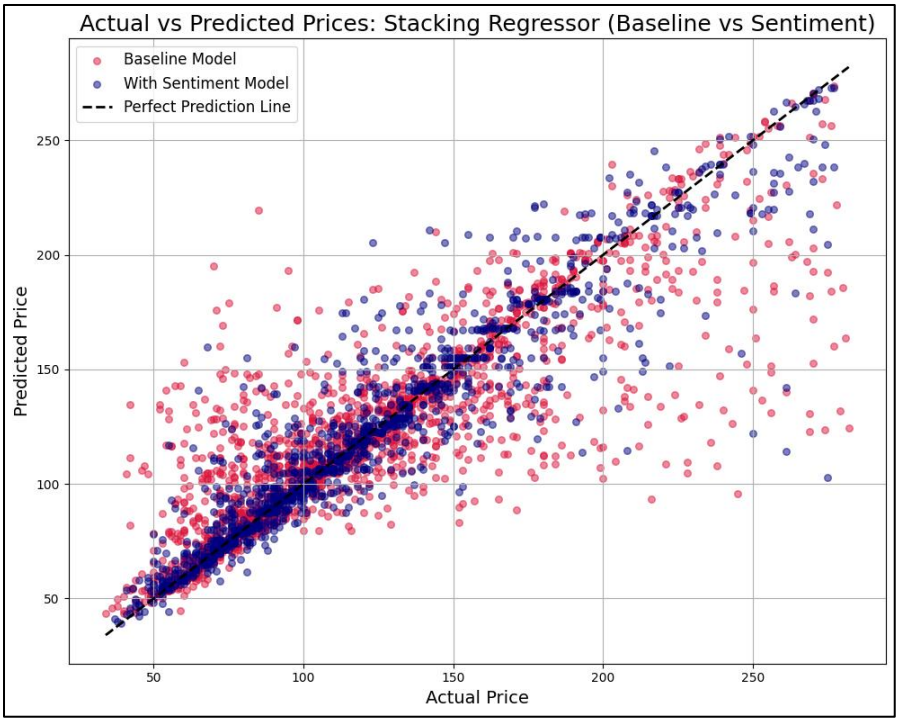
Model Comparison

Model Type	*Best R^2*	*Best RMSE*
Baseline	0.695	28.797
Sentiment	0.869	18.034

Incorporating sentiment features led to a **24.9% increase** in R^2 and a **37.4% decrease** in RMSE compared to the baseline model. This highlights the value of leveraging textual guest reviews alongside numerical listing features.



- The **blue dots** (with Sentiment Model) are **more tightly clustered** around the Perfect Prediction Line compared to the **red dots** (Baseline Model).
- This shows that **adding sentiment features** improved the model's ability to predict prices more accurately.
- The **Baseline Model** had greater spread and under/over-prediction errors, especially at higher price ranges.



- Across **RMSE, MAE, and MSE**, the **Sentiment Model consistently has lower errors** for all models compared to the Baseline.
- In the **R^2 Score Comparison**, the Sentiment models **achieved higher R^2** values across every model, especially for Stacking and XGBoost.
- **Biggest improvements** were observed for complex models like Stacking, Voting, and Random Forest.

Business Insights and Recommendations

Our analysis highlights how data-driven strategies—especially those incorporating guest sentiment—can enhance pricing accuracy and satisfaction across the Airbnb ecosystem. We offer tailored recommendations for three key stakeholders:

For Hosts

- **Prioritize guest experience:**
  Listings with highly positive sentiment consistently achieved higher prices. Hosts should invest in hospitality, cleanliness, and timely communication to improve review scores and justify premium pricing.
- **Monitor and manage reviews:**
  Even a few neutral or negative reviews can reduce perceived value. To avoid pricing penalties, hosts should regularly review feedback, respond constructively, and make targeted improvements.

- **Optimize revenue per review:**
  Rather than maximizing booking volume alone, hosts should aim for high satisfaction in each stay. Strong reviews contribute directly to higher revenue per review, driving long-term profitability.

- **Leverage high-occupancy strategies:**
  Our model showed that higher occupancy rates correlate with higher pricing. Hosts can improve occupancy by offering longer-stay discounts, flexible check-in/out times, and repeat guest incentives.

- **Adopt data-driven pricing tools:**
  Hosts should consider tools incorporating sentiment signals and real-time market data to adjust prices dynamically, particularly during holidays or high-demand periods.

For Airbnb (Platform)

- **Integrate sentiment into listing algorithms:**
  Airbnb can enhance the search experience by surfacing listings with consistently positive sentiment and flagging those with recurring negative themes.

- **Enhance pricing guidance tools:**
  By integrating sentiment-based predictors into Smart Pricing algorithms, Airbnb can offer hosts more accurate, performance-based pricing suggestions.

- **Support host improvement loops:**
  Highlight common areas of complaint (e.g., noise, temperature control, slow responses) from reviews to help hosts take action and reduce churn or guest dissatisfaction.

For Customers

- **Use sentiment as a quality indicator:**
  Guests can benefit by filtering listings with consistently positive sentiment in areas like cleanliness, location, and host interaction—features strongly tied to value and satisfaction.

Balance price with perceived value:
Our models suggest that slightly higher-priced listings with strong sentiment scores often deliver better overall experiences, reducing the risk of disappointment

Contributions

Team members contributed the following:

Common Contributions: All team members equally contributed to data cleaning, feature engineering, model development, exploratory data analysis, evaluation metrics setup, and presentation preparation.

Team Member	Contributions
AJ Payzant	Led the writing of both Milestone 1 and Milestone 2 reports, ensuring clarity, structure, and alignment with project goals. Assisted in early-stage code troubleshooting, managed organization of project files and deliverables, and co-led creation and delivery of the PowerPoint presentation.
Rohit Sarna	Responsible for the majority of the coding effort throughout the project, including data preprocessing, feature engineering, model building, and evaluation. Contributed to final model integration, validation, and visualization. Also helped present key findings during the final presentation.
Iseet Panja	Collaborated on building and refining code for data preprocessing, sentiment analysis, and modelling. Took the lead in designing and constructing the PowerPoint presentation, organizing slide content, visuals, and formatting.

Use of Generative AI Tools

We utilized the ChatGPT generative AI assistant for support in the following:

- Assistance with creation of evaluation function and model performance comparison visualizations
- Used for creating batch function for sentiment analysis (XLM Sentiment Model)
- Support in refining the wording, structure, and flow of the final report to maintain clarity and consistency.
- Occasionally, code-related suggestions were reviewed via AI for debugging or exploration methods. Final code was manually implemented and reviewed by group members.

No code generation or modeling was done solely by AI. All final implementations were coded, tested, and validated manually by the team.