UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

BUDT 758T Final Project Report Spring 2024

Chien-Jui "Jerry" Huang, Jie "Wendy" Yun Guan, Pravah Malunjkar, Navya Gehlot, Gautam Bhatia

Section 1: Team member names and contributions

a) Jerry:
   i) XgBoost Model
   ii) Logistic Regression Model
   iii) Cross-validation
   iv) XgBoost ROC curve generation
   v) Data cleaning/feature engineering
   vi) Report writing

b) Pravah:
   i) Ridge Regression Model
   ii) Team Lead: all project final prediction submissions
   iii) Finalize ensemble submissions
   iv) Feature engineering
   v) External data research and testing
   vi) Hyperparameter chart generation
   vii) Report writing

c) Wendy:
   i) Lasso Regression Model
   ii) Text mining
   iii) Feature engineering
   iv) External data research and testing
   v) Learning curve chart generation
   vi) Report writing

d) Navya:
   i) kNN Model
   ii) Random Forest Model
   iii) Feature engineering
   iv) Final code compilation
   v) Report writing

e) Gautam:
   i) Exploratory & explanatory data analysis (EDA) and insights
   ii) Feature engineering
   iii) Report writing

Section 2: Business Understanding

Executive Summary:
Data-driven decision-making has become essential in the quickly changing hospitality sector. Our project's main goal is to create a prediction model that leverages information from Airbnb listings to forecast whether or not a listing will receive a perfect rating. In order to improve visitor contentment, maximize listings, and eventually spur revenue growth, hosts, property managers, and platform developers can use this model as a strategic tool.

To achieve the goal and analyze our models, the evaluation metrics being used are true positive rate (TPR) and false positive rate (FPR). We aim to boost TPR given the 10% FPR limitation. The model will serve as a crucial tool for stakeholders such as hosts, leasing companies, or house-renting investors in several scenarios.

The Model's Relevance to the Business:
For Airbnb Hosts and Property Managers:

1. Targeted Improvements: Hosts can set priorities for particular enhancements to their communication, accommodations, or service by determining which elements have the greatest impact on attaining positive ratings.
2. Pricing Strategy: Listings that are expected to receive good scores may be able to support a premium pricing approach, which would boost profitability and keep the listing in a competitive position.
3. Operational Excellence: By providing hosts with guidance on best practices and areas that require attention, the model's insights can improve the visitor experience and raise the possibility that they will return.

For Airbnb Platform Developers:

1. Feature Development: The model's results can guide improvements to the platform, such as suggesting to hosts high-value additions or services, or even putting predictive scores right into the host dashboard.
2. Quality Control: The platform can maintain strict guidelines and user confidence by anticipatorily suggesting remedies for listings that are expected to underperform in terms of guest satisfaction.
3. Marketing and Promotions: High-rated lodgings can draw in more customers by emphasizing their listings in search results and marketing materials.

Business Actions Based on Model Output:

1. Personalized Recommendations: Personalized reports that offer specific steps to increase the chances of each listing receiving a perfect rating can be generated for each listing based on the prediction findings.
2. Alert Systems: When a listing approaches the threshold for a perfect score, put alert systems in place to trigger prompt updates and inspections from the host.
3. Performance Tracking: To monitor the success of improvements made by hosts over time and make necessary adjustments.

Value Generation from the Predictions
The ultimate value of our predictive model lies in its potential to increase both guest satisfaction and host profitability:

1. Higher Occupancy Rates: Listings with a strong rating history have a greater chance of seeing higher occupancy rates, which translates to higher revenue for hosts.
2. Improved visitor Experience: Hosts can increase the overall quality of the visitor experience by aiming for perfect ratings. This is important since it helps with customer retention and word-of-mouth marketing.
3. Reputation Management: A host's marketability and success can be greatly increased by being able to forecast and optimize listing performance on a platform where reputation is crucial.

To sum up, the predictive model created using data from Airbnb listings helps with operational decision-making and can be a strategic tool for gaining a competitive edge. Businesses and individuals within the Airbnb ecosystem can use this model to inform decisions that support long-term growth and client pleasure.

Section 3: Data Understanding and Data Preparation

Table 1: Feature names and a brief description of the data cleaning process as well as corresponding code lines in the final code.

ID	Feature Name	Brief Description	Code Line
1	bed_type	original feature from dataset	33
2	cancellation_policy	regroup into fewer levels	34,35
3	license	replace NA with 0; otherwise, 1	36,37
4	market	replace NA with OTHER	38,39
5	property_type	regroup into fewer levels	40-47
6	room_type	original feature from dataset	48
7	state	original feature from dataset	49-56
8	accommodates	replace NA with median	57
9	availability_365	if it's > 0, then 1; otherwise 0	58
10	bathrooms	replace NA with median	59
11	bedrooms	replace NA with median	60
12	beds	NA with median	61
13	cleaning_fee	if it's > 0, then 1; otherwise, 0	62
14	extra_people	replace NA with 0 and take logarithm	63
15	host_acceptance	regroup host_acceptance_rate_ into fewer levels	64-71
16	log_host_listings_count	replace NA with 0 and take logarithm	72
17	HLC_bin	discretize log_host_listings_count	73,74
18	host_response	regroup host_response_rate into fewer levels	75-78
19	log_host_total_listings_count	replace NA with 0 and take logarithm	79,80
20	log_maximum_nights	replace NA with 0 and take logarithm; regroup into fewer levels	81-85
21	log_minimum_nights	replace NA with 0 and take logarithm; regroup into fewer levels	86-91
22	monthly_price	if monthly price < price*30, then 1; otherwise, 0	92-94
23	log_price	replace NA with median and take logarithm	95,96
24	security_deposit	if it's > 0, then 1; otherwise, 0	97-106
25	host_since_days	day difference between the system date and since days	107,108
26	description_length	replace NA with 0 and count character number	109
27	pets_allowed	if pets allowed, then 1; otherwise, 0	110-113
28	neighborhood_overview	replace NA with none and convert to lowercase	114,115
29	is_market_enterain	replace NA with none, lowercase, and check if grocery stores or entertainment nearby	116-119
30	is_interaction	replace NA with 0; otherwise, 1 to check if there are contacting ways	120
31	house_rules	replace NA with 0; otherwise, 1 to check if there are rules	121,122,133
32	house_rules_no_count	replace NA with none, lowercase, and count how many "no" and take logarithm	123-128
33	summary_length	replace NA with 0 and count character number	131
34	host_about	replace NA with 0; otherwise, 1	132
35	amenities_count	Counts by splitting by on ','	136
36	host_verfications_count	Counts by splitting by on ','	137

37	timezone		145-167
38	amenities_result	Text data is preprocessed by replacing separators, converting to lowercase, removing numbers, punctuation, and stopwords, and then tokenizing. The cleaned text is then vectorized into a Document Term Matrix after pruning the vocabulary based on frequency thresholds (75 most frequent words)	253
39	features_result	Same as amenities_result	256
40	description_result	Same as amenities_result	259
41	space_result	Same as amenities_result	262
42	summary_result	Same as amenities_result	265
43	name_result	Same as amenities_result	268
44	neighborhood_result	Same as amenities_result	271
45	access_result	Same as amenities_result	274
46	host_about_result	Same as amenities_result	277
47	interaction_result	Same as amenities_result	280
48	notes_result	Same as amenities_result	283
49	transit_result	Same as amenities_result	286
50	house_rules_result	Same as amenities_result	289
51	top10happy	States ranked top 10 in *HappiestStatesCommunityAndEnvironmentRank* from the external dataset are marked 1, else 0.	365-373
52	top10pop	States ranked top 10 in *State Population 2024 (pop_2024)* from the external dataset are marked 1, else 0.	382-390
53	top10weather	States ranked top 10 in *StatesWithBestWeatherNumofExremeWeatherEvents* from the external dataset are marked 1, else 0.	400-408
54	crime_rate	States categorized as *"Very Low Crime", "Low Crime", "Moderate Crime", "High Crime", and "Very High Crime"* from the external dataset, else “MISSING” if NA	425-431

Feature Exploration:

Correlation between Numerical Variables

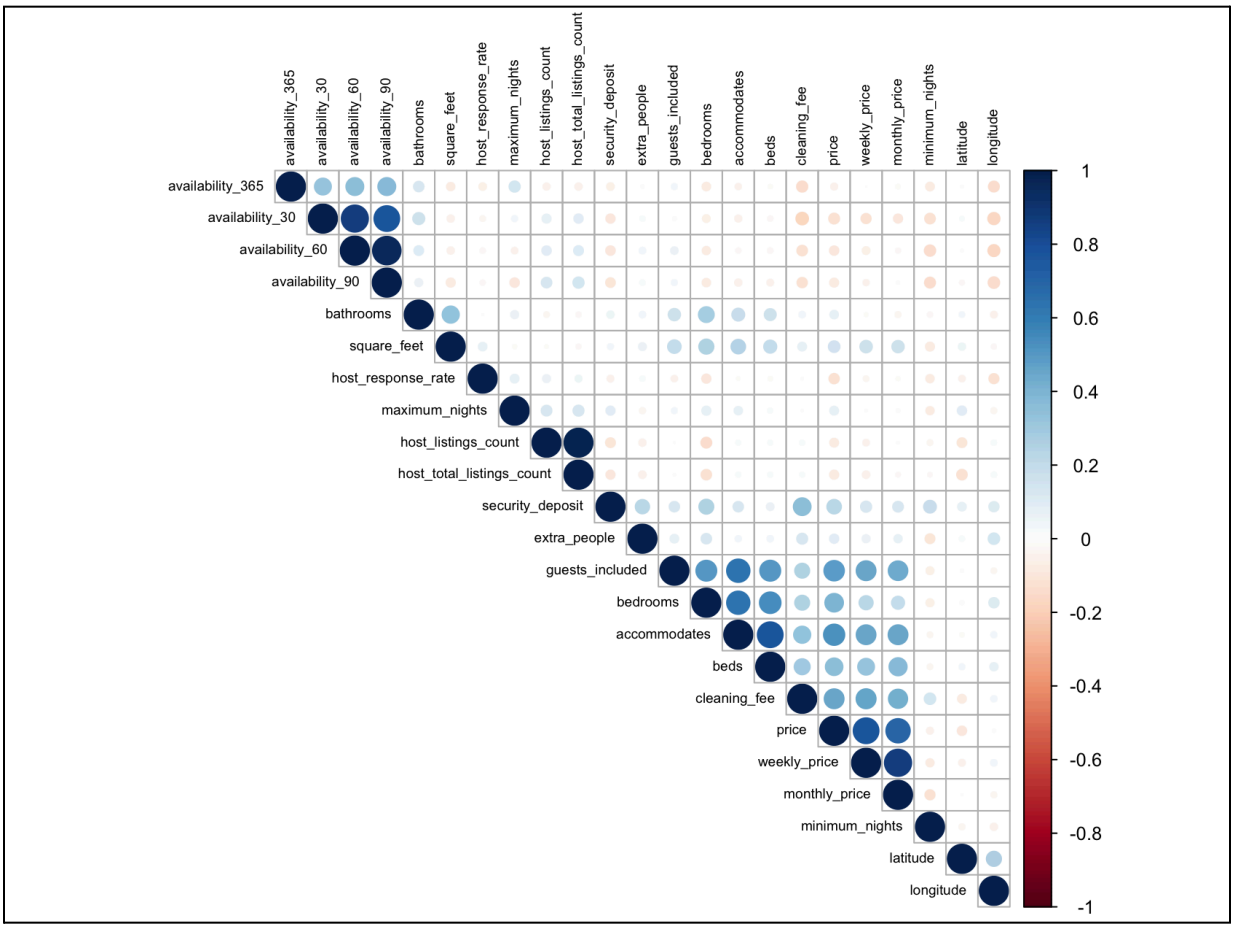

Figure 1: Correlation plot between raw numerical features

The correlation plot of numerical variables in the Airbnb dataset offers insightful glimpses into the interconnectedness of features that likely possess strong predictive power regarding listings' performance and guest satisfaction. Key correlations such as those between accommodates and both bedrooms and beds indicate that the physical capacity of a listing is crucial for attracting groups and determining the comfort level, directly impacting guest reviews and booking decisions. Similarly, the tight correlation between pricing variables (price, weekly_price, monthly_price) and between various availability metrics (30, 60, 90, 365 days) suggests that hosts' strategies around pricing and availability management are pivotal in influencing both short-term and long-term bookings. These relationships underscore the importance of strategic feature management and presentation in optimizing a listing's appeal and profitability, serving as potential predictors of a listing's success in achieving high occupancy and favorable ratings.

1. market

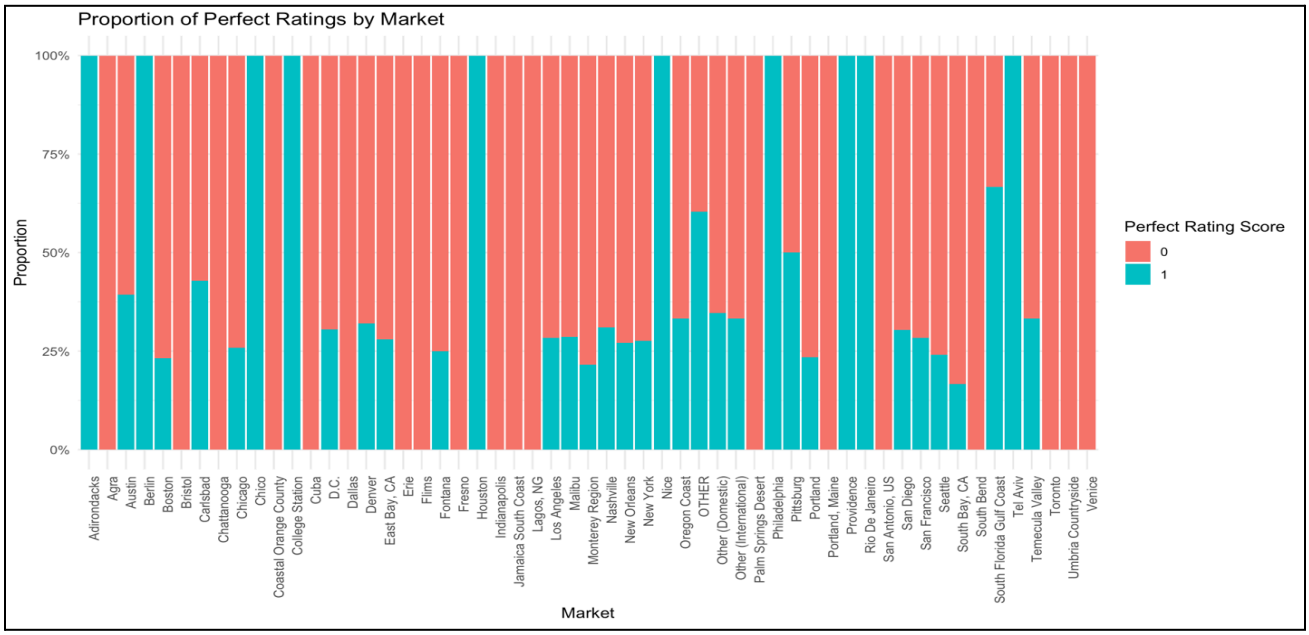

Figure 2: Stacked bar chart of perfect rating scores across markets

The distribution of perfect ratings across different markets reveals disparities, with places like "Rio de Janeiro" and "Houston" achieving high scores, while "Berlin" and "Agra" fare less favorably. These variations indicate that markets with higher proportions of perfect ratings are likely to exhibit stronger customer satisfaction and competitive service standards. Local factors such as customer expectations, regional hospitality practices, and market competition significantly influence the rating patterns, while cultural norms also shape how lenient or critical customers are in their reviews. By identifying markets with lower perfect ratings, businesses can strategically target service improvements and marketing efforts to enhance customer satisfaction.

2. host_response

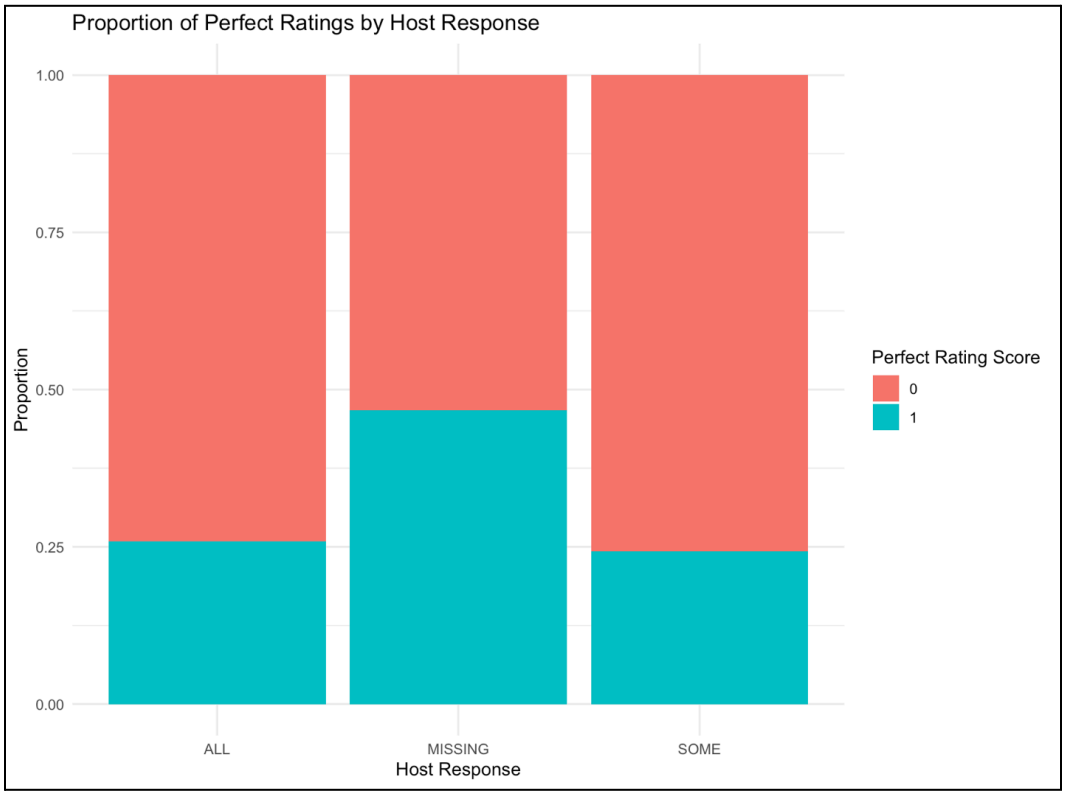

Figure 3: Stacked bar chart of perfect rating score grouped by host response category

The graph illustrates that listings with complete host responses tend to have a slightly higher proportion of perfect ratings, while those with some responses have a noticeably lower proportion. This trend suggests that full host responsiveness is predictive of guest satisfaction. Active host engagement, through transparent and accurate communication, helps establish trust, manage guest expectations, and improve overall experiences. By providing comprehensive responses, hosts create a reliable environment that meets or exceeds guests' expectations, thereby increasing the likelihood of perfect ratings.

3. host_since_days

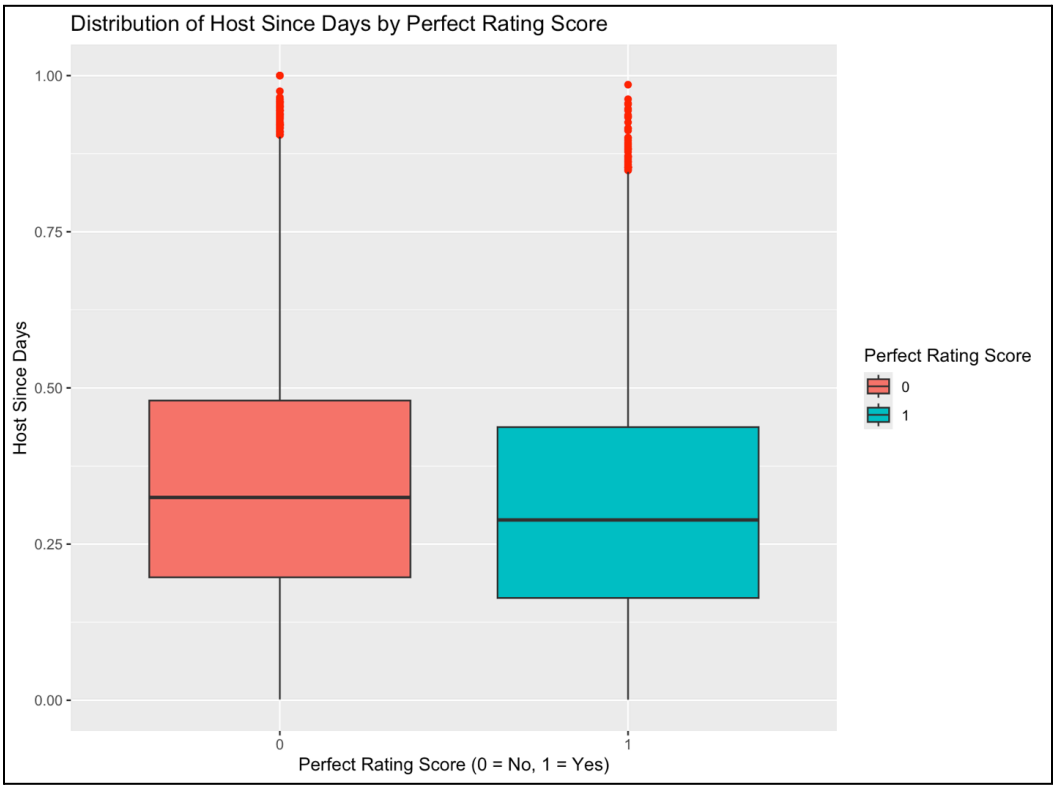

Figure 4: Boxplot of the distribution of days since the host's first listing, grouped by perfect rating score.

Given the difficulty in achieving perfect ratings, the box plot shows that less experienced hosts (1 - perfect rating) have a higher proportion of perfect scores compared to those who have hosted for longer periods (0 - not perfect). This trend suggests that newer hosts may initially benefit from guests' leniency or goodwill when ratings are given. However, as hosts accumulate more hosting experience, maintaining the high standards needed for perfect ratings becomes challenging due to evolving expectations, increased scrutiny, or a larger volume of reviews. Therefore, it's challenging for longer-term hosts to consistently achieve perfect scores.

4. summary_length

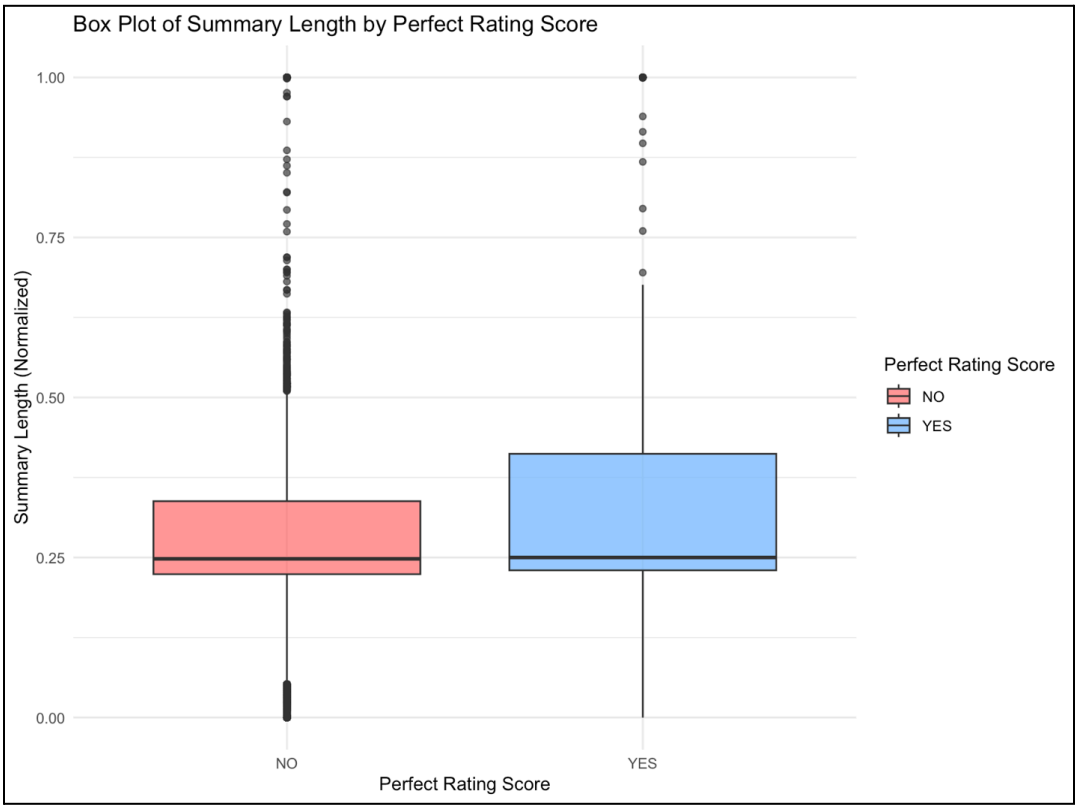

Figure 5: Box plot of the distribution of the length of the summary for each listing grouped by perfect rating score.

The box plot indicates that listings with perfect ratings (1) generally have consistent summary lengths & fewer outliers, while those without perfect ratings (0) display a wider range of extremes. This pattern suggests that maintaining summaries within a consistent range helps meet guest expectations and fosters higher ratings. By providing clear, detailed information that aligns with what guests are seeking, hosts minimize surprises and ensure positive experiences. Thus, achieving a consistent summary length is predictive of perfect scores due to its ability to set realistic expectations and communicate effectively.

5. monthly_price

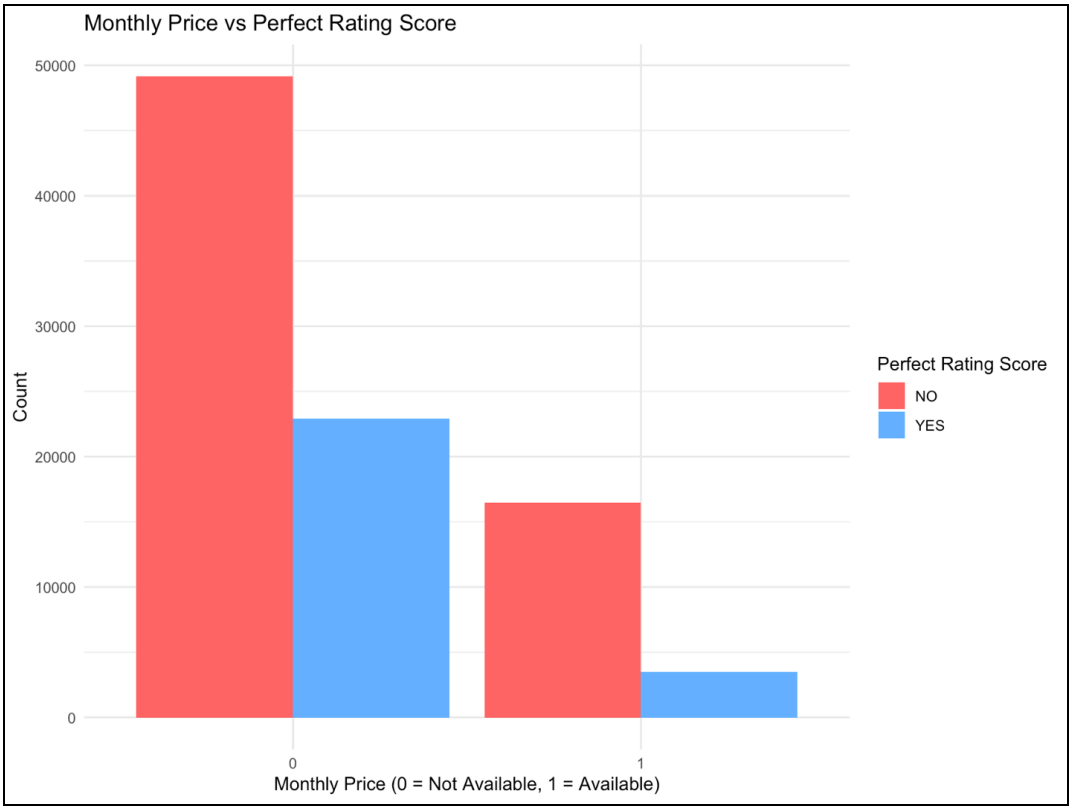

Figure 6: Bar graph comparing the count of hosts with and without perfect rating scores, grouped by the availability of monthly pricing.

The bar chart shows that listings offering monthly pricing are more likely to receive perfect ratings compared to those without. This pattern suggests that guests staying for a longer period tend to have a better experience, possibly because they value the stability and good pricing for longer stays. As a result, they are more likely to leave a perfect review.

6. availability_365

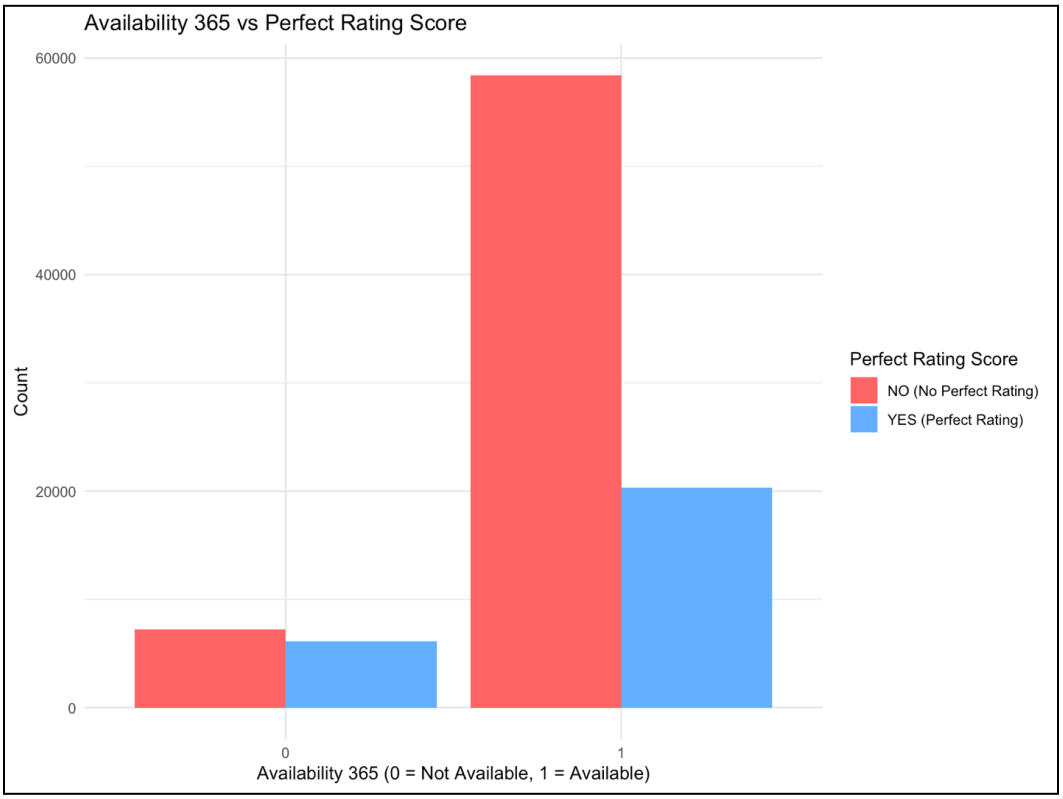

Figure 7: Bar graph comparing counts of hosts with and without perfect rating score, grouped by the availability over the next 365 days.

The bar chart shows that listings available all year round (1) have a significantly higher count of non-perfect ratings (NO) compared to perfect ratings (YES), while those not available throughout the year (0) have a similar count for both perfect and non-perfect ratings. This trend suggests that while year-round availability attracts a broader guest base, the challenge of consistently maintaining high standards across a diverse guest profile may result in fewer perfect scores. Limited availability could be linked to a more focused guest experience, where higher quality and attention to detail lead to better ratings.

7. description_length

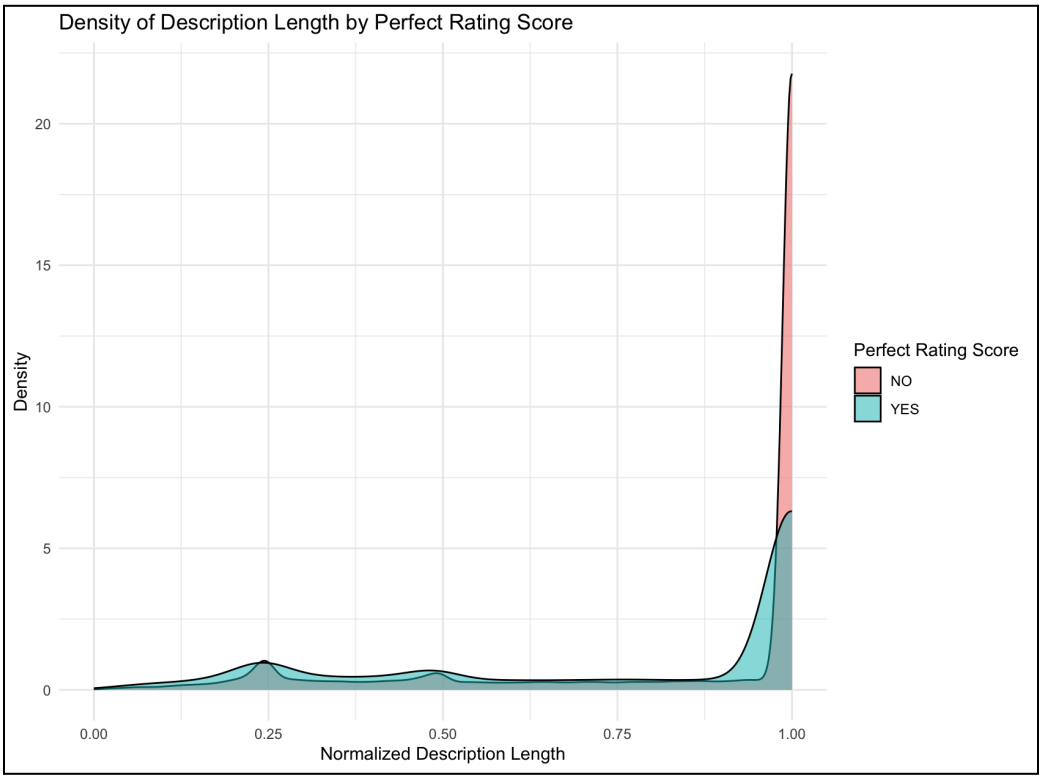

Figure 8: Density plot displaying the distribution of normalized description length, comparing listings with and without perfect rating score.

The density plot reveals that both listings with and without perfect ratings tend to have longer description lengths, yet listings without perfect ratings (NO) show a higher density near the maximum length, suggesting that overly lengthy descriptions may not always correlate with higher satisfaction. In contrast, listings with perfect ratings (YES) feature a more balanced distribution across the description length spectrum. This suggests that while comprehensive details are necessary, clarity and conciseness are also crucial for achieving perfect ratings, indicating that the most effective descriptions are those that provide essential information in a clear and engaging manner.

8. cancellation_policy

Figure 9: Stacked bar charts showing the perfect rating score grouped by the strictness of the listing's cancellation policy

The bar chart illustrates that listings with flexible cancellation policies tend to receive a higher proportion of perfect ratings (1), while those with strict policies see a greater share of non-perfect scores (0). Moderate policies fall between the two, with a more balanced distribution. Flexible policies provide guests peace of mind and adaptability, fostering a positive experience and leading to more perfect ratings. Moderate policies offer a compromise, giving some leeway for changes without being too lenient, resulting in slightly more perfect rating scores than strict cancellation policies but not much. Stricter policies, however, can lead to dissatisfaction due to limited options for guests when plans change, reducing the likelihood of perfect scores. Overall, providing flexibility helps enhance guest satisfaction.

9. log_price

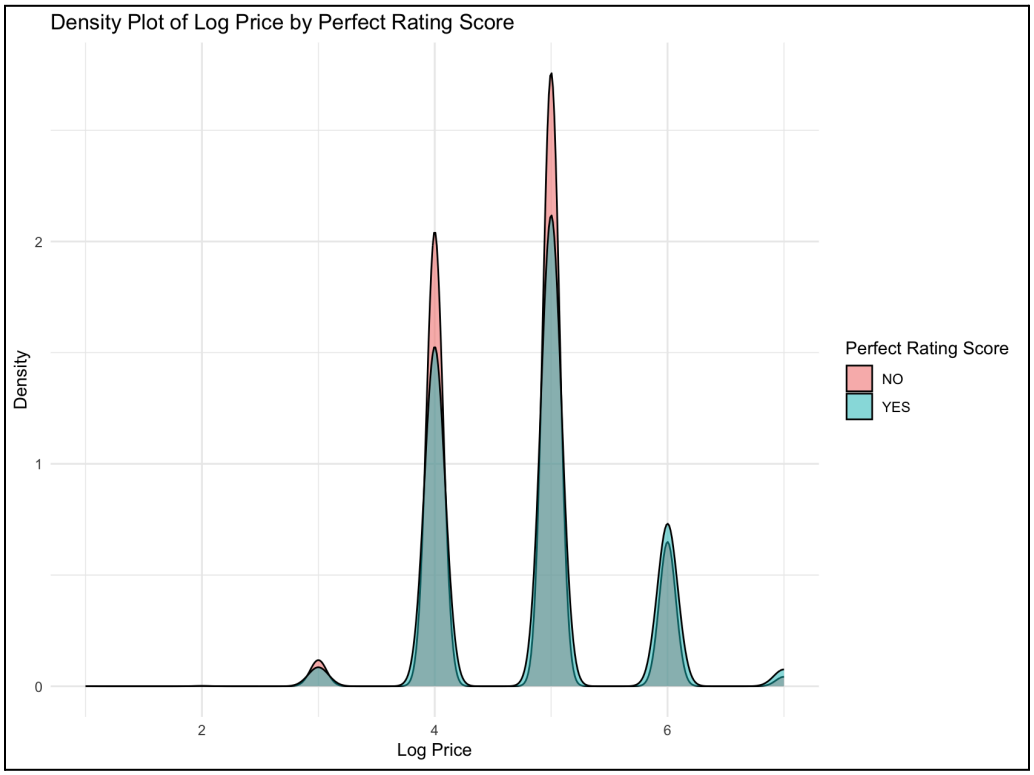

Figure 10: Density plot illustrating the distribution of log-transformed prices for hosts, segmented by their perfect rating score

The density plot of log prices shows that listings with perfect ratings (1) tend to have a wider distribution of prices compared to those without (0), although both groups peak around a similar price range. This suggests that listings with perfect ratings can achieve high scores across various pricing levels. The key insight is that aligning accommodation quality with guest expectations at these price points is essential for securing perfect ratings, as guests prioritize value and quality that matches the price they are paying.

10. log_minimum_nights

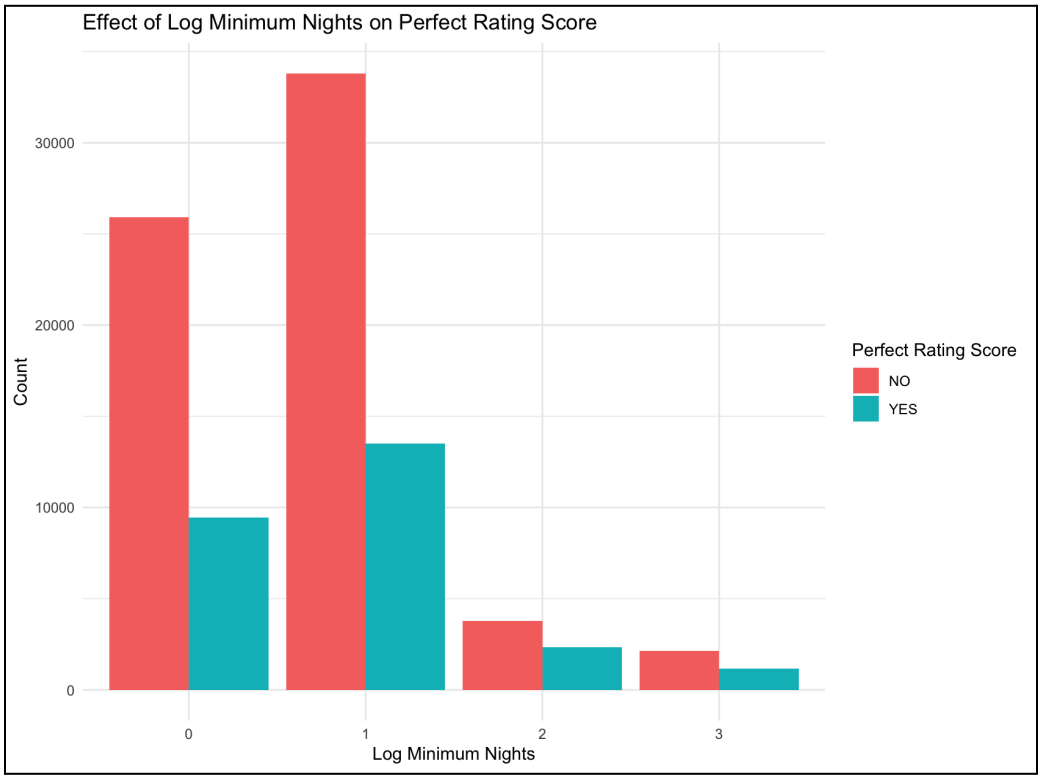

Figure 11: Bar chart showing the count of perfect rating scores grouped by the log-transformed minimum nights that are required for a booking.

The bar chart shows that listings with lower log values for minimum nights required (0 and 1) have higher counts of perfect ratings (YES) compared to those with higher values. This trend suggests that listings with shorter minimum night requirements are more likely to receive perfect scores, as they provide the flexibility that aligns with guests' needs for shorter stays. Listings with higher minimum night requirements may deter guests from seeking more flexible options, resulting in fewer perfect ratings.

Section 4: Evaluation and Modeling

During the model selection phase, a methodology involving a 10% subsample of the original training dataset and employing 5-fold cross-validation was implemented. The reason for selecting the 10% subsample was due to three concerns related to computation intensity.

1. Training Time: Training models on large datasets and using complex algorithms require significant computational resources and time. This was the bottleneck we encountered, especially when we experimented with multiple models and hyperparameters. In terms of training duration, we observed that logistic regression, Lasso, and Ridge models were relatively quicker to train, each earning a rank of 1 for efficiency. In contrast, kNN models were significantly slower, ranking 4, while Xgboost and Random Forest models were the most time-intensive, ranking 5 and 6 respectively. These rankings reflect not only the computational intensity required by each model type but also highlight the trade-offs between training time and model complexity in our project.
2. Memory Requirements: Storing and manipulating large datasets in memory can be resource-intensive. Subsampling the data reduces memory usage, making it more manageable for training and evaluation.
3. Computation Power: Running cross-validation on a full dataset with multiple folds can require substantial computation power, which is not practical.

The generalization performance in the below chart revealed that logistic regression yielded the highest TPR on the validation set across the five folds. Consequently, initial attention was directed solely towards logistic regression.

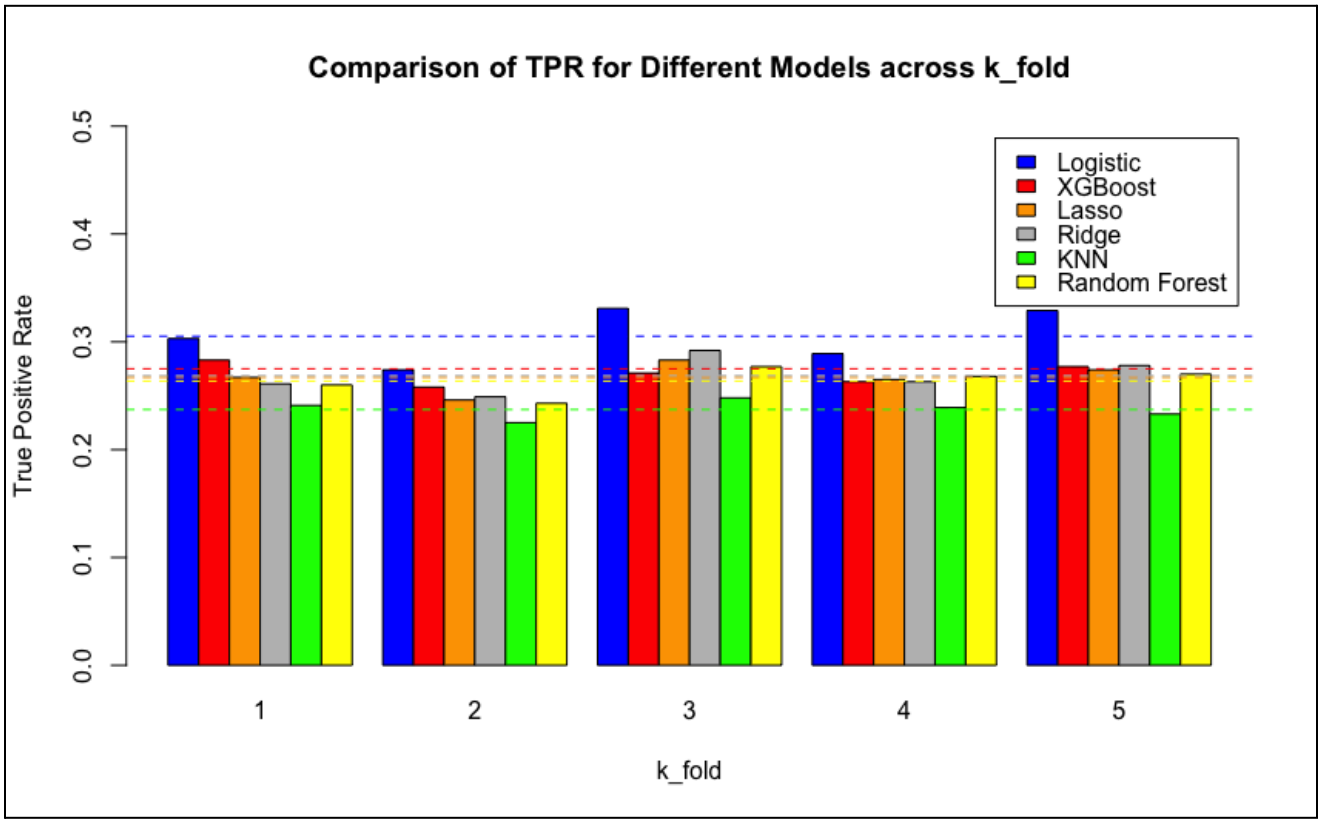

Figure 12: Bar chart showing the true positive rate performance of various predictive models (Logistic, XGBoost, Lasso, Ridge, kNN, and Random Forest) across five folds of cross-validation.

However, the utilization of a 10% sample size has the risk of not providing adequate representativeness for certain models to perform optimally, and this is the exact problem we encounter after the second prediction evaluation. Hence, our team decided to pursue a diversified approach, encompassing logistic regression, lasso

regression, ridge regression, kNN, random forest, and XGBoost to do ensembling. Each model was trained using a simple train/validation split ratio of 70% for training and 30% for validation. Subsequently, the TPR on the validation set was used to determine the optimal model specification which was XGBoost in our case.

Table 2: Overview of 6 different predictive models (Logistic, XGBoost, Lasso, Ridge, kNN, and Random Forest) and their performance during both training and validation phases. In addition the code line is provided for each model as well as the library and functions used.

Model	Train		Validation		Code line	Function/ Library used
	TPR	FPR	TPR	FPR		• tidyverse • caret • class • ggplot2 • reshape2 • dplyr • glmnet • pROC • text2vec • tm • SnowballC • randomForest • gbm • ISLR • corrplot • tidytext
logistic	0.327	0.096	0.293	0.088	856-832	
Lasso	0.323	0.095	0.310	0.094	1079-1182	
Ridge	0.331	0.095	0.324	0.094	1189-1356	
kNN	0.258	0.095	0.229	0.097	981-1015	
Random Forest	0.261	0.062	0.235	0.067	1020-1073	
XGboost	0.356	0.081	0.328	0.089	836-937	

After trying various model families, we settled on XGBoost with 150 trees and a depth of 10, based on its superior performance in terms of TPR and FPR compared to other model families. The XGBoost model incorporated features listed in the following table. During training, it achieved a TPR of 0.356 and an FPR of 0.082 on the train set, while on the validation set it attained a TPR of 0.328 and an FPR of 0.089. We determined its superiority through the mentioned evaluation metrics comparison across different models. The line numbers in our R code where the final predictions were generated for the contest are as follows: 933-937

Table 3: Features used in our final predictive model, categorized by feature type.

Categorical	Numerical	Text
• state • market • property_type • room_type • bed_tye • monthly_price • security_deposit • cleaning_fee • license • cancellation_policy • host_acceptance • HLC_bin • host_response • pets_allowed	• bathrooms • bedrooms • beds • extra_people • availability_365 • log_host_total_listings_count • log_maximum_nights • log_minimum_nights • log_price • host_since_days • description_length • house_rules_no_count • summary_length	• amenities • features • description

• is_market_entertain • is_interaction • host_about • house_rules • top10happy		

In this analysis, the Lasso and Ridge models were evaluated using error vs. model complexity plots (Figure 13) to optimize the balance between complexity and error rates. Their performance was further assessed through learning curves (Figure 14), examining the relationship between training size and model accuracy. For our final XGBoost model, evaluation was conducted using a ROC curve, providing a comprehensive assessment of model effectiveness across various thresholds. These evaluations help determine the most appropriate modeling approaches for our dataset, ensuring optimal performance and accuracy.

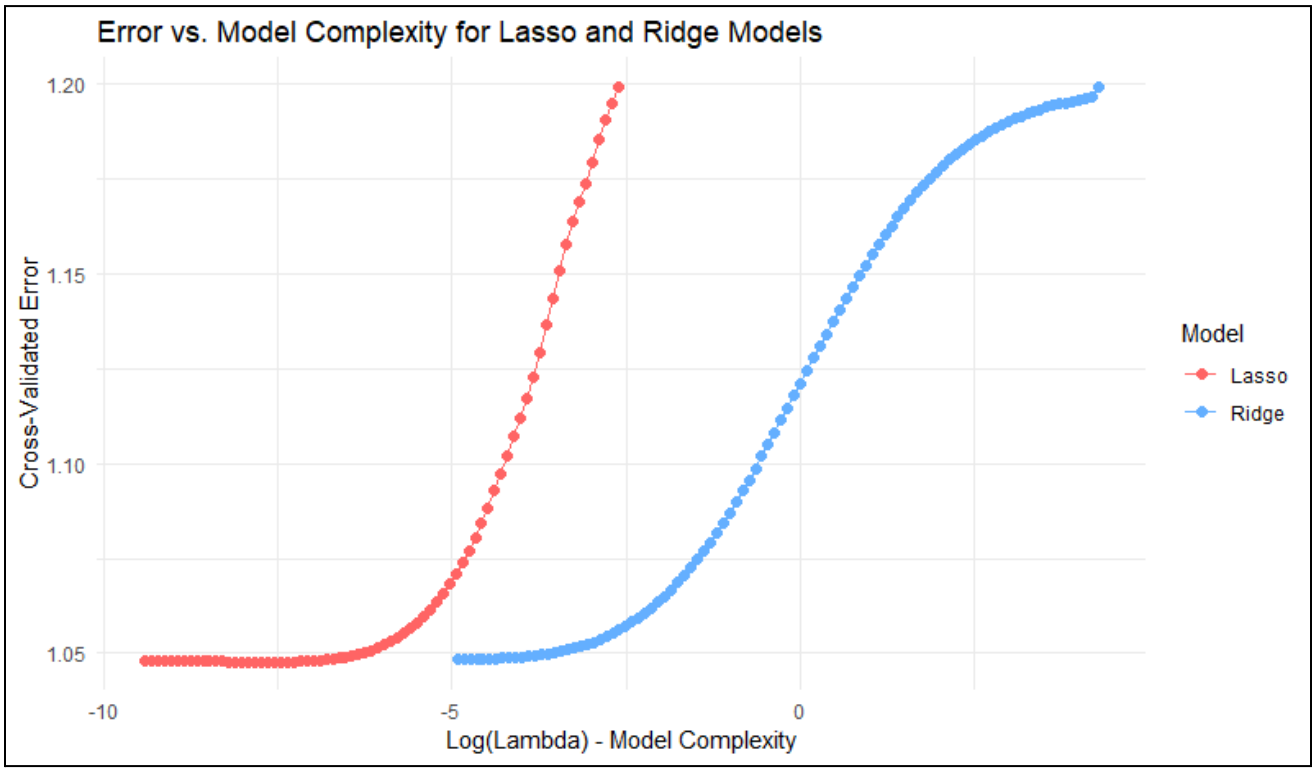

Figure 13: Plot illustrating relationship between model complexity (logarithm of lambda) and cross-validated error for Lasso and Ridge models.

Lasso Model: When λ is very low, the Lasso model begins with large cross-validated errors, suggesting that the model may be overfitting as a result of excessive complexity (too many features making a significant contribution). The error falls off quickly as λ rises, reaches a minimum, and then begins to rise gradually. This increase at high λ values points to underfitting, in which the model is oversimplified by excessive regularization and ignores important variables.
Ridge Model: Like Lasso, the Ridge model shows possible overfitting at lower λ values, when errors are larger at first. Up to a certain point, the error decreases as λ increases, suggesting an improvement in the model's performance. As λ increases, the Ridge curve, in contrast to Lasso, exhibits a smoother transition into larger errors, indicating a gradual shift towards underfitting.
We may assess which model (Lasso or Ridge) works better for your particular dataset at different regularization levels by comparing the two curves. Plotting can be used to determine the optimal regularization type and intensity (λ value) for a given situation.

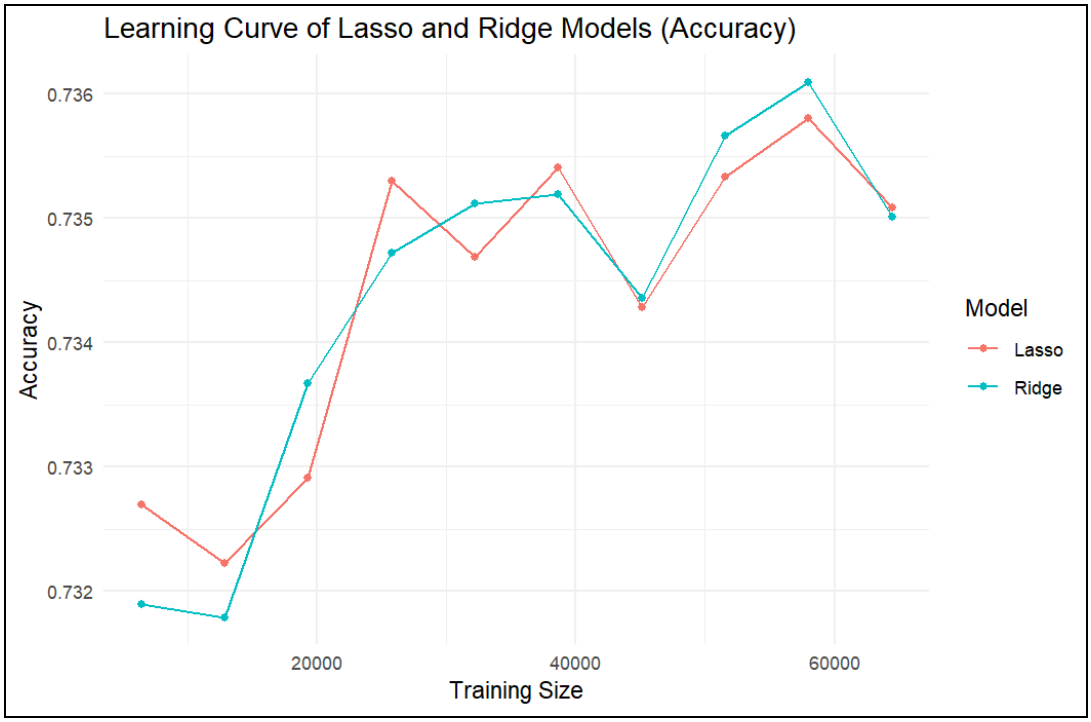

Figure 14: Line plot demonstrating the learning curves of lasso and ridge regression models, graphing the model accuracy as a function of training size.

The learning curve graph for the Lasso and Ridge models demonstrates how model accuracy changes with different training sizes. The accuracy of both models shows significant improvement as training size increases from about 13,000 to 26,000, reaching a peak accuracy of approximately 0.7353 for Lasso and 0.7347 for Ridge. Beyond 26,000 data points, the accuracy continues to show slight improvements, albeit with fluctuations that hint at diminishing returns from increasing the training data size. These observations underscore the constraints of data scalability in boosting model performance. Furthermore, our analysis is confined by the data limit of 65,000 points (70% of total train data), restricting deeper exploration into the scalability potential. Due to computational constraints, our learning curve concentrated on Lasso and Ridge models, precluding us from extending to more computationally demanding models like XGBoost, Random Forest, and kNN, thereby limiting our comparative analysis across a broader array of model types.

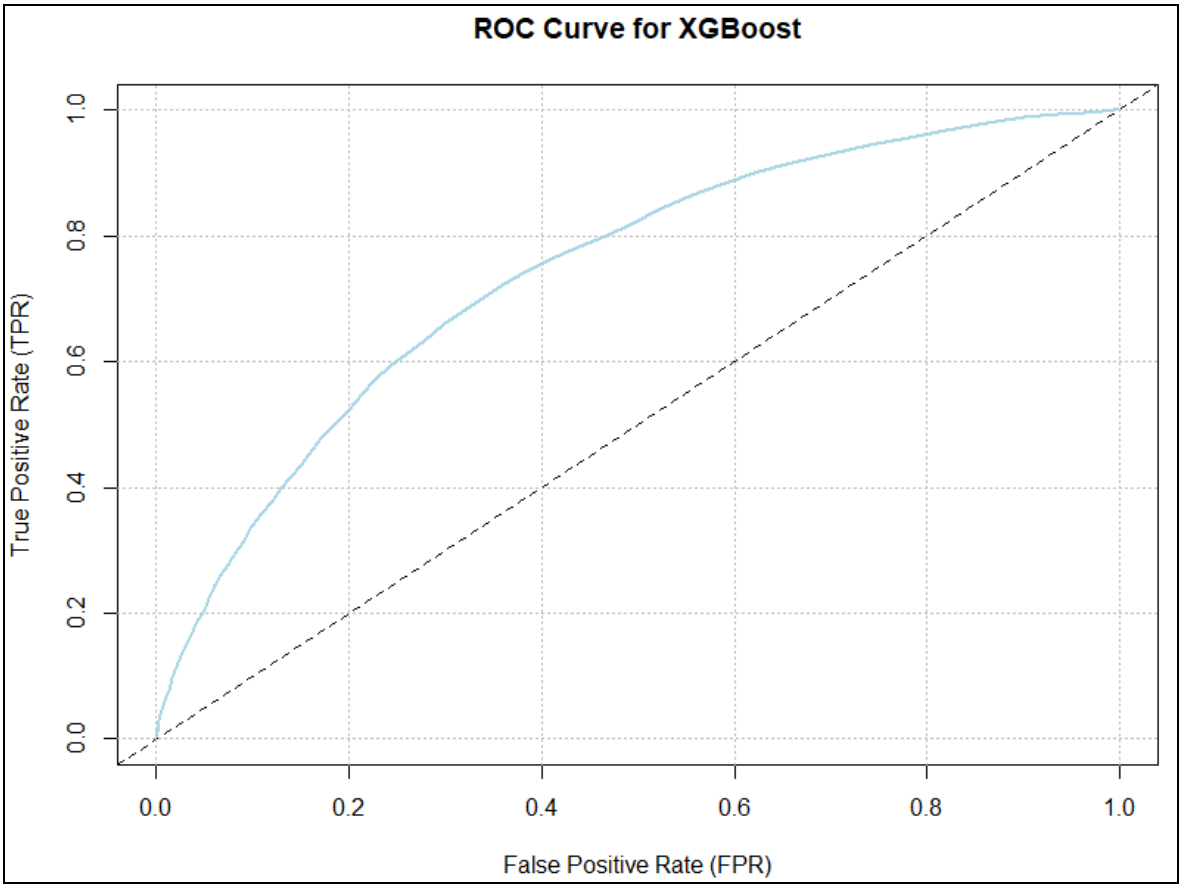

Figure 15: ROC curve for final prediction XgBoost Model.

The ROC curve depicted in Figure 15 illustrates the performance of our XGBoost model in terms of its ability to distinguish between classes. This curve plots the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. The model demonstrates a satisfactory balance between sensitivity and specificity, particularly relevant given our specified criterion for model selection which includes maintaining a FPR below 0.1. In this context, our XGBoost model achieves an FPR of 0.089, which is within the FPR limit, and a TPR of 0.328, indicating a moderate ability to correctly identify positive cases while effectively limiting the rate of false positives. This model has the best performance among all models that were created and lead to the decision of the final model being the XgBoost model. This performance supports the XGBoost model as a viable option under the constraints of our project requirements, focusing on minimizing the FPR while maintaining a reasonable TPR.

Section 5: Reflection

1. What did our group do well?
   The first meeting of our group was crucial in establishing the framework for our group endeavor. We began with a thorough feature analysis to give everyone a chance to voice their opinions and voice concerns on the unique qualities, intricacies, and data integrity of each feature that was being considered. By thoroughly examining the characteristics, we were able to gain a comprehensive understanding and make well-informed decisions about data cleansing and enhancement through the integration of additional features. This laid a strong foundation for our early modeling endeavors. Building on these foundational elements, we judiciously distributed modeling duties based on each team member's area of expertise, guaranteeing a productive division of labor and maximizing our unique advantages.
   i. Jerry decided to apply the XgBoost and Logistic Regression models, which are well-known for their reliable results in binary classification issues.
   ii. Pravah concentrated on Ridge Regression, a method that penalizes the size of coefficients and is useful in managing multicollinearity.
   iii. Wendy chose Lasso Regression, which is comparable to Ridge but is perfect for feature selection because it can totally remove the weight of less significant characteristics.
   iv. Navya was the creator of two separate models: the k-Nearest Neighbours (kNN) Model, which is well-known for its simplicity and efficacy in categorization through in-depth feature space analysis, and the Random Forest Model, which excels at managing big datasets with increased dimensionality.
   v. Gautam's final focus was on exploratory data analysis (EDA) and insights. This was important because it helped us find patterns, trends, and anomalies in the data, which helped us make decisions about our modeling techniques and presumptions.

   This method allows us to compare and cross-validate the efficacy of various strategies in a controlled environment, in addition to individually tailoring models that best suit the same set of attributes. The variety of approaches enhanced our study by providing different viewpoints on the accuracy and dependability of our models. By working together, we were able to optimize our group's analytical skills and obtain important knowledge about the real-world uses of different regression and classification methods.

2. What were the main challenges?
   Throughout the progress of this project, we found there are three main challenges:
   i. Understanding the data and exploring the useful features:
      Understanding the data proved to be a significant hurdle. Given the complexity of the dataset, which comprised more than sixty features, data cleaning became pivotal for model usage. In the initial stages, while building our first model using logistic regression, we employed a proof of concept (POC) approach. This method involved basing our understanding of the data on logical assumptions and then training models to validate whether these assumptions held true. We initially analyzed all features within the context of the house hosting field to determine their utility. Additionally, handling missing values was informed by our past experiences in data cleaning, such as imputing using mean or median values. However, the poor generalization performance of our first model highlighted issues with our POC approach. This prompted us to revisit and refine our data cleaning process, particularly in how we treated each individual feature.

   ii. Tuning hyperparameters in different model families:
      Tuning hyperparameters for XGBoost presents a significant challenge, particularly when contending with an extensive feature set and a substantial number of trees. Our current methodology involves manually testing a handful of values for parameters like n.tree and depth, assessing their impact on performance through trial and error. This process becomes time-consuming, often exceeding 10 minutes per iteration with our dataset containing over 100 features (including textual features). This approach lacks systematic exploration of the parameters. To address this inefficiency, adopting a grid

search strategy could prove advantageous. By systematically evaluating a predefined set of parameter combinations, grid search can efficiently pinpoint the most promising configurations for maximizing performance.

iii. Compiling code from all group members:
Another significant challenge we faced during the project was the difficulty in compiling and collaborating on R code among various group members. Unlike more interactive and cloud-based coding environments like Google Colab, which allow for real-time collaboration, R generally requires more traditional methods of code sharing and integration, we updated our code by uploading R files to Google Drive which can be less immediate and visually intuitive. Although Google Colab does offer a potential solution by allowing simultaneous edits and execution of code in a shared cloud environment, we found that its computational power was somewhat limited, impacting our ability to run complex models efficiently. This situation underscored the need for more robust collaborative tools that can handle the demands of heavy computational tasks without sacrificing the ease and flexibility needed for effective team collaboration.

3. What we would have done differently if we could start the project over again?
While reflecting on our process, we recognized that a strong focus on feature engineering is essential for laying a solid foundation for subsequent modeling efforts. Looking back, dedicating more time to feature exploration would have been advantageous, as it allows for the identification of innovative ways to engineer features and leverage external resources that could significantly enhance model performance. Our group initially concentrated more on modeling techniques, but we soon realized the transformative impact that well-designed features could have on our results. If we could start over, we would plan to prioritize extensive feature exploration and the integration of external data sources early in the project to maximize the effectiveness of our predictive models.

4. What would you do if you had another few months to work on the project?
If we had more time on our hands for the project, our strategy would involve expanding the dataset by integrating additional external sources. This broader data scope would provide richer insights and allow for a more comprehensive analysis of underlying patterns. We would also focus on refining and adding to our feature engineering techniques to extract maximum predictive power from the available variables. By implementing more sophisticated feature extraction methods, we aim to uncover subtle patterns that could enhance model accuracy. Additionally, thorough hyperparameter tuning would be a priority. This process is crucial for optimizing model performance across various algorithms. By fine-tuning model parameters, we can ensure the most effective predictive outcomes.

5. What advice do you have for a group starting this project next year?
For a group starting a similar project next year, we have several pieces of advice based on the challenges and lessons learned from our project:

i. Prioritize understanding the data
Invest time and effort into understanding the dataset thoroughly. Focus on exploring features and identifying their relevance to the problem. Consider different ways apart from what's being taught in the lectures to engineer features and leverage external data sources to enhance model performance.

ii. Emphasize feature engineering
Place a strong emphasis on feature engineering, as it can significantly impact model performance. Dedicate time to exploring and refining feature extraction techniques to extract maximum predictive power from the dataset.

iii. Expand data scope
Consider integrating more external data sources to enrich the dataset and provide deeper insights into underlying patterns. Expanding the data scope can lead to more robust and accurate predictive models.

iv. Hyperparameter tuning
Recognize the importance of hyperparameter tuning in optimizing model performance. Instead of manually testing parameters, consider employing systematic techniques to efficiently explore the parameter space and identify optimal values of hyperparameters.

v. Enhance code management
Establish effective methods for collaborating on code among group members. Explore tools and platforms that facilitate real-time collaboration and version control to streamline the development process and ensure code integrity.