|  | **BUDT 758T**  **Final Project Report**  **Group 14**  **Spring 2024** |
| --- | --- |

# **Section 1: Team member names and contributions**

1. **Sai Manohar Beeraka**- Contributed to the modeling efforts, created an ensemble of XGBoost and Random Forest models and managed numerical and categorical variables for XGBoost modeling. Implemented Logistic Regression with Ridge. Cross validation for Logistic lasso model and Logistic Lasso. Performed SMOTE analysis to handle class imbalance.
2. **Vedant Kamat**- Utilized exploratory data analysis techniques to gain a comprehensive understanding of the datasets. In addition to his data preparation and analysis responsibilities, made significant contributions to the report writing process.
3. **Amoghvarsh Kulkarni**- Conducted comprehensive analysis of external data sources relevant to the project. This involved sourcing, cleaning, and processing diverse datasets to extract meaningful insights and incorporate that in our model.
4. **Jasvinder Singh**- Contributed to modeling efforts including Logistic Regression with Ridge, Logistic Regression with Lasso, and Random Forest models. Additionally, contributed to data engineering tasks and feature selection processes. Developed interaction terms during feature engineering. Conducted hyperparameter tuning for model optimization.
5. **Vrushti Shah**- Contributed to data engineering tasks and feature selection processes. Additionally, contributed to modeling with Logistic Regression, and performed model evaluation through fitting curves and learning curves. Conducted Text Mining for the unstructured text columns in the dataset.

# **Section 2: Business Understanding**

### **Introduction**

Airbnb.com, a pioneer in the sharing economy, has transformed the way people travel and utilize their living spaces. By connecting hosts who wish to rent out their properties with travelers seeking unique accommodations, Airbnb has cultivated a diverse marketplace that offers everything from private rooms to luxury homes globally. As the platform grows, the competition among listings intensifies, making it crucial for hosts to differentiate their offerings to attract bookings and achieve high guest satisfaction. Understanding the dynamics that lead to exceptional ratings and consistent bookings is key to maximizing a listing's potential. This business case proposes the development of predictive analytics models that will identify the characteristics of Airbnb listings that are most likely to achieve perfect ratings and high booking rates. These insights will empower hosts to optimize their listings, thereby enhancing guest experiences and increasing host earnings.

### **Business Need**

In the competitive landscape of Airbnb, achieving a perfect rating and maintaining a high booking rate are critical for the success of a host. These metrics influence a listing’s visibility and attractiveness, impacting the host's ability to secure future bookings and generate steady income. However, with numerous factors affecting guest satisfaction and booking decisions—from the quality of amenities to the accuracy of the listing description—hosts often struggle to identify and implement the changes that will most significantly improve their performance. There is a pressing need for a sophisticated, data-driven approach to decode the complexities of guest preferences and booking behaviors. Predictive models that can accurately forecast performance outcomes for listings will fill this gap, providing actionable insights that hosts can use to refine their offerings and align better with market demands.

### **Target Audience**

The predictive models developed from this project serve multiple stakeholders within the Airbnb ecosystem:

1. Airbnb Hosts: Individual hosts or property managers will use these models to gain insights into how different features of their listings, like amenities, pricing, or host responsiveness, influence the likelihood of achieving perfect rating. This knowledge will enable them to make targeted improvements that enhance guest satisfaction and optimize their listing’s performance.
2. Airbnb Corporate Team: The insights derived from the models can assist Airbnb's internal teams in developing new features and tools that help hosts enhance their listings. For example, predictive analytics could be integrated into the host dashboard, providing personalized tips and benchmarks against similar listings.
3. Real Estate Investors: Investors interested in properties for short-term rental purposes can use the model outputs to evaluate potential returns based on the predicted booking rates and guest satisfaction scores of different properties in various locations.
4. Tourism Consultants: Professionals advising on tourism and hospitality can leverage these models to provide strategic guidance to clients looking to enter or optimize their positions in the short-term rental market.

### **Model Overview**

Our project involves creating a predictive model as follows:

1. Perfect Rating Score Prediction (Binary YES/NO): This model predicts whether a listing will achieve a 100% perfect rating score, helping hosts understand the likelihood of achieving top-tier guest satisfaction.

The models will utilize **56** features extracted from Airbnb's data, including amenities, location, price, and host response rates, using machine learning techniques constrained to the R programming environment.

### **Business Actions and Value**

The implementation of these predictive models will drive several strategic actions and create substantial value across the Airbnb platform:

1. Listing Optimization: Hosts can use model insights to adjust critical aspects of their listings, such as redesigning interiors, upgrading amenities, or revising descriptions to highlight unique features. Such optimizations can directly enhance guest experiences, leading to better reviews and increased bookings.
2. Pricing Strategy: By understanding factors that contribute to high booking rates, hosts can adopt dynamic pricing strategies that reflect demand patterns, maximizing their revenue while remaining competitive.
3. Marketing Initiatives: Airbnb can utilize the findings from the models to craft targeted marketing campaigns that promote high-performing listings or encourage hosts to adopt practices that lead to higher satisfaction and booking rates.
4. Investment Decisions: Real estate investors can make informed decisions about where and what type of properties to acquire for Airbnb purposes, based on predictions of booking rates and potential guest satisfaction.
5. Platform Enhancements: Airbnb can integrate these analytics into its platform to offer automated suggestions to hosts for improving their listings, based on data-driven insights, thus maintaining a high standard of quality across the platform and improving overall user satisfaction.

**Takeaways**

Developing predictive models for Airbnb's platform serves as a strategic initiative to enhance host performance, guest satisfaction, and platform competitiveness. These models not only provide valuable insights for individual hosts but also support broader business objectives by fostering an optimized, data-driven ecosystem. This approach is essential for sustaining Airbnb's leadership and innovation in the evolving landscape of the travel and hospitality industry.

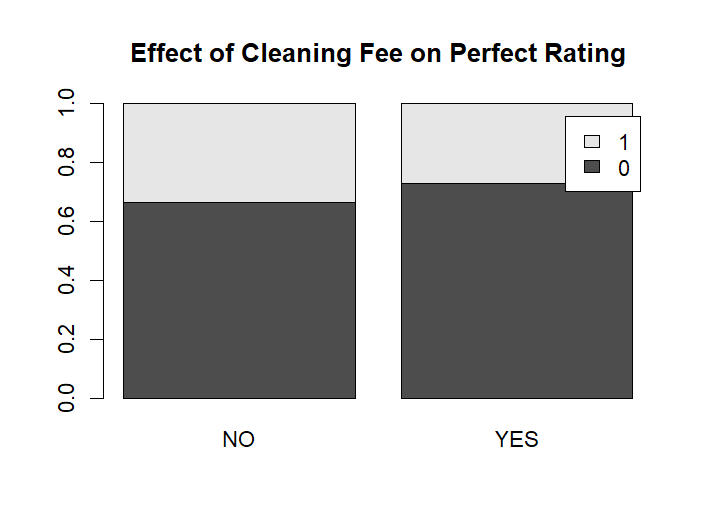
# **Section 3: Data Understanding and Data Preparation**

### **1) Below is a table explaining the features that were chosen by our team and the R code line numbers**

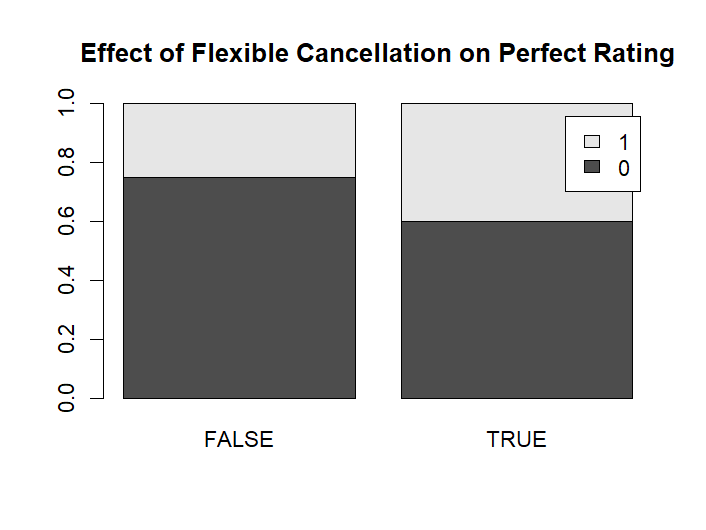
| **ID** | **Feature Name** | **Brief Description** | **R Code Line Numbers** |
| --- | --- | --- | --- |
| 1 | accommodates | Original feature from dataset | 287 |
| 2 | bedrooms | Number of bedrooms in the property | 287 |
| 3 | bathrooms | Number of bathrooms in the property | 287 |
| 4 | beds | Number of beds in the property | 287 |
| 5 | guests\_included | Number of guests included in the booking price | 287 |
| 6 | host\_response\_rate | Percentage rate at which host responds to inquiries | 287 |
| 7 | minimum\_nights | Minimum nights required for booking | 287 |
| 8 | has\_cleaning\_fee | Indicator if cleaning fee is charged | 287 |
| 9 | has\_extra\_people | Charges for extra people | 287 |
| 10 | price\_per\_person | Average price per person | 287 |
| 11 | bed\_category | Type of beds in the property | 287 |
| 12 | flexible\_cancellation | Cancellation policy is flexible | 287 |
| 13 | has\_security\_deposit | Whether a security deposit is required | 287 |
| 14 | ppp\_ind | Price per person indicator | 287 |
| 15 | property\_category | Category of the property (e.g., house, apartment) | 287 |
| 16 | amenity\_count | Number of amenities provided | 287 |
| 17 | host\_duration | How long the host has been with Airbnb | 287 |
| 18 | time\_since\_first\_review | Time since the property received its first review | 287 |
| 19 | booking\_availability\_ratio\_30 | Availability ratio over next 30 days | 287 |
| 20 | booking\_availability\_ratio\_60 | Availability ratio over next 60 days | 287 |
| 21 | booking\_availability\_ratio\_90 | Availability ratio over next 90 days | 287 |
| 22 | booking\_availability\_ratio\_365 | Availability ratio over next 365 days | 287 |
| 23 | response\_time\_categorical | Categorical response time of the host | 287 |
| 24 | high\_end\_amenities | Presence of high-end amenities | 287 |
| 25 | pet\_friendly | Property allows pets | 287 |
| 26 | child\_friendly | Property is suitable for children | 287 |
| 27 | public\_transit\_access | Accessibility to public transit | 287 |
| 28 | urban\_setting | Property is located in an urban area | 287 |
| 29 | rural\_setting | Property is located in a rural area | 287 |
| 30 | multi\_listing\_host | Host lists multiple properties | 287 |
| 31 | host\_verifications\_ratio | Ratio of verification methods used by the host | 287 |
| 32 | price | Price of the property per night | 287 |
| 33 | price\_per\_amenity | Price calculated per amenity provided | 287 |
| 34 | Households\_Total | Total number of households in the area | 287 |
| 35 | Households\_Mean\_income | Mean income of households in the area | 287 |
| 36 | Families\_Total | Total number of families in the area | 287 |
| 37 | Families\_Mean\_income | Mean income of families in the area | 287 |
| 38 | i1 | Interaction between price per person and high-end amenities | 228 |
| 39 | i2 | Interaction between moderate response time and host duration | 229 |
| 40 | i2\_1 | Interaction between slow response time and host duration | 230 |
| 41 | i3 | Interaction between host duration and host verifications ratio | 231 |
| 42 | i4 | Interaction between booking availability ratios for 30 and 365 days | 232 |
| 43 | i5 | Interaction between booking availability ratio for 30 days and urban setting | 233 |
| 44 | i6 | Interaction between amenity count and child-friendly properties | 234 |
| 45 | i7 | Interaction between urban setting and public transit access | 235 |
| 46 | i8 | Interaction between rural setting and pet-friendly properties | 236 |
| 47 | i9 | Interaction between flexible cancellation and multi listing host | 237 |
| 48 | i10 | Interaction between moderate response time and booking availability ratio for 365 days | 238 |
| 49 | i10\_1 | Interaction between slow response time and booking availability ratio for 365 days | 239 |
| 50 | i11 | Interaction between moderate response time and urban setting | 240 |
| 51 | i11\_1 | Interaction between slow response time and urban setting | 241 |
| 52 | i12 | Interaction between booking availability ratio for 30 days and price per person | 242 |
| 53 | i13 | Interaction between booking availability ratio for 30 days and flexible cancellation | 243 |
| 54 | i14 | Interaction between rural setting and high-end amenities | 244 |
| 55 | i15 | Interaction between multi listing host and host verifications ratio | 245 |
| 56 | i16 | Interaction between host duration and having a cleaning fee | 246 |

We have also used two unstructured text columns, Features and Amenities. Created TF-IDF of these two columns and added it into our original data set, then passed the whole data into our XGBoost Model.

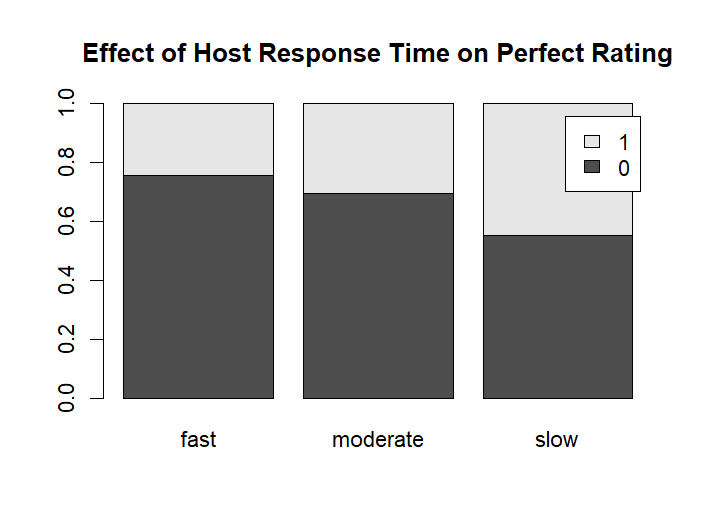
### **2) Graphs or tables demonstrating useful or interesting insights regarding features in the dataset**



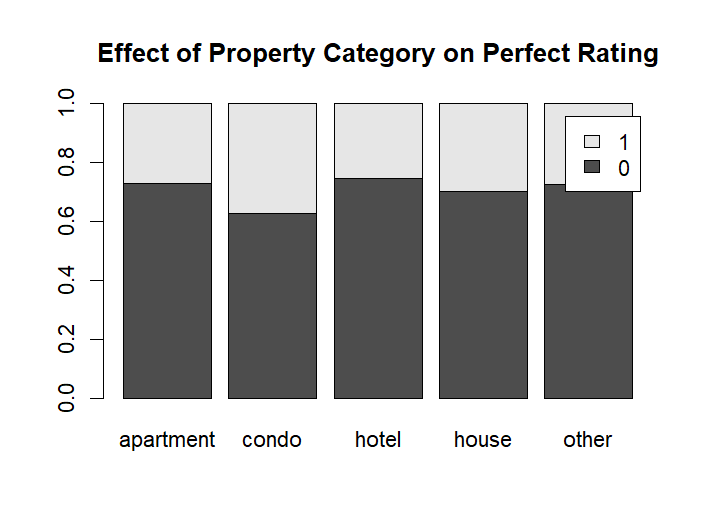




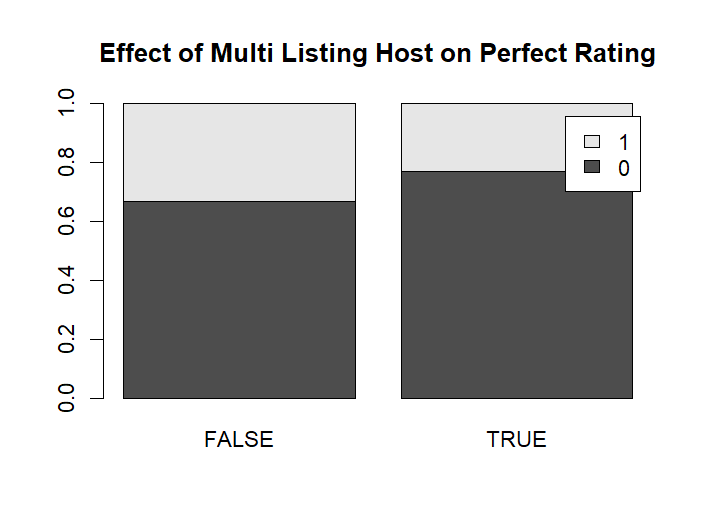




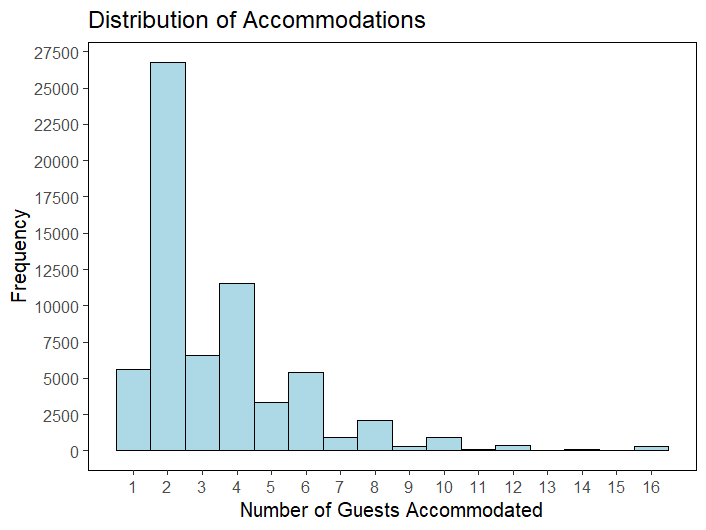




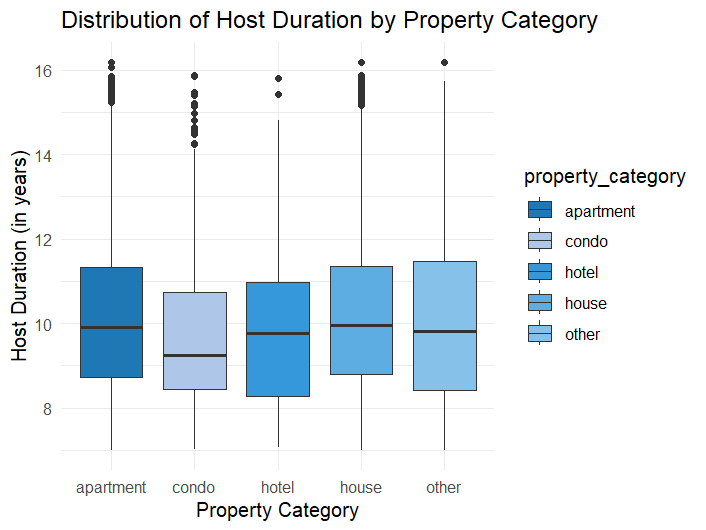




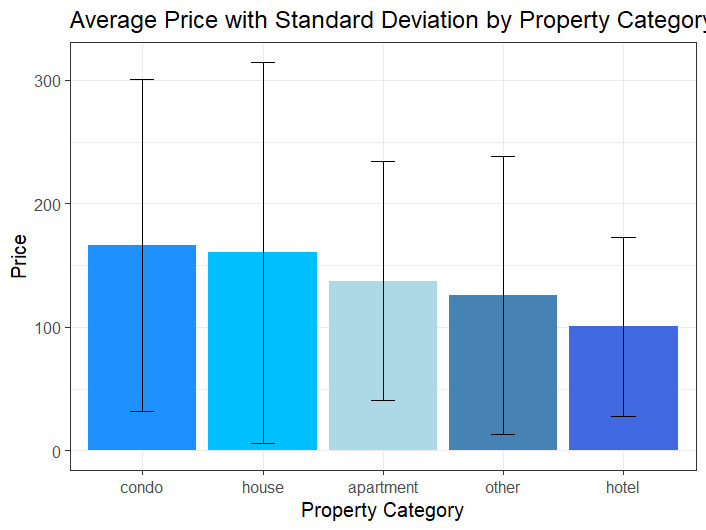




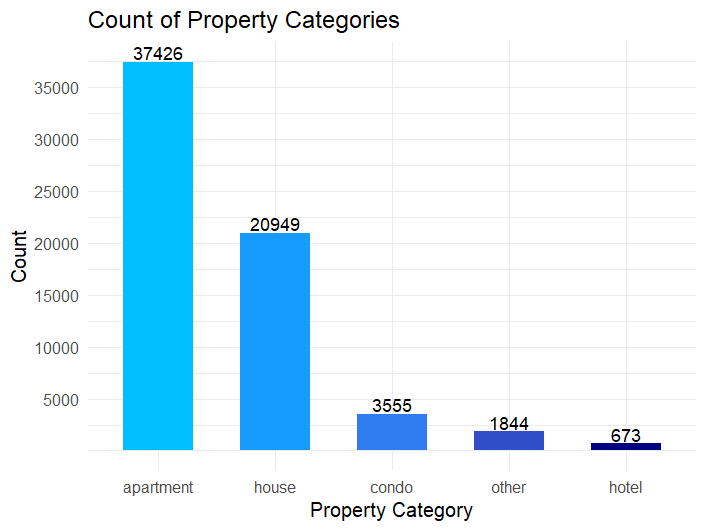




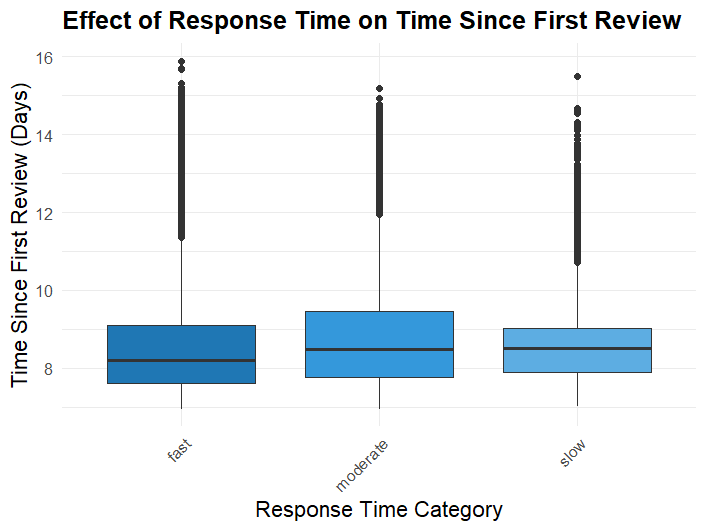




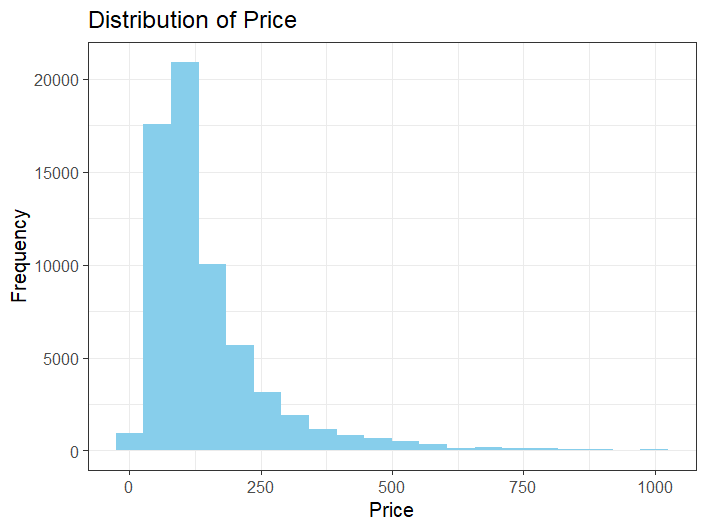




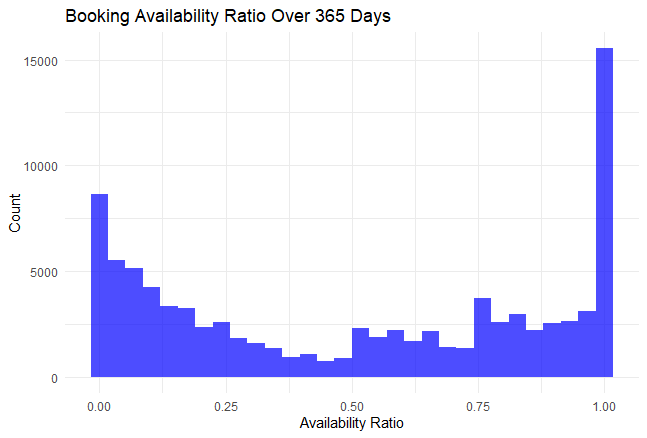




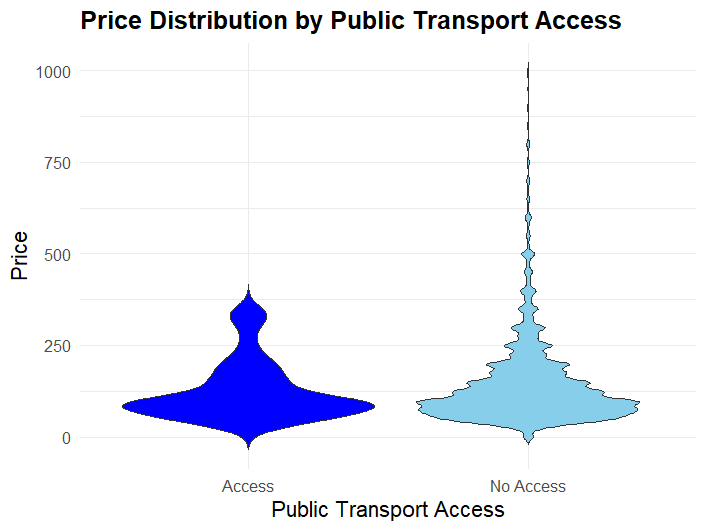




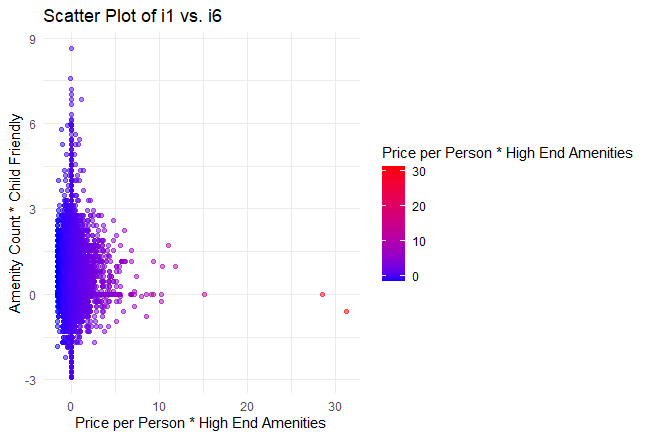




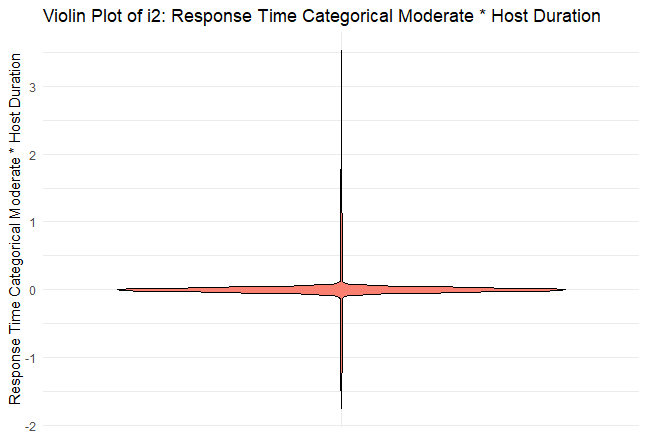














### **3) Additional Data Considered:** **Analysis of Socioeconomic Factors Influencing Airbnb Rating Scores**

**Introduction**

This section of the report examines the influence of key socioeconomic indicators on the likelihood of achieving perfect rating scores for Airbnb listings. The focus is on four specific variables: Total Households, Mean Household Income, Total Families, and Mean Family Income. These variables are integral in understanding the economic environment of the areas where Airbnb properties are located, which can significantly impact guest experiences and host capabilities.

**Variables and Their Relevance-**

1. **Total Households (Households\_Total)**
   * **Description:** This variable represents the total number of households within a given area.
   * **Relevance:** The number of households is indicative of the population density and urbanization of an area. Areas with higher household counts may exhibit greater demand for short-term rentals, influencing the volume of Airbnb transactions and potentially affecting the competition among hosts, which can drive improvements in service quality to achieve high ratings.
2. **Mean Household Income (Households\_Mean\_income)**
   * **Description:** This variable measures the average income per household in a specified area.
   * **Relevance:** Mean household income is a direct indicator of the economic status of a region. Higher average incomes are often correlated with better-maintained properties and more disposable income to invest in amenities that enhance guest experiences, thereby contributing to higher satisfaction and better reviews.
3. **Total Families (Families\_Total)**
   * **Description:** Reflects the total number of family units within an area.
   * **Relevance:** Similar to total households, the total number of families provides insights into the community and demographic structure. Areas with more families might prioritize family-friendly accommodations, influencing Airbnb hosts to adapt their offerings to cater to this demographic, potentially affecting ratings.
4. **Mean Family Income (Families\_Mean\_income)**
   * **Description:** This variable indicates the average income of families in the area.
   * **Relevance:** Family income can suggest a different economic dynamic than household income, as it often involves considerations for children and dependents. Properties in areas with higher family incomes might offer more spacious and comfortable accommodations suitable for families, which can positively impact guest reviews and ratings.

**Justification for Variable Selection-**

The selection of these variables is based on the hypothesis that economic factors significantly influence hosting capabilities and guest expectations. An area's economic health can affect a host’s ability to provide quality service and accommodations. Furthermore, guests from higher income areas might have higher expectations, which influences their ratings. Analyzing these variables allows us to assess whether hosts in wealthier or denser areas achieve higher ratings due to better property conditions, more amenities, or superior service quality.

**Takeaway-**

Understanding the socioeconomic context of Airbnb listings through variables such as household and family counts and incomes provides valuable insights into the dynamics that influence guest satisfaction and ratings. This analysis helps identify key economic factors that could be leveraged to improve Airbnb experiences and achieve perfect ratings, supporting targeted strategies for hosts to enhance their offerings.

**Evidence-**

The p-values for the variables Households\_Total, Households\_Mean\_income, Families\_Total, and Families\_Mean\_income in a logistic regression analysis were all under 0.05, this indicates that each of these variables is statistically significant in predicting the perfect rating score.

1. **Statistical Significance:** A p-value under 0.05 typically indicates that there is a statistically significant relationship between the predictor and the response variable. In this context, it means that changes in the household and family income variables are associated with changes in the likelihood of a perfect rating.
2. **Influence on the Outcome:** Since these variables are significant, they contribute meaningful information to the model about the outcome. This can be interpreted as these factors having a real effect on whether a rating is perfect or not.
3. **Model Impact:** The significance of these variables also suggests they should be included in the model for better prediction accuracy. Ignoring them could lead to a less effective model.

# **Section 4:Evaluation and Modeling:**

### 

### **Winning Model: XG BOOST (Code Line Numbers: 1-895)**

**(a) The type of model**: The winning model is an XGBoost model, a type of gradient boosting framework that is renowned for its performance in structured data prediction tasks.

**(b) The R function and/or library used:** The model was built using the xgboost() function from R's base package.

Variables Included: The model incorporates a wide range of features including both raw and engineered features:

* Numerical features like accommodates, bedrooms, bathrooms, beds, guests\_included, host\_response\_rate, and many more.
* Categorical features converted to dummy variables such as property\_category, bed\_category, has\_cleaning\_fee, etc.
* Text-derived features from amenities and other descriptive text fields processed into TF-IDF vectors.
* Interaction terms that combine multiple features to capture their combined effects on the response variable.
* Performance Estimation:
* Training Performance: High accuracy on the training set, indicating that the model captures the underlying patterns well.
* Generalization Performance: Assessed using a combination of validation data and cross-validation during the tuning process, ensuring the model is robust and performs well on unseen data.

Model Selection: The model was selected based on the highest area under the ROC curve (AUC) from the cross-validation results on the validation dataset. Parameters were tuned iteratively to maximize this metric, indicating strong predictive performance especially for binary classification tasks.

**(c) Estimated training and generalization performance**:

The model's performance is typically assessed using metrics TPR (True Positive Rate), and FPR (False Positive Rate).

**(d) Generalization Performance Estimation Methodology**: Cross-validation is utilized within the xgb.cv function for hyperparameter tuning which also provides an estimate of generalization performance through multiple folds ( 5 folds in the nfold argument).

Specific parameters for validation include splitting data into training (70%), validation (20%), and testing (10%).

**(e)** **Best-Performing Set of Features:** Features include amenities, host-related metrics, booking ratios, and categorically transformed features like has\_security\_deposit, flexible\_cancellation

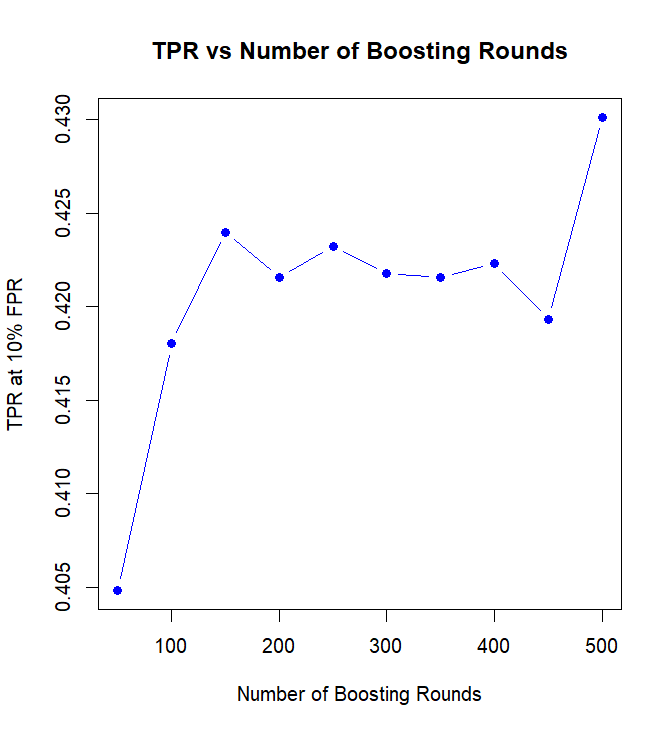
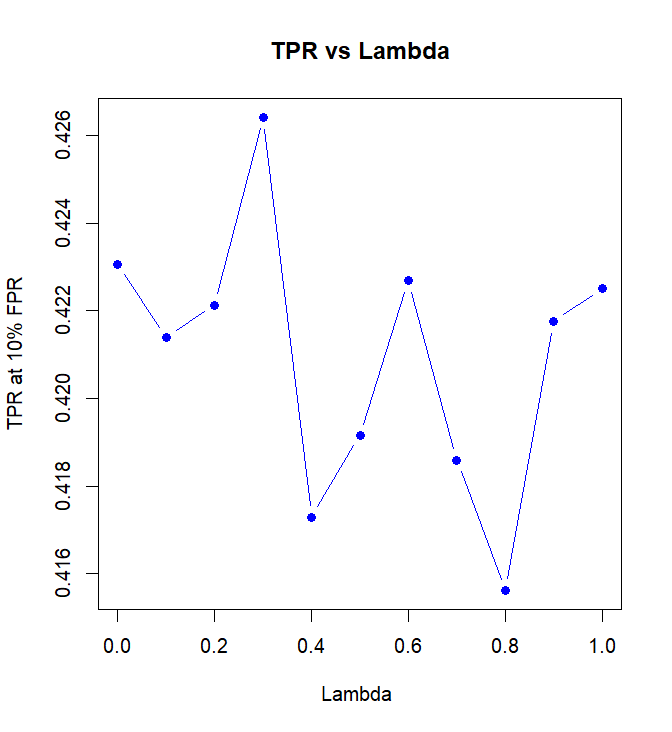
**(f) Model Training and Performance Estimation:**

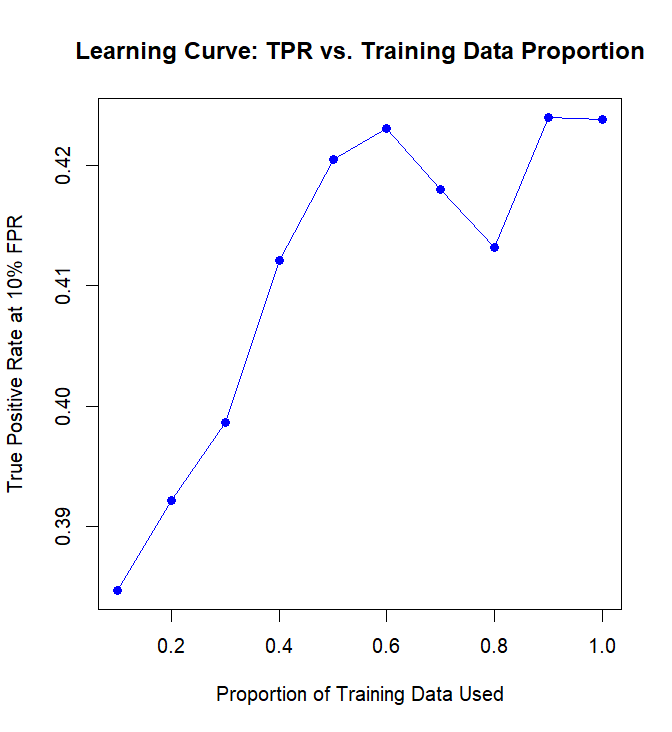
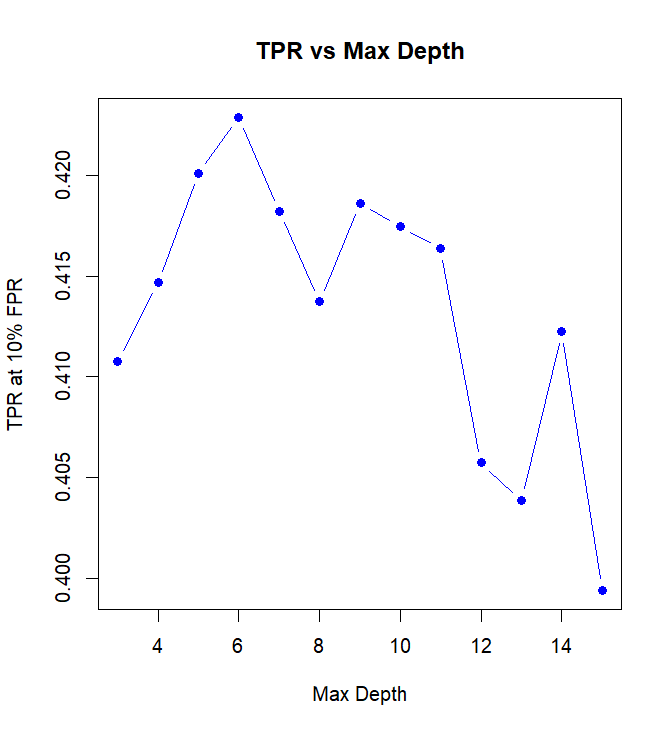
Model training with XGBoost: The setup starts around the line xgboost\_bst <- xgboost(...).

Generalization performance estimation using ROCR: Starting with prediction(preds\_xg, valid\_y\_xg\_boost).

**(g) Hyperparameters Tuned:**

max.depth, eta, subsample, colsample\_bytree, lambda, alpha, min\_child\_weight, and gamma are the hyperparameters tuned in the script. Specific values tested are mentioned in the param\_grid definition and include various levels primarily for alpha in the grid search setup.

**Learning Curve Insight**: The TPR improves initially with more data, dips slightly at a 0.6 to 0.8 proportion, then stabilizes and peaks as the training data nears full utilization. This pattern indicates that initially more training data helps the model perform better, but performance variability suggests potential overfitting or the need for further model tuning. The plateau at full data usage highlights a performance limit with the current model and dataset.



**Evidence of Using Unstructured Text Column**:

The True Positive Rate (TPR) achieved was 0.452 at a False Positive Rate (FPR) of 0.098. This improvement resulted from using text columns to create features.Without the features from text columns, TPR was 0.4126 at FPR of 0.098.

The chosen model and its evaluation strategy demonstrate a robust approach to predicting perfect rating scores using Airbnb data. The use of advanced feature engineering, careful model tuning, and rigorous validation ensures that the model not only fits the training data well but also generalizes effectively to new, unseen data. This comprehensive approach underpins the reliability of your predictive insights, making them valuable for practical applications.

### **LOGISTIC REGRESSION ( Code Line Numbers: 907-1085)**

1. **The type of model** : Logistic Regression
2. **The R function and/or library used** : The model was built using the glm() function from R's base package.
3. **Estimated training and generalization performance**: The model's performance is typically assessed using metrics TPR (True Positive Rate), and FPR (False Positive Rate). These were calculated from a confusion matrix obtained via predictions on the validation dataset.
4. **The generalization performance of the model was estimated:** using a simple train/validation/test split:

* 70% training data to fit the model.
* 20% validation data to assess model performance and tune hyperparameters.
* 10% test data for a final unbiased evaluation.

The data splitting was done using custom function dataSplit() which randomized data assignments to ensure a diverse representation in each subset.

1. **Best-Performing Set of Features:**

The model included features related to accommodation specifics, host details, and property type:

Accommodation features: availability, bathrooms, bedrooms, beds, price\_per\_person, etc.

Host-related features: host\_listings\_count, host\_response\_rate, host\_total\_listings\_count.

Property attributes: property\_category, has\_cleaning\_fee, has\_extra\_people, has\_security\_deposit.

1. **Model Training and Performance Estimation:**

* Model Training: The logistic regression model was trained at the line where glm() is called.

model\_logistic <- glm(perfect\_rating\_score ~ availability\_30 + availability\_365 +

availability\_60 + availability\_90 + bathrooms + bedrooms + beds +

has\_cleaning\_fee + has\_extra\_people + guests\_included +

host\_listings\_count + host\_response\_rate + host\_total\_listings\_count +

latitude + minimum\_nights + price\_per\_person + bed\_category +

cancellation\_policy+has\_security\_deposit+ ppp\_ind + property\_category,

data = train, family = "binomial")

* Performance Estimation: The performance was evaluated using the validation set, and relevant metrics were extracted from the confusion matrix.

cf\_logistic <- logistic\_pred(valid, model\_logistic, 0.42)

1. **Hyperparameters Tuned**

Cutoff for classification: Different thresholds for converting predicted probabilities to class outputs have been tested, with 0.42 used in the script.

1. **Fitting Curves Created**

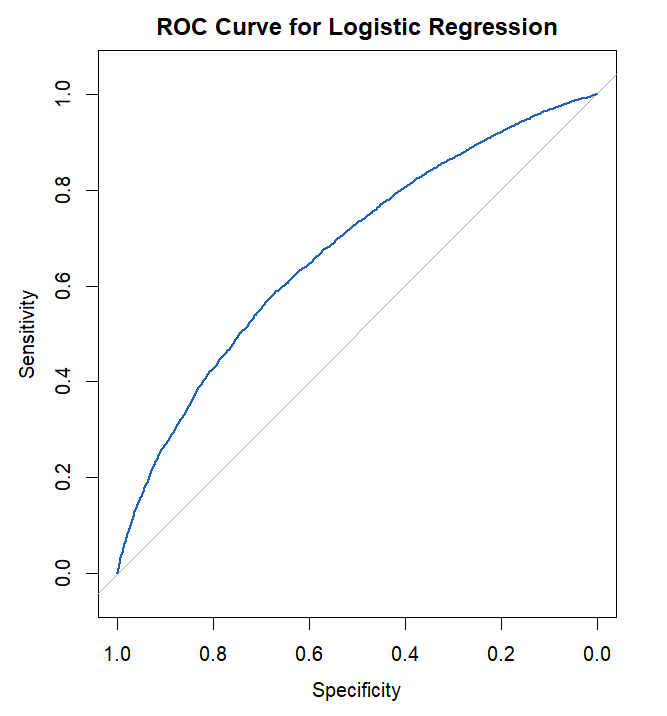
A ROC curve was plotted to assess model performance across various thresholds.

# Compute the ROC curve

roc\_obj <- roc(response = actual\_binary, predictor = predictions\_prob)

# Plot the ROC curve

plot(roc\_obj, main = "ROC Curve for Logistic Regression", col = "#1c61b6", lwd = 2)



### 

### **LOGISTIC REGRESSION WITH LASSO (Code Line Numbers: 1097-1545)**

1. **Type of Model** : The model used is Logistic Regression with Lasso Regression
2. **R Function and/or Library Used** : The logistic regression model is trained using the glmnet function from the glmnet package
3. **Estimated Training and Generalization Performance**: Training and generalization performance can be estimated using metrics like ROC, TPR, FPR.
4. **Generalization Performance Estimation Methodology**:

* Generalization performance was estimated using:
* Cross-validation: The script uses a k-fold cross-validation approach (k=5) within the cv.glmnet function to determine the regularization parameter (lambda) and avoid overfitting.
* Train/Validation/Test Split: Data is split into training (70%), validation (20%), and test (10%) sets to evaluate model performance.

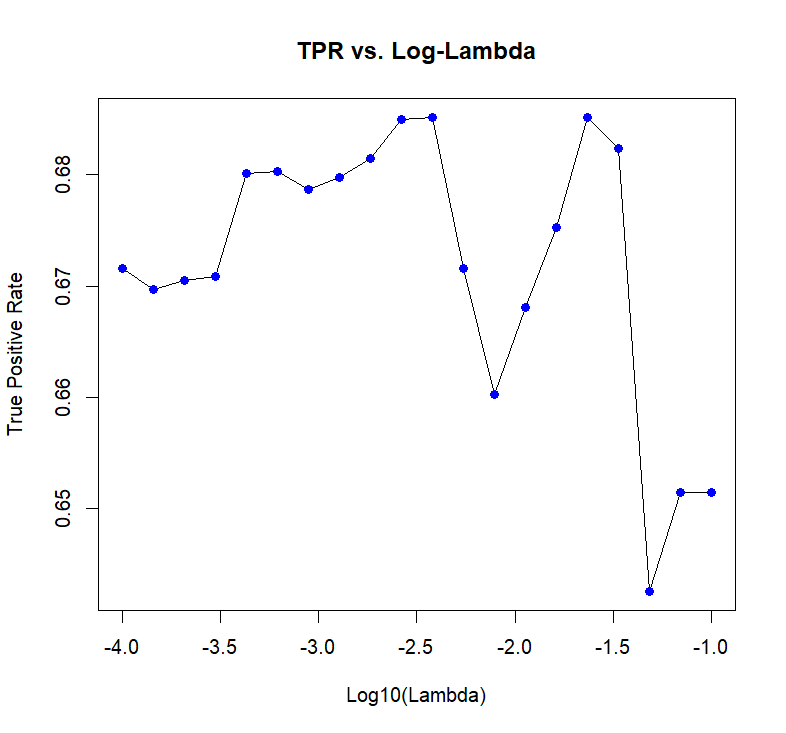
1. **Best-Performing Set of Features**:

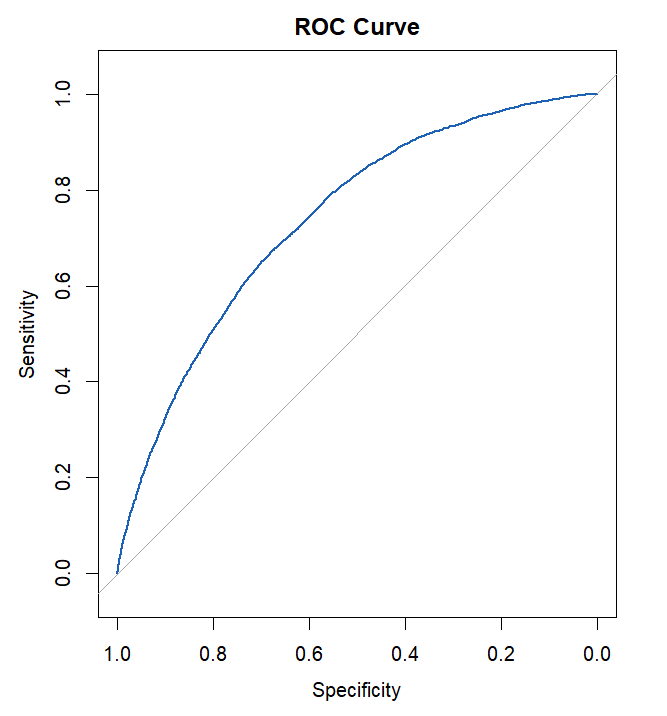
Included Features: Accommodates, bedrooms, bathrooms, beds, guests included, host response rate, minimum nights, price per person, amenity count, host duration, time since first review, various booking availability ratios, host verifications ratio, price, price per amenity, household and family income statistics.

1. **Line Numbers for Model Training and Performance Estimation**:

* Model Training: The model is trained with cv.glmnet at the section where cv.out is created.
* Generalization Performance Estimation: Performance is primarily estimated using the predictions made on the validation set using predict function.

1. **Tuned Hyperparameters**:

* Lambda (lambda): Regularization parameter in logistic regression, determined through cross-validation. The grid of lambda values tested ranges from 10^-1 to 10^-4.
* Alpha (alpha): The elastic-net mixing parameter, set to 1 (Lasso penalty).

1. **Fitting Curves Created (ROC Curve)**

### **LOGISTIC REGRESSION WITH RIDGE (Code Line Numbers: 1554-1938)**

1. **Type of Model :** Logistic Regression with Ridge
2. **R Function and/or Library Used:**

* glmnet: Used for fitting generalized linear models via penalized maximum likelihood. The cv.glmnet function from the glmnet package
* pROC: Used for ROC curve analysis, specifically the roc() function to calculate and plot ROC curves.

1. **Estimated Training and Generalization Performance:**

The performance metrics such as ROC, TPR,FPR metrics computed to estimate the performance.

1. **Generalization Performance Estimation Methodology:**

* Cross-validation: Utilized via cv.glmnet with k-fold cross-validation (k=5), which is crucial for estimating the model's ability to generalize to new data and for selecting the optimal regularization parameter (lambda).
* Data Split: The data was split into training (70%), validation (20%), and testing (10%) sets to evaluate model performance, following a typical train/validation/test split methodology.

1. **Best-Performing Set of Features:**

Included Features: Features such as accommodation details (e.g., accommodates, bedrooms), booking details (availability\_30, availability\_365), host details (host\_listings\_count), and financial aspects (price, cleaning\_fee).

1. **Line Numbers for Model Training and Performance Estimation**

* Model Training: Occurs where cv.glmnet is called to train the model with cross-validation to select lambda.

cv.out <- cv.glmnet(train\_x, as.matrix(train\_y), family="binomial", alpha=0, lambda=grid, nfolds=k)

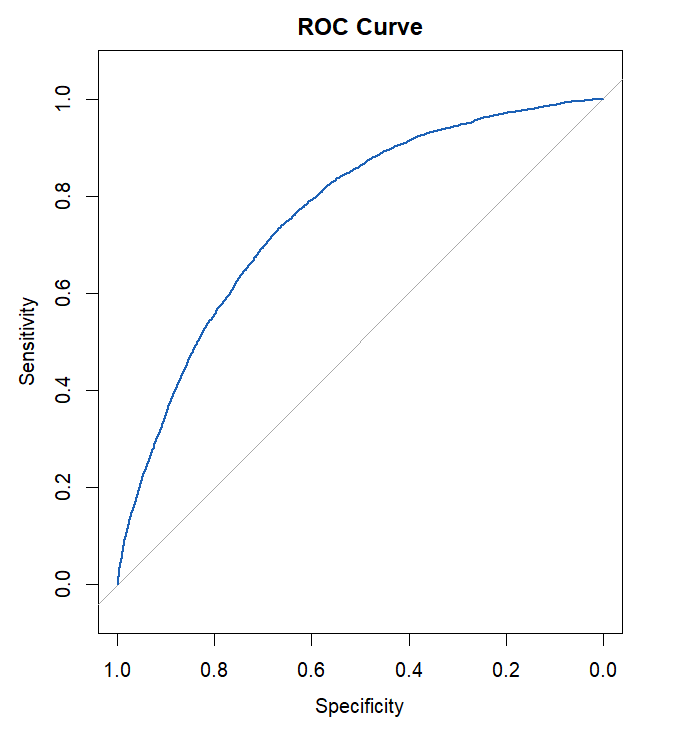
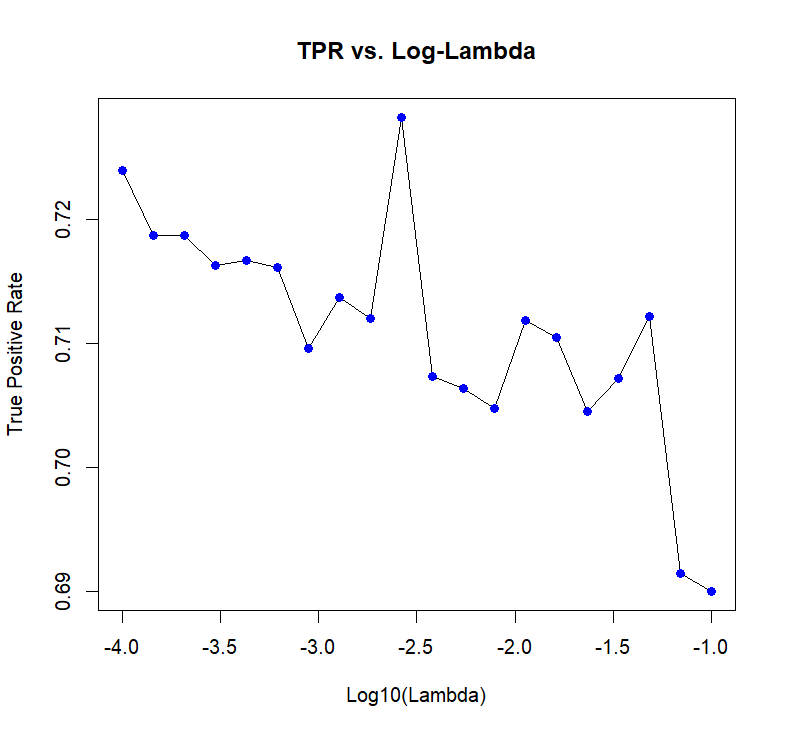
* Performance Estimation: The ROC curve and prediction outputs are generated to assess model performance.

pred\_prob <- predict(cv.out, s = bestlam, newx = valid\_x, type="response")

roc\_obj <- roc(valid\_y$perfect\_rating\_score, pred\_prob)

1. **Tuned Hyperparameters:**

* Lambda (lambda): A range of lambda values from 10^-1 to 10^-4 was tested to determine the optimal amount of regularization.
* Alpha (alpha): Set to 0 for Ridge penalty, focusing on minimizing a slightly modified RSS (Residual Sum of Squares) that adds a penalty equivalent to the square of the magnitude of the coefficients.

1. **Fitting Curves Created:** 

### **RANDOM FOREST (Code Line Numbers: 1947-2495)**

1. **Type of Model:** Random Forest
2. **R Function and Library Used:**RandomForest from the randomForest package is used for training the model.
3. **Estimated Training and Generalization Performance:** Performance Metrics: The model's performance is typically evaluated using ROC, TPR,FPR, etc.,.
4. **Methodology for Estimating Generalization Performance:**

* Validation Setup: You used a simple train/validation split (70% training, 20% validation, 10% test) as described in your data splitting function.
* vary ntrees and mtry to get the optimal value to maximize TPR at 10% FPR

1. **Best-performing Set of Features:**

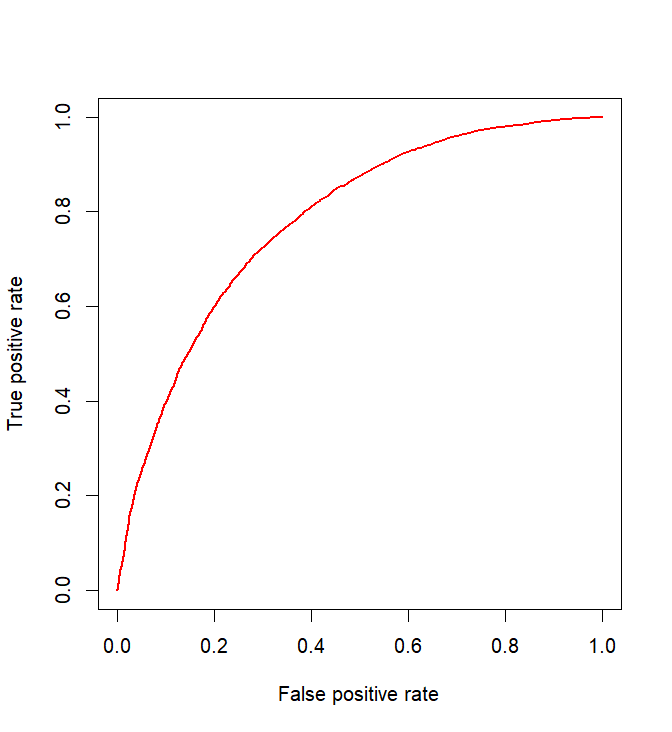
* Feature Set: Features include accommodations, bedrooms, bathrooms, etc., all of which appear to be related to Airbnb listing attributes. The final model appears to use many interaction terms and transformed features for prediction.
* Notable Features: High booking rate, availability ratios, and personalized features like price\_per\_person multiplied by high\_end\_amenities.

1. **Lines of Code for Model Training and Performance Estimation:**

* Model Training: The random forest model is trained in the block starting with rf.mod <- randomForest(...) which is within the final block of the code.
* Performance Estimation: Performance estimation is set up immediately after the model training, where predictions are made and performance metrics like TPR and FPR are calculated.

1. **Hyperparameters Tuned and Values Tried:**

* Hyperparameters: mtry and ntree are the hyperparameters tuned.
* Values: mtry varied from 2 to 20 by 2s in a loop, and ntree was used at 350 for the model execution mentioned, though it was tested at other values in previous setups.



1. **Fitting Curves Created:**

### **XGBOOST WITH NUMERICAL AND CATEGORICAL VARIABLES (Code Line Numbers: 2504-3294)**

1. **Type of Model:** Xgboost
2. **R Function and Library Used:**xgboost from the xgboost package for XGBoost modeling.
3. **Estimated Training and Generalization Performance:** Performance Metrics: The model's performance is typically evaluated using ROC, TPR,FPR, etc.,.
4. **Methodology for Estimating Generalization Performance:**

* Cross-Validation: Particularly with the XGBoost model, using parameters like nfold and early\_stopping\_rounds during xgb.cv for robust estimation of performance.
* Train/Validation/Test Split: Data is split into training, validation, and test sets, with the validation set used to tune the models and the test set to evaluate final model performance.

1. **Best-performing Set of Features:**

* Features used include accommodates, bathrooms, price\_per\_person, and several engineered features like booking\_availability\_ratio\_30 and interactions such as price\_per\_person\*high\_end\_amenities.
* Interaction terms and specific property features like location and amenities turned out to be highly indicative of the outcomes.

1. **Lines of Code for Model Training and Performance Estimation:**

xgboost\_bst <- xgboost(

data = as.matrix(comb\_train),

label = as.numeric(train\_y$perfect\_rating\_score),

max.depth = 6, # Reduced model complexity

eta = 0.05, # Fine-tuned learning rate

subsample = 0.6, # Reduced subsample ratio

colsample\_bytree = 0.6, # Reduced column subsampling

lambda = 0.5, # Increased L2 regularization

alpha = 0.1, # Increased L1 regularization

min\_child\_weight = 18, # Increased minimum child weight

gamma = 0.6, # Adjusted gamma

scale\_pos\_weight = 3.5, # Optimized class weights

nrounds = 250,

objective = "binary:logistic"

)

preds\_xg <- predict(xgboost\_bst, as.matrix(comb\_valid))

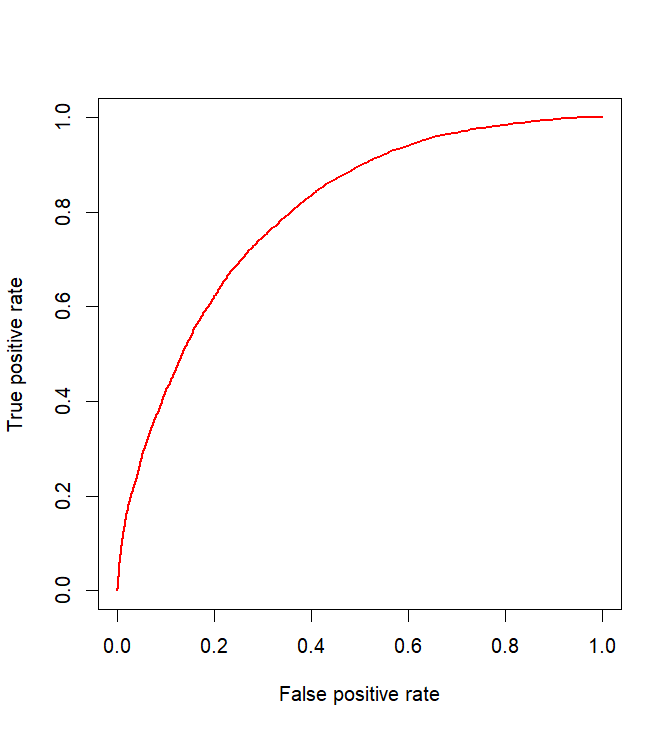
valid\_y\_xg\_boost <- as.numeric(valid\_y$perfect\_rating\_score)

1. **Hyperparameters Tuned and Values Tried:**

For the XGBoost model:

* max.depth: Controls the depth of the trees (tried values like 6, more profound trees for complex patterns).
* eta: Learning rate (values like 0.05 to slow learning and prevent overfitting).
* subsample and colsample\_bytree: Controls the fraction of the subsample and features used per tree (values around 0.6 to ensure model isn't too biased or variance heavy).
* lambda and alpha: L1 and L2 regularization terms (varied across ranges like 0.1 to 1 to control overfitting).
* min\_child\_weight and gamma: Control overfitting by making the model conservative through the tuning of split decisions.

1. **Fitting Curves Created:**

* TPR vs. FPR curves plotted using ROCR to visualize model performance under different thresholds.
* Learning curves to show how model accuracy improves with an increase in the data size.
* Parameter tuning effects illustrated through plots of metrics like AUC as functions of hyperparameters such as max\_depth, lambda, etc.

### **ENSEMBLE MODEL (RANDOM FOREST+ XGBOOST) (Code Line Numbers: 3310-3937)**

1. **Type of model:** Random Forest + XGBoost
2. **R function and/or library used :**

* Random Forest: Implemented using the ranger function from the ranger package.
* XGBoost: Implemented using the xgboost function from the xgboost package.

1. **Estimated training and generalization performance**

* Random Forest: ROC, TRP, FPR.
* XGBoost: ROC, TRP, FPR.

1. **Methodology for estimating generalization performance:**

The generalization performance is estimated using a simple train/validation/test split method. This is evident where the data is split into training, validation, and test datasets with proportions of 70%, 20%, and 10% respectively.

1. **Best-performing set of features:** The features used include both raw and engineered features such as:

* Basic room and property attributes (accommodates, bedrooms, bathrooms, etc.).
* Host-related features (host\_response\_rate, host\_listings\_count).
* Text-based features derived from amenities.
* Interactions and polynomial features such as price\_per\_person \* high\_end\_amenities.

1. **Line numbers for training the model and estimating performance:**

XGBoost Training:

* Training is performed around the lines involving xgboost() function call.

Random Forest Training and Performance Estimation:

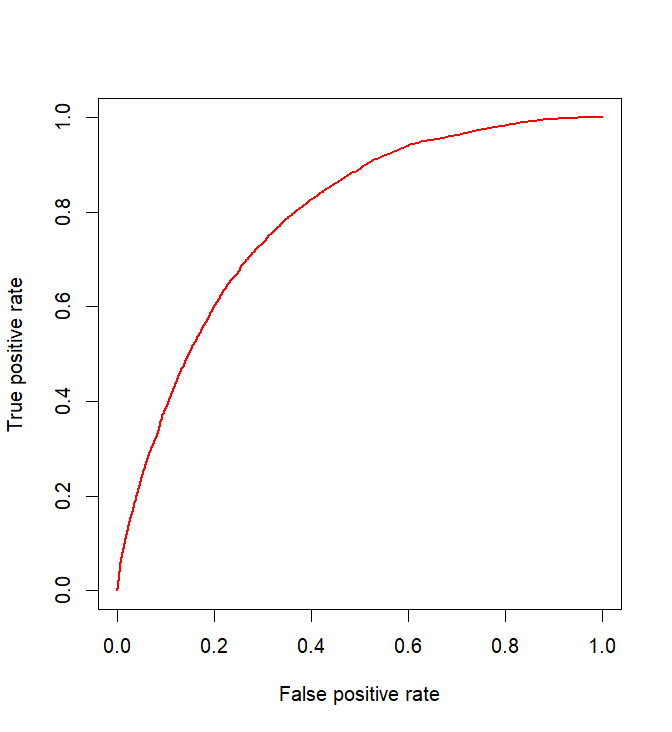
* Training occurs around the line involving ranger() call.
* Performance estimation (AUC calculation) is mentioned in the lines discussing the prediction() and performance() functions from the ROCR package.

1. **Hyperparameters tuned and their values:**

For XGBoost, the following hyperparameters are tuned:

* max.depth (tree depth), typically between 3-10.
* eta (learning rate), common values range from 0.01 to 0.3.
* subsample and colsample\_bytree (both related to the fraction of samples and features), usually set between 0.5 and 1.
* Regularization parameters like lambda and alpha to control model complexity and prevent overfitting.

For Random Forest, the following hyperparameters are tuned:

* mtry, tuned between 5-20.
* Num.trees values range from 200 to 800.

1. Fitting curves:

# **Section 5: Reflection/Takeaways:**

### **1)** **What did your group do well?**

Our group excelled in several key areas during the project:

* **Collaboration and Communication**: We maintained open and frequent communication throughout the project, ensuring that all members were aligned with the project goals and aware of their responsibilities.
* **Technical Skills and Resourcefulness**: We effectively leveraged our collective technical skills in data analysis, machine learning, and programming to handle complex datasets and apply advanced modeling techniques.
* **Problem-Solving**: We demonstrated strong problem-solving skills, especially in tackling unexpected challenges related to data quality and model performance.

### **2) What were the main challenges?**

We faced several challenges:

* **Data Quality and Preparation**: A significant amount of time was devoted to cleaning and preparing the data for analysis, which was more complex than anticipated due to inconsistencies and missing values.
* **Model Complexity and Selection**: Balancing the complexity of the model with the interpretability and computational efficiency was challenging, especially when integrating various data sources and feature sets.
* **Overfitting**: Initially, our models tended to overfit the training data, requiring careful tuning of parameters and validation strategies to generalize better to unseen data.

### **3) What would your group have done differently if you could start the project over again?**

If we were to start the project over, we would make a few strategic changes:

* **Earlier and More Rigorous Validation**: Implement cross-validation and other model validation techniques earlier in the process to better guide the feature selection and model tuning.
* **Enhanced Data Exploration**: Spend more time initially understanding and exploring the data to uncover more insights that could inform our modeling strategy and feature engineering**.**

### **4) What would you do if you had another few months to work on the project?**

Given more time to work on the project, we would do the following:

* **Explore Alternative Modeling Techniques**: Experiment with other machine learning algorithms and ensemble methods that might yield better performance or insights.
* **Implement Advanced Feature Engineering**: Explore more sophisticated text analysis techniques and interaction effects among features to improve model accuracy.
* **Deploy and Monitor the Model**: Develop a deployment strategy to implement the model in a real-world setting and establish a monitoring system to evaluate the model's performance over time and adjust as necessary.

### **5) What advice do you have for a group starting this project next year?**

For groups undertaking this project next year, we offer the following advice:

* **Plan and Prioritize**: Early in the project, prioritize tasks and allocate time and resources efficiently. Understand that data preprocessing might take longer than expected and plan accordingly.
* **Maintain Documentation**: Keep thorough documentation from the start. This practice is invaluable for troubleshooting, understanding your process, and ensuring reproducibility.
* **Be Adaptive:** Be prepared to pivot or adjust your strategies based on interim findings and challenges. Flexibility and adaptability is crucial in data science projects.
* **Utilize Version Control:** Use tools like Git for version control to manage changes and collaborate effectively, especially in group settings.
* **Engage with Professor/TA:** If possible, regularly engage with the professor/ TA who can provide insight or feedback on your project direction and outcomes.

Reflecting on the project, our experiences provide a foundation of learning and growth in data science that will benefit our future academic and professional endeavors. The challenges we faced were matched by the satisfaction of solving complex problems and producing a model that could have real-world applications.