UNIVERSITY OF MARYLAND
ROBERT H. SMITH
SCHOOL OF BUSINESS

BUDT 758T
Final Project Report
Spring 2024

Section 1
Team Members:

Akshatha Prabhu:

- Actively participated in the initial collaborative review of the dataset to identify key features for model training.
- Focused on Exploratory Data Analysis (EDA) and Data Cleaning to evaluate the influence of different features on the models.
- Focused efforts on building and evaluating multiple models, including Logistic Regression, Decision Trees, Random Forest, and XGBoost.
- Explored the impact of hyperparameter tuning on model performance, employing cross-validation techniques with varying fold values.

Jagadeesh Harinarayanan:

- Actively participated in the initial collaborative review of the dataset to identify key features for model training.
- Focused efforts on building and evaluating multiple models, including Logistic Regression, Decision Trees, Random Forest, and XGBoost.
- Explored the impact of hyperparameter tuning on model performance, employing cross-validation techniques with varying fold values.
- Provided support during Exploratory Data Analysis (EDA) and Data Cleaning.

Sanya Gangwar:

- Actively participated in the initial collaborative review of the dataset to identify key features for model training.
- Applied text mining techniques to investigate the importance of text features in model accuracy.

- Set up the training and validation datasets to ensure the models were robust and representative.
- Provided invaluable assessments and insights, contributing to the iterative improvement of the models.

Tejaswini Kandula:

- Actively participated in the initial collaborative review of the dataset to identify key features for model training.
- Focused on Exploratory Data Analysis (EDA) and Data Cleaning to evaluate the influence of different features on the models.
- Applied text mining techniques to investigate the importance of text features in model accuracy.
- Set up the training and validation datasets to ensure the models were robust and representative.

Venkat Nikhil Yerramilli:

- Actively participated in the initial collaborative review of the dataset to identify key features for model training.
- Focused efforts on building and evaluating multiple models, including Logistic Regression, Decision Trees, Random Forest, and XGBoost.
- Explored the impact of hyperparameter tuning on model performance, employing cross-validation techniques with varying fold values.
- Rigorously analyzed the performance of each model using various metrics, such as AUC (Area Under the Curve), accuracy, and confusion matrices.

Through this synergistic approach, the team harnessed the collective expertise of its members, fostering a dynamic and engaging environment for model development and optimization. The collaborative spirit and dedication displayed by each team member played a pivotal role in achieving remarkable results.

Section 2
Executive Summary

Unlocking High Booking Rates

In the ever-competitive world of travel and hospitality, maximizing occupancy rates and optimizing pricing strategies is a constant challenge. Recognizing the immense potential within this treasure trove, our team has developed a predictive analytics model that estimates the probability of a new listing achieving a high booking rate. This innovative solution empowers hosts to unlock a competitive edge in a crowded marketplace, enabling them to make data-driven decisions that drive growth and profitability.

Target Audience: Hosts, Property Managers, and Investors

The primary beneficiaries of our model are hosts who are eager to increase their property's visibility and booking rates. Additionally, our solution caters to property management companies and real estate investors seeking to maximize the profitability of their short-term rental portfolios. By leveraging our model's insights, these stakeholders can strategically position their offerings to attract more guests and generate higher return.

A Data-Driven Approach to Success

Our model harnesses the power of a comprehensive dataset, encompassing features such as location, price, number of bedrooms, amenities, and host reputation. Through the application of advanced machine learning techniques – including Decision Trees, Random Forest, and XGBoost – we have crafted a predictive powerhouse that accurately forecasts the likelihood of a property being booked frequently.

Actionable Insights for Strategic Decisions

The output of our model equips hosts and property managers with invaluable insights to make informed decisions that drive business success:

- Property Enhancements: Our models show that features such as availability of a laundry facility, kitchen and parking increase booking rates. Based on these recommendations hosts can tailor amenities and accommodation to align with high-demand features that attract more bookings, staying ahead of the competition.

- Dynamic Pricing: Adjust prices based on predictive insights to maximize occupancy without compromising revenue, striking the perfect balance between profitability and guest satisfaction.

- Targeted Marketing: Enhance listing descriptions and visuals to highlight the features that are statistically significant in influencing booking rates, capturing the attention of potential guests.

Unlocking Unparalleled Value

1. Increased Revenue: By optimizing listings to align with high-booking probability features, hosts can significantly increase their occupancy rates and, consequently, their earnings, driving sustainable growth.
2. Operational Efficiency: Property managers can streamline operations by focusing on properties and features that yield the highest return on investment, maximizing resource allocation and minimizing waste.
3. Market Competitiveness: With actionable insights derived from our model, hosts can maintain a competitive edge by adapting to evolving market trends and guest preferences, staying ahead of the curve in the dynamic landscape of short-term rentals.

Conclusion

Our predictive model represents a game-changing solution for hosts and property managers seeking to enhance booking rates and operational efficiency. By leveraging the powerful insights provided by our cutting-edge technology, users can make strategic adjustments to their properties, pricing, and marketing strategies, ultimately leading to increased profitability and market competitiveness. Embrace the future of data-driven decision-making and unlock the full potential of listings with our innovative predictive analytics model.

Section 3: Data Understanding and Data Preparation

In our modeling process, a critical step was identifying the features that effectively explained our target variable, high booking rates. With a dataset comprising 61 columns, over half of which were text-based, the potential permutations and combinations for exploratory features were vast. We aimed to refine our features to focus on those exhibiting the most significant variation and correlations with our target variables.

To streamline our data preparation process, we segmented it into three distinct stages:

I. Preliminary feature Extraction and Manipulation:

In order to prepare our data for the model and add additional variables we implemented a couple of changes. Variables with categories were converted to factors. Variables with null values were imputed with the mean or with zero values. For e.g null values in cleaning fee were replaced with zero as it represented a lack of a cleaning fee. Another change made was to create secondary features that represent important business metrics. For e.g metrics such as price_per_person, metrics that consumers might take into account while deciding on an airbnb rental.

ID	Feature Name	Brief Description	R Code Line #
1	cancellation_policy	Updated original feature from the dataset by mapping ("strict", "super_strict_30") to strict and converted to factor	213
2	cleaning_fee	Original feature from dataset (replaced nulls to mean and converted to numeric)	214
3	price	Original feature from dataset (replaced nulls to mean and converted to numeric)	217
4	cleaning_fee	Original feature from dataset (replaced nulls to 0 and converted to numeric)	216
5	price_per_person	Feature engineered by dividing price by accommodates	219
6	accommodates	Original feature from dataset	295

7	has_cleaning_fee	Feature engineered to factor by using the following condition: if cleaning_fee == 0 then "NO" else "YES"	220
8	bed_category	Feature engineered to factor by using the following condition: if bed_type == "Real Bed" then "Bed" else "Other"	221
9	property_category	Feature engineered to factor, by using the original feature from the dataset - property_type based on several conditions	222
10	guests_included	Updated the original feature to factor	231
11	market	Updated original feature from the dataset by mapping na's to "Unknown" and if the market value is appearing less than 300 times, replaced it with "Other"	232
12	room_type	Updated the original feature to factor	233
13	ppp_ind	Feature engineered to factor, if the price per person factor is greater than median price per person, it is marked as 1 else 0	246
14	bathrooms	Original feature from dataset (replaced nulls to median and converted to numeric)	251
15	extra_people	Original feature from dataset (converted to numeric)	252
16	charges_for_extra	Feature engineered to factor by using the following condition: if cleaning_fee == 0 then "NO" else "YES"	253
17	host_response_rate	Original feature from dataset (converted to numeric)	254
18	host_acceptance_rate	Original feature from dataset (converted to numeric)	255
19	host_acceptance	Feature engineered to factor by using the following condition: if host_acceptance_rate is na then "Missing" if else host_acceptance_rate == 100 then "All" else "Some"	256
20	host_response	Feature engineered to factor by using the following condition: if host_response_rate is na then "Missing" if else host_response_rate == 100 then "All" else "Some"	258
21	has_min_nights	Feature engineered to factor by using the following condition: if minimum_nights > 1 then "YES" else "NO"	260

22	months_since_host_listed	Feature engineered to numeric by using the following process: used host_since to get the date and calculated the number of months the host has been in Airbnb and replaced the na by mean	276
23	average_listings_per_month	Feature engineered to numeric by using the following process: calculated the average listing the host has in Airbnb by host_total_listings_count dividing by months_since_host_listed and replaced na's by mean	277
24	Free_parking_flag	Feature engineered to factor by using the following condition: if amenities feature in the dataset has the word "Free Parking" then 1 else 0	285
25	Host_Is_Superhost_flag	Feature engineered to factor by using the following condition: if features feature in the dataset has the word "Host is Superhost" then 1 else 0	289
26	G_verified_flag	Feature engineered to factor by using the following condition: if host_verifications feature in the original dataset has the word "google" then 1 else 0	292
27	interaction_term	Feature engineered to an interaction term by using the following process: multiplied accommodates by bedrooms	295
28	accommodates_s	Feature engineered to an numeric by using the following process: raised the accommodates value to the power of 2	296
29	superhost_response_interaction	Feature engineered to an interaction term by using the following process: used interaction function on Host_Is_Superhost_flag and host_response	301
30	cleaning_fee_price_interaction	Feature engineered to an numeric by using the following process: if has_cleaning_fee feature = "Yes" then 1 else 0 which then multiplied by price	302
31	workspace	Feature engineered to factor by using the following condition: if amenities feature in the dataset has the word "workspace" then 1 else 0	306
32	dryer	Feature engineered to factor by using the following condition: if amenities feature in the dataset has the word "dryer" then 1 else 0	307

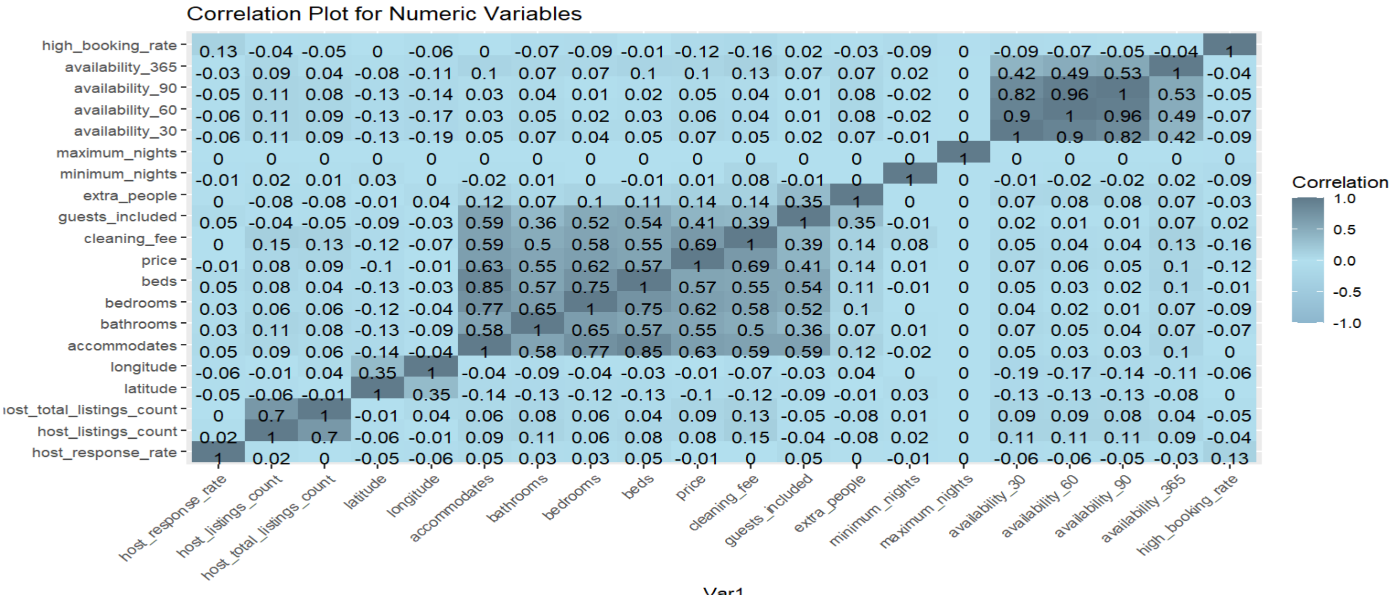
33	board_game	Feature engineered to factor by using the following condition: if interaction feature has the word "board game" then 1 else 0	308
34	home	Feature engineered to factor by using the following condition: if interaction feature has the word "home" then 1 else 0	309
35	wonderland	Feature engineered to factor by using the following condition: if neighborhood_overview from the dataset feature has the word "wonderland" then 1 else 0	311
36	description	Original feature from the dataset used for text mining	126
37	summary	Original feature from the dataset used for text mining	126
38	neighborhood_overview	Original feature from the dataset used for text mining	126
39	transit	Original feature from the dataset used for text mining	126
40	access	Original feature from the dataset used for text mining	126
41	house_rules	Original feature from the dataset used for text mining	126
42	city	Original feature from the dataset used for text mining	126

II. EDA and Identifying features with poor data quality that needed to be dropped:

To begin with, we identified features with a high percentage (over 40%) of missing data. A high percentage of missing values suggested that the variable had poor variation. Feature imputation for these variables was not an option as imputing more than 50% of the data with alternatives such as averages could potentially cause bias in the model. This led us to drop 9 features, dropping the total count of features from 61 to 53, the variables dropped included: notes, neighborhood_group, square_feet, weekly_price, monthly_price, license and jurisdiction_names.

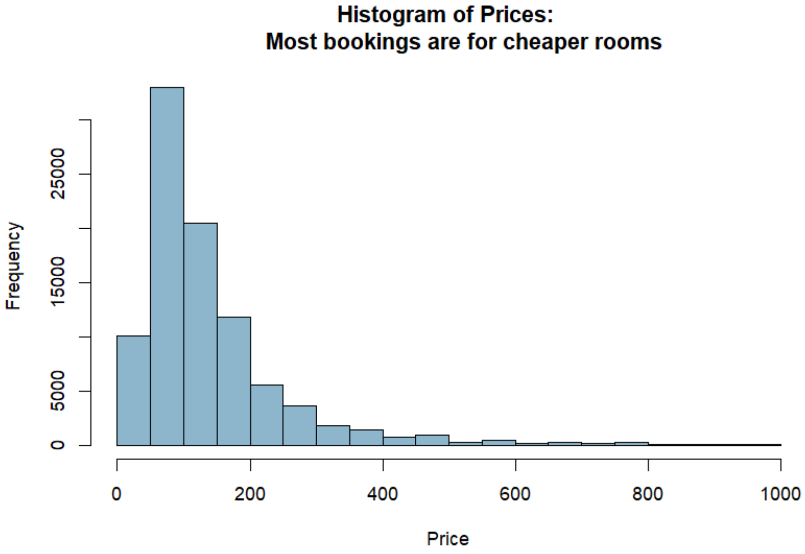
The second step was to eliminate features lacking variability, such as "experiences_offered," "country_code," and "country." These features contained only one value; for instance, "country_code" exclusively had "US" values. Given that the dataset solely contained data on Airbnb rentals solely within the United States, it negated the need for country-level analysis. This further reduced our dataset to 49 features.

Once we removed these basic features, we moved on to understanding the correlation of features in the dataset. This involved correlation at two levels, correlation of the explanatory variables with our target variables, high booking rate, and intercorrelation of variables. The aim was to remove features that have poor correlation and to drop features that have high intercorrelation and could cause multicollinearity issues. We ran a correlation plot to see how the correlations were distributed. The plot revealed that some features were highly correlated. For example the variable ‘beds’ was highly correlated (> 75% correlation) with the variables ‘accommodates’ and bedrooms. We decided to drop beds to reduce multicollinearity and reduce model variance. On the other hand, we had variables such as maximum_nights and host_listings_count that had poor correlation with the target variable, high_booking_rate. Based on the correlation values and logic outlined above we dropped another seven variables and landed on 42 variables (as seen below)

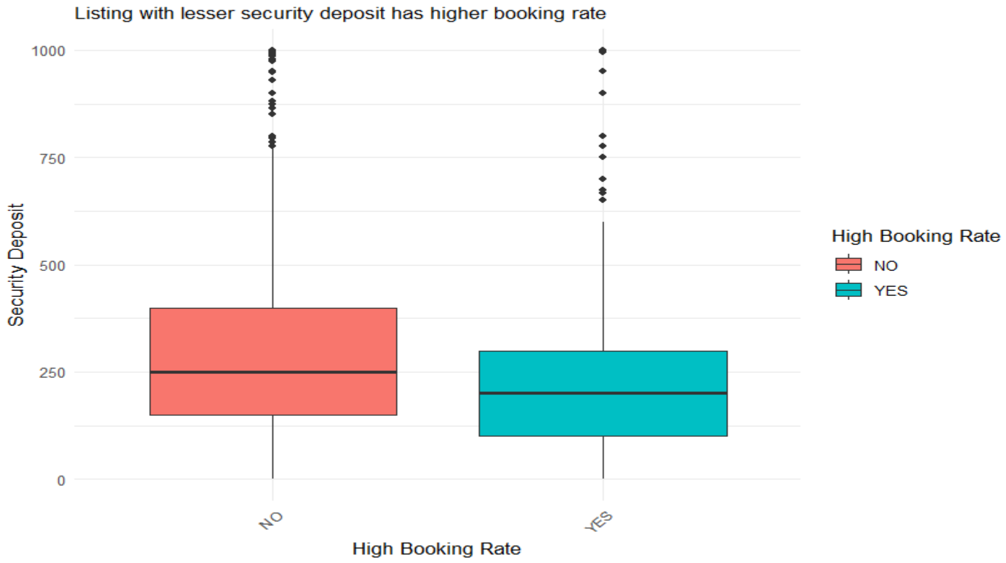


What quantitative factors should hosts take into account to increase booking rates?

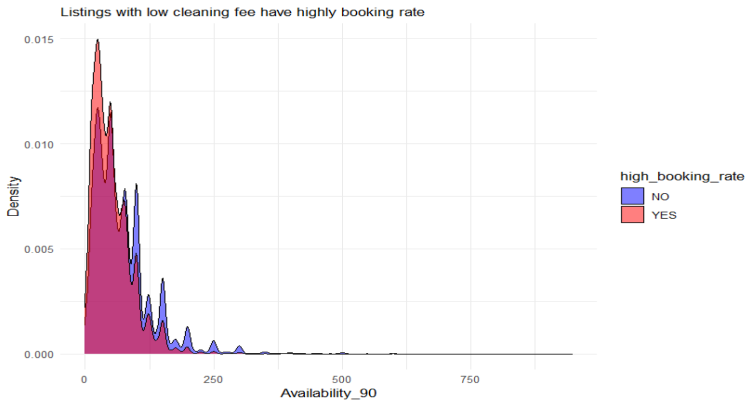
Identifying features based on EDA:



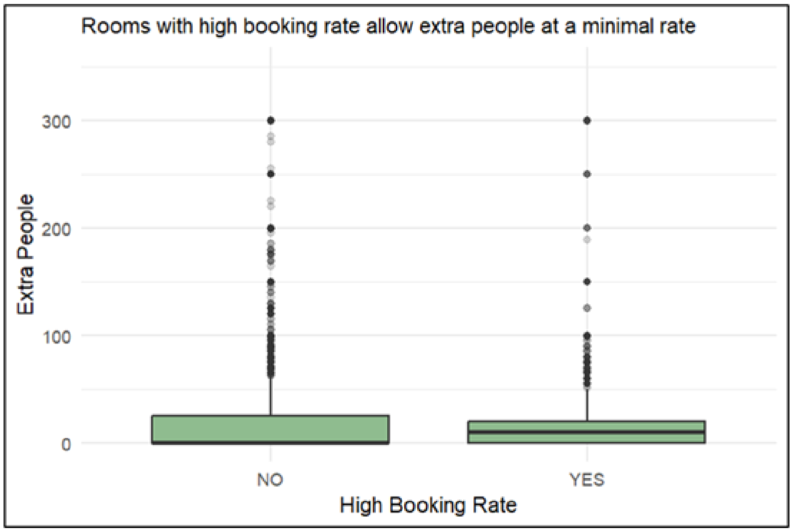
The histogram illustrates the distribution of prices for room bookings. It is evident that most bookings are concentrated at the lower end of the price range, with a significant decrease in frequency as the price increases. This suggests a preference for more budget-friendly options among consumers. The highest frequency of bookings occurs for rooms priced below $100, and the frequency steadily declines for rooms priced above $200.



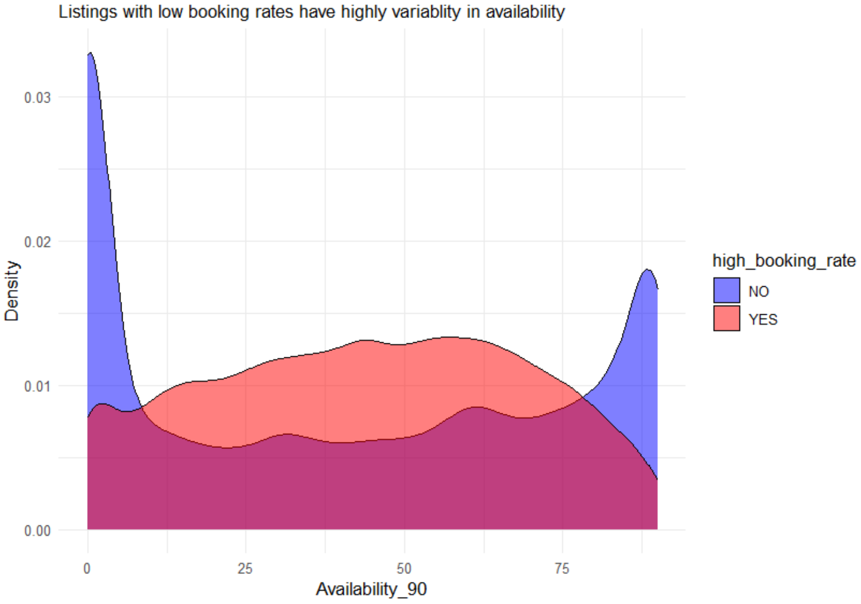
The boxplot illustrates a clear relationship between security deposit amounts and booking rates for rental listings. Listings with high booking rates tend to have notably lower security deposits, often with a wider range of amounts and some higher outliers. Conversely, listings with low booking rates generally require higher security deposits and exhibit a narrower range of amounts. This suggests that renters may prioritize listings with lower upfront costs, as indicated by the inverse relationship between security deposit amounts and booking rates.



The graph shows the connection between a listing's availability over 90 days and its booking rate (high or low). Listings with high booking rates are typically booked more often and have less availability overall. In contrast, there's no clear pattern for low booking rates - some listings may be rarely available, while others are almost always available.



The boxplot illustrates that regardless of a room's booking rate (high or low), it's unlikely to allow a large number of extra people. While there might be a few outliers allowing extra guests, most rooms in both categories don't accommodate many additional people. Rooms with high booking rates are slightly more likely to allow a larger number of extra people in some cases, as indicated by a few more outliers on that side of the graph.



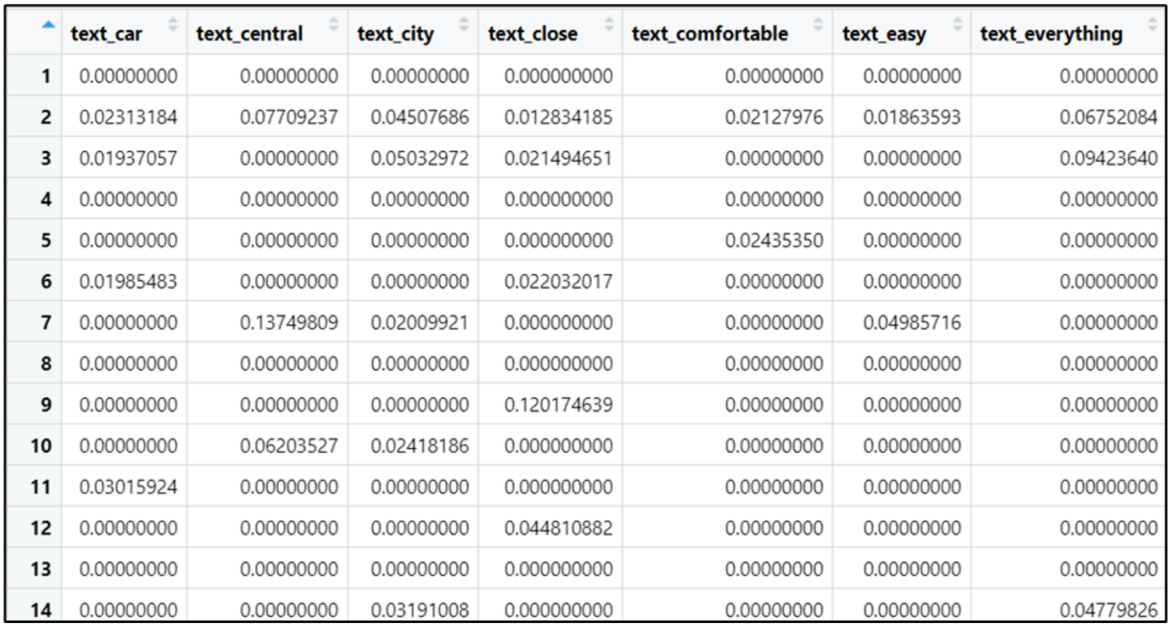
The graph illustrates that listings with high booking rates are typically less available, suggesting a negative correlation between booking rate and availability. However, for listings with low booking rates, availability varies widely with no discernible pattern, indicating that availability alone does not determine booking success for these listings.

III. Advanced feature extraction with text mining:

Within the finalized set of 42 preliminary features, we had seven text features that hosted a wealth of information on airbnb descriptions. Given this information, it was important to understand what qualitative factors lead customers to rent an airbnb. Is a customer more likely to avoid a rental if the host has a no smoking rule? Does the availability of a gym entice more customers? These are important factors to explore that require a deeper dive into the descriptive variables.

Our strategy with text mining was to use the TF-IDF approach. We began by creating a generic function to remove unwanted terms so that we could apply the function to the seven text features. This function consisted of a cleaning tokenizer that removed punctuation, numbers and stop words. Additionally it changed the terms to lower case and removed white space to avoid duplication. Post cleaning, the function gives each term a TF-IDF score and removes terms that are less than 2 characters (for e.g: “to”, “a”) and also removes infrequent terms (appears in less than 5 documents). We then applied the function to our seven text columns in the dataset: description, summary, neighborhood overview, transit, access, house rules and city. This gave us the table seen below with the TF-IDF scores for the text features for each document. We included these features in our model and then used our final XGboost model to understand the best features. More of which will be explored in section 4.

TF-IDF score matrix



	text_car	text_central	text_city	text_close	text_comfortable	text_easy	text_everything
1	0.00000000	0.00000000	0.00000000	0.000000000	0.00000000	0.00000000	0.00000000
2	0.02313184	0.07709237	0.04507686	0.012834185	0.02127976	0.01863593	0.06752084
3	0.01937057	0.00000000	0.05032972	0.021494651	0.00000000	0.00000000	0.09423640
4	0.00000000	0.00000000	0.00000000	0.000000000	0.00000000	0.00000000	0.00000000
5	0.00000000	0.00000000	0.00000000	0.000000000	0.02435350	0.00000000	0.00000000
6	0.01985483	0.00000000	0.00000000	0.022032017	0.00000000	0.00000000	0.00000000
7	0.00000000	0.13749809	0.02009921	0.000000000	0.00000000	0.04985716	0.00000000
8	0.00000000	0.00000000	0.00000000	0.000000000	0.00000000	0.00000000	0.00000000
9	0.00000000	0.00000000	0.00000000	0.120174639	0.00000000	0.00000000	0.00000000
10	0.00000000	0.06203527	0.02418186	0.000000000	0.00000000	0.00000000	0.00000000
11	0.03015924	0.00000000	0.00000000	0.000000000	0.00000000	0.00000000	0.00000000
12	0.00000000	0.00000000	0.00000000	0.044810882	0.00000000	0.00000000	0.00000000
13	0.00000000	0.00000000	0.00000000	0.000000000	0.00000000	0.00000000	0.00000000
14	0.00000000	0.00000000	0.03191008	0.000000000	0.00000000	0.00000000	0.04779826

Section 4.1: Evaluation and Modeling

The model evaluations included two aspects. The first aspect was understanding which model to pick and which parameters to choose. The second part involved model evaluation methods and choosing the best method for cross validation based on the finalized model.

I. Winning Model Description:

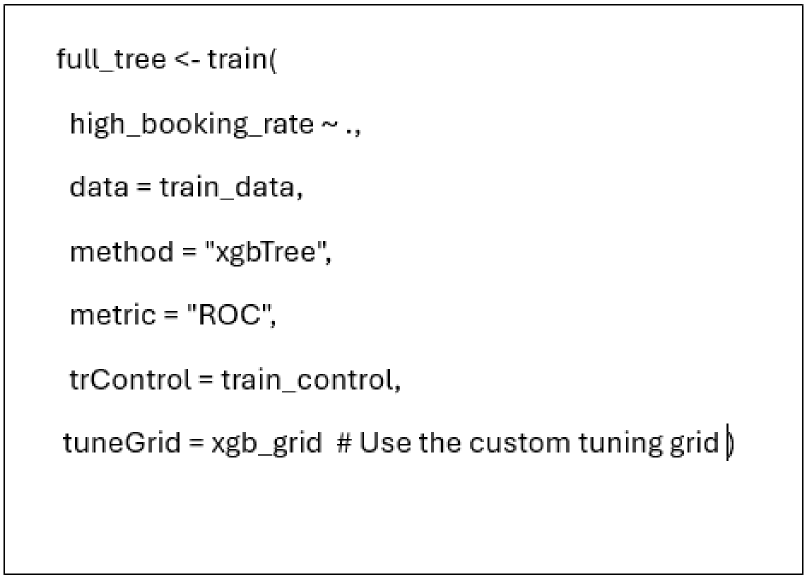
The winning model in our contest is an XGBoost model, a powerful and flexible machine learning algorithm known for its efficiency and performance. The model was trained using a variety of features, including but not limited to features related to property characteristics, location, pricing, booking history, and customer behavior. After thorough experimentation and validation, the XGBoost model demonstrated superior performance in both training and generalization phases, achieving high accuracy and robustness on unseen data. The decision to choose XGBoost as the winning model was based on its consistently strong performance across various evaluation metrics, including accuracy AUC score, as well as its ability to handle complex relationships and non-linear patterns in the data. In our R code, the final predictions generated by the XGBoost model for submission to the contest can be found in line numbers 650 - 653. And the model evaluation was done from lines 444 - 480.

II. Exploring Various Models and Metrics of the Winning Model :

Model	AUC
Logistic Regression (w/o Text features)	*0.76*
Decision Tree (w/o Text features)	*0.80*
Random forest (w/o Text features) - folds = 2	*0.82*
Random forest (w Text features) - folds = 2	*0.84*
Random forest (w Text features) - folds = 5	*0.85*
Xgboost (w text features) - rounds = 500	*0.87*
Xgboost (w text features) - rounds = 600	*0.88*

We designed our code to ensure ease of switching between models, making evaluation easier and clear. The following code snippet was used:

1. Changing method = "lm", metric = "RMSE" and dropping tuneGrid allowed us to perform linear regression
2. Changing method = "glm", family = "binomial" and dropping tuneGrid allowed us to perform Logistic Regression
3. Using method = “rpart” and dropping tuneGrid allowed us to perform Decision Trees
4. Using method = “rf” and dropping tuneGrid allowed us to perform Random Forest
5. Finally the above code was used to perform XGBoost. The model is at line numbers 393-400



```
full_tree <- train(

 high_booking_rate ~ .,

 data = train_data,

 method = "xgbTree",

 metric = "ROC",

 trControl = train_control,

tuneGrid = xgb_grid  # Use the custom tuning grid )
```

Our winning model specification was Xgboost with the following features: 600 rounds, a depth of 6, learning rate of 0.1, gamma = 0, 0.75 subsample ratio of columns, 3 minimum sum of instance weight and subsample of 0.75. This model gave us our highest AUC at 0.8743.

But to get to these specifications we had to tune and test various model specifications. Our main metric of evaluation with all models was the AUC as that was the target metric for maximization.

We initiated our model testing by starting with a logistic regression model without text features. This initial attempt gave us a good AUC score of 0.76. However, we remained apprehensive about its suitability, considering its limitation in capturing non-linear relationships. To try an alternative, we opted for a decision tree model next, without text features in the initial trial. The decision tree outperformed our expectations, achieving an improved AUC score of 0.80. Despite this progress, we recognized that both models exhibited susceptibility to overfitting or underfitting, indicating the need for further refinement and evaluation.

To avoid or reduce fitting issues we decided to try boosting and bagging methods. We first experimented with a Random Forest model. Given that a random forest model is computationally intensive, we decided to first test the model without text features. The first round with two folds gave us an AUC improvement of 0.2 and increased our AUC to 0.82 from 0.80 (with decision trees). To further improve the model, we tried the model with text features. For the text features, we included the TF-IDF value for every document for each text variable. We then ran the random forest with two and five folds. As seen below, the five fold model outperformed the two fold model with a higher AUC of 0.85.

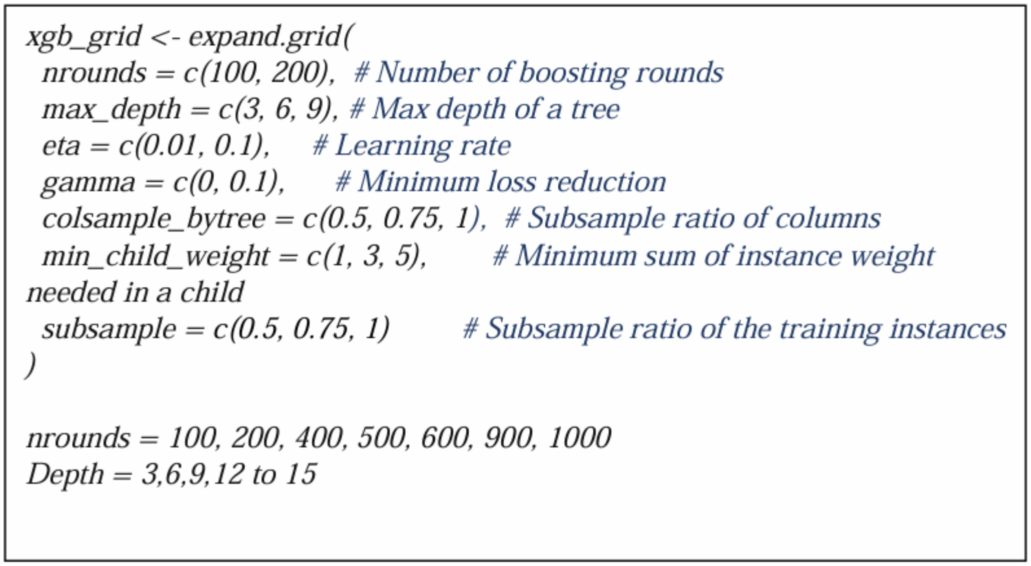
While random forest gave us high AUC scores, we found that after a while, the AUC values were plateauing around 0.85. The random forest method was also very computationally intensive with each model variation taking more than 30 minutes to run. This also made it very difficult to compare and contrast different variations of the model. Hence, in order to try an alternative model we tried boosting.

To ensure that our models were not biased, and to test the results on different variations, we employed a range of evaluation methods. Since we did not have a test y dataset to compare predictions with, we began with splitting our train dataset into 70% train and 30% test data to evaluate model strength. This was the basic split that we used while evaluating our preliminary models: logistic regression and decision trees.

Our second approach was to use cross validation metrics when comparing the more advanced (ensemble) models, random forest and XGboost. For the Random forest and XG boost we tried 2 and 5 fold cross validation. With random forest we noticed a significant improvement in performance with 5 folds (as mentioned above and seen in the table above). With XGboost, both 2 and 5 folds provided similar results, with 5 folds giving only a slight 0.1/0.2 improvement in AUC. Given that 5 fold variation took longer to run, we decided to stick to 2 fold variation.

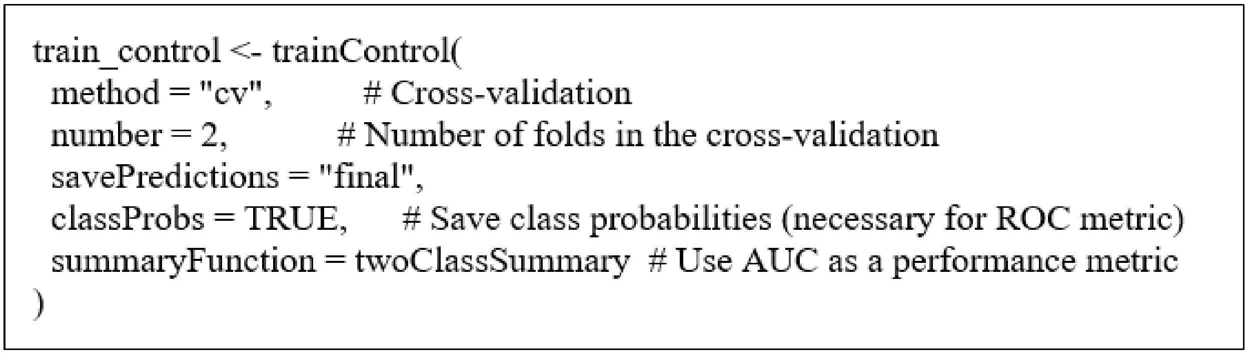
For the XGboost model, we also used a grid search method as explained above. We found that after 600 rounds the variation in AUC was not extensive. Hence, in the interest of time and model complexity we capped the number of rounds at 600 and stuck to 2 fold variation.

We experimented with the following hyperparameter variations for the XGboost model:



```
xgb_grid <- expand.grid(
  nrounds = c(100, 200),  # Number of boosting rounds
  max_depth = c(3, 6, 9), # Max depth of a tree
  eta = c(0.01, 0.1),     # Learning rate
  gamma = c(0, 0.1),      # Minimum loss reduction
  colsample_bytree = c(0.5, 0.75, 1),  # Subsample ratio of columns
  min_child_weight = c(1, 3, 5),        # Minimum sum of instance weight
needed in a child
  subsample = c(0.5, 0.75, 1)        # Subsample ratio of the training instances
)

nrounds = 100, 200, 400, 500, 600, 900, 1000
Depth = 3,6,9,12 to 15
```



```
train_control <- trainControl(
  method = "cv",          # Cross-validation
  number = 2,           # Number of folds in the cross-validation
  savePredictions = "final",
  classProbs = TRUE,      # Save class probabilities (necessary for ROC metric)
  summaryFunction = twoClassSummary  # Use AUC as a performance metric
)
```

We used the following 2 code snippets to alter our hyperparameters and perform Cross Validation with multiple folds including 2,5 and 10.

This gave us our winning model combination, Xgboost with the following parameters: 600 rounds, a depth of 6, learning rate of 0.1, gamma = 0, 0.75 subsample ratio of columns, 2 fold cross validation, 3 minimum sum of instance weight and subsample of 0.75. This model gave us our highest AUC of 0.88. We preferred XGboost to Random Forest for a host of reasons. Primarily, the XGboost model was quicker to run and easier to tune hyperparameters for. Additionally, it gave us a higher AUC value. The XGboost model may have given us better results than Random Forest as it prioritizes reduction of misclassified values by focusing on the mistakes made by previous trees. Random Forest on the other hand samples different variations of datasets but runs independent trees. As our focus was on increasing accuracy and reducing misclassification to improve AUC, XGboost gave us the best results.

We used code snippets from the lines 649-653 in the final file to generate the predictions file.

Libraries used in modeling

For our model building we used the following list of libraries:

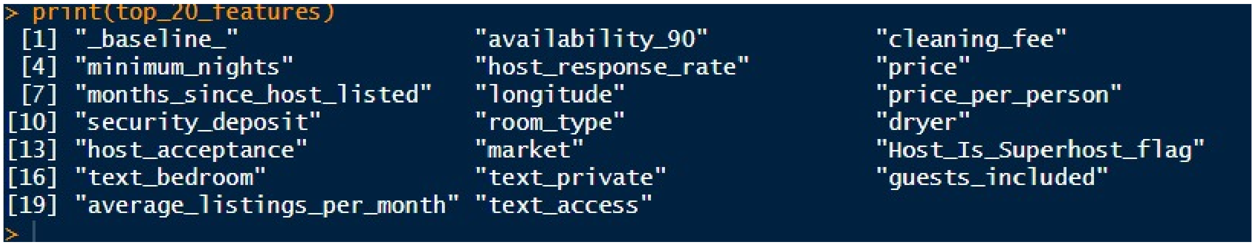
1. tm
2. SnowballC
3. text2vec

4. tidyverse
5. caret
6. lubridate
7. tree
8. class
9. caret
10. ggplot2
11. ROCR
12. rang

Best Features:

We computed our top features based on the metric of drop loss from XgBoost. Drop loss represents a minimum improvement in logloss, the features below represent the top 20 features that gave us the best improvement in log loss.

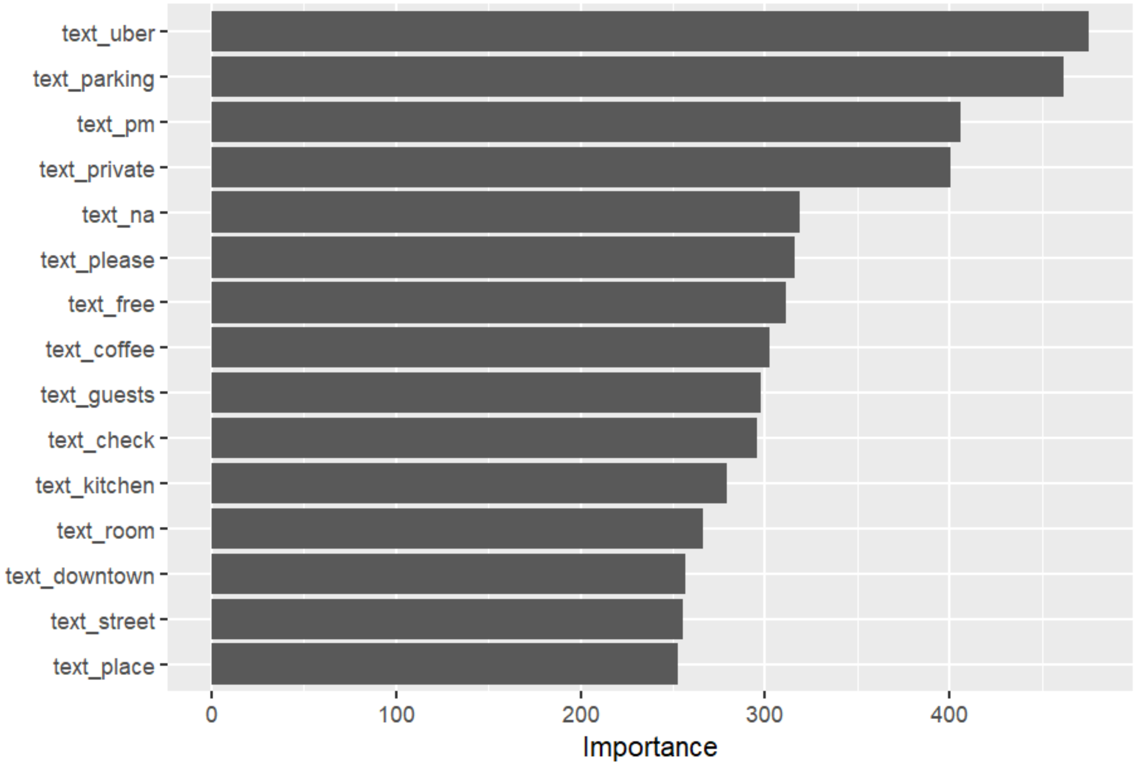
Our top 20 features based on drop loss:



```
> print(top_20_features)
 [1] "_baseline_"                "availability_90"          "cleaning_fee"
 [4] "minimum_nights"            "host_response_rate"       "price"
 [7] "months_since_host_listed"  "longitude"                "price_per_person"
[10] "security_deposit"          "room_type"                "dryer"
[13] "host_acceptance"           "market"                   "Host_Is_Superhost_flag"
[16] "text_bedroom"              "text_private"             "guests_included"
[19] "average_listings_per_month" "text_access"
> 
```

In order to check the top text features we ran a random forest using the text features and the high booking rate variable:

The top 15 text features are seen below:



From the plot above, it appears that features related to location and airbnb facilities are some of the most important factors correlated with high booking rates. Adding descriptions on location advantages such as easy availability of Ubers or mentioning downtown proximity entices more customers. Additionally, added free in house facilities are also more likely to increase booking rates for a listing. As seen above the use of the word 'free' itself in an airbnb description increases booking rates. Some of the other facility related top features seen in the plot above include facilities such as free kitchen and parking.

Section 5: Reflection

Reflecting on our project, we've garnered several insights into the process of data cleaning, feature engineering, modeling and practical application. This reflection encompasses what we excelled at, the challenges we faced, areas for potential improvement, and advice for future groups.

Our team leveraged individual strengths to handle each project phase comprehensively, from data cleaning to model implementation, with regular meetings ensuring efficient task distribution. We placed significant emphasis on feature selection and conducted extensive exploratory data analysis (EDA) and data cleaning early on. This helped us manage missing data and enhance our dataset's quality, forming a solid foundation for our predictive models.

Our team put considerable effort into selecting the most effective model for our predictive task. We implemented a range of machine learning algorithms, including Linear Regression, Logistic Regression, Decision Trees, Random Forest, and XGBoost. Each model underwent rigorous cross-validation to ensure its robustness across different data subsets. Our focus was on identifying the model with the highest AUC, reflecting our commitment to prioritizing predictive accuracy. By fine-tuning the models and evaluating their performance metrics, we aimed to make data-driven decisions and achieve optimal outcomes in our predictive modeling endeavor.

Main Challenges:

Our main challenges predominantly revolved around three key areas: Exploratory Data Analysis (EDA), text mining, and feature engineering.

Firstly, during the EDA phase, we encountered difficulties in understanding the underlying patterns and distributions within the dataset. Uncovering meaningful insights from the data proved to be challenging due to its complexity and size.

Secondly, incorporating text mining techniques posed a significant hurdle, particularly in extracting relevant information from unstructured text data such as property descriptions and reviews. Managing the vast amount of textual information and transforming it into structured features required careful consideration and innovative approaches.

Finally, feature engineering presented its own set of challenges, particularly in determining which features would be most informative for our predictive models. Crafting meaningful features that effectively captured the underlying relationships in the data while avoiding overfitting demanded thorough domain knowledge and creativity.

Overcoming these challenges necessitated a comprehensive understanding of the data, expertise in text mining methodologies, and adeptness in feature engineering techniques.

Opportunities for Improvement:

If we were to start the project from scratch, we'd definitely rethink how we handled our variables. Initially, we focused on selecting the most effective ones, but looking back, we realized we might have missed out on valuable insights by doing so. Keeping all variables would avoid loss of information and could have given us a more comprehensive understanding of our data.

Additionally, we could have tried extensive text mining and sentiment analysis wherever possible. Text data always holds a treasure trove of information, and by leveraging it to the full potential, we could enhance the model , discover a different trend or find support for our interpretation of the model . While we did implement some text mining, we did not cover sentiment analysis and would have definitely explored that given more time.

Future Enhancements:

Given a few more months on the project, we would focus on:
Extending the project to include Lasso and Ridge regression could enhance our model significantly. Lasso would improve feature selection and model simplicity, while Ridge would stabilize the model by addressing multicollinearity. Using both techniques with optimized cross-validation would refine our predictions and increase analytical depth.

We could also explore importing and integrating external datasets to have more data, leading to more information and finally a more confident prediction.

Real-World Application: Developing a prototype or a minimal viable product (MVP) to demonstrate the model's application, gathering user feedback to refine the approach.Analyze user interaction data to identify common usage patterns and potential areas for improvement. This data can help understand which features are used most and which are ignored or cause confusion.

Advice for Future Teams:
Understanding the business context would make the EDA efficient and easier. Understand the domain deeply to ensure the relevance and applicability of your model.
Investing time in data exploration can provide critical insights and should be prioritized.
We would also like to emphasize the importance of feature engineering and the potential it holds. Take the time to explore, experiment, and craft features that truly capture the essence of the data.