UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

# BUDT 758T Final Project Report Spring 2023

## Section 1: Team member names and contributions

***1. Pruthvi Shyam Billa*** - Worked on the R script code: Building and evaluating different models, submitted predictions for the contest, organized team status calls, and formatted the final report.

***2. Afia Simeen*** - Successfully wrote, implemented, and tested code for predictions using various models for the contest, generated predictions for the test data, created necessary tables and charts to analyze the feature variables, and provided comprehensive documentation for the models used.

***3. Amuktha Malyada Nadipelli*** - Successfully incorporated the required external data source, performed analysis on learning curves, experimented with different tuning parameters to improve the AUC, and worked on documentation required for the report.

***4. Thanmai Reddy Kadire*** - Conducted extensive model experimentation, parameter tuning, cross-validation, and comparison of six models to optimize AUC. Meticulously documented all relevant details for inclusion in the report.

***5. Lekhya Mudda*** - Worked on multiple models to optimize the AUC, diligently worked on fitting curves to compare their performance, contributed to the report and documentation, providing valuable insights for the project.

## Section 2: Business Understanding

***Executive Summary:***

This report presents a business case for a predictive model that aims to forecast the high booking rate of Airbnb listings. The model was built using the R programming language, specifically employing boosting techniques with various features such as accommodation capacity, number of beds, and price. The goal of this model is to provide valuable insights to individuals and businesses in the Airbnb industry, enabling them to optimize their listing strategies and maximize booking rates.

The target audience for this predictive model includes both individual Airbnb hosts and larger property management companies. By leveraging the model's predictions, these stakeholders can make informed decisions to increase the probability of high booking rates and maximize their rental income. Additionally, travel agencies and vacation rental platforms can benefit from incorporating this model into their recommendation systems, improving the overall user experience and satisfaction.

The model output can drive several actionable business strategies:

*1.* ***Pricing Optimization****:* By analyzing the impact of pricing on the booking rate, hosts and property managers can adjust their rates accordingly. The model can identify the optimal price range for maximizing bookings while ensuring competitive pricing in the market.

*2.* ***Listing Optimization****:* The model's insights on accommodation capacity, number of beds, and other features can guide hosts and property managers in optimizing their listings. This includes identifying the ideal number of guests a listing can accommodate, adjusting the number of beds, or adding amenities to enhance the attractiveness of the property.

*3.* ***Marketing Campaigns****:* With the model's predictions, marketing efforts can be strategically targeted towards listings with a high probability of achieving a high booking rate. This allows hosts and property managers to allocate their resources effectively and increase the visibility of listings with greater potential.

*4.* ***Inventory Management****:* Property management companies can utilize the model to forecast demand and adjust their inventory accordingly. This enables them to allocate resources efficiently and ensure an optimal balance between supply and demand.

The predictions generated by this model have the potential to generate significant value for Airbnb hosts, property management companies, and the broader vacation rental industry. By leveraging the insights provided, businesses can improve their profitability, enhance customer satisfaction, and gain a competitive edge in the market.

In conclusion, the predictive model presented in this report offers valuable insights for optimizing Airbnb listing strategies. By leveraging the model's predictions, hosts and property management companies can make data-driven decisions to increase their booking rates and maximize rental income. This model has the potential to generate substantial value and drive actionable business actions within the Airbnb industry.

Based on the insights provided by the predictive model, here are two important recommendations for Airbnb:

*1.* ***Enhanced Listing Guidance****:* Airbnb should provide hosts with comprehensive guidance on optimizing their listings based on the model's predictions. This guidance could include specific recommendations on pricing, accommodation capacity, and other relevant features. By equipping hosts with data-driven insights and actionable recommendations, Airbnb can help hosts maximize their booking rates and improve the overall guest experience.

*2.* ***Dynamic Pricing Algorithm****:* Airbnb should develop and implement a dynamic pricing algorithm that takes into account real-time market conditions, demand patterns, and the predictions generated by the predictive model. This algorithm would enable hosts to automatically adjust their pricing to match the optimal price range for maximizing bookings. By continuously analyzing and adapting to market dynamics, hosts can ensure competitive pricing and maximize their revenue potential.

By implementing these recommendations, Airbnb can empower its hosts to make informed decisions and optimize their listings, leading to increased booking rates, improved customer satisfaction, and overall business growth.

## Section 3: Data Understanding and Data Preparation

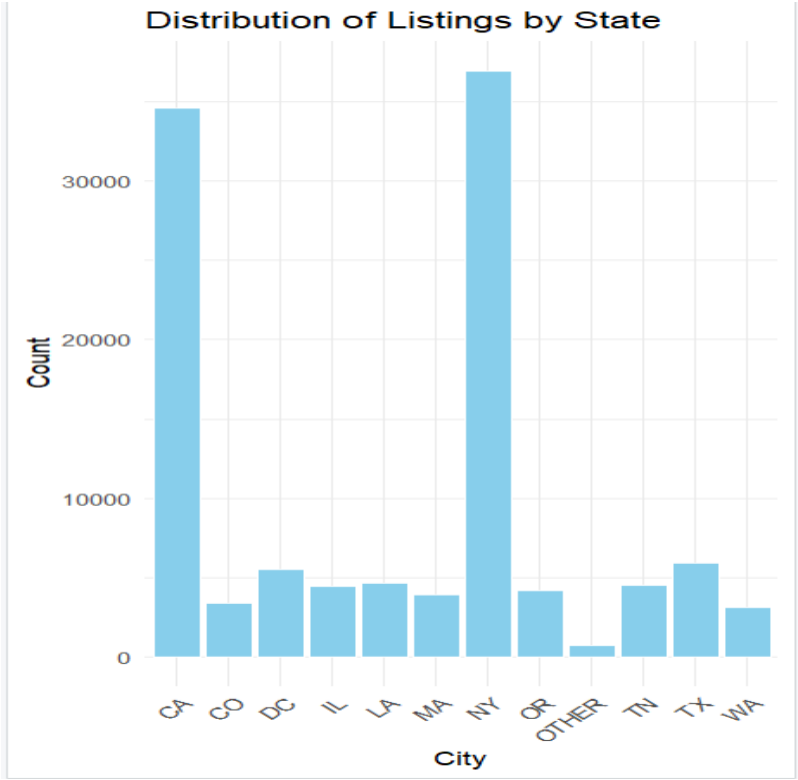
1) *A table for feature variables used:*

| ID | Feature Name | Brief Description | R Code Line Numbers |
|---|---|---|---|
| 1 | accommodates | Original feature from dataset | 782 - 794 |
| 2 | availability_30 | Original feature from dataset | 782 - 794 |
| 3 | availability_90 | Original feature from dataset | 782 - 794 |
| 4 | availability_365 | Original feature from dataset | 782 - 794 |
| 5 | bathrooms | Original feature from dataset | 151 - 152 |
| 6 | bed_category | New added feature | 209, 250 |
| 7 | bedrooms | Original feature from dataset | 147 |
| 8 | beds | Original feature from dataset | 148 |
| 9 | charges_for_extra | New added feature | 219 |
| 10 | host_total_listings_count | Original feature from dataset | 171 |
| 11 | host_acceptance | New added feature | 220 |
| 12 | host_response | New added feature | 221 |
| 13 | has_min_nights | New added feature | 222 |
| 14 | host_is_superhost | Original feature from dataset | 153 - 154 |
| 15 | has_guests_included | New added feature | 226 |
| 16 | host_has_profile_pic | Original feature from dataset | 164 - 165 |
| 17 | has_security_deposit | New added feature | 229 |
| 18 | host_response_time | Original feature from dataset | 159 - 160 |
| 19 | has_monthly_price | New added feature | 228 |
| 20 | has_weekly_price | New added feature | 227 |
| 21 | has_minnight_rental | New added feature | 231 |
| 22 | has_maxnight_rental | New added feature | 232 |
| 23 | is_business_travel_ready | Original feature from dataset | 167 - 168 |
| 24 | is_location_exact | Original feature from dataset | 786 |
| 25 | instant_bookable | Original feature from dataset | 786 |
| 26 | market | Original feature from dataset | 307 - 322 |
| 27 | price | Original feature from dataset | 120 |
| 28 | ppp_ind | New added feature | 246, 253 |
| 29 | property_category | New added feature | 210 - 216 |
| 30 | require_guest_phone_verification | Original feature from dataset | 787 |
| 31 | square_feet | Original feature from dataset | 157 |
| 32 | requires_license | Original feature from dataset | 177 |
| 33 | bed_convenience | New added feature | 197 - 198 |
| 34 | overpriced_market | Original feature from dataset | 329 - 334 |
| 35 | host_identity_verified | Original feature from dataset | 173 - 174 |

| | | | |
|---|---|---|---|
| 36 | bathroom_luxury | New added feature | 233 |
| 37 | privacy | New added feature | 234 |
| 38 | cluster | New added feature | 736 - 759 |
| 39 | sent_score | New added feature | 679 - 689 |
| 40 | high_deposit | New added feature | 394 - 398 |
| 41 | overpriced_category | New added feature | 265 - 268 |
| 42 | verification_for_underpricing | New added feature | 341 - 345 |
| 43 | privacy_accomodates | New added feature | 201 - 202 |
| 44 | total_price | New added feature | 272 - 274 |
| 45 | year | New added feature | 280 - 282 |
| 46 | host_experience | New added feature | 288 |
| 47 | host_is_experienced | New added feature | 292 - 293 |
| 48 | zipcode | Original feature from dataset | 130 - 132, 403 - 421 |
| 49 | sent_score_neighbour | New added feature | 721 - 729 |
| 50 | host_preferred | New added feature | 295 - 296 |
| 51 | host_preferred_accceptance | New added feature | 298 - 300 |
| 52 | host_preferred_location | New added feature | 301 - 302 |
| 53 | population | New added feature | 73 |
| 54 | room_type | Original feature from dataset | 252 |
| 55 | property_type | Original feature from dataset | 255 - 256 |
| 56 | extra_people | Original feature from dataset | 123 - 125 |
| 57 | weekly_price | Original feature from dataset | 128 |
| 58 | monthly_price | Original feature from dataset | 127 |
| 59 | minimum_nights | Original feature from dataset | 792 |
| 60 | maximum_nights | Original feature from dataset | 793 |
| 61 | cancellation_policy | Original feature from dataset | 116 - 117 |
| 62 | city_name | Original feature from dataset | 258 - 259 |
| 63 | cleaning_fee | Original feature from dataset | 119 |
| 64 | guests_included | Original feature from dataset | 793 |
| 65 | require_guest_profile_picture | Original feature from dataset | 793 |

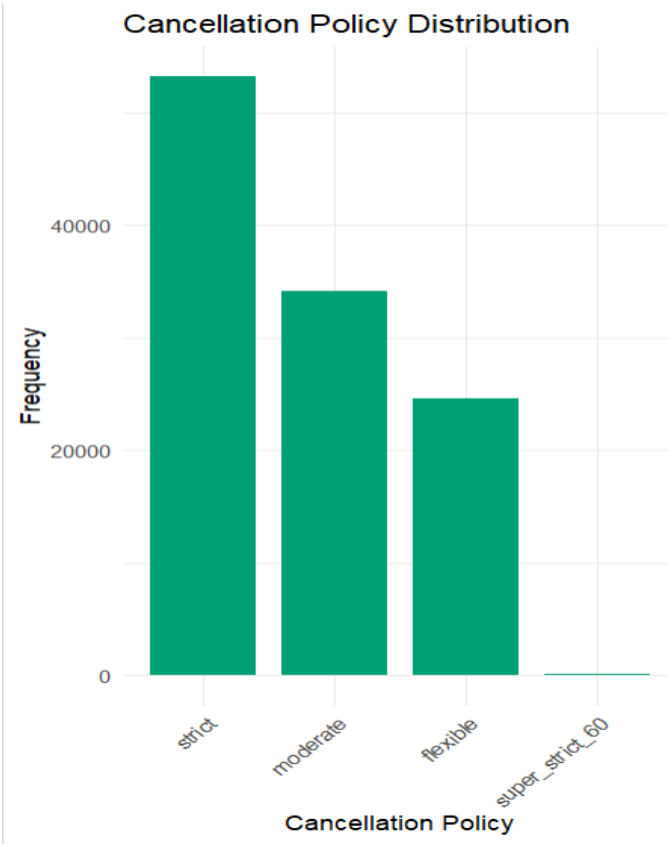
2) *Graphs or tables regarding features in the dataset:*

1. *Distribution of Airbnb listings by state*



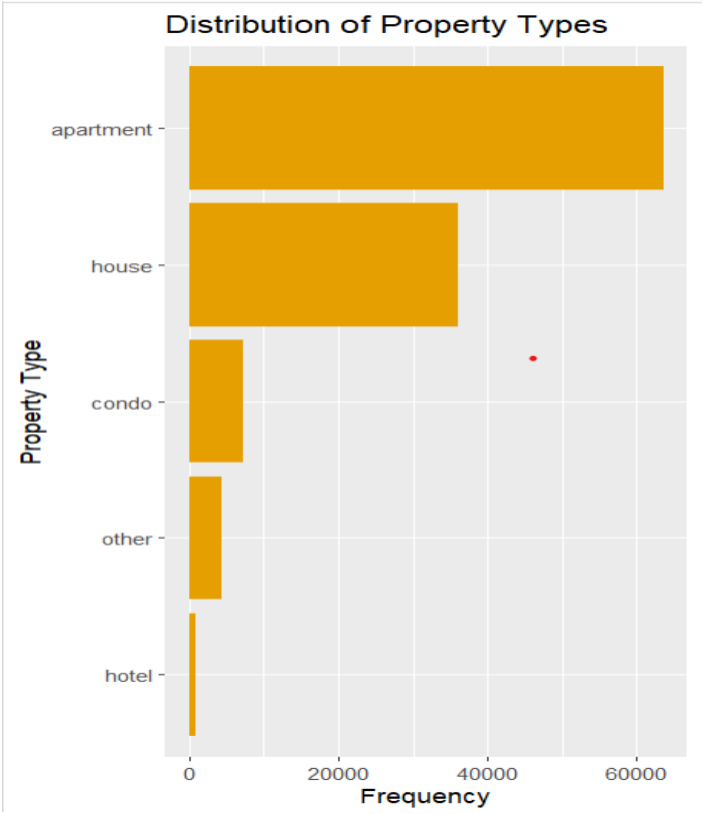

From the graph above, we can clearly observe that New York has the largest proportion of property listings on Airbnb followed by California. Also, other states in the US have no more than 5000 property listings on Airbnb except for D.C. and Texas which demonstrate a slightly higher figure.

2. *Distribution of Airbnb Listings by Cancellation Policy*



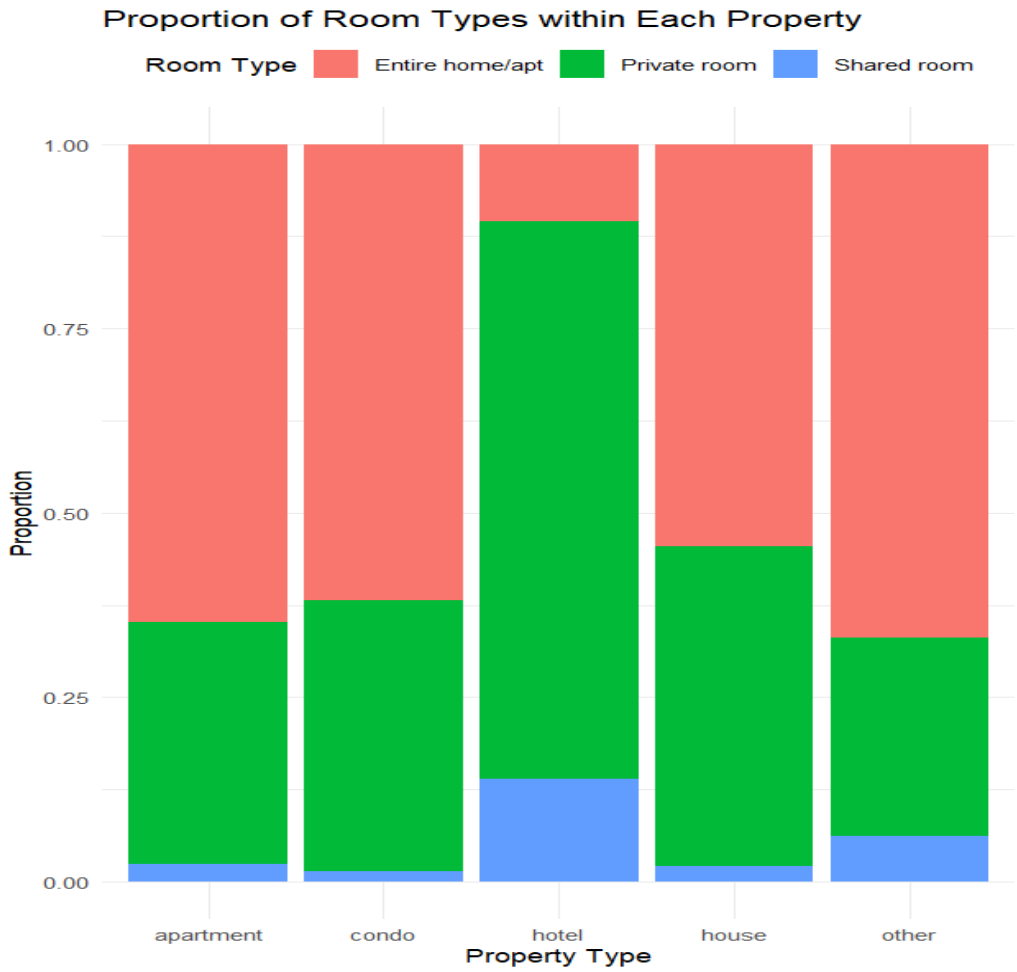

From the graph above, we can observe that most property listings on Airbnb have a strict cancellation policy. Also, a considerable portion of these listings also have moderate or flexible cancellation rules.

3. *Distribution of Property Type Listings on Airbnb*



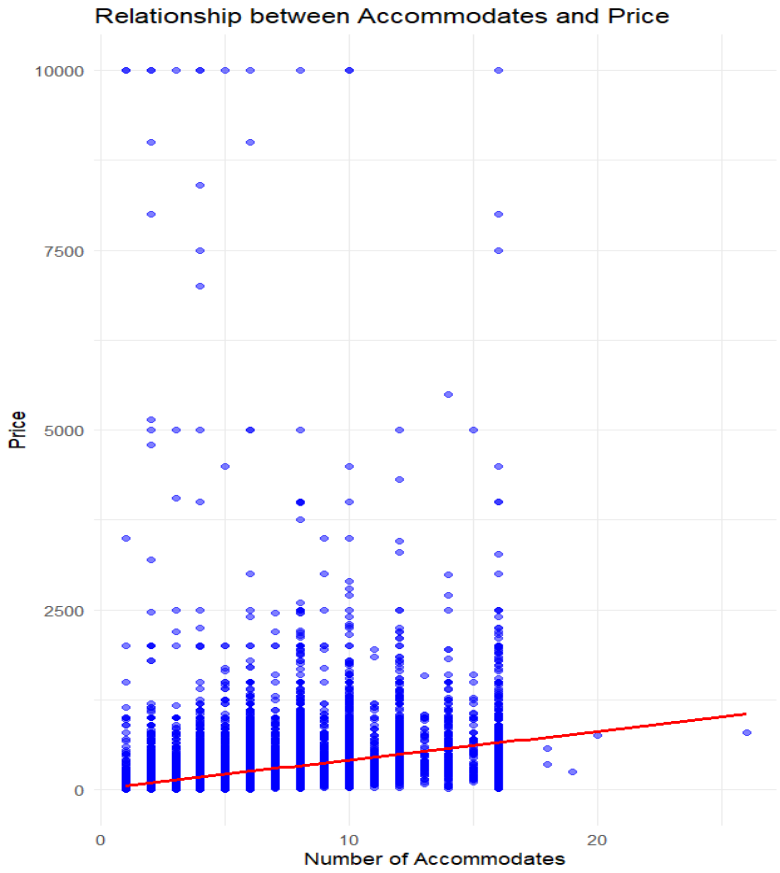

From the graph above, we can determine that a significant portion of property listings on Airbnb are apartments followed by independent homes hosting guests. Also, properties such as condos, hotels, and other estates have fewer listings on Airbnb compared to other categories.

4. *Distribution of Room Types in Each Property Category on Airbnb*



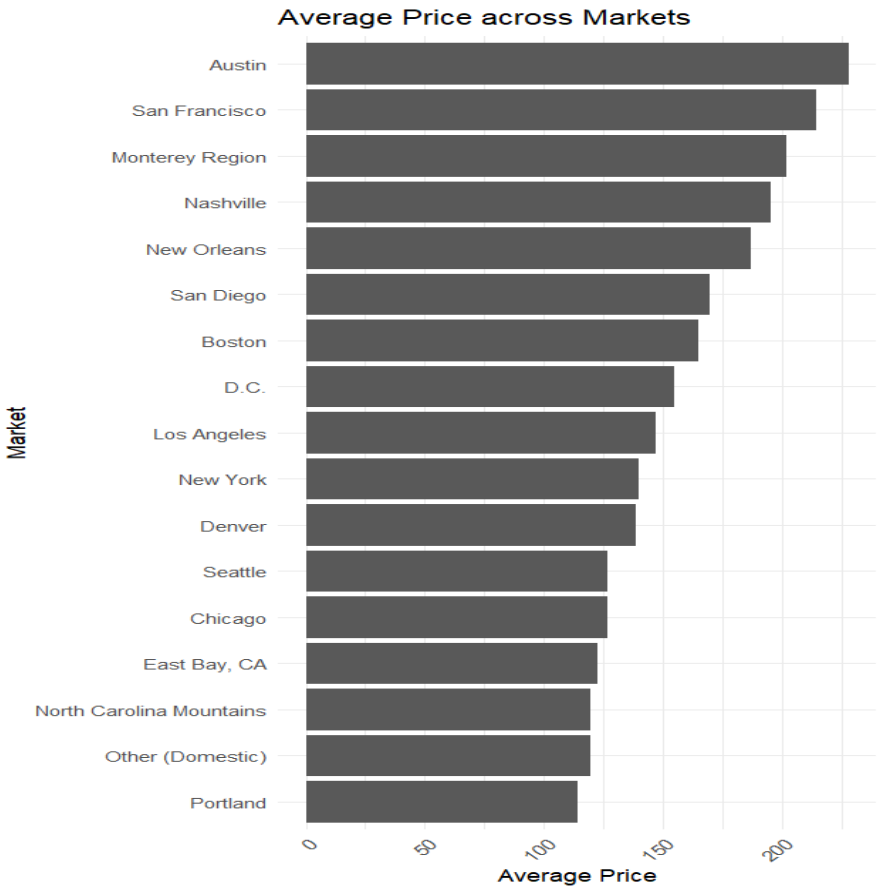

From the graph above, we can observe that a substantial percentage of apartment listings on Airbnb offer the entire property for leasing to its customers. Additionally, other property listings such as condos, individual homes, and other estates also exhibit a similar trend except for hotels which significantly offer more private or individual spaces to the guests for their stay.

5. *Analyzing the relationship between number of accommodates and pricing of listing*



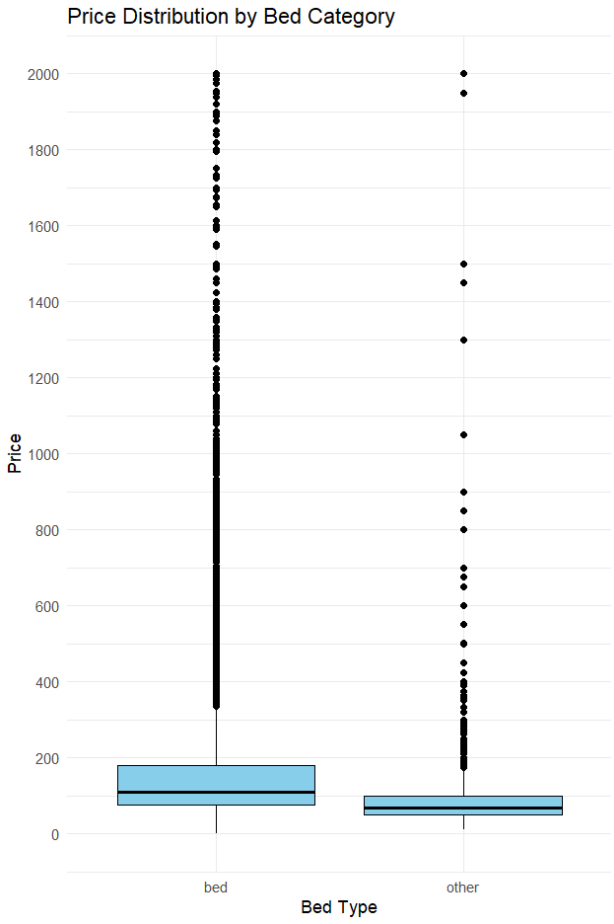

From the graph above, we can observe that as the number of accommodates increase the price of the Airbnb listings also increases. The relationship between these two variables follows a directly proportional linear trend. Also, a few outliers exist where the price of the Airbnb listing is significantly higher compared to the number of people it can accommodate.

*6. Distribution of Average Listing Price across Markets on Airbnb*



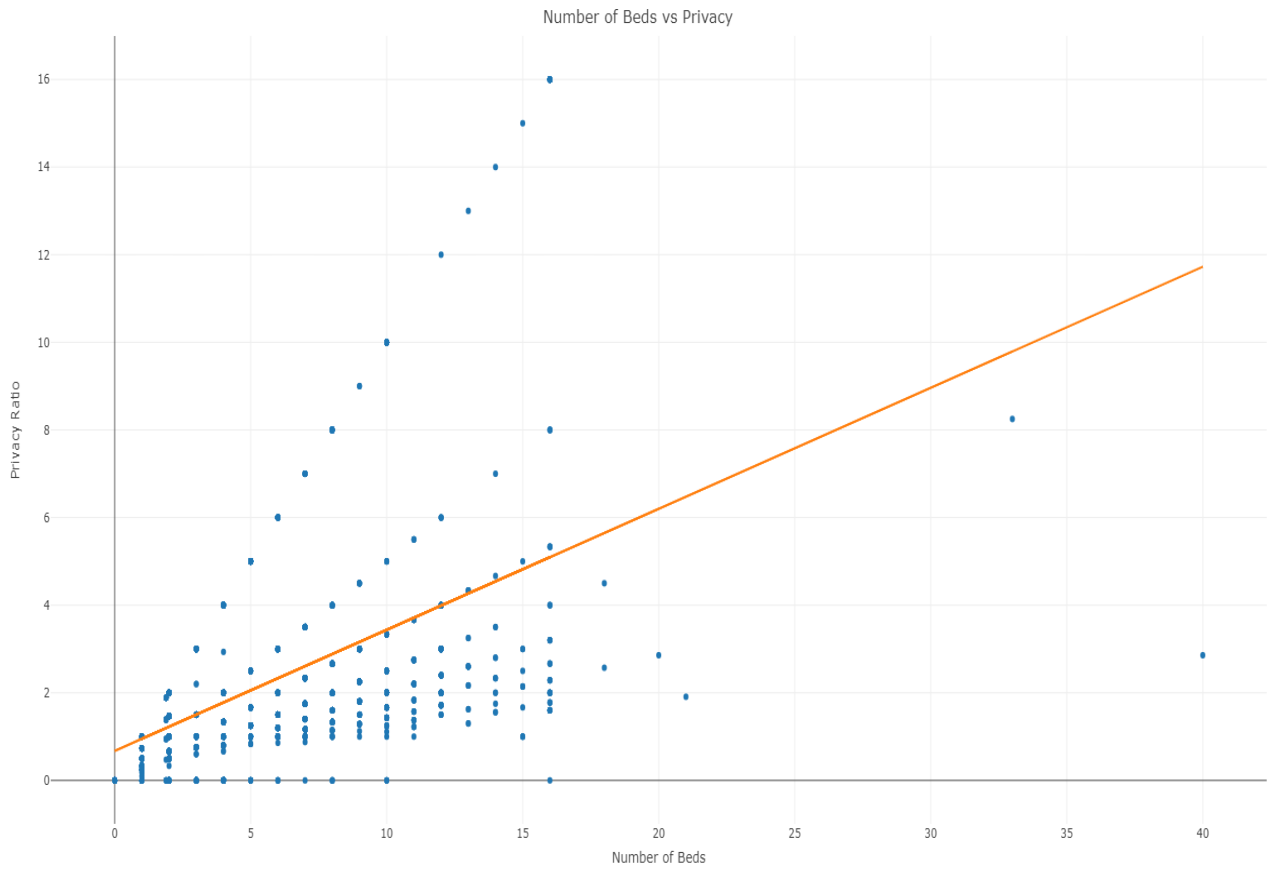

From the graph above, we observe that Austin has the highest average listing price for Airbnb properties followed by San Francisco, Monterey, and Nashville. Other metropolitan cities such as New York, Boston, and Seattle have surprisingly lower mean prices for Airbnb estates compared to other markets.

*7. Price Distribution by Bed Category on Airbnb*



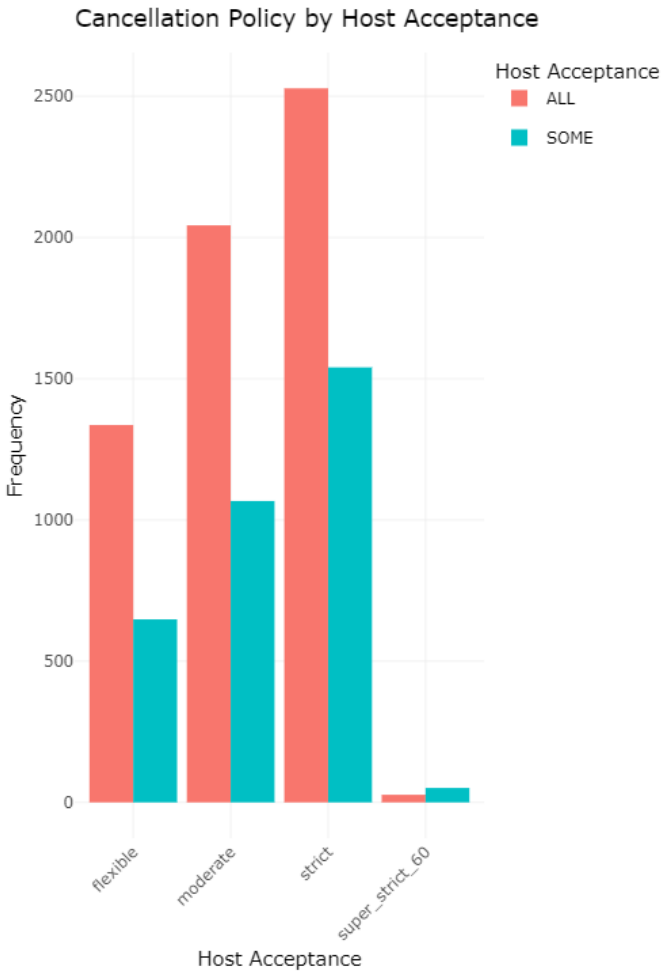

From the chart above, we determine that Airbnb listings offering proper beds to their guests have a higher median price compared to other listings offering mock beds such as futons, pull-over sofas, and couches. Also, all of these mock bed listings charge no more than $100 per day to their customers for their stay.

8. *Analyzing the Relationship between the number of beds and privacy ratio of customers on Airbnb*




From the graph above, we observe that as the number of beds increases the privacy of guests also increases. This can be explained by the increase in the number of bedrooms in the Airbnb property as the beds increase. Hence, the privacy enjoyed by the guests also increases proportionately .

*9. Analyzing Host Acceptance vs Cancellation Policy*




From the graph above, we observe that a significant proportion of Airbnb properties that have a stricter cancellation policy usually have a host acceptance rate of 100%. Hence, they have rigid cancellation rules in place for the guests if they wish to cancel their booking. Also, all other Airbnb listings with flexible or moderate cancellation policy also demonstrate a similar trend in their acceptance outcomes. However, Airbnb properties that have extremely rigid cancellation rules have a considerably lower host acceptance rate compared to others.

*10. Analyzing overpriced listings in each market on Airbnb*

| | Market | Total Overpriced Listings |
|---|---|---|
| 1 | New York | 12826 |
| 2 | Los Angeles | 6912 |
| 3 | Austin | 1667 |
| 4 | D.C. | 1658 |
| 5 | Boston | 1536 |
| 6 | Nashville | 1518 |
| 7 | Chicago | 1488 |
| 8 | New Orleans | 1436 |
| 9 | San Diego | 1428 |
| 10 | Portland | 1351 |
| 11 | San Francisco | 1342 |
| 12 | Denver | 1117 |
| 13 | Seattle | 1098 |
| 14 | East Bay, CA | 389 |
| 15 | North Carolina Mountains | 271 |
| 16 | Monterey Region | 206 |
| 17 | OTHER | 174 |
| 18 | Other (Domestic) | 139 |

*From the table we can observe that New York has the highest number of overpriced listings. This insight is significantly aligned with the metropolitan status of NYC. Also, a large proportion of the overpriced listings on Airbnb are located in major metropolitan cities such as L.A., Austin, D.C and Boston. Additionally, other domestic or suburban areas in the US have very few Airbnb listings that are overpriced.*

## Section 4: Evaluation and Modeling

*Include a short (one-paragraph) description of the "winning" model, the variables included in the model, your estimated training and generalization performance, and how you decided that it was the winning model. Also list the line numbers in your R code where you generated the final predictions that you submitted for the contest.*

"Boosting" is a powerful ensemble learning technique that combines multiple weak learners (often decision trees) to create a strong predictive model. It works by iteratively fitting models on the training data, where each subsequent model focuses on the samples that the previous models misclassified. By giving more weight to these misclassified samples, boosting effectively learns to correct the errors and improve overall prediction accuracy. The final model is an ensemble of these weak learners, and its predictions are made by aggregating the predictions from all the individual models. Boosting is known for its ability to handle complex relationships in data and often delivers excellent predictive performance.

*The variables included in this model are:*
accommodates, availability_30, availability_90, availability_365, bathrooms, bed_category, bedrooms, beds, charges_for_extra, host_total_listings_count, host_acceptance, host_response, has_min_nights, host_is_superhost, has_guests_included, host_has_profile_pic, has_security_deposit, host_response_time, has_monthly_price, has_weekly_price, has_minnight_rental, has_maxnight_rental, is_business_travel_ready, is_location_exact, instant_bookable, market, price, ppp_ind, property_category, require_guest_phone_verification, square_feet, requires_license, bed_convenience, overpriced_market, host_identity_verified, bathroom_luxury, privacy, cluster, sent_score, high_deposit, overpriced_category, verification_for_underpricing, privacy_accomodates, total_price, year, host_experience, host_is_experienced, zipcode, sent_score_neighbour, host_preferred, host_preferred_accceptance, host_preferred_location, population, room_type, property_type, extra_people, weekly_price, monthly_price, minimum_nights, maximum_nights, cancellation_policy, city_name, cleaning_fee, guests_included, require_guest_profile_picture.

Estimated Generalization Performance:
Validation auc: 0.8884116

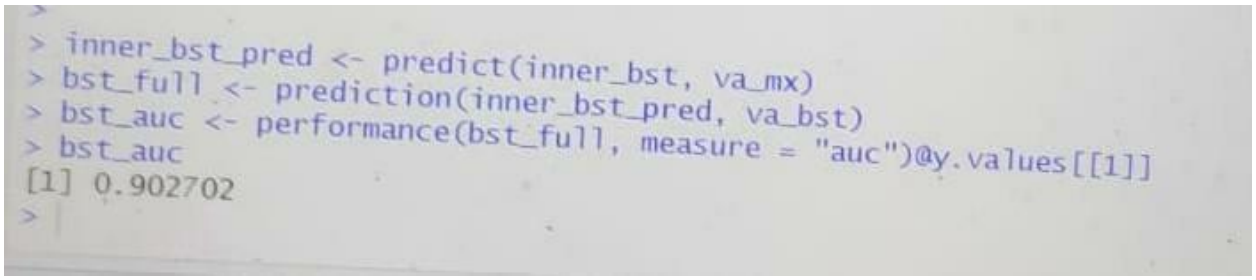
```
> #Compute the AUC for boosting model
> auc_boost <- performance(prediction_boost, measure = "auc")@y.values[[1]]
> auc_boost
[1] 0.8884116
```

The winning model, Boosting, was determined based on its superior performance in optimizing the AUC (Area Under the Curve) metric. We evaluated and compared multiple models using

various techniques such as cross-validation, fitting curves, and analyzing performance metrics. After thorough analysis and consideration of the model's performance on the given dataset, Boosting emerged as the top-performing model, making it the winning choice for our analysis.

Additionally, we also performed comprehensive analysis on the unstructured data to increase the performance of the model and its prediction accuracy. For instance, feature variables such as amenities, description, house rules, and neighborhood overview were processed, tokenized, and converted into structured data for further analysis and evaluation of predictions on the target variable. Moreover, the text processing significantly increased the AUC of the prediction models by a notable proportion.

**Note: Below contest model includes text columns, which helped to improve our AUC.**




As can be seen above in the picture, the AUC of the winning model was 90.27%. This was determined by evaluating predictions on the validation or unseen data. Additionally, the AUC derived from the predictions on the test data was approximately 89.93% thereby, concluding this model as the best performing evaluation among all other models that were implemented.

*2) For each type of model:*

- *a) The R function and/or library used:*
- *b) Estimated training and generalization performance for this model:*
- *c) How you estimated the generalization performance for this model. This should include the methodology used (i.e. simple train/validation split, cross validation, nested holdout) as well as the specific parameters of the validation setup (i.e. how much data, how many folds, etc).*
- *d) The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set).*
- *e) The line numbers in your R code where you trained the model and estimated its generalization performance.*
- *f) (Required to receive at least 80 points): Any hyperparameters that you tuned and a list of the values that you tried.*
- *g) (Required to receive at least 80 points): Any fitting curves that you created for this model.*

***## Logistic Regression:***

- The R function and/or library used:
    - R function: glm()
    - Library: R base package

- Estimated training and generalization performance for this model:
    - Validation accuracy:  0.8015003
    - Validation auc: 0.8434613

```
> # Compute the AUC of the Logistic model
> auc_logistic <- performance(perf_logistic, "auc")@y.values[[1]]
> auc_logistic
[1] 0.8434613
```

To estimate the generalization performance for this model, we ran a simple training/validation split on the data by 70:30 ratio. Additionally, we tuned the threshold parameter to adjust the trade-off between sensitivity and specificity and to optimize performance metrics such as accuracy, and recall.

- The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set):
accommodates, price, ammenities, host interactions, bedrooms, bed, ppp_ind , cancellation_policy, has_cleaning_fee, host_total_listings_count, property_category, bathrooms, charges_for_extra, host_acceptance, host_response, has_min_nights, market, description, zip code, latitude and longitude cluster.

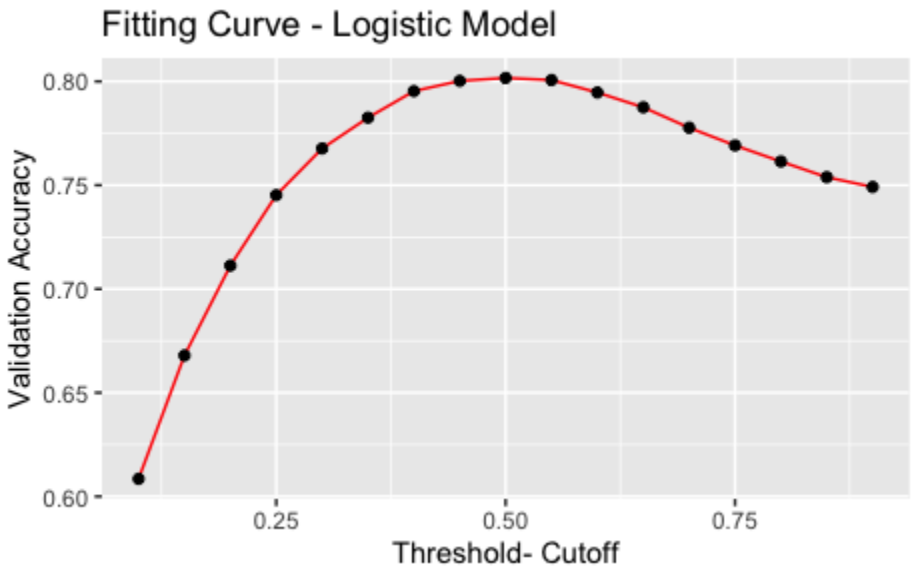
- The line numbers in the R code where we trained the Logistic regression model and estimated its generalization performance.

  Logistic regression: Lines Of Code (846 - 863)

- Any hyperparameters that we tuned and a list of the values that we tried are:

  Threshold or cutoff: Tuned the threshold between 0 and 1 to choose the best classifying cutoff with respect to accuracy, TPR, and FPR.

- Fitting Curve:




***Note: After removing External data source feature 'population' (merged using zip code), auc decreased.***

```
> # Compute the AUC of the logistic model without external dataset feature-population
> auc_logistic <- performance(perf_logistic, "auc")@y.values[[1]]
> auc_logistic
[1] 0.8428882
```

The AUC after removing external data has reduced from 0.8434613 to 0.8428882. So, the evidence suggests a decrease in the AUC for the logistic model. This indicates that the inclusion of the external data had a positive impact on the model's performance, and its removal led to a reduction in predictive accuracy.

***## Trees***

- The R function and/or library used.
    - tree

- Estimated generalization performance for this model.
  Validation accuracy: 0.7982
  Validation AUC: Full - 0.8171528
  Pruned - 0.805473

```
> ## accuracy
> # make a confusion matrix given a cutoff of 0.5
> predicted_classifications_trees <- classify(full_tree_probs, 0.5)
> predicted_classifications_trees <- factor(predicted_classifications_trees, levels = unique(va_y))
> valid_actuals_airbnb <- factor(va_y, levels = unique(va_y))
> CM2 = confusionMatrix(data = predicted_classifications_trees,
+                       reference = valid_actuals_airbnb,
+                       positive = "1")
> print(CM2)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 20250  3909
         1  2144  3692

               Accuracy : 0.7982
```

```
> auc_full
[1] 0.8171528
> auc_pruned
[1] 0.805473
```

To estimate the generalization performance for this model, we ran a simple training/validation split on the data by 70:30 ratio. Additionally, we tuned tree parameters such as the minimum number of observations in a terminal node (min size, mincut) and the maximum depth of the tree (maxdepth) to control the complexity of the decision tree model and improve its performance.

- The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set).

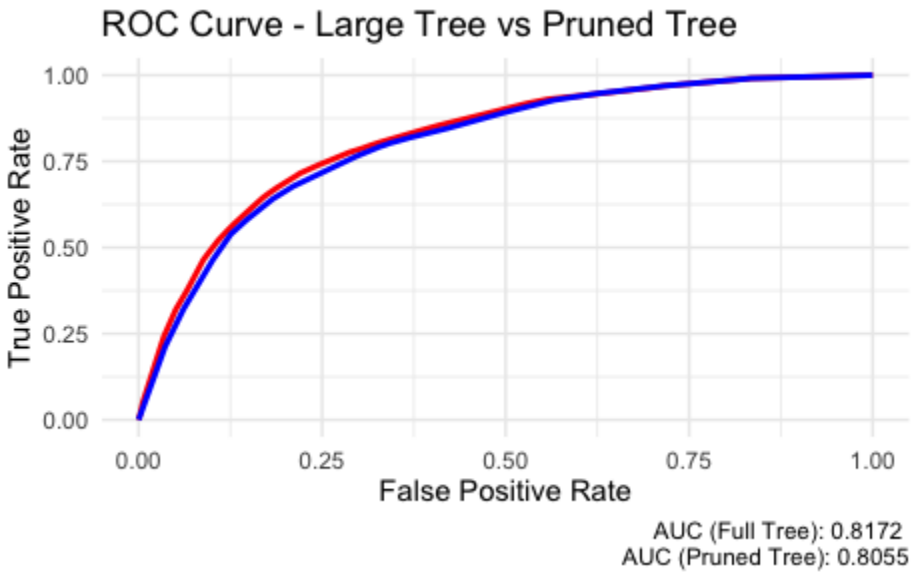
  accommodates, price, ammenities, host interactions, bedrooms, bed, ppp_ind , cancellation_policy, has_cleaning_fee, host_total_listings_count, property_category, bathrooms, charges_for_extra, host_acceptance, host_response, has_min_nights, market, description, zip code, latitude and longitude cluster.

- The line numbers in the R code where we trained the Tree model and estimated its generalization performance.

  Trees : Lines Of Code (1019 - 1045).

- Any hyperparameters that we tuned and a list of the values that we tried are:
  minsize: Set the minimum number of observations in the terminal node to 2.
  mincut: Set the minimum number of observations required to create a terminal node to 1

- ROC Curve:




***## Random Forest***

- The R function and/or library used:
    - randomForest

- Estimated generalization performance for this model:
  Validation AUC: 0.8817467

```
> preds_rf <- predict(rf.mod, newdata=va_x)
> predictions_rf <- prediction(preds_rf, va_y)
> performance(predictions_rf, measure = "auc")@y.values[[1]]
[1] 0.8817467
```

  To estimate the generalization performance for this model, we ran a simple training/validation split on the data by 70:30 ratio. Additionally, we tuned parameters such as the number of trees to grow in the random forest (ntree), and the number of randomly selected predictors at each split in the tree (mtry) to control the complexity of the model and improve its predictive accuracy.

- The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set).

  accommodates, price, ammenities, host interactions, bedrooms, bed, ppp_ind , cancellation_policy, has_cleaning_fee, host_total_listings_count, property_category, bathrooms, charges_for_extra, host_acceptance, host_response, has_min_nights, market, description, zip code, latitude and longitude cluster.

- The line numbers in the R code where we trained the Random Forest model and estimated its generalization performance.

  Random Forest: Lines Of Code (1089 - 1106).

- (Required to receive at least 80 points): Any hyperparameters that you tuned and a list of the values that you tried.

  mtry: Set the number of features randomly sampled at each split to 10
  ntree: Set the number of decision trees to be grown in the ensemble to 500

- (Required to receive at least 80 points): Any fitting curves that you created for this model.

***## Random Forest - Ranger***

- The R function and/or library used:
  - randomForest

- Estimated generalization performance for this model:
  Validation AUC: 0.8822971

```
> #Compute the AUC for ranger model
> auc_rf <- performance(prediction_ranger, measure = "auc")@y.values[[1]]
> auc_rf
[1] 0.8822971
```

  To estimate the generalization performance for this model, we ran a simple training/validation split on the data by 70:30 ratio. Additionally, we tuned parameters such as the number of trees to grow in the random forest (ntree), and the number of randomly selected predictors at each split in the tree (mtry) to control the complexity of the model and improve its predictive accuracy.

- The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set).

  accommodates, price, ammenities, host interactions, bedrooms, bed, ppp_ind , cancellation_policy, has_cleaning_fee, host_total_listings_count, property_category, bathrooms, charges_for_extra, host_acceptance, host_response, has_min_nights, market, description, zip code, latitude and longitude cluster.

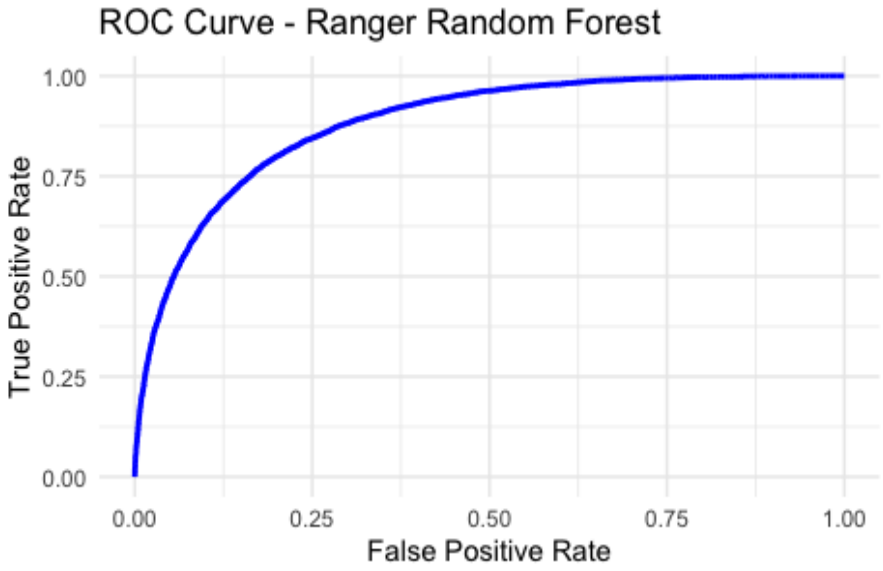
- The line numbers in the R code where we trained the Random Forest model(Ranger) and estimated its generalization performance.

Random Forest(Ranger): Lines Of Code (1265 - 1284).

- Any hyperparameters that we tuned and a list of the values that we tried are:

  mtry: Set the number of features randomly sampled at each split to 10
  ntree: Set the number of decision trees to be grown in the ensemble to 500

- *ROC Curve:*




### *## Ridge Model*

- The R function and/or library used:
  - R function - cv.glmnet()
  - Library: glmnet

- Estimated generalization performance for this model:
  Validation AUC: 0.8430068

```
> # Compute the AUC of the ridge model
> auc_ridge <- performance(prediction_ridge, "auc")@y.values[[1]]
> auc_ridge
[1] 0.8430068
```

To estimate the generalization performance for this model, we ran a simple training/validation split on the data by 70:30 ratio. Additionally, we tuned the regularization parameter i.e., lambda (λ) to optimally balance the bias and variance, and prevent overfitting.

- The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set).

  accommodates, price, ammenities, host interactions, bedrooms, bed, ppp_ind , cancellation_policy, has_cleaning_fee, host_total_listings_count, property_category, bathrooms, charges_for_extra, host_acceptance, host_response, has_min_nights, market, description, zip code, latitude and longitude cluster.

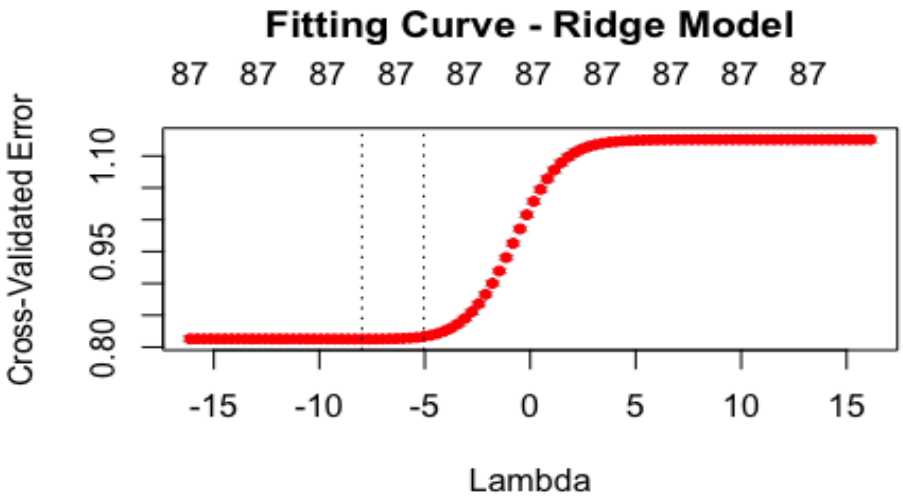
- The line numbers in the R code where we trained the Ridge model and estimated its generalization performance.

  Ridge model: Lines Of Code (1107 - 1142).

- Any hyperparameters that we tuned and a list of the values that we tried are:

  Lambda: Set the lambda to a value that gave the lowest cross-validated error i.e.,
  best lamba value: 0.000343

- Fitting Curve:



***## Lasso***

- The R function and/or library used:
    - glmnet

- Estimated generalization performance for this model:
  Validation AUC: 0.8430931

```
> # Compute the AUC of the lasso model
> auc_lasso <- performance(prediction_lasso, "auc")@y.values[[1]]
> auc_lasso
[1] 0.8430931
```

  To estimate the generalization performance for this model, we ran a simple training/validation split on the data by 70:30 ratio. Additionally, we tuned the regularization parameter i.e., lambda (λ) to optimally balance the bias and variance, and select the most relevant features.

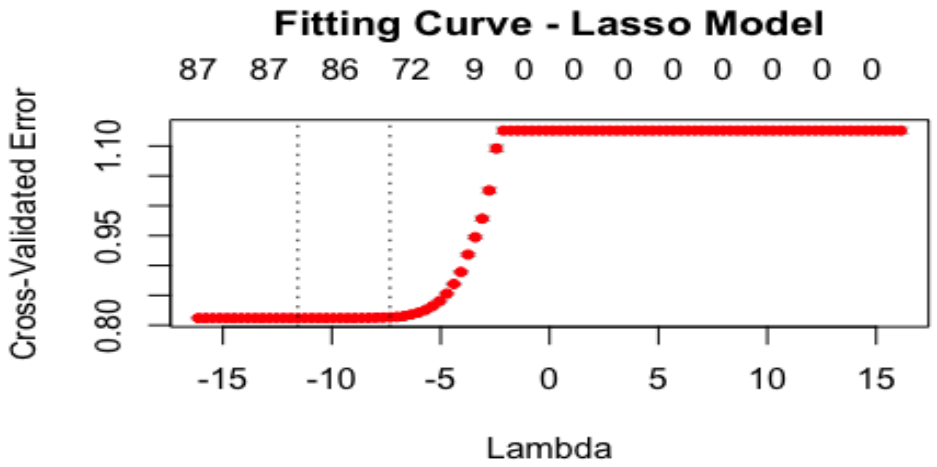
- The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set):
  accommodates, price, ammenities, host interactions, bedrooms, bed, ppp_ind , cancellation_policy, has_cleaning_fee, host_total_listings_count, property_category, bathrooms, charges_for_extra, host_acceptance, host_response, has_min_nights, market, description, zip code, latitude and longitude cluster.

- The line numbers in the R code where we trained the Lasso model and estimated its generalization performance.

  Lasso model: Lines Of Code (1145 - 1171).

- Any hyperparameters that we tuned and a list of the values that we tried are:
  Lambda: Set the lambda to a value that gave the lowest cross-validated error i.e.,
  best lamba value: 0.00000954

- Fitting curve:




***## KNN***

- The R function and/or library used:
    - KNN

- Estimated generalization performance for this model:
  Validation accuracy: 0.7023, 0.6951, 0.7254, 0.7218, 0.7344

```
> # Define the K values to test
> k_values <- c(1,2,3,4,5)
> for (k in k_values) {
+
+   # Fit the KNN model
+   knn_model <- knn(train = tr_x, test = va_x, cl = tr_y, k = k)
+
+   # Make predictions on validation data
+   knn_predictions <- as.factor(knn_model)
+
+   # Calculate accuracy
+   knn_accuracy <- calculate_accuracy(knn_predictions, va_y)
+
+   # Print accuracy
+   print(paste("KNN Accuracy:", knn_accuracy))
+
+ }
[1] "KNN Accuracy: 0.702350391731955"
[1] "KNN Accuracy: 0.695182530421737"
[1] "KNN Accuracy: 0.725420903483914"
[1] "KNN Accuracy: 0.721820303383897"
[1] "KNN Accuracy: 0.734455742623771"
>
```

To estimate the generalization performance for this model, we ran a simple training/validation split on the data by 70:30 ratio. Additionally, we tuned the number of neighbors (k) to increase the model's performance and generalization ability.

- The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set).

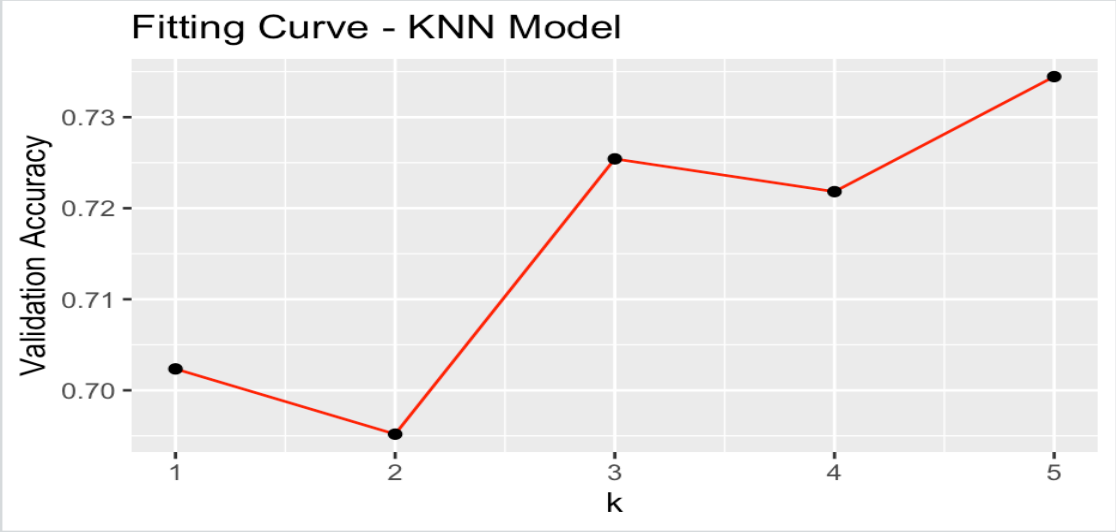
  accommodates, price, ammenities, host interactions, bedrooms, bed, ppp_ind , cancellation_policy, has_cleaning_fee, host_total_listings_count, property_category, bathrooms, charges_for_extra, host_acceptance, host_response, has_min_nights, market, description, zip code, latitude and longitude cluster.

- The line numbers in the R code where we trained the K-NN model and estimated its generalization performance.

  K-NN model: Lines Of Code (1174 - 1220).

- (Required to receive at least 80 points): Any hyperparameters that you tuned and a list of the values that you tried.
  K values: Tuned the number of nearest neighbors for values 2, 5, 10, 20, 100, 200.
  We were able to tune better for higher K values, that is we got the highest value for k=200.

- Fitting curve: we can see that validation accuracy is less for k=1 (overfit). As we increase k, generalization performance also increased.



## *Boosting*

- *T*he R function and/or library used:
    - xgboost

- Estimated generalization performance for this model:
  Validation auc: 0.8884116

```
> #Compute the AUC for boosting model
> auc_boost <- performance(prediction_boost, measure = "auc")@y.values[[1]]
> auc_boost
[1] 0.8884116
```

  To estimate the generalization performance for this model, we ran a simple training/validation split on the data by 70:30 ratio. Additionally, we tuned boosting parameters such as the number of boosting rounds or iterations (nrounds), the contribution of each weak learner to the ensemble (eta), and the maximum depth or complexity of each individual weak learner (max.depth) to control model complexity and increase the performance of the model.

- The best-performing set of features that you used in the model (you may also optionally note other features that you tried but didn't include in the final feature set).

  accommodates, price, ammenities, host interactions, bedrooms, bed, ppp_ind , cancellation_policy, has_cleaning_fee, host_total_listings_count, property_category, bathrooms, charges_for_extra, host_acceptance, host_response, has_min_nights, market, description, zip code, latitude and longitude cluster.

- The line numbers in the R code where we trained the Boosting model and estimated its generalization performance.

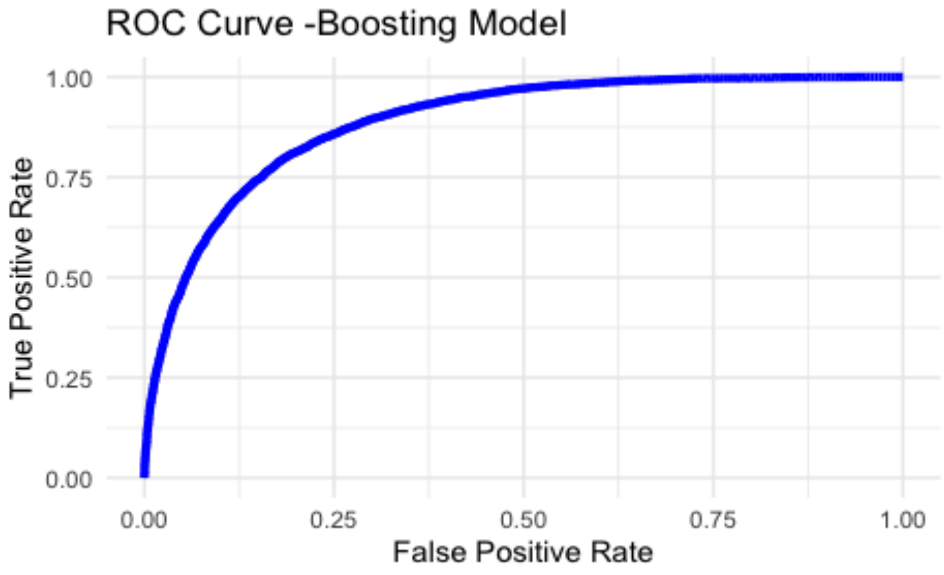
  Boosting model: Lines Of Code (1304 - 1316).

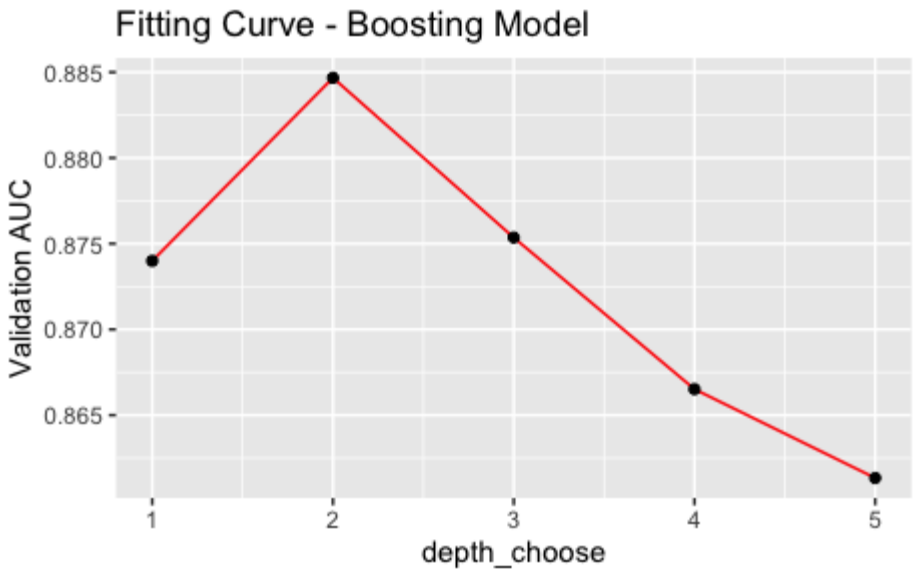
- Any hyperparameters that we tuned and a list of the values that we tried:

  nrounds: Tuned the number of boosting rounds or iterations to values (500, 1000, 1500)
  eta: Tuned the contribution of each weak learner between 0.02 - 1
  max_depths: Tuned the maximum depth of each weak learner to values (1-15)
  We got the highest AUC for :
  nrounds: 1000; max_depth: 15 ; eta: 0.04. Below is an example for a set of tuning:

```
> grid_search()
[1] "depth, nrounds, eta, auc"
[1] "1, 500, 0.8, 0.873856163099852"
[1] "1, 500, 0.9, 0.873858219301222"
[1] "1, 500, 1, 0.873996052351919"
[1] "2, 500, 0.8, 0.887236884510923"
[1] "2, 500, 0.9, 0.885947834265965"
[1] "2, 500, 1, 0.884671773338708"
[1] "3, 500, 0.8, 0.882046873706239"
[1] "3, 500, 0.9, 0.878419860851142"
[1] "3, 500, 1, 0.875357751127626"
[1] "4, 500, 0.8, 0.872996794311619"
[1] "4, 500, 0.9, 0.871190336248468"
[1] "4, 500, 1, 0.866515603624871"
[1] "5, 500, 0.8, 0.869425090335096"
[1] "5, 500, 0.9, 0.865450726324952"
[1] "5, 500, 1, 0.861333926310494"
```

- ROC and Fitting curves:

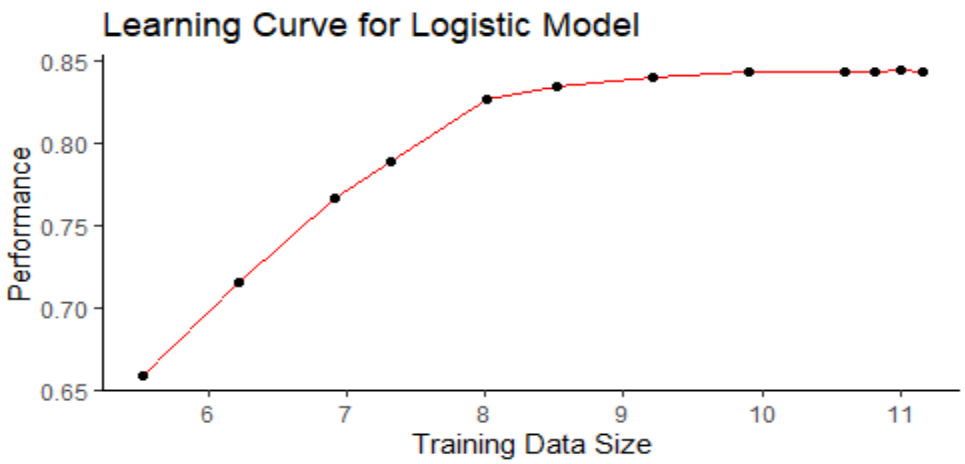




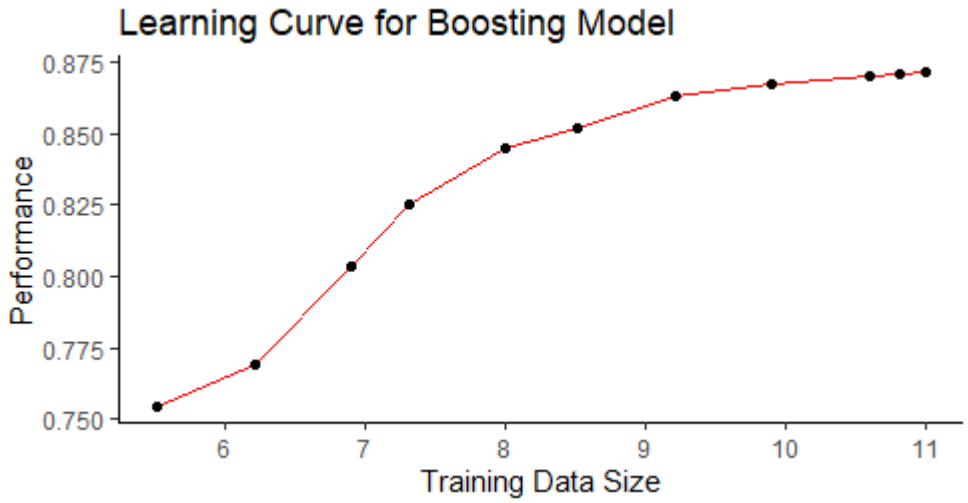

*3) (Required to receive at least 85 points) Also include any learning curves that your team created as well as any insights generated by these curves.*

***Learning curve**:* Learning curve gives an idea of how well the model is learning.

To assess the performance and behavior of the models, we utilized performance learning curves for both Logistic Regression and Boosting. These curves were computed based on the evaluation metric used for model selection, which is the AUC (Area Under the Curve). By analyzing these learning curves, we gained insights into how the models' performance evolved and how they were influenced by varying amounts of training data.




For the Logistic model, the learning curve indicates that as the training data size increases, the model learns and performs better. However, the curve levels off after a certain point, indicating diminishing returns in terms of model improvement.




The learning curve for the Boosting model demonstrates that as the amount of data fed into the model increases, its performance improves. This finding leads to the conclusion that the model benefits from a larger training dataset, indicating that increasing the size of the training data leads to better performance.

## Section 5: Reflection/takeaways

*1) What did your group do well?*

Firstly, we effectively collaborated and communicated with each other, ensuring that everyone was aware of the tasks to be performed. We established regular team status calls, which facilitated effective coordination and allowed us to address any issues promptly. We believe that our success was due to our careful selection and identification of relevant variables, and optimization of model parameters.

Our team worked diligently to incorporate various features, which significantly contributed to the increase in AUC and reliability of our predictions. We spent a lot of time on random forest models and utilized boosting techniques effectively, resulting in a well-performing model.

*2) What were the main challenges?*

The main challenges we encountered during the project included feature engineering and hyperparameter tuning of the models. We spent significant time, cleaning and preprocessing the data to ensure its suitability for model development.

Additionally, the time constraints posed a challenge in terms of model refinement and optimization. As the project and contests had a specific deadline, we had to allocate our time efficiently to ensure the completion of all project deliverables. Striking a balance between model complexity and time constraints was a continuous challenge.

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

If we could start the project over again, we would allocate more time to data exploration and preprocessing. This would have streamlined the model development process and potentially improved the overall AUC of our predictions.

Furthermore, we would have implemented an iterative approach for model development and refinement. By incorporating feedback loops and conducting regular evaluations, we could have identified areas for improvement earlier and adjusted our models accordingly.

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

Given additional time, we would focus on further model refinement and optimization. We would explore alternative machine learning algorithms and ensemble methods to improve the model's predictive performance. Additionally, we would conduct more extensive feature engineering to identify and incorporate additional relevant variables that could enhance the model's accuracy.

Moreover, we would leverage external data sources to enrich our dataset, such as incorporating external factors like locations, weather, local events, and holidays that may impact booking rates. This would provide a more comprehensive understanding of the factors influencing booking rates and further enhance the model's predictive power.

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

For a group starting this project in the future, we have a few recommendations:

a) Iterate and refine the models continually. Implement an iterative approach, where feedback and evaluations are incorporated to drive improvements in model performance.

b) Maintain clear and frequent communication within the team. Regular status calls or meetings will ensure that everyone is on the same page, progress is monitored, and any issues are addressed promptly.

c) Pay attention to the interpretability of the model. While focusing on predictive accuracy is crucial, understanding the factors driving predictions is equally important. This will provide valuable insights for actionable business strategies and facilitate better decision-making.

We hope that this advice will be useful for future groups working on this project. We are also happy to provide additional guidance or support if needed. Thank you for the opportunity to undertake this project, and we wish future groups the best of luck.