Airbnb Analysis

BUDT 758T – Eaman Jahani

By: Katie Dyachenko, Isabella Gaitan-Salanga, Yashvi Mohta, Nagasri Peri & Zarafsha Uzzaman

May 7, 2024

Section 1: Team Members & Contributions

Ekaterina (Katie) Dyachenko: Feature Cleaning, text mining and text analysis, model tuning, experimentation with models like adaboost, random forest, and SVM,

Isabella (Bella) Gaitan-Salanga: Team Leader, feature cleaning, text mining & analysis, sentiment analysis, xgboost model iterations and experimentation, main communication correspondence, business understanding.

Yashvi Mohta: Data preparation, Exploratory Data Analysis on Features (insights), Feature cleaning, text mining on selected features, feature selection through lasso regression, xgboost model building and tuning, random forest model tuning, KNN model building and tuning. Training performance outputs for all models, debugging the entire code.

Nagasri (Anusha) Peri: Data Preparation, feature cleaning, text mining, random forest building and tuning, knn building and tuning, fitting and learning curves, xgboost tuning, debugging, external dataset cleaning and prediction.

Zarafsha Uzzaman: Data Preparation with feature cleaning and text mining, logistic regression model building with cross validation, Light GBM model building, generalization performance outputs for all models.

Executive Summary

The aim of this report is to relay the purpose, processes, and reflections of analyzing Airbnb datasets to determine which features have an impact on property rating scores. After thorough data cleaning, various models were designed, tested, and altered to predict which properties would have “perfect” rating scores. These perfect scores were then compared against true values to determine the accuracy and intelligence of the developed models. At length, this report will give an extensive synopsis of the methodology used to obtain the results.

Section 2: Business Understanding

When Airbnb was first introduced, there was nearly nothing like it. With such a unique model, it gained traction and attention from the masses. Since its founding in 2008, the platform has grown exponentially, attracting individuals to become hosts as well as patrons. Airbnb's business model is dependent on the satisfaction of guests and a host's ability to effectively and consistently accrue new clients.

This study is designed for Airbnb hosts to help them understand their consumer market, either matching competitors or leveraging their own distinctive features to earn perfect ratings. On the rating scale, the closer their score to perfect, the more desirable their property becomes, increasing in value and potential. Specific to this study, the Airbnb dataset was analyzed to determine which attributes and features of a location were the most influential. Then, predictions were developed, using predictive analytics and models, labeling locations as ones that would or would not receive perfect scores.

The value proposition of this study includes:

1. **Consumer Analysis**: Understanding the consumer market gives the opportunity to learn more about consumer needs and preferences. This information can be used to improve properties or seek a new consumer segment. Directly listening and catering to consumers increases satisfaction rates, as well as referral and return rates.
2. **Competitive Analysis**: Learning about competitors gives hosts options, references to compare to, and the ability to learn and improve. If they are underperforming in certain aspects, they can mimic the behavior of their peers who perform well. Alternatively, where they see others underperforming, they can avoid certain actions and behaviors.
3. **SWOT (Market) Analysis**: Strengths, Weaknesses, Opportunities, and Threats. Markets for Airbnbs drastically vary based on where they are located. This study aids hosts in seeing these patterns, answering questions like: How does a property excel? What does a property lack?

Where can it improve, and what is this market segment missing? As well as what poses potential challenges.

4. **Marketing & Advertising Strategies**: Combining consumer and market knowledge, marketing and advertising strategies can be amended to attract the type of guests a host prefers. Moreover, improved, and intentional marketing and advertising typically increases demand and success.

Section 3: Data Understanding & Data Preparation

ID	Features Name	Description	R Code Lines
1	access	Defined clean_access function to convert text to lowercase, and select main keywords from the variable, applied the function to each row of the variable.	614 – 625, 627- 631, 636-640
2	accommodates	Kept original value in provided dataset; substituted null values with the average of the entire feature.	178, 420
3	amenities	Processed the column, to make multiple binary columns to see if an amenity listed is available(has_amenityname) and removed the original column from the train and test set	138, 408
4	availability_30	Replaced null values with the mean of the variable, and converted the variable to numeric	225, 226, 466, 467
5	Availability_365	Replaced null values with the mean of the variable, and converted the variable to numeric	211, 452
6	availability_60	Replaced null values with the mean of the variable	210, 451
7	availability_90	Kept original value in provided dataset; substituted null values with the average of the entire feature.	179, 421
8	bathrooms	Replaced null values with the median of the variable	117, 389
9	bed_type	Converted bed_type to bed_category with values such as "Real Bed", "bed", and "other"	231-233, 472-474
10	bedrooms	Replaced null values with the mean of the variable	192, 435
11	beds	Replaced null values with the mean of the variable	212, 454
12	cancellation_policy	Redefined and combined categories “strict” and “super_strict_30” into “strict”. Kept original values otherwise. Converted values into factors.	185-185, 428-429
13	city	Created the cleaning tokenizer, converted all characters in the column to lowercase, removed white space, punctuation, replaced null values with “Missing”, and converted the variable to factor	127- 133, 400-404
14	cleaning_fee	Converted the variable to numeric and replaced the null values with the mean of the variable	238-240, 478-480
15	country	Removed the variable from the train and test sets	202
16	country_code	Replaced the null values with “Unknown”, and removed the variable from the train and test sets	213, 455, 1123, 1126

17	description	Created cleaning_tokenizer function, vocabulary from tokenized itoken object, vectorizer object, and converted training documents into DTM. Used top 10 terms from DTM corpus and grouped into categorical variables.	715, 729-739 781, 794-804, 1062-1079, 1087-1104
18	experiences_offered	Converted the variable to factor, and removed the variable from the train and test set, because all the values were NONE	113, 384, 1121, 1124,
19	extra_people	Created a new factor variable "charges_for_extra" which has the value "YES" if extra_people > 0 and "NO" if extra_people is 0 or null, removed the extra_people variable from the train and test sets	245-247, 271, 484-487, 509
20	features	Created `features_process` function, split and cleaned features from `features` column, converted to binary indicators for each unique value, and merged these into the main dataset as new columns while removing the original `features` column.	989-1022, 1024-1058
21	first_review	Replaced the null values with "01/01/1999"	214, 456, 1218
22	guests_included	Kept original value in provided dataset; substituted null values with the average of the entire feature.	180, 423
23	host_about	Processed the variable using a custom tokenizer to create tokenized iterables. Constructed vocabularies and pruned them to the top 500 terms, converted texts into Document Term Matrices (DTMs) for feature extraction. Utilized regular expressions to categorize host descriptions into categories such as Travel, Home Living, and Health Wellbeing, adding these as a new categorical variable `host_mentions` based on content relevance.	846, 859, 887-896, 908-917, 1121, 1124
24	host_acceptance_rate	Created a new factor variable "host_acceptance" from host_acceptance_rate with the values "ALL" if host_acceptance_rate = 100%, "SOME" if host_acceptance_rate < 100%, and "MISSING" if it's null, deselected the host_acceptance_rate variable from the train and test sets.	254-259, 270, 493-498, 507, 592, 601,
25	host_listings_count	Converted the variable to factor in the train and test sets.	193, 436
26	host_location	Defined a `categorize_location` function to standardize and categorize `host_location` into geographical regions. Applied this function to both train and test datasets to create a new `region` variable, then removed the original `host_location` column.	590, 599, 660, 663, 666, 669
27	host_name	Excluded – Nonessential for prediction models. Only first names were provided, and most hosts had the same or similar names making data invaluable.	1121, 1124
28	host_neighbourhood	Replaced the null values with the value "Not Mentioned" and converted the variable to factor	87-88, 115-116, 142-143, 155, 386-387, 1417
29	host_response_rate	Created a new factor variable "host_response" with the values "ALL" if host_response_rate = 100%, "SOME" if host_response_rate < 100%, and "MISSING" if it's null	255, 263-265, 270, 494, 502-

		and deselected the original host_response_rate variable from the train and test sets.	504, 507, 591, 600
30	host_response_time	Converted the variable into factor, replaced the null values with the value "missing" in train and test sets	194, 436
31	host_since	Replaced the null values with the value "01/01/1999" in train and test sets	214, 455
32	host_total_listings_count	Kept original value in provided dataset; substituted null values with the average of the entire feature.	181, 423
33	host_verifications	Processed the column, to make multiple binary columns to see if the verification type is listed (has_verificationname) and removed the original column from the train and test set	137, 407, 411
34	house_rules	Processed the variable by converting to lowercase, extracting binary indicators for smoking and pets rules through keyword detection, and combined these into a single `has_rules` variable. Adjusted missing values to "No rules" and appended this new binary indicator to the dataset.	273, 513, 589, 598, 1121, 1124
35	interaction	Made categories of interaction types based on most common words in the text and named the column interaction_categor. Removedd the variable from the train and test sets	105, 1081, 1106
36	jurisdiction_names	Defined a `categorize_jurisdiction_by_state` function to standardize and categorize `jurisdiction_names` based on specific geographical keywords, converting descriptions into state categories. Applied this function to train and test to create a new `jurisdiction_state` variable, removed the original `jurisdiction_names`	595, 604, 689, 693, 696, 700
37	latitude	Excluded – Nonessential for prediction models. More insightful location details were provided in other features.	-
38	license	Created the `license_status` variable, categorizing each entry based on the presence of license data: set to "No" if missing, "Pending" if containing "pending," and "Yes" for entries with numeric values or other content. Converted `license_status` to a factor and removed the original `license` column for data streamlining.	96, 98, 120-122, 134, 391-393, 405
39	longitude	Removed the variable from the test and train sets	-
40	market	Imputed missing `market` values with the first non-missing market within each group. Assigned "Unknown" for missing entries. Converted `market` into factor	204, 446, 593, 602, 1114, 1117, 1417
41	maximum_nights	Replaced null values with the mean of the variable in train and test sets	215, 456
42	minimum_nights	Kept original value in provided dataset; substituted null values with the average of the entire feature.	184, 426
43	monthly_price	Replaced null values with the mean of the variable in train and test sets	118, 389
44	name	Replaced null values with "No Name", removed the variable from the train and test sets	295, 298, 536, 539
45	neighborhood	Merged "La Jolla Village" into "La Jolla" and converted the variable factor	195, 203, 437, 445, 1417

46	neighborhood_group	Replaced the null values with the value "Unknown" in train and test sets	216, 457
47	neighborhood_overview	Reused cleaning_tokenizer function. Created new vocabulary from tokenized itoken object, vectorizer object, and converted training documents into DTM. Used top 10 terms from DTM corpus and grouped into categorical variables.	740, 754-765, 806, 820-831, 1121, 1124
48	notes	Removed the variable from the train and test sets	1121, 1124
49	price	Converted the variable to numerical and replaced null values with the mean of the variable.	303-304, 544-545, 1208, 1210, 1243, 1328-1329
50	property_type	Categorized the variable into broader groups, converted these groups into a factor under `property_category`. Replaced null values with "Other"	196-199, 438-441, 594, 603, 1233, 1236, 1240, 1243
51	room_type	Replaced null values with "Other" in train and test sets	217, 458
52	security_deposit	Kept original value in provided dataset; substituted null values with the average of the entire feature.	183, 425
53	smart_location	Converted the variable to factor in train and test sets	114, 385, 1418
54	space	Processed the `space` description by tokenizing the text, filtering out stop words and numbers, counting word frequencies. Identified key features to create a binary indicator. Assigned "No space features" for null values, removed the space variable	310, 334, 549-550
55	square_feet	Replaced null values with the mean of the variable in train and test sets	200, 442
56	state	Converted the variable to upper case, replaced the null values with “Unknown” in train and test sets	218, 220, 459-461, 1289, 1310
57	street	Excluded – Nonessential for prediction models. More insightful location details were provided in other features.	-
58	summary	Removed the variable from the train and test sets	1121, 1124
59	transit	Used a customized tokenizer to create tokenized text, build a vocabulary, and prune to the top 500 terms. Converted these texts into DTMs. Categorized transit information into specific categories and converted `transit_category` to factor, deselected the variable from the train and test sets	934. 948-957, 960, 974-983, 1121, 1124
60	weekly_price	Replaced null values with the mean of the variable in train and test sets	201, 443, 1334-1335
61	zipcode	Replaced null values with value "Unknown" converted the variable to character type.	221-222, 462-463, 1435

To accurately clean the better of 60+ features, a multitude of strategies had to be implemented. While mainly using the tidyverse and dplyr packages to mutate and convert data, other functions had to be

utilized to categorize certain features more efficiently. Finally, it ended up having around 185 features (through dummies, binary column creation, etc.)

Text features were the most robust in this dataset, often overlapping in content and serving the same or similar functions. Before categorizing and cleaning occurred, text analysis was done on these features. Firstly, our group defined a tokenizer function that removed numbers, punctuation, and typical stop words. Documents were converted to tokens, and that information was transformed into feature vocabulary. Keeping a sample portion (500 words) of the original vocabulary, it was then converted into the document-term matrix (DTM). The top 10 most frequent words were selected from this DTM and used to create a new feature, which could then be split into dummy variables. The new features were created using the grepl function found in the base R package. In this instance, grepl was used to sieve through specific columns for certain words. If the word was located in the phrase(s), it would be sorted into that group within a new feature column.

1. Feature Insights

1.1. Amenities Column: The column originally outputted as **Figure-1**. After deliberation and careful consideration, our team decided to clean the data, ultimately resulting in **Figure-2** with improved values.

1.2. Host_Neighbourhood Variable: Replaced NA values in **Figure-3** with "Not Mentioned" and transformed variable into **Figure-4.**

1.3. License_Status: Originally had more than 2,800 levels and values **(Figure-5)**. For better predictive efficiency and readability, grouped the original values **(Figure-6)**.

1.4. Interaction Variable vs Perfect_Rating_Score: Converted robust text variable into levels and categories **(Figure-7)**. From this, we concluded the interaction with the host was extremely impactful and correlated to customer satisfaction.

1.5. Price Variable: Performed log transformation **(Figure-9)** on originally skewed variable **(Figure-8).**

1.6. CHARGES_FOR_EXTRA Variable: Created a new variable based on the behavior of the extra_people feature. Intuitively wanted to determine if there was a significant relationship. Then compared results to cancellation_policy to see the distribution **(Figure-10)**

1.7. First_review vs perfect rating score: Experimental variable testing between first_review vs perfect_rating_score. **(Figure 12).**In random forest model, variable was provided to be influential.**(Figure13)**

1.8. Price Distribution by Property Type: The box plot in **Figure-14** illustrates significant price variability across different Airbnb property types, with "Loft" and "House" showing higher median prices and a broader range of outliers, suggesting a mix of standard and luxury offerings. In contrast, types like "Apartment" and "Bed & Breakfast" exhibit more consistent and moderate pricing, indicating uniformity and potential appeal for budget-conscious travelers.

1.9. Popular Themes in Airbnb Host Mentions: Used text mining to discern popular themes in the host_mentions feature; travel and nearby locations were popular themes. **(Figure 15)**

1.10. Prices according to latitude and longitude: Heat map was plotted to see the density difference compared among cities **(Figure-16)**. Narrowed attention and focused on the Airbnb density in San Fransico / Bay Area **(Figure-17).**

Section 4: Evaluation & Modeling

1. Winning Model

The "winning" model in our competition was a Random Forest Classification model, chosen for its robustness and ability to handle the complexity of 185 carefully cleaned and engineered features. We achieved a training performance with an accuracy of 99.20%, a True Positive Rate (TPR) of 97.40%, and

a remarkably low False Positive Rate (FPR) of 0.07831%. The generalization performance on validation showed an accuracy of 76.67%, a TPR of 42.46%, and an FPR of 9.34%. Our Random Forest model underwent hyperparameter tuning through a trial-and-error process. The decision to declare this model as "winning" was based on its overall balance between accuracy, TPR, and FPR compared to other models, as well as its competitive standing (5th place) in the final evaluation, where it showed a TPR of 0.369 and an FPR of 0.07 on the testing data. Further tuning was possible to improve its generalization, but time constraints limited these efforts. The final predictions were generated in our R code at line numbers 1472, 1623, 1653, 1734, 1937, 1965, 2029, 2053, 2141, 2214, 2242, 2303, and 2355.

2. Comparative Analysis for the Models

Model	R Functions & Libraries	Training Performance	Generalization Performance	Generalization Performance Explained	Features Used in Model	R Code Lines
Lasso	Libraries - glmnet, caret	Accuracy: 73.32%	Accuracy: 72.93%	Model Uses Cross Validation with 5 folds,	Features Selected by Lasso Regression[1]	1535-1580

[1] Features Selected by Lasso Regression:

{ host_response_rate, host_acceptance_rate, host_listings_count, host_total_listings_count, latitude, accommodates, bathrooms, bedrooms, beds, square_feet, price, weekly_price, security_deposit, guests_included, minimum_nights, maximum_nights, availability_30, availability_90, availability_365, has_phone, has_facebook, has_reviews, has_kba, has_jumio, has_governmentid, has_google, has_offlinegovernmentid, has_linkedin, has_workemail, has_amex, has_manualonline, has_manualoffline, has_identitymanual, has_none, has_sentid, has_sesame, has_sesameoffline, has_tv, has_cable_tv, has_internet, has_wireless_internet, has_air_conditioning, has_kitchen, has_heating, has_familykid_friendly, has_washer, has_dryer, has_smoke_detector, has_carbon_monoxide_detector, has_first_aid_kit, has_essentials, has_shampoo, has_24hour_checkin, has_hangers, has_hair_dryer, has_iron, has_laptop_friendly_workspace, has_self_checkin, has_smartlock, has_private_entrance, has_free_parking_on_premises, has_pets_allowed, has_pets_live_on_this_property, has_cats, has_safety_card, has_lockbox, has_bathtub, has_changing_table, has_high_chair, has_crib, has_roomdarkening_shades, has_childrens_dinnerware, has_smoking_allowed, has_dogs, has_suitable_for_events, has_lock_on_bedroom_door, has_buzzerwireless_intercom, has_smart_lock, has_translation_missing_enhostingamenity49, has_translation_missing_enhostingamenity50, has_gym, has_wheelchair_accessible, has_breakfast, has_elevator_in_building, has_hot_tub, has_doorman, has_pool, has_indoor_fireplace, has_keypad, has_baby_bath, has_babysitter_recommendations, has_pack_n_playtravel_crib, has_other_pets, has_private_living_room, has_free_parking_on_street, has_refrigerator, has_oven, has_stair_gates, has_window_guards, has_fireplace_guards, has_game_console, has_NA, has_hot_water, has_private_bathroom, has_cooking_basics, has_bbq_grill, has_garden_or_backyard, has_luggage_dropoff_allowed, has_long_term_stays_allowed, has_ethernet_connection, has_table_corner_guards, has_single_level_home, has_wide_doorway, has_paid_parking_off_premises, has_washer__dryer, has_pocket_wifi, has_flat_smooth_pathway_to_front_door, has_firm_mattress, has_accessibleheight_toilet, has_accessibleheight_bed, has_host_is_superhost, has_host_has_profile_pic, has_host_identity_verified, has_is_location_exact, has_instant_bookable,

	Functions- cv.glmnet(), glmnet(), coef(), predict(), which(), rownames(), select(), table(), mean(), caret()	TPR: 22.18% FPR: 6.13%	TPR: 21.34% FPR: 6.23%	Generalization Performance conducted on 90% validation data		
Random Forest	Library - ranger Functions - ranger(), predict(), mean()	Accuracy: 99.20% TPR: 97.40% FPR: 0.07831%	Accuracy: 76.67% TPR: 42.46% FPR: 9.34%	Model Uses Simple training/validation split (30% for validation), Generalization Performance conducted on 90% Validation data	All Features	1611-1661
XGBoost	Library – xgboost Functions – as.integer(), range(), dummyVars(), predict(), xgb.DMatrix(), xgboost(), mean(), caret()	Accuracy: 86.04% TPR: 57.07% FPR: 2.31%	Accuracy: 73.96% TPR: 31.82% FPR: 9.00%	Model Uses Simple training/validation split (30% for validation), Generalization Performance conducted on 90% Validation data	Features Selected by Lasso Regression[1]	1823-1879
Light GBM	Library- lightgmb Functions- list(), lgb.train(), lgb.Dataset(), predict(), mean(), table(), caret()	Accuracy: 87.90% TPR: 65.65% FPR: 3.15%	Accuracy: 73.48% TPR: 35.05% FPR: 10.00%	Model Uses Simple training/validation split (30% for validation), Performance conducted on 90% Validation data	All Features	1929-1973
KNN	Library – class			Model Uses Simple	All Features	2029-2069

has_require_guest_phone_verification, has_require_guest_profile_picture, has_missing_value_placeholder, log_price, log_cleaning_fee }

	Functions- knn(), as.matrix(), table(), seq_along()	Accuracy: 75.54% TPR: 29.61% FPR: 6.00%	Accuracy: 71.59% TPR: 23.16% FPR: 8.83%	training/validation split (30% for validation), We manually tested values from 1:20 because the loops were lagging. After k=15 the TPR dropped, k=13 had the highest rate, so we kept that), Performance conducted on 90% Validation data		
Logistic Regression	Libraries - stats Functions - glm(), predict(), table(), factor(), mean()	Accuracy: 73.18% TPR: 23.07% FPR: 6.61%	Accuracy: 73.06% TPR: 22.66% FPR: 7.30%	Model Uses Cross Validation with 15 folds; Performance conducted on 90% Validation data	Features Selected by Lasso Regression[1]	2089-2163
Random Forest with External Dataset	Libraries: ranger() Functions: ranger(), predict(), mean()	-	Accuracy: 76.7 % TPR:40.5% FPR:9.01%	Model Uses Simple training/validation split (30% for validation), Generalization Performance conducted on 90% Validation data	All Features + All External Dataset Features[2]	1711-1753

3. **External Dataset:** Merged a dataset of energy rates zipcode-wise to see its impact on where a listing gets a perfect_rating_score. For this, we ran a random forest model using the ranger() using all the features from Airbnb and the new dataset with similar tuning parameters to our

[2] External Dataset Features:
{ zip, eiaid, utility_name, state, service_type, ownership, comm_rate, ind_rate, res_rate }

winning model. **The TPR = 0.405647985989492, FPR= 0.0911489361702128 and Accuracy = 0.7679863 proving to be better than our lasso model.** This could be due to the following reason: The "res_rate" column, reflecting residential utility rates, is key for predicting Airbnb ratings. Lower rates suggest affordability, enhancing guest satisfaction and potentially leading to higher ratings. They also hint at quality amenities and a host's commitment to sustainability and responsiveness—factors that resonate with guests' preferences for comfort, eco-friendliness, and attentive service. In essence, "res_rate" is a significant indicator of a listing's appeal and the host's attentiveness, both influential in securing perfect guest ratings.

4. **Hyperparameters for Tuning**: We hyper-tuned our Random Forest, XGBoost, and LightGBM models extensively due to their high complexity. This process involved rigorous trial and error, during which we determined which parameter adjustments led to better True Positive Rates (TPR) and lower False Positive Rates (FPR), and continuously made those alterations, to find the best values. For example, in Random Forest, we adjusted the class weights based on the class imbalance of 'NO' and 'YES' in the target variable. For XGBoost, we found that reducing the eta helped control the model's overfitting. The final tuning parameters for each model were:

Model Name	Final HyperTuning Paramters	Reasoning/Alteration of Values
Random Forest Classification	num.trees = 600, mtry = sqrt(185), importance = 'impurity', probability = TRUE, class.weights = c(1.5, 3.5) threshold = 0.46	**num.trees = 600**: Increasing the number of trees enhances model accuracy, we tried values 300,400,500,600,1000,1500 **mtry = sqrt(185)**: Adjusting the number of features to consider at each split to optimize the balance between model bias and variance. This is the number of features **probability = TRUE**: Enables the model to output prediction probabilities, useful for making threshold-based decisions. **class.weights = c(1.5, 3.5)**: We used weights for classes to address imbalances and improve

		minority class predictions. As NO was 72% and YES was 28%. We calculated weights by inversing the percents and then kept on increasing/decreasing them by 0.3 increments to see changes. **threshold = 0.46**: We kept changing this from 0.5, to 0.48,0.52, to balance it correctly with the TPR and keep the FPR below 0.10
XGBoost Classification	objective = "binary:logistic", eval_metric = "auc", max_depth = 47, eta = 0.18, nthread = 3, gamma = 5.9, scale_pos_weight = 0.72, min_child_weight = 3, nrounds = 50	**objective = "binary:logistic":** We used this as we wanted the model to predict probabilities for binary classification. **eval_metric = "auc"**: Since, we wanted to distinguish betweeen"YES", "NO" classes. **max_depth=47**: We figured that this helps to see how specialized a tree gets, we used values 0.5(lead to underfitting), 3 to 9, and figured that around 5.7-6 was the correct spot as increasing it too much (around 9) started overfitting it, so we chose 5.9. **eta = 0.18**: We worked on different values starting from 0.08 till 3 and found that reducing the eta helped control the model's overfitting. 0.18 was the best spot. **nthread = 3**: this helps in limiting the parallel thread used for resources, we kept this low (2,3,5) as we didn't have alot of computational resources **gamma = 5.9:** we tried values from 2 to 10, with increments of 0.3 saw increasing it too much

		makes it underfit, while decreasing too much made it overfit. **scale_pos_weight = 0.72**: We used weight for classes to address imbalances and improve minority class predictions. As NO was 72% and YES was 28%. **min_child_Weight= 3:** we tried 1, to 10, and saw that at higher values it was again overfitting. Way too low values led to overgeneralization. At3 the balance between TPR and FPR was the best. **nrounds = 50:** We did values 50,100,200 and noticed around 200 the model computation time was way too long and was not giving the best TPR and FPR values we needed according to the rubrics. 50 was the best in terms of computational time and balance between metrics**.**
LightGBM Classification	objective = "binary", metric = "binary_error", learning_rate = 0.28, num_leaves = 55, max_depth = 34, min_data_in_leaf =35, bagging_fraction = 0.8, bagging_freq=2, feature_fraction =0.8, verbosity = 1	**Learning rate :** controls step size and controls the speed at which learning occurs. If the speed is too fast, this may result in “skipping over” the optimal solution. For this reason, we started at a learning rate of 0.1 and increased to 0.9 in steps of 0.05. Found that 0.2 to 0.3 showed increased TPR. Then we explored this range and found the best number to be 0.28. **num_leaves:** Increasing num_leaves increases complexity and may increase TPR, however a number too large could cause overfitting. Therefore, we began with 20 and increased in steps of 5 until

		reaching 70, after which we saw signs of overfitting. Within the range found that 55 was best for TPR. **max-depth**:Maximum tree depth for base and allows for more complexity. Larger numbers mean better fitting the data, however, numbers that are too large may result in overfitting. Similar approach to Num leaves. We began with 20 and increased to 70 in steps of 2. We found that range between 30 and 40 was best. Within this range, we experimented in steps of 1 and found 34 to be a good value for this parameter. **Min_data_in_leaf:** controls the minimum amount of data in one leaf. Our approach was similar to the approach of max-depth. We experimented and found the best value to be 35. **Bagging-fraction-** controls the fraction of data for each iteration. Larger values for this lead to more diverse exposure of the data, but like other parameters, values that are too large will lead to overfitting. For this reason, tried values between 0.1 and 0.9 and found. 0.8 to be best for increasing TPR. **Bagging_frequency**- controls the frequency of bagging. Increasing this value allows for the ensemble to be trained on a larger fraction of features. Large values allow for better fitting, however, values that are too large may lead to overfitting. For this reason, we tried values between 1 and 15 and found 2 to be the best value for TPR.

		Feature-fraction- controls the fraction of features for each iteration. Larger values for this lead to more diverse exposure of the data, but like other parameters, values that are too large will lead to overfitting. For this reason, tried values between 0.5 and 0.9 and found. 0.8 to be best for increasing TPR.

5. Fitting Curve Analysis

The ROC curve **(Figure 18)** analysis reveals a model with an initial true positive rate (TPR) surge from 0.350 to a peak of 0.425, against a false positive rate (FPR) increase from 0.080 to 0.095. This sharp rise in TPR at the lower FPR levels indicates effective class separation early on. However, the curve's later fluctuations suggest that the model's ability to discriminate between classes diminishes as the FPR slightly increases. These numbers are critical for pinpointing the threshold where the model maintains high sensitivity without sacrificing specificity, guiding the fine-tuning process for optimal model performance.

6. Learning Curve Analysis

The learning curve **(Figure 19)** for the Random Forest Model reveals that initial training with a smaller dataset yields a rapid increase in accuracy. However, as the training set size exceeds 60%, the model's accuracy improvement rate diminishes, eventually plateauing. This suggests that beyond a certain point, additional data contributes minimally to performance enhancement. The model's highest accuracy level stabilizes around 0.766, indicating the optimal amount of training data needed for effective learning and generalization capabilities.

Section 5: Reflection/takeaways

1. What did your group do well?

Our group demonstrated various strengths across various aspects of the project. One of the major strengths of our group was excellent communication and innovation. The constructive collaboration within our team was a significant driving force behind our creative solutions and the robustness of our predictive system. We kept on debugging our code and very well understood the concepts required. Besides that, our team excelled in the deep understanding and meticulous preparation of the data and selection and optimization of our predictive models. We were able to thoroughly analyze the Airbnb datasets and explore multiple models, rigorously testing and comparing their performances to identify the best fit for our goals.

2. **What were the main challenges?**

In this project, we encountered some challenges that tested our problem-solving skills and analytical abilities. Attempting to choose the most accurate model and tuning its parameters was particularly challenging given the contest's requirements for False Positive Rate. Experimenting with various algorithms and their parameters to find the optimal balance between TPR and FPR was time-consuming and required a deep understanding of each model's characteristics. Another aspect of the project we found challenging was constructing models in a way that ensures that the models have similar TPR and FRP on both the training and the validation datasets. Finding the right balance between overfitting and underfitting demanded careful attention and rigorous checks.

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

When reflecting on our performance, there are some things that we would do differently if we could start the project over again. Initially, we focused quite narrowly on a few models. With a fresh start, we would explore a wider array of models early in the project. We would also make sure to figure out if our factor levels and features for train and test data were same from the very beginning. This approach would include experimenting with ensemble methods and more complex models that would potentially improve our predictive accuracy.

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

If we were given another few months to work on and refine our project, we would take advantage of integrating external data sources. Incorporating economic indicators, and seasonal trends could improve our understanding of fluctuations in Airbnb ratings and bookings. Besides that, producing visualizations for the report and developing an interactive user interface would enhance the interpretability and usability of our findings.

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

Some of the pieces of advice that we think would benefit the groups starting this project next year are thorough planning of the project timeline, extensive experimentation, and having the ability to work under pressure. Having a plan for the timeline of the project and a schedule that a group can stick to would be crucial for this project, as by breaking down the project into manageable stages, a team can ensure effective time management, which is crucial for efficient progress. Experimentation with various models and techniques, including classical machine learning models and more advanced algorithms, is a particularly important aspect of the project. By conducting research to find various models that could be implemented in the project, future teams will expand their horizons when it comes to model selection, which might result in an advantage compared to other contestants. Lastly, future participants should be prepared to adapt their strategies based on feedback they will receive, which can sometimes be unexpected. The teams must be ready to embrace challenges as they come, think quickly, and adapt effectively.

Figure Appendix

Figure-1

amenities
TV,Cable TV,Internet,Wireless Internet,Air conditioning,Kitch...
TV,Wireless Internet,Air conditioning,Kitchen,Free parking o...
TV,Wireless Internet,Air conditioning,Kitchen,Free parking o...
TV,Cable TV,Wireless Internet,Air conditioning,Kitchen,Free ...
TV,Cable TV,Internet,Wireless Internet,Air Conditioning,Kitch...
TV,Internet,Wireless Internet,Air Conditioning,Kitchen,Free P...
TV,Cable TV,Internet,Wireless Internet,Air Conditioning,Kitch...
TV,Internet,Wireless Internet,Air Conditioning,Kitchen,Free P...
TV,Wireless Internet,Kitchen,Heating,Family/kid friendly,Smo...
TV,Internet,Wireless Internet,Kitchen,Heating,Smoke detecto...
TV,Internet,Wireless Internet,Air conditioning,Kitchen,Pets al...
TV,Cable TV,Internet,Wireless Internet,Air conditioning,Kitch...
TV,Internet,Wireless Internet,Air conditioning,Kitchen,Gym,P...

Figure-2

```
[28] "has_kitchen"
[29] "has_heating"
[30] "has_familykid friendly"
[31] "has_washer"
[32] "has_dryer"
[33] "has_smoke detector"
[34] "has_carbon monoxide detector"
[35] "has_first aid kit"
[36] "has_fire extinguisher"
[37] "has_essentials"
[38] "has_shampoo"
[39] "has_24hour checkin"
[40] "has_hangers"
[41] "has_hair dryer"
[42] "has_iron"
[43] "has_laptop friendly workspace"
[44] "has_self checkin"
[45] "has_smartlock"
[46] "has_private entrance"
[47] "has_free parking on premises"
[48] "has_pets allowed"
[49] "has_pets live on this property"
[50] "has_cats"
[51] "has_safety card"
```

Figure-3

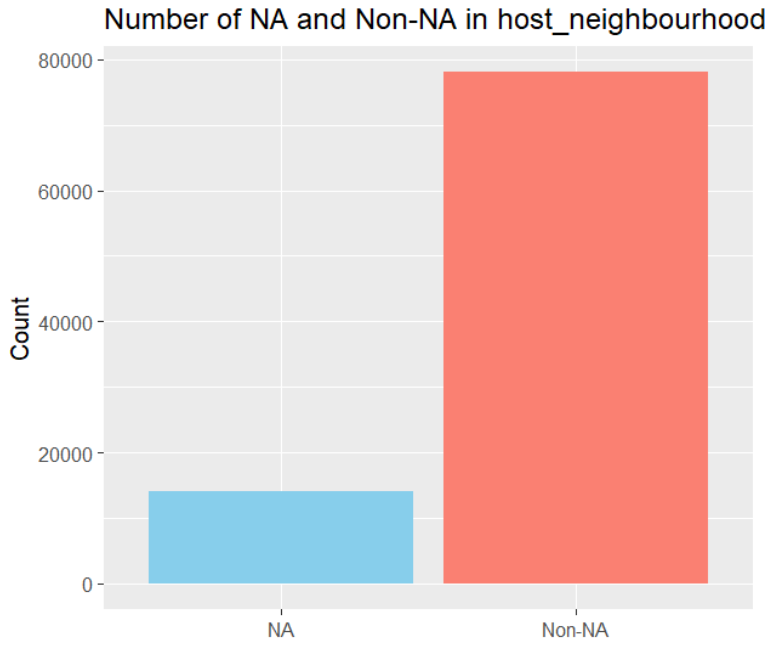

Figure-4

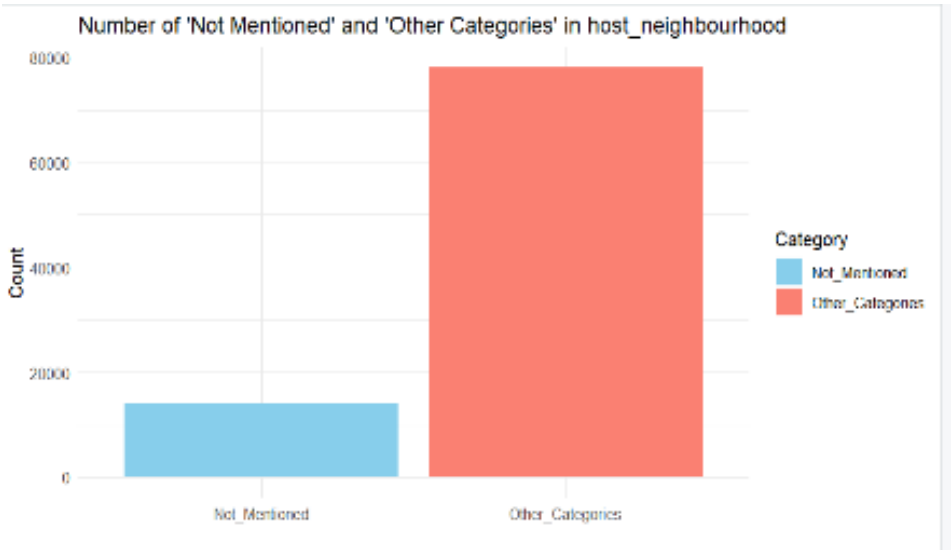

Figure-5

	License	Count
1	-16-291789-HO	1
2	"City registration pending"	3
3	"City Registration Pending"	1
4	"San Francisco Short-term Residential Registration number ...	1
5	#0306645-03-001	1
6	#1013914 (Business Registration Certificate for San Francisco)	1
7	#128220795	1
8	#14-231-355. The first legal Airbnb in Eastmoreland. Safety i...	1
9	#17STR-05552	1
10	#17STR-05913	1
11	#17STR-07575	1

Figure-6

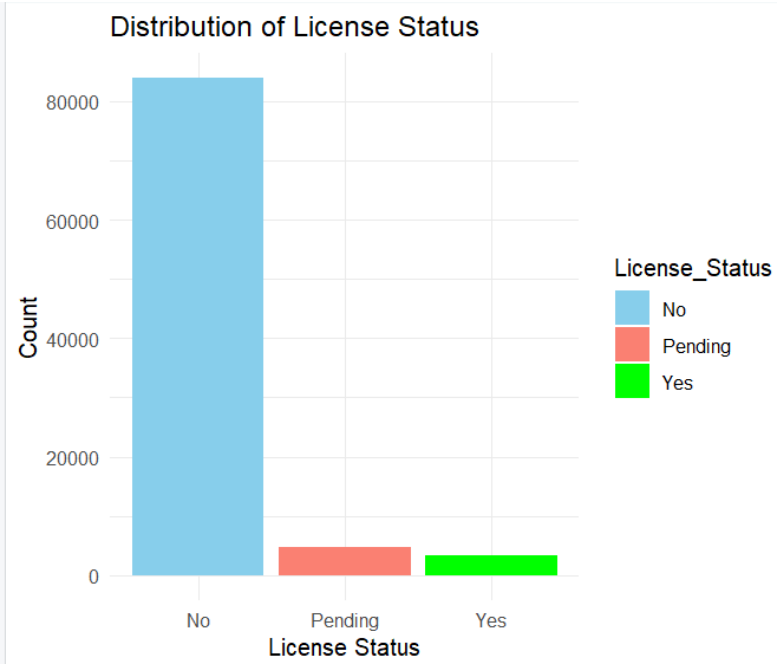

Figure-7

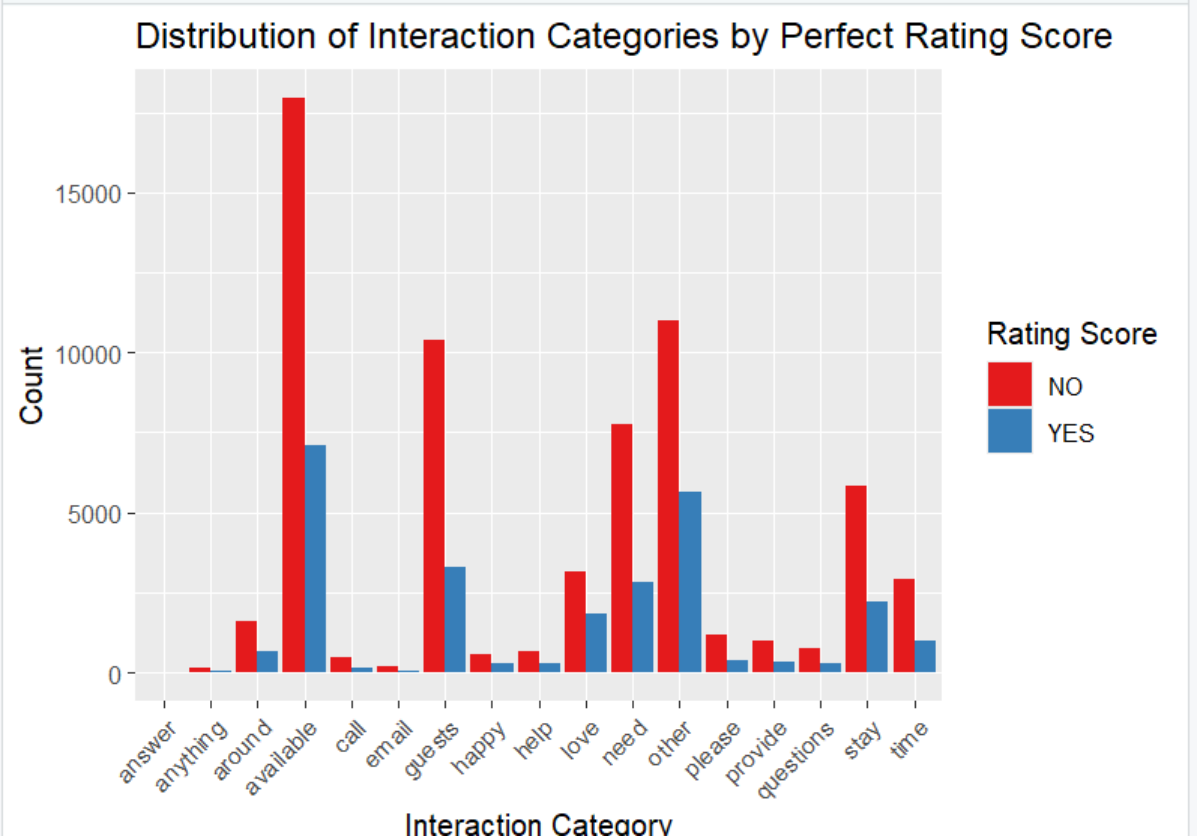

Figure-8

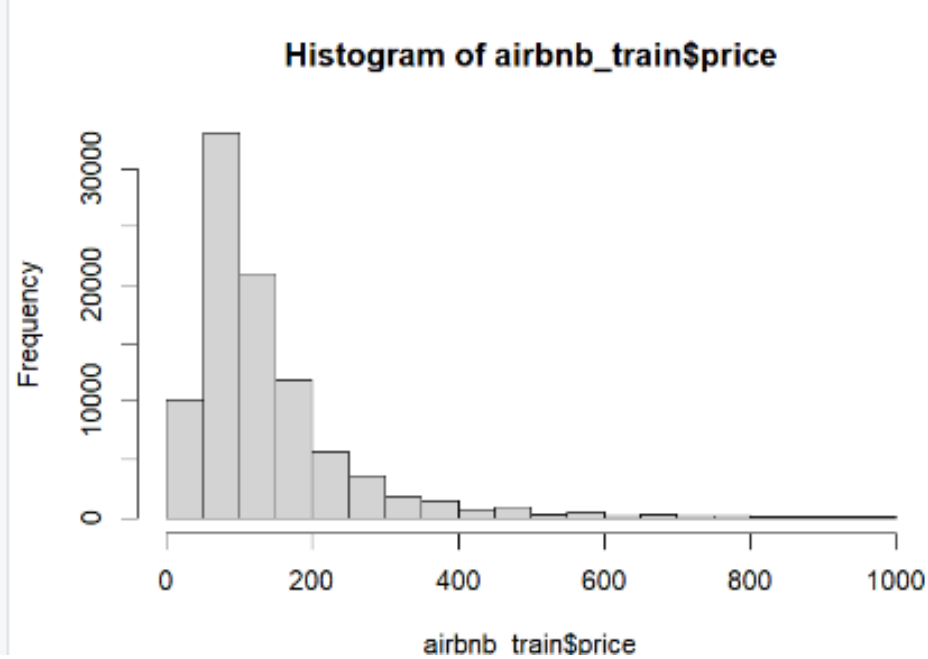

Figure-9

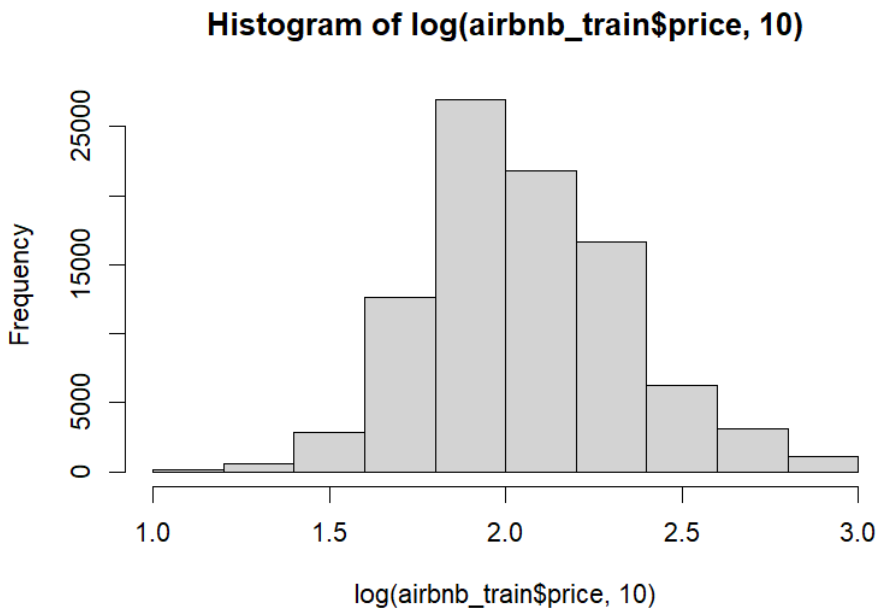

Figure-10

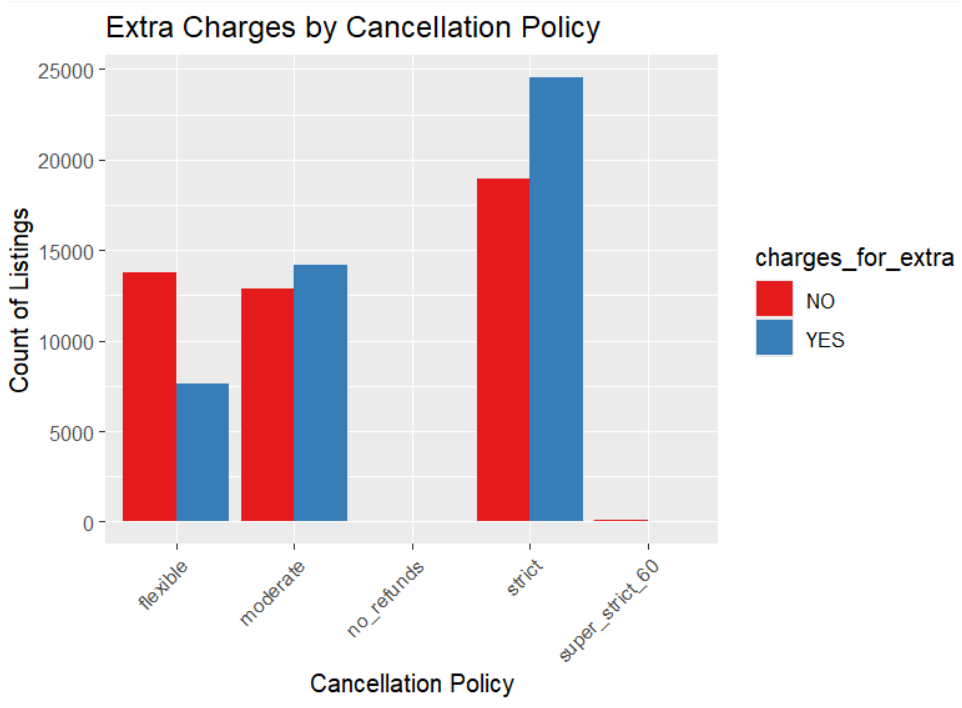

Figure-11

```
Levels: NO YES
> table(airbnb_train$charges_for_extra)

   NO   YES
45665 46402
```

Figure-12

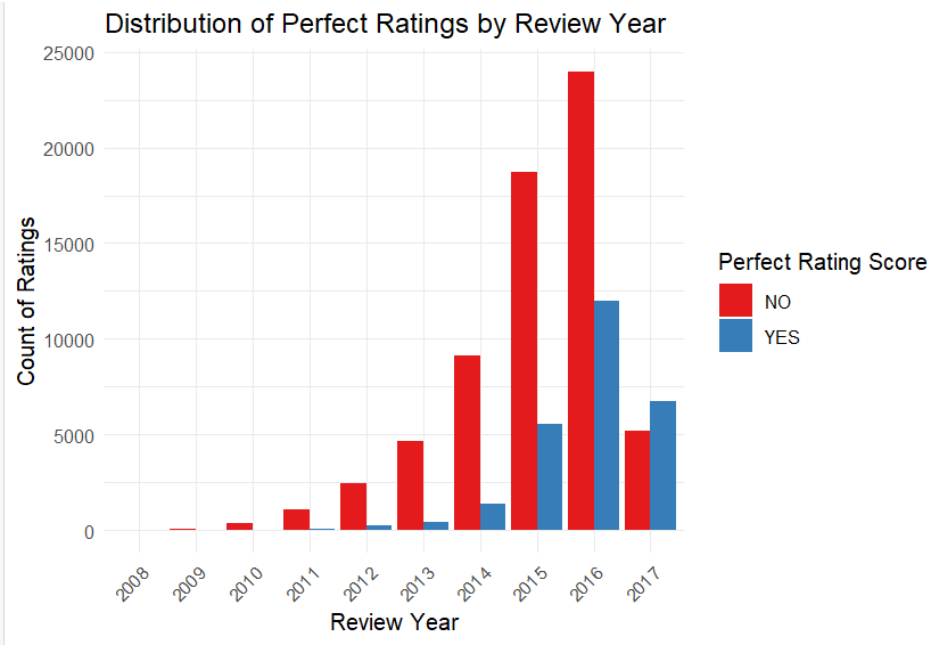

Figure-13

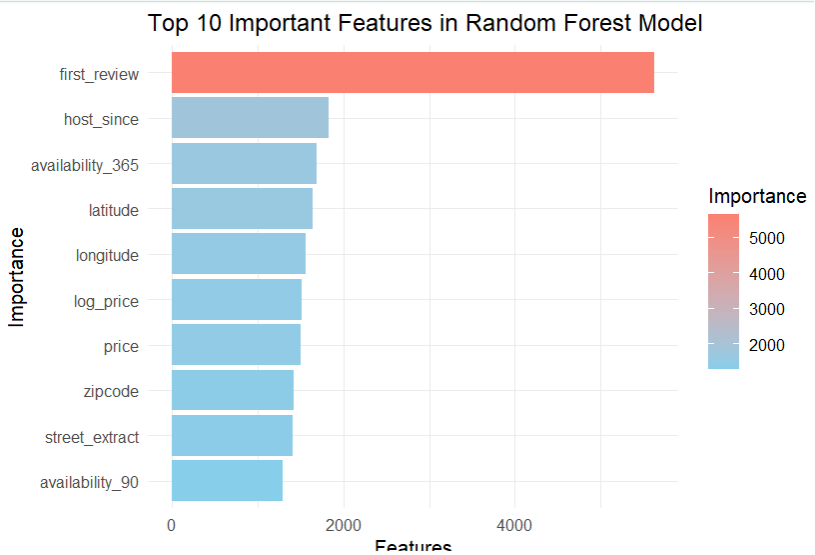

Figure-14

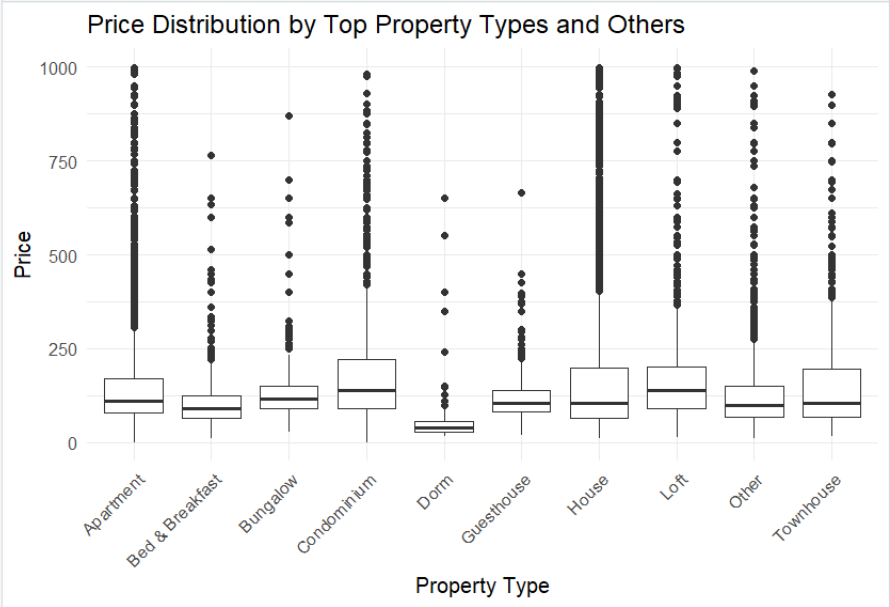

Figure-15

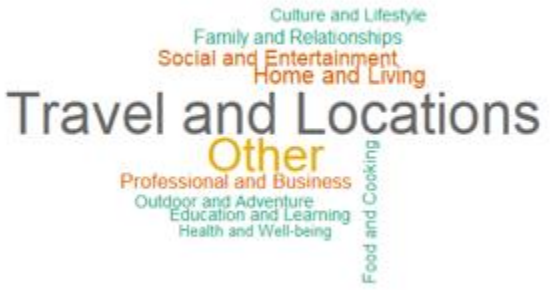

Figure-16

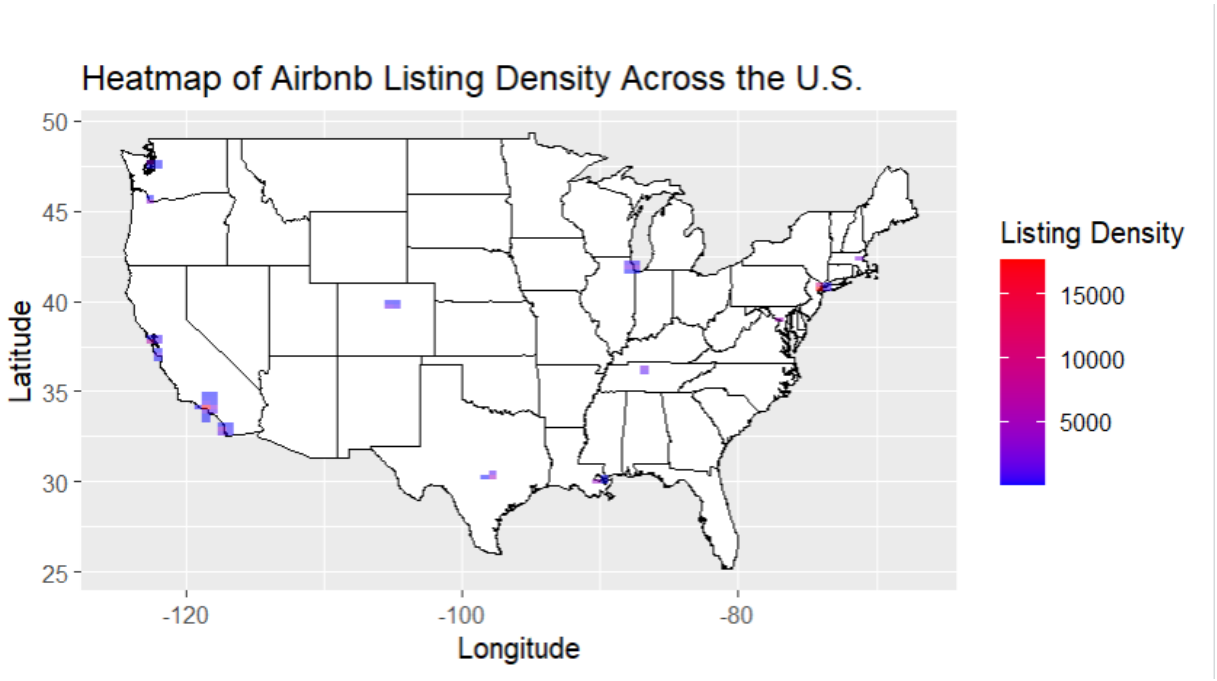

Figure-17

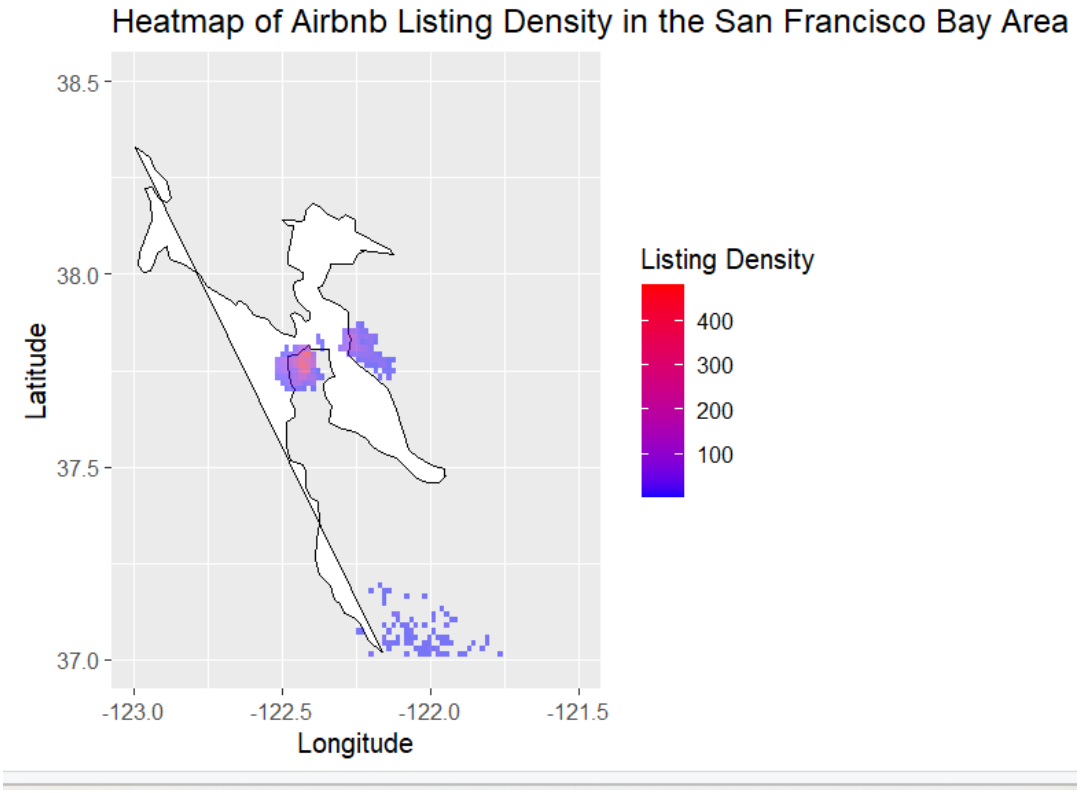

Figure 18

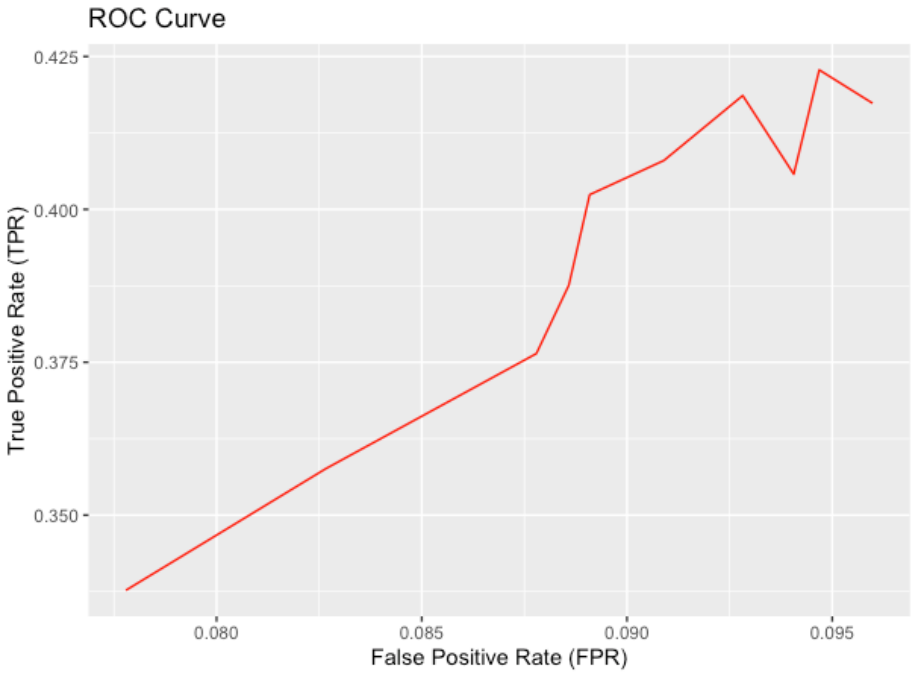

Figure 19

References

National Renewable Energy Laboratory (NREL), & Huggins, J., U.S. Electric Utility Companies and Rates: Look-up by Zipcode (2020) (2021). National Renewable Energy Laboratory (NREL). Retrieved May 12, 2024, from https://catalog.data.gov/dataset/u-s-electric-utility-companies-and-rates-look-up-by-zipcode-2020.