UNIVERSITY OF
MARYLAND

ROBERT H. SMITH
SCHOOL OF BUSINESS

BUDT 758T Final Project Report
Group 19
Spring 2025

Table of Contents

Section 1: Team member names and contributions

Devika Raheja: Coding and Report Writing
Linda Tom: Coding and Report Writing
Manasvi Surasani: Coding and Report Writing
Prakruthi Kaganti Shivakumar: Coding and Report Writing
Rasika Pande: Coding and Report Writing
Shriya Goyal: Coding and Report Writing

Section 2: Business Understanding

Airbnb is a global home-sharing platform where property owners can rent out homes, apartments, or rooms to short-term renters. Hosts earn income from their listings, while guests provide public feedback through reviews and star ratings. The platform thrives on guest satisfaction, trust, and reputation. A key driver of customer bookings, pricing power, and long-term success for Airbnb hosts is their listing's rating score. A perfect rating, defined as a 100% five-star review from guests signals high-quality service, boosts visibility in search results, and allows hosts to command premium pricing.

The goal of this data mining project is to build a predictive model that estimates whether a listing will receive a perfect rating. We developed a binary classification model using a wide range of features derived from structured listing metadata, host attributes, review history, and external crime statistics. Our external dataset was pulled from Kaggle, which contained the number of crime incidents in each state. We also engineered text-based features from guest reviews using topic modeling, allowing the model to capture subjective experience factors like cleanliness, responsiveness, and location quality. Our model can have extensive uses for various use cases and generate business value for –

1. Hosts

Independent hosts and property owners stand to gain the most benefit from our model. By identifying the key predictors of perfect ratings—such as missing host response time, months to first review response speed, instant booking, and missing host acceptance—our model can be used as a personalized diagnostic tool. Hosts can receive targeted recommendations on how to improve their listing (e.g. improving responsiveness or allowing instant bookings) which might allow them to increase the likelihood of receiving perfect scores. In doing so, Hosts would be able to:

- Boost their visibility through Airbnb's algorithm
- Justify higher nightly rates
- Improve booking conversion
- Build long-term brand trust and guest loyalty

2. Property Managers

Property managers who oversee multiple Airbnb units can integrate the model into their performance dashboards. By scoring listings based on their likelihood of receiving perfect reviews, they can:

- Allocate resources efficiently across underperforming units

- Prioritize upgrades to properties with high revenue potential but suboptimal guest feedback
- Benchmark properties across cities or regions
- Train new staff or cleaners based on data-driven feedback loops

3. Airbnb Corporate: Platform Trust and Quality Control
Airbnb can use predictions from our model to proactively identify listings at risk of poor performance and intervene before negative guest experiences occur or use it to predict upcoming perfect rated hosts. This could involve:

- Suggesting listing improvements during the onboarding process
- Automating nudges or messages to struggling hosts
- Personalizing search ranking algorithms to emphasize likely-to-be-perfect listings

Our binary classification model draws from over 60 variables, including review sentiment, host tenure, booking frequency, property type, and guest-reported amenities. We trained multiple algorithms and selected XG Boost as the best-performing model based on validation metrics, specifically prioritizing sensitivity (TPR) and the false positive rate in identifying listings with genuine 5-star potential.

By accurately forecasting whether a listing is likely to achieve a perfect score, our model supports strategic decision-making for various Airbnb stakeholders. Hosts can receive actionable insights to improve their listings, property managers can optimize resource allocation across portfolios, and Airbnb can use these predictions to support quality control and trust on the platform. Through a combination of advanced feature engineering, model evaluation, and external data integration, our approach delivers both predictive accuracy and business value across the Airbnb ecosystem.

Section 3: Data Understanding and Data Preparation

In the table below, Python code blocks correspond to those displayed when previewing the notebook in Jupyter. Each line number indicates where the feature was first defined or, in the case of original features, where it was initially accessed from the dataset.

ID	Feature Name	Brief Description	Python Code Blocks
1	host_acceptance_MISSING	Missing bucket or partial acceptance level flag	7
2	host_acceptance_SOME	Missing bucket or partial acceptance level flag	7
3	has_security_deposit_YES	Binary value based on 'has_security_deposit_YES' condition	7
4	has_cleaning_fee_YES	Binary value based on 'has_cleaning_fee_YES' condition	7
5	charges_for_extra_YES	Binary value based on 'charges_for_extra_YES' condition	7
6	has_host_verification_YES	Binary value based on 'has_host_verification_YES' condition	7
7	has_min_nights_YES	Binary value based on 'has_min_nights_YES' condition	7
8	host_response_rate	Original feature from dataset	3
9	host_listings_count	Original feature from dataset	3
10	host_total_listings_count	Original feature from dataset	3
11	accommodates	Original feature from dataset	3
12	bathrooms	Original feature from dataset	3
13	bedrooms	Original feature from dataset	3
14	beds	Original feature from dataset	3
15	price	Original feature from dataset	3
16	security_deposit	Original feature from dataset	3
17	cleaning_fee	Original feature from dataset	3
18	guests_included	Original feature from dataset	3
19	extra_people	Original feature from dataset	3
20	minimum_nights	Original feature from dataset	3
21	maximum_nights	Original feature from dataset	3
22	availability_30	Original feature from dataset	3
23	availability_60	Original feature from dataset	3
24	availability_90	Original feature from dataset	3
25	availability_365	Original feature from dataset	3
26	host_months_since	Months since the host created their account	7
27	first_review_months	Months since the listing received its first review	7
28	num_verifications	Count of host verification methods	8
29	num_features	Count of features present across categories	8
30	require_guest_phone_verification	Flag for if host requires guest phone verification	8
31	host_has_profile_pic	Flag for if host uploaded a profile picture	8
32	host_is_superhost	Flag for if host is marked as a Superhost	8
33	require_guest_profile_picture	Flag for if guest must have a profile picture	8

34	requires_license	Flag for if listing requires a license	8
35	is_location_exact	Flag for if listing location is exact	8
36	instant_bookable	Flag for if listing can be instantly booked	8
37	host_identity_verified	Flag for if host's identity is verified	8
38	num_amenities	Total number of amenities in listing	8
39	free_parking	Amenity group: free parking related keywords	8
40	pets_allowed	Amenity group: pet-friendly accommodation	8
41	checkin_24hr	Amenity group: allows 24-hour check-in	8
42	amenity_buzzer/wireless_intercom	Flag for if amenity 'buzzer/wireless intercom' is listed	8
43	amenity_self_check_in	Flag for if amenity 'self check in' is listed	8
44	amenity_fire_extinguisher	Flag for if amenity 'fire extinguisher' is listed	8
45	amenity_translation_missing:_en.hosting_amenity_50	Flag for if amenity 'translation missing: en.hosting 50' is listed	8
46	amenity_washer	Flag for if amenity 'washer' is listed	8
47	amenity_laptop_friendly_workspace	Flag for if amenity 'laptop friendly workspace' is listed	8
48	amenity_first_aid_kit	Flag for if amenity 'first aid kit' is listed	8
49	amenity_indoor_fireplace	Flag for if amenity 'indoor fireplace' is listed	8
50	amenity_heating	Flag for if amenity 'heating' is listed	8
51	amenity_wireless_internet	Flag for if amenity 'wireless internet' is listed	8
52	amenity_internet	Flag for if amenity 'internet' is listed	8
53	amenity_cat(s)	Flag for if amenity 'cat(s)' is listed	8
54	amenity_pets_live_on_this_property	Flag for if amenity 'pets live on this property' is listed	8
55	amenity_breakfast	Flag for if amenity 'breakfast' is listed	8
56	amenity_hair_dryer	Flag for if amenity 'hair dryer' is listed	8
57	amenity_dryer	Flag for if amenity 'dryer' is listed	8
58	amenity_private_entrance	Flag for if amenity 'private entrance' is listed	8
59	amenity_air_conditioning	Flag for if amenity 'air conditioning' is listed	8
60	amenity_free_parking_on_premises	Flag for if amenity 'free parking on premises' is listed	8
61	amenity_kitchen	Flag for if amenity 'kitchen' is listed	8
62	amenity_carbon_monoxide_detector	Flag for if amenity 'carbon monoxide detector' is listed	8
63	amenity_hot_tub	Flag for if amenity 'hot tub' is listed	8
64	amenity_shampoo	Flag for if amenity 'shampoo' is listed	8
65	amenity_smoke_detector	Flag for if amenity 'smoke detector' is listed	8
66	amenity_wheelchair_accessible	Flag for if amenity 'wheelchair accessible' is listed	8
67	amenity_iron	Flag for if amenity 'iron' is listed	8
68	amenity_elevator_in_building	Flag for if amenity 'elevator in building' is listed	8

69	amenity_lock_on_bedroom_door	Flag for if amenity 'lock on bedroom door' is listed	8
70	amenity_essentials	Flag for if amenity 'essentials' is listed	8
71	amenity_gym	Flag for if amenity 'gym' is listed	8
72	amenity_family/kid_friendly	Flag for if amenity 'family/kid friendly' is listed	8
73	amenity_safety_card	Flag for if amenity 'safety card' is listed	8
74	amenity_translation_missing:_en.hosting_amenity_49	Flag for if amenity 'translation missing: en.hosting 49' is listed	8
75	amenity_cable_tv	Flag for if amenity 'cable tv' is listed	8
76	amenity_dog(s)	Flag for if amenity 'dog(s)' is listed	8
77	amenity_lockbox	Flag for if amenity 'lockbox' is listed	8
78	amenity_24_hour_check_in	Flag for if amenity '24 hour check in' is listed	8
79	amenity_hangers	Flag for if amenity 'hangers' is listed	8
80	amenity_tv	Flag for if amenity 'tv' is listed	8
81	amenity_pets_allowed	Flag for if amenity 'pets allowed' is listed	8
82	amenity_suitable_for_events	Flag for if amenity 'suitable for events' is listed	8
83	amenity_pool	Flag for if amenity 'pool' is listed	8
84	price_per_person	Price divided by guest count	10
85	price_per_bed	Price divided by number of beds	10
86	price_per_bedroom	Price divided by number of bedrooms	10
87	price_per_bed_plus_bath	Price divided by sum of beds and baths	10
88	complete_price	Total cost including base price, security deposit and cleaning fee	10
89	beds_per_bedrooms	Number of beds divided by bedrooms	10
90	bathrooms_per_accommodates	Bathrooms divided by accommodates	10
91	bathrooms_per_bedrooms	Bathrooms divided by bedrooms	10
92	bedrooms_per_accommodates	Bedrooms divided by accommodates	10
93	accommodates_into_num_amenities	Ratio of accommodates to number of amenities	10
94	host_response_rate_into_availability_365	Host response rate × availability_365	10
95	host_response_rate_into_num_verifications	Host response rate × num verifications	10
96	host_experience	Host months × host response rate	10
97	availability_experience	availability_365 × host months	10
98	host_acceptance_rate_into_listings_count	Host acceptance rate × listing count	10
99	num_verifications_into_listings_count	Num verifications × listing count	10
100	response_speed	Ordinal speed score from response text	10
101	host_responsiveness	Numeric value from response time mapping	10
102	host_experience_load	Host months divided by listings count	10
103	host_nearby	Flag indicating host lives near the listing	10

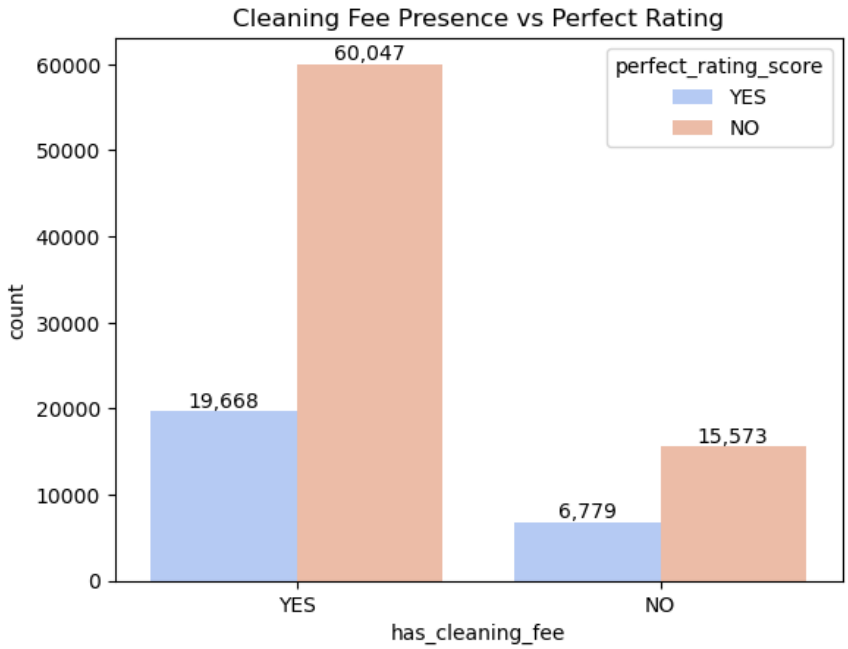
104	crime_incidents	Number of crime incidents joined by state	27
105	tfidf_clean	TF-IDF indicator: keyword 'clean' within top terms	35
106	tfidf_quiet	TF-IDF indicator: keyword 'quiet' within top terms	35
107	tfidf_cozy	TF-IDF indicator: keyword 'cozy' within top terms	35
108	tfidf_comfortable	TF-IDF indicator: keyword 'comfortable' within top terms	35
109	tfidf_spacious	TF-IDF indicator: keyword 'spacious' within top terms	35
110	tfidf_modern	TF-IDF indicator: keyword 'modern' within top terms	35
111	tfidf_renovated	TF-IDF indicator: keyword 'renovated' within top terms	35
112	tfidf_safe	TF-IDF indicator: keyword 'safe' within top terms	35
113	tfidf_private	TF-IDF indicator: keyword 'private' within top terms	35
114	tfidf_new	TF-IDF indicator: keyword 'new' within top terms	35
115	tfidf_large	TF-IDF indicator: keyword 'large' within top terms	35
116	tfidf_plentywifi	TF-IDF indicator: keyword 'plentywifi' within top terms	35
117	tfidf_tv	TF-IDF indicator: keyword 'tv' within top terms	35
118	tfidf_washer	TF-IDF indicator: keyword 'washer' within top terms	35
119	tfidf_dryer	TF-IDF indicator: keyword 'dryer' within top terms	35
120	tfidf_kitchen	TF-IDF indicator: keyword 'kitchen' within top terms	35
121	tfidf_bathroom	TF-IDF indicator: keyword 'bathroom' within top terms	35
122	tfidf_queen	TF-IDF indicator: keyword 'queen' within top terms	35
123	tfidf_queen_size	TF-IDF indicator: keyword 'queen_size' within top terms	35
124	tfidf_sharedpool	TF-IDF indicator: keyword 'sharedpool' within top terms	35
125	tfidf_laundry	TF-IDF indicator: keyword 'laundry' within top terms	35
126	tfidf_cable	TF-IDF indicator: keyword 'cable' within top terms	35
127	tfidf_air	TF-IDF indicator: keyword 'air' within top terms	35
128	tfidf_furnished	TF-IDF indicator: keyword 'furnished' within top terms	35
129	tfidf_towels	TF-IDF indicator: keyword 'towels' within top terms	35
130	tfidf_living_room	TF-IDF indicator: keyword 'living_room' within top terms	35
131	tfidf_walk	TF-IDF indicator: keyword 'walk' within top terms	35
132	tfidf_walking	TF-IDF indicator: keyword 'walking' within top terms	35
133	tfidf_walking_distance	TF-IDF indicator: keyword 'walking_distance' within top terms	35
134	tfidf_blocks_away	TF-IDF indicator: keyword 'blocks_away' within top terms	35
135	tfidf_minute_walk	TF-IDF indicator: keyword 'minute_walk' within top terms	35
136	tfidf_bus	TF-IDF indicator: keyword 'bus' within top terms	35
137	tfidf_metro	TF-IDF indicator: keyword 'metro' within top terms	35
138	tfidf_train	TF-IDF indicator: keyword 'train' within top terms	35
139	tfidf_subway	TF-IDF indicator: keyword 'subway' within top terms	35
140	tfidf_station	TF-IDF indicator: keyword 'station' within top terms	35
141	tfidf_transportation	TF-IDF indicator: keyword 'transportation' within top terms	35
142	tfidf_easystreet_parking	TF-IDF indicator: keyword 'easystreet_parking' within top terms	35

143	tfidf_parking	TF-IDF indicator: keyword 'parking' within top terms	35
144	tfidf_access	TF-IDF indicator: keyword 'access' within top terms	35
145	tfidf_central	TF-IDF indicator: keyword 'central' within top terms	35
146	tfidf_downtown	TF-IDF indicator: keyword 'downtown' within top terms	35
147	tfidf_minutes	TF-IDF indicator: keyword 'minutes' within top terms	35
148	tfidf_city	TF-IDF indicator: keyword 'city' within top terms	35
149	tfidf_welcome	TF-IDF indicator: keyword 'welcome' within top terms	35
150	tfidf_enjoy	TF-IDF indicator: keyword 'enjoy' within top terms	35
151	tfidf_amazing	TF-IDF indicator: keyword 'amazing' within top terms	35
152	tfidf_friendly	TF-IDF indicator: keyword 'friendly' within top terms	35
153	tfidf_best	TF-IDF indicator: keyword 'best' within top terms	35
154	tfidf_perfect	TF-IDF indicator: keyword 'perfect' within top terms	35
155	tfidf_good	TF-IDF indicator: keyword 'good' within top terms	35
156	tfidf_great	TF-IDF indicator: keyword 'great' within top terms	35
157	tfidf_view	TF-IDF indicator: keyword 'view' within top terms	35
158	tfidf_beautiful	TF-IDF indicator: keyword 'beautiful' within top terms	35
159	tfidf_light	TF-IDF indicator: keyword 'light' within top terms	35
160	tfidf_love	TF-IDF indicator: keyword 'love' within top terms	35
161	tfidf_nice	TF-IDF indicator: keyword 'nice' within top terms	35
162	tfidf_welcoming	TF-IDF indicator: keyword 'welcoming' within top terms	35
163	tfidf_helpful	TF-IDF indicator: keyword 'helpful' within top terms	35
164	tfidf_responsive	TF-IDF indicator: keyword 'responsive' within top terms	35
165	tfidf_heart	TF-IDF indicator: keyword 'heart' within top terms	35
166	tfidf_historic	TF-IDF indicator: keyword 'historic' within top terms	35
167	tfidf_available	TF-IDF indicator: keyword 'available' within top terms	35
168	tfidf_free	TF-IDF indicator: keyword 'free' within top terms	35
169	tfidf_restaurants	TF-IDF indicator: keyword 'restaurants' within top terms	35
170	tfidf_bars	TF-IDF indicator: keyword 'bars' within top terms	35
171	tfidf_coffee	TF-IDF indicator: keyword 'coffee' within top terms	35
172	tfidf_shops	TF-IDF indicator: keyword 'shops' within top terms	35
173	tfidf_loft	TF-IDF indicator: keyword 'loft' within top terms	35
174	tfidf_studio	TF-IDF indicator: keyword 'studio' within top terms	35
175	tfidf_patio	TF-IDF indicator: keyword 'patio' within top terms	35
176	tfidf_garden	TF-IDF indicator: keyword 'garden' within top terms	35
177	tfidf_dining	TF-IDF indicator: keyword 'dining' within top terms	35
178	tfidf_shoppingsolo_adventurers	TF-IDF indicator: keyword 'shoppingsolo_adventurers' within top terms	35
179	tfidf_local	TF-IDF indicator: keyword 'local' within top terms	35
180	tfidf_neighborhood	TF-IDF indicator: keyword 'neighborhood' within top terms	35
181	tfidf_art	TF-IDF indicator: keyword 'art' within top terms	35

182	tfidf_music	TF-IDF indicator: keyword 'music' within top terms	35
183	tfidf_park	TF-IDF indicator: keyword 'park' within top terms	35
184	tfidf_close	TF-IDF indicator: keyword 'close' within top terms	35
185	tfidf_beach	TF-IDF indicator: keyword 'beach' within top terms	35
186	verbosity_word_count	Verbosity metric from word count	44
187	verbosity_scaled	Verbosity metric from word count	45
188	description_sentiment	VADER sentiment score from description field	48
189	summary_sentiment	VADER sentiment score from summary field	48
190	house_rules_sentiment	VADER sentiment score from house_rules field	48
191	host_about_sentiment	VADER sentiment score from host_about field	48
192	interaction_sentiment	VADER sentiment score from interaction field	48

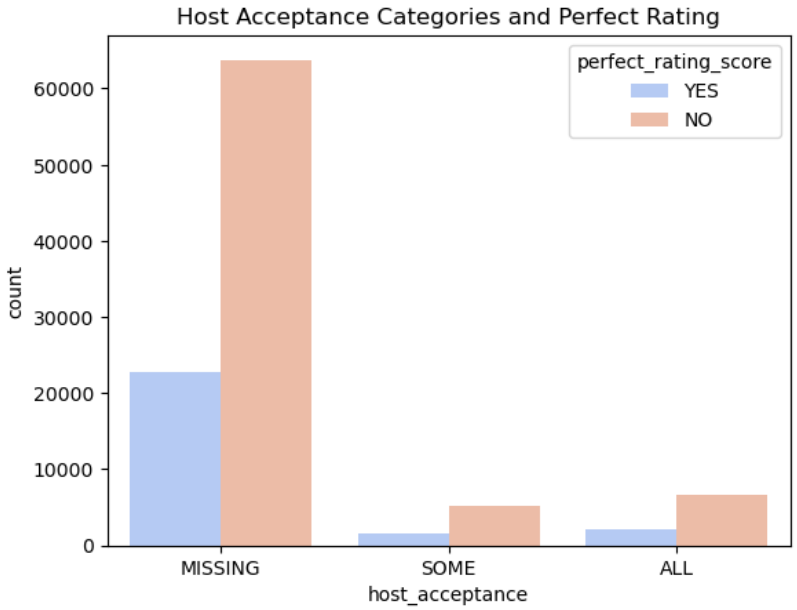
1) Graphs or tables demonstrating useful or interesting insights regarding features in the dataset.

Graph 1:



While listings with a cleaning_fee are more frequent, those without a cleaning_fee have a higher proportion of perfect ratings, suggesting that avoiding extra fees may improve guest satisfaction

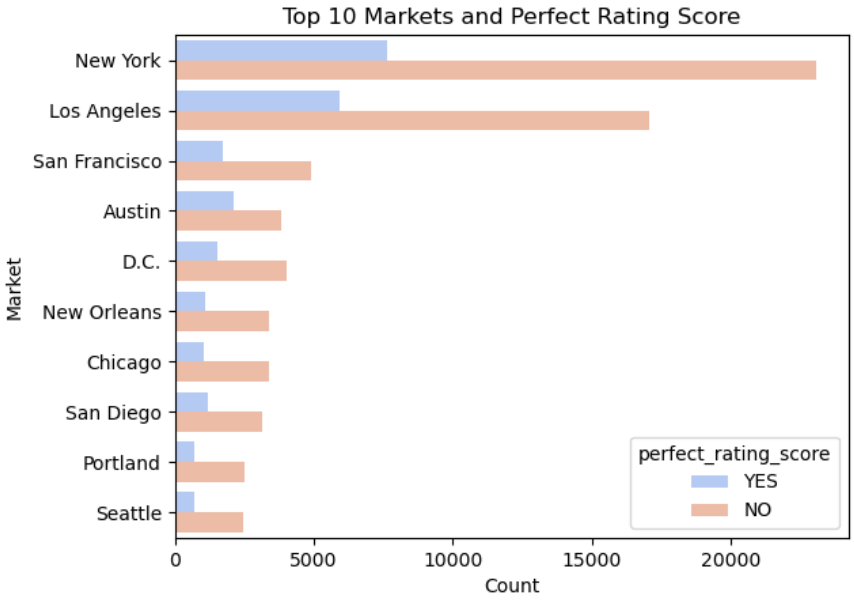
Graph 2:



A huge number of listings have missing values for the host_acceptance field, yet they still account for a considerable volume of perfect ratings. This may indicate that:

- Many hosts likely accept most requests without explicitly setting acceptance rules, leading to a good guest experience even if the field is blank
- The missing data may reflect system defaults, older listings, or hosts who never updated optional profile sections, not necessarily poor hosting standards

Graph 3:

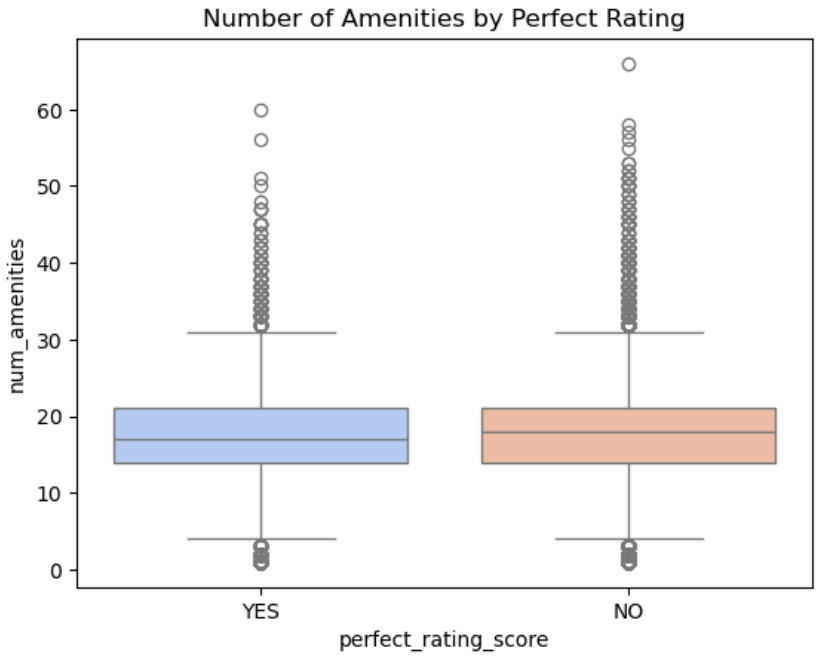


Cross Tabular Table:

perfect_rating_score	NO	YES
market		
Austin	0.646582	0.353418
D.C.	0.720813	0.279187
San Diego	0.726555	0.273445
San Francisco	0.742616	0.257384
Los Angeles	0.742717	0.257283
New York	0.751685	0.248315
New Orleans	0.754351	0.245649
Chicago	0.762750	0.237250
Seattle	0.773888	0.226112
Portland	0.785378	0.214622

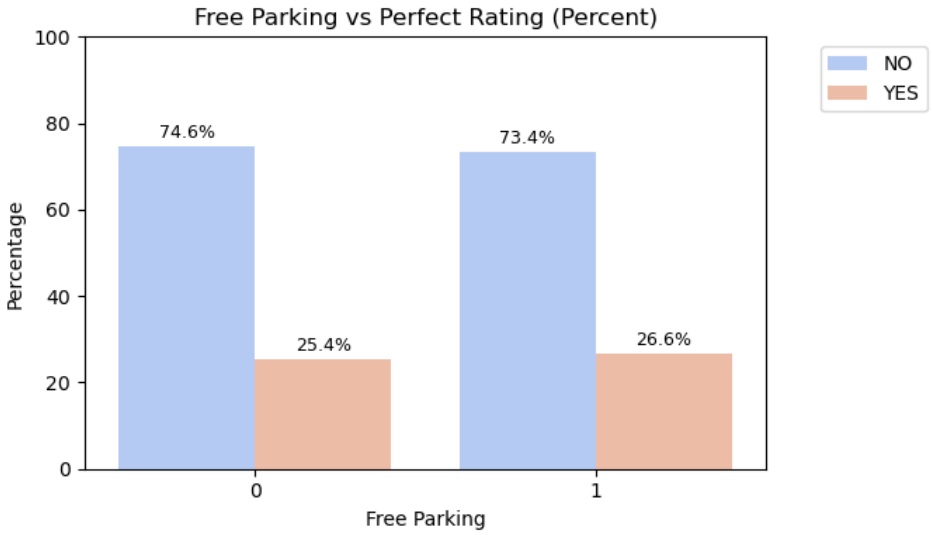
While Los Angeles & New York dominate in listing volume, cities like Austin and D.C. show a higher proportion of perfect ratings. In contrast, markets like Portland and Seattle have the lowest share of perfect scores. This suggests that guest satisfaction varies by market

Graph 4:



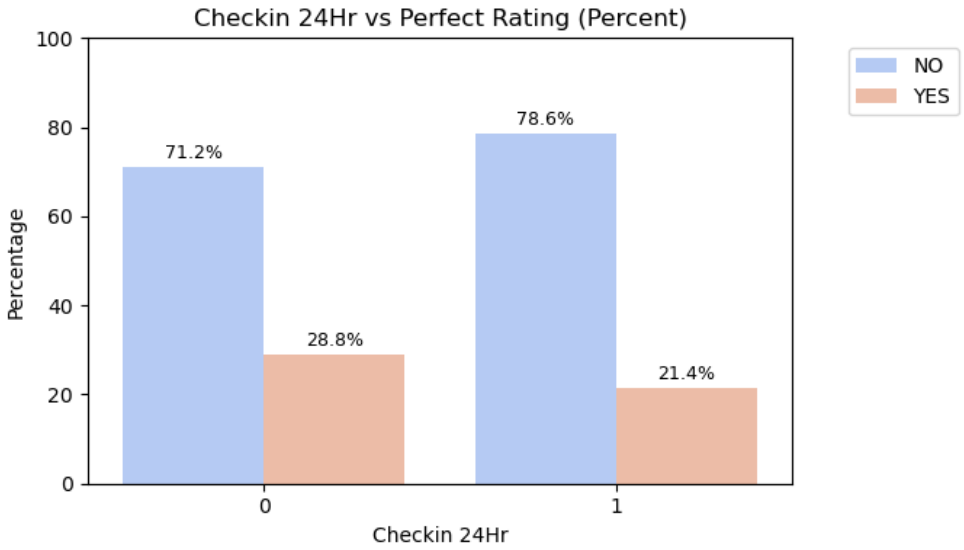
Listings with perfect ratings tend to have a slightly higher median number of amenities compared to those without. While the difference isn't large, it suggests that providing more amenities may contribute positively to guest satisfaction

Graph 5:



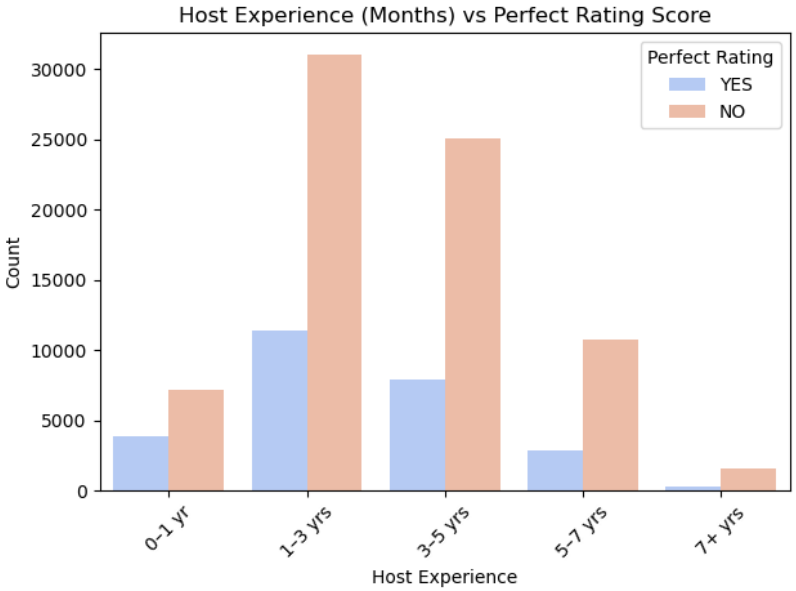
The perfect rating proportion is slightly higher when free parking is available (26.6% vs 25.4%). The difference is small but may signal added guest convenience

Graph 6:



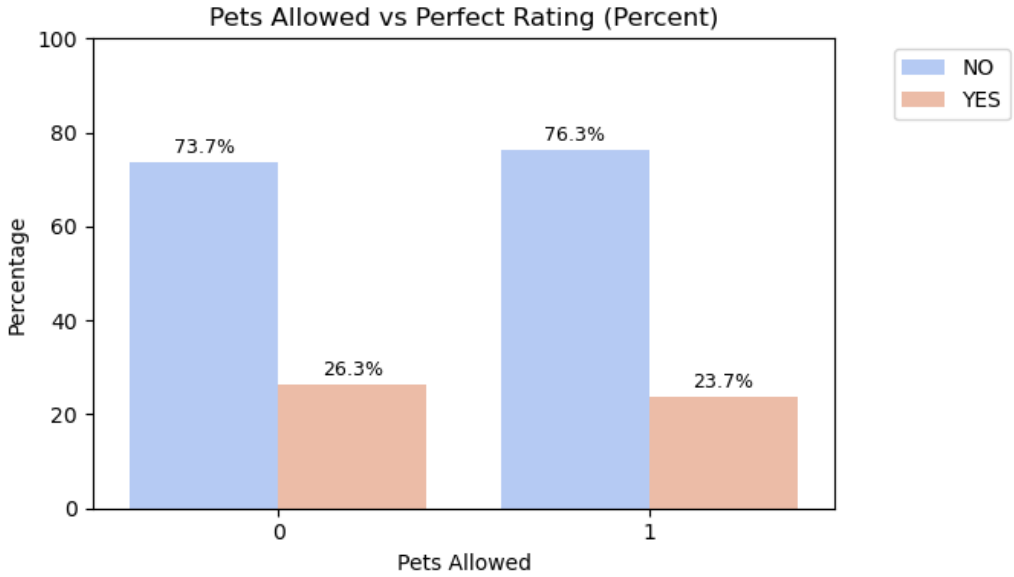
Listings without 24-hour check-in have a higher proportion of perfect ratings (28.8% vs 21.4%). It could be because 24-hour check-in listings are often automated, high-turnover rentals where convenience replaces personalized service.

Graph 7:



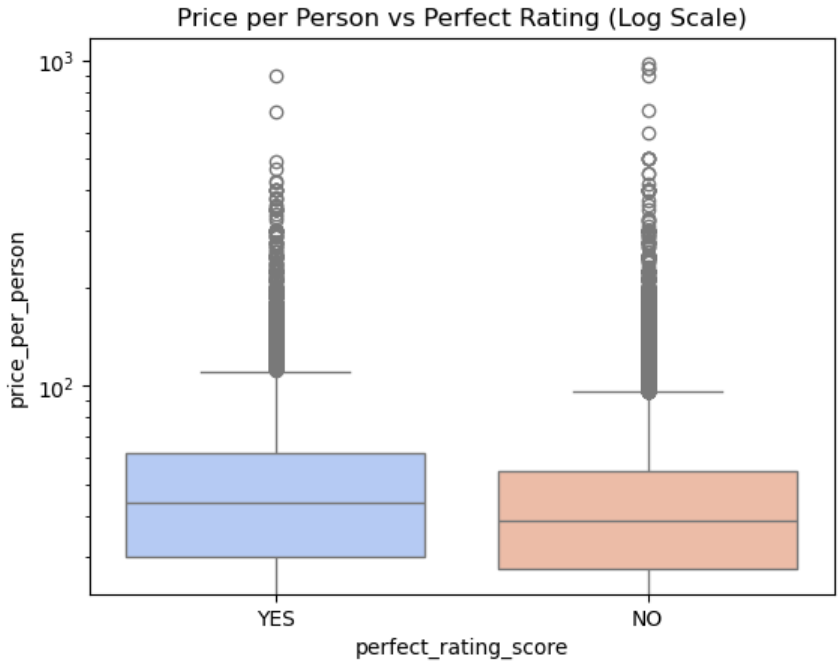
There is a clear decline in the proportion of perfect ratings as host experience increases. Hosts with less than a year of experience have the highest share of perfect ratings (34.7%), while those with over 7 years drop to just 16.8% possibly because newer hosts tend to put in extra effort to impress guests and build a strong reputation early on. Also, recent listings may be more updated/aligned with current guest preferences, while more experienced hosts might become complacent over time, leading to slightly lower ratings.

Graph 8:



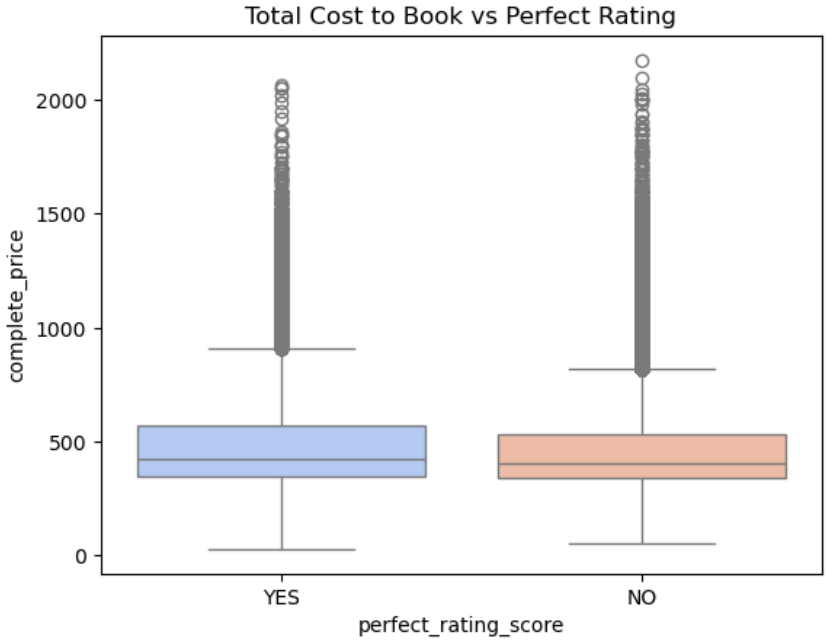
Listings not allowing pets have a higher share of perfect ratings (26.3% vs 23.7%). Pet-friendly listings might bring more variables (noise, cleanliness), affecting guest satisfaction

Graph 9:



The log-scaled plot shows that listings with perfect ratings generally have a slightly lower median price_per_person compared to those without. However, both distributions have long tails with expensive outliers, indicating that while affordability may help, high prices don't necessarily prevent perfect ratings if value is delivered.

Graph 10:

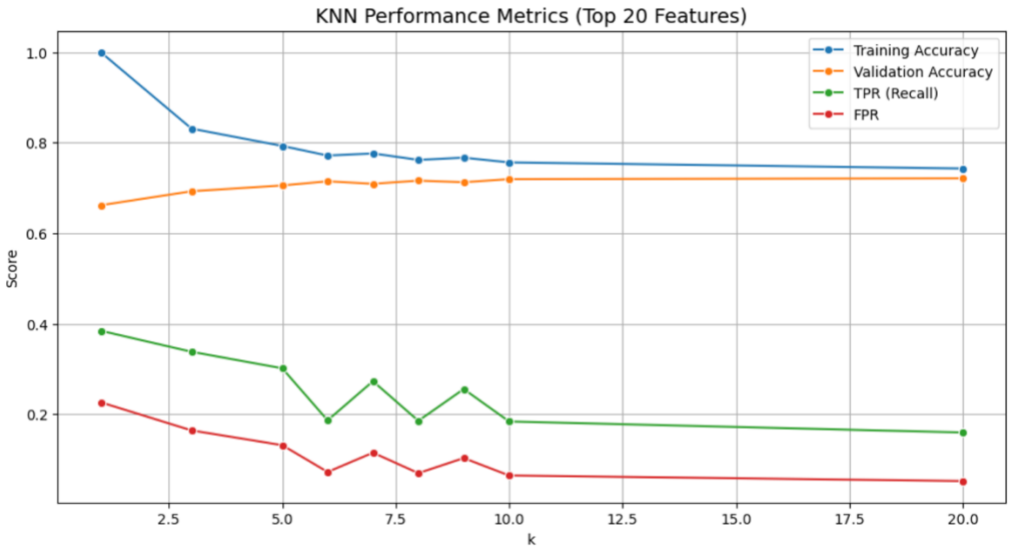


The distribution of complete_price (total booking cost) is very similar between listings with and without perfect ratings. Both groups exhibit a wide range of prices with several high outliers, suggesting that total cost alone does not strongly differentiate highly rated listings.

Section 4: Evaluation and Modeling

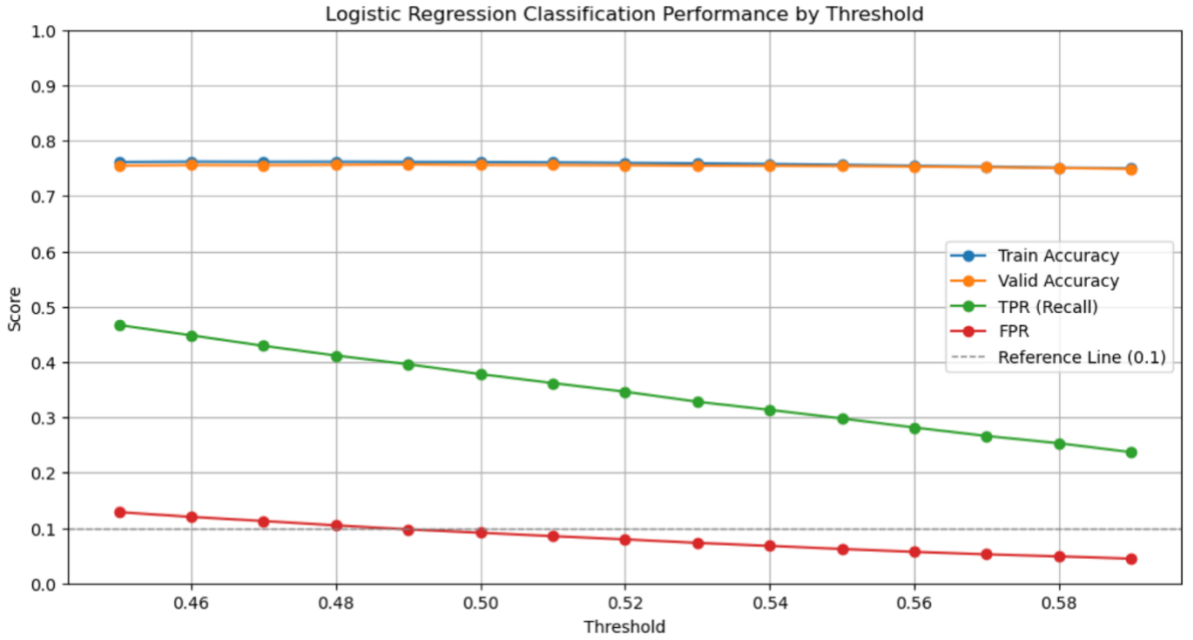
The winning model is the XGBoost model. Post intensive feature engineering, the final dataset had over 250 features consisting of original listing characteristics, engineered ratios, amenity indicators, sentiment scores, and selected TF-IDF tokens from listing descriptions. Greedy forward selection was used to enhance the True Positive Rate (TPR) under a 10% False Positive Rate (FPR) constraint, creating a model that balanced sensitivity with overfitting. The model that was finally used was trained with 85 selected features, included the external dataset from Kaggle which detailed crime_incidents in each state and tuned using cross-validation. Estimated training performance was a TPR of 0.4822 at FPR $\leq$ 0.10, with validation performance closely matching, implying good generalization. This model was chosen as it maximized the contest metric without going above the false positive threshold. Final predictions submitted for the contest were generated in line [91] of the Python code.

Graph 11



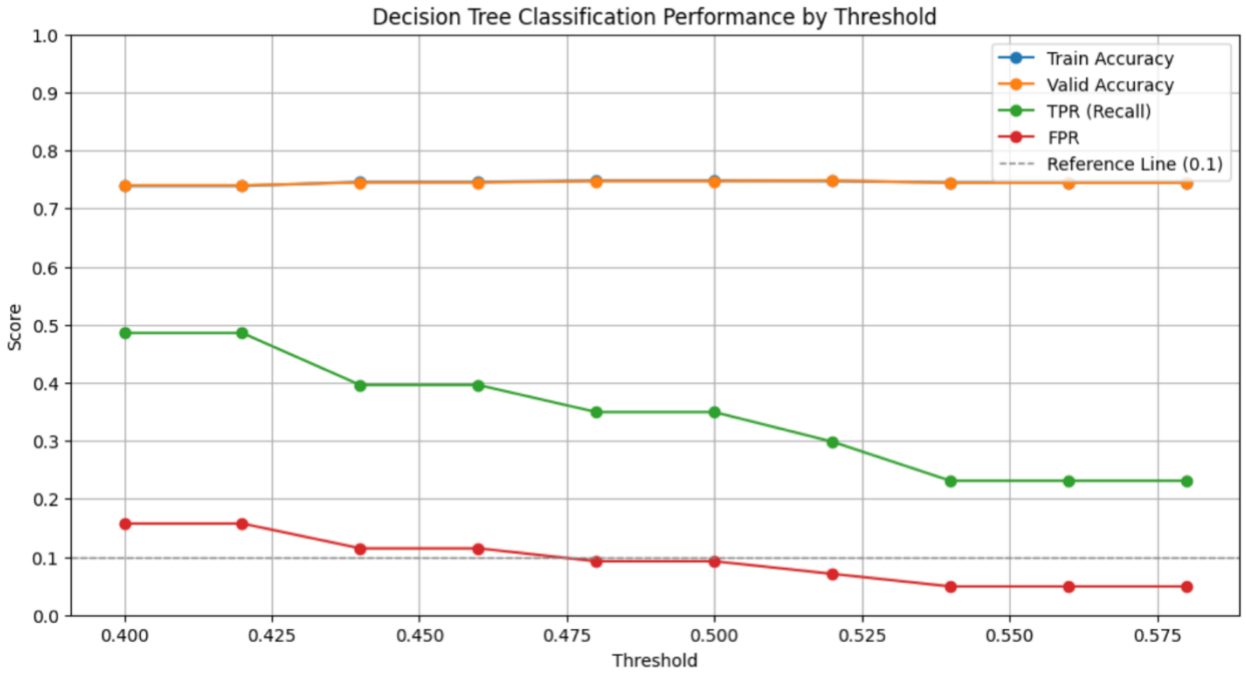
One of the first models we explored was K-Nearest Neighbors (KNN), which falls under the instance-based learning family. We implemented it using KNeighborsClassifier from sklearn.neighbors, training it with scaled numeric features. The model's performance obtained a a training accuracy of 75.65%, validation accuracy of 71.95%, with a True Positive Rate (TPR) of 18.40% and a False Positive Rate (FPR) of 6.47% (while being under threshold of 0.51). We analyzed the performance using a simple 70/30 train/validation split. Many different k values were tested including [1, 3, 5, 6, 7, 8, 9, 10, 20]. Performance was evaluated on the basis of TPR, FPR, and accuracy. The top 20 features identified through forward feature selection were adopted in this model. The model was implemented at lines [65] - [67] of the notebook. Graph 11 was created to show how each performance metric changed across different k values, enabling us to evaluate the sensitivity of the model to k. The fitting curve visualizes that although training accuracy goes down with higher k, validation accuracy stabilizes, implying decreased overfitting and enhanced generalization. Although this model was eventually outperformed by more complex models, KNN provided a useful baseline for the modeling process.

Graph 12



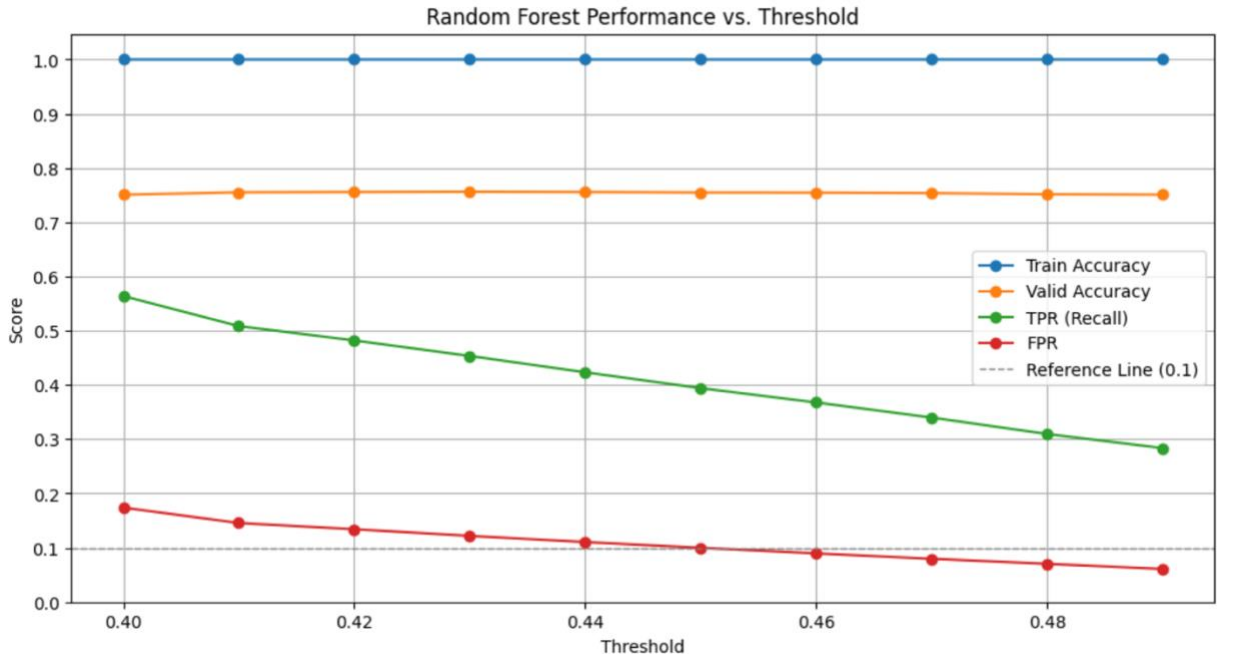
We also adopted a logistic regression model with LogisticRegression from sklearn.linear_model. This model was trained using scaled numeric features with StandardScaler. Model was implemented at lines [61] - [64] of the notebook. The model obtained a training accuracy of 76.17%, validation accuracy of 75.69% and a TPR (Recall) of 39.59% with an FPR of 9.76% when evaluated at a threshold of 0.49. Generalization performance was computed through both a simple train/validation split (70/30) and 5-fold stratified cross-validation. Cross-validation suggested consistent performance with a mean accuracy of 75.89%, TPR of 39.03%, and FPR of 9.25%. We investigated a range of classification thresholds that fall between 0.45 and 0.59. Then, we visualized how accuracy, recall, and false positive rate changed in response. Even though no traditional hyperparameter tuning (e.g., grid search) was adopted beyond increasing max_iter to 1000, threshold selection was applied as a prime tuning mechanism. Graph 12 was made to analyze how accuracy, recall, and false positive rate change amongst different classification thresholds, assisting us with selecting the most balanced operating point in terms of sensitivity and specificity. The final feature set was decided on through a greedy forward selection strategy that gave priority to TPR while maintaining FPR below 10%.

Graph 13



We used a decision tree model with DecisionTreeClassifier from sklearn.tree, which is part of the decision tree family of models. The model was trained using a max_depth of 5 and random_state of 42, using scaled numeric features. Model was implemented at lines [68] - [70] of the notebook. To evaluate performance, we adopted a 70/30 train-validation split, where the validation set has 30% of the training data. The best-performing set of features originated from numeric preprocessing applied before in the pipeline. The model obtained a training accuracy of 74.79% and a validation accuracy of 74.72%, with a True Positive Rate (TPR) of 34.94% and False Positive Rate (FPR) of 9.24% at a threshold of 0.48. Even though no hyperparameter tuning via grid search was conducted, threshold tuning was applied using a range between 0.4 and 0.6. Graph 13 was plotted to show how accuracy, TPR, and FPR changed across thresholds, assisting us with choosing an operating point that balanced recall and specificity.

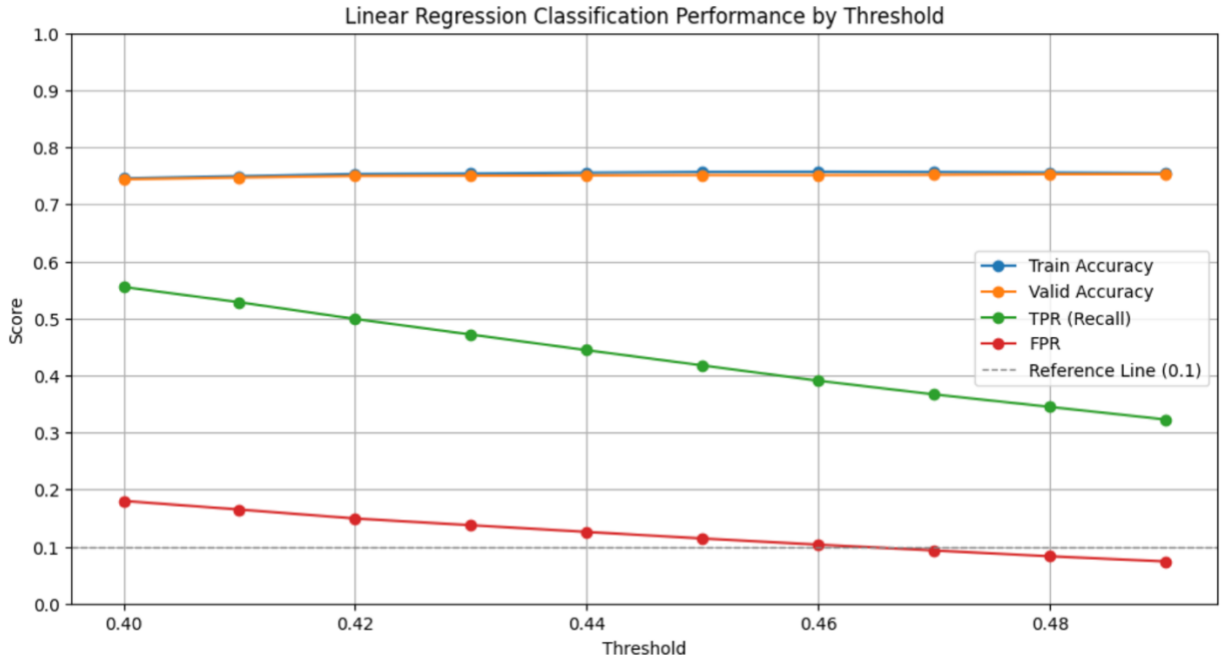
Graph 14



We adopted a Random Forest classifier (with RandomForestClassifier from sklearn.ensemble), which is part of the ensemble tree-based model family. The model was trained using scaled numeric features withStandardScaler. Model was implemented at lines [71] - [73] in the notebook. The model's generalization performance provided a a training accuracy of 99.99%, validation accuracy of 75.45%, TPR of 36.78%, and FPR of 8.96% when analyzed at a threshold of 0.46. These metrics were computed through a simple 70/30 train/validation split, with predicted probabilities used to evaluate performance amongst thresholds between 0.40 and 0.49.

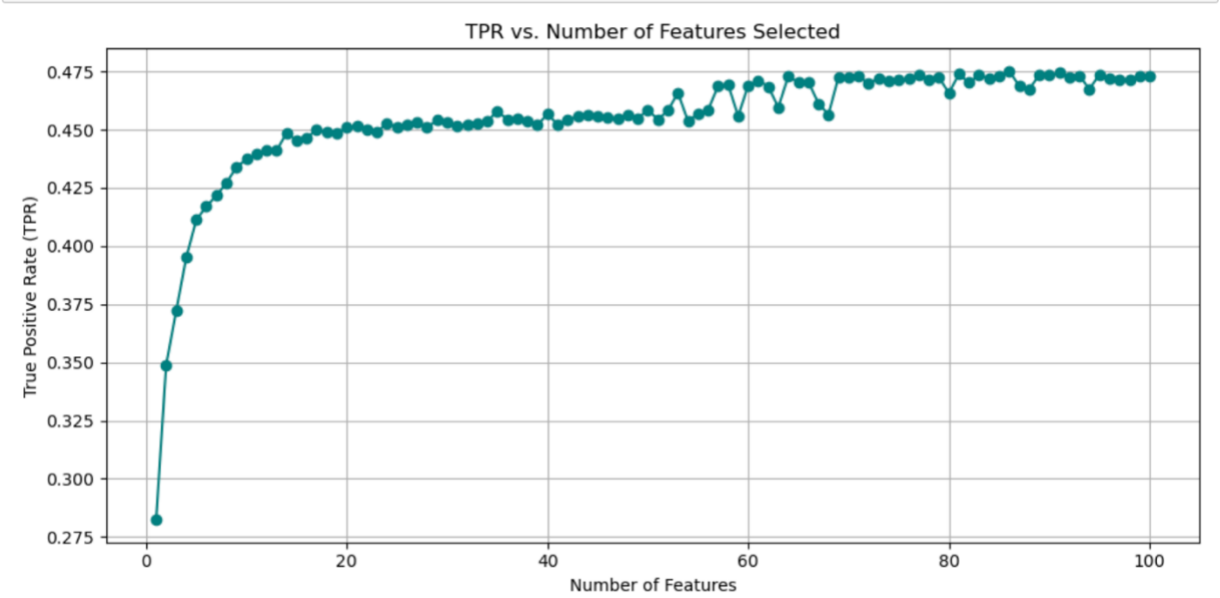
Although the n_estimators parameter was set as 100 and random_state was set as 42 for reproducibility, no exhaustive hyperparameter tuning (e.g., grid search) was done. This was a conscious choice as we witnessed diminishing returns in performance, and we wanted to focus effort on model interpretability and feature selection across competing classifiers. Graph 14 was created to show how accuracy, TPR, and FPR shifted with various thresholds. This graph helped find the optimal threshold (0.46) that maximized TPR while maintaining FPR under 10%. The Random Forest model generally acted as a good benchmark in our modeling pipeline, balancing predictive power and interpretability.

Graph 15



We also used a regression-based approach using the LinearRegression model from sklearn.linear_model. The model was trained with scaled numeric features (categorical variables were left out), and prediction outputs were changed into binary values through adopting a classification threshold of 0.5. Generalization performance was analyzed through both a simple 70/30 train-validation split and a sweep across many thresholds that ranged from 0.40 to 0.49. At the chosen threshold of 0.47, the model obtained an R^2 of 0.1844 and RMSE of 0.4086. In terms of classification performance, it earned a training accuracy of 75.66%, validation accuracy of 75.17%, a TPR of 36.68%, and an FPR of 9.32%. The model was applied and evaluated in line [57] - [60] of the code. No traditional hyperparameter tuning was applicable as linear regression has no primary hyperparameters in its basic form. Still, threshold tuning was applied to find the optimal classification point under performance trade-offs. Graph 15 was also made to see how validation accuracy, recall, and false positive rate shifted across the threshold range, assisting with the interpretation of model behavior

Graph 16



We chose XGBoost (Extreme Gradient Boosting) as our final model because of its high predictive performance and flexibility with high-dimensional data. The model was built with the XGBClassifier from the xgboost Python package. Training and evaluation were done across lines [80] - [89] in the notebook. The model was trained with scaled numeric features, and predictions were thresholded at 0.4901 for optimal sensitivity below a 10% false positive constraint. The model obtained a validation accuracy of 78.00 %, with a True Positive Rate (TPR) of 48.22 % and a False Positive Rate (FPR) of 10 % on the validation fold and a train accuracy of 79.58% on the training fold. Generalization performance was evaluated with a 70/30 train-validation split

We did a forward feature selection process focused on maximizing TPR while having FPR below 10%, adopting the same performance metrics from each fold. This method resulted in the identification of the top 85 features, which consistently gave the best balance between sensitivity and generalization. The model also picked crime_incidents at feature 21(with an importance of 1.33%) which was added from our external data source. As seen in the TPR vs. Number of Features plot (Graph 16), TPR rapidly went up in the early stages of feature inclusion but started to plateau around 85 features. This supports our decision to choose this cutoff point and prevent unneeded model complexity.

The 85 features along with their importance which the model used were the following:

	feature_name	Importance
0	response_speed	0.075622
1	first_review_months	0.054504

2	host_response_time_within_an_hour	0.052485
3	host_responsiveness	0.044947
4	host_acceptance_MISSING	0.040880
5	instant_bookable	0.038156
6	amenity_washer	0.032478
7	availability_30	0.028676
8	amenity_24_hour_check_in	0.028425
9	state_WA	0.019744
10	city_New_York	0.018547
11	property_category_house	0.018366
12	availability_365	0.016651
13	charges_for_extra_YES	0.016545
14	state_MA	0.015350
15	price_per_person	0.015231
16	minimum_nights	0.014229
17	host_is_superhost	0.014185
18	bathrooms	0.013681
19	cancellation_policy_strict	0.013655
20	crime_incidents	0.013333
21	amenity_indoor_fireplace	0.013288
22	state_NY	0.013017
23	city_Seattle	0.011285
24	host_acceptance_rate_into_listings_count	0.011148
25	amenity_hair_dryer	0.010796

26	checkin_24hr	0.010009
27	state_TN	0.009577
28	availability_60	0.009183
29	price	0.009134
30	state_CO	0.009019
31	city_San_Diego	0.008623
32	amenity_dog(s)	0.008367
33	has_min_nights_YES	0.008291
34	bedrooms_per_accommodates	0.008288
35	price_per_bed	0.008097
36	availability_90	0.008066
37	tfidf_art	0.008000
38	cancellation_policy_moderate	0.007929
39	num_features	0.007909
40	host_experience_load	0.007744
41	amenity_tv	0.007689
42	amenity_pets_live_on_this_property	0.007298
43	host_response_rate	0.007284
44	room_type_Private_room	0.007107
45	host_response_time_missing	0.007044
46	host_listings_count	0.007014
47	require_guest_phone_verification	0.006999
48	guests_included	0.006975
49	extra_people	0.006873

50	amenity_elevator_in_building	0.006799
51	verbosity_scaled	0.006776
52	cleaning_fee	0.006699
53	amenity_self_check_in	0.006672
54	maximum_nights	0.006592
55	amenity_first_aid_kit	0.006443
56	beds_per_bedrooms	0.006188
57	tfidf_patio	0.005789
58	amenity_internet	0.005731
59	tfidf_view	0.005675
60	market_Los_Angeles	0.005326
61	amenity_translation_missing:_en.hosting_amenit...	0.005230
62	num_verifications_into_listings_count	0.005191
63	city_Portland	0.004879
64	tfidf_dryer	0.004860
65	tfidf_station	0.004820
66	tfidf_access	0.004466
67	amenity_fire_extinguisher	0.004430
68	amenity_shampoo	0.004411
69	market_New_Orleans	0.004282
70	tfidf_furnished	0.004040
71	bedrooms	0.003934
72	amenity_lock_on_bedroom_door	0.003901
73	beds	0.003839

74	tfidf_train	0.003589
75	amenity_suitable_for_events	0.003522
76	summary_sentiment	0.003393
77	tfidf_coffee	0.003350
78	tfidf_queen_size	0.003264
79	num_verifications	0.003245
80	tfidf_cozy	0.003228
81	has_cleaning_fee_YES	0.003054
82	neighborhood_group_Brooklyn	0.003038
83	tfidf_comfortable	0.002952
84	tfidf_new	0.002650

To choose the best hyperparameters for XGBoost, we did an extensive grid search with a predefined parameter grid. The grid spanned a wide range of values across critical tuning parameters, such as n_estimators [200, 300, 400, 500], learning_rate [0.02, 0.05, 0.07], max_depth [3, 4, 5, 6], min_child_weight [2, 3, 5], subsample [0.6, 0.8, 1.0], colsample_bytree [0.5, 0.7, 0.9], gamma [0, 1, 2], scale_pos_weight [1, 2, 4], reg_alpha [0, 0.1, 0.3, 0.5], and reg_lambda [1, 3, 5, 7]. We also made use_label_encoder=False and eval_metric='logloss' for compatibility with the binary classification task. This hyperparameter tuning process was adopted to assist with the selection of the final model configuration, which ultimately made use of the following parameters:

{'use_label_encoder': False, 'subsample': 1.0, 'scale_pos_weight': 4, 'reg_lambda': 5, 'reg_alpha': 0.5, 'n_estimators': 400, 'min_child_weight': 3, 'max_depth': 5, 'learning_rate': 0.07, 'gamma': 2, 'eval_metric': 'logloss', 'colsample_bytree': 0.7}

Section 5: Reflection/takeaways

1) What did your group do well?

One of the best things we did was clean the data really well in the beginning. We removed the extra columns, handled missing values, and created new useful variables like *price per person* and *host since (in months)* early on. This groundwork really helped us get a clearer picture of the dataset and made the rest of the process smoother. Even though we added more features later, having a

solid base meant we didn't waste time fixing things later. It also allowed us to concentrate more on testing models and improving performance instead of constantly going back to clean the data again.

2) What were the main challenges?

One of the biggest challenges we encountered was choosing which features and models to use. In most machine learning projects, it's generally more effective to lean on automated methods like random feature selection or techniques that let the model determine which features are significant. These methods help minimize bias and make the process more objective. However, in our case, we discovered that manually selecting features after ranking them gave us better results.

Even though manual selection can sometimes introduce bias, it worked better for us with this specific dataset. The data was not very clean, and some features needed human judgment to understand their value. We were careful to test our selections and make sure they were improving the model's performance. If we had more time, we would have liked to try a mix of manual and automated methods to see if we could get even better results.

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

Looking back, we could have decided to fully switch to Contest 1 a bit sooner. While we didn't delay too much, we did use up two of our eight submissions testing out Contest 2 just to be sure, which limited the number of final attempts we had for the main contest. If we had made that decision sooner, we could have utilized all our submissions more effectively to boost our performance.

Another thing we would have approached differently is experimenting with a higher threshold earlier on. For most of the project, we played it safe with a threshold around 9 because we didn't want to take risks. It wasn't until the end that we tested with a threshold of 10, which worked well for the team. If we had tested that earlier, we could have avoided using extra submissions just for that purpose. So, planning our iterations better and experimenting with different thresholds sooner would have helped. This taught us that making small adjustments and testing them early can save a lot of time and effort later.

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

If we had additional time, we would have explored the external datasets more thoroughly to better understand how they interacted with the original data. For instance, the crime dataset we used contributed meaningfully to our top variables, but we were unable to include the California crime data due to limited time for preprocessing and integration. Given more time, we would have incorporated that as well and considered other external datasets that could further improve model performance.

We also would have focused more on model tuning. While we tested different models, we were not able to experiment extensively with hyperparameter settings. With more time, we could have refined our models further to achieve better accuracy, TPR, and FPR values.

Lastly, we considered using neural networks but refrained due to time limitations and tuning complexity. They're powerful but prone to overfitting without careful control, which we didn't have the capacity to handle within the scope of this project.

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

Start early and clean your data properly from the beginning. Don't rush into modeling before fully understanding what the target column means and how the evaluation works. Keep track of everything you try so you can see what helps and what doesn't. Try to submit something for every contest, even if it's just a small change. This helps you see how even small changes affect the actual predictions. Sometimes your local model looks better, but the leaderboard results shows something else. Also, don't be afraid to take calculated risks. Experiment with higher thresholds or bolder feature engineering ideas, especially in the later stages. You never know what might work unless you try. Start with something simple and keep refining it as you go.

6) Which skills did we improve the most during the project?

Throughout the project, we experienced significant growth in three main areas: data cleaning, modeling, and evaluation metrics. When it came to data cleaning, we gained a much clearer insight into how missing values can hold valuable information. We realized that simply removing them could mean losing out on important patterns. Instead of dismissing nulls as just noise, we learned to take a moment to consider what they might signify.

In modeling, we became more confident with iterative experimentation. We started with simple baseline models and gradually added complexity, which helped us understand how each change impacted performance. This step-by-step approach taught us the importance of tuning and testing carefully instead of jumping to complex solutions too early.

We also deepened our understanding of evaluation metrics, especially the tradeoff between True Positive Rate and False Positive Rate. Learning how to navigate this balance was critical since our contest had a strict False Positive Rate threshold. Now, we have a much clearer and stronger sense of how to align model performance with specific business goals.

References

Agrawal, A. (2023, October 2). *US crime dataset*. Kaggle. https://www.kaggle.com/datasets/mrayushagrawal/us-crime-dataset