UNIVERSITY OF MARYLAND
ROBERT H. SMITH SCHOOL OF BUSINESS

BUDT 758T Final Project Report Spring 2023

Section 1: Team member names and contributions

Team member	Contribution
Deepak Dhole	1. Data cleaning & preparation, Data modeling, and evaluation. 2. Ensuring the report is in accordance with the rubric.
Abdulazeez Olukose	1. Data cleaning and the majority of the final report tasks. 2. Error debugging and creating insights/charts on various features.
Vishnu Kothagadi	1. Data cleaning & preparation, Data modeling, and evaluation. 2. Final report tasks and overall compilation of the report.
Aditya Shenoy	1. Generating learning curves and fitting curves for various models. 2. Compiling the final R code and proper documentation.
Hriday Mehta	1. Hyperparameter tuning for various models and choosing the best model. 2. Majority of the final report tasks.

Section 2: Business Understanding

The business case:

The predictive model we built can be beneficial to real estate firms planning to expand their current business by investing in buying new properties and rolling out a rental business model, similar to Airbnb.

Our model can act as a core profit driver for the firms as they plan to invest in select properties that have a promising return on investment (ROI). And our model predicts just the right indicator of profitability, i.e., perfect_rating_score which predicts whether or not a particular property listing is likely to have a high chance of getting a perfect rating score. Therefore the predictive model will act as a great guide in the business decision-making process of investing only in those properties that will have a high ROI.

Business actions:

On the other hand, the predictive model will also provide further insights into the specific features of different properties that translate into the company's profits. For instance, if we find out that 'host_is_superhost' predictor has a significant weightage in predicting the accurate perfect_rating_score for a given listing, then the firm can translate the same into their business decisions. That is, they can ensure that the owners of their property listings follow a certain protocol that aligns with the objective of turning them into a super host, and overall, driving the company's profits.

Value of the predictions:

By utilizing the power of our predictive model, Airbnb, and similar firms can further expand their businesses in new geographical locations across the world. They don't have to take the risk of a failed expansion resulting from uncertainty of various factors pertaining to the new location. By augmenting more variables relating to these factors, we can make the current predictive model adjust to the new location and therefore drive those predictions with a high perfect_rating_score to exactly find those properties that have a high probability of resulting in a perfect rating score in the future; if bought and rented out as a listing.

However, it is to be noted that since the model's True positive rate (TPR) is not very significant we cannot entirely rely on the positive 'Yes' predictions to make a decision of buying out a property. We would need to consider a lot of other market factors as well. However, since our model has a low False positive rate (FPR) of < 10%, it significantly reduces the risk of buying out those properties that would otherwise result in a bad rating score and thereby result in a potential loss for the company. This means that our model can support a risk-averse business strategy for incumbent firms that are trying to survive in the market.

In addition to the model acting as a guide in driving a higher ROI, it can also aid in the process of pinpointing the necessary steps that need to be taken to improve the current listings that are not receiving higher ratings. This implies that we can specifically target certain aspects of the listings and alter them by just the right amount via investing in underperforming areas to drive a higher number of perfect rating scores.

Section 3: Data Understanding and Data Preparation

1) ***Table***

ID	Feature Name	Brief Description	R Code Line Numbers
1	accommodates	Original feature from dataset	1-5
2	availability_365	Original feature from dataset	200-204
3	bed_type	Original feature from dataset	131
4	availability_30	Original feature from dataset	200-204
5	availability_90	Original feature from dataset	200-204
6	bathrooms	Original feature from dataset	141
7	bedrooms	Original feature from dataset	101
8	beds	Original feature from dataset	104
9	cancellation_policy	Original feature from dataset	95

10	cleaning_fee	Original feature from dataset	96
11	extra_people	Original feature from dataset	84
12	guests_included	Original feature from dataset	200-204
13	host_listings_count	Original feature from dataset	98
14	instant_bookable	Original feature from dataset	200-204
15	ppp_ind	New feature	126
16	has_min_nights	New feature	146
17	host_is_superhost	Original feature from dataset	106
18	host_identity_verified	Original feature from dataset	107
19	has_extra_fee	New feature	115
20	host_response	Original feature from dataset	143
21	room_type	Original feature from dataset	130

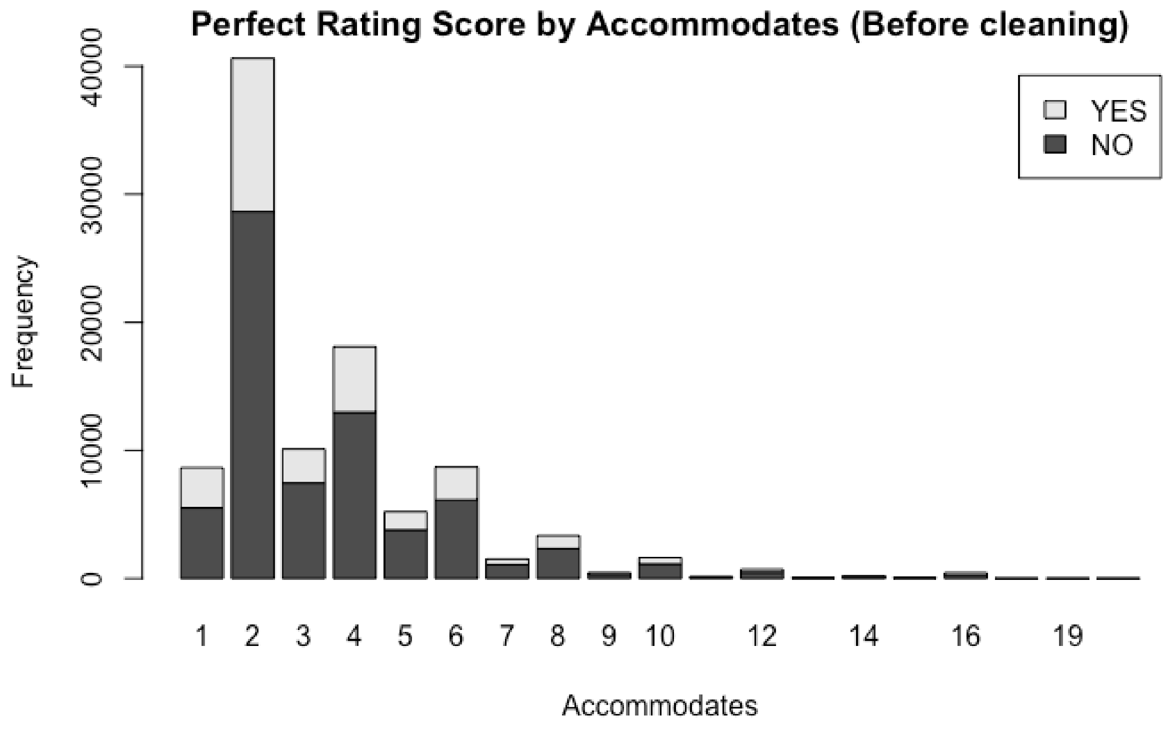
22	is_location_exact	Original feature from dataset	200-204
23	is_available	New feature	102
24	no_rules	New feature	103
25	market	Modified feature	158
26	maximum_nights	Original feature from dataset	200-204
27	minimum_nights	Original feature from dataset	200-204
28	monthly_available	New feature	157
29	price	Original feature from dataset	97
30	require_guest_phone_verification	Original feature from dataset	200-204
31	require_guest_profile_picture	Original feature from dataset	200-204
32	requires_license	Original feature from dataset	200-204
33	weekly_available	New feature	156
34	Amenities	Modified feature as dummy	76-79

35	transit_available	New feature	155
36	square_feet	New feature (**from external dataset**)*	192-196
37	is_available	New feature	98
38	is_available_60	New feature	99
39	is_available_90	New feature	100
40	is_available_365	New feature	101
41	no_rules	New feature	103
42	beds	Modified feature	104
43	host_is_superhost	Modified feature	105
44	host_identity_verified	Modified feature	106
45	zipcode	Modified feature	107
46	smart_location	New feature	108

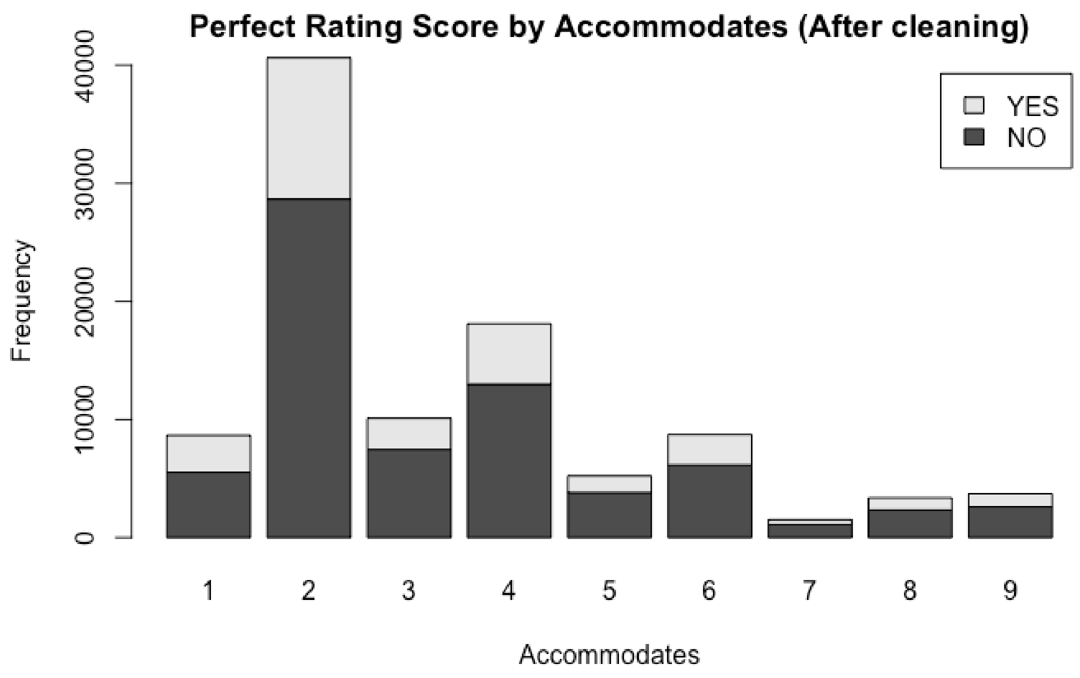
47	state	Modified feature	109
48	per_day_price_weekly	New feature	111
49	per_day_price_monthly	New feature	113
50	host_total_listings_count	New feature	114
51	host_info_available	New feature	116
52	guests_included	New feature	118
53	has_cleaning_fee	New feature	126
54	has_deposit	New feature	128

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

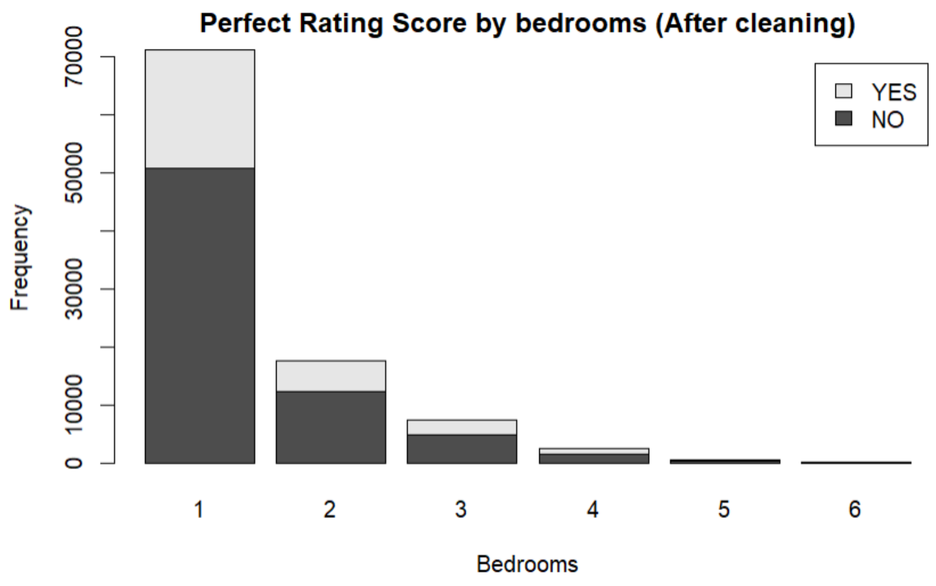
1. **Accommodates** - This is an original feature in the dataset which tells us about the number of guests that can stay in the listing. Based on the original dataset, it is not evenly spread. In order to clean this feature, we have limited the maximum number of accommodates to 9. Following graphs shows the distribution of the accommodation across all given listings.


Perfect Rating Score by Accommodates (Before cleaning)
Frequency
0
10000
20000
30000
40000
YES
NO
1
2
3
4
5
6
7
8
9
10
12
14
16
19
Accommodates

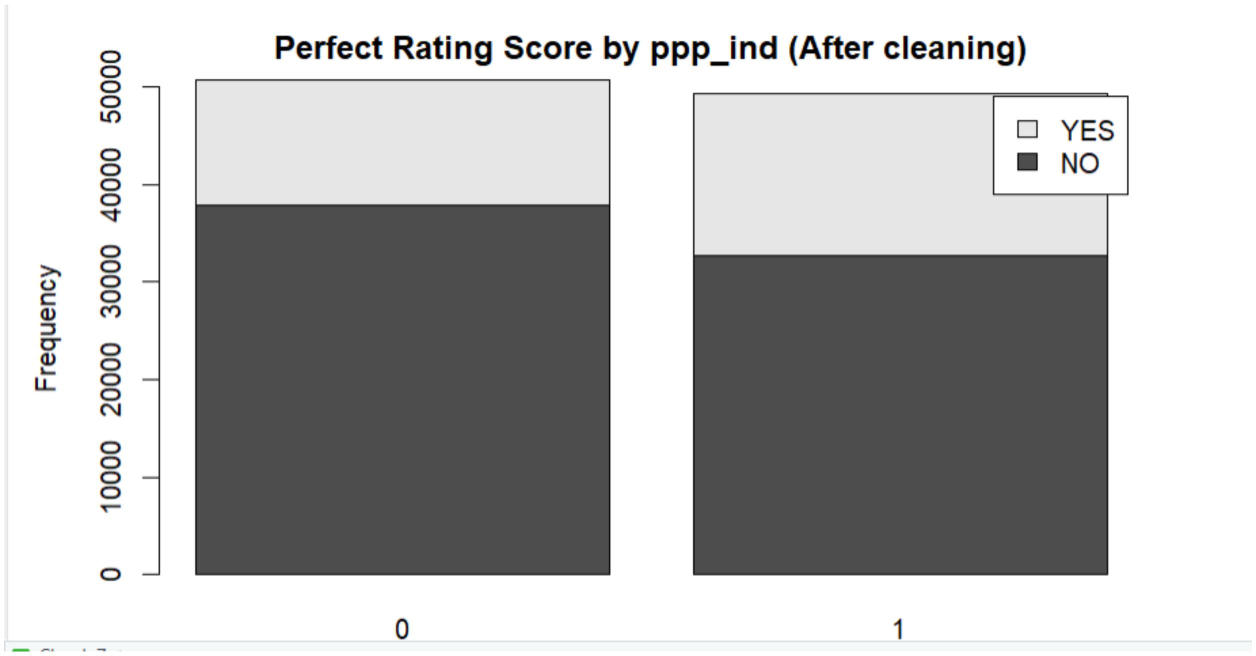
The chart below depicts the distribution of accommodates after cleaning process. We can observe that the accommodates variable has a better distribution across YES and NO categories, unlike the previous impure distribution:



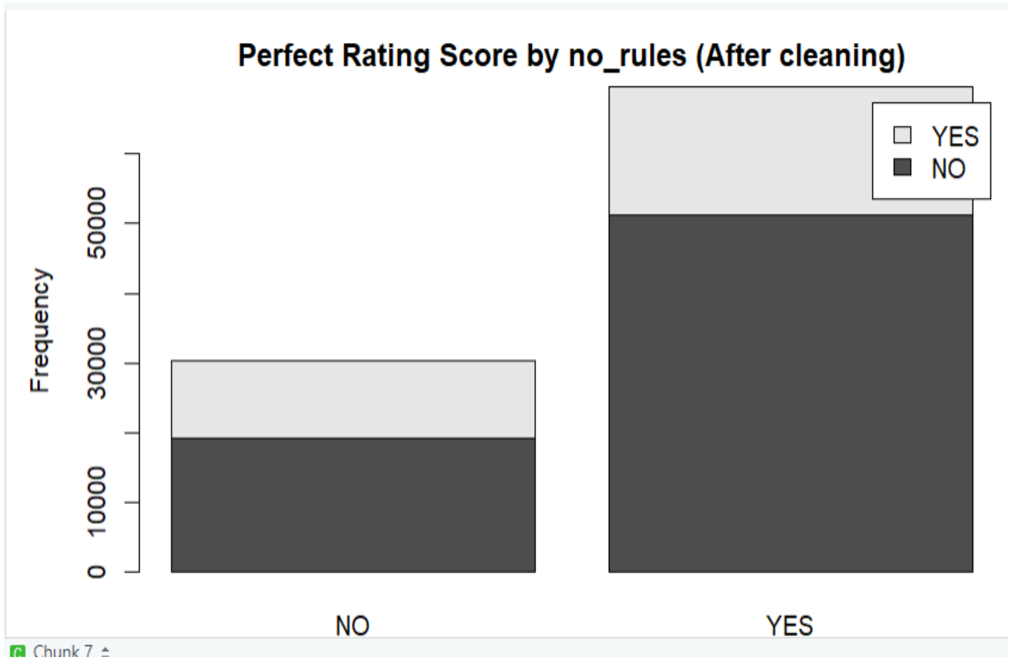
2. **Bedrooms -** This is the original feature from the dataset. However, we limited the number of bedrooms to just 6, as the distribution in this case is more purely spread across YES and NO variables compared to the uncleaned (original) version.



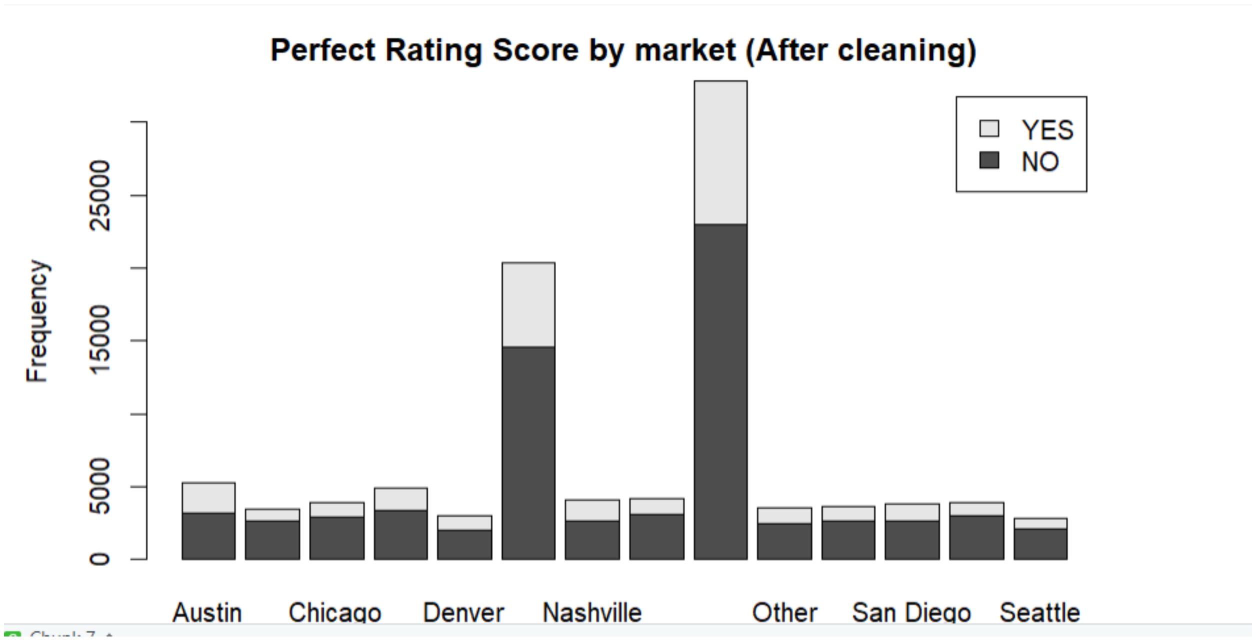
3. **ppp_ind** - It's a new feature added. It's 1 if the price per person (price/accommodates) is greater than the median value, otherwise it's 0. It's clear that if ppp_ind is 1, we have greater number of perfect_rating_score instances classified as YES.



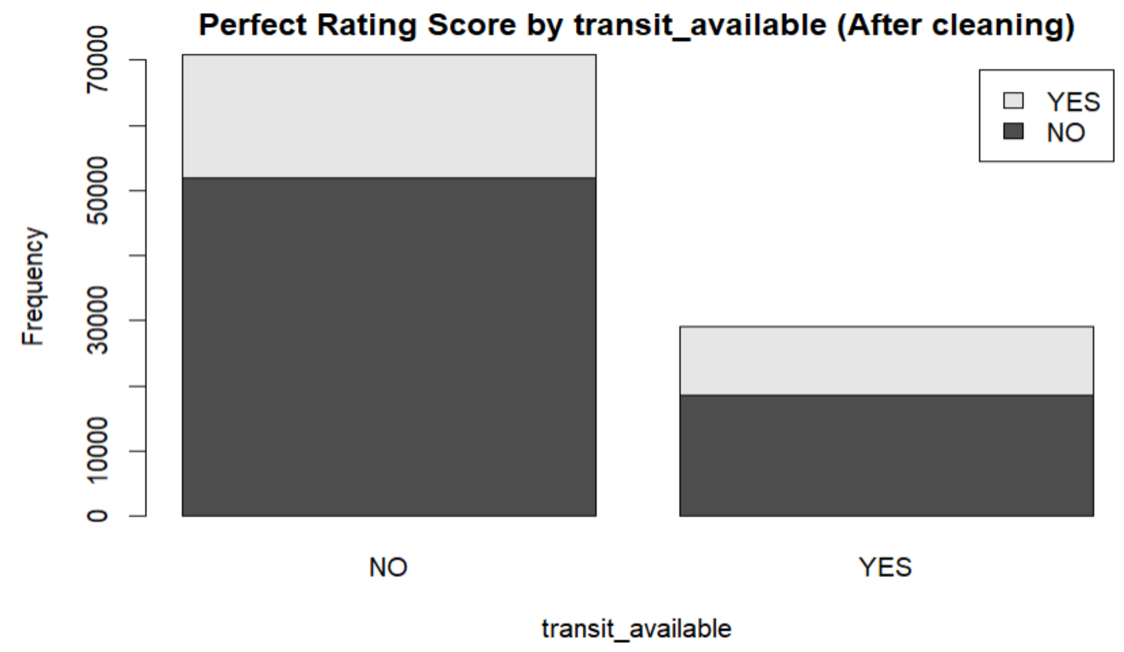
4. **No_rules** - It's an added feature, that holds a value of YES if there's no rules in the listing and NO otherwise. It's clear that the listings with no_rules as YES have a greater number of perfect rating scores.



5. **Market -** Market feature has a special significance pertaining to the biased dataset. That is, if we have any one market show up in a slightly more number of instances than other markets in the training dataset of nearly 100,000 instances; it provides biased/wrong results when tested on nearly 12,000 testing instances. Therefore we hard-coded the markets into following categories: "Los Angeles","New Orleans","Boston", "Chicago", "Denver", "San Diego","San Francisco", "Nashville","Portland", "Seattle","Other".

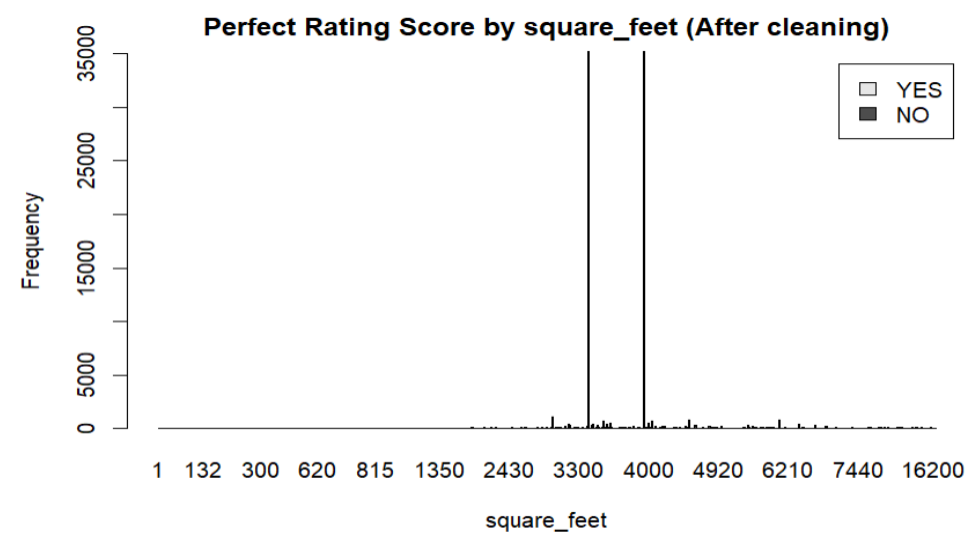


6. **Transit_available** - This is a modified feature from the original dataset that takes the value of YES if there’s a transit available near the listing’s location and NO otherwise. As expected, there’s a higher number of NO ratings for the listings if there’s no transit available nearby.

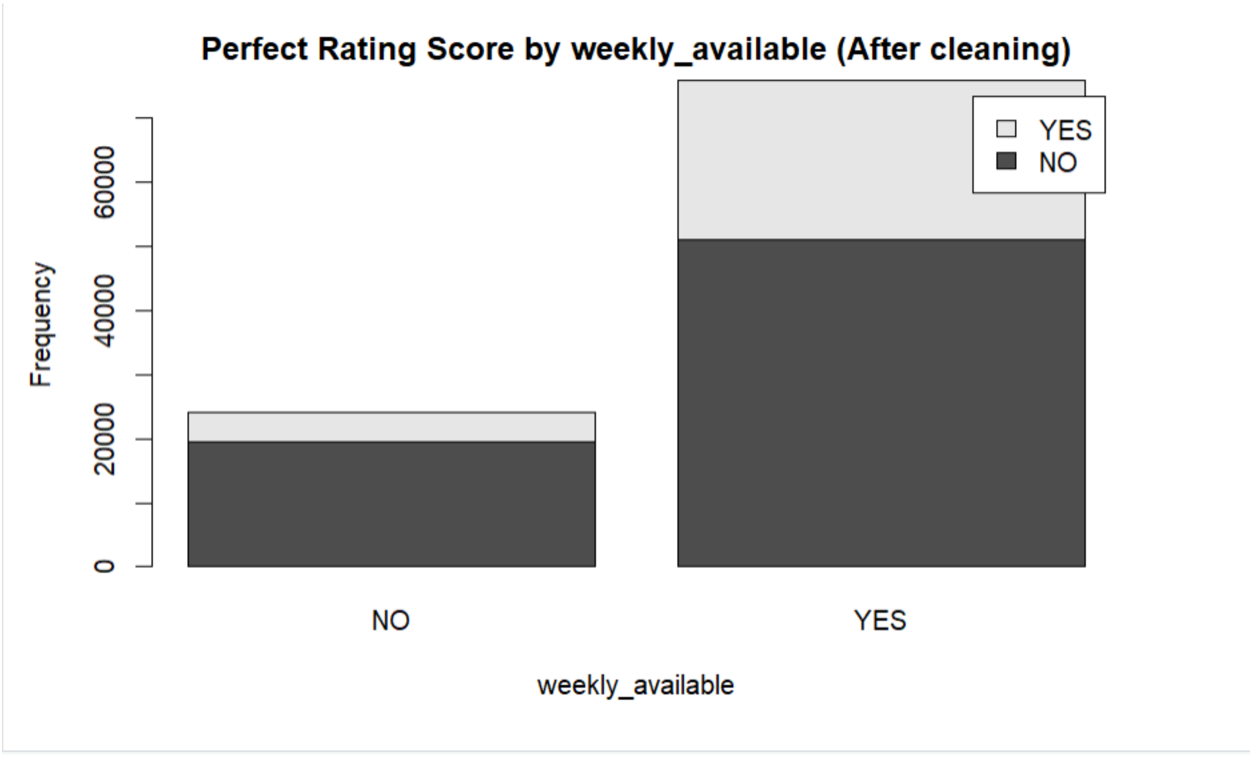


7. **Square_feet** **(Taken from **External Data** -
https://www.kaggle.com/datasets/yasserh/housing-prices-dataset?resource=download) -

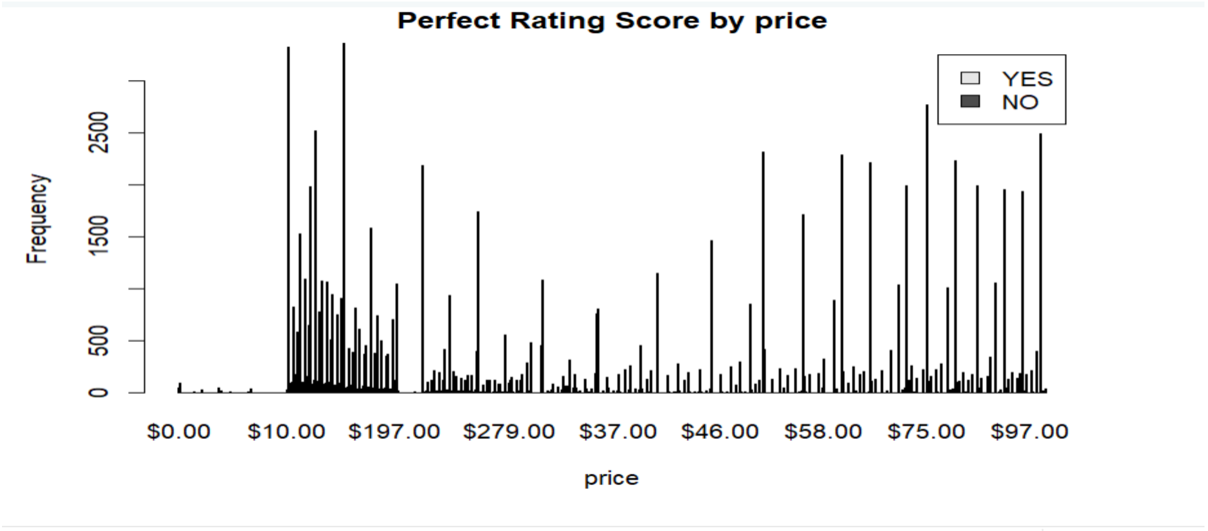
Since 90% of the instances in the training model has NAs for the square_ft variable, we import external dataset and add a new feature square_feet to the current dataset, relatively based on the no. of bedrooms and their corresponding area in sq. ft.



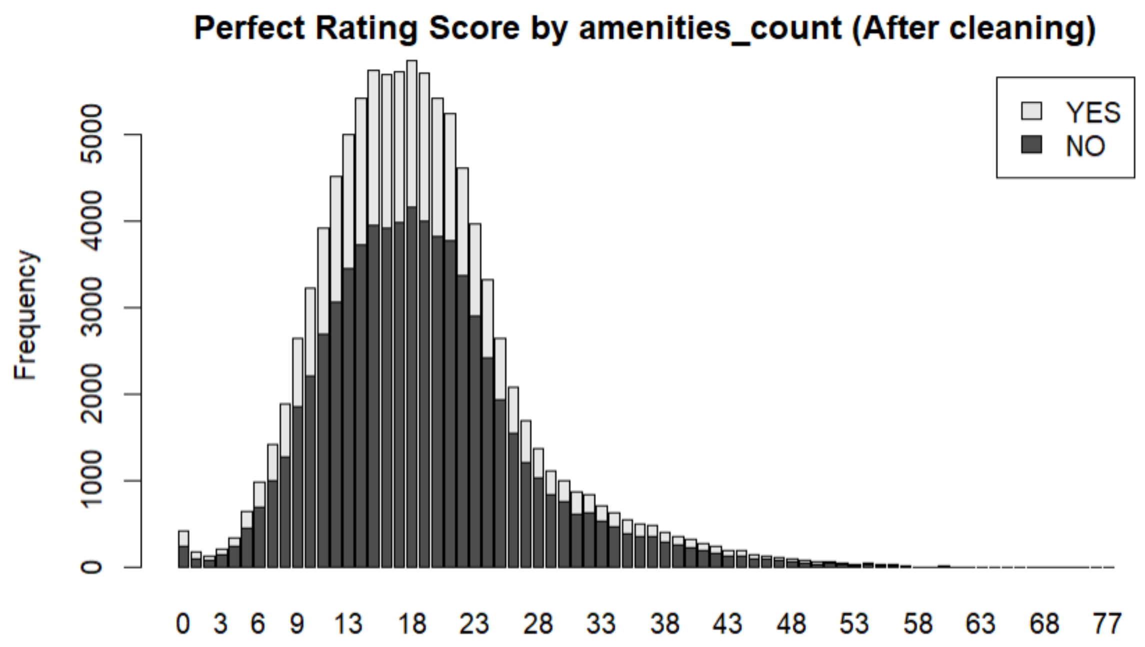
8. **Weekly_available:** Modified feature that's YES if a listing is rented out on a weekly basis and NO otherwise. Its clear that better ratings were given to the former case.



9. **Price** - Original feature from the dataset.



10. **Amenities (dummy variable)** - Amenities has a lot of unstructured data in a single column, therefore cleaning it and converting it into dummy variables provides better control on the individual categories. Following is a chart for the no. of amenities.



3) (optional) Any additional comments about the data or the steps you undertook for data preparation.

As a note, one of the features that we augmented from the external dataset is 'square_feet', which is basically the overall area of the given listing in sq. ft. However, in the given dataset, the area feature has around 90% missing/NA values. Therefore we have integrated an external data source to fill in the NA values in the current dataset. We have imputed the square feet area in accordance with the number of bedrooms.
Link to the external housing data source- https://www.kaggle.com/datasets/yasserh/housing-prices-dataset?resource=download

That is, for a given instance in the current dataset with an area value as NA, we have imputed the square_feet area of a random instance chosen from the external dataset having the same no. of bedrooms in that house. This feature has significantly improved the performance of our best-winning model as it is one of the significant (important) variables.

Section 4: Evaluation and Modeling

1) Winning model:

The winning model in our case is the Gradient Boosting model. The variables included in the model are:

accommodates, availability_365, bed_type, availability_30, availability_90, bathrooms, bedrooms, bedrooms, beds, cancellation_policy, cleaning_fee, extra_people, guests_included, host_listings_count, instant_bookable, ppp_ind, has_min_nights, host_is_superhost, host_identity_verified, has_extra_fee, host_response, room_type, is_location_exact, instant_bookable, is_available, no_rules, market, maximum_nights, minimum_nights, monthly_available, price, require_guest_phone_verification, require_guest_profile_picture, requires_license, weekly_available, TV, X.Cable.TV., Internet, X.Wireless.Internet., X.Air.conditioning., Kitchen, X.Free.parking.on.premises., Heating, X.Family.kid.friendly., X.Family.Kid.Friendly., Washer, X.Carbon.Monoxide.Detector., Essentials, Shampoo, Hangers, X.Hair.dryer., Iron, X.Laptop.friendly.workspace., X.Self.Check.In., Keypad, X.Safety.card., X.Hot.water., X.Cooking.basics., X.Lock.on.bedroom.door., Gym, X.Pets.allowed.,transit_available, square_feet.

Based on the relative generalization performance of different models on the validation dataset, the boosting model significantly outperformed the other models with a high accuracy of 0.757 or approximately 75%, TPR of 0.376~ 37% , and FPR of 0.084~ 8%. In addition, it also significantly improved the baseline model performance having a 71% accuracy. Therefore, we went ahead and decided that this was our winning model.

Line numbers in R code:
786-800

2) All models:

1. Logistic Regression:

The first type of model used is Logistic regression. Logistic regression is performed using glm( ) function. It does not need to be loaded from any external libraries; it is a built-in module.
Estimated training performance & generalization performance is as below:

Training performance:
Training Accuracy for the logistic model: 0.7298174
Training TPR: 0.28310524521502
Training FPR: 0.0847066912017471

Generalization performance:
Validation Accuracy for the logistic model: 0.7299217
Validation TPR: 0.285039901090255
Validation FPR: 0.0824722722532941

The generalization performance was estimated using a simple train/validation split. And we used all of the dataset's instances in training the model, inspite of the skewness in the perfect_rating_score variable that has almost 70% "NO"s in the training set.
The best performing set of features in the model include:

availability_365, availability_30, bathrooms, bedrooms, beds, cancellation_policymoderate, cancellation_policystrict, cleaning_fee, extra_people, guests_included, instant_bookableTRUE, ppp_ind1, has_min_nightsMIN, has_min_nightsMO, host_is_superhostTRUE, marketSeattle, price and Internet among others that have a significantly low P-value (indicated by '***' in the model summary output) , implying they're significant predictors in the model.

The line numbers in the R code where we trained the model: 266 to 306.

2. Ridge-Lasso model for logistic regression:

The second type of model we used is the Ridge-Lasso model that is used to compensate for the large values of coefficients obtained using the first logistic regression model used.
Ridge and lasso models can be implemented by incorporating a new argument while training the logistic regression model. If we set the argument 'alpha' to 0, it acts the ridge model. And if we set 'alpha' to 1, it acts as Lasso.

Estimated training performance & generalization performance is as below:
For Ridge:
Training performance:
Training Accuracy for the Ridge model: 0.7297602
Training TPR for ridge model: 0.273705741976331
Training FPR for ridge model: 0.080884880593695

Generalization performance:
Generalization Accuracy for the Ridge model: 0.7295883

Generalization TPR for Ridge model: 0.276385298415196
Generalization FPR for Ridge model: 0.0792966157929662

For Lasso:
Training performance:
Training Accuracy for the Lasso model: 0.7297602
Training TPR for Lasso model: 0.27243948765402
Training FPR for Lasso model: 0.080359128869836

Generalization performance:
Generalization Accuracy for the Lasso model: 0.7296883
Generalization TPR for Lasso model: 0.276160503540519
Generalization FPR for the Lasso model: 0.079059626504882

The generalization performance was estimated using a simple train/validation split. And again, we utilized entire training dataset in the model training process.

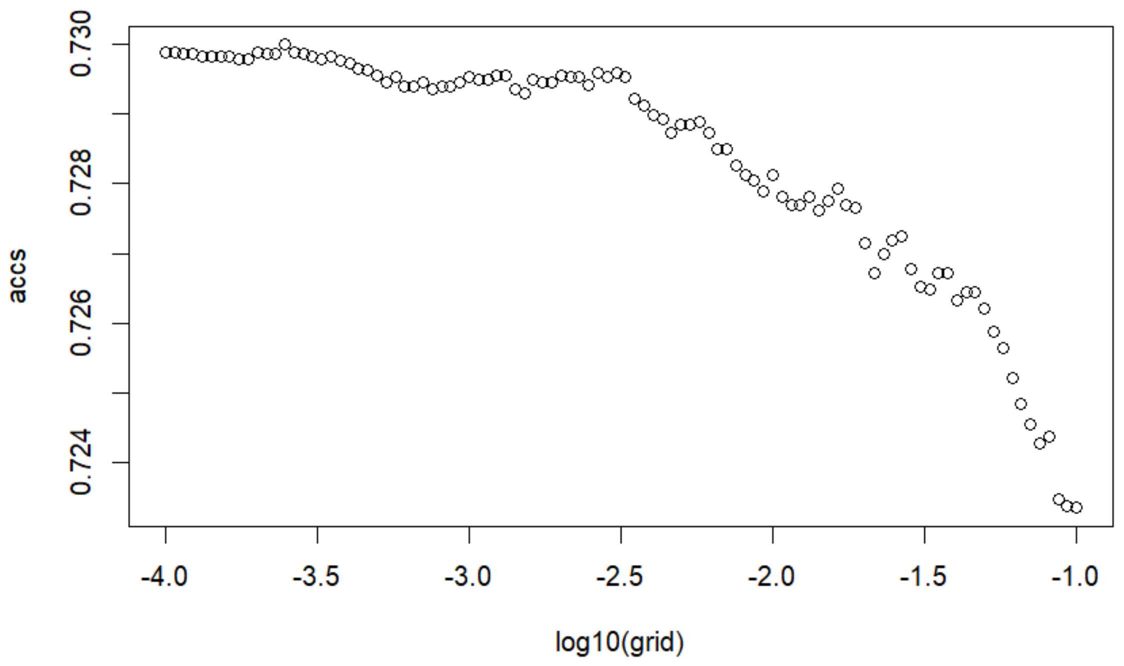
The best performing set of features include:

Bathrooms, X.Air.conditioning., X.Free.parking.on.premises., Essentials, X.Laptop.friendly.workspace., X.Self.Check.In. and transit_available.YES (dummy). These features have a higher coefficient values compared to others and therefore hold a higher significance. They in fact also translate into the real-world requirements of customers willing to provide a perfect rating score.

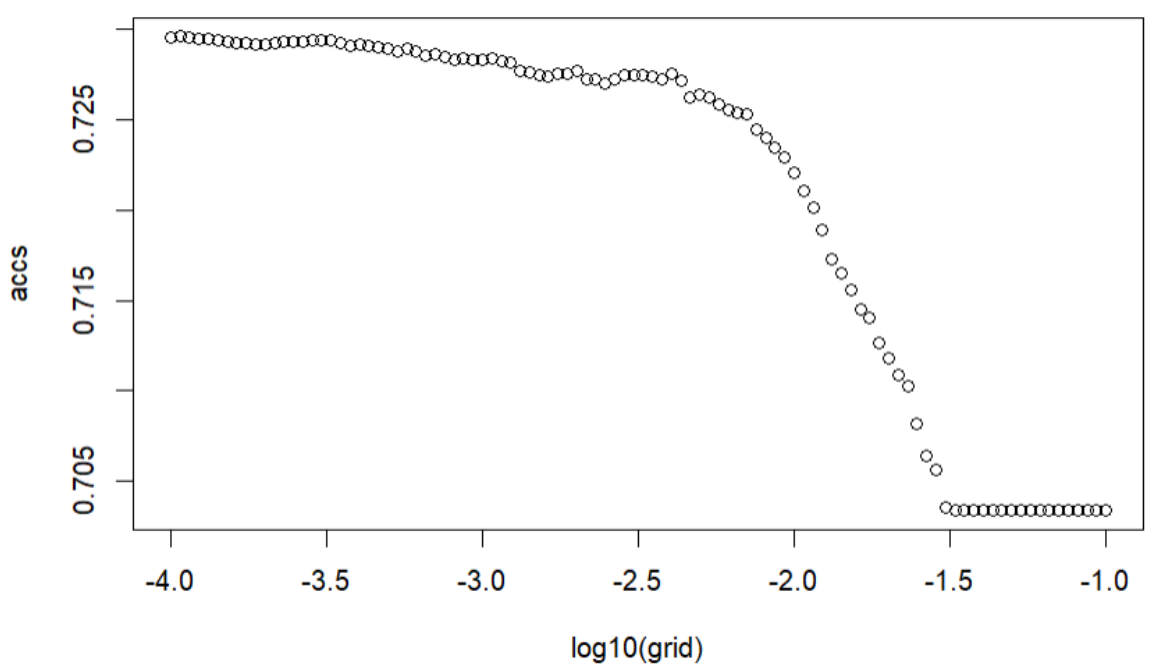
The line numbers in the R code where we trained these models:
Ridge model: 347 to 427
Lasso model: 438 to 518

The hyperparameters used in these 2 models were a set of 100 values, ranging from 0.0001 to 0.1. These values are the various lambda values that are inputted into the Ridge/Lasso models to find out the best lambda for which relatively maximum accuracy can be achieved.

Fitting curve for Ridge model:



Fitting curve for Lasso model:



3. Decision Trees

The third type of model we used is the Decision Trees. The library we need to import is 'trees' in order to use the tree( ) function.
The estimated training performance & generalization performance is as below:

Training performance:
Training Accuracy for the Decision Tree model: 0.7129712
Training TPR: 0.252325524765012
Training FPR: 0.0957676986229349

Generalization performance:
Validation Accuracy for the Decision Tree model: 0.7091849
Validation TPR: 0.248285939080589
Validation FPR: 0.0964546402502607

The generalization performance was estimated using a simple train/validation split. And again, we utilized entire training dataset in the model training process.

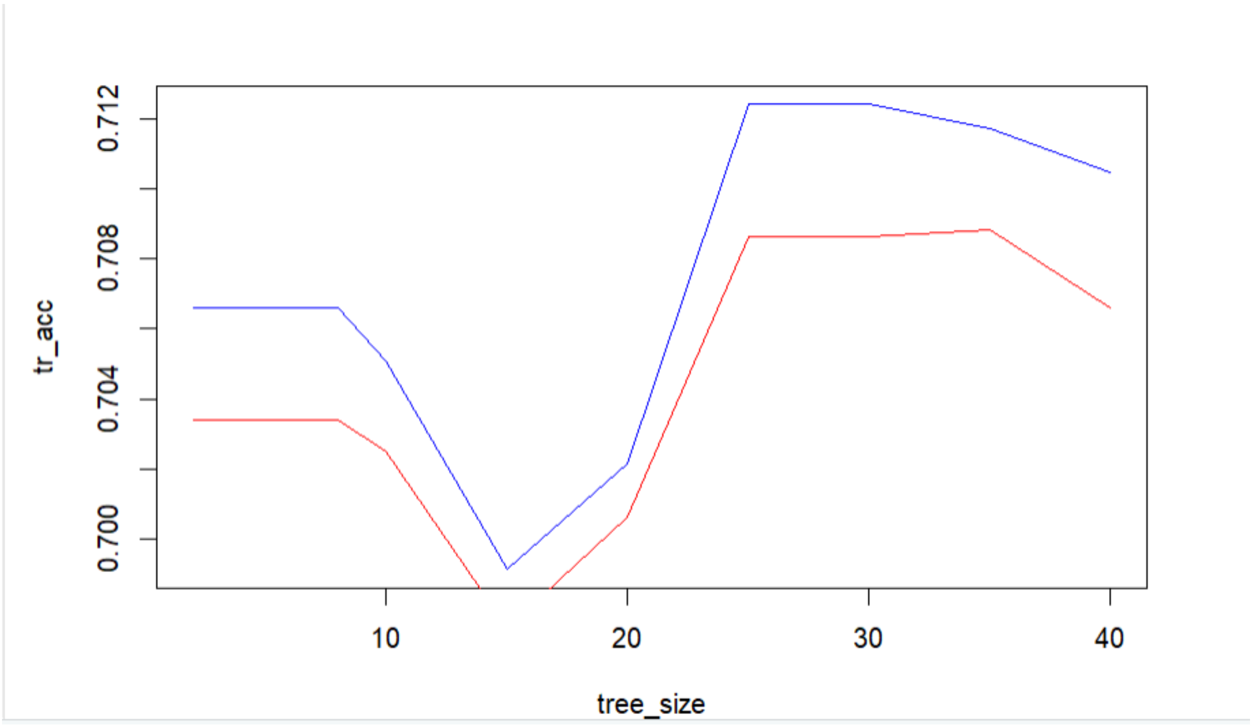
The best performing set of features include:
"availability_365", "Internet", "host_is_superhostTRUE", "host_listings_count", "price", "host_response.MISSING", "monthly_available.YES", "instant_bookableTRUE", "availability_90", "availability_30", "minimum_nights", "Washer", "cancellation_policy.strict", "extra_people", "X.Wireless.Internet.", "ppp_ind.1", "transit_available.YES" as these features have been chosen as the nodes in the resulting decision tree and therefore they would have led to a higher reduction in impurity of the overall tree when selected.

In fact, it also translates into the real-world requirements of customers willing to provide a perfect rating score.

The line numbers in the R code where we trained these models:

638 to 675

Fitting curve for decision tree: The hyperparameter tree_size is plotted against the training accuracy.



4. Gradient Boosting Model (Winning model):

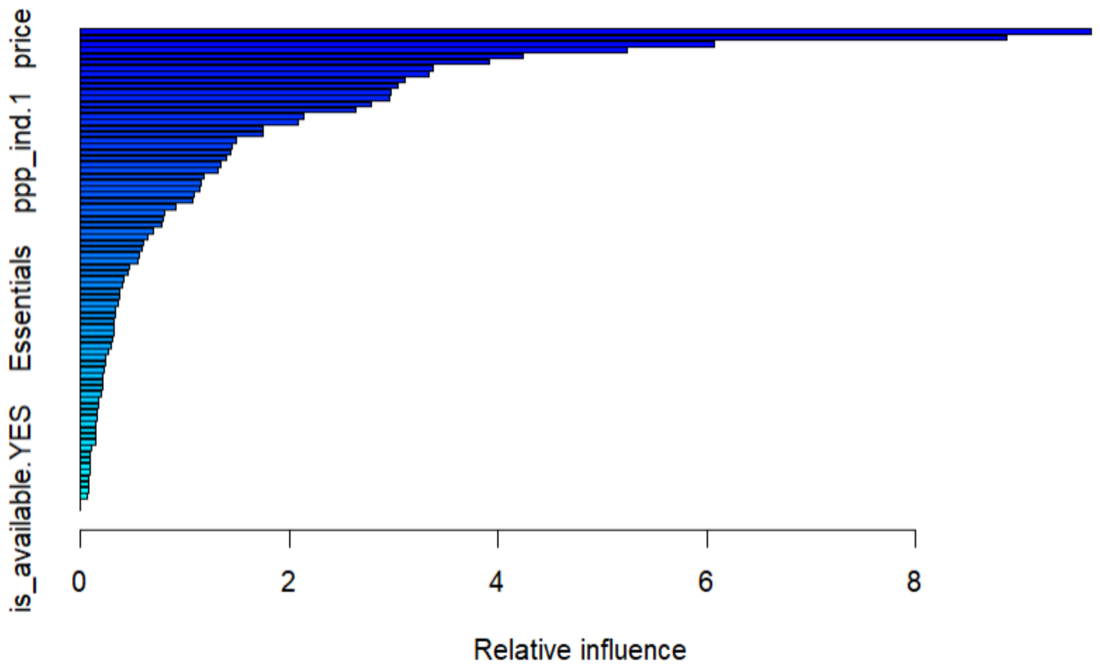
The fourth type of model we used is the Gradient Boosting model which belongs to the family of ensemble methods in machine learning. The library we need to import is 'gbm' in order to use the gbm( ) function.
The estimated training performance & generalization performance is as below:

Training performance:
Training Accuracy for the Boosting Model: 0.7576515
Training TPR for Boosting model: 0.376710660887352
 Training FPR for Boosting model: 0.0841809394778881

Generalization performance:
Validation Accuracy for the Boosting Model: 0.7385898
Validation TPR for the Boosting Model: 0.346521299314376
Validation FPR for the Boosting Model: 0.096075457389326

The generalization performance was estimated using a simple train/validation split. And again, we utilized entire training dataset in the model training process.

The best performing set of features include:
Availability_365, price, host_listings_count, Internet, host_is_superhostTRUE, minimum_nights, weekly_available.YES, cleaning_fee, availability_30, maximum_nights. There are the top 10 variables that gradient boosting model has considered to have relatively higher importance. The following graph also depicts the same:



The line numbers in the R code where we trained these models:

722 to 778

5. Random Forest:

The last type of model we used is the Random Forest model which belongs to the family of ensemble methods in machine learning. The library we need to import is ‘randomForest’ in order to use the randomForest( ) function.
The estimated training performance & generalization performance is as below:

Training performance:
Training accuracy for the Random Forest model: 0.9998428
Training TPR for the random forest model: 0.999512979106804
Training FPR for the random forest model: 2.0221220148442e-05

Generalization performance:
Validation Accuracy for the Random Forest model: 0.7311219
Validation TPR for the Random forest model : 0.313813645048893
Validation FPR for the Random forest model: 0.092899800928998

The generalization performance was estimated using a simple train/validation split. And again, we utilized entire training dataset in the model training process.

The best performing set of features include:
Availability_365, price, market, cleaning_fee, availability_90, square_feet, maximum_nights, availability_30 and extra_people are the top 10 variables that gradient boosting model has considered to have relatively higher importance.

The line numbers in the R code where we trained these models:

523 to 573

Section 5: Reflection/takeaways

1) *What did your group do well?*

Our group managed to get good accuracy and TPR values of about 73% and 34% respectively through Gradient Boosting model, although owing to a risky FPR value of about 9.6%, we were unable to make it to the top ranks in the contest.
However, we were able to successfully integrate an external data source (https://www.kaggle.com/datasets/yasserh/housing-prices-dataset?resource=download) into the existing data set and improve the performance significantly.

2) *What were the main challenges?*

One of the challenging parts was to work with the given dataset to increase the TPR while maintaining a low FPR, despite the dataset having around 70% of ‘NO’ perfect_rating_scores. We achieved high TPR value of around 43% using Random Forest model but it had an FPR>10%. Therefore due to the tight time constraints, we went ahead with submitting the predictions made using Logistic Regression that had a safe FPR value.

In addition, working with large dataset like this opened gates to many error creeping in and also being very time consuming to resolve. As running models like Random Forest and Knn on large datasets takes a lot of runtime, running them multiple times to evaluate the model was a daunting task. Thereby, starting as early as possible and reaching the tuning parameters stage early on would have been a better way of doing it again.

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

Firstly, our group didn’t have a synchronous workflow at the beginning. We initially had some brainstorming sessions to lay out all the big-picture ideas of how to go about preparing the data cleaning function. Later we decided to work on the function individually till we reach a good number of features cleaned in the dataset; and later planned to combine them and then work together.

However it turned out that this divide and conquer approach was not so effective; working in teams right from the beginning stages would have saved a lot of time and re-work of the same tasks. Later when we switched to working together on the same tasks or deliverable simultaneously, we saw it was very much efficient although it takes a little effort to ensure all the team members find a time to hop on a call.

If we could start the project over again, we would approach the predictive modeling process in an iterative manner of building the model; which is analogous to the AGILE approach that most real-world firms tackle their problems. To elaborate, we would start working on various parts of the data modeling process starting from data cleaning & preparation, data modeling and data evaluation to coming up with the ‘best model’ in a cyclical manner, instead of completing one level at a time until perfection.

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

If we had a few more months, we would have tried out the approaches we didn’t try yet. Firstly, that includes incorporating text mining into the feature engineering process. Furthermore, performing sentiment analysis on the descriptive variables for each listing like ‘Note’, will help in gaining further insight into zeroing down to best/ most important features. Later we would also leverage more features from other external data sources to account for the many missing/NA values in the dataset.

Not to mention that we would also put effort in coming up with a couple of ensemble models by trying out different combinations of models. Later, we would try out Deep Learning approaches to leverage the power of neural networks that perform well in classification tasks.

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

For a group starting this project next year, I would suggest them to start working on the project as soon as it’s posted on canvas. That way they can reach the stage of model evaluation and hyperparameter tuning earlier and later work on improving the generalization performance. In addition, it would also be beneficial to start working on the final report simultaneously. Not because it’s time consuming, but because working on the report can in fact help in making the team aware and mindful of where they currently are in their overall prediction journey. Documenting various milestones achieved (including small) in the form of report will act as a great guide in reaching the desired performance objectives on time.