

Data Science Case Study



Author: Vinayak (Vin) Kannan

Vanguard[®]




Agenda

1. **Dataset Overview:** Airbnb listing information
2. **Problem:** How can we help hosts with high ratings, yet few visitors, receive more traffic?
3. **Approach:** Model to predict Average Number of Monthly Ratings
 1. Data Cleaning
 2. Modeling
 3. Conclusions
4. **Next Steps:** Synthesis and proposed actions to improve analysis

Dataset Overview

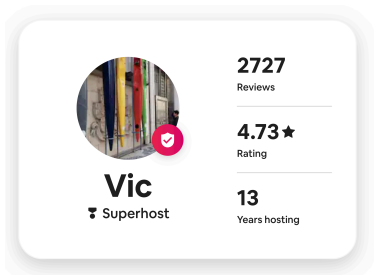
Dataset Overview: Bay Area Airbnb Listing dataset provides relevant information about listing / host / geography, and will be focus of analysis

Dataset consists of three sources (listings, neighborhoods, reviews)

Sources	 Listings	 Neighborhoods	 Reviews
Description	Scraped Airbnb listing information (e.g., Name, Price, Description, etc.) on a per-property basis	Metadata describing neighborhoods to neighborhood groups in Bay Area	Limited information on a per-review basis (related property, date of review)
Key Attributes	<ul style="list-style-type: none">• 7221 rows representing unique listings• Features consist of the following:<ul style="list-style-type: none">• Listing information (free text descriptions, property flags, included amenities, pricing, reviews)• Host information (verification, profile picture)• Geography (neighborhood)	<ul style="list-style-type: none">• 16 rows detailing information about neighborhoods and their associated groupings• Same information is represented in Listings dataset	<ul style="list-style-type: none">• 213K rows of reviews• Features include date of the review and the related listing ID• Same information is largely represented by review information present in Listings dataset

Problem to Solve

Problem: Great hosts, with highly rated listings, are not getting traffic



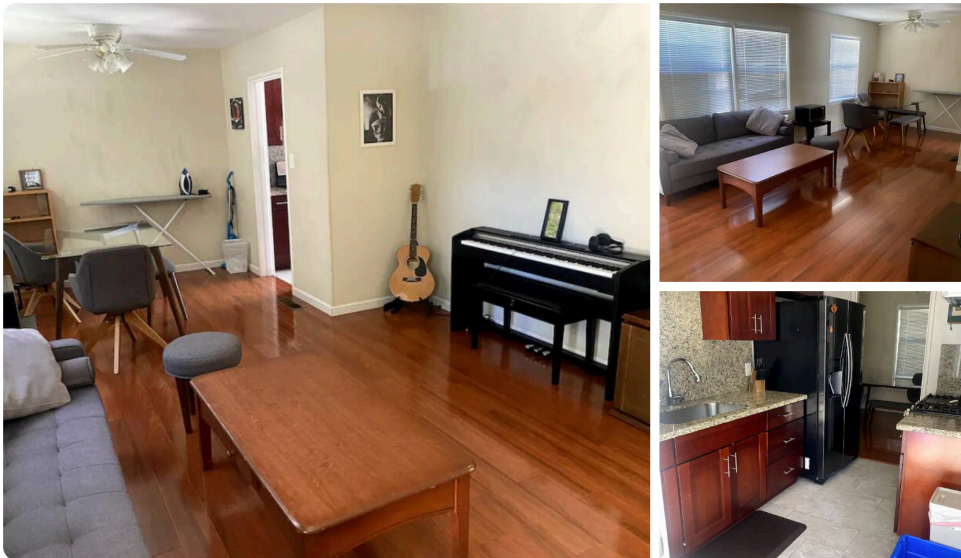
About Vic

Vic is a seasoned Airbnb super host with ~50 listings; reviews mention him as 'kind', 'hospitable', and 'caring'. He is the prototypical Airbnb success story

Despite this, not all his listings do well

HackerHome - Downtown Sunnyvale: private room

★ 5.0 · 19 reviews · Superhost · Sunnyvale, California, United States



Vic's HackerHome has high ratings yet below avg. traffic

HackerHome has perfect rating (5.0) across several reviews

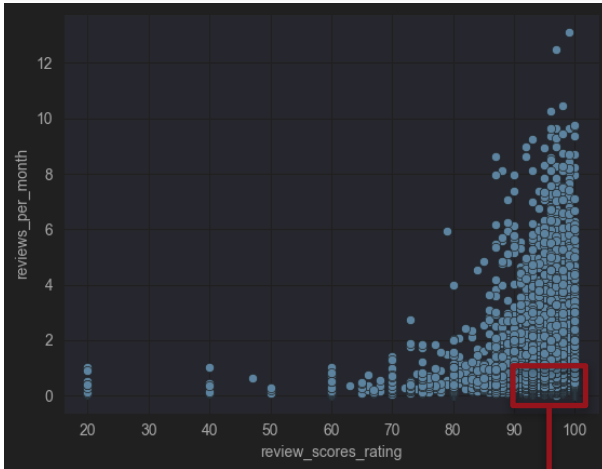
Listing is in the bottom 30% of listings by traffic ('reviews_per_month', proxy for average number of visits per month to the relevant Airbnb listing)

Vic needs help understanding two things:

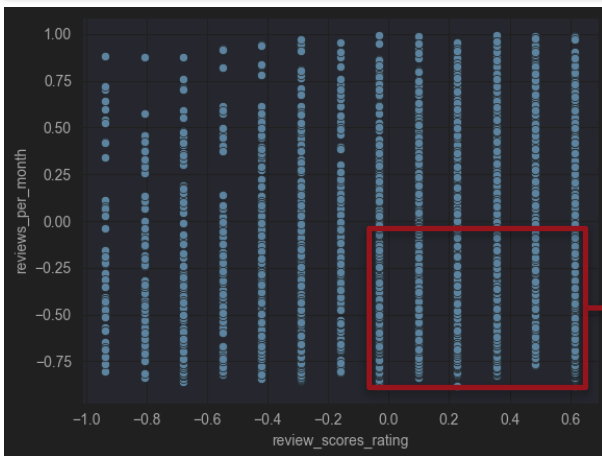
- 1) Why his listing performing poorly? What can he realistically change about this listing to improve its traffic (e.g., moving the listing to a new neighborhood isn't realistic)?
- 2) If he were to open future listings in the future, could he predict their traffic performance?

Problem: ~33% listings face Vic's problem

Reviews vs Avg. Rating



Normalized View



33% of listings

Approach

- Filtered Listing dataset's columns to 'reviews_per_month' and 'review_scores_rating'
- 'reviews_per_month' is a proxy for average number of visits to the relevant Airbnb listing; 'review_scores_rating' is a proxy for visitor satisfaction
- Scatterplot created representing this relationship, both using data as-is and also normalized in order to have both features on the same 'scale'
- Expectation is linear curve, where popular listings receive more visits, in the form of higher 'reviews_per_month'

Takeaways

- Initial view (top left) shows this largely holds; however, there are a set of listings with high ratings yet low review counts
- Normalized view (bottom left) confirm this. Quadrant of listings with below average review counts yet above review scores exists, breaking from expectation
- Magnitude of problem is significant, as ~33% of listings fall in this unexpected quadrant

A predictive, explainable model could help hosts like Vic with underperforming listings reposition their assets on Airbnb

Problems to solve



A host, after creating a new listing, wants to know how it will perform



A host, with an existing listing, wants to know what factors (that they can control) drives traffic

Model capabilities

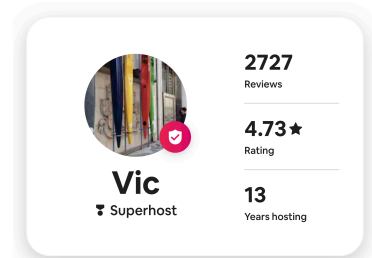
1.

Model will be trained to predict 'reviews_per_month', as a proxy for traffic

2.

Interpretable models (e.g., regression, decision trees) will be used to provide some insight into factor importance

How model will help



Airbnb hosts, like Vic, will be able to make informed decisions about how to position their current and future listings

Approach

Approach - Data Cleaning: Cleaning primarily consists of relevant feature selection, cleaning, and converting raw text fields to PCA embeddings

Data Cleaning step (not exhaustive)

Why step was performed?

Feature Selection

Removed features which a Airbnb listing owner likely cannot control (e.g., neighborhood); model predicting that a listing owner change neighborhoods likely isn't helpful to end user

Filtered out features with high sparsity (e.g., square_feet) with no clear way to impute

Removing features will improve future model performance by reducing computational cost and focusing model on features user is interested in

Dimensionality reduction will make dataset easier to work with; additional techniques such as Lasso Regression are applied later to further reduce dimensionality

Cleaning / Imputation

Ran one hot encoding on categorical features

Simplified unstructured fields (e.g., amenities) into count of amenities, to enable modeling

Imputed missing values for host related fields, like host_response_rate, by replacing values with average for associated host_id

Normalized numerical values to manage scale

One hot encoding ensures categorical features are represented in a bias-free, unranked manner

Data cleaning / imputation helps preserve some features / rows that the model would otherwise be unable to leverage

Raw Text Handling

Converted free text features (e.g., listing summary) into vector embeddings; reduced dimensionality by collapsing embeddings via PCA

Embeddings help model ingest raw text fields that are potentially important; PCA helps maintain dimensionality

Approach - Modeling: Two approaches were taken using three model types to capture baseline and measure efficacy of text features

Approach 1: Drop raw text embeddings

Approach 2: Keep raw text embeddings

Dataset

48 features by 3993 rows

603 features by 3993 rows
(Dimensionality spike caused by raw text embeddings)

Modeling Approach

Approach consistent across different datasets to facilitate comparison

Several steps were taken prior to modeling:

- 'reviews_per_month' is set as the numeric target variable to predict
- MSE is used as scoring metric
- Grid Search set up to tune alpha for lasso regression and learning rate, max depth, number of estimators, and minimum child weight for XGBoost
- 10-Fold Randomized Cross Validation created to assist with hyper parameter tuning
- 80/20 dataset split between train / holdout test; performance measured across both

3 models were selected to establish baselines for comparison within / across approaches:

1. Linear Regression
2. Lasso Regression
3. XGBoost

Why models were chosen?

Linear regression was selected as a baseline to compare other models against

XGBoost selected as a 'SOTA' alternative to Linear Regression from performance standpoint; handles collinearity and is interpretable

Lasso regression selected to assist with dimensionality reduction / potential collinearity, as Lasso regression adds a penalty which can eliminate unimportant variables

Approach - Modeling: Two approaches were taken using three model types to capture baseline and measure efficacy of text features

- Poor performance
- Moderate performance
- Strong performance

Approach 1: Drop raw text embeddings

Approach 2: Keep raw text embeddings

	R^2	MSE	R^2	MSE
Linear	~0	>10	~0	>10
Lasso	0.22	2.16	0.35	1.81
XGBoost	0.49	1.42	0.50	1.40

All metrics shown are based on running model against holdout dataset; metrics against training dataset in appendix

Conclusions

- Linear regression performs poorly and underfits compared to alternative approaches. This is likely driven by potential collinearity / complexity the model cannot capture
- Models which can reduce dimensionality, such as Lasso / XGBoost are similar in performance
- Raw text embeddings do not add any meaningful performance improvement; as such, simpler model without text embeddings is selected as the proposed model

Approach - Conclusions: Several features arose as important to predicting 'traffic' for an Airbnb listing, which hosts can keep in mind

Note: 'Feature importance' calculated based on XGBoost calculation using feature's weight, gain, and cover

'Expected' findings



Host_Acceptance_Rate (how often hosts accept visitors): If a host wants more traffic, they should try to accept as many guests as possible



Review_Scores_Rating (listing's rating): If a listing is rated well, it will likely receive more traffic



Cleaning_Fee: Having a higher cleaning fee is more likely to reduce traffic

Insightful findings



Minimum_Nights_Avg_NTM (minimum nights host requires for a stay on average): Having lower minimum nights for bookings can improve traffic



Guests_Included (how many guests are included in the listing): If a listing allows for more guests, it is likely to be more busy



Price (price of listing): Price of listing was actually less important than Cleaning_Fee; potential opportunity for hosts to lower Cleaning_Fee's in exchange for higher price

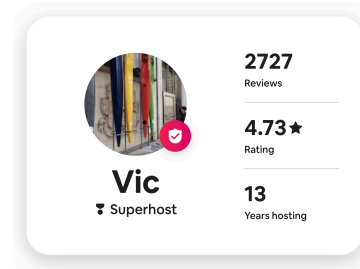
Next Steps

Next Steps: Model addresses the problems identified by helping Airbnb hosts identify means to increase their listings' traffic

Recall problems to solve ➡ How model solves problem if deployed



A host, after creating a new listing, wants to know how it will perform



Airbnb hosts, like Vic, could log into the Airbnb app and view their listings



A host, with an existing listing, wants to know what factors (that they can control) drives traffic

1.

Vic can check his current listings and see both his predicted traffic / ways he can improve his listing

2.

Vic can create a new listing, and during the creation process, receive input on how to improve his listings projected performance

Next Steps: Several steps can be taken to improve data cleansing / modeling methodology

Possible methods to improve model

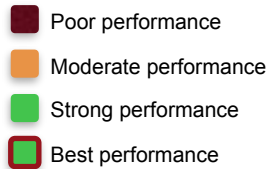


Proposed next steps

- 1.** Improve dataset
Additional relevant features could be added to the dataset to improve underlying models (e.g., revenue, revenue per guest, review content)
- 2.** Neighborhood-by-neighborhood analysis
Models could be created on a neighborhood-by-neighborhood basis, so hosts can compare their listings to relevant comparable
- 3.** Improved text vector embedding analysis
One text vector embedding was created for each free text entry; performance may be improved if free text entries are broken into 'chunks' and multiple embeddings are created for each free text entry
- 4.** Host / Listing picture embedding analysis
Host and Listing images are provided in the dataset. These images can be converted into embeddings and similarly passed into the model as an input to potentially improve model performance
- 5.** Additional dimensionality reduction to improve Linear Regression performance
PCA and rotations can be performed on the remaining numeric / boolean columns outside of text columns for linear regression. This will improve performance by handling collinearity and rotations can be interpreted to facilitate explainability
- 6.** Further unit testing
Additional unit tests can be created to ensure that code functions as expected

Appendix

Approach - Modeling: Performance metrics along training dataset



Approach 1: Drop raw text embeddings

Approach 2: Keep raw text embeddings

	R^2	MSE	R^2	MSE
Linear	0.24	1.95	0.56	1.12
Lasso	0.24	1.95	0.5	1.28
XGBoost	0.96	0.11	0.98	0.05

Proposed model

Conclusions

- Linear regression performs poorly and underfits compared to alternative approaches. This is likely driven by potential collinearity / complexity the model cannot capture
- Models which can reduce dimensionality, such as Lasso / XGBoost are similar in performance
- Raw text embeddings do not add any meaningful performance improvement; further work could be done to improve performance with raw text (to be discussed)