# **What Features Drive Demand High?**

Airbnb in New York, Fitting a Demand Surface

Final Report for D3M @ Stern by Daniel Chin | Andrew Liu | Wendi Tang

**Table of Contents**

[0. Research Question and Context 4](#_Toc39584345)

[1. Estimate Occupancy 4](#_Toc39584346)

[2. Feature Cleaning 7](#_Toc39584347)

[3. Text Mining 8](#_Toc39584348)

[4. Principal Component Analysis 11](#_Toc39584349)

[5. Hierarchical Clustering 12](#_Toc39584350)

[6. Demand Curve and Recommendations for the Host 12](#_Toc39584351)

[7. ML Models and Metrics 16](#_Toc39584352)

[8. Future Work 17](#_Toc39584353)

[9. Questions 18](#_Toc39584354)

[10. References 18](#_Toc39584355)

[11. Appendices 18](#_Toc39584356)

## Research Question and Context

In this project, we use data analysis to inform Airbnb hosts in NY what are the best features in terms of boosting demand.

The demand curve describes the relation between P (price) and Q (quantity). In the context of Airbnb, Q is translated to “occupancy rate”, i.e. the percentage of available days that end up being booked by the guests. We will fit a model to predict occupancy rate from the price and the features of a listing. The model will reveal the features that influence demand.

## Estimate Occupancy

The dataset is scraped from Airbnb public listings and hence does not contain the occupancy rate for the listings. Yet, we need the occupancy rate to calculate the sales per month for any given host. To estimate the occupancy rate, most analysts chose to use the number\_of\_reviews. For example, Inside Airbnb's "San Francisco Model" assumed a review rate of 50% [1]. In contrast, Airbnb's CEO Brian Chesky reported a review rate of 72% [2]. Their numbers diverge a lot, so we decide to calculate our own estimation from the dataset.

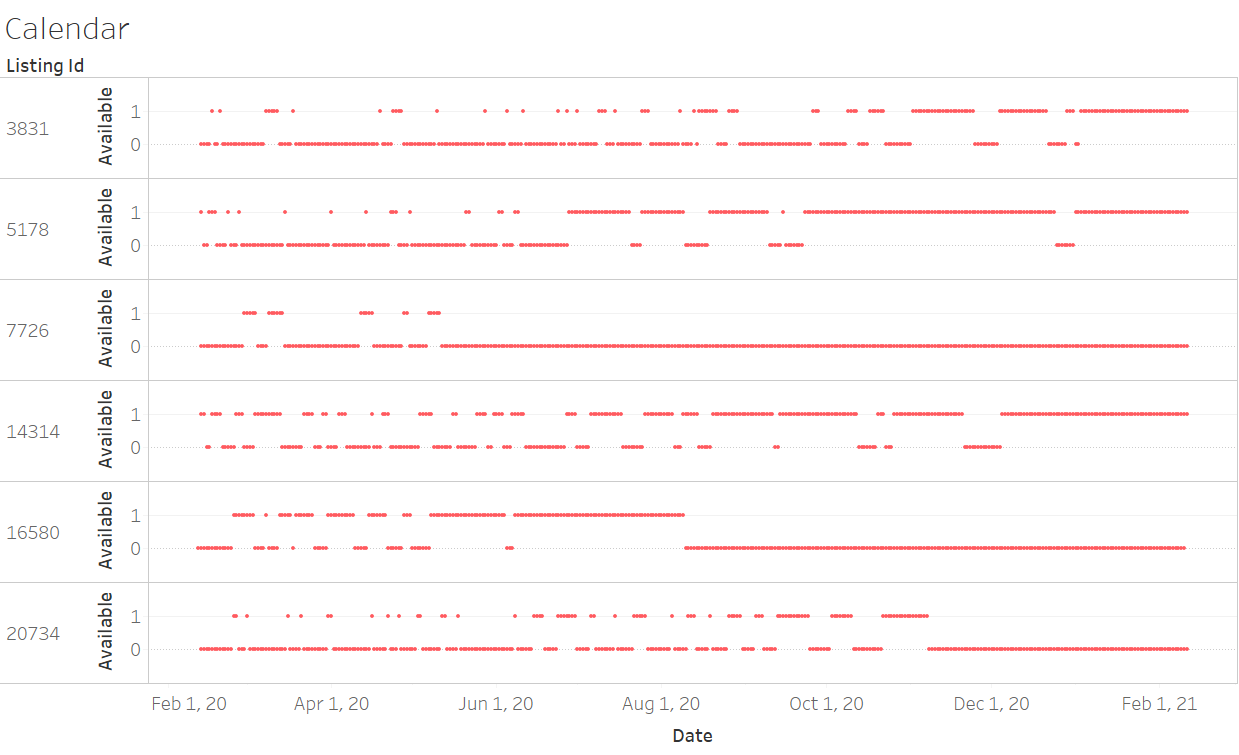
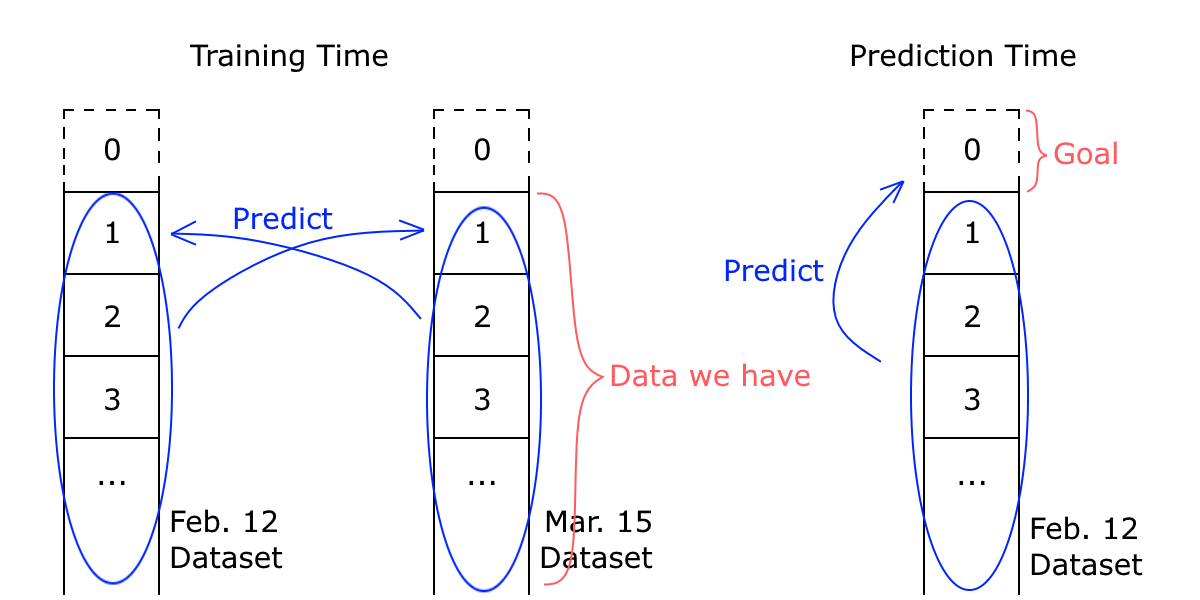
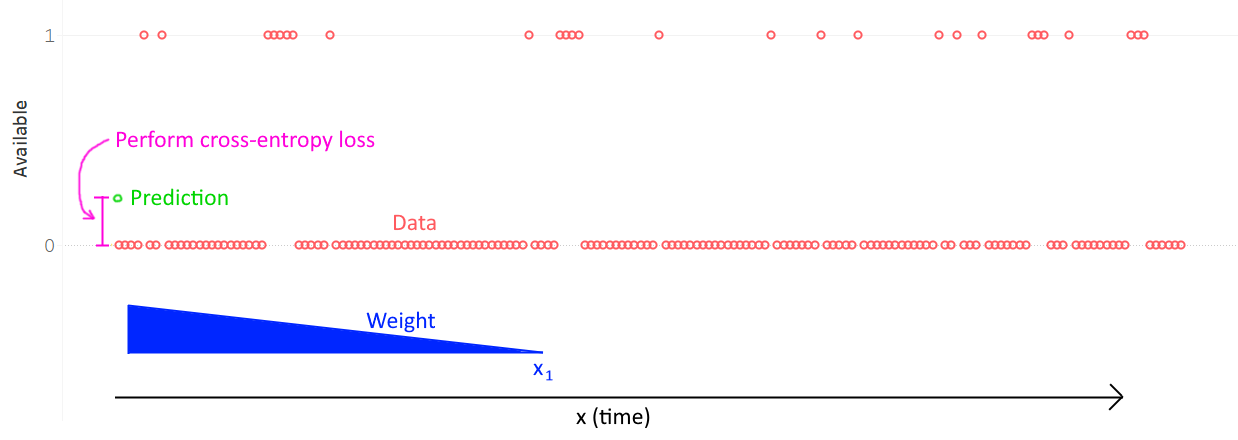
  
**Figure 1**

Figure 1 shows the availability calendar for six selected listings. It is a snapshot taken on scrape\_day (Feb. 12-13, 2020). “Available” means the day is open for booking. “Unavailable” has two possible causes: a) the day is occupied by another guest, or b) the host closed it off. Usually, the dots trend upwards (Figure 1 listing 3831, 5178, 14314), hinting that the unavailable days on the left are sales. However, the trend is not observed in listing 7726, 16580, and 20734. That is due to large chunks of days being closed off by the hosts.

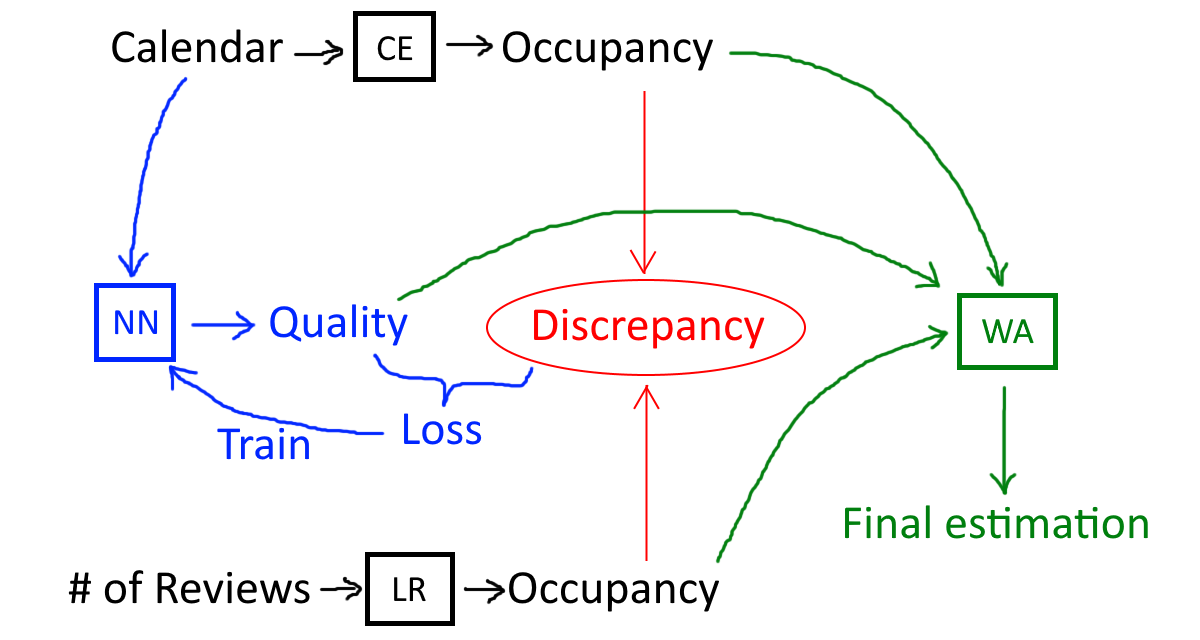
Thus, the process to estimate occupancy is clear. First, estimate probability\_unavailable for x=0. Second, estimate probability\_closed. Third, subtract them, and we will get the estimation of the occupancy rate. We use the left half (0<x<x1) to estimate probability\_unavailable for x=0. We use the right half (x2<x<end) to estimate probability\_closed.[[1]](#footnote-1)

We use machine learning to find the optimal x1 and x2. We download two instances of the dataset. For any given x1, we can try to use *one dataset* to predict *the other dataset*’s availability at x=1. The goodness of x1 is thus indicated by the cross-entropy loss. See figure 2. We find the optimal x1=9 and x2=93 (see Appendix 1).

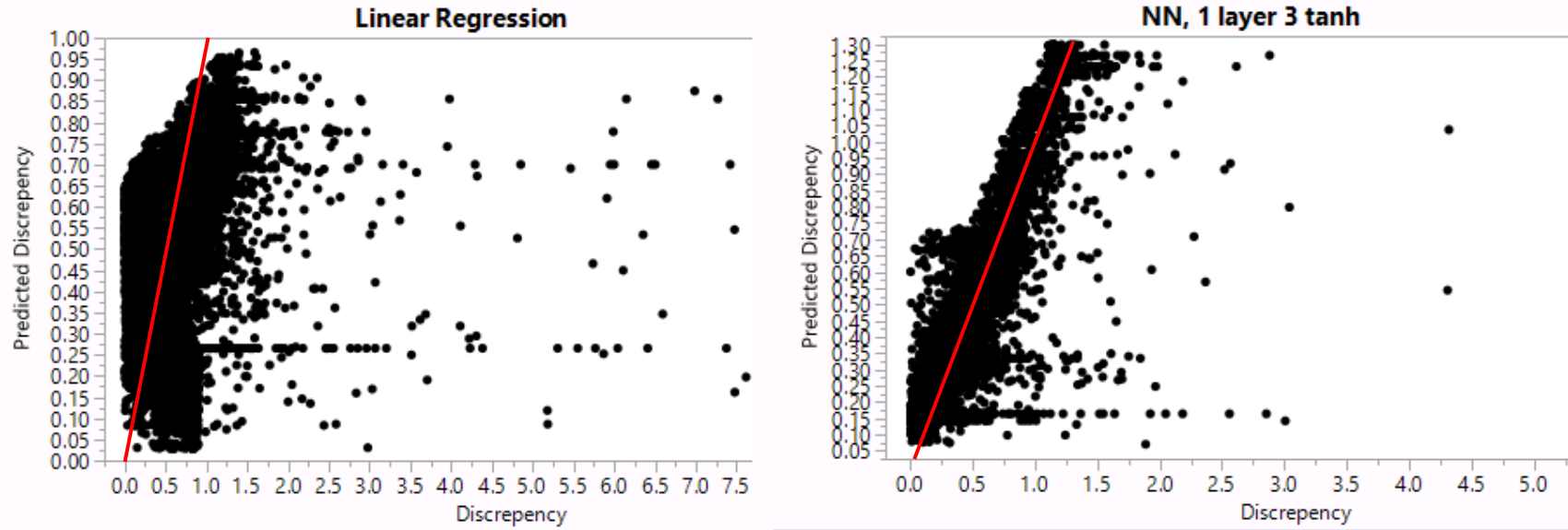
  
**Figure 2**

In this way, we calculate a rigid estimation of the occupancy rate for every listing. The next step is to bring number\_of\_reviews back into the picture. We run linear regression between number\_of\_reviews, length\_of\_stay and occupancy\_rate (from calendar), filtering on the condition that last\_review is recent enough[[2]](#footnote-2). The results show [result here]. See Appendix 2.

Now we have two estimations for occupancy\_rate, one from the calendar, and one from the reviews. Our final estimation is a weighted average of the two. Intuitively, at times when the calendar produces bad estimations, we should put a higher weight on the reviews estimation. Bad calendars are those with large chunks of closed days, and those with occupancy>0 near x=end.[[3]](#footnote-3) A neural net is trained to look at the calendar[[4]](#footnote-4) and predict the discrepancy between the calendar estimation and the review estimation. The predicted discrepancy is taken as the quality score for how bad the calendar is.[[5]](#footnote-5) See figure 3.

  
**Figure 3.**CE: Calendar Estimation; NN: Neural Net; LR: Linear Regression; WA: Weighted Average

The performance of our neural net[[6]](#footnote-6) is evaluated using a Predicted vs Actual plot (figure 4). Alongside is a comparison with a linear regression model, whose inadequacy is visible to naked eyes. The usage of a neural net is thus justified because the linear model fails to capture the non-linear surface.

  
**Figure 4. Predicted vs Actual, on validation set.**

With the predicted discrepancy[[7]](#footnote-7) as the quality score, we calculate the final estimation for occupancy rate:  
 occupancy\_rate\_final = (calendar\_weight \* occupancy\_rate\_calendar   
 + review\_weight \* occupancy\_rate\_reviews  
 ) / (calendar\_weight + review\_weight)

Where calendar\_weight = max(1 - predicted\_discrepancy, 0),   
and calendar\_weight = constrain(1.5 - 0.5 \* review\_z\_score, 0, 1)

The sum of the two weights is stored as a confidence indicator of our final occupancy estimation.

## Feature Cleaning

The dataset provides 105 different features for every listing. For us to fit a model to predict demand, we shall try to lower the number of x variables. There are two purposes of doing so: 1) lower the computation time, and 2) avoid overfitting.

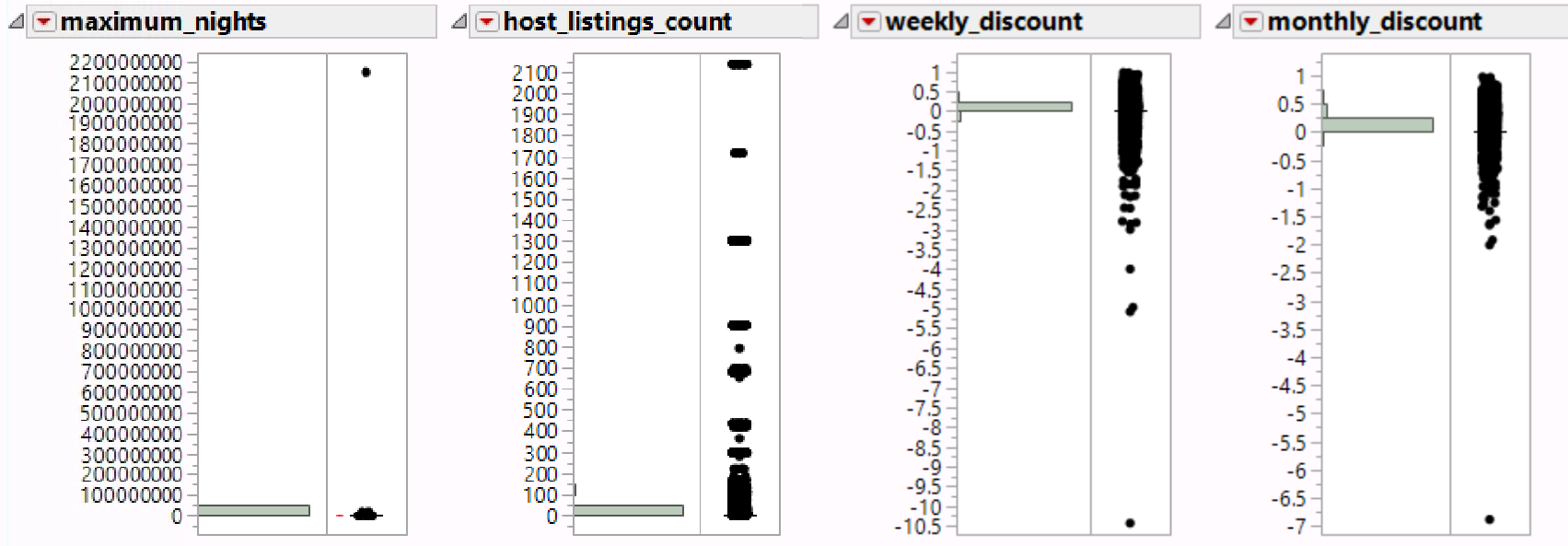
The first thing we do is to exclude 34 obviously irrelevant features: listing\_url, scrape\_id, last\_scraped, host\_id, host\_url, host\_name, state, market, country\_code, country, calendar\_updated, availability\_30, availability\_60, availability\_90, availability\_365, calendar\_last\_scraped, number\_of\_reviews, number\_of\_reviews\_ltm, first\_review, last\_review, review\_scores\_rating[[8]](#footnote-8), review\_scores\_accuracy, review\_scores\_cleanliness, review\_scores\_checkin, review\_scores\_communication, review\_scores\_location, review\_scores\_value, reviews\_per\_month, and 6 image urls. We discard notes, access because it has too many missings. This leaves us with 69 features.

Then, we remove duplicate information. We remove neighbourhood, latitude, longitude, smart\_location, is\_location\_exact, and keep neighbourhood\_cleansed and neighbourhood\_group\_cleansed. We remove minimum\_minimum\_nights, maximum\_minimum\_nights, minimum\_maximum\_nights, maximum\_maximum\_nights, minimum\_nights\_avg\_ntm, maximum\_nights\_avg\_ntm, and keep minimum\_nights, maximum\_nights. We remove requires\_license and keep license. We remove calculated\_host\_listings\_count\_entire\_homes, calculated\_host\_listings\_count\_private\_rooms, calculated\_host\_listings\_count\_shared\_rooms, and keep calculated\_host\_listings\_count. This further reduces the number of columns to 54.

After inspecting the distribution of each feature, we find that some features have extremely skewed distributions. Therefore, we further discard license, experiences\_offered, host\_neighbourhood, square\_feet, has\_availability, jurisdiction\_names, is\_business\_travel\_ready. Host\_listings\_count and host\_total\_listings\_count are always equal, so we discard the latter. Amenities need to be expanded into 30 columns (See Appendix 3). This way, we have 75 features left to deal with.

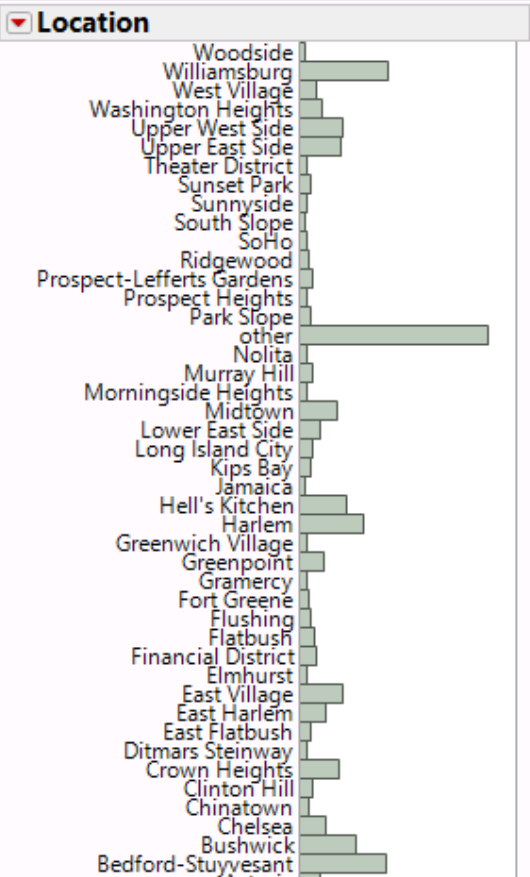
Host\_response\_rate, host\_acceptance\_rate needs additional cleaning to be parsed as numerical. Host\_verifications is converted to num\_of\_host\_verifications. Weekly\_price and monthly\_price are converted[[9]](#footnote-9) into weekly\_discount and monthly\_discount. Missing values are filled with ones for the following variables: host\_listings\_count, host\_total\_listings\_count, bedrooms, beds. Missings are filled with zeros for Security\_deposit, cleaning\_fee. Bathrooms missings are left as is. (Maybe leaving missing data could be a host strategy)

Lastly, we examine extreme outliers.[[10]](#footnote-10) Apparently one host put the largest integer that can be stored in a 32-bit computer word (2,147,483,647) as maximum\_nights:

  
**Figure 5. Strange Distributions**

We set a cap of 2000 to maximum\_nights. The distribution of host\_listings\_count also looks funny, but we decide it just needs a log transform, so we keep the outliers. Weekly\_discount and monthly\_discount, to our surprise, have negative values. Evidently, some hosts set the weekly prices such that it is actually more expensive than the per-night prices. Albeit strange, we decide that it is a valid feature of a listing. We thus set a lower cap of -1 (you pay double!) for Weekly\_discount and monthly\_discount.

The location originally has 224 levels. We group the minority locations into “other” and end up having 45 levels. See figure 6.

  
**Figure 6**

## Text Mining

The data contain many text features. We must extract quantitative features from the texts. Here is how we handle each text feature.

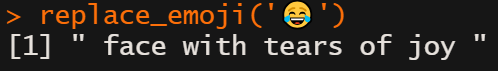
|  |  |  |  |
| --- | --- | --- | --- |
| Text Variable | Word count | Sentiment | Topics |
| name | ✔ |  |  |
| summary | ✔ | ✔ | ✔ |
| space | ✔ | ✔ | ✔ |
| description | ✔ | ✔ | ✔ |
| interaction | ✔ | ✔ | ✔ |
| host\_about | ✔ | ✔ | ✔ |
| neighborhood\_overview | ✔ |  |  |
| transit |  |  | ✔ |
| house\_rules |  |  | ✔ |

For name, we also generate is\_all\_cap and average\_word\_length.

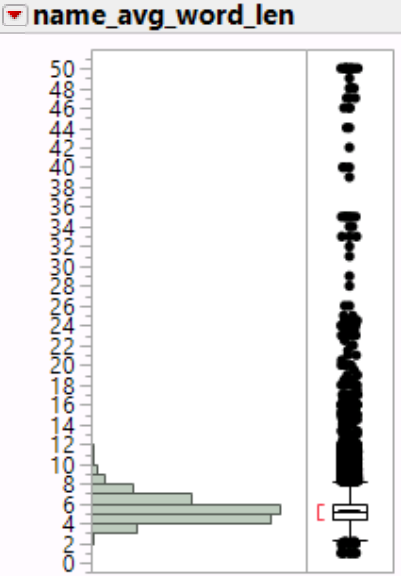
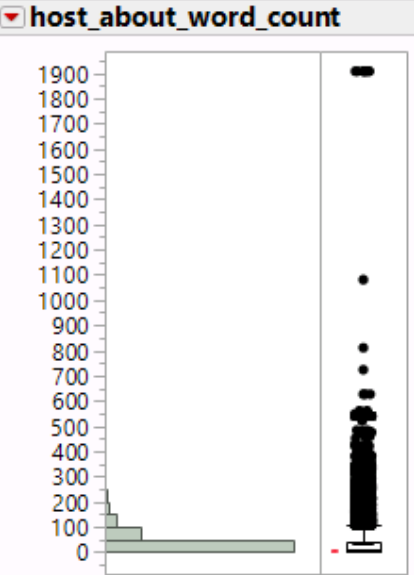
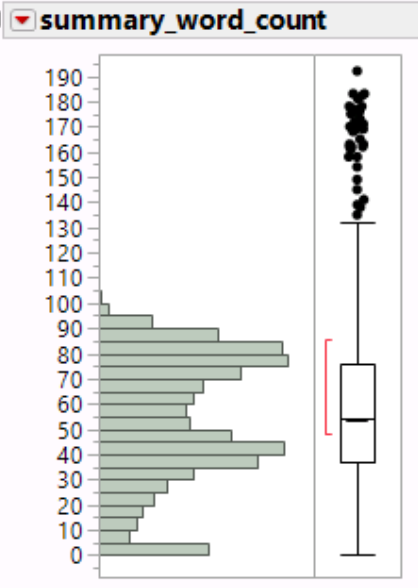
We first clean the text data. It is particularly annoying to see multiple *emojis not separated by space*, since every unique string of emojis are regarded as a unique word in the corpus.

  
**Figure 7. A list of emojis displayed incorrectly in R terminal**

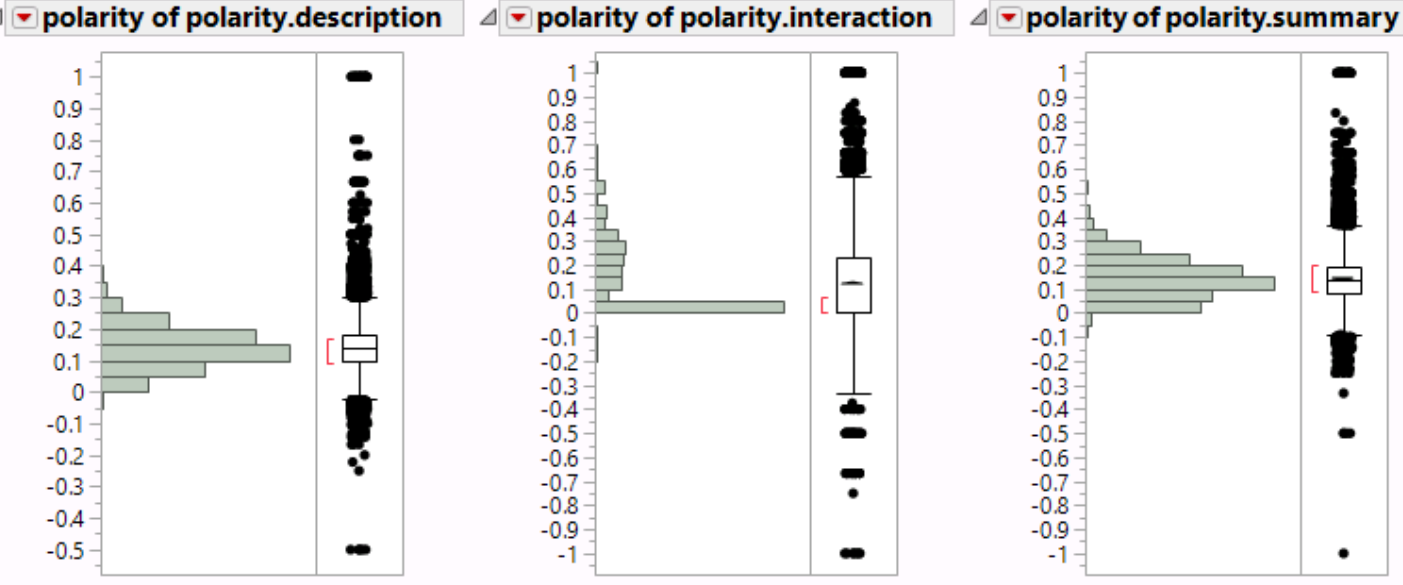
We use library “textclean”[[11]](#footnote-11) to resolve this issue. We replace emojis with word equivalences and remove non-ASCII characters. We replace word elongation (“soooo good” becomes “so good”).

  
**Figure 8**

We then inspect for outliers (figure 9). The word count of summary is not normally distributed, so we add a column denoting whether the summary is empty, in case a linear model couldn’t handle its nonlinearity. The word count of host\_about has extreme outliers, so we cap the value at 300. Similarly, name\_avg\_word\_len is capped at 14.

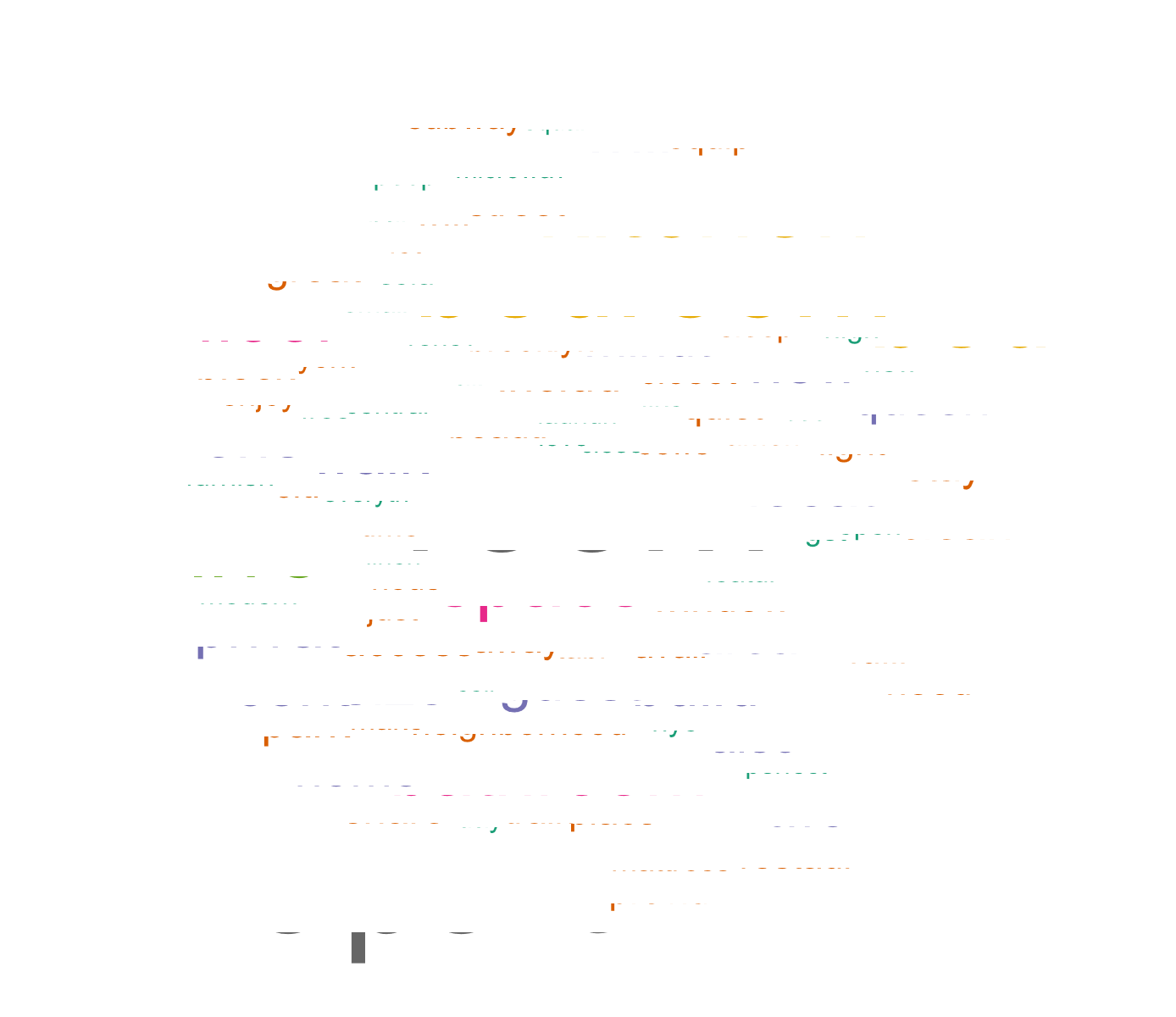
  
**Figure 9**

Most text fields are neutral in terms of sentiment; some are mostly positive.

  
**Figure 10**

For topic modeling, we use top ~500 terms and the Gibbs method with 5000 iterations to find 20 topics for each text field. We identify the topics in long-appendix B.

Figure 11 shows the word cloud for listing description. See long-appendix A for more word clouds.

  
**Figure 11**

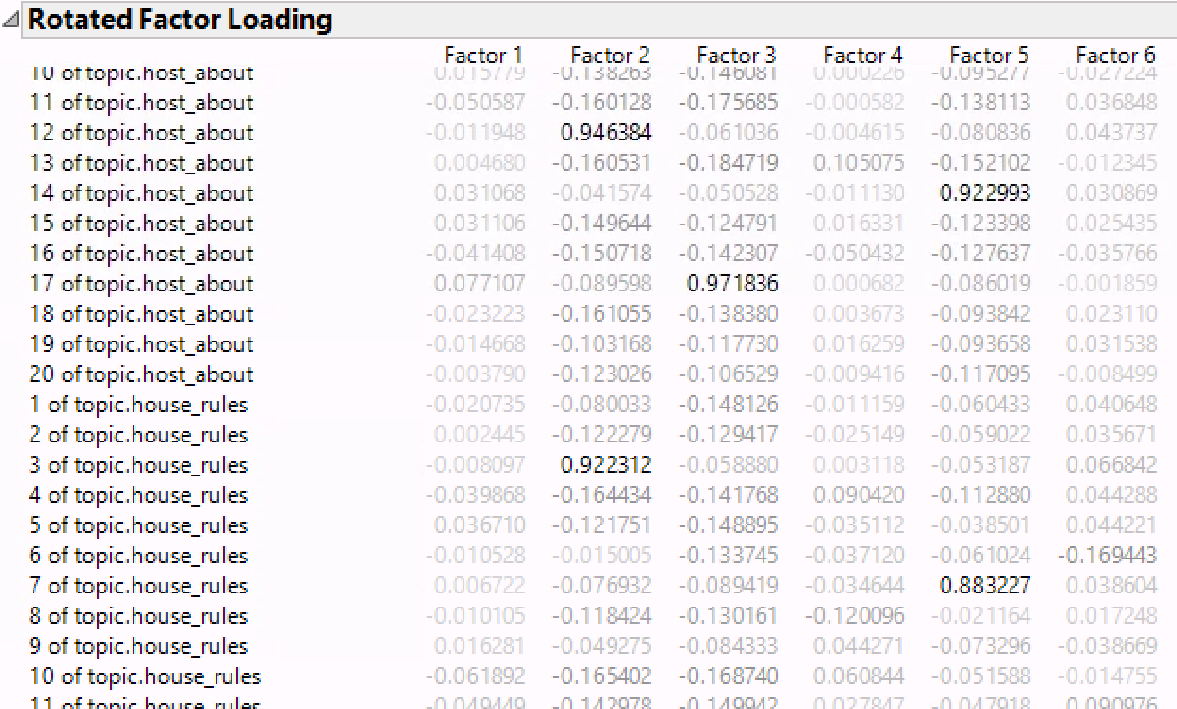
We end up having 155 columns of features related to text. Adding that to the 75-9=66 non-text features, our model is finally ready to run with 221 columns!

## Principal Component Analysis

The next step is Principal Component Analysis (PCA). We create a temporary copy of the data table. We exclude nominal variables that are unusable. We recode some nominal variables to continuous numerical values. The results show that PCA manages to summarize 57 columns to 15 factors while still preserving 59% of overall diversity. See Appendix 4.

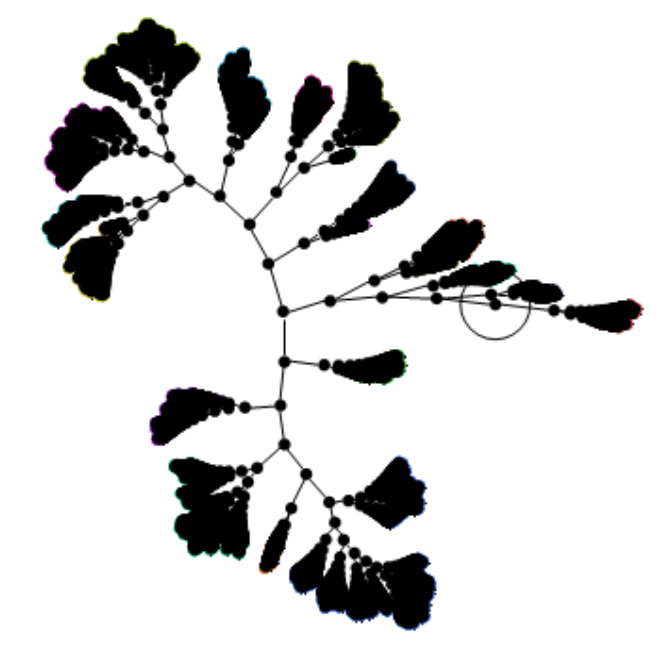
Although the rotated factors are interesting to look at, we do not intend to use the rotated vectors as model input. Because the project goal is to provide guidance for the host to improve the listing’s features, we do not want to add a layer of indirectness to the insights that the final model will give us.

Next, we add text mining results to the PCA. This time, 205 features can be summarized by 66 factors. The rotated factor loading matrix shows that some amenities are correlated with certain topics in a very sparse manner. We are amazed to see this result. See figure 12.

  
**Figure 12**

Another interesting observation is that the hosts whose listings have all-cap names are extraneous and usually have many verifications. Unfortunately, because many text fields are optional for the host, 80% of the columns have missing data; so this PCA result is unsuitable for cluster analysis. In the next section, we use the previous 15-factor PCA results to perform clustering.

## Hierarchical Clustering

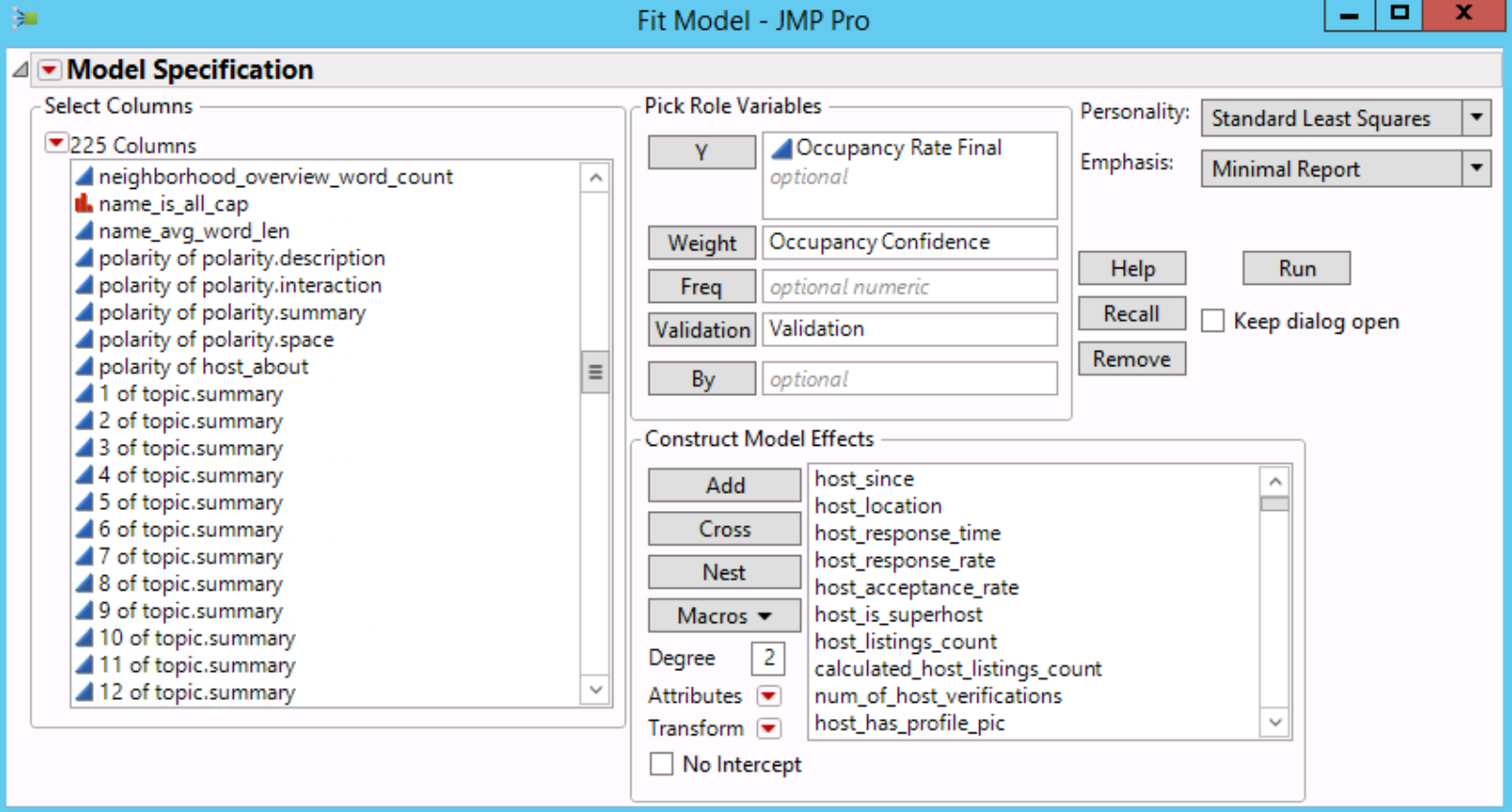
  
**Figure 13. Constellation Plot.**

The cluster analysis suggested 20 clusters. See figure 13 for the constellation plot and Appendix 5 for the dendrogram.

## Demand Curve and Recommendations for the Host

To answer the research question, we need to find the demand curve.[[12]](#footnote-12) The demand curve predicts occupancy rate when supplied with prices and features. The shape of the demand curve then informs the host 1) what features are important and 2) what price is optimal.

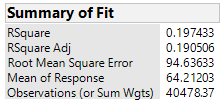
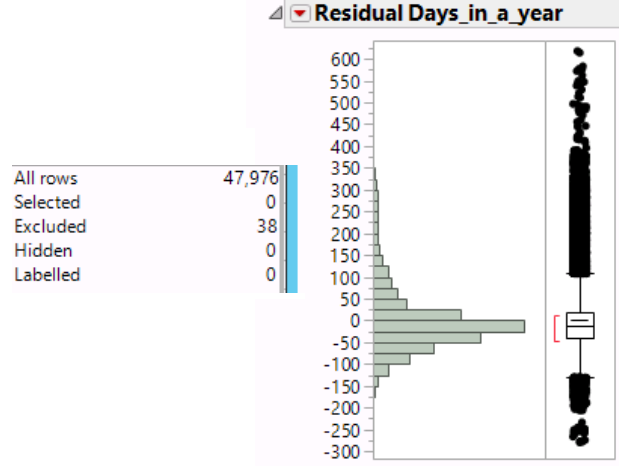
We first try a linear regression without regularization. 20% of the dataset is allocated to the validation set. Regression is weighted by the confidence attached to occupancy prediction.

  
**Figure 14**

We encounter an “ill-conditioned regression” warning. That means there are perfectly correlated model effects. However, when we run a multivariate analysis of the 172 continuous model effects, we find no perfect correlations. We then look at the singularity report and realize there is collinearity since the distribution of 20 topics must sum to 1. Therefore, we exclude topic 20. We also exclude host\_location, zipcode, city and street because they somehow correlate with neighborhood\_cleansed. We discard minimum\_nights because we used it to estimate occupancy rate. We now have 209 model effects left.

We apply log transformation to price, accommodates, bathrooms, bedrooms, beds, security\_deposit, cleaning\_fee, guests\_included, and extra\_people. Informative missing is turned on, and indicator function parameterization is turned off. We multiply occupancy\_rate by 365.25 so that the model parameters would be easier to comprehend.

We run the regression. It fits with adjust RSquare = 19%. Residuals look pretty normally distributed except 38 positive outliers. See figure 15 and long-appendix C.

  
**Figure 15**

Next, we will look at the significant model effects one by one, and offer suggestions to a host looking to list a place on Airbnb. However, the host should ***beware of the confounder effect***. For example, it may appear that a positive sentiment in listing description helps; however in reality, a host only writes positive-sentiment description if the place is nice and they feel proud of it. Faking a positive sentiment in the description without having a nice place may be ineffective in attracting guests.

**Log(price)**. -9.9 days/year. P-value: <.0001

The price sensitivity. For every 11% increase in price, the number of days booked per year decreases by 1. This sensitivity is lower than expected. There must be model effects not captured in the dataset that positively correlate with price and sales.

**Host\_since**. 1 year newer → +0.9 days/year. P-value: 0.0011

HOVAC, if the host is 1 year newer, the number of days booked per year increases by 0.9 on average.

**Host\_acceptance\_rate**. Missing → -19.7 days/year. P-value: <.0001

This is a confounder effect. If the listing is bad, then nobody books it, so the host acceptance rate is missing.

**Host\_is\_superhost**. Yes → +11.5 days/year. P-value: <.0001

Being a super host may be effective on boosting sales. Or, maybe the causality is in reverse?

**Log(calculated\_host\_listing\_count)**. +9.9 days/year. P-value: <.0001

If you double the listing count, yearly sales +6.9 days. This may also be confounded.

**Location[Flushing]**. -12.7 days/year. P-value: 0.0297

**Location[Greenwich Village]**. +12.9 days/year. P-value: 0.0238

**Location[Harlem]**. -6.4 days/year. P-value: 0.0312

**Location[Morningside Heights]**. +34.5 days/year. P-value: <.0001

**Location[Ridgewood]**. -12.0 days/year. P-value: 0.0458

**Location[Theater District]**. +27.0 days/year. P-value: 0.0001

**Location[Woodside]**. -14.2 days/year. P-value: 0.0368

All locations are compared against “other”.

**Neighborhoob\_group\_cleaned[Brooklyn]**. -5.8 days/year. P-value: 0.0098

HOVAC, Manhattan listings have higher demand compared to Brooklyn!

**Log(Accommodates)**. -3.6 days/year. P-value: 0.0236

The more people you accommodate, the less demand there will be? This may be wrong.

**Log(beds)**. +5.4 days/year. P-value: 0.0017

For every 10% more beds you have, you can increase your price by 5.5%... The underlying confounder here, however, should be the size of the room. Increasing the size of the room and having more beds significantly raises the demand!

**Log(security\_deposit)**. +5.5 days/year. P-value: <.0001

This is, again, confounded. The better the place, the higher the demand, as well as the average security deposit.

**Cleaning\_fee**. Missing → +3.0 days/year. P-value: 0.0227

If you don’t collect a cleaning fee, there will be a higher demand.

**Log(guests\_included)**. +5.0 days/year. P-value: 0.0004

For every 22% more guests you include with the offer, the demand grows by 1 day/year.

**Log(extra\_people)**. -3.3 days/year. P-value: 0.0030

Extra\_people is the fee you collect for every additional guest. If you collect a 35% higher fee, the demand drops by 1 day/year.

**Room\_type[Shared room]**. -13.2 days/year. P-value: <.0001

Compared to a full house/apartment, offering a shared room lowers the demand.

**Amen: Air conditioning**. Yes → +2.3 days/year. P-value: 0.0018

**Amen: Hangers**. Yes → 3.3 days/year. P-value: <.0001

**Amen: Carbon monoxide detector**. Yes → +1.6 days/year. P-value: 0.0138

**Amen: Shampoo**. Yes → -2.4 days/year. P-value: <.0001

Whoever mentions shampoo in the listing must have a bad place? We suspect some sort of confounder effect here.

**Amen: Hair dryer**. Yes → +1.6 days/year. P-value: 0.0130

**Amen: Iron**. Yes → +1.3 days/year. P-value: 0.0451

**Amen: Hot water**. Yes → -2.0 days/year. P-value: 0.0050

Again, if you have to mention hot water, you probably don’t have anything else.

**Amen: Refrigerator**. Yes → +3.4 days/year. P-value: 0.0055

**Amen: Dishes and silverware**. Yes → +2.1 days/year. P-value: 0.0383

**Amen: Fire extinguisher**. Yes → +2.4 days/year. P-value: <.0001

**Amen: Lock on bedroom door**. Yes → -1.4 days/year. P-value: 0.0171

This suggests it’s a shared apartment, hence lower demand.

**Amen: Free street parking**. Yes → +3.7 days/year. P-value: <.0001

**Amen: Internet**. Yes → +2.7 days/year. P-value: <.0001

To sum up, it helps when you have more amenities.

**Weekly\_discount**. -23.9 days/year. P-value: <.0001

For every 10% discount you offer if the guests book in bulks of weeks, you lose 2.4 days/year of sales? No, that is false. The confounder here is whether the listing is self-consistent or not.

**Monthly\_discount**. +33.1 days/year. P-value: <.0001

For every 10% discount you offer if the guests book in bulks of weeks, you win 3.3 days of sales per year!

**Maximum\_nights**. +.007 days/year. P-value: <.0001

It is not clear what this means.

**Instant\_bookable**. Yes → +1.5 days/year. P-value: 0.0062

Making your listing instantly bookable creates value for your guests.

**Cancellation\_policy[flexible]**. +18.6 days/year. P-value: 0.0015

**Cancellation\_policy[moderate]**. +18.7 days/year. P-value: 0.0013

**Cancellation\_policy[strict\_14\_with\_grace\_period]**. +22.5 days/year. P-value: <.0001

Compared to having a strict cancellation policy, it is better to be flexible.

**Space\_word\_count**. +.015 days/year. P-value: 0.0227

Consider talking about space! If you leave this field blank, you may be missing out a lot of guests.

**Polarity.interaction**. -7.8 days/year. P-value: 0.0381

In the “interaction” text field of the listing, positive sentiment leads to lower demand.

**Summary topic 8**. -90 days/year. P-value: 0.0157

**Summary topic 15**. +107 days/year. P-value: 0.0033

**Summary topic 17**. -83 days/year. P-value: 0.0302

**Transit topic 2**. -102 days/year. P-value: 0.0150

**Interaction topic 2**. +200 days/year. P-value: 0.0023

**Interaction topic 13**. -176 days/year. P-value: 0.0058

**Description topic 5**. +93 days/year. P-value: 0.0095

**Description topic 10**. +84 days/year. P-value: 0.0113

**Host\_about topic 1**. +242 days/year. P-value: <.0001

**Host\_about topic 2**. +110 days/year. P-value: 0.0172

**Host\_about topic 3**. +140 days/year. P-value: 0.0038

**Host\_about topic 5**. +82 days/year. P-value: 0.0375

**Host\_about topic 10**. +187 days/year. P-value: <.0001

**Host\_about topic 12**. +207 days/year. P-value: <.0001

**Host\_about topic 13**. +95 days/year. P-value: 0.0446

**Host\_about topic 14**. +165 days/year. P-value: <.0001

**Host\_about topic 19**. +123 days/year. P-value: 0.0027

**House\_rules topic 2**. +112 days/year. P-value: 0.0032

**House\_rules topic 3**. +173 days/year. P-value: <.0001

**House\_rules topic 4**. +148 days/year. P-value: 0.0002

**House\_rules topic 6**. +168 days/year. P-value: <.0001

**House\_rules topic 8**. +88 days/year. P-value: 0.0241

**House\_rules topic 9**. +231 days/year. P-value: <.0001

**House\_rules topic 16**. +155 days/year. P-value: <.0001

**House\_rules topic 17**. +124 days/year. P-value: <.0001

**House\_rules topic 18**. +108 days/year. P-value: 0.0024

**House\_rules topic 19**. +100 days/year. P-value: 0.0023

**Space topic 1**. +83 days/year. P-value: 0.0195

**Space topic 10**. +81 days/year. P-value: 0.0305

**Space topic 11**. +165 days/year. P-value: <.0001

**Space topic 12**. +118 days/year. P-value: 0.0039

Again, merely talking about a topic may not be the underlying causal effect; your place must actually have what the topic describes. Please see topic identification in long-appendix B. The coefficients are very big because the topic distributions are usually << 1.

The model also gives us the profiling ability. A host can input their specific features and plot a demand curve of sales vs price. For illustration purposes, here we solve for the optimal price assuming all other features are the population median. The optimal price that maximizes revenue turns out to be $160,000 per night. For comparison, the average price of NY Airbnb listings is $100 per night. We significantly underestimated the demand’s sensitivity to prices! There must be model effects not captured in the dataset that positively correlate with price and sales. Although this result is disappointing, we know we did our best and we hope the other conclusions are still somewhat effective.

## ML Models and Metrics

This section is unrelated to the research question. We use Random Forest, Boosted Tree, and NN to fit the demand curve. We then compare their prediction power.

We do not use these models to answer the research question for two reasons. 1) A lot of these models do not have human-interpretable parameters. 2) They do not offer p-value and are prone to overfitting. A simple linear model, on the other hand, gives conservative and interpretable insights. The linear regression presented in the last section performs very well on the validation set, so there is no need to run an elastic net model, as long as we only focus on those statistically significant coefficients.

The way we assess the performance of the ML models differ from what we learnt from the class (ROC, lift curve...). This is because instead of a classification problem, our ML models try to predict a continuous value.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |

**Figure 16. ML Model Results**

**Table 1. Asses ML Performance**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Linear | Random Forest | Boosted Tree | NN1 | NN2 |
| RSquare, Training Set | 20% | 32% | 30% | 21% | 24% |
| RSquare, Validation Set | 26% | 27% | 24% | 21% | 23% |
| Comment | Baseline. | 100 trees. | 50 layers. | 3 tanh. | 3 + 3 tanh. |

In terms of prediction power, the linear regression and random forest do the best job.

## Future Work

We are very pleased with our results. However, there are many improvements we can make. Here are a few.

Remove outliers. In this project, we remove outliers only when it occurs to us. We often forget to remove outliers. For the study to be more rigid, one should always remember to remove outliers.

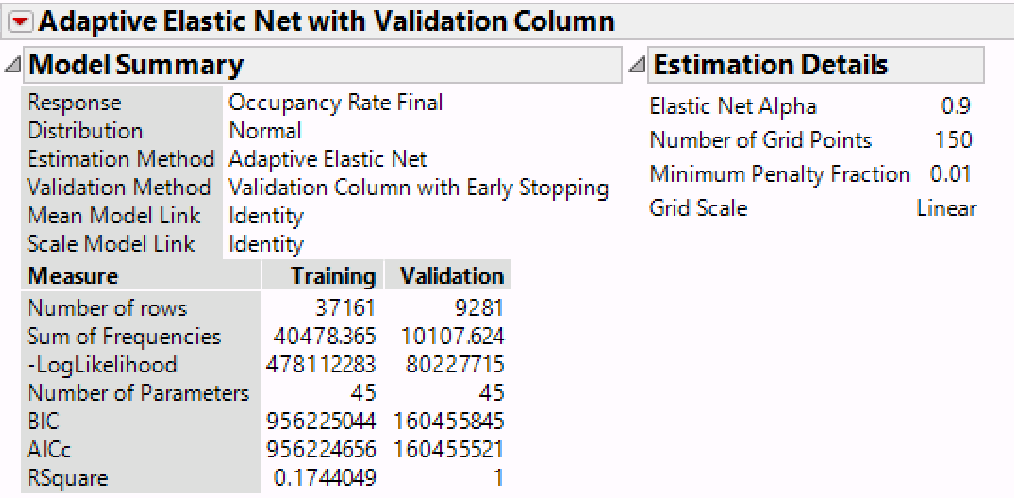
Cleanse amenities. The amenities values are provided by the hosts and are not cleansed. For example, instead of {“Pets allowed”, “Wifi”} (2 items), one host listed {“Pets allowed, Wif”} (1 item, typo). We suspect that Airbnb provides some tags for the hosts to pick and also allows the hosts to create new tags, and hence there are typos and mistakes. If we had more time, we could use edit distance to smartly interpret the tags the hosts provide. We could also enlist typos as a model effect and see whether typos affect your sales.

Analyze images. The dataset contains urls to photos and profile images. It would be helpful if we can extract quantitative data from the images.

Take out confounded variables.

## Questions

This section lists out questions we need help with.

Validation RSquare = 1?

## References

1. Inside Airbnb. San Francisco Model. [insideairbnb.com/about.html](http://insideairbnb.com/about.html), Section “The Occupancy Model”.
2. Brian Chesky. Review Rate. Quora. [qr.ae/pNn4gn](https://qr.ae/pNn4gn).
3. Airbnb Economic Impact. [blog.atairbnb.com/economic-impact-airbnb/#new-york](https://blog.atairbnb.com/economic-impact-airbnb/#new-york).

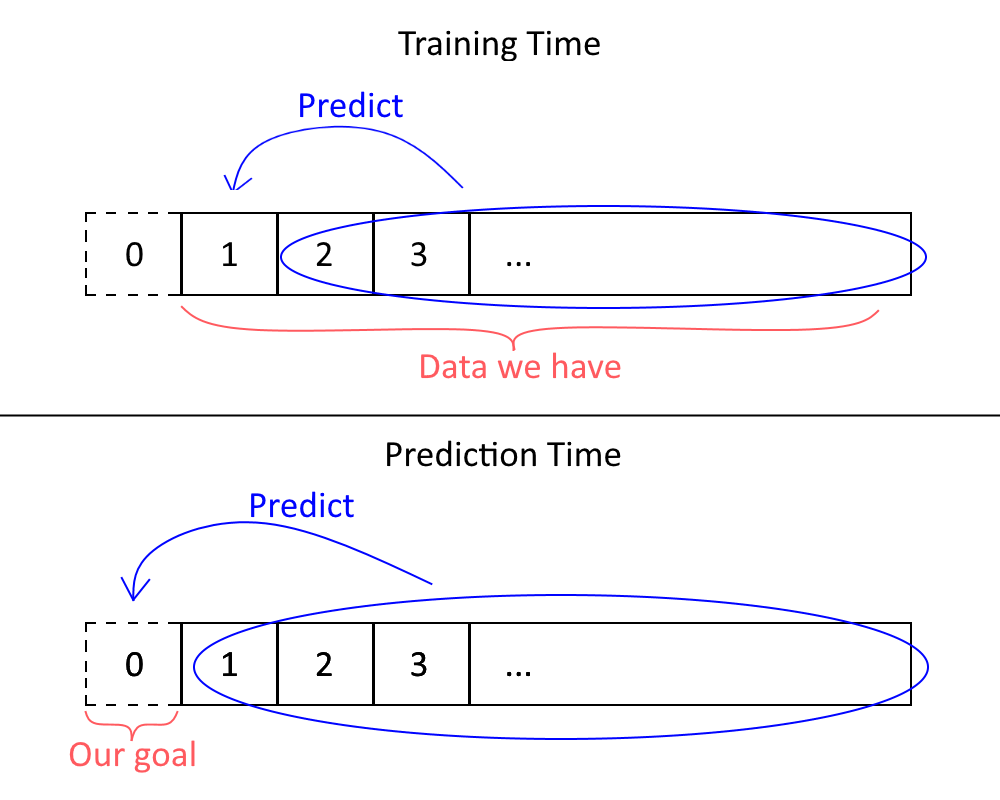
## Appendices

All source code can be found at [github.com/Daniel-Chin/airbnb](https://github.com/Daniel-Chin/airbnb).

### Optimize x1 and x2

We tried two methods to optimize x1 and x2. The first method (legacy method) failed. The second method yields satisfying results.

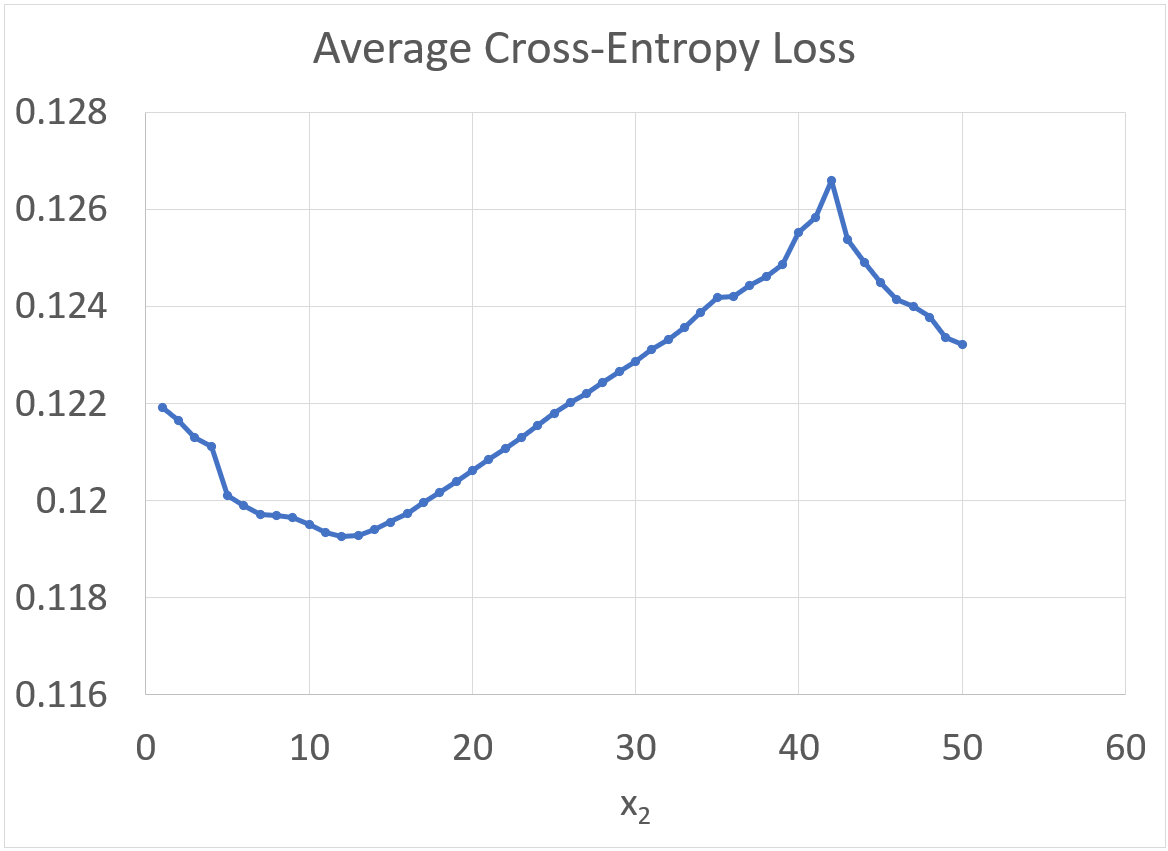
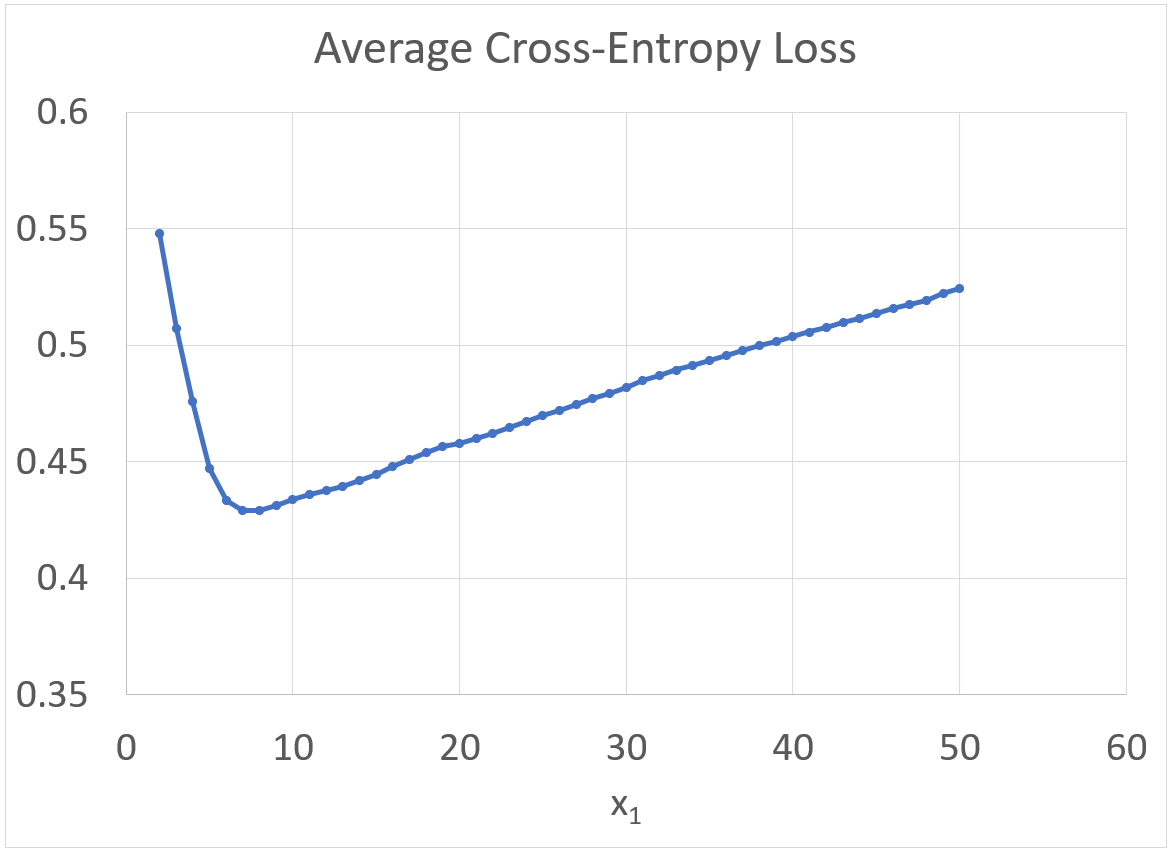
Here goes the legacy method. For any given x1, we can pretend we do not know the availability of x=1 and predict probability\_unavailable for x=1. The goodness of x1 is thus indicated by the cross-entropy loss on x=1. With the optimal x1, we then make x=1 visible to the algorithm and predict probability\_unavailable for x=0. See figure 17.

  
**Figure 17**

x1=6 yields minimum average loss 0.429, x2=12 yields minimum average loss 0.119

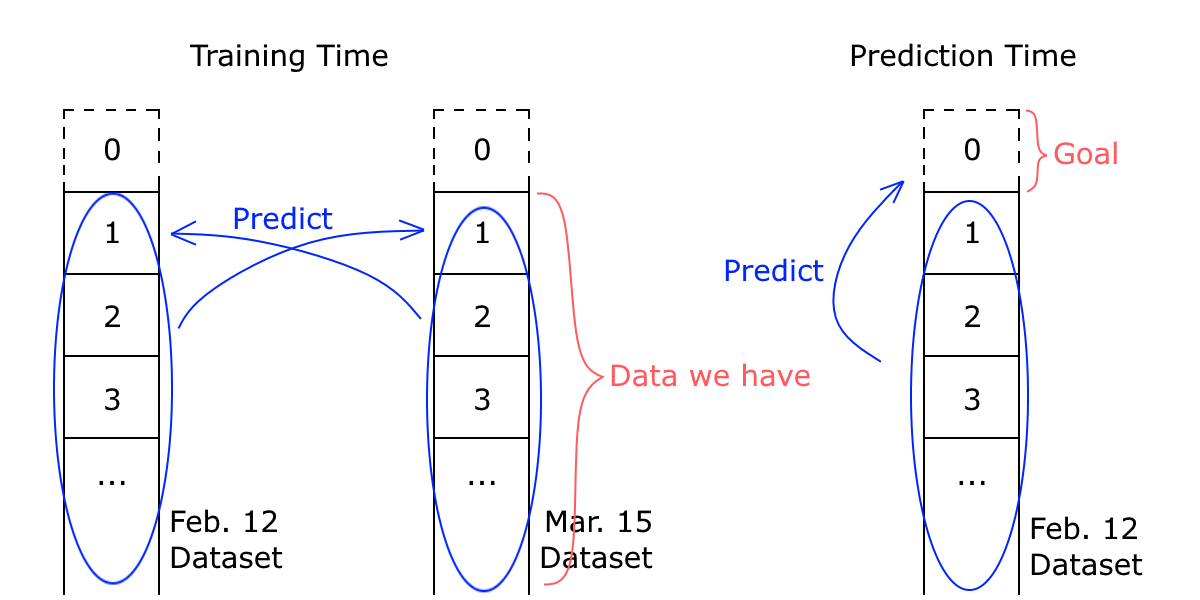
When |y\_hat - y|=1, we adjust the loss from infinity to 5.

Source code is at ./demandAnalysis/x1x2.py

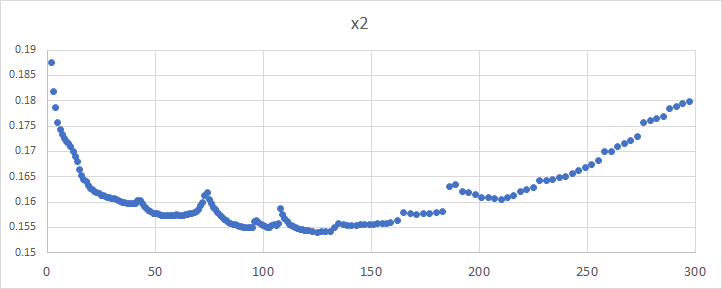
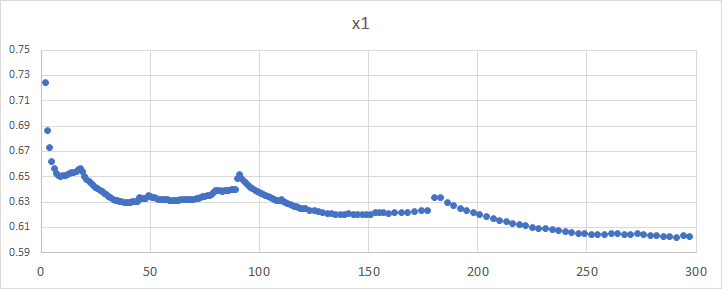


However, the occupancy estimated by the legacy method has a very low R-square with review scores. Unsatisfied, we look for reasons. Closer analysis reveals a serious mistake: this model assumes availability is independant across days. In reality, because days are booked in consecutive lumps, nearby days have positively correlated availablity. The ML model can easily exploit that fact and give a very small x1.

To fix that, we waited one month and downloaded a newer version of the calendar availability data. The ML method is described in the main text. Note that with the extra dataset, there can now be many better methods of estimating occupancy; however, we want to retain our project scope of using only the Feb. 12 dataset, so we do our best to leave the extra data untouched. Consider the extra data we use to obtain optimal x1 as a “one-time local hack”.



Plotting average loss against x1 x2:



We set x2=93. For x1, we take 9, 39, 135, 258, 291 as four candidates. To assess their validity, we check the correlation between number\_of\_reviews and occupancy\_rate predicted by the calendar using x1. Legacy method gives R-Square = .0051. The improved method gives the following results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x1 | 9 | 39 | 135 | 258 | 291 |
| R-Square | .184 | .139 | .134 | .123 | .121 |
| Estimated Review Rate | 37% | 38% | 43% | 49% | 51% |

We set x1=9 to maximize R-Square.

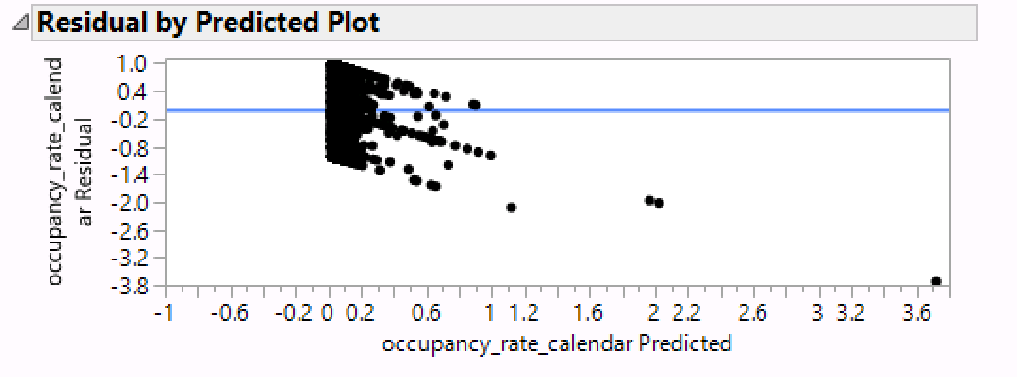
Source code is at ./demandAnalysis/m2\_x1x2.py

### Linear Regression Model between number\_of\_reviews and occupancy\_rate

We first calculate the average length\_of\_stay per listing. According to Airbnb, “visitors stay on average 6.4 nights” in New York [3], which we use as the baseline. Then, we observe that the acceptable minimum\_nights is <7 for 41180 listings and >=7 for 9917 listings. We use the minimum\_nights for those >=7. For those <7, we use 6.4. We initially wanted to adjust them so that the overall average remains 6.4, but the adjusted average turned out to be 1.031 if we assume equal weight to all the listings. This is obviously unrealistic, so we decided to use 6.4 instead. See discussion and source code at ./demandAnalysis/length\_of\_stay.py

Divide occupancy\_rate by length\_of\_stay, we get y. Remove outliers. Run a regression from review\_per\_day (calculated) to y. The probability of guests leaving reviews turns out to be 38.3% ± 1.2%. It is significantly lower than [1] and [2].

A linear regression between the two versions of occupancy\_rate predicted leaves residuals:



This does not look too good; however, the relation between review counts and occupancy is, by nature, linear. Therefore, we are happy with the linear model.

### Most Common Amenities

Wifi 49830

Heating 48124

Essentials 47281

Kitchen 46473

"Smoke detector" 44595

"Air conditioning" 43493

Hangers 38471

"Carbon monoxide detector" 36249

TV 35086

Shampoo 34568

"Hair dryer" 34045

"Laptop friendly workspace" 32520

Iron 32353

"Hot water" 28744

Refrigerator 22867

"Dishes and silverware" 22132

Washer 20821

Dryer 20463

"Fire extinguisher" 20056

Microwave 19339

"Lock on bedroom door" 19046

Stove 18960

"Cooking basics" 18924

Oven 18553

"Free street parking" 18496

"Coffee maker" 17698

"First aid kit" 17583

"Bed linens" 16954

Internet 13857

Elevator 13631

For the sake of efficiency, we take the 30 most common amenities tags.

### PCA Results

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Row | F 1 | F 2 | F 3 | F 4 | F 5 | F 6 | F 7 | F 8 | F 9 | F 10 | F 11 | F 12 | F 13 | F 14 | F 15 |
| amen\_"Hair dryer" | 2 | 1 | 6 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | -1 | -1 | 0 | 1 |
| amen\_Iron | 2 | 1 | 6 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | -1 | 0 | 1 |
| amen\_Hangers | 2 | 0 | 5 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 1 | 0 | -1 | 0 | 1 |
| amen\_Shampoo | 1 | 0 | 5 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | -1 | -1 | 0 | 1 | 0 |
| amen\_"Laptop friendly workspace" | 2 | 1 | 4 | 0 | 0 | -1 | 0 | 0 | 1 | 0 | 1 | -1 | 0 | 0 | 1 |
| amen\_Essentials | 1 | 0 | 4 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | -1 | 0 | 0 | 0 |
| amen\_"Air conditioning" | 1 | 1 | 3 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 1 | 1 | 0 | -1 |
| amen\_Heating | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| amen\_"Hot water" | 6 | 0 | 3 | 0 | 0 | -1 | 0 | 1 | 1 | 0 | 1 | 0 | -2 | 1 | 1 |
| amen\_TV | 1 | 1 | 3 | 1 | 0 | 1 | 0 | -1 | 1 | 0 | 0 | 2 | 1 | 1 | -1 |
| amen\_"Smoke detector" | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 1 | 0 | -1 |
| amen\_Wifi | 0 | 0 | 2 | 0 | 0 | 0 | 0 | -1 | 1 | 0 | 0 | 0 | 1 | 0 | -1 |
| amen\_"Carbon monoxide detector" | 2 | 1 | 2 | 0 | 0 | 0 | 0 | 1 | 7 | 0 | 1 | 0 | 0 | 0 | 0 |
| amen\_"First aid kit" | 1 | 1 | 2 | 0 | 0 | -1 | 0 | 0 | 3 | 0 | -1 | 0 | -1 | 1 | 1 |
| amen\_"Coffee maker" | 7 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | -1 |
| amen\_"Bed linens" | 6 | 0 | 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | -1 | 1 | 1 |
| amen\_"Fire extinguisher" | 2 | 1 | 2 | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 0 | 1 | -1 | 1 | 1 |
| host\_is\_superhost | 3 | 0 | 2 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | -1 | -1 | 1 | 0 |
| amen\_"Dishes and silverware" | 9 | 0 | 2 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | -1 |
| amen\_"Free street parking" | 5 | 1 | 2 | -1 | 0 | -2 | 0 | 0 | 1 | 0 | 0 | -1 | -2 | 0 | 0 |
| cancellation\_policy | 2 | 1 | 2 | 0 | 1 | 0 | 0 | -1 | 0 | 0 | 3 | 0 | 0 | 1 | 0 |
| cleaning\_fee | 2 | 4 | 2 | 1 | 0 | 3 | 0 | 0 | 0 | 0 | 5 | 0 | 1 | 1 | -1 |
| amen\_"Cooking basics" | 9 | 0 | 2 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 |
| amen\_Refrigerator | 9 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| amen\_Microwave | 8 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | -1 | 0 | -1 |
| accommodates | 1 | 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | -1 |
| host\_acceptance\_rate | 2 | 0 | 1 | -1 | 4 | 0 | 0 | 4 | 0 | 0 | -1 | -1 | 0 | 1 | -1 |
| amen\_Stove | 9 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 1 | 1 |
| amen\_Oven | 9 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 3 | 0 | 2 |
| amen\_"Lock on bedroom door" | 1 | -1 | 1 | -1 | 0 | -2 | 0 | 2 | 2 | 0 | -1 | 0 | -2 | 0 | 2 |
| host\_response\_rate | 1 | 0 | 1 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 |
| beds | 1 | 8 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| guests\_included | 1 | 5 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 4 | -1 |
| amen\_Dryer | 0 | 1 | 1 | 10 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| amen\_Washer | 0 | 1 | 1 | 10 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 |
| amen\_Elevator | -1 | -1 | 1 | 4 | 0 | 2 | 0 | -1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| amen\_Kitchen | 1 | 1 | 1 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | -1 | 4 | -1 | 0 |
| instant\_bookable | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 5 | 0 | 0 | -1 | 0 | 0 | 0 | -1 |
| extra\_people | 1 | 1 | 1 | 0 | 0 | -1 | 0 | 0 | 1 | 0 | 2 | -1 | -1 | 4 | 0 |
| calculated\_host\_listings\_count | 1 | 0 | 1 | 1 | 0 | 8 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| host\_since | 0 | 0 | 1 | 0 | 0 | 1 | -2 | 6 | 0 | -1 | -1 | 5 | 0 | 0 | 0 |
| num\_of\_host\_verifications | 1 | 0 | 0 | 0 | 0 | 1 | 0 | -1 | 0 | 0 | 0 | -4 | 0 | 0 | 0 |
| host\_listings\_count | 1 | 0 | 0 | 1 | 0 | 8 | 0 | 1 | 0 | 0 | 1 | -1 | 0 | 0 | 0 |
| security\_deposit | 0 | 1 | 0 | 1 | 0 | 0 | 0 | -1 | 0 | 0 | 5 | 0 | 0 | 1 | 0 |
| price | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| require\_guest\_phone\_verification | 0 | 0 | 0 | 0 | 0 | 0 | 9 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| require\_guest\_profile\_picture | 0 | 0 | 0 | 0 | 0 | 0 | 9 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| host\_identity\_verified | -1 | 0 | 0 | 0 | 0 | -1 | 0 | -4 | 0 | 0 | 0 | -4 | 0 | 0 | -1 |
| bedrooms | 1 | 8 | 0 | 0 | 0 | -1 | 0 | 0 | 0 | 0 | 1 | -1 | 0 | -1 | 1 |
| bathrooms | 0 | 5 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | -1 | 1 |
| host\_has\_profile\_pic | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | 0 |
| amen\_Internet | -1 | 0 | 0 | 0 | 0 | 0 | 1 | -5 | 0 | 1 | 0 | -1 | 1 | 0 | -1 |
| maximum\_nights | -1 | 0 | 0 | 1 | 0 | 2 | -1 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | -1 |
| minimum\_nights | 0 | 0 | 0 | 1 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | -1 | 1 |
| weekly\_discount | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 |
| monthly\_discount | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -2 | 0 | 10 | 1 | 0 | 0 | 0 | 0 |
| host\_response\_time | -1 | 0 | -1 | 0 | -9 | 0 | 0 | -1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

\* Loading values multiplied by 10 and rounded to the nearest integer.

Factor 1: Appliances. Factor 2: Capacity. Factor 3: Good shower. Factor 4: Laundry. Factor 5: Host response. Factor 6: Host listings count. Factor 7: Require guest profile. Factor 8: No internet. Factor 9: Smoke detector. Factor 10: Bulk discount. Factor 11: Additional fees. Factor 12: Unverified. Factor 13: Cooking. Factor 14: Guest fee. Factor 15: Unknown.

### Dendrogram of Hierarchical Clustering



**Long-Appendix A. Word Clouds**

**Long-Appendix B. Identify the Topics**

**Long-Appendix C. Linear Regression Report**

1. We also considered trying linear regression or sigmoid regression on the calendar, but we rejected those methods. The current model assumes: 1) the position of the closed-day chunks is random and independent of other listing features; 2) occupancy is 0 near x=end. (1) is reasonable since scrape\_day randomizes the horizontal offset of the calendar. (2) makes our method a *conservative* estimation in extremely popular listings. [↑](#footnote-ref-1)
2. 30 / reviews\_per\_month gives the average review blank time. Multiply by 2 and we get the 95% confidence threshold that the listing is more inactive than usual. [↑](#footnote-ref-2)
3. The bad features correspond nicely with the assumptions of the calendar model. [↑](#footnote-ref-3)
4. Input includes: probability\_unavailable, probability\_closed, and the lengths of the longest three chunks of unavailable days. [↑](#footnote-ref-4)
5. We justify the use of NN by comparing it to a linear regression model. [results here] [↑](#footnote-ref-5)
6. We tried various NN structures. We found that giving the model more layers and parameters does not yield better results, so we settled with 1 layer of 3 tanh nodes to minimize room for overfitting. In the end, the validation set R-Square was higher than that of the training set, so we concluded there was no significant overfitting. [↑](#footnote-ref-6)
7. Why not use actual discrepancy? If we used actual discrepancy, the final equation could be simplified (everything is linear!) to a weighted average, just as if we did not consider the quality score. It is also wrong because the calendar’s quality score should only depend on the calendar, but not on both the calendar and the review counts. The predicted discrepancy does exactly the job. Without overfitting, the NN let no information from the review counts’ side sneak into the predicted discrepancy. [↑](#footnote-ref-7)
8. The reviews data are excluded because as a host, when we try to optimize our features, we do not want to hold the review scores as constant. The review scores can be a result of our features. [↑](#footnote-ref-8)
9. For example, $10 a day, $63 a week would yield weekly\_discount = 10%. Missing values in weekly\_price yields weekly\_discount = 0. This transformation helps, because instead of tasking the machine to interpret *informative missings*, we interpret them correctly beforehand (using common sense) for the machine. [↑](#footnote-ref-9)
10. We tried PCA and PCA gave an unreasonably high loading to maximum\_nights. Only then did we realize we forgot to clean outliers. [↑](#footnote-ref-10)
11. [github.com/trinker/textclean](https://github.com/trinker/textclean). [↑](#footnote-ref-11)
12. Due to its high dimensionality (200+), the demand curve we are solving for may as well be called “demand surface”. [↑](#footnote-ref-12)