ORIE 4741 Final Project Report

Yanyun Chen yc2295, Jenny Li jl2872, Jiwon Kim jk2332

December 10th

Contents

1 Introduction **2**

2 Data Analysis **2**
2.1 Data Description . . . 2
2.2 Data Visualization and Data Cleaning . . . 2
2.2.1 Distribution of Prices . . . 2
2.2.2 Correlation of real values . . . 3
2.2.3 Set Data - amenities . . . 3
2.2.4 Categorical Data - property type . . . 4
2.2.5 Missing Values - Imputing Matrix vs Dropping Values . . . 4
2.2.6 Embedding for text data-NLP Model . . . 4
2.3 Features Selected for Training . . . 5

3 Model Selection and results **5**
3.1 Back-Slash . . . 5
3.2 Proximal Gradient . . . 5
3.2.1 Huber-Loss . . . 5
3.2.2 L1 Loss . . . 6
3.3 Binary classification . . . 6
3.4 Categorical and Ordinal Classification . . . 6
3.4.1 One-vs-All Hinge Loss . . . 7
3.4.2 Multinomial Logit Loss . . . 7
3.4.3 Big vs Small Loss with Logistic Loss . . . 7

4 Discussion of results **7**

5 Weapon of Math Destruction and Fairness **8**
5.1 Weapon of Math Destruction . . . 8
5.2 Fairness . . . 8

6 Future Improvement **9**
6.1 Learning from Pictures . . . 9
6.2 More effective embeddings using NLP Model . . . 9

7 Conclusion **9**

1 Introduction

The data set we are investigating in this project is the New York City Airbnb data from InsideAirbnb.com. Using the listings data, our goal is to predict the price of a given listing. We aim to not only train a model that can successfully make prediction of a price of a listing, but also to gain insights on the potential key factors that influence pricing, all the while acknowledging any limitations and assumptions of the data and our model. Such a model would have the following benefits: first, house owners can have a reference when they set the price, and by making their initializing process less confusing, Airbnb might be able to attract more prospective hosts; second, consumers of Airbnb can also benefit from this predicted price serving as a reference, as they feel more comfortable and confident when having better understanding of how prices are determined; last but not the least, regulatory policies regarding prices can be set accordingly by Airbnb.

2 Data Analysis

2.1 Data Description

Our raw data, the listing.csv file, has 48,377 examples and 106 features with real values, nominal values, categorical values, set values and texts. After a preliminary investigation into each of the columns, we decided to discard the following features from our model:

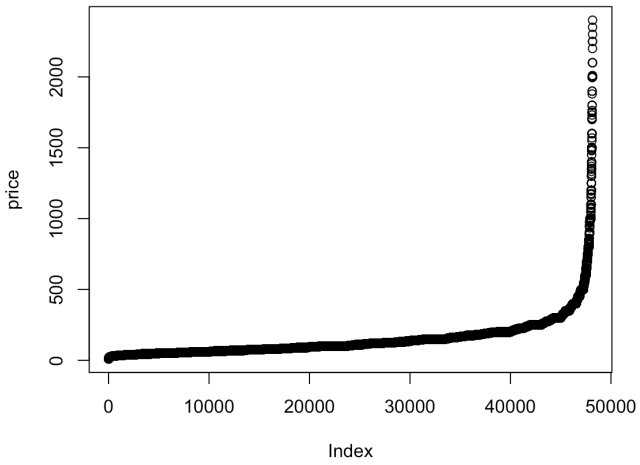
Name of Column Discarded	Reason of Discarding
"thumbnail_url" "medium_url" "xl_picture_url" "square_feet" "weekly_price" "monthly_price" "license" "jurisdiction_names"	Over 80 percent of data missing
"experiences_offered" "host_acceptance_rate"	Columns have very low variance
"host_id","host_url","host_name"	Column probably has low correlation with price
"host_thumbernail-url", "host_picure_url"	Column contains url
"city" "state" "market"	Already narrowed location down to NYC
"has_availability", "calendar_last_scraped",	Value dependent on data collection time
"first_review","last_review"	Date probably has low relevance in predicting price
"host_neighbourhood"	Current location of host not relevant

Table 1: Discarded Columns

2.2 Data Visualization and Data Cleaning

2.2.1 Distribution of Prices

To first get an understanding of the distribution of prices, we provide a sorted price plot (Figure 1). We observe that the majority(98.2%) of the data fall under the $500 threshold, and then the price distribution shoots up. After some investigation into listings priced at $500+, they are either luxury villas that only account for a very small portion of listings in NYC or a measure taken by hosts to prevent their listing



10.7

Figure 1: Prices of listing, sorted in increasing order

from booking temporarily. As a result, we decided to only include listings priced under $500 in our model.

2.2.2 Correlation of real values

To gain a more concrete understanding of the relationship between the real-valued features and the price, a correlation plot (Figure 2) is created. The 'accommodates' ,'bedrooms','beds', and 'guests_included' features have the strongest correlations with pricing, which agrees with common sense and the direction is as expected. The plot also reveals a surprising finding of a strong correlation between 'cleaning_fee' and price as well. Our choice of real-valued features included in the model is also based on this correlation plot. We further confirm the strength and direction of correlation between these features and prices with boxplots (Figure 3a,3b).

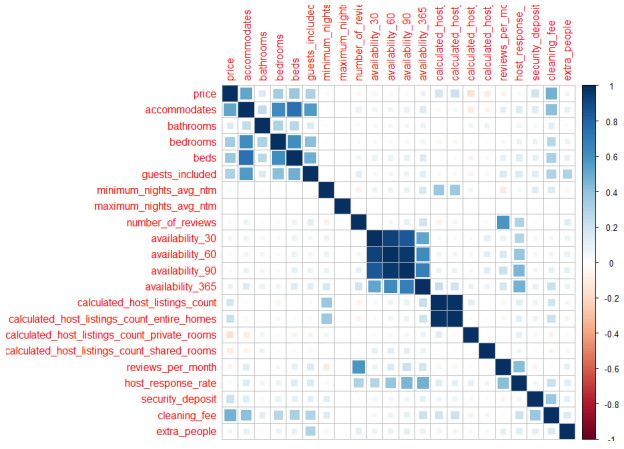
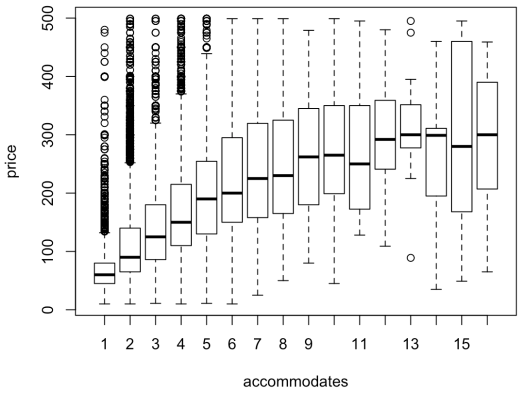


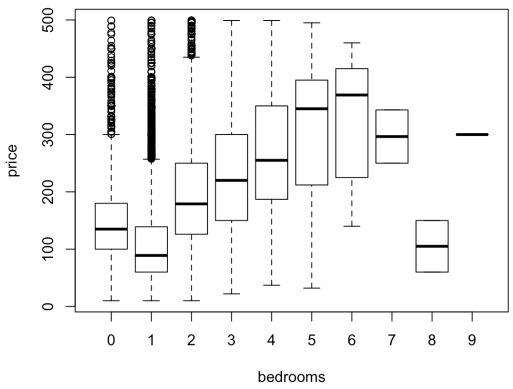
Figure 2: Correlation Visualization

2.2.3 Set Data - amenities

Airbnb allows hosts to show amenities up to a very large number and are often too detailed. Thus doing one-hot encoding on unique amenities will give rise to an unnecessary expansion of the feature space. We decided to organize the amenities into broader categories like the followings: 1.tv = "TV","Cable



(a) Number of accommodates with price



(b) Number of bedrooms with price

Figure 3

TV"]; 2.bath = ["Hot tub", "Hot water"]; 3.internet = ["Internet", "Wifi"]; 4.conditioning = ["Air conditioning", "Heating"]; 5.cook1 = ["Cooking basics", "Oven","Stove", "Refrigerator" , "Dishes and silverware","Microwave"]; 6.cook2 = ["Kitchen","Microwave"]; 7.essentials = ["Shampoo","Hair dryer","Iron", "Washer","Dryer"]; 8.park1 = ["Free street parking"]; 9.park2 = ["Paid parking off premise"], etc.

2.2.4 Categorical Data - property type

The issue with this specific feature is that some categories only have less than ten examples, such as "cave" and "barn". We decide to combine such categories into one–"Other". We list out some examples of the property types and the corresponding number of listings in our dataset. "Apartment" 37388.0, "Guest suite" 378.0, "Townhouse" 1523.0, "Bed and breakfast" 71.0, "Loft" 1266.0, "Condominium" 1424.0, "House" 3829.0, etc.

2.2.5 Missing Values - Imputing Matrix vs Dropping Values

Upon exploring the seven review scores features, we noticed that there are nearly 1/4 of all examples missing values for these columns (10,087 rows out of 47,075 rows). We explored the option of imputing missing values or to simply drop examples with missing values since we have a relatively large raw dataset. Using Low Rank Models, we investigated the different loss functions we could use to perform matrix completion. We found that L1 Loss produces an imputed matrix with the lowest mean absolute error. Then, we combined the completed review score matrix to the training data. A simple backslash calculation shows that adding this imputed matrix barely has MSE around 2700 on training set and 2800 on testing set.

We then tried to simply dropping rows with missing values for the review scores. The MSE then decreased to around 2300, which is a significant improvement. Thus, considering that we still have a good amount of data to work with, we decided to discard any data that is missing value for the seven review score columns.

2.2.6 Embedding for text data-NLP Model

We initially started off with basic many-hot encoding (bag of words such as 'luxury', 'spacious', 'train', etc.) but ended up with[1] universal-sentence-encoder-lite from Tensorflow hub to create word embeddings for a more complex model. Specifically, we incorporated these columns: 'name', 'summary', 'space',

[1]The authors found that Tensorflow embedding produces smaller error compared to bag-of-words encoding.

'neighborhood_overview', 'notes', 'transit', 'access', 'interaction' and 'house_rules'. We put them into a single paragraph for each data point and ran universal-sentence-encoder-lite to convert it into a 512-length vector.

2.3 Features Selected for Training

The real-valued features that we selected are: 'accommodates', 'bathrooms','bedrooms', 'beds','guests_included', 'calculated_host_listings_count', 'calculated_host_listings_count_entire_homes', 'cleaning_fee', 'security_deposit', 'review_scores_rating', 'review_scores_value', 'review_scores_accuracy', 'review_scores_cleanliness', 'review_scores_communication', 'review_scores_checkin', and 'review_scores_location'
For categorical and set data, we produced one-hot vectors along with many-hot vectors for the features: 'amenities', 'property_type', 'room_type', 'bed_type' , 'cancellation_policy', 'neighbourhood_cleansed'. 'host_is_superhost', 'instant_bookable','is_business_travel_ready', 'requires_license', 'require_guest_profile_picture' and 'require_guest_phone_verification' are the boolean features we chose.
Finally, we also used the embeddings on text data as we mentioned in section 2.2.6. The size of the feature space ends up to be 618. We split 80% (30127 rows) of our cleansed data into training dataset and the rest (7532 rows) goes to test dataset. To further validate our model, we also do 5-fold cross validation on splitting the data.

3 Model Selection and results

In this section, we document the different models and approaches we experimented, along with the error rates for each of the models. For all of the models we tried, we selected 80 percent of the data into the training set and put the remaining 20 percent into the testing set.

3.1 Back-Slash

Our first attempt is utilizing the backslash shortcut that Julia provides, which fits a linear regression by minimizing the quadratic loss function. This will be the baseline model that will be compared with other models we explore. With the simple backslash calculation and a 5-fold cross-validation, the training and testing MSE we found converge at around 2300 and 2400 respectively.

3.2 Proximal Gradient

Our second trial is the proximal gradient method, where we experimented with many potential loss functions and regularizers.

3.2.1 Huber-Loss

Note that the listings with prices higher than 250 dollars tend to be luxury homes, with highly fluctuating price points. These data consist of a much smaller portion of the total data, and we want to prevent our model from being affected by those outliers. Therefore, we considered using the Huber Loss function. Our intuition is that the model should penalize less for the houses with higher price point, and still perform the same quadratic penalization for prices under the 250 threshold. The objective of a model with Huber Loss function is $min\ \frac{1}{n}\sum_{i=1}^{n} huber(y - w^t x_i)$, where:

$$huber(x) = \begin{cases} \frac{1}{2}x^2 & |x| \le k \\ k(|x| - \frac{1}{2}k) & |x| > k \end{cases}$$

We tried out Huber Loss with no regularizer, with quadratic regularizer, and with l1 regularizer. With just the Huber Loss function, the MSE is around 3751.7 for training error, and 3452.67 for testing . For the two other regularizers, we performed cross-validation to find the best lambda value as well.

L1 Regularizer

When looking at the features we selected, we realized that some of the columns might not be independent. A potential problem might be that the solution we found is not unique. Thus, using a l1 regularizer places the guarantee that we arrive at an unique solution. Using this regularizer can also help eliminate any features that do not contribute strongly to price prediction. After training a model with Huber Loss and L1 Regularizer,and experimenting with different λ values, we found that the MSE is at 3513.479 for training and 3236.541 for testing

L2 Regularizer

Compared to the L1 Regularizer, a L2 regularizer might not produce a solution that is quite as sparse, but we are unsure whether a sparse solution would improve our model. The MSE for Huber loss with Quadratic Regularizer is 3513.637 for training and 3236.675 for testing.

3.2.2 L1 Loss

Since the Huber Loss function still have relatively high MSE, we also tried the L1 loss function. Compared to Huber Loss, the L1 loss function is even more robust, and even less sensitive to outliers.

Again, similar to Huber Loss, we first tried to combine L1 Loss function with no regularizer, with the L1 regularizer, and with quadratic regularizer. With no regularizer, the L1 Loss function achieves an MSE of 3552.29 for training error and 3271.64.

L1 Regularizer Following the logic of Huber Loss, the second approach we took was to combine the L1 Loss with L1 Regularizer. The error rate we get is 3529.491 for training MSE, and 3251.105 for testing MSE.

L2 Regularizer Using this combination of L1 Loss with Quadratic Regularier, we get a training MSE of 3552.989, and a testing MSE of 3272.277.

3.3 Binary classification

Another model worth considering is a binary classification on high/low price. We set the threshold at $250 mainly because the price prediction model we presented above generally underestimate listings priced above $250. This model is trained using the perceptron algorithm where a positive prediction is classified as high-price listing. After applying a Hoeffding's bound on our result, with a confidence level of 95%, the sample error rate 0.3 is within 0.02 of true error rate.

Algorithm 1 Perceptron algorithm

Initialize w = 0
while *there is a misclassified example (x,y)* **do**
  w = w + yx
end

3.4 Categorical and Ordinal Classification

Instead of predicting the exact number of prices, we also consider the possibility of predicting price ranges. This is practically meaningful in that providing a range to the hosts or the customers gives room for adjustment and alleviates deviation. Thus we attempted to train categorical and ordinal classification models on price ranges with a stepsize of $50 and a total of 10 classes.

Since computing the MSE for a multi-classification model does not make sense, we will measure the performance using two error metrics. One is an error rate, where an error is when the predicted class is different from the actual value. The other one is a loose error rate, where an error is when the predicted label is more than 1 label away from the actual label (i.e. a predicted label of 6 with an actual label of 4 is considered an error, while a prediction of 5 is not).

3.4.1 One-vs-All Hinge Loss

We first trained a model using one-vs-all hinge loss function, one of the two loss functions we've learned that performs categorical classification. It separates the bigger classification problem into reducing a series binary problems. Specifically, the loss function is$l(z, y) = (1 - z_y)_+ + \sum_{y' \neq y}(1 + z_{y'})_+$. The exact error rate for training is 0.624 and 0.622 for testing. If we were to use the loose error rate, then the training error goes down to 0.269, and the test error is 0.273.

3.4.2 Multinomial Logit Loss

We also tried the multinomial logit loss, which performs classification through learning probabilities. In this case, our model is minimizing $l(y, z) = -log(\frac{exp(z_y)}{\sum_{j=1}^{k} exp(z_j)})$. The error rate for this model is quite similar to the one-vs-all model, with a training error of 0.625 and testing error of 0.62. Using the looser error calculation we get a 0.273 training and testing error.

3.4.3 Big vs Small Loss with Logistic Loss

Finally, we also tried to predict the price ranges using ordinal classification. The BvS matches the label into a vector of length 10, then makes prediction using the designated loss function - we used the logistic function for our model. The training error rate is 0.631 and the test error rate is 0.629. With the loose error calculation, the training error is 0.268, and the test error is 0.276.

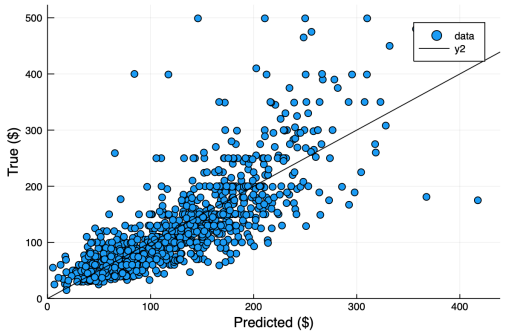
4 Discussion of results

Out of the different possibilities we've explored so far, it's surprising for us to find out quadratic loss produces the smallest error in the presence of obvious outliers. One potential reason we presume is that since high-price consist a very small portion in our data (a total of 6% of $250+ examples), quadratic loss did a better job at estimating normal price levels. Another thing that can't be neglected is that the results for L1 and Huber loss were estimated (proximal). We also found out that regularizers are not very helpful and that's because our models were not producing any crazy predictions based on the data we've carefully cleaned.

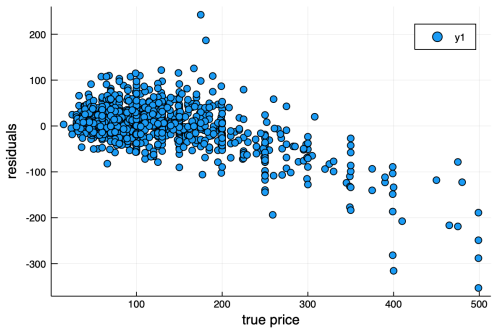
Onto the interpretation of results produced by our models, we report that the number of beds, bedrooms, accommodates, location and room type are the most powerful features judging by the magnitudes of coefficients, which aligns with what one would intuitively expect.

As for the performance of our models, after a 5-fold cross-validation on splitting data and a k=100 bootstrap of variance estimation, mean of RMSE is around $47 and bootstrap showed us a relatively stable prediction. However, we have to point out that the target has a mean of $127. As a result, it's hard to say it's a model of high accuracy, but we do not have prominent any overfitting issue. The underfitting is not too surprising for us, after all, the fact that prices on Airbnb are currently set solely by the hosts themselves can introduce many inevitable noises, such as individuals' different perception of prices. To take a closer look at our prediction (Figure 4b), it's easy to tell that we are underestimating high price listing. And

indeed if you look at MSE separately for listings under $250, the error went down to around 1200 while for high price it went up to 10000+. To deal with this issue, we initially thought choosing models and loss function that are insensitive to outliers would help but didn't. At this point, we have to assume that the issue is within the data itself – we are missing important indicative features for high price listings.
To incorporate what we've discussed in to practical settings, we would not recommend Airbnb to restrict hosts to only using predicted price, rather serve merely as a reference.



(a) Prediction vs. True



(b) Residuals

Figure 4

5 Weapon of Math Destruction and Fairness

5.1 Weapon of Math Destruction

A model is considered a "Weapon of Math Destruction" if its outcome is not easily measurable, can have negative consequences, or has self-fulfilling feedback loop [2].The model we trained has measurable outcome - namely, an estimate of the pricing of an Airbnb listing. This estimate most likely would not have major negative consequences: the intention of the model is to gain an insight of Airbnb pricing from the listings' quality. Finally, there is not any prominent self-fulfilling feedback loop. Most features, and especially features that have high predicting power, in our model reflect the quality of the listing itself, such as location and the number of accommodates. Theoretically speaking, those features should not get affected from a low nor high predicted price.

5.2 Fairness

It's clear that our model does not take in any features that directly relate to ethnic issues, such as hosts' races or gender. However, chances are some features could shadow those information in a way that we do not realize. As a result, we evaluate the fairness of our model in this section, using two metrics chosen from the data we have – "host_neighbourhood" and "host_has_profile_pic"[3].
An ideally fair prediction model in this setting should not have a bias towards whether the host is willing to reveal their physical appearance, which could signal their races and gender. Indeed, when we split the test data set based on this metric (one batch that has profile picture and one that does not), the mean values of the predicted price are almost identical, 125.58 vs.125.37.
Hosts' neighbourhood is another metric we should consider since it signals racial identity of the hosts to some extent. We look at the mean values of the predicted price again for different hosts' neighborhood,

[2]Definition from "Weapons of Math Destruction", Cathy O'Neil 2016
[3]To clarify, these two features are not included in our model. We only use them here as metrics for fairness testing

controlling for the accommodates of the listing. We report results from selected hosts' neighbourhoods that we think best represent ethnic issue (Table 2). Fortunately, we do not observe any consistent or significant trend from different hosts' neighbourhoods.

Host's neighbourhood	$ Price of listing (1-4 accommodate)	5-10 accommodates	10+ accommodates
Chinatown	60	91	160
Harlem	88	109	185
Upper east side	102	98	150

Table 2: Predicted price for different hosts' neighbourhodds

6 Future Improvement

6.1 Learning from Pictures

When considering other factors that highly influence a costumer's decision when choosing a listing, we realized that pictures of the listing play a very strong role. A listing with better visuals can easily have significantly higher prices than another listing with the exact same neighborhood, rating, number of beds, bedrooms, and bathrooms. Unfortunately our data is missing such info, and even if the pictures are included, it is out of the scope of this class to train a model that also learns from images. However, we believe that a model with such ability would definitely have a significantly better performance.

6.2 More effective embeddings using NLP Model

Intuitively speaking, one would expect text features such as summary and neighbourhood overview should be of great help to prediction. But one thing we found in our utilization of Tensorflow is that the contribution is rather trivial. We suspect it's most likely due to hosts striving to 'beautify' their property as much as possible and it could be hard to extract actual merits for NLP models. A future improvement is to find some specific words-embedding models that perform well under such settings.

7 Conclusion

In this project, we explored the NYC Airbnb data and made various attempts for a price prediction model, including a loss-minimizing model for exact price number, a binary classification model for high/low price and a multi-classification model for price ranges. Our final results on the performance of these models are a MSE of 2400, an error rate of 0.3 and a loose error rate of 0.27 respectively. Our models are easy to interpret, stable and do not have any evident fairness issue. However, our models still suffer from underfitting and we expect future work in images embedding or more complex models would alleviate the problem.