

---

# Neighborhood and Price Prediction for San Francisco Airbnb Listings

---

**Emily Tang**

Departments of Computer Science, Psychology  
Stanford University  
emjtang@stanford.edu

**Kunal Sangani**

Department of Economics  
Stanford University  
ksangani@stanford.edu

## 1 Introduction

Since its founding in 2008, Airbnb has become a symbol of the sharing economy and has changed the way people travel. As of 2015, the site advertises over 1.5 million listings in 34,000 cities around the world. In this study, we focus on listings in San Francisco, where the company first started.

Airbnb listings not only give us a window into how participants in the new sharing economy market their offerings, but also give us a unique insight into the characteristics of a city and its neighborhoods. San Francisco in particular is known for its diverse neighborhoods and cultural enclaves, and the city's Airbnb listings give us an opportunity to catalogue similarities and differences between neighborhoods.

Using the Inside Airbnb project's complete set of listings available in San Francisco, we develop classifiers able to predict a listing's neighborhood and its price. We choose these two outputs for their potential applications: Predicting a listing's neighborhood gives us insight into the cultural elements visible through text, image, and amenities that might link neighborhoods together, and could potentially be applied for a recommendation system (e.g. "If you enjoyed your stay in the Haight Ashbury, we recommend trying Alamo Square!"), and predicting the appropriate pricing of a listing, especially taking into account features unique to the sharing economy of how hosts describe their offering, may be helpful in internal pricing tools that Airbnb offers to its hosts.

The Inside Airbnb project provides data on the complete set of listings available in San Francisco. We input text features, image features, and other information associated with each listing to an SVM classifier to predict the (1) the neighborhood a listing is in, and (2) the listing's price (separated into discrete ranges). In the following sections, we explain the context for this work, describe our dataset and methods for feature extraction and classification, and present an analysis of our results.

## 2 Related Work

As far as we are aware from our literature search, there are no published studies that apply machine learning techniques to data from the Inside Airbnb project. (The project's data has largely been used for visualizations of listing types in each city.) Of a small number of papers that analyze Airbnb data, a relevant paper is by Lee et al. [3], in which the authors find that "social factors" like number of references, host response rate, and number of reviews, are even more important determinants of room booking than conventional factors like price and amenities offered. The importance of such social factors suggests that the ways in which hosts describe their listings and themselves may well be of large importance when determining price visitors are willing to pay.

We also draw from the literatures on price prediction and neighborhood detection. One notable example of machine learning used for neighborhood detection comes from a paper by Zhang et al. [1] that uses Foursquare user check-ins and semantic information about places to detect neighborhoods in cities. The study finds for three urban areas—New York, San Francisco, and London—that measuring homogeneity between places and the users frequenting them allows the authors to detect neighborhoods as clusters, and that the neighborhoods identified match qualitative descriptions of the cities neighborhoods. The objective of the neighborhood classification part of our project is similar, in that we use data from several listings that fall within neighborhoods to understand characteristics of each neighborhood and the diversity between them.

The literature on algorithmic pricing is more extensive. Hill [2] explains Airbnb's AI-based dynamic pricing tools, discussing how the original regression-based tool released in 2013, which used amenities of a listing and information about neighboring properties to predict the appropriate pricing, was refashioned into the company's most recent, reinforcement-learning based

tool, Aerosolve. The tool uses microneighborhoods to inform pricing and uses user booking as a feedback tool to improve future pricing tips. Our work is based on a freeze frame of data and so does not include any dynamic component, but the text and image features we explore may add to the accuracy of Airbnb’s price tools as well as provide the basis for additional tools on suggested listing marketing to property owners.

Finally, we draw from the literature on machine learning techniques. These references are covered in the methodology section.

### 3 Dataset

Our dataset of Airbnb listings, available through the “Inside Airbnb” project [4], includes 7,029 listings posted in San Francisco as of November 2, 2015. Each listing contains text information—a name, general description, a host bio, and a description of the space offered, neighborhood, and local transit—as well as an image thumbnail (72 pixels per inch, usually 275x183) and several other fields describing the amenities offered, number of bed- and bathrooms, type of property, type of bed, square-footage of the space, etc. The per-night price of each listing was also included, as well as the neighborhood of the listing (of 37 possible SF neighborhoods) and a cumulative review score.



ID 25463: “*Modern Zen in the Lower Haight*”  
 “Located in one of San Francisco’s most bohemian and central neighborhoods, I offer a large sunny room with a very comfortable queen bed and shared bathroom. My home is one block from the trendy shops, restaurants, cafes, art galleries and salons of the Lower Haight.” (\$115.00 per night)  
 {property\_type: ‘house’, room\_type: ‘private room’, bed\_type: ‘real bed’, bathrooms: ‘1’}

Figure 1: An example of an Airbnb listing

In order to reduce the burden on our multiclass classifier for neighborhood prediction, we first pre-process the data to exclude listings that belonged to neighborhoods containing fewer than 70 listings (1% of the dataset). This reduces the number of listings to 6764 and the number of neighborhoods to 27. The remaining listings are split into train, dev, and test sets (80:10:10).

### 4 Feature Extraction & Methods

For each listing, we extract five sets of features: (1) listing information features, (2) multinomial bag of words features, (3) multinomial word class features, (4) text sentiment features, and (5) visual features.

Listing information features include the property type (apartment, condo, bed & breakfast, dorm, house, etc.), the bed type (futon, airbed, real bed, etc.), the host’s cancellation policy, and the number of beds, bedrooms, bathrooms, and guests accommodated.

To create our bag of words features, we aggregate each listing’s name, summary, space, description, experiences offered, notes, and host bio fields. We then use the NLTK package’s PorterStemmer to stem words to their root and choose the 1,000 stems that occur across the most entries in the test set. After filtering out all stop words and all neighborhood words (to prevent our neighborhood classifier from simply using mentions of neighborhoods for prediction), we arrive at a list of 973 stems: the multinomial feature vectors were then calculated as the frequencies of each stem in each listing’s text, with the sum of the features normalized to sum to one.

A closer look at the dataset reveals that words like comfortable and cozy might be used interchangeably between listings. In order to address this, we handpick 9 word classes: people, nightlife, activities, style, accessibility, culture, nature, amenities, and comfort. For instance, the “style” word class includes the stems of the words modern, brand, victorian, style, decor, gorgeous, marble, elegant, boutique, and trendy. Multinomial word class features are counts of the number of words belonging to each of the nine word classes in each listing, with the sum of the features normalized to sum to one. For sentiment features, we use the TextBlob package, which calculates the polarity of a segment of text by averaging the polarity of each word in the text included in the package’s lexicon.

Finally, we download all listing images and extract visual features using the standard bag of words model in computer vision. We randomly sample 100 images to create a dictionary of 1000 words. To create the dictionary, we extract Speeded Up Robust Features (SURF) descriptors from the 100 images using OpenCV. SURF is a performant scale- and rotation-invariant interest point detector and descriptor used to find similarities between images [5]. We use K-means clustering to form 1000 clusters, or visual words for the dictionary. Using this visual word dictionary, we create feature vectors for each listing by extracting SURF descriptors from the listings image. For each descriptor, we select the closest cluster in the dictionary. In brief, the visual feature vector contains the frequency of each visual word in the listing image, and is normalized.

We input the features listed above into a Support Vector Machine (SVM) with a linear kernel, implemented using sklearn’s SVM package. We discretize our listing prices into two buckets separated at the median (166.5), and develop two classifiers to (1) predict the range in which a listing’s price falls and (2) predict the listing’s neighborhood. (Since the number of listings per neighborhood is not identical across neighborhoods, we use sklearn svm’s ‘balanced’ class weight setting.) Using  $l_1$

regularization, our SVM optimizes the objective [6]

$$\begin{aligned} \min_{\gamma, w, b} \quad & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t.} \quad & y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i, \quad i = 1, \dots, m \\ & \xi_i \geq 0, \quad i = 1, \dots, m \end{aligned}$$

Note that the parameter  $C$  adjusts the sensitivity of the classifier to outliers. In order to tune our parameter for  $C$ , we use gridsearch and vary  $C$  from 0.001 to 100,000. The value of  $C$  for which performance on the dev set is highest is then used for the final classifier, which is trained on the train set and tested on our held-out test set.

We plot learning curves for each classifier, and note that our model has high variance, indicating an overfitting problem. To account for this problem, we use feature selection, specifically sklearn’s Recursive Feature Elimination (RFE). RFE is an algorithm that recursively considers smaller subsets of all features, by removing features with the smallest weights after training on a set of features for each iteration. Since RFE requires us to choose the number of features we want to ultimately use, we conduct an experiment varying the number of features to select the best  $K$ , which is shown in our results below.

To better understand our classifier, we run ablative analysis, by removing one component at a time, building our classifier using the train set, and seeing how our classifier accuracy changes on the dev set. In addition, we run feature selection on only the bag of words text features for both neighborhood and price predictions, because we are interested in comparing and analyzing the words between neighborhoods and prices.

## 5 Results and Discussion

### 5.1 SVM Performance for Price and Neighborhood Prediction

After tuning the  $C$  parameters (example for price prediction model shown in Table 1) on the dev set, we test our models on a held-out test set. The train and test accuracies, precision, and recall for each set of features, as well as for the entire system, are presented in Tables 2 and 4 (for price prediction and neighborhood prediction, respectively). We note that the price prediction model achieves a test accuracy of 81.2% using all features, which is significantly higher than the chance accuracy of 50%. For the neighborhood prediction model, we find that, interestingly, the highest accuracy achieved is 42.2% using only the bag of words text features. In neighborhood prediction, the chance accuracy baseline is 8%, which can be achieved by simply predicting the neighborhood with the greatest number of examples.

The receiver operating characteristic (ROC) curve for price prediction is included in Fig. 2. The AUC reported is 0.883, which suggests that the classifier performs quite well in differentiating between true and false positives. (We also binarize our labels for neighborhood prediction and generate 26 ROC curves—one for each neighborhood—but exclude this figure for sake of space. In neighborhood prediction, the AUCs average 0.84 across neighborhoods.) We also inspect confusion matrices for both the price and neighborhood classifiers. In particular, use our confusion matrix from the neighborhood classifier to generate a heat map of prediction accuracy by neighborhood, as seen in Fig. 8. Note that prediction accuracies tend to be highest for the most central neighborhoods (including the Mission, South of Market, and Castro/Upper Market). This makes sense, since those neighborhoods are best known for their unique characteristics, and those characteristics might be best detected in the features we extract.

### 5.2 Learning Curves

Since the train accuracies are incredibly high for neighborhood predictions, we plot the learning curves for both models, to determine whether our model is overfitting. In Fig. 4, we see that gap between the training scores and the cross-validation scores for price prediction is quite large, indicating a problem of high variance and thus overfitting. In Fig. 6, we note that training scores for predicting neighborhoods has maxed out at nearly 1 and are extremely far from the cross-validation scores, which also points to an issue with overfitting.

### 5.3 Feature Selection

There are two solutions to high variance: finding more training examples or reducing the number of features. Since our dataset was limited to listings in San Francisco, finding more training examples was not possible. So, to help mediate overfitting, we use feature selection, specifically sklearn’s RFE algorithm, to reduce the number of features. Because recursive feature elimination (RFE) requires us to select the end  $k$  number of features, we conduct an experiment to determine the best  $k$ . We vary the number of features, run RFE on our models for each number of features, and test the models on our dev set. The resulting train and dev accuracies from this experiment are plotted in Fig. 3 and 7 (for price and neighborhood prediction, respectively).

From Fig. 3, we see that selecting 800 features for RFE gives us the best dev accuracy for prices, and that selecting 150 features does so for neighborhoods. We build our models, run RFE using the best number of features, and plot the learning curves again to determine whether the overfitting problems for neighborhoods and prices have been reduced. For our price prediction model, when comparing the two learning curves in Fig. 4 and 5, we see that the train accuracy has decreased, and approaches the dev accuracy curve. In addition, we see that the dev accuracy after feature selection remains very close to 81%. Putting these together, we conclude that feature selection slightly alleviate overfitting of our price prediction model.

Comparing the two learning curves for our neighborhood prediction model, we find similar improvements. After feature selection on 150 components, the training accuracy is no longer maxed out at 1, and has decreased to around 0.8. (We exclude our learning curve after running RFE for sake of space.) In addition, we find that feature selection actually improves our accuracy slightly from 42.3% to 43.1%. Finally, we notice that the cross validation score increases as number of training samples increases. This upward trend suggests that, if more training examples were available, they may have helped improve our classifier.

## 5.4 Ablative Analysis

To determine which component of our five features accounts for the most improvement in our SVM models, we run ablative analysis, and remove one component at a time to see how our classifier accuracy on the dev set changes. For price prediction, we see from Table 3 that the listing information features are the most significant factor. This makes sense because housing prices are very much determined by the property type and size, the number of bedrooms and bathrooms, and so on. For neighborhood prediction, we see from Table 5 that the Bag of Words text features are the most important component. This suggests that neighborhoods are quite diverse and are hard to capture from the listing information, sentiment, or images, but that the richness of the word choice used to describe the surroundings of the neighborhoods (such as vibrant, bars, nightlife, museums) is best at differentiating between neighborhoods.

## 5.5 Comparing Prices and Neighborhoods

Interested in seeing how word importance differed across price and neighborhood models, we run feature selection on just the bag of words features. We discover that the word features that remain after RFE for price prediction are actually a subset of the word features that remain for neighborhood prediction. Most of the words important in both prices and neighborhoods describe the house and its interior, such as patio, remodel, luxurious, clean, modern, and cozy.

We can intuitively see that these words, though they describe an individual listing home, could also be applied to neighborhoods that include housing units of similar style and age. Thus, the words that describe a listing can also distinguish one neighborhood from another. On the other hand, the words found to be important in only neighborhood prediction have a larger scope and are related to culture and larger surroundings. These words include vibrant, cafe, church, peace, bustling, culture, museum, and history.

## 6 Conclusion and Future Work

Our analysis of the Inside Airbnb project’s listings for San Francisco shows that we can successfully predict neighborhood and price range using an array of features extracted from listings. While our classifiers initially suffer from high variance, our feature selection at least partially remedies this overfitting. Both classifiers we develop perform significantly above baseline, and suggest that listing information, text features, and more can be harnessed effectively for predicting neighborhood and price.

With more time and computational resources, we hope to develop our model for price prediction. The results we’ve presented here split prices into two discrete ranges. (In our work on this project, we also developed a classifier to predict price across four buckets, split at the data’s quartiles, and found encouraging accuracy rates.) To provide meaningful price tools to Airbnb users, we would need to discretize our price range into smaller buckets to offer targeted and narrow price range suggestions.

Our final goal in neighborhood prediction is to provide a recommendation system to Airbnb users as mentioned in this paper’s introduction. Our work in this paper established that listing text is rich enough to allow us to predict in which neighborhood a listing belongs. In future work, we hope to take advantage of this diversity of word choice to cluster neighborhoods with similar cultures or characteristics together. We may even be able to do this across cities, as the Inside Airbnb project provides data for several cities around the world, and catalogue both cultural differences across cities and neighborhoods from city to city that are somewhat similar.

Finally, we notice in our analysis that there is significant overlap between the features that predict neighborhood and price. In future work, it would be interesting to understand the correlations between neighborhoods and prices more deeply. Upscale neighborhoods naturally have higher priced listings, and tracing out the qualities of neighborhoods that encourage consumers to pay higher prices may be an important application for real estate / urban planning.

Price Prediction

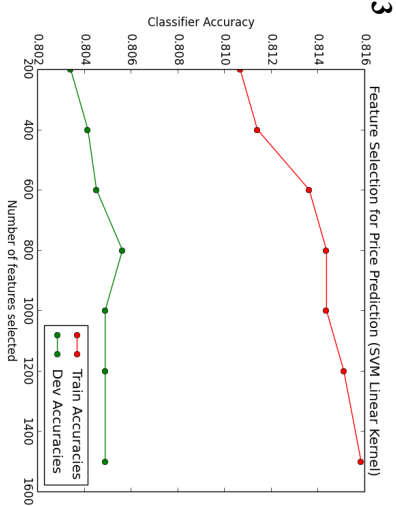
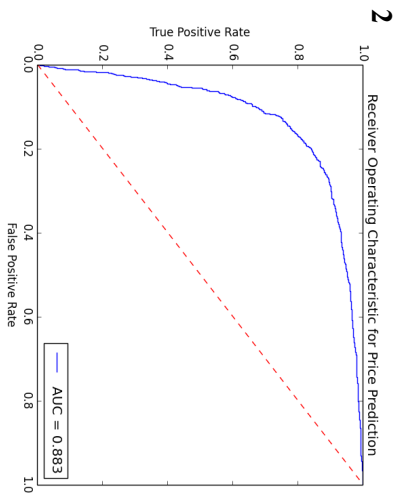
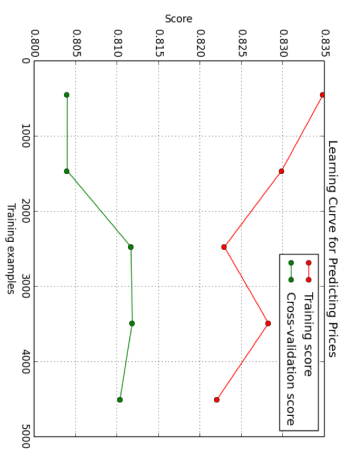


Figure 2. ROC curve for price prediction (AUC = 0.883)

Figure 3. Results from RFE for price prediction

Figure 4. Learning curve for price prediction

Figure 5. Learning curve for price prediction after running RFE using 800 features



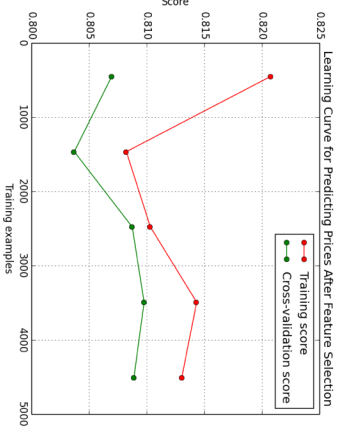
4

Table 2. Train and test accuracies, precision, and recall of SVM for price prediction

Features used		Train Acc.	Test Acc.	Precision	Recall
Sentiment features		0.5533	0.5499	0.5845	0.3498
Word classes		0.6413	0.6353	0.6316	0.6517
Listing information		0.8040	0.8101	0.7683	0.8886
Bag of words text feats.		0.7700	0.7350	0.6901	0.8546
Visual features		0.5666	0.5185	0.5104	0.9417
All features		0.8173	0.8115	0.7700	0.8893

Table 3. Ablative analysis for price prediction

Component	Train Acc.	Test Acc.
Overall system	0.8173	0.8052
Sentiment	0.8084	0.8027
Word classes	0.8151	0.8045
Listing information	0.8129	0.8038
Bag of words text feats.	0.7848	0.7299



5

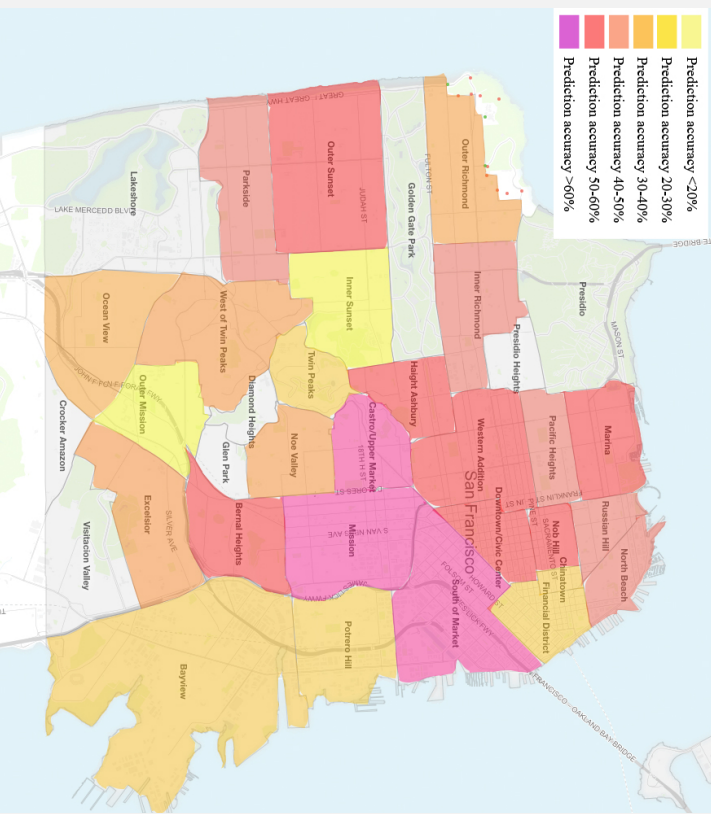
Table 4. Train and test accuracies, precision, and recall of SVM for neighborhood prediction

Features used	Train Acc.	Test Acc.	Precision	Recall
Sentiment features	0.0695	0.0301	0.0564	0.0310
Word classes	0.1302	0.0636	0.1112	0.0636
Listing information	0.1960	0.1027	0.1425	0.1027
Bag of words text feats.	0.9948	0.4228	0.4235	0.4228
Visual features	0.7115	0.0835	0.1329	0.0835
All features	0.9993	0.3119	0.2894	0.3119

Table 5. Ablative analysis for price prediction

Component	Train Acc.	Test Acc.
Overall system	0.9993	0.3156
Sentiment	0.9993	0.3370
Word classes	0.9993	0.3533
Listing information	0.9985	0.3917
Bag of words text feats.	0.7115	0.0928

Figure 8. Heat map of San Francisco neighborhoods by prediction accuracy



Neighborhood Prediction

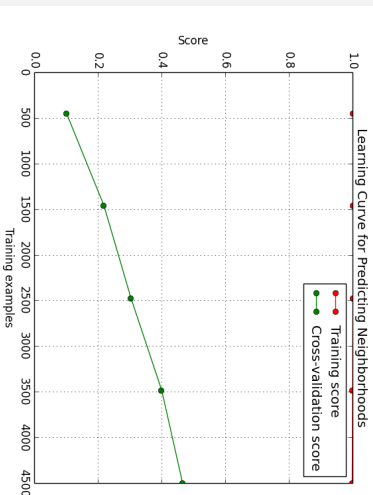
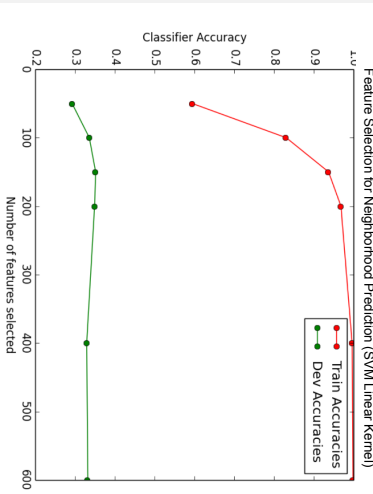


Figure 7. Results from RFE for neighborhood prediction



## 7 References

- [1] Zhang, A. X., Noulas, A., Scellato, S., & Mascolo, C. (2013, September). Hoodsquare: Modeling and recommending neighborhoods in location-based social networks. In Social Computing (SocialCom), 2013 International Conference on (pp. 69-74). IEEE. <http://arxiv.org/pdf/1308.3657.pdf>
- [2] Hill, D. (2015). How much is your spare room worth?. *Spectrum, IEEE*, 52(9), 32-58.
- [3] Lee, D., Hyun, W., Ryu, J., Lee, W. J., Rhee, W., & Suh, B. (2015, February). An Analysis of Social Features Associated with Room Sales of Airbnb. In Proceedings of the 18th ACM Conference Companion on Computer Supported Cooperative Work & Social Computing (pp. 219-222). ACM.
- [4] Dataset available here: <http://insideairbnb.com/get-the-data.html>
- [5] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In Computer visionECCV 2006 (pp. 404-417). Springer Berlin Heidelberg.
- [6] Ng, Andrew. (2015). 229 Lecture Notes: Support Vector Machines (Notes 3). (pg. 19).