# NYC Condo Price Estimation Using NYC Open Data

Hari Arul
Andres Morales

## Introduction

This project explores the structure of the New York City housing market by predicting the price of condominiums in New York City using the publicly available NYC Open Data dataset. Estimating property prices is a well-known problem in statistics and machine learning. Many different parametric and non-parametric models have been shown to achieve good levels of accuracy on certain datasets. We attempt to use this novel dataset to tailor brand new features and gain a high accuracy in prediction. Furthermore, by harnessing the wide variety of features given by the dataset, we hope to gain insight into what groups of features have the highest impact on generalization error.

This paper explores the procedure we took in learning about a new dataset by using clustering, normalization, and PCA. We then discuss the impact of using different learning algorithms on the accuracy of prediction. To do so we used Multinomial Naïve Bayes, SVM, SVR, and Locally Weighted Linear Regression on our new features. We saw that our features were not as predictive as the UCI Boston Housing dataset features given our results when we ran these same algorithms on that dataset. This demonstrated that we could improve upon our features, and so we moved on to using feature selection and adding location-based features to better our dataset and our prediction accuracy. Given these new techniques our results improved considerably and did a good job of predicting housing prices.

## NYC Open Data

NYC Open Data is a 2011 project by New York City that has made available troves of data generated by public agencies about the city. The data is updated as new information comes in, and is provided in a machine-readable format for people (like us) to examine. This data includes information as varied as federal stimulus data in NYC regions, housing data split by boroughs, and wifi hotspot locations in the city.

NYC BigApps is a competition for developers sponsored by New York City that intends to find the best applications that utilize NYC Open Data [1]. We are applying to this competition with a housing price prediction app that utilizes the machine learning described in this paper for the backend price calculations. For the year 2011-2012, there were data points given for 4950 condominiums in the Manhattan Area.

## Previous Research

Previous research in this problem area has focused on examining housing data, and trying to predict prices using machine learning techniques. Almost all of the articles use the UCI Machine Learning Repository Housing dataset comprising housing data from Boston. The features include crime data by town, house-specific information, and socioeconomic and education data, and the predicted value was the value of homes in thousands of dollars.

Typically, these papers try to combine qualities about different learning models to come up with an optimal model to train the housing data, and then show how it does a good job of prediction. For example, a Yann LeCun's 2007 paper combines both a parametric and non-parametric model to train the final model with a form of EM [2]. Specifically, the authors make the assumption that house prices are a function of both the features of the house (the parameterized model) and the suitability of the neighborhood (non-parametrically modeled). This model ended up being better at predicting housing prices than pure parametric or non-parametric algorithms, showing that new models can better represent housing prices.

The same NYU professors wrote another paper that added geographical information to help reduce housing price prediction error (an approach which we eventually take with our NYC data) [3]. Lastly, SVM's and SVR's have been used to predict housing prices [4,5]. Using the Boston data, it was shown that as long as information was known about the surrounding Boston cities and towns, the value of housing could be fairly accurately determined. From this previous research, we know there are multiple ways we can train our data, and are hoping to find the best prediction scenario.

## Initial Features and Dimensionality

After gathering housing market information from the NYC Big Data database, we decided to begin solely with features directly related to the price of the house. Our initial feature vector is in 10 dimensions, and includes the Borough, Block, Neighborhood, Building Classification, Total Units, Year Built. Estimated Gross Income, Gross Income per square foot, Estimated Expense, Expense per square foot, and Operating Income of the property and/or property owner. We used 10-fold cross-validation to generate our training and testing data.

Initially, we did not normalize our feature data, and when we attempted to use PCA to find the principal components of the data, we found that the first component explained 100% of our data, and basically was comprised of one feature. After normalizing the data, PCA gave 7 principal components which explained 98.3% of the data (see Figure 1). The first principal component placed high coefficients on the features corresponding to Building Classification, Year Built, Gross Income, and Expense per square foot. This makes sense because each of these features provide different types of data, and there is a lot of overlap between these features and the ones which were not weighted as heavily, such as Gross Income per square foot and Expense per square foot. This does not explain which features will help predict housing prices, but it did help give us a sense of how representative our features were.

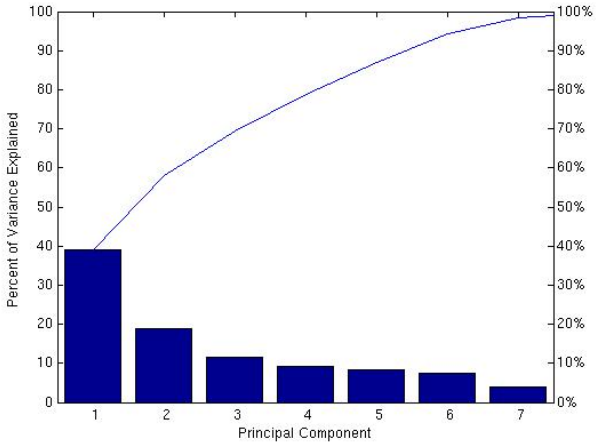


Figure 1: % of Data Explained by Principal Components

## Multinomial Naïve Bayes

In previous research, multinomial Naïve Bayes was used in a feature selection paper on the UCI Boston Housing dataset, where the problem became a binary classification problem with the y-values were separated according to the mean value of the target [6]. On our dataset, we ended up classifying our prices per square foot into the proper bucket 71.105% of the time. We used k-means to find our two buckets, and they roughly split the training into low-priced houses (<$200 per square foot) and high-priced houses (>$200).

However, we found binary classification of housing prices an inaccurate way of actually predicting what the price of a house should be, given it only really selects whether our house is cheap or expensive. So, as our baseline algorithmic test, we decided to run Naïve Bayes across N buckets, where each bucket represents a price differential of 30% (e.g. if the min price of a bucket is $100, the max price will be $130). We did the bucketing this way to ensure that whenever we correctly identified a bucket, our estimated price would be at most 15% away from the median price of that bucket. All the previous research that we looked at attempted to specify the percentage of houses that were correctly predicted within 15% of the actual price, and when we implemented classification, this was the best way of adhering to these error predictions. Since we used Naïve Bayes, we also had to discretize our input feature space.

With n buckets, for a given feature j, we bucketed our feature as follows:

$$s_j = \frac{max_i(x_j^{(i)}) - min_i(x_j^{(i)})}{n-1}$$

$$b_j^{(i)} = \frac{floor(x_j^{(i)} - min_i(x_j^{(i)}))}{s_j}$$

where $s_j$ is the size of a bucket and $b_j$ is the bucket for a training example j. Given a different number of input buckets, and our output buckets as specified above, Table 1 illustrates the classification performance of this baseline test.

| Number of input buckets | Accuracy |
|---|---|
| 8 | 18.64% |
| 16 | 25.58% |
| 32 | 22.84% |
| 64 | 23.37% |
| 128 | 20.73% |

Table 1: Naïve Bayes Results

When we run the same algorithm on the UCI Boston Housing data that was used in research papers with 16 input feature buckets, our accuracy was 31.68%. As our future algorithms show, we can greatly improve performance from Naïve Bayes.

# Support Vectors

## SVM

Given we were already working on classification with Naïve Bayes, our next step was using Support Vector Machines to try and better classify our test data into proper buckets. SVM is a learning algorithm developed by Vapnik in order to obtain an optimal margin classifier in a high-dimensional feature space [5]. SVMs have been used in the past for predicting housing prices, so we thought it would be useful to try on our dataset [4]. We looked at both linear kernels as well as Gaussian radial-basis kernels, whose value depends on the distances between input vectors.

| **Bucket Size:** | **Accuracy** | |
|---|---|---|
| | Linear Kernel | Radial Basis Kernel |
| 15% | 35.105% | 33.895% |
| 30% | 47.571% | 41.263% |
| 60% | 57.053% | 53.053% |

Table 2: SVM Accuracy (no location data nor feature selection)

As we can see, increasing our bucket size naturally helps improve our classification ability. However, compared to the results in LeCun et al, our SVM is not giving nearly as good performance. As the rest of this paper demonstrates, we can still improve on this SVM classification prediction through a combination of feature selection and location data.

## SVR

Support Vector Regression is a regression technique developed in 1996 by Drucker and Vapnik et al which extends upon Vapnik's SVM concept to do regression analysis [5]. Similar to SVM's for classification, SVR determines the support vectors from the Lagrangian of the loss function described in the paper, which is slightly different from the original SVM function. We wanted to use SVR because not only has it been used for predicting Boston housing prices, but using regression can give us a better indication

of what our overall generalization error is, because we do not have to deal with bucketing. Our SVR results, without including any location data, nor doing any feature selection, looks as follows:

| | |
|---|---|
| Mean Squared Error | 3,993.40 |
| Average Error ($ per square foot) | 48.07 |
| Average Housing Price ($ per square foot) | 125.69 |
| % Error | 38.2% |

Table 3: SVR Results (no location data nor feature selection)

We saw that using SVR, our results averaged about $48 per square foot off from the actual price of the house. Given the average housing price of $125.69 per square foot for the condominiums given, this lead to a 38.2% average error for the condominiums tested. The percentage of homes tested where our predicted price was within 15% of the actual price was 36.11%. We will be able to improve on this data when we do feature selection, but our results will still not be as good as running SVR on a dataset such as the UCI Housing set.

## Location Data and Feature Selection

One of the raw features provided by NYC Open Data that we did not initially include was the address of the condominiums. However, after reading a set of papers highlighting the importance of location in predicting housing prices, either as a separate model, or as an added feature, we decided to find a way to add it in [2,3,5]. Using GPS data for each of the addresses provided by Yahoo!, we used longitude and latitude as additional features. We condensed both these features into one by calculating their distance to the origin.

Not only did we incorporate location data to obtain better results, but we used feature selection in order to build up our best set of features using forward search without including extraneous ones that only worsened our predictive ability. As a result, we ended up getting the minimal percent error by ignoring the features Block, Neighborhood, and Total Units (see Figure 4). It makes sense that we don't want to add Block and Neighborhood because we are already adding in longitude and latitude information for our location data, and this would only serve to potentially increase overfitting. Total Units ends up being extraneous as well because we decided to predict over price per square foot, which makes the number of units for the condominium not as worthwhile as if it was total price.

With our optimal seven features (including longitude-latitude), running SVR gives us: (see Table 4):

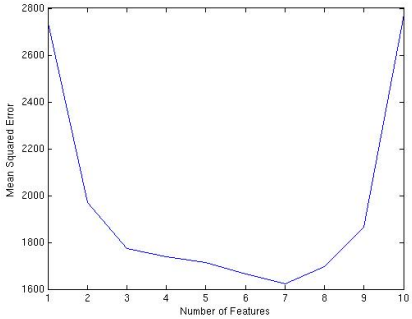


Figure 2: Number of Features vs. MSE

| | |
|---|---|
| Mean Squared Error | 1,697.33 |
| Average Error ($ per square foot) | 27.46 |
| Average Housing Price ($ per square foot) | 125.69 |
| % Error | 21.8% |

Table 4: SVR Results with Feature Selection

These results are better than what we previously had, showing the importance of having the right features when training data. 49.1% of our predicted prices were within 15% of the actual price per square foot.

When we ran SVR on our Boston dataset with feature selection, we saw our results improve (on the basis that the UCI dataset has better and more predictive features for housing prices). Specifically, our average percent error is 17.5%, and the predicted MSE is $13.77 (in thousands), which is in line with the original SVR paper that tested on the Boston data. 54.4% fo our predicted prices were within 15% of the actual price of the data (and 68% of our data was within 20%), which is in line but slightly worse than Yann LeCun's paper which used a specific Los Angeles housing data [2]. Since our algorithms did better

on those datasets, it made us realize that even though our predictions were ok, having better and more predictive features for NYC Open Data would yield much better results.

## Locally Weighted Linear Regression

Now that we had location data as part of our feature set, we thought it would be interesting to run locally weighted linear regression, where 'local' is used in a literal and figurative sense. We weight each of our test inputs by how close its GPS location (i.e. the location feature), giving closer training points a higher weight. So we just run locally weighted regression using one feature for our weighting, and then using our full set of features for our predicted price estimate. A scatterplot of our predicted and actual results against the GPS distance from the origin is plotted here.

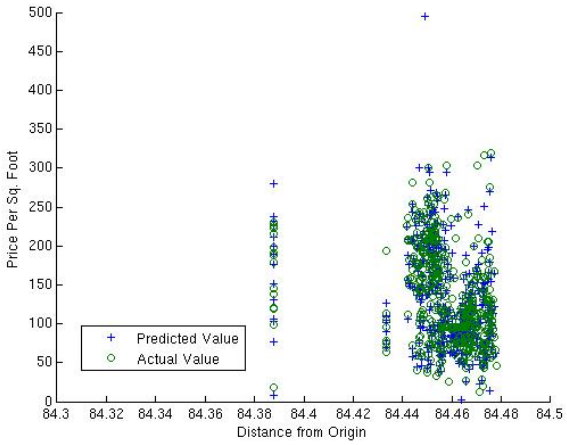


Figure 3: Location vs. Predicted / Actual Housing Prices

Our average error using LWLR was 24.97%, so it is fairly reasonable to assume that location (even as granular as our data) plays a big part in determining housing prices, given these results are in line with SVM and SVR on the dataset.

## Conclusion

The results from our data did reasonably well on our training set. However, we don't think our results are good enough to report on or to fully accurately make predictions in an application. Given that our more complicated tests ran well on the more fine-tuned Boston dataset, we are fairly confident that an added feature set than the one we have, along with feature selection, would lead to significantly better prediction results. Data we could include, along with our current house specific data and location data, are crime data, school district data, and socioeconomic data; all of which was used in the other datasets but was not readily available. We believe thus that our algorithms will scale well with better data and will allow us to confidently make predictions in the market.

## References

[1] http://2011.nycbigapps.com/
[2] Chopra, Sumit, et al. "Discovering the Hidden Structure of House Prices with a Non-Parametric Latent Manifold Model." *ACM* (2007):
[3] Sumit, Chopra, Yann LeCun, and Andrew Caplin. "Machine Learning and the Spatial Structure of House Prices and Housing Returns." *Working Paper - SSRN* (2008):
[4] Wang, Chaoyang, Yanfeng Sun, and Yanchun Liang. "An Improved SVM Based on Similarity Metric " *Journal of Universal Computer Science* 13.10 (2007):

[5] Drucker, Harris, and Vladimir Vapnik. "Support Vector Regression Machines." *Advances in Neural Information Processing Systems* (1996):
[6] Radivojac, Predrag, Zoran Obradovic, and A. Keith Dunker. "Feature Selection Filters Based on the Permutation Test." *ECML* (2004):