NYC Real Estate Price Prediction

Ben Jakubowski and Haonan Zhou

***Abstract*—We built supervised learning models to predict NYC property prices using data scraped from StreetEasy's website. Our best model for this problem was XGboost with hyper-parameters optimized using grid search and cross-validation. The final model had an out-of-sample median absolute percentage error of 5%, which is comparable to the results of Zillow's real estate price prediction model.**

***Keywords*—NYC Real Estate, StreetEasy, XGBoost**

I. Introduction

The value of real estate units are most often understood by examining sales of real estate comps (comparables). These comps are used by property owners to assess the market values of their existing units and property developers to decide whether it is profitable to construct new units. Very often, much of the process of determining real estate comps are conducted in a qualitative and haphazard method that only uses basic information such as price per square foot of recent sales in similar neighborhoods. Moreover, often this information is gather informally (through professional networks) as a limited sample of known recent sales [1].

In this project, we used an automated, systematic, and extensible machine learning approach to determine real estate values. We developed a prediction model of NYC real estate prices using supervised learning models instead of relying on expert intuition and a priori definitions of comparability. We trained various models using features engineered from publicly available property sales and listings records, and we evaluated the models by examining various percentiles of out-of-sample absolute percent error in predicted price. This objective is also used by the real estate website Zillow to evaluate the accuracy of their price predictions, and thus it can be viewed as the industry standard evaluation metric.

II. Data Collection

When initially approaching this predictive modeling problem (predicting real estate prices in NYC), we began by exploring civic datasets related to NYC real estate; these datasets included the Department of Finance Annualized Sales data, which provides records of property sales for each year, and the PLUTO dataset, which provides information about tax lots (including features describing building age and assessed tax value). Unfortunately, these data were very feature poor; when predicting property values, current modeling approaches often use features like the number of beds/baths and the square footage [1]. Since civic datasets don't include these or other property-level features, we sought out alternative datasets.

Ultimately, we decided to make our primary dataset sale pages from StreetEasy, an online real estate database with a wealth of NYC property listings and associated features. We used the web-crawling tool Scrapy in conjunction with the Tor network to download the HTML source of all historical listings available on the StreetEasy website by systematically querying all sales pages (fetching pages indexed by integers from 1-1400000). This yielded 516k entries of properties listings dating from 1997 to the present day (November 2016). We proceeded to use the BeautifulSoup python package to extract data from various HTML tags on the sales page based on our knowledge of the web-page structure. The set of raw features that we obtained from the StreetEasy property listings include the following:

- Sale or final listing price
- Sale or delisting date
- Final status of entry (sold, delisted, etc)
- Property size (square-footage, number of bed and baths)
- Property type (condo, co-op, house, etc)
- Monthly costs (taxes, maintenance, etc)
- Building data (amenities, number of units, age, etc)
- Distance to public transportation (MTA, PATH, etc)
- Textual description of listing (human written)
- GPS coordinates of property

After the raw features were extracted, we chose to discard records of sales outside the five boroughs (a large number of sales were reported in New Jersey and the greater NYC metro region). Then we immediately proceeded to split the sales records into 80-20 train-test data sets ensure our final model evaluation provided an accurate estimate of generalization error. All data points collected from StreetEasy contain a final sale or listing price, which is required for the construction the target variable in our supervised learning models. Nevertheless, we still performed some minimal data filtering in order to obtain a data set that is more suitable for our modeling goals. We removed all records without an associated GPS coordinate because these are mostly spurious or misnamed listings with incomplete or incorrect address information. We also removed entries with price outside of the 5th to 95th percentile price band of the training set, as these outliers have erroneous price or represent special properties with different data generating processes (i.e. the market for sales of buildings is anticipated to operate differently than the market for sales of condos). The size of the training and test sets are reduced to 286k and 71k records respectively after the data is filtered.

III. Feature Engineering

The raw data extracted from Streeteasy is abundant in missing entries. We used standard techniques to eliminate these NaN entries before using the data for our models:

- Filled missing categorical features with dummy label
- Filled missing numerical features with mean value

In addition, we transformed raw features and constructed new features using various techniques:

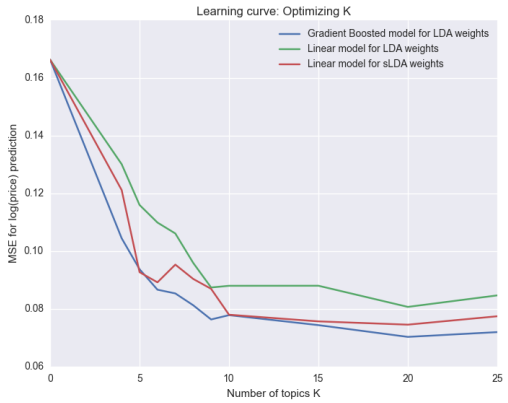

Fig. 1: Validation set MSE for sLDA and LDA topic models for grid search over K

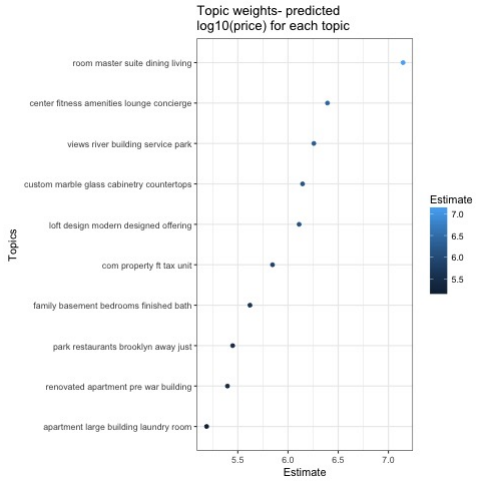

Fig. 2: Topic weights are learned along with topics in sLDA, and the top five words for each topic and the topic's associated weight are shown.

- Used one-hot-encoding to represent categorical features
- Inferred neighborhood, borough, and community district from property GPS coordinates and constructed new geographical indicators from this analysis
- Constructed comp features for listing price and unit size from comparable units in (i) the same building and (ii) the same neighborhood
- Modeled listings description text field using SLDA to generate topic weights as features

LDA Analysis

While the scraped HTML pages provided many features that were useable with minimal processing (ex: number of bedrooms), one of the major components of the sale pages was a human written property description. After reviewing a number of these descriptions, we hypothesized they reflected several latent topics with semantics related to property aesthetics and amenities (for example, luxury condos in large buildings versus prewar buildings in quiet residential neighborhoods). Thus, we proceeded to

- Further split the training data into training and validation subsets
- Run a grid search over K in [0,4,5,6,7,8,9,10,15,20,25] topics, for two types of topic models: Latent Dirichlet Allocation [2] and supervised Latent Dirichlet Allocation [3]

The learning curves for the grid search are shown in Fig. 1. Based on these results, we chose to use sLDA with $K = 10$ topics. This was based on the observations that (i) additional topics did not improve validation set performance, and (ii) linear modeling using sLDA topics performed as well as non-linear modeling using LDA topics, so non-linear modeling with sLDA topics would perform at least as well. To further validate our use of this topic representation, we compared the learned topic weights to the top 5 words for each topic (Fig. 2), and observed higher topic weights (i.e. higher predicted price) clearly were associated with words related to higher-end properties. For reference, note the topic weights can be interpreted as the predicted log(price) for a property with a one-hot topic vector.

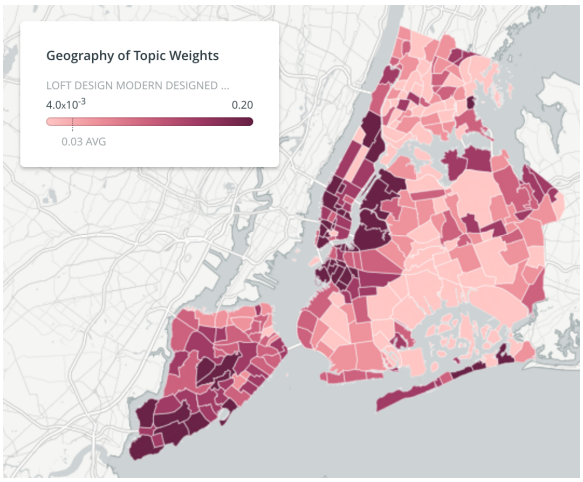

Fig. 3: Properties were aggregated by neighborhood, and median topic weights for each topic were mapped. This map shows median topic weights for topic 7: designed loft properties.

Additionally, we also aggregated properties by neighborhood (specific neighborhood tabulation area or NTA, a geographic division constructed by the city) and mapped NTA-level median topic weights for each topic. As shown in the sample map (Fig. 3), the geographic clustering of topics by neighborhood further support the use of the sLDA topics in our predictive model.

IV. Modeling

A. Linear modeling

Our baseline model for the regression problem was a standard MSE-loss elastic-net model trained using cross-validation to determine optimal hyper-parameters. From the

cross-validation curve in Fig. 4, we see that the model never enters a region of high variance even as the regularization strength is reduced significantly. This suggests that the linear model is highly biased, and a more expressive non-linear model would be more suitable for the price prediction problem.

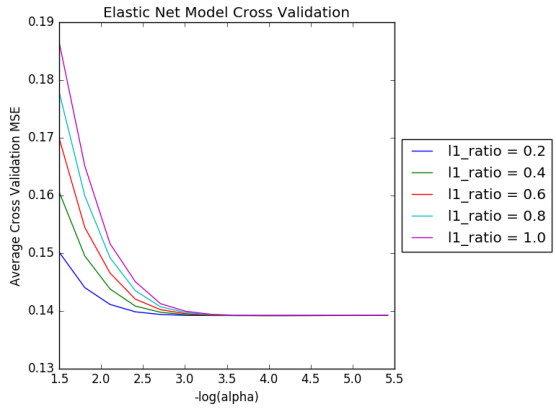

Fig. 4: Cross validation MSE of elastic-net model with various l1/l2 regularization ratio and regularization strength (alpha).

We also tried to fit the elastic-net model for sub-regions of NYC (boroughs, community districts, and neighborhoods). Although this yielded better evaluation scores, the improvement is not significant compared to the city-wide model. Table 1. summarizes the model evaluation results for the city-wide and four borough-wide models in comparison with the industry benchmark from Zillow. It is clear that there is still a lot more to improve before our model can be competitive with the benchmark.

Borough-Level Elastic Net Models

Model	Median Absolute Percentage Error
All NYC	21.2
Manhattan	17.91
Brooklyn	20.58
Queens	18.59
Bronx	21.85
Zillow Benchmark	5.2

Table 1: Test set evaluation errors of normalized prices using elastic-net model with regularization coefficient and l1/l2 regularization ratio chosen by 3-fold cross-validation.

B. Non-linear Modeling

Random Forest vs. XGBoost

Model	hyper-parameter Search Space	MSE	R^2
RF	First search: Tree depth - [4,6,8,10] Second search: Learning rate - [0.01,0.1,0.3,0.7]	0.0456	0.934
XGBoost	Tree depth - [4,8] Number of Trees - [50,100,200,400] Number of Features - [P, sqrt(P)]	0.196	0.720

Table 2: Test set errors following initial hyper-parameter optimization for Random Forests and XGBoost

Since the linear models demonstrated significant bias, we proceeded to explore two approaches to city-wide non-linear modeling: random forests and tree boosting with XGBoost [4]. The first round of hyper-parameter optimization revealed that random forests underperformed compared to XGBoost, so subsequent tuning efforts focusing on the XGBoost model. These initial RF and XGBoost results in shown in Table 2.

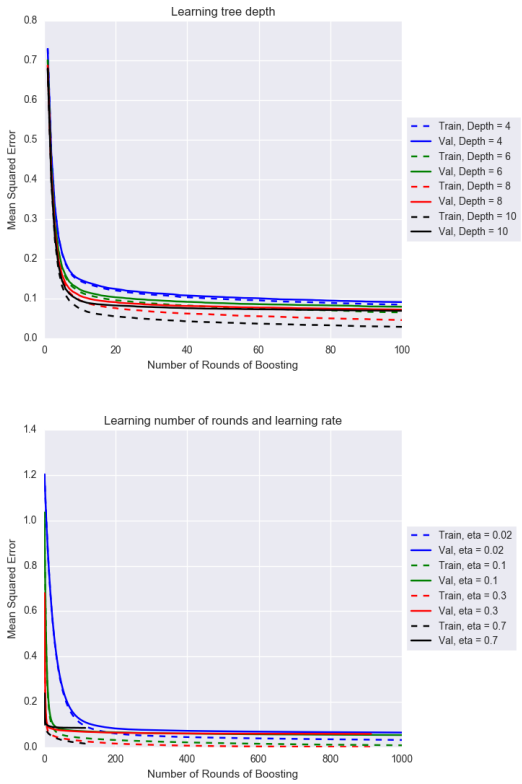

Fig. 5: Learning curves for XGBoost hyper-parameter optimization. To constrain runtime of hyper-parameter optimization, we first used gird search to identify the optimal tree depth on 100 rounds of boosting, then optimized the learning rate given this optimal depth

Since XGBoost was clearly outperforming the random forest (though further tuning of the random forest could potentially have produced further gains), subsequent experiments were conducted to further optimize the XGBoost model. As shown in Fig. 5, these experiments included testing:

- Loss functions: (i) RMSE, (ii) Root Mean Squared Percent Error (RMSPE), implemented as a custom less function
- Target transformations: (i) Scaled, (ii) log-transformed and normalized.
- Learning rate: Grid search over [0.01,0.1,0.3,0.7]
- Max tree depth: Grid search over [4,6,8,10]

Based on these experiments, the final optimized model was 1000 rounds of boosting, a maximum tree depth of 10, a learning rate of 0.1, with the objective of RMSE on the log-transformed and normalized price. Finally, using the optimal

XGBoost configuration, the model was refit to the entire training set. Feature importance is shown for the 10 most important features in the final XGBoost model in Fig. 6.

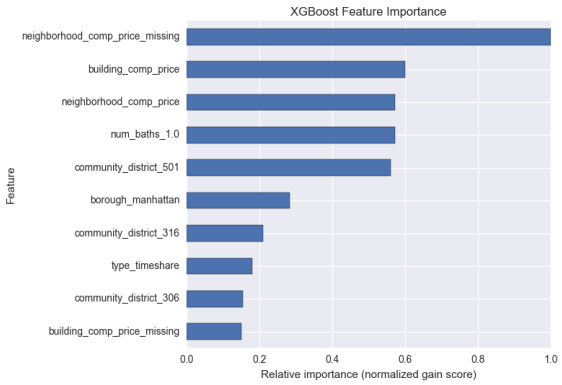

Fig. 6: Feature importance for final model

V. Future Work

We went through a tremendous amount of effort to obtain a large amount of high quality data for supervised learning problem, but there is still scope to improve the quality of the data. The target variable that we used for the supervised learning problem was a price scraped from the StreetEasy sales pages. In many cases, this number is the not the final sale price (such as when a listing doesn't terminate in a sale), so there is some error in our target variable and the actual market value of the property.

One aspect of the project that we did not address in detail was the date associated with each listing. We used the final listing date as a feature for our models, but we simply treated it as a categorical feature with each year being a distinct label. There are potential improvements that can be made with the prediction accuracy if we take the continuous nature of date into account and construct more complicated features to represent changes over time.

In addition to more expressive and robust features, we can also improve our model by increasing the size of our training data set. The StreetEasy NYC listings are primarily focused on Manhattan and Brooklyn apartment style properties, so the model performs poorly on properties in other boroughs. The highly non-linear nature of the XGBoost algorithm also means that generalization error would reduce as the size of the training data set is increased. One conceptually straightforward method of getting more is to scrape other real estate webpages such as Trulia and Zillow. We can reuse many components of our StreetEasy data collection pipeline for this task, but removing duplicate entries from multiple sources and merging features in a consistent manner is a task that would require a great amount of effort

In terms of modeling, the XGBoost algorithm is extremely computationally expensive, and while we have converged on a good set of hyper-parameters, we cannot guarantee that it is optimal or even approximately optimal. Future work could be done to setup a pipeline to perform model selection in a parallel computing environment so that more model configurations can be tried within a tractable amount of time.

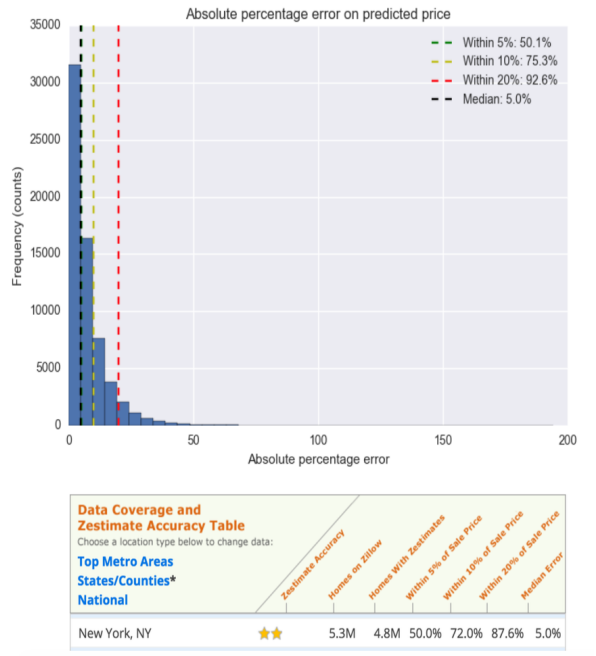

Fig. 7: Final model performance compared to Zillow's deployed NYC price prediction model

VI. Conclusion

As shown in Fig. 7, our final predictive model achieved a median out-of-sample absolute percent error of 5.0%, identical to that achieved by Zillow's NYC zestimate model. Hence, based on this industry standard evaluation metric, we conclude we learned a predictive model that is competitive with the model deployed for NYC by one of the largest online real estate sites. Importantly, our thresholding rule (training on sale records between the 5^{th} and 95^{th} percentiles) also resulted in us only making predictions on $\sim 90\%$ of the records with valid addresses, comparable to the percentage of Zillow.

References

[1] E. Pagourtzi, V. Assimakopoulos, T. Hatzichristos, and N. French, "Real estate appraisal: a review of valuation methods," *Journal of Property Investment & Finance*, vol. 21, no. 4, pp. 383–401, 2003.

[2] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[3] J. D. Mcauliffe and D. M. Blei, "Supervised topic models," in *Advances in neural information processing systems*, 2008, pp. 121–128.

[4] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *arXiv preprint arXiv:1603.02754*, 2016.