NYC Real Estate Price Prediction

Benjamin Jakubowski, Haonan Zhou

> Introduction

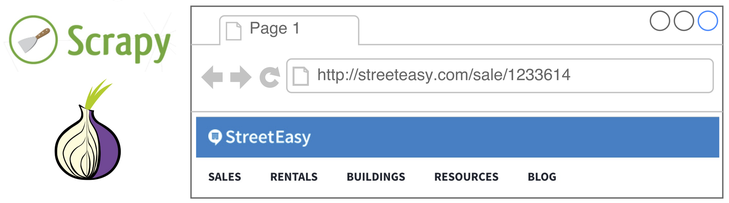
Problem

- In real estate markets, prospective buyers and sellers must valuate properties to inform their asking and offering prices.
- To do so, many buyers and sellers rely on real estate agents' comparative market analyses, which valuate real estate using past sale prices for comparable properties.
- We aimed to develop a price prediction model using machine learning algorithms instead of expert intuition and a priori definitions of comparability.

Objective

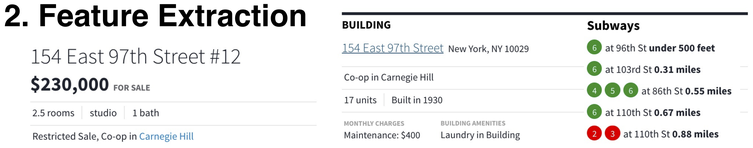
- In our modeling, we aimed to minimize median absolute percent error in predicted price. This objective was selected as it is used by the real estate website Zillow to evaluate the accuracy of their price predictions, and thus can be viewed as the industry standard evaluation metric.

1. Getting Data

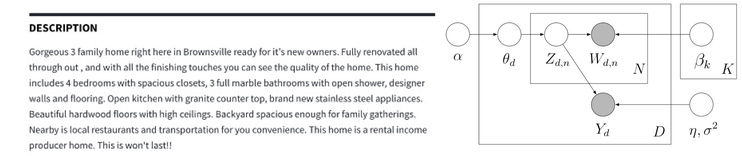


1. Initially tried using NYC public data, but available data were feature-poor.
2. Ultimately scraped raw HTML for 516K sale pages on real estate website streeteasy.com, using Scrapy framework and Tor network.

2. Feature Extraction



1. Extracted features available directly from raw HTML, used binning and one-hot encodings for categorical and transit features, and used mean imputation to fill missing values.
2. Used Rtrees to infer Neighborhood, Community District, and Borough from lat/long.
3. Constructed comps features for sale price and unit size from comparable units in (i) the same building, and (ii) the same neighborhood.
4. Thresholded the target variable, passing records with sale prices between the 5th and 95th percentiles (so similarly to Zillow we predicted the price of ~90% of the available units).

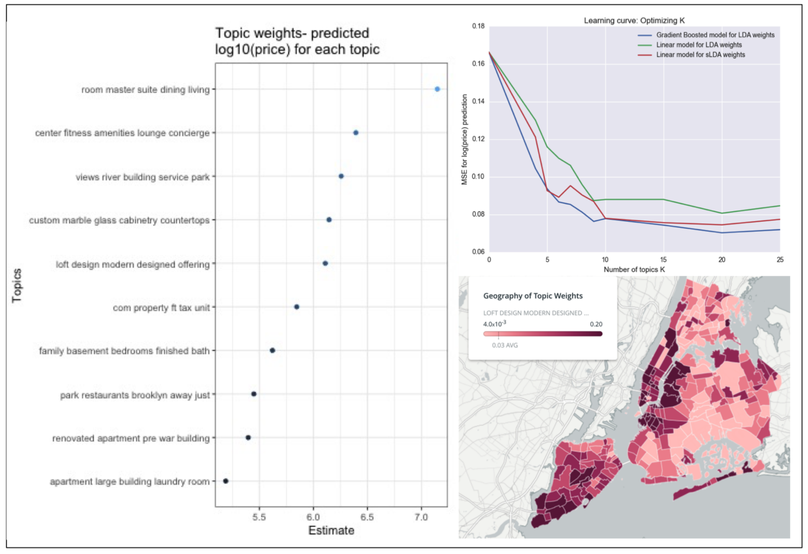


5. Used supervised latent Dirichlet allocation (sLDA) to learn topic representation of description.

> sLDA Featurization

In streeteasy, much of the information regarding the property type (ex: luxury apartment, pre-war building near the park) is provided by a descriptive paragraph. Hypothesizing these latent style variables could be identified using LDA or sLDA, we learned a number of topic models. These models are summarized in the figures below:

- **Top right**: Learning the optimal K for sLDA and LDA topic models based on validation set log(price) MSE.
- **Bottom right**: Maps of neighborhood median topic weights reveal geographic coherency of learned topics. This example map shows geographic distribution of Topic 7: designed loft properties.
- **Left**: Coefficients learned in the sLDA model can be interpret as the predicted log(price) for the corresponding single topic (one-hot) vector.



> Price Prediction Models

Following featurization, we tried the following approaches to predictive modeling:

- **Linear Models**: We tested linear models for city-wide, borough-level, and community-district level price prediction.
- **Non-linear Models**: We tested non-linear models (random forest and additive regression trees using XGBoost) for city-wide price prediction.

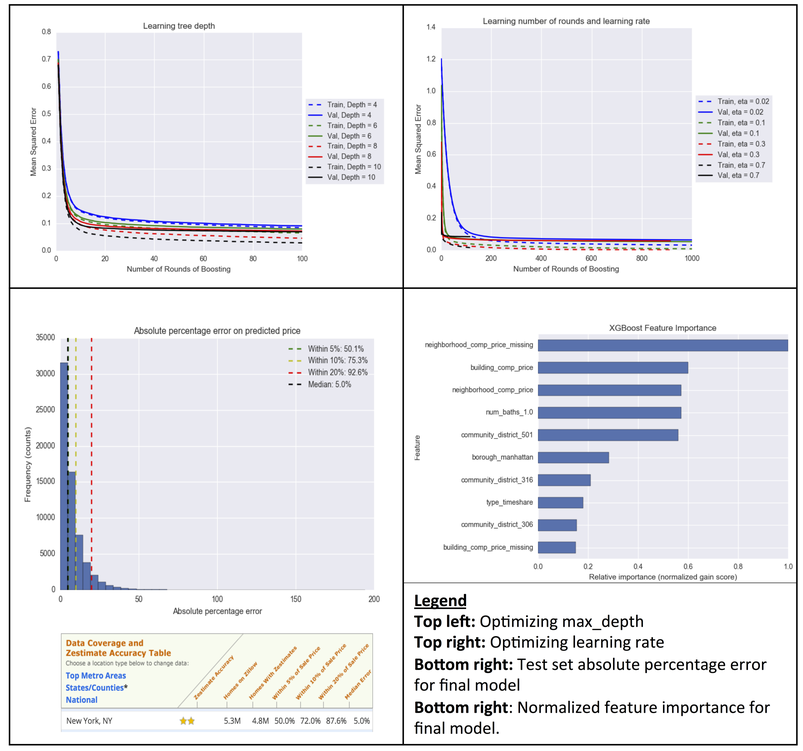
For the regression problem we initially trained our model with normalized price as the target variable. However as we delved deeper into model optimization, we found that using normalized log price as the target yielded the best validation set performance. Additionally, based on initial hyper-parameter optimization, it was clear XGBoost provided the best performance, so subsequent optimization efforts were focused on tuning XGBoost.

> Model Tuning

Since XGBoost outperform other models in initial experiments, the focus of model tuning was optimizing XGBoost parameters. Experiments included testing:

- **Loss functions**: (i) RMSE, (ii) Root Mean Squared Percent Error (RMSPE)
- **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]

First tree depth was optimized, then learning rate and number of rounds of boosting. Loss functions and transformations were tested within this grid search framework.



Legend
Top left: Optimizing max_depth
Top right: Optimizing learning rate
Bottom right: Test set absolute percentage error for final model
Bottom right: Normalized feature importance for final model.

> Conclusion

- We achieved test set performance comparable to the deployed NYC "Zestimate" model.
- XGBoost performed best for this problem, compared to random forests and regularized linear models. This is most likely due to its ability learn complex non-linear interactions between the different features we engineered.