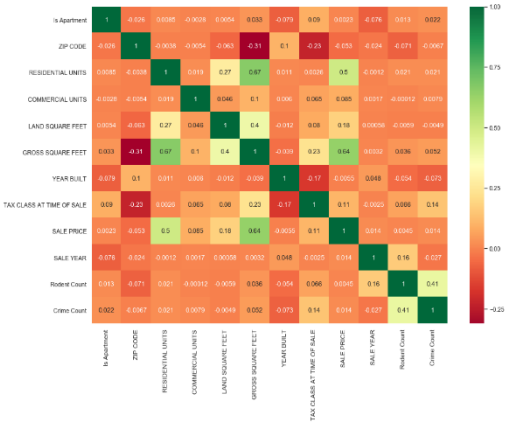
NYC Housing Predictions – Machine Learning Analysis

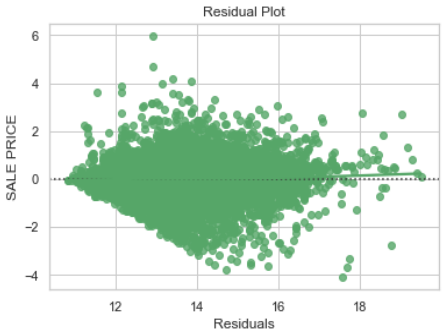
So, for this project, I was interested in whether there was a better way to predict a house's price. The main thing I wanted to see was whether construction projects had an impact on the prices of sales. I predicted that it would have a strong positive correlation due to anecdotal experience.

There was about a dozen features that were not classification data that seemed to have some effect on the sales price of the properties. After all the data cleaning and one hot encoding was done, there were roughly 500 features for the datasets. Because there are so many features in the dataset and since we do not have a clear understanding of which of those features are important, Lasso regression helps automatically determine which coefficients and features should be selected or removed. Lasso regression removes coefficients that are unnecessary and "shrinks" features without a big risk of increasing bias. At the time, it seemed Lasso was the way to go, however, after scoring the Lasso regression model, Lasso did not do as well as I hoped.



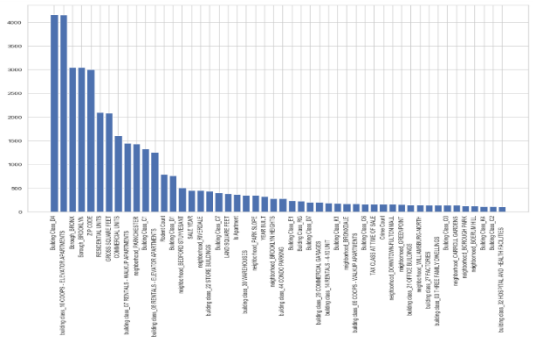
Correlation heatmap, before one hot encoding

Moving on from that, I tried out a Random Forest model, it also does somewhat of an automatic (albeit random) feature selection using bootstrap resampling. Due to time constraints of running the model I could not hyper tune it to be extremely precise however I did attempt a few parameter tunings to determine what number of trees and number of features would generate a good model score. After plotting the predicted vs actual, I noticed that there were too many points on the plot, I had to add an alpha value to each point so that I could see where most of the points were. There did not seem to be too many outliers, which was a good sign, however I noticed that there was a large variance between the predicted and actual. I felt like this was due to either an incomplete feature cleaning or it could be due to other external features that were not available in the dataset that was constructed.

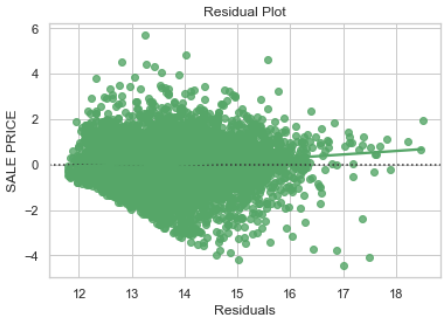


Residual Plot: Random Forest

Along with random forest, I also tried a Jackknife resampling method for random forest. It was a new technique that was gaining popularity and was mentioned to me. However, I could not find an easy to use implementation of it, so unfortunately, I had to implement it myself with a twist. So instead of doing a complete random jackknife resampling method (which is just basically a bootstrap resampling without replacement) I did an average between gotten from a normal jackknife random forest regression and a decision tree regression of the best n (equal to the rest) features. I thought it would be appropriate since I already formulated the features that were most impactful using SelectKBest from sklearn. Unfortunately, the R^2 score was just slightly lower than the bootstrapped random forest one, a 63% vs a 73% for random forest. This could be for several reasons, but I personally believed that the implementation of the jackknife random forest was not perfect.



SelectKBest Plot: best 50 features



Residual Plot: Jackknife Random Forest

As always, more data and more feature cleaning would probably help the model better. There was a lot of generalization that was done to make data handling easier. For example, I generalized rodent data (count), crime data(count), construction data (average sales) into zip codes. Doing that did not account for several things like how far the crime, rodent infestation, or construction plan was from the specific address. I think if I had better processing power I would probably do that instead of just lumping them into groups.