Predicting New York City Property Prices Using Machine Learning Models

Taruna Kar

Abstract Today, machine learning is a powerful tool that can be used to facilitate computation and modeling in many real-world applications. In this paper, I will detail how I predicted housing/property prices in New York City, given data from a fairly large and comprehensive public dataset. After applying extensive preprocessing and cleaning techniques on this data, such as filtering outliers and normalizing numerical features, I fit several machine learning models, including but not limited to many variations of linear regression, like lasso/ridge regression and elastic nets. I found that out of the 5 models I did, random forests consistently generated the lowest root mean square error (RSME) of around .610. While working throughout the entire pipeline, I also came across a few areas that could serve as several extensions, which I will also detail in this report, that I could add to my model at a later point, in order to minimize the error further.

I. Introduction

There are several uses for machine learning in this day and age. The use case I studied was using data from a given dataset to predict property values of New York City housing units. One of the fundamentals of machine learning is understanding what kind of problem this task is asking us to solve. Since we are trying to compute a value that is on a continuous range, and not discrete, we must opt for an algorithm that uses any sort of regression. Therefore, for this task, I opted to use various machine learning models that all used regression.

There are online datasets that are available to the public that provides not only sale price data for various units, but up to twenty separate features that describe the data. In the dataset I will be using to solve this task, about 20% of the data does not have available pricing data, which is what we need to try to predict using a machine learning model. This also makes regression models increasingly appealing, due to the number of numerical features and the ability to also use categorical variables to our advantage. Some of the most prominent and relevant features are the number of units, type of building, square footage, and various location elements like borough, zip code, block, and lot.

The next few sections will detail my pipeline, which primarily consists of numerous pre-processing steps followed by constructing my machine learning models and evaluating them using popular error metrics, and concluded by running my chosen model on sample data that have unknown pricing data.

II. Pre-Processing Steps

A. Initial Feature Analysis

As stated in the introduction, the public dataset that I used for this task had unknown values when it came to *sale price*, the primary label we're interested in predicting, and for some of our features, like *land square footage* and *gross square footage*.

Before I began to make any modifications to the dataset, scanned the feature list to see if there were any I could drop from the dataset, based on intuition or common sense. I dropped *ease-ment* simply due to the fact that it was an empty column. Features like *address* and *apartment number* were not general enough, and were probably specific enough to have their own category for every data point. This would not contribute anything of substance to our machine learning model, so I chose to drop them. The features *tax class at present* and *building class at present* seemed to be redundant because of two other features: *tax class at date sold* and *building class at date sold*. Upon closer inspection, comparing the values between the two respective categories, most pairs were the same, so as such, I dropped *tax class at present* and *building class at present*.

There were a couple of features on the list that did not seem absolutely necessary to include in the final feature list, like *zip-code* and *neighborhood*. Since these features basically describe similar things regarding a property value, perhaps the only difference being that a zip-code might encompass multiple neighborhoods, there is a chance that there would be a high level of correlation between the features. Statistically speaking, *multi-collinearity* among features does not necessarily improve a machine learning model and in fact can have introduce a higher level of variance and error, as well as inconsistent results **[1]**.

Therefore, for those reasons, I dropped the features *lot*, *block*, and *zip-code*, and kept *neighborhood* and *borough* as my primary location features.

B. Feature Modification

This section will detail what sort of initial pre-processing steps I took for the features that are remaining.

Sale Date

This feature was given in the format: *YYYY-MM-DD HH:MM:SS*. Since this feature is very specific, in order to generalize it, I truncated the object values to only be of the format *YYYY-MM*. That way, there would be around eleven or so categories that we can use as a feature, which is relatively generalized.

Building Class Category

There is really no pre-processing that's necessary for this feature, but I simply removed any and all character literals, and changed number categories like "01" to "09" to "1" to "9", respectively. This way, we just have numbers one to around forty-eight.

Total Units

In order to see how to best pre-process this data, I tabulated data regarding *total units* along with how many units had that *total unit* count. Upon inspecting the data, approximately 25% of the data listed that they had zero total units. This seemed very skeptical to me, as it made sense that all housing should have at least one unit, which the table verified, because it said that half of the dataset contained housing in which there was one unit. Therefore, I removed all data points that listed that they had zero units.

Since I expect to see a high degree of correlation between *total units* and *residential units* and *commercial units*, I gave myself the choice of getting rid of this feature at some later point down the pipeline. This is something that I will discuss in the *Results* section.

Residential Units & Commercial Units

Since these two features are highly correlated with *total units* (in fact, total units is just equal to the number of residential units plus commercial units), I did not feel a need to perform any preprocessing on these two features.

Sale Price

Since approximately 20% of the data points had no sale price data, I split the dataset into two datasets: one that had no known sale price data (for testing at the end of our pipeline) and one that had known sale price data, which I further processed before using it to develop my machine learning.

Land Square Feet & Gross Square Feet

For these features, alongside *sale price*, I saw that there were a lot of missing values. However, rather than getting rid of the data samples that contained no information for these two features, I decided to impute the missing values with the mean of the sale price from each borough. There wasn't enough missing data for this pre-processing step to introduce a large magnitude of variance but there was enough to make a difference, if removed from the dataset.

C. Numerical and Categorical Feature Classification

The next step was to classify each remaining feature as either numerical or categorical. It is important to make a distinction because of features like *borough*, for example. That feature take on one of five values: one to five, representing each of the five different boroughs in New York City (Brooklyn, Bronx, Queens, Staten Island, Manhattan). However, for example, if borough 1 is Brooklyn and borough 2 is Manhattan, it does not necessarily mean that Manhattan's housing prices will be greater than Brooklyn's. In other words, *borough* would be a categorical feature because there is no numerical significance to the numbers given to represent the boroughs. I therefore classified *borough* as a categorical variable.

D. Correlation Matrix Analysis

After all of that pre-processing, I wanted to take a look to see how correlated the numerical features are to each other (see above section II.A to see discussion regarding *multi-collinearity*). Therefore, I calculated the correlation matrix, which can be seen in the Fig. 1 below.

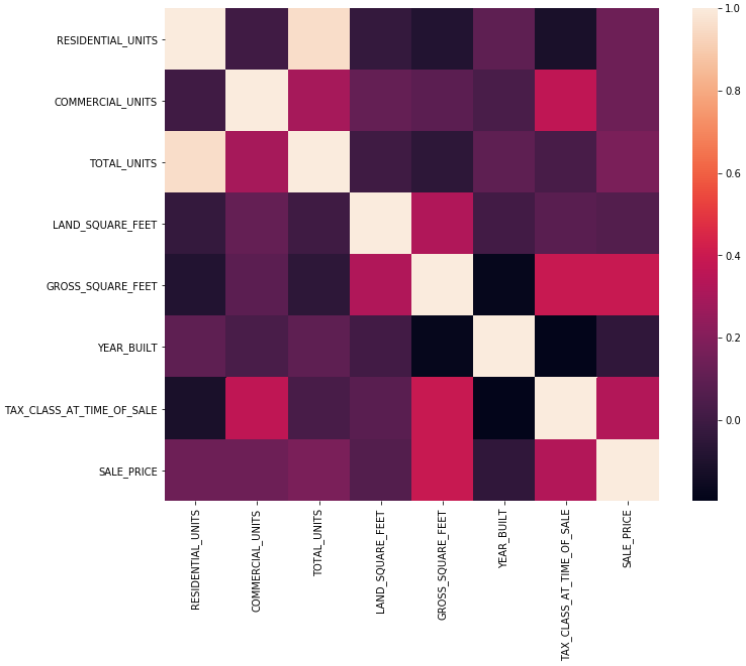

Fig. 1. Above is the correlation matrix that shows the correlation between all 8 numerical features that remains a part of my dataset at this point in the pipeline.

According to this correlation matrix, the only features that seem to have a high level of correlation (defined as greater than .7) are *total units* and *residential units* **[2]**. I therefore chose to test a model with *total units* and without *total units* because I was not straight away convinced that doing one or the other would give me better results.

Additionally, it would seem that there is no strong correlation between *sale price*, which is our feature of interest, and the other seven numerical features. The highest two correlation values are .391 with *gross square feet* and *.328* with *tax class at time of sale*, both of which make somewhat intuitive sense **[3].**

E. Statistical Analysis of Features

In this section, I will describe what sort of statistical approach I took to analyzing each numerical feature.

Sale Price

Statistically speaking, we would like to see that the distribution of any numerical feature is approximately normal and not skewed. However, before we do that, we should filter outliers in our data so that they do not have an impact on our chosen machine learning model later on. According to the figure 2 below, there are quite a few points

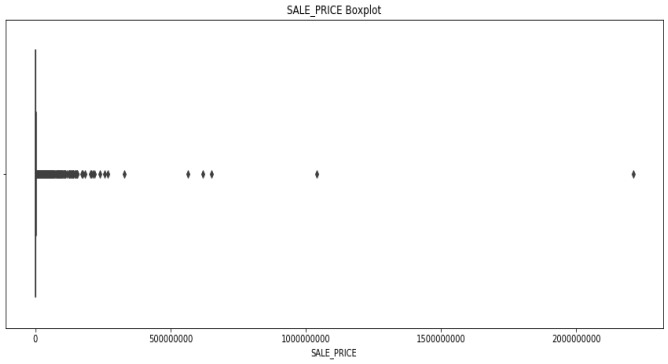

Fig. 2. The above figure shows the single element distribution of *sale price* across our whole dataset

that we can filter. To do this, I set two variables named *upper bound* and *lower bound* as bounds for cutoff. As in, if the *sale price* is below *upper bound* and higher then *lower bound*, then it can remain in the dataset. In the results section, I will talk about effects of changing these variables (namely *upper bound*) on the performance of our machine learning model. We then get a normal distribution as shown in figure 3.

Since we may possibly normalize our numerical features later on in the pipeline, it's important to see that at least the log transformation of the *sale price* distribution is at least normal and un-skewed. See figure 4 for our log transformation.

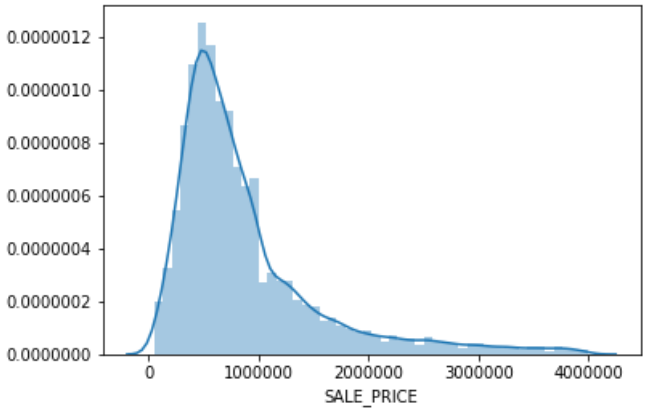

Fig. 3. The above figure shows the right-skewed normal distribution of *sale price*. The skew value is 1.988. Lower bound is 5000 and upper bound is 4000000

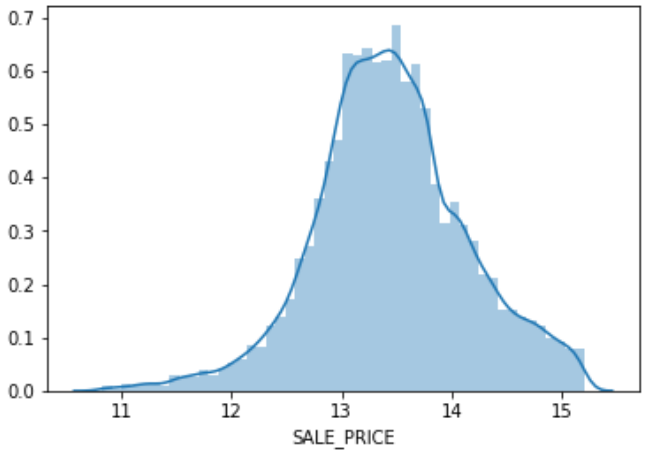

Fig. 4. A visualization of the log transformed distribution of *sale price*. This distribution has a skew value of 0.066.

Total Units

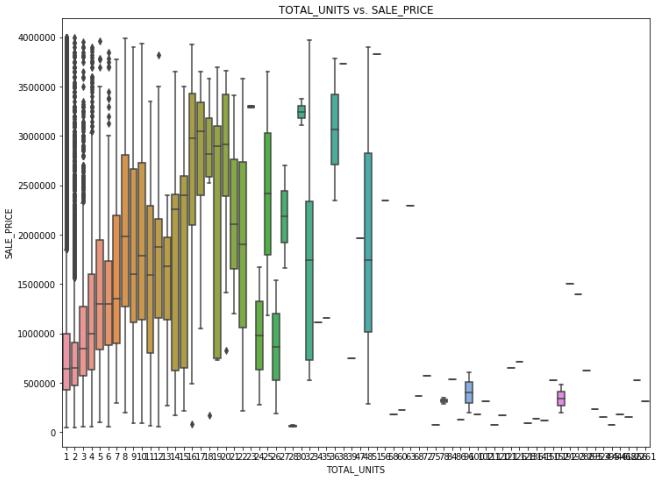

Fig. 5. A visualization of the boxplot distribution of *total units* vs. *sale price* before removing outlier data.

Upon initial visualization of the *total units* versus *sale price* boxplot distribution (see Figure 5), we see that there is

a lot of outlier data that don't seem to follow the basic intuition that sale price increases with increasing number of total units. Therefore, I removed all data points that contained more than thirty total units to get the distribution seen in Figure 6. We can clearly see that there is a positive correlation between *total units* and *sale price.*

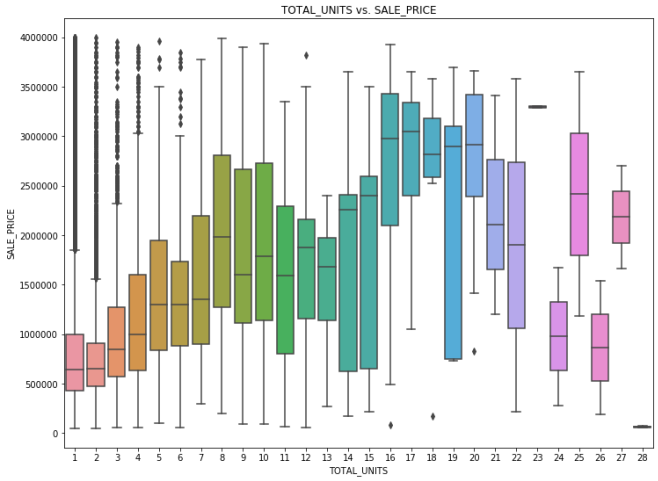

Fig. 6. A visualization of the boxplot distribution of *total units* vs. *sale price* after removing outlier data.

Residential Units

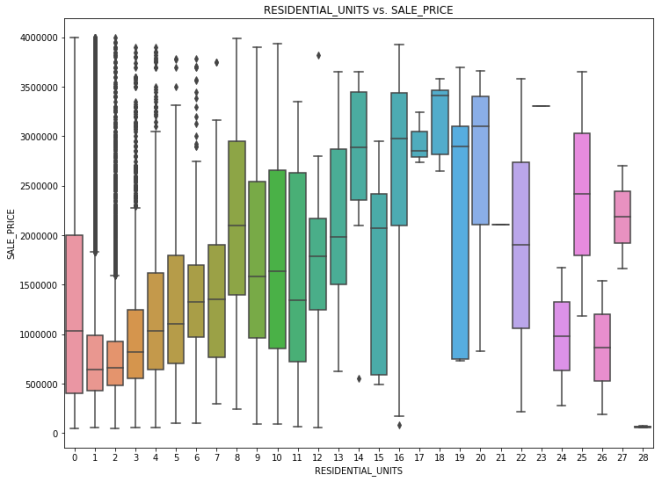

Fig. 7. A visualization of the boxplot distribution of *residential units* vs. *sale price* before removing outlier data.

Figure 7 shows the boxplot distribution of *residential units* versus *sale price*. There seems to be outliers that do not follow expected behavior past 21 residential units, so I got rid of those respective data samples to get the box plot distribution seen in Figure 8.

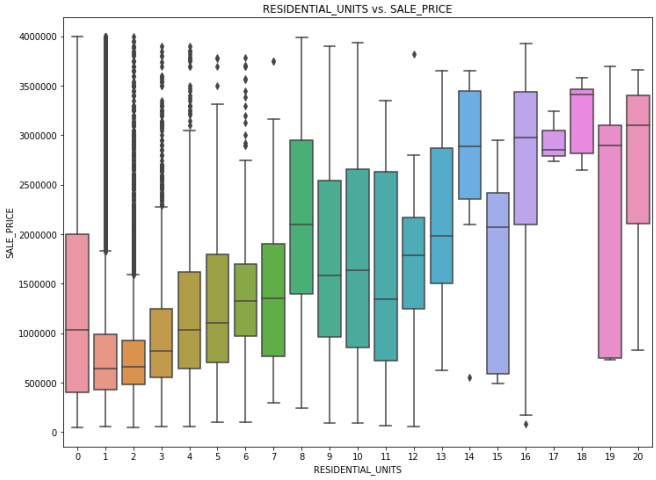

Fig. 8. A visualization of the boxplot distribution of *residential units* vs. *sale price* after removing outlier data.

Commercial Units

Figure 9 shows the boxplot distribution of *commercial units* versus *sale price*. There seems to be some outlier data here as well, but I chose not to void those samples.

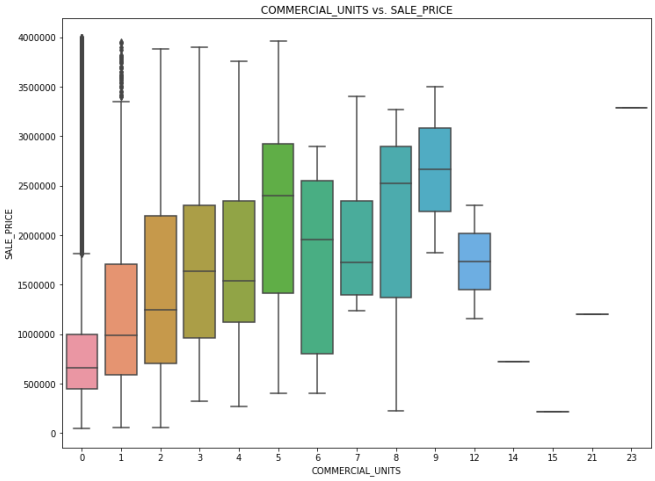

Fig. 9. A visualization of the boxplot distribution of *commercial units* vs. *sale price*.

Year Built

Figure 10 shows the boxplot distribution of *year built* versus *sale price*. When it comes to these particular features, it is hard to guess any sort of correlation. Perhaps the newer the building then the higher the sale price is one intuitive relationship one can put hypothesize, but the data does not seem to show that sort of correlation. Therefore, we will leave the dataset as is.

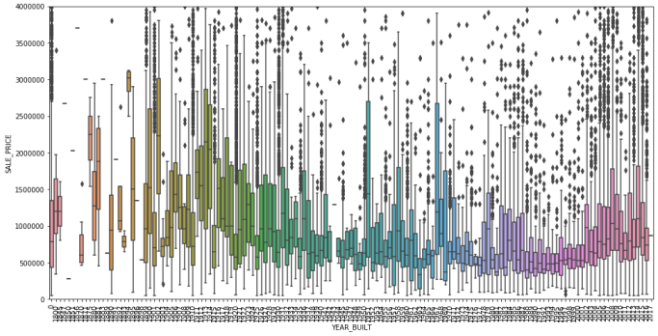

Fig. 10. A visualization of the boxplot distribution of *year built* vs. *sale price*. The range is from 1800 to 2017.

Tax Class at Time of Sale

Figure 11 shows the boxplot distribution of *ax class at time of sale* versus *sale price*. This follows the trend one would expect, according to how tax classes are defined by the state of New York [3].

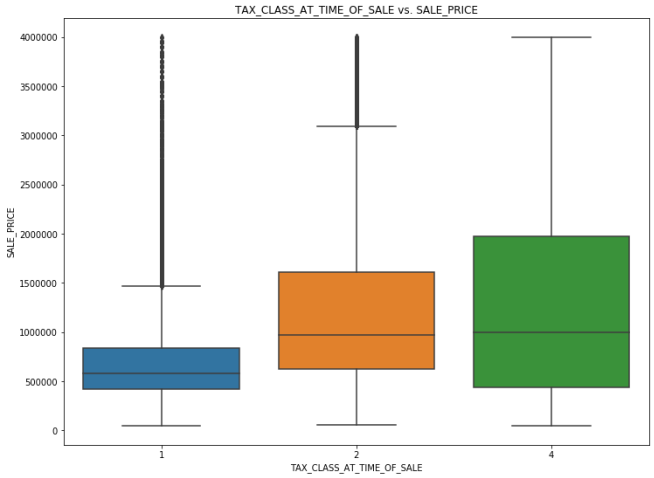

Fig. 11. A visualization of the boxplot distribution of *tax class at time of sale* vs. *sale price*.

Land Square Feet

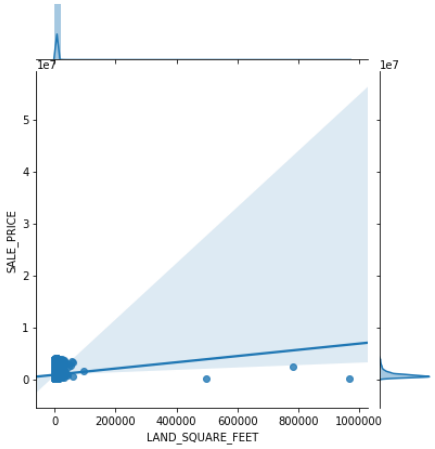

Fig. 12. A visualization of the boxplot distribution of *land square feet* vs. *sale price* before removing outlier data.

Figure 12 shows the scatter plot distribution for *land square feet* versus *sale price.* Here, I just got rid of the few outlier data points to get the distribution seen in Figure 13.

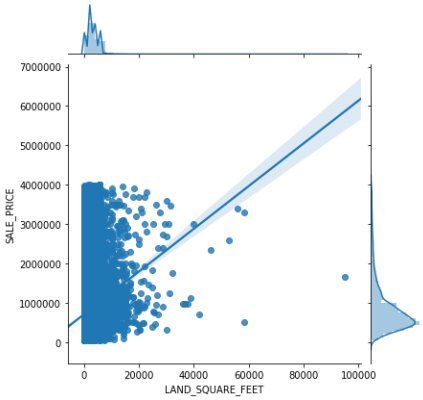

Fig. 13. A visualization of the boxplot distribution of *land square feet* vs. *sale price* after removing outlier data.

Gross Square Feet

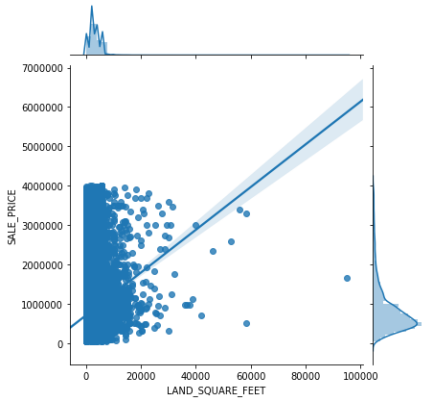

Fig. 14. A visualization of the boxplot distribution of *gross square feet* vs. *sale price* before removing outlier data.

Figure 14 shows the scatter plot distribution for *land square feet* versus *sale price.* Here, I just got rid of the few outlier data points to get the distribution seen in Figure 15.

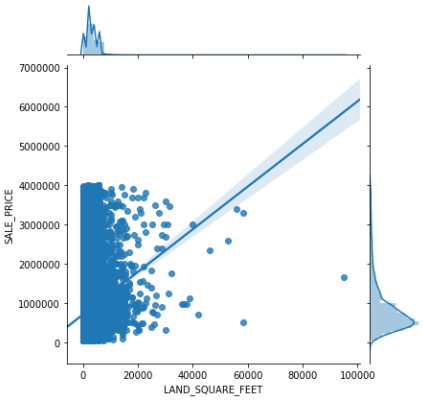

Fig. 15. A visualization of the boxplot distribution of *gross square feet* vs. *sale price* after removing outlier data.

F. Numerical Feature Transformation

Just like how I performed a log transformation on *sale price*, naturally I went ahead and performed log transformations on the rest of our numerical features. This had some implications on our model performance that I will describe in the results section.

G. Categorical Feature Transformation

It is highly likely that there are categorical variables that are correlated with *sale price*. Even though we cannot just plug them in as numerical features, with the *one-hot encoding* technique, I was able to transform the categorical features into true/false feature columns. This way, we can incorporate these features into our model. After performing this pre-processing step, my dataset consisted of 429 features/columns.

Note that it is not sufficient to do *label encoding* in this case. This is because all of our categorical variables are ordinal, meaning that there is no numerical significance in the way the values are ordered (see discussion in section II.C)

III. Machine Learning Models

Now that I had finally cleaned the data to the best of our ability, I started building some machine learning linear regression models.

A. Linear Regression

This is the simplest of models one can use to perform this task. In the results section, I will detail the issues that I ran into when using this model.

B. Lasso

The lasso linear regression model seemed particularly attractive to me because of its built-in regularization term [4]. My dataset contains 429 features because of *one-hot* encoding. This way, if there were features that did not contribute much to our machine learning model (which is highly likely), then they would not be incorporated into the model whatsoever. One disadvantage I saw with the lasso model was that it was took the longest to fit.

C. Ridge

Similar to the lasso method, the ridge method also makes use of penalty regularization. It was computationally inexpensive and had similar performance measures to the lasso method.

D. Elastic Net

Naturally, I also chose the elastic net method so I had the freedom of choosing my *alpha* hyper-parameter so that I could optimize this model's performance.

E. Random Forest

Finally, I chose the random forest model because of the strengths of ensemble learning, and its advantages compared to its more basic form, the *decision tree*.

F. Error/Performance Metric

I chose the *root mean square error* (RMSE) to be my primary performance metric. As I will explain in my results, I used a non-normalized RMSE for my dataset with non-normalized numerical features and a normalized RMSE for my dataset with normalized numerical features. Also, I compared the errors of each model to each other to evaluate performance relative to one another.

G. Cross Validation

Another common metric used to determine which model/hyper-parameters to select given a dataset is *cross-validation*. I performed cross-validation with all of the machine learning classifiers to use it as one of my metrics, along with RMSE, to evaluate which model works the best.

I ran into problems with running cross-validation with some of my models due to the sheer amount of time it takes to fit the model *k* times what it usually takes to fit the classifier, if *k* is defined as the number of folds used for *cross-validation*.

H. Train-Test-Split

After performing cross-validation, I went on to split the dataset that I had been using in my pipeline up until this point in the process into a training data set consisting of around 29,000 data points and a testing data set of roughly

9000 data points, with a split fraction of 0.25. After doing so, I then proceeded to fit my models and calculate the RMSE's after using my models to predict *sale price* values for my testing data set.

IV. Results and Analysis

Due to the time required for cross-validation and the lasso method, I was not able to secure consistent results for cross-validation, but the results for the five machine learning models are as follows (for which I used an *alpha* value of 0.005, where applicable):

	Average Cross-Validation Score	*Non-Normalized RMSE*	*Normalized RMSE*
Linear Regression	*1.02*	*551,485*	*N/A*
Lasso	*1.06*	*524,489*	*.794*
Ridge	*.67*	*470,850*	*.662*
Elastic Net	*.345*	*492,247*	*.769*
Random Forests	*.936*	*443,599*	*.648*

From cross-validation alone, we cannot exactly rule out any model due to its inconsistent relationship with the RMSE. However, the RMSE tells us that our random forest regression model works the best in the context of this task.

Figure 15 below shows a random sample consisting of 250 test samples with the actual sale prices (in red) and the predicted sale prices (in blue) from our random forest model.

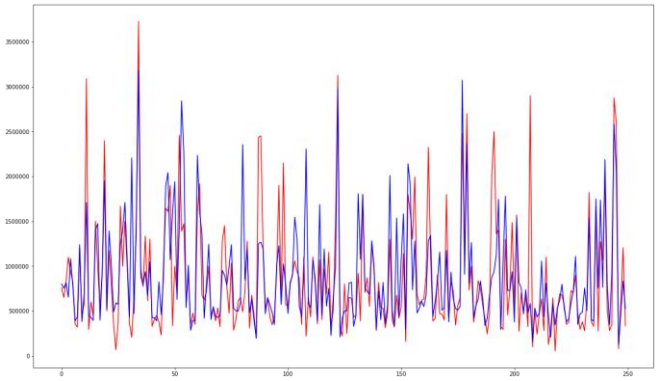

Fig. 15. A random sample from my testing data set consisting of 250 test samples. The data in red represents actual sale prices while the data in blue represents my random forest model's predictions.

Peaks Analysis

For most of the data points, we see that the predicted values are somewhat close to the actual sale prices. The biggest discrepancies that contribute the most to the RSME would be the peaks in the dataset. Half of the time, for the relatively larger sale prices, my model does not predict those values nearly enough and are off by up to almost 50%. This may suggest that for these larger peaks, there may be some features that I had initially dropped that perhaps could actually have been a better indicator for my model.

For example, when I was defining my *upper bound* and *lower bound* variables, perhaps I should left those data points that seemed to be outliers. However, in exchange, the data would have been even more right skewed and thus have decreased the performance.

V. Conclusion and Possible Extensions

After all of the pre-processing/cleaning steps that I detailed thus far, I was able to build linear regression machine learning models that were able to predict housing prices of properties in New York City, given various other information like location, square footage, and number of units. My random forest model came out on top as having the lowest RSME of .648. For the same *alpha* values, my ridge model worked best, followed by elastic net and then lasso. Simple linear regression proved to have one of the worst RSME's, but this was to be expected because of the importance of all the comprehensive pre-processing steps I performed.

As possible extensions, we could use *principal component analysis* as a form of dimensionality reduction, to reduce computation speed. We could have also imputed the missing land and gross square footage values using linear regression on them using the other features (with the exception of *sale price*). Finally, although removing outliers helped us normalize and generalize the data, perhaps it may have been good to keep them so that maybe we could have predicted larger sale price values more accurately.

References

[1] C. F. Mela, "The impact of collinearity on regression analysis: the asymmetric eVect of negative and positive correlations." .

[2] "14: Correlation." [Online]. Available: http://www.sjsu.edu/faculty/gerstman/StatPrimer/correlation.pdf.

[3] *NYC*. [Online]. Available: https://www1.nyc.gov/site/finance/taxes/property-tax-rates.page. [Accessed: 05-Dec-2018].

[4] C. Guo, "Entity Embeddings of Categorical Variables."