Applying Machine Learning to New York City Housing Data

Jennie Coughlin

Data Analysis and Visualization, CUNY Graduate Center

DATA 71200: Advanced Data Analysis

Dr. Johanna Devaney

July 12, 2024

Introduction

For this series of projects, I used a New York City housing sales data set from Kaggle. Housing is always a big issue in New York City, so when I saw a reasonably clean data set of housing data, I decided to explore it. I was primarily interested in if machine learning could predict, based on the information about the property, if it was likely to be a condo, co-op, townhouse, single-family house or some other type of property.

Cleaning the Data

Fortunately, the data set on Kaggle wasn't missing any values, which streamlined the cleaning immensely. I did have at least one row where the property was marked as having zero bathrooms, and I ended up subsetting the data to remove that both because it would cause an issue when I transformed the data and because a legal residence in New York would have to have a bathroom, so that row either had incorrect data or was not a true residential property.

My biggest struggle ended up being with the two categories of data that I thought could be particularly useful, largely because they were categorical fields with many, many values. The Broker Title variable seemed like it could be useful if brokers specialize either in types of housing or in parts of the city dominated by one type of housing. The location data also seemed like it could be useful, given that "location, location, location" is a cliche about its importance in real estate.

The locality and sublocality fields were really messy in a way that I know how to deal with in Sheets/Excel where I can see the data as I'm working through cleaning steps, but I wasn't confident I could clean it in Python. The ZIP codes were easily sliceable, though, so I settled on that for location data. But that presented new challenges: There are 179 ZIP codes in New York City, and the need to one hot encode that column meant a very sparse matrix. I tried that, but ended up discarding the idea of using location data.

The Broker Title field had a similar issue, but with an even larger number of options, many of which were brokers with just one property and a handful that represented several properties.

As you will see, both decisions limited the usefulness of machine learning on this dataset.

Finally, I needed to do some filtering on the target variable. The original Kaggle data was sales data and the TYPE field represented the status of the property. Most rows were 'XXX for sale' where XXX represents the type of property (condo, co-op, single-family, etc.) for sale. But some were the status of the sale (Contingent, Pending, etc.) and were not useful for this project. So I took only the subset that showed the actual type of property and were property types common in New York City. Mobile homes and land were two of the types excluded at this point. I did, however, leave in cond-op. It was also rare, but it is a type of property unique to New York and it seemed worth including.

Visualizing the Data

One issue I spotted right away when visualizing the testing data was there were a few outliers that were really flattening out most of the dataset. It was unclear if these were errors in the data or just unusual properties, but they were so far outside the range of the rest of the data that I was able to exclude them using a fairly loose parameter to subset the data.

I also determined that to scale the bedrooms, bathrooms and square footage so they would work in conjunction with the sale price, I needed to use a logarithmic transformation, which was the only one of the options that didn't leave all the values piled on top of each other.

That said, even with the transformation, the data was still fairly skewed. If I could do it over, this is the point where I would have gone back to find a way to include useful location data so my model had enough features to work off of.

Supervised Learning Models

Because my label field was the type, and thus categorical, I was restricted to classifier models. I decided to try K-Neighbors and Decision Tree to see how they did. In both cases, the models worked fine on the training sets, but terribly on the testing data.

K-Neighbors was my first choice because the idea of classifying a property based on the ones most similar to it made logical sense, although I suspected it would take some tuning to figure out if it would work with the actual six clusters I knew were labeled or if it couldn't classify the data to that level of granularity. When I set it n=5, which was one fewer than the number of labels, the accuracy on the test set was just 0.23, and all of the ways of looking at the F-1 scores were terrible.

That remained true when I applied cross-validation and re-ran the model. I was limited to three folds because the smallest label group only had three rows. The training accuracy was much higher than the testing accuracy, and got worse the bigger the value of K, but the best performance on the testing set was n=4, one smaller than my default of 5.. None of the cross-validation scores even hit 50 percent, but they all were quite similar —

between .460 and .475. Given how much the model seemed to be overfitting, I was surprised that cross-validation didn't turn up more variation.

When I used Grid Search to evaluate different values, the top three were 6, 1 and 7, but I also tested out 2, which was ranked in the middle of the pack. The n=1 option performed the best (0.497), but the n=2 was next-best (0.479). At that point, I decided that KNN was unlikely to be the best model for the task.

The second model I picked was the Decision Tree Classifier because I thought that might do well by narrowing down the options for labels as it went. Its accuracy on the training data was pretty terrible, but it did do better on the testing data, though at 0.492 it was still below 50%. The biggest class had the best F-1 score, but three of the six classes had F-1 scores of zero.

When I ran cross-validation on this model, I got more range in the results — from 0.545 to 0.592 — but even the low end of that range is a much better performance than anything the KNN model produced. When I looked at the feature importance, none of the features had values of 0.4 or higher, though the two least important features were similar (0.154 vs. 0.190). So I tried limiting the depth, first to two features, then to three features.

Despite the similar importance for the two lowest features, knocking both out gave me similar results to using all four. When I decided to split the difference and limit the tree to three features, the model improved to 0.594 from 0.492, only two of the classes had F-1 scores of zero, and all the classes that had positive scores in the first version improved them.

Decision Tree Classifier still doesn't perform well, but it was a notable improvement over KNN, so when I moved on to Project 3, I used the Decision Tree model.

PCA for Feature Selection

I had hoped that using PCA for feature selection would help since my problem seemed to be overfitting the training set. Unfortunately, PCA ended up actually making the model's performance on the testing data worse when I used the 95% variance option. Even though it wasn't required, I opted to also test the model using PCA with two principal components. That performed even more poorly than the 95% variance model. Typically, I would think that the answer to overfitting would be a less complex model, but in this case it was not. My working theory is that the features in the model aren't capturing enough data to provide useful labels.

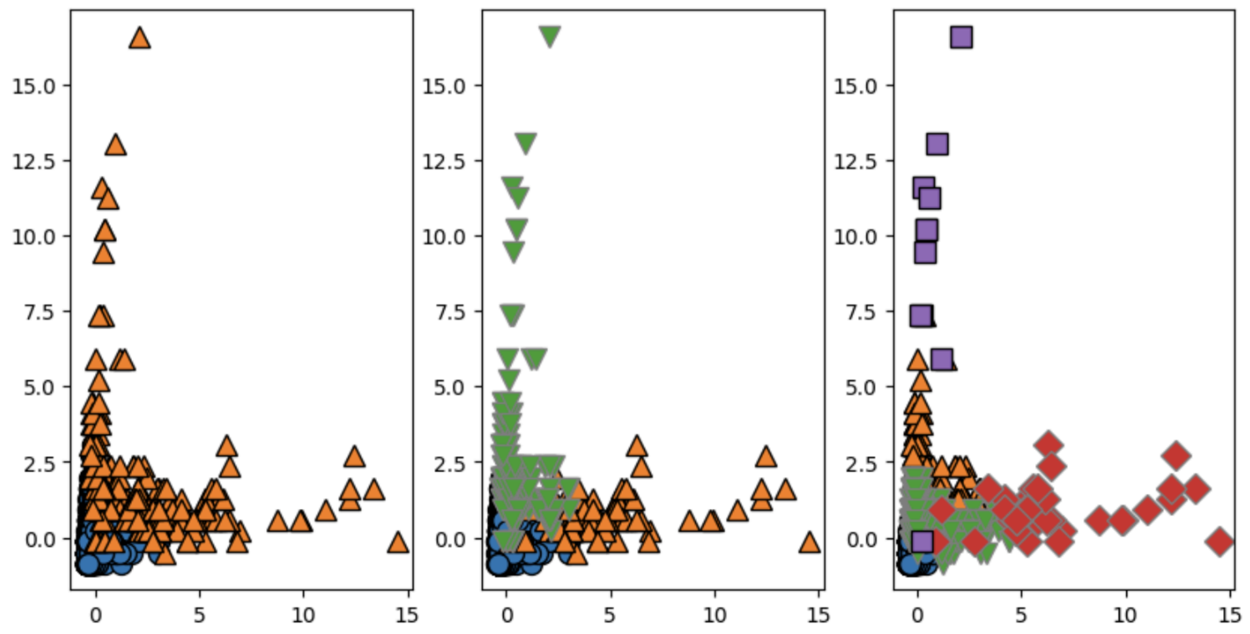
I suspect that if broker or location had been fields, the PCA would have been more useful in sorting out which of the six variables were most important to the models.

To make this model useful enough to be worth applying, I would aggregate the ZIP codes into bigger neighborhood clusters, and perhaps add borough as a second collection of one-hot-encoded variables. The same code could be used for both, a series of if-else statements in a loop that would look at the ZIP code for the property and based on the value, add a NEIGHBORHOOD and BOROUGH value to the row. It would take a fair amount of work to set up the best way to aggregate the 179 ZIP codes, so beyond the scope of this project, but the code could be re-used for any location-based models for the city.

PCA for Pre-Processing

I had hoped that pre-processing the data with PCA would yield some improvements, but the results looked similar to the scaled data when I visualized it. However, the dendrograms did seem to show distinct clusters that suggested there were differences in the data and the scatter plots also showed similar points bunched together in sideways-V shapes. That said, the points were very tightly packed. It wasn't until I looked at the dendrograms that I got a better sense of the data distribution.

One difference with the scaled data, other than the orientation of the "corner" of the dot distribution, was that there were more points in a cluster that visually appeared closer to a different cluster than to the one it was labeled with.



That said, when I ran the dendrograms for both scaled and PCA, they were similar, so the variation I saw in the scatter plots doesn't seem to make as much difference when you use the dendrogram visualization.

I didn't pay enough attention to the details on the ARI and Silhouette portion of the assignment, so I didn't have all six of the combinations. That said, the results were also generally terrible, like the other models I evaluated.

Summary

My initial instincts when I was working on cleaning the data seem to bear out: The data I excluded because it was too complicated to use given the number of options for each categorical variable was probably required for any of these models to perform well. My biggest takeaway from the series of projects is that until I can add in more features to the dataset in a way that's useful in machine learning models, I won't know if the problem is my lack of data or that the underlying challenge of classifying property types based on features isn't something where machine learning can be a useful tool.