Capstone

Metro Portland, OR Housing Prices

Creating a predictive model

Mitchell Rosenfeld

Data Scientist

The prices of homes across the United States do not seem to be trending down. The prices for homes keep going up, as wages stay consistent. Even if you are financially ready to put down a deposit, do you know if you are getting the best deal? Have you done enough research to know what you want, where you really want to be? Does your real estate agent have your best interests in mind? As someone who doesn’t want to keep paying for rent and wants to own something one day, I wanted to find a way to look at my current market, Portland Oregon, and determine what are the best prices for the surrounding area. To do this, I will be making a model based on previous sales data to determine future housing prices. I will be telling you my process of finding my data, how I played around with it, and what models I tried making to perfect something that can help future home buyers.

I first started looking for data on government websites, but I couldn’t find anything specific enough to get the answers I desired. With a simple google search, I was able to find a couple real estate websites with some datasets. Redfin.com had the best dataset to fit my needs. At first glance, the dataset had over four million rows and 58 features. These features were repetitive a bit. A feature such as price per square foot held other features with the same data but given as month over month and year over year averages. What was also interesting about the data was what my target variable would have to be, median sale price. It seemed the data provided date ranges, 30 day or month ranges, of all sales that happened between 2013 and the current year. This dataset is updated monthly on Redfins website. I also found out that this data wasn’t fully cleaned. Although it had no duplicates, 30 percent of the features had 10 percent of its data filled with nulls. My plan to action here was to first filter down the data to be just the cities I would need, fortunately there was a feature specific to metro areas that allowed me to do this easily. I would then look at the columns I believe would be useful and drop the ones I believe I didn’t need. I was looking to take an agile approach to my cleaning, just get something down for now and we would fix it up later if need be. I would then backfill any features that were missing data. The last step would be to one hot encode a couple features, specifically property type and city name. With that done, I was off to further analyzing.

Chart, line chart

Description automatically generated I first wanted to look at the features I had and try to see what correlations there were between them. To do this, I created a seaborn heatmap. There were very few features with great correlations between them. What I classified as “great” was anything above 0.3. Due to this cutoff, three features were removed. After completing the correlations, I felt prepared enough to run a simple linear regression model on the data. My first results seemed promising as my R-squared was 0.75. I found that many features though held high P values. Because of how many there were, I made a for loop to drop features with P values at certain levels and create a list of their respective R-squared values once dropped. R-squared values can be seen on the diagram to the left labeled A. It turns out that the more columns dropped made the R-squared value go down. It seems very minimal still though as the most it dropped was by 0.005. From here I decided to conduct a test train split and continue to work on a linear regression model, along with any other model I could think of.

A

I split by data up by year. Anything from 2020 and up with my test, and the rest of the data was my train. The testing split equated to around 20 percent of all data. This was a major mistake I found later on, as the model would not be able to fit years it has never seen. I will still present my findings but explain how I fixed the issue further down. My linear regression was extremely under fit, with a test R-squared of 0.38 and a train of 0.79. I tried Laso and ridge regression, which gave around the same results as linear regression. A decision tree regression gave interesting results with a train of 1 and a test of 0.24, the train could not predict the unseen data very well at all. I tried a few other models, but the one I want to highlight is Random Forest Regression. I was able to get a train of 0.92 and a test of 0.4. With the test being higher than the one from Laso, ridge, and linear regression, I thought I would keep looking into it.

Chart, line chart

Description automatically generated After realizing my mistake with how I split my data, I decided to perform a proper test train split. Before I did this however, I also realized my data had some data leakage. I kept a feature in my model called median price per square foot. I had to be removed, as it is just a calculation based on the price of houses sold. After completing these two steps, I tried modeling again. I tried linear regression first, and found the model extremely overfitting, with a train of 0.36 and a test of 0.72. I received around the same results for a Laso regression. With Laso, I thought I could change the alpha hyper parameter and it might help better fit the model. I took a range of alphas from .0001 to 1000 and created a for loop to calculate R-squared values for my test and train data. You can see the results to the right labeled B. Increasing or decreasing the alpha parameter did not change the outcome of the model for the better, increasing it made it perform considerably worse. The last model, which I had the highest hopes for, pumped out a test of 0.93 and a train of 0.82. These results seemed a bit too good to be true. To test this, I ran a 5 fold cross validation on the train and test sets. I end up finding that same overfitting with an average train of 0.55, and a test of 0.72.

B

I believe these results came from how I originally cleaned my data, so I would go back to the top and start from scratch.

Looking back at my data, I realized I took out two features that should have been kept in from the beginning, median list price and median list price per square foot. I took these out before because I didn’t see them as much different from just the sell price. Although that could be the case, my logic to adding it back was whenever you look up houses on the market, the list price is the easiest piece you can find. If someone were to plug in data on the current market to calculate what current prices are, that would be a relevant feature. I also added back the number of homes sold within a given month range and city, as well as inventory, or number of houses available on the market within a month and city. With these all added, and backfilled for null values, we can go and try modeling again. Our linear and Laso models are still perform around the same, with a train of 0.68 and a test of 0.72. There is Overfitting still, but very minimal. Our random forest seemed it was doing good as well, with a train of 0.98 and a test of 0.84. I wanted to test this again, so we performed another 5 fold cross validation, giving out a train of 0.82 and a test of 0.7.

Chart, box and whisker chart

Description automatically generated The last step before looking at a single model and trying to evaluate it was dropping outliers in my data. Something I didn’t touch yet was that there are homes, or complexes I believe, in this dataset that go for multi millions of dollars. I created a box plot to show the variation of median sale price. You can see it to the right labeled C. There is a fair number of multimillion-dollar homes as well. It’s hard to see, but the boxplot represents 991 outliers. That is a fair amount of my data that I did not want to give up. I believe multimillion-dollar homes should be represented to an extent, so I decided to look at any home prices that exceeded 3 standard deviations away from the mean. I found there were 386, which is far less than 991, and should take out the real extremes. With that down, we iterate again with modeling.

C

Graphical user interface

Description automatically generated with low confidence Random Forest Regression kept performing the best with all iterations of modeling I performed, so I am only going to focus talking out its results. The last test performed on it, without any hyperparameter tuning, gave R-squared results of 0.98 for test and 0.85 for train. Overall, all though it was just a little, taking out some outliers did make the model better perform for both train and test data. I tried a bit of hyperparameter optimization looking at min sample split, max depth, and a couple others. Almost all really showing adjusting any of them would make my model worse. I have some charts representing these finding that I will share in an appendix at the end l labeled D through F. The only adjustment that I feel could work for me was looking at max features. You can see my findings to the left labeled G. Although it doesn’t affect the model too much, I found by having only 50 percent of my features would generalize my model better, at least for training purposes. I don’t believe this should be included still though because most of the columns being removed are most likely cities, as the features represented are mostly one hot encoded from my original city column. Without certain cities, if people were to look up prices for those location, they would be inaccurate, but at the same time, if my model is saying they are not relevant, most likely those cities would have inaccurate prices anyways. For now, I will keep the model the way it is.

G

To really evaluate the model, we are going to look at the mean absolute error. We can see the model is giving an MAE of $38,220.89. This might seem like a lot, but it’s very relative to the area you are looking to purchase a home in. Our above box plot shows a mean price of houses around 3 to 4 hundred thousand dollars. The closer you get to Portland though, it is well known that prices are going to sky rocket up. This is a fact for really any heavy populated city. For someone looking for a house around the price of 800k to one million, 40 thousand dollars up or down might be a bit more statistically relevant to them for when to buy. For someone with a lower income, looking for a house between 3 to 4 hundred thousand, the model might not be helpful, at least not yet.

Chart, histogram

Description automatically generated I want to look at the features giving most of the weight in calculating home prices. I created a chart showing the top 10 features and their percentage of importance to the model. It can be seem below labeled H. What can be first seen is that median list price takes up over 40 percent of the calculation, which isn’t too surprising. Housing being a Condo or Co-op is interesting to be the second largest determining factor. What truly is crazy is to see how everything besides those first two have so little importance. I believe this means there are many other factors out there that are just not represented in the dataset that have heavy effects on the current housing market.

H

If I was to continue fine tuning this model, I believe the most important step I could take would be finding other relevant datasets that are affecting the housing market and adding them in. My first thoughts on what to add would be features about the surrounding area, such as how many schools there are within a city. I could even see if the amount of covid cases effected how the market has been. I would want to dig deeper into any and all factors. I would also consider expanding the data to more than just the Portland area to help many other people. To do that as well, I would create a web application that anyone could just plug in features they are seeing within the market, and have my model calculate the best price range for the area.

**Appendix**

**Chart, line chart

Description automatically generated**

E

D

**Graphical user interface

Description automatically generatedChart, line chart

Description automatically generated**

F