Municipal and Market Housing Pricing: Using Machine Learning to Predict Home Values

Sergei Neznanov

Department of Economics, Rutgers University

570: Economics of Data Science

Dr. Yuan Liao

August 31, 2022

Abstract

Home prices have rocketed to their highest levels in history in the United States (Harrison 2022), as a result buyers need a tool to best predict the overall cost of their mortgage payment. As the mortgage payment is split between paying the bank the principal and interest on the loan and paying the government property taxes, I will attempt to find a model that accurately predicts both the bank's and the government's valuation of a given property. This would be a useful tool for consumers to be able to have an accurate prediction of their monthly costs once they have purchased a home and would be able to budget accordingly. I built this tool by using two datasets, one that would be used by the bank and the other compiled by the government, to select the best machine learning method to predict property values. After building machine learning models using Linear Regression, Decision Tree, Random Forest, and Deep Neural Networks, I found Random Forest to be the best overall model for predicting property pricing from both a government and market standpoint.

Introduction

A home is the largest purchase an average household will make in their lifetime. With the prices of homes having grown 20.4% year over year (Standard and Poors 2022), it is important to get a full view of the cost of homeownership.

A mortgage generally refers to the loan given by a bank to a borrower, but the mortgage payment itself is a single payment by the borrower to several entities (Bank of America 2022), with the bank and the government being the largest benefactors. These entities use different metrics to determine the value of a property, and thus can calculate differing values on the same home.

The government assesses a theoretical sales value on all properties for a given year, and bases its tax off that. The government bases its valuation as a neutral reflection of the property values, only taking into consideration hard characteristics, such as size, location, and physical characteristics.

The bank gives a loan based on actual market value of a home at the time of sale. This method of valuation is open to more subjective characteristics, including the location of the home relative to desirable landmarks or the racial composition of the neighborhood.

Goal

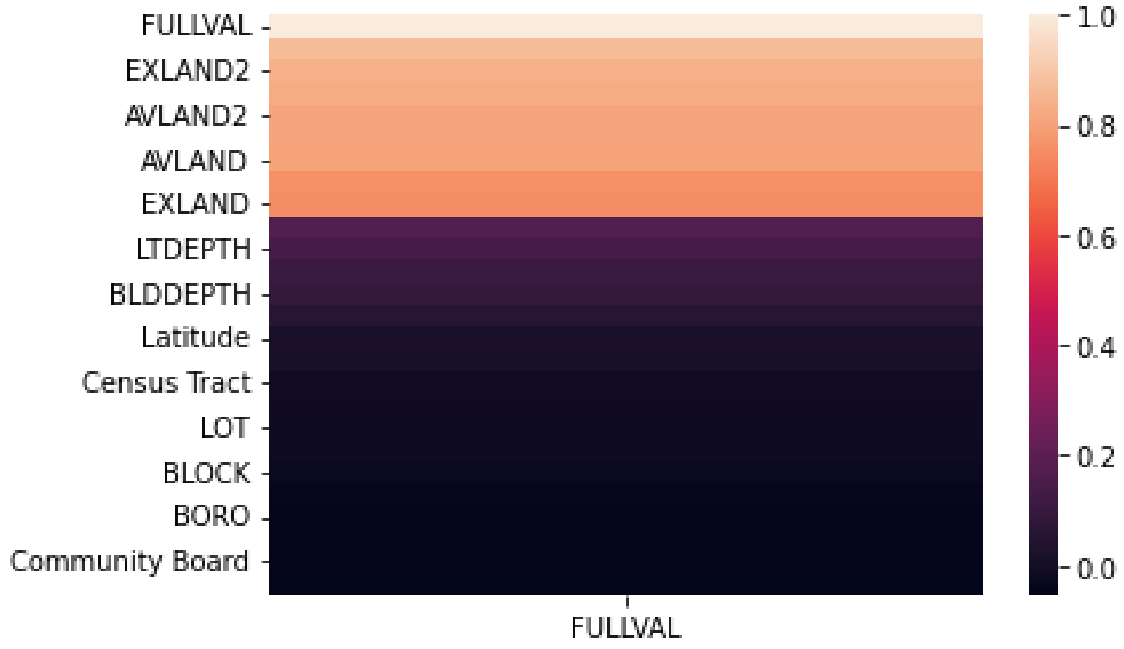
As the government and the banks use different methods to determine the value of a house, they will give different values for the same home. For this project, it is my intention to see if there is a single machine learning method which is best suited for both types of valuations. If there isn't, I hope to identify the reasons why one method may be superior to another for the given data set. To test this, I used two separate datasets, tested using identical methods to identify the best machine learning method to predict housing prices.

Data

As I've described, the two main entities which receive money from a mortgage payment are the government, in the form of property taxes, and the bank, receiving payment to the principle value of the house and the interest rate being charged.

I chose two separate datasets to train my models, the City of New York's Property Valuation and Assessment Data (City of New York 2022), which is a compilation the value and objective characteristics of every piece of property in New York City, and Kaggle's Boston Housing Dataset (Kaggle 2016). I chose these datasets because they reflect very similar cities, both being major population centers on the East Coast of the United States. Both are used by Standard and Poor's as part of their framework of the S&P CoreLogic Case-Shiller Home Price Index (Standard and Poors 2022). Prices and demand are similarly high in both cities, as they are in the top ten most expensive real estate markets in the United States (Dehan 2022).

The New York City data is a raw dataset compiled by the City of New York, and is publicly available through their NYC Open Data website (City of New York 2022). As such, it contains the characteristics of over 9.8 million properties in New York City. The variables are the borough, block ID, lot ID, the presence of an easement, the owner's name, building class, tax class, lot width, lot depth, extension indicator, number of stories in the building, market value of the property, actual land value, actual total value, actual exempt land value, actual exempt land total, exemption code 1, street address, post code, exemption class, building width, building depth, transitional land value, transitional total value, transitional exemption land value, transitional exemption land total, exemption code 2, assessment period, assessment year, Latitude, Longitude, Community Board, Council District, and Census Tract.

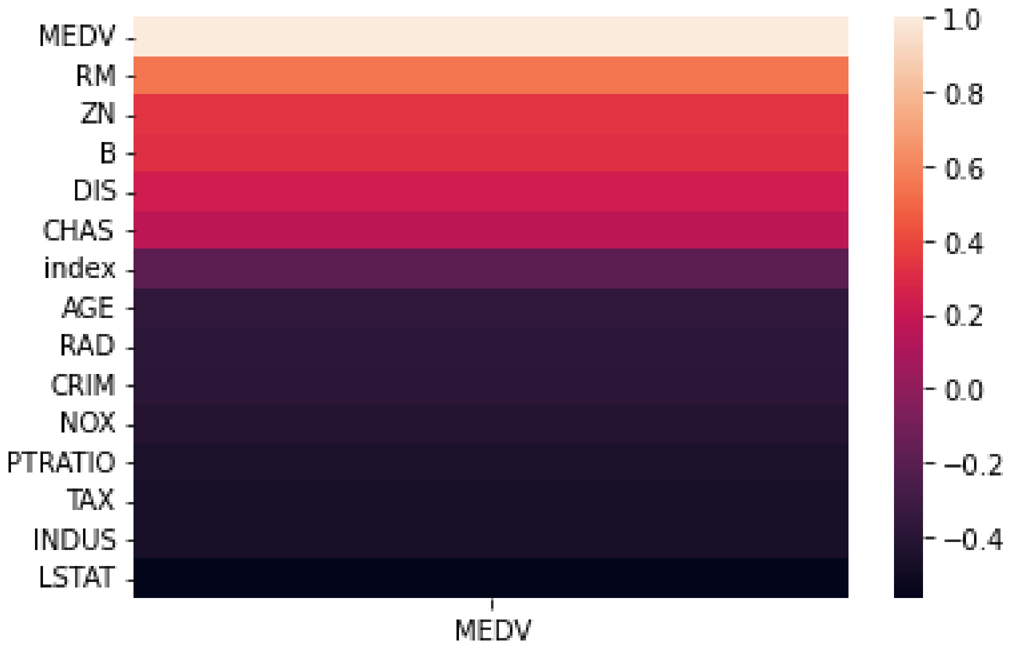


Seaborn heat map showing correlation with market value of the property for the NYC dataset.

As this would be too many variables to efficiently train a machine learning model, I ran a Seaborn heat map showing only the variables that positively correlated with the market value of the property. This reduced the number of variables to twelve, of which I chose the top four variables with which to train my models. They are: transitional exemption land value, transitional land value, actual land value, and actual exempt land value. This was surprising, as I'd expected the size of the building to be at least somewhat influential on the market value of a property, but with the scarcity of land in New York City, the variables with the most correlation to market value are all related to the land it sits on.

I further pared down the size of my dataset from 9.8 million to 10,000 randomly selected samples to train my models. This was out of practical necessity, as my computer doesn't have the resources to functionally analyze that many inputs to generate a model. I ran the random selection once, saved those results, and loaded the same output to each machine learning method, to ensure that the same random survey would be used to train all models.

The Boston dataset was a prepared dataset from an existing Kaggle competition (Kaggle 2016), thus there was much less data cleaning necessary to proceed with training my models. The Boston dataset is much sparser than the NYC dataset, only containing 506 samples, with 14 variables. The variables are the per capita crime rate by town, the proportion of residential land zoned for large lots, the proportion of non-retail businesses acres per town, proximity to the Charles River, nitrogen oxides concentration, the number of rooms in the building, age of the building, weighted mean of distances to five Boston employment centers, an index of accessibility to radial highways, the full value property tax rate, the pupil-teacher ratio by town, the proportion of African-Americans per town, the proportion of lower status residents, and the market value of the homes.



Seaborn heat map showing correlation with market value of the property for the Boston dataset.

I also ran a Seaborn heat map to help narrow down which variables to train my machine learning models. Since I had a much smaller sample size, I chose my five variables which correlated most strongly with the market values of the homes: number of rooms per house, the proportion of residential land zoned for large lots, the proportion of African-Americans per town, weighted mean of distances to five Boston employment centers, and the proximity to the Charles River. The number of rooms, zoning for large lots, distance to work, and being near the Charles River made sense to me, as I would associate a large home, on a large piece of land, near work, in a picturesque location would command a higher price. I was, however, surprised to see that given the choice of crime rate, teacher-pupil ratio, and pollution, the proportion of African-Americans in a given town had the highest correlation with the price of the house. This provided numerical evidence for Boston's reputation as America's most racist city (Adams 2021).

As this dataset only had 506 samples, there was no additional data cleaning necessary to reduce it to a functional length for machine learning.

Methodology

Given the vast differences between the two datasets, I chose four machine learning methods to train in an attempt to find the best one for predicting home values as either a buyer would, to present to their bank for a loan, or as the government does when they're calculating the tax value of the home. The four models are Linear Regression, Decision Tree, Random Forest, and Deep Neural Networks. I also chose to use two methods of splitting the data to train the models, train-test split and k-fold cross validation. I used the r^2 score as the methodology for testing the accuracy of my models.

Total Data Set	
Testing Sample	Training Sample

Test-Train Split

A train-test split is a relatively simple validation method. It simply splits a preset percentage of the total dataset into a testing sample, the rest being reserved for training. The user defines X values used to train the model, against Y values being used to predict the predicted value. For my NYC Dataset, I used transitional exemption land value, transitional land value, actual land value, and actual exempt land value as my X Values, trying to predict my Y value, total market value. For my Boston Dataset, I used number of rooms per house, the proportion of residential land zoned for large lots, the proportion of African-Americans per town, weighted mean of distances to five Boston employment centers, and the proximity to the Charles River as my X values, trying to predict my Y value, the market value of a given home. I set my train-test split to a 20% testing sample, with 80% being used to train the models for all models and datasets. I also set the random state to 973 for all models and datasets, ensuring the same composition of training samples and testing samples throughout the process.

Total Data Set					
Testing Sample	Total Training Sample				
Fold 1	Testing	Training	Training	Training	Training
Fold 2	Training	Testing	Training	Training	Training
Fold 3	Training	Training	Testing	Training	Training
Fold 4	Training	Training	Training	Testing	Training
Fold 5	Training	Training	Training	Training	Testing

K-Fold Cross Validation

A k-fold cross validation works by iterating over the same data set k times. The image above shows a model where k equals five. During each iteration, a small subsection of the total training sample is removed and used as a validation sample against the remaining training samples. This ensures that each sample in the dataset has an opportunity to be tested and trained on, this quality being especially useful in a sparse dataset, I expect that as a result the k-fold validation should be more accurate than the train-test split. The disadvantage of using k-fold validation is that it is a more resource intensive process than a regular train-test split, and I noticed this in particular with the NYC dataset taking a noticeably longer amount of time to run the machine learning methods for k-fold cross validation than the train- test split. For my NYC Dataset, I used transitional exemption land value, transitional land value, actual land value, and actual exempt land value as my X Values, trying to predict my Y value, total market value. For my Boston Dataset, I used number of rooms per house, the proportion of residential land zoned for large lots, the proportion of African-Americans per town, weighted mean of distances to five Boston employment centers, and the proximity to the Charles River as my X values, trying to predict my Y value, the market value of a given home. To ensure consistency across my testing, I ran all my k-fold splits with k equaling five and my random state set to 973. Not changing the k value ensured that no model was trained any more or less than any other model. Setting the random state to one value ensures that every time the same dataset is split, the same individual properties would be used in every individual fold and separation from the testing and validation samples. This is to ensure consistency of inputs into the models.

$$r^2 = 1 - SS_{res}/SS_{tot}$$

r squared formula

r^2 is a measure of performance for a linear regression model. In the mathematical formula above, SS_{res} is the sum of the squares of residual errors and SS_{tot} is the sum total of the errors. A perfect score for the r^2 is 1, where all of the variability of the output can be totally explained by the model. The closer the score is to 1, the better the model is at predicting an accurate outcome. It is possible that the r^2 score is negative if the sum of the squares of residual errors is greater than the sum total of the errors. This occurs when the SS_{tot} is derived from only the input data and the SS_{res} is derived from both the model predictions and the input data. A negative r^2 score is then indicative of a notably poor functioning model for predicting the desired outcome.

Linear regression as a machine learning method is a relatively simple protocol. When fed the training sets, the model would calculate weights to assign to the given variables to best fit the x values to predict y values. This should predict the y values in the testing set, when given the x values to predict from. I set the maximum number of iterations at 5000 for both datasets. For my Boston dataset, I received an r^2 score of -10.669 when using the k-fold validation and -9.032 when using the train-test split. This suggests that the training model overfit, and the testing parameters returned values that the model functioned worse than a random guess for final property values. For my NYC dataset, I received an r^2 score of -71.265 for both k-fold validation and train-test split. This model was even more extremely overfit than the model for my Boston dataset. As neither the NYC dataset nor the Boston dataset returned statistically accurate r^2 scores, I am inclined to disregard it as a potential model to estimate both government and bank determined values for mortgage payments.

Decision tree as a machine learning method is a slightly more complex method than linear regression. A decision tree predicts the target value based on learning rules from the input data. Decision trees may are also prone to being overfit. For my Boston dataset, I received an r^2 score of -78.478 for the k-fold validation, and an r^2 score of -63.665 for the train test split. This shows me that the using the k-fold validation particularly overfit vs the train-test split. For my NYC dataset, the r^2 score for the k-fold validation was .641 and my r^2 score for the train-test split was .459. Both of these scores are much better than the scores for the linear regression testing. Compared to the Boston dataset I'm inclined to conclude that the 10,000 value dataset for NYC had the advantage of having a smaller proportion of outliers than the 506 entry Boston dataset, where any outlier would have an outsize effect on the model. In this case, the decision tree seems to be a better machine learning model for government prediction of property values, but not for banks.

Random forest is an evolution of the decision tree classifier. Random forest functions by fitting decision trees over subsamples of the dataset, and finds the averages of those subsamples to improve accuracy and reduce overfitting. I set a constant random state for all random forests, which controlled for the randomness of the sampling, ensuring each model would sample from the same subsets of the dataset. I then ran three different values for the number of trees in the forest to see which one would provide the most accurate model. First I set the number of trees at 1000. For the Boston dataset, the r^2 score was .724

for the k-fold validation and .509 with train-test split. For the NYC dataset, the r^2 score was .700 for k-fold validation and .584 for train-test split. I then reduced the number of trees to 500. For the Boston dataset, the r2 score was .729 for the k-fold validation and .509 with train-test split. For the NYC dataset, the r2 score was .702 for k-fold validation and .581 for train-test split. I then increased the number of trees to 1500 . For the Boston dataset, the r2 score was .724 for the k-fold validation and .509 with train-test split. For the NYC dataset, the r2 score was .698 for k-fold validation and .585 for train-test split.

Compared to using single decision trees, in all cases the random forest machine learning method gave me significantly more accurate predictions. Also, in every case the k-fold validation proved to be more accurate than the train-test split. For both the NYC and Boston datasets, setting the number of trees to 500 gave the highest r^2 scores, .702 for NYC and .729 for Boston.

I used the Keras python extension for running my neural network. For all models, I set the number of epochs to 800. I started with a three-layer neural network, with each layer having 16, 8, and 4 nodes, respectively. For the Boston dataset, my r^2 score was .502 for k-fold and .438 for train-test split. For the NYC dataset, my r^2 score was .580 for k-fold validation and .577 for train-test split. I then tested another three-layer neural network, with each layer having 16 nodes. For my Boston dataset, my r^2 score was .527 for k-fold validation and .497 for train-test split. For my NYC dataset, my r^2 score was .506 for k-fold validation and .493 for train-test split. Next I tested a nine-layer neural network, with each layer having three nodes. For my Boston dataset, my r^2 score was -.005 for k-fold validation and -.03 for train test split. For my NYC dataset, my r^2 score was -.044 for k-fold validation and -.023 for train-test split. Having found that a lower number of nodes, with more layers wasn't the right direction, I returned to a three-layer neural network, and increased the number of nodes to 20. For the Boston dataset, my r^2 score was .520 for k-fold validation and .454 for train-test split. For the NYC dataset, my r^2 score was .710 for k-fold validation and .490 for train-test split. I maintained the three-layer neural network, increasing the number of nodes to 25. In the Boston dataset, the r^2 score was .486 for k-fold validation and .525 for train-test split. For the NYC dataset, the r^2 score was -.134 for k-fold validation and .108 for train-test split.

Overall, the best performing setup for Boston was a three-layer neural network, with 16 nodes per layer. This returned an r^2 score of.527 for the k-fold split. The best performing setup for NYC was a three-layer neural network, with 20 nodes per layer. This had an r^2 of .710 with the k-fold validation.

Conclusion

Using two very different datasets, I was able to find some overall trends in using machine learning to predict price. Simple models, such as linear regression and decision trees proved to be too susceptible to overfitting and interference from outliers, and returned r^2 scores which were essentially useless for calculating the value of a property, which could be used by both the government in calculating property taxes and the bank in determining the market value of a home to loan a mortgage. More complex models, such as random forests and neural networks were significantly better at predicting home valuations for the banks and government. The single best model for predicting based on the government's criteria was a neural network with three layers, and 20 nodes per layer, with an r^2 score of .710. But, the overall best model for predicting both the bank and government's valuation was the random forest with 500 trees, with r^2 scores for the bank's classification, using the Boston dataset, of .729 and the government's r^2 score, using the NYC dataset, of .702. Overall, the k-fold cross validation served to be more accurate than the train-test split, as was expected because of its more thorough training.

When the consumer is trying to identify a property with the lowest overall mortgage payment, I would suggest using a random forest model, as it proved to be most accurate at predicting both the government's valuation of a property and the bank's. Especially in the current climate of high demand for housing and high interest rates, the prices of real estate are at historic highs, so buyers are particularly price sensitive.

References

Adams, Dart. 2021. "Is Boston America’s Most Racist City? Ask a Black Bostonian for Once." *Boston Magazine*, September 9.

Bank of America. 2022. "Sergei Neznanov's Morgage Statement for the Month of August." Charlotte, August 1.

City of New York. 2022. "Property Valuation and Assessment Data." *NYC Open Data.* May 9. Accessed June 9, 2022. https://data.cityofnewyork.us/City-Government/Property-Valuation-and-Assessment-Data/yjxr-fw8i.

Dehan, Andrew. 2022. *10 Most Expensive Cities In The US.* August 23. Accessed August 30, 2022. https://www.rocketmortgage.com/learn/most-expensive-cities-in-the-us.

Harrison, David. 2022. "U.S. Home Prices Hit Record of $416,000 in June as Sales Continued to Slide." *Wall Street Journal.* July 20. Accessed August 31, 2022. https://www.wsj.com/articles/u-s-home-sales-fell-again-in-june-economists-estimate-11658309401.

Kaggle. 2016. "Boston Housing." *Kaggle.* June 2. Accessed June 10, 2022. https://www.kaggle.com/competitions/boston-housing/overview/description.

Standard and Poors. 2022. *S&P CoreLogic Case-Shiller.* New York: S&P Dow Jones.