Makenzie Barber

Course Project

9 August 2022

Predicting Real Estate Prices

Appendix

Cover………………………………………………………………………….Page 1

Executive Summary…………………………………………………………..Page 3

Methods……………………………………………………………………Pages 4-5

Interpretations, Integrity and Future Research……………………………….Page 6

References……………………………………………………………………Page 7

**Executive Summary**

At Barber Real Estate, we pride ourselves on providing our customers with the future value of what their home could be worth. In order to accurately do so, our company relies on us to provide them with the best metrics and tools to keep our customers informed.

In this report, I will test three models that essentially do the same thing, in different ways. These models produce algorithms that return accuracy metrics on the real pricing of houses in 2020, for the future. These predictions are the basis of what we do, so it is imperative we get it right. Today, we will walk through the GLM, GLMNet and Random Forest model to make accurate predictions based on past data. All three models were tested against data to be transformed, and data that remained the same. This control data is our holdout, and the transformed data is the train.

Pre-processing data is an essential step to produce results that could mean the difference in giving out bad financial advice, and great advice. Cleaning out correlating columns to avoid endogeneity, and running suggested transformations for later algorithms are just a few of the pre-processing steps that must be taken in order to retain accurate information. For this dataset, I removed the Tax column because it was too highly correlated with the Median value of owner-occupied homes. That is because every house price has a tax associated with it, as we know. Since that column would not be of much use today, it was cut. Our suggested transformation was a log10, so you will see those appear in the sequences of my tuning parameters to keep our data in line!

The GLM model was tested on the MEDV data, which is the Median Value of Owner-occupied homes (in hundreds of thousands of dollars). GLM was ran on a 3 fold, cross-validated algorithm and is great for plotting. This model did not create accurate predictions for our company, and should not be used for predictions in the future by any means.Narrowing the tuning parameters for this model would make it potentially useful for our company.

The GLMNet model is our second best, and is a viable option for creating predictions in the future, accounting for the miss it provides. The GLMNet model is great for thorough, accurate predictions on the future pricing of houses. This model only misses the mean by 5.02, and the predictions typically miss by 7.14. Considering our actual data is in hundreds of thousands, this miss is considered low and fair.

The Random Forest model is our last, most accurate model created. This model is by far the most accurate. It only misses 4.13 times on the raw data, and is 3.01 times better than the GLMNet. Again, these misses might seem high, but with tax and inflated prices added in, it essentially will not make a difference as our median set is in the hundreds of thousands of dollars. The Random Forest model should be the model we use for all predictions going forward if we want to provide our customers with the best home ownership advice on the market!

**Methods**

Shape, rectangle

Description automatically generated To begin understanding the Real Estate data in order to make predictions for prices, we must first go through the steps of making sure the data frame we have is something we can work with. I began with finding correlation between columns. It is easiest to nest a function to omit NA’s within the cor() function and takes care of two problems at once. This function returns the data frame with zero NA’s, and from here we can further cleaning. Next, I created an empty vector to store any columns with correlation between other columns to avoid an endogeneity problem. This function reveals that one column, ‘Tax’ is highly correlated with ‘Rad’ as displayed in Table 1. I removed this column so that any future predictions will not be influenced by the correlation in the ‘Tax’ column. To continue the data process, I checked for variance between all rows and columns, and discovered that this dataset was greatly varied and ready for use. ran suggested transformations, and log10 was returned, so this will be the basis for my tuning parameters later on. This dataset does not contain any categorical variables, so no further processing was required. Next, we will begin forming vectors for predictions.

The variable in which I will be predicting is called MEDV, which is described as the median value for homes in thousands of dollars. To begin pre-processing this data, I created a train, a vector with a reserved percentage of the data from the MEDV column for the model to learn off of. The second part of the train is the holdout, which is a vector with a percentage of data that will be untouched and will be useful for validating models against the train data once it is finished. The train today will be 40% of the data from the data frame.

Shape, rectangle

Description automatically generated Making predictions for any dataset begins with creating your train control, this protects your model from overfitting or accidentally computing numbers that do not line with your data. These will be our tuning parameters, and they help us construct data with integrity and interpretability. Today, these tuning parameters will help us model a 3-fold, cross-validated, generalized linear model. The 3-fold chops our train data up in 3 sections to run through and cross-validate against all 3 sections. This GLM (Generalized Linear Model) was made using the train function, using data from our train vector. The method for this model was glm, which gives us our desired accuracy results, and a pre-processor of centering. This allows our data to be consistent and uphold the interpretability across all models we will build. Originally upon running this code alone, I came upon a na.fail code; I quickly fixed this with na.action; which excludes the NA’s and allows the model to fit. Finally, to create observations for the predictions, I isolated the results for the MEDV column from the train data. For the predictions, I used data from the glm model, and for the new data to test against I used the train data. For the accuracy measure, I used a a postResample function to test standard deviations throughout my predictions, and utilized the predict.trains function, which loops through the data to find estimates of variance. The results are presented in Table 2.

Shape, rectangle

Description automatically generatedA picture containing graphical user interface

Description automatically generated The next model I will be building is a GLMNet. This is like the model we just built, however, GLMNet uses a penalty term for large coefficients. Due to this, GLMNet explains as much variance in the data and is more thorough than a regular glm. For this graph, we will have defined tuning parameters, alpha and lambda, which will control our coefficients as mentioned above. For the model itself, we will again be predicting the MEDV column through a 3-fold repeated cross-validation method. For repeated cross-validation, after looping through the folds 3 times, the algorithm circles back and loops again. To find the lowest RMSE, I used the which.min() function on the results of the graph, which returned all metrics for the GLMNet, and is presented in Table 3, however I only included the metrics we will be comparing. The difference in the predictions for these two models is that the glm was tested against data from the train, and the GLMNet was tested against the holdout. I did this to compare the results and get the accuracy for how the train data tested against the holdout, and will be presented in Table 4.

Finally, the last model we will be building to test the predictions of house prices for the real estate market is a random forest model. This model builds decision trees based on the train data and is said to produce the most accurate results. This data begins with a tuning grid, with parameters of mtry(). This parameter is the number of variables randomly sampled at each split in the decision tree. For this model, as the number of variables sampled increased, the RMSE decreased. This is modeled in Table 5. Next, the model itself was predicted witText

Description automatically generated with low confidenceh the MEDV data, centered, and the method used here is “rf”, or random forest. The verboseIter is the training log for the algorithm and teaches the model how to fit. To interpret the accuracy for this model, I found the best tune and it is presented in Table 5.

Overall, the best method presented is a matter of comparing RMSE’s (typical misses) and seeing how the models line up. One obvious standout is the GLM model, as it did not return an RMSE, so the decision is between the GLMNet and the Random Forest. Both of these models hone in on thoroughness.

**Interpretation, Integrity and Future Research**

To begin with, the GLM model did not return an RMSE. I believe this issue was caused because of the way it was modeled. I used a log10 sequence for my tuning parameters, however, no matter how I changed them I was unable to get a value for this model. So, immediately, I cancelled out this model as a contender for predicting demand.

Next, we have the GLMNet model. The typical miss for this model was about 5.02, which means the algorithm misses the average data around 5.02 times. While that RMSE might seem high, the data for MEDV data is in hundreds, so it can considered close to accurate. When compared with the GLM model, there is no comparison, as the GLM did not produce an RMSE. When compared to the Random Forest Model with an RMSE of 4.31, however, there is a clear winner. When comparing the prediction accuracy for the GLMNet model, we can realize that the predictions tend to miss 7.14 times, which is 2.02 times more than the raw RMSE from the model. This can be interpreted as: When the model is created, and ran through the train data, it typically misses the mean of the data set 5.02 times. When predictions from that model are compared to data from the holdout, it typically misses 7.14 times. Therefore, we can conclude that the model itself would be better at predicting values for housing prices than the predictions themselves. I also believe it can be interpreted as not producing great predictions, and therefore could be adjusted through the tuning parameters for more accurate results.

Finally, we have the Random Forest models. This model builds decisions trees based on the number of random variables given to sample, and creates predictions based on what tuning parameter fits the model best. For this, we saw the model become more accurate with predictions as the number of variables sampled increase.

For the future, I would suggest this data to predict housing prices in a more updated economy. Even though this data was updated two years ago, it would not make great predictions for the current market we have for housing. The predictions could be compared against what housing prices rose to and tested how accurate each model really was. I believe that future models could be made to account for inflation, consumer preferences and the growing American population.

Overall, this dataset was close to perfect. There were missing errors that were not “NA”. This caused issues once it was time to model, as “NaN” and “Inf” codes were produced. I accounted for these issues by using a na.action function that allowed the original unction it was nested in to ignore the missingness. For our predictions for housing prices at Barber Real Estate, we should continue forward with the Random Forest model for all predictions.

**REFERENCES**

Ali, Arslan. “Real Estate Dataset.” *Kaggle*, 28 Sept. 2020, https://www.kaggle.com/datasets/arslanali4343/real-estate-dataset?resource=download.

Stevens, Brian. **“**Stat 201” *Youtube,* 10 July 2022. Accessed until 9 August 2022.

<https://www.youtube.com/playlist?list=PL-blpDu7mdw0pH7ixeY2MQIftNFO6b68T>