Kaggle: House Prices with Advanced Regression Techniques

Trevor Isaacson, Hawas Alsadeed, Evan Kessler

2/28/2022

Introduction

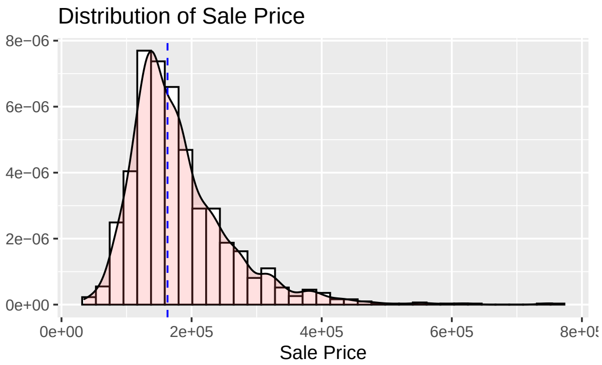
Imagine you're looking to purchase a home in the near future. How do you know the price of that home is accurate and fair. What factors or variables are you looking at to gauge the final price? For many, home prices are a number associated with a few key variables. For example, many buyers are looking at the number of bedrooms, bathrooms and the color of the fence. What other things affect the price of a home? Location, building materials, condition and quality are just a few that could potentially impact the price of a house. In this Kaggle competition, we look at home prices in Ames, Iowa and 79 variables associated with those homes. These 79 variables describe just about everything regarding these homes.

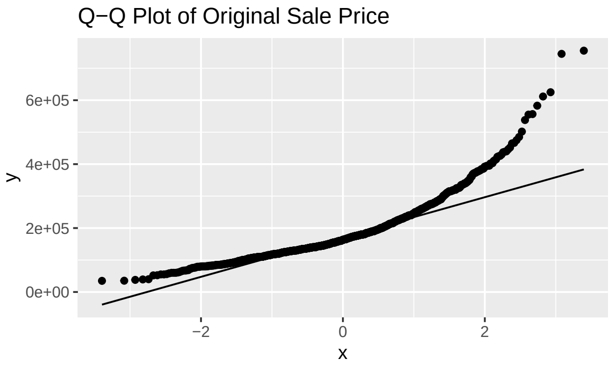
Problem Discription

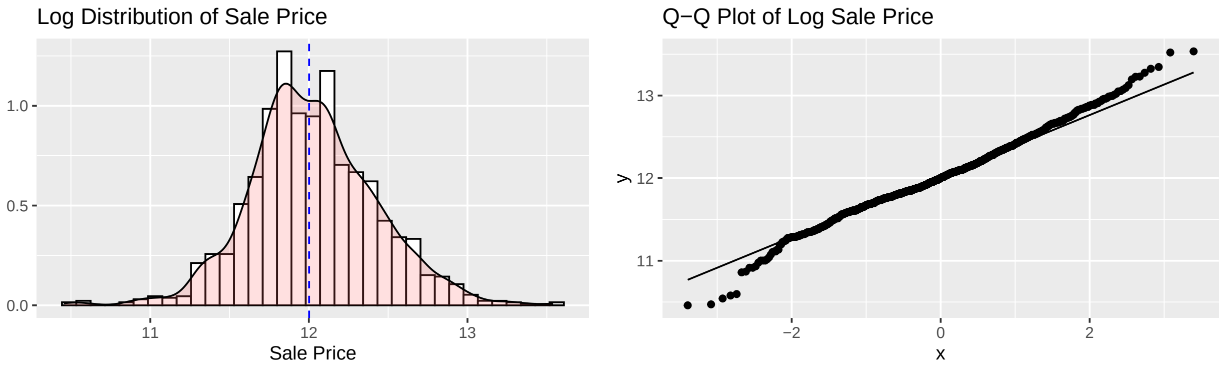
The problem is quite clear, can we accurately predict the price of a home? Given a lengthy set of variables regarding a house, are we able to predict the final sale price of that home? Using several different regression techniques, which method is able to best predict the final sale price and is it accurate enough to potentially use?

Data Cleaning

Because the original sales price data is very heavily skewed, we needed to log transform the prices. As shown in the histogram, we have a very heavy right tail because there are a some listings with very high prices compared to the median price of $180,921 (blue line). This non-normal shape and distribution is clearly evident in the Q-Q plot. By applying a log transformation, we fix this problem.







Additionally, after looking at several different variables we noticed there were some containing upwards of 1400 NA or 0 values in a dataset of 1460 observations. For some of them it made sense; such as fireplaces or half baths, a 0 value in this place just means the house simply doesn't have it in which many houses may not. In others however we noticed an issue could arise in fitting our models. We got rid of some of the values that were character values that had over 1000 of 1460 observations being NA. We figured this many values with NAs proved no relevance to our models, and only clogged up our GAMs and trees with unnecessary fittings. The other changes we made to our dataset was getting rid of NAs and categorizing a value of 'other' for variables with less than 5 values including Neighborhood, Exterior1st and Exterior2nd. Otherwise, we split our training data into a training and testing set 70% to 30% in order to check our models before submitting the final testing data to Kaggle.

Multiple Linear Regression

To begin, we started with a simple multiple linear regression model. We wanted to give ourselves a baseline root mean squared error value and because linear regression is the easiest to apply and interpret, we determined this is the best place to start. The model is fit using 73 variables and the training set. It reported a RMSE of 0.14 and some statistically significant variables include OverallCond, OveralQual, X1stFlrSF, X2ndFlrSF, WoodDeckSF, ScreenPorch and GarageCars.

```
## [1] "RMSE of Testing Set:  0.14"
```

Splines

After finding a baseline using linear regression, the next idea was to try fitting regression splines. When looking at some of the more explanatory variables, it was decided to perform a spline on the variables OverallCond, GarageCars, X1stFlrSF and X2ndFlrSF. The best performing spline was X2ndFlrSF with degrees of freedom of 2. However, none of the splines performed remotely close to our baseline. This was expected because there are over 70 variables in the baseline model and a single spline of a significant variable isn't likely to perform better.

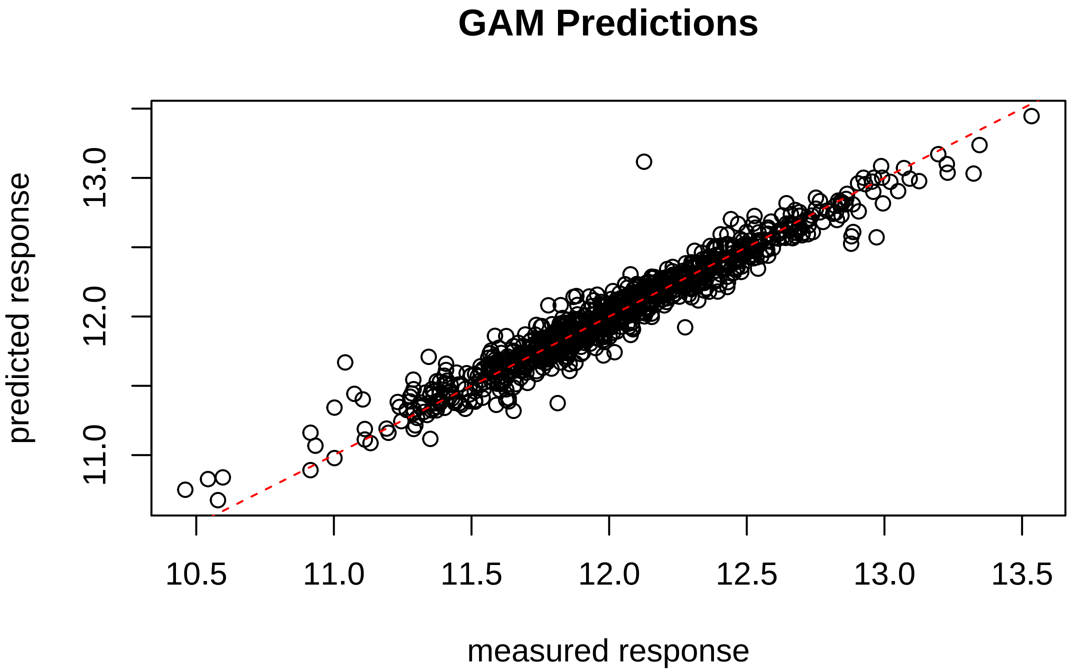
Splines	RMSE	RMSE_Dollars
X2ndFlrSF	0.4253	1.53
OverallCond	0.4456	1.56
X1stFlrSF	0.4695	1.60
GarageCars	0.4767	1.61

GAM Model

After fitting some regression splines, the next model we attempted was a General Additive Model or GAM. In general, this model is used for nonlinear relationships with splines on various variables. All available predictors were used to fit the GAM, along with the splines calculated above. There is definitely some improvement over the baseline as the RMSE = 0.1227 was below the baseline of 0.14. The final RMSE can be shifted depending on the which splines are used but, in general, this RMSE number is an improvement from our baseline model. The GAM combined with the splines is currently our best performing regression method.

```
## [1] "Test RMSE of GAM:  0.1227"
```

Below is a measured versus predicted plot. In general, our predictions follow the direction of the red target line with most points centered around the line. There are no visible splits in the predicted values and the measured responses. This compliments our lower RMSE score.

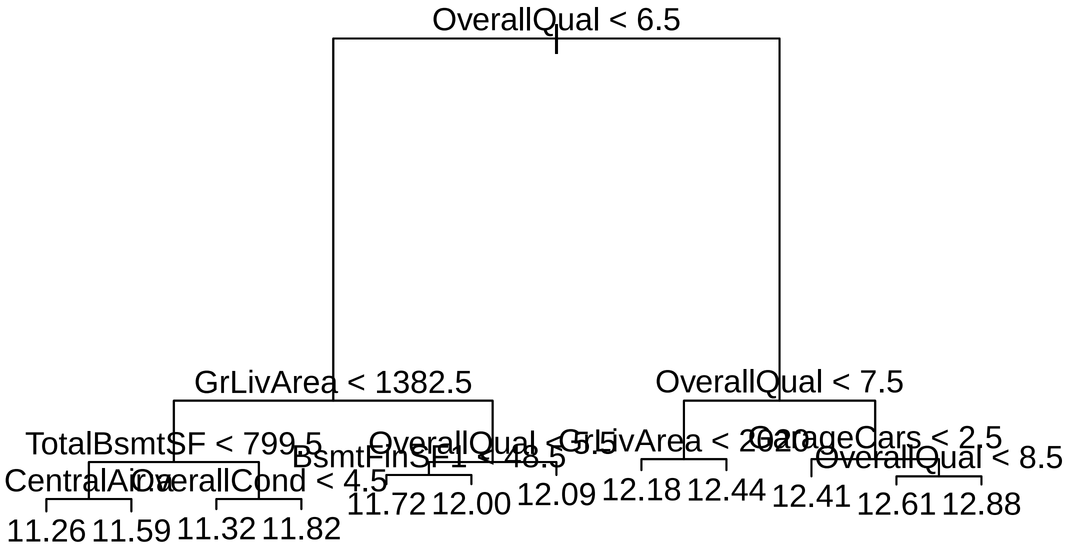


Trees, Bagging, and Random Forests

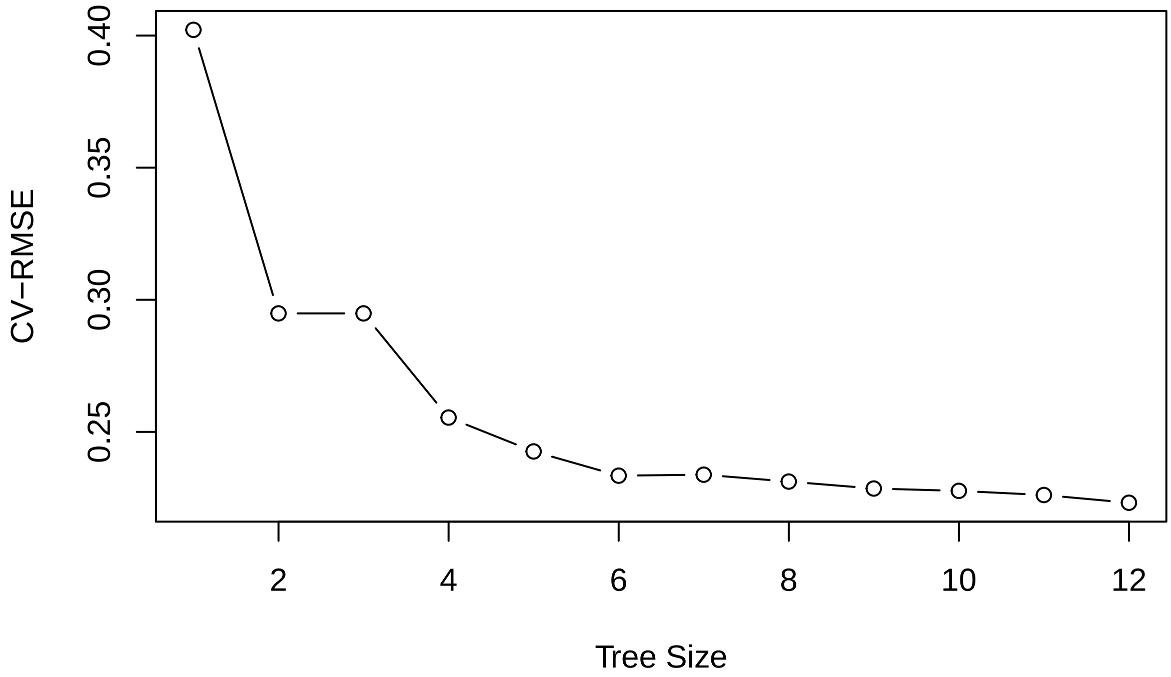
Finally we are doing to try and do trees, bagging, and random forests. To begin we fit a normal tree without any editing. From this it returned 6 significant variables out of the 80 we are testing. These included: OverallQual, GrLivArea, TotalBsmtSF, OverallCond, BsmtFinSF1, and GarageCars yielding 12 terminal nodes and a root mean squared error of .2135 (not the best). After this we looked to prune the tree to see what the best utilization of terminal nodes. After pruning we graphed the RMSE to terminal nodes to discover that anything above 6 would yield similar results. We noticed that throughout all of our terminal node sizes nothing we changed it to would change our RMSE significantly enough to care. Next we tried our luck at bagging.

```
##
```

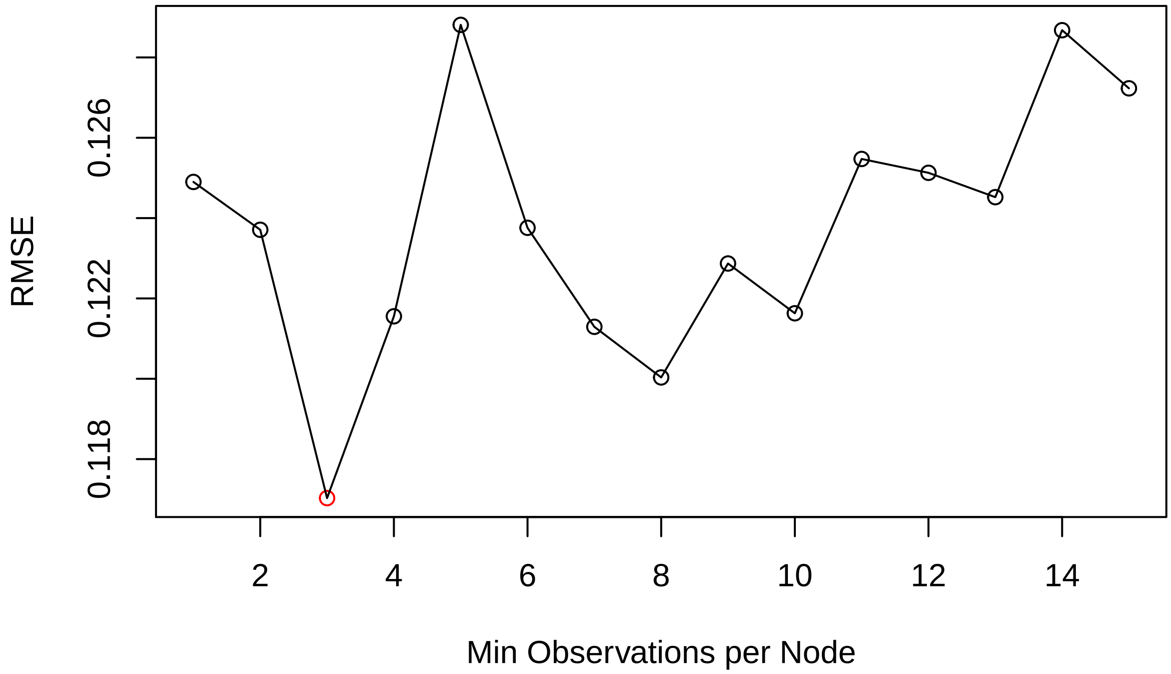
```
## Regression tree:
## tree(formula = SalePrice ~ ., data = training)
## Variables actually used in tree construction:
## [1] "OverallQual" "GrLivArea"   "TotalBsmtSF" "CentralAir"  "OverallCond"
## [6] "BsmtFinSF1"  "GarageCars"
## Number of terminal nodes:  12
## Residual mean deviance:  0.03781 = 38.16 / 1009
## Distribution of residuals:
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.803800 -0.111700  0.003485  0.000000  0.122400  0.652300
```

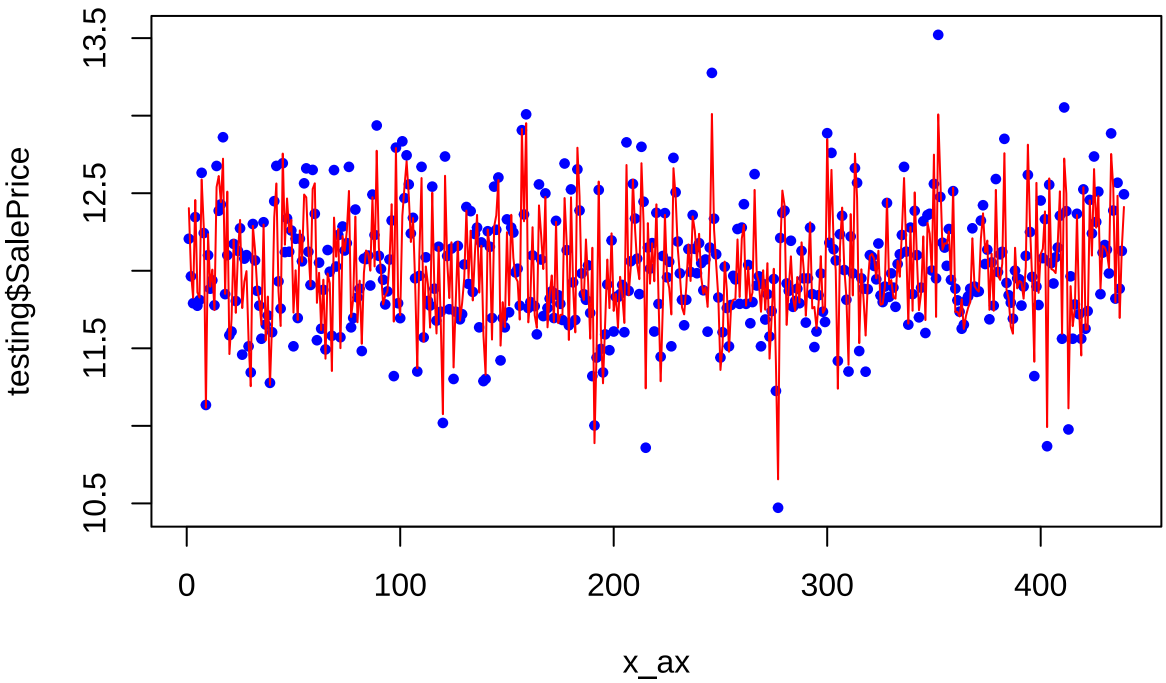


```
## [1] "Test RMSE of Tree:  0.208"
```



Before we start bagging we need to make sure to change all of our character values to factors. This is because our gbm model stand for gaussian bagging model and it doesn't take characters. In bagging we do it a little bit different than trees. We are not going to start out with a very basic model and prune it, we are instead going to use some of the information we learned from the previous trees. For example, our value for cv.folds in our gaussian bagging model holds the value of 7. This was chosen based off our CV.RMSE values in our graph above as 7 was around when the value of tree sizes begun to fall off and not change too significantly. The value shrinkage we left at the basic value of .1 as we didn't want to unnecessarily use too many trees. We set our trees at a large value of 1460 in order for the bagging model to use as many trees as necessary (most of the time it sets around 400-1000 before not reaching the shrinkage parameter). Finally, the last variable we used in our model was 'n.minobsinnode'; this was used as a constraint on the minimum number of observations in the terminal nodes of the tree. In order to find the best model we looped through n.minobsinnode being 1 through 15 and reported the best RMSE value. Below we will return three different graph. The first being a graph depicting the RMSE values of each bagging model. From this graph we can see that a value of 5 for our n.minobsinnode would return the lowest RMSE. Knowing this, we redid the model with this value in place and graphed the actual values of SalePrice (depicted in blue) and where are model went through predicting.


RMSE
0.118
0.122
0.126
2
4
6
8
10
12
14
Min Observations per Node



Just like with our trees we started with the most basic model we could in random forests, although we did add a few more parameters. One including the mtry, defined as the number of variables sampled as candidates at each split. We chose this to be the number of our columns divided by 3 because as stated in The Elements of Statistical Learning: Data Mining, Inference, and Prediction written by Trevor Hastie, Robert Tibshirani, Jerome Friedman, the text stated, “For regression, the default value for m is [p/3] and the minimum node size is five” (Haste 610). In doing our forest model we got a RMSE value of .14, considering this isn’t as small as our bagging was and the only reason to do forests over bagging is because of overfitting we can conclude that our bagging from before holds our best model.

```
## [1] "Test RMSE for Random Forest 0.1408"
```

Conclusion

Methods	RMSE	RMSE_Dollars
Linear Regression	0.1400	1.15
Splines	0.4253	1.53
GAM	0.1227	1.13
Trees	0.2080	1.23
Bagging	0.1170	1.12
Random Forest	0.1408	1.15

Conclusion

After fitting our model to the proper testData provided by Kaggle, we built our csv file and submitted it to the competition. We thought bagging would have proven to be our best bet for placement in the competition. It yielded our lowest rmse value using our training and testing data at 0.1170287. Based off of our training and testing above we were predicting placing around 100th place with a RMSE of 0.1170287. Because the dataset was different than what we were testing on, it could have been anything, but we were hopeful. After submission, we ended with a RMSE value of .135 placing around 1000th place.

With over 4,410 teams competing in this competition, we happened to beat ~75% of the other teams. Without overfitting, we think our results are accurate enough to potentially continue further research.

Github

All of our work can be found at: https://github.com/trevorisaacson3/KaggleHomePrices

Sources

Hastie, Trevor, et al. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2017.