Housing: The Price is Right!

Yizhou Zhang (yzhng114), Group member 1, Group member 2

DUE: Thursday, December 21, 2017 at 10:00PM

Abstract

We used all King County, WA home sales during May 2014 to May 2015 to develop a supervised learning model of home sale prices, in the hope of helping future home buyers/sellers make informed decisions. Models such as ordinary least squares, KNN, boosting, and random forests were considered. A linear ordinary least square model is found to be most accurate in terms of having the lowest RMSE.

Introduction

The price of a house is greatly affected by its conditions, such as the socio-demographic conditions of its size, number of bedrooms, condition, and year of construction. Its price is also greatly affected by the surrounding environment, such as the community's crime rate or the average sales price of surrounding properties. Furthermore, homesale prices also significantly depend on the year and month of the sale. Although the impact factors of housing prices are well known, it is important to quantify such relationships such that both home buyers and sellers would make have more accurate expectations of the houses and make informed decisions. Although the current study only investigated King County in Washington, its analytical approach and findings are generalizable to future studies covering the whole US.

Material and Methods

Data

The King County Dataset is a Kaggle dataset that contains data for homes sold in King County, WA between May 2014 and May 2015. There are a total of 21,613 homes recored for this time frame and 21 different variables:

Variable Name	Description
id	Unique ID for each home sold
date	Date of the home sale
price	Price of each home sold
bedrooms	Number of bedrooms
bathrooms	Number of bathrooms
sqft_living	Square footage of the apartments interior living space
sqft_lot	Square footage of the land space
floors	Numbers of floors
waterfront	If the apartment overlooks the waterfront or not
view	An index from 0 to 4 of how good the view is
condition	An index from 1 to 5 on the condition of the home
grade	An index from 1 to 13 for construction and design
sqft_above	The square footage of the interior housing space that is above ground level
sqft_basement	The square footage of the interior housing space that is below ground level
yr_built	The year the house was initially built
yr_renovated	The year of the house's last renovation
zipcode	The zipcode of the house
lat	Latitude

Variable Name	Description
long	Longitude
sqft_living15	The mean square footage of the interior housing space of the nearest 15 neighbors
sqft_lot15	The square footage of the land lots of the nearest 15 neighbors

We performed the following pre-processing of the data.

Factorize the zipcode. The geographic location of a house greatly impacts the price. We factorize the zipcode to control for neighborhood fixed effects.

Factorize the waterfront variable. The dataset contains a binary variable to indicate whether a house has waterfront view. We factorize that variable from integer to factor.

Factorize the condition variable. We factorize the condition variable.

Factorize the view variable. We factorize the view variable.

Factorize the grade variable. We factorize the grade variable.

Factorize the renovation variable. The yr_renovated variable specifies the year of the most recent renovation, if any. However, since a lot of houses never underwent a renovation (have zero for this variable), we transform it into a binary variable and factorize it (see Figure 1).

Factorize the basement square feet. Although the original dataset contains the square footage of the basement, if any, we factorize this variable into a binary indicator of the presence of any basement in the house for the sake of model simplicity (see Figure 1).

Extract the year/month of the sale. We used the substring function to extract the date and month information from the "date" variable and factorize them respectively. In this way, we control for the year and month fixed effects.

Interaction of longitude and latitude. We also added an interaction term between latitude and longitude of the house. This term, together with the latitude and longitude, should account for unobserved spatial characteristics that are not covered in the dataset. For example, houses in western King County, which is closer to the sea, should be more expensive than those in the east

Removal of bad observations. From our bedroom plot in Figure 1, we can there is an outlier with a house with 33 bedrooms. This is most likely a typo, so we will remove this point. We also found that houses with a grade of 1 or 3 are very rare and create several warnings in our various methods, so we will remove these observations as well.

Models

For our project, we ran through several different models with several different transformations to our data. Some of our models took extremely long to run. To combat this we saved our models in our **/models** folder. Then we set the actual chunks within this document that evaluates our models to `eval = FALSE` and loaded our models at the beginning of this document. If you want to to run any of our models, you are welcome to, but again they will take quite a while, especially GBM, random forest, and KNN.

Linear Model

For our first model, we wanted to use a simple linear model. We first used the Box-Cox power transformation to determine if we should transform our response (`price`) or not. The Box-Cox function in `R` creates normality plots (see Figure 2). The λ the corresponds with the maximum of our log-likelihood determines the power our response should be transformed by (i.e. $\lambda = 3$, would be our response cubed). For our first Box-Cox normality plot, our λ is close to 0, which indicates a log transformation. Then we run a Box-Cox on our log

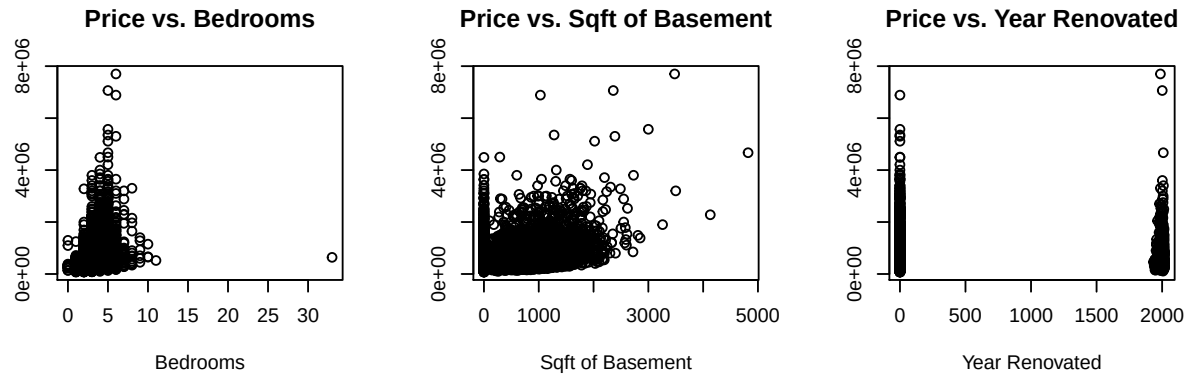

Figure 1: These three plots show our price plotted against three variables: number of bedrooms, the square footage of the basement, and the year a house was renovated.

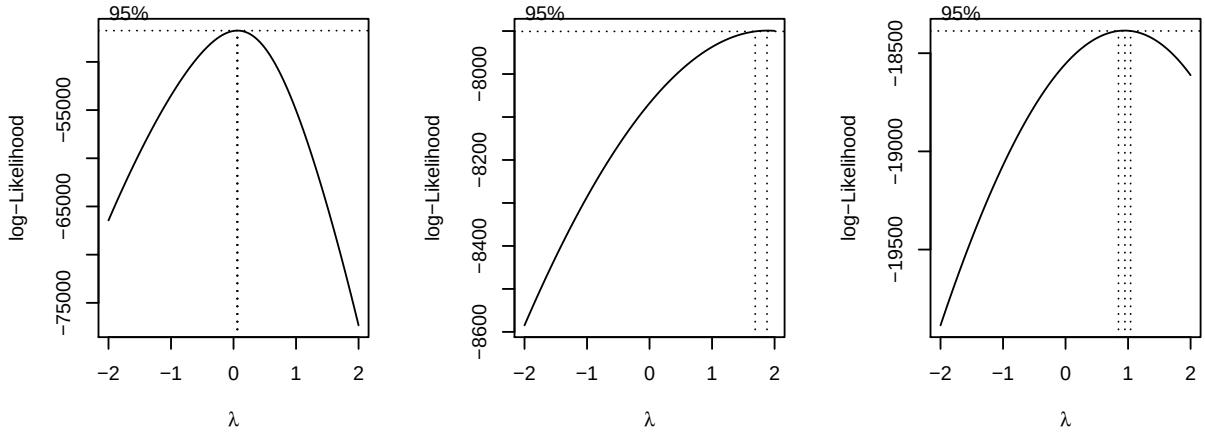

Figure 2: These are our Box-Cox normality plots using our full linear model. On the left is the Box-Cox normality plot for our untransformed response, in the middle is the Box-Cox normality plot for our log transformed response, and on the right is the Box-Cox normality plot for our log squared transformed response.

Table 2: These are the test RMSE values for our three linear models with various transformations on our response (price).

Models	Test RMSE
Linear Model	159140.7
Linear Model with Log Transform	143304.2
Linear Model with Log^2 Transform	126523.9

Table 3: These are the test RMSE values for our three lasso models with various transformations on our response (price).

Models	Test RMSE
Lasso Model	159242.0
Lasso Model with Log Transform	145298.7
Lasso Model with Log^2 Transform	127718.9

transformed response, our λ is near 2, which would indicate a square of our log response. Our last Box-Cox plot has a λ around 1 which indicates that our squared log response should not be transformed again.

To illustrate the benefit of this transformation, we ran a linear model on our centered and scaled predictors against all three transformation possiblies for our response: untransformed `price`, log of `price`, and square of the log of `price`.

As you can see in Table 2, our best model is the one that has the squared log `price` as the response. This is the linear model that will be used to compare with our other methods.

Lasso

We chose a lasso model, because our linear regression model seems to work rather well, so we wanted to see if a lasso model that shrinks unimportant predictors, might be able to beat our linear model. We again did all three possible response transformations and build lasso tuning grids using our λ_{min} and λ_{1se}.

The best lasso, model is again the one using the square of the log of our response (see Table 3). And our choosen lambda value is shown in Table 4.

K-Nearest Neighbor (KNN)

Next, we wanted to use the non-parametric K-Nearest Neighbor (KNN) method to predict from our data. As we discussed in class, a KNN model can drastically change when we scale our data. So, we ran each of our Box-Cox transformations in our *Linear Model* section above using scaling and not using scaling. We also ran our models with 5-fold cross-validation and with a `tuneLength` of 10. Our comparisons of our test RMSE for our KNN models are shown in Table 5.

Table 4: These are the tuned parameters of our lasso model with a squared log transformed response.

	alpha	lambda
Tuned Parameters	1	0.0057068

Table 5: These are the test RMSE values for our six KNN models.

	Scaled	Unscaled
Untransformed	192710.7	279213.2
Log Transformed	196152.2	289028.4
Log^2 Transformed	195652.2	288216.2

Table 6: This is our best k value for our scaled KNN without transforming our response variable.

	k
Tuned Parameter	5

The best result for our KNN model is scaling our predictors while keeping our response untransformed. For this model, `train()` choose the tuning parameter, k, shown in Table 6.

Boosting

We also tried several boosting methods: stocastic gradient boosting, eXtreme linear gradient boosting, and eXtreme tree gradient boosting.

Stocastic Gradient Boosting (GBM)

For GBM, we used 5-fold cross-validation to tune over the following tuning grid:

```
gbm_grid = expand.grid(
  interaction.depth = c(1, 2, 3),
  n.trees = (1:30) * 100,
  shrinkage = c(0.1, 0.3),
  n.minobsinnode = 20
)
```

As shown in Table 7, we performed a log transform of our response as well as the square of our log transform for our GBM. Our best is our GBM with a log transformed response, with the tuning parameters shown in Table 8.

eXtreme Linear Gradient Boosting

For eXtreme linear gradient boosting, we also performed 5-fold cross-validation to tune our parameters. Our tuning parameters were the default values chosen by `caret`.

As shown in Table 9, we performed a log transform of our response as well as the square of our log transform for our eXtreme Linear Gradient Boosting. Our best is our eXtreme Linear Gradient Boosting with a log

Table 7: These are the test RMSE values for our two random forest models with various transformations on our response (price).

Models	Test RMSE
GBM with Log Transform	128104.8
GBM with Log^2 Transform	130189.6

Table 8: These are the tuned parameters of our gradient boosted model with a log transformed response.

	n.trees	interaction.depth	shrinkage	n.minobsinnode
Tuned Parameters	2600	3	0.1	20

Table 9: These are the test RMSE values for our two extreme linear gradient boosting models with various transformations on our response (price).

Models	Test RMSE
eXtreme Linear Gradient Boosting with Log Transform	131600.9
eXtreme Linear Gradient Boosting with Log^2 Transform	145769.1

transformed response, and with the tuning parameters shown in Table 10.

eXtreme Tree Gradient Boosting

Our last boosting method is the eXtreme Tree Gradient Boosting method, we again allowed `caret` to specify the tunning grid and used 5-fold cross-validation to tune our models.

As shown in Table 11, we performed a log transform of our response as well as the square of our log transform for our eXtreme Tree Gradient Boosting. Our best is our eXtreme Linear Tree Boosting with a squared log transformed response, and with the tuning parameters shown in Table 12.

Random Forest

Lastly, we wanted to use an ensemble method to evaluate our data and we chose to use a random forest. For this method, we only chose to check our model for an untransformed response and a log transformed response. This was mostly done to save time, since we did not see any improvement of our model by log transforming our response. We choose to run our random forest with `ntree = 150` and with `mtry` values from 1 to 20. We tuned our random forest using out-of-bagging instead of cross-validation, as suggested in class.

From Table 13, we can see that our test RMSE does not improve when we log transformed our response, thus we will use our untransformed response in our random forest model. Our tuning parameter chosen for our random forest is shown in Table 14.

Results

The test RMSE for our various chosen model described in our *Methods* section can be seen in Table 15. From this table we can see that our linear model with a squared log transformation of our response variable has the lowest test RMSE, followed closely by our lasso model, our GBM model, and our eXtreme Tree Gradient Boosting Model. Both our KNN and random forest do not seem to perform well for our data.

Table 10: These are the tuned parameters of our gradient boosted model with a log transformed response.

	nrounds	lambda	alpha	eta
Tuned Parameters	100	0.1	0.1	0.3

Table 11: These are the test RMSE values for our two extreme linear gradient boosting models with various transformations on our response (price).

Models	Test RMSE
eXtreme Tree Gradient Boosting with Log Transform	129103.3
eXtreme Tree Gradient Boosting with Log^2 Transform	128623.4

Table 12: These are the tuned parameters of our gradient boosted model with a log transformed response.

	nrounds	max depth	eta	gamma	colsample bytree	min child weight	subsample
Tuned Parameters	150	3	0.3	0	0.8	1	1

Table 13: These are the test RMSE values for our two random forest models with various transformations on our response (price).

Models	Test RMSE
Random Forest	151919.1
Random Forest with Log Transform	161729.5

Table 14: This is our tuned mtry value for our random forest without transforming our response variable.

	mtry
Tuned Parameter	17

Table 15: These are the test RMSE's for our various models explained in our methods section.

Models	Test RMSE
Linear Model with Log^2 Transform	126523.9
Lasso with Log^2 Transform	127718.9
Scaled KNN	192710.7
Stochastic Gradient Boosting with Log Transform	128104.8
eXtreme Linear Gradient Boosting with Log Transform	131600.9
eXtreme Tree Gradient Boosting with Log^2 Transform	128623.4
Random Forest	151919.1

Table 16: This table shows the variables with the five largest coefficient values.

	Coefficients
Spatial	-193.08612
lat	-123.30287
long	88.03328
sqft_living	33.93747
sqft_above	29.46969

Figure 3: An map of King County, WA

Discussion

The results above indicates that a linear model outperforms other approaches. From Table 15, we see a linear model with the dependent variable log-transformed and squared has the greatest predictive power. In contrast, KNN approaches with or without dependent variable transformation have the lowest predictive power. This means that a linear, parametric model outperforms a non-parametric model that works well with non-linear trends.

Since each predictor in our linear model is in the range of [0, 1] because of our `preProcess`, we also looked at the largest (in magnitude) coefficients to determine which predictors most influenced our model. The five predictors with the largest β values can be seen in Table 16. The first three variables are all location variables, which goes to show that the mantra "location, location, location" holds for our dataset. For example, larger values of our `Spatial` predictor, indicate a more southeast home, since our coefficient is negative it means the more northwest our home is the larger its price, in general. If you look at our map of King County (see Figure 3), you can see that Seattle, WA is located in the northwest of King County and which we would expect to have higher home prices. Our last two variables, have to do with the size of our house, in general, the larger our home the more it costs.

Conclusion

This study generated a supervised learning model of all home sale prices in King County Washington between 2014 and 2015, in order to help home buyers/sellers develop reasonable prediction of their properties. We found that a linear model, with the dependent variable transformed, has the greatest predictive power among a series of models, and the most important predictors of home prices are related to the location and size of our homes.