Chen Xu 14941132
Woo Lee 25080136

Best Statistical Model for Predicting House Prices

Introduction

Arguably, buying a house is one of the most important moments of everyone's lives. There are many factors outside the typical ones such as number of bedrooms or the area of a backyard that determine the final sale price of a house. We have found a dataset containing 79 explanatory variables that describes almost every aspect of a house in Iowa. Our goal here is to train various models and methods learned in STAT406 using the dataset and predict the final sale price of a house. Finally, we will also determine which model or method has the best predictive power.

Data Description

The dataset has 1460 houses with 79 explanatory variables: there are 46 categorical variables and 33 quantitative variables. These explanatory variables describe various house features such as the quality of a house and the number of bedrooms. The response variable, SalePrice, is continuous. Each house has its own ID, but it was not used to predict SalePrice. The data can be found here: https://www.kaggle.com/c/house-prices-advanced-regression-techniques.

There are also numerous "NA" entries in the dataset that represent specific levels in categorical variables. As R handles "NA" entries as missing data, they were replaced with "None" or "Unknown". The following variables were removed, because they have very low variance which implies that they do not make much positive impact on predictive power: Alley, Utilities, PoolQC, MiscFeature, and MiscVal. 9 of the 1460 houses were removed because of missing values not accounted for by the data description. We also partitioned all the year variables with the exception of YrSold into decades (ie. 1900-1910, 1990-1999 etc.) in an effort to reduce factor levels to increase computation speed.

Methodology

We first log transformed our response variable in order to equally treat errors in predicting expensive houses and cheap houses when computing MSPE. For example, the following equation is true: log(1200) - log(1000) = log(120) - log(100).

Using our dataset with a continuous response variable, we explored both linear regression models and model-free methods:

- Linear regression* (Full, Ridge and LASSO)
- K-nearest neighbors (KNN) regression

- Regression trees
- Random forest
- Gradient boosting machines (GBM)

In order to assess how each algorithm would perform with new dataset, we used 5-fold cross-validation (CV) to estimate the mean squared prediction error (MSPE). In addition, CV was performed 50 times to reduce the variability

Analysis

Linear Regression:

We wanted to explore if we could use linear regression models for our data. In order to check our assumptions of linearity, we made a qq plot (Figure 1) and residual plot (Figure 2) for the log transformed response. According to the plots shown below, we came to the conclusion that even after having a log transformed response, the data is nonlinear and that the error terms are not normally distributed. This means that the results from any linear regression models we will apply are unreliable and thus they should not be used to predict house prices.

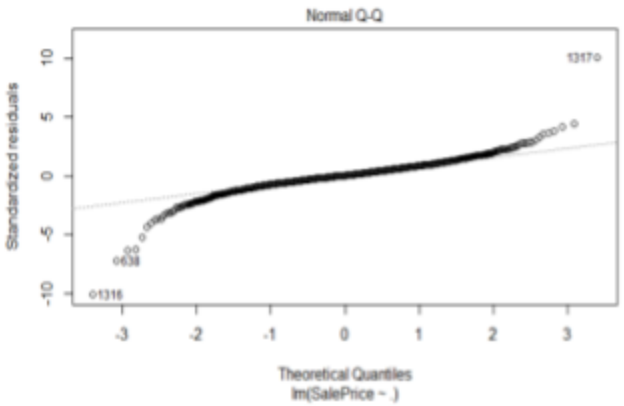

Figure 1. Quantile-quantile plot

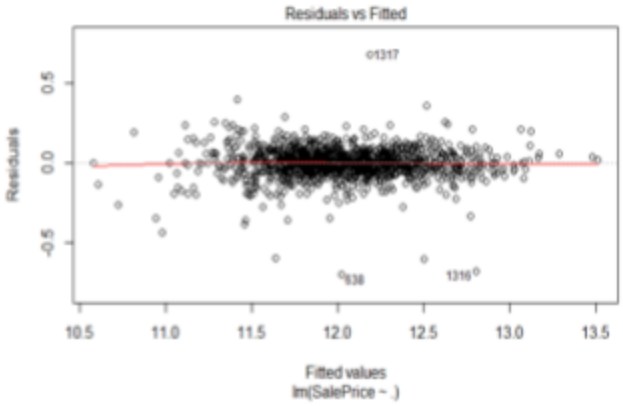

Figure 2. Residual plot

K-nearest Neighbors (KNN):

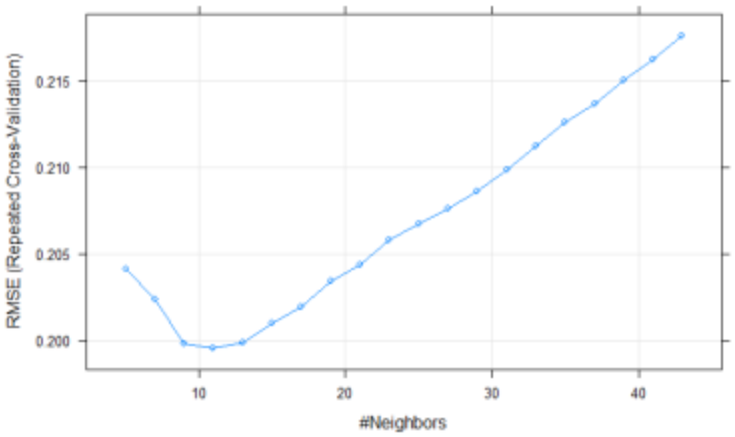

Figure 3. RMSE of KNN regression

We chose to use a KNN regression model, because it was an intuitive approach to predict the house price by finding a set of K most similar houses for a given house and returning the average of K house prices. KNN is a non-parametric method for both regression and classification. It is similar to the kernel regression

estimator in the continuous-response regression setting, which more specifically, can be thought of as a variable-bandwidth kernel estimator. Through repeatedly conducting CV, we plotted RMSE against k, the number of neighbors. Based on Figure 3. we used K=11 as the number of neighbors to find in the algorithm.

Tree-based Methods:
Due to our data having many explanatory variables, we cannot avoid the 'curse of dimensionality.' In other words, local neighbourhoods found in KNN algorithm may not really be local. Thus, we wanted to try a different non-parametric algorithm.

Trees provide a non-parametric regression estimator that is able to overcome a serious limitation of "classical non-parametric" methods like spline and kernel regression, the limitation being performing poorly when a dataset has many explanatory variables. In addition, trees are simple to understand and can be extended to ensemble models such as bagging, random forest and boosting. In order to have better performance, a regression tree is built by pruning an overfitting tree.

Figure 4. Regression tree: *Left: overfitted regression tree of our data. Right: pruned from left tree*

Although a decision tree has low bias, it has high variance. So, we decided to use the random forest method which has lower variance due to tree bagging and uncorrelated trees.

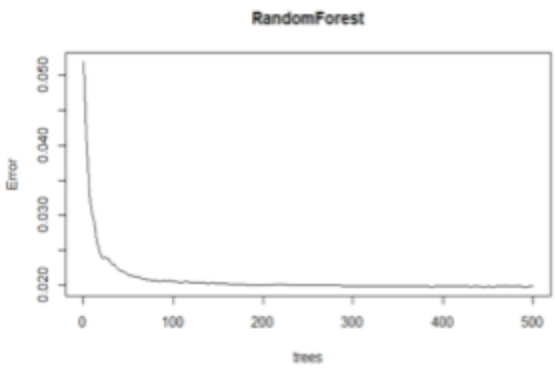

Figure 5. Out-of-bag error of random forest

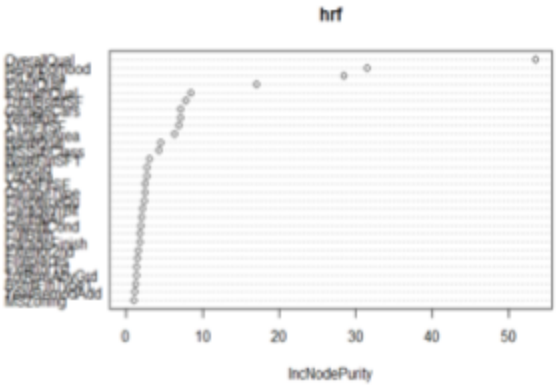

Figure 6. Variable importance plot

Figure 5 suggests using 500 trees in a random forest is enough as the out-of-bag error has been stabilized in that region

Figure 6 represents the variable ranking measured by the random forest, (suggesting that OverallQual, followed by Neighborhood, GrLivArea and ExterQual are strong predictors variables.)

Apart from random forests, another tree-based method we used was gradient boosting machines (GBMs). Like the random forest model, GBMs produce predictions by using an ensemble of weak models such as shallow regression trees. However, it does so by building the model in a stage-wise fashion. We used the 'caret' package to use this algorithm. In order to choose the optimal values of GBM parameters, we generated a grid of 50 combinations of parameter values and used the grid to greedly train GBM to find optimal values..

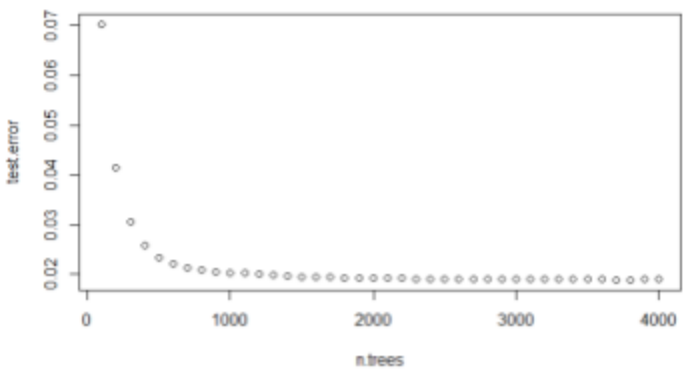

Figure 7. MSPE of gradient boosting machine

Based on Figure 7, the number of iterations of GBM was chosen to be 2000 trees as the MSPE has been stabilized in that region.

We expect GBM to have the best predictive power. Due to the curse of dimensionality, tree-based models which can resist to this curse would perform better than KNN regression. Within the tree-based models, random forest and GBMs are the strong contenders. Random forest can reduce the variance by tree bagging and random feature selection, but bias can not be further reduced. However, GBM not only reduces variance by combining multiple trees, but it can also reduce bias by adding each new tree in the sequence so that any house price that was not appropriately predicted by the preceding tree is captured. As GBM can reduce both bias and variance, we believe GBM is the best algorithm for our dataset based on our analysis.

Results and Conclusion

Each algorithm's predictive power is estimated by MSPE computed from 50 runs of 5-fold CV. Figure 9 represents the MSPEs of 4 analyzed algorithms. The order of algorithms from highest MSPE to lowest MSPE is KNN, regression trees, random forest, and GBM. We conclude that the best algorithm to predict house price is GBM.

Random forests are relatively easy to calibrate, since its default parameters of most implementations in R can achieve great results. On the other hand, GBMs are hard to tune as there are more parameters to consider: the number of trees or boosting iterations, the complexity of the tree, and the learning rate. However, if we can fine tune the parameters of GBMs, then GBMs may better predict house prices than random forests.

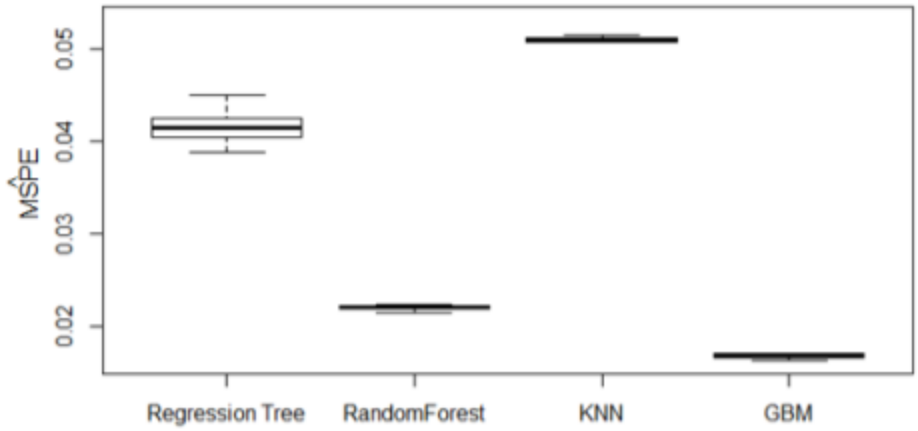

Figure 9. MSPEs of selected algorithms