House Price Prediction: Advanced Regression Techniques

Emily Strong
CSYE 7245 Spring 2018

Abstract

Housing price calculations can be a black box to those who do not work as assessors. In this project I develop a model to perform that calculation, comparing the performance of ensemble-method decision tree algorithms, specifically Random Forest and XGBoost, to the similarly non-linear MLP neural network. I hypothesize that decision trees are the best method for performing these calculations, and my results support that hypothesis with XGBoost proving to have the greatest accuracy without overfitting. These analyses use a Kaggle competition variant of the Ames Housing data set.

Introduction

The Ames Housing data set[1] was released to provide an alternative to the frequently used but decades-old Boston Housing data set. This project is an analysis of that data and exploration of regression techniques appropriate for house price prediction. Predicting house sale prices is a complex problem, so complex Zillowis offering a $1,200,000 prize[3] for improving their algorithm. The particular version of the Ames data set using here is released as part of a Kaggle competition[3] to allow data science students to gain experience with advanced regression techniques.

I propose that ensemble-method decision tree algorithms are the best approach to this data set. Data about a home is inherently mixed data, typically containing information such as various square footages, heating type, zoning information, and room counts. Location has a significant effect on sale price as comparable sales of nearby houses are used to assess the value of a house during the mortgage application process[4]. Thus any model selected to predict sale prices would need to be well-suited to handling mixed data with a large number of categorical variables.

For my analysis I examine the Random Forest and Gradient Boosted Decision Trees (XGBoost) algorithms, and also use MLP neural network to provide contrast to the effectiveness of these two algorithms.

Methodology

Data Set

The Ames Housing data set represents home sales in Ames, Iowa from 2006-2010. Homes in this area have an average value of \$180,000 ± \$80,000. As the high standard deviation suggests, the prices are highly skewed, with a long tail towards higher prices.

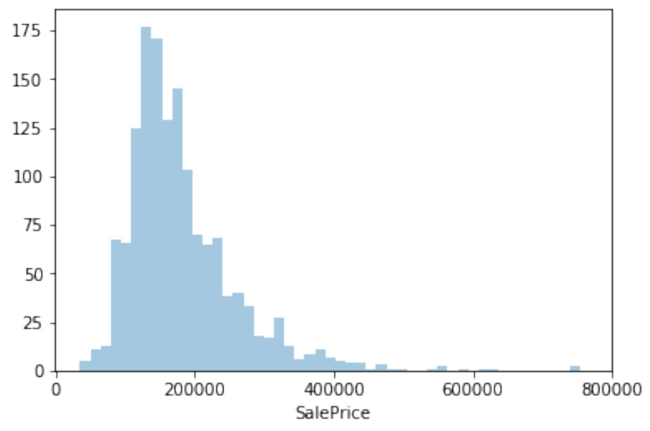

Fig 1: Distribution of sale prices

The data contains 79 features, 28 of them are continuous or discrete, 5 are years or months, and the remaining 46 are categorical and ordinal. The original data set includes the parcel ID however this feature is excluded from the Kaggle set. Additionally, the original data set has 2930 entries, however in dividing the data into train and test sets some rows appear to have been removed as the train set contains 1460 rows and the test set contains 1459.

Feature Selection

Through my exploratory data analysis, I selected the following features for use in developing a prediction model:

1. 1stFlrSF – the square footage of the first floor
2. GarageArea – the square footage of the garage
3. GarageCars – the capacity of the garage
4. GrLivArea – the square footage of the above ground living area
5. TotalBsmtSF – the square footage of the basement
6. TotRmsAbvGrd – the number of rooms above ground
7. FullBathNet – the number of full bathrooms
8. OverallQual – a rating of the quality of the home
9. YearBuilt – the year the home was built
10. YearRemodAdd – the year the home was remodeled
11. BldgType – the type of house (single family, duplex, etc)
12. BsmtQual – a rating of the quality of the basement
13. CentralAir – a flag for central air
14. FireFlag – a flag for one or more fireplaces
15. ExterCond – a rating of the condition of the exterior of the house
16. HouseStyle – how many stories the house has and whether the 2nd level is finished
17. KitchenQual – a rating of the quality of the kitchen
18. LotShape – indicates a regular or irregular shape
19. MSZoning – zoning classification
20. Neighborhood – 25 neighborhoods in Ames

Other features were rejected based on too much missing data (≥75%), <0.5 correlation with the sale price for numeric features, and low variance in sale price for the categorical features. Some were also rejected as redundant (e.g. there are multiple quality metric that have a duplicate condition metric).

Random Forest
The first algorithm I applied to the data set is the Random Forest regressor. The Random Forest algorithm is an ensemble method in which at each split in a decision tree a random subset of features are selected and evaluated, keeping the one that provides the best split point. The greater number of features that are permitted to be selected (the number of estimators), the better the final tree will fit the data. This algorithm has problems with overfitting that can be countered by limiting the depth of the tree and by using cross validation. In my analysis, I used both of these methods to limit overfitting with marginal success.

XGBoost
Gradient Boosted Decision Trees, or XGBoost, is another ensemble method for decision trees. In XGBoost a decision tree is generated and then the accuracy of the prediction is iteratively improved by adding other trees with weights assigned to them (the gradient). As with Random Forest, this algorithm can lead to overfitting. To counter the overfitting you can set a parameter that stops the model fitting when the accuracy has not improved over several iterations. You can also set the learning rate which decreases the weight given to each additional tree. I used both of these parameters following a tutorial from the Kaggle competition[5]. In my own exploration with the parameters, I found that though decreasing the learning rate does improve overfitting, there is a point at which it is too low and decreases the accuracy of the prediction.

MLP Neural Network
To provide a contrast to the two decision tree methods, I also used the Multi-Layer Perceptron neural network algorithm. In this algorithm, variables are divided into neurons and transformed in hidden layers through weighted sums and activation functions (on or off). Each layer has a specified number of neurons and eventually converge to an output. In addition to the number of layers and number of neurons in each layer, the model can be tuned through regularization set with an alpha parameter to reduce overfitting. I used Grid Search to identify the optimal alpha value.

Results
With each of the three algorithms I performed parameter tuning to improve the accuracy and reduce overfitting. Below are the best results for each algorithm. Raw results for each of the trials as I tuned parameters are in my Jupyter Notebook, as well as the mean absolute error and mean absolute percent error for each trial. I report only accuracy and RMSE here since the Kaggle competition evaluates submissions based on RMSLE (root mean square log error).

Algorithm (parameters)	**Train Score**	**Test Score**	**Train RMSE**	**Test RMSE**
Random Forest (n_estimators=200, max_depth=8)	0.961	0.857	15586.62	29816.87

Random Forest with KFold cross validation	0.962	0.859	(not calculated)	(not calculated)
XGBoost (n_estimators=1000, learning_rate=0.05, early_stopping_rounds=5)	0.951	0.913	17556.17	23146.52
MLP Neural Network (hidden_layer_sizes=(50,20,10), max_iter=500, alpha=.1)	0.012	-0.233	79123.48	87587.98

The XGBoost model had the greatest accuracy and the least overfitting. This is easily visualized by examining plots of the actual vs predicted values for each model.

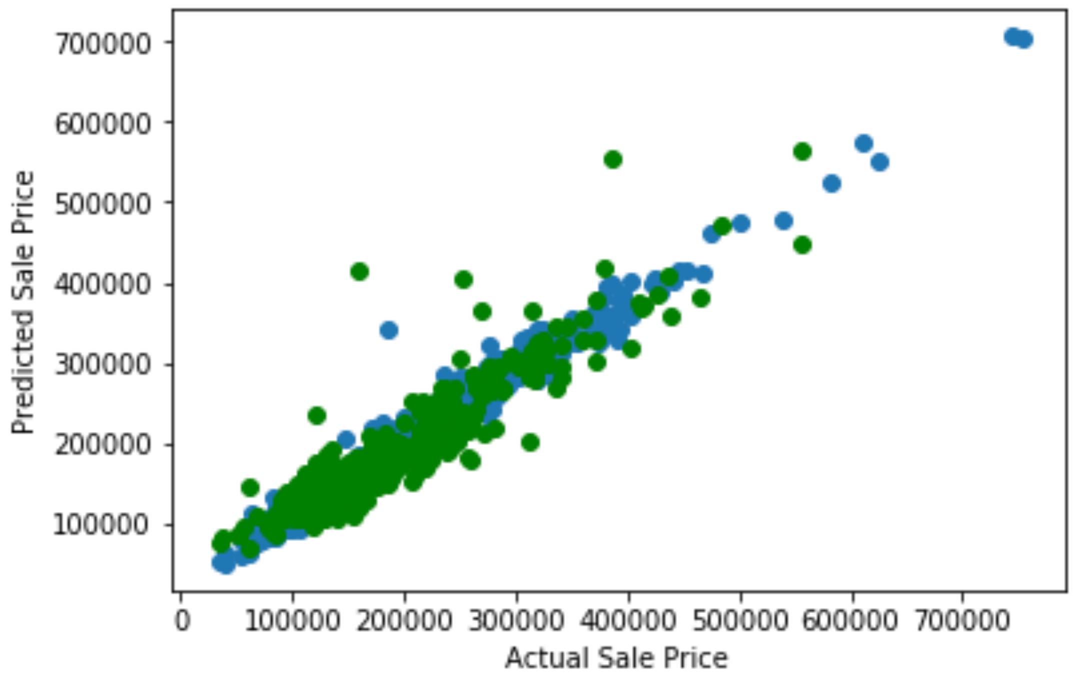

Fig 2: Random Forest: Actual vs. Predicted Sale Prices.
Key: blue is train data, green is test data

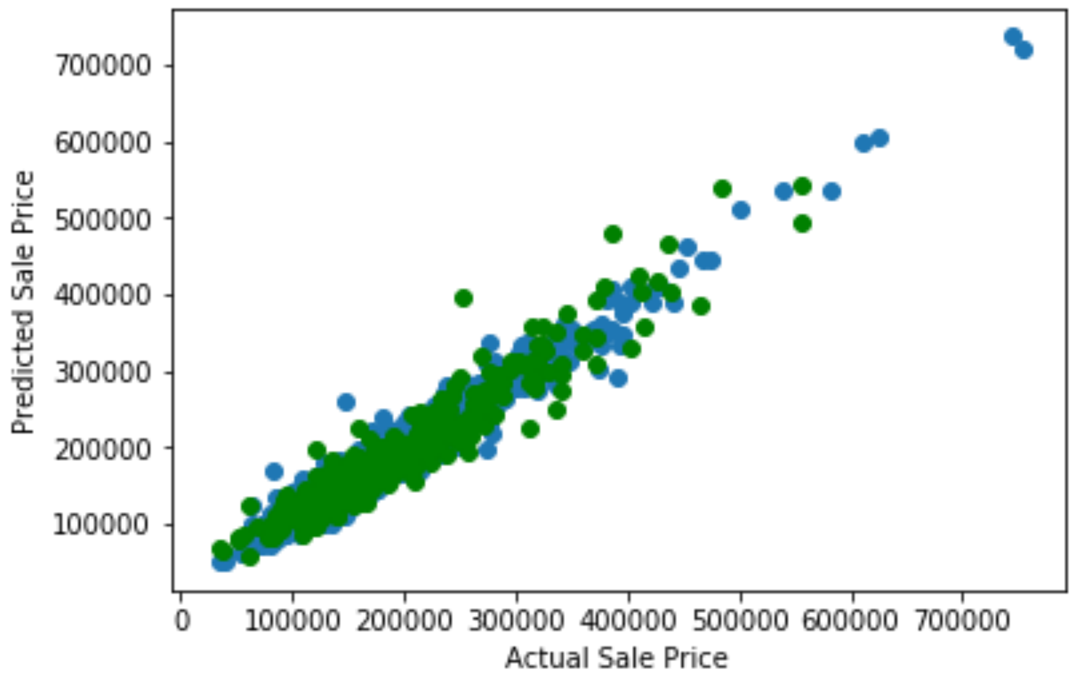

Fig 3: XGBoost: Actual vs. Predicted Sale Prices
Key: blue is train data, green is test data

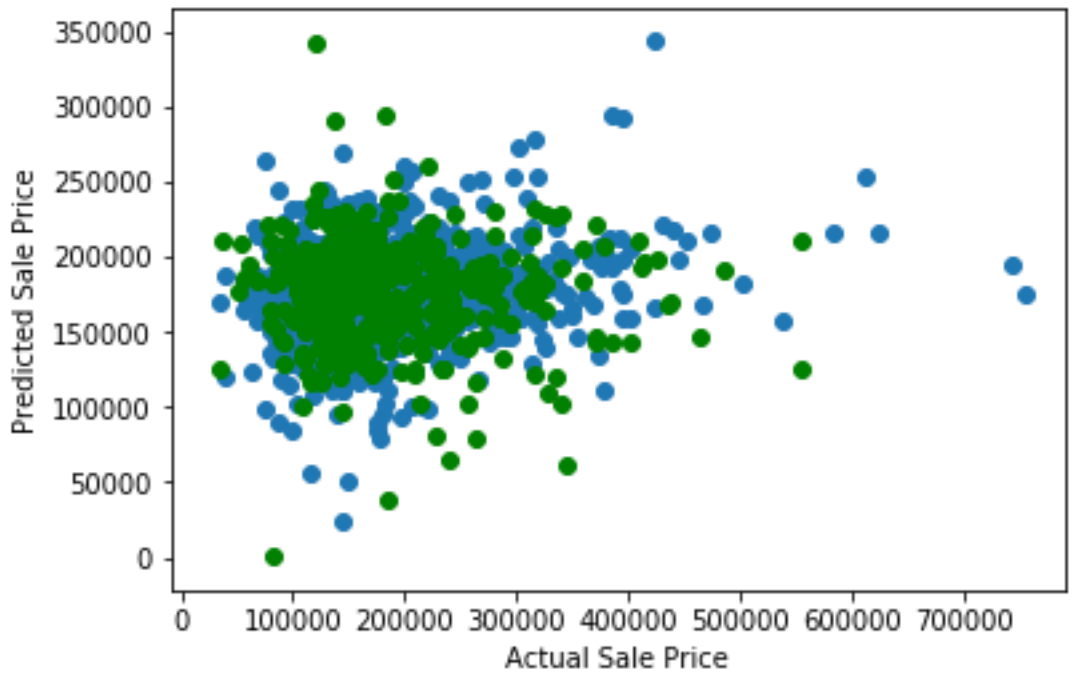

Fig 4: Neural Network: Actual vs. Predicted Sale Prices
Key: blue is train data, green is test data

The Random Forest model is generally linear but has several outliers, with a bias to predict home values that are too high. The XGBoost model forms a much neater linear plot with fewer outliers and those outliers have less extreme error. The MLP neural network model shows no linear relationship between predicted and actual values, and is grossly under predicting values to the extent the y axis maximum is 50% lower than it should be.

My Kaggle submission has a RMSLE of 0.15125, giving me a rank of #2086. RMSLE is calculated using the logs of the predicted and actual values and can be expressed as

$$\log((\text{pred} + 1) / (\text{act} + 1))$$

This metric is used so that the error of a larger number is to reduce the effect of having a large range of values, so that for example the $700k homes do not have a larger effect on the error than the $50k homes.

Discussion

My hypothesis that decision tree ensemble methods are appropriate for predicting house prices was correct. Both decision tree algorithms performed better than the neural network. Furthermore, gradient boosted decision trees proved to have the best accuracy and the least overfitting due to the extent of parameter tuning they allow. In further improving my model I would instead focus on feature engineering, as I retained a larger number of features and in choosing between redundant categorical features could have eliminated ones that have a greater effect on the variance.

References

[1] De Cock, D. Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project. Journal of Statistics Education 19, no. 3 (2011), www.amstat.org/publications/jse/v19n3/decock.pdf
[2] https://www.kaggle.com/c/zillow-prize-1
[3] https://www.kaggle.com/c/house-prices-advanced-regression-techniques
[4] https://www.bankrate.com/finance/real-estate/comps.aspx
[5] https://www.kaggle.com/dansbecker/learning-to-use-xgboost