House Prices Prediction

Anh Le, Eric Spehlmann

December 2019

I Abstract

The housing market is a complex domain that is influenced by multiple variables impacting the price of individual houses. Machine learning has been applied to the local and national housing market to create models that are used to guide appraisers, sellers, and buyers of houses. The data set used in this model represents a suburban community in Ames Iowa. The goal of the project was to experiment with data analysis methods to establish the best estimator of housing prices. The analysis used in this project ranged from varying preprocessing, algorithms, parameterization, and attribute strategies. After, intensifying attention to attribute selection based on size, age, quality, and location, the model became increasingly more accurate. The research finished with a working model that produced an accuracy of 91% using gradient boosting regression as the estimator.

II Introduction

The data used for this particular Kaggle competition contained data on houses in Ames Iowa and the selling price of the house. Each data point had 81 attributes describing almost all information of the house. The housing data included 43 categorical attributes 38 numerical attributes. This would later represent a problem when dealing with the immense volume of categorical data. The training set contained 1460 elements while the testing set contained 1459. Knowing the particulars of the data was helpful; however, further understanding of the housing domain led to better understanding of the dataset. Housing prices are greatly determined by the location of the house, the size of the house, time sold, and condition of the house. These insights were crucial in preprocessing and selecting predictors for the algorithm. The values that correlated most directly with sales price were selected using pandas .corr method to obtain the top 10 attributes. These attributes described what was predicted to affect the house price. After forming an educated understanding of the data set, data cleaning was the next step in getting the data ready for fitting an algorithm.

	Correlation
SalePrice	1.000000
OverallQual	0.790982
GrLivArea	0.708624
GarageCars	0.640409
GarageArea	0.623431
TotalBsmtSF	0.613581
1stFlrSF	0.605852
FullBath	0.560664
TotRmsAbvGrd	0.533723
YearBuilt	0.522897

Table 1: Top 10 Attributes Relating to Sale Prices

III Pre-Processing

The data proved to have missing values in both the testing and training set. First, we eliminated attributes with 90% of their data missing using the thresh parameter in the *dropna()* method with *thresh* parameter, provided by pandas. This made matters less tricky when deciding to fill those missing values. Next, the attributes were divided into numerical and categorical data. To fill in the values of the missing numerical data, the mean was calculated from the training set and was imputed into the row. This ensured no chance of data leakage to the testing set.

Categorical data was handled using advice from the website “towards data science”; which detailed filling the missing categorical data with the mode (highest frequency). If the categorical data had more than 10 missing values, the values were set to ‘None’. Once missing values were filled or removed, attribute engineering was attempted to improve the model in the future.

Attribute engineering is the process by which new attributes are used to encapsulate multiple attributes or describe a feature that could be interpreted from values. The attributes engineered for the model included, total bath number, total square footage, and age. Total baths was created by adding the number of baths, counting a full bath as one and a half bath as 0.5, respectively. This allowed for the one attribute to encapsulate 4. The total square footage was created to sum the total area of the house. Age was devised by subtracting the year built from 2019 to get years old. Lastly, the year remodeled attribute was manipulated to be “remodeled” which contained a 1 or 0 if the house was remodeled.

Encoding of categorical information varied based on the attribute, location and condition were handled similarly. After investigating the most expensive neighborhoods, by summing sales prices from neighborhoods using the *groupby()* method provided by pandas, a map was created to change the categories into ordinal values. This same concept was used for condition which represented key location information. Other categorical attributes represented ordinal values that could be handled with similar cases. Many of the qualitative attributes used varying scales of “excellent, good, avg, poor”. This presented an opportunity to map the strings to a scale of 1-4 or 1-6 depending on the particular scale.

Lastly, binary encryption and one hot encoding were incorporated into the data set to encrypt categorical data. The roof style seemed of minor importance, but figuring certain roofs were better for the cold weather in Ames, it was decided to incorporate it. Binarization using Sklearn Binarizer was used to create labels for the roofs. This only affected the model's variance if using gradient boosting. In addition to binary encoding, one hot encoding was used to encode the foundation attribute. Foundation represents an expensive construction aspect of the house. Using the pandas *get_dummies()* method, foundation was encrypted.

Algorithms:

Linear regression, linear ridge, lasso, and gradient boosting were all experimented with to determine which created the best model for predicting housing price. All the models were linear regression models except gradient boosting were linear regressors. Linear regression was used as a guiding light throughout the project. The preprocessing models used focused largely on creating all numeric values so that the linear models could be fitted. Linear ridge is a type of linear regressor that works on data sets with sparse information. Thinking the data set was considerably small, ridge was used to get started with hyper tuning. Ridge regression implements a penalty term that enforces a simpler model to be developed. The results were marginally better than linear regression.

$$\sum_{i=1}^{M}(y_i - y_i)^2 = \sum_{i=1}^{M}(y_i - \sum_{j=0}^{p}(w_j * x_{ij})^2 + \lambda \sum_{j=0}^{p} w_j^2$$

Cost function for Ridge Regression

$$\sum_{i=1}^{M}(y_i - y_i)^2 = \sum_{i=1}^{M}(y_i - \sum_{j=0}^{p}(w_j * x_{ij})^2 + \lambda \sum_{j=0}^{p}|w_j|$$

Cost function for Lasso Regression

Both ridge and lasso implement regularization, a method used to limit individual coefficients from weighing more than others, preventing overfitting. Lasso shrinks the number the attributes to a mean to avoid complexity. Lasso eventually would return a cv score of 83%.

Lastly, gradient boosting was used to venture away from linear regression. Gradient boosting regressor represents an ensemble, an algorithm that produces multiple groups of models to converge to one best fitting model. However, with this power comes the increased potential of over fitting. The learning rate was key in producing a model that did not overfit the data.

Hyperparameterization:

Trying to create a universal tuning method for hyperparameterizing the algorithms took trial and error. Eventually, using ranges of values and mapping each value to a new algorithm produced an easy way to test tuning values. Linear ridge and lasso were manipulated using the alpha value. 1000 alpha values range from 0.01 to 1 with a step of 0.01, were used to graph varying ‘accuracies’. The gradient boost regressor was tuned with the same process however the learning rate was adjusted instead. Eventually, optimal values were obtained.

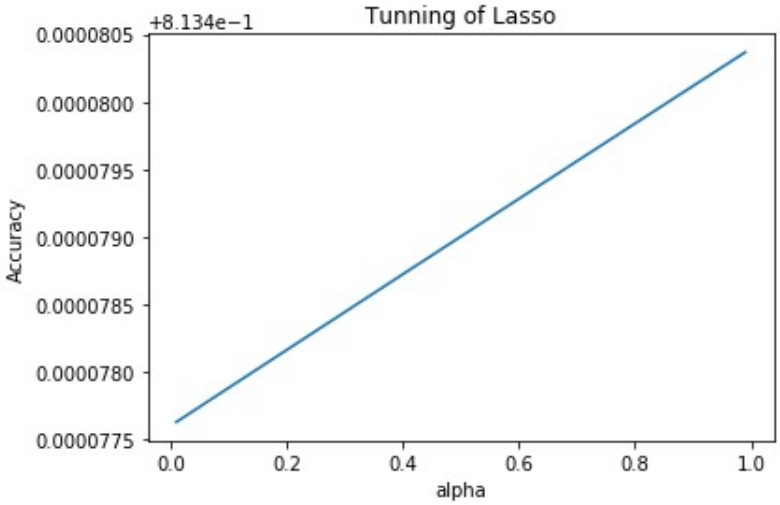

Figure 1: Relative Accuracies Corresponding to Alpha

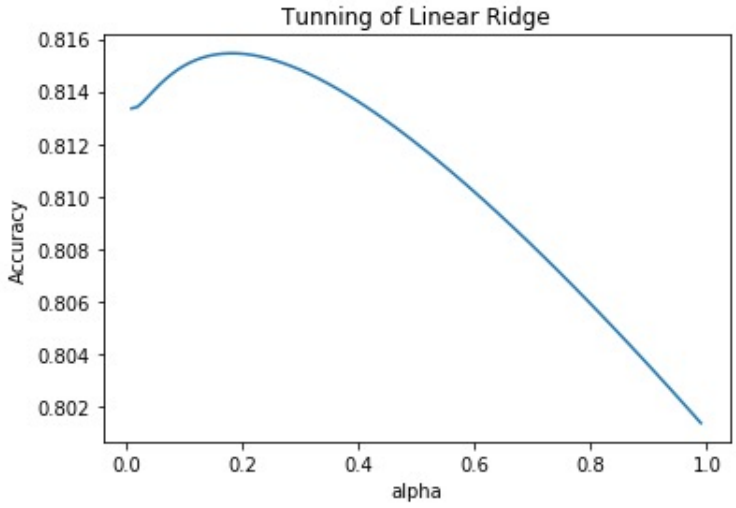

Figure 2: Relative Accuracies Corresponding to Alpha

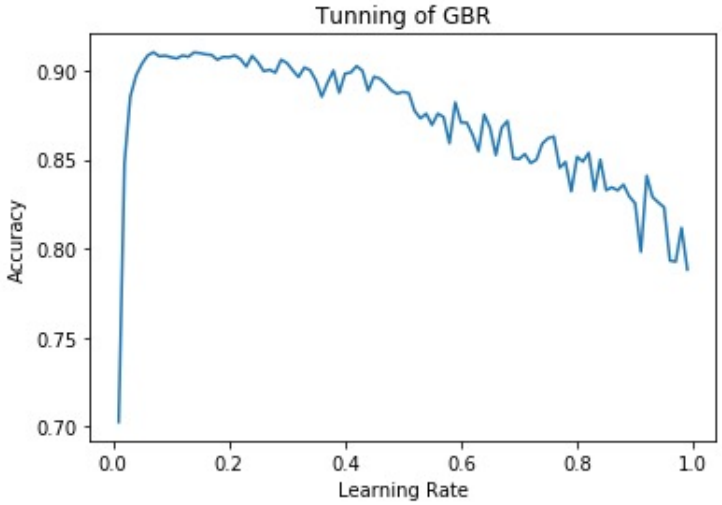

Figure 3: Relative Accuracies Corresponding to Learning Rate

Results:

After hyper tuning the models, predictions started to become more accurate. Linear ridge, and lasso had similar r2 scores but varied widely in their kaggle score. Lasso out performed ridge in the kaggle test, this could be due to the difference in score calculations between the average variance and mean squared error. Gradient Boost had the best results of the three models used in the kaggle competitions. The r2 score was 0.91 kaggle score resulted in 0.153.

Model	Tuning Parameter	Accuracy (r2)	Kaggle Score
Linear Ridge	Alpha: 0.069	0.815	0.783
Lasso	Alpha: 0.99	0.813	0.177
Gradient Boosting	Learning Rate: 0.069	0.911	0.153

Table 2: Comparation of CV scores between different models

Analysis:

Concluding from the results, gradient boosting had comparative edge on the other models. Boosting is a powerful technique that uses multiple models to improve upon the errors made by previous models. This is clearly an advantage over linear regression; Although, requiring more processing power to run. Secondly, hyper tuning was crucial in creating accurate scores. The graphs above illustrate the high drop offs in accuracy due to over or under stepping the hyper parameter.

IV Conclusion

The preprocessing seemed most important in creating accurate models. Key factors affecting the models seems to be missing values, and predictors used. Given the following predictors related to location, size, age, and quality of the house, the algorithms had enough information to create semi accurate results. Values that correlated highly with price played an important role in effective preprocessing. Linear regression worked well with numeric data; although, ultimately they didn't have enough information alone without using categorical inputs. Using an ensemble, gradient boosting

regressor, the model was able to produce the most accuracy. Concludingly, knowing the domain of the data is most important at intuiting a beautiful model.

References

[1] Swalin, Alvira, "How to Hanndle Missing Data", *Medium*. Towards Data Science, 19 Mar. 2018, https://towardsdatascience.com/how-to-handle-missing-data-8646b18db0d4

[2] Hale, Jeff, "Smarter Ways to Encode Categorial Data for Machine Learning", *Medium*. Towards Data Science, 16 July. 2019, https://towardsdatascience.com/smarter-ways-to-encode-categorical-data-for-machine-learning-part-1-of-3-6dca2f71b159

[3] Jain, Shubham, "Linear, Ridge and Lasso Regression Comprehensive Guide for Beginners.", *Analytics Vidhya*. 17 Sept. 2019, www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/.

[4] ""3.2.4.3.5. Sklearn.ensemble.GradientBoostingClassifier¶.", *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html.