Housing Price Regression

Robin Bista[1], Shriyanshi Shikha[2]
[1]Department of Computer Science, DePauw University
Greencastle, IN 46135, U.S.A.

Abstract

We are given a training and testing data set with 80 features related to the house properties. The goal is to predict the unknown sales price in the test data set. Usually, house price index represents the summarized price changes of residential housing. While for a single family house price prediction, it needs more accurate method based on location, house type, size, build year, local amenities, and some other factors which could affect house demand and supply. This is a supervised machine learning regression problem to predict the most accurate housing sales prices practical and composite data pre-processing, regression techniques enhanced by feature engineering and regularization will be examined in this paper. We will perform the feature selection by checking the correlation of the features and handing it later by modeling the data with single and ensemble of models. The precision of each model will be evaluated by Root Mean Squared Error(RMSE) of the logarithm of the sales price. The model will be most accurate with the least error.

Keywords: Gradient booster, random forest, ensemble

1 Data Description

The housing price prediction depends on the attribute we supply to our regression model. It is very important to play with the data set and analyze it as well as to pick the attributes those are the most important in evaluating the final score. In the Kaggle housing data set we were given 80 attributes to predict the price. And among them the ratio of qualitative data to quantitative data was almost 1. We had 43 categorical variables and 38 numerical variables. Numerical variables are divided into two categories namely discrete and continuous data set with 17 and 16 variables respectively.

Qualitative Variables

Alley, BldgType, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, BsmtQual, CentralAir, Condition1, Condition2, Electrical, ExterCond, ExterQual, Exterior1st, Exterior2nd, Fence, FireplaceQu, Foundation, Functional, GarageCond, GarageFinish, GarageQual, GarageType, Heating, HeatingQC, HouseStyle, KitchenQual, LandContour, LandSlope, LotConfig, LotShape, MSZoning, MasVnrType, MiscFeature, Neighborhood, PavedDrive, PoolQC, RoofMatl, RoofStyle, SaleCondition, SaleType, Street, Utilities

Quantitative Variables

1stFlrSF, 2ndFlrSF, 3SsnPorch, BedroomAbvGr, BsmtFinSF1, BsmtFinSF2, BsmtFullBath, BsmtHalfBath, BsmtUnfSF, EnclosedPorch, Fireplaces, FullBath, GarageArea, GarageCars, GarageYrBlt, GrLivArea, HalfBath, KitchenAbvGr, LotArea, LotFrontage, LowQualFinSF, MSSubClass, MasVnrArea, MiscVal, MoSold, OpenPorchSF, OverallCond, OverallQual, PoolArea, ScreenPorch, TotRmsAbvGrd, TotalBsmtSF, WoodDeckSF, YearBuilt, YearRemodAdd, YrSold

Above are the listed names of the attributes which has been categorized into qualitative and quantitative variables. As, quantitative data are numerical values we can standardize it and use it in our regression algorithm if needed, whereas the quantitative data can only be used by converting it into ordinal or interval values. Most of the variable of qualitative values are related to each other like GarageArea and GarageCars, BsmtFinSF1, BsmtFinSF2, BsmtFullBath, BsmtHalfBath, and BsmtUnfSF. After converting the repeating qualitative variables into quantitative variable these can be used to in the final algorithm. so that we are not over fitting the data.

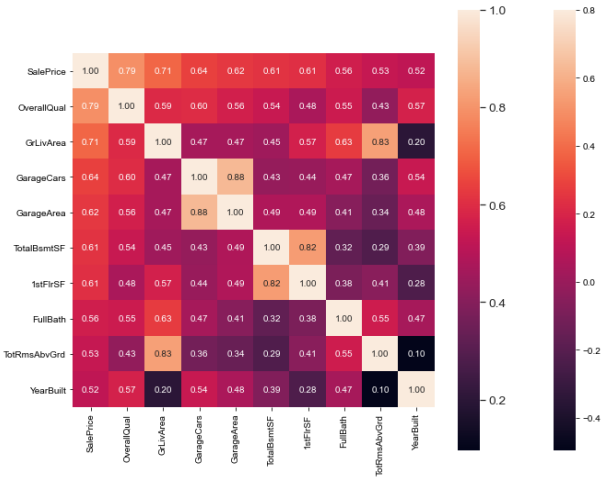

Figure 1: Co-related data

2 Experiments

2.1 Pre-Processing

Pre-processing of the data is heart of data mining. Every data must be cleaned and analyzed well so that only attributes those are highly related to the sales price must be introduced to the regressor or else we might risk over fitting the data. Our pre-processing includes checking the co-relation between 10 most correlated attributes to figure which variable is most likely to influence the values in the end. Similarly, outliers are one of the few other anomaly that deflect our prediction from the actual value. We plotted the graph of few of the attribute being used, vs sales price to check the outliers and dropped it. We also checked the missing values and handled it by Imputer by filling in the values with the mean by default. We did this because it's usually better to fill in the missing values rather than dropping them entirely because the missing values might show some valuable pattern in the data set. Nominal values were one hot encoded and converted into a series of binary variables. After the data cleaning, we standardized the data to convert the value of every feature into standardize z-score that has a mean of 0 and a unit variance. This brings all our features into the same scale. Finally we used feature engineering to create new variable that can add predictive value to the model such as YearsOld. We also used the Label Encoding feature to replace the categorical variable to numeric value. We did this because some categorical variables might contain some useful information in the data set.

2.2 Algorithm and Hyperparameterization

After the descriptive analysis of the data set we moved into the predictive analysis. We used different machine learning model to predict the data set and selected the most promising one. The different models were Gradient Boosting Tree and Random Forest Regression.
Hyperparameter Tuning: Tuning hyperparameters require splitting the training data set into training and validation data. We did rough parameter tuning of gradient boosting regressor and we also used cross validation procedure to calculate the the 10-fold-CV. Cross validation has a mechanism to stop wasting data so it was very useful. ScikitLearn provides high level estimator and it's impossible to modify the implementation of these estimators. Only tunable elements in the data set is the estimator hyperparameters. One issue we face while modeling the overfitting of data that results in the model not generalizing the test data set. This can happen when there are too many variables involved in the training set and can be handled by removing the complexity and keeping the valuable components intact. Overfitting can be reduced by standardization of data which is a feature elimination technique that reduces the less influential features coefficients to near zero while sustaining the important variables. We have used three different types of regularization in our model: Lasso, Ridge and Bayridge. Lasso has an advantage of eliminating the feature entirely which cannot be done by other regularization algorithm.

Since regularization uses hyperparameters it is difficukt to say which value will result in the best performance. Hyperparameterization is time-consuming and it is tedious to run over cross-val-score repeatedly with different combinations of hyperparameter. To over come this we introduced the GridSerchCV method, which has a built-in cross-validation that iterates through all possible hyperparameters in our case XGbooster and random forest. GridSearchCV will choose the best performing model of all the combination to execute the final fit. It is time consuming as it needs to fit k models for every hyperparameter but

at the same time it fits the entire process into one single execution and gives us an unbiased approach to identify the hyperparameter values.

Before proceeding to fitting the data set we applied the logarithmic function to the entire column of the numerical variables to make the model more log normal.

2.3 Results

Model Name	RMSE
Gradient Boosting Reg (GBR)	0.112745
XGBoost (XGB)	0.114289
Random Forest (RF)	0.159538
K-Nearest Neighbour (KNN)	0.164800
AdaBoost (ADA)	0.171082
Lasso	0.108088
Bayridge	0.110420
Ridge	0.112873

Observing the table carefully we can see that GBR and XGB have almost the same RMSE as they both use the gradient boosting implementation. Lasso is performing the best out of all the models. The tree based ensemble like XGB has many hyperparameters and hence is more computationaly intensive to tune. Lasso on the other hand performs both variable selection and regularization to enhance the accuracy and reduce the MSE. Therefore, Lasso is outperforming all other models and proved to be the best to calculate the accuracy of the from the data provided.

2.4 Analysis

Ensemble is collection of predictor that together makes an aggregate prediction.

$$h(x) = \sum_t \rho_i h_t(x)$$

We assume

$$\forall t, \rho_i = 1 \; and \; so \; h(x) = \sum_t h_t(x)$$

An additive ensemble h takes a weighted sum of results from individual predictors(h_i)

	MSE of Selected Columns	MSE of all except missing	MSE of all except missing or correlated
GBR (n=100)	791247769.4970871	509748650.4332256	507494992.2551949
GBR (n=200)	785264052.1571097	487423608.3765214	492558540.2862732
RF	870019837.4839373	636474747.3803004	648262094.7962344

Figure 2: `Table of MSE`

$$h^{(0)}(x) = h_o(x) = \sum_j y^{(j)}/n$$

$$h^{(t)}(x) = h^{(t-1)}(x) + h_t(x)$$

$$= \sum_{t=0}^{t} h_i(x)$$

The outcomes of the iteration can be seen in Figure2. After this step we made an ensemble of ensembles described above:

$$h(x) = \rho_x h_x(x) + \rho_r h_r(x)$$

With weights:

$$\begin{cases} \rho_x, \rho_r >= 0 \; and \; \rho_x + \rho_r = 1 \end{cases}$$

where x and r denotes XGB and Random Forest respectively and h(x) represents the final predicted score of the ensemble.

In our experiments, we begin with testing each algorithm namely Gradient Boosting Regression (GBR) and Random Forest Regression (RF) individually in order to see their root mean squared error is with different encoding strategies and attribute selection. According to the above table we can see that the Gradient Boosting Regressor is a better ensemble method in our model. Random Forest regressor uses decision trees which are very prone to over-fitting whereas GBR is is boosting method. In RF, for each iteration the classifier is trained independently

from the rest but in GBR one classifier at a time is added so that the next classifier is trained to improve the already trained ensemble. This specific distinction between both the ensemble makes the latter better in evaluating the error.

3 Conclusion

By preprocessing the data, using feature engineering, dropping outlier, standardizing the data we reduced the anomaly, substituted the missing data with more meaningful values in this way the data was cleaned and was ready to be fed into the regressor algorithm. Once, the data ran through each model, score from each model was recorded and compared to identify the best regression model.