Real Estate Price Prediction with Regression and Classification

CS 229 Autumn 2016 Project Final Report

Hujia Yu, Jiafu Wu
[hujiay, jiafuwu]@stanford.edu

1. Introduction

Housing prices are an important reflection of the economy, and housing price ranges are of great interest for both buyers and sellers. In this project. house prices will be predicted given explanatory variables that cover many aspects of residential houses. As continuous house prices, they will be predicted with various regression techniques including Lasso, Ridge, SVM regression, and Random Forest regression; as individual price ranges, they will be predicted with classification methods including Naive Bayes, logistic regression, SVM classification, and Random Forest classification. We will also perform PCA to improve the prediction accuracy. The goal of this project is to create a regression model and a classification model that are able to accurately estimate the price of the house given the features.

2. Data and Preprocessing

The dataset is the prices and features of residential houses sold from 2006 to 2010 in Ames, Iowa, obtained from the Ames Assessor's Office. This dataset consists of 79 house features and 1460 houses with sold prices. Although the dataset is relatively small with only 1460 examples, it contains 79 features such as areas of the houses, types of the floors, and numbers of bathrooms. Such large amounts of features enable us to explore various techniques to predict the house prices.

The dataset consists of features in various formats. It has numerical data such as prices and numbers of bathrooms/bedrooms/living rooms, as well as categorical features such as zone classifications for sale, which can be 'Agricultural', 'Residential High Density', 'Residential Low Density', 'Residential Low Density Park', etc. In order to make this data with different format usable for our algorithms, categorical data was converted into separated indicator data, which expands the number of features in this dataset. The final dataset has 288 features. We splitted our dataset into training and testing set with a roughly 70/30 split, with 1000 training examples and 460 testing examples. Besides, there were some features that had values of N/A; we replaced them with the mean of their columns so that they don't influence the distribution.

3. Models

We would perform two types of supervised learning algorithms: classification and regression. While it seems more reasonable to perform regression since house prices are continuous, classifying house prices into individual ranges of prices would also provide helpful insight for the users; also, this helps us explore different techniques which might be regression- or classification-specific. Since there are 288 features in the dataset, regularization is needed to prevent overfit. In order to determine the regularization parameter, throughout the project in both classification and regression parts, we would first perform K-fold cross validation with k = 5 on a wide range of selection of regularization parameters; this helped us to select the best regularization parameters in the training phase. In order to further improve our models, we also performed principal component analysis pipeline on all models, and cross validated number of components to fit in each of the model to give the optimized results.

3.1 Classification

Data Preprocessing

The house prices were classified into the buckets of prices. Based on the distribution of the housing prices in the data set, the price buckets were followed: [0, 100K), [100K, 150K), [150K, 200K), [200K, 250K), [250K, 300K), [300K, 350K), [350K, ∞), and we would need to perform multi-class classification to predict house prices into these seven buckets. The performance of each model can be characterized by accuracy rate, which is the number of test examples correctly classified over the number of total examples.

Models and Results

Our baseline model for classification is Naive Bayes. We implemented two types of Naive Bayes: Gaussian Naive Bayes and Multinomial Naive Bayes. Our initial expectation is that Multinomial might perform better than Gaussian, since most of the features are binary indicator values, while only a minority of them are continuous. The test result showed that Gaussian Naive Bayes had 21%

accuracy while Multinomial Naive Bayes had 51% accuracy. Optimistically speaking, even the Gaussian Naive Bayes model performed better than random guess (14% or 1/7 with 7 price buckets). Besides, to better characterize how bad the Multinomial Naive Bayes misclassified, we would assign the indexes on price buckets according to their orders, and we would compute the average absolute difference between the expected indexes and the computed indexes of all examples, which can be viewed a root mean square error. The Multinomial Naive Bayes had an average absolute difference of 0.689, which means that in average the indexes were misassigned by less than 1.

In order to improve our classification, we turned to Multinomial Logistic Regression on the same dataset. We tuned the L2 regularization parameters using the 5-fold cross validation (we would address that with more details later); we also fit an intercept into the features. Nevertheless, its performance was actually similar with Multinomial Naive Bayes; it had an accuracy of 50% compared with 51% of Multinomial Naive Bayes. After the tuning of the parameters, it appeared that the performance of both Naive Bayes and Multinomial Logistic Regression was capped at around 50%.

We continued to explore other models for our multiclass classification. One choice was Support Vector Machine Classification(SVC), and we chose linear kernel as well as Gaussian kernel. Similar to Multinomial Logistic Regression, we added an L2 regularization parameter and tuned it using cross validation. We found out that the SVC with linear kernel outperformed our past model with an accuracy of 63%, while the SVC with Gaussian kernel only had an accuracy of 41%.

At last, our final choice of classification model is random forest classification. One important parameter to control overfitting is the maximum depth that we allow the trees to grow; as a result, similar to the L2 regularization parameters of Multinomial Logistic Regression and SVC, we performed cross validation to tune this maximum depth parameters for regularization. After tuning, we obtain the accuracy of 67%, which is actually similar to SVC with linear kernel.

So far, we can observe that the SVC with linear kernel and Random Forest Classification had the best performance, with the accuracy of 67%.

3.2 Regression

Data Preprocessing

Before we fit the regression models, we preprocessed the data with log-transform on the skewed features, including the target variable SalePrice, to have normal distributions.

Models and Results

For regression models, we try to solve the following problem: given a processed list of features for a house, we would like to predict its potential sale price. Linear regression is a natural choice of baseline model for regression problems. So we first ran linear regression including all features, using our 288 features and 1000 training samples. The model is then used to predict sale prices of houses given features in our test data and is compared to the actual sale prices of houses given in test data set. The performance was measured by Root Mean Square Error (rmse) of the predicted results and the actual results. Our baseline model generated a rmse of 0.5501. Note that since the target variable SalePrice is log-transformed before model fitting, the resulting rmse is based on differences in the log-transformed sale prices, which accounts for the small values of rmse for regression models.

After using linear regression model as the baseline model, we included the regularization parameters in linear regression models to reduce overfitting. Linear regression with Lasso after 5-fold cross validation generated a rmse of 0.5418, which is better than our baseline model. Also, linear regression with lasso automatically picked 110 variables and eliminated the other 178 variables to fit in the mode. The plot on selected features and their weights in lasso regularized model is attached in part 6.

Other than lasso regularizer, we also applied ridge regularizer with cross validation in our linear regression model, which generate a rmse of 0.5448. This rmse is also better than our baseline model, meaning that regularized linear regression helped with overfitting.

Support vector regression (SVR) with Gaussian and linear kernels are also fitted to the features. Parameters Cs of both models are cross validated to pick the best performing parameters. SVR with Gaussian kernel model generated a rmse of 0.5271, and that of linear kernel generated a high rmse of 5.503. SVR with Gaussian kernel performed 4% better than our baseline model. Whereas SVR with linear kernel generated a relatively high rmse due to the kernel's unfit with the dataset in this case.

Lastly, we fitted our training dataset with random forest regression model, with max_depth parameter cross validated to be 150. Our random forest regression model

generated a rmse of 0.5397, which is also better than our baseline model.

Overall, all of the models performed better than the basic linear regression model, except SVR with linear kernel. Specifically, SVR with Gaussian kernel performed the best among all models, which generated the lowest rmse of 0.5271. We suggest that this regression model be used for future house price predictions.

4. Performance Optimization

For our classification and regression algorithms, we found two ways for optimizations: regularization and dimensionality reduction; both ways were useful to reduce overfit and increase accuracy.

4.1 Regularization

For most of our algorithms, the regularization term played an important role on model performance, and tuning the regularization term was one of the most important parts when applying these algorithms. Cross validation were performed during the Multinomial Logistic Regression, SVC and Random Forest Classification on classification algorithms, as well as Lasso/Ridge Regression, SVR and Random Forest Regression on our regression algorithms.

In order to optimize our regularization term, we would tune this regularization coefficient using cross validation. Let us take Multinomial Logistic Regression for example. We would like to tune its L2 regularization coefficient C. Here coefficient C can be considered the inverse of the regularization parameter; larger C corresponds to smaller regularization. We first selected a wide range of data points C; for each C, we would perform K-fold cross validation and compute the cross validation error score (RMSE). Then we plotted out the RMSE versus C. Based on the graph, we would hope that our range of C would roughly cover the particular C corresponding to the minimum error score, and as a result we would pick that as our designated regularization coefficient for training. Our graph is shown below. Based on the plot below, we can observe a global minimum at C = 10.

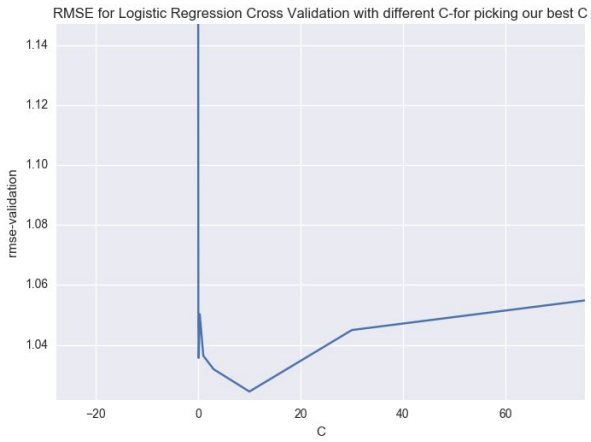

Figure 1. Cross Validation RMSE versus C

For all the algorithms except Random Forest would use cross validation to tune their regularization coefficient C, while Random Forest Classification and Regression would need cross validation to tune their maximum depth, which is also a parameter to prevent overfitting.

4.2 Dimensionality Reduction with PCA

Beside regularization, another way to prevent overfit is to reduce the dimension of our data. As we described before, our dataset contained 288 features; as a result, it might get overfitted due to the large amount of features. One algorithm for dimensionality reduction is Principal Component Analysis (PCA).

With PCA, we would be able to use the principal components instead of all the features in the dataset. Though the PCA is capable to produce the principal components, we still need to decide how many components we should include in our training and testing. As a result, we would need to tune the number of principal component and pick the number using cross-validation, using the same method as shown before in the regularization part: we first selected a wide range of data points N; for each N, we would perform K-fold cross validation and compute the cross validation error score (RMSE). Then we plotted out the RMSE versus N and we would pick the N corresponded to the global minimum or close to global minimum as our designated regularization coefficient for training.

Using PCA with tuned numbers of principal components, we would use the reduced-dimension dataset to perform our classification and regression algorithms again. For most of the algorithms we can observe improvements of accuracy, as shown in tables in part 5.

5. Model Comparison

Classification	Accuracy w/o PCA	Accuracy w/ PCA
Gaussian Naive Bayes	0.2087	0.4978
Multinomial Naive Bayes	0.5109	-
Multinomial Logistic Regression	0.5000	0.5587
SVC linear kernel	0.6740	0.6913
SVC Gaussian kernel	0.4109	0.4109
Random Forest Classification	0.6652	0.5674

Table 1. Classification Results

Regression	RMSE w/o PCA	RMSE w/ PCA
Linear Regression w/o regularization	0.5501	0.5473
Lasso	0.5418	-
Ridge	0.5448	0.5447
SVR (linear kernel)	5.503	-
SVR (Gaussian kernel)	0.5271	0.5269
Random Forest Regression	0.5397	0.5323

Table 2. Regression Results

For classification problem, adding PCA generally helped the result. Our original best algorithms are SVC with linear kernel and Random Forest Classification with accuracy of 67% without PCA. Now with the data after PCA, the accuracy SVC with linear kernel raised slightly to 69% while the accuracy of Random Forest Classification dropped to 57%. Also, even the Gaussian Naive Bayes with bad performance as baseline had a significant improvement from 21% to 50%. One interesting point to notice is that the PCA dataset cannot work on Multinomial Naive Bayes algorithm because it requires the dataset to be non-negative while the PCA transformation might introduce negative value inside the dataset and formid Multinomial Naive Bayes to be usable.

For regression problem, PCA improved the results of all the models slightly. For example, Our original best algorithms is SVR with Gaussian kernel with 0.5271 rmse without PCA. Now with the data after PCA, the RMSE of SVR with Gaussian kernel dropped slightly to 0.5269. PCA helped improve performance of random forest regression model the most, whose rmse dropped from 0.5397 to 0.5323, showing a 1.37% decrease in rmse. We did not perform pca on lasso regularized linear regression since lasso regularization already performs feature reduction during model fitting. PCA was not fitted to the SVR model with linear kernel either due to the kernel's unfit for the data set.

6. Visualizations and Analysis

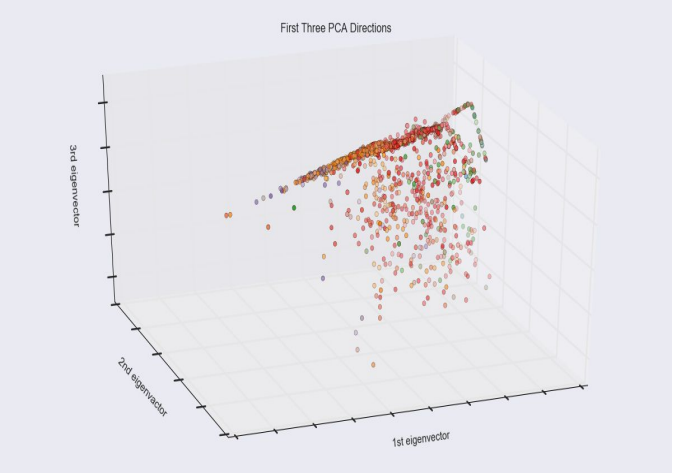

Figure 2. First three PCA directions. The 3D axes are three largest eigenvectors of the training data.

In PCA, instead of regressing the dependent variable on the explanatory variables directly, the principal components of the explanatory variables are used as regressors. By regressing on only a subset of all the principal components, PCA can result in dimension reduction through substantially lowering the effective number of parameters characterizing the underlying model. According to Figure 2, the three largest eigenvectors of the PCA model above clearly demonstrates a linear relationship between components and target variable.

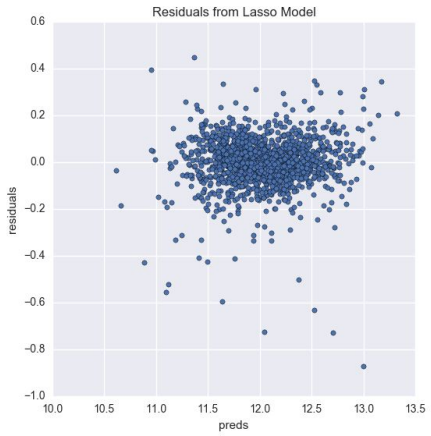

Figure 3. Residuals from Lasso Model.

Figure 3 presents a visual realization of residuals obtained from fitted lasso regression model, we can see that, for the predicted values, residuals are closely clustered around 0, which indicates that the model does not have significant misfit problem due to the randomly distributed residuals.

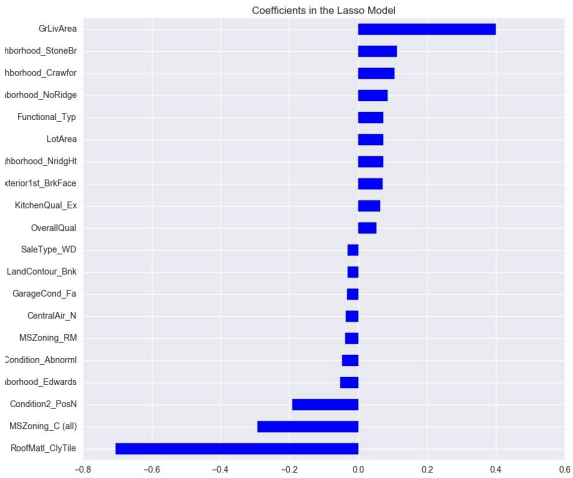

Figure 4. Coefficients of Covariates Selected in the Lasso Model

Figure 4 presents selected variables from the lasso model, and value of coefficients for each fitted covariate. The variable that has the greatest coefficient is GrLivArea (Continuous): Above grade (ground) living area square feet. This makes intuitive sense that the sale price of a real estate property is strongly correlated with its living area. The variable that has the greatest negative impact on the housing prices is Roof Matl (Nominal): Roof material : ClyTile, which indicates the material of the roof. This offers a new perspective of the housing prices as well, since the cost of the materials of the roof (which could be very expensive sometimes) can have significant impact on housing prices.

7. Conclusion

For classification problem, the best-performing model is SVC with linear kernel, with the accuracy of 0.6740; with PCA preprocessing, the accuracy can be increased to 0.6913, which is also the best among all other algorithms with PCA preprocessing.

For regression problem, the best-performing model is SVR with gaussian kernel, with rmse of 0.5271, however, visualization for SVR is difficult due to its high-dimensionality. On the other hand, lasso regression model can provide insights about chosen features, which is helpful in helping us understanding the correlations of house features and its sale prices.

According to our analysis, living area square feet, material of the roof, and neighborhood have the greatest statistical significance in predicting a house's sale price.

8. Future Work

When PCA were performed, we cross-validated its number of dimension given the tuned regularization coefficient. To obtain an lower test error, we might need to cross-validate both the number of dimension after PCA as well as regularization coefficient together, though this would be more computationally intensive.

For Random Forest Classification/Regression, besides the depth, we might need to examine further variations to optimize this algorithm, such as considering the splits of nodes, the requirements of leaf nodes, etc.

9. Reference:

[1] De Cook, Dean. "Ames, Iowa: Alternative to the Boston Housing Data as an End of Semester Regression Project." *Journal of Statistics Education*, vol. 19, no. 3, 2011.