Capstone Project

Machine Learning Engineer Nanodegree

Andy Webber

July 8, 2017

Definition

Project Overview

One problem that humans have always done that could be assisted by computers is evaluating housing prices. This is traditionally done by real estate professionals and many inputs go into this evaluation such as size of house, condition of house, neighborhood, etc. If a house is appraised too high, it could sit on the market too long without selling. If it is evaluated too low, the seller walks away with too little. A tool to give an objective measure of a house’s value could be of great aid and give reassurance to all parties that their estimates are within bounds. In addition the tool could assist homeowners about making improvements. With the knowledge of how much the improvement will increase a home’s value the homeowner could make better judgements on these investments. A third use of the tool could be for tax assessors. An objective determination of a home’s value could put much squabbling to rest.

Figure Errors for refinement methods

Problem Statement

This is clearly a regression problem and will be solved by a regression technique in scikit. The best solution could be a simple model or an advanced model involving boosting. With 78 features and only 1460 samples, it is clearly important to identify the most important features and use those or apply dimensionality reduction. The solution will minimize the mean root square error between the logarithm of the predicted price and logarithm of the actual price as dictated by Kaggle.

Metrics

This project will do regression and calculate the error between the logarithm of the predicted selling price and logarithm of the actual selling price. The logarithm is used so that errors on less expensive houses count as much as logarithms of more expensive houses. The mean root square errors of the differences in logarithms is the metric that will be used. This is the method defined by Kaggle.

Analysis

Data Exploration

This project evaluates the values of homes sold in Ames Iowa. Data came from a Kaggle competition found at <https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data>. Two files can be found there called train\_NAL.csv and test\_NAL.csv. The train\_NAL.csv file contains 78 features of houses along with the selling prices of these houses. Test\_NAL.csv contains the same 78 features for a different set of houses but no house prices. The objective is to train a model on the houses in train\_NAL.csv and predict the values of the houses in test\_NAL.csv. This being a kaggle competition, the values of the houses in test\_NAL.csv are submitted and a metric is formed. These two files have been changed from the ones downloaded because one features is called “Alley” where the value “NA” means no alley and will incorrectly interpreted as not applicable. The values were changed to NAL and the files were renamed.

With 78 features, the first task is to evaluate the available features to see which ones we expect to help us the most. The data consists of numerical data, data that can take on a range of values and categorical data, data that can take on discrete values. I started by evaluating each feature to see which one gave the most information by itself. The evaluation of the numerical features is done by doing a linear regression of the selling price to that feature and judging the features by its correlation coefficient. A high correlation coefficient tells us the feature is linearly correlated with the selling price. The categorical features are evaluated by analysis of variance that tells us how well a feature separates values. The metric of analysis of variance is the F value. A high F value tells us the feature separates the selling price value while a low one tells us it does not. With so many features tables a table below shows the 15 best numerical features and 15 best categorical features.

Table Numerical features with highest correlation coefficient

|  |  |  |
| --- | --- | --- |
| Feature name | Feature description | Correlation coefficient |
| GrLivArea | Above grade (ground) living area square feet | .71 |
| GarageCars | Size of garage in car capacity | .64 |
| GarageArea | GarageArea: Size of garage in square feet | .62 |
| 1stFloorSF | First Floor square feet | .61 |
| FullBath | Full bathrooms above grade | .56 |
| TotRmsAbvGrd | Total rooms above grade (does not include bathrooms) | .53 |
| YearBuilt | Original construction date | .52 |
| YrRemodAdd | Remodel date (same as construction date if no remodeling or additions) | .51 |
| GarageYrBlt | Year garage was built | .49 |
| MasVnrArea | Masonry veneer area in square feet | .48 |
| Fireplaces | Number of fireplaces | .47 |
| BsmtFinSF1 | Type 1 finished square feet | .39 |
| LotFrontage | Linear feet of street connected to property | .35 |
| WoodDeckSF | Wood deck area in square feet | .32 |
| OpenPorchSF | Open porch area in square feet | .32 |

The categorical features with the highest F values measured in an analysis of variance

Table Categorical features with highest F value

|  |  |  |
| --- | --- | --- |
| Feature Name | Feature description | F value |
| ExterQual | Evaluates the quality of the material on the exterior | 443 |
| KitchenQual | Kitchen quality | 408 |
| BsmtQual | Evaluates the height of the basement | 392 |
| OverallQual | Rates the overall material and finish of the house | 349 |
| GarageFinish | Interior finish of the garage | 251 |
| MasVnrType | Masonry veneer type | 111 |
| Foundation | Type of foundation | 100 |
| CentralAir | Central air conditioning | 98 |
| HeatingQC | Heating quality and condition | 88 |
| GarageType | Garage location | 72 |
| BsmtExposure | Refers to walkout or garden level walls | 71 |
| Neighborhood | Physical locations within Ames city limits | 71 |
| BsmtFinType1 | Rating of basement finished area | 68 |
| SaleCondition | Condition of sale | 46 |
| MSZoning | Identifies the general zoning classification of the sale | 43 |

This is just a preliminary examination of features. The project will loop through all features and choose the one that gives the most accurate cross validation score. It will then loop through the remaining features and give the one that gives the greatest improvement to the accuracy of the cross validation score. Two features could be highly correlated such as GarageArea and GarageCars. In this case while adding one will improve the accuracy of the cross validation score, adding the other probably will not help much because there is little additional information.

Some categorical variables are really ordinal, that is they have an order from high to low. An example of this is OverallQual. The feature has values that range from 1 to 10 in the csv files and these represent quality from very poor to very excellent. A true categorical feature such as LotConfig has values of ‘Inside’, ‘Corner’, ‘CulDSac’, ‘FR2’, and ‘Fr3’ that have no ordered value. The ordinal feature can be treated as a categorical variable or numerical variable. A first attempt will be made to treat it as categorical but each will be tested as numerical and if treating it as numerical increases the accuracy, that use will continue.

Among the numerical features, LotFrontage had 259 Nans out of 1460. For all these values the Nans were replaced by the mean values of the readable numbers. GarageYrBuilt also had 81 Nans but was not a significant feature and was not used in training. All categorical features involving garages also had 81 Nans. This is probably because 81 properties did not have garages. There were 37 or 38 Nans in all categorical variables involving basements. In all these cases the Nans were made into a separate category. Features such as FireplaceQu, PoolQc, and Fence had many Nans. This is probably because some houses did not have fireplaces, pools, or fences. Again these Nan values were made into separate categories.

Exploratory Visualization

The above grade living area in square feet (GrAbvGrade) was the numerical feature that best correlated with the sale price seen in the first table. A plot of this saleprice vs. living area is shown in the plot below. This plot shows many points roughly around a diagonal line. It is important to note the two points in the lower right with large living area and modest saleprice. These two points appear on many feature plots. Better results were obtained with the points removed.

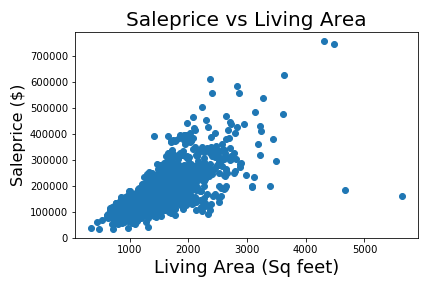


Figure Saleprice vs. Living Area

Algorithms and Techniques

To fit a model to the data I first used linear regression. This involves finding a set of weights wi such that is minimized where yi is the known Saleprices of the houses and Xij are the features of the houses that are sold. In order to keep from overfitting one more term called a regularization term is added so we really try to minimize . The regularization makes this cost function increase as the magnitude of weights get very large to accommodate outlying points. This alpha is a hyperparameter and we will test out various values to find the one that helps the cross validation data fit the data best.

A full description of Support Vector Machine is beyond this this report but an excellent description can be found in the paper “A Tutorial on Support Vector Regression” by Smola and Scholkopf found at <http://www.svms.org/regression/SmSc98.pdf>. A SVM model with a linear kernel will try to do a linear fit to the data just as the linear regression model but will try to minimize the largest error rather than the sum of the squares of the errors. It uses regression as well but it is important to note that where the equation above uses the term alpha, SVM will try to vary a parameter C = 1/. Support vector machines with rbf kernels will fit a sum of functions that are Gaussian in feature space. The SVM with Gaussian kernels come with one more hyperparameter which is the rate the functions drop off in space.

Gradient Boosting Repressor is a method of combining an ensemble of weak prediction models. A better description can be found on Wikipedia at <https://en.wikipedia.org/wiki/Gradient_boosting>. Gradient Boosting Repressor builds a series of models each based on the error of the previous model. It uses decision trees so hyperparemeters maximum depth. As more models are added each one must contribute less for the model to converge. The learning rate is the parameter that sets the rate at which sequential models contribution decreases. The number of estimators is the number boosting stages to perform.

Principal Component Analysis is about using eigenvalue analysis to determine subspaces that define points well. Imagine many points in a three dimensional space that are close to a plane. For example our solar system consist of the Sun, planets, moons and comets that are in a three dimensional space but there is a two dimensional plane that goes through the sun and comes close to much of the matter of the solar system. If we define all positions in the solar system with their coordinates on this two dimensional plane and ignore the third dimension we would be fairly accurate. The feature data we use could be similar although in much higher dimensions. We will see later that a complete description of our data could be well over a hundred dimensions but we could find it close to hyperplane of much smaller dimension. Being able to accurately describe high dimensional points in lower dimensions helps avoid “The curse of dimensionality” in machine learning. It is especially helpful when we have a limited amount of data such as our problem. Anytime we do data compression using PCA, some information will be lost while we are operating in a lower dimensional space. The question is whether the benefit of working in the lower dimensional space outweighs the information loss.

Benchmark

The evaluation metrics were described in the Metrics section. The regression analysis is predicting a scaler value and seeing how close it comes to a known measurement of that value. Because home prices can vary significantly we want a 10% error on a modest house to impact our metric as much as a 10% error on an expensive house so we measure the prediction the logarithm of the sale price.

The Kaggle page on this competition has 2106 entries ranging from 0.00 (perfect fit) to 12. The median score of these values is 0.1336. A preliminary entry of mine was 0.1348 and used only linear regression, very close to the median. This first measurement used all the data as is. The benchmark will be to see how high I can climb on the leaderboard using advanced regression techniques and data engineering.

Methodology

Data Preprocessing

Data is separated into training and testing data by Kaggle. The two outlier points described above are removed. For all numerical data, nan values are converted into the mean of the feature. Because we want to train to predict the logarithm of the sale price, this logarithm is taken.

A visual inspection of the numerical features was taken by plotting a scatter plot of each features with the sale price. I wanted to see if the sale price might be better related to the square of a numerical feature rather than the features itself. From my inspection, I determined that the feature ‘BsmtFinSF1’ was a candidate for this transformation. As it turned out, introducing this feature of BsmtFinSF1SQ did not help the regression and was ignored in the future.

Implementation

Implementation is about training a baseline evaluation involving training a first model and measuring its accuracy. Baseline evaluation involves

1. Load data from csv files into pandas DataFrames. All instances are included and all ordinal data is treated as categorical data.
2. Massaging the data to form training matrices X and y. For numerical features, nans must be modified to the mean of the feature. For categorical variables, encoding must be done using the OneHotEncoder provided by scikit. Encoding involves identifying all unique valuables and creating a feature for these unique value.
3. Establishing the single best feature. This is done by looping through all features numerical and categorical and training a model on this feature only. The feature is trained on the Ridge model with regularization parameter of 1.0. The accuracy is evaluated by 10 fold cross validation. In this evaluation the first 10% of points are put aside and the model is trained on the remaining 90%. The accuracy of the 10% gives the first measurement. We then put the second 10% aside and train on all remaining points and continue 10 times so we will then have 10 measurements of accuracy. The mean of these 10 measurements is the overall measurement of accuracy. The standard deviation is also measured to tell us how much the true mean might vary from the measured mean.
4. With the best feature chosen, we loop through the remaining features to determine the best feature to add to the originally chosen feature giving the best accuracy score. This procedure of adding features until the accuracy reaches a minimum and starts to increase.
5. With this set of best chosen features, a learning plot is generating showing how the accuracy of testing and training datasets changes with the number of points used.
6. Finally a validation plot is made to show how accuracy changes with the regularization parameter.

Refinement

I refer to the first model tested as the “standard model”. This model uses all points including the outliers shown in the previous plot. It treats all ordinal features as categorical features and trains the data using the Ridge model. The standard model produces a learning curve shown below.

The best result came with the use of 35 features. As described above, I chose the features that gives the best fit to the model when used alone then choose the best of the remaining features to add to the original feature. I then choose the best feature to add to the first two and continue adding features in this way while the cross validation error decreases. At some point the addition of features causes the cross validation error to increase and this is where I stop adding features. The first 10 features chosen are shown in the table below

Table First features chosen to create model

|  |  |  |
| --- | --- | --- |
| Feature | Type | Cross validation error |
| OverallQual | categorical | 0.2286 |
| GrLivArea | numerical | 0.1941 |
| Neighborhood | categorical | 0.1639 |
| BsmtFinSF1 | numerical | 0.1518 |
| OverallCond | categorical | 0.1446 |
| YearBuilt | numerical | 0.1373 |
| TotalBsmtSF | numerical | 0.1308 |
| GarageCars | numerical | 0.1260 |
| SaleCondition | categorical | 0.1234 |
| Fireplaces | numerical | 0.1214 |

The features chosen by my code show some similarity to the features predicted in the data exploration but show some surprises as well. The first features chosen was the categorical feature OverallQual which had the fourth highest F value in the analysis of variance. ExterQual was the feature with the highest F score did not show up at all. It was likely highly correlated with the OverallQual feature so once OverallQual was included, ExterQual added very little new information. The first numerical feature used was GrLivArea which was the feature with the highest correlation coefficient measured in the data exploration. Neighborhood is the second categorical feature used but was 12th on the list of F scores. This is probably because it is highly uncorrelated with the first two features and therefore adds new information.

This plots shows error rather than the accuracy shown in the Boston Housing project at Udacity because Kaggle grades entries on error rather than accuracy. The plot shows an increase in training error as the number of points increases and a decrease in cross validation error as the number of points increases as expected. The shaded bands around the curves show the standard deviation of these errors. The errors were calculated by 10 fold cross validation where 10% of the data is put aside and the model is calculated on the remaining 90% then repeated 10 times with a different piece of data put aside each time. This generates 10 error results and the plot shows the mean of these 10 errors and the width of the shaded band shows the standard deviation of these ten errors. The difference between the two plots is less than the cross validation error for larger number of points used. The cross validation error shows some distinct increases as we add points. This is probably due to the outlying points mentioned previously. This is the first indication that our results could possibly be improved by eliminating these entries.

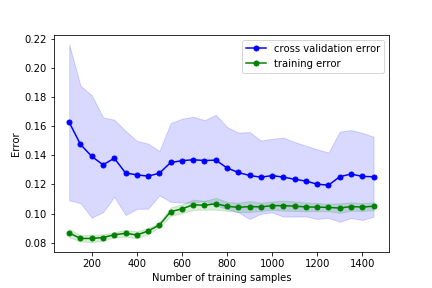


Figure Learning curve for standard model

The cross best cross validation error of this model was 0.125 and when I entered the model on Kaggle the score was 0.135. The Kaggle score is the true testing score. A difference between training and testing score is expected because the cross validation data is used to choose which features to use and other hyperparameters. This Kaggle score ranked at about 980 out of 1876.

The first attempt at improving the scores came from eliminating the outlying points as mentioned above. Doing this reduced the cross validation error to .1083 and when I entered the model on Kaggle the score was 0.1285. Both scores improved but the cross validation score improved more than the Kaggle score. This is because there were two points that were not representative of other prices. By removing these points the cross validation score improved substantially but the Kaggle score much less. The testing data could also have outlying points that cannot be removed. The new model did an overall fit of the testing model better than the original model but outliers in the testing data still exist that show up when score are submitted to Kaggle.

The next attempt to improve the model came from treating some of the categorical features as numerical features. For example the feature OverallQual (overall quality) ranges from Very Excellent which is assigned the number 10 to Very Poor which is assigned the number of 1. This is clearly an ordinal feature where Average is obviously greater than Very Poor and less than Very Excellent. Rather than using One Hot Encoder on this feature, I just treated it as a numerical feature ranging from 1 to 10 and this dropped the cross validation error to 0.1079 and a Kaggle score of 0.1251. Similar attempts were made on all other ordinal features. By treating HeatingQC as a numerical feature, the cross validation score dropped to .1077 and the Kaggle score dropped to .1251. By then changing the Functional feature to numerical the cross validation error was reduced to 0.1075 and the Kaggle score was reduced to 0.1249.

Other scored did not show improvement and were kept as categorical. I have about 20 ordinal features so to try all combinations would take just over a million tests and is not practical. I simply tried to make a feature numerical and if it improved the score I kept it there and moved on. If not I left it as categorical

Table Errors for data modifications

|  |  |  |
| --- | --- | --- |
| Modification | Cross validation error | Kaggle error |
| Standard Model (no modification) | .1250 | .1350 |
| Removing outlying points | .1083 | .1285 |
| OverallQual as numerical | .1079 | .1251 |
| HeatingQC as numerical | .1077 | .1251 |
| Functional as numerical | .1075 | .1249 |

All other improvements involved using different models. I started by trying ElasticNet and Lasso Regression. ElasticNet is linear regression with a regularization that is a combination of Ridge and Lasso. The parameter l1\_ratio determines where Elasctic net is on the range from Ridge to Lasso. L1\_ratio = 0 means it is really just the Ridge regression and l1\_ratio = 1 means it is really just the Lasso regression. The chart below shows a plot of error vs. l1\_ratio using ElasticNet where alpha = 1.0 which gave the best results.

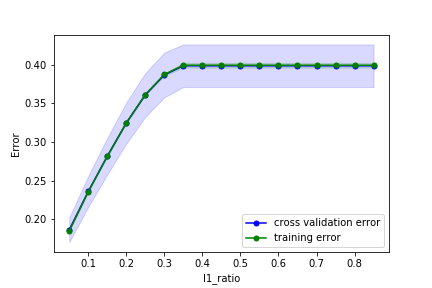


Figure Error vs. l1\_ratio for ElasticNet

The plot clearly shows that the error grows as we go from Ridge regression to Lasso regression. It is clearly best to keep with Ridge regression. The Scikit-learn algorithm cheat-sheet suggests these should be used if a few features should be important. From the plot this is obviously not the case. Because results got better as we added features up to 35 features tells us that many features are needed to get the best result so it is not surprising that Ridge regression performed better than Lasso or ElasticNet. The best result from ElasticNet came with l1\_ratio = 0.05 and this error was .1861. The error for Lasso regression was .3988.

The next model that was attempted was Support Vector Machine with a linear kernel. This model produced results that were slightly worse than the Ridge model but very close. The best error obtained was 0.1078 at alpha = 91. The regularization parameter for SVM is really C which is 1/alpha but for comparison to other results I just reported alpha. The fact the two error scores are so close is not surprising as they are just different ways of applying a linear fit to the data.

The Support Vector Machine was done with the rbf kernel as well. This was tried over a range of epsilon values and regression values. In all cases this performed noticeably worse than the Ridge model. I got the best results of 0.1569 error at epsilon = 0.005 and alpha = 1.0.

The last model attempted was Gradient Boosting Regressor. Although many paremeters were tweaked including n\_estimators, learinging\_rate, and max\_depth, no results from Gradient Boosting were better than what was attained from Ridge Regression. The best result I got with Gradient Boosting Regressor was 0.1111 at n\_estimators = 4000, learning\_rate = 0.015, and max\_depth = 3.

The final attempt to improve the fitting came from using Principal Component Analysis to reduce the number of columns in my testing and training matrices. If all features were numerical then the number of features would match the number of columns in the training matrix. With encoding on categorical features, each unique value of category will add one more column to the training matrix so the number of columns will exceed the number of features. The preliminary learning curve shown above shows the testing and training errors being within one standard deviation of the errors for the full data set. This means the model shows little variance. PCA is a tool that captures a large portion of the training information in a smaller number of columns in the training matrix i.e. it compresses the data. This compression in data is especially useful if we do not have enough data to fully fit the model.

I started with the initial 35 features that translated to a training matrix of 124 columns. When I did any compression of the data by reducing columns through PCA, the errors rose. I tried to start with more data and do compressions as well. When I used 40 features, the training matrix had 144 columns and I got the best result of 0.1074 when I brought this down to 122 with PCA. While this was slightly better than the result from the Ridge model, when I submitted this to Kaggle, the result as 0.1258 which was slightly worse. The results with 45 features and 130 columns using PCA was again 0.1074 but the Kaggle score was 0.1273, slightly worse. Attempts with PCA using 50 features and 140 columns 0.1076 and did not improve the score so PCA was not a useful tool.

All of these models were tested using the results the data changes discussed previously. Because they did not improve scores, many were not submitted to Kaggle. The table below shows the errors obtained with these models. The table below summarizes the results from method refinement. This table shows only the cross validation errors.

Table Errors for refinement methods

|  |  |
| --- | --- |
| Method (parameters) | Cross Validation error |
| ElasticNet (l1\_ratio = 0.05, alpha = 1.0) | 0.1861 |
| Lasso (alpha = 1.0) | 0.3988 |
| Support Vector Machine (linear kernel)alpha = 91) | 0.1078 |
| Support Vector Machine (rbf kernel, alpha =1.0, epsilon=.005) | 0.1569 |
| Gradient Boosting Regressor (n\_estimators = 4000, learning\_rate = 0.015, max\_depth = 3) | 0.1111 |
| Ridge with PCA 40 features compressed to 122 columns | 0.1074 |
| Ridge with PCA 45 features compressed to 130 columns | 0.1074 |
| Ridge with PCA 50 features compressed to 140 columns | 0.1076 |

Results

Model Evaluation and Validation

So the best results were obtained with the Ridge model using 35 features. Two points were eliminated and three ordinal features were converted to numerical. The only parameter to solve for was the regression parameter alpha. A validation plot examining the error as the regularization parameter, alpha varies is shown below.

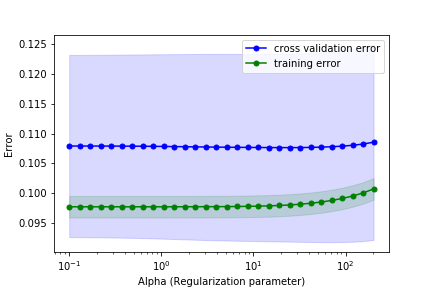


Figure Validation plot with refined data

The errors show little variation with alpha although there is a distinctive rise for alpha values above 100. Close examination of the plot shows that the cross validation error reaches a minimum near alpha = 60. This lowest cross validation error is 0.10775 and when the results are submitted to Kaggle the error is .12466. This is the best value I obtained and it ranks at about number 640 on the leaderboard out of 1890 entries or about at the 33% mark. The Kaggle evaluation is the way to test the model on unseen data.

The testing was done with 10 fold cross validation. The standard deviation of the errors on these folds was 0.016 or about 15% of the mean. These errors ranged from 0.09 to 0.14. These numbers show the robustness of the model.

Justification

By refining the methods we reduced the error from 0.135 to 0.125 as measured by Kaggle. This raised me from 980 to 634.

The final solution is significant enough to solve the problem. The purpose of the learning is to develop a tool to help Real Estate brokers price houses. The model was able to do this to about 13% accuracy. A final asking price will be set by the broker and seller but to get this close will help all parties, seller, buyer and broker know they are within range. There will sometimes be things the model cannot take into account that only a human can access. The model within 13% will also be effective in evaluating whether to do a home upgrade or not. Rather than rely on antidotal judgements, the owner can assess what upgrades will pay back their investments to 13% accuracy. This is much more effective than any friend and certainly more trustworthy than listening to a contractor that is not unbiased.

Conclusion

Free-Form Visualization

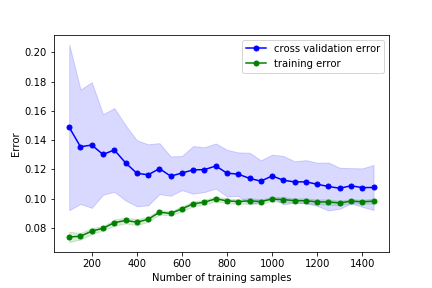
As we started by looking at the learning plot for the standard model it is important to look at the learning plot for the final solution. This is different from the learning curve shown above because two points were removed and 3 categorical features were treated as ordinal, and it was done with alpha = 60.

Figure Learning plot for final solution

This curve is smoother than the previous curve with cross validation error showing an almost uniform decrease as samples are added. For the full set of points the training error and cross validation error are within one standard deviation. The plots are close to flat near the end meaning there are enough samples and more data would not help.

Reflection

In this project, I downloaded information about homes that sold in Ames, Iowa over a 5 year period. I explored the data identifying numerical features and categorical features. I did an analysis on each features to identify whether it would do a good job in predicting the price. The numerical features were done with a correlation coefficient and the categorical features were done with an analysis of variance. Scatter plots were done of all numerical features to identify if a power or root might predict the price better than the feature as is i.e. would price vary better with LotFrontage squared than LotFrontage. A preliminary model was constructed and tested which proved to be close to the median score on Kaggle. The first attempts to improve the score came from data manipulation which included removing outlying points and treating ordinal features as numerical. Using more sophisticated learning models such Support Vector Machines and Boosting did not improve the scores.

The interesting aspects of the project is that so much improvement came from data manipulation. I originally thought that sophisticated models were the clue to better results. I turns out that careful examination of that data and asking questions of exactly how to treat the data were much more important than models.

A 13% error is within my expectation of the project. A first look at the leaderboard on Kaggle shows this to be the case. The third best error of all entries was about 10.5% and the median error was about 13.5% which to real estate professional using the tool is not much above noise. My model at 12.5% is one that could be trusted. It could give adequate guidance to buyers, sellers, and real estate professionals while a final price will be set by humans.

Improvement

There are two improvements that I would consider. The first is that I choose features based on ones that best reduced the error. I have read that another technique is Sequential Backward Selection. That is start with all features then one by one whose removal gave the lowest error.

Another improvement I have imaged it to develop a technique to set an optimal map for translating ordinal features to numerical values. For example, HeatingQC takes on 6 different values ranging from ‘Ex’ or excellent to ‘Po’ or Poor then ‘NA’ for not applicable. I choose an arbitrary linear scale for these numbers with ‘Ex’ set to 1 and ‘Po’ set to 5 and ‘NA’ set to 6. I could set this linear scale as an initial guess then measure the correlation coefficient they give when compared to selling price. I could then change the assigned values and measure a new correlation coefficient. This would involve a gradient descent model to maximize the value of r squared. I would then use this new map in my learning algorithm. I don’t know that the NA value for heating would be greater than the Poor value, it could be in the middle of my map.