Project 1

Predicting Boston Housing Prices

By Robert Lutz

December 19, 2015

Rev 1.1

1) Statistical Analysis and Data Exploration

In this investigation we will attempt to predict the price a buyer in the Boston housing market would be willing to pay for a house provided information on **13** numeric or categorical predictive attributes related to location, demographics, structure of the dwelling, etc. We use as our data set a population of **506** houses sold over an undisclosed period in the late 1970's. The house prices ranged from a minimum of **$5.0k** to a maximum of **$50.0k**. The mean and median house prices are **$22.5k** and **$21.2k** respectively. The fact that the mean is greater than the median indicates that housing prices are not distributed normally, but are somewhat skewed toward high values. The standard deviation of the house prices is **$9.2k**. Table 1 summarizes the relevant data.

Table 1: Relevant Data from the Boston Housing Dataset

Parameter	Value
Number of Houses	506
Number of Features	13
Minimum Price (k$)	5.0
Maximum Price (k$)	50.0
Mean Price (k$)	22.5
Median Price (k$)	21.2
Price Standard Deviation (k$)	9.2

2) Evaluating Model Performance

The problem we are dealing with is one of regression as opposed to one of classification. Because the price of a house is a continuous variable that can conceivably take on any positive value--rather than a descriptive variable that could be readily placed into one of numerous categories--we must obtain it using regressive techniques. We can immediately rule out using accuracy, precision, recall or F_1-score as our performance metric, as they are strictly applicable to problems of classification. We are thus left with a choice between median absolute error, mean absolute error, mean squared error, R^2-score and explained variance. Let's take a closer look at each in turn.

The median absolute error is obtained by taking the median of the absolute error for each data point, defined as

$$\left|\varepsilon_i\right| = \left|y_i - f\left(X_i\right)\right| \qquad (1)$$

where ε_i is the i^{th} error value, y_i is the actual value of element *i* and $f(X_i)$ is the value of element *i* predicted by our model function *f* mapping the feature vector X_i. We see from

equation (1) why we must first take the absolute value of the error for each data point: since the error can be positive or negative with equal probability, the expected value of the median error would be zero if we did not!

The mean absolute error produces a result very similar to the median absolute error, but is more influenced by outliers. It is defined as

$$\overline{|\varepsilon|} = \sum_{i=1}^{n} \frac{|y_i - f(X_i)|}{n} \quad (2)$$

where n is the total number of samples in the data set.

Mean squared error is similar to mean absolute error, but instead of taking the absolute value of the individual errors, we square them:

$$\overline{\varepsilon^2} = \sum_{i=1}^{n} \frac{(y_i - f(X_i))^2}{n} \quad (3)$$

Mean squared error is even more influenced by outliers than is mean absolute error, due to the process of squaring a linear measure.

R^2-score is more complicated; rather than reporting an error value directly, R^2 reports a normalized metric defined as [adapted from scikit-learn]

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - f(X_i))^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad (4)$$

where $\bar{y}$ is the mean value of parameter *y*. We see that the quotient is the mean squared error divided by the variance of *y* (the parameter *n* in both denominators have cancelled out). Basically, R^2-score represents the improvement in error by fitting the data with our chosen model compared to a model that was nothing more sophisticated than taking the mean of the data. It is equal to unity when the model fits the data perfectly and can be no higher; it's lower bound is limitless for extremely poor models.

Explained variance is a normalized metric similar to R^2-score but slightly more elaborate. It is defined as [adapted from scikit-learn]

$$\sigma_E{}^2 = 1 - \frac{\sum_{i=1}^{n} \left(y_i - f(X_i) - \overline{(y_i - f(X_i))}\right)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \quad (5)$$

It's not clear how subtracting the average error from the squared quantity in the numerator makes explained variance superior to R^2 score. It seems this procedure would give high scores to apparently poor models that have low variance but high bias. Perhaps we would use explained variance in situations in which we cared only about minimizing variance error and would be willing to accept high bias error to achieve it.

Knowing what we now know, if we wish to have a metric that represents error then we must exclude R^2-score and explained variance score because they report normalized scores rather than measures of error. Beyond that, we would need to decide if we wish the model to be more or less sensitive to outliers. A good argument would be that extreme residual outliers are evidence that a model is missing something important, thus a good model should fit the data in such a way that residual outliers are eliminated. One observation that supports this view is that the `DecisionTreeRegressor`, which we will use in this project, uses mean squared error by default (and mandatorily) to find the optimal fit. Because it is most sensitive

to outliers, and because it gives us the clearest insight into what the regressor is 'thinking', it makes sense to use the **mean squared error** as our performance metric.

In order to obtain an unbiased estimate of mean squared error, we must split the available data into two sets: one for training and one for testing. Using the training data, the machine learning algorithm will determine a best-fit set of regression parameters by means of minimizing the mean square error. The regression model will naturally be idealized to fit the training data, hence the model will fit the training data better than it will most arbitrary data sets. We would like to get an unbiased estimate of the residual mean squared error in an arbitrary data set fit by the regression model, so it behooves us to keep an independent test data set in reserve for this purpose. If one were to use as test data a set of data that contains some or all of the training data, the fit to the test data would be compromised and the mean squared error artificially small.

The best predictive model will be the one whose hyperparameters have the most universal applicability. Hyperparameters are those parameters that must be programmed into the machine learning model before the model takes a look at any data, as opposed to more customary parameters that are extracted from analysis of the data themselves. In the case of the `DecisionTreeRegressor`, the principle hyperparameter is `max_depth`, the maximum depth of the recursion tree. So how do we determine what the most appropriate hyperparameters are for general model applicability? A technique called grid search allows us to do just that. By specifying arrays of possible values for each hyperparameter (the grid) and iterating through a series of models that incorporate each hyperparameter combination, we can find the optimize the model by selecting hyperparameters that result in minimized mean squared error.

Cross validation is a very useful technique that allows us to maximize the amount of data in both the training and test data sets. Without cross-validation, increasing the size of the test data set decreases the size of the training data set and vice versa, leading to sub-optimal model assessment. Using the cross validation technique, we split the original data set into *k* smaller data sets of approximately equal size. Then we train the estimator *k* times in the following way. We follow an iterative process and loop through those *k* data sets, selecting each one in turn to be used as the test data. Once the test data has been identified, we group the remaining *k-1* data sets into a training data set on which a model is trained and subsequently tested on the test data. By doing so, we can obtain a mean squared error that contains contributions from each and every data point in the original data set.

But wait, what exactly do we have in our hands after performing a cross validation? We have *k* models for which the hyperparameters are identical but the modelling parameters are most certainly different. To use the concrete example of the decision tree regressor, each model produced by the cross validation has depth no greater than `max_depth`, but the precise branches and leaves of the decision tree will be unique to each. Which of these is the 'best' one? The 'best' model hasn't been created yet--it's the one that is trained on the entire data set. While the cross validation procedure hasn't provided us with a usable model, it has told us if a model with particular hyperparameters fits the data well or not. We can now imagine how the strength of cross validation is amplified in conjunction with grid search. By searching over a grid of hyperparameter combinations we can discover the combination that

Figure 1: Training and Test Error vs Training Size for Depths of 1, 2, 5 and 10

produces the least error in cross validation. We then select this as our model of choice, and obtain the optimal modelling parameters by training it on the full complement of data. In the case of the decision tree regressor, we will use cross validation in conjunction with grid search to reveal the optimal `max_depth` to use, then train a regression tree with this `max_depth` on the full data set.

3) Analyzing Model Performance

Figure 1 shows the training and test error versus training size for decision tree depths of 1, 2, 5 and 10. In general, we see that the training error increases as we increase training size, while the test error decreases rapidly up to a training size around fifty, then either flattens out or at best slowly shrinks as training size increases beyond 50. The increase in mean square error of the training data with sample size is analogous to the increase in sample standard deviation with sample size. The test error shows quite different behavior than the training data. In part this is due to the fact that the test data set size is constant, being independent

of the training data set size in this analysis. But more interestingly, we expect models developed on larger training data sets to be better house price predictors for arbitrary feature data. The fact that the improvement in test error performance diminishes beyond a training size of about 50 tells us that we have sufficient data in our training set (354 data points) to get good performance out of this particular model of housing prices.

We see also from Figure 1 that the training error shows more improvement with tree depth than does the test error. The training error at maximum training size improves from 45.0 at a tree depth of 1 to 0.5 at a depth of 10. Meanwhile, the test error at maximum training size improves from 52.3 at tree depth 1 to 28.6 at depth 10. At a tree depth of 1, both the training and test mean squared errors are rather high at maximum training size, which indicates that the model is oversimplified and suffers predominantly from bias error. This makes sense, as the data has been branched into only two leaves by the decision tree algorithm, which is certainly too few to develop an accurate model. At a tree depth of 10, the training mean squared error is quite small, whereas the test mean squared error is still relatively high. These facts indicate that the model at tree depth 10 overfits the data and suffers predominantly from variance error. At depth 10, the decision tree algorithm has branched the data into some large number of leaves, perhaps as many as one per data point. This is certainly far too many leaves, and noise in the training data attributed by the model as signal will manifest itself once again as noise upon modeling the test data.

Figure 2 shows the model complexity graph, in which training and test error are plotted as a function of maximum recursion tree depth. The training error decreases rapidly with increasing recursion tree depth. The test error decreases rapidly up to a depth of around 5, and then flattens out or even increases slightly. It is somewhat surprising that we see no substantial increase in error as the model becomes overly complex. We should expect to see an increase in mean squared error, due to the increase in variance error as a model becomes overly complex and overfits the data. We probably need a deeper understanding of the inner workings of the decision tree regressor to interpret and explain this observation. Nevertheless, it appears the ideal recursion depth is approximately **5**. Below a depth of 5 we run the risk of poor predictive capability of the model as evidenced by the elevated mean square error. Above a maximum recursion depth of 5 we would not expect to pay much penalty in predictive capability, but we would pay a penalty in unnecessary computing time.

4) Model Prediction

Table 2 shows an analysis of the reproducibility of the regressor model as we vary the cross validation parameter k from 3 to 6. In this analysis, 128 unique grid searches with fully randomized cross validation splits were performed for each of the *k* values listed in Table 2. From this analysis we get a sample of 128 `max_depth` values as determined by 128 grid search procedures. In all cases we find that the mode of, and therefore best choice for, `max_depth` is **5**, just as we suspected from looking at Figure 2. We report the mode rather than the mean (or median) because the best choice for `max_depth` is the one that grid search produces most often, not (necessarily) the one closest to average value of

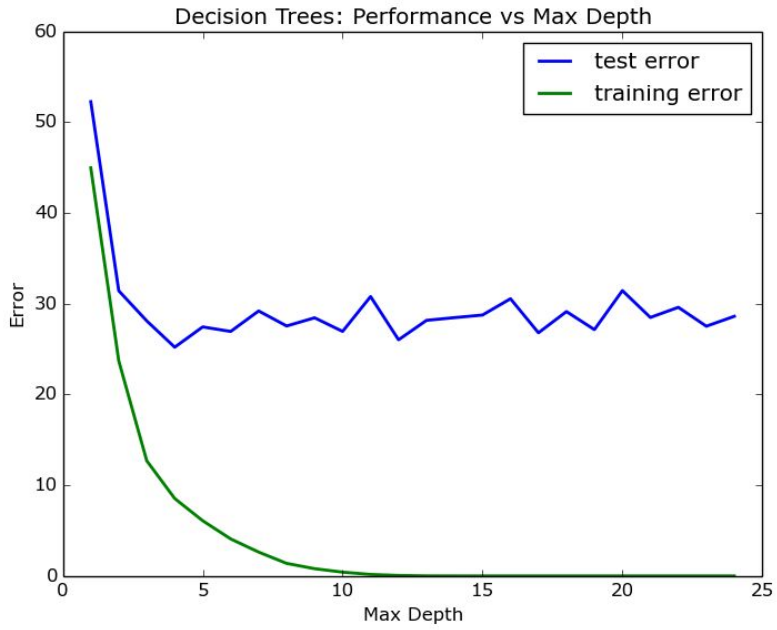

Figure 2: Training and Test Error vs Max Recursion Tree Depth

`max_depth` produced over the series of grid searches. We also see in Table 2 that grid searches performed with cross validation parameter *k* greater than 3 result in the more reproducible models with lower `max_depth` sample standard deviation than that obtained with the default *k* value of 3. There is no noticeable improvement in standard deviation upon increasing *k* beyond 4 to 5 or 6, so for this reason and to minimize computing time we choose to use a value of *k* equal of **4** in our cross validation.

With max_depth set to 5, we obtain an estimate of **$21.0k** for our house of interest. This is very close to the median price of the full data set as described in Table 1, and is very near the value we would expect had we drawn a house at random from the general population. Table 3 summarizes the key results obtained in this analysis.

Table 2: Optimum and Stability of Hyperparameter `max_depth` for Various Number of Cross Validation Split Groups *k* Obtained from 128 Grid Searches

k	`max_depth` Mode	`max_depth` Standard Deviation
3	5	1.9
4	5	1.7
5	5	1.7
6	5	1.7

Table 3: Key Values of Parameters Obtained Through Detailed Grid Search with Cross Validation

Parameter	Value
Target House Price (k$)	21.0
`max_depth`	5
k	4

Bibiography

‘Cross Validation for Detecting and Preventing Overfitting’, A.W. Moore

Scikit-Learn, http://scikit-learn.org/, especially sections 1.10, 3.2, 3.2 and 3.3 of the User Guide

Udacity, www.udacity.com, especially videos related to Machine Learning Nanodegree

‘Understanding the Bias-Variance Tradeoff’, S. Fortmann-Roe, 2012