Udacity Machine Learning Nanodegree
Project One: Boston Housing Prices
Josh Casale
1/31/2015

1) Statistical Analysis and Data Exploration

- Number of data points (houses)? 506
- Number of features? 13
- Minimum and maximum housing prices? Minimum: 5.0, Maximum: 50.0
- Mean and median Boston housing prices? Mean: 22.5328, Median: 21.2
- Standard deviation? Standard Deviation: 9.188

2) Evaluating Model Performance

- Which measure of model performance is best to use for predicting Boston housing data and analyzing the errors?
    - Because we are conducting a regression analysis, candidates for measures of model performance include mean absolute error and mean squared error. In our case, as we are predicting real estate pricing, I would argue that mean absolute error is preferable to mean squared error.
- Why do you think this measurement most appropriate? Why might the other measurements not be appropriate here?
    - For our use-case, mean absolute error is the superior metric because mean squared error gives extra weight to, and punishes, large errors. Real estate is prone to outliers for a number of reasons, and given that with 13 features, our data set likely not be capable of accounting for all eventualities that could cause an outlier, (Like a house being built by a famous architect, or previously owned by a celebrity) I do not see the utility of harshly punishing outliers.
- Why is it important to split the Boston housing data into training and testing data? What happens if you do not do this?
    - As is the case with training a model on any data set, if we do not split our data into training and testing sets, we cannot guarantee that patterns in the data used to train our model will generalize to examples outside of the training set. If we do not split our data, we have no means to evaluate the predictive power of our resulting model; we may end up with a model that perfectly describes the current data, but does not perform well on new data.
- What does grid search do and why might you want to use it?
    - Grid search performs optimization for the parameters of an estimator that are not directly trained within the estimator itself, using cross-validation on different combinations of these parameters to determine which configuration give the best results.

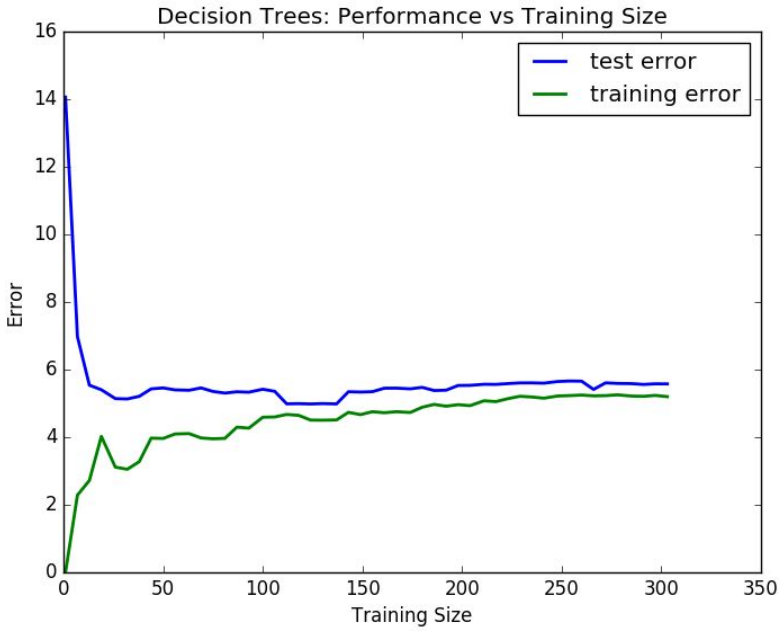
      In our case, we train a Decision Tree regressor on housing data, but give it a max_depth parameter as input, rather than learning this parameter from the training data. By utilizing a grid search, we can find which value of max_depth input parameter gives us the best performance.

- Why is cross validation useful and why might we use it with grid search?
    - The fact that a model does a good job of fitting a training set does not guarantee that the model has predictive power to generalize to examples that it has not yet seen. Using cross-validation to evaluate our model gives us an a test set to evaluate model performance that is independent of the data used to train the model. This allows us to ensure that our model generalizes to examples outside of our training set (i.e. that we are not overfitting our training data.) This is particularly useful in our case, because given that we allow a decision tree to grow deep enough, we can perfectly fit (overfit) any finite training set. However, as we have seen through this exercise, the test accuracy of decision trees stops increasing, and can even decrease, after a certain point.
    - ***REVISION***: Cross validation is a method of validating a model against held-out test data that allows us to maximize the amount of available data that we can use as training data for our model. To perform cross-validation, we split our training set into ***k*** smaller sets and train our model on each of ***k-1*** of these sets. We then use the remaining set as our test set to measure each of the trained models' performances against this test set, and average the results. Thus, we can evaluate our model's performance with need to reserve only ***n/k*** of our available ***n*** samples as a test set. Another useful feature of cross-validation is that it reduces the impact of the particular partition of our dataset. If we split the test set into only test, train, and validation sets, it's possible that our choice of partition affects the results of model training and testing. Because cross-validation uses an average of errors calculated across all of the training sets, we reduce the impact of the particular choice of how we partition our data.
    - Cross validation is a useful tool with grid search because in order to perform a grid search, we need to evaluate many iterations of our model. If we were to use a basic train/validation/test split, the particular partitioning of our dataset has a large chance of influencing the individual performance results of the different hyperparameter configurations; cross-validation prevents this from affecting our end results.

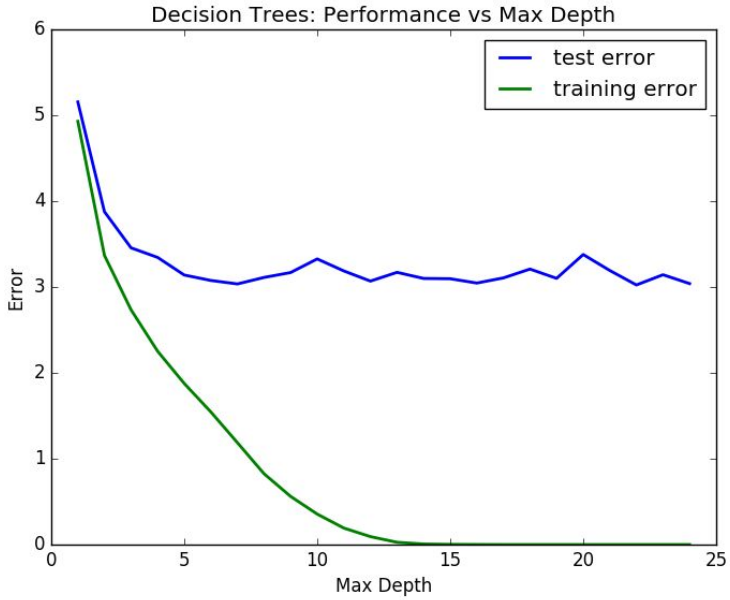
3) Analyzing Model Performance

- Look at all learning curve graphs provided. What is the general trend of training and testing error as training size increases?
    - In general, as training size increases, testing error and training error asymptotically approach some optimal values. For the training set, the optimal mean squared error value is 0 and for test data, the optimal value is not perfect, but for my model, mean absolute error approach a value of approximately 2.64.
    - ***REVISION***: As we can see from the following example figure, as **training size** increases, test error decreases and training error increases. (I misread the question upon the initial submission; I thought that the question was referring to the trends in

error as max_depth increases.)



- Look at the learning curves for the decision tree regressor with max depth 1 and 10 (first and last learning curve graphs). When the model is fully trained does it suffer from either high bias/underfitting or high variance/overfitting?
    - At a max_depth of 1, we have relatively high error (5.2 and 5.6 for train and test data respectively). This indicates that we are experiencing high bias/underfitting the data.



    - As our max_depth increases, we see both training and test error decrease until a max_depth of 7, where test error then levels off, and even begins to increase. The training error, however, continues to drop to zero with the depth of the tree. At a

max_depth of 10, training error is approximately .355 and test error is approximately 3.33; given that at a max_depth of 10, the test error is slowly increasing with max_depth, the fact that test error is almost 10 times as large as the test error here is a strong indicator that we may be overfitting the training data.

- Look at the model complexity graph. How do the training and test error relate to increasing model complexity? Based on this relationship, which model (max depth) best generalizes the dataset and why?
    - ***REVISION:*** Training and test error both initially decrease with model complexity at a roughly logarithmic scale. The training error continues to approach zero indefinitely with increasing model complexity, but test accuracy reaches a global minimum at around, and then begins to gradually increase. This is due to the bias-variance trade-off, as we begin to overfit the training data.
    - ***REVISION:*** I would argue that the model that best generalizes the dataset is that with a max_depth of 7, because at this point, testing error ceases to decrease with model complexity, and though training error is not quite at its global minimum, it is beginning to level off. As explained in the video lecture section “Occam’s Razor,” all other things being equal, the simpler of two models typically will generalize more effectively to new data. Assuming this principle, we can conclude that if the test error of a set of models is relatively static, as is the case with all models of max_depth greater than or equal to 7, the simplest model will generalize best to new data. (This is also the choice of best parameter reported by the grid search algorithm for this particular runtime.)

4) Model Prediction

- Model makes predicted housing price with detailed model parameters (max depth) reported using grid search. Note due to the small randomization of the code it is recommended to run the program several times to identify the most common/reasonable price/model complexity.
- Compare prediction to earlier statistics and make a case if you think it is a valid model.
    - Our model achieves a mean absolute error of approximately 2.6. Given that the mean of our data set is ~ 22.5, the model is, on average, within about 11.6% of the actual sale price. Given that the absolute mean error is within 10-15% of the mean sale price, I would say that the model fits the **general trend** of the data well, but is far from perfect. I’m no real-estate expert, but I would probably expect the predictions of a decent human real-estate agent to be, on average,within 10% of the actual price.