Predicting Boston Housing Prices

Greg Schafer

13 December 2015

1) Statistical Analysis and Data Exploration

1. Number of data points (houses)?
    a. 506
2. Number of features?
    a. 13
3. Minimum and maximum housing prices?
    a. Minimum = 5.0
    b. Maximum = 50.0
4. Mean and median Boston housing prices?
    a. Mean = 22.53
    b. Median = 21.2
5. Standard deviation?
    a. 9.19

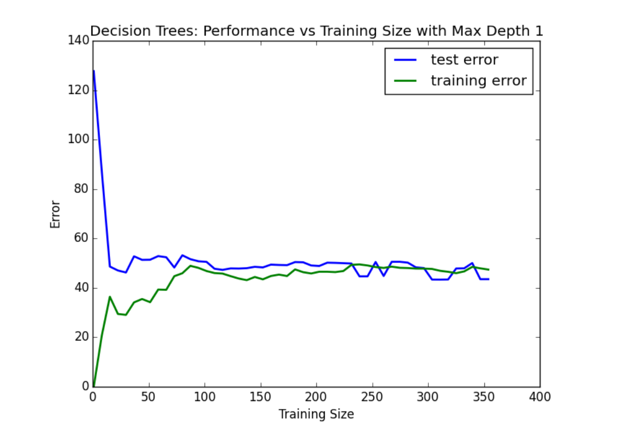
2) Evaluating Model Performance

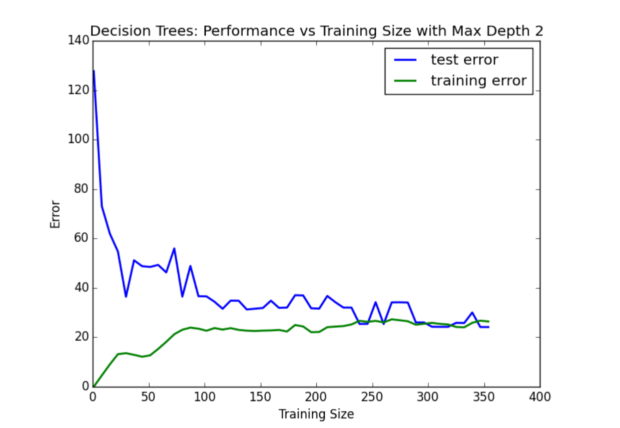
1. Which measure of model performance is best to use for predicting Boston housing data and analyzing the errors? Why do you think this measurement most appropriate? Why might the other measurements not be appropriate here?
    a. Mean squared error is likely the best metric for evaluating the model's performance, because it emphasizes large errors (due to squaring the error) compared to a metric like mean absolute error. It makes sense to emphasize large errors because an estimate of a house's price that is off by $20k is "worse" than 20x estimates that are each off by $1k (which will often be "close enough" in the mind of consumers of the model's estimates).
    b. Other model performance metrics covered in this course (like F1 score, confusion matrices, and precision/recall) are for evaluating classification models, and are therefore not applicable to this model, which is a regression.

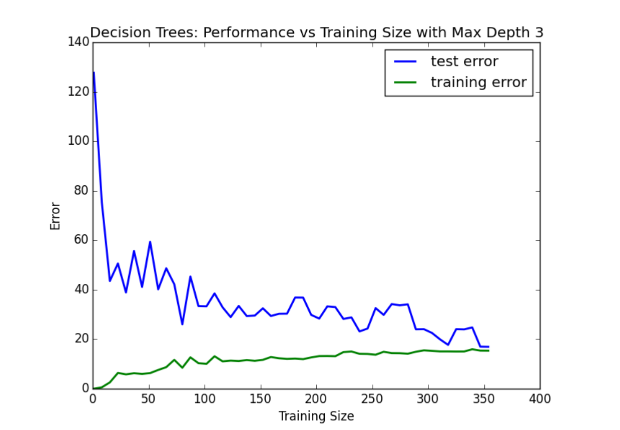
2. Why is it important to split the Boston housing data into training and testing data? What happens if you do not do this?
    a. It is important, in general, to split data into training and testing so that the model does not see (and train upon) the test data (which is used to evaluate the model's performance). If the model is trained on the test data, then it may overfit that data and not generalize well to new instances (even though you may believe that the model performs excellently because it scored well on the test data).
3. What does grid search do and why might you want to use it?
    a. Grid search exhaustively iterates all combinations of model parameters (sklearn documentation refers to these as *hyperparameters*) using a K-Fold with 3 folds (by default) cross-validation strategy to figure out the best model. For the Boston housing price regression, we use grid search to test how deep of a decision tree performs best on test data without overfitting.
    b. A max depth of 4 seems to be the most common best-scoring parameter.
    c. You might want to use grid search to find the best set of hyperparameters for your model while still avoiding overfitting. If you have a consistent train/test split of data and you manually adjust model hyperparameters to minimize error, then you might end up with a set of hyperparameters that are uniquely good at that split of train/test data (overfitting). Cross validation helps avoid this overfitting, and grid search has k-fold cross validation built in, plus it automates generating parameter combinations to make it easy.
4. Why is cross validation useful and why might we use it with grid search?
    a. Mostly answered above already, but to reiterate: cross validation helps us avoid picking hyperparameters that are overfit to a particular train/test data split. We use it with grid search so that the models trained with each combination of hyperparameters are consistent/comparable and avoid overfitting, giving more accurate scoring/error and a better final model.

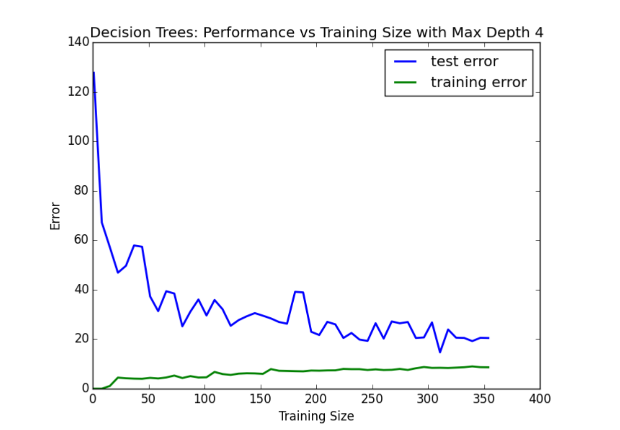
3) Analyzing Model Performance

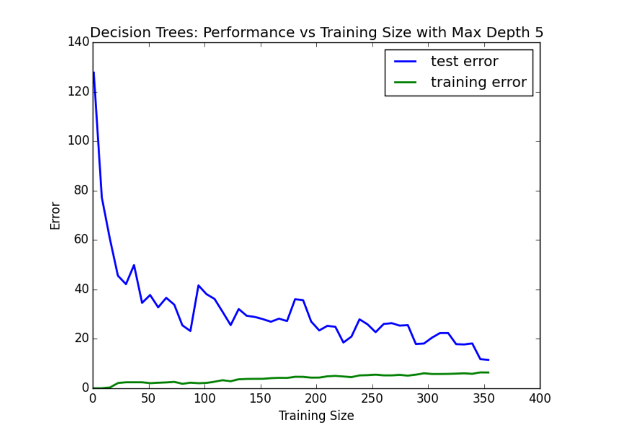
1. Look at all learning curve graphs provided. What is the general trend of training and testing error as training size increases?
    - As training size increases, training error increases (the model sees more different examples, so it is bound to estimate more of them incorrectly) and test error decreases (the model sees more examples that are similar to test instances, allowing it to estimate better on the test set).

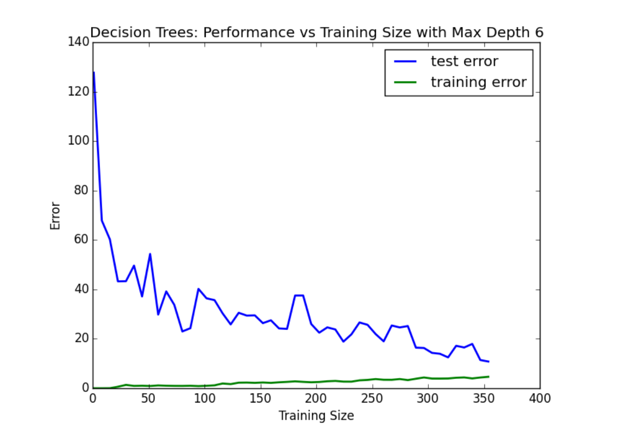

Decision Trees: Performance vs Training Size with Max Depth 1
Error
0
20
40
60
80
100
120
140
Training Size
0
50
100
150
200
250
300
350
400
test error
training error

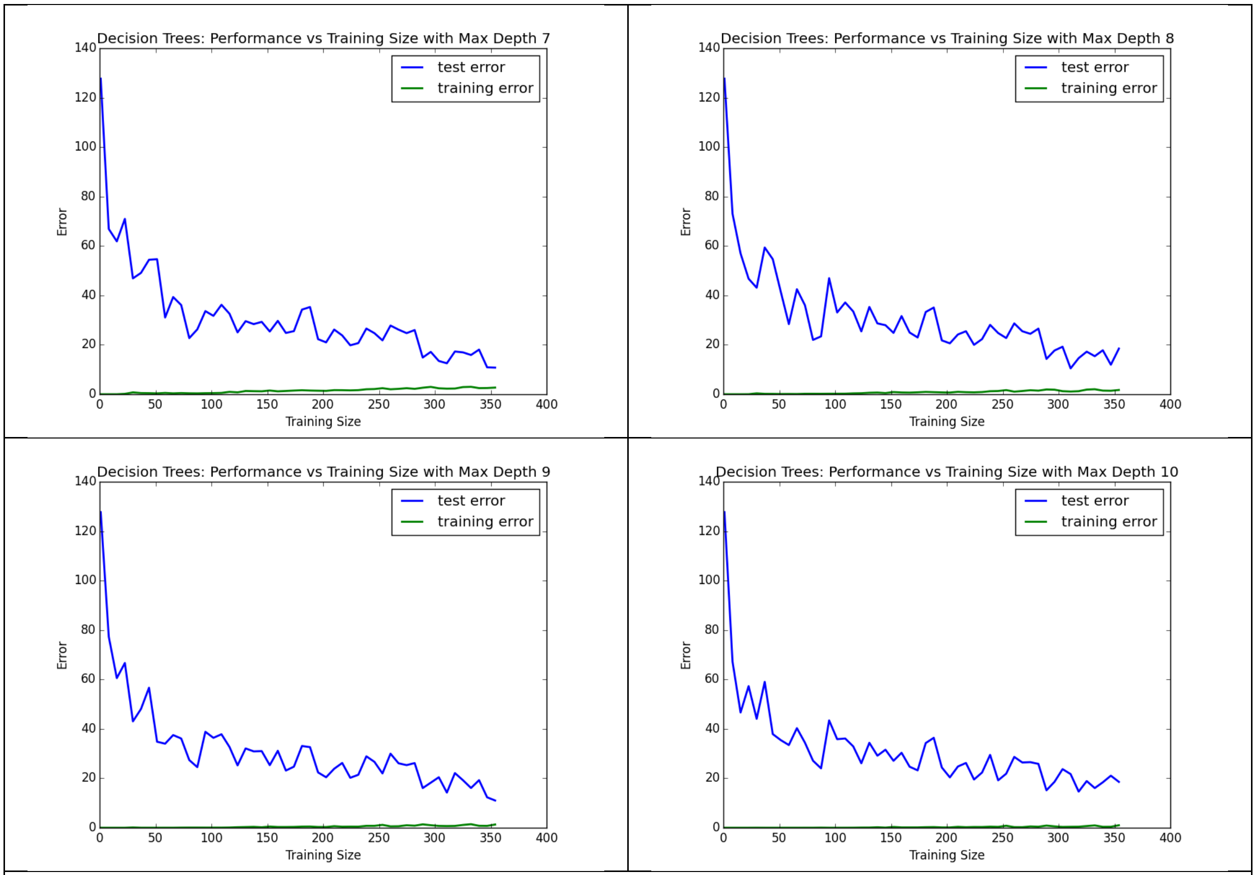

Decision Trees: Performance vs Training Size with Max Depth 2
Error
0
20
40
60
80
100
120
140
Training Size
0
50
100
150
200
250
300
350
400
test error
training error


Decision Trees: Performance vs Training Size with Max Depth 3
Error
0
20
40
60
80
100
120
140
Training Size
0
50
100
150
200
250
300
350
400
test error
training error


Decision Trees: Performance vs Training Size with Max Depth 4
Error
0
20
40
60
80
100
120
140
Training Size
0
50
100
150
200
250
300
350
400
test error
training error


Decision Trees: Performance vs Training Size with Max Depth 5
Error
0
20
40
60
80
100
120
140
Training Size
0
50
100
150
200
250
300
350
400
test error
training error


Decision Trees: Performance vs Training Size with Max Depth 6
Error
0
20
40
60
80
100
120
140
Training Size
0
50
100
150
200
250
300
350
400
test error
training error


Decision Trees: Performance vs Training Size with Max Depth 7

Decision Trees: Performance vs Training Size with Max Depth 8

Decision Trees: Performance vs Training Size with Max Depth 9

Decision Trees: Performance vs Training Size with Max Depth 10

Figure 1 – Graphs of training and test set error as training set size increases, with different DecisionTreeRegressor max_depth values.

2. 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?
   - With a max depth of 1, the model suffers from high bias (underfitting), because the model is too simple to adequately represent the data. We can tell that it is underfitting, because there is high error on the training set.
   - With a max depth of 10, the model may be suffering from a small amount of variance/overfitting. I would expect that a very deep decision tree may learn the training set so well/completely that it would not generalize to instances in the test set (which would be an example of overfitting). However, the fully-trained error on the test set at a max depth of 10 is about 20 errors, which is only slightly worse than the error rates (around 12 errors) at max depths of 5, 6, 7, and 9. Re-generating the graphs multiple times yields different exact values, but the test error at a max depth of 10 generally appears to be greater than the test error at depths of 4-6. To get a

better idea if the above graphs are consistent, I generated the fully-trained error on the test set 10 times; the results are in Figure 2, below.

Max depth	Error on test set
1	43.5
2	24.2
3	17.1
4	16.8
5	14.0
6	12.3
7	11.7
8	11.8
9	12.9
10	12.1

Figure 2 – Fully-trained error on test set, averaged over 10 runs. Due to the minimum error occurring around a max depth of 7, the model may be overfitting at greater max depths.

3. 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?
   - As model complexity increases, training error decreases, because the model can more fully capture all of the features and labels of all the training examples (though this likely leads to overfitting).
   - As model complexity increases, test error decreases to a point, then stays relatively flat or increases slightly. The test error does not continue decreasing, because there are probably examples in the test set that are different from anything in the training set, so no amount of training will prepare the model to accurately estimate those test instances. The test error may increase slightly as the model gets very complex due to overfitting the training data, causing it to generalize poorly when estimating on the test set.
   - From the model complexity graph, the max depth that best generalizes the dataset seems to be around a max depth of 5-7, which is where test error is minimized and training error is low. The training error is lower around a max depth of 15, but we care more about the test error, and adding model complexity beyond the point of minimized test error will add noise and overfitting – the model won't generalize to other instances as well as it should around a max depth of 5-7. One thing to keep in

mind is that all of the learning curve and model complexity graphs were made with a static train/test split of the data, so these conclusions may be subject to a happenstance arrangement of the data (e.g. all the instances of a particular kind are in the test set, so performance behavior may be a bit different compared to when cross validation is used).

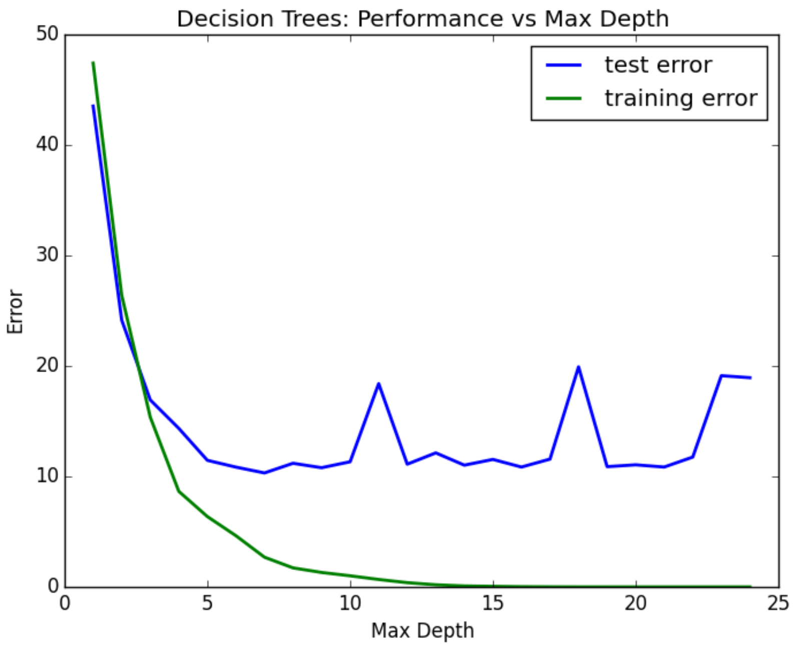


Figure 3 – Comparison of training and test set errors at different DecisionTreeRegressor max_depth values.

4) Model Prediction

1. 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.
   - After running grid search a few times, the below parameters/results are a representative example of common output:
     - Max_depth: 6
     - MSE: 34.2
     - Prediction: 20.77
2. Compare prediction to earlier statistics and make a case if you think it is a valid model.

- The predicted instance falls within the price range of the dataset (and within the dataset’s ranges for each feature), so the model won’t be extrapolating, which bodes well for the accuracy of the prediction. The predicted price is near the mean and median of the dataset, but we can’t infer anything about prediction accuracy from that. I think it is a valid model due to low error on the test set and due to the model following expected behavior in the learning curve and model complexity graphs.