Boston Housing Project Report

1) Statistical Analysis and Data Exploration

- Number of data points (houses)? 506
- Number of features? 13
- Minimum housing prices? 5.0
- Maximum housing prices? 50.0
- Mean housing prices? 22.5328063241
- Median housing prices? 21.2
- Standard deviation? 9.18801154528

2) Evaluating Model Performance

- 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?

  I chose mean squared error as my performance metric among the available score metrics for regression (e.g. mean absolute error, etc.) as it gives more weight to bigger error values (predictions further away from actual value) and hence tries to fit better to outlier/non-regular data points, which based on my assumption, might be common in housing data.

- Why is it important to split the Boston housing data into training and testing data? What happens if you do not do this?

  The test set should be separate from training set so the accuracy and generality of the model can be measured/tested. If test data is included in training, as the model gets more complex, it could overfit the training and test data and hence artificially reach a low error score on the specific test data, while having a big error on other test data.

- What does grid search do and why might you want to use it?

  Grid search could be used to search over different model parameters among a list of values and automatically find the best fit according to a scoring metric.

- Why is cross validation useful and why might we use it with grid search?

  Cross validation is generally used to track the performance of the model trained by training data over some test data. When a dataset is limited in size cross validation becomes extremely useful as it allows for an extensive exploitation of available data allowing assessing the real potential of our algorithm in terms of performance metrics. cross validation can be easily done using grid search as it can take the train and test data separately as arguments and automatically check for best model parameters using CV given the scoring algorithm.

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?

  Training error increases while testing error decreases and they both plateau after some point.

- 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?

  For max depth of 1: Looking at the right side of the graph where the model is fully trained, it can be seen that the model is underfitting the training data as evident by the high error on training data itself. This is because the model is unable to capture the relationship between the input examples and the target values.

  For max depth of 10: Looking at the right side of the graph where the model is fully trained, the model is overfitting training data as evident by the fact that the model performs well on the training data but does not perform well on the test data as it cannot generalize well to unseen examples.

- 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. However, it seems that the test error monotonically decreases up to max depth of 5, after which it oscillates. Hence, max depth of 5 would result in a model that best generalizes data without falling into the problem of overfitting.

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.

  The predicted house price lies between the min and max Y, and is close to both median and mean. Also, given the low MSE for max_depth=4, it seems to be a valid model.

House Price Prediction

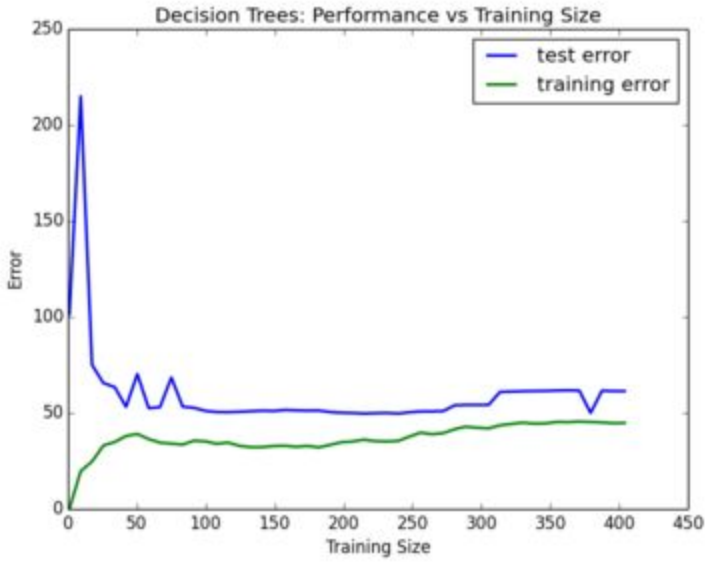
House: [11.95, 0.0, 18.1, 0, 0.659, 5.609, 90.0, 1.385, 24, 680.0, 20.2, 332.09, 12.13]
Prediction: [21.62974359]

Python output:

```
data_size: 506
feature_size: 13
min_value: 5.0
max_value: 50.0
mean_value: 22.5328063241
median_value: 21.2
std: 9.18801154528
train size: 404
Final Model:
GridSearchCV(cv=None, error_score='raise',
       estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
          max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, presort=False, random_state=None,
          splitter='best'),
       fit_params={}, iid=True, n_jobs=1,
       param_grid={'max_depth': (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)},
       pre_dispatch='2*n_jobs', refit=True,
       scoring=make_scorer(mean_squared_error, greater_is_better=False),
       verbose=0)
/Library/Python/2.7/site-packages/sklearn/utils/validation.py:386: DeprecationWarning: Passing 1d arrays as data is deprecated in 0.17 and willraise ValueError in 0.19. Reshape your data either using X.reshape(-1, 1) if your data has a single feature or X.reshape(1, -1) if it contains a single sample.
  DeprecationWarning)
House: [11.95, 0.0, 18.1, 0, 0.659, 5.609, 90.0, 1.385, 24, 680.0, 20.2, 332.09, 12.13]
Prediction: [ 21.62974359]
Best model parameter:  {'max_depth': 4}
```

Graphs

Decision Tree with Max Depth: 1



Decision Tree with Max Depth: 2

Decision Tree with Max Depth: 3

Decision Tree with Max Depth: 4

Decision Tree with Max Depth: 5

Decision Tree with Max Depth: 6

Decision Tree with Max Depth: 7

Decision Tree with Max Depth: 8

Decision Tree with Max Depth: 9

Decision Tree with Max Depth: 10

Model Complexity: