Questions and Report Structure

1) Statistical Analysis and Data Exploration

- Number of data points (houses)? Number of features?
  Data is available for 506 houses, with a feature set of size 13.

- Minimum and maximum housing prices?
  Minimum is 5.0 and Maximum is 50

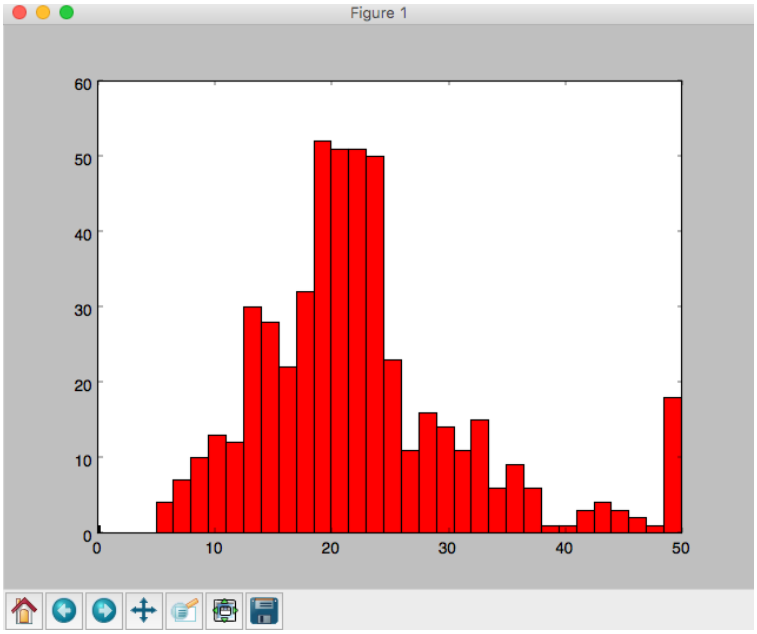
- Mean and median Boston housing prices?
  Mean price is 22.533, and Median is 21.2

- Standard deviation?
  Standard Deviation is 9.188

2) Evaluating Model Performance

- Q: 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: House Prices follow a skewed normal distribution:



There are several model evaluation metrics available for regression (http://scikit-learn.org/stable/modules/model_evaluation.html). But I created a custom metrics that is Root-Median-Square (not root-mean-square); median will avoid impact of outliers and square will highlight the bigger values, and then take square-root to balance sensitivity / fitment.

Following are model complexity graphs using different performance metrics:
(From left) Root Mean Square, Root Median Square, R^2
Of all 3, the training error seems to converge best in root median square (middle one). Although if we consider scale of error, than that is least in R^2. Since the scales and graphs vary, am not sure if median is

best choice for performance metrics here.

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

  A: Have now split data into train=70 / test=30.
  It is always important to keep testing data separate from training data, to avoid leaking of testing data into training. Test data will allow validation of trained model.

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

  A: Grid Search makes it easy to optimize hyper-parameters of the estimator, like those used by decision tree, e.g. min_samples_split and criterion. GridSearchCV also implicitly does Cross-validation (default 3 fold), so that no explicit shuffle-and-split of data is needed.

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

  A: Cross-validation shuffles and splits the data, into training, validation, and testing subsets. This could either be train_test_split or more refined K-Fold. This is needed to avoid bias in training. Otherwise it could happen that training data only includes a specific type / range of data, and the resulting model is never able to predict on test data. This also allows separating test data from training data,which is important for evaluating the model's performance.

3) Analyzing Model Performance

- Q: Look at all learning curve graphs provided. What is the general trend of training and testing error as training size increases?

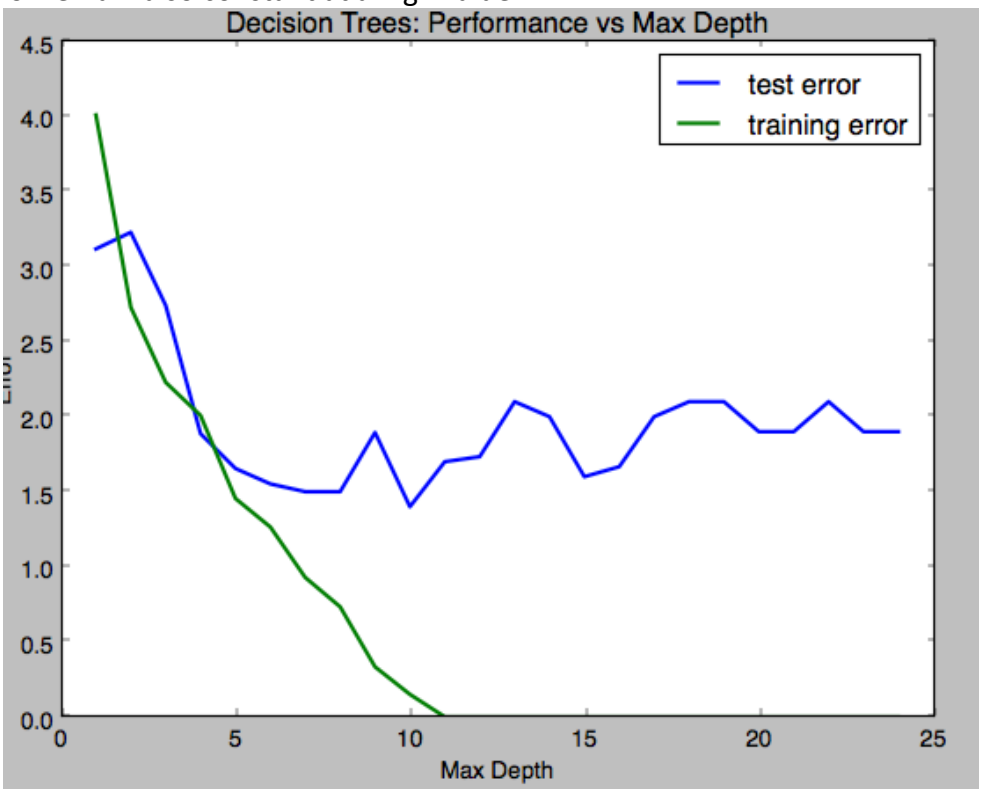
  A: With increase in training size, error in training increases while testing errors are reduced. This trend mostly continues, until both converge on a minimum constant gap. This means that model is learning more with data, getting generalized, and can make more accurate predictions. But if under some condition, gap between training and testing graphs does not reduce and remains significant, it means there is either bias or data for training model is insufficient.

- Q: 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?

A: When max-depth is 1, there is high variance, as there are significant errors in both training and testing phase while both are still converging and often overstepping each other.
When depth is 10, there is high bias since testing error remain high while training error remain low, and never the two converge.

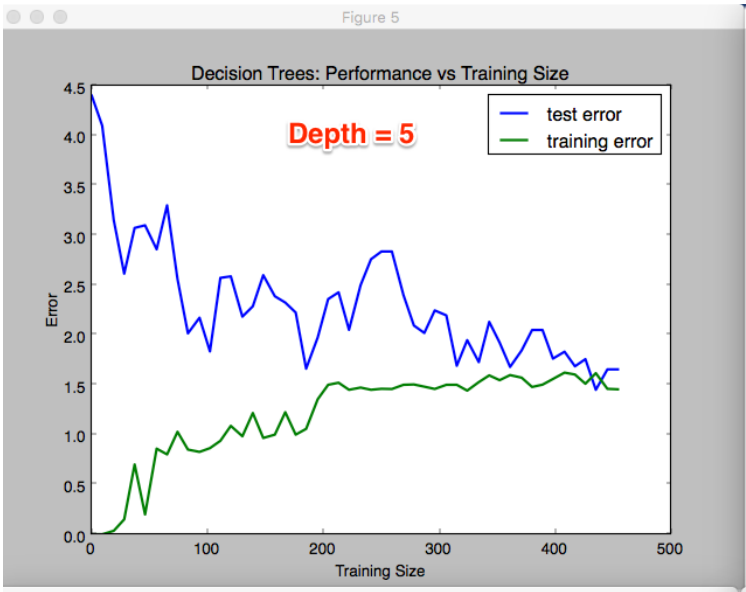
- Q: Look at the model complexity graph. How do the training and test error relate to increasing model complexity? Based on this relationship, which model (max depth) best generalizes the dataset and why?

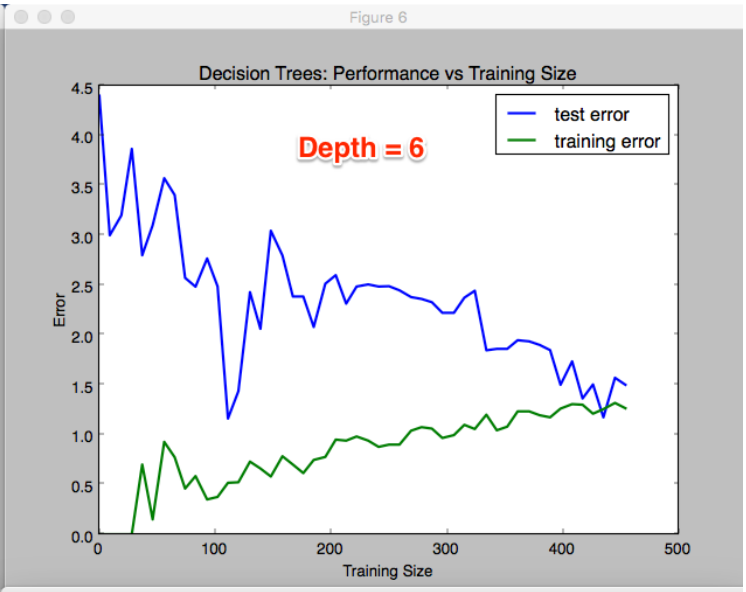
A: Training and tests errors reduce considerably with increasing complexity upto a certain depth, after which Test Error remain also constant at a high value.

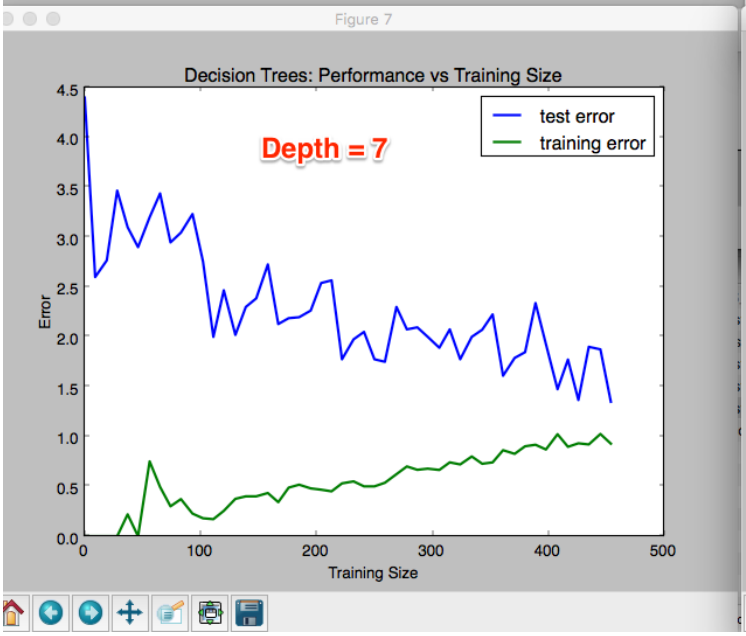


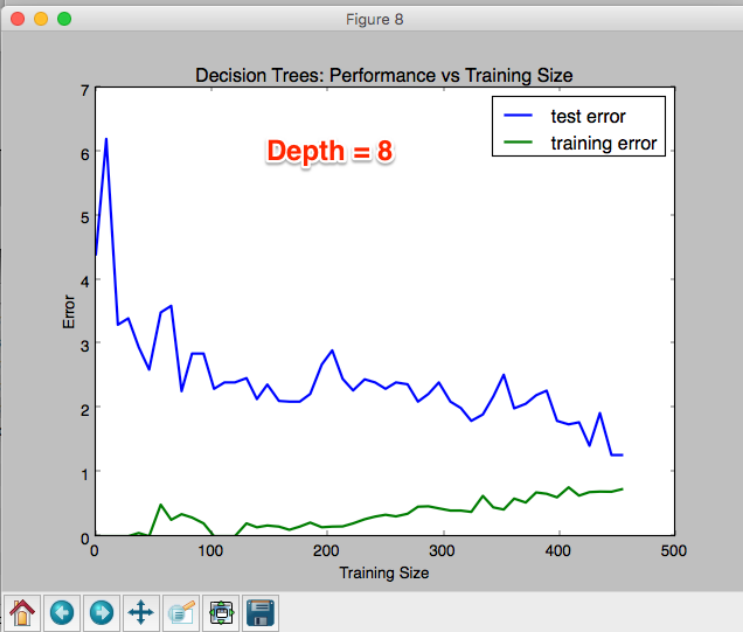
The model depths between 5 and 7 are best, since it is when both training and test errors have similar values and test error has reached its lower limit.

Depth of 5, 6, and 7 are also good since, the learning curves are also best for these depths.









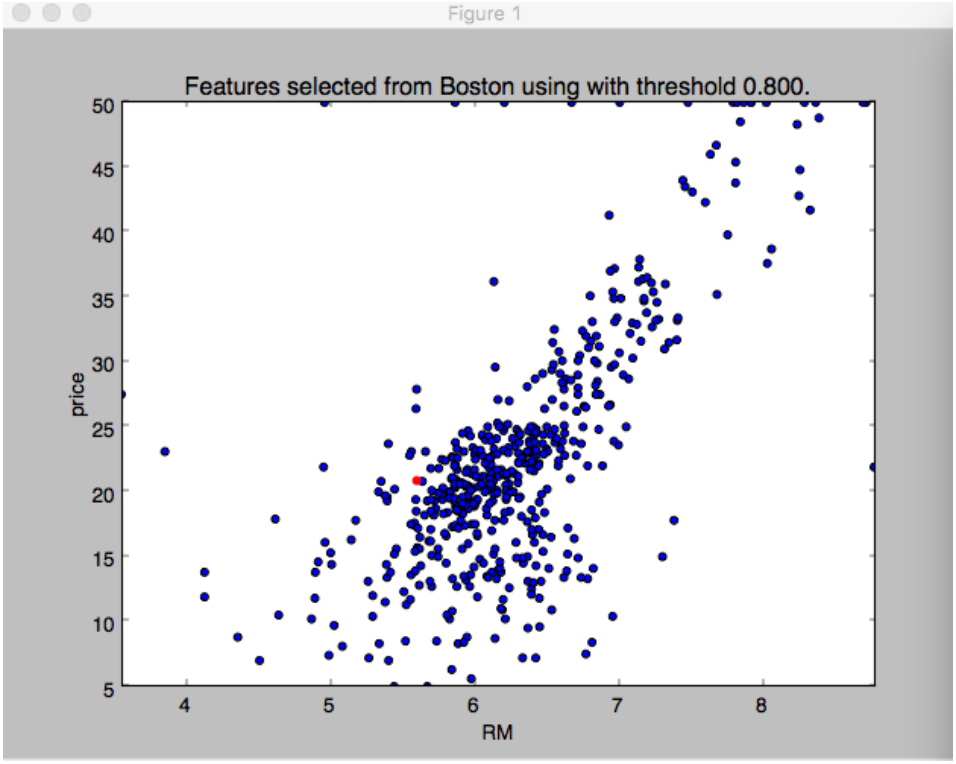
4) Model Prediction

- 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.

I ran the program several times, and got max_depth=5,6, and 7, but **most common max_depth was 5**.
- Using GridSearchCV with default 3-Fold CV, and max_depth =5, the model predicts price of **20.968**.
  Predicted value is within **0.17** σ (Standard Dev.)
- Using max_depth=7, predicted price was **19.997**, which is within .**276 Std. Dev**.

Also tried to identify most significant features using LassoCV, and found 'RM' to be the most significant feature. Plotted correlation of Price against RM and highlighted the value for predicted price (20.968) against predictor values (RM=5.6090).

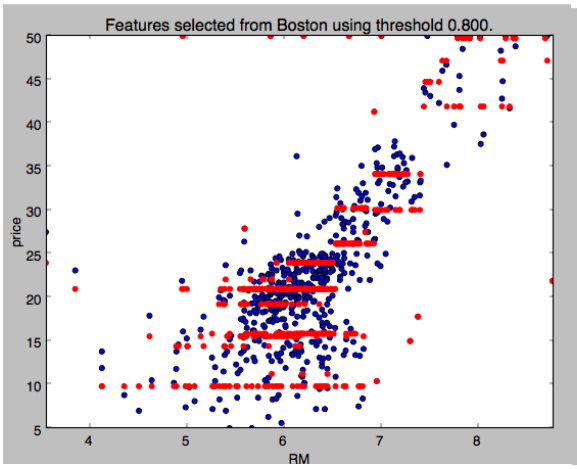
Here is the plot of Price against RM (using max_depth=5). This plot also confirms the predicted value.



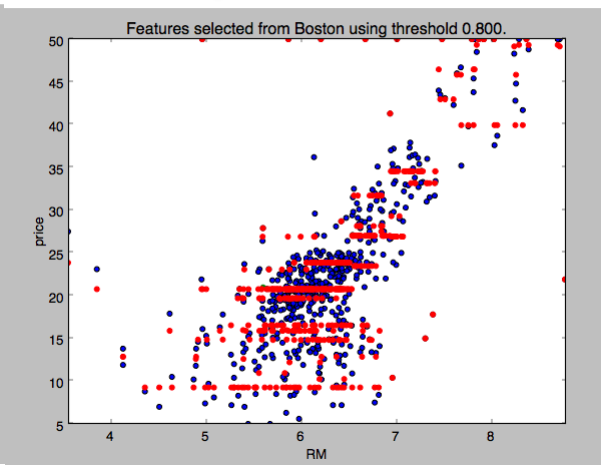
Additional Question:
Code file used: boston_feature_selection.py

I used this for feature minimization.
Here there are 2 plots, both having RM values on X-axis, and prices o y-axis.
Blue dots are given prices (“target”), while Red dots are predicted prices.
I plotted these for max_depth values of 5,6, and 7.
Although max-depth was most common, but the dots seems best fit when max_depth=7.
How can I explain this? Is this because GridSearch and LassoCV use different performance metric?

Depth = 5



Depth = 6



Depth = 7