Statistical Analysis and Data Exploration

Size of data (number of houses) = 506
Number of features = 13
Minimum price = 5.0
Maximum price = 50.0
Mean price = 22.5328063241
Median price = 21.2
Standard deviation price = 9.18801154528

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?

  The Boston housing is a regression problem; hence we have to use regression metrics, e. g. explained variation, mean absolute error, mean squared error, median absolute error, the coefficient of determination etc. At the same time, the coefficient of determination and explained variation are not the error metrics. Hence, we have to choose one from mean absolute error, mean squared error, median absolute error:

  - *median absolute error – the metric is robust to outliers, but looking at minimum, maximum, average and median prices we can notice that there are not outliers;*
  - *mean squared error – looking at standard deviation price we can notice that it is not big and we don't need higher weight for large errors;*
  - *mean absolute error – the simplest metric look quite suitable here.*

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

  It allows us to to estimate the performance on an independent data set and to check on overfitting. Otherwise, we run into issues evaluating a model because it has already seen all the data.

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

  The grid search is a way of systematically working through multiple combinations of parameter tunes, cross-validating as it goes to determine which tune gives the best performance. We use it since we want to find the best model.

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

  Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set.

  When evaluating different settings for estimators, there is still a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the *test set can “leak” into the model and evaluation metrics no longer report on generalization performance. To solve this problem, yet another part of the dataset can be held out as a so-called “validation set”: training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.*

However, by partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.

A solution to this problem is a procedure called cross-validation. It is very useful to use with grid search since it does not waste too much data and estimates how well the model generalizes; hence, it allows to build better model.

Additionally, it is quite easy to implement method.

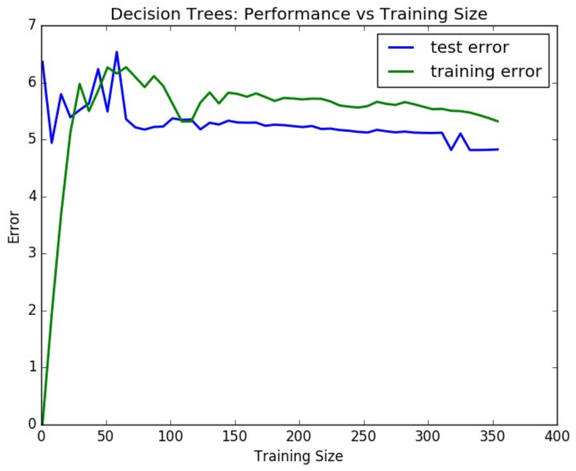
Since our data are scarce, we will finesse the problem using K-fold cross-validation: we split the data into K roughly equal-sized parts. In our case, let's use K = 10.

Analyzing Model Performance

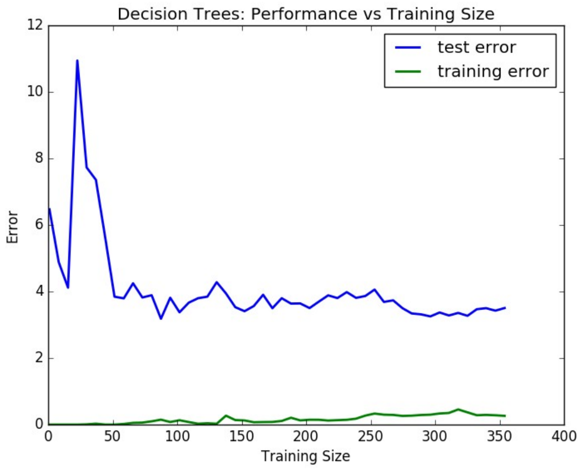
- Look at all learning curve graphs provided. What is the general trend of training and testing error as training size increases?

  Usually training and testing error converges as training size increases; training error increases, test error decreases.

- 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?



max depth 1



max depth 10

The model, with max depth equals to 1, suffers from bias/underfitting since how we can see at the picture test and train error are both high; the test error is even greater than the train one – usually it means that the model suffers from bias.

The model, with max depth equals to 10, suffers from high variance/overfitting since the train error is *low and the test error is much greater – usually it means that the model suffers from variance.*

- 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?

  The training error is decreasing; the test error is increasing. The model, with max depth equals 6, gives the best generalization; after the overfitting takes place – the error is increasing.

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.

  Average house prediction: 20.548873888

  Average best model parameter: 6

- Compare prediction to earlier statistics and make a case if you think it is a valid model.

  Yes, it looks like a quite valid model – the forecasted price is pretty close to the median. We can check that the predicted value is located in the following interval [-1σ; +1σ] from median value as well.