P1 - Predicting Boston Housing Prices.

Julio Rodrigues (juliocezar.rodrigues@gmail.com) December 2015

1 Statistical Analysis and Data Exploration

1. **Number of data points (houses)?**
    - 506
2. **Number of features?**
    - 13
3. **Minimum and maximum housing prices?**
    - Minimum: 5,000
    - Maximum: 50,000
4. **Mean and median Boston housing prices?**
    - Mean: 22.53
    - Median: 21.2
5. **Standard deviation?**
    - 9.188

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?**

    I decided to use the **mean squared error** metric for predicting Boston housing data. Besides this being a standard metric for regressions, I also considered it a suitable metric for the fact that it penalizes larger variations more severely than small variations, what is desirable when predicting values which are subject to small variations which do not entirely fit the data, but are still normal (which is the case for housing prices). A metric like the **median absolute error**, for example, did not prove itself suitable for this task, since errors are not penalized proportionally and even negligible variations can strongly affect performance and in this case sometimes causing the model to over-fit.

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

   In general, it's a common practice to split data into training and test data so we are able to test our trained model on unseen data and be sure that it is not over-fitted. Not having a test set whose data is unknown to the model, means that we are not able to realistically evaluate the performance of our model for new data.

3. **What does grid search do and why might you want to use it?**

   Grid search (GS) is a method for hyper-parameter optimization. It is an exhaustive search for an optimal set of hyper-parameters. GS searches through all - or sometimes only a subset of - hyper-parameter combinations and after evaluating the performance of each of them, chooses the one which performs best. The performance is measured using some predefined metric and usually cross-validation.

   In this current project, specifically, we use GS together with cross-validation (k-fold = 3) to choose the tree depth for which our model performs best on the test set. In this case, since we are doing a search in a 1-dimensional configuration space, a straight-forward search using depths from 1 up to 10 is enough to find an optimal depth (or a range within optimal).

   At first, I decided to use a k-fold=10, instead of the default 3. My assumption was that testing the performance on several different partitions of the data would give us a stronger evidence that the trained model was not over-fitted. After running the program several times I noticed that the optimal depth would oscillate very often with this higher k-fold value. My hypothesis is that skewed distributions in the samples with only 1/10 of the size of this already relatively small data set causes this oscillation.

4. **Why is cross validation useful and why might we use it with grid search?**

   Cross-validation (CV) is useful for assessing the performance of a given model on different portions of the available data, thus giving an estimate of how well our model is able to generalize to an unseen data set. It is also useful when we do not dispose of a large amount of data and do not want to have even less data available for training after splitting it into a training and test set.

   Using it together with grid search, we can make sure that our hyper-parameter optimization does not only apply to a very specific set of data, but that it will also perform well for most independent data sets.

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?**

   When the training size is very small, the error on the test set is very high (while it is usually very low on the training set). This happens because when given only a few data points to learn from, all the model does is to "memorize" these few data points from the training set and thus performing poorly on new unseen data. With an increasing size of the training set, the error rate on this set

itself increases by a few points, but it also decreases considerably on the test set. In other words, it does not suffer from over-fitting and is able to better generalize to unseen data.

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 su□er from either high bias/under-fitting or high variance/over-fitting?**

   Training the decision tree regressor with max depth 1 causes the model to suffer from high bias/under-fitting. This can be seen on the poor performance of the model on both, training and test sets (both $\approx 50\%$).

   On the other side, when training the decision tree with max depth 10, the error rate on the training set drops to nearly zero and the performance on the test set does not improve much and is still around 50%. In this case we speak of an over-fitting, where we have a virtually perfect score on the training set (or on data very similar to it) and a poor performance on unseen data.

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 data set and why?**

   An increasing model complexity causes the model to suffer from high variance, while a very low complexity causes the model to be biased or under-fitted. The latter happens because a shallow decision tree is barely able to model any kind of specificity in the data. The former happens here because allowing the tree to branch off very deeply, concretely means allowing the model to represent the training data in a very detailed and specific way, which does not necessarily reflect how the general population for this kind of data should be modeled. Grid search and cross-validation indicated that the optimal tree depth for this data set lies around 5.

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.**

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

   The prediction returned for the given sample, lies around 20.0. The prices of the 3 nearest data points in the data set are $27.9, 19.1, 14.2$, (in ascending order of distance). Considering this range of variation, the predicted value around 20.0 is reasonable. Also, if we consider the standard deviation in the data set of ≈ 9.0 and the fact that its mean and median are, respectively, 22.53 and 21.2, the results are very plausible.