Project 1: Predicting Boston Housing Prices

Submitted by Benjamin Tanz
benjamintanz@gmx.de

November 19, 2015

1) STATISTICAL ANALYSIS AND DATA EXPLORATION

1a) Number of data points:
506

1b) Number of features:
13

1c) Minimum and maximum housing prices:
5.0 (min) / 50.0 (max)

1d) Mean and median housing prices:
22.5328 (mean) / 21.2 (median)

1e) Standard deviation:
9.188

2) EVALUATING MODEL PERFORMANCE

2a) Which measure of model performance is best to use for regression and predicting Boston housing data? Why is this measurement most appropriate? Why might the other measurements not be appropriate here?

I consider the mean squared error the best model performance measure to use for this regression problem. This measure is appropriate, since for the regression problem at hand, we need a measure that captures the deviations of the data from the approximated function in a continuous way (as opposed to the discrete nature of classification problems). Otherwise, the measure would not help us in becoming the best real estate agent out there. Now, many functions satisfy this requirement and have desirable properties, such as the absolute value function (leading to the mean absolute error), the square function (leading to the mean squared error), or a function that takes the error to the fourth power.
For the case at hand, it is debatable whether the mean squared error is the MOST appropriate model, since once can make an argument for other loss functions, such as mean absolute deviation, as well. The decision on which loss function to use ultimately is a decision on how to trade of larger errors with smaller ones. The mean squared error provides a sensible tradeoff between large and small deviations and hence is a good choice for housing price problem.
Other metrics that are often used for classification problems, such as accuracy, precision or recall, are not appropriate for this case, as they only help for classification. For example, we could use them to evaluate how many points sit exactly on the regression function or

how many are correctly identified as being "above" or "below" the regression function, but this is not the point of this exercise. A measure such as mean absolute deviation may make sense, as discussed before.

2b) Why is it important to split the data into training and testing data? What happens if you do not do this?

Splitting the data is important to evaluate the quality of a predictive model. A model that is fit to the data always approximates some of the truth (i.e. the true and generally unknown generation process of the data) and some of the noise. Ideally, we want our model to capture the truth only, and treating the noise as noise, not truth. We can test this by applying the predictive model on data it has not seen before. This way, we can test whether we captured the true relationship sufficiently well and/or whether our model structure is to sensitive to noise. If a lot of noise is captured, the predictive performance of the model on the test and training sets will be very different, which allows us to identify this problem (which is often termed overfitting). If we would not split the data, we could not run this test and hence would not be able to analyze potential problems due to overfitting.

2c) Which cross validation technique do you think is most appropriate and why?

I think k-fold cross validation with k around 10 is the most appropriate cross validation technique. It makes efficient use of the data by splitting all data in k groups of samples and does training on k-1 groups and testing on one group. We do not need to stratify, as there are no indications that the housing data are stratified in some way. Also, k-fold cross-validation with k strictly smaller n was in many cases shown to be a better cross validation strategy than the alternative leave-on-out strategy (Kohavi 1995, Intl. Jnt. Conf. AI).

2d) What does grid search do and why might you want to use it?

With grid search, one can partition the parameter space in grids, run the model on each grid point in the parameter space, compute a score for the model, and finally compare the achieved scores on the grid to find the best parameters. Conceptually, grid search is a way to find the optimal set of hyperparameters for a given estimator, data, score function and sampling/validation scheme.

Grid search can be used to fine-tune hyperparameters and optimize the predictive ability of a model.

3) ANALYZING MODEL PERFORMANCE

3a) Look at all the learning curve graphs provided. What is the general trend of training and testing error as the training size increases?

As the training size increases, the training error increases and the test error decreases. The intuition for the training error is that with less training samples, the model can better fit the data and hence produces less error on the training data. For a given model, more training data generally (but not always) lead to a higher training error. The test error, on the other hand, decreases with a bigger training sample, as the model becomes better in capturing the truth with more training data and hence leads to better predictions on the test set.

3b) Look at the learning curves for the decision tree regressor with max depth 1 and 10. When the model is fully trained does it suffer from either high bias/underfitting or high variance/overfitting?

In the case of max depth 1, the model suffers from high bias/underfitting. This is indicated by the high error for both the training and test set. In the case of max depth 10 on the other hand, the model suffers from high variance/overfitting. On the learning curve, this is indicated by the large difference between the test and training error. Whereas the training error seems to be very low (certainly below 5), the test error is around 25. This is a clear indication for overfitting.

3c) 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?

According to the model complexity graph, the training error decreases with model complexity. This makes sense, as a more complex model leads to a better fit and a lower error on the training data. The test error also decreases with higher model complexity until a max depth of around 5 is reached. Afterwards, it remains constant around 25 for all higher max depth's.

By inspection of the model complexity graph, a max depth of 5 generalizes the dataset best. The reason is that with a higher max depth, the error on the test set does not decrease and the model starts to suffer from overfitting. With a lower max depth on the other hand, we can still decrease the test error and hence would tend to underfit the data. Hence, a max depth of 5 looks like the sweet spot in this case.

4) MODEL PREDICTION

The model predicts a house value of 21.6297 (this may vary slightly, depending on the initial conditions of the algorithm), which is roughly equal to the mean and median of the sampled housing prices computed earlier.

The final algorithm used for obtaining this prediction is a decision tree with max depth of 6, the details of which are provided in the following scikit-learn representation.

```
DecisionTreeRegressor(criterion='mse', max_depth=6, max_features=None,
           max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
           min_weight_fraction_leaf=0.0, random_state=None,
           splitter='best')
```