# Udacity Machine Learning Engineer Project 1

Boston Housing Prices

Martin Oberg

## 1 Data Exploration

The Boston housing dataset is a collection of house prices and associated features.

| | |
|---|---|
| Number of houses | 506 |
| Number of features | 13 |
| Minimum price (in $1,000s) | 5.0 |
| Maximum price (in $1,000s) | 50.0 |
| Mean price (in $1,000s) | 22.532 |
| Median price (in $1,000s) | 21.2 |
| Standard deviation (in $1,000s) | 9.188 |

Table 1: Statistics of the Boston housing dataset

We can see from the distribution of house prices (Figure 1) that the median and mean house prices are in the low $20K range. The collection of houses in the $45-50K range and the general lower frequency of any house above $25K shows that prices are not normally distributed. This will have an impact on the evaluation metric.

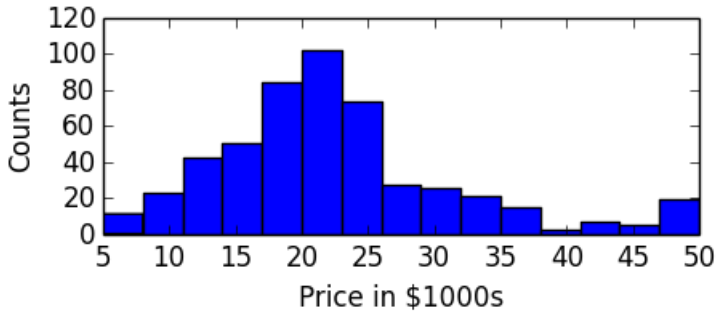


Figure 1: Histogram of Boston house prices

# 2 Evaluating Model Performance

## 2.1 Performance Metric

Predicting house price is a regression problem and requires a metric designed for such a case. The scoring metric `sklearn.metrics.mean_absolute_error` was used. As seen in Figure 1, the distribution of house prices is positively skewed with a collection of houses being sold for very high prices. Using a mean squared error would disproportionately affect those more expensive houses and result in skewing the model.

## 2.2 Testing/Training Split

The best models are ones that generalize. It is more important to fit a model that finds an underlying pattern rather than fitting a particular pattern in a sample of data. Therefore, it is necessary to test a model on unseen data to ensure the pattern holds for new cases. For the learning curve analysis the data were split in to 70% training and 30% testing.

```
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X, y,
test_size=0.3, random_state=1)
```

## 2.3 Grid Search

Grid search, `sklearn.grid_search.GridSearchCV` allows many models to be computed across combinations of parameter settings. This can be particularly powerful when used in conjunction with cross-validation.

## 2.4 Cross-Validation

Cross-validation is a method by which subsets of data are partitioned and each is used to test a model against the remainder of the data. Just as the degree of over- or under-fitting may be influenced by the particular learning curves made for this project from a single random sampling, cross validation aims to better assess the quality of fit by evaluating a model using the entire dataset.

In k-fold cross-validation the dataset is split into k sets of equal size and iteratively held out as test data while the remainder are used to train a model. This occurs automatically in `GridSearchCV` which then returns the best fit classifier. By using each subset for test purposes we are able to test the model against all of the data maximizing the usefulness of all observations. This also ensures that a single test does not dictate the model's evaluation if by chance an unrepresentative sample is taken.

Using the default 3-fold `cv` parameter for GridSearchCV cross-validates of over 3 set test sets each 33% of the entire set and trained on 67% which is close to the 30/70 split used for the learning curve visualization.

Grid search is implemented by supplying a regressor function, regressor parameters, and a scoring metric. A decision tree regressor object from `sklearn.tree` was instantiated:

```
regressor = DecisionTreeRegressor(min_samples_leaf=2)
```

Setting a minimum sample number for leaves ensures that models will have to consider more than a single data point.

Regressor parameters were set as:

```
parameters = {'max_depth':  [1,2,3,4,5,6,7,8,10]}
```

to allow the model to find the optimal depth. Although models with various depths will be found, we will see that intuitions following from interpreting learning curve graphs are well founded.

The scorer was defined by `sklearn.metrics.make_scorer` with `greater_is_better` set to `False` since errors are to be minimized.

```
bos_scorer = make_scorer(mean_absolute_error, greater_is_better=False)
```

Finally, all together:

```
grid_search.GridSearchCV(
estimator=regressor, param_grid=parameters, scoring=bos_scorer)
```

# 3 Analyzing Model Performance

## 3.1 Learning Curves and Training Analysis

Learning curves for a minimization problem, minimizing mean absolute error in this case, show training error to increase and test error to decrease as training size increases. A small sample size will yield a simple models that has low training error due to its simple nature but high test error because of its lack to generalize. Training error increases as sample size increases but approaches a limit given the allowed complexity of the model. Testing error will decrease with increased sample size as more data are considered and the model is better able to generalize to new data.

## 3.2 Learning Curves and Bias & Variance Analysis

Figure 2 shows the learning curve for a model trained with a flat decision tree of max depth 1. The test error remains high despite increased sample size showing that the model is overly simple and underfit for the complexity of data and underlying patterns. The training error is also very high across all training sizes (generally higher than even the overfit test error of Figure 3) and approaches the same underfit level of the test error.


Figure 2: Learning Curve - Max Depth 1

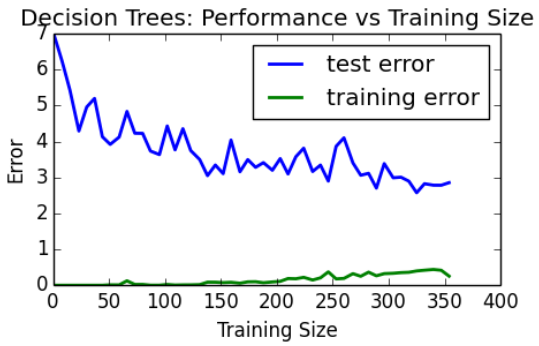


Figure 3: Learning Curve - Max Depth 10

The learning curve for max depth 10 (Figure 3) shows that the fully trained model suffers from high variance and over-fitting. The training error stays close to 0 because a perfect training fit is achieved with all of the data. However, because the model is over-fit and cannot generalize the test error remains high.

### 3.3 Error Curves and Model Complexity

In choosing the optimal model one must balance the tradeoffs of model bias and model over-fitting. An oversimplified model will miss out on patterns in the data while a model suffering

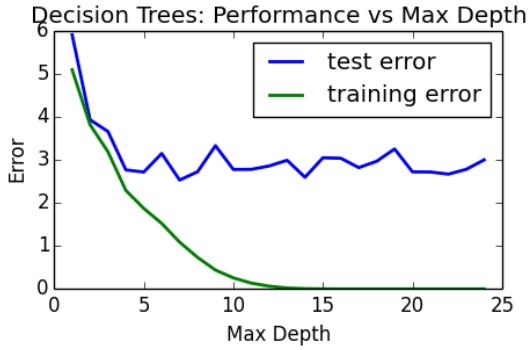


Figure 4: Learning Curve - Model Complexity

from over-fitting will have "found" patterns in the training data that do not generalize to new data. In Figure 4 we see that as the decision tree depth increases the training error drops to zero meaning that all aspects of the training set are being modelled. However, the test error quickly starts oscillating around 3 indicating that model performance cannot improve beyond this point for unseen data.

At a feature depth of 6 the training error has departed from the test error. Using one less feature, i.e. 5, yields good training error without over-fitting.

## 4 Model Prediction

The predicted house price for the featureset `[11.95, 0.00, 18.100, 0, 0.6590, 5.6090, 90.00, 1.385, 24, 680.0, 20.20, 332.09, 12.13]` was computed for 25 models with a mean price of \$20,406 median of \$20,766 and a standard error ($std/\sqrt{N}$) of \$137. This would appear to be a slightly below average house when compared to the overall median price of \$21,200.

The max depth of the model's best estimator is usually 5 for more than half of each GridSearch fitting. This depth is in line with the intuitions from reading Figure 4. This house would appear to be slightly below average for the area when compared to the overall median price of \$21,200.

A scatterplot of Target price vs Predicted price for one of these models is shown in Figure 5. This is a reasonable model as it shows a small variance without over-fitting the model and predicting prices perfectly.

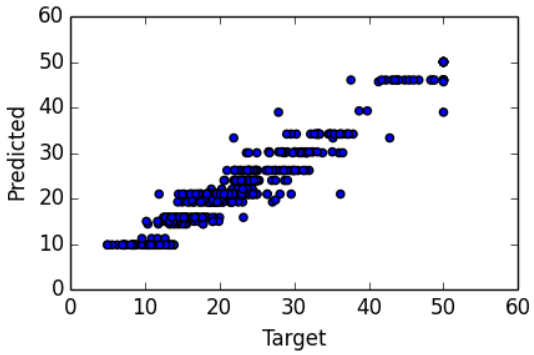


Figure 5: Scatterplot of Target and Predicted Prices for one model