# PREDICTING BOSTON HOUSING PRICES

DIEGO ROQUE MONTOYA

Abstract. We use Decision Trees[2] with model selection to predict the prices of houses in the Boston market using data from ICS[1].

## 1. Statistical Analysis and Data Exploration

First we calculate some properties of the dataset. Note that the housing prices are measured on thousands of dollars.

1.1. **What is the number of data points?** There are 506 data points.

1.2. **What is the number of features?** There are 13 features.

1.3. **What are the minimum and maximum housing prices?** The minimum and maximum housing prices are 5 and 50 thousand respectively

1.4. **What are the mean and median of housing prices?** The mean of the housing prices is 22.532 thousand and the median price is 21.2 thousand.

1.5. **What is the standard deviation of the housing prices?** The standard deviation of the housing prices is 9.188 thousand.

Table 1. Summary

| Measurement | Boston Dataset |
|---|---|
| Data Points | 506 |
| Features | 13 |
| Minimum Price (thousands) | 5 |
| Maximum Price (thousands) | 50 |
| Mean Price (thousands) | 22.532806 |
| Median Price (thousands) | 21.2 |
| Standard Deviation (thousands) | 9.188012 |

*Date*: January 29, 2016.

## 2. Evaluating Model Performance

To compare different models, there are several measurements at our disposal, like $r^2$, Explained Variance, Mean Absolute Error (MAE), and Mean Square Error (MSE). We discard the first two because we want to measure error and the first two measure how good the model fits the data.

When compared to MAE, MSE gives a greater penalty to large errors. We want to find the price our client should sell their house, so greater errors would significantly impact the opportunities for selling the house. Because of this we prefer MSE over MAE.

Furthermore, this is the metric that makes most sense, as internally it's what the Decision Tree Regressor uses for splits. If it's optimizing over a metric, it should be measured over the same metric.

We split the data in training and test data, in a ratio of 75% to 25%, to obtain a reasonable estimate of how well our algorithm would fare in the real world, with new data, assuming we trained over a representative sample of the real world data. This split is done randomly to avoid patterns in sequential data. The training data is used in the learning phase of the classifier and the testing data to score the resulting learner.

Otherwise we would not be able to get a reportable score, that's comparable with other results. The test data serves as a way to estimate performance on novel data and as a check for over fitting. Having test data helps when comparing different models and analize the bias variance tradeoff, making sure our model generalizes.

Decision Trees can be prone to overfitting. To avoid that, we will tune the parameter of the maximum depth and maximum number of features allowed to prevent overfitting while minimizing error, thus getting a model that generalizes better.

Our strategy to find the best parameters is as follows. Grid Search does a brute force search, evaluating every combination of the parameter options, and selects the one that generalizes best. We can't train using the test set, and to train with the whole training set would be prone to overfitting.

So we scored with a 3-fold cross validation from the training set, which splits the data in three parts, trains with two of those parts and tests with the remaining one, then averages the three scores to get the overall score.

By doing cross validation, we maximize the usefulness of our data. This is especially useful when having small data sets. Together with a random shuffle, it also prevents splitting the data in an unbalanced way. This will be the strategy that we will use to find the best parameter.

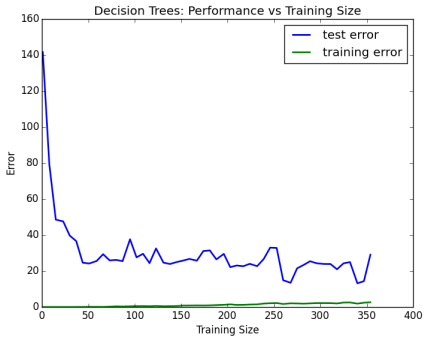
## 3. Analyzing Model Performance

We can see from Figure 1, that across the different models, as the data increases the testing error decreases for the first data sizes for the first few values. After that, it stays around a value, with varying levels of perturbations. It's similar to an asympthotic behavior with noise. The more complex models tend to have greater perturbations.

Similarly, the training error increases for the first few sizes, because for those values the classifier basically remembers the whole set. Intuitivelly, the complexity of the classifier is greater than the complexity of the data for small sizes.


(A) Max Depth = 1


(B) Max Depth = 3




(C) Max Depth = 7


(D) Max Depth = 10

FIGURE 1. Performance vs Training Size

Comparing the cases where the maximum depth is 1 and when the maximum depth is 10, we see that the former has a significant bigger error on both the training and the test sets than the latter. On the other hand, there's a closer relation between the training and test errors when the maximum depth is 1 than when it's 10, which has a training error of almost zero. This indicates that training the decision tree with a maximum depth of 1 underfits the data and it's left with high bias, while training with a maximum depth of 10 overfits the data and it's left with high variance.

We can see from Figure 2 that while increasing the maximum depth decreases to almost zero the training error, the testing error decreases until a maximum depth of around 6, then increases slightly. So lower maximum depths have high bias while high depths have high variance. This suggest we can select a model that best generalizes with a maximum depth of 6, which is the global minimum.

## 4. Model Prediction

Running Grid Search with 3-fold cross validation gives us a model with maximum depth of 6 and maximum of 3 features searched per split. This is consistent with the model performance graph in Figure 2. This model predicts that a house representing

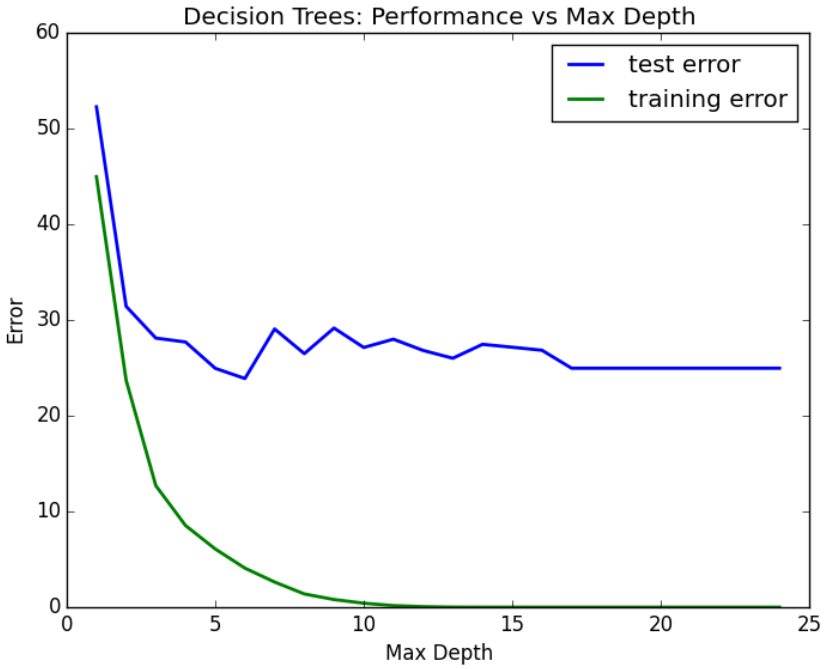



FIGURE 2. Performance vs Max Depth. Ignores Max Features.

the datapoint [11.95, 0.00, 18.100, 0, 0.6590, 5.6090, 90.00, 1.385, 24, 680.0, 20.20, 332.09, 12.13] would have a predicted price of 20.5133 thousand.

This is close both to the mean price of 21.532 thousand and the median price of 21.2 thousand, so it makes sense given that the standard deviation is 9.118 thousand. Note that we would expect that around 68% of the prices be one standard deviation from the mean. It's five nearest neighbors have prices of 27.9, 19.1, 14.2, 50.0, and 16.7 (thousand). All these values, except for the 50, are within 1 standard deviation.

## REFERENCES

1. M. Lichman, *UCI machine learning repository*, 2013.
2. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

*E-mail address*: droque@mit.edu