Udacity - Machine Learning Engineer Nanodegree

P1: Predicting Boston Housing Prices

A model evaluation and validation project using Boston housing data from Scikit Learn.

Project Description

You want to be the best real estate agent out there. In order to compete with other agents in your area, you decide to use machine learning. You are going to use various statistical analysis tools to build the best model to predict the value of a given house. Your task is to find the best price your client can sell their house at. The best guess from a model is one that best generalizes the data.

For this assignment your client has a house with the following feature set: [11.95, 0.00, 18.100, 0, 0.6590, 5.6090, 90.00, 1.385, 24, 680.0, 20.20, 332.09, 12.13]. To get started, use the example scikit implementation. You will have to modify the code slightly to get the file up and running.

When you are done implementing the code please answer the following questions in a report with the appropriate sections provided. Answers provided in blue. Instructor Feedback in red.

1. Statistical Analysis and Data Exploration

Statistic to Analyse	Explored Data
Number of data points (houses)	506
Number of features (13 dimensions per)	506
Min Housing Price	5.00 ($5,000)
Maximum Housing Price	50.00 ($50,000)
Mean Housing Price	22.53 ($22,530)
Median Housing Price	21.20 ($21,200)
Standard Deviation of Housing Prices	9.19

2. 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?

  Mean Squared Error is best for this project because it gives more weight to larger errors. The Boston housing data is a continuous range of prices and therefore the model requires a regression

metric. Therefore, classification metrics like accuracy, precision, and recall are not appropriate. Some regression metrics like R^2 are not a measure of "error", but rather of "score" or how well the model fits the data. Therefore for the purpose of analysing error, R^2 is not appropriate, Although MSE was chosen, here's a list of useful regression error metrics:

- **Mean Absolute Error**: MAE is the averaging of the list of absolute values of the difference between(the error) actual and predicted values. Taking the absolute value ignores the direction of the error (positive or negative), but keeps its magnitude. Unlike Mean Squared Error, all errors have equal weight when they are averaged.
- **Median Absolute Error**: MedAE is the median of the list of absolute values of the difference between(the error) actual and predicted values. The median is great for highly skewed distributions and is therefore not useful metric for the Boston housing data.
- **Mean Squared Error**: MSE is the averaging of the list of squared values of the difference between(the error) actual and predicted values. MSE is used when large errors are undesirable because errors are squared before they are averaged, which gives higher weight to large errors.
- **Root Mean Squared Error**: RMSE is the square root of MSE and thus keeps the original units of the data. In the case of this project we are looking for trends in the testing and training error over our training data so it's not exactly necessary.

You are correct, Mean Squared Error (MSE) is an appropriate metric for this case, because the problem requires the use of a regression error metric and this metric penalizing larger error more than smaller.
Excellent overview about different regression scoring errors.

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

  It is necessary to split the housing data into separate training and testing data because each dataset performs unique roles. The training data finds the parameters of the model, while the testing data verifies the performance of the model after it has been fully trained, mimicking real-world data. Splitting the data makes the model generalized because it evaluates itself on the test set by using separate data than it was trained. If the housing data is not uniquely split, the model will train and test on the exact same data which will will always evaluate to be 100% correct. This is a misleading result because the model overfitted its own data and when unique data is applied the model will likely give unintended results.

  Well Done!! Splitting the data allows us to estimate how well the model generalizes to unseen data.

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

  Grid search automatically tweaks model parameters to optimize performance of the model through iteration. As it works through multiple parameter tunes it cross-validates to determine which tune gives the best model performance. It is a much faster and less stressful option for finding optimal model parameters rather than do it manually.

  You are correct, Grid search perform hyper-parameter optimization or model selection simply by exhaustive searching through a manually specified subset of the hyperparameter space.

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

Cross validation is a method for reducing overfitting and generalizing our model to unseen inputs. In particular, K-fold cross validation allows us to use all of the data for training and testing even though we split the data into separate training and validation sets.

You are correct, "K-fold cross validation allows us to use all of the data for training and testing even though we split the data into separate training and validation sets". I strongly urge you to expand this section and explain how cross validation does that .You might find these links useful, wiki http://www.anc.ed.ac.uk/rbf/intro/node16.html

3. Analyzing Model

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

  As training size increases, initially training error increases and testing error decreases. Eventually both training and testing error plateau.

- 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?

Figure 1. Max Depth 1 Learning Curve

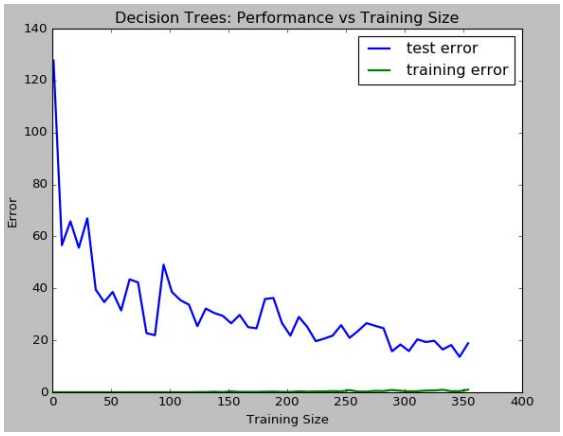

Figure 2. Max Depth 10 Learning Curve

The learning curve with max depth 1 suffers from high bias/underfitting because the training and testing errors are both high and close together when the model is fully trained. This suggests the model is not complex enough to generalize the data.

The learning curve with max depth 10 suffers from high variance/overfitting because the training error is significantly lower than the testing error when the model is fully trained. This suggests the model is too complex because it performs very well during training but not so well during testing.

You are correct, training error continues to decrease while the testing error reach a minimum, while model complexity increases. I want to point your attention to the fact that as model complexity increases, the model requires more iterations to learn the underlying data.

- 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?

  As model complexity increases, training error decreases and starts to plateau towards zero. Furthermore, test error decreases until after certain a point where the further model complexity just adds noise to our model. This point occurs when test error starts to increase as training error continues to decrease.

  My initial idea for the model that best generalizes the dataset is a “sweet spot” where the test error trend (red curve in Figure 3.) is minimum. With this I thought the best model appears to be around max depth 7. However upon further analysis of the graphs my initial idea of finding the red trend line is invalid because the test error after max depth 5 is volatile. By running many trials of model complexity graphs and looking at the test error (blue curve), test error is is around 18 (unitless) in Figure 3. and around 10 in Figure 4. This is a significant range of difference in errors and can not possibly be a stable generalization of the dataset!

  Max depth 5 is the true “sweet spot” where the model best generalized the dataset. At this depth the test error remains constant between 11 or 12 over these many graphs. Therefore, everything to the left of max depth 5 is underfitted while everything to the right is overfitted.

  Yes you are correct, the idea is that we measure the performance on the test set, because this is a data set that the model do not know. For the test error we looking for the "max depth" that will minimize the error, at this minimum the model is optimal. In addition it is important to note that just because we can increase the model complexity does not mean we should. Increasing model complexity increases the amount of data required to train the model, which may or may not be available. This might also increase the computational resources need for training and testing a model since some model might require additional memory or calculations for additional features.

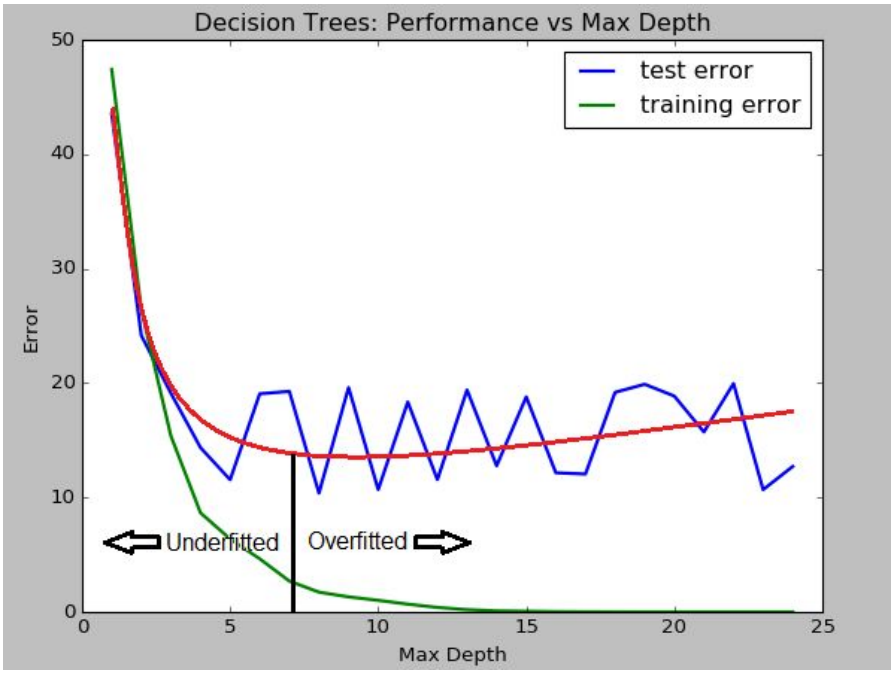

Figure 3. Performance vs. Max Depth Trial 1

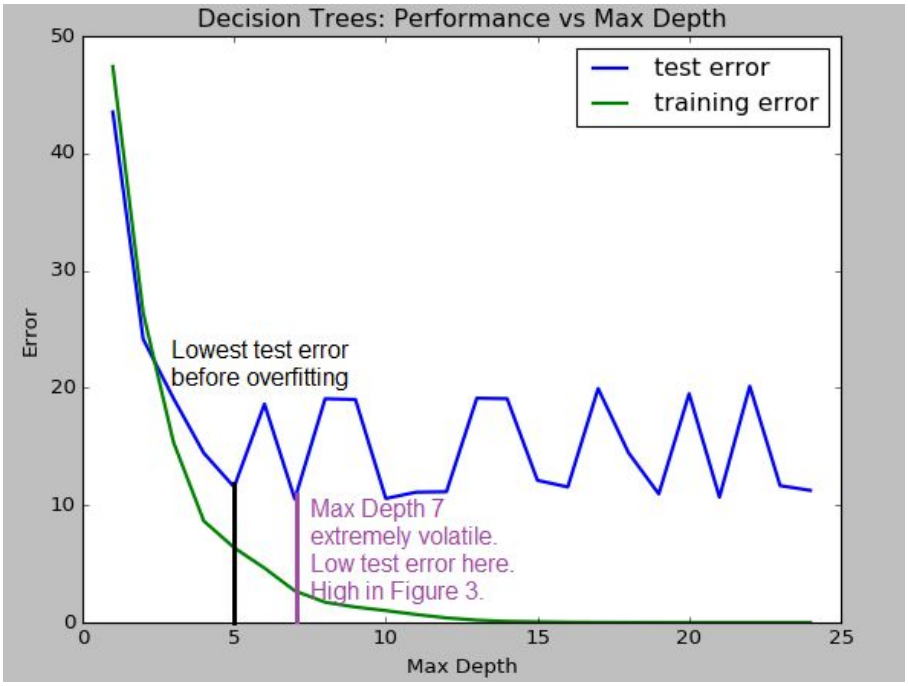

Figure 4. Performance vs. Max Depth Trial 10

4. 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.

  DecisionTreeRegressor(criterion='mse', max_depth=4, max_features=None,
  max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
  min_weight_fraction_leaf=0.0, presort=False, random_state=None,
  splitter='best')
  House: [[11.95, 0.0, 18.1, 0, 0.659, 5.609, 90.0, 1.385, 24, 680.0, 20.2, 332.09, 12.13]]
  Prediction: [21.62974359]

  Other Tests:
  Prediction: [20.96776316]
  Prediction: [19.32727273]
  Prediction: [20.96776316]
  Prediction: [19.99746835]
  Prediction: [20.76598639]

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

  The prediction 21.63 (~$21,630) falls in the range of Boston housing price data, 5.00 ($5,000) to 50.00 ($50,000), is very close to the median 21.20 (~$21,200) and mean 22.53 (~$22,530), and is well within 1 standard deviation of the data. Multiple trials show that the prediction is repeatable within a range from about 18.00 to 22.00. Based on this analysis it's a valid model.