P1: Prediction Boston Housing Prices

1) Statistical Analysis and Data Exploration

Number of data points (houses)	506
Number of features	13
Minimum price (in $1000's)	5.000000
Maximum price (in $1000's)	50.000000
Mean price (in $1000's)	22.532806
Median price (in $1000's)	21.200000
Standard deviation	9.188012

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 calculates the average of the sum of squared residual errors (the difference between predicted and true value) for all data points. This function would be the most appropriate measurement since squaring the residual error converts all errors as positives and also emphasizes larger errors rather than smaller errors. Deriving the derivative from function can also be obtained for gradient descent calculation.

In comparison:

- Explained Variance Score: This function measures the residual performance by computing the explained variance regression score in the model and is not an error metric as originally requested.
- Mean Absolute Error: While mean absolute error takes the total absolute error of each example and averages the error based on the number of data points, each individual error is weighted equally and therefore this is a linear error function.
- Median Absolute Error: Median absolute error is similar to mean absolute error except it considers the median absolute error rather than the average. As a consequence, this measurement is more appropriate for describing data that is heavily skewed.
- R2 Score: R2 score computes the coefficient of determination which measures how well future samples are likely to be predicted by the model and not an error metric.

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

Splitting the housing data into training and testing data sets performs two functions: the training data set is used to train the given model, whereas the testing data set (which is independent from the training data set) is used to evaluate the performance of the trained model by computing the test error, i.e. the average loss on data points in the test data set. Using the housing data without splitting would not be a good measure of predictive performance since it would be overly optimistic to measure the performance when the model was fitted with the training data in the first place.

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

Grid search is a method that uses an estimator object (ex: DecisionTreeRegressor), a parameter grid (as a dictionary or list of parameter dictionaries) whose parameters are associated with the passed estimator object, and methodically builds and evaluates a model for each combination of parameter specified in the grid via cross validation.

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

The idea of cross validation involves partitioning the data into subsets and calculating the average validation error; in the case of the common used k-fold cross validation, the procedure involves partitioning the data into k folds and, for each iteration in a loop that is run k times, calculate the validation error where:

- one different fold is designated as the validation data (note: this will result in each fold being designated as the validation data exactly once after processing the loop)
- the remaining k-1 folds are designated as the training data

The average of the validation errors is then computed to yield the overall performance measure. This corrects the optimistic nature of training error and derive a more accurate estimate of the model prediction performance. As well, this procedure is useful where the data is too small to divide into training, validation, and test sets. Finally, the grid search parameters (max_depth, in our case) that resulted in the best k-fold average will be applied when fitting the entire data set.

3) Analyzing Model Performance

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

As observed in Figure 1 and Figure 2, the training error increases and the testing error decreases as the training size increases, where both curves eventually approach their respective error values.

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?

In Figure 1 where the max depth is 1, high bias/underfitting is observed for a low complexity learning curve since both the training and test errors are high when the model is fully trained.

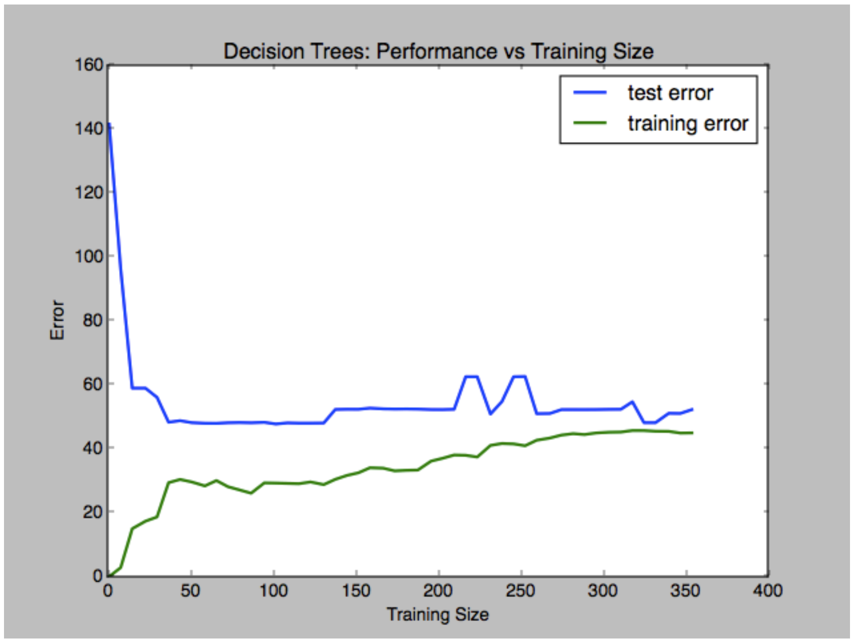

Figure 1 Performance vs. Training Size (max depth 1)

In contrast, as shown in Figure 2 where the max depth is 10, high variance/overfitting is observed for a high complexity learning curve since there is a large gap between training and test errors. Also note that the training error is close to 0 for all training sizes.

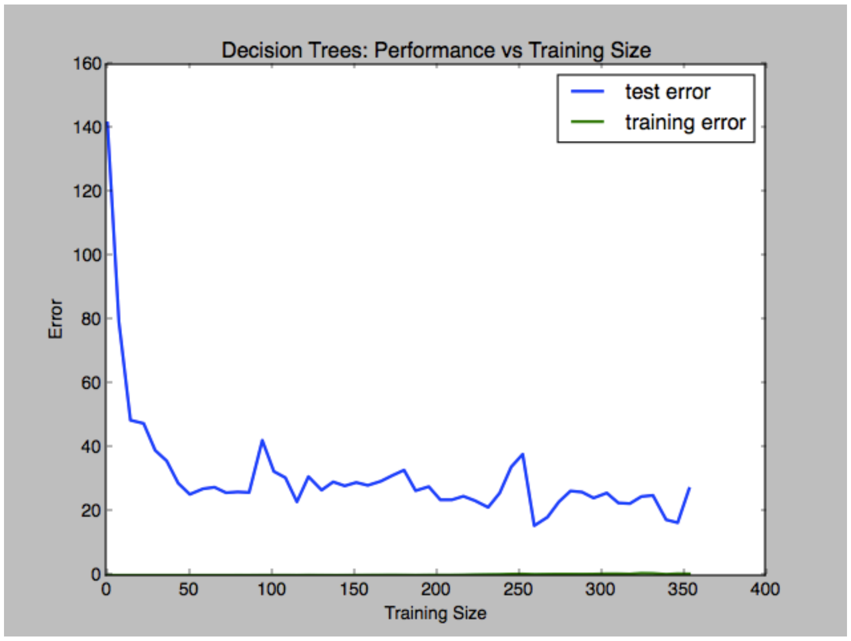

Figure 2 Performance vs. Training Size (max depth 10)

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 shown in Figure 3, the training error decreases towards zero as the max depth increases, but the testing error decreases at a slower rate than the training error from max depth 1 to 6 and then slightly increases (overall) with more variance afterwards. Based on this observation, max depth of 6 would best generalize the dataset since it yielded the minimum test error and provided a balance between bias and variance; as the model complexity increases, bias decreases and variance increases. Please note that a defined random_state value (say, 0) was defined in order to obtain reproducible randomization.

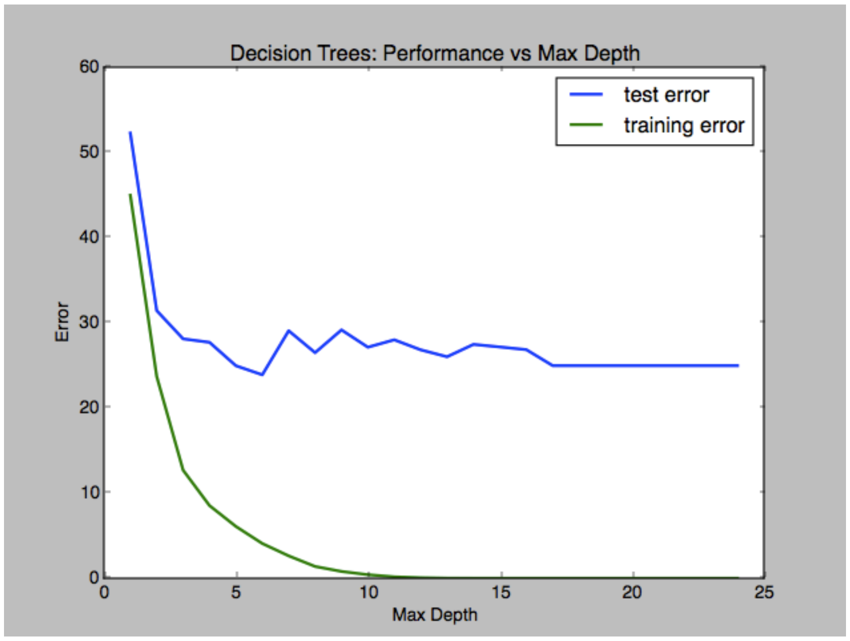

Figure 3 Performance vs Max Depth

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.

Based on using model prediction, it was observed that max_depth == 6 yielded the predicted price (in $1000's) of 20.76598639 (assuming that we pre-define the random_state for reproducible randomization).

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

In comparison to earlier statistics found boston_housing.csv, consider some similar samples (A and B) from the existing data set and compare with sample x:

Sample	*CRIM*	*ZN*	*INDUS*	*CHAS*	*NOX*	*RM*	*AGE*	*DIS*	*RAD*	*TAX*	*PTRATIO*	*B*	*LSTAT*	*MEDV*
A	11.9511	0	18.1	0	0.659	5.608	100	1.2852	24	666	20.2	332.09	12.13	27.9
B	12.0482	0	18.1	0	0.614	5.648	87.6	1.9512	24	666	20.2	291.55	14.1	20.8

vs.

Sample	*CRIM*	*ZN*	*INDUS*	*CHAS*	*NOX*	*RM*	*AGE*	*DIS*	*RAD*	*TAX*	*PTRATIO*	*B*	*LSTAT*	*MEDV*
x	11.95	0	18.1	0	0.659	5.609	90	1.385	24	680	20.2	332.09	12.13	20.8

Given the above observation, it seems that the predicted price of sample x is reasonable when considering the similarity in terms of features with samples A and B, as well as the mean and median prices (in $1000s) of the data set (22.5328063241 and 21.2, respectively).

Also, the learned model was fitted using the most appropriate max depth parameter obtained through cross validation via grid search as explained earlier, where it yielded a low mean square error value (4.75866115391) and high R2 score (0.94363082003).