Machine Learning Nano Degree - Project 1
Reynard Hilman

Predicting Boston Housing Prices Report

1) Statistical Analysis and Data Exploration

Number of data points (houses): **506**
Number of features: **13**
Minimum price: **5.0**
Maximum price: **50.0**
Mean price: **22.533**
Median price: **21.2**
Standard deviation: **9.188**

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?

I run the GridSearchCV over 100 iterations for each performance metric to find the trend of max_depth chosen by the grid search. Here is the result:

performance metrics	max_depth			
	mode	mean	range	std. dev
mean_squared_error	4	5.71	4-10	1.99
mean_absolute_error	**4**	**5.2**	**4-8**	**1.1**
r2_score	4	5.39	4-10	1.6
median_absolute_error	5	5.72	5-7	0.69
explained_variance_score	4	5.13	4-10	1.62

From the chart observation using different metrics, all of them show that the best max_depth is at 4. That is the max_depth where adding complexity to the model does not reduce the test error.

So looking at the max_depth result for each metrics, **mean absolute error** stands out as the metric that returns max_depth=4 with the lowest standard deviation of the max_depth and the smallest

range, meaning it is more consistent in returning the max_depth that is closer to the optimum complexity based on our chart observation.

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

We split the data into training and testing set so we can validate that the model we have built using the training data will in fact predict the test data with good accuracy. It lets us optimize the model for both training and testing data set.
If we use all the data set for training, we can run into problem where we overfit the model to the training data. Overfitting might look good on training data because we can minimize error on training data, but when given a new data set the model might have a very high error.

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

Grid search does an exhaustive search to find the best parameters combination for our model. We use grid search to find a model complexity that is optimum for a specific performance metric.

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

Cross validation lets us use the limited data set to train and test the model, and then cross validating it N times (depending on how many fold cross validation is used). It prevents overfitting the model to the data set. Because grid search's goal is to find the best parameters for the model, cross validation is an important tool to prevent finding the best parameters for a model that overfit the 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 the training size increases, the trend for training error is to increase at a slower and slower rate. The trend of testing error is to decrease at a slower and slower rate. Eventually both of training and testing error start to level off.

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?

With a max_depth of 1, it suffers from high bias/underfitting, because the error is still relatively high for both the training and testing data. With a max_depth of 10, the model suffers from high variance/overfitting, because the error is low for the training data, but the testing error is much higher than the training error.

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?

Increasing the model complexity eventually reduces the training error to 0. Increasing complexity to a certain point reduces the testing error. Beyond that point, increasing complexity does not have any effect in reducing the testing error.

The max_depth that best generalizes the dataset is 4, because increasing the max_depth beyond 4, does not reduce the testing error.

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.

To run the GridSearchCV 100 times and find the max_depth trend returned by the grid search:

```
python boston_housing.py -mdt
```

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

Using sklearn.neighbors.NearestNeighbors, I find 3 nearest neighbors and their prices. The nearest neighbors mean price is **20.4,** with standard deviation of **5.6**. Our model with a max_depth of 4, predicts the price to be **21.63**, which is pretty close to the nearest neighbors mean price, and well within the standard deviation.
Note: I did not choose more than 3 nearest neighbors because the 4th nearest neighbor has a price of 50 (outlier), which would skew the nearest neighbor comparison.