Project 1 - Predicting Boston Housing Prices

1) Statistical Analysis and Data Exploration

- **Number of data points?**

Number of data points / rows: 506

- **Number of features?**

Number of features: 13

- **Minimum and maximum housing prices?**

Minimum housing price: 5.0 on this scale
Maximum housing price: 50.0 on this scale

- **Mean and median prices of Boston housing prices?**

Mean house price: 22.5328063241
Median house price: 21.2

- **Standard deviation?**

Standard deviation of housing prices: 9.18801154528

2) Evaluating Model Performance

- **Which measure of model performance is best to use for regression and predicting Boston housing data? Why is this measurement most appropriate? Why might the other measurements not be appropriate here?**

The best model performance for regression is 'Mean Squared Error', which hard-coded in python is np.sum((y_true - y_pred)**2). This shows the residual distance squared and then summed up and the lowest error score wins. We could also use mean absolute error or even root mean squared error, however mean squared error is more common and requires less computation time. Mean absolute error is more robust to outliers than MSE because MAE assigns equal weight to the data whereas MSE emphasizes the extremes. Root mean squared error just square roots the MSE to make it a more of an interpreted statistic, since it has the same units as the quantity plotted. Obviously we cannot use any classification metrics to score a regression problem so MSE is the most commonly used and understood.

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

We need to split our data into a train / test split because we need a hold out set to test our model on and determine how the model is doing. If we don't do this and use all of the data to build our model, then basicly our model is just 'memorizing' our data and therefore our model will not scale well to new, unseen data. We do this as we do not want to over-fit our data and the testing performance is a better estimate on how well our model is doing than training score.

- **Which cross validation technique do you think is most appropriate and why?**

The problem with the train / test split approach is the chance of high variance estimate since changing which observations happen to be in the testing set can significantly change testing accuracy especially with low testing points. The most appropriate cross validation technique is K fold cross validation since it is better for dealing with the splitting on the dataset. It allows for a more accurate estimate of out-of-sample accuracy and allows for a more ‘efficient’ use of data (every observation is used for both training and testing).

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

Grid Search is mostly used for parameter tuning. It as an awesome addition to sklearn as it allows you not to have to write for loops and run your model for each parameter tuning. Grid Search will search for desired parameters like the max-depth in a decision tree or gamma in a SVC and automatically update the model to use the best parameter.

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?**

The general trend of the testing and training error as the training size increases is that the training error and testing error get closer together. This is because as the size increases the training set with more data will decrease the variance and the model will always get a better estimate when there is more data to train on, therefore the testing set error will be better.

- **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?**

When the model is fully trained for the max depth of 1 the model suffers from high bias / underfitting because the training error and testing error is very high and are close in the middle of the graph. When the max depth is 10 the model suffers from high variance / overfitting because for the training set the model is simply memorizing the data as the MSE is very low.

- **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 the max depth of the model increases the training set error decreases showing that the model is overfitting the data while the test set error drops in the beginning and levels off at about around a max depth of about 5, Based on this relationship, in my opinion a simpler model is always a better one, so I would conclude that the a max depth of about 5 best generalizes the data. This is because there is no real major drop in testing set error and the bias - variance trade-off seems appropriate. And the grid search choice was {'max_depth': 4}.

4) Model Prediction

- **Model makes predicted housing price with detailed model parameters**

By predicting on the feature set of [11.95, 0.0, 18.1, 0, 0.659, 5.609, 90.0, 1.385, 24, 680.0, 20.2, 332.09, 12.13] the corresponding features the best model returned by the grid_search tuning algorithm predicted the value of [21.62974359] for the price of the home.

- **Compare prediction to earlier statistics**

This predicted price of 21.62974359 is very close to the mean and median of the prices of the original data, therefore I would have high confidence in this prediction as it falls within the bulk of the data and not toward the ends of the spectrum where the confidence interval would be much larger definitely with the high range of potential housing prices.