Udacity Machine Learning: Jean Ro

Project 1 Report
Predicting Boston Housing Prices

Various statistical analysis tools such as NumPy and scikit-learn are used to build the best model to predict the housing price based on the Boston area housing dataset.

1) Statistical Analysis and Data Exploration

- Number of data points (houses)? 506
- Number of features? 13
- Minimum and maximum housing prices? Minimum price is 5.0 and maximum price is 50.0
- Mean and median Boston housing prices? Mean price is 22.5328 and median price is 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've used mean absolute error with the function mean_absolute_error to measure the error and to evaluate the performance. Mean squared error is useful for measuring variance and spread of error, but it does not reflect the true errors for housing price type. Median absolute error does not accurately represent the error since it's just a number that falls in the mid point. R2 is not a measure of error and cannot be used in this context.

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

The training data are used to construct the model and the test data are used for model evaluation and selection. If there is no splitting of data and only one set of data is used, then there is a danger of model getting overly complicated leading to overfitting.

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

  Grid search is used to optimize the parameters that are passed to estimators. It does it by searching a parameter space for the best cross validation performance score. GridSearchCV generates all combinations of parameters by constructing grids with elements in param_grid value. Then it generates scores for all parameter combinations and parameter settings that gave the best results.

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

  Using cross validation, we can take care of the problem of having to reduce the number of training data. This approach can be computationally expensive but does not waste data by not having to reserve fixed test set. It's used with grid search since it's an effective way to measure performance when the number of samples is not very large. Instead of default 3 fold cross validation, I've used 5 fold cross validation. I noticed from the learning curve that the error has not stabilized with the 3 fold training size of 330. Using default 3 fold also resulted in inconsistent results for each run. Therefore, I have found that the best number to use for k fold validation is 5, which gives consistent results after each run.

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?

  For very small training sizes, the training error is very small since it's easy to fit small number of data. As the training size increases, the training error increases since it's harder to fit all data points without error as the number increases. For very small training sizes, the test error is very large but as the training size increases, the test error decreases. For models with high bias, for example with low max_depth numbers less than 4, the training error and test error converge to certain values at relatively low number of training set size and increasing the training size does not help. However, for models with high variance as in the case of max_depth number around 10, there are big gaps between the training error and test error and increasing the training size does make the gap smaller (but in a very slow rate).

- 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 the max depth of 1, it suffers from bias/underfitting and it can be seen by both training and testing error converging very quickly at higher error values at smaller number of training samples. With the max depth of 10, the training error became very low but the test error didn’t change beyond max depth of 4 or 5. Therefore, it suffers from high variance/overfitting.
- 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 model becomes too complex, the difference between test error and train error increases. the max depth of 4 best generalizes the dataset because it’s the point where the test error and train error are about the same and the model is not too complex nor is too simple.

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 is used for modeling and its parameters are optimized using GridSearchCV. The following parameters are optimized: splitter, max_depth, min_samples_leaf. For other parameters, the default values seem to work best. The following are the best values for those optimized parameters: splitter: ‘best’, max_depth: 8, min_samples_leaf: 3. The best score was 2.719. Using these parameters, the predicted house price for the provided feature array is 18.81666667

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

  One approach to validate the predicted data is to compare it to statistical mean/median and spread. For the housing dataset, the mean price is 22.5328, median price is 21.2 and the standard deviation is 9.188. The predicted value 18.81666667 is within the one standard deviation range of 13.345 and 31.72.

  The other way to see if the prediction makes sense is to find the nearest neighbors of the feature set used for prediction and then compare the predicted value to the average of neighbor's values. When NearestNeighbors module with n_neighbors = 25 is used, the mean of neighbors is 18.912, which is very close to the predicted value of 18.8167. Based on this, the model is valid.