Boston housing prices - A report on model evaluation and validation

1) Statistical Analysis and Data Exploration

Number of data points (houses)?

506

Number of features?

13

Minimum and maximum housing prices?

Minimum price: 5 Maximum price: 50

Mean and median Boston housing prices?

Mean: 22.533

Median: 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?

"mean_squared_error" is used as the measure of model performance.

Why do you think this measurement most appropriate? Why might the other measurements not be appropriate here?

Another error score available is mean_absolute_error. Since mean_squared_error squares the error for every sample, the outliers get heavily penalized and their impact in the model is reduced. we don't get this benefit from mean_absolute_error. Also, since the square function is differentiable, it lends itself to finding the minimum of the cost function algebraically.

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

If we don't split the data into test and training set, and train the model on complete data set we may end up with the model that overfits just the training data and fails to generalize. So it is important to split the data into training and testing sets, and use the testing set for validation of the model.

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

"grid search" takes a parameter grid as one of the inputs (parameter grid depends on the estimator used) and runs training for each of the parameter from the grid. It also takes scorer function as input and uses it to compare the runs and choose the best parameter set. It is useful when we don't know what parameter will be ideal for an estimator and want to try a set of parameters and choose the best one.

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

Splitting data into training and test set makes less data available for training. With cross validation in place, we get the advantage of training and testing on the complete data set. This is because, in each iteration of cross validation, portion of data is used for testing and the remaining for training. This repeats till the model is trained and tested on all the available data.

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 training error increases as the training size increased. This is because, when the data size is small, any model can easily fit it reasonably well. But as the training size increases, it is not so easy to find a model that fits all the data points well. So the training error increases with the size of the data.

On the other hand, the testing error is high when the data size is small. This is because when the size of the data is less, any trivial model can fit the data well but the model won't generalize for the new data points. But as the training size increases, the model gets complex in an attempt to fit the data reasonably well, and it is likely that it fits the testing data also well. So, the training error reduces with the increase in the size of the data.

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 max_depth 1, when the model is fully trained, it suffers from high bias(underfitting) as the model is not complex enough for the data. This can be seen from the high testing error. Throwing more data into the model doesn't seem to help because the model couldn't get complex enough to represent the data accurately.

With max_depth 10, when the model is fully trained, it suffers from high variance i.e. it overfits the data. This was shown in the learning curve while observing the changes to training error. With high enough max_depth, the model got complex, closely followed the data (we can say it pretty much memorized the data) but failed to generalize for new set of 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?

The training and testing error reduces as the model complexity increases. But after certain stage, the model becomes more complex as there is no improvement in the model performance. With a model of higher complexity, it tends to overfit the data leading to minimal training error but high testing error. From the model complexity graph, the optimal max_depth is about 4 (actually seems be the range 4-6 with 4 being more frequent among different runs) where we seem to get a reasonable balance between the model complexity and training error. With this model, the testing error is minimal and hence the model generalizes the dataset well.

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.

Desired model complexity: 4
Predicted price: $21.63

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

Here are the steps I followed to validate my model:

For all the training examples, I predicted the house prices using my model. The results were stored in an array say, predictions. When I plot a scatter plot between actual prices (from the given training samples) and predictions, I expected that I should get a scatter plot which could be generalized by line 'y = x'. This was the case as can be seen below except for the distortions because of the training error.