Predicting Boston Housing Prices

Model Evaluation & Validation Project

Project Description

You want to be the best real estate agent out there. In order to compete with other agents in your area, you decide to use machine learning. You are going to use various statistical analysis tools to build the best model to predict the value of a given house. Your task is to find the best price your client can sell their house at. The best guess from a model is one that best generalizes the data.

For this assignment your client has a house with the following feature set: [11.95, 0.00, 18.100, 0, 0.6590, 5.6090, 90.00, 1.385, 24, 680.0, 20.20, 332.09, 12.13]. To get started, use the example scikit implementation. You will have to modify the code slightly to get the file up and running.

When you are done implementing the code please answer the following questions in a report with the appropriate sections provided. Answers provided in blue.

1) Statistical Analysis and Data Exploration

Statistic to Analyse	Explored Data
Number of data points (houses)	506
Number of features (13 dimensions per)	506
Min Housing Price	5.00 ($5,000)
Maximum Housing Price	50.00 ($50,000)
Mean Housing Price	22.53 ($22,530)
Median Housing Price	21.20 ($21,200)
Standard Deviation of Housing Prices	9.19

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 is best for this project because it gives more weight to larger errors. The Boston housing data is a continuous range of prices and therefore the model requires a regression metric. Therefore, classification metrics like accuracy, precision, and recall are not appropriate. Some regression metrics like R^2 are not a measure of "error", but rather of "score" or how well the model fits the data. Therefore for the purpose of analysing error, R^2 is not appropriate, Although MSE was chosen, here's a list of useful regression error metrics:

  - **Mean Absolute Error**: MAE is the averaging of the list of absolute values of the difference between(the error) actual and predicted values. Taking the absolute value ignores the direction of the error (positive or negative), but keeps its magnitude. Unlike Mean Squared Error, all errors have equal weight when they are averaged.
  - **Median Absolute Error**: MedAE is the median of the list of absolute values of the difference between(the error) actual and predicted values. The median is great for highly skewed distributions and is therefore not useful metric for the Boston housing data.
  - **Mean Squared Error**: MSE is the averaging of the list of squared values of the difference between(the error) actual and predicted values. MSE is used when large errors are undesirable because errors are squared before they are averaged, which gives higher weight to large errors.
  - **Root Mean Squared Error**: RMSE is the square root of MSE and thus keeps the original units of the data. In the case of this project we are looking for trends in the testing and training error over our training data so it's not exactly necessary.

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

  It is necessary to split the housing data into separate training and testing data because each dataset performs unique roles. The training data finds the parameters of the model, while the testing data

verifies the performance of the model after it has been fully trained, mimicking real-world data. Splitting the data makes the model generalized because it evaluates itself on the test set by using separate data than it was trained. If the housing data is not uniquely split, the model will train and test on the exact same data which will will always evaluate to be 100% correct. This is a misleading result because the model overfitted its own data and when unique data is applied the model will likely give unintended results.

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

  Grid search automatically tweaks model parameters to optimize performance of the model through iteration. As it works through multiple parameter tunes it cross-validates to determine which tune gives the best model performance. It is a much faster and less stressful option for finding optimal model parameters rather than do it manually.

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

  Cross validation is a method for reducing overfitting and generalizing our model to unseen inputs. In particular, K-fold cross validation allows us to use all of the data for training and testing even though we split the data into separate training and validation sets.

3) Analyzing Model

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

  As training size increases, initially training error increases and testing error decreases. Eventually both training and testing error plateau.

- 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?

Figure 1. Max Depth 1

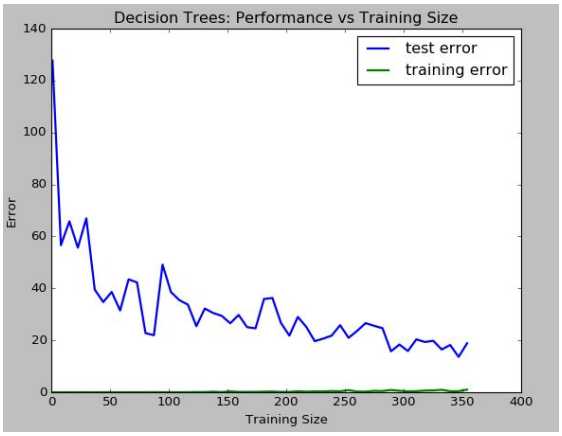

Figure 2. Max Depth 10

The model suffers from high variance/overfitting when it is fully trained because there is a large gap between test and training error on the last learning curve graph. When training error is almost nothing while there is relatively large testing error suggests the model is 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?

Figure 3. Performance vs. Max Depth

The model complexity graph indicates very simple models (low max depth) have a high training and testing error. Although the training and testing errors converge, low complexity models have high bias and are underfitted. Conversely, very complex models (high max depth) have a lower training

and testing error, but with a greater spread between the two which indicates high variance and overfitting.

The model that best generalizes the dataset appears to be about a max depth of 3 because training and testing errors converge with relatively low 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.

```
DecisionTreeRegressor(criterion='mse', max_depth=4, max_features=None,
        max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
        min_weight_fraction_leaf=0.0, presort=False, random_state=None,
        splitter='best')
House: [[11.95, 0.0, 18.1, 0, 0.659, 5.609, 90.0, 1.385, 24, 680.0, 20.2, 332.09, 12.13]]
Prediction: [ 21.62974359]

Other Tests:
Prediction: [ 20.96776316]
Prediction: [ 19.32727273]
Prediction: [ 20.96776316]
Prediction: [ 19.99746835]
Prediction: [ 20.76598639]
```

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

The prediction 21.63 (~$21,630) falls in the range of Boston housing price data, 5.00 ($5,000) to 50.00 ($50,000), is very close to the median 21.20 (~$21,200) and mean 22.53 (~$22,530), and is well within 1 standard deviation of the data. Multiple trials show that the prediction is repeatable within a range from about 18.00 to 22.00. Based on this analysis it's a valid model.