boston_housing_submission

June 18, 2016

1 Machine Learning Engineer Nanodegree

1.1 Project 1: Predicting Boston Housing Prices

Submitted by: CHIRAG JHAMB

```
In [16]: import numpy as np
         import pandas as pd
         import visuals as vs # Supplementary code
         from sklearn.cross_validation import ShuffleSplit

         # Pretty display for notebooks
         %matplotlib inline

         # Load the Boston housing dataset
         data = pd.read_csv('housing.csv')
         prices = data['MDEV']
         features = data.drop('MDEV', axis = 1)

         # Success
         print "Boston housing dataset has {} data points with {} variables each."
```

```
Boston housing dataset has 489 data points with 4 variables each.
```

TO DO- Calculate the minimum, maximum, mean, median, and standard deviation of 'MDEV', which is stored in prices.

```
In [17]: # TODO: Minimum price of the data
         minimum_price = min(prices)

         # TODO: Maximum price of the data
         maximum_price = max(prices)

         # TODO: Mean price of the data
         mean_price = np.mean(prices)

         # TODO: Median price of the data
         median_price = np.median(prices)

         # TODO: Standard deviation of prices of the data
         std_price = np.std(prices)
```

```
In [18]: # Show the calculated statistics
         print "Statistics for Boston housing dataset:\n"
         print "Minimum price: ${:,.2f}".format(minimum_price)
         print "Maximum price: ${:,.2f}".format(maximum_price)
         print "Mean price: ${:,.2f}".format(mean_price)
         print "Median price ${:,.2f}".format(median_price)
         print "Standard deviation of prices: ${:,.2f}".format(std_price)
```

```
Statistics for Boston housing dataset:

Minimum price: $105,000.00
Maximum price: $1,024,800.00
Mean price: $454,342.94
Median price $438,900.00
Standard deviation of prices: $165,171.13
```

1.2 Question 1 - Feature Observation

As a reminder, we are using three features from the Boston housing dataset: `'RM'`, `'LSTAT'`, and `'PTRATIO'`. For each data point (neighborhood): - `'RM'` is the average number of rooms among homes in the neighborhood. - `'LSTAT'` is the percentage of all Boston homeowners who have a greater net worth than homeowners in the neighborhood. - `'PTRATIO'` is the ratio of students to teachers in primary and secondary schools in the neighborhood.

*Using your intuition, for each of the three features above, do you think that an increase in the value of that feature would lead to an **increase** in the value of `'MDEV'` or a **decrease** in the value of `'MDEV'`? Justify your answer for each.*
Hint: Would you expect a home that has an `'RM'` value of 6 be worth more or less than a home that has an `'RM'` value of 7? * Answer:** For each of these 3, the impact on the `'MDEV'` value will be the following: - Increase in `'RM'`, that is the number of rooms would certainly *increase* the price of the house. Chances are the size of the house will also increase with `'RM'`. - `'LSTAT'`, the percentage of all Boston homeowners who have a greater net worth than homeowners in the neighborhood. People with higher worth would probably prefer living in a better house having higher price. Thus, increase in the value of `'LSTAT'` would *increase* the value of `'RM'`. - In case of `'PTRATIO'`, though the factor of correlation might not be as high as the other features, but as per intuition, more the number of students in a school means less attention to each student, which is usually less preferred by people. Increase in the `'PTRATIO'` would result in a *decrease* in the value of `'RM'`.

```
In [19]: # TODO: Import 'r2_score'
         import sklearn.metrics as sm
         def performance_metric(y_true, y_predict):
             """ Calculates and returns the performance score between
                 true and predicted values based on the metric chosen. """
```

```
        # TODO: Calculate the performance score between 'y_true' and 'y_predic
        score = sm.r2_score(y_true, y_predict)

        # Return the score
        return score
```

1.2.1 Question 2 - Goodness of Fit

Assume that a dataset contains five data points and a model made the following predictions for the target variable:

| True Value | Prediction | | :————-: | :———: | | 3.0 | 2.5 | | -0.5 | 0.0 | | 2.0 | 2.1 | | 7.0 | 7.8 | | 4.2 | 5.3 | *Would you consider this model to have successfully captured the variation of the target variable? Why or why not?*

Run the code cell below to use the `performance_metric` function and calculate this model's coefficient of determination.

```
In [20]: # Calculate the performance of this model
         score = performance_metric([3, -0.5, 2, 7, 4.2], [2.5, 0.0, 2.1, 7.8, 5.3]
         print "Model has a coefficient of determination, R^2, of {:.3f}.".format(s
```

```
Model has a coefficient of determination, R^2, of 0.923.
```

Answer: Since the coefficient of determination has a value of 0.923, this means the model is very efficient! **Conclusion**: Model has successfully captured the variation of the target variable by 92.3%.

```
In [21]: # TODO: Import 'train_test_split'
         from sklearn.cross_validation import train_test_split
         # TODO: Shuffle and split the data into training and testing subsets
         X_train, X_test, y_train, y_test = train_test_split(features, prices, test

         # Success
         print "Training and testing split was successful."
```

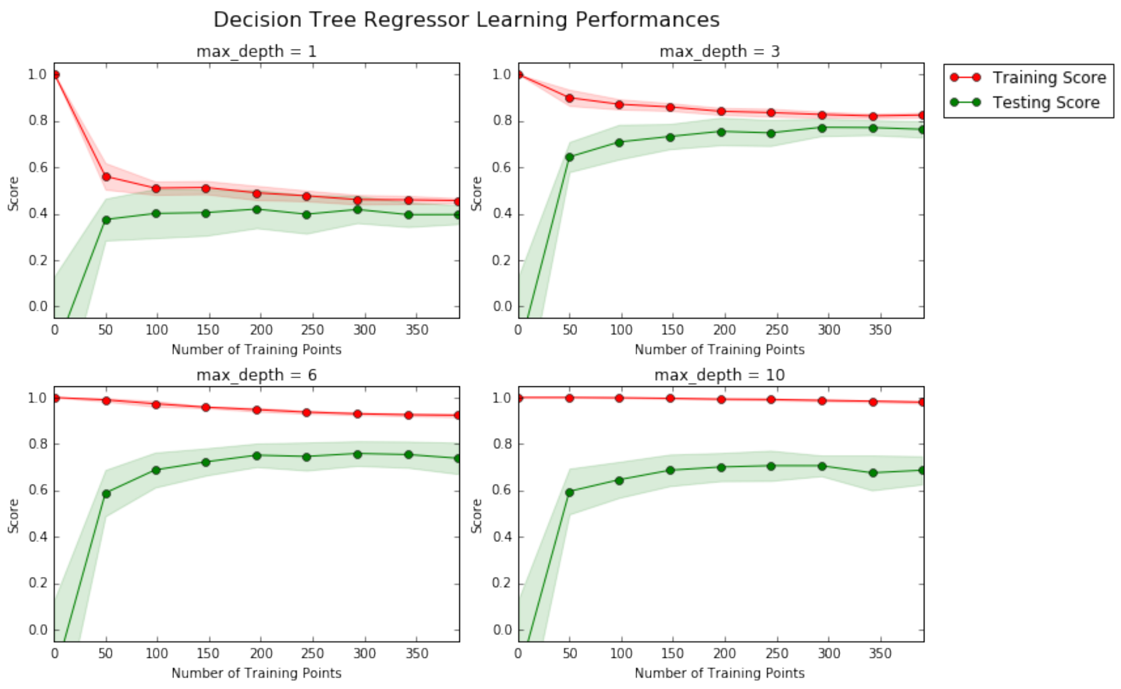
```
Training and testing split was successful.
```

1.2.2 Question 3 - Training and Testing

What is the benefit to splitting a dataset into some ratio of training and testing subsets for a learning algorithm?

Hint: What could go wrong with not having a way to test your model? * Answer: ** A test set helps in building a validation model. After training a model, testing it's accuracy is equally important. Before passing any real data having unkown results, it's better to estimate how accurately the learning algorithm would be able to predict the results. A test set helps in validating the corectness of the learning algorithm. If there is no way to test the model then the results given by the model cannot be trusted.

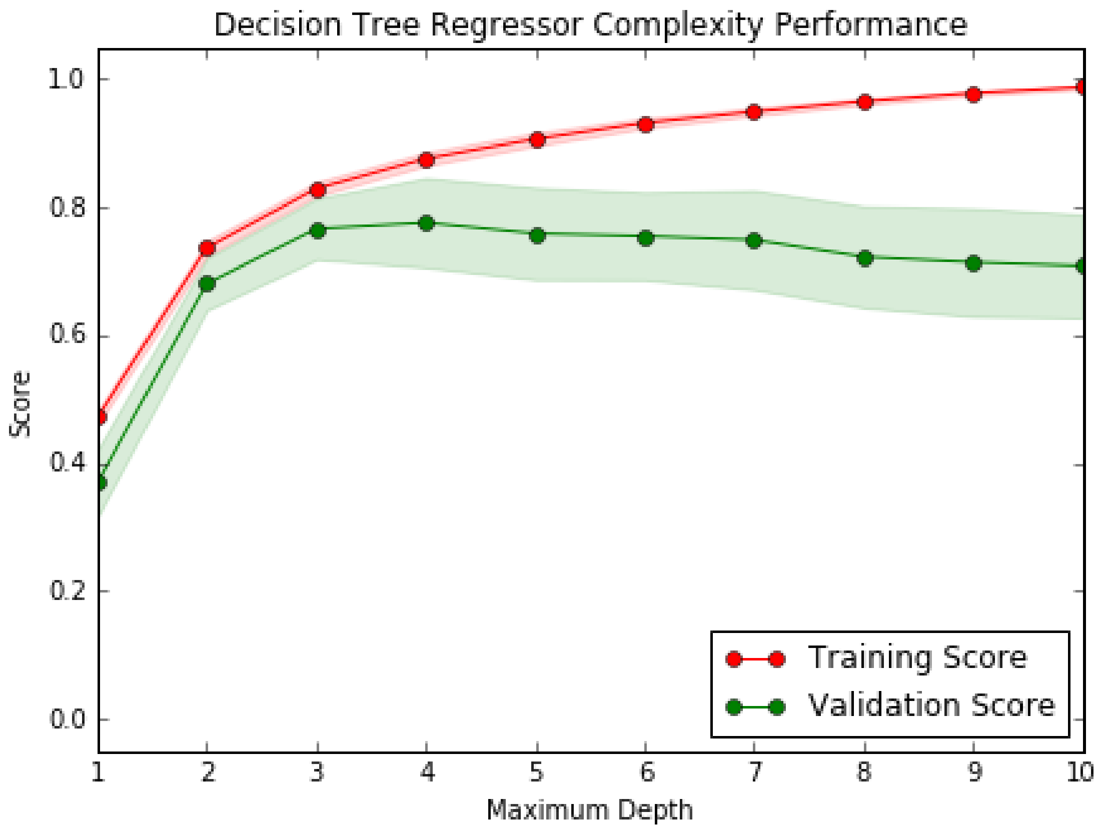
```
In [22]: # Produce learning curves for varying training set sizes and maximum depth
         vs.ModelLearning(features, prices)
```



1.2.3 Question 4 - Learning the Data

Choose one of the graphs above and state the maximum depth for the model. What happens to the score of the training curve as more training points are added? What about the testing curve? Would having more training points benefit the model?
Hint: Are the learning curves converging to particular scores? * Answer: ** Graph with `"max_depth=3"` represents the most precise model. The score of training set is decreasing with a very slow rate as the number of training points are being increased. On the other hand, testing curve is slightly increasing with the number of training points. Gradually, both the curves will converge as the training points increase. Thus, adding more training points for this model will not make much of a difference.

```
In [23]: vs.ModelComplexity(X_train, y_train)
```



1.2.4 Question 5 - Bias-Variance Tradeoff

When the model is trained with a maximum depth of 1, does the model suffer from high bias or from high variance? How about when the model is trained with a maximum depth of 10? What visual cues in the graph justify your conclusions?
Hint: How do you know when a model is suffering from high bias or high variance? * Answer: **The model suffers from high bias when trained with the maximum depth of 1.** Reason:** Lower score of training set means high error, which is an indicatore of high bias.

The model trained with the maximum depth of 10 suffers from high variance. **Reason:** The validation score is relatively less than the training score, which concludes a high error rate on the test set, an indicator of high variance.

1.2.5 Question 6 - Best-Guess Optimal Model

Which maximum depth do you think results in a model that best generalizes to unseen data? What intuition lead you to this answer? * Answer:** The validation score at maximum depth 4 is the highest and thus the value of maximum depth with highest score and least uncertainity, that is maximum depth=4, should be the best model.

1.2.6 Question 7 - Grid Search

What is the grid search technique and how it can be applied to optimize a learning algorithm? * Answer:** In grid search technique, all combinations of parameters are passed through the method to find the right combinations of parameters acting as the best parameters for the `"estimator"`, also known as hyperparameters. Grid search can work through many combinations to find the right one. To optimise a learning algorithm, the grid search method takes the algorithm and the possible hyperparameters as an input and returns the best set of parameters for prediction.

1.2.7 Question 8 - Cross-Validation

What is the k-fold cross-validation training technique? What benefit does this technique provide for grid search when optimizing a model?
Hint: Much like the reasoning behind having a testing set, what could go wrong with using grid search without a cross-validated set? * Answer:** In k-fold cross-validation, data is randomly partitioned into k equal subsets of data. One subset is used as the validation data for testing the model, and the remaining k − 1 subsets of data are used as training data. Grid search works through many combinations to find the right set of parameters over a single dataset. K- fold cross validation provides more number of combination of datasets so that the grid search works through more data to find the right combination. If cross validation is not used, the grid search might train certain set of data and apply the model on a different subset having completely different model. Thus application of grid search over different training subsets is preferred.

```
In [36]: # TODO: Import 'make_scorer', 'DecisionTreeRegressor', and 'GridSearchCV'
         from sklearn.metrics import make_scorer
         from sklearn.tree import DecisionTreeRegressor
         from sklearn.grid_search import GridSearchCV

         def fit_model(X, y):
             """ Performs grid search over the 'max_depth' parameter for a
                 decision tree regressor trained on the input data [X, y]. """

             # Create cross-validation sets from the training data
             cv_sets = ShuffleSplit(X.shape[0], n_iter = 10, test_size = 0.20, ranc

             # TODO: Create a decision tree regressor object
             regressor = DecisionTreeRegressor(random_state=0)

             # TODO: Create a dictionary for the parameter 'max_depth' with a range
             params = {'max_depth': list(range(1,10))}


             # TODO: Transform 'performance_metric' into a scoring function using
             scoring_fnc = make_scorer(performance_metric)

             # TODO: Create the grid search object
             grid = GridSearchCV(regressor, params, scoring=scoring_fnc, cv=cv_sets

             # Fit the grid search object to the data to compute the optimal model
```

```
            grid = grid.fit(X, y)

            # Return the optimal model after fitting the data
            return grid.best_estimator_
```

1.2.8 Question 9 - Optimal Model

*What maximum depth does the optimal model have? How does this result compare to your guess in **Question 6**?*

Run the code block below to fit the decision tree regressor to the training data and produce an optimal model.

```
In [38]: # Fit the training data to the model using grid search
         reg = fit_model(X_train, y_train)

         # Produce the value for 'max_depth'
         print "Parameter 'max_depth' is {} for the optimal model.".format(reg.get_
```

```
Parameter 'max_depth' is 4 for the optimal model.
```

Answer: The maximum depth is 4, just as predicted in answer 6.

1.2.9 Question 10 - Predicting Selling Prices

Imagine that you were a real estate agent in the Boston area looking to use this model to help price homes owned by your clients that they wish to sell. You have collected the following information from three of your clients:

| Feature | Client 1 | Client 2 | Client 3 | | :—: | :—: | :—: | :—: | | Total number of rooms in home | 5 rooms | 4 rooms | 8 rooms | | Household net worth (income) | Top 34th percent | Bottom 45th percent | Top 7th percent | | Student-teacher ratio of nearby schools | 15-to-1 | 22-to-1 | 12-to-1 | *What price would you recommend each client sell his/her home at? Do these prices seem reasonable given the values for the respective features?*
Hint: Use the statistics you calculated in the **Data Exploration** section to help justify your response.

Run the code block below to have your optimized model make predictions for each client's home.

```
In [39]: # Produce a matrix for client data
         client_data = [[5, 34, 15], # Client 1
                        [4, 55, 22], # Client 2
                        [8, 7, 12]]  # Client 3

         # Show predictions
         for i, price in enumerate(reg.predict(client_data)):
             print "Predicted selling price for Client {}'s home: ${:,.2f}".format
```

```
Predicted selling price for Client 1's home: $407,480.77
Predicted selling price for Client 2's home: $225,642.86
Predicted selling price for Client 3's home: $976,150.00
```

Answer: These prices are somewhat reasonable. Client 3 is recommended the highest price since he/she requires the highest number of rooms, has the highest household income. Client 2 gets the least price because of just 4 rooms, high student-teacher ratio and low household income. Client 1 gets a fair price since all the values lie between the highest(Client 3) and the lowest(Client 2).

```
In [40]: vs.PredictTrials(features, prices, fit_model, client_data)
```

```
Trial 1: $324,240.00
Trial 2: $302,400.00
Trial 3: $346,500.00
Trial 4: $420,622.22
Trial 5: $302,400.00
Trial 6: $411,931.58
Trial 7: $344,750.00
Trial 8: $407,232.00
Trial 9: $352,315.38
Trial 10: $316,890.00

Range in prices: $118,222.22
```

1.2.10 Question 11 - Applicability

In a few sentences, discuss whether the constructed model should or should not be used in a real-world setting.

Hint: Some questions to answering: - *How relevant today is data that was collected from 1978? - Are the features present in the data sufficient to describe a home? - Is the model robust enough to make consistent predictions? - Would data collected in an urban city like Boston be applicable in a rural city?*

Answer: - Since the economy has changed a lot since 1978, the data is not of much relevance. There must be a large number of new houses, new nearby places and schools etc affecting the price of each house and thus diminishing the relevance of the features given in the dataset. - There can be a few more features since people often check for a lot more things before buying a house. For example- the nearby supermarkets, number natural disasters in the past, parking space provided, distance from shopping complexes etc. would affect the prices as well. - The model has a high range as seen above, and a total of 506 instances(as per UCI Machine Learning Repository). The model can be considered robust. Although we might want to take more features for better predictions. - People in urban areas have completely different preferences over people in rural areas. The model would be completely different for a rural city and thus this data cannot be used.