Machine Learning Engineer Nanodegree

Model Evaluation & Validation

Project 1: Predicting Boston Housing Prices

Welcome to the first project of the Machine Learning Engineer Nanodegree! In this notebook, some template code has already been provided for you, and you will need to implement additional functionality to successfully complete this project. You will not need to modify the included code beyond what is requested. Sections that begin with **'Implementation'** in the header indicate that the following block of code will require additional functionality which you must provide. Instructions will be provided for each section and the specifics of the implementation are marked in the code block with a 'TODO' statement. Please be sure to read the instructions carefully!

In addition to implementing code, there will be questions that you must answer which relate to the project and your implementation. Each section where you will answer a question is preceded by a **'Question X'** header. Carefully read each question and provide thorough answers in the following text boxes that begin with **'Answer:'**. Your project submission will be evaluated based on your answers to each of the questions and the implementation you provide.

> **Note:** Code and Markdown cells can be executed using the **Shift + Enter** keyboard shortcut. In addition, Markdown cells can be edited by typically double-clicking the cell to enter edit mode.

Getting Started

In this project, you will evaluate the performance and predictive power of a model that has been trained and tested on data collected from homes in suburbs of Boston, Massachusetts. A model trained on this data that is seen as a *good fit* could then be used to make certain predictions about a home — in particular, its monetary value. This model would prove to be invaluable for someone like a real estate agent who could make use of such information on a daily basis.

The dataset for this project originates from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Housing). The Boston housing data was collected in 1978 and each of the 506 entries represent aggregated data about 14 features for homes from various suburbs in Boston, Massachusetts. For the purposes of this project, the following preprocessing steps have been made to the dataset:

- 16 data points have an `'MEDV'` value of 50.0. These data points likely contain **missing or censored values** and have been removed.
- 1 data point has an `'RM'` value of 8.78. This data point can be considered an **outlier** and has been removed.
- The features `'RM'`, `'LSTAT'`, `'PTRATIO'`, and `'MEDV'` are essential. The remaining **non-relevant features** have been excluded.

- The feature `'MEDV'` has been **multiplicatively scaled** to account for 35 years of market inflation.

Run the code cell below to load the Boston housing dataset, along with a few of the necessary Python libraries required for this project. You will know the dataset loaded successfully if the size of the dataset is reported.

In [2]:
```
# Import libraries necessary for this project
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['MEDV']
features = data.drop('MEDV', axis = 1)

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

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

Data Exploration

In this first section of this project, you will make a cursory investigation about the Boston housing data and provide your observations. Familiarizing yourself with the data through an explorative process is a fundamental practice to help you better understand and justify your results.

Since the main goal of this project is to construct a working model which has the capability of predicting the value of houses, we will need to separate the dataset into **features** and the **target variable**. The **features**, `'RM'`, `'LSTAT'`, and `'PTRATIO'`, give us quantitative information about each data point. The **target variable**, `'MEDV'`, will be the variable we seek to predict. These are stored in `features` and `prices`, respectively.

Implementation: Calculate Statistics

For your very first coding implementation, you will calculate descriptive statistics about the Boston housing prices. Since `numpy` has already been imported for you, use this library to perform the necessary calculations. These statistics will be extremely important later on to analyze various prediction results from the constructed model.

In the code cell below, you will need to implement the following:

- Calculate the minimum, maximum, mean, median, and standard deviation of `'MEDV'`, which is stored in `prices`.
    - Store each calculation in their respective variable.

In [3]:
```
# TODO: Minimum price of the data
minimum_price = np.min(prices)

# TODO: Maximum price of the data
maximum_price = np.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)

# 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
```

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 homeowners in the neighborhood considered "lower class" (working poor).
- `'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 `'MEDV'` or a **decrease** in the value of `'MEDV'`? 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:  RM: An increase in the value of RM feature means increase in number
of rooms. Given that these houses
             are among homes in the neighborhood they will command greater
prices for the increase space.
```

```
        LSTAT: An increase in homeowners with lower income than average will
result in smaller and houses not in peak
               condition to be purchased. This will ultimately drive the Median
prices down for houses in the                        neighborhood.
        PTRATIO:In most cases the income of the neighborhood determines the
taxes paid. Higher income results in                       higher taxes levied.
The higher taxes results in more resources available for schools. This will lead
               to a lower student to teacher ratio as the school district will
be able to employ more teachers.Given
               this analysis of PTRATIO a lower student to teacher ratio will
increase house prices.
```

Developing a Model

In this second section of the project, you will develop the tools and techniques necessary for a model to make a prediction. Being able to make accurate evaluations of each model's performance through the use of these tools and techniques helps to greatly reinforce the confidence in your predictions.

Implementation: Define a Performance Metric

It is difficult to measure the quality of a given model without quantifying its performance over training and testing. This is typically done using some type of performance metric, whether it is through calculating some type of error, the goodness of fit, or some other useful measurement. For this project, you will be calculating the *coefficient of determination* (http://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination), R^2, to quantify your model's performance. The coefficient of determination for a model is a useful statistic in regression analysis, as it often describes how "good" that model is at making predictions.

The values for R^2 range from 0 to 1, which captures the percentage of squared correlation between the predicted and actual values of the **target variable**. A model with an R^2 of 0 always fails to predict the target variable, whereas a model with an R^2 of 1 perfectly predicts the target variable. Any value between 0 and 1 indicates what percentage of the target variable, using this model, can be explained by the **features**. *A model can be given a negative R^2 as well, which indicates that the model is no better than one that naively predicts the mean of the target variable.*

For the `performance_metric` function in the code cell below, you will need to implement the following:

- Use `r2_score` from `sklearn.metrics` to perform a performance calculation between `y_true` and `y_predict`.
- Assign the performance score to the `score` variable.

In [4]:
```
# TODO: Import 'r2_score'
from sklearn.metrics import r2_score

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_predict'
    score = r2_score(y_true, y_predict)

    # Return the score

    return score
```

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 [5]:
```
# 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(score)
```

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

```
**Answer:Model has a coefficient of determination of :0.923

        With a coefficient of Determinaiton of 0.923 it would seem the
regression model approximates the target
        variable well; this given the fact that the range of values are from 0
to 1 (1 represents a perfect fit).

        There is however concern about the fact that working with five variables
drawn from a larger dataset the
        results maybe misleading, so while this model appears to capture the
variation fairly well it could be
        misleading.
```

Implementation: Shuffle and Split Data

Your next implementation requires that you take the Boston housing dataset and split the data into training and testing subsets. Typically, the data is also shuffled into a random order when creating the training and testing subsets to remove any bias in the ordering of the dataset.

For the code cell below, you will need to implement the following:

- Use `train_test_split` from `sklearn.cross_validation` to shuffle and split the `features` and `prices` data into training and testing sets.
  - Split the data into 80% training and 20% testing.
  - Set the `random_state` for `train_test_split` to a value of your choice. This ensures results are consistent.
- Assign the train and testing splits to `X_train`, `X_test`, `y_train`, and `y_test`.

In [6]:
```
# TODO: Import 'train_test_split'
from sklearn import cross_validation

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 = cross_validation.train_test_split(features,pri

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

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

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: The benefit of splitting the data into some ratio of training and testing subsets of a learning algorithm is to ensure our predictions will be relatively close to the outcomes we get to see. It also should provide a good idea of how large our prediction errors are when they are incorrect.

The training Subset is fed clean data from the dataset then the knowledge or information is extracted, in the testing stage we put this information to use. Here is where we verify if the outcomes are accurately predicted and to what level the outcomes vary if they are not accurate. Both stages are thus very beneficial. **

Analyzing Model Performance

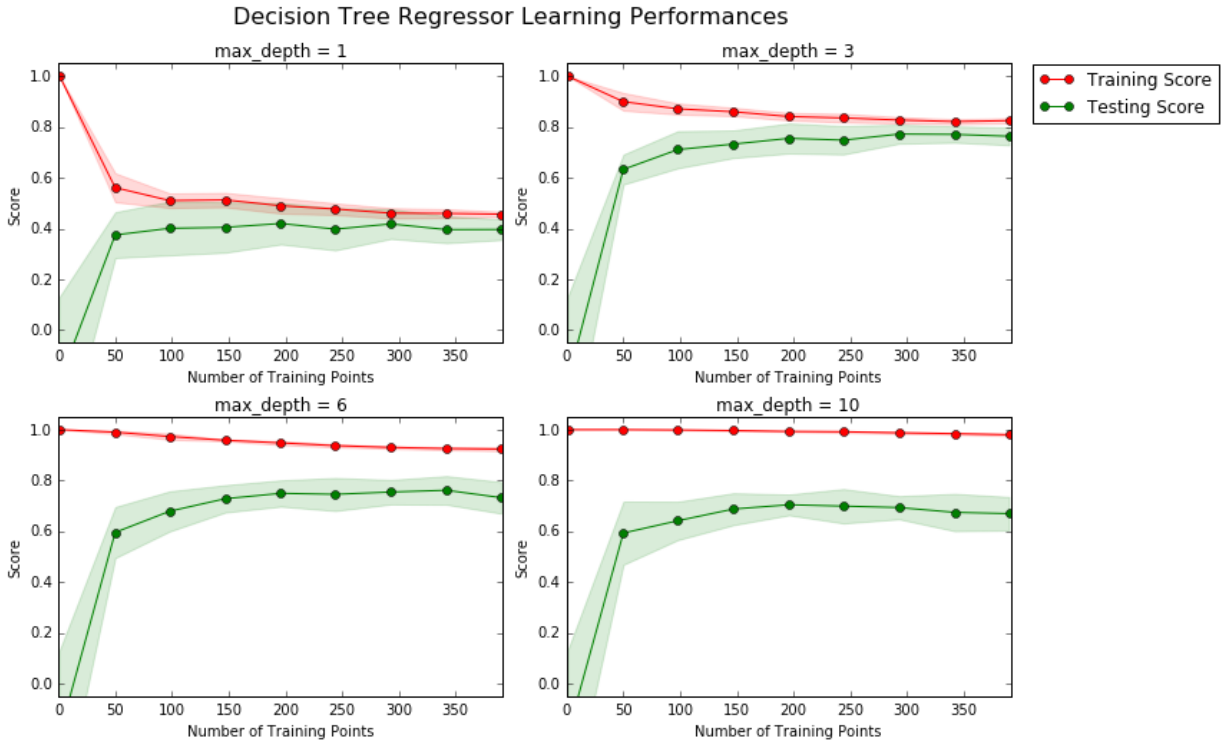
In this third section of the project, you'll take a look at several models' learning and testing performances on various subsets of training data. Additionally, you'll investigate one particular algorithm with an increasing `'max_depth'` parameter on the full training set to observe how model complexity affects performance. Graphing your model's performance based on varying criteria can be beneficial in the analysis process, such as visualizing behavior that may not have been apparent from the results alone.

Learning Curves

The following code cell produces four graphs for a decision tree model with different maximum depths. Each graph visualizes the learning curves of the model for both training and testing as the size of the training set is increased. Note that the shaded region of a learning curve denotes the uncertainty of that curve (measured as the standard deviation). The model is scored on both the training and testing sets using R^2, the coefficient of determination.

Run the code cell below and use these graphs to answer the following question.

In [7]:
```
# Produce learning curves for varying training set sizes and maximum depths
vs.ModelLearning(features, prices)
```



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: I will choose max_depth = 1
```

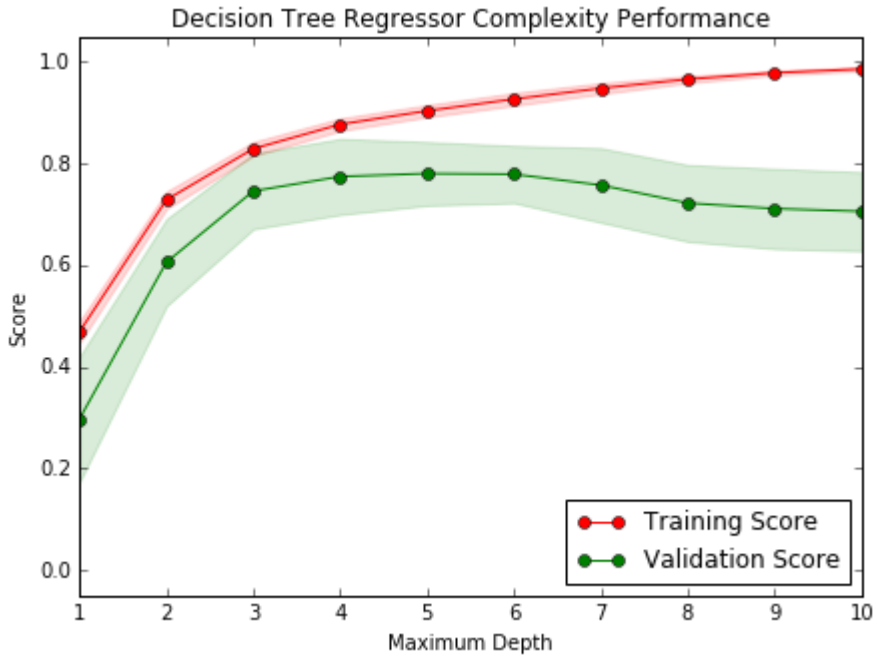
```
          In depth look at effect of increasing the number of training points on
testing and training curve:

          Training curve:
          With very few training points the score is very high in fact it is
nearly at 1. This is mostly due to
          overfitting, here for a small dataset using a few trainig points the
model may be useful. As more training
          points are added the score of the training curve starts decreasing
which could mean the results more
          accounts for the variance inherent in modelling data that has a level
of randomness. .

          Testing Curve:
          With a few training points the testing curve is close to zero. As more
training points are added it
          increases drastically and converges close to the value of the testing
curve. Given the size of the datasets
          with a few training points it is obvious the results will reflect a
high bias as it will not be indicative
          of what the results of training on more members of the dataset will
produce.

          As more training points are added the training and testing curve begin
to converge this means adding more
          training points will not benefit the model.   **
```

Complexity Curves

The following code cell produces a graph for a decision tree model that has been trained and validated on the training data using different maximum depths. The graph produces two complexity curves — one for training and one for validation. Similar to the **learning curves**, the shaded regions of both the complexity curves denote the uncertainty in those curves, and the model is scored on both the training and validation sets using the `performance_metric` function.

Run the code cell below and use this graph to answer the following two questions.

In [8]:
```
vs.ModelComplexity(X_train, y_train)
```



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:   maximimum depth of 1:With a low training and validation score the
model will underfit the data, this usually          happens with maximum depth
of 1. The result is a high bias, here the model pays little attention to the
data          and the result is oversimplified. With a max_depth of 1 the model
faces difficulty of accounting for the              variance that will likely
exist.

         max depth 10:With a maximum depth of 10 the model will tend to have a
low bias and high variance. The result          is usually overfitting this
usually occurs with a high coefficent of determination which tend to result with
         a high maximum depth. The validation score peaks at around max-depth
of 4 and start decreasing          as          the values increase the results
could be explained as a consequence of increase variance. With a max_depth
      of 10 the validation score is higher than with a low max_depth but lower
than the training score with a              max_depth of 10.

         The training and validation score are comparatively low in relation to
other areas of the graph. Also with
         max_depth 4 and below, the increase in both training and validation
score are very similar.

         Visual cues from the graph shows that as the Maximum Depth increases
the score increases form a little above
         0.4 to close to 1.0.
```

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: Somewhere between between 3 and 5 Max_depth the validation score is at its highest. The value at 4 would more represent where the model best generalizes to the unseen data. Validation score tends to be more accurate as it is an average of running different iterations of the training data, hence it will best represent a clearer value for the variance.  **
```

Evaluating Model Performance

In this final section of the project, you will construct a model and make a prediction on the client's feature set using an optimized model from `fit_model`.

Question 7 - Grid Search

What is the grid search technique and how it can be applied to optimize a learning algorithm?

```
**Answer: Grid search technique is one of the methods that can be used to efficiently search for the optimal tuning parameters for a machine learning model to maximize its performance.To optimize a learning model the Grid search technique runs exhaustive searching via cross-validation on the training set. One could select finite values from two  pair and then  run a SVM with each pair from the two sets then evaluate their performance on a held out validation.  It then outputs the settings that achieve the highest score in the validation procedure.  **
```

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: K-fold cross-validation training technique involves splitting the data set into equal portions 'k' times. Despite the fact that their exist the train_test_split method, it was proven that using the k-fold cross-validation technique was many times more effective. In the train_test_split method the data is divided in some proportion into a training set, used to train the model and a testing set used to evaluate the model. By using the k-fold cross-validation we increase the number of evaluation from 1 to k which results in a more reliable evaluation.
```

```
The process of doing a k-fold evaluation works like this:
 1.We first divide the data into k-folds(equal proportions k times)
 2.We then iterate through k portion('bucket') individually using the current
portion('bucket') for validation and the remaining k-1 buckets for training.
 3.We then average all the validation from scores from all the iterations; the
reultant is a more reliable result than if we had used the train_test_split
method.

By using k-fold cross-validation we have a larger validation set size and hence
we are better able to optimize the model.  **
```

Implementation: Fitting a Model

Your final implementation requires that you bring everything together and train a model using the **decision tree algorithm**. To ensure that you are producing an optimized model, you will train the model using the grid search technique to optimize the `'max_depth'` parameter for the decision tree. The `'max_depth'` parameter can be thought of as how many questions the decision tree algorithm is allowed to ask about the data before making a prediction. Decision trees are part of a class of algorithms called *supervised learning algorithms*.

For the `fit_model` function in the code cell below, you will need to implement the following:

- Use `DecisionTreeRegressor` (http://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html) from `sklearn.tree` to create a decision tree regressor object.
  - Assign this object to the `'regressor'` variable.
- Create a dictionary for `'max_depth'` with the values from 1 to 10, and assign this to the `'params'` variable.
- Use `make_scorer` (http://scikit-learn.org/stable/modules/generated/sklearn.metrics.make_scorer.html) from `sklearn.metrics` to create a scoring function object.
  - Pass the `performance_metric` function as a parameter to the object.
  - Assign this scoring function to the `'scoring_fnc'` variable.
- Use `GridSearchCV` (http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html) from `sklearn.grid_search` to create a grid search object.
  - Pass the variables `'regressor'`, `'params'`, `'scoring_fnc'`, and `'cv_sets'` as parameters to the object.
  - Assign the `GridSearchCV` object to the `'grid'` variable.

In [9]:
```
# TODO: Import 'make_scorer', 'DecisionTreeRegressor', and 'GridSearchCV'
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import make_scorer
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, random_stat

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

    # TODO: Create a dictionary for the parameter 'max_depth' with a range from 1
    params = {'max_depth': range(1,11)}

    # TODO: Transform 'performance_metric' into a scoring function using 'make_sc
    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_
```

Making Predictions

Once a model has been trained on a given set of data, it can now be used to make predictions on new sets of input data. In the case of a *decision tree regressor*, the model has learned *what the best questions to ask about the input data are*, and can respond with a prediction for the **target variable**. You can use these predictions to gain information about data where the value of the target variable is unknown — such as data the model was not trained on.

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 [10]:
```
# 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_params(
```

```
Parameter 'max_depth' is 6 for the optimal model.
```

**Answer: The Maximum depth of the Optimal model is 5. In my answer six i deduced the max_depth was between 3 to 5 settling at 4. This is a bit more definite, I was a little way off.

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
Neighborhood poverty level (as %)	17%	32%	3%
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 [11]:
```
# Produce a matrix for client data
client_data = [[5, 17, 15], # Client 1
               [4, 32, 22], # Client 2
               [8, 3, 12]]  # Client 3

# Show predictions
for i, price in enumerate(reg.predict(client_data)):
    print "Predicted selling price for Client {}'s home: ${:,.2f}".format(i+1, pr
```

```
Predicted selling price for Client 1's home: $424,935.00
Predicted selling price for Client 2's home: $284,200.00
Predicted selling price for Client 3's home: $933,975.00
```

```
**Answer:Predicted selling price for client's 1 through clients's 3 are
$419,700.00, $287,100.00 and $927,500.00 respectively.

Using the statistics from Data Exploration I will make a case for whether the
prices seem reasonable.

Client 1: Given the Median price is : $438,900.00, the predicted price of
$419,700.00 may be reasonable, the standard
```

```
          deviation is much lower then the dataset value of $165,171.13. If
their is a high demand in that area the              client could hold out for
the mean price or a bit more as the school to teacher ratio is relatively good
at           15-1, the number of rooms at 5 might be a plus factor too,  the
drawback to not holding out could be the              poverty level which is at
17%.

Client 2: Of the three homes client's 2 home has the least favorable features
affecting its attractiveness, the poverty         level is 32%, teacher to pupil
ratio is 22-1 and it only has 4 rooms. The predicted price is much lower than
        the median price but it is within the standard deviation of prices in
the neighborhood. Given that fact it may         still be fairly priced. If it
is high demand period they could hold out for a bit more , however if demand is
        low then this price maybe reasonable.

Client 3 : Client 3 has the most favorable features going for his/her home.
There are 8 rooms, low poverty and the best            teacher to pupil ratio of
12-1. Its price however in comparison to the maximum price maybe a bit low. The
             poverty level of 3% could be indicative of the fact that the house
is situated in an affluent neighborhood            which means its value could
be in line with the maximum price value. its more than a $100k less the maximum
           price if the demand is high then it could be undervalued however if
demand is tepid it could be reasonable            priced. Given the good
features I would say they could ask for more. **
```

Sensitivity

An optimal model is not necessarily a robust model. Sometimes, a model is either too complex or too simple to sufficiently generalize to new data. Sometimes, a model could use a learning algorithm that is not appropriate for the structure of the data given. Other times, the data itself could be too noisy or contain too few samples to allow a model to adequately capture the target variable — i.e., the model is underfitted. Run the code cell below to run the `fit_model` function ten times with different training and testing sets to see how the prediction for a specific client changes with the data it's trained on.

In [12]:
```
vs.PredictTrials(features, prices, fit_model, client_data)
```

```
Trial 1: $391,183.33
Trial 2: $419,700.00
Trial 3: $415,800.00
Trial 4: $420,622.22
Trial 5: $418,377.27
Trial 6: $411,931.58
Trial 7: $399,663.16
Trial 8: $407,232.00
Trial 9: $351,577.61
Trial 10: $413,700.00

Range in prices: $69,044.61
```

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:The Dataset contain data whose determining factors may have changed over the years from what they were in 1978. Particularly the LSTAT and PTRATIO data; the LSTAT represented the lower income of the neighborhood, during the course of over 30 years the demographic may have change. The income change will affect the median value of houses. The PTRATIO represented teacher to student ratio; this also maybe different as the public school system have not remain constant over the period of thirty years. With these data being different the dataset maybe outdated and it may not hold the same relevance as it did in 1978.

We used a select few relevant features from the total presented in fact we chose three from the fourteen.It maybe possible other features in the data would have given a more precise description when combibele other features in the data would have given a more precise description when combined with the three we focused on.

The results of the model showed from the sensitivity data that the variance from the mean price of the houses which was $454,000 was a bit high. We obtained trial values from low of 350K this is a a bit of a gap from the maean value. We might conclude that the model is not robust enough to make a consistent prediction.

It is highly unlikely that data collected in Boston be applicable to a rural city. The factors that affect how attractive a home is to a buyer are a bit different depend on locale. The PTRATIO, Median value, LSTAT maybe be totally different and so the results a dataset will produce due to the varying values of the features will I would say produce differing outcomes.