boston_housing

September 12, 2018

1 Machine Learning Engineer Nanodegree

1.1 Model Evaluation & Validation

1.2 Project: 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.

1.3 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. 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 [1]: # Import libraries necessary for this project
        import numpy as np
        import pandas as pd
        from sklearn.cross_validation import ShuffleSplit
        from IPython.display import display

        # Import supplementary visualizations code visuals.py
        import visuals as vs

        # 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)

        display(data.head())

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

```
/opt/conda/lib/python3.6/site-packages/sklearn/cross_validation.py:41: DeprecationWarning: This
  "This module will be removed in 0.20.", DeprecationWarning)
/opt/conda/lib/python3.6/site-packages/sklearn/learning_curve.py:22: DeprecationWarning: This mo
  DeprecationWarning)
```

	RM	LSTAT	PTRATIO	MEDV
0	6.575	4.98	15.3	504000.0
1	6.421	9.14	17.8	453600.0
2	7.185	4.03	17.8	728700.0
3	6.998	2.94	18.7	701400.0
4	7.147	5.33	18.7	760200.0

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

1.4 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.

1.4.1 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 [2]:
```
# 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: ${}".format(minimum_price))
print("Maximum price: ${}".format(maximum_price))
print("Mean price: ${}".format(mean_price))
print("Median price ${}".format(median_price))
print("Standard deviation of prices: ${}".format(std_price))
```

```
Statistics for Boston housing dataset:

Minimum price: $105000.0
Maximum price: $1024800.0
Mean price: $454342.9447852761
Median price $438900.0
Standard deviation of prices: $165171.13154429474
```

1.4.2 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 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: This problem can phrased using examples like below.
* Would you expect a home that has an `'RM'` value(number of rooms) of 6 be worth more or less than a home that has an `'RM'` value of 7? * Would you expect a neighborhood that has an `'LSTAT'` value(percent of lower class workers) of 15 have home prices be worth more or less than a neighborhood that has an `'LSTAT'` value of 20? * Would you expect a neighborhood that has an `'PTRATIO'` value(ratio of students to teachers) of 10 have home prices be worth more or less than a neighborhood that has an `'PTRATIO'` value of 15?

**Answer: After reading the following questions very carefully here are the points which justify my answer of the following question:

Awnswer 1:-The value of the MEDV value increases when the RM value increases that is more the value of RM then more the value of MEDV. For example the value of the house price increases/decreases according the size of the land and the number of the floors of the house but in this case the value of the house price increases when there are more number of rooms in the house. Therefore the house that has the RM value of 7 be more worth than the value of RM of 6.

Answer 2:-The value of MEDV decreases when the value of LSTAT increases. When more neighbourhood occupies in the boston area then value of MEDV will decrease dracticsally.

Answer 3:-When the PTRATIO increases then the value of MEDV decreases. When the student-teacher ratio increases it could be that the number of students are increasing i.e 120/1 which means 1 teacher is teaching to 60 students which indicates that there is lack of teachers for the particular school or the funding of the school is not proper where the value of MEDV of the houses decreases. If the student-teacher ratio decreases i.e if more number of schools are devloped then the teacher can give proper attention to all of the students which will later increase the education qaulity of the school where the value of MEDV also increases. The value of PTRATIO of 10 would be more worth than 15 for the purpose of the increase of the value of MEDV.

1.5 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.

1.5.1 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*, R2, 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 R2 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 R2 of 0 is no better than a model that always predicts the *mean* of the target variable, whereas a model with an R2 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 R2 as well, which indicates that the model is **arbitrarily worse** than one that always 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 [3]:
```
# 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
```

1.5.2 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

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

In [4]:
```
# 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.
```

- Would you consider this model to have successfully captured the variation of the target variable?
- Why or why not?

** Hint: ** The R2 score is the proportion of the variance in the dependent variable that is predictable from the independent variable. In other words: * R2 score of 0 means that the dependent variable cannot be predicted from the independent variable. * R2 score of 1 means the dependent variable can be predicted from the independent variable. * R2 score between 0 and 1 indicates the extent to which the dependent variable is predictable. An * R2 score of 0.40 means that 40 percent of the variance in Y is predictable from X.

Answer: As we have seen R^2 score have a score between 0 and 1 and the score which we have obtained is 0.923 which s very close to 1.So yes we would consider the model have successfully captured the variation of the target variable.

From the score we can predit that 92.3% of the variance in Y is predictable from X and as the R2 score is very close to 1 it really means that dependent variable can be predicted from the indepenedent variable.

So here in the boston housing price the features which we are using is size of the land,

1.5.3 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 [5]: # 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_size=0.2, ran

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

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

1.5.4 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: Think about how overfitting or underfitting is contingent upon how splits on data is done.

Answer:

The benefit of splitting the training and testing data are as follows:- The training data from which the data which is to be trained to get the best accuracy of the model and the training is done on the data to get the best model of the algorithm with the accuracy with more than 90% or 85%.

The testing data is used to test the data in the real world from which the model is implemented in the real world where the testing data is used to check the model whether the model is accurate or not.If the testing data is not used then the model of the algorithm will be not applicable to use in the real world.

Warning:- From the instructor we got the warning that the testing data cannot be used for training.

In the case of errors of the model i.e overfitting and underfitting of the model can be determined by the training and testing model of the algorithm from which sum of the errors of training model and the sum of the testing model can help us to plot the graph whether the model is overfitting or underfitting.

1.6 Analyzing Model Performance

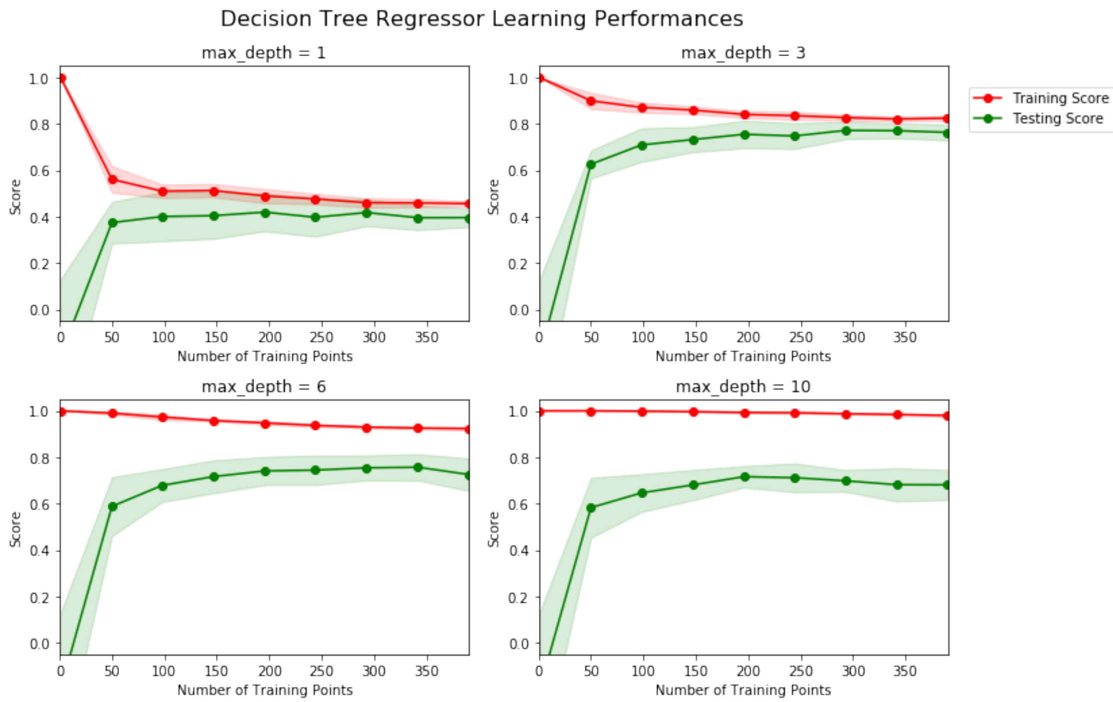
In this third section of the project, you'll take a look at several models' learning and testing performances on various subsets of training data. Additionally, you'll investigate one particular algorithm with an increasing `'max_depth'` parameter on the full training set to observe how model complexity 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.

1.6.1 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 R2, the coefficient of determination.

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

In [6]:
```
# Produce learning curves for varying training set sizes and maximum depths
vs.ModelLearning(features, prices)
```



1.6.2 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? Generally speaking, the more data you have, the better. But if your training and testing curves are converging with a score above your benchmark threshold, would this be necessary? Think about the pros and cons of adding more training points based on if the training and testing curves are converging.

Answer:

I would like to choose graph no 2 with max_depth=2. In the case with the other three graphs after carefully observing i would like to give some points which are as follows:- Graph 1 I have observe that the score is reducing at some point and the more training data testing data is stuck at one score with slightly ups and downs. Graph 3 I have observe that the testing data is not making good score with large gap between the training data and testing data. With more training data the score of the testing data is not scoring very well. Graph 4 I have observe that the testing data and the training data is making large gap and the score is not very well which makes it difficult to use it in the real world for testing the data whether the data is complex or simple.

Hence the graph 2 with max_depth=2 you can observe that the score is increasing between the training and testing data with more training of the model.

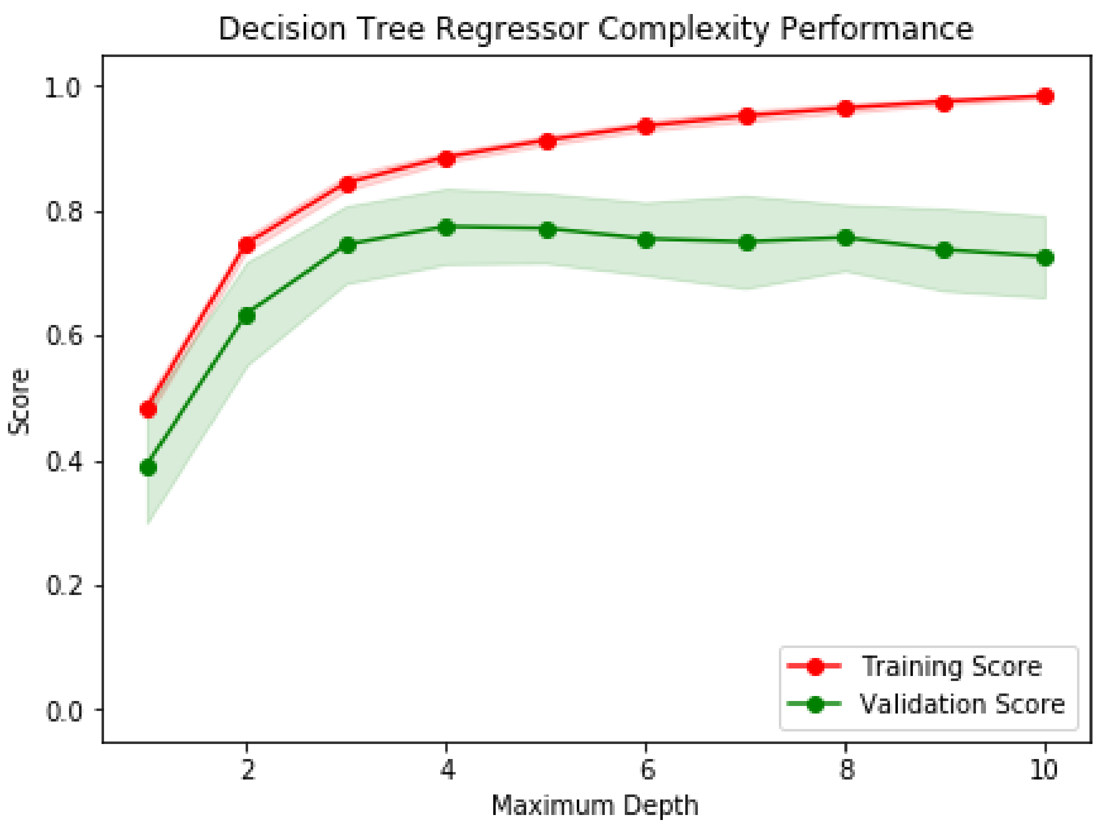
Yes more data for the training model will make the programmer whether the model he have been used for training the data is accurate or not and after the splitting the data into training and testing from where the checking the score between the training and testing if the model is overfitting or underfitting or no errors. But sometimes at some point of the training model with more training data cannot benifit as the score between the training and testing will decrease and the accuracy of the model will decrease.

1.6.3 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 Q5 and Q6. **

```
In [7]: vs.ModelComplexity(X_train, y_train)
```



1.6.4 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: High bias is a sign of underfitting(model is not complex enough to pick up the nuances in the data) and high variance is a sign of overfitting(model is by-hearting the data and cannot generalize well). Think about which model(depth 1 or 10) aligns with which part of the tradeoff.

Answer:

After visualizing the model very carefully it have been observed that the maximum depth of 1 the model will suffer from high bias with low variance which leads to underfitting as the model gets more training then the model will suffer from low bias and high variance whic leads to overfitting.

When the model is trained with maximum depth of 10 then the model will lead to overfitting where the model have high variance and low bias. At one point of the training the model was coming from underfitting to good fitting to overfitting where the score at the maximum depth of 10 it leads to overfitting as the model is memorizing where the model is getting more complex and the score is reducing where more training is given to the model.

1.6.5 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?

** Hint: ** Look at the graph above Question 5 and see where the validation scores lie for the various depths that have been assigned to the model. Does it get better with increased depth? At what point do we get our best validation score without overcomplicating our model? And remember, Occams Razor states "Among competing hypotheses, the one with the fewest assumptions should be selected."

Answer:

The max depth of 3 would be best because when the training is given to the model the score is making a very good score from which is neither model is leading to underfitting or overfitting it is a good fitting where the model is neither memorizing when the model is too complex nor the model is simple.

The model would be perfect at the max depth at 3 where the training data would give the good accurate and testing the data in the real world the model would predict the prefect answer for the data when testing with the good accuracy. The model at the max_depth of 4 the score is slighly decreasing where it is leading to overfitting when the model is goven more training

The model would be perfect which is neither underfitting nor overfitting.

1.7 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`.

1.7.1 Question 7 - Grid Search

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

** Hint: ** When explaining the Grid Search technique, be sure to touch upon why it is used, what the 'grid' entails and what the end goal of this method is. To solidify your answer, you can also give an example of a parameter in a model that can be optimized using this approach.

Answer:

GridSearch is an exhaustive search over specified parameter values for an estimator, used to search the best combination of parameters from a grid with possible parameter values. It can be used to optimize learning algorithm by returning the best classifier for the combination of provided parameters.

1.7.2 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: When explaining the k-fold cross validation technique, be sure to touch upon what 'k' is, how the dataset is split into different parts for training and testing and the number of times it is run based on the 'k' value.

When thinking about how k-fold cross validation helps grid search, think about the main drawbacks of grid search which are hinged upon **using a particular subset of data for training or testing** and how k-fold cv could help alleviate that. You can refer to the docs for your answer.

Answer:

The k-fold cross-validation training technique is the process of dividing your data points into smaller number of k bins. Testing then occurs on one of the k bins while training occurs with the other k-1 bins. This process, testing and training, occurs k times across all bins for testing and training. The average of the k testing experiments are used as the overall result of the model.

Although grid search automates the parameter selection and tuning for best performance, not using cross-validation could result in the model being tuned only to a specific subset of data. This is because without using a technique such as cross-validation, for example, only using kfold to create testing and training data, will not shuffle your data points, i.e if your dataset is ordered or in any pattern, grid search would only perform tuning on the same subset of training data. Utilizing cross-validation, eliminates this issue by using the entire dataset allowing grid search to optimize parameter tuning across all data points.

1.7.3 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*.

In addition, you will find your implementation is using `ShuffleSplit()` for an alternative form of cross-validation (see the `'cv_sets'` variable). While it is not the K-Fold cross-validation technique you describe in **Question 8**, this type of cross-validation technique is just as useful!. The `ShuffleSplit()` implementation below will create 10 (`'n_splits'`) shuffled sets, and for each shuffle, 20% (`'test_size'`) of the data will be used as the *validation set*. While you're working on your implementation, think about the contrasts and similarities it has to the K-fold cross-validation technique.

Please note that ShuffleSplit has different parameters in scikit-learn versions 0.17 and 0.18. For the `fit_model` function in the code cell below, you will need to implement the following: - Use `DecisionTreeRegressor` 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` 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` 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 [10]:
```
# 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
    # sklearn version 0.18: ShuffleSplit(n_splits=10, test_size=0.1, train_size=None, r
    # sklearn versiin 0.17: ShuffleSplit(n, n_iter=10, test_size=0.1, train_size=None,
    cv_sets = ShuffleSplit(X.shape[0], n_iter = 10, test_size = 0.20, random_state = 0)

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

    # TODO: Create a dictionary for the parameter 'max_depth' with a range from 1 to 10
    params = {'max_depth':list(range(1,11))}

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

    # TODO: Create the grid search cv object --> GridSearchCV()
    # Make sure to include the right parameters in the object:
    # (estimator, param_grid, scoring, cv) which have values 'regressor', 'params', 'sc
    grid = GridSearchCV(estimator=regressor, param_grid=params, scoring=scoring_fnc, cv

    # 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_
```

1.7.4 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.

1.7.5 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 [11]: # 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()['max
```

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

** Hint: ** The answer comes from the output of the code snipped above.
Answer:
I would like to argue that the result produced have the max_depth of 4 and the max_depth which I have answered is 3 but if you carefully observe that model is best for trainig is at max_depth of 3 till the max_depth of 4 where we would have a very good accuracy after that the accuracy would fall down where more training is given which leads to overfitting and the model memorizes.

1.7.6 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. Of the three clients, client 3 has has the biggest house, in the best public school neighborhood with the lowest poverty level; while client 2 has the smallest house, in a neighborhood with a relatively high poverty rate and not the best public schools.

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

```
In [12]: # 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, price))
```

```
Predicted selling price for Client 1's home: $391,183.33
Predicted selling price for Client 2's home: $189,123.53
Predicted selling price for Client 3's home: $942,666.67
```

Answer:

Cilent 1:-The price which the cilent is selling is 391K which is at the reasonable price.If we check the median price which is 454K and the number of rooms & student-teacher ratio which is quite reasonable where the price will attract the meduim-earning person where the poverty around the area is not much higher percentage which will not keep negative impact of the area.

Cilent 2:-The price which the cilent is keeping the quote of 189K which is quite lower and having poverty very high percentage,the number of rooms is 4 where room is paid on rent by each person (depends on the situation) & the student-teacher ratio is quite high which the selling price is quite low.

Cilent 3:-The Cilent have quoted very high price which high number of rooms,student-teacher ratio very low and the poverty level is also very low where the price which is quite close maximum selling price then the price is resonably priced for the features the cilent is providing.

1.7.7 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 respect to the data it's trained on.

```
In [13]: 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: $413,334.78
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
```

1.7.8 Question 11 - Applicability

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

Hint: Take a look at the range in prices as calculated in the code snippet above. Some questions to answering: - How relevant today is data that was collected from 1978? How important is inflation? - Are the features present in the data sufficient to describe a home? Do you think factors like quality of apppliances in the home, square feet of the plot area, presence of pool or not etc should factor in? - Is the model robust enough to make consistent predictions? - Would data collected in an urban city like Boston be applicable in a rural city? - Is it fair to judge the price of an individual home based on the characteristics of the entire neighborhood?

Answer:

1:- The data which was collected at 1978 which was very revelant where the prices of the houses can be predicted how much infltion can happen in the future depending upon the features such as the number of rooms,poverty level,student-teahcer ratio,etc.

2:-The features which we have discussed so far is insufficient in the real world where we would need more featureas such as land size of the house which is given in the question. There could be any features such as population of the area,types of people who are staying,types of shops around the area and other features should be considered in the real world to predict the real price of the house/land.

3:-The model is not robust to make predictions of the house.

4:-The price which was collected like the Urban City Boston would not be applicable in the rural city where there is a large differences between urban city and rural city.For the case of the prediction of the prices of house in rural city there are many features which we have to considered.

5:-Yes It is fair to judge the price of the house based on the entire neighbourhood where the entire neighbourhood makes the society and from the society to the area/colony as we say. Depending upon the neighbourhood the types of people would buy the house if the price of the house is convinient for them to buy and live happily in the area with neighbourhood.

> **Note**: Once you have completed all of the code implementations and successfully answered each question above, you may finalize your work by exporting the iPython Notebook as an HTML document. You can do this by using the menu above and navigating to
> **File -> Download as -> HTML (.html)**. Include the finished document along with this notebook as your submission.