P1: Predicting Boston Housing Market

Question 1

Of the features available for each data point, choose three that you feel are significant and give a brief description for each of what they measure.

- CRIM: Crime rate per capita by town.
- RM: Average number of rooms per dwelling.
- DIS: weighted distances to five Boston employment centers.

Question 2

Using your client's feature set `CLIENT_FEATURES`, which values correspond with the features you've chosen above?

```
np_CLIENT_FEATURES = np.array(CLIENT_FEATURES[0])
feature_names = np.array(city_data.get('feature_names'))
feature_columns = np.array([np.where('CRIM'==feature_names)[0][0],
                            np.where('RM'==feature_names)[0][0],
                            np.where('DIS'==feature_names)[0][0]])

print np_CLIENT_FEATURES[feature_columns]
```

```
[ 11.95 5.609 1.385]
```

Question 3

Why do we split the data into training and testing subsets for our model?

We split the data into training and testing sets so that we develop the model using the training set and then measure the performance of the model using an out of sample set (the test test). Also, the testing dat provides us with better insights about whether we are overfitting or under fitting (variance and bias). We must always split the data into training and test sets unless our data contains **all** the population.

Question 4

Which performance metric below did you find was most appropriate for predicting housing prices and analyzing the total error. Why?

- *Accuracy*
- *Precision*
- *Recall*
- *F1 Score*
- *Mean Squared Error (MSE)*
- *Mean Absolute Error (MAE)*

For a classifier model where the target values are labeled data, the first four metrics would be appropriate.

however, the target values of our data `housing_prices` are numerical ones and therefore a regression model would be suitable choice. For regression models, the MSE or MAE would be the appropriate ones to use. Both of two metrics measures the relative distance of the predicted values from the actual values. The two formulas of those two metrics are:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{predicted} - y_{actual})^2$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_{predicted} - y_{actual}\right|$$

Because the MSE takes the square distance instead of taking the absolute one in MAE, it penalizes extreme values more than the MAE does. Therefore, for this project I will go for the MSE.

Question 5

What is the grid search algorithm and when is it applicable?

Grid search algorithm is brute force solution for choosing the optimal hyper-parameters to select the best model. It is applicable when we have a range of values for each hyper-parameter and we would like to find a combination of those values that will result the best performance scores.

Question 6

What is cross-validation, and how is it performed on a model? Why would cross-validation be helpful when using grid search?

Cross-validation is a method used for evaluate the generalized performance of the model and fine tuning hyperparameters. There are many different methods for cross validation. One popular choice is dividing the data into training set, validation set and testing set (a 60%, 20%, 20% split for example). However, since our data is not big enough, using this method will take out valuable data that could be used to train the model. Therefore, a better method in this situation would be K-fold cross validation where it matters less how the data gets divided. In the case of K-fold cross validation, the training set is divided into K equal sized folds. Then, a single fold is hold out and treated as a validation data, the remaining K-1 folds are used for the training the model and then the trained model is tested on the validation data (the excluded fold). The process is repeated K times and than the average error across all the fold is computed. A pseudocode for the process is summarized below:

```
Divide the data into K folds.
for `i=i:K`:
    Train the model on the data exluding fold `i`.
    Test the trained model on fold `i`
    Compute the error
Return the average error.
```

Moreover, cross-validation is helpful when using grid search over a set of hyperparameters because ensure the model does not overfit its data by tuning. Grid search exhaustively searches through a manually specified subset of hyperparameter space of a learning algorithm. Then, the resulting models are evaluated using cross-validation.

Question 7

Choose one of the learning curve graphs that are created above. What is the max depth for the chosen model? As the size of the training set increases, what happens to the training error? What happens to the testing error?

The graph chosen is the one with depth 1. The training error when the size of the training set is small is small since there is less variance to accommodate while the testing error is high since the model hasn't been trained with enough examples. As the size increases, the training error starts to increase. if the model is underfiting, the increase will be high and if it is overfitting, the increase will be low. our chosen curve clearly indicates that the model is underfitting since the increase in the training error is high when the size increases. The testing error decreases as the size increases since now the model is better trained with more training example.

Question 8

Look at the learning curve graphs for the model with a max depth of 1 and a max depth of 10. When the model is using the full training set, does it suffer from high bias or high variance when the max depth is 1? What about when the max depth is 10?

With a depth of 1, the model suffers from high bias since both of the training and testing set have high errors. On the other hand, the model with depth 10 suffers from high variance since there is a large gap between the two errors.

Question 9

From the model complexity graph above, describe the training and testing errors as the max depth increases. Based on your interpretation of the graph, which max depth results in a model that best generalizes the dataset? Why?

As the model complexity increases, the model go through two stages. The first stage is when the model complexity is small. In this stage, the model suffers from a high bias (underfitting) and both the training and testing errors are high. The second stage is when the model complexity is high. Here the model suffers from high variance (overfitting) where the training error is low and the testing error is high. Therefore, the max depth that will best generalizes the dataset is the one which the training error is low and testing error is at global minimum. According to the curve above, this would be around a depth of 5 or 6.

Question 10

Using grid search on the entire dataset, what is the optimal `max_depth` parameter for your model? How does this result compare to your intial intuition?
Hint: Run the code block below to see the max depth produced by your optimized model.

According to optimized model, the maximum depth would be 4. This is not far from my initial tuition since the learning curves with depth of 3 and 6 above indicate that the best model would be somewhere between them. That is because those two models have small errors and have the smallest gaps between the two errors.

Question 11

With your parameter-tuned model, what is the best selling price for your client's home? How does this selling *price compare to the basic statistics you calculated on the dataset?*

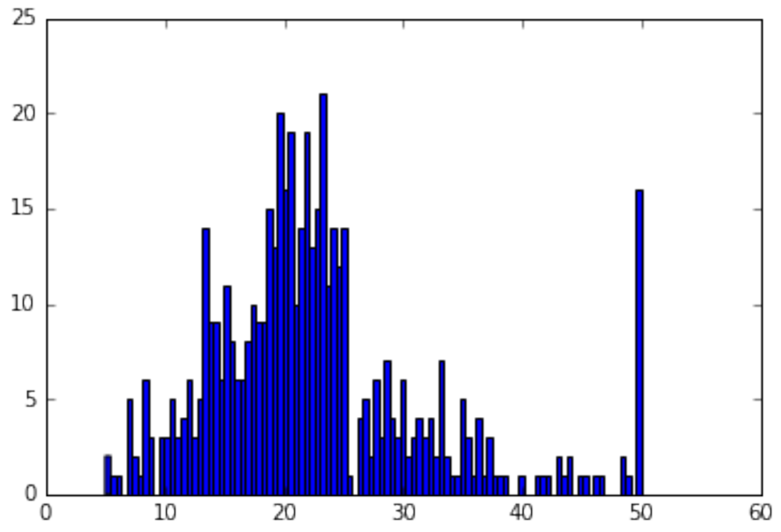
According to out trained model, the best selling price would be $21,630, which is close of the mean housing prices in Boston and within one standard deviation from the mean.

Question 12 (Final Question):

In a few sentences, discuss whether you would use this model or not to predict the selling price of future clients' homes in the Greater Boston area.

According to the learning curves above, there is still a small gap between the train and testing error due to over fitting. Therefore, I might use this model if we are able to improve it. One way would be collecting more data. Note that collecting more data does not necessarily solve the over fitting problem. Or perhaps we need to select smaller set of features. Also, perhaps the outliers in our data are affecting the model's performance. The histogram below clearly shows that there are somee extreme outliers in our `housing_prices` data.

```
_ = pl.hist(housing_prices,bins=100)
```


25
20
15
10
5
0
0
10
20
30
40
50
60