UDACITY Logout

PROJECT

Predicting Boston Housing Prices

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW | CODE REVIEW | NOTES

Meets Specifications

SHARE YOUR ACCOMPLISHMENT

Dear student,

well done addressing every previous issue, you made it for a very good submission. I've posted some pro tips for you in case you might be interested in learning more about some specific topics. Please consider my optional comments regarding question 10 as that answer could be more thorough, the Pro Tip there is quite interesting as well :)

Congratulations on passing you exam!

Data Exploration

✓ All requested statistics for the Boston Housing dataset are accurately calculated. Student correctly leverages NumPy functionality to obtain these results.

✓ Student correctly justifies how each feature correlates with an increase or decrease in the target variable.

Developing a Model

✓ Student correctly identifies whether the hypothetical model successfully captures the variation of the target variable based on the model's R^2 score.
The performance metric is correctly implemented in code.

✓ Student provides a valid reason for why a dataset is split into training and testing subsets for a model. Training and testing split is correctly implemented in code.

Analyzing Model Performance

✓ Student correctly identifies the trend of both the training and testing curves from the graph as more training points are added. Discussion is made as to whether additional training points would benefit the model.

✓ Student correctly identifies whether the model at a max depth of 1 and a max depth of 10 suffer from either high bias or high variance, with justification using the complexity curves graph.

✓ **Student picks a best-guess optimal model with reasonable justification using the model complexity graph.**

Evaluating Model Performance

✓ **Student correctly describes the grid search technique and how it can be applied to a learning algorithm.**

Pro tip: There are other techniques that could be used for hyperparameter optimization in order to save time like RandomizedSearchCV, in this case instead of exploring the whole parameter space just a fixed number of parameter settings is sampled from the specified distributions. This proves useful when we need to save time but is not necessary in cases in cases like ours where the data set is relatively small.

✓ **Student correctly describes the k-fold cross-validation technique and discusses the benefits of its application when used with grid search when optimizing a model.**

Excellent!

✓ **Student correctly implements the `fit_model` function in code.**

✓ **Student reports the optimal model and compares this model to the one they chose earlier.**

✓ **Student reports the predicted selling price for the three clients listed in the provided table. Discussion is made as to whether these prices are reasonable given the data and the earlier calculated descriptive statistics.**

Ideally you would discuss the price of each house individually using its features to support your argument. Do you think that the model did an appropriate job in pricing those homes? Why? (For instance: What happens when you compare the number of rooms of the first house with the average number of rooms or with the other two houses? Which conclusions can you draw from that comparison? Can you extend the same methodology to the net worth and the student-teacher ratio?)

Pro tip: To assess if your prediction is reasonable, besides from comparing it with the median, the mean and checking if it is included in one standard deviation range, you could use SKlearn to find the nearest neighbours of the feature vector. You can then contrast your results with the closest neighbours, the ones that have similar characteristics.

```
from sklearn.neighbors import NearestNeighbors
num_neighbors=5
def nearest_neighbor_price(x):
    def find_nearest_neighbor_indexes(x, X):  # x is your vector and X is th
e data set.
        neigh = NearestNeighbors( num_neighbors )
        neigh.fit(X)
        distance, indexes = neigh.kneighbors( x )
        return indexes
    indexes = find_nearest_neighbor_indexes(x, features)
    sum_prices = []
    for i in indexes:
        sum_prices.append(prices[i])
    neighbor_avg = np.mean(sum_prices)
    return neighbor_avg
print nearest_neighbor_price( [4, 55, 22])
index = 0
for i in client_data:
    val=nearest_neighbor_price(i)
    index += 1
    print "The predicted {} nearest neighbors price for home {} is: ${:,.2f}
".format(num_neighbors,index, val)
```

http://scikit-learn.org/stable/modules/neighbors.html#finding-the-nearest-neighbors

✓ **Student thoroughly discusses whether the model should or should not be used in a real-world setting.**

DOWNLOAD PROJECT