UDACITY

PROJECT

Predicting Boston Housing Prices

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Congratulations on passing this assignment, on your first shot! You have done a great job to demonstrate very good understanding of the concepts and techniques. Keep up your excellent work!

Data Exploration

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

Well done on getting the statistics using `numpy`. For most real world projects, getting statistics of the data is the first step to understand the data before subsequent feature engineering can take place. We may also examine if there are any missing values and outliers.

There are other statistical measures supported by `numpy`. For example, we can look at the percentile of the data:

```
first_quartile = np.percentile(prices, 25)
third_quartile = np.percentile(prices, 75)
```

Suggestion on code: we can use `prices` in place of `data[MEDV]`.

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

Your reasoning makes perfect sense. You can also visualize the features vs. prices to verify your intuition:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 5))
for i, col in enumerate(features.columns):
    plt.subplot(1, 3, i)
    plt.plot(data[col], prices, 'o')
    plt.title(col)
    plt.xlabel(col)
    plt.ylabel('prices')
```

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.

With this high R^2 value we may say that the model has done a good job to capture variation of the data.

Suggestion: It would be good if you can elaborate on how does the result compare with the range of R^2? Is it close to the upper bound?

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.

Well done on splitting the data, and you have successfully captured the key benefit of doing it.

Without a testing set being reserved, there is no data to validate the model to ensure it is working well before being used in the general context. Therefore we need to train the model using training subset and test it over the testing subset to confirm whether the predictions being made on *unseen* data are correct.

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.

Indeed, as we have more data, we can generalize better on the test set initially, but after certain limit more data does not really help much.

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.

You are right. Visually, for high bias, the training score is low and close to the test score. On the other hand, for high variance, there is a large gap between training and test scores.

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

I would pick max depth of 3 as well, as this seems to be the turning point between underfitting and overfitting.

Evaluating Model Performance

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

To be more precise, grid search is an exhaustive search algorithm, and it searches over *all* combinations of parameters we specify to find the optimum combination that yields the best performance.

Due to its exhaustive search nature, grid search can be computationally expensive, especially when data size is large and model is complicated. Sometimes we resort to randomized search in this case to search only *some* combinations of the parameters.
Ref: http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.RandomizedSearchCV.html#sklearn-grid-search-randomizedsearchcv

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.

Nice description of k-fold cross validation, which improves accuracy and robustness of grid search by making use of all available data.

K-fold CV does have its limitations:

- it is more computationally expensive than hold out method
- it does not work well when data is not uniformly distributed (e.g. sorted data).

Student correctly implements the `fit_model` function in code.

Well done on implementing `fit_model`, which may be the most important code block for this assignment.

Suggestion on code: It would be good to set the `random_state` of `DecisionTreeRegressor` to ensure the result is reproducible.

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

In fact I would say both depths pf 3 and 5 are acceptable, and the difference arises as a result of different data used for in the graphical representation and the grid search. Arguably grid search should be more reliable.

As an interesting note, you may also read about Occam's Razor principle, which favours a simpler model to more complicated ones.

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

Well done and awesome justification. This is really important, as in real-world projects, often we are not only required to get the result, but also to explain the result as well.

We can also visualize the result by overlaying the prediction with histogram of the data as:

```
import matplotlib.pyplot as plt
plt.hist(prices, bins = 20)
for price in reg.predict(client_data):
    plt.axvline(price, lw = 5, c = 'r')
```

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

Great justification here.

Suggestion: You can also discuss about the robustness of the model based on the sensitivity analysis earlier. Do you think the results are consistent?

DOWNLOAD PROJECT

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH

Rate this review

Student FAQ