UDACITY

Back to Machine Learning Engineer Nanodegree

Predicting Boston Housing Prices

REVIEW

CODE REVIEW

HISTORY

Meets Specifications

Dear student

Great job on this project! Your submission now meets all specifications. You've shown that you can diagnose problems with a model using the training/testing performance and that you have a strong understanding of the relationship between model complexity and behavior. These are principles that you can use in any area of machine learning so pat yourself on the back! Congratulations on passing the project and good luck with the next section of the course!

Cheers!

Data Exploration

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

Great job using NumPy to document the dataset statistics!

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

Nice analysis here! Your response is correct and represents what we see if each feature is plotted vs. the housing 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.

> R2 is .923 which is very high between 0 and 1 - so yes the model has shown to capture most of the variance in Y resulting from X.

Great job! While it's true that this is certainly a positive result, please keep in mind that we only have 5 data points so far and we don't know anything about the model. This means we probably shouldn't completely trust the model yet.

https://en.wikipedia.org/wiki/Coefficient_of_determination#Caveats

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.

```
X_train, X_test, y_train, y_test = train_test_split(features, prices, train_size=0.8
, random_state=10)
```

Looks good!

> Train-Test-validate datasets alerts model developers for overfitting/underfitting and also exposes errors based on bias, variance.

Nice job! If we don't have sufficient *independent* data for testing, we can't tell if the model is generalizing to patters in the dataset or simply memorizing the training data (overfitting). Too little training data and the model will regress to the mean.

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.

> Adding more training point beyond 300 datapoints may not provide additional benefits as the curve has become flat (not learning / not improving accuracy).

This is correct. In fact, adding even more data at this point could be counter productive, since it will make the model more computationally expensive to train at no benefit to the model's performance.

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.

> when trained with max_depth=1, the above model shows around 0.4 for training and testing score which suggests that the model is not complex (fewer decision tree rules) enough to better fit the data and runs the risk of underfitting (high bias).

Nice job! Keep in mind that the key indicator of high bias is poor training performance (since testing/validation performance will be poor in both cases of high bias or high variance).

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.

Nice explanation. Because the grid search algorithm is trying each combination (the outer or Cartesian product of the matrix) of parameters one-at-a-time, this algorithm is going to be computationally expensive if the parameter space is very large or continuous. In these cases, a gradient decent approach or a randomized search algorithm would be better (like this one):

http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

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.

> The entire dataset is split into "k" parts/folds. The experiment is run "k" times. Each time, 1 fold is

> chosen as testing set and remaining (k-1) folds as training sets. Testing fold is never repeated across "k" runs. Each run gives a CV accuracy. Finally we calculate the average of CV accuracy over all runs.

Great job! This is very clear. One thing that I'd caution you about is using the term "accuracy" as a synonym for performance. Accuracy is a specific statistical metric, so it's best to only talk about accuracy in that context when you're writing about a machine learning problem.

> The benefit of using this technique is to avoid the variance that might creep in while doing random train-test split. Since each datapoint is used as testing set atmost once and also features in training set, this is helpful to work with small/limited available data. K-fold CV is also used during model selection phase (in that case, the final model is again tested upon unseen data). This works best on uniformly distributed data.

Well stated! CV will prevent the model from being overfit on a single split of the dataset. If the dataset was split (by chance) in a way that created a training subset that didn't represent the dataset, this could cause big problems. Folding the data multiple times makes this much less likely.

Student correctly implements the `fit_model` function in code.

```
regressor = DecisionTreeRegressor(random_state=0)
```

Great job setting a random state variable here!

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

> Answer: Parameter 'max_depth' is 4 for the optimal model.

Suggested:

- It would also be a good idea to note how this result compares to your answer to Question 6 (you were spot on!).

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.

Great job using the dataset statistics to bolster your answer! You've also noted that the features of the house are correlating with the predicted prices. This type of analysis can act as a crucial 'sanity check' on our models

when they become large and complex.

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

DOWNLOAD PROJECT

RETURN TO PATH