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 for meeting all specifications for this project! I added some comments and suggestions below, I hope they'll be useful. Keep up the good work!

Data Exploration

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

Awesome

All statistics are calculated using `numpy`. Well done! :)

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

Suggestion

Another way to visualize your data is using the seaborn package, which abstracts away much of the code needed for a plot:

```
import matplotlib.pyplot as plt
import seaborn as sns

for var in ['RM', 'LSTAT', 'PTRATIO']:
    sns.regplot(data[var], prices)
    plt.show()
```

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.**

Awesome

> However, whether the model is useful ultimately depends on the application and we know nothing about it. Therefore I don't really feel comfortable saying the model is a successful or unsuccessful.

Excellent answer. Indeed, how good an R^2 score is really depends on the domain of the problem. In a well controlled environment such as laboratory testing, you would expect R^2 to be higher than in a "messy" study full of noisy data, uncertainty regarding measurements, etc.

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.

Comment

It's important to understand what's going here, and why the learning curves behave like they do:

For the **training curve**, the thinking goes like this:

- With few training points, it's quite easy for the model to modify its parameters so that the train targets are fully approximated by the model's output function.
  - To be a little extreme: if we have just two training points, even a simple linear model will be able to perfectly fit them, since you can connect any two points with a straight line.
- As the number of training points grows, perfectly fitting them becomes more difficult, so the train score will tend to go down.
  - This tendency will show itself quicker if the model is less flexible. Using the extreme example above again, for a simple linear model three training points may be enough for a perfect fit to be impossible.

For the **testing curve**, the idea is that:

- With few training points, very little information is passed to the learner. It may be able to perfectly fit the training points, but it has very little to go by when it's time to predict unseen data, so the test score will probably be quite low.
- As more training points are added, the model receives more information and is able to make better predictions for unseen data, so the test score will grow.
- But each model has only so much flexibility, and can extract only so much information from the training data. That's why, at some point, test score will stabilize, and additional training data will not help the model.
  - The more flexible (or complex) the model is, the harder it is for this "information limit" to be met, and the more training data the model will need until its capability to extract information from it is exhausted.

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.

Awesome

You show a great understanding of learning curves, model complexity curves, and visual cues in them to identify bias and variance problems in our model. Nice job!

Evaluating Model Performance

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

Comment

- Because it exhaustively searches over all possible parameter combinations, grid search can be very time consuming and computationally expensive.
- One alternative to it is randomized search, which will search through a predefined number of parameter combinations randomly selected from the full grid.
- This page from scikit-learn's documentation shows that randomized search can achieve results that are comparable to those of full grid search.

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.

Awesome

> 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.

Precisely. Cross-validation decreases the risk of "overfitting by overtuning", or selecting an instance of the model that *fits the validation set well* instead of one that generalizes well to unseen data.

Student correctly implements the `fit_model` function in code.

Comment

Note that in Python 2 `range` returns a list, so you can write `'max_depth': range(1, 11)` instead of `list(range(1,11))`. Python 3 is a different matter, though.

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 for each of the three predictions as to whether these prices are reasonable given the data and the earlier calculated descriptive statistics.

Suggestion

To go above and beyond on this question, you could also compare the features of the three examples to some statistics of the distribution of features for the whole dataset. You can check these statistics by running `features.describe()`, which will print out the five-number summary, the mean, and the standard deviation of each feature.

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

Comment

> I don't think this features taken in the datasets are sufficient for predicting housing price.

- Adding **meaningful features** to our model may be a good way of reducing the so-called **irreducible error**, or the error that comes from variation in our target that our predictors are not able to explain.
- More predictors (as long as they have a true relationship with the target that is not captured by the other predictors in our data set) will give useful information to our model, which can use this information to improve its predictions.
- You can read more about the irreducible error, bias, and variance in the resources linked in this post from the project's discussion forum.

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

Student FAQ