UDACITY

‹ Return to "Machine Learning Engineer Nanodegree" in the classroom

DISCUSS ON STUDENT HUB

Predicting Boston Housing Prices

REVIEW

HISTORY

Meets Specifications

Udacity student,

your project shows how committed you are to the course. You probably spent hours or days on this project and really should be proud of your work here and I'm proud of being part of your journey!

Keep the great work on the next sections of this amazing Udacity course!!!

I share with you some extra links from Medium:

A guide to start your path in Data Science and Machine Learning:

Fundamental Python Data Science Libraries

This last one is not for only a job interview, but it contains a lot of useful information about some great topics for a machine learning professional:

Data Science and Machine Learning Interview Questions

😄

Data Exploration

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

Correct! 😄

You have answered all statistics questions using the Numpy library. Nice job! 👍

It is important here to know why Udacity ask students to use the NumPy library:

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Numpy Documetation

It's always important to be aware of tools you use. For example, the Pandas' Series.std() will by default give you different result than numpy.std(). It's because Numpy takes in count the whole population while pandas assumes that you are evaluating the standard deviation for a sample of your dataset.

This article has a very good explanation about it.

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

Correct! 😄

You have correctly justified how each feature correlates with an increase or decrease in the target variable.

In addition, you can also plot your data to confirm your intuition and practice some coding skills using a library called matplotlib

```
import matplotlib.pyplot as plt
plt.figure(figsize=(15, 5))
for i, col in enumerate(features.columns):
    plt.subplot(1, 3, i+1)
    plt.plot(data[col], prices, 'x')
    plt.title('%s x MEDV' % col)
    plt.xlabel(col)
    plt.ylabel('MEDV')
```

Matplotlib Documentation

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.

Great job! 😄

Since R2 Score is close to one we may say it is a good model.

The R-Squared is very common score used in Data Science / Machine Learning. But we can have another type of R-squared called Adjusted R-Squared.

Also, in this article from Duke University you can find a very nice opinion about how good R2 Score can be in a ML mode.

I'd like to share one more excellent article: Mean Squared Error, R2, and Variance in Regression Analysis

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.

Correct! 😄

You have implemented a random_state for the train_test_split function properly.

The benefit of splitting a dataset into training and testing subsets is that the training set can be used to train a model while the testing set can be used to estimate how well a model generalizes to previously unseen data. A model that is overfitting the data may perform well on the training set but its generalization error (which is calculated using the testing set) will be high. A model that is underfitting neither will perform well on the training set nor generalize well. This allows estimating how well a model generalizes before launching it in production.

The training set is bigger than the testing set because a model trained on a bigger dataset is most likely to perform better than one trained on a smaller one. A higher training set may also prevent a model from overfitting (because the model will be trained on different training examples).
The data should be split into training and testing subsets randomly to prevent bias in the order of the data points.

In this article about "what's the purpose of splitting data up into test sets and training sets?" you may go deeper on your studies.

Splitting the dataset into a reasonable ration, we can generalize the model and avoid to perform worse when an unseen data is inputted.

In this video, you may have insights of how to split your data set and how to use Sckit-Learn for it:

Machine Learning Tutorial Python - 7: Training and Testing Data

Great job!

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.

I hope you got the importance of learning curves. If you interested about others learning curves for the same project, you can modify the file `visual.py` to plot other learning curves, for example with `max_depth=4` or `max_depth=5`. Just look for this part of the code:

```
# Create three different models based on max_depth
    for k, depth in enumerate([1,3,6,10]):

        # Create a Decision tree regressor at max_depth = depth
        regressor = DecisionTreeRegressor(max_depth = depth)
```

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.

Correct! 😄

Good job identifying high bias and high variance for different `max_depth` parameters.

The model which is trained with a maximum depth of 1 suffers from high bias because it performs well neither on the training nor on the validation set. The graph indicates this by showing that training and validation scores are low (close to 0.4) and both training score and validation score curves are close to one another.

The model which is trained with a maximum depth of 10 suffers from high variance because it performs really well on the training set but its performance on the validation set is not as high. This can be seen in the complexity graph by looking at the big gap between the training score and validation score curves. The training score is close to 1.0 while the validation score is less than 0.7.

In this link you can read more about high bias and high variance of data and boost your understanding about this topic!

Also in Wikipedia you may find a nice article about the tradeoff regarding to a model with high bias and high variance.

You may also check this website. It brought me a lot of light about this issue.

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

Correct! 😄

Evaluating Model Performance

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

Correct! 😄

You show really a very good understanding about the Grid Search.

I'd like just to point out that Grid Search will test EVERY combination of hyperparameters. So it can be very computationally expensive if you want a model with many hyperparameters or many sets of them.

Usually I have some tips and links about Grid Search and I'd like to share with you. It can be good and complete your studies and I hope so!

1. Official sklearn page on gridSearch
2. How gridSearch works
3. Specifying multiple metrics for evaluation
4. The scoring parameter: defining model evaluation rules
5. Defining your scoring strategy from metric functions

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.

Correct! 😄

You summarized well the K-fold concepts. I really liked your explanation and it shows that you understood this important machine learning tool.

The k-fold is a way to validate the parameters you've chosen for your model. It divides the trainset in k subsets and train the model in k-1 of them and evaluate it in the left set. To make it a stronger measure, it'll repeat the process k times, using a different subset from the k ones as the evaluation one, finally, takes the mean of the score used to evaluate the hyperparameters used in the k-fold cross-validation.

The benefits of grid search using k-fold cross-validation:

- Validation itself (it would be an advantage even with a single validation set), that keeps the test set really unseen. If we optimize the hyperparameters using an evaluation measure in the test set, it would kind leak information to the model, we may overfit the parameters to the test set because we're choosing them looking for a measure in it. And that's not good! The final result, the model assessment will be compromised, it would overestimate the generalization power of our model;
- The k-fold makes a robust measure taking the mean of a measure in different validation sets. The greater the k, more robust the measure;
- It's a good technique to use when we don't have many data points and couldn't split the data into a single and big validation set;

I'd like to share some extra readings about K-fold:

1. What is Cross Validation?
2. K-fold and Cross Validation

Student correctly implements the `fit_model` function in code.

Correct! 😄

Very nice implementation. 👍

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

Correct! 😄

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.

Correct! 😄

The selling prices you evaluated are correct and your discussion shows your very good understanding about this project. Also, it shows that you can extract insights from data using your intuition and abilities.

You can also plot the data and the predictions to have an intuition about the model:

```
prediction_data = np.transpose(client_data)
pred = reg.predict(client_data)
for i, f in enumerate(house_features):
    plt.scatter(features[f], prices,alpha=0.25, c='green')
    plt.scatter(prediction_data[i], pred, color='red', marker='D')
    plt.xlabel(f)
    plt.ylabel('MEDV')
    plt.show()
```

Good job!

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

Correct! 😄

Nice answers regarding to all questions in this last section.

DOWNLOAD PROJECT

RETURN TO PATH