UDACITY

PROJECT

Predicting Boston Housing Prices

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Requires Changes

1 SPECIFICATION REQUIRES CHANGES

You analysis is very impressive, as this program will be a breeze for you. Just need one minor adjustment and actually use Numpy for these stats and you will be good to go. Keep up the awesome work!!

Data Exploration

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

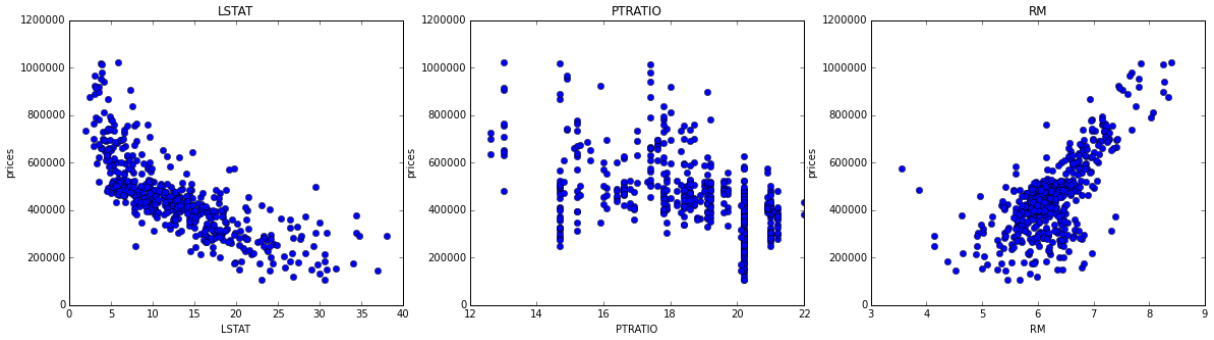
Although these are the correct ideas, for this section you should leverage NumPy functionality to obtain these results. Therefore you are actually using pandas `Series()` commands to receive these statistics in your implementation. Therefore I bet numpy has the same methods!!

```
minimum_price = np.min(prices)
....
```

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

Nice observations for the features in this dataset. As we can confirm these ideas by plotting each feature vs MEDV housing prices.

```
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')
```



Note: With your comment of "*At first glance it seems to me that the feature set is weak. While the reasons exposed may have some significance they seem quite inadequate to explain the significant price variance. We should proceed to draw correlation diagrams (or simply compute the correlation) between the features and the target variable to confirm this suspect or refuse it.*"

- Good intuition, you are correct that we probably need more features to model a true house.
- And performing EDA is a great idea. We can check out the correlation with

  ```
  data.corr()
  ```

  And check out the visual above.

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.

Nice job. As we can see with this high r^2 score of 92.3% (0.923) we can see that we have strong correlation between the true values and predictions.

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.

Spot on with your comment of "*If we don't verify the model on unseen data and use all data for training there is no way to determine if the model is overfitting and doesn't generalize well. There is no way to select a better model.*"

- As we definitely need to protect against overfitting
- As we can get a good estimate of our generalization accuracy on this testing dataset
- As our main goal is to accurately predict on new unseen data

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.

Great intuition. You are correct that the training score decreasing and testing score increasing makes sense, since with little amounts of the data we simply memorize the training data(no generalization), then when we receive more and more data points we can't simply memorize the training data and we start to generalize better(higher testing accuracy).

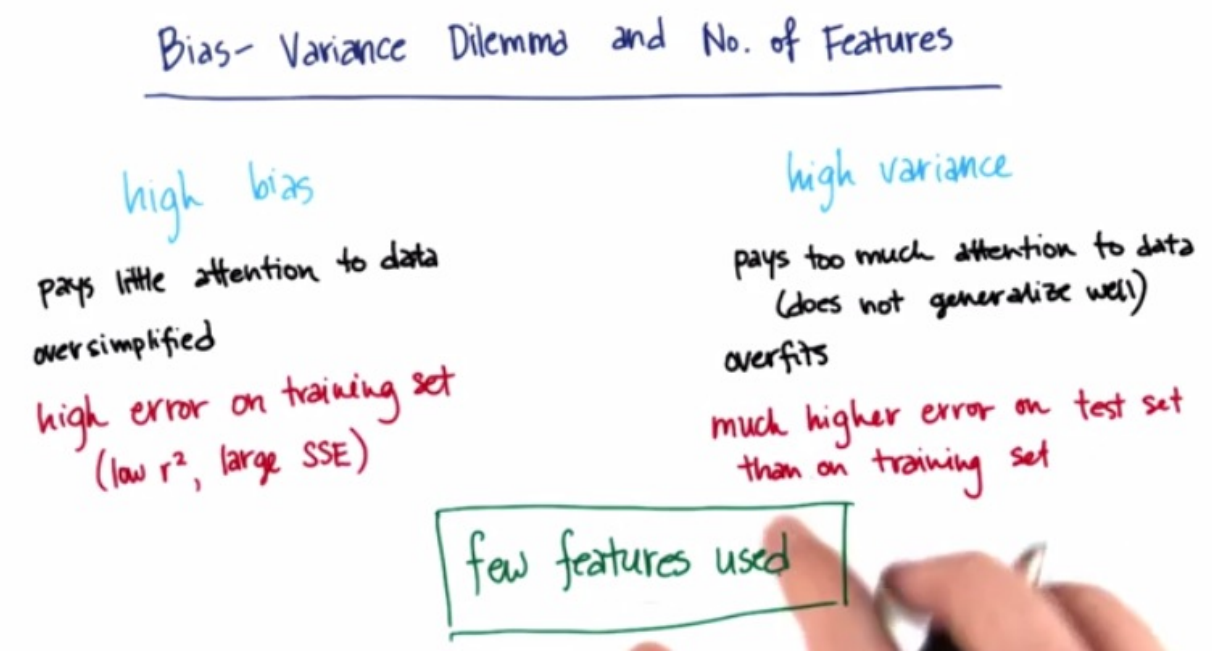
And correct with your comment of " *It seems to me that more data won't significantly benefit performance, since in this case the curves reach a plateau very early as the training set size grows.*" Since the testing curve has converged.

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.

Like your comment of "*We need graphs like those above to make absolute comparisons between different hyperparametrizations and choose the best trade off between bias and variance.*" As anytime we can plot this the better. Just sometimes is difficult when we are optimizing multiple parameters.

- As a max_depth of 1 suffers from high bias, visually this is due to the low training and validation scores(also note that they are close together). As this model is not complex enough to learn the structure of the data
- And a max_depth of 10 suffers from high variance, since we see a large gap between the training and validation scores, as we are basically just memorizing our training data and will not generalize well to new unseen data

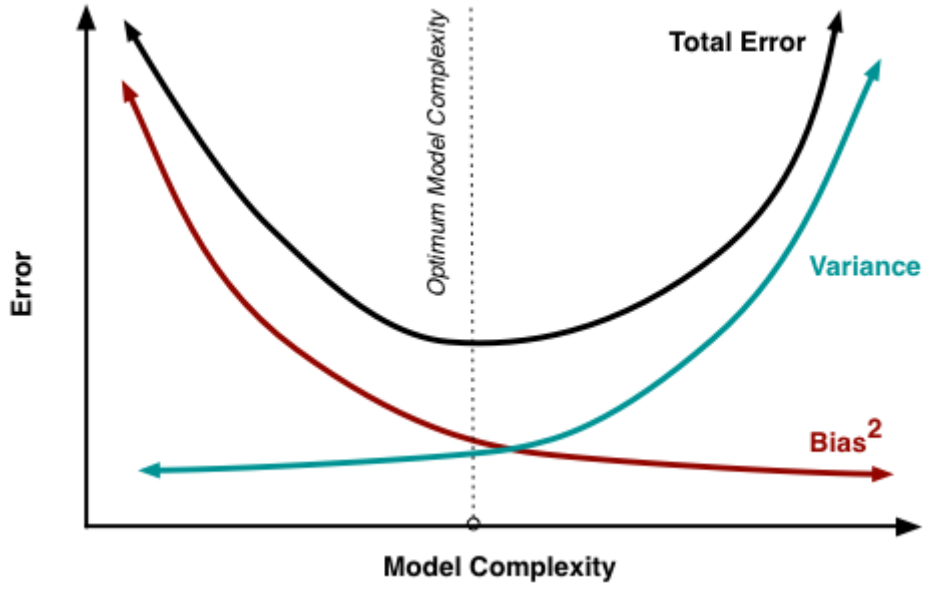
Could also discuss if more features are necessary for a high bias or high variance model?



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

" *The model that has the best score on the test data is the one that better generalizes to unseen data.*"

Good. As we(and GridSearch) are looking for the highest validation. And you are correct that we also need to consider the bias / variance tradeoff as well.



Evaluating Model Performance

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

Correct as this is an exhaustive search through a parameter space. Because this "*technique will automatically test all the combinations of them*", one limitation of GridSearch is that it can be very computationally expensive when dealing with a large number of different hyperparameter and much bigger datasets. Therefore there are two other techniques that we could explore to validate our hyperparameter

- RandomizedSearchCV which can sample a given number of candidates from a parameter space with a specified distribution. Which performs surprisingly well!
- Or a train / validation / test split, and we can validate our model on the validation set. Often used with much bigger datasets

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.

Great job! And spot on with your comment of "*In this way the result of Grid Search is not affected by an arbitrary choice of a single validation set.*" As if we use single validation set and perform grid search then it is the chance that we just select the best parameters for that specific validation set. But using k-fold we perform grid search on various validation set so we select best parameter for generalize case.

Note: Good with "*The original test data is preserved to verify the performance of the best model on unseen data.*" As many people don't realize this, and just use all the data, resulting in data leakage.

Student correctly implements the `fit_model` function in code.

This is perfect! Just note that we can also use the self-created scoring function as well

```
scoring_fnc = make_scorer(performance_metric)
```

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

Good. As gridSearch searches for the highest validation scores on the different data splits.

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.

Love your analysis here by comparing the features and housing prices of the dataset to your predictions. Just remember to keep in mind the testing error.

We can also plot a histogram of all of the housing prices in this dataset and see where each of these predictions fall

```
import matplotlib.pyplot as plt
plt.hist(prices, bins = 30)
for price in reg.predict(client_data):
    plt.axvline(price, c = 'r', lw = 3)
```

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

Would completely agree here, as this dataset is quite old and probably doesn't capture enough about housing features to be considered robust!

RESUBMIT

DOWNLOAD PROJECT

Learn the best practices for revising and resubmitting your 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