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

2 SPECIFICATIONS REQUIRE CHANGES

This is a very impressive submission. Just need a couple minor adjustments and you will be golden, but also check out some of the other ideas presented in this review. One tip here would be that some of these topics are extremely important as you embark on your journey throughout your Machine Learning career and it will be well worth your time to get a great grasp on these topics before you dive deeper in. Keep up the hard work!!

Data Exploration

✓

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

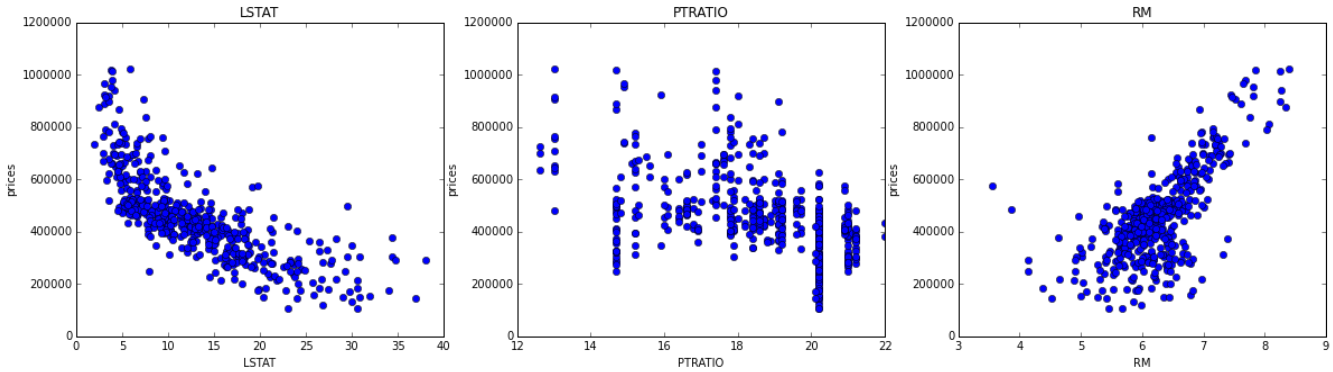
Good job utilizing the power of Numpy!! Always important to get a basic understanding of our dataset before diving in. As we now know that a "dumb" classifier that only predicts the mean would predict $454,342.94 for all houses.

✓

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')
```



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 justification here. Would recommend also mentioning the optimal score of 1. Could also think about if more data points would allow us to be more confident in this model?

R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression. The definition of R-squared is fairly straight-forward; it is the percentage of the response variable variation that is explained by a linear model. Or:

- R-squared = Explained variation / Total variation

R-squared is always between 0 and 100%:

- 0% indicates that the model explains none of the variability of the response data around its mean.
- 100% indicates that the model explains all the variability of the response data around its mean.

In general, the higher the R-squared, the better the model fits your data. So with a high value of 92.3% (0.923) we can clearly 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.

Excellent! Since we need a way to determine how well our model is doing! As we can get a good estimate of our generalization accuracy on this testing dataset. Since our main goal is to accurately predict on new unseen data. Also that we can try and protect against overfitting with this independent dataset.

If you would like to learn some more ideas in why we need to split our data and what to avoid, such as data leakage, check out these lectures

- https://classroom.udacity.com/courses/ud730/lessons/6370362152/concepts/63798118300923
- https://classroom.udacity.com/courses/ud730/lessons/6370362152/concepts/63798118310923

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.

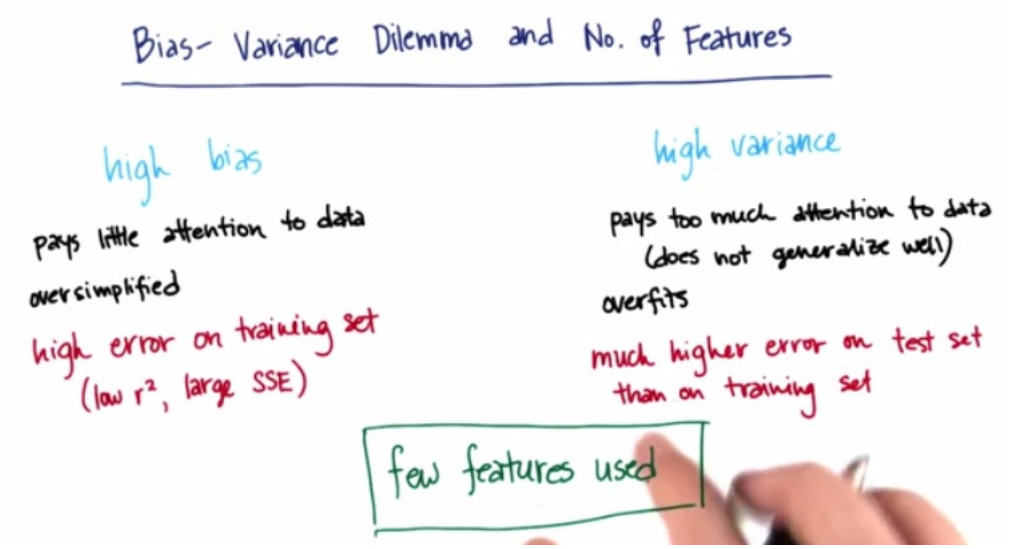
You are correct with your comment of "*given that this convergence is asymptotic, adding more training data beyond a certain point won't benefit the model...and in fact, the additional data could decrease model peformance due to overfitting*" At the end if we look at the testing curve here, we can clearly see that it has started to converge to / diverge from its optimal score, so more data is not necessary. Therefore lastly for this section make sure you also describe the initial trends of the training and testing curves, before they converge(increasing or decreasing)?

✓

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.

Solid ideas here! You clearly understand the bias/variance tradeoff. I would recommend expanding a bit more in term of your visual justification for these

- As a max_depth of 1 suffers from high bias, visually this is due to the low training and validation scores(also note that it has low variance since the scores 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



✓

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

Spot on with your ideas! Either 3 or 4 are acceptable

- As a max depth of 4 might have a higher validation score(which is what gridSearch searches for)
- But you are correct that maybe a max depth of 3 has a better bias / variance tradeoff(with closer training and validation scores), also a simpler model, which is what is recommend based on Occam's razor

Check out this visual, it refers to error, but same can be applied to accuracy(just flipped)

Evaluating Model Performance

✓

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

Very nice! As you can see that we are using max depth in this project!

"*exhaustively testing lots of different combinations can be computationally expensive, which is one reason some people prefer 'randomized search' over 'grid search.*"

Great intuition here. You are correct that one limitation of GridSearch is that it can be very computationally expensive when dealing with a large number of different hyperparameters and much bigger datasets. Therefore there are two other techniques that we could explore to validate our hyperparameters

- 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.

You are spot on regarding the benefits when used with grid search when optimizing a model with your comment of "*in the process of evaluating different sets of hyper-parameters, knowledge about the test set can "leak" into the model and cause it to overfit.*" As this allows for multiple testing datasets and is not just reliant on the particular subset of partitioned data. For example, 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.

Even though I am sure that you know this, the reason this is marked as *Requires Changes* is that make sure you also mention exactly how many 'folds' are used for the testing sets in these K models?

Links

- (https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation)
- Video
- (https://www.cs.cmu.edu/~schneide/tut5/node42.html)

✓

Student correctly implements the `fit_model` function in code.

Looks good! Could also look into using the `range` command

```
params = {'max_depth':range(1, 11)}
```

✓

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

Can note that GridSearch searches for the highest validation score on the different data splits. So maybe 4 was a bit higher in the graph above.

✓

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.

Good simple justification for these predictions by comparing them to the descriptive stats of the features. Just remember to keep in mind the testing error here.

```
reg.score(X_test, y_test)
```

Pro Tip: 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
for i,price in enumerate(reg.predict(client_data)):
    plt.hist(prices, bins = 30)
    plt.axvline(price, lw = 3)
    plt.text(price-50000, 50, 'Client '+str(i+1), rotation=90)
```

✓

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

Very nice ideas. Would agree, 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.

RETURN TO PATH

Student FAQ