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

5 SPECIFICATIONS REQUIRE CHANGES

This is a very impressive submission. Just need some 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

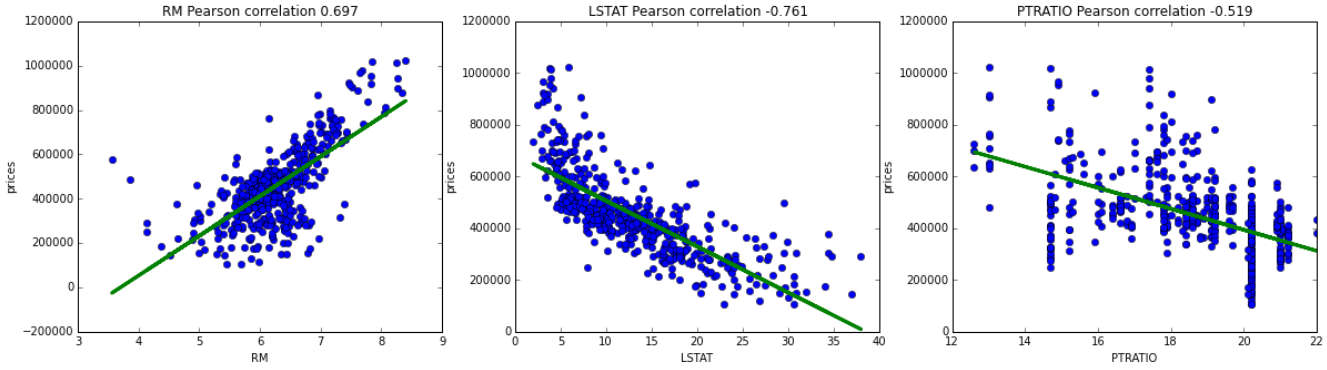
**All requested statistics for the Boston Housing dataset are accurately calculated. Student correctly leverages NumPy functionality to obtain these results.**

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+1)
    plt.plot(data[col], prices, 'o')
    fit = np.polyfit(data[col], prices, 1)
    plt.plot(data[col], data[col] * fit[0] + fit[1], lw=3)
    plt.title(col + ' Pearson correlation ' + str(np.round(np.corrcoef(data[col], prices)[1][0], 3)))
    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.**

> "that 92.3% of the target variable can be predicted from the features(Good model)."

Should actually read that "92.3% of the **"variance"** in the target variable can be predicted from the features." R2 Score isn't about exactly predicting the values, but R2 score is the proportion of the 'variance' in the dependent variable that is predictable from the independent variable.

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.

**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.**

Great reasons! 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. And correct 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.**

Great analysis of the training and testing curves here. As in the initial phases, 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).

---

> "the model doesn't benefit from adding more training points beyond TP=300"

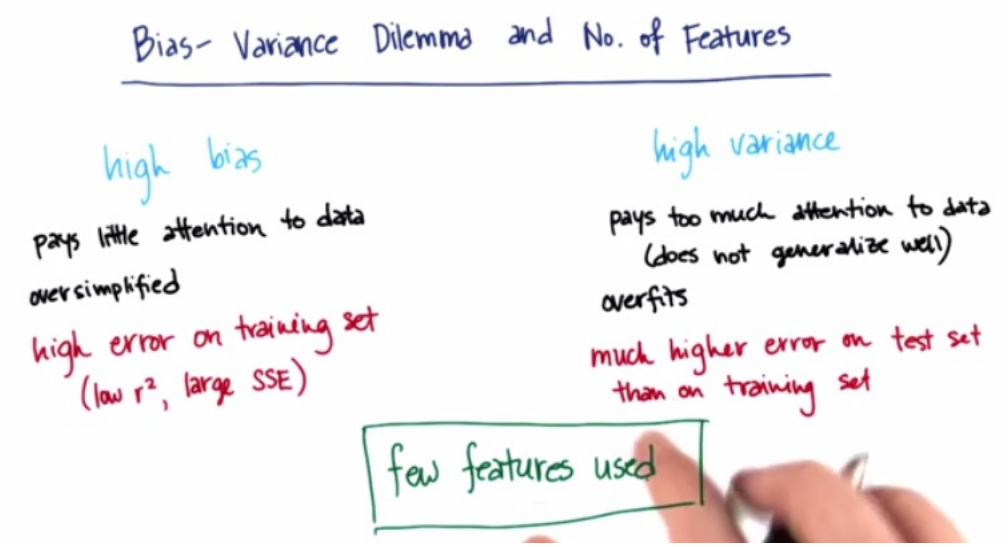
Correct! As in the beginning it is beneficial, but at the end if we look at the testing curve here, we can clearly see that it has converged to its optimal score, so more data is not necessary.

Also note that in practice collecting more data can often be time consuming and/or expensive, so when we can avoid having to collect more data the better. Therefore sometimes receiving very minor increases in performance is not beneficial, which is why plotting these curves can be very critical at times.

**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.**

Please note that it is the **low training score** that truly depicts high bias.

- As a max_depth of 1 suffers from high bias, visually this is due to the low training score(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

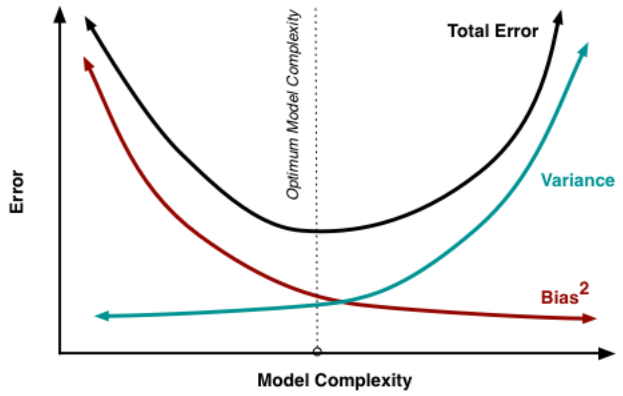



**Student picks a best-guess optimal model with reasonable justification using the model complexity graph.**

> "because it has the highest validation score"

Exactly! As we are definitely looking for the highest validation score(which is what gridSearch searches for). And we are also looking for a good bias / variance tradeoff(with close training and validation scores while the training score is high).

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.**

You are correct that gridSearch is a parameter optimization technique. Therefore lastly for this section please also mention which hyper-parameter value combinations does it test (i.e. a random sample of them, every other combination, all of the exhaustively)? As there are many different search types. What does `GridSearchCV` use?

Links

- (http://scikit-learn.org/stable/modules/grid_search.html)
- (https://en.wikipedia.org/wiki/Hyperparameter_optimization#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.**

> "benefits for grid search: -->with k-fold CV, all data is used for training and also for testing only once."

Remember when optimizing a model with gridSearch, we only run k-fold cross-validation on the **training dataset**. As we still need to split the data into a training and testing set, since we still need that last testing set for the final model evaluation. You can see that we are using `fit_model(X_train, y_train)` in this project. When training a model to evaluate different hyperparameters, the testing set should never be used to train the model.

Therefore we don't need to break off a piece of our training dataset solely for validation purposes, which is why this is ideal with smaller datasets. It is common to use a train/validation/test split when tuning machine learning models to prevent overfitting on the test dataset. But setting aside a validation set reduces the number of samples which can be used for training, which is why we often run k-fold cross-validation on the **training dataset** to keep as much training data as possible.

Links

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

But other than that, nice ideas!

**Student correctly implements the `fit_model` function in code.**

Nice implementation! Could also set a `random_state` in your DecisionTreeRegressor for reproducible results.

```
regressor = DecisionTreeRegressor(random_state = "any number")
```

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

Congrats! GridSearch searches for the highest validation score on the different data splits in this ShuffleSplit.

**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 justification for Client 2 and Client 3, just make sure you also give some brief ideas in why Client 1's prediction is reasonable given its features.

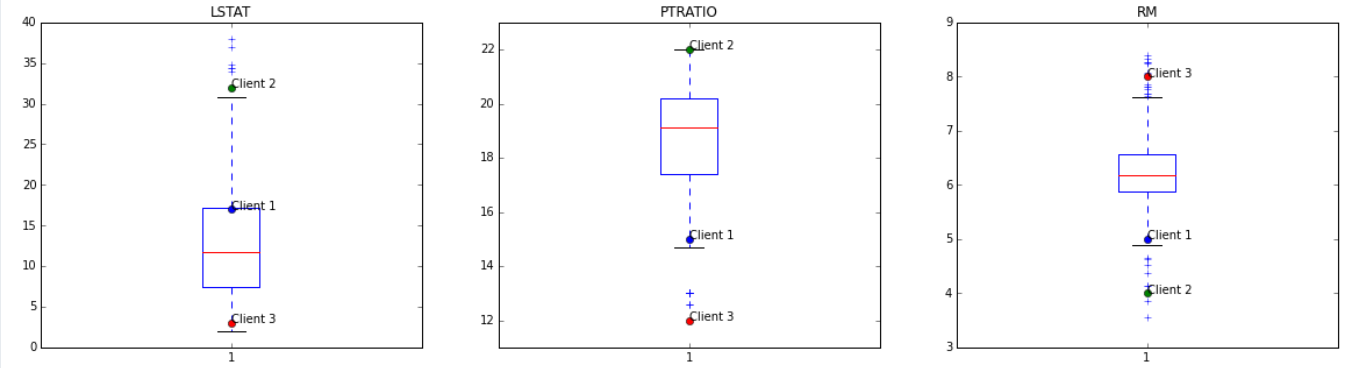
**Optional**: A more advanced and great idea would be to compare these to the descriptive stats of the features. We can compute the five number summary of the descriptive stats of the features with

```
features.describe()
```

For example -- Client 3 RM value is in the top 25 percentile in the data, while being near the minimum value for LSTAT, and since an increase in RM would lead to an increase in MEDV. An increase in LSTAT would lead to a decrease in MEDV. The predicted price near the maximum value in the dataset makes sense.

Maybe also plot some box plots and see how the Client's features compare to the interquartile range, median and whiskers

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20, 5))
y_ax = [[3,9],[0,40],[11,23]]
for i, col in enumerate(features.columns):
    plt.subplot(1, 3, i+1)
    plt.boxplot(data[col])
    plt.title(col)
    for j in range(3):
        plt.plot(1, client_data[j][i], marker="o")
        plt.annotate('Client '+str(j+1), xy=(1,client_data[j][i]))
        plt.ylim(y_ax[i])
```




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

You have good ideas here(nice job answering the questions). Therefore lastly for this section make sure you also explicitly mention if the constructed model should or should not be used in a real-world setting?

RESUBMIT

DOWNLOAD PROJECT


## Best practices for your project resubmission

Ben shares 5 helpful tips to get you through revising and resubmitting your project.

Watch Video (3:01)

RETURN TO PATH