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

6 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

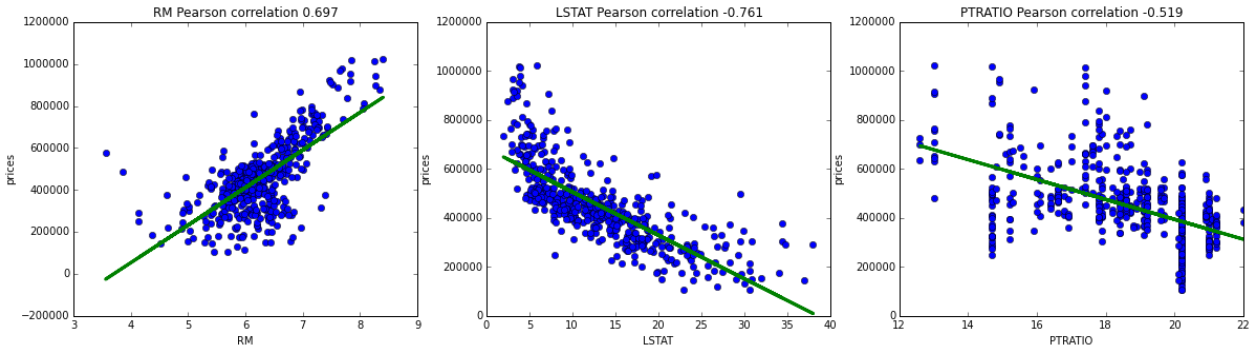
All requested statistics for the Boston Housing dataset are accurately calculated. Student correctly leverages NumPy functionality to obtain these results.

Although these are the correct ideas, for this section you should leverage NumPy functionality to obtain all these results. Therefore you are actually using basic python commands to receive the `minimum_price` and `maximum_price` statistics in your implementation. Therefore I bet numpy has the same methods!!

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.

> "The R^2 score seems to indicate that the model is fairly predictable, but intuitively, from the data, it doesn't feel like it's a close fit."

This comment is a bit unclear. Remember that this is just a hypothetical model. Therefore solely based on this model R2 score of 0.923, would you consider this model to have successfully captured the variation of the target variable? Why or why not? How does this score compare to the optimal score?

(https://en.wikipedia.org/wiki/Coefficient_of_determination)

Code Note: We can also check out the linear relationship between the 'True Values' and 'Predictions' with a Seaborn plot

```
import seaborn as sns
sample_df = pd.DataFrame([3, -0.5, 2, 7, 4.2], [2.5, 0.0, 2.1, 7.8, 5.3]).r
eset_index()
sample_df.columns = ['True Value', 'Prediction']
sns.regplot('True Value', 'Prediction', sample_df)
```

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.

> "first, that if there isn't enough data to train the model on, it will be underfitted"

We can have underfitting or overfitting if the training data is small(as it depends on the type of model). As you can see that a decision tree overfits the training data with a small training set(shown in the next section).

Therefore just focus on *why* we actually need a testing set. What does this testing set represent? What can we try and 'protect against' or 'detect' being able to test it with an independent dataset? etc...

Links

- Lecture
- (http://info.salford-systems.com/blog/bid/337783/Why-Data-Scientists-Split-Data-into-Train-and-Test)

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).

> "Around 300 training points, the two scores really start to converge. After about 400 training points, more training data seems to have strongly diminishing returns."

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.

> "At a maximum depth of 10, the training score is high, but the validation score is decreasing, which is a symptom of overfitting."

Just make sure you mention what a max depth of 10 suffers from in terms of high bias or high variance. What does "overfitting" refer to?

Links

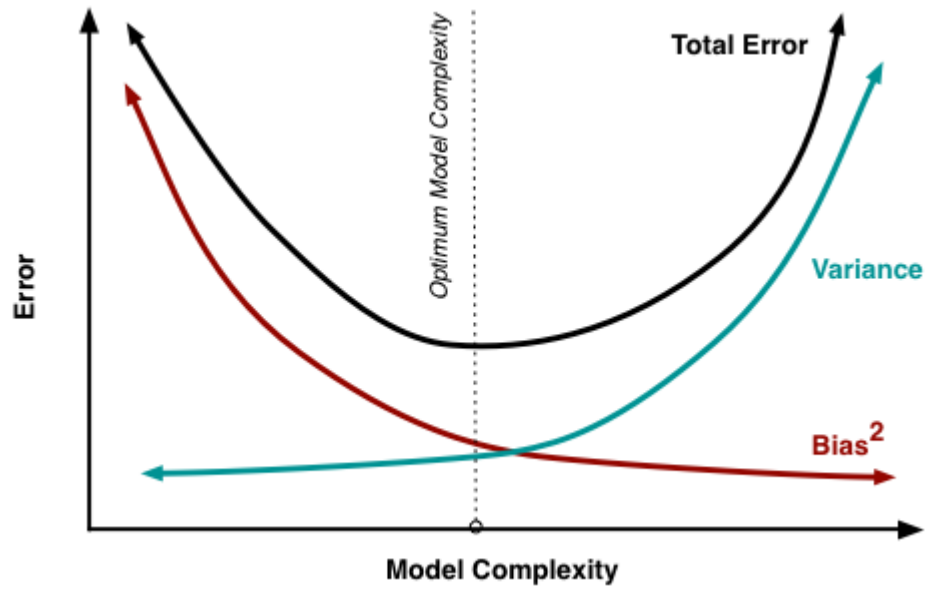
- http://scott.fortmann-roe.com/docs/BiasVariance.html
- https://insidebigdata.com/2014/10/22/ask-data-scientist-bias-vs-variance-tradeoff/
- http://machinelearningmastery.com/gentle-introduction-to-the-bias-variance-trade-off-in-machine-learning/

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

> " the highest validation score is at max_depth=4"

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). And the less complex one is definitely recommended 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.

> "For each cell in this matrix, the F1 score is computed and the cell with the highest F1 score is picked."

First off, just want to make you aware, that we are not restricted to using F1 score as our evaluation metric when optimizing a model with gridSearch, as we can use any evaluation metric we please. As you can see that we are using max depth, a decision tree and **r squared** score in this project. Here might be a list of common evaluation metrics.

Also please 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.

> "In k-fold cross-validation, the training data is broken up into n buckets; training is done n times, using n-1 buckets for training and the unused bucket for cross-validation."

Just make sure you also mention the last step. How we get that one last final score from k-fold cross-validation? What do we do with all the error rates at the end of the K-Training? As this is a key part of the k-fold cross-validation technique and why it is beneficial for parameter tuning.

> "then use F1 scores of the remaining (CV) bucket as the value for the grid search cell."

As mentioned above, we are not restricted to using F1 score as our evaluation metric when optimizing a model with gridSearch

---

> "If you use the full training set to pick a model, there's the possibility of overfitting the data (because thou shalt never use your testing data to make a decision about the model)."

This comment is a bit unclear. Why would we "overfit the data" if you use the full training set to pick a model? Can you expand on this comment.

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.

This is an extremely important concept in machine learning, as this allows for multiple validation 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 there 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 sets so we select best hyperparameter for generalization. Cross-validation better estimates the volatility by giving you the average error rate and will better represent generalization error.

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.

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.

Can note that GridSearch searches for the highest validation score on the different data splits in this ShuffleSplit. So 5 was a bit higher in these splits.

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.

Excellent justification for these predictions by comparing them to the features and descriptive stats of the housing prices. This is always an important step in any machine learning project.

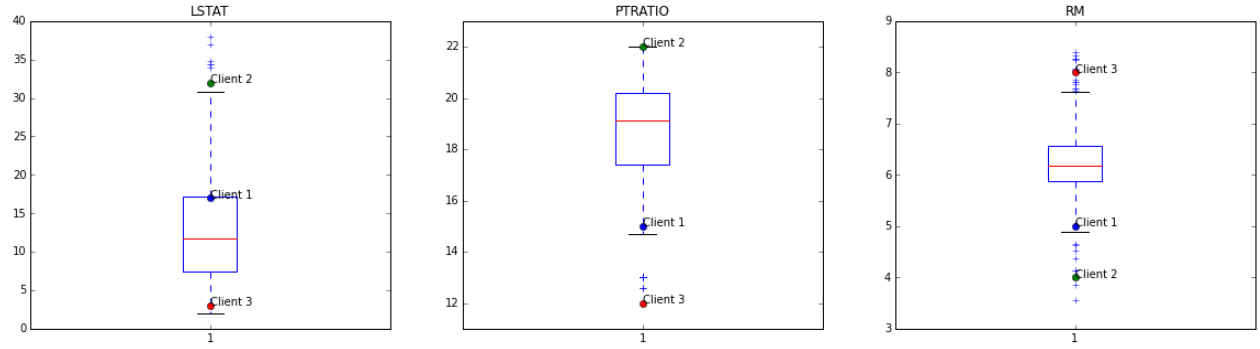
A more advanced and great idea would be to compare these to the descriptive stats of the features. We can compute the five number summary of the descriptive stats of the features with

```
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])
```



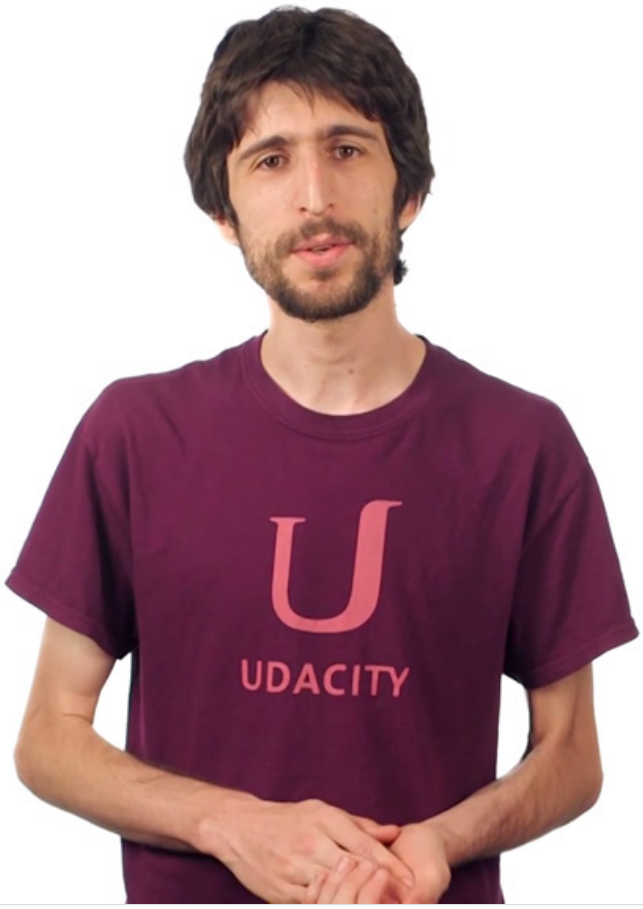
Student thoroughly discusses whether the model should or should not be used in a real-world setting.

Would agree. This dataset is quite old, probably doesn't capture enough about housing features, and the range in predictions are quite large. Therefore this model would not be considered robust!

If you would like to learn more about how to analyze the uncertainty in the output of a mathematical model, can check out these ideas (https://en.wikipedia.org/wiki/Sensitivity_analysis)

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

Rate this review

Student FAQ