UDACITY

PROJECT

Predicting Boston Housing Prices

A part of the Machine Learning Engineer Nanodegree Program

PROJECT REVIEW

CODE REVIEW

NOTES

SHARE YOUR ACCOMPLISHMENT!

Meets Specifications

Quality of Code

Student's code runs successfully and produces results similar to those in the report. No modifications are made to the template code beyond what is requested without justification.

Statistical Analysis & Data Exploration

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

Well done for making use of NumPy library getting the result correctly.

Student adequately describes three separate features of the dataset. The corresponding values in

the client's feature set are correctly identified for the chosen features.

- Impressive work here for the feature selection.
- Please look at the code as following which would be getting the corresponding values in the client's feature set in a programmable manner:

```
chosen_features = ['LSTAT', 'RM', 'PTRATIO']
features = city_data.feature_names.tolist()
for feature in chosen_features:
  index = features.index(feature)
  print CLIENT_FEATURES[0][index]
```

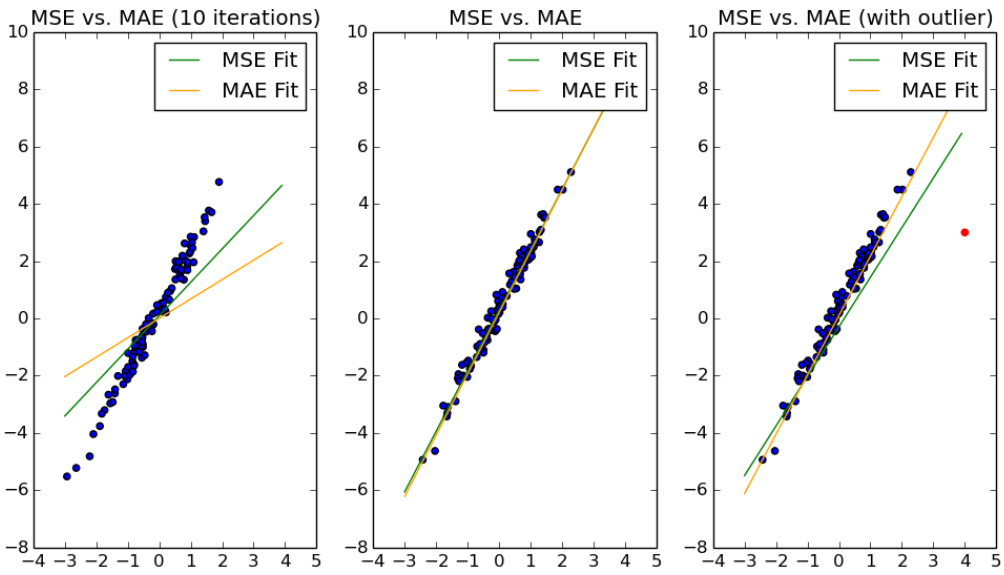
Evaluating Model Performance

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.

- Well done for making use of the train_test_split and specify the test_size as 0.3 and supply random_state, since the random_state is similar to the seed in the random function to ensure that each time the split is reproducible.
- It's great that you have briefly mentioned about the benefit of splitting the dataset into training and testing subset - provide two independent dataset to give estimate of performance on an independent dataset; it also serves as check on overfitting.

An appropriate performance metric is chosen with thorough justification. The metric is correctly implemented in code.

Both MSE and MAE are appropriate. There are instances however where one error metric can help us achieve better performance. Below is an example where a linear model has been trained on some synthetic data by means of stochastic gradient descent based on both MSE and MAE:



Note that MSE converges faster to the solution while MAE is more robust to outliers. Notice that although MSE punishes larger errors more heavily than MAE they both guide the model to almost the same solution if no outliers are present and enough iterations are performed.

Student correctly describes the grid search algorithm and briefly discusses its application.
GridSearchCV is properly implemented in code.

- Similar to what you have included regarding to grid search algorithm which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set
- It would be worth mentioning about fine tuning a learning algorithm for a more successful learning/testing performance in terms of the application for grid search.

Student correctly describes how cross-validation is performed on a model, and why it is helpful when using grid search.
Modifications beyond the default 3-fold cross-validation for GridSearchCV are reasonably justified.

- More on the k-fold cross validation: - the data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other k-1 subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set k-1 times
- It maximize the dataset for training and testing, so that the data we can use to provide best learning result and best validation - this is extremely useful when the dataset is limited in size as

Grid Search will allow an extensive exploitation of available data.

- As you mentioned in the report, CV helps with grid search to ensure that overfitting would not happend. Kindly note that if we limit grid search to single testing set, we may accidentally over fit our model if the split testing set is imbalanced. Using cross validation the parameters will be optimized on the entire data set and any random anomalies due to random splitting will be removed.

Analyzing Model Performance

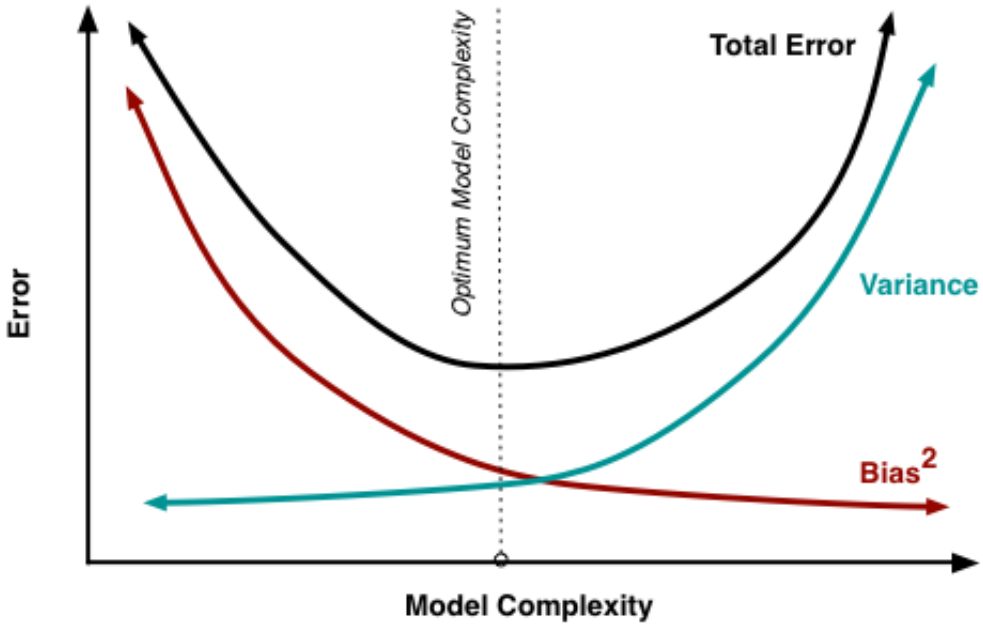
Student correctly identifies significant qualities of the training and testing errors as the training set size increases.

- There are two phases to the error rates - the testing error approaches some form of an asymptote and begins perturbing while the training error continues to increase.
- To give more context, when the training set is small, the trained model can essentially “memorize” all of the training data. As the training set gets larger, the model won’t be able to fit all of the training data exactly.
- The opposite is happening with the test set. When the training set is small, then it’s more likely the model hasn’t seen similar data before. As the training set gets larger, it becomes more likely that the model has seen similar data before.

Student provides analysis for both a max depth of 1 and a max depth of 10. Reasonable justification is given for each graph if the model suffers from high bias or high variance.

Student identifies how the training and error curves relate to increasing the model’s complexity.

Please look at the following diagram, which clearly shows how the testing and training error evolves with the increasing model complexity



- Please note that with the increasing model complexity, the model goes through two stages from underfitting to overfitting - Please consider to include this in your report
- The first phase is where the model is underfitted and the training error is exceedingly high.
- The second phase is where the model is overfitted and the difference between testing and training error is high.

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

The optimal model is where the turning point at, which the training error is low and testing error is at global minimum.

Model Prediction

Student determines the optimal model from parameter tuning and compares this model to the one they chose.

Student's model produces a valid result. The predicted selling price is adequately justified by the calculated descriptive statistics.

In the report, I suggest mentioning that the predicted value is between one standard deviation from the

mean.

- Another way to justify the prediction is to find similar data points and compare their housing price. Sklearn has a nearest neighbor module that makes it easy to find similar data points: http://scikit-learn.org/stable/modules/neighbors.html

Student thoroughly discusses justification for or against using their model for predicting future selling prices.

It would be better that could consider the following questions:

- Would additional data points (or the inclusion of data per year) benefit the model? Please look at the dataset for when the data is collected.
- Is there a possibility of outliers in the data that can drastically change predictive results?
- Does this dataset feature enough characteristics about homes to be considered robust?
- Does performing grid search on the entire dataset affect your confidence in the model?

Moreover, it's impressive work to consider the principle component to reduce the dimensionality

DOWNLOAD PROJECT

Have a question about your review? Email us at review-support@udacity.com and include the link to this review.

RETURN TO PATH