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

4 SPECIFICATIONS REQUIRE CHANGES

Almost there, excellent work so far!
👏👏👏

I can see you put a lot of effort in your project.

Just a few improvements and you'll be good to go, but first let's make sure you have a firm grasp of the concepts presented here.

Keep going and good luck!
Paul

PS. If you have further questions you can find me on Slack as `@viadanna`

Data Exploration

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

Good job calculating statistics, this exploration is very important as a starting point to understand the dataset.

Just make sure to use numpy's equivalents for all statistics as required by this rubric.

Using `numpy` is recommended as it's a very fast and efficient library commonly used in machine learning projects. Check this reference on statistical functions included.

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

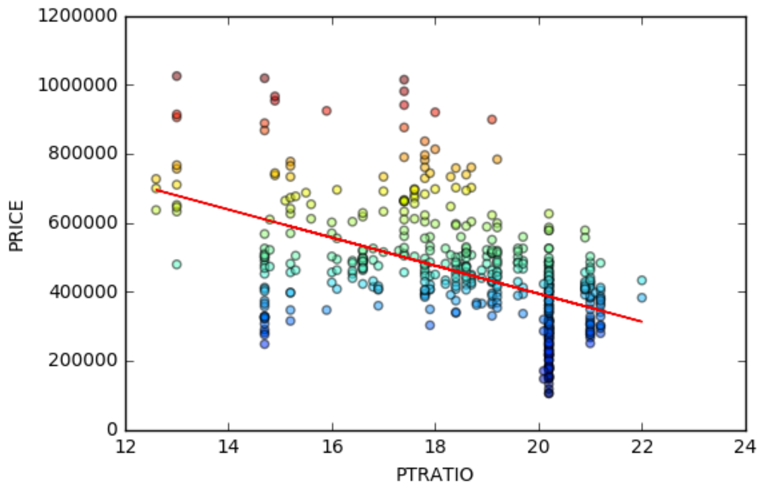
Nice intuition on the correlation between MEDV and the features.

This intuition is essential in machine learning projects to evaluate if results are reasonable.

An alternative to explore this is plotting a linear regression.

```python
from sklearn.linear_model import LinearRegression

reg = LinearRegression()
pt_ratio = data['PTRATIO'].reshape(-1, 1)
reg.fit(pt_ratio, prices)
plt.plot(pt_ratio, reg.predict(pt_ratio), color='red', linewidth=1)
plt.scatter(pt_ratio, prices ,alpha=0.5, c=prices)
plt.xlabel('PTRATIO')
plt.ylabel('PRICE')
plt.show()
```



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.

Great answer!

Metrics like this are important since the datasets found on Machine Learning projects rarely can be evaluated visually as this example.

It is possible to plot the values to get a visual representation in this scenario:

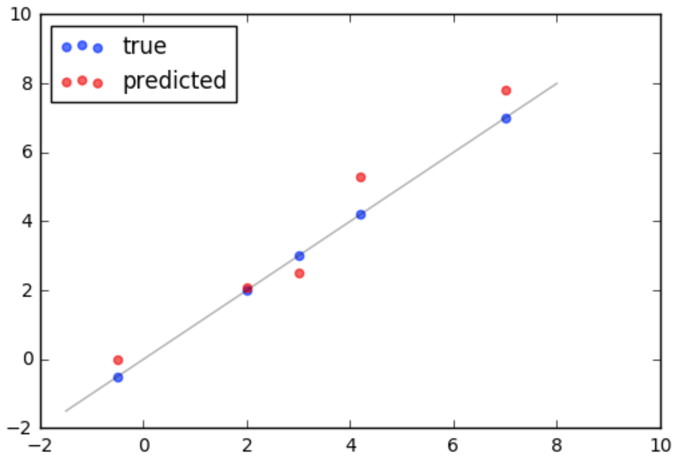
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

true, pred = [3, -0.5, 2, 7, 4.2], [2.5, 0.0, 2.1, 7.8, 5.3]
# Plot true values
true_handle = plt.scatter(true, true, alpha=0.6, color='blue', label='true')

# Reference line
fit = np.poly1d(np.polyfit(true, true, 1))
lims = np.linspace(min(true) - 1, max(true) + 1)
plt.plot(lims, fit(lims), alpha=0.3, color='black')

# Plot predicted values
pred_handle = plt.scatter(true, pred, alpha=0.6, color='red', label='predicted')

# Legend and show
plt.legend(handles=[true_handle, pred_handle], loc='upper left')
plt.show()
```



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.

Good answer!

It's really important to have an independent dataset to validate a model and identify its ability to generalize predictions.

Remember that this alone doesn't prevent overfitting, but allows us to identify it.

Check out this article for an interesting further reading on this subject.

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.

Good job, there's little benefit in increasing the size of training dataset further.

Notice that it's easy for any model to memorize a few training points, that's why there's a large gap between training and testing accuracy with a small training size. As more points are added, it's increasingly difficult for the model to memorize all data points, and it starts to learn how to generalize instead.

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.

Nice answer, you correctly identified high bias and high variance.

In short, high bias means the model isn't complex enough to learn the patterns in the data, resulting in low performance on both the training and validation datasets.

On the other hand, high variance means the model starts to learn random noise, losing its capacity to generalize previously unseen data, resulting in a very high training score but low testing score.

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

Good choice! I'd choose `max_depth = 4` as well for the highest validation score.

Evaluating Model Performance

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

Nice description of GridSearchCV, just a small confusion on *In grid search, we take parameters as columns (with each value as one column) and hyperparameters as rows (each different value as one row).*

Remember that the grid is created by cartesian product, generating all hyperparameters combinations.

For further information I suggest checking the following resources:

- Course
- http://scikit-learn.org/stable/modules/grid_search.html#exhaustive-grid-search
- http://machinelearningmastery.com/how-to-tune-algorithm-parameters-with-scikit-learn/
- 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.

Almost there!

Your description of how the k-fold technique works is correct.

For the benefit think what could happen to the hyperparameters selected by grid search if a simple train/validation split was used. Can you guarantee that the selected model would be the one that best generalizes the whole dataset?

Further information can be found on:

- https://en.wikipedia.org/wiki/Cross-validation_(statistics)#k-fold_cross-validation
- http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
- https://www.cs.cmu.edu/~schneide/tut5/node42.html
- http://stats.stackexchange.com/a/104750

Student correctly implements the `fit_model` function in code.

Nice implementation 👍

Just a suggestion, you can instead use

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

as `range(a, b)` returns an integer list where `a <= x < b`

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

Well done!

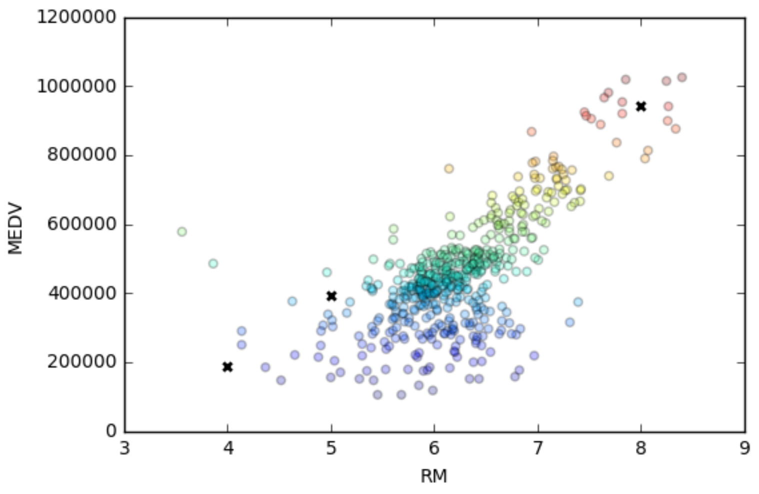
Student reports the predicted selling price for the three clients listed in the provided table. Discussion is made for each of the three predictions as to whether these prices are reasonable given the data and the earlier calculated descriptive statistics.

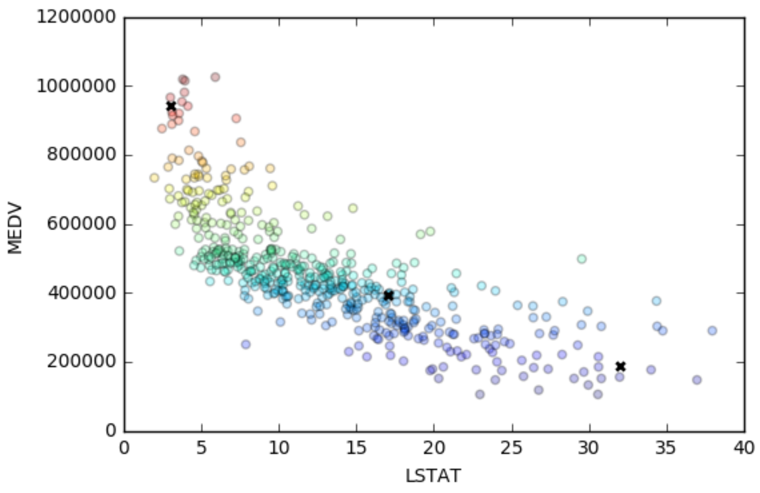
That's a good start, but it's still missing a justification based on the features and statistics as required. For instance, why is it reasonable that the house for Client 2 is priced close to the minimum?

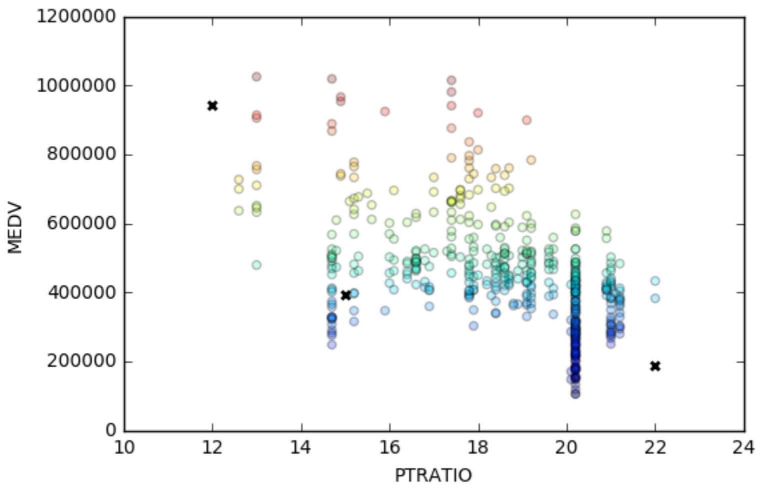
Here are some plots of client data to help you, but remember to add a written justification if you decide to add these plots to your answer:

```
from matplotlib import pyplot as plt

clients = np.transpose(client_data)
pred = reg.predict(client_data)
for i, feat in enumerate(['RM', 'LSTAT', 'PTRATIO']):
    plt.scatter(features[feat], prices, alpha=0.25, c=prices)
    plt.scatter(clients[i], pred, color='black', marker='x', linewidths=2)
    plt.xlabel(feat)
    plt.ylabel('MEDV')
    plt.show()
```







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

You have good arguments that show this model shouldn't be trusted in a real-world setting, lacking robustness, features and updated data.

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