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 understanding the dataset.

The notebook states the `numpy` library must be used here to get you acquainted, as it's very fast, efficient and commonly used in machine learning projects. It's also the foundation of pandas actually. ***Since numpy has already been imported for you, use this library to perform the necessary calculations.***

Check this reference on statistical functions included.

Besides, `pandas.std` results will be different from `numpy.std` as their default degrees of freedom isn't the same.

Notice that the MEDV column has already been queried to the `prices` variable.

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

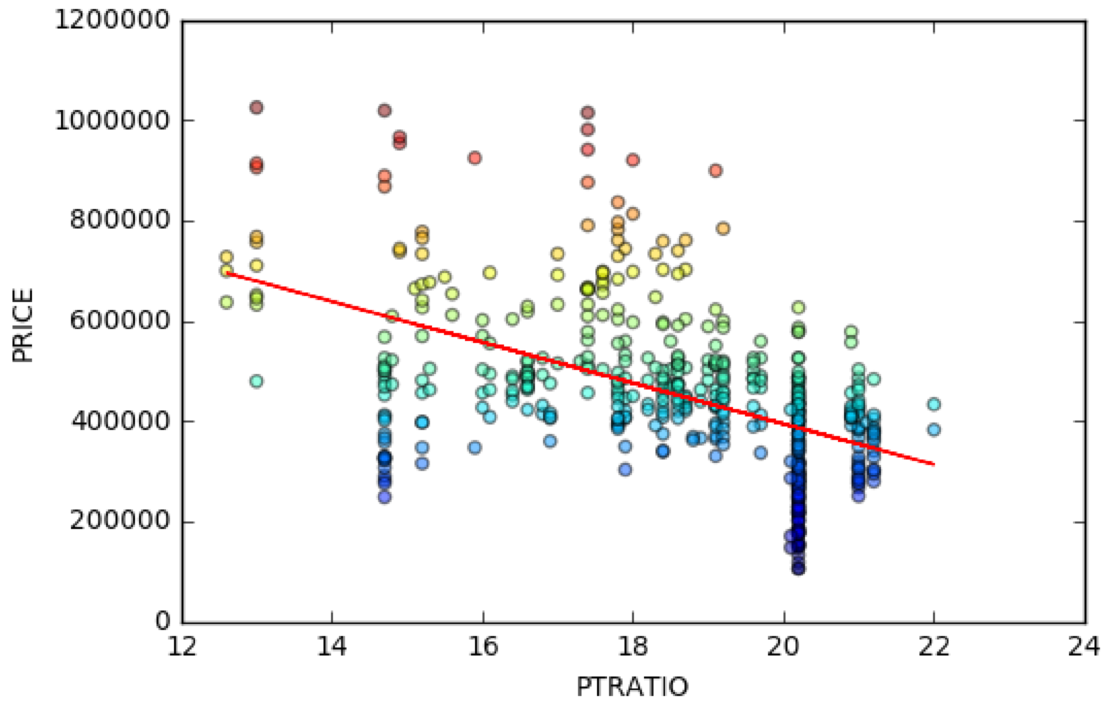
Nice intuition on the correlation between MEDV and the features.

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

An alternative to explore this is plotting a linear regression.

```
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:

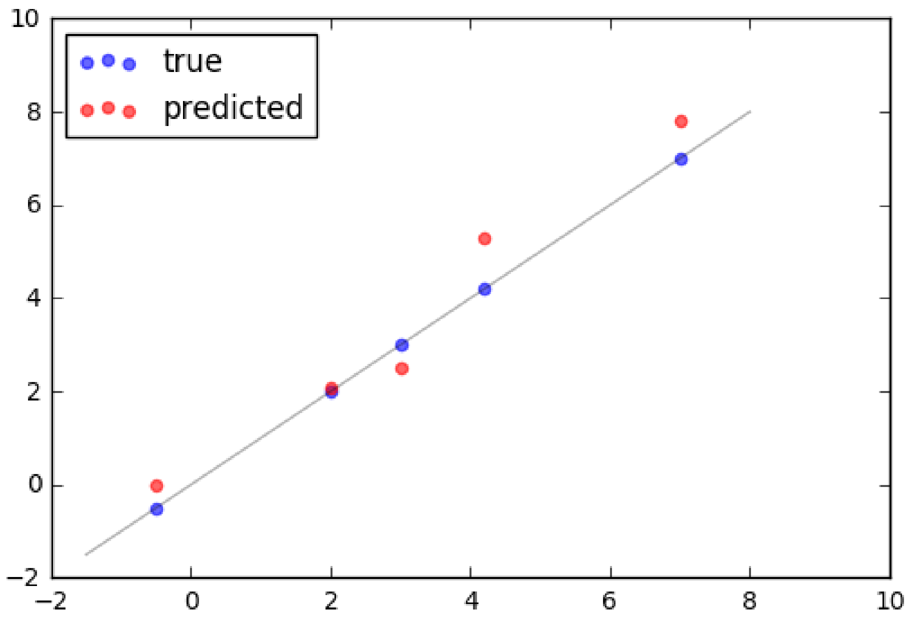
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

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.

You are on the right path. To get to the specific reason why the dataset is split into training and testing, try answering:

- What could happen if you test your model with the same data used to train it?

You can check the following resource for further information:

- http://machinelearningmastery.com/a-simple-intuition-for-overfitting/

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.

Just remember to add a description of the behaviour of both curves. Are those ascending or descending? Are they converging?

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.

Great 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 the curve for `max_depth = 4` as well for the highest validation score.

The curve also shows a high validation score for `max_depth = 5`, but I'd still pick the above considering Occam's Razor which applied here basically means that for two or more equivalent answers, the simplest one is the correct one.

Evaluating Model Performance

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

Nice description of GridSearch!

Just notice that GridSearchCV might use kfold cross-validation, but not necessarily.

In short, GridSearch exhaustively searches for a combination of hyperparameters resulting in a model with the best performance.

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. For completeness, try answering:

- How many iterations are there and how one differs from the others?

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.

Excellent implementation 👍

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

Well done!

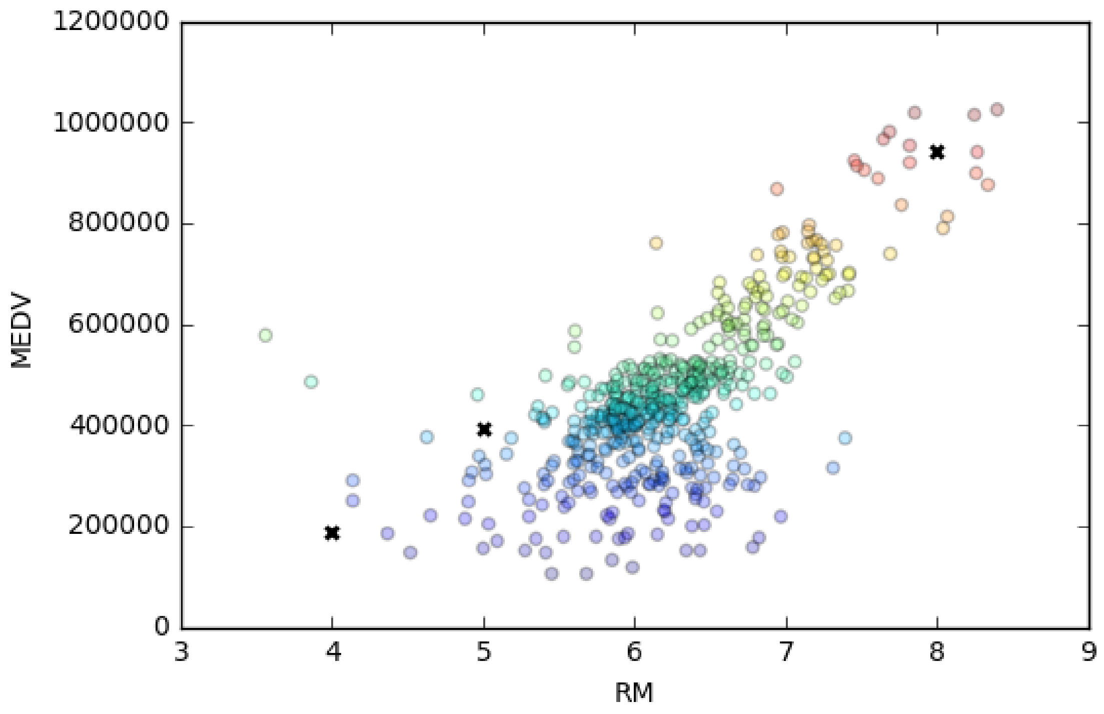
Student reports the predicted selling price for the three clients listed in the provided table. Discussion is made for each of the three predictions as to whether these prices are reasonable given the data and the earlier calculated descriptive statistics.

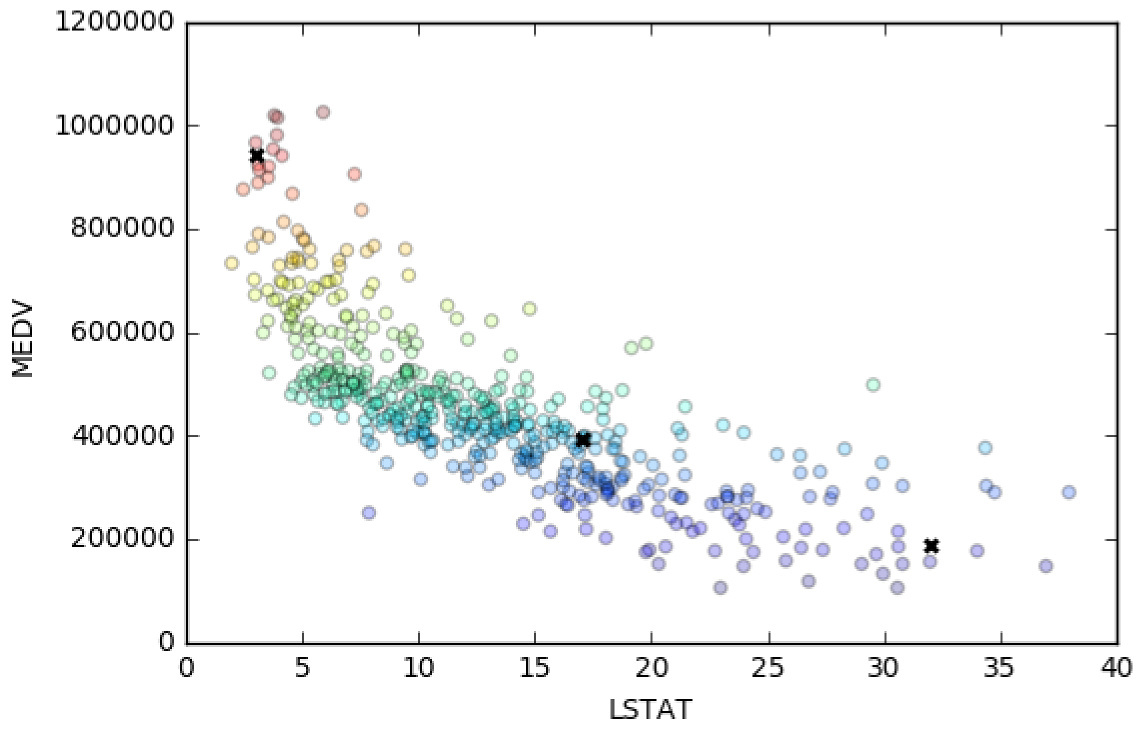
Awesome work using the features and statistics to justify your answer!

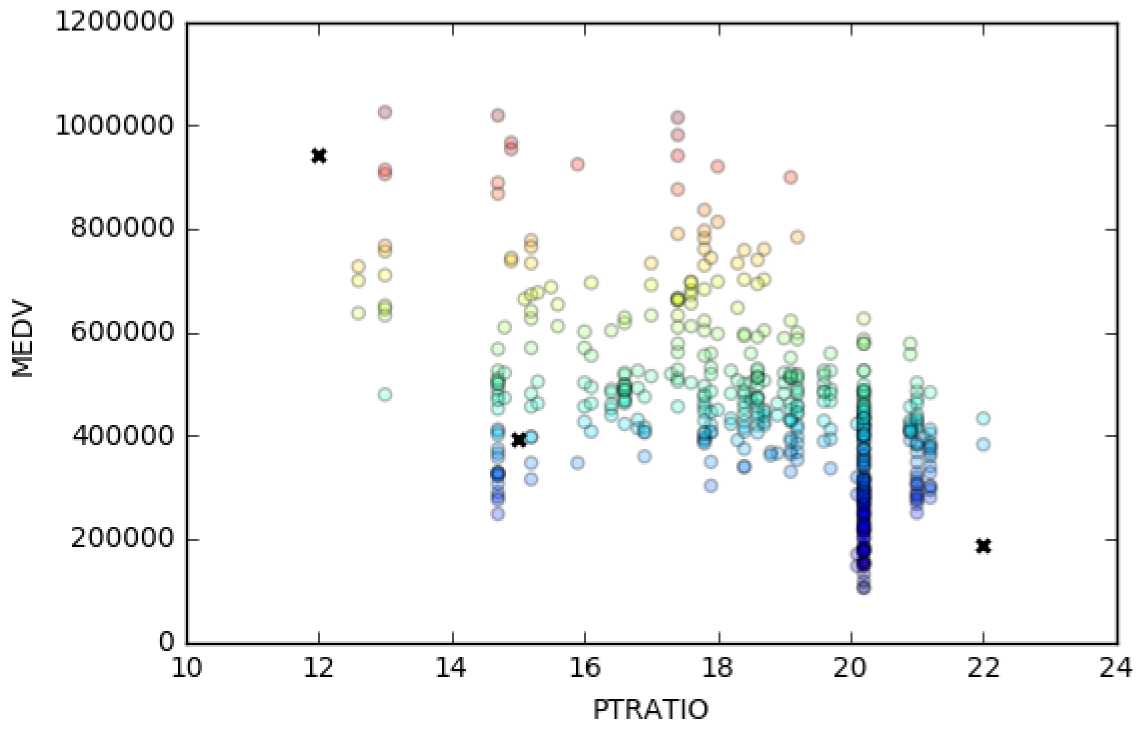
You could also plot the client features against the dataset to get a better view:

```
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 features and updated data.

I just missed the ***Range in prices*** shown in ***Sensitivity*** in your discussion of the robustness of this model.

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

Student FAQ