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 and with just a few improvements you'll be good to go. But first let's make sure you have a firm grasp on the concepts presented here that are fundamental to supervised learning projects.

Keep going and good luck!
Paul

PS. If you have questions you can find us on Slack channel `#ml-model-validation`. If you are not part of the Slack community yet you can request an invite here.

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.

Also 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.

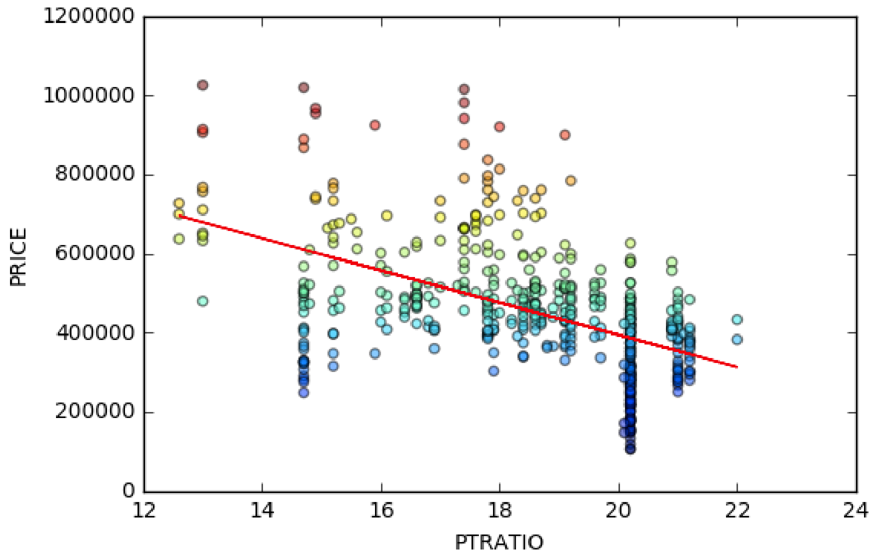
Student correctly justifies how each feature correlates with an increase or decrease in the target variable.

Excellent intuition on the correlation between MEDV and the features. This intuition is essential in machine learning projects to evaluate if results are reasonable.

Another way to explore this is graphically with a linear regression, as you will learn next.

```
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.

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

A correlation coefficient (R2) of 0.923 mean 92.3% of the variation of the dependent variable can be explained by the model. If you are interested in further reading, check this article.

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

Just check your code, as you are not using the train_test_split function to split the dataset and instead are attributing these values to the subdatasets. And remember to use the features variable as X and not data.

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.

It seems you forgot your answer to this question.

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.

Correct! You correctly identified high bias and high variance.

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

High variance means the model starts to learn random noise in the data, losing it's 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 GridSearch!

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

Just make sure you understand the *exhaustively* part, training and validating a model for every combination of hyperparameters available, because this is computationally intensive and can be unfeasible.

If you are curious check the alternative RandomSearch.

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.

The rubric states you must describe how k-fold works. Try answering the following:

- How does the technique divides the dataset?
- How many folds are used for training and testing?
- How many iterations are there and how one differs from the others?

You should also explain the main benefit of using it along GridSearch. What can happen to GridSearch's optimal model if a simple train-test split is used instead of this technique?

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.

Good start, but your implementation of the GridSearch does not use the `cv_sets` cross-validation sets which you created earlier.

You should add `cv=cv_sets` to the parameters of `GridSearchCV` to use it. Check the documentation for more information.

This will actually impact all the next answers, so I recommend that after you fix this, you restart the notebook using the menu `Kernel > Restart and Run All` and review your results and answers on the next questions.

And a suggestion, instead of

```
params = {'max_depth': [1,2,3,4,5,6,7,8,9,10]}
```

you can instead code

```
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.

Good job!

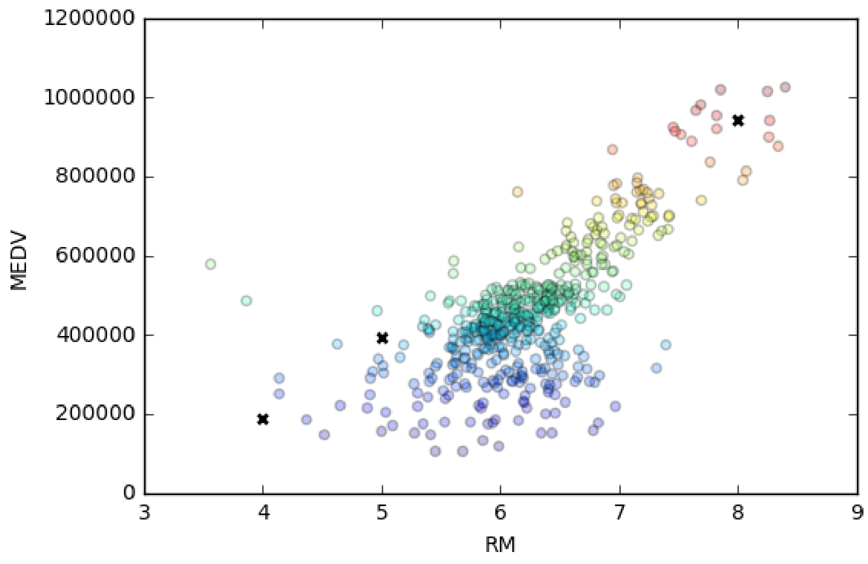
Student reports the predicted selling price for the three clients listed in the provided table. Discussion is made as to whether these prices are reasonable given the data and the earlier calculated descriptive statistics.

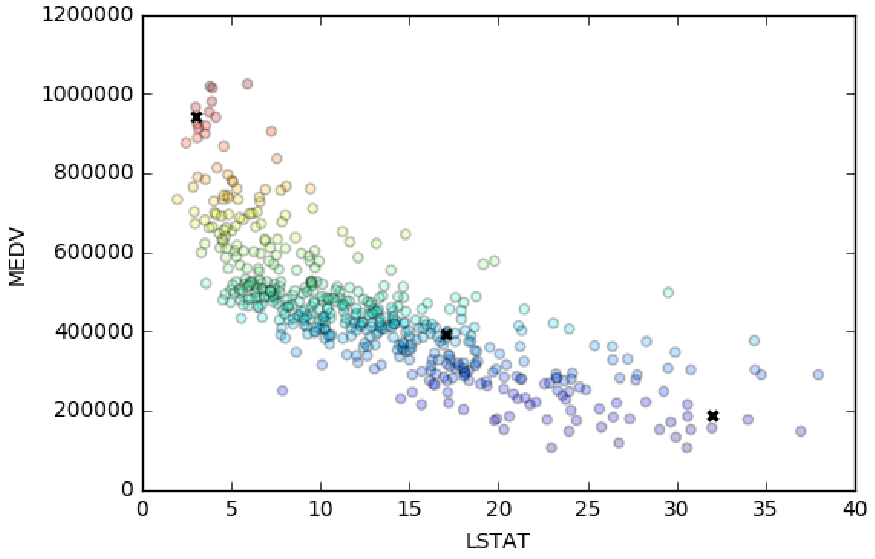
Excellent work using the statistics and features to justify your answer.

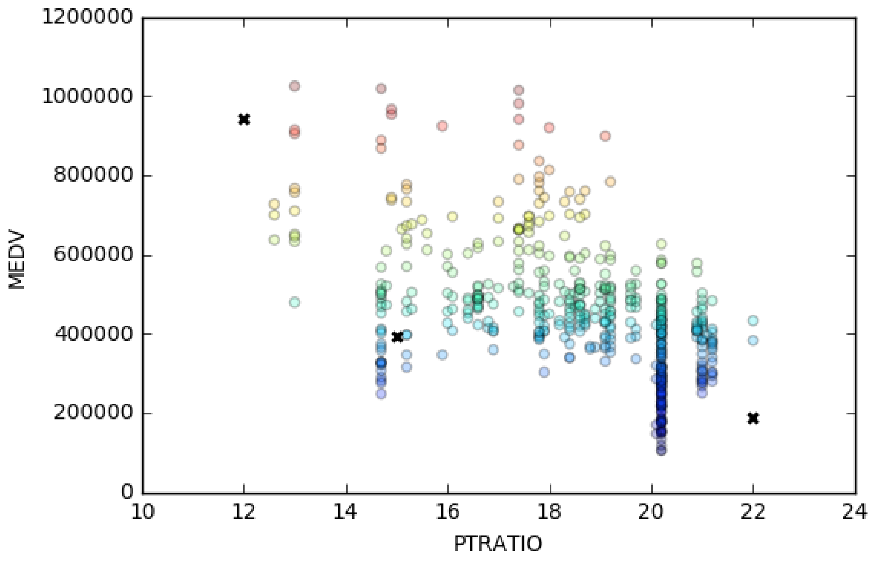
You could also plot the client data against the dataset to get a better view of their features:

```
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 not to use this model in a real-world setting.

Remember to rerun the Sensitivity section after fixing GridSearchCV.

Oh, and what seems a topographic mistake the dataset is 38 years old, not 28.

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)

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

RETURN TO PATH

Student FAQ