1) Statistical Analysis and Data Exploration

NUMBER OF DATA POINTS (HOUSES)?
506

NUMBER OF FEATURES?
13

MINIMUM AND MAXIMUM HOUSING PRICES?
minVal: 5.0
maxVal: 50.0

MEAN AND MEDIAN BOSTON HOUSING PRICES?
mean: 22.5328063241
median: 21.2

STANDARD DEVIATION?
9.18801154528

2) Evaluating Model Performance

WHICH MEASURE OF MODEL PERFORMANCE IS BEST TO USE FOR PREDICTING BOSTON HOUSING DATA AND ANALYZING THE ERRORS? WHY DO YOU THINK THIS MEASUREMENT MOST APPROPRIATE? WHY MIGHT THE OTHER MEASUREMENTS NOT BE APPROPRIATE HERE?
I used a regression metric, because we want to predict continuous data and the regression metric does not care if we predicted it to 100% correctly. Furthermore I used mean squared error metric. I implemented it in both absolute and squared error metric, however I came to the result, that the mean squared error metric had better results. Some benefits of mean squared error metric are that there always are positive values, which does not have an effect at this example, and there are larger differences, so the learning by errors will be more precise. At this example we definitely want to have better results by the learning of errors so thats another reason why I chose squared over absolute.
Classification does not work in this case, because we do not have any categories where we can fit the data.

WHY IS IT IMPORTANT TO SPLIT THE BOSTON HOUSING DATA INTO TRAINING AND TESTING DATA? WHAT HAPPENS IF YOU DO NOT DO THIS?
In machine learning it is important to split up the data into training and testing data to train a model and validate this model. We need the training data to train the model. The testing data is used to validate unseen data. The validation through the testing data estimates the performance of the algorithm and show how well the model fits.
If we do not split the data in training and testing data, we will run into issues concerning the model, because the model does know all data and therefore it can not say how well the model fits. The model will be overfitted and would fail if it predict unseen data. Also we will have no clue about the performance of the algorithm.

Reviewer Comment:
It is not entirely correct to state: "The model will be overfitted and would fail if it predict unseen data." we are not automatically causing over-fitting if we are not splitting, it might be that our model is great or it might not be the case, the problem is that we wouldn't be able to assess if the model is over-fitting.

WHAT DOES GRID SEARCH DO AND WHY MIGHT YOU WANT TO USE IT?
Grid search tries multiple combinations of parameters and finds the best combination of the parameters which give the best performance. It does a huge impact to the performance with just some lines of code.

Reviewer Tip:
Pro tip: There are other techniques that could be used for hyperparameter optimization in order to save time like RandomizedSearchCV, in this case instead of exploring the whole parameter space just a fixed number of parameter settings is sampled from the specified distributions. This proves useful when we need to save time but is not necessary in cases in cases like ours where the data set is relatively small.

WHY IS CROSS VALIDATION USEFUL AND WHY MIGHT WE USE IT WITH GRID SEARCH?

Cross validation helps to prevent overfitting and decides which model gives the best performance. It defines a dataset on which the model is "tested" in the training phase. The training data is used to be split in k smaller sets. A model will be trained on k-1 of the training data. The trained model will be validated on the remaining parts of the data.
Cross validation helps grid search to evaluate the models which were trained. It used to perform model selection for the grid search.

Reviewer Comment:
Optional: It is true that using cross validation with grid search helps preventing over fitting, as hinted out in my previous review there is another element to consider: When a dataset is limited in size cross validation becomes extremely useful as it allows for an extensive exploitation of available data allowing assessing the real potential of our algorithm in terms of performance metrics.

3) Analyzing Model Performance

LOOK AT ALL LEARNING CURVE GRAPHS PROVIDED. WHAT IS THE GENERAL TREND OF TRAINING AND TESTING ERROR AS TRAINING SIZE INCREASES?
At the beginning the test error is much more higher than the training error. When the training size is a little bit higher than the training error and test error are more or less on the same height.
Every time when the depth increases the training error comes closer to an asymptote to the X axis. Also the training error does have an high increase when the training size becomes bigger. While the training error has a small increase, the test error is decreasing, which means, that the learning is done right.

LOOK AT THE LEARNING CURVES FOR THE DECISION TREE REGRESSOR WITH MAX DEPTH 1 AND 10 (FIRST AND LAST LEARNING CURVE GRAPHS). WHEN THE MODEL IS FULLY TRAINED DOES IT SUFFER FROM EITHER HIGH BIAS/UNDERFITTING OR HIGH VARIANCE/OVERFITTING?
When the max depth is 1, there is a high error level on the training error. The test error is also very high and it is more or less on the same level with the training. I think that it is just memorising from the training data instead of learning, since the two curves look very similar expect the beginning. So I guess that it suffers from high bias, so the model is underfitting. Also when the train and test errors are close each other and both are high, this is an underfitting symptom.
When the max depth is 10, the model does not suffer from high bias, because there it has a low error on the training set. So the model is not underfitting.
When the max depth is 10, the model suffers from high variance, because it has a much higher error on the test than on the training set. So the model is overfitting.

Reviewer:
The answer is correct, well done. Just a small nitpick when you write: "Also when the train and test errors are close each other and both are high, this is an underfitting symptom." As you correctly acknowledge before the problem is in the high error nothing the fact that the two errors are close.

LOOK AT THE MODEL COMPLEXITY GRAPH. HOW DO THE TRAINING AND TEST ERROR RELATE TO INCREASING MODEL COMPLEXITY? BASED ON THIS RELATIONSHIP, WHICH MODEL (MAX DEPTH) BEST GENERALIZES THE DATASET AND WHY?

When the depth increases, than the graphs of test error and training error get an higher distance between. At beginning they are close together and they have an high test and train error rate which is a symptom for underfitting. With every increase of the depth, the gap between the two graphs gets higher. When the gap between test and train error graph gets to high, than this indicates as a symptom of overfitting.
I think somewhere at 4/5 it does the best job, because I think that it is better for the model with the test and training error graph are not to closely to each other. The model should not suffer from underfitting or overfitting at this point.

Reviewer:
I think we are on the same page here: What matters here is that we try to find the point where the testing error is minimized. Normally after that point the testing error is flat to slightly increasing meaning that introducing further complexity is just causing over fitting. The training error continues to decrease but we don't bother about that.

4) Model Prediction

MODEL MAKES PREDICTED HOUSING PRICE WITH DETAILED MODEL PARAMETERS (MAX DEPTH) REPORTED USING GRID SEARCH. NOTE DUE TO THE SMALL RANDOMIZATION OF THE CODE IT IS RECOMMENDED TO RUN THE PROGRAM SEVERAL TIMES TO IDENTIFY THE MOST COMMON/ REASONABLE PRICE/MODEL COMPLEXITY.
COMPARE PREDICTION TO EARLIER STATISTICS AND MAKE A CASE IF YOU THINK IT IS A VALID MODEL.

```
GridSearchCV(cv=None, error_score='raise',
      estimator=DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
         max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
         min_weight_fraction_leaf=0.0, presort=False, random_state=None,
         splitter='best'),
      fit_params={}, iid=True, n_jobs=1,
      param_grid={'max_depth': (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)},
      pre_dispatch='2*n_jobs', refit=True,
      scoring=make_scorer(mean_squared_error, greater_is_better=False),
      verbose=0)
Best model parameter:  {'max_depth': 4}
Best estimator:  DecisionTreeRegressor(criterion='mse', max_depth=4, max_features=None,
          max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=2,
          min_weight_fraction_leaf=0.0, presort=False, random_state=None,
          splitter='best')
House: [11.95, 0.0, 18.1, 0, 0.659, 5.609, 90.0, 1.385, 24, 680.0, 20.2, 332.09, 12.13]
Prediction: [ 21.62974359]
Nearest Neighbors average: 21.52
```

My prediction was usually between 21 and 22. Since this is very close to the mean and median and the predicted housing price is from an average house then I think that the price is reasonable. Also the nearest neighbors is very close.