MACHINE LEARNING ENGINEER NANODEGREE

Project Report

P1: PREDICTING BOSTON HOUSING PRICES

Author: Andrejs Zujevs, Latvia, Riga, skype name: sholc2005, email: andrejs.zujevs@gmail.com

Table of Contents

1Data Exploration....1
2Evaluating Model Performance....2
2.1Performance Metric....2
2.2Testing/Training Split....4
2.3Gridsearch implemetation....4
2.4Cross Validation....5
3Analyzing Model Performance....5
3.1Learning Curves and Training Analysis....5
3.2Learning Curves and Bias & Variance Analysis....8
3.3Error Curves and Model Complexity....9
3.4Picking the Optimal Model....9
4Model Prediction....10
4.1Predicted Housing Price....10
4.2Comparing Model Price to Housing Statistics....11

1 Data Exploration

Boston data statistical analysis:

Data size: 506
Number of features 13
Price minimum value: 5.000
Price maximum value: 50.000
Price mean value: 22.533
Price median: 21.2
Price STD: 9.188
Skewness = 1.1048, kurtosis = 1.4686 (frequency of prices)

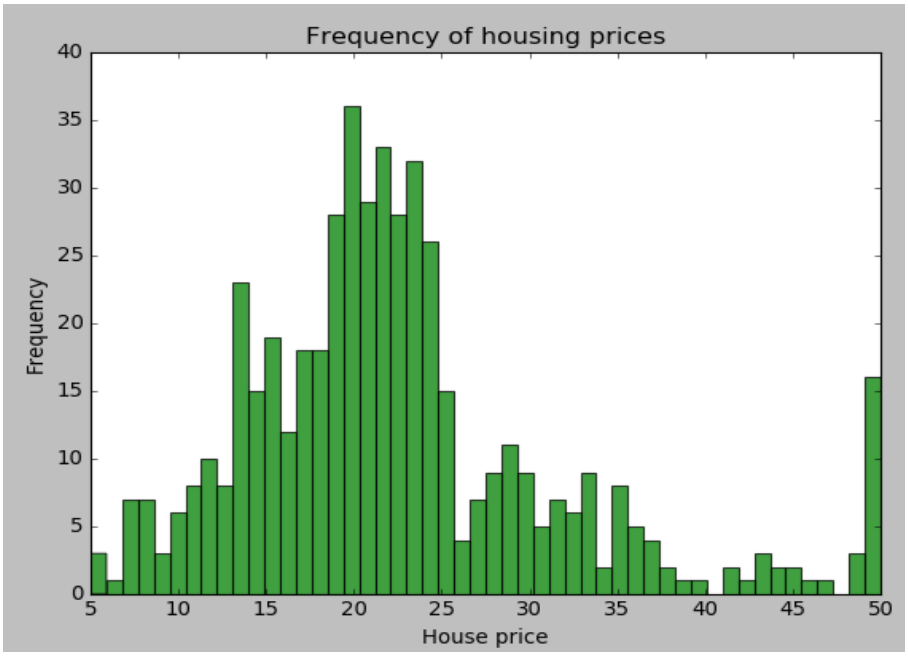


Figure 1. Frequency of housing prices

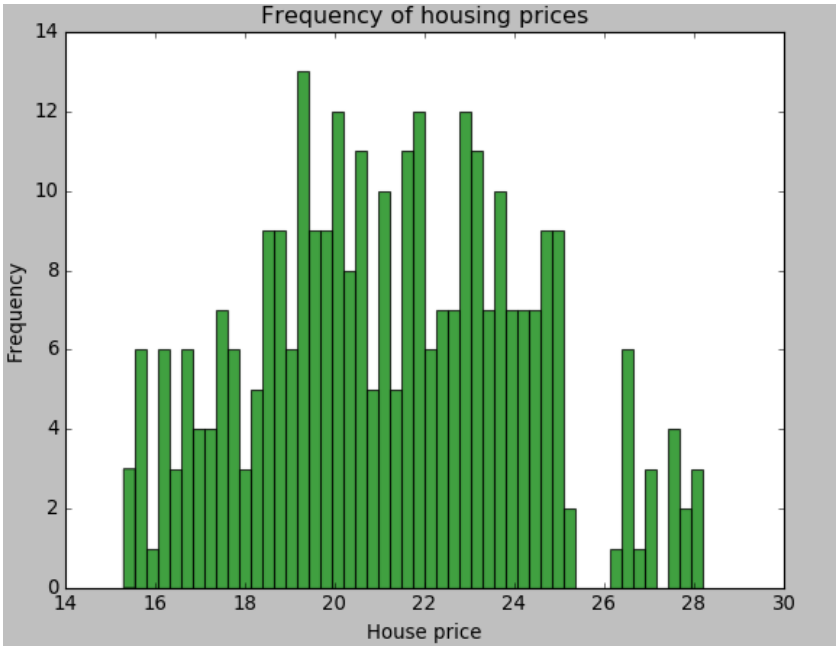


Figure 2. Frequency of housing prices without outliers (0.25)

2 Evaluating Model Performance

2.1 Performance Metric

For the P1 project the Mean Square Error (MSE) error scoring metric was used. The reasons of this are:

1. MSE is classic error measuring metric;
2. MSE has more mathematical properties than, for example, the Mean Absolute Error (MAE). MSE is differentiable and that allow for different methods (ANN, regression);
3. MSE is more sensitive for errors in data (difference between fact and predicted value) than MAE.

In fact, Fig.1 shows that houses in range 48-50K are more than 15 and that we need to take into account. This means that MSE is more precise for the model building. This is true also for the houses with price under 48K.

MSE implemented in the code, see `performance_metric` function.

MAE	**MSE**
 max_depth=1	 max_depth=1
 max_depth=3	 max_depth=3
	

max_depth=6	max_depth=6
	
max_depth=10	max_depth=10
MAE	**MSE**

In the graphs before MAE and MSE metric are very similar for Boston (with single random state for regressor). If we evaluate square root of MSE we would obtain value a little more than MAE value. So I choose MSE because DecisionTree regressor can find gradient or slope to minimize error.

2.2 Testing/Training Split

Boston data set was split into testing and training data sets. If we will use all data for training it would produce model overfitting problem. If a model will overfitted it become too complex and generally have poor performance. The overfitted model losing their generality and become very sensitive for errors in data.

The author was used `ShuffleSplit` (from sklearn.cross_validation) function for data split with 25% part of testing data. This function belongs to the holdout cross validation method.

See implementation in the `split_data` function.

2.3 Gridsearch implemetation

Foe GridSearch function we need to provide:

- **estimators.** In this case I provide regressor object which is DecisionTreeRegressor estim;
- **estimator parameters.** In the my case I provide 'max_depth':(1,2,3,4,5,6,7,8,9,10) for DecisionTreeRegressor and GridSearch check each depth level.
- **scoring function**. In my case I provide my_scorer object which was defined with function make_scorer with parameter greater_is_better=False (for reducing lose) and performance function: performance_metric (particularly MSE).
- **cv parameter (default 3-folds)**. Number of Cross-validation folds used in model fitting. I used 11 because Boston data not large and cross-validation with many fords gives more accurate results.

See implementation in the `fit_predict_model` function.
After GridSearch work we would take best parameter: max_depth of model. The GridSearch search best combination of estimator parameters. I my case the lowest value of performance metric. As

scoring function defined with parameter greater_is_better=False, then values of score function are negative, we only need minimal absolute value in my case.
Best model complexity level - is minimal value of scoring function. Generally a curve of model complexy is concave. The best level of model depth is lowest place in curve.

2.4 Cross Validation

Cross validation is a model evaluation method. The main idea is to split data to a training set and testing set (minimum of two folds - holdout method). After model training it will be tested of data that has not already seen. Holdout method has disadvantage – the model training and testing results much depends on individuals in a samples and this method provide high variance of error evaluation. To resolve such problem the K-fold method will used.
Cross validation is useful in GridSearch, where would be small variation of error score value and best parameter set is more robust for regression of classification function.
The author used CV=11 parameter in GridSearch that means using of 11 folds for model evaluation and mean error calculation. Instead of 3-folds (default) author using 11 folds – because Boston data size is relatively small. In experiments author obtained more robust and error less results.

3 Analyzing Model Performance

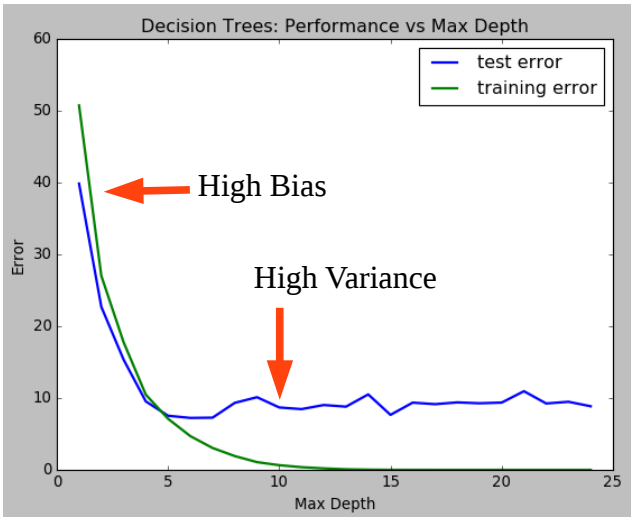
3.1 Learning Curves and Training Analysis

MSE permanence metric, data split and regressor with random state=1111	
	I think this is good model (max_depth=6). Where testing error has global minimum (in my case of data spiting and max_depth range) and where training error decreasing if we are looking on testing and training error in max_depth range figure.

Generally: with increasing training size the training error increasing and the testing error decreasing until they meet in the middle.

Then training size is small the training error is near zero due a model is too simplistic and well fits into the data, but increasing size of training set the training error increasing and opposite the testing error decreasing.

3.2 Learning Curves and Bias & Variance Analysis



The model with max_depth=1 is high biased (high error for both the training and the testing set) and model is under-fitted or too simplistic.
The high biased model is to simple (DecisionTree with 1 level) and not will ever be a good to fit to Boston data – the data is poorly fit.

The model with max_depth=10 is with high variance, where the training error is very low, while the testing error is high. The model is overfitted or too complex.
Overfitted model tends to fit a training data very well, hence low training error, while for testing data the model is too sensitive or no general, hence high testing error.

3.3 Error Curves and Model Complexity

max_depth=6

Increasing model complexity training and testing curves converge. Hence, variance decrease. With inceasing model complexity decrease bias. After the optimal level of the model complexity the error curve sloping to zero and large gap between curves become (high variance).

In this graph you can see correlation of testing and training errors.

Before max_depth=5 testing and training curves converge (variance decreasing), but after the 5-th model becoming overfitted with high variance and training error towards to zero.

Best model max_depth is near 6, where training and testing error reduced as possible together, but after 5 diverge. This means that model becoming overfitted.

3.4 Picking the Optimal Model

In this graph you can see correlation of testing and training errors.

Before max_depth=5 testing and training curves converge (earlier model is too general or model under-fitted), but after 5 the model becoming overfitted and very sensitive for an errors in testing set.

Best model is with max_depth 5, where training and testing error reduced as possible together, but after 5 diverge. This means that model becoming overfitted.

4 Model Prediction

4.1 Predicted Housing Price

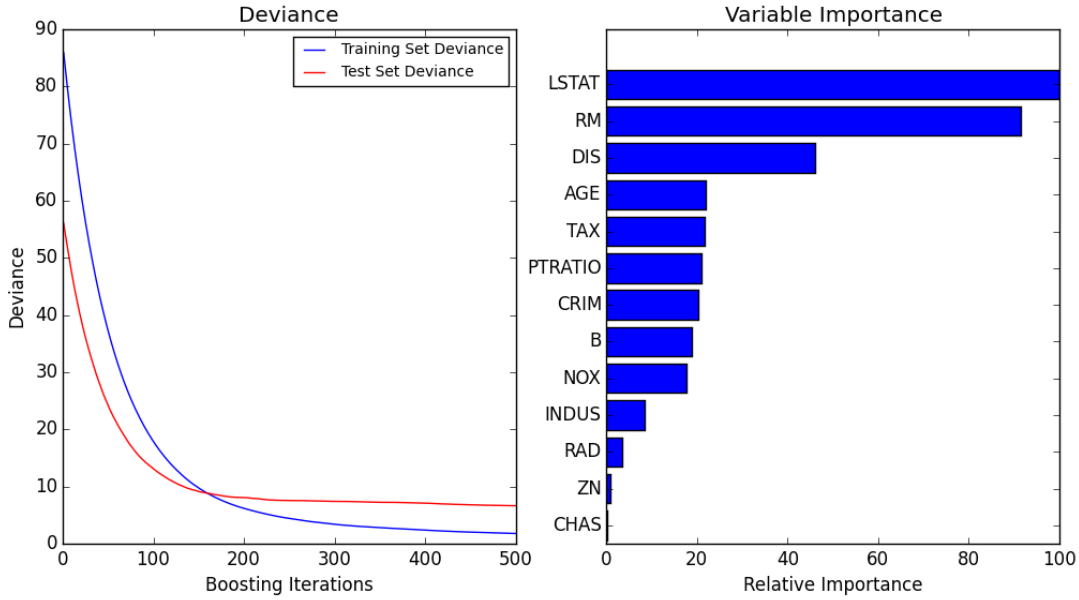
Grid search used with CV=11 (cross-validation folds).
For estimator were used such parameters:

- "max_depth':(4,5,6), I`m reduced
- 'max_features': [3,5,6,7,8,9,10] – looking on feature importance I conclude get futures number 3 to 10 (see graph below).

The outliers was reducing with range 5%.

Fitting model function I continuously run 10 times and after that choose particular run with minimal MSE.

See details below:



from: http://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_regression.html

The best score is 17.41 and if we compute square root then error is: +/-4.17K

DecisionTreeRegressor(criterion='mse', max_depth=6, max_features=7,
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')
Best parameters:
{'max_features': 7, 'max_depth': 6}
Details of SearchGridCV:
mean: -29.44738, std: 39.19877, params: {'max_features': 3, 'max_depth': 4}
mean: -23.63674, std: 20.45759, params: {'max_features': 5, 'max_depth': 4}
mean: -23.21252, std: 19.61036, params: {'max_features': 6, 'max_depth': 4}
mean: -21.55645, std: 18.66106, params: {'max_features': 7, 'max_depth': 4}

mean: -21.94794, std: 18.15543, params: {'max_features': 8, 'max_depth': 4}
mean: -18.87749, std: 13.84446, params: {'max_features': 9, 'max_depth': 4}
mean: -21.94349, std: 20.32550, params: {'max_features': 10, 'max_depth': 4}
mean: -31.11077, std: 31.33363, params: {'max_features': 3, 'max_depth': 5}
mean: -20.44386, std: 18.63846, params: {'max_features': 5, 'max_depth': 5}
mean: -23.31186, std: 23.87093, params: {'max_features': 6, 'max_depth': 5}
mean: -20.72869, std: 17.35518, params: {'max_features': 7, 'max_depth': 5}
mean: -22.69857, std: 20.38567, params: {'max_features': 8, 'max_depth': 5}
mean: -21.47821, std: 18.55807, params: {'max_features': 9, 'max_depth': 5}
mean: -19.94218, std: 18.05281, params: {'max_features': 10, 'max_depth': 5}
mean: -20.20094, std: 16.68052, params: {'max_features': 3, 'max_depth': 6}
mean: -23.69743, std: 22.13642, params: {'max_features': 5, 'max_depth': 6}
mean: -20.63111, std: 18.07963, params: {'max_features': 6, 'max_depth': 6}
mean: -17.41893, std: 10.53325, params: {'max_features': 7, 'max_depth': 6}
mean: -21.47617, std: 22.24779, params: {'max_features': 8, 'max_depth': 6}
mean: -19.83095, std: 13.30438, params: {'max_features': 9, 'max_depth': 6}
mean: -21.97624, std: 15.61239, params: {'max_features': 10, 'max_depth': 6}
Best score
-17.4189304172

And predicted with max_depth=6 and features_num=7
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: [21.35555556]

Using RandomizedSearchCV for best parameter estimation the little better results (by MSE) were obtained (multiple runs):
Best estimator:
DecisionTreeRegressor(criterion='mse', max_depth=5, max_features=10,
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')
Best parameters:
{'max_features': 10, 'max_depth': 5}
Details of SearchGridCV:
mean: -24.52015, std: 22.30312, params: {'max_features': 3, 'max_depth': 4}
mean: -24.61778, std: 18.79132, params: {'max_features': 3, 'max_depth': 6}
mean: -24.90807, std: 23.81418, params: {'max_features': 5, 'max_depth': 5}
mean: -18.24851, std: 11.21149, params: {'max_features': 9, 'max_depth': 5}
mean: -23.37192, std: 21.00552, params: {'max_features': 7, 'max_depth': 6}
mean: -17.36383, std: 12.60198, params: {'max_features': 10, 'max_depth': 5}
mean: -19.68312, std: 13.05250, params: {'max_features': 6, 'max_depth': 4}
mean: -21.72699, std: 20.56604, params: {'max_features': 9, 'max_depth': 4}
mean: -23.42656, std: 26.25868, params: {'max_features': 3, 'max_depth': 5}
mean: -19.89605, std: 19.01176, params: {'max_features': 10, 'max_depth': 4}
Best score
-17.3638306179

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: [20.4338843]

4.2 Comparing Model Price to Housing Statistics

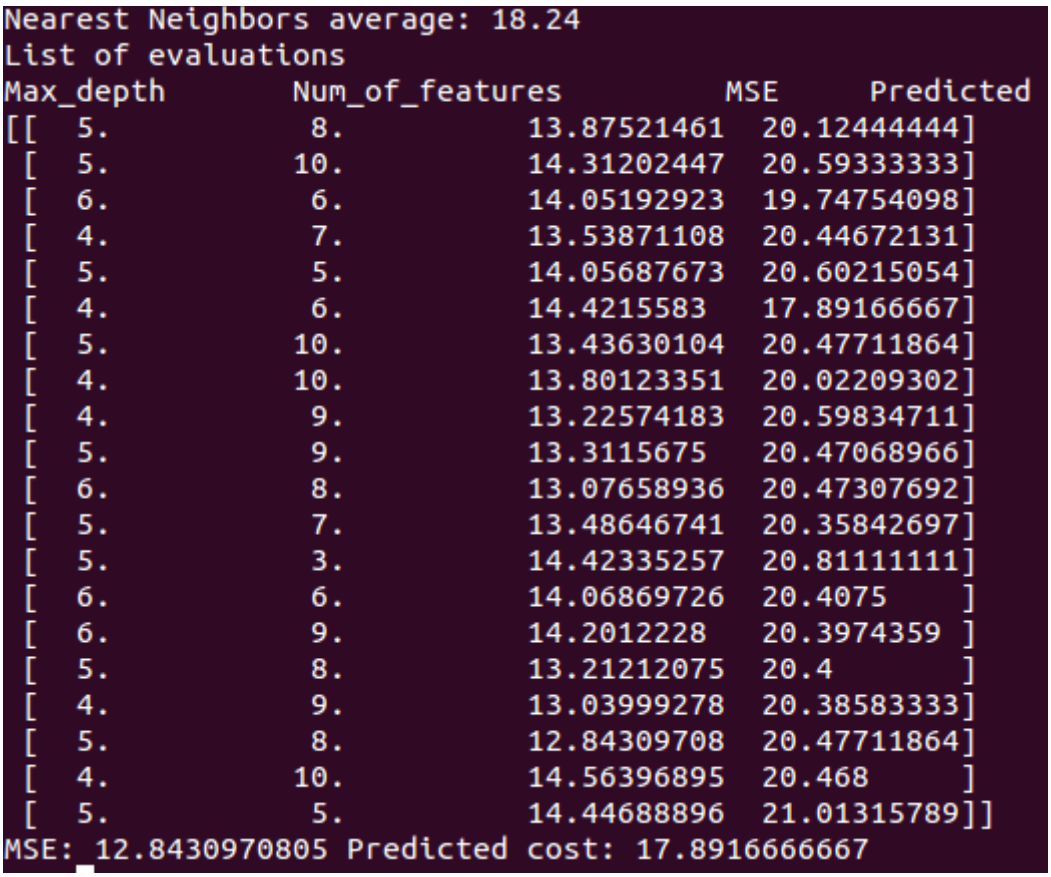
In previous section predicted house price is 21.35K for such case:
[11.95, 0.0, 18.1, 0, 0.659, 5.609, 90.0, 1.385, 24, 680.0, 20.2, 332.09, 12.13] .
And model best MSE is 17.42
Comparing with Chapter 1 model MSE value is inside STD range. Also predicted house price is near median and mode of Boston data prices. I think predicted value is reasonable.

Comparing model parameters obtained by using GridSearchCV and predicted with NearestNeighbors the values is very near: **Nearest Neighbors average: 18.24**

```
List of evaluations
Max_depth       Num_of_features         MSE     Predicted
[[  6.          10.          13.43137175  21.06956522]
 [  4.           8.          13.74483054  20.38583333]
 [  5.           8.          14.37212414  20.34754098]
 [  5.           5.          13.62818479  20.73055556]
 [  6.          10.          13.45462652  21.35555556]
 [  5.           7.          13.03126003  20.18111111]
 [  5.           6.          13.59362936  20.27142857]
 [  6.          10.          12.7859732   20.3974359 ]
 [  6.           9.          13.93104244  19.66969697]
 [  5.           6.          12.4185919   19.83116883]
 [  6.           6.          13.75432123  21.24262295]
 [  5.           8.          13.41120498  20.24823529]
 [  6.           9.          13.65988156  18.06666667]
 [  4.          10.          13.74976208  20.38583333]
 [  4.          10.          13.74643354  20.37226891]
 [  5.          10.          13.30560446  20.37037037]
 [  4.           8.          13.54793143  20.56835443]
 [  4.           5.          13.95711081  20.88510638]
 [  4.           9.          13.44885059  20.4338843 ]
 [  4.          10.          13.54380616  20.76179775]]
MSE: 12.4185918977 Predicted cost: 18.0666666667
```

The results in this case are very close and model model parameter are reasonable.

Another experiment was with using RandomizedSearchCV:



In this case results also are very close and model model parameters are reasonable.