# P1 - Predicting Boston Housing Prices

*Michaël Lambé*
*mic0331@gmail.com*

## 1/ Statistical Analysis and Data Exploration

Size of data: 6578.00 records
Number of features : 13.00 + 1 (the target = MEDV - Median value of owner-occupied homes in $1000's)
Min target value: 5.0
Max target value: 50.0
Mean target value: 22.53
Median target value: 21.20
Std. dev. target value: 9.19

## 2/ Evaluating Model Performance

- **Which measure of model performance is best to use for regression and predicting Boston housing data? Why is this measurement most appropriate? Why might the other measurements not be appropriate here?**

When the outcome is a number, the most common method for characterising a model's predictive capabilities is to use the mean squared error (MSE). This metric is a function of the model residuals, which are the observed values minus the model predictions. The MSE is calculated by squaring the residuals and summing them.
Another common metric is the coefficient of determination (R^2). This value can be interpreted as the proportion of the information in the data that is explained by the model. R^2 is a measure of correlation, not accuracy.

Practically speaking, the dependance on the outcome variance can have a drastic effect on how the model is viewed. In our case as the range of the houses is large from $5K to $50k, the variance of the sale price would also be very large. One might view a model with a 90% R^2 positively, but the MSE may be in the tens of thousands of dollars — poor predictive accuracy for anyone selling a moderately priced properly. therefore, we prefer using the MSE for our performance analysis.

An other possible measure we would consider is the MAE (mean absolute error). This metric is more robust to outliers since it does not make use of squares. However, MSE is more useful if we are concerned about large errors whose consequences are much bigger than equivalent smaller ones. so again, MSE is the preferred metrics for this dataset.

- **Why is it important to split the data into training and testing data? What happens if you do not do this?**

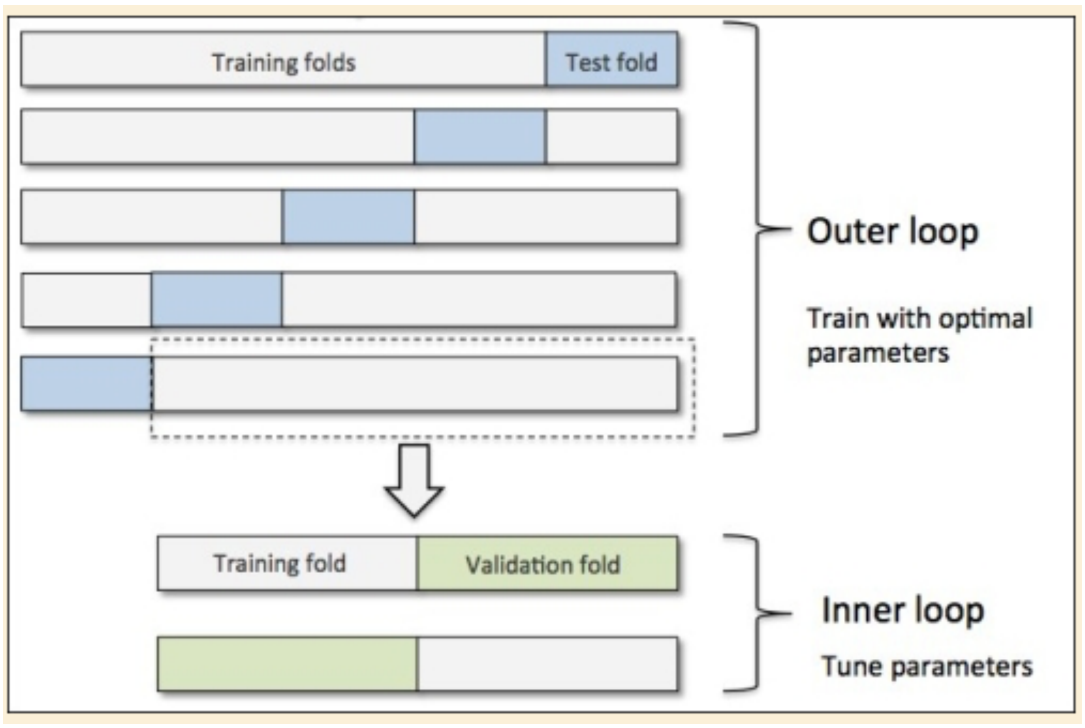
We split (randomly partition) our initial dataset into a separate training and test dataset —

the former is used for model training, and the latter is used to estimate it's performance. This process is important in order to determine whether our machine learning algorithm not only performs well on the training set but also generalises well to new (unseen) data. This is particularly important to detect overfitting (high variance / the model is too complex) and undercutting (high bias / the model is not complex enough)

- **Which cross validation technique do you think is most appropriate and why?**

We use nested cross-validation technique. We have an outer k-fold cross-validation loop to split the data into training and test folds, and an inner loop is used to select the model using k-fold cross-validation on the training fold. After model selection, the test fold is then used to evaluate the model performance.

The following figure explains the concept of nested cross-validation with five outer and two inner folds, which can be useful for large data sets where computational performance is important; this particular type of nested cross-validation is also known as 5x2 cross-validation.



- **What does grid search do and why might you want to use it?**

In machine learning, we have two types of parameters: those that are learned from the training data and the parameters of a learning algorithm that are optimised separately. The latter are tuning parameters, also called hyperparameters, of a model (for example, the depth parameter of a decision tree).
In order to find the optimal combination of hyperparameter values, we use a technique

called grid search. This approach is quite simple, it's a brute-force exhaustive search paradigm where we specify a list of values for different hyperparameters, and the computer evaluates the model performance for each combination of those to obtain the optimal set.
In our case, the optimal "max_depth" parameter is an example of parameter that is search by grid search for us.

## 3/ Analyzing Model Performance

- **Look at all learning curve graphs provided. What is the graph trend of training and testing error as training size increases?**

The training error is decreasing to become null and the testing error increase slowly, the testing error graph is also squeezing. This is an indication of overfitting.

- **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?**

The initial learning curve presented to us is showing high bias. This model underfits the training data. As the training size increases, we see that the training error decrease and the gap between training error and testing error is minimum at about max_depth=2.
At higher depth we see that the training error is getting minimum but the gap with the test error is increasing. This indicates an overfitting of the training data (high variance).
The deeper the decision tree, the more complex the decision boundary becomes, which can easily result in overfitting.

- **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?**

As the complexity of the model increases we always improve the fitting to the training data (by adding mode predictors). Some predictors indeed capture the main features behind the data. Those features are shared by both training and test data, and the test data prediction error can be reduced by including those predictors.
Some other predictors may only describe the noise in the training data which is not reproducible in the test data. Therefore, the test data prediction error is raised by the other (less important) predictors.

The best model that generalises the dataset is the model with max_depth = 5 because this is the point where the test error and training error are balancing each other well. The model complexity decreases prediction error until max_depth = 5 where we are adding just noise. The training error goes down as it has to, but the test error is starting to go up (overfitting).

## 4/ Model Prediction

- **Model makes predicted housing price with detailed model parameters**

```
GridSearchCV(cv=10, 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(two_score, greater_is_better=False), verbose=0)
       GRIDSCORES     MSE mean: -70.40126, std: 47.80094, params: {'max_depth': 1}
       GRIDSCORES     MSE mean: -40.12898, std: 30.76690, params: {'max_depth': 2}
       GRIDSCORES     MSE mean: -40.17378, std: 31.43985, params: {'max_depth': 3}
       GRIDSCORES     MSE mean: -39.16489, std: 34.05245, params: {'max_depth': 4}
       GRIDSCORES     MSE mean: -36.18948, std: 31.31878, params: {'max_depth': 5}
       GRIDSCORES     MSE mean: -33.36084, std: 26.36594, params: {'max_depth': 6}
       GRIDSCORES     MSE mean: -33.65017, std: 27.34841, params: {'max_depth': 7}
       GRIDSCORES     MSE mean: -36.68978, std: 31.69844, params: {'max_depth': 8}
       GRIDSCORES     MSE mean: -37.00423, std: 29.05275, params: {'max_depth': 9}
       GRIDSCORES     MSE mean: -38.61797, std: 30.79695, params: {'max_depth': 10}
MSE    HP    {'max_depth': 6}    33.360840

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: [ 20.76598639]
```

- **Compare prediction to earlier statistics**

According to the prediction we can see that the predicted price is acceptable. The price is between the max and min value but bellow the mean and median.
We can also noticed that the price is included in the main portion of our data distribution as it is included in the one standard deviation below the mean (34.13% of the population bellow the mean) assuming our data are normally distributed.