1) Statistical Analysis and Data Exploration

- Number of data points (houses): **506**
- Number of features: **13**
- Minimum and maximum housing prices: **5 - 50**
- Mean Boston housing prices: **22.53**
- Median Boston housing prices: **21.20**
- Standard deviation: **9.19**

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?

  Measure of model performance has to be one suitable for regression tasks: mean square error, mean absolute error, etc.
  Of course measures like accuracy, precision, recall, or the like, are pointless here, being those meant for classification tasks.
  Which to choose among the ones that fit our task? After trying out mean square error and mean absolute error, results don't look that different, apparently it makes no difference in choosing among this two. I could have used the median absolute error, which is robust to outliers, but the analyzed dataset doesn't look particularly affected by this kind of problem – in fact, mean and median are not that different.
  Eventually I used **mean squared error** measure.

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

  Missing to split dataset in training and test datasets won't give us the possibility to verify that our model generalizes well, that is that it's able to interpret correctly data that it has never "seen" before - which is the goal of ML, eventually.
  Specifically we couldn't verify that our model suffers from the overfitting problem: training data – and possibly the noise that comes along with it – could be fit too well, eventually missing the real shape of the regression curve – or of the separation boundary, in case of classification tasks.

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

Besides learning the parameters that represent the model, and that will be used in the production phase to make predictions, there are a few parameters that have to be tuned separately, usually called hyperparameters: these could be the max depth for a decision tree, as in our case.
Grid search helps us in finding the optimal values for the hyperparameters in an automated way.
In its simplest form, it is a brute force search: for each of the hyperparameter, a set of values is given, properly chosen by the experimenter.
For each tuple in the Cartesian product of those value sets, a model is trained and then estimated. The tuple corresponding to the best estimation is then selected as the optimal hyperparameters.
Since the brute force search could be computationally too expensive, a variant exist for which grid search performs a random search more than an exhaustive one.

- Why is cross validation useful and why might we use it with grid search?

Cross validation is fundamental in performing model selection without incurring in overfitting problem. During model selection phase, one wishes to find the optimal values for the hyperparameters of the learning algorithm. Those parameters will be used for the final model.
In the most basic form of validation, the available dataset is split into three subsets: training set, validation set and test set.
Training set is used to train models, each with its own setting of hyperparameters values, while the validation set is used for scoring them. The best scoring model is selected, and its hypeparameters values are used to define the final model that is eventually trained (training set + validation set can be used for the training of the final model) and evaluated against the test set.
This arrangement suffers from a couple of major problems:
- few samples can be left for training the models, during model selection phase, because of the split in three different partitions. This is particularly true when data is scarce, that might be often the case.
- the results of model selection can depend on the particular way the training and validation sets has been sampled.
In order to overcome these problems, cross validation can be applied. In its most basic form, named k-fold cross validation, we have to split the available dataset in two subsets: training set and test set: validation set is not needed anymore, so more samples can be used for training.
During the model selection phase, training set is split in k subsets. In a rotating fashion, we select k-1 of the k available subsets and use those to train a model, while we use the

remaining subset for scoring. This way, we end up training a model k times: the final score is computed as the average of the scores from each training.
This process has to be repeated for each model experiment we wish to run, that is, for each particular arrangement of hyperparameters we wish to test.
Eventually, the hyperparameters of the best scoring model are selected, and are used for the final model that is trained with the training set and then evaluated with the test set.

It's useful to apply cross validation in a grid search session, since grid search after all performs a model selection, in an exhaustive and systematic way.

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?

  Training error increases until it reaches a value and then remains around that value.
  This could be explained whit the fact that it's easier for the model to fit/overfit the data when training examples are few.
  Testing error decreases until it reaches a stable value, as for training error.
  This can be explained with the fact that, increasing the training size, we increase the chance to avoid the overfitting problem and to improve the generalization ability of the learnt model.

- 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?

  For **max depth = 1**, train ad test errors converge to a quite high value: this means that our model is suffering from high bias/underfitting and it could be too simple for the problem at hand and we need to raise its complexity
  For max depth = 10, train and test error don't converge to the same value, but train error is much lower. This means that the model is suffering from high variance/overfitting and that it is too complex for the problem/data at hand.

- 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?

Training error tends to decrease consistently while increasing the model complexity, that is the max depth value in this case.
Test error instead reaches a minimum for a max depth value that could be 5 or 6, then for bigger values it starts having worst performances.
By a qualitative inspection of this graph, 5 or 6 could be the best choice for the max depth value.

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.

  Predicted price for the given example is **20.76**.
  The best model as selected by grid search has **max depth = 6**

- Compare prediction to earlier statistics and make a case if you think it is a valid model.

  Prediction, that is **20.76**, looks following the tendency as described by the computed mean/median values in earlier statistics, this could be a hint that the model is quite valid. Also, the mean squared error obtained so far looks acceptable, compared to the quantities involved. Nevertheless, it could be interesting to evaluate the magnitude of the error respect to the real value of the houses: for example, while a +/-2 delta in the predicted price could be acceptable respect to the value of a house that is worth 50, the same error could be problematic if the real value is 5, in which case the error would be about 40% of the the real price.