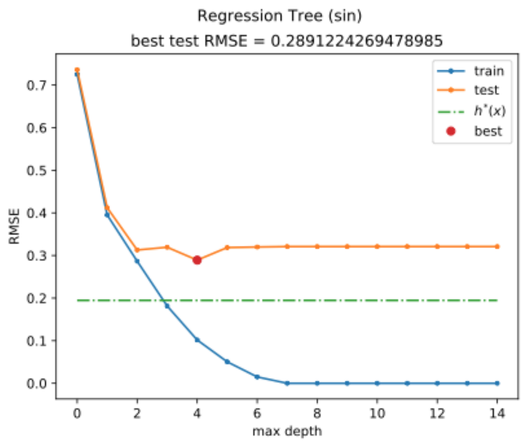
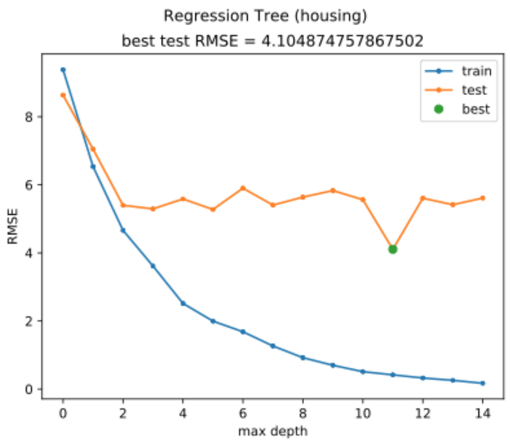
Ensembling

by José Ananías Hilario Reyes

Assignment 1

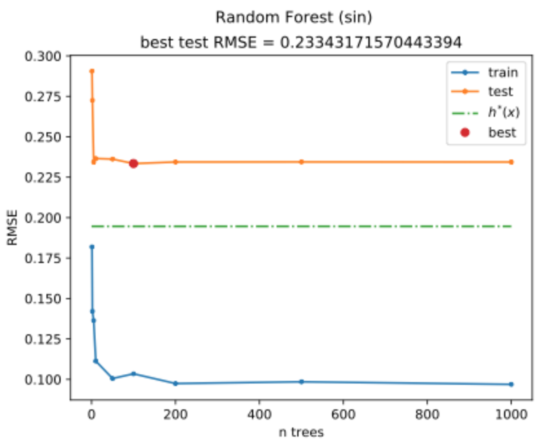


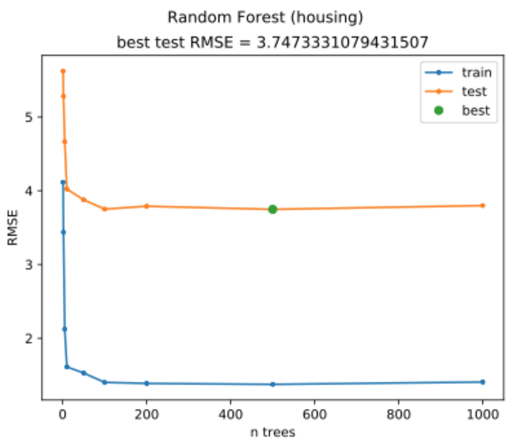


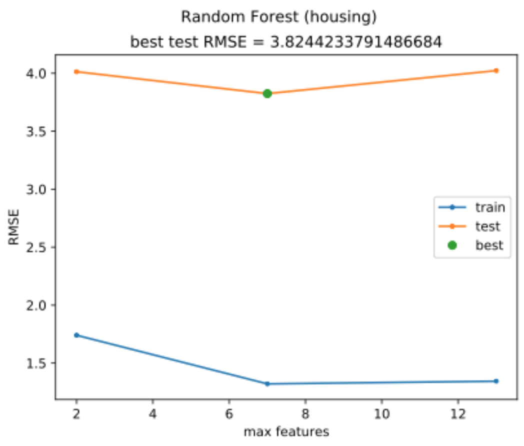
We can see from the results measuring the influence of the depth in building regression trees, that optimal point with respect to error varies from case to case, and could be found by cross validation. We know, however, that it is preferable to create the trees to maximal depth, so that they have low bias, and can be combined using bagging ensemble methods. If they are to be combined using boosting, usually we want to make them much shallower, having 8 levels top.
The results we get show us that nothing is lost by increasing the depth of the tree, in what respects to regression: after certain minimum point, the error attains a steady state which remains constant. Therefore, we can get the benefits of maximal depth, without risking damaging the model.

It also can be seen that for pure functions, regression trees make their optimum fit at a fairly shallow depth, and that for this kind of functions, the steady state is constant. In the housing data case, however, we see that the depth required to reach a good estimate is bigger, and that the steady state is oscillating around some mean point.
The error in the case of a pure function approximates very well the optimal predictor. Not so in the case of real data, where we get a much bigger error.

Assignment 2



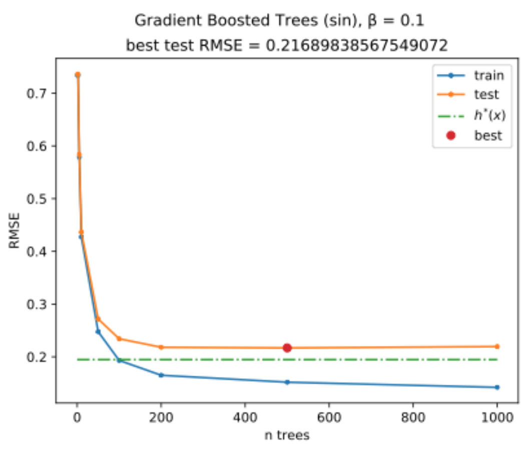


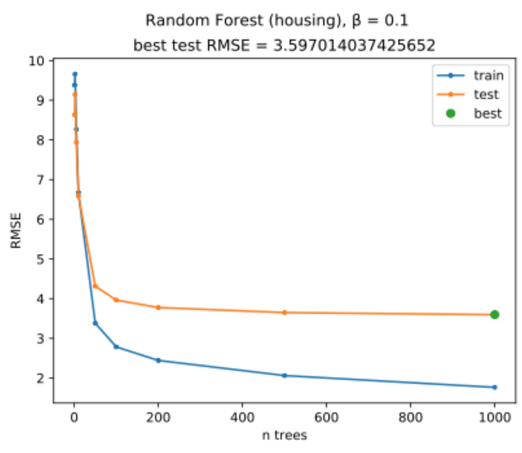


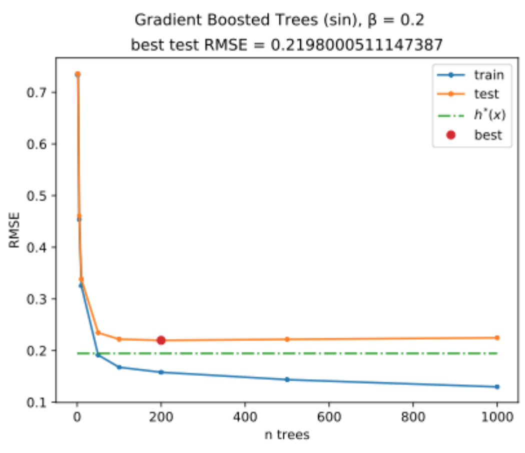
Even though adding a lot of trees reduced dramatically the training error, their addition stopped influencing the result on the test data at certain point. We can conclude from these results that some lack in generalization is harming the final result. This is happening also for the pure sin function data.

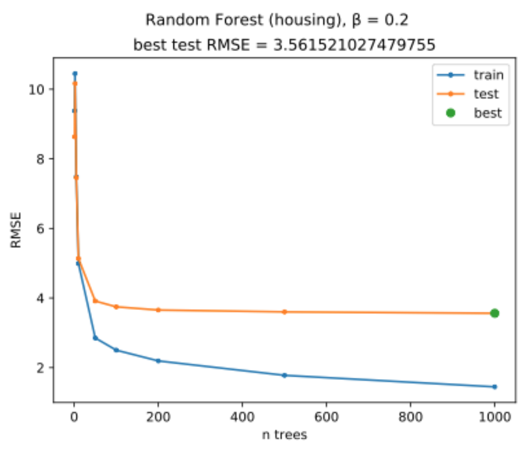
In the case “c” we see that the optimal choice of attributes to use for the trees, is around half of the maximum. Empirically, it has been shown that good results can be obtained using ⅓ n and sqrt(n). We know that the randomization of the split in attribute selection decorrelates the trees, thus, enhancing various aspects such as immunity to outliers, and effectively reducing the variance of the final result.

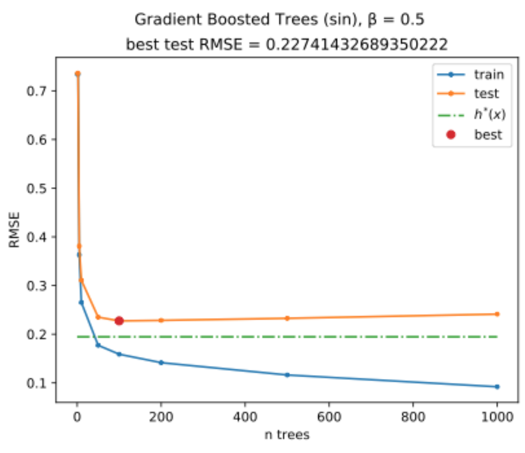
Assignment 3

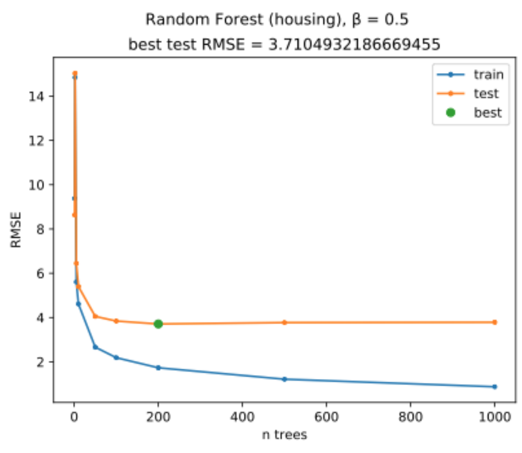


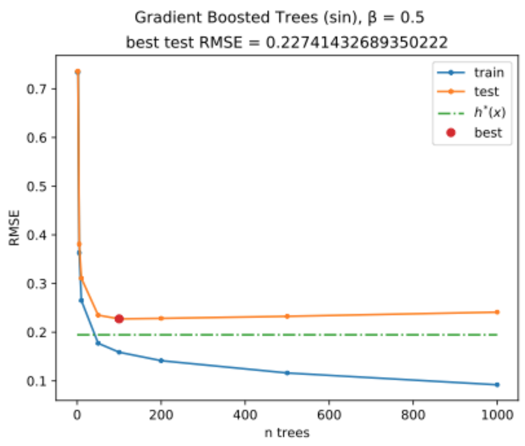


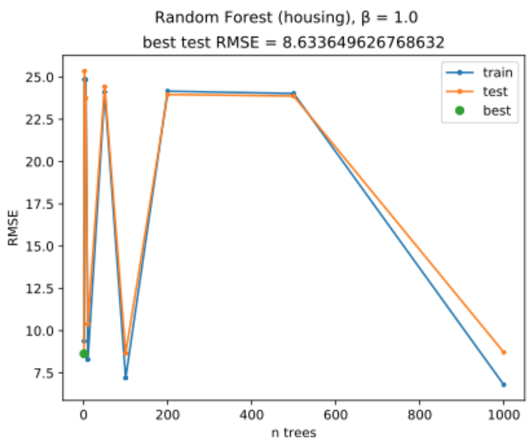












We can see that the results from the Gradient Boosted Trees improve over the Random Forests. It's not by chance that many of the top 1% champions at Kaggle have resolved problems using variants of this method.

In what respects to its parameters, the learning rate has an influence that varies from one problem to another, from data to data. Because of this, it is usually recommended to learn its value by cross-validation, or by using line search. In our specific cases, worst of them all was the learning rate of 1, which gave away very unstable results. Usually after a learning rate of 0.5, the results worsened.

Another parameter that is usually learned by cross-validation is the number of trees needed to make a good classifier. Having more trees than its needed doesn't have a negative effect on the classification, so what one would do in practice is increment trees for as long as one sees improvements.

Assignment 4

A parallelization approach was taken for the Random Forest implementation. The idea was to distribute the task of constructing each individual tree, so that time could be used efficiently.

The difference between serial run and parallel run are illustrated in the following table.

Random Forest (housing): n_trees = 1000	
Serial	**Parallel**
249.98794198036194 seconds	139.68777179718018 seconds