Classification trees, bagging, random forests

Machine Learning for Data Science Homework #1

Vid Stropnik; 63200434

ABSTRACT

In the first homework in the Machine Learning for Data Science class, I worked with Classification trees and algorithms that use them. Throughout my work, I gained deeper understanding in how a decision tree is built, how it predicts, what are the method's shortcomings and how we can circumvent them. Through the requested plots and reports, I describe my lessons herein.

1 DATA

All reporting is done on the *housing3.csv* dataset. For the remainder of this report, the initial 80% (400 samples) of the data are referred to as the *training set*, while the latter 20% (100 samples) are adressed as the *testing set*, as these were their roles throughout the work.

2 DECISION TREE

I implemented my decision tree algorithm in such a way, that it finds potential splitting points in between distinct values of any given attribute. For each binary split, the algorithm computes a candidate splitting point's error as *a weighed sum of gini indices of its potential children*. The selected splitting point is the one with the minimal error. The building of the decision tree is stopped when the number of training sample in a given node recedes below the hyper-parametric threshold or the node is pure (contains only a single class). When predicting, the leaf returns the target-variable (class) value of the majority of its contained samples.

Misclassification rates from hw_tree_full *(min_samples=2)*:
Training set : 0.0
Testing set : 0.17

2.1 Internal Cross Validation

An ideal lower threshold for splitting a tree node was determined, using cross validation on the 400 samples of the training set. The validation used 5 folds and shuffled the data once using the random seed *1* for the *numpy.random.shuffle* method. In Figure 1, we can notice that the performance starts plateuing after a certain size of *min_samples*; around 40.

Misclassification rates from hw_min_samples:
Training set : 0.055
Testing set : 0.160
Optimal splitting threshold : 6

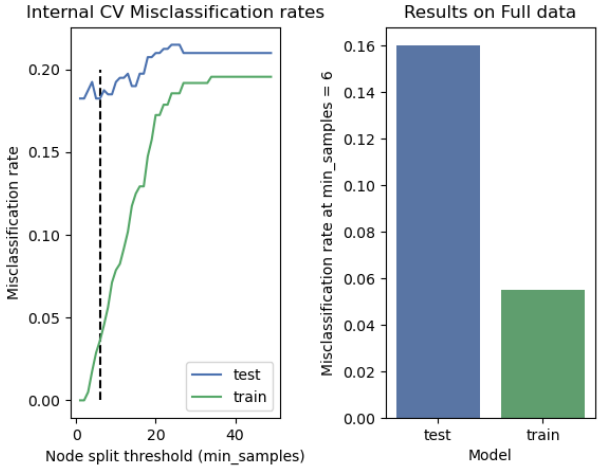

Figure 1: Results of hw_min_samples. The left plot shows the CV performance across different lower thresholds. The right plot shows the final results on the full dataset.

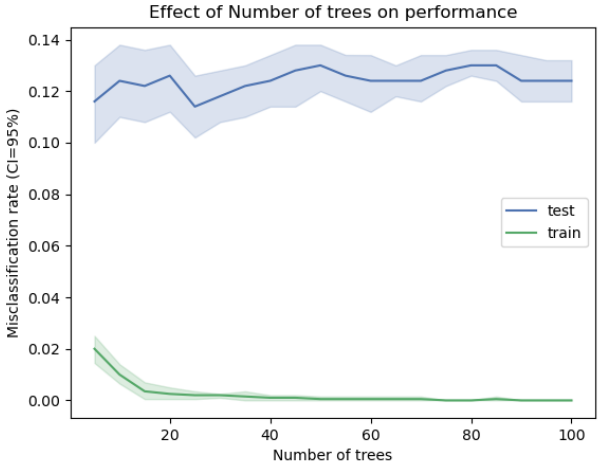

Figure 2: Bagging performance for different numbers of trees.

3 BAGGING

With bagging, a hyper-parametric number of trees are creating by bootstrapping the entire training data set. Any given prediction is then decided by the majority output of all of the created trees. The outputs are visualised in figure 3. Notice that the deviation between different random seeds is quite high and that the model performs significantly better than the standard Decision Tree.

Misclassification rates from hw_bagging *n=50*:
Training set : 0.0025 ± 0.0001
Testing set : 0.12 ± 0.02

4 RANDOM FOREST

When training a random forest, an additional step is taken in comparison to bagging. At each splitting point, only $\sqrt{N}$ features are

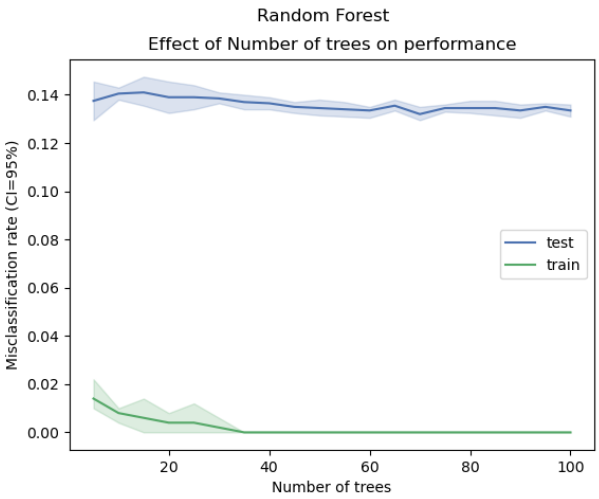

Figure 3: Performance of Random Forests for different number of trees.

considered, where N is the number of all possible features in the dataset. Consequently, the building algorithm is significanlty faster, while also reducing the effect of the bootstraping randomness on the model's performanc, as is visible when comparing Figures 2 and 3

Misclassification rates from hw_randomforests *n=50*:
Training set : 0 ± 0
Testing set : 0.135 ± 0.0025

5 OBSERVATIONS

From the conducted experiments, we can clearly deduce that the problems with the variance of Decision Trees is a tangible concern – seeing as just minor changes, introduced in the bootstrap method were able to significantly effect the model's performance. We can also observe that the selection of the number of trees does not play a very large role in the performance of either of the ensemble algorithms. The effect of the choice of random seed does, however, slightly decrease over time with the increase of this n. The final observation of note is the vast reduction of the effect of the random seed in the Random Forest algorithm. We can conclude that bagging and random forests tend to achieve comparable results, as shown in Figure 4. While this specific set does not conclude this on the optimal splitting data, results in prior plots have shown that, due to its lower variance (and lower computational complexity), Random forests should be considered favorably for future applications.

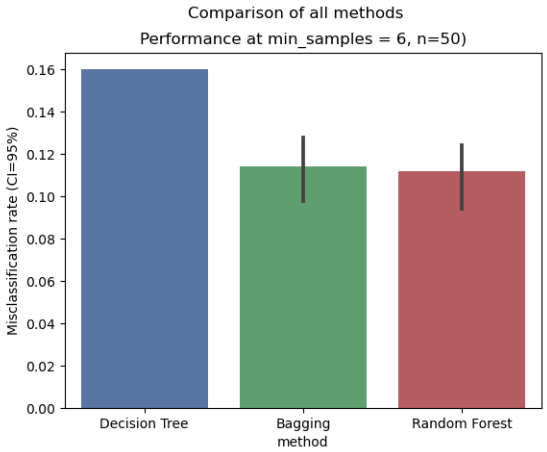

Figure 4: Comparative evaluation of results on optimal splitting points and provided number of trees