Classification trees, bagging, random forests

Urša Zrimšek

uz2273@student.uni-lj.si, 63200441

Introduction

This is the report of first homework for class Machine Learning for Data Science 1, in which we implemented classification trees, bagging, and random forests. Let us first describe the classes and their parameters.

First we implemented a flexible classification tree, in class `Tree`, with the following attributes:

- `rand`: a random generator, for reproducibility, of type `random.Random`,
- `get_candidate_columns`: a function that returns a list of column indices considered for a split (needed for the random forests),
- `min_samples`: the minimum number of samples, where a node is still split,
- `left`: `None`, if we are in leaf, `Tree` if current node is a split,
- `right`: `None`, if we are in leaf, `Tree` if current node is a split,
- `prediction`: prediction of a leaf or `None`, if current node is not a leaf,
- `split_criteria`: `None` if this node is a leaf or (column index, value) of the split otherwise.

Our next class was `Bagging`, with the following attributes:

- `rand`: a random generator, for reproducibility, of type `random.Random`,
- `tree_builder`: an instance of `Tree` used internally,
- `n`: number of bootstrap samples,
- `tree_list`: a list of `n` trees.

Last implemented class was `RandomForest`, with the following attributes:

- `rand`: a random generator, for reproducibility, of type `random.Random`,
- `n`: number of bootstrap samples,
- `min_samples`: the minimum number of samples, where a node is still split,
- `bag`: an instance of `Bagging` used for building the forest.

Each of this classes has methods `build`, that returns the model as an object and `predict`, that returns the predicted target class of given input samples. They use *Gini index* to decide the best split at building the trees.

The methods are applied to `housing3.csv` dataset, that is composed of 500 rows with 13 numeric columns and one column that represents binary target variable. We selected first 400 rows as training set and last 100 as testing set.

Our tasks were following:

1. In function `hw_tree_full`, build a tree with `min_samples=2`. Return misclassification rates on training and testing data.
2. In function `hw_cv_min_samples`, find the best value of `min_samples` with 5-fold cross-validation on training data. Return misclassification rates on training and testing data for a tree with the best value of `min_samples`. Also, return the best `min_samples`.
3. In function `hw_bagging`, use bagging with `n=50` trees with `min_samples=2`. Return misclassification rates on training and testing data.
4. In function `hw_randomforest`, use random forests with `n=50` trees with `min_samples=2`. Return misclassification rates on training and testing data.

Results

In this section, we will provide answers to the tasks above, and show some more graphical results.

Classification tree

Misclassification rates for a tree with `min_samples=2` are:

- 0 on training set,
- 0.17 on testing set.

The result for the training set is expected. If `min_samples=2`, that means that we will have only one type of target value in each leaf (we can still split even if there are only 2 different values in a node), that is why our tree must have perfect score on training data.

Cross validation for min_samples

To answer the second task, we did a 5-fold cross validation, calculated misclassification rates on training and testing data for different values of attribute `min_samples`. We tested values from 2 (the smallest possible, since we cant split a node with only one sample) to 50. The results can be seen on Figure 1.

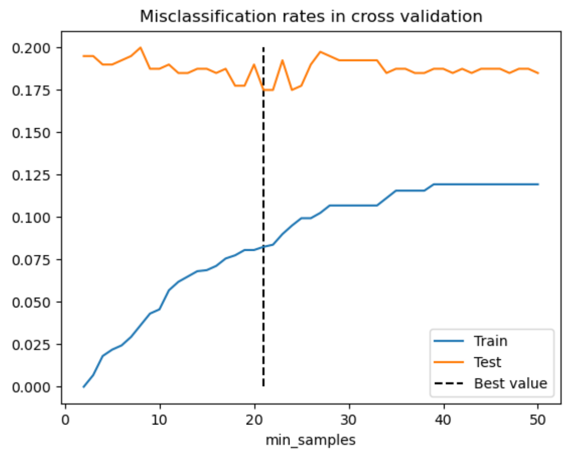

Figure 1. Misclassification rates versus `min_samples`. Here we can see that misclassification on the training set starts at 0 for value 2 (as expected) and monotonously rises almost until value 40, then it platous. That is also one of the reasons we didn't continue for bigger values (another one being the size of the train set). Misclassification on test set was the factor for selecting the best value. It is the smallest for values 21, 22 and 24, with all reaching the same rates on the original test data.

From the cross validation we selected value 21 for the best one. We trained a tree with it on our whole training set, and calculated misclassification rates:

- 0.08 on training set,
- 0.13 on testing set.

We can see they are indeed better than with `min_samples=2` The reason is less overfitting to training data.

Bagging

Misclassification rates from the third task (bagging with 50 trees and `min_samples=2`) are:

- 0.0025 on training set,
- 0.14 on testing set.

On Figure 2 we can see the influence of number of trees on misclassification rates on training and testing set.

On Figure 2 we plotted results for $n \in \{1,2,3,5,10,20,30,40,50,75,100,150,200\}$. On Figure 3 we plotted them for all values from 1 to 50.

Random forest

Misclassification rates from the fourth task (random forest with 50 trees and `min_samples=2`) are:

- 0 on training set,
- 0.11 on testing set.

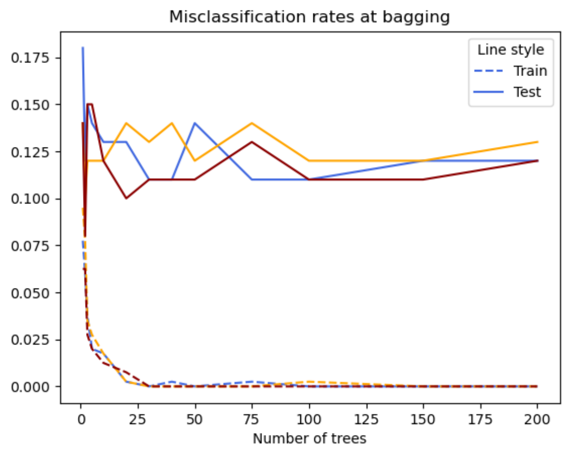

Figure 2. Misclassification rates versus the number of trees. Here we can see that the misclassification on training set is almost monotonously falling, but on the testing set is more jumpy. Different color accounts for different seeds, and we can see that it has a lot of influence on the misclassification rates.

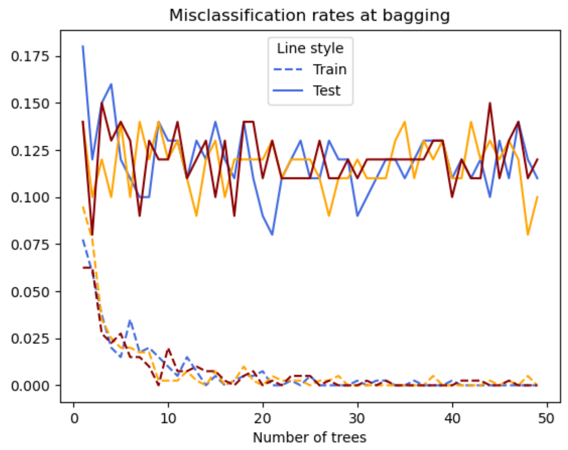

Figure 3. Misclassification rates versus the number of trees. On this plot we can see big influence of different random seeds, having completely different numbers of trees as the best. From it I would conclude that we can't select the best number of trees with this kind of testing.

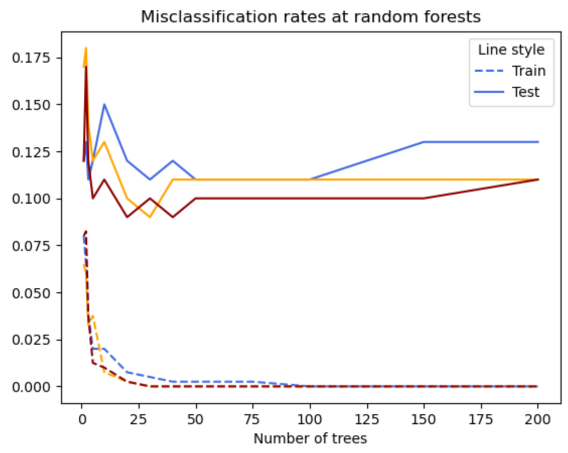

Figure 4. Misclassification rates versus the number of trees. Here we can see that the misclassification on training set is almost monotonously falling, but on the testing set we can see again that the seed has a lot of influence on the misclassification rates, but we could say that the best number should be in between 20 and 50.

On Figure 4 we can see the influence of number of trees on misclassification rates on training and testing set.

On Figure 4 we plotted results for $n \in \{1,2,3,5,10,20,30,40,50,75,100,150,200\}$. On Figure 5 we plotted them for all values from 1 to 50, to get a better view of the numbers that looked best on Figure 4.

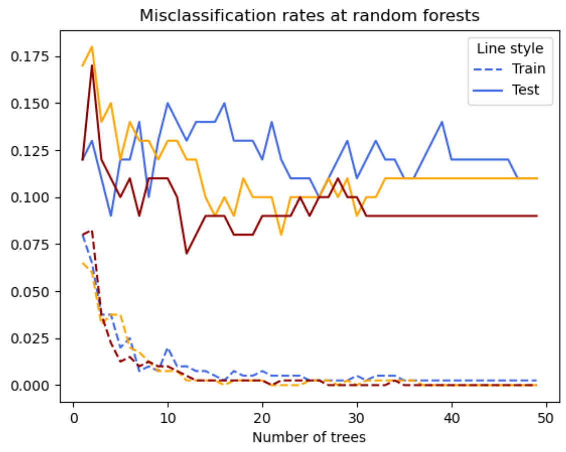

Figure 5. Misclassification rates versus the number of trees. This plot gives us different best values for different seeds - around 4, 12 and 22. Again I would say that the influence of the seeds is too big to select the best value from this kind of testing.