Classification trees and random forests

Blaž Erzar

Machine Learning For Data Science 1 2023/24
Homework 1 report

Abstract

This report presents the implementation of a decision tree and random forest models, as well as their evaluation on the dataset tki-resistance*. We also look at the feature importance of this dataset and provide further support that our implementation is correct using artificial datasets. Our implementation achieves results comparable to classifiers implemented in a Python package, i.e., a misclassification rate of* 19.0% *for the tree and* 1.7% *for the random forest model.*

1 Introduction

In the first homework for the course Machine Learning for Data Science 1 we implemented decision tree and random forest models for classification. A decision tree is built using the CART algorithm with the Gini index as an impurity measure. A random forest builds multiple decision trees internally on bootstrap samples and makes them less correlated by only using a sample of all features.

Since random forest uses bootstrap sampling, we have access to out-of-bag samples, which can be used to compute feature importance, as described in [1].

Both models are evaluated on a small FTIR spectral dataset.

2 Methods

2.1 Decision tree

A tree is built recursively by splitting the dataset into two parts in each step. Building stops when the node size is too small or when the region is already pure, i.e., all data points have the same class.

To find the best split, we first sort each column of the dataset and compute averages between consecutive values to get the threshold values we consider for splitting. We select a feature f and a threshold t, which split the dataset $\mathcal{D}$ into $\mathcal{D}_L$ and $\mathcal{D}_R$ that minimize the weighted Gini impurity

$$G(\mathcal{D}, f, t) = |\mathcal{D}_L| G(\mathcal{D}_L) + |\mathcal{D}_R| G(\mathcal{D}_R).$$

The Gini impurity $G(\mathcal{D})$ is calculated by multiplying 2 with the product of probabilities of both classes.

When predicting, we follow the tree structure by comparing feature values to thresholds and predict the majority class in the final leaf node.

2.2 Random forest

A random forest is built by building n trees ($n = 100$ in our case). Each tree is trained on a bootstrap sample (sample with replacements) of the original train data and in each split, only $\sqrt{n}$ features are considered, where n is the number of all features.

When predicting, we compute the majority of predictions of all trees.

2.3 Feature importance

To compute the feature importance in a random forest, we use the out-of-back samples, i.e., samples which were not used for training the trees. For each feature, we compute the average decrease in accuracy over all trees, after permuting feature values.

3 Results

For evaluation, we use the provided *tki-resistance* dataset and train both models using the first 130 rows. Prediction results can be seen in Table 1. Models successfully learn the patterns in the data and achieve comparable results to the implementation in the library. All models perform better than the majority classifier at 39.7%. Because the misclassification rate is the expected value of a Bernoulli random variable with 0 and 1 corresponding to correct and incorrect classification, the standard error can be computed as the standard deviation of this random variable divided by $\sqrt{n}$.

	Ours		scikit-learn	
	MR [%]	SE [%]	MR [%]	SE [%]
Tree	19.0	5.1	20.7	5.3
Forest	1.7	1.7	1.7	1.7

Table 1: ***Misclassification rates (MR) and standard errors (SE)*** *for both models computed on the test set, including a comparison of our implementation with the one in the scikit-learn Python package.*

To further support the correctness of our tree implementation, we test it on an artificial dataset with

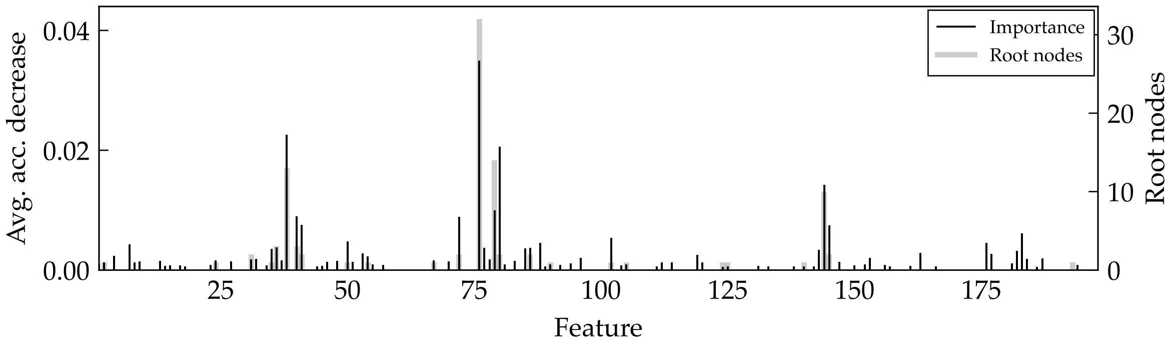

Figure 1: ***Random forest feature importance and root nodes of a*** **100** ***trees*** *trained on the* tki-resistance *dataset. Trees were built using* 20% *of the train set. Both random forest and tree models prefer similar features. Importance values lower than* 0.005 *were dropped for clarity. Out of those, some values were also negative.*

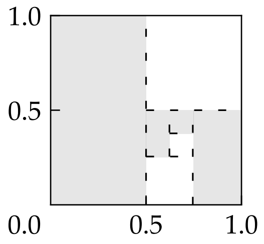

Figure 2: ***An example of a tree trained on an artificial dataset****. Ground truth class is shown using background colour, while the tree splits are shown using dashed lines. Both feature and threshold were chosen correctly in each split.*

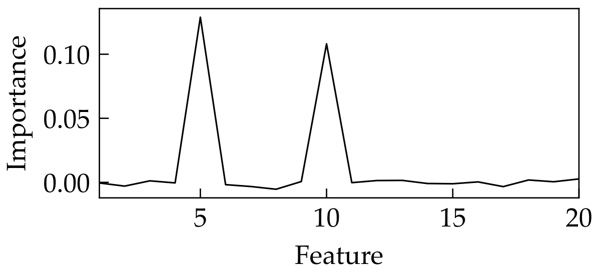

Figure 3: ***Random forest feature importance for an artificial dataset****. The class variable is calculated using features* 5 *and* 10*. Both are detected correctly using the feature importance.*

spiral splits of the 2D domain. As can be seen in Figure 2, the model successfully finds the correct feature and threshold in each split.

Similarly, we support the correctness of the feature importance calculation for the random forest. In Figure 3, we see the importance calculated for an artificial dataset containing 20 features. The class variable is calculated only using features 5 and 10, which are correctly detected by the algorithm.

The feature importance calculated on the *tki-resistance* dataset is shown in Figure 1, as well as roots of a 100 trees trained on a sample of the train set.

Since random forest has a parameter n that can be tuned, we also look at the dependence of the error rate on n. It can seen from Figure 4 that the misclassification rate drops with the number of trees.

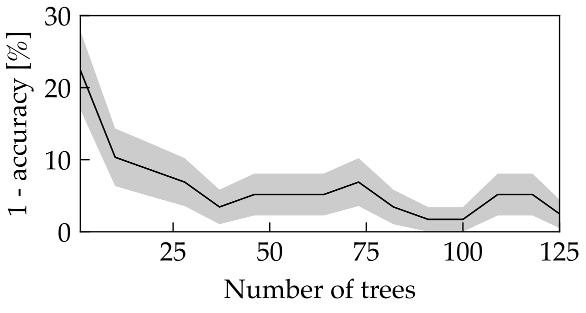

Figure 4: ***Misclassification rate depending on the number of trees*** *in the random forest. The standard error is shown with the grey band around the plot line.*

4 Discussion

From Figure 1 we can observe that feature importance calculated using the out-of-bag samples in a random forest model does correlate with the features in the root of trees trained on a sampled dataset. Both approaches show peaks around features 37, 75 and 140. This does seem sensible since it is more likely that more important features will provide a good first split. The main difference between these results is that using trees we do not obtain the distribution over all the features, as we do with the permutation test. We only have 100 root nodes that are placed into almost 200 histogram buckets.

The drop in misclassification rate in Figure 4 seems sound as well, because the variance of the model drops with a higher number of trees.

We have also shown that our implementation does achieve the expected accuracy, not only on artificial datasets we used for testing correctness, but also on an FTIR spectral dataset.

References

[1] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: `10.1023/A:1010933404324`. URL: `https://doi.org/10.1023/A:1010933404324`.