U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

IACEC

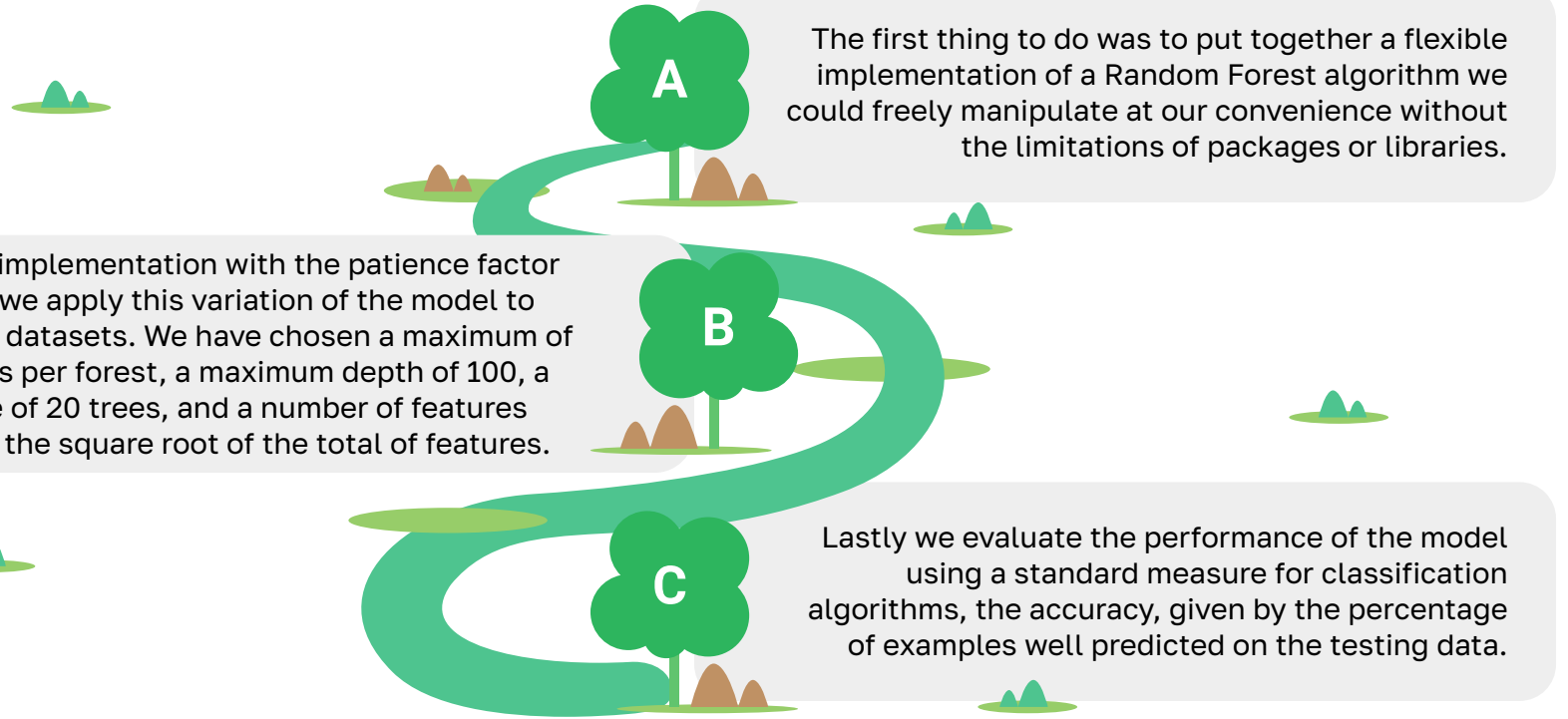
Bruno Silva - up201508756
João Carvalho - up201507023

Algorithm and Data

1. Random Forest

2. Classification-specific datasets with a maximum number of 500 features for computational reasons.

Experimental Setup



The first thing to do was to put together a flexible implementation of a Random Forest algorithm we could freely manipulate at our convenience without the limitations of packages or libraries.

With an implementation with the patience factor built-in, we apply this variation of the model to OpenML datasets. We have chosen a maximum of 250 trees per forest, a maximum depth of 100, a patience of 20 trees, and a number of features equal to the square root of the total of features.

Lastly we evaluate the performance of the model using a standard measure for classification algorithms, the accuracy, given by the percentage of examples well predicted on the testing data.

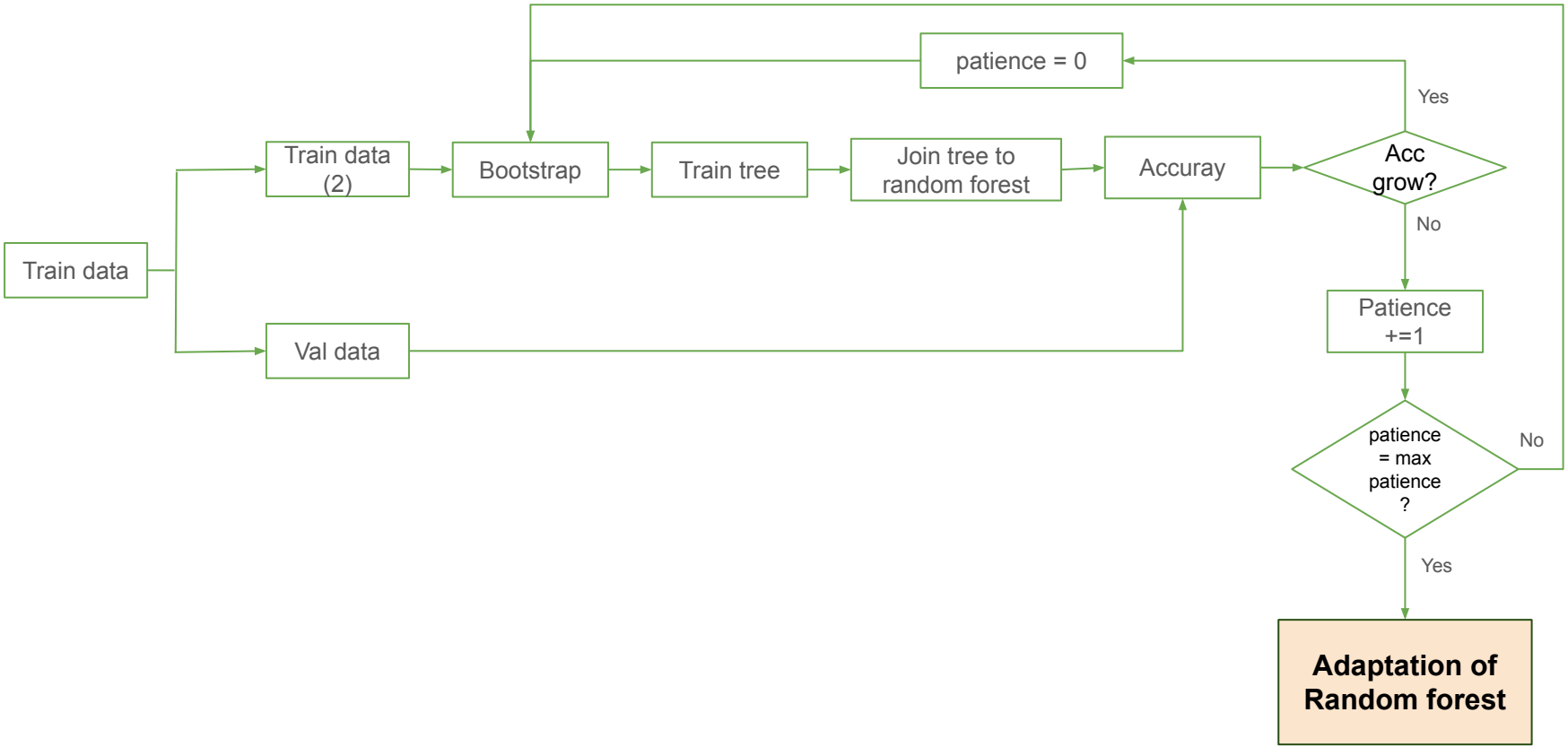
Proposal and Algorithm variation

1. In order to avoid excessive growth of a Random Forest, we propose a new addition to the implementation of the model: patience.

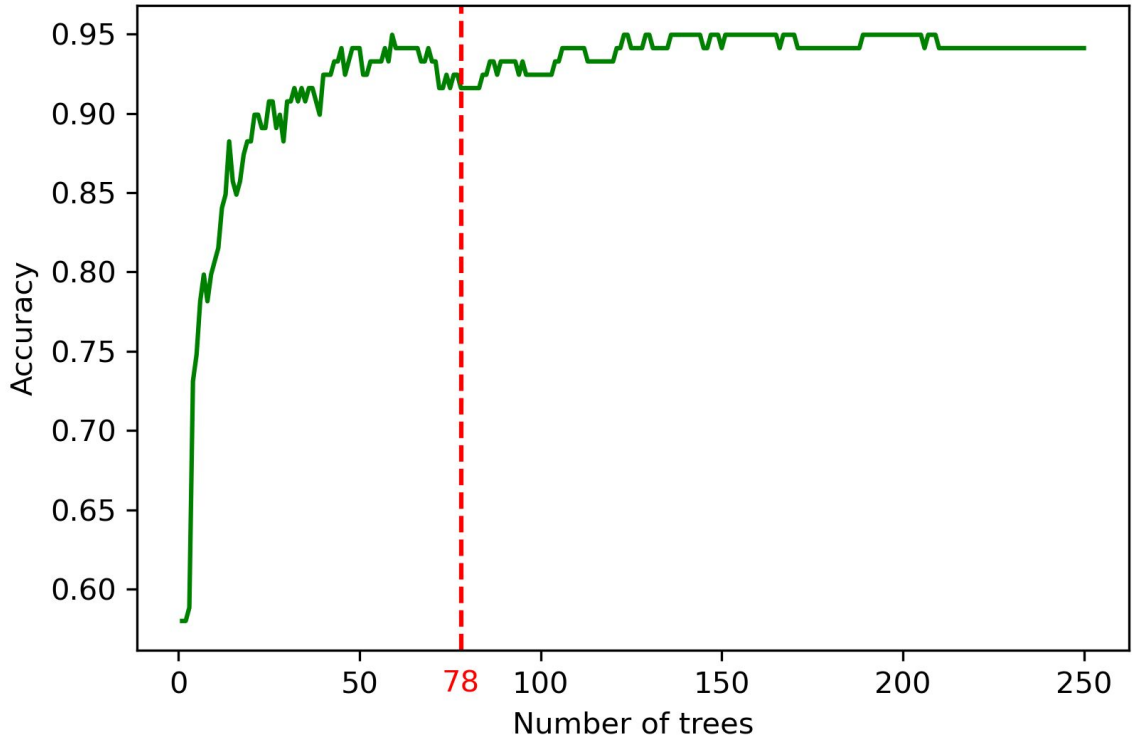
2. This added parameter will continuously evaluate the performance of the forest when each new tree is created and if the performance does not improve for a number of consecutive trees, it stops growing.

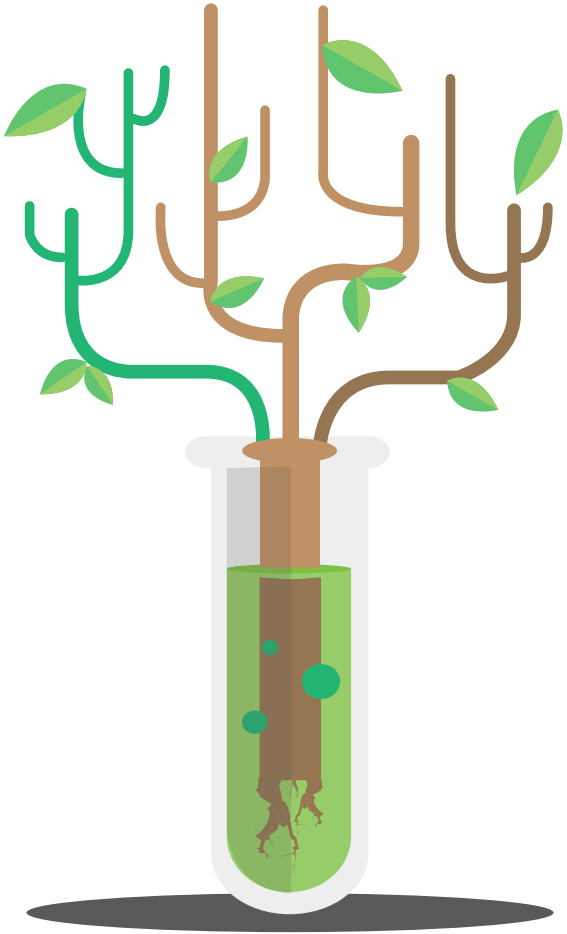
3. We aim to study if this pre-pruning method brings efficiency to the algorithm by avoiding unnecessary trees without sacrificing accuracy and performance.

Adaptation of Random Forest



How the model learns?





Benchmarks

A

Patience = 20

B

Without Patience

C

Scikit-Learn

We tested the performance and results of three different implementations of a Random Forest on the same datasets so we could assess the impact of the addition of patience to the algorithm. The most important comparison is between our implementation with (A) and without (B) patience since they are built on top of the same basis in terms of code. We also compared the results with the Scikit-Learn implementation primarily for accuracy benchmarking.

Experimental Results

Statistics

Metric	Scikit-learn random forest	Random forest implementation	Random forest variation (patience of 20)
Average trees (min/max)	250	250	49 (22/172)
Average accuracy (Z)	**0.871**	0.863 (0.15)	0.859 (0.30)
Standard deviation	0.145	0.149	0.143
Rank	2.41	2.31	**1.76**

1. The scikit learn variation achieved better average accuracy but closer to the variation of random forest

2. Despite the worst average accuracy random forest variation achieved better accuracy in most of the datasets

3. **This implementation allows to save the computation of trees in 80% and the time of processing in 74% (average of 409s to 107s).**

Conclusions and Future Work

1. Our variation of a Random Forest does not perform worse than the standard Random Forest in terms of accuracy.

2. On top of the good performance our implementation of the Random Forest also outperforms in computation and speed.

3. There is work that can be done regarding the optimization of the patience parameter. Some measure of the allowed drop in accuracy before stopping the forest might be taken into account as well.

Attachments

Metric	Random forest implementation	Random forest variation (patience of 30)	Random forest variation (patience of 10)
Average trees (min/max)	250	62 (34/181)	27 (12/51)
Average accuracy (Z)	**0.863**	0.857 (0.21)	0.855 (0.28)
Standard deviation	0.149	0.148	0.149
Metric	Random forest variation (patience of 5)	Random forest variation (patience of 2)	Random forest variation (patience of 1)
Average trees (min/max)	17 (7/35)	10 (4/28)	3 (3/3)
Average accuracy (Z)	0.849 (0.50)	0.842 (0.76)	0.754 (3.79)
Standard deviation	0.147	0.147	0.165

The Z statistic shown next to the average accuracy figures represents the test statistic of a difference of means test between the average accuracy of the random forest implementation without patience and the variation with the respective patience value.

Attachments

- In the previous slide we show the results for similar studies with different patience parameters. The main study was done with a patience of 20 and now we compare the same implementation with patience ranging from 1 to 30. When using a patience as low as 2, it seems that the performance of the random forest is not significantly affected. Although the number of trees decreases substantially, the average accuracy does not.

- We also studied the extreme case where the forest growth is stopped immediately after the first time that the addition of a tree decreases the overall accuracy (patience = 1). In this case the decrease is significant and enough to reject H0 in the hypothesis test, meaning the accuracy is lower than in the model with no patience, statistically speaking.

Attachments

- Additionally, these results show that the accuracy of the Random Forest goes almost straight up until it plateaus. With a margin of only 2 trees, the accuracy of the forest is virtually unchanged so from the first tree until peak accuracy, there were few datasets where the forest growth was stopped due to a “false” signal given by just two consecutive accuracy drawdowns.

- Although it is remarkable that using a patience as low as 2 (average of 10 trees) the accuracy of the model doesn’t seem to be affected on average, we think that this is only possible due to the fact that we only included small/medium datasets. An analysis on specific dataset sizes would probably be a good suggestion of future research that we suspect would show that different datasets demand different values of patience for the accuracy to be optimized.

- Another improvement to the model could include a post-pruning of the trees that didn’t add any value in terms of accuracy to the overall Random Forest. This solution would also consider the patience parameter initially set to train the model. For example, for a patience of x, after training and testing the model, the last x trees to be included could be disregarded since the overall accuracy is either the same or perhaps lower with these last x trees. Therefore, the last x trees are not adding anything of value, on the contrary, on top of potentially decreasing the model’s performance, they are also increasing complexity when applying the model to new data.