Random Deforestation

Henry Stievater, Eric Cai
1/12/2026
Machine Learning 1
Dr. Yilmaz
Quarter 2 Project

Abstract

Decision trees tend to be prone to overfitting, so various pruning methods have been used, including pre-pruning, where the model is pruned during construction, and post-pruning, where the model is pruned after the tree has been made. Both approaches have benefits and drawbacks – pre-pruning is easier but potentially less helpful (risks underfitting), while post-pruning is more expensive but can be more effective. Trees in random forest models are not typically pruned, so we took the novel approach of applying both pre-pruning and post-pruning to trees inside a random forest. We tested this model on two datasets and found no degradation in performance, meaning that pruned random forests have potential to reduce the size and complexity of traditional random forest models.

Introduction

Random forests as a type of ensemble learning are extremely modular, as each decision tree in the forest can vary in many different ways, including but not limited to aspects such as different attributes, instances, size, etc. The tendency of decision trees to overfit is partially alleviated by the sheer number of the trees there are in a random forest along with their variability, but utilizing pruning methods on top of this can resolve the issue of overfitting even more. While different pruning techniques have their own advantages and disadvantages, we wanted to try to bring out the advantages and limit the disadvantages of the pre-pruning/post-pruning techniques.

We tested the Random Deforestation model on two datasets – the Pimas Indians Diabetes dataset, and the Iris dataset. The Pimas Indian dataset was a binary classification problem. The input to the random forest model was health data for an individual, and the classification target was whether or not the individual had diabetes. For the Iris dataset, the inputs to the model are plant measurements for a single plant and the random forest model and the classification target is the type of Iris. We got into more detail about the datasets in the Dataset and Features section. In both cases, we can compare the performance of the model against a random forest model with no pruning. The performance of the model will be evaluated across a different number of decision trees inside the random forest, and performance will be measured using accuracy for the multiclass classification and accuracy, precision, and recall for the binary classification.

Related work

While random forests in general reduce overfitting through bagging and choosing features randomly, the individual decision trees can still become too complicated which amounts to at least some level of overfitting. Many different approaches have been researched to further address this issue.

- https://arxiv.org/abs/2512.10445 (Maximum Risk Minimization with Random Forests)
- [1] F. Freni, A. Fries, L. Kühne, M. Reichstein, and J. Peters, “Maximum Risk Minimization with Random Forests,” 2025, arXiv. doi: 10.48550/ARXIV.2512.10445.

One example of an approach is explored in Maximum Risk Minimization with Random Forests, where the authors created a risk based framework to improve the generalization of the model. Instead of modifying the internal structure of each decision tree, this method tries to estimate the “generalization risk” (or likelihood of overfitting) of each tree and then varies the weight of its

prediction accordingly. This method of controlling the forest at an “ensemble level” allows the model to artificially weigh the “likely to overfit” trees less, which reduces their influence and thus improves performance without explicitly modifying the structure of individual trees.
In contrast to our own approach explored in this project, we focus on controlling the overfitting at the “tree level” by using a hybrid pruning strategy that combines pre-pruning and post-pruning within a random forest. By simplifying the individual trees themselves on top of the ensemble diversity given to us for free by using a random forest, our method tries to reduce overfitting while simultaneously preserving accuracy. This allows us to look into whether this structural pruning can help random forest generalization even more beyond just the ensemble-level risk minimization alone.

- https://proceedings.neurips.cc/paper_files/paper/2016/file/3948ead63a9f2944218de038d8934305-Paper.pdf
- [2] F. Nan, J. Wang, and V. Saligrama, “Pruning Random Forests for Prediction on a Budget,” Jun. 16, 2016, arXiv: arXiv:1606.05060. doi: 10.48550/arXiv.1606.05060.

This paper is more directly related to our work, as it’s studying post-pruning random forests just like us, but their goal they hope to achieve from this pruning is different. While our goal is to try to reduce overfitting in our model for predictive accuracy, their goal is to optimize the cost of classification at prediction time, trying to address a resource constraint. In short, they are trying to reduce cost and time, while we are trying to improve accuracy.

- https://www.sciencedirect.com/science/article/pii/S0306437923001461
- [3] F. A. Khalifa, H. M. Abdelkader, and A. H. Elsaid, “An analysis of ensemble pruning methods under the explanation of Random Forest,” Information Systems, vol. 120, p. 102310, Feb. 2024, doi: 10.1016/j.is.2023.102310.

This paper is related to our work in that it also focuses on reducing the complexity of random forests by pruning, but instead of pruning individual decision trees by removing nodes/leaves, it removes entire trees from the forest. This type of “ensemble pruning” is selected based on the contribution of the trees to the overall model in terms of performance or diversity, so the goal of this is to improve efficiency and reduce redundancy which in a sense is similar to the previous one that’s trying to reduce cost. Our work specifically focuses on controlling overfitting of the trees by simplifying the structure of each individual tree. In short, they prune which trees to be used in the forest at all, while we prune how the trees themselves are built/trimmed down.

- https://arxiv.org/abs/1703.05430
- [4] K. B. Ravi and J. Serra, “Cost-complexity pruning of random forests,” Jul. 19, 2017, arXiv: arXiv:1703.05430. doi: 10.48550/arXiv.1703.05430.

This paper is very similar to our work in that it post-prunes the individual trees inside a random forest as opposed to pruning entire trees from the ensemble like the previous one. The authors use cost-complexity pruning and rely on “out of bag” samples to choose how much each tree should be pruned, so while our goals overlap in reducing overfitting and unnecessary complexity, we focus more on combining pre-pruning and post-pruning rather than just relying on a single pruning method alone. Basically, they study how to choose the best subtree with “out of bag” error, while we study how different pruning strategies affect model generalization.

- https://hal.science/hal-02534421/
- [5] L. Giffon, C. Lamothe, L. Bouscarrat, P. Milanesi, F. Cherfaoui, and S. Koço, “Pruning Random Forest with Orthogonal Matching Trees,” 2020.

This paper also focuses on reducing the size of a random forest after training like us, but it prunes entire trees instead of nodes within individual trees so it's like the third paper we mentioned. The authors use Orthogonal Matching Pursuit to select a subset of trees that have combined predictions which best match the training labels, so it's kind of like a tree-selection problem with learned weights. This reduces model size and can improve performance, but it operates at the "ensemble level", pruning entire trees and not modifying the internal structure of the individual decision trees themselves, so it's different from our work which focuses on structural pruning within each tree by using pre-pruning and post-pruning to control the tree complexity and therefore reduce overfitting. In short, they decide which trees to keep and how to weight them, while we focus on how complex each individual tree should be.

Datasets and Features

The first dataset we used was the Pimas Indians dataset, obtained from Kaggle. The data had the following features:

- Number of times pregnant
- Plasma glucose concentration at 2 hours in an oral glucose tolerance test
- Diastolic blood pressure (mm Hg)
- Triceps skin fold thickness (mm)
- 2-Hour serum insulin test (mu U/ml)
- Body mass index (weight in kg/(height in m)^2)
- Diabetes pedigree function
- Age (years)
- Class variable (1:tested positive for diabetes, 0: tested negative for diabetes)

We did no preprocessing of the data. All attributes were numeric except the class. The decision trees inside the random forest were implemented with sklearn, which can handle continuous data. The data contains 768 instances, each representing one person. We split it into 537 testing instances and 231 testing instances. Here are some example instances:

No.	preg	plas	pres	skin	test	mass	pedi	age	class
1	1.0	147.0	94.0	41.0	0.0	49.3	0.358	27.0	1.0
2	8.0	84.0	74.0	31.0	0.0	38.3	0.457	39.0	0.0
3	8.0	109.0	76.0	39.0	114.0	27.9	0.64	31.0	1.0
4	8.0	188.0	78.0	0.0	0.0	47.9	0.137	43.0	1.0
5	6.0	125.0	78.0	31.0	0.0	27.6	0.565	49.0	1.0
6	4.0	197.0	70.0	39.0	744.0	36.7	2.329	31.0	0.0
7	5.0	95.0	72.0	33.0	0.0	37.7	0.37	27.0	0.0

8	4.0	90.0	0.0	0.0	0.0	28.0	0.61	31.0	0.0

The second dataset we used was the Iris dataset, from the datasets that came installed with Weka. The Iris dataset contains 150 instances, where each instance is a flower. Each instance has four attributes and the class:

- Sepal length
- Sepal width
- Petal length
- Petal width
- Species - Iris setosa, Iris virginica, or Iris versicolor

The species of Iris is our classification target. We split the dataset into 105 training instances and 45 testing instances.
Here is a sample of the data:

No.	Sepal length	Sepal width	Petal length	Petal width	Class
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa

Methods

Random forest is a kind of ensemble learning that uses decision trees. On a high level, random forest creates some number of models and then aggregates each model's prediction to classify an instance. The way each decision tree is created in a random forest is by using a random subset of both samples and features. Samples are chosen randomly with replacement, so a single sample can appear multiple times in a model's training data. However, features are randomly selected without replacement. Since our training datasets were both relatively small, we went with the number of samples for each model equal to the number of samples total, a typical choice. We also went with the number of features for each model as equal to the square root of the total number of features (rounded). This is also a standard choice for random forests.

Our model, random deforestation, is a variation of a standard random forest. Decision tree pruning is a technique of reducing decision tree overfitting, a common problem. There are two kinds of pruning: pre-pruning, and post-pruning. Pre-pruning is less computationally expensive to implement and a simple way to reduce the model's complexity, which helps limit variance. Post-pruning is a more computationally difficult approach of pruning certain leaves after the full tree is constructed, based on an evaluation of each leaf's usefulness in discriminating instances. Random forest trees are not typically pruned, because the trees are smaller due to only receiving a subset of the data. However, recent evidence has emerged

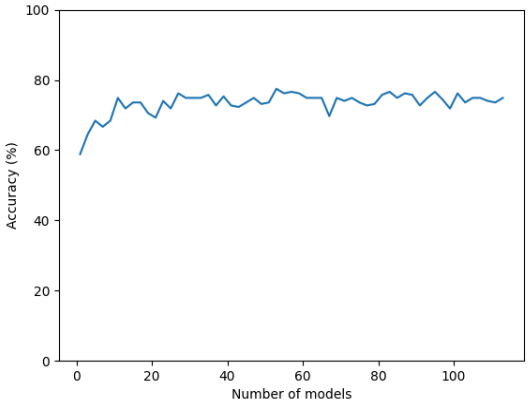
showing that pruning decision trees inside random forests may be useful and produce better results. Our idea was an extension of this technique, by implementing both pre-pruning and post-pruning on the same random forest model, in equal proportions on different trees inside the forest. We call this technique “random deforestation”.

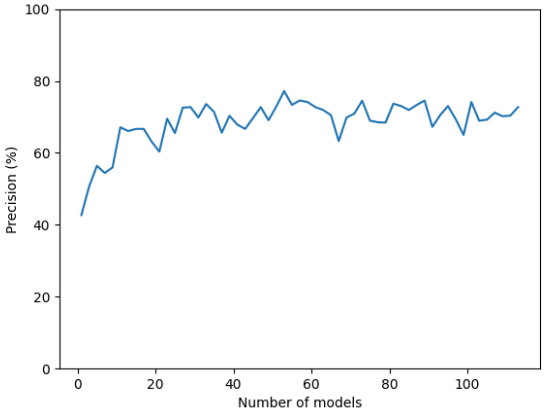
Decision tree pre-pruning was done by limiting the tree’s growth using a “max height”. The maximum height of all pre-pruned trees were determined before the creation of any trees. Post-pruning is slightly more complex and done using a method called cost complexity pruning. Cost complexity pruning is done by recursively removing what is calculated as the tree’s weakest node, where the calculation is based on both the cost (number of training instances misclassified because of that node) and complexity (number of branches from a node). The recursion continues until the total cost-complexity calculation of the tree goes below a threshold supplied at training time, called the ccp_alpha parameter. Appropriate values for ccp_alpha can vary widely across datasets and usually requires testing to determine a good threshold.

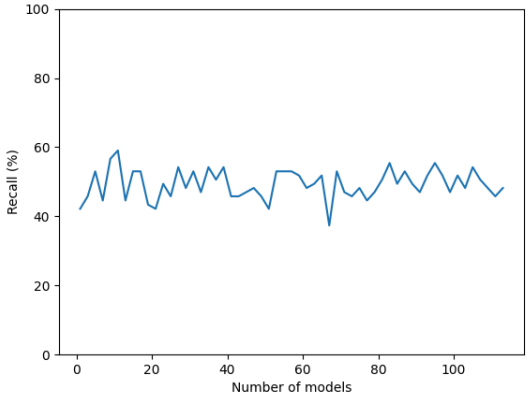
In total, this gave us three numerical parameters for the random deforestation model: the number of trees, the max height threshold, and the ccp_alpha threshold.

Results and discussion

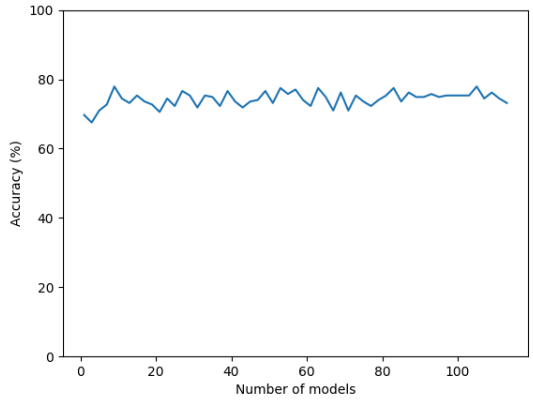
We evaluated many random deforestation models on the Diabetes dataset and the Iris dataset. Since the Diabetes dataset was a binary classification problem, our metrics were accuracy, precision, and recall. Our point of comparison is to a baseline random forest with no pruning at all. The following graphs show how accuracy, precision, and recall change as a result of increasing the number of trees in the random forest.

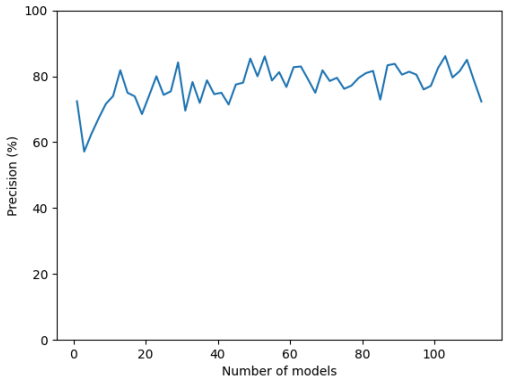


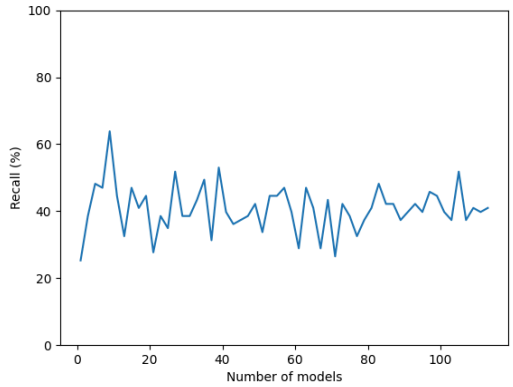




The average accuracy was 73.46%, the average precision was 68.55%, and the average recall was 49.29%. Another comparison model was one with only prepruning, where half the trees were pruned with an aggressive max depth of 2 (the other half were untouched). We chose this value arbitrarily to get a sense of the impact of prepruning. The model performed very slightly better, with an average accuracy of 74.28%, an average precision of 77.40%, and an average recall of 40.71%.

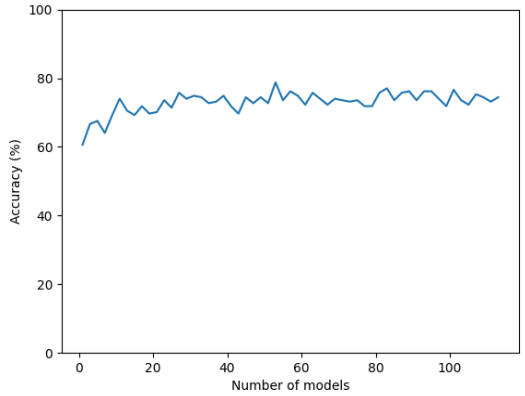


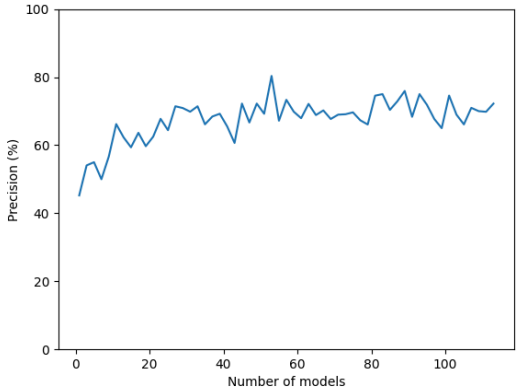


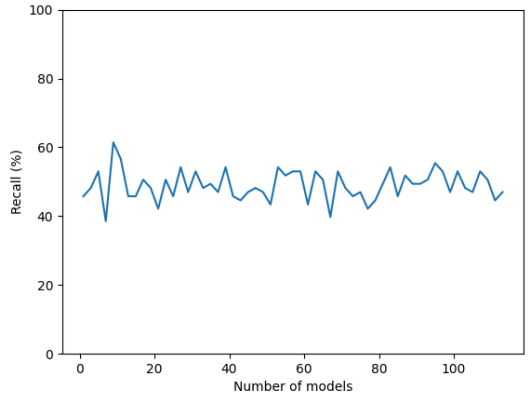


As you can see on the graphs, the model performed about the same as with no pruning in terms of accuracy. However, the pruned model had better precision and lower recall than the unpruned model. In the case of the Diabetes dataset, where detection is more important than false positives, recall is more important than precision, so this pruned model would not be the best choice.

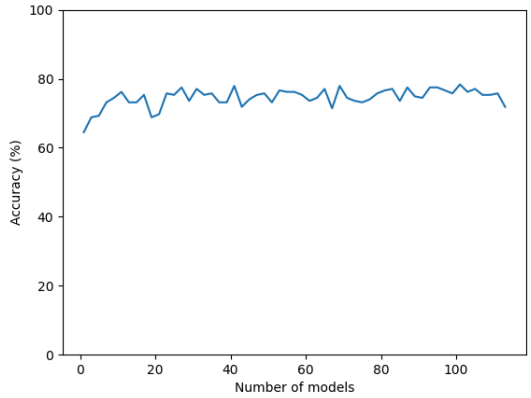
We also tried only post-pruning, with an alpha value of 0.001 – which is about as aggressive as the max height of 2. For this model, the average accuracy was 72.99%, the average precision was 67.52%, and the average recall was 48.95%. This model performed about as well as the baseline random forest model.

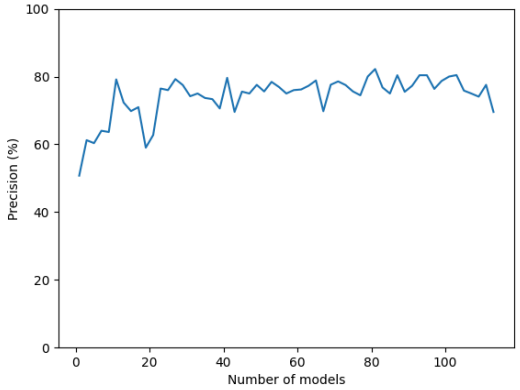


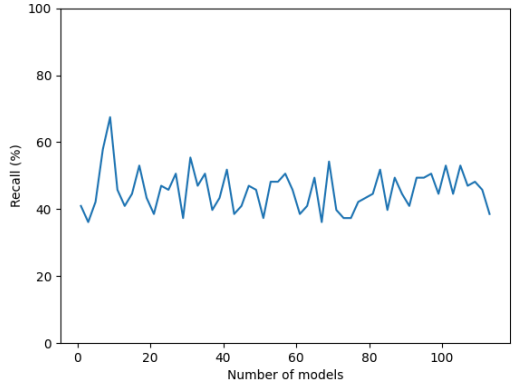




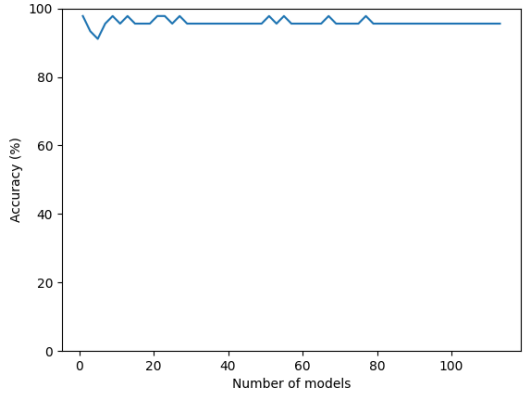
Finally, for the Diabetes dataset, we experimented with many different values for parameters to make the best full random deforested model. The most successful model was with half of the trees prepruned with max height of 6 and half postpruned with an alpha value of 0.004. This model achieved an average accuracy of 74.60%, an average precision of 74.22%, and an average recall of 45.55%. The graphs can be seen below.



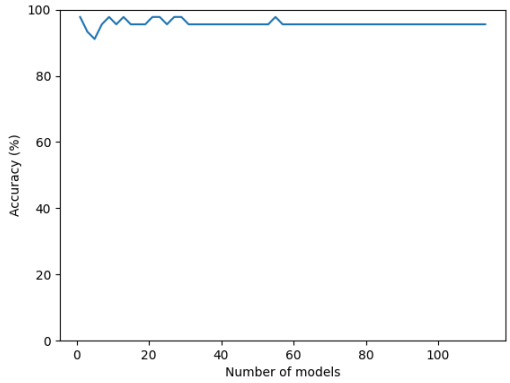




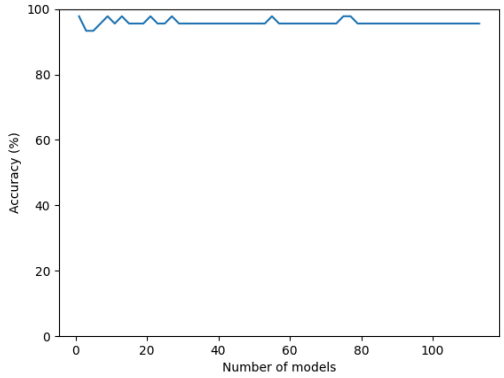
For the Iris dataset, our main metric is accuracy. Random forest performs much better on this dataset, so our baseline accuracy (random forest with no pruning) is much higher, starting at 95.83%.



On the accuracy graph, you can also see that random forest is much more stable than on the Diabetes dataset and that not as many trees were necessary to achieve best performance. Another baseline is the pre-pruning only model. This time, we tested a random forest where half the models were capped at a maximum height of 6. This achieved an average accuracy of 95.75%.



To see the effect of post-pruning, we did a test with only half the models post-pruned, using an alpha threshold of 0.03. The results were very similar, with an accuracy of 95.79%:



In fact, every combination of pre- and post–pruning produced nearly identical results, with an average accuracy of around 95.7%. The only exceptions were when trees were pruned with a maximum height less than 6 or a ccp_alpha more than 0.03, as at this point the trees began to be too aggressively pruned and underfit to the training data. All pruning less aggressive than that gave essentially the same results.

Overall, our random deforested model performed on the same level as a baseline random forest with no pruning. Performance gains were slight and may be within the normal random variation of the model's performance. However, these results should not be taken to mean that pruning random forest decision trees is a waste of time. As with many things in computer science, there is a tradeoff between time and space. For example, I saved to file the largest Iris random forest models, unpruned and pruned versions. The unpruned model was about 300kb of space, compared to 160kb for the pruned model, and both achieved essentially the same accuracy. Additionally, the shorter height of the pruned trees means that classification of instances using the model would be faster for a deforested random forest model than the more typical version with full trees. Although the additional time of pruning the trees must be taken into account, we believe our random deforested model may have important benefits that are useful in certain situations – when model size is critical, or when a large number of instances need to be classified quickly.

Conclusions

In summary, random deforestation was not a large success. Across both datasets, Diabetes and Iris, performance gains were marginal compared to the baseline random forest. However, performance was not hurt even with moderately aggressive pruning thresholds, meaning that deforested models were much smaller than full random forests without sacrificing performance. This could have benefits in specific use cases when model size and classification speed are worth the sacrifice of the computational cost to prune the trees. In the future, additional research could explore alternative pruning strategies. However, it may be that since the nature of instance and attribute selection in random forest is designed to prevent overfitting, decision tree pruning in a random forest may be trying to solve a problem that does not exist.

References

[1] F. Freni, A. Fries, L. Kühne, M. Reichstein, and J. Peters, “Maximum Risk Minimization with Random Forests,” 2025, arXiv. doi: 10.48550/ARXIV.2512.10445.

[2] F. Nan, J. Wang, and V. Saligrama, “Pruning Random Forests for Prediction on a Budget,” Jun. 16, 2016, arXiv: arXiv:1606.05060. doi: 10.48550/arXiv.1606.05060.

[3] F. A. Khalifa, H. M. Abdelkader, and A. H. Elsaid, “An analysis of ensemble pruning methods under the explanation of Random Forest,” Information Systems, vol. 120, p. 102310, Feb. 2024, doi: 10.1016/j.is.2023.102310.

[4] K. B. Ravi and J. Serra, “Cost-complexity pruning of random forests,” Jul. 19, 2017, arXiv: arXiv:1703.05430. doi: 10.48550/arXiv.1703.05430.

[5] L. Giffon, C. Lamothe, L. Bouscarrat, P. Milanesi, F. Cherfaoui, and S. Koço, “Pruning Random Forest with Orthogonal Matching Trees,” 2020.

https://github.com/hstievat/ml-q2-project/

Contributions

Eric did the sections “Introduction” (the first half) and “Literature review”. Henry did the sections “Abstract”, “Datasets and Features”, “Methods”, “Results and discussion”, and “Conclusions”. Henry did the coding and model testing.