Protocol presentation: Tuning strategies for hyperparameters of random forests

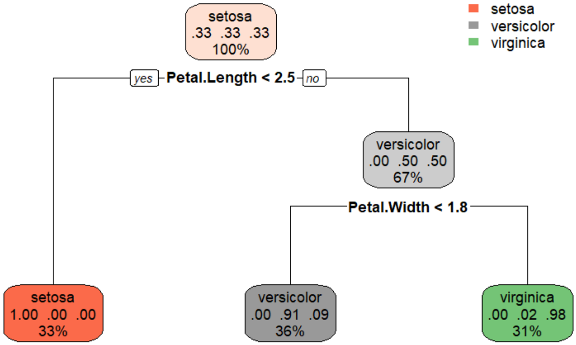
Judith Neve

Department of Methodology and Statistics, UU
Julius Center, UMCU

Outline

What is a random forest?

A single decision tree:



What is a random forest?

Main idea: combine many decision trees to classify an observation

- For each tree:
  - For each split: randomly select m of the candidate predictors
- An observation is assigned to a class by majority vote

Hyperparameters

Lots of settings to choose!

- Number of trees
- Number of candidate predictors
- Proportion of the sample used for fitting the tree
- Sample with or without replacement
- Minimum node size
- Split rule
- Respect unordered factors

Tuning hyperparameters

Options:

- Software defaults
- Educated guess
- Tune them to identify the best ones for the data

Aims

1. Which (combination of) hyperparameters have the most influence on model performance?
2. Which metric is the most closely related to model performance?
3. Which hyperparameter search algorithm is the best for model performance and runtime?

Data-generating mechanism: scenarios

Varying across the datasets:

- Number of candidate predictors p: 8, 16, 32
- Event fraction EF: 0.1, 0.3, 0.5
- Sample size n: 0.5, 1, 2 times the minimum required sample (as defined by Riley, 2020)

1000 datasets for each scenario $\rightarrow$ 27000 datasets per study

Data simulated under **logistic regression with strong interactions**

Data-generating mechanism: coefficients

For each combination of event fraction EF and number of candidate predictors p:

1. Simulate predictors: 10,000 draws from a p-variate normal distribution with mean 0, variance 1, covariance 0.2
2. Optimise the intercept and slopes to obtain the desired EF and an AUC of 0.7
   - The strength of the effect is the same for all predictors
3. Using these, optimise $0.25p$ interaction slopes to obtain an AUC of 0.8
   - The strength of the interaction is the same for all interactions

Data-generating mechanism: data simulation

For each scenario:

1. Simulate predictors: n draws from a p-variate normal distribution with mean 0, variance 1, covariance 0.2
2. Simulate outcomes: n draws from a Bernoulli distribution. The probabilities will be computed using the optimised coefficients and simulated predictors.

Estimands

Model performance:

- Discrimination (AUC)
- Calibration (calibration in the large, calibration slope)
- **For study 3**: runtime

Methods: Study 1 - hyperparameter combinations

Probst et al. (2019): number of candidate predictors and sample fraction have the largest effect on accuracy

- All combinations that include these two predictors + no tuning: 33
- Fit a random forest tuning each combination on each dataset: 891,000 tuning procedures

Optimised metric: accuracy
Hyperparameter search algorithm: grid search

Methods: Study 1 - hyperparameter combinations

Get a table of the form:

Data simulation settings			Hyperparameters tuned	AUC	Calibration slope	CIL
p	Event fraction	Sample size				
8	0.1	*0.5	replace	NA	NA	NA
16	0.1	*0.5	replace	NA	NA	NA
8	0.1	*1	replace + sample fraction	NA	NA	NA
16	0.1	*1	replace + sample fraction	NA	NA	NA

1. Extract top 3 combinations giving the best of each metric
2. Is there a combination that appears in each metric's top 3? If so, select that combination. If there are multiple, give them each a value (weighted sum of position in top 3 & performance metric)
3. If not, take the product of calibration slope (transformed: 1 - absolute deviation from 1) and AUC and select the highest **such that both metrics are at least as good as the default hyperparameters**

Methods: Study 2 - optimisation metric

We tune the hyperparameter combination selected from study 1.
Fit a random forest on each dataset, optimising each of the following candidate metrics:

- Accuracy [default]
- Kappa [as is an option via caret]
- Brier score [as is the default in tuneRanger]
- AUC [as is an option via tuneRanger]
- Logarithmic loss [as is an option via tuneRanger]

i.e., each dataset is tuned 5 times: 135,000 tuning procedures.
Hyperparameter search algorithm: grid search

Methods: Study 2 - optimisation metric

Get a table of the form:

Data simulation settings			Optimisation metric	AUC	Calibration slope	CIL
p	Event fraction	Sample size				
8	0.1	*0.5	Accuracy	NA	NA	NA
16	0.1	*0.5	Accuracy	NA	NA	NA
8	0.1	*1	Kappa	NA	NA	NA
16	0.1	*1	Kappa	NA	NA	NA

1. Extract the optimisation metric that leads to the best of each performance metric
2. Is this consistent? If so, select that combination.
3. If not, take the product of calibration slope (transformed: 1 - absolute deviation from 1) and AUC and select the highest **such that both performance metrics are at least as good as that for accuracy**

Methods: Study 3 - hyperparameter search algorithms

We tune the hyperparameter combination selected from study 1.
We optimise the metric selected from study 2. Fit a random forest on each dataset, using each of the following candidate search algorithms:

- Model-free search algorithms:
  - Grid search [caret]
  - Random search [caret]
- Bayesian optimisation: SMAC [tuneRanger]
- Multifidelity: undefined - have not yet identified an R package
- Metaheuristic: genetic algorithm [GA]

i.e., each dataset is tuned 5 times: 135,000 tuning procedures.

Methods: Study 3 - hyperparameter search algorithms

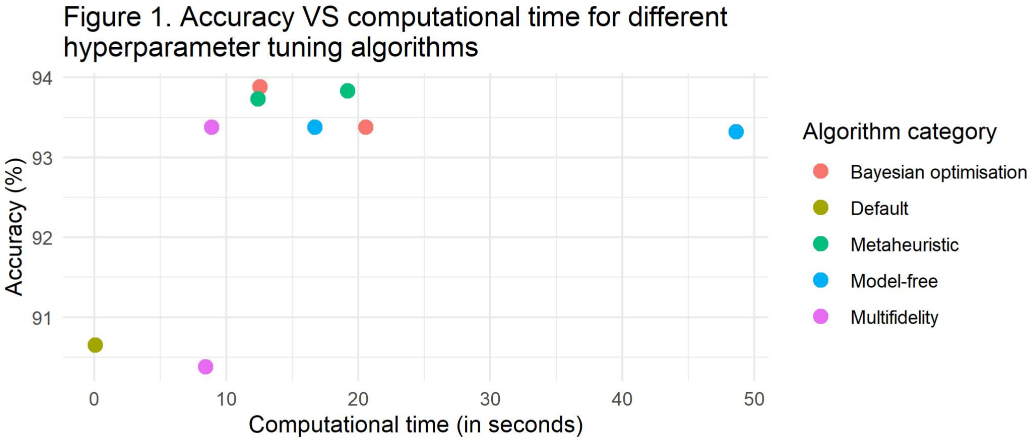
Get a table of the form:

Data simulation settings			Hyperparameter search algorithm	AUC	Calibration slope	CIL	**Time**
p	Event fraction	Sample size					
8	0.1	*0.5	Grid search	NA	NA	NA	NA
16	0.1	*0.5	Random search	NA	NA	NA	NA
8	0.1	*1	Grid search	NA	NA	NA	NA
16	0.1	*1	Random search	NA	NA	NA	NA

As of yet unsure how to assess the best one.

Methods: Study 3 - hyperparameter search algorithms

Get a figure of the form:



Still to consider

- Error handling:
  - Degenerate datasets
  - Non-converging calibration slopes
- Study 3:
  - Search algorithms to include
  - How to draw conclusions
- Runtime: pilot studies