

Introduction to Machine Learning

Random Forest Basics



Learning goals

- Know how random forests are defined by extending the idea of bagging
- Understand general idea to decorrelate trees
- Understand effects of hyperparameters
- RFs and overfitting

MOTIVATION

CARTs offer several appealing features:

- **Interpretability:** Easy to understand and explain
- **Invariance to rank-preserving transformations:**
E.g., unaffected by scaling or shifting of features
- **Versatility:** Work on categorical and numerical data
- **Robustness to missing values:** Can work with missings



Despite these benefits, CARTs are not without drawbacks:

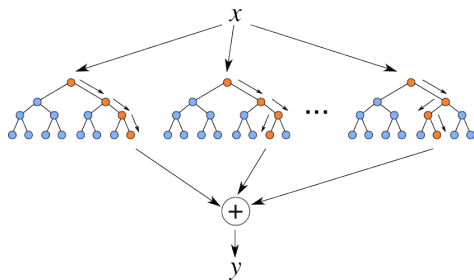
► Hastie, Tibshirani, and Friedman 2009

"Trees have one aspect that prevents them from being the ideal tool for predictive learning, namely inaccuracy."

RANDOM FORESTS

► Breiman 2001

- RFs use **bagging with CARTs as BLs**
- **Random feature sampling** decorrelates the base learners
- **Fully expanded trees** further increase variability of trees



INTUITION BEHIND DECORRELATION

- Since bootstrap samples are similar, models $\hat{b}^{[m]}$ are correlated, affecting the variance of an ensemble \hat{f}
- We would like variance to go down linearly with ensemble size, but because of correlation we cannot really expect that
- Assuming $\text{Var}(\hat{b}^{[m]}) = \sigma^2$, $\text{Corr}(\hat{b}^{[m]}, \hat{b}^{[j]}) = \rho$, semi-formal analysis, without proper analysis of prediction error:

$$\begin{aligned}\text{Var}(\hat{f}) &= \text{Var}\left(\frac{1}{M} \sum_{m=1}^M \hat{b}^{[m]}\right) = \frac{1}{M^2} \left(\sum_{m=1}^M \text{Var}(\hat{b}^{[m]}) + 2 \sum_{m < j} \text{Cov}(\hat{b}^{[m]}, \hat{b}^{[j]}) \right) \\ &= \frac{1}{M^2} \left(M\sigma^2 + 2 \frac{M(M-1)}{2} \rho\sigma^2 \right) = (1 - \rho) \frac{\sigma^2}{M} + \rho\sigma^2\end{aligned}$$

- Ensemble variance is “convex-combo of linear-reduction and no-reduction, controlled by ρ ”
- Maybe we can decorrelate trees, to reduce ensemble variance?
And get less prediction error?



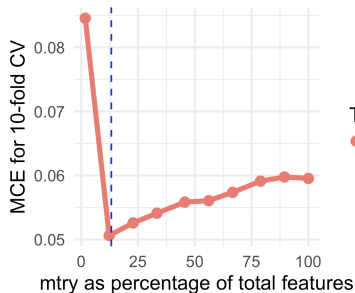
RANDOM FEATURE SAMPLING

RFs decorrelate trees with a simple randomization:

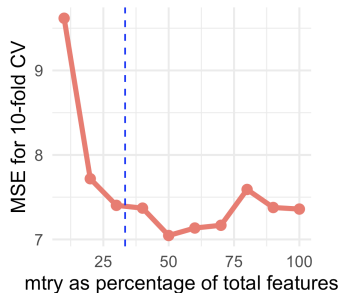
- For each node of tree, randomly draw $m_{\text{try}} \leq p$ features (m_{try} = name in some implementations)
- Only consider these features for finding the best split
- Careful: Our previous analysis was simplified! The more we decorrelate by this, the more random the trees become!
This also has negative effects!



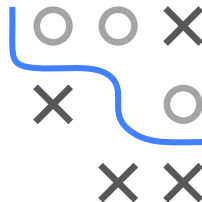
EFFECT OF FEATURE SAMPLING



Task
● spam



Task
● mtcars



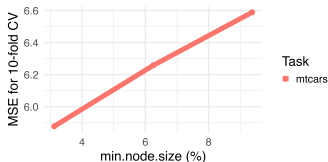
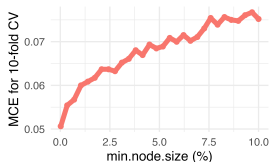
- Optimal `mtry` typically larger for regression than for classification
- Good defaults exist, but still most relevant tuning param
- Rule of thumb:
 - Classification: $mtry = \lfloor \sqrt{p} \rfloor$
 - Regression: $mtry = \lfloor p/3 \rfloor$

TREE SIZE

In addition to `mtry`, RFs have two other important HPs:

- Min. nr. of obs. in each decision tree node

Default (ranger): `min.node.size = 5` ▶ Breiman 2001

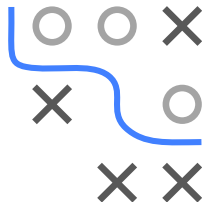


- Depth of each tree

Default (ranger): `maxDepth = ∞`

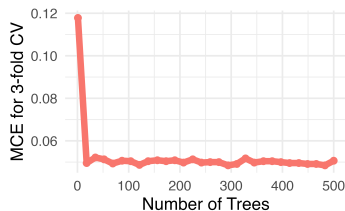
- There are more alternative HPs to control depth of tree: minimal risk reduction, size of terminal nodes, etc.

RF usually use fully expanded trees, without aggressive early stopping or pruning, to further **increase variability of each tree**. ▶ Louppe 2015

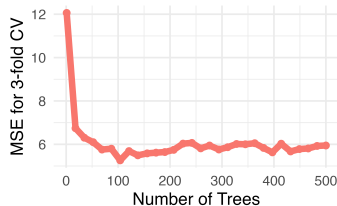


ENSEMBLE SIZE

- RFs usually better if ensemble is large ► Breiman 2001
- But: Increases computational costs, and diminishing returns
- 100 or 500 is a sensible default
- Can also inspect the OOB error (see later)

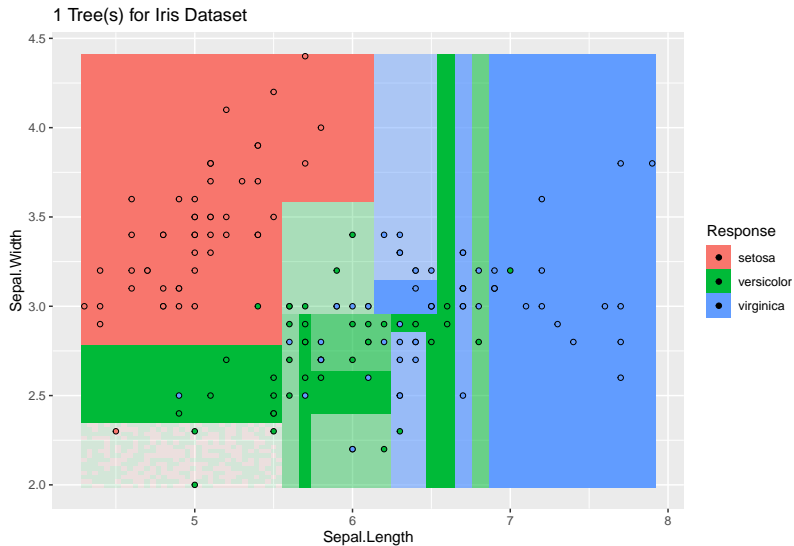


Task
■ spam

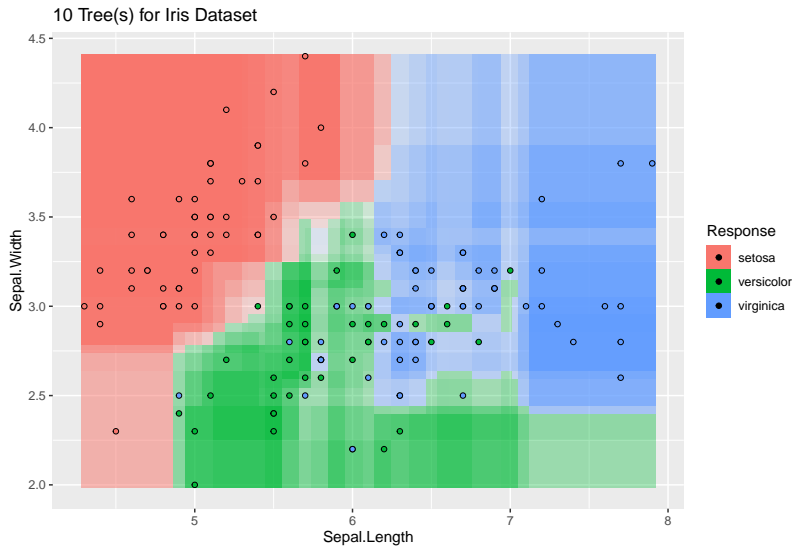


Task
■ mtcars

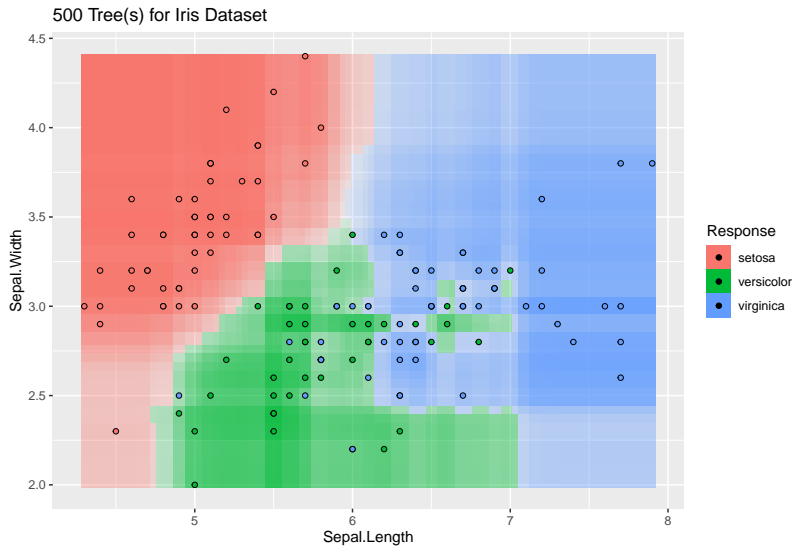
EFFECT OF ENSEMBLE SIZE



EFFECT OF ENSEMBLE SIZE



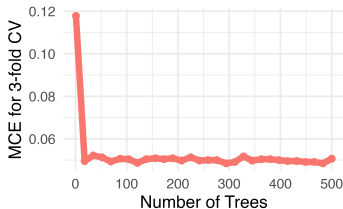
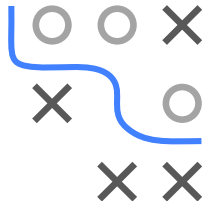
EFFECT OF ENSEMBLE SIZE



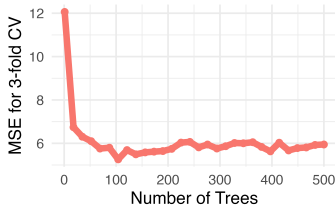
CAN RF OVERFIT?

► Probst and Boulesteix 2018

- Just like any other learner, RFs **can** overfit!
- However, RFs generally **less** prone to overfitting than individual CARTs.
- Overly complex trees can *still* lead to overfitting!
If most trees capture noise, so does the RF.
- But randomization and averaging helps.



Task
■ spam

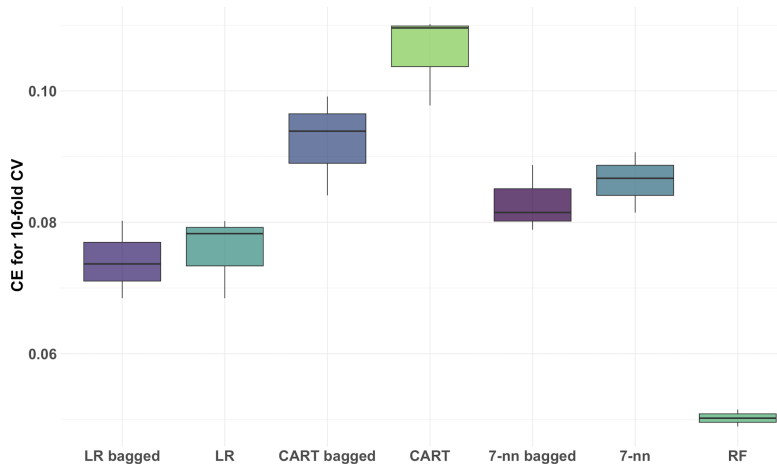


Task
■ mtcars

Since each tree is trained *individually and without knowledge of previously trained trees*, increasing `ntrees` generally reduces variance **without increasing the chance of overfitting!**

RF IN PRACTICE

Benchmarking bagged ensembles with 100 BLs each on spam
versus RF (ntrees = 100, mtry = \sqrt{p} , minnode = 1),
we see how well RF performs!



DISCUSSION

Advantages:

- Most advantages of trees also apply to RF: not much preprocessing required, missing value handling, etc.
- Easy to parallelize
- Often work well (enough)
- Works well on high-dimensional data
- Works well on data with irrelevant “noise” variables



Disadvantages:

- Same extrapolation problem as for trees
- Harder to interpret than trees
(but many extra tools are nowadays available for interpreting RFs)
- Implementation can be memory-hungry
- Prediction can be computationally demanding for large ensembles