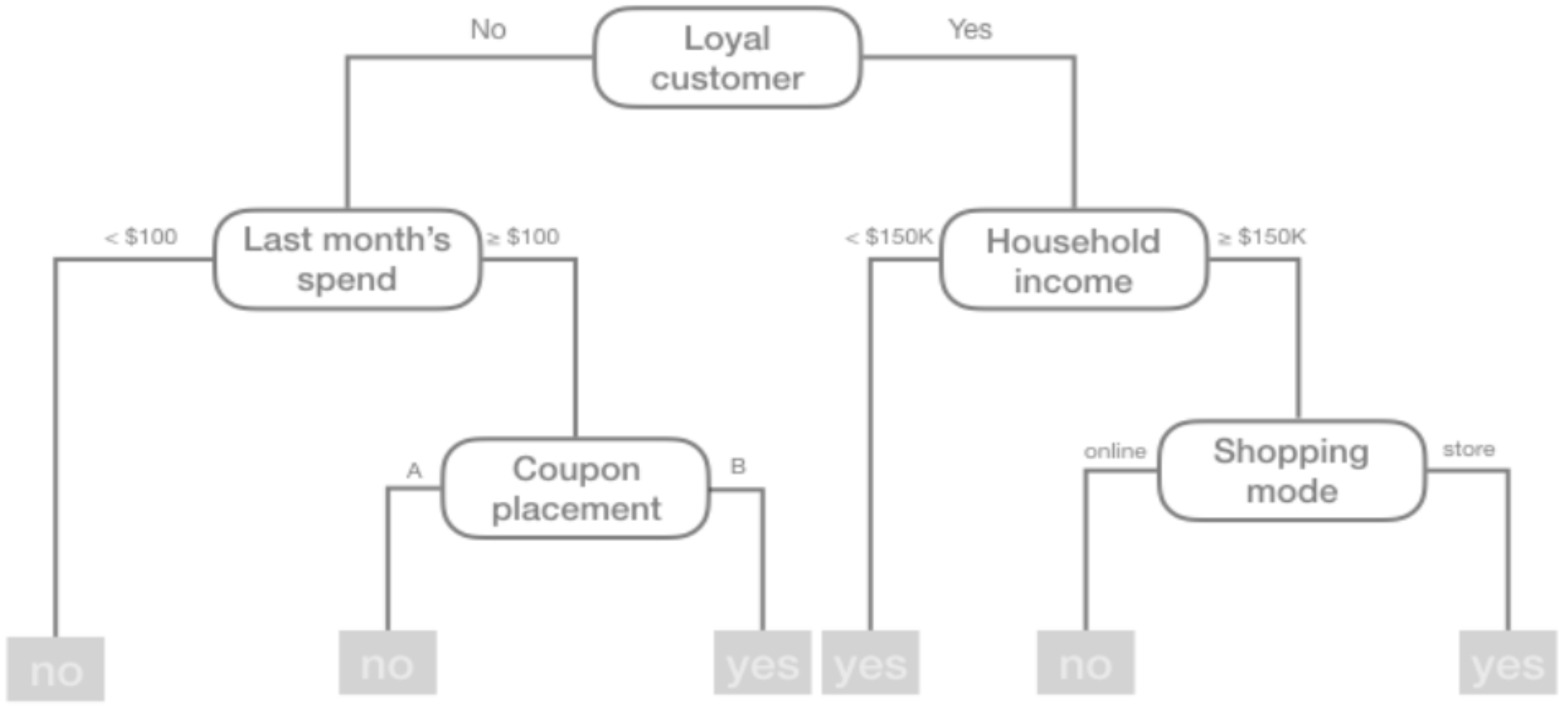
CS135
Introduction to Machine Learning

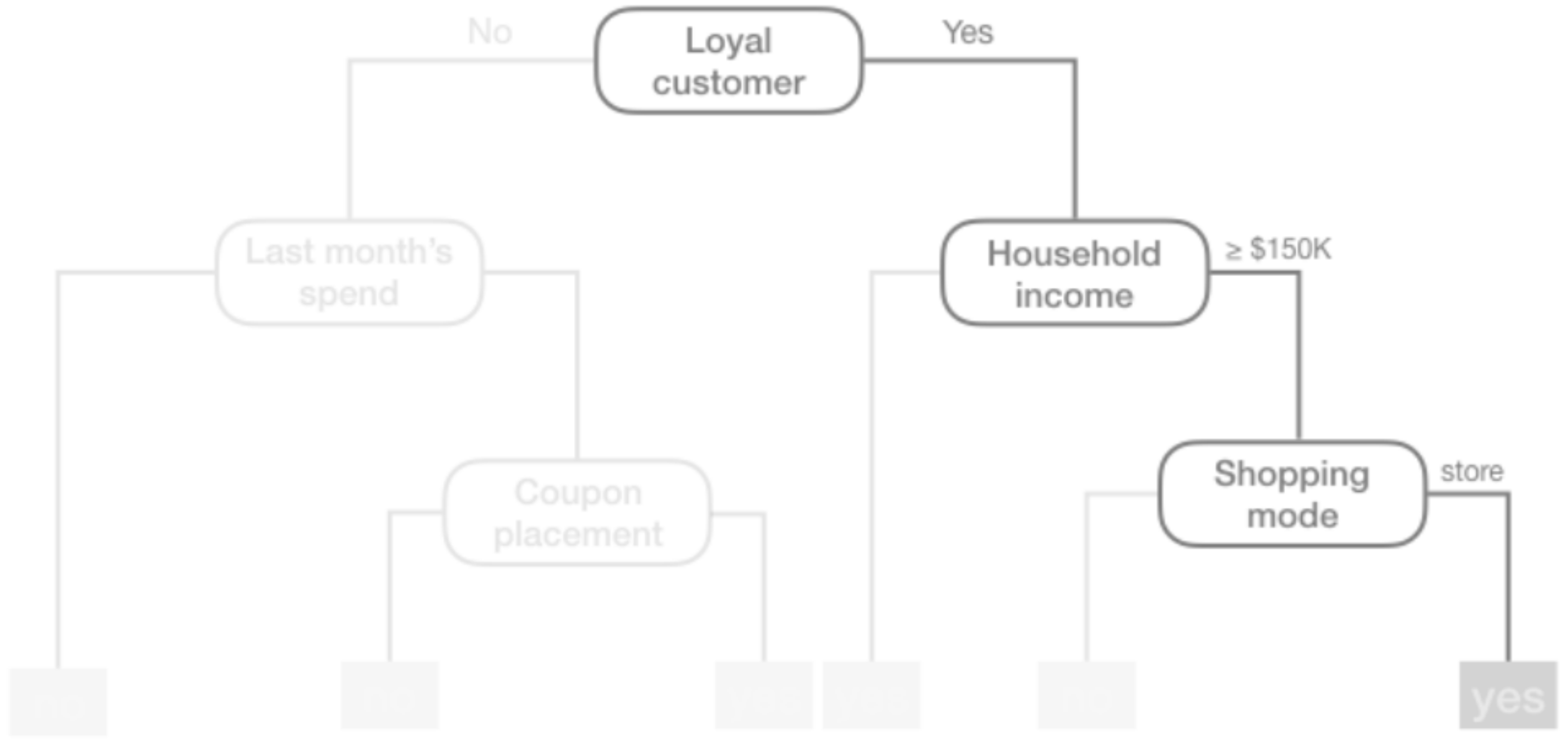
Lecture 10: Classifiers (Decision Tree, Random Forest)

Example Decision Tree – Retail Data



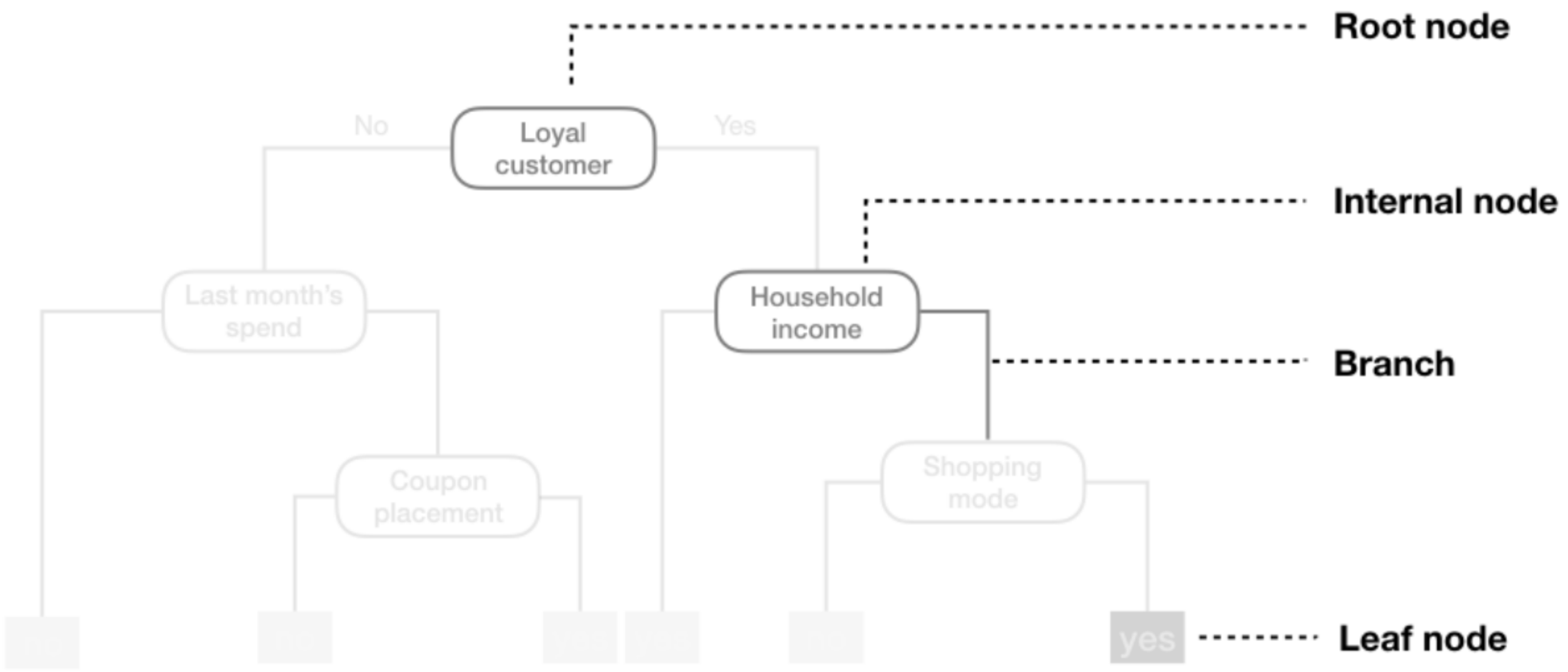
Will a customer redeem a coupon

Decision Tree - Ruleset Model



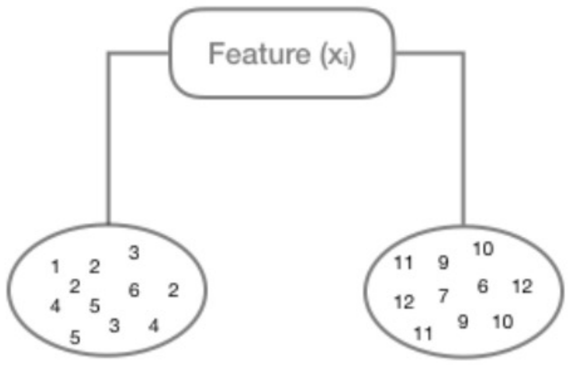
```
if Loyal Customer = Yes and Household income >= $150K and Shopping mode = store then coupon redemption = Yes
```

Terminology



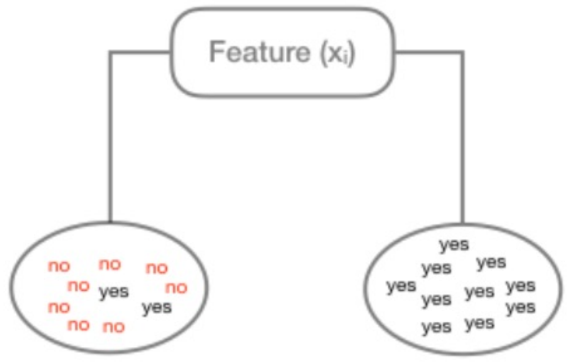
Best Binary Partitioning

Regression tree



Prediction:	3.36	9.7
SSE:	24.55	36.1

Classification tree



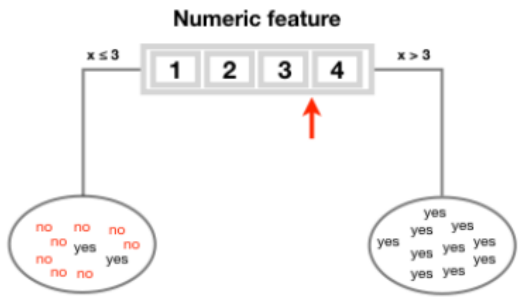
Feature (x_i)
no no no no yes no no yes no no
yes yes yes yes yes yes yes yes yes yes

Prediction:	no	yes
Gini:	.16	0

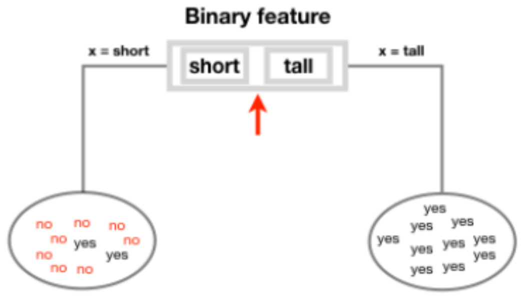
Objective: Minimize disimilarity in terminal nodes

Best Binary Partitioning

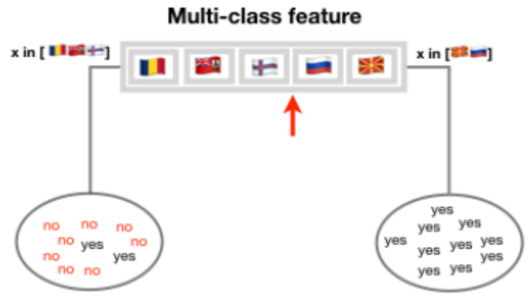
- **Numeric feature**: Numeric split to minimize loss function


Numeric feature
x ≤ 3
x > 3
1
2
3
4
no no no no yes no no yes no no
yes yes yes yes yes yes yes yes yes yes

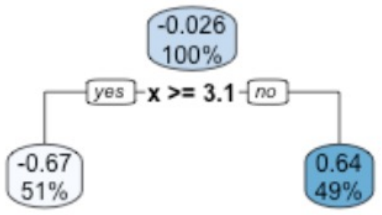
- **Binary feature**: Category split to minimize loss function


Binary feature
x = short
x = tall
short
tall
no no no no yes no no yes no no
yes yes yes yes yes yes yes yes yes yes

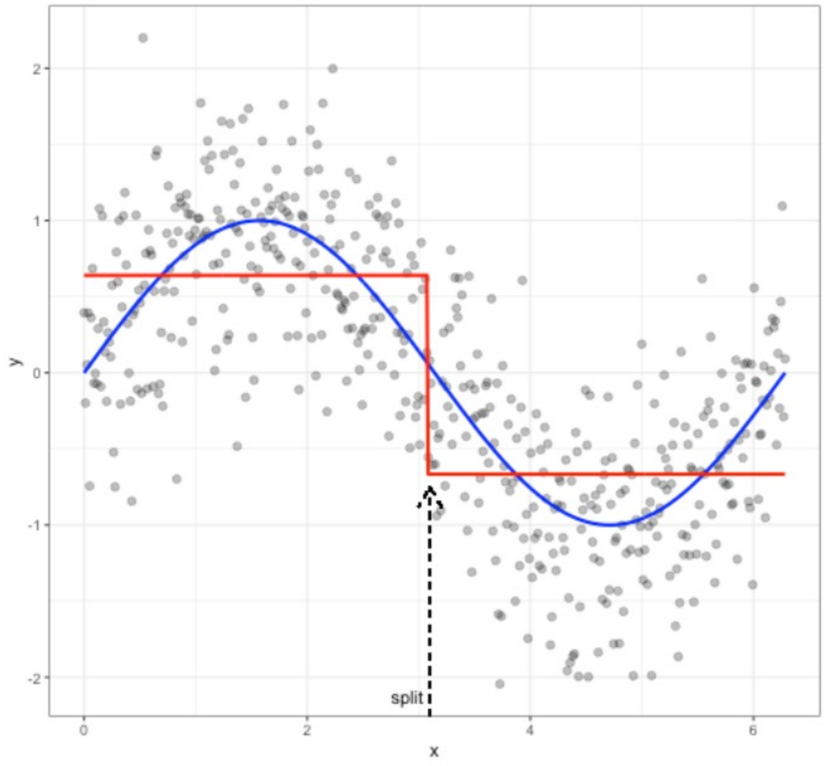
- **Multiclass feature**: Order feature classes based on mean target variable (regression) or class proportion (classification) and choose split to minimize loss function


Multi-class feature
x in [ ]
x in [ ]
no no no no yes no no yes no no
yes yes yes yes yes yes yes yes yes yes

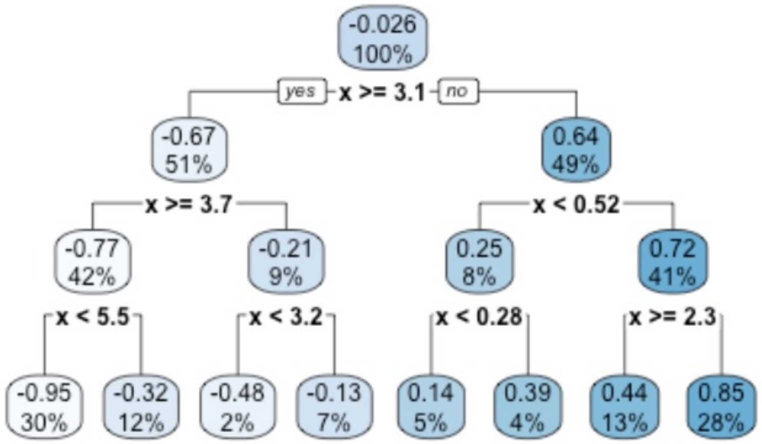
Tree Depth = 1 (Decision Stump)

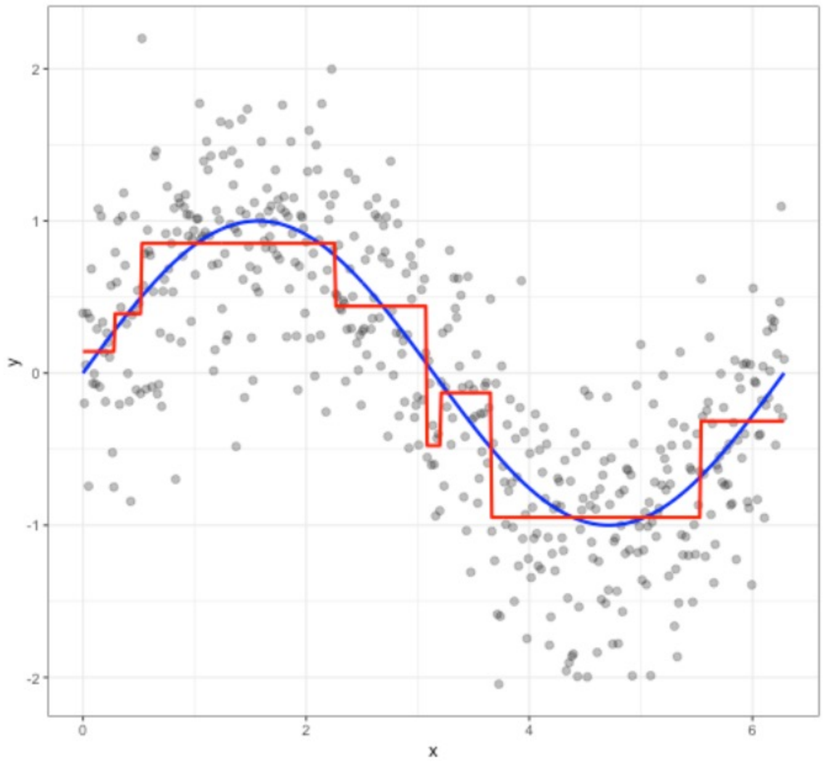


```
##
## Model formula:
## y ~ x
##
## Fitted party:
## [1] root
## |   [2] x >= 3.07863: -0.665 (n = 255, err = 95.5)
## |   [3] x < 3.07863: 0.640 (n = 245, err = 75.9)
##
## Number of inner nodes:    1
## Number of terminal nodes: 2
```



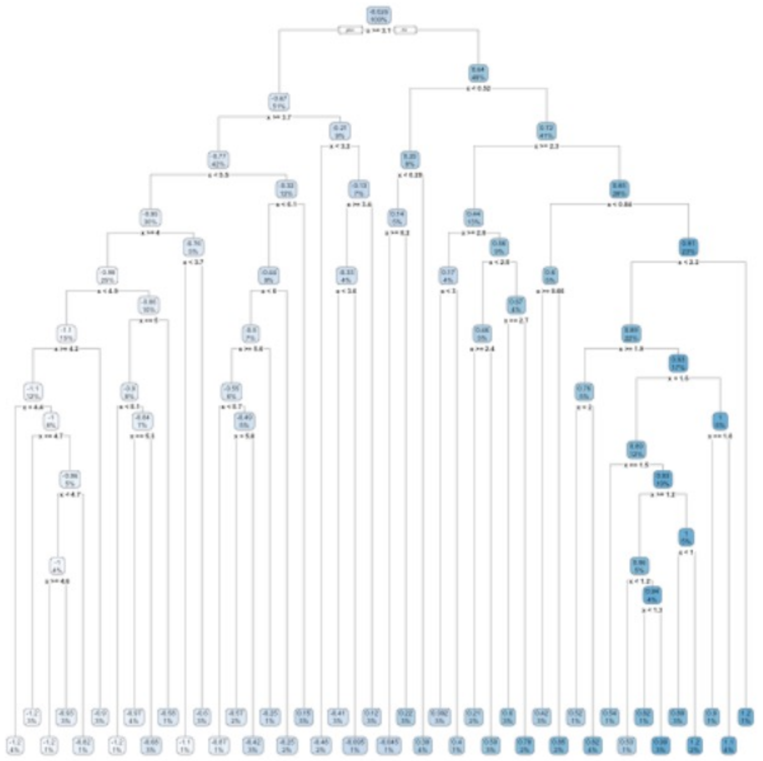
Tree Depth = 3

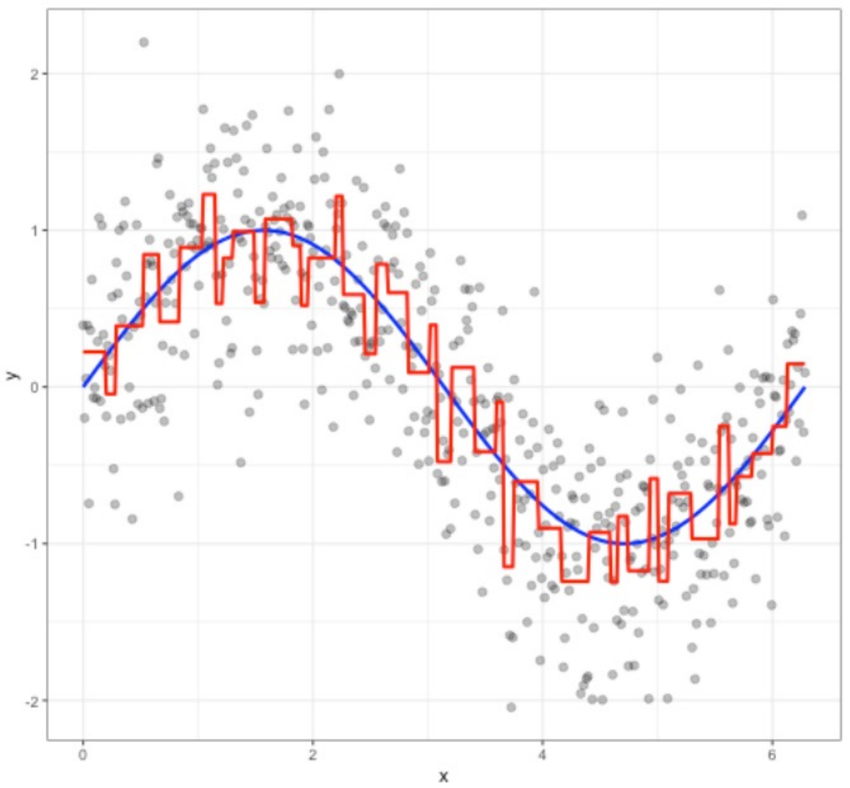




```
##
## Model formula:
## y ~ x
##
## Fitted party:
## [1] root
```

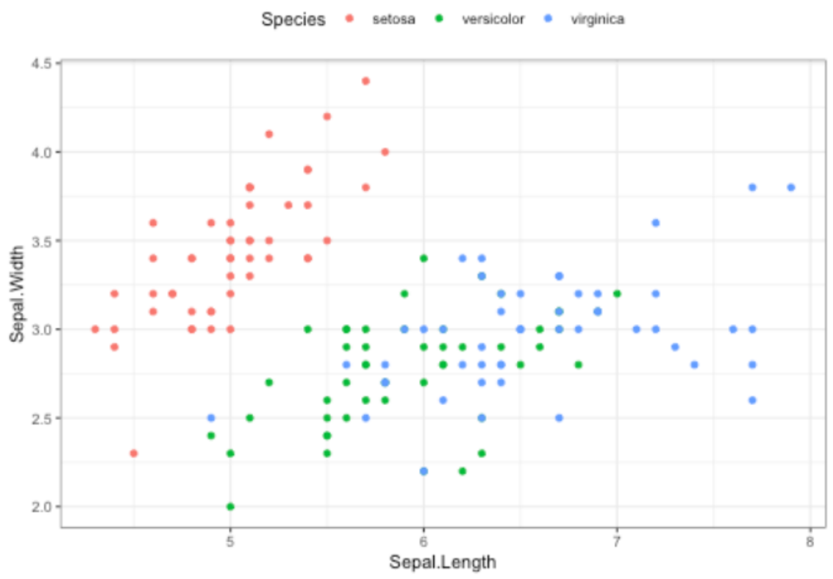
Tree Depth = 20 (Complex Tree)





Two Predictor Decision Boundaries

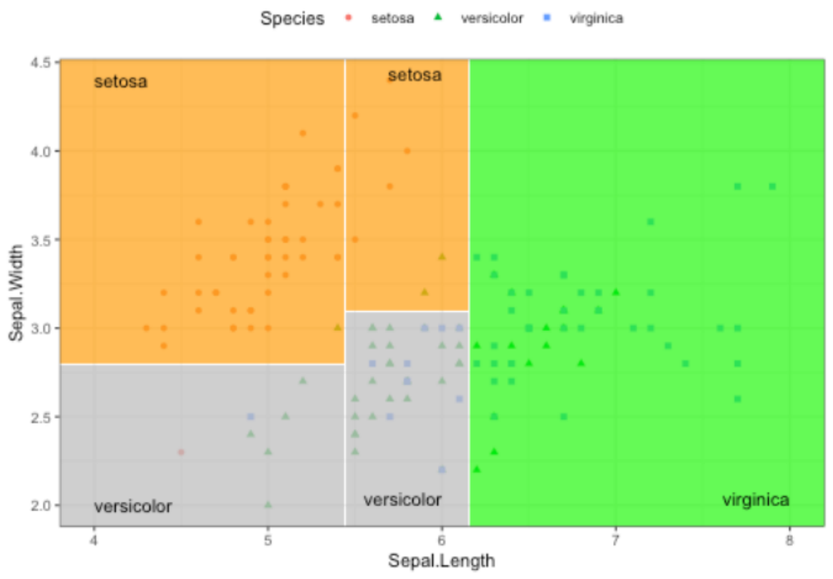
Classification problem: Iris data



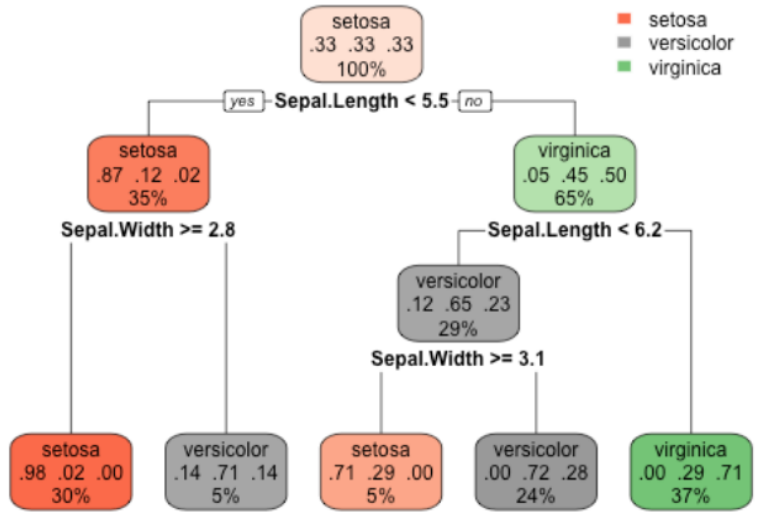
Tufts

Two Predictor Decision Boundaries

Classification problem: Iris data

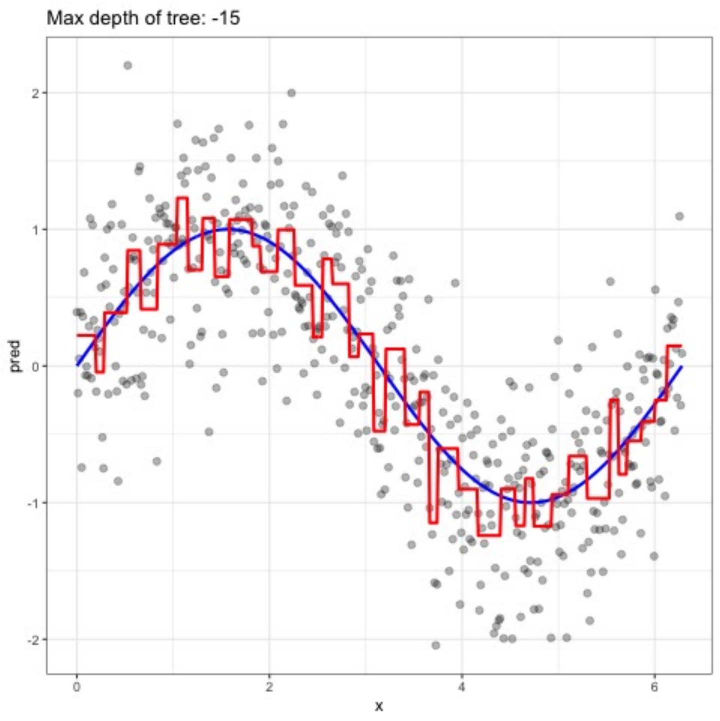


Classification tree

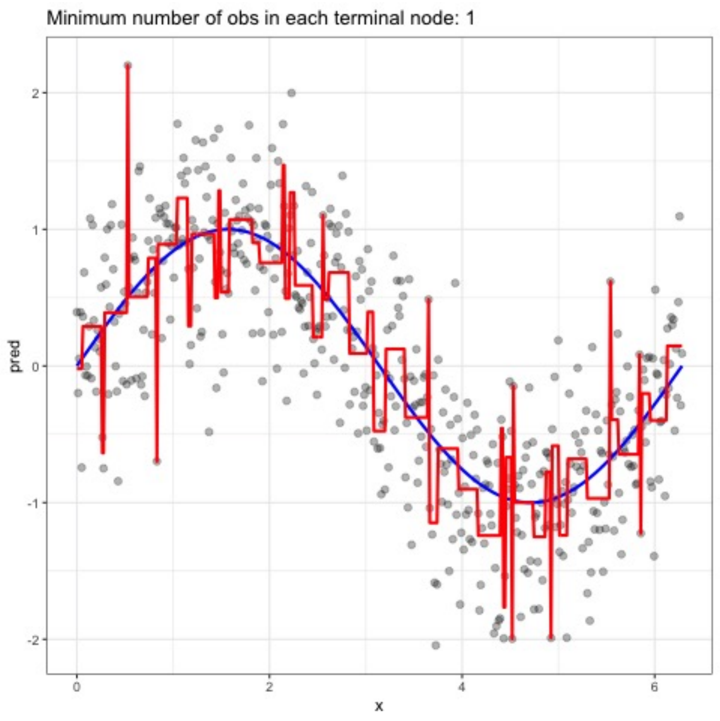


Minimize overfitting: Early stopping

Limit tree depth: Stop splitting after a certain depth



Minimum node "size": Do not split intermediate node which contains too few data points

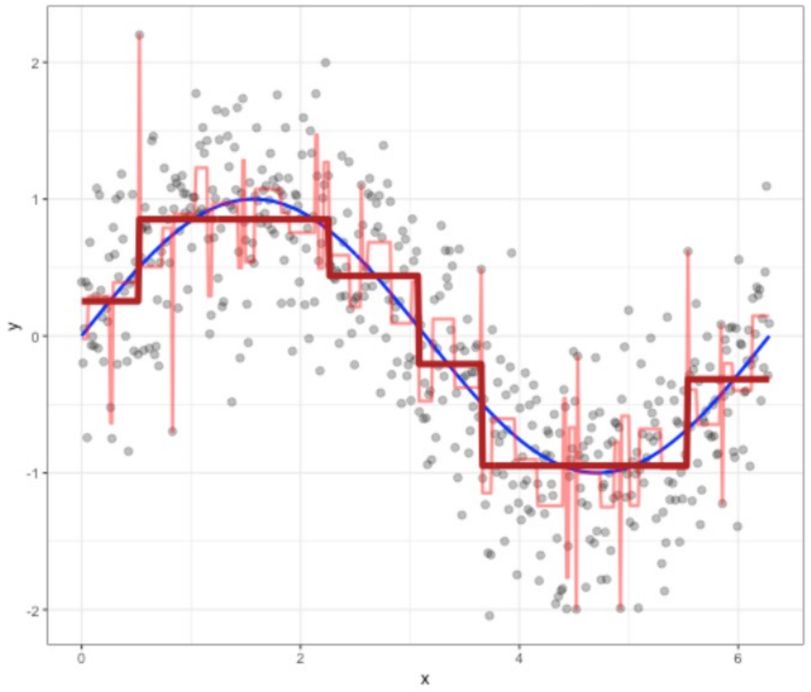


Minimize overfitting: Pruning

1. Grow a very large tree
2. Prune it back with a *cost complexity parameter* (α) × number of terminal nodes (T) to find an optimal subtree:
   - Very similar to lasso penalty in regularized regression
   - Large α = small tree
   - Small α = large tree
   - Find optimal α with cross validation

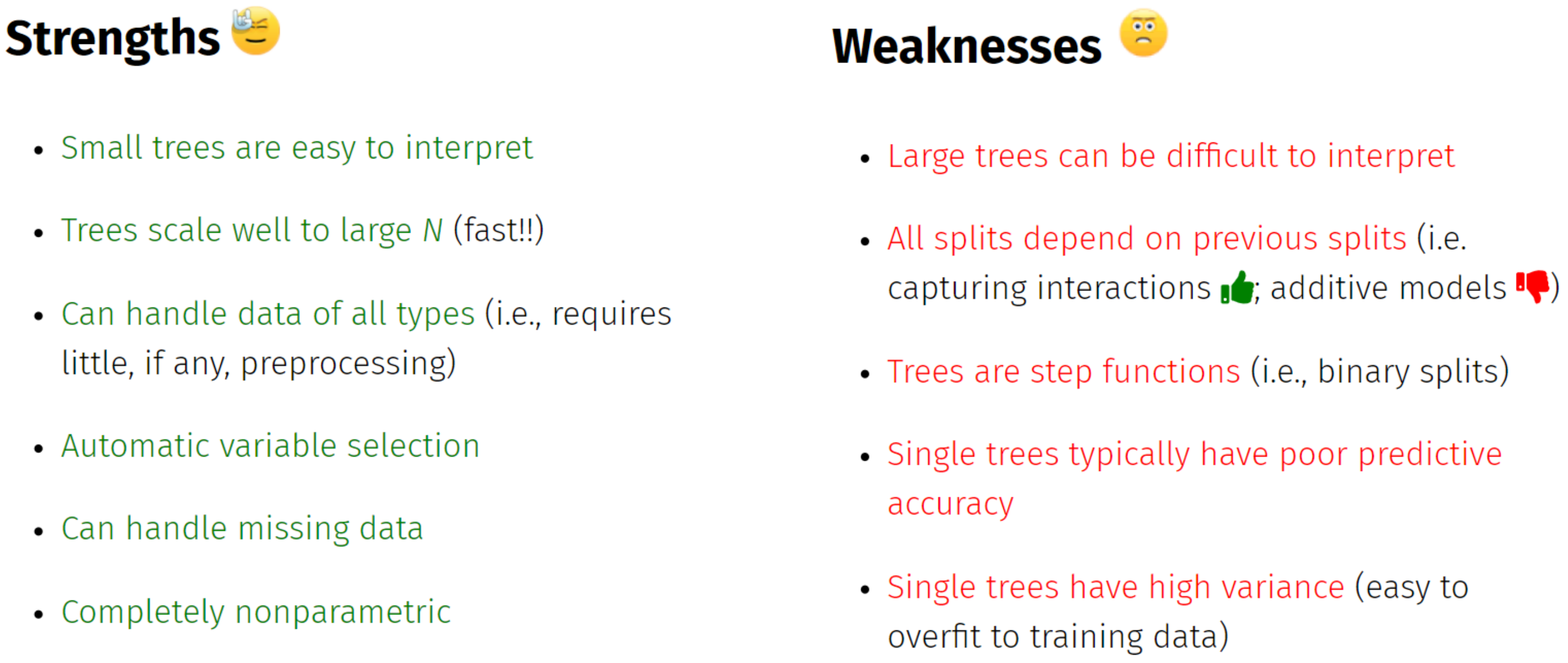
$$\text{minimize: loss function} + \alpha|T|$$

Deep trees overfit



Penalize depth to generalize

Decision Tree - Strengths & Weaknesses



Strengths

- Small trees are easy to interpret
- Trees scale well to large *N* (fast!!)
- Can handle data of all types (i.e., requires little, if any, preprocessing)
- Automatic variable selection
- Can handle missing data
- Completely nonparametric

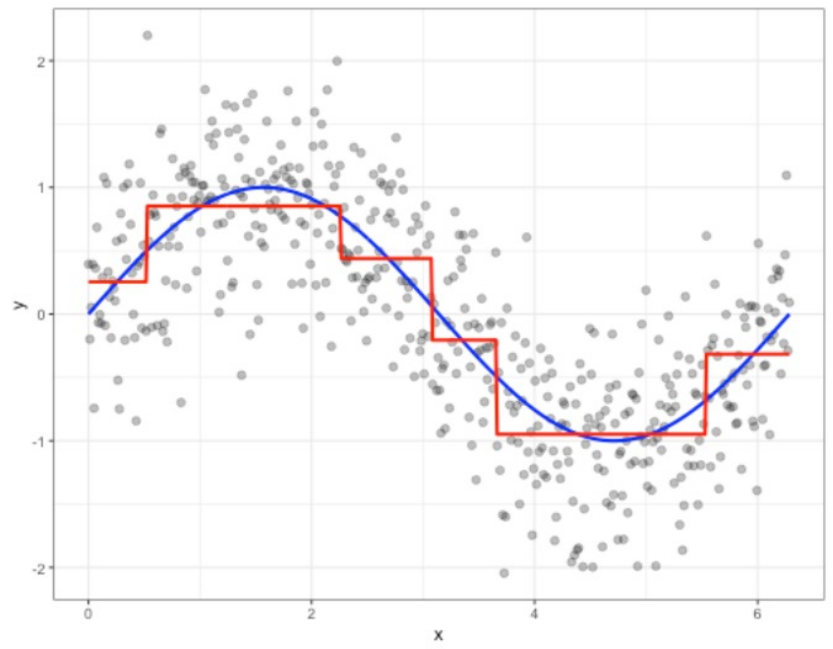
Weaknesses

- Large trees can be difficult to interpret
- All splits depend on previous splits (i.e. capturing interactions 👍; additive models 👎)
- Trees are step functions (i.e., binary splits)
- Single trees typically have poor predictive accuracy
- Single trees have high variance (easy to overfit to training data)

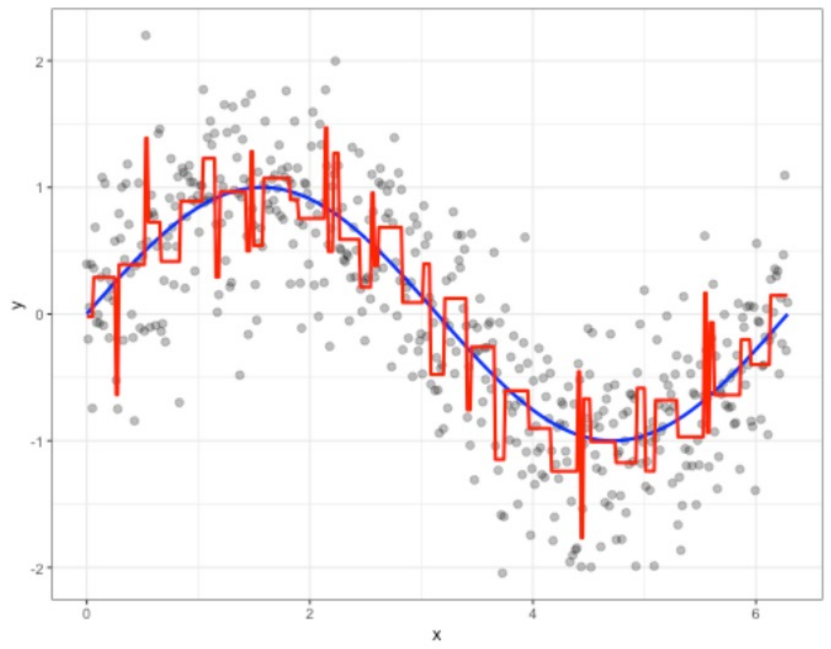
Tufts

The Problem with Single Decision Trees

Single pruned trees are poor predictors



Single deep trees are noisy



Bagging uses this high variance to our advantage ↑

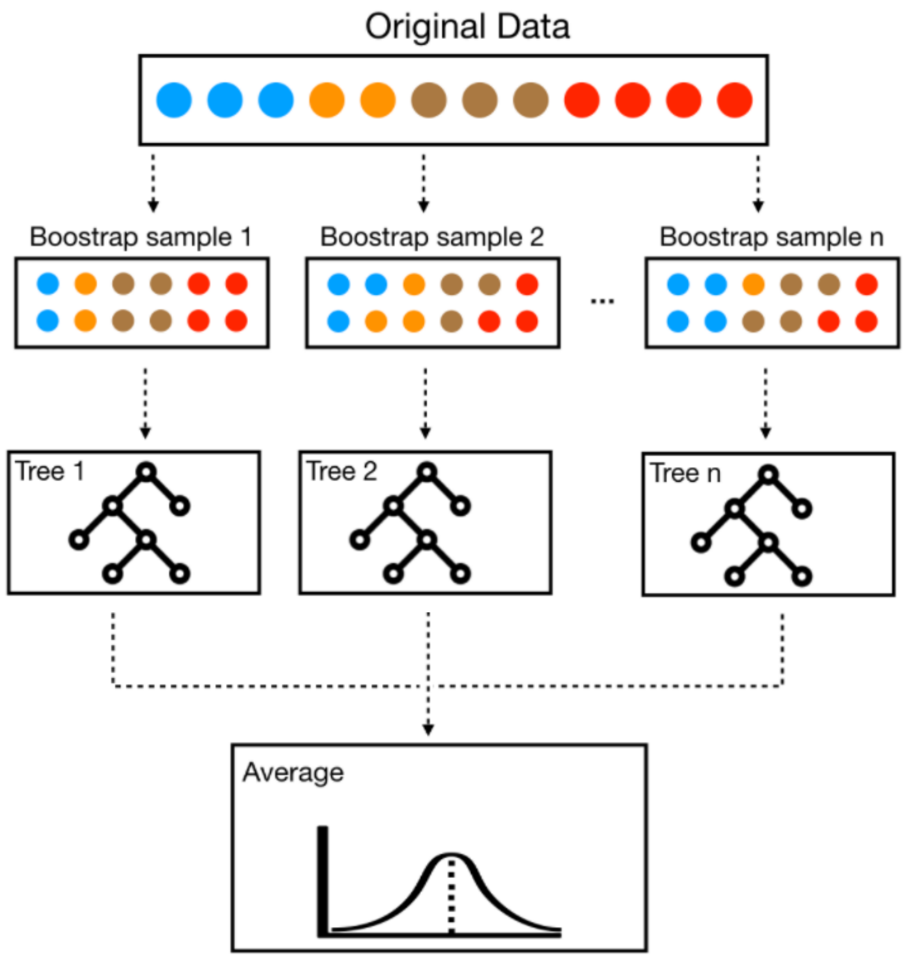
Bagging : Bootstrap Aggregating : wisdom of the crowd

1. Sample records w replacement (aka "bootstrap" the training data)

Sampling is selecting a subset of items from a vast collection of items.
Bootstrap = Sampling with replacement. It means a data point in a drawn sample can reappear in future drawn samples as well.

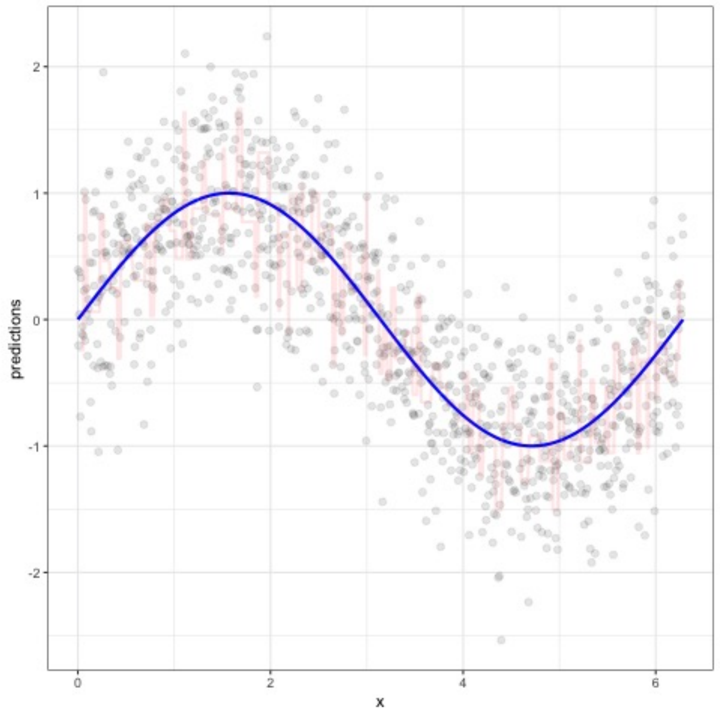
2. Fit an overgrown tree to each resampled data set

3. Average predictions

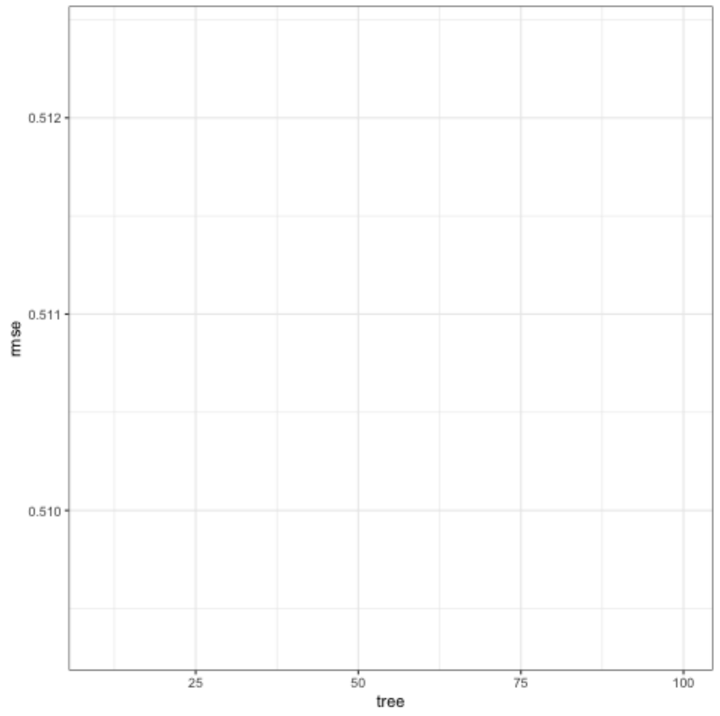


Bagging : Bootstrap Aggregating : wisdom of the crowd

As we add more trees…

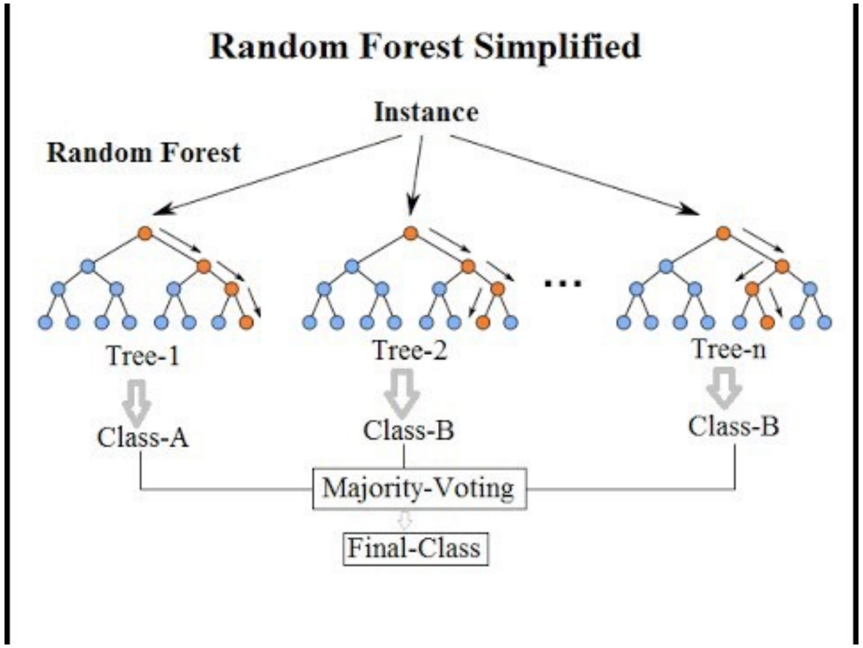


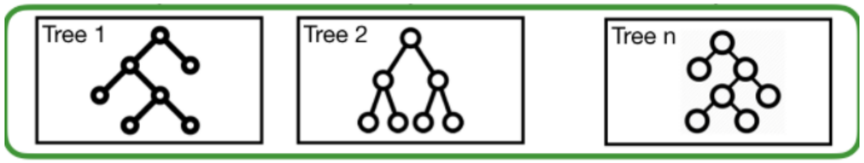
our average prediction error reduces



Random Forest

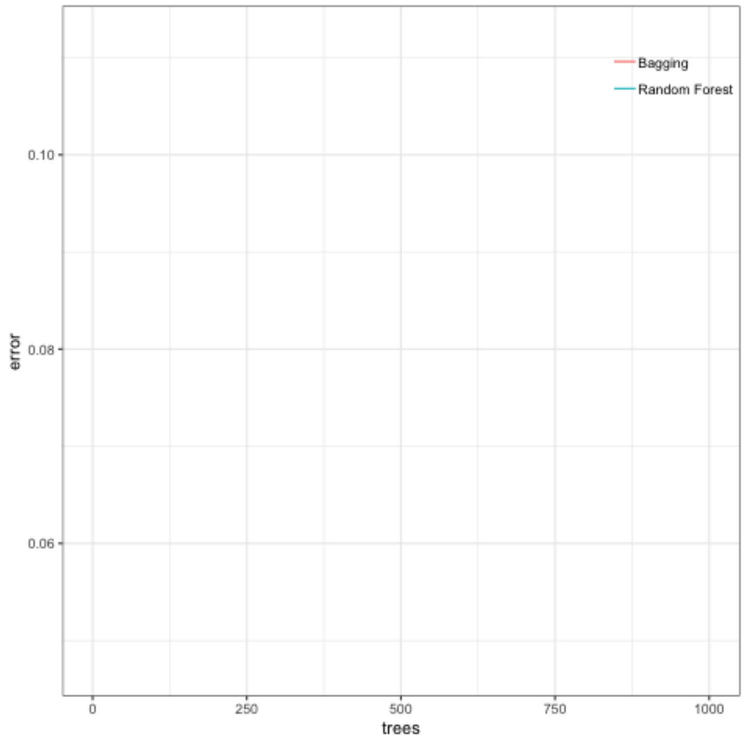
- Random forest is identified as a collection of decision trees. Each tree estimates a classification, and this is called a "vote". Ideally, we consider each vote from every tree and chose the most voted classification (Majority-Voting).
- Random Forest follow the same bagging process as the decision trees but each time a split is to be performed, the search for the split variable is limited to a random subset of m of the p attributes (variables or features) aka Split-Attribute Randomization :
  - classification trees: $m = \sqrt{p}$
  - regression trees: $m = p/3$
  - m is commonly referred to as mtry
- Random Forests produce many unique trees.





Bagging vs Random Forest

- Bagging introduces randomness into the rows of the data.
- Random forest introduces randomness into the rows and columns of the data
- Combined, this provides a more diverse set of trees that almost always lowers our prediction error.



Split-Attribute Randomization : Prediction Erro

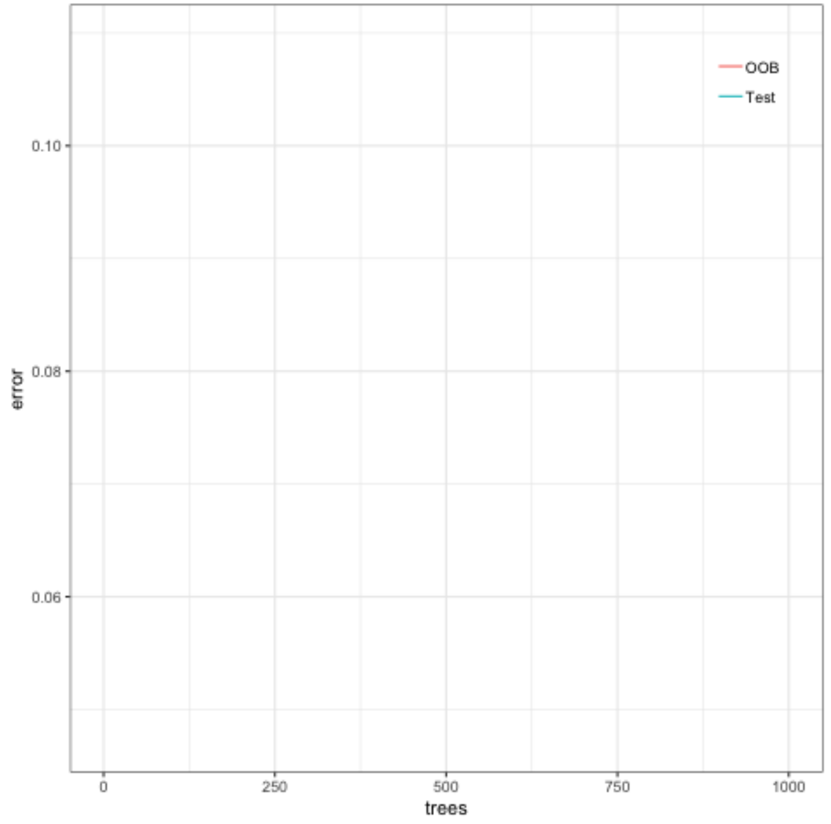
Tufts

Random Forest: Out-of-Bag (OOB) Observations

- For large enough N, on average, 63.21% or the original records end up in any bootstrap sample
- Roughly 36.79% of the observations are not used in the construction of a particular tree
- These observations are considered out-of-bag (OOB) and can be used for efficient assessment of model performance (**unstructured, but free, cross-validation**)

Pro tip:

- When N is small, OOB is less reliable than validation
- As N increases, OOB is far more efficient than *k*-fold CV
- When the number of trees are about 3x the number needed for the random forest to stabilize, the OOB error estimate is equivalent to leave-one-out cross-validation error.



Tufts

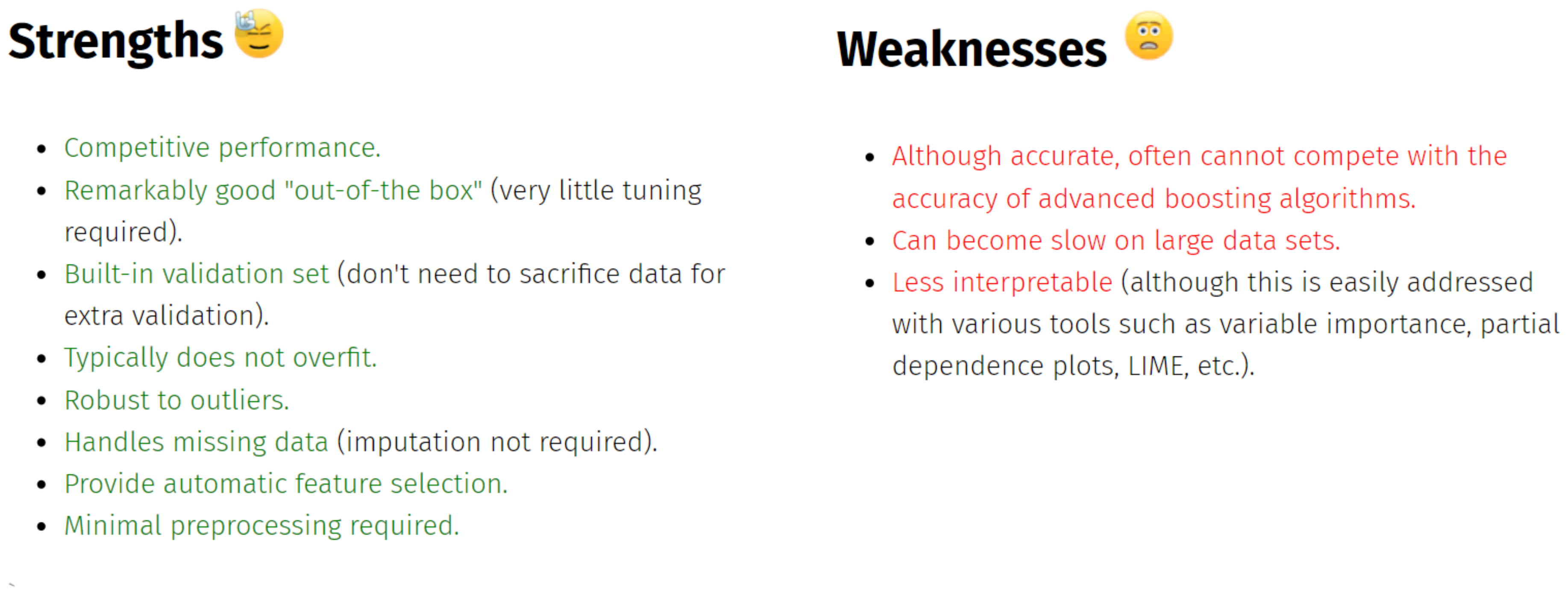
Random Forest : Tuning

Random forests provide good "out-of-the-📦" performance but there are a few parameters we can tune to increase performance.

- Number of trees
- mtry
- Node size
- Sampling scheme
- Split rule

- Typically have the largest impact on predictive accuracy.
- Tend to have marginal impact on predictive accuracy but still worth exploring. Can also increase computational efficiency.
- Generally used to increase computational efficiency

Random Forest - Strengths & Weaknesses



Strengths

- Competitive performance.
- Remarkably good "out-of-the box" (very little tuning required).
- Built-in validation set (don't need to sacrifice data for extra validation).
- Typically does not overfit.
- Robust to outliers.
- Handles missing data (imputation not required).
- Provide automatic feature selection.
- Minimal preprocessing required.

Weaknesses

- Although accurate, often cannot compete with the accuracy of advanced boosting algorithms.
- Can become slow on large data sets.
- Less interpretable (although this is easily addressed with various tools such as variable importance, partial dependence plots, LIME, etc.).