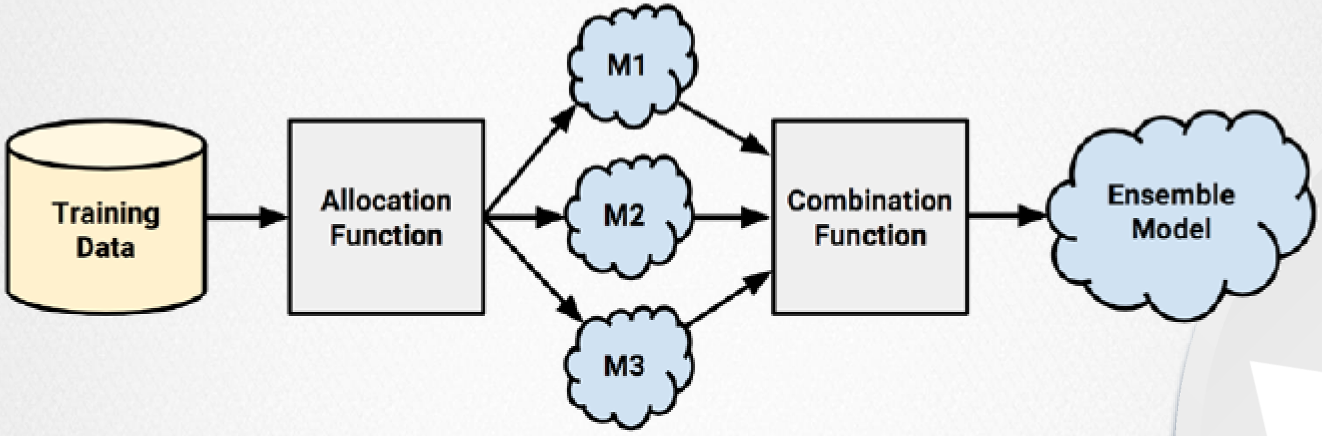
DECISION PATTERNS

Model Ensembles

Practical Machine Learning (with R)
UC Berkeley

KNOW YOUR DATA

Ensembles



Source: *Machine Learning with R*

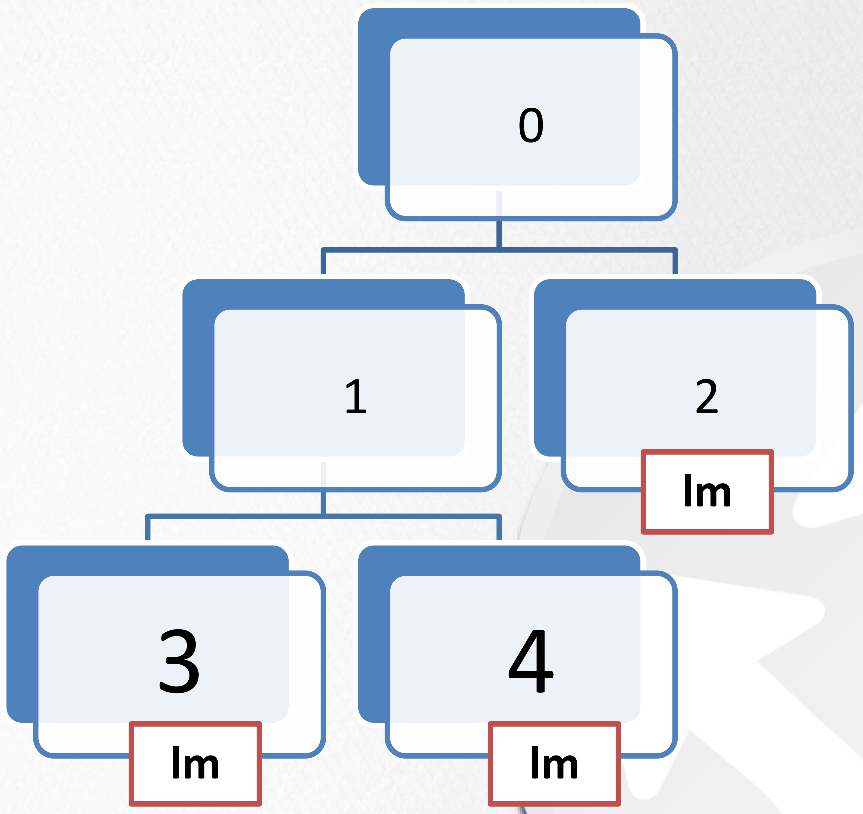
Two Big Ideas

- **Wisdom of the crowds**
  It is better to make estimates from multiple models (***ensembles***) than individual models
  - Better predictions
  - Lower variance for the same model

- **Greed is bad. Patience is good.**
  It is better to slowly approach your solution than arrive at an answer directly.
  - More accurate solutions

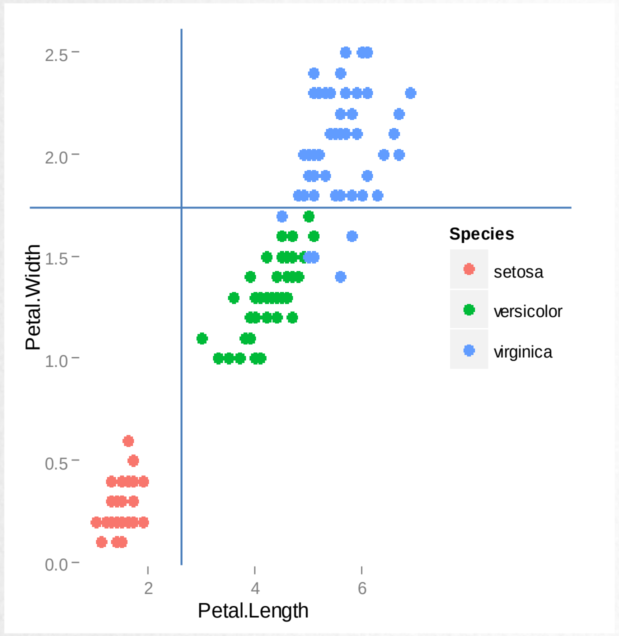
Tree Enhancement: M5

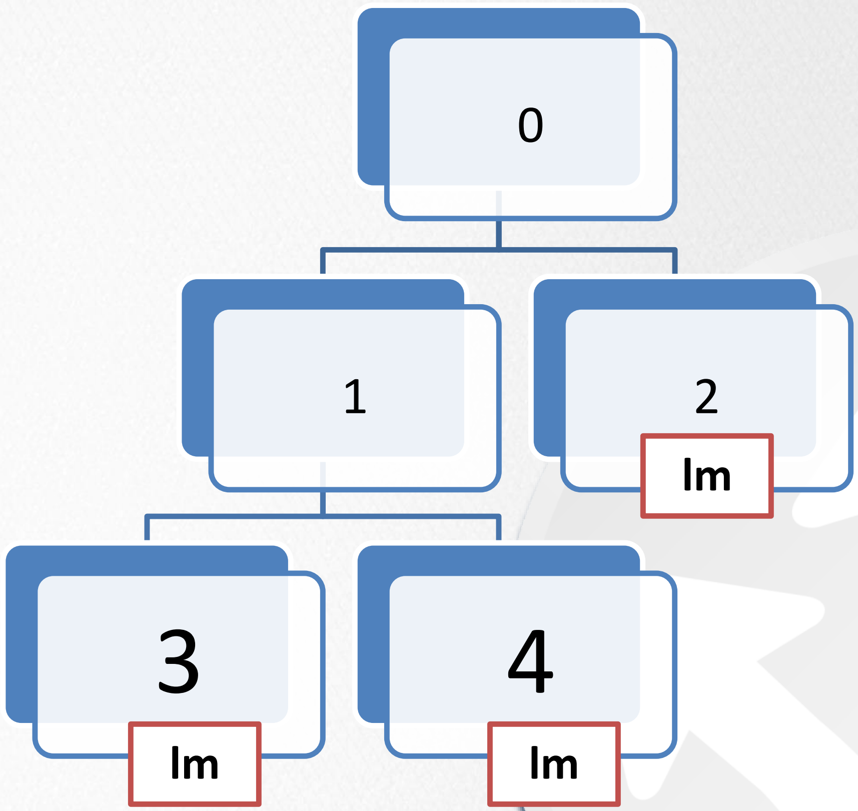
- **Wisdom of the Crowd!**
- Having one value represent the entirety of the node leaves information in the node.
- Function in the node is a simple average
- Use something better
  - **M5** put linear models in nodes of trees



M5 Tree Enhancement (cont.)

- **Greed is bad**
  - linear models are built on the residuals of the tree model.
  - Models are recursive



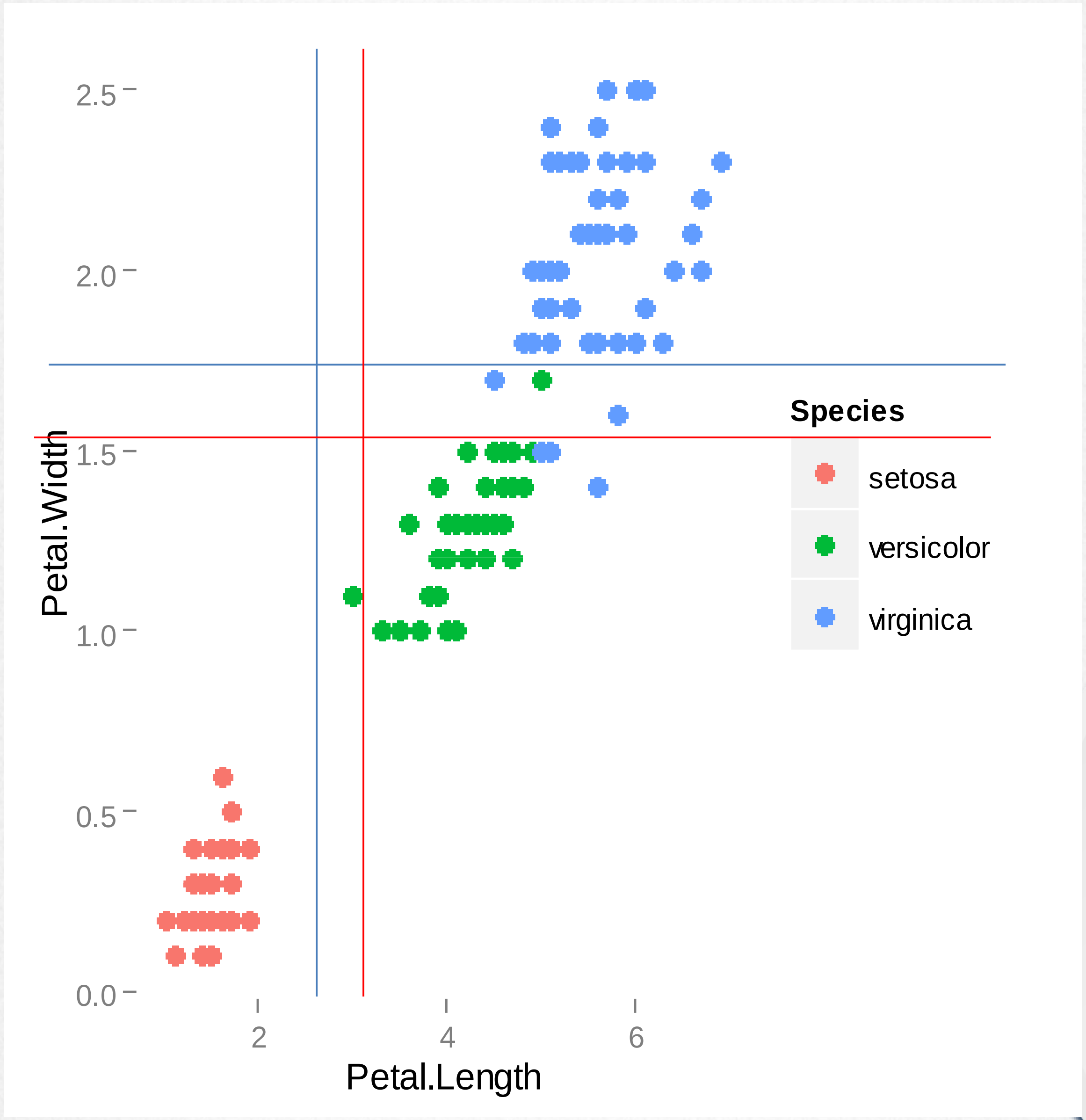


Bagging Models

- Brieman:
  "Bagging is a general approach that uses bootstrapping in conjunction with any regression (or classification) model to construct an ensemble.“

```
1 for $i = 1$ to $m$ do
2     Generate a bootstrap sample of the original data
3     Train an unpruned tree model on this sample
4 end
```

$$\hat{y} = \frac{\sum_i \hat{y}_i}{m}$$


2.5
2.0
1.5
1.0
0.5
0.0
Petal.Width
2
4
6
Petal.Length
Species
setosa
versicolor
virginica

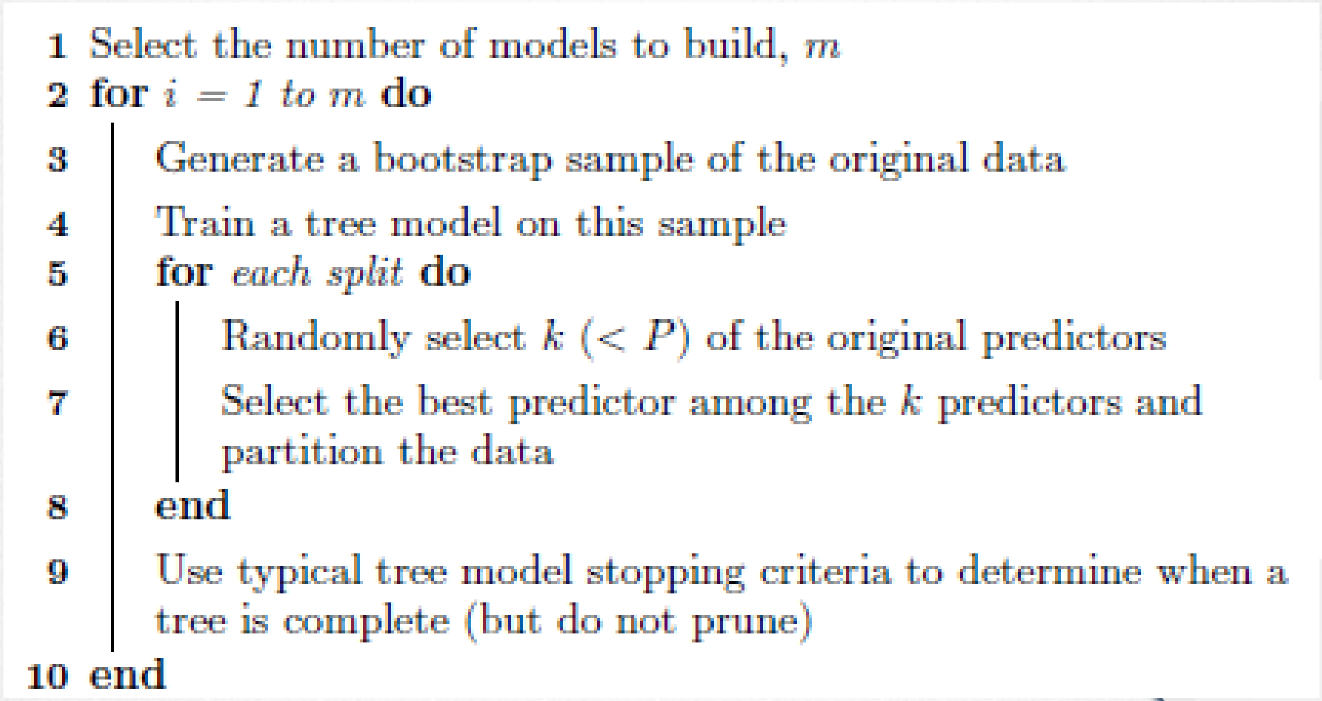
Bagging Notes

- Lowers variance
  - Increases stability
  - Has less effect on lower variance models (e.g. linear models)
  - More effect on weak learners

- Disadvantages
  - Computational cost → but parallelizable
  - Reduces Interpretability

Random Forest

- **Wisdom of the Crowds:** Bagging
- **Greed is bad: consider subset of predictors at each split**



```
Select the number of models to build, $m$
for $i = 1$ to $m$ do
    Generate a bootstrap sample of the original data
    Train a tree model on this sample
    for each split do
        Randomly select $k$ ($< P$) of the original predictors
        Select the best predictor among the $k$ predictors and
        partition the data
    end
    Use typical tree model stopping criteria to determine when a
    tree is complete (but do not prune)
end
```

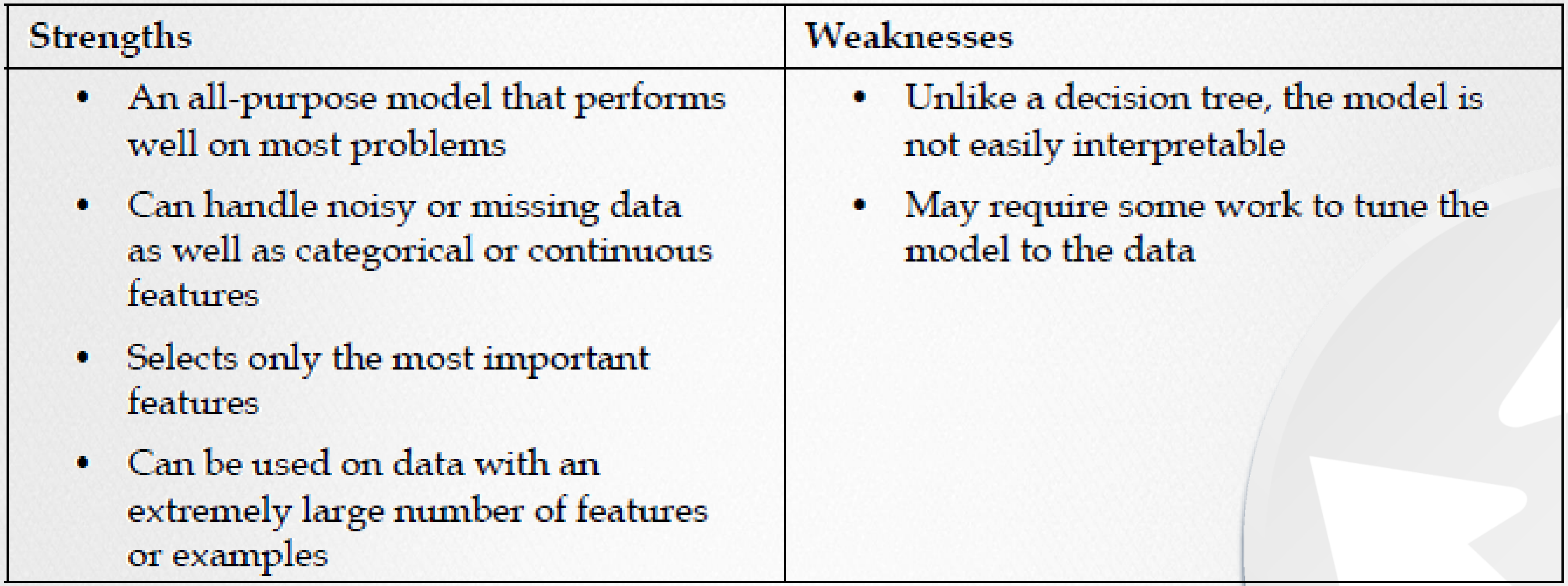
Tuning Parameter

$\mathtt{m}_{\mathtt{try}}$: number of predictors to use at each split

- **regression** 1/3rd of number predictors
- **classification** sqrt(number of predictors)

➲ Kuhn: “Starting with five values of k that are somewhat evenly spaced across the range from 2 to P”.

Random Forests



Strengths	Weaknesses
• An all-purpose model that performs well on most problems • Can handle noisy or missing data as well as categorical or continuous features • Selects only the most important features • Can be used on data with an extremely large number of features or examples	• Unlike a decision tree, the model is not easily interpretable • May require some work to tune the model to the data

Advantages

- No overfitting
- More trees better (limited by computation time/power only)
- In caret, parameters are considered independently
- Because each learner is selected independently of all previous learners, Random Forests is robust to a noisy response
- Computationally efficient -- each tree built on subset of predictors at each split.
- Use any tree variants as "base learner": CART, ctree, etc

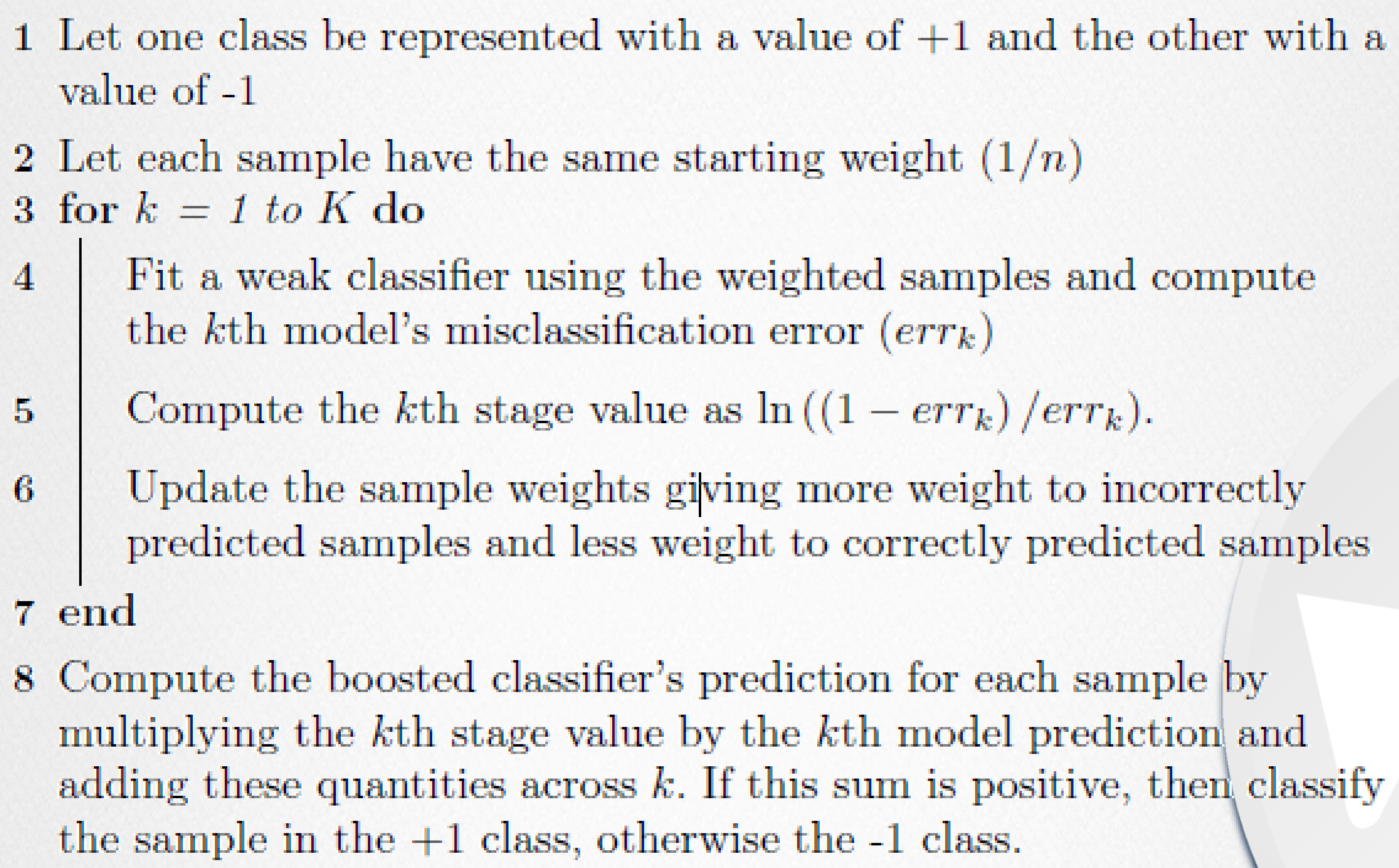
BOOSTING

BOOSTING

- Single models work;
  - Multiple models work better

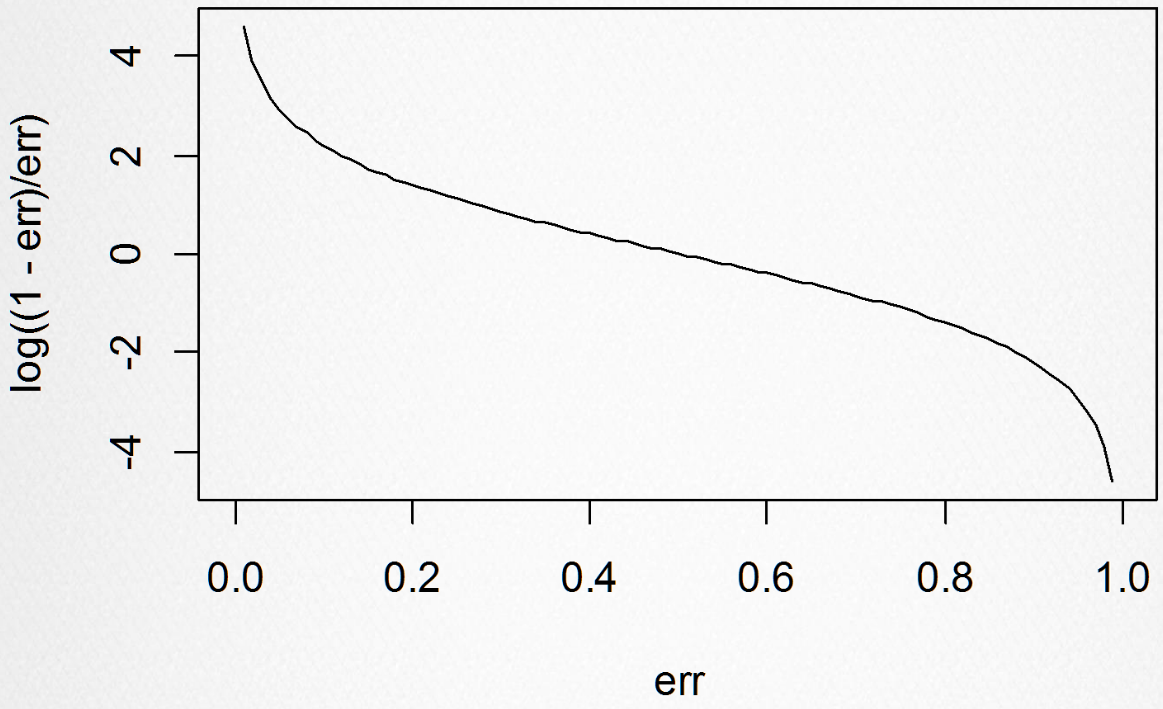
- Idea is simple:
  - **Fit first** model: $\hat{y}_1 \sim f_1(x)$
  - **Fit** errors/residuals: $\hat{y}_2 = f_2(y - \hat{y}_1)$
    $= f_2\left(y - f_1(x)\right)$
    $= f_2(x)$
  - **Iterate:** $\hat{y}_i = (y - \hat{y}_{i-1}) \sim f_i(x)$
  - **Predict:** $\hat{y} \sim \sum_i f_i(x)$

ADABOOST (SHAPIRE/FREUND)



```
1 Let one class be represented with a value of +1 and the other with a
  value of -1
2 Let each sample have the same starting weight ($1/n$)
3 for $k$ = 1 to $K$ do
4 |    Fit a weak classifier using the weighted samples and compute
  |    the $k$th model's misclassification error ($err_k$)
5 |    Compute the $k$th stage value as $\ln\left(\left(1 - err_k\right)/err_k\right)$.
6 |    Update the sample weights giving more weight to incorrectly
  |    predicted samples and less weight to correctly predicted samples
7 end
8 Compute the boosted classifier's prediction for each sample by
  multiplying the $k$th stage value by the $k$th model prediction and
  adding these quantities across $k$. If this sum is positive, then classify
  the sample in the +1 class, otherwise the -1 class.
```

Adaboost Stage Value



Boosting Notes

- Additive models
- Works best with "weak learners"
  - i.e. ungreedy, low bias, low variance
  - ~~Any~~ Most models with a tuning parameter can be a weak learner
  - Trees are excellent weak learners
    - Weak → "restricted depth"
- Residuals or errors define a gradient
- Interpreted as forward step-wise regression with exponential loss

Simple Gradient Boosting (regression)

```
1 Select tree depth, $D$, and number of iterations, $K$
2 Compute the average response, $\overline{y}$, and use this as the initial
  predicted value for each sample
3 for $k = 1$ to $K$ do
4     Compute the residual, the difference between the observed value
      and the *current* predicted value, for each sample
5     Fit a regression tree of depth, $D$, using the residuals as the
      response
6     Predict each sample using the regression tree fit in the previous
      step
7     Update the predicted value of each sample by adding the
      previous iteration's predicted value to the predicted value
      generated in the previous step
8 end
```

Naïve Guess

Simple Gradient Boosting (Classification)

```
1 Initialized all predictions to the sample log-odds: $f_i^{(0)} = \log \frac{\widehat{p}}{1-\widehat{p}}$.
2 for  iteration $j = 1 \dots M$ do
3     Compute the residual (i.e. gradient) $z_i = y_i - \widehat{p}_i$
4     Randomly sample the training data
5     Train a tree model on the random subset using the residuals as
      the outcome
6     Compute the terminal node estimates of the Pearson residuals:
      $r_i = \frac{1/n \sum_i^n (y_i - \widehat{p}_i)}{1/n \sum_i^n \widehat{p}_i (1-\widehat{p}_i)}$
7     Update the current model using $f_i = f_i + \lambda f_i^{(j)}$
8 end
```

Stochastic Gradient Boosting

- Gradient Boosting Susceptible to Overfitting
  - Apply "regularization/shrinkage"
    - Use λ ("Learning Rate")
      Rather than add the entirety of the residuals, add a fraction of the residuals at each iteration.

$$\hat{y} \sim \lambda \sum_i f_i(x) \qquad 0 < \lambda \leq 1$$

    - Small values for λ (~0.01) work best
    - λ ~ 1/computational time ~ 1/storage size

- Use bagging, as well
  - Bagging Fraction: a sample of data in each loop iteration

APPENDIX