Tree-based Models and Ensembles

Lecture 11

Supervised Learning Techniques

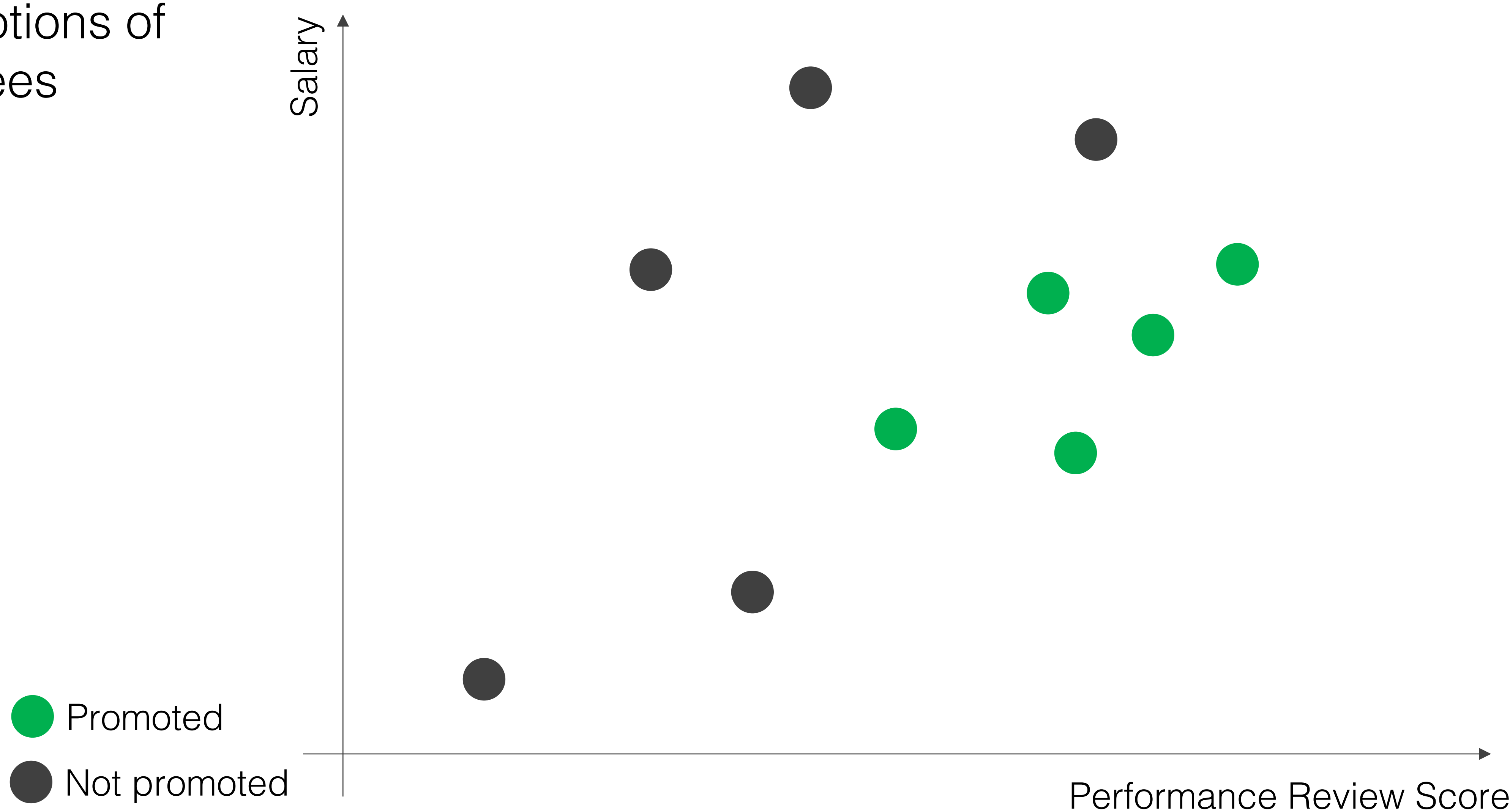
Covered so far

- Linear Regression
- K-Nearest Neighbors
- Perceptron
- Logistic Regression
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Naïve Bayes
- Decision Trees and Random Forests
- Ensemble methods (bagging, boosting, stacking)

Classification and Regression Trees (CART)

Classification trees = decision trees

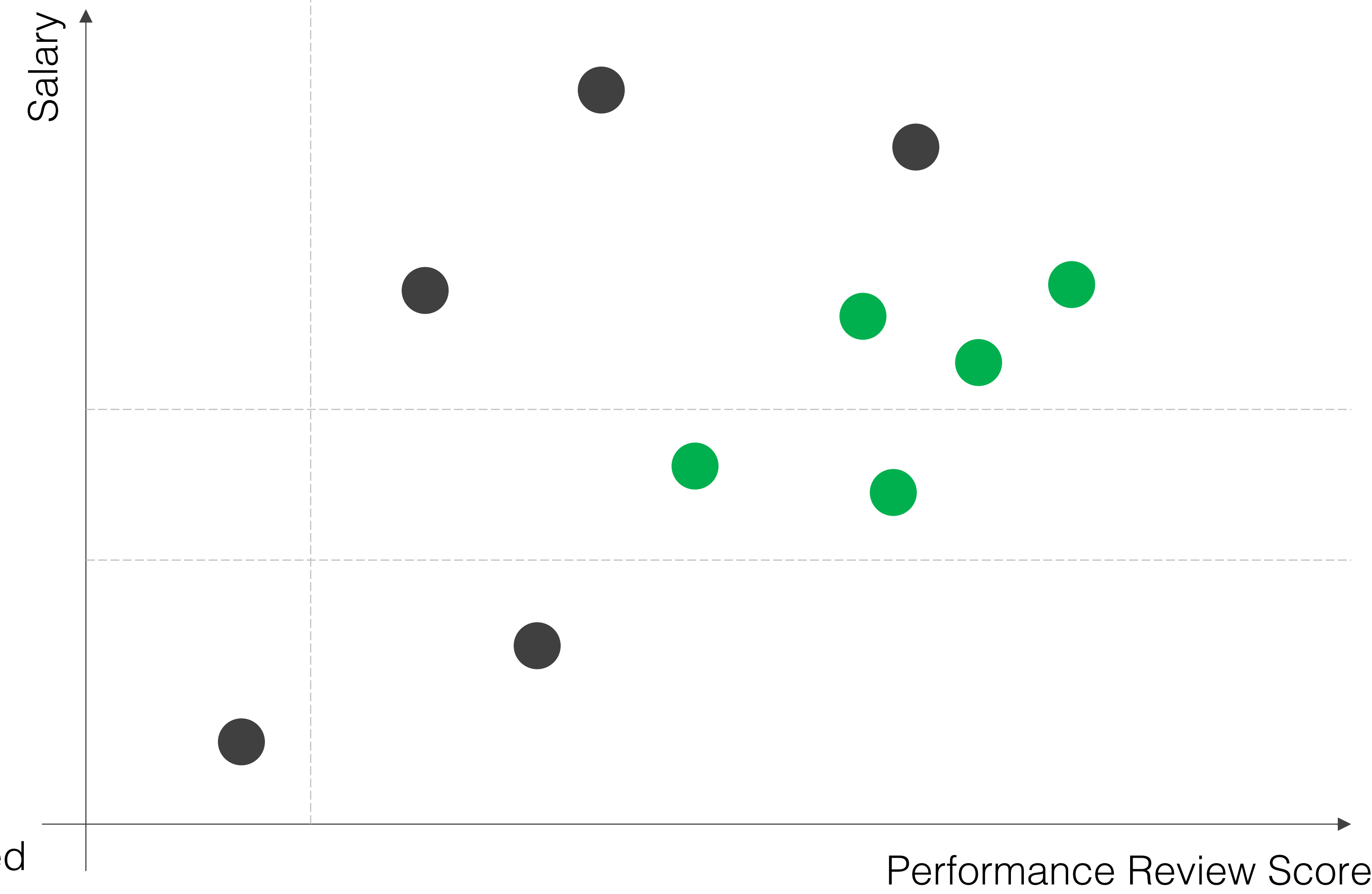
Predicting promotions of salaried employees



Classification and Regression Trees (CART)

Predicting promotions of salaried employees

1 Find the best "split" in any one feature (that best classifies the data) that divides the region in two



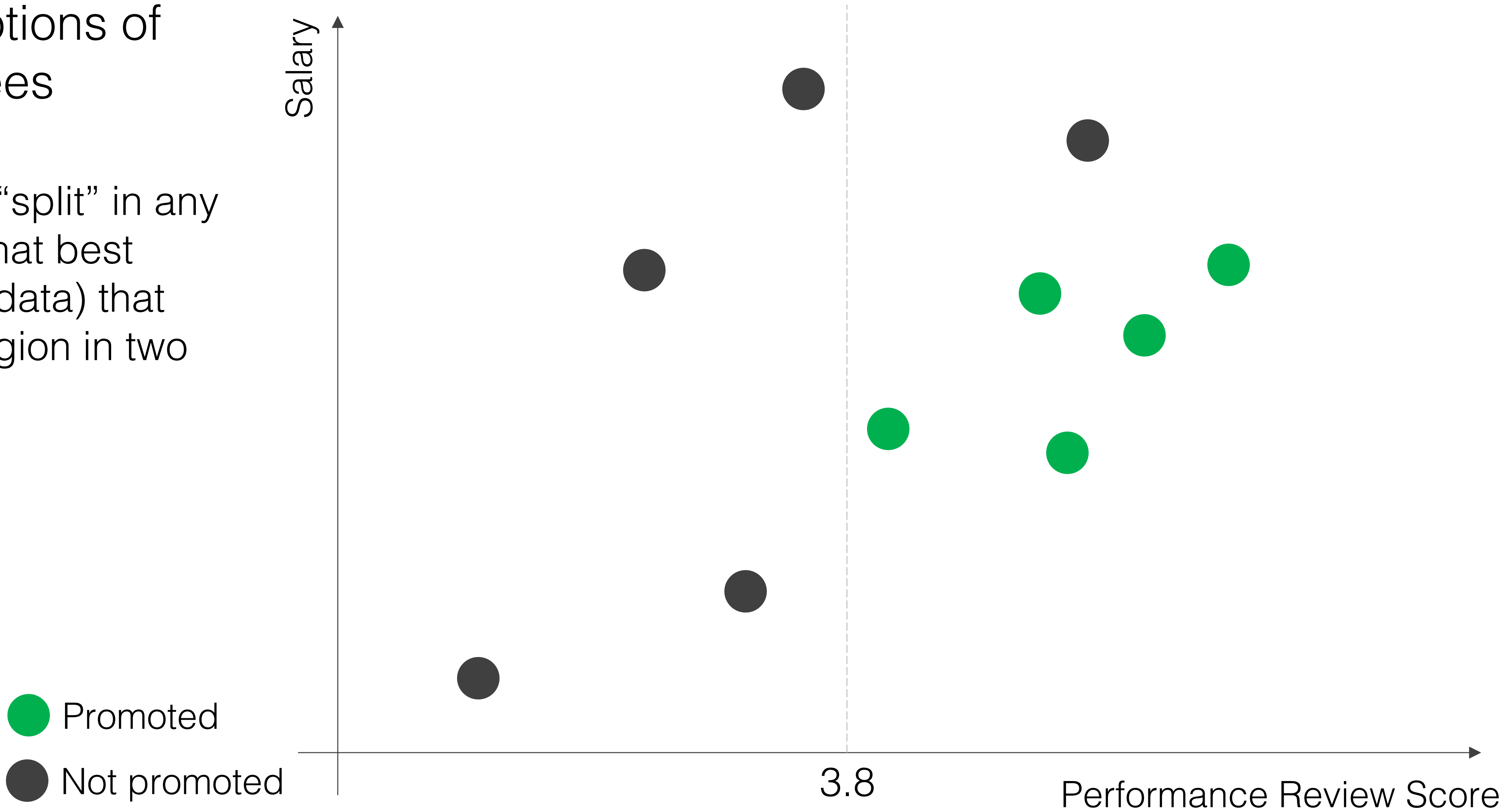
● Promoted

● Not promoted

Classification and Regression Trees (CART)

Predicting promotions of salaried employees

1 Find the best “split” in any one feature (that best classifies the data) that divides the region in two



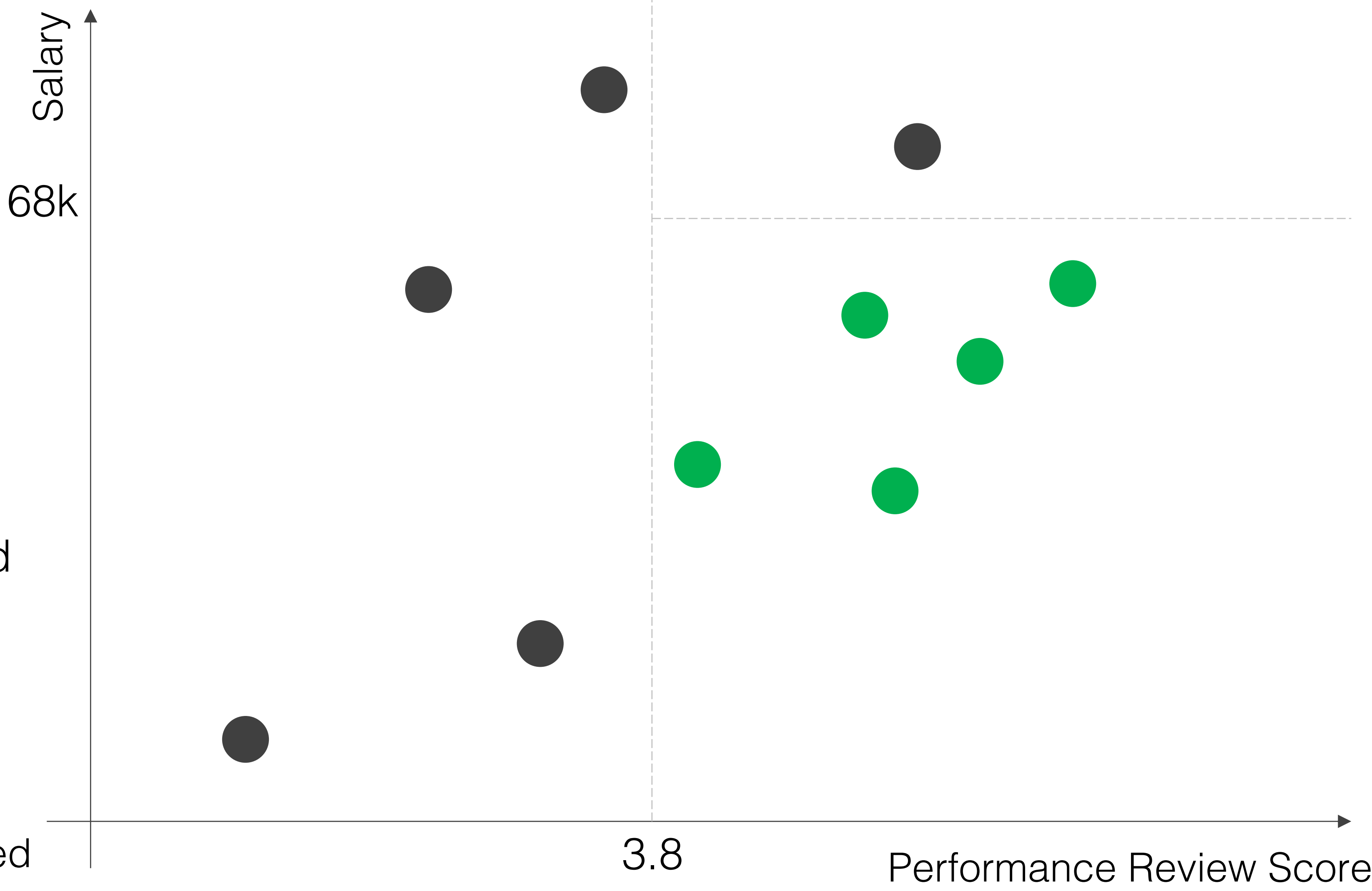
Classification and Regression Trees (CART)

Predicting promotions of salaried employees

1 Find the best "split" in any one feature (that best classifies the data) that divides the region in two

2 Continue splitting regions (1 feature at a time) until a stopping criterion is reached (e.g. there are at most N samples in any region

Greedy, recursive binary tree

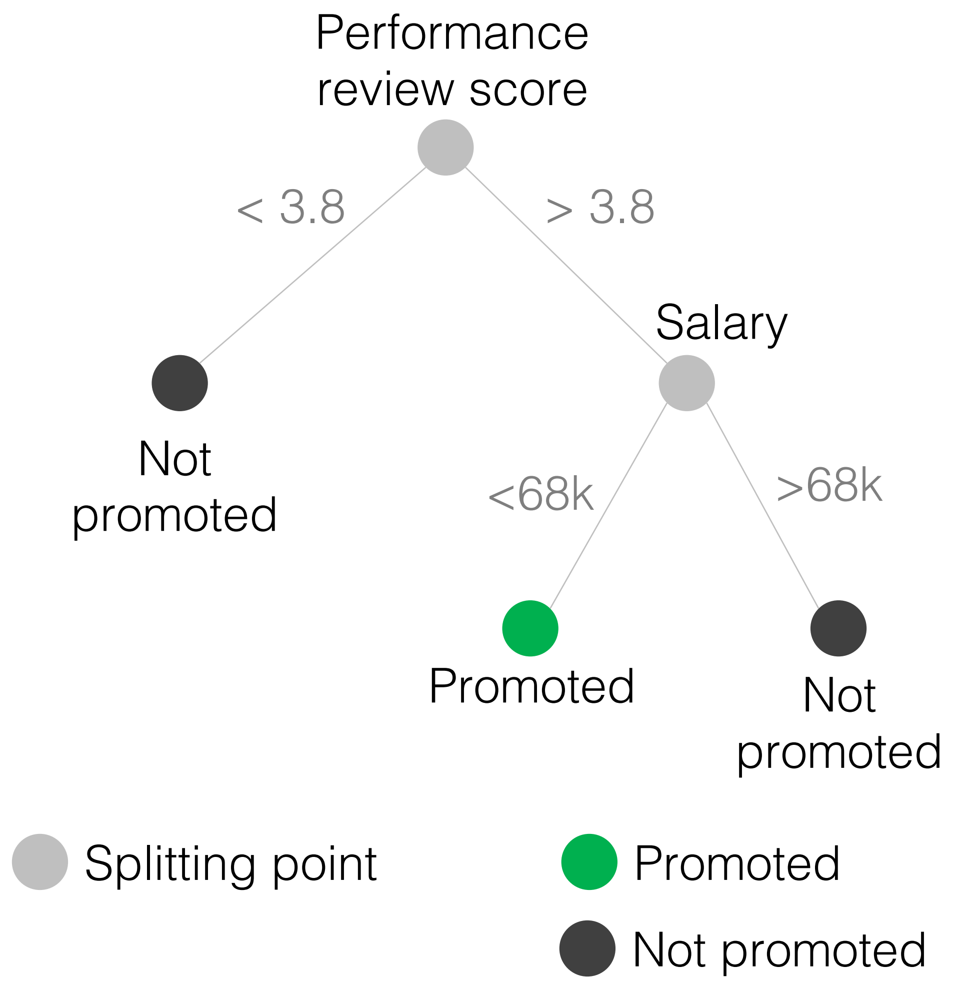


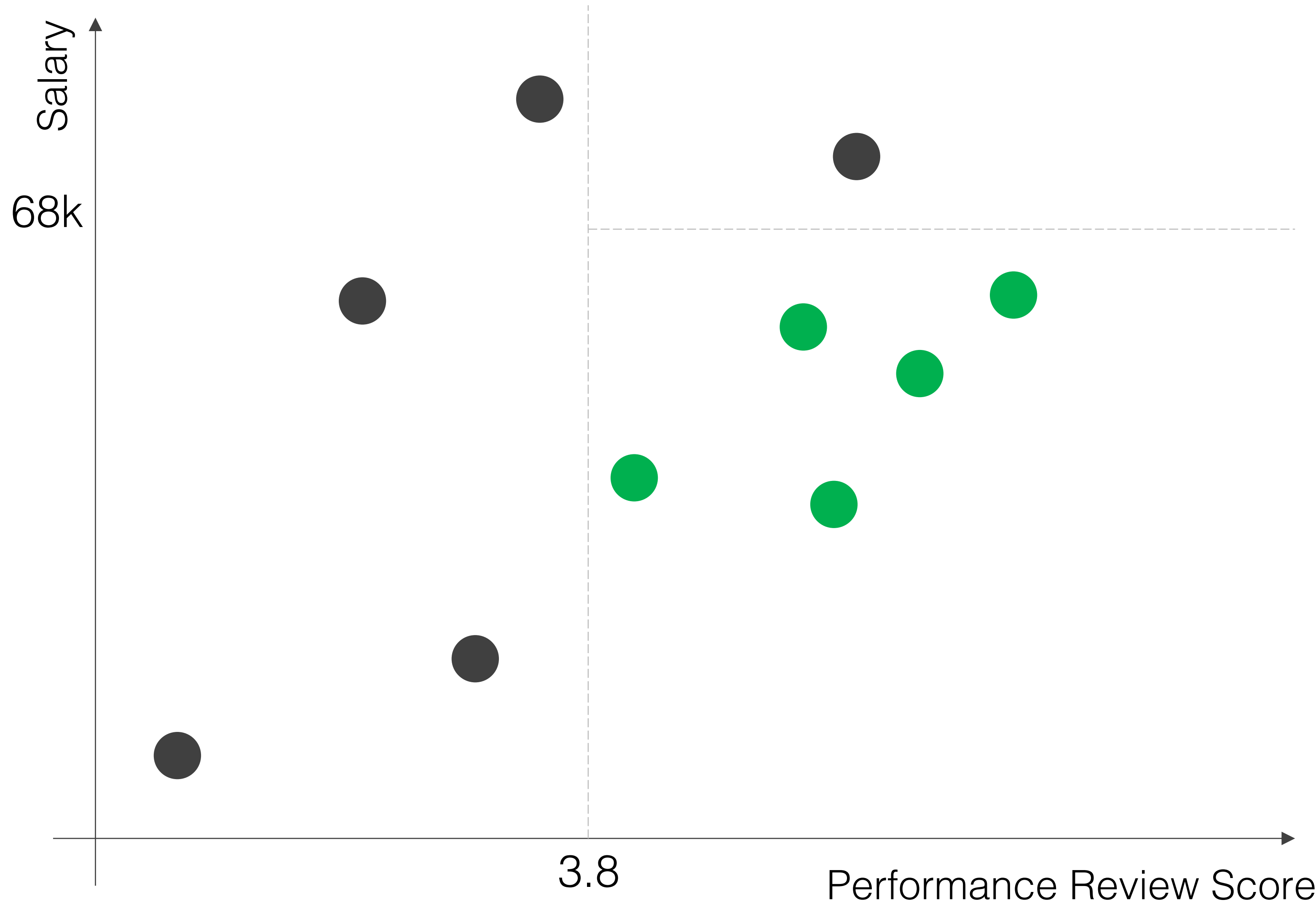
● Promoted

● Not promoted

Classification and Regression Trees (CART)

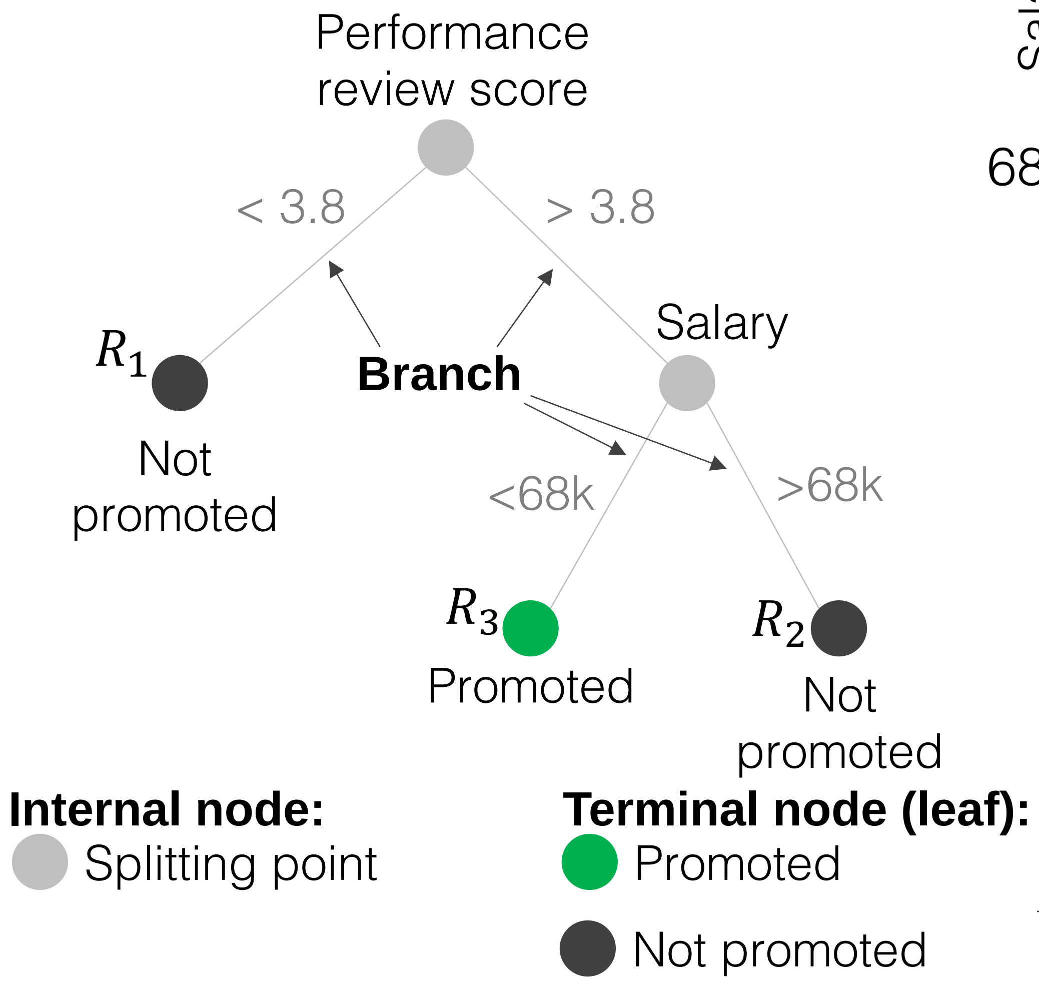
Tree representation:

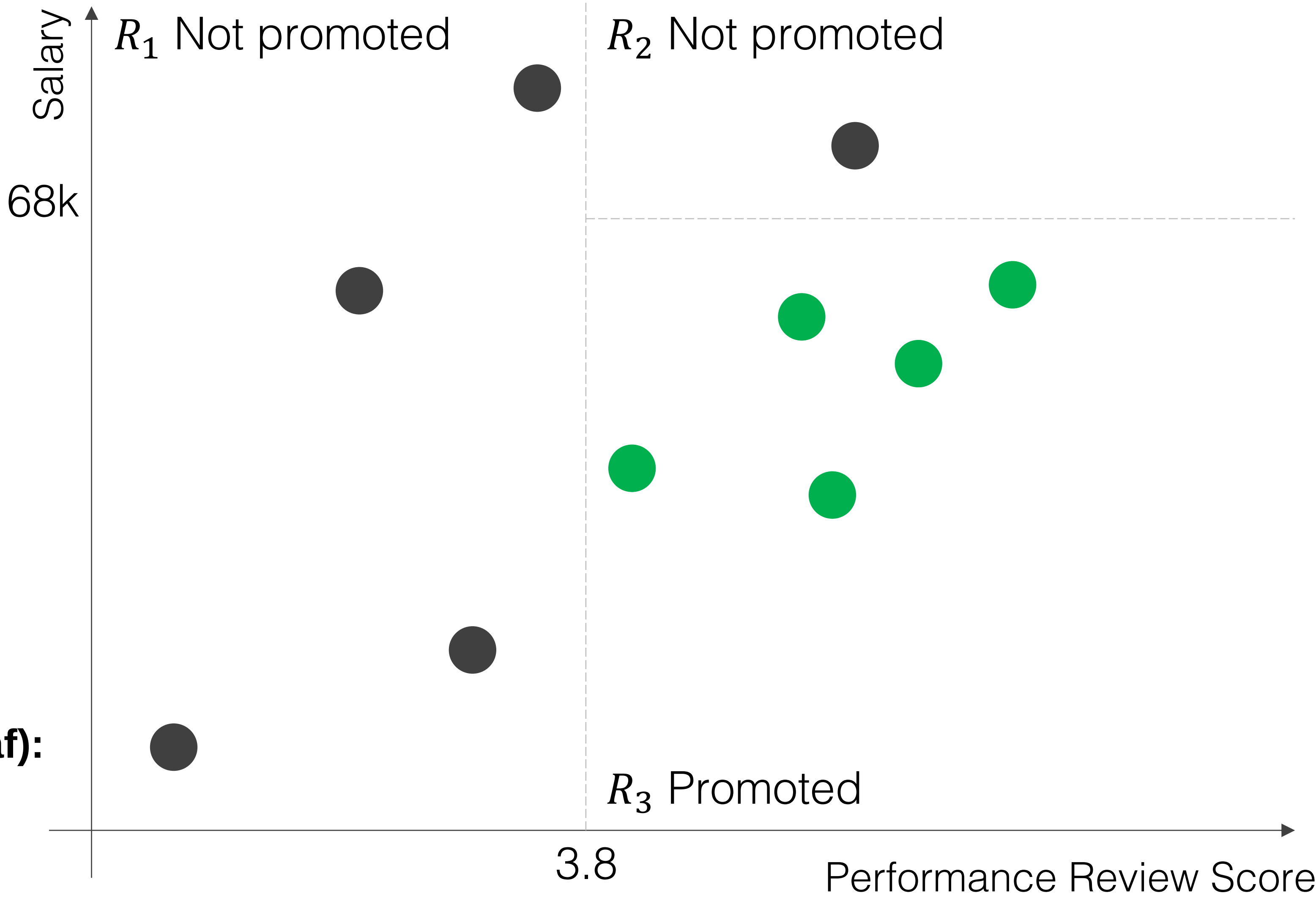




Classification and Regression Trees (CART)

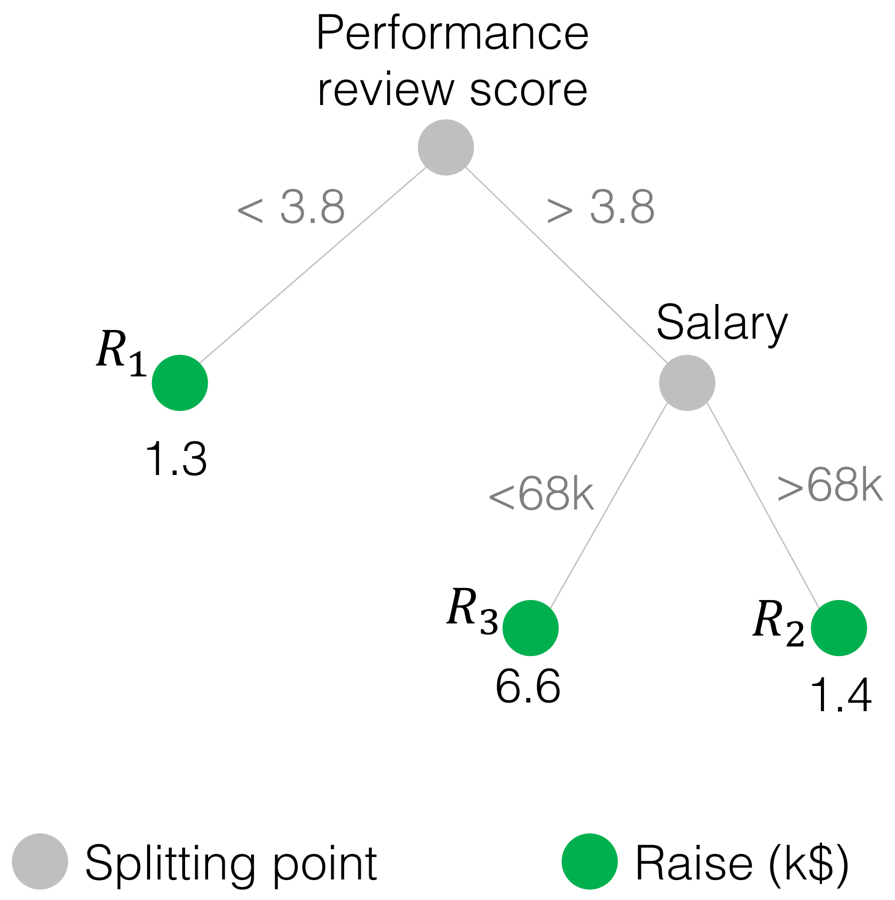
Tree representation:

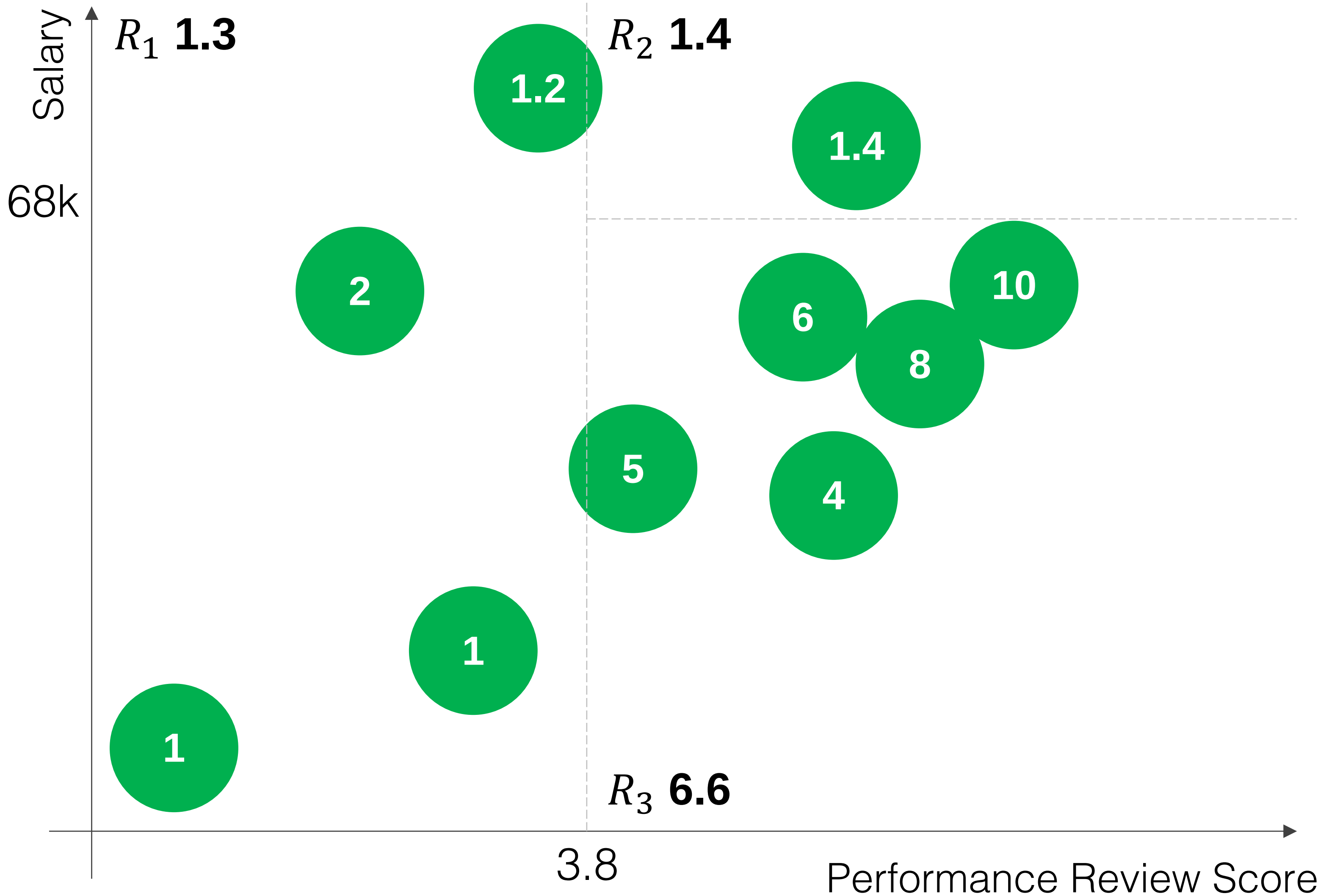




The Regression Setting

In this case, each region is represented by an average of the values it contains





How do we determine which split to make?

Pick the split that reduces the error/cost criterion most after the split

Splitting criterion

$$C = \sum_{r=1}^{R_{tot}} Q(r)$$

Regression

Mean square error

$$Q_{MSE}(r) = \sum_{i \in R_r} \left(y_i - \hat{y}_{R_r}\right)^2$$

y_i = training data response i

$\hat{y}_{R_r}$ = mean value in region r,
(where R_{tot} is the total # of regions)

Classification

Misclassification rate

$$Q_{Misclass} = 1 - \max_k \left(\hat{p}_{rk}\right)$$

Gini impurity

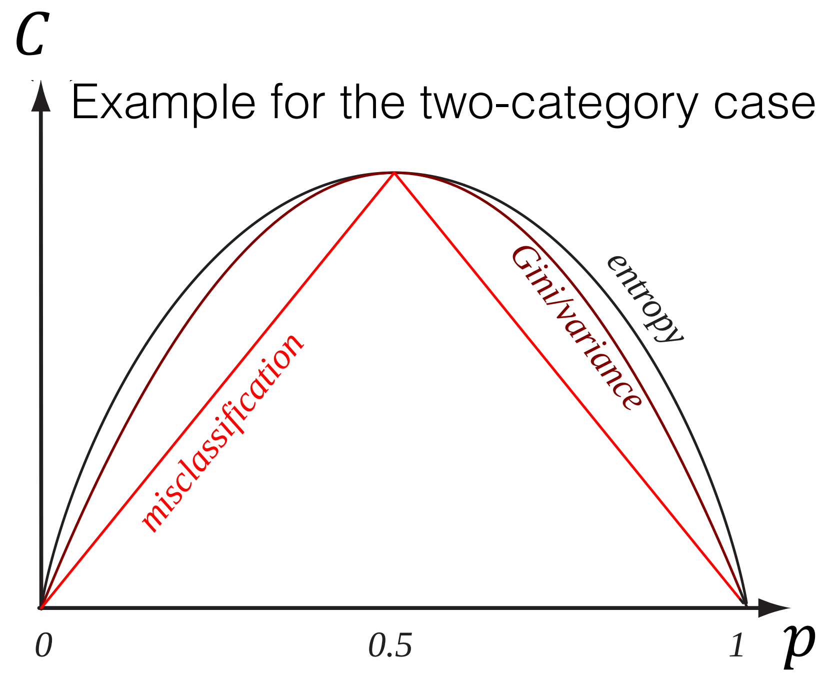
$$Q_{Gini} = \sum_{k=1}^{K} \hat{p}_{rk}(1 - \hat{p}_{rk})$$

Cross-entropy

$$Q_{entropy} = -\sum_{k=1}^{K} \hat{p}_{rk} \log \hat{p}_{rk}$$

$\hat{p}_{rk}$ = proportion of training observations in the r^{th} region from the k^{th} class

Example for the two-category case

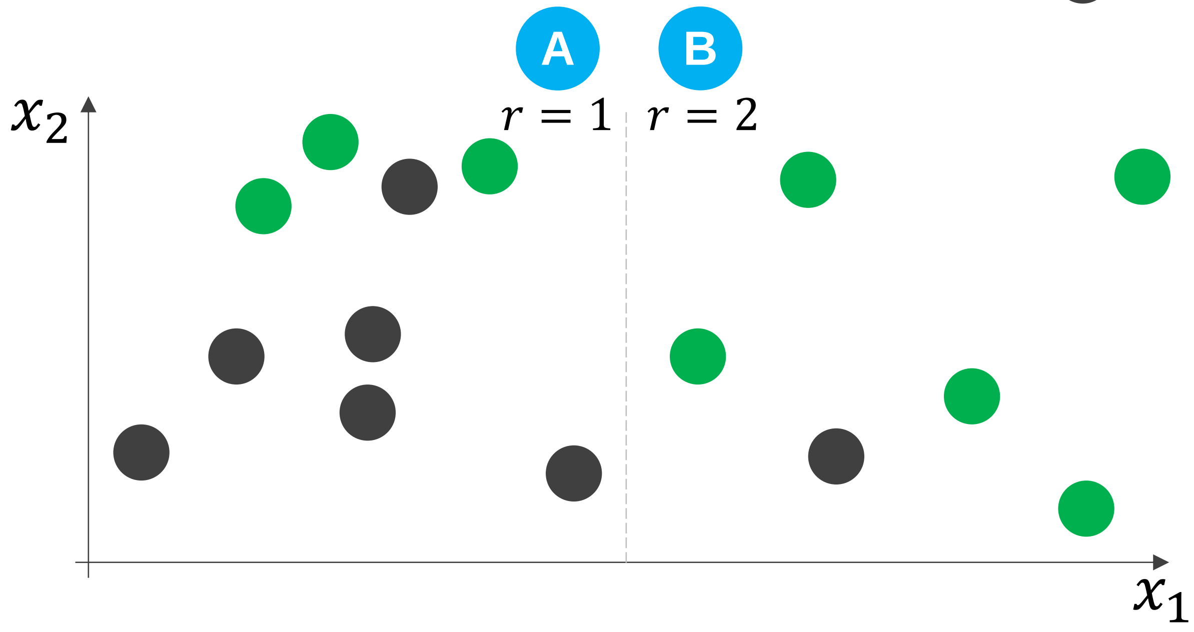


Duda, Hart, and Stork., Pattern Classification

How to measure quality of split for classification?

$\hat{p}_{rk}$ = proportion of training observations in the rth region from the kth class

Class 1 ●
Class 2 ●



$\hat{p}_{11} = 3/9$ $\hat{p}_{21} = 5/6$

$\hat{p}_{12} = 6/9$ $\hat{p}_{22} = 1/6$

For each region:

	A	B
Misclassification rate $Q_{Misclass} = 1 - \max_k (\hat{p}_{rk})$	0.333	0.167
Gini impurity $Q_{Gini} = \sum_{k=1}^{K} \hat{p}_{rk}(1-\hat{p}_{rk})$	0.444	0.278
Cross-entropy $Q_{entropy} = -\sum_{k=1}^{K} \hat{p}_{rk} \log \hat{p}_{rk}$	0.637	0.450

Tree Pruning

Trees have the tendency to overfit the data

- Consider the stopping rule: stop splitting once there is only 1 observation in each region (leads to complete overfit)

Pruning the tree back reduces this overfit (removing splits after the tree is formed)

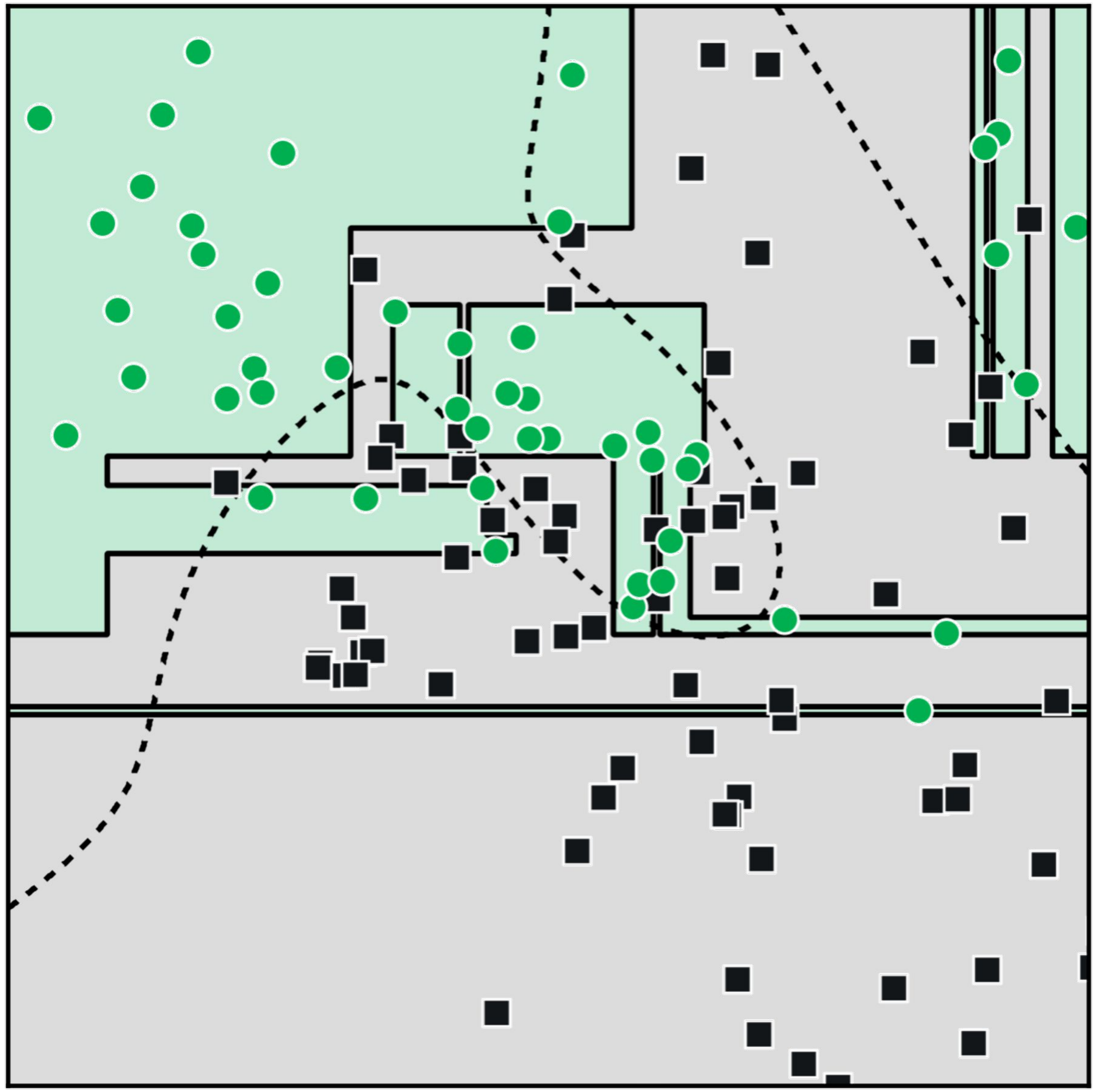
Pruning can be optimized through a penalty on the number of terminal nodes:

$$C_{Prune} = \sum_{j=1}^{T} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2 + \alpha T$$

penalty on number of terminal nodes

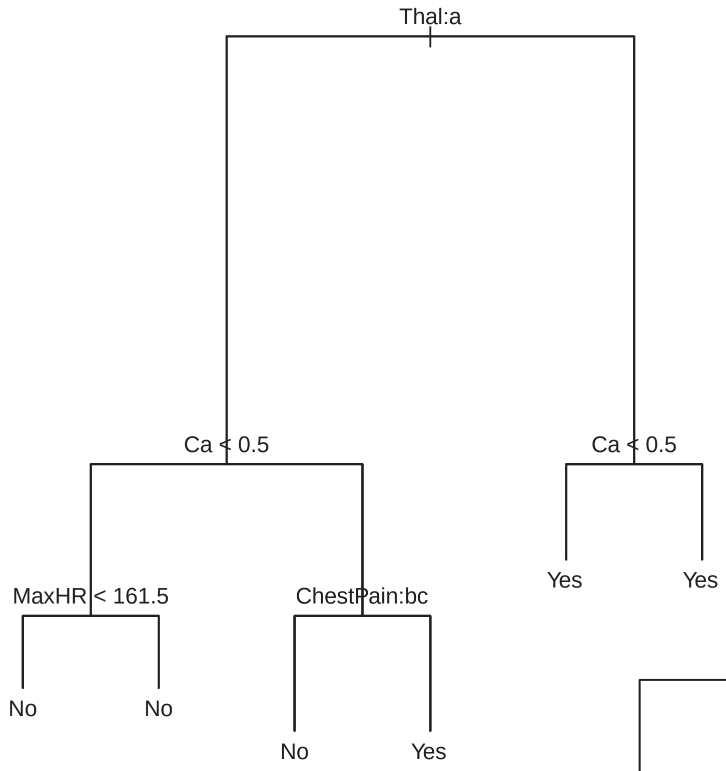
number of terminal nodes

Decision Tree



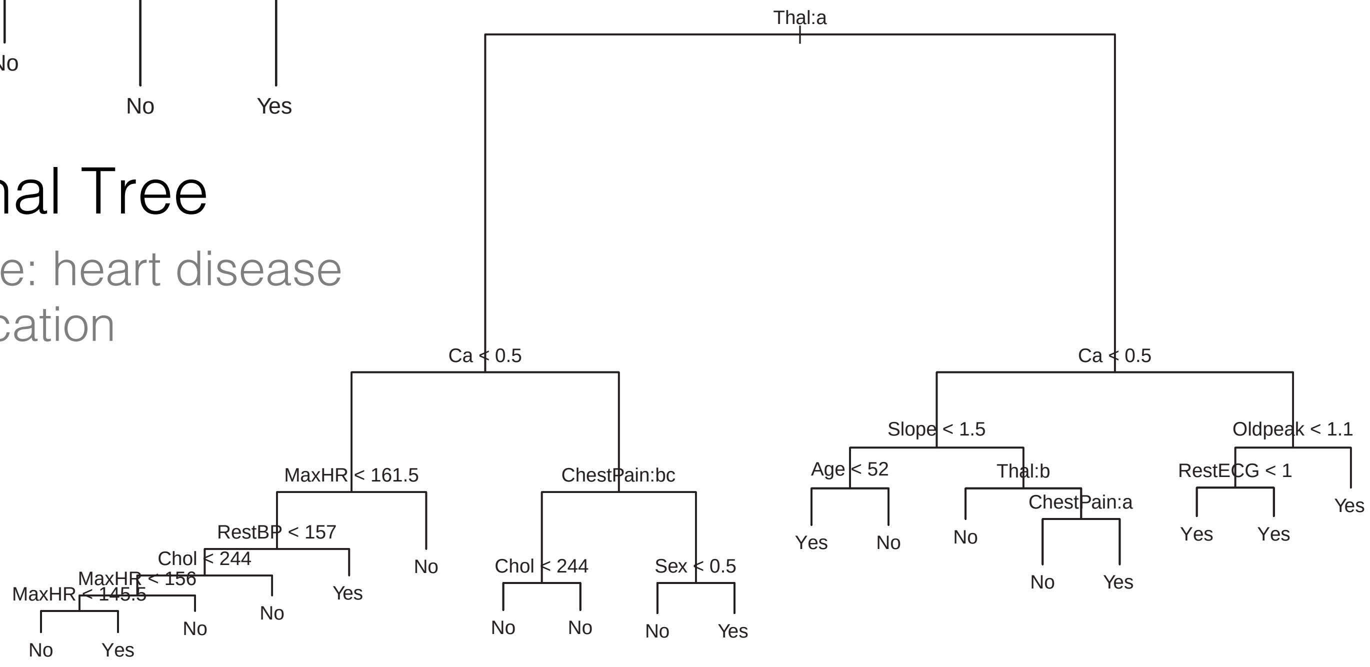
Pruning example

Pruned Tree

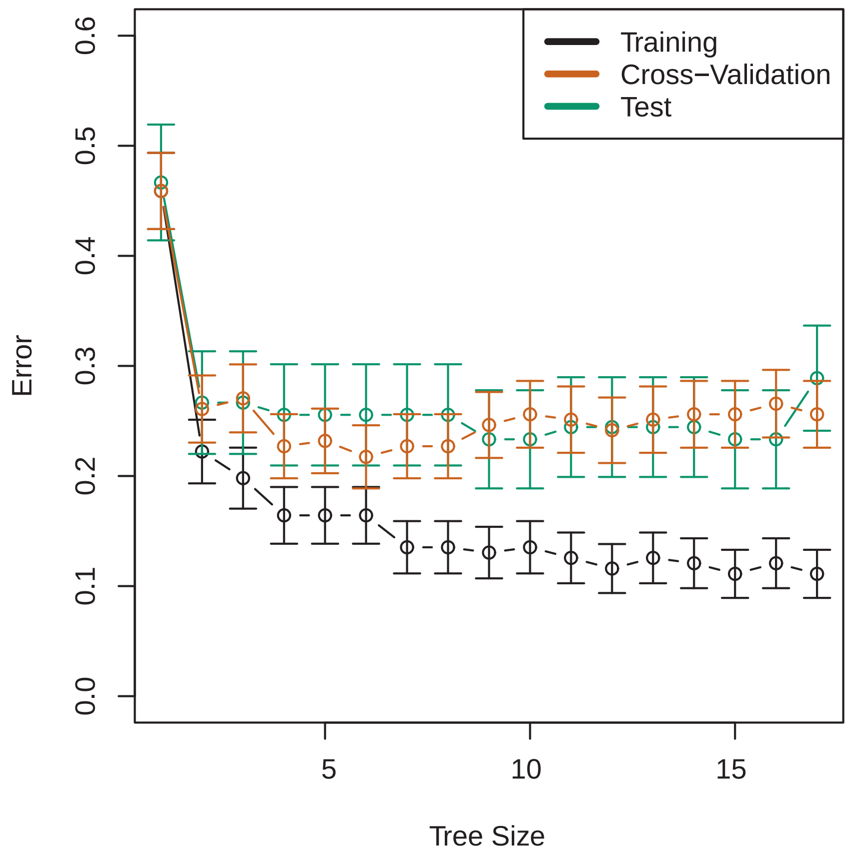


Original Tree

Example: heart disease classification

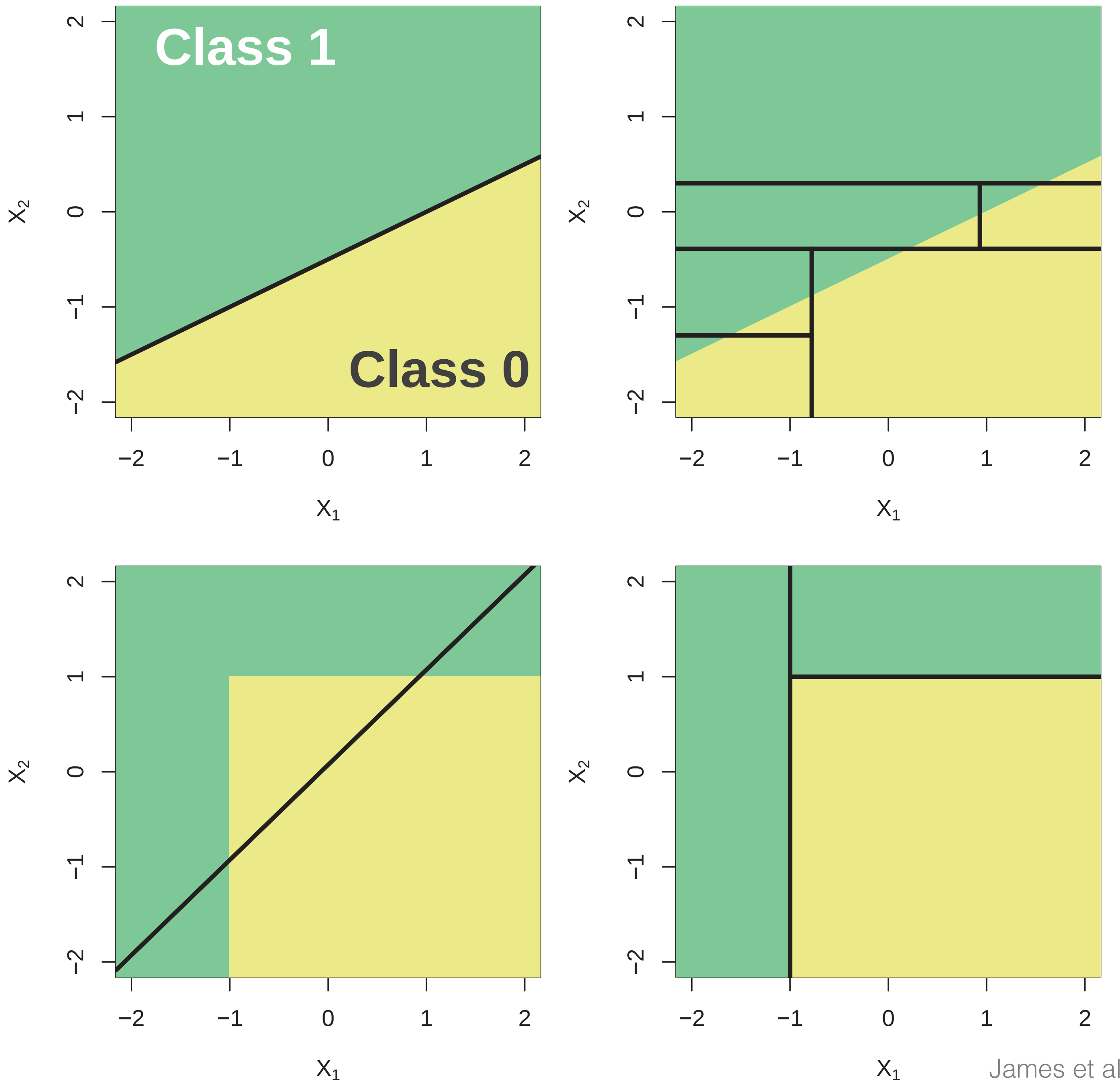


Performance



James et al., An Introduction to Statistical Learning

Linear model



Classification Tree

Struggle when the boundary is not parallel to an axis

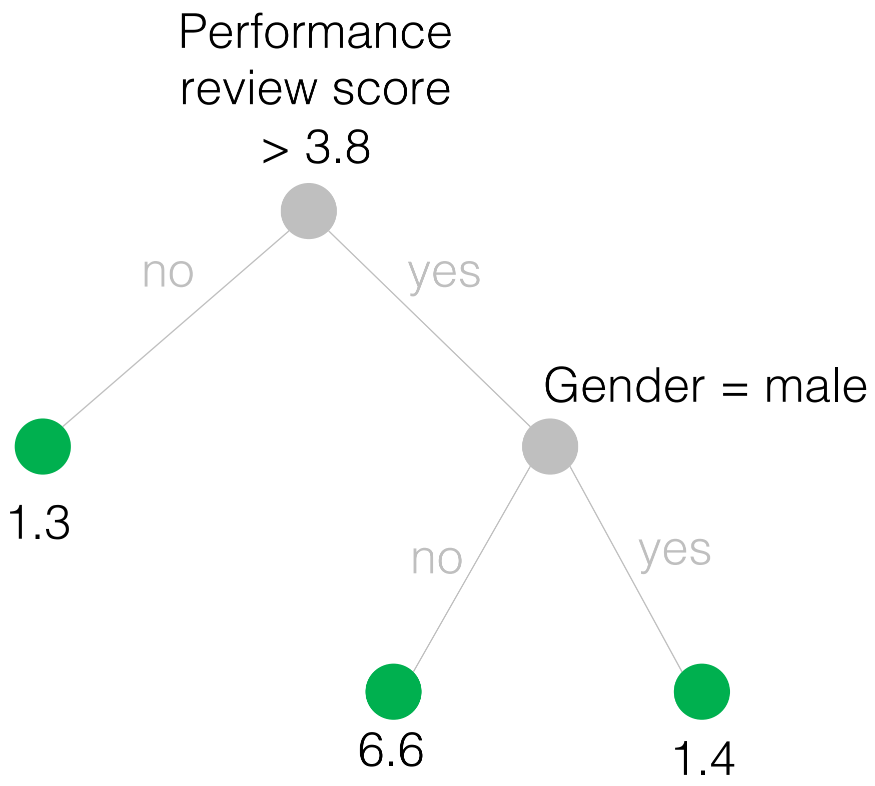
…nonlinear feature transforms could help…

James et al., An Introduction to Statistical Learning

Pros/Cons

Numerical data

Categorical data



Pros:

Trees easily handle multiple types of data

Trees are easy to interpret

Cons:

Trees do not typically have the same level of predictive accuracy of many methods

Tend to overfit (have high variance)

Ensemble learning

How can we combine models to improve performance?

Bagging (bootstrap aggregation)
Random forests (tree-specific modification of bagging)
Gradient boosting

Applicable to many models (not just trees)

Reducing Variance or Bias through ensembles

	Bagging	**Boosting**
Models in ensemble:	high variance, low bias (i.e. overfit models)	high bias, low variance (i.e. underfit models)
Effect of aggregating:	Reduce variance through averaging output	Reduce bias through sequentially fitting models to previous model errors

Bagging

Bootstrap aggregation

Trees **overfit** (have high variance). Averaging over observations **reduces variance**

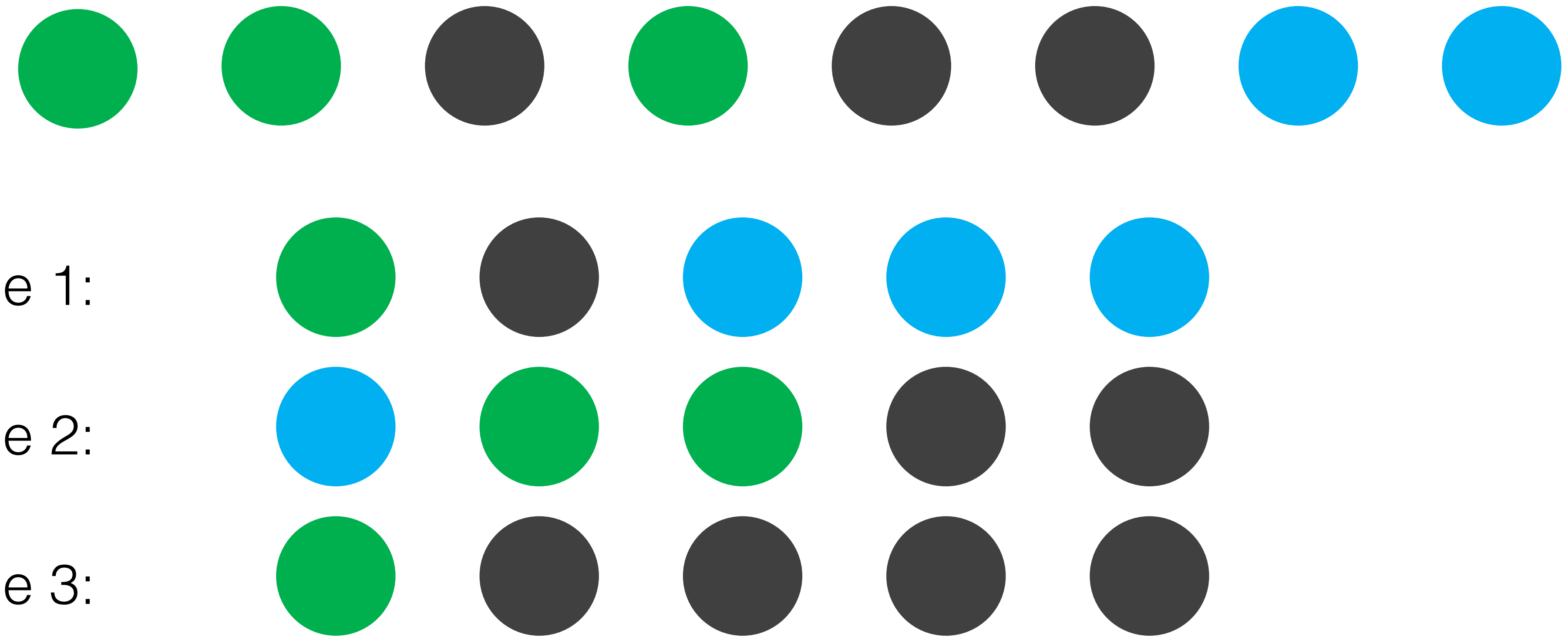
Recall bootstrap sampling (sampling with replacement):

Original Data:

Bootstrapped sample 1:

Bootstrapped sample 2:

Bootstrapped sample 3:



Bagging

Bootstrap aggregation

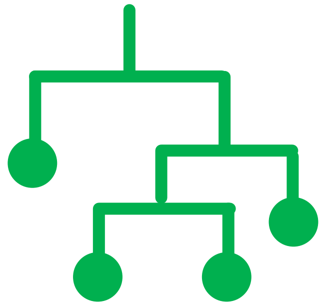
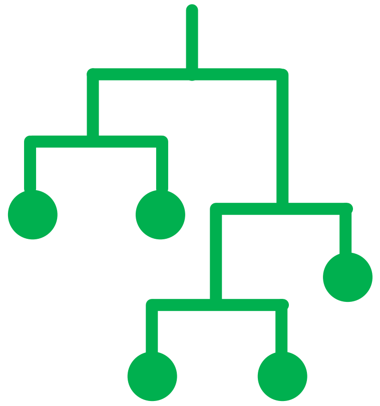
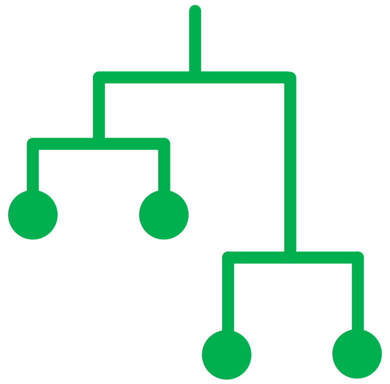
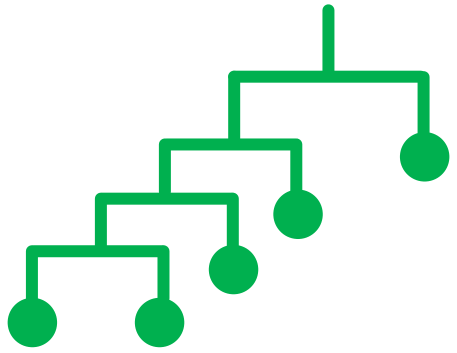
Can be applied to many machine learning techniques!

1. Create a random bootstrap sample from the training data
2. Train a model on that bootstrap sample and call it $\hat{f}_i(\boldsymbol{x})$
3. Repeat 1 and 2 until we have B models trained on different bootstrap samples
4. Take the average of the output for our new model estimate:

$$\hat{f}_{bag}(\boldsymbol{x}) = \frac{1}{B}\sum_{i=1}^{B}\hat{f}_i(\boldsymbol{x})$$

(for classification models we can get the average class confidence or take a majority vote)

Bagging

Tree Number:	1	2	3	4
Observations Included: (out of 1-9)	[1,2,3,3,8]	[1,2,4,7,7]	[1,5,6,8,9]	[2,2,2,4,9]
Features list:	$[A, B, C, D]$	$[A, B, C, D]$	$[A, B, C, D]$	$[A, B, C, D]$
First split:	D	B	B	D
Trees:				

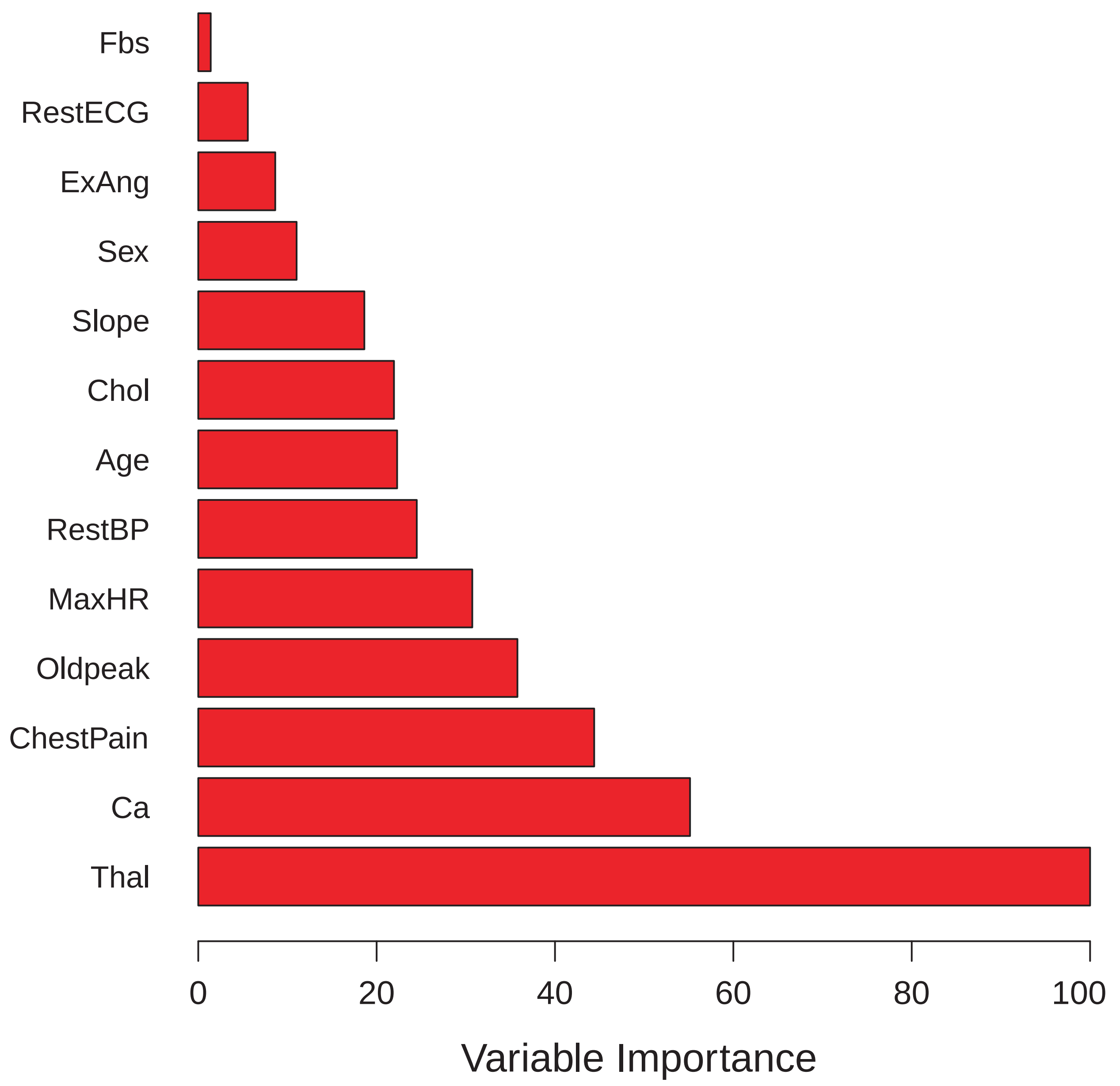
Variable Importance

Decision trees are very interpretable, but this is lost with bagging

We can construct another measure called "variable importance" to **compare feature contributions**

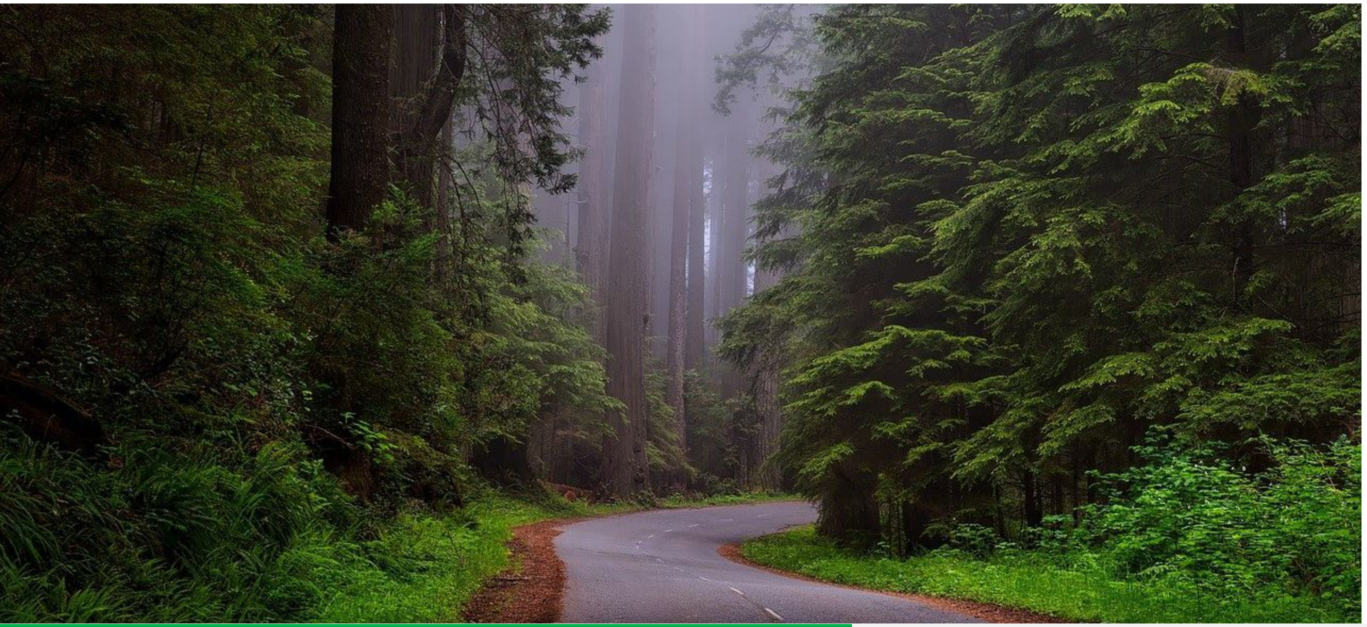
1. Calculate the total amount the error (or impurity) decreased by splitting on each feature.

2. Average over all the trees resulting from bagging



James et al., An Introduction to Statistical Learning

Random Forests



Random Forests

Random forests **decorrelate** **the bagged trees**

A **small tweak on bagging**

Decision trees are constructed greedily

This can lead to highly correlated trees

“Strong” features will typically be split before moderately strong predictors.

Each time a split is considered, a **random subset of m features** is selected as candidates from the full set of p features

Typically chose: $m = \sqrt{p}$

(If $m = p$, then we would be back to the bagging approach)

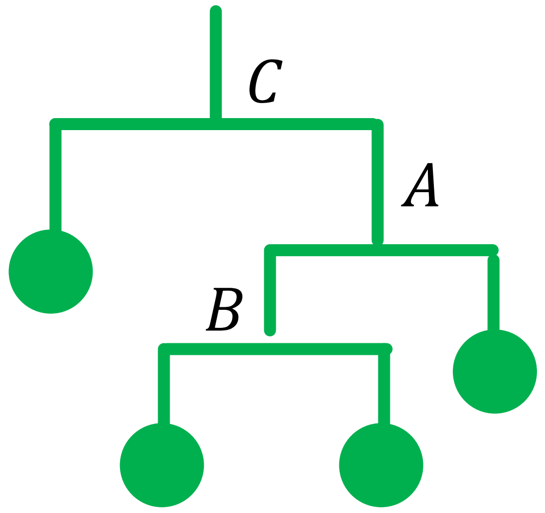
	Bagging	Random forests
Observations Included: (out of 1-9)	[1,2,3,3,8]	[1,2,3,3,8]
Features list:	$[A, B, C, D]$	$[A, B, C, D]$

Feature options for each split:

$[A, B, C, D]$

$[A, B, C, D]$

$[A, B, C, D]$

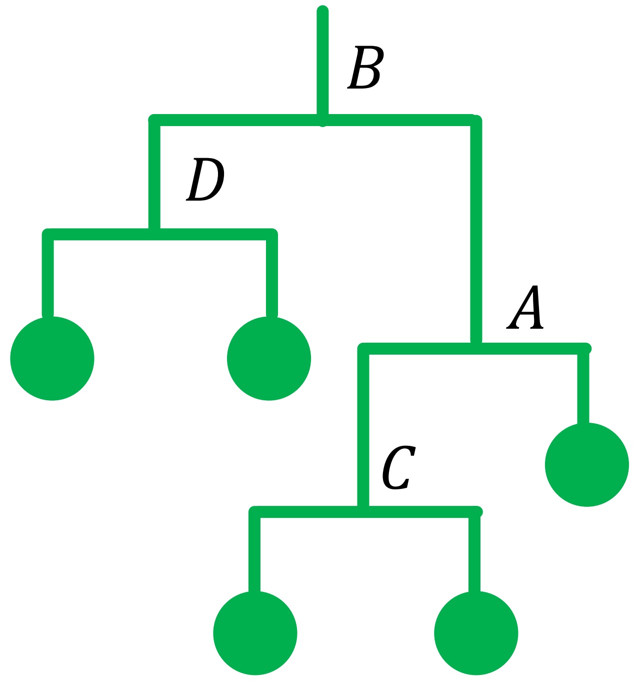


$[A, B]$

$[C, D]$

$[A, B]$

$[B, C]$

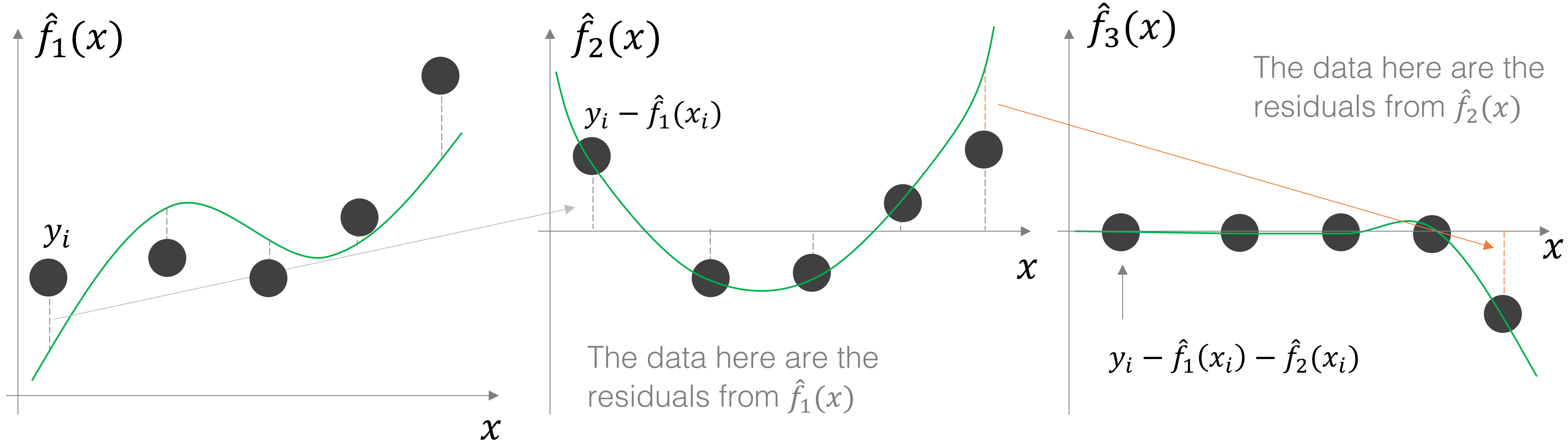


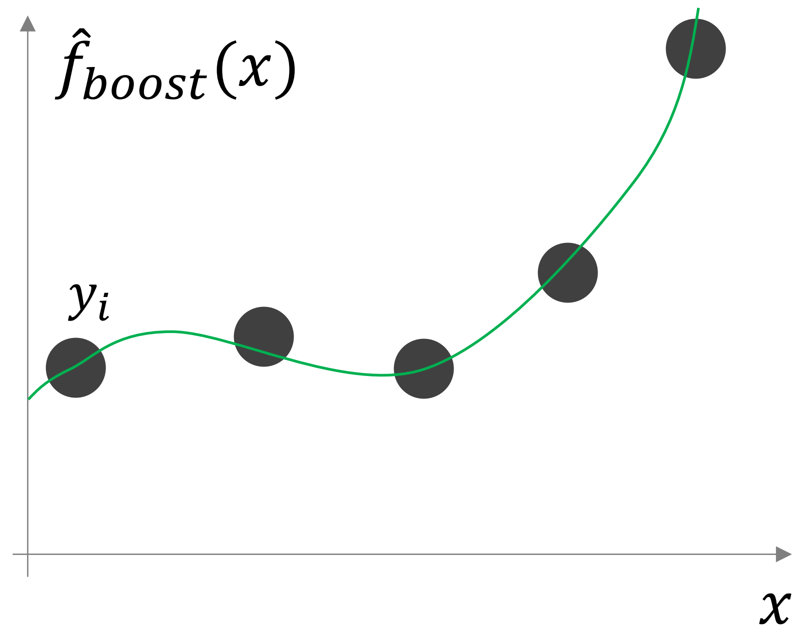
Boosting

Can be applied to many machine learning techniques!

Bagging created trees that were designed to be as independent as possible

Boosting involves building trees **sequentially**, each building on the errors of the last


$\hat{f}_1(x)$
y_i
x
$\hat{f}_2(x)$
$y_i - \hat{f}_1(x_i)$
x
The data here are the residuals from $\hat{f}_1(x)$
$\hat{f}_3(x)$
The data here are the residuals from $\hat{f}_2(x)$
x
$y_i - \hat{f}_1(x_i) - \hat{f}_2(x_i)$



We build consecutive models, each fit to the residuals of the last model

We sum models output to get the boosted prediction

$$\hat{f}_{boost}(x) = \hat{f}_1(x) + \hat{f}_2(x) + \hat{f}_3(x)$$

Boosting

Boosting for regression trees

1. Select the number of models to train, B, and learning rate λ

   λ slows down the learning process to avoid overfitting

2. Set $\hat{f}(\boldsymbol{x}) = 0$ and $r_i = y_i$ for all the training data

Repeat B times:

3. Fit a tree, $\hat{f}_i(\boldsymbol{x})$ to the residuals, r_i (with d splits)

   Often this is just a small number of splits

4. Update $\hat{f}(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) + \lambda \hat{f}_i(\boldsymbol{x})$

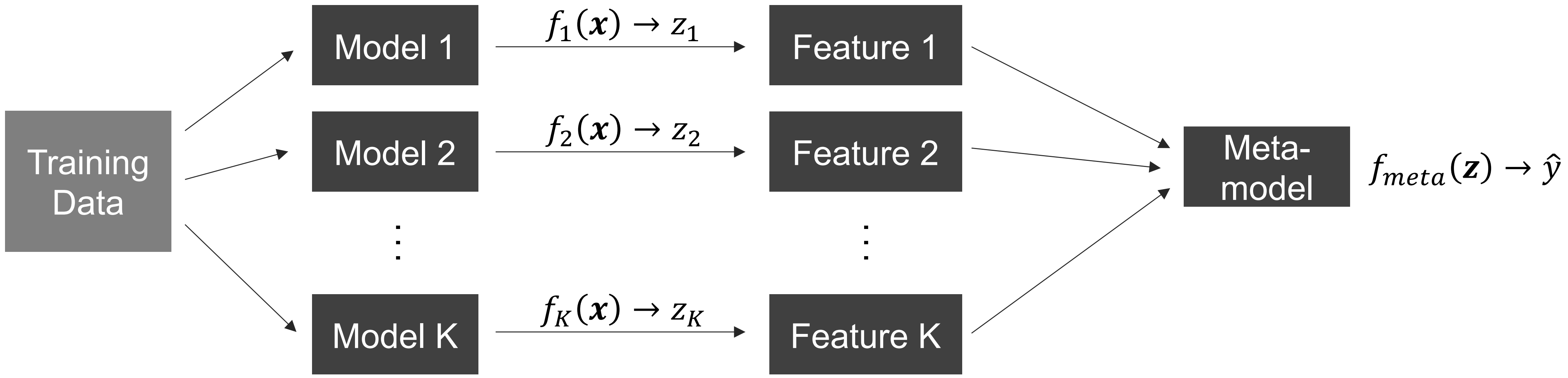
5. Update the residuals $r_i = r_i - \lambda \hat{f}_i(\boldsymbol{x}_i)$

6. Output the boosted model: $\hat{f}(\boldsymbol{x}) = \sum_{i=1}^{B} \lambda \hat{f}_i(\boldsymbol{x})$

Model Stacking

Train multiple supervised learning techniques (could be different models)

THEN Train a supervised learning technique that includes the **outputs** of the other models as **features**



Training Data
Model 1
Model 2
Model K
$f_1(\boldsymbol{x}) \rightarrow z_1$
$f_2(\boldsymbol{x}) \rightarrow z_2$
$f_K(\boldsymbol{x}) \rightarrow z_K$
Feature 1
Feature 2
Feature K
Meta-model
$f_{meta}(\boldsymbol{z}) \rightarrow \hat{y}$

Supervised Learning Techniques

Covered so far

- ● Linear Regression
- ●● K-Nearest Neighbors
- ● Perceptron
- ● Logistic Regression
- ● Linear Discriminant Analysis
- ● Quadratic Discriminant Analysis
- ● Naïve Bayes
- ●● Decision Trees and Random Forests
- ●● Ensemble methods (bagging, boosting, stacking)

Can be used with many machine learning techniques

Appropriate for:
- 🟢 Classification
- ● Regression