# Tree-based Models and Ensembles

# Supervised learning in practice

## Preprocessing
### Explore & prepare data

**Data Visualization and Exploration**

Identify patterns that can be leveraged for learning

**Data Cleaning**

- Missing data
- Noisy data
- Erroneous data

**Scaling (Standardization)**

Prepare data for use in scale-dependent algorithms.

**Feature Extraction**

Dimensionality reduction eliminates redundant information

## Model training

**1** Supervised Learning Models: Linear models and KNN
(enough to get started using supervised learning)

Select model options

**5** Other algorithms and concepts:
- Generative vs discriminative models
- **Parametric vs nonparametric models**
- **Model ensembles**
- Feature/representation learning (neural networks, deep learning)

**4** How to control model overfit: regularization strategies for model refinement

Iteratively fine tune the model

## Performance evaluation

**Make a prediction on validation data**

**2** Evaluating model performance and comparing models

Classification

Precision, Recall, $F_1$,

**3** How to make decisions using models

**Regression**

MSE, explained variance, $R^2$

# 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)

# Parametric vs Nonparametric techniques

**Non-parametric Models**

Complexity of the model grows with the size of the training data

- K-Nearest Neighbors
- Decision Trees

**Parametric Models**

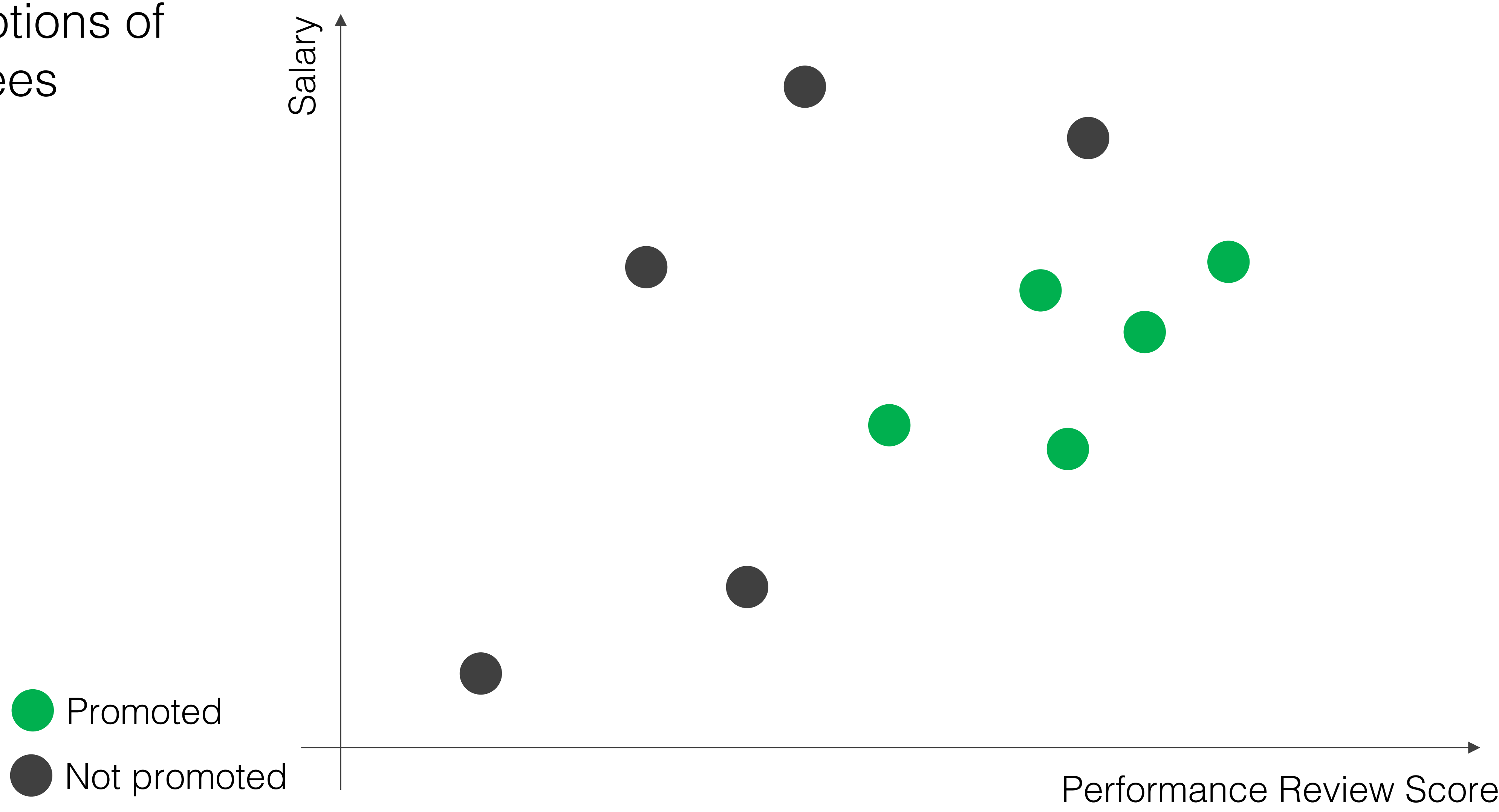
Fixed number of parameters (i.e. a fixed structure)

- Linear regression
- Logistic regression
- LDA, QDA
- Naïve Bayes with Gaussian likelihoods

# Classification and Regression Trees (CART)

Classification trees = decision trees

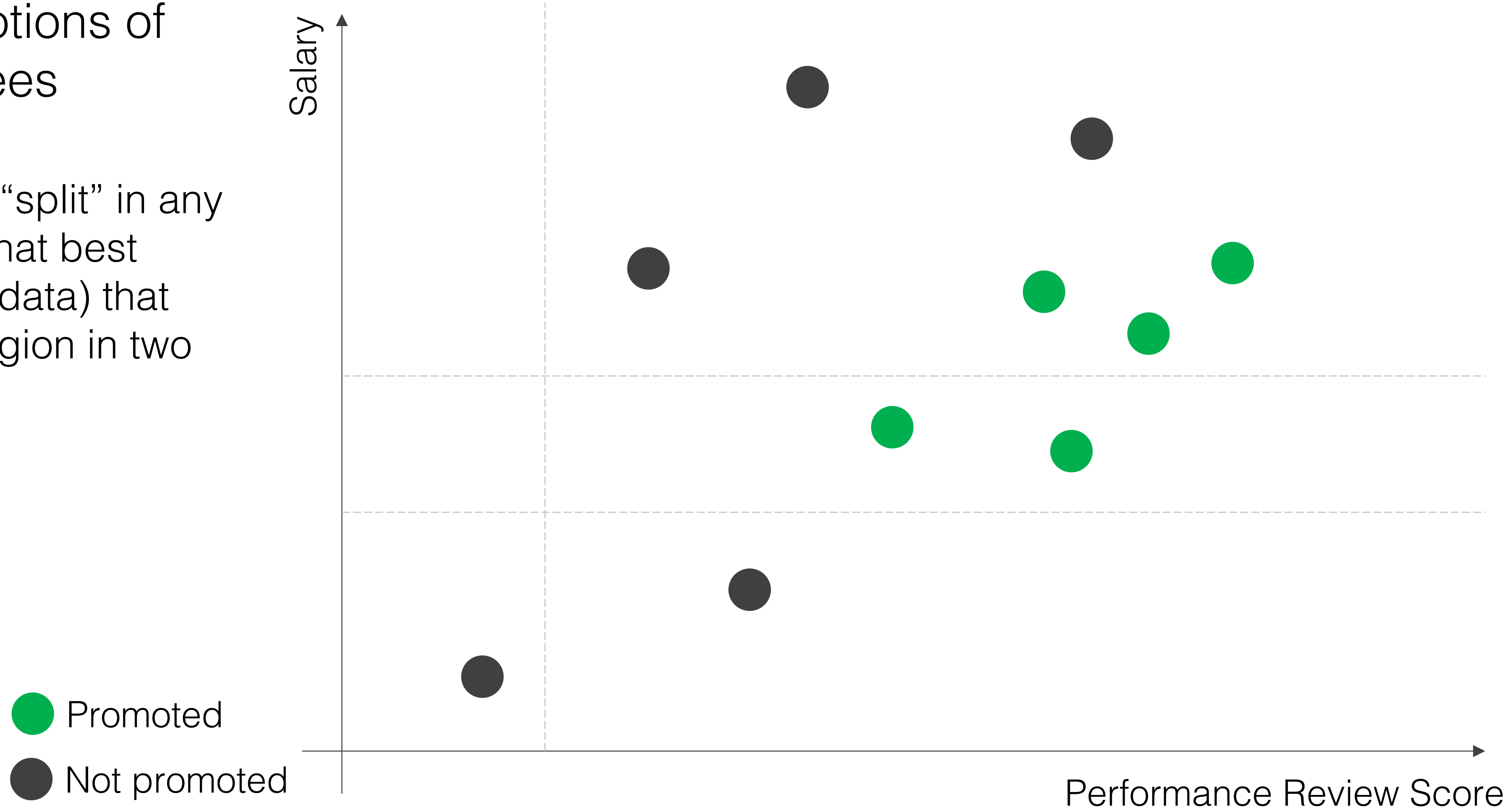
Predicting promotions of salaried employees



# Classification and Regression Trees (CART)

Predicting promotions of salaried employees

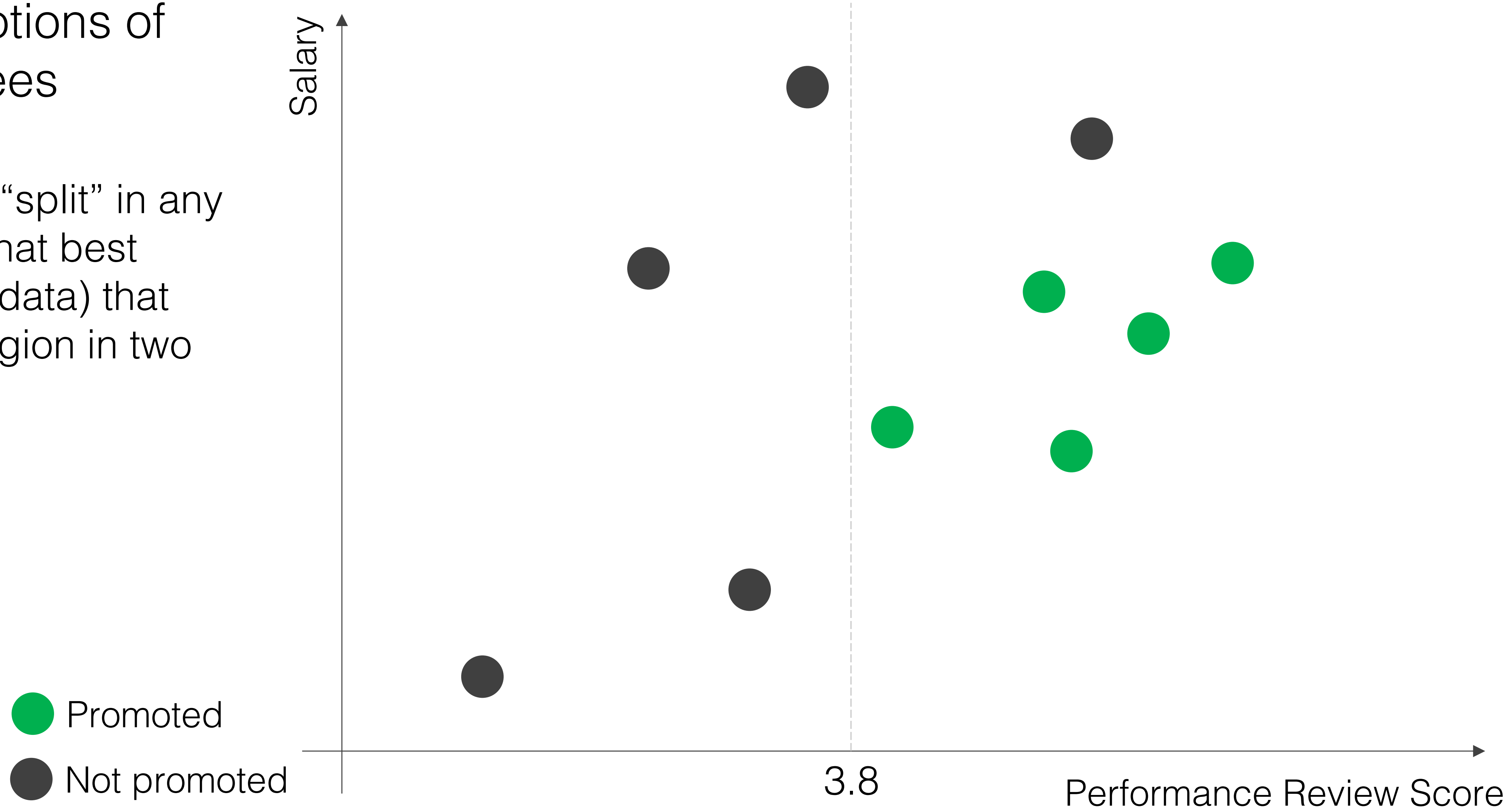
1. Find the best "split" in any one feature (that best classifies the data) that divides the region in two



# 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

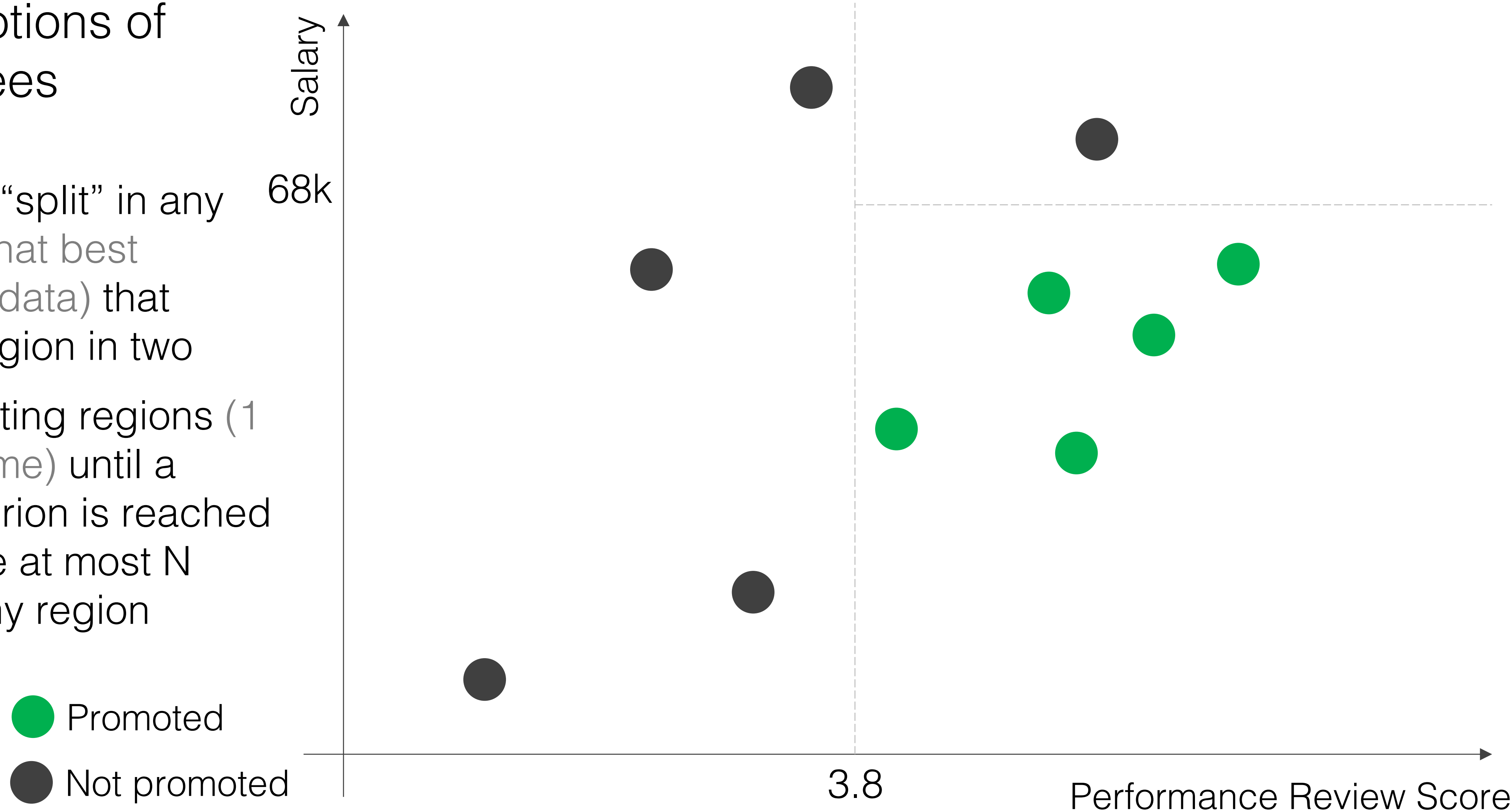


# Classification and Regression Trees (CART)

Predicting promotions of salaried employees

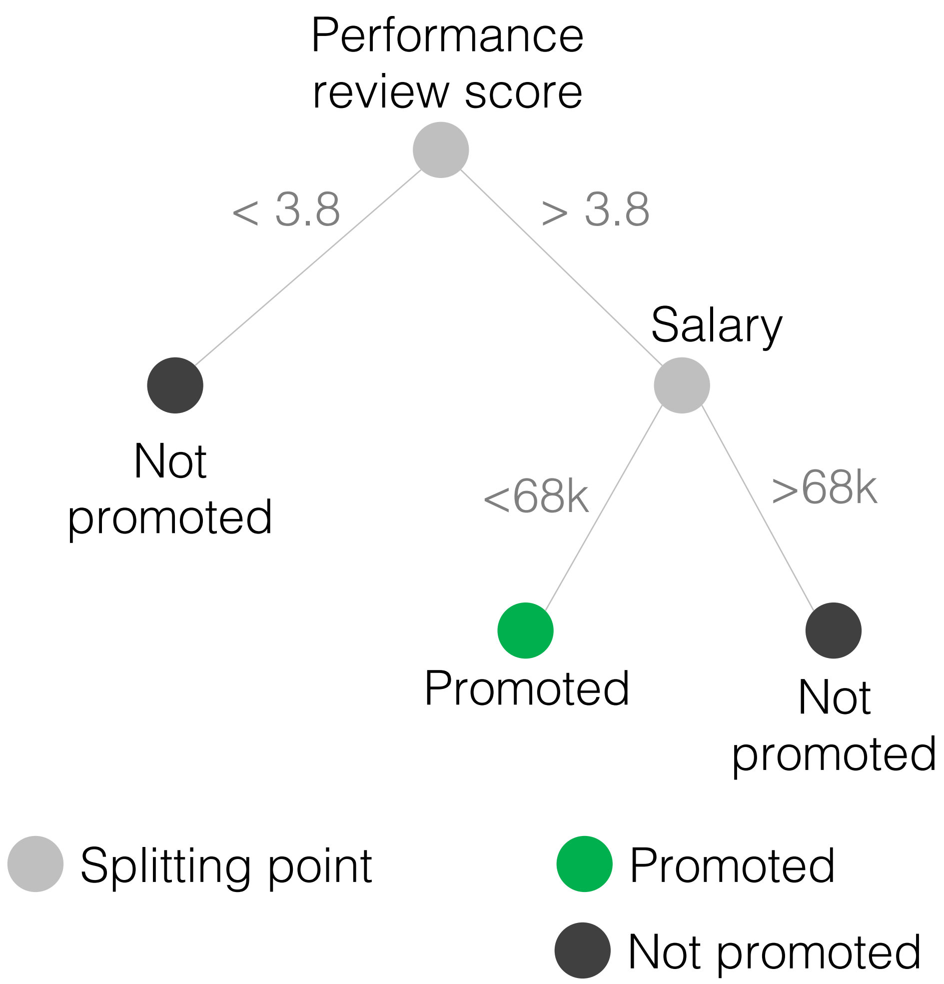
1. Find the best "split" in any one feature (that best classifies the data) that divides the region in two
2. Continue splitting regions (1 feature at a time) until a stopping criterion is reached (e.g. there are at most N samples in any region

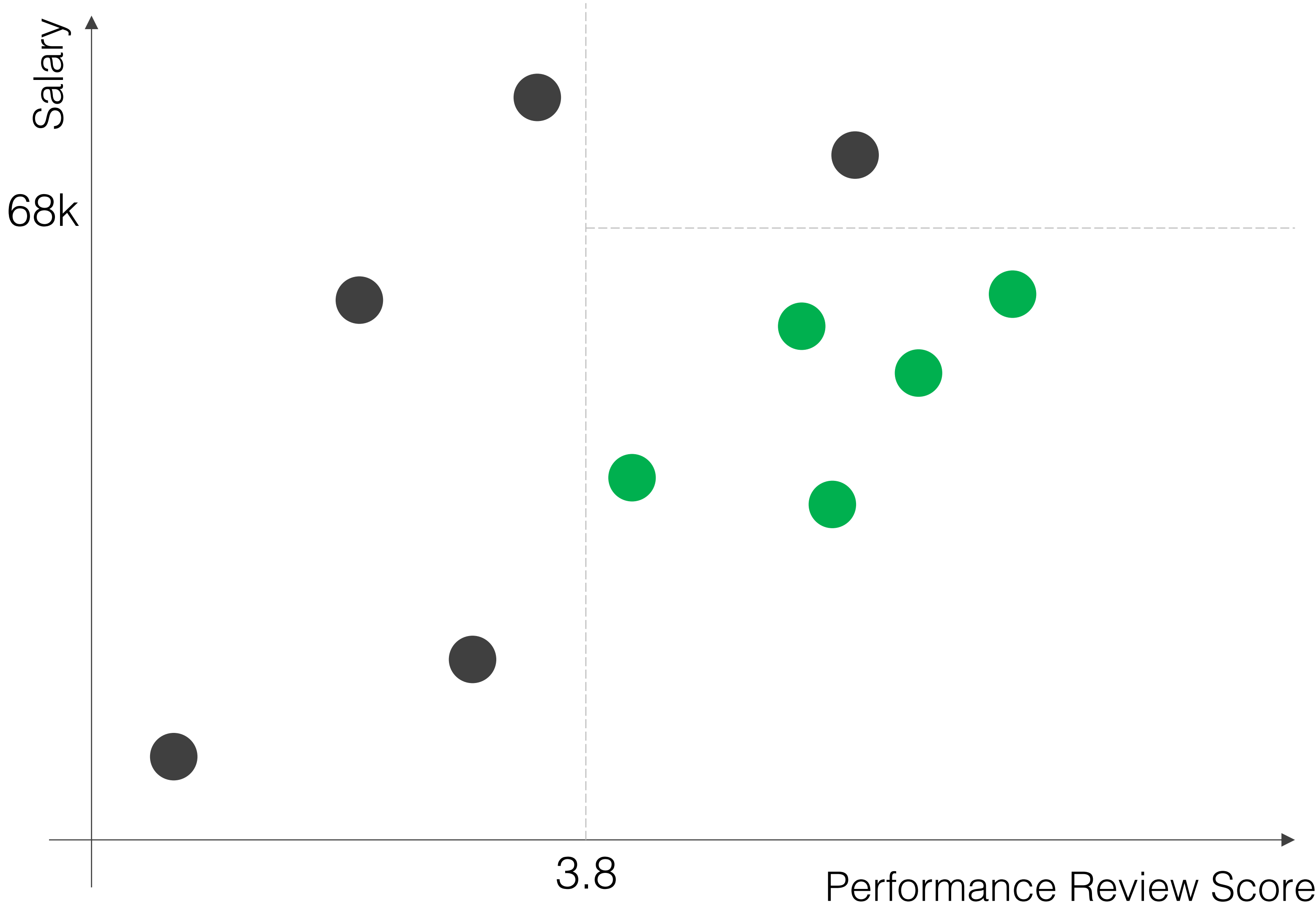
**Greedy, recursive binary tree**



# Classification and Regression Trees (CART)

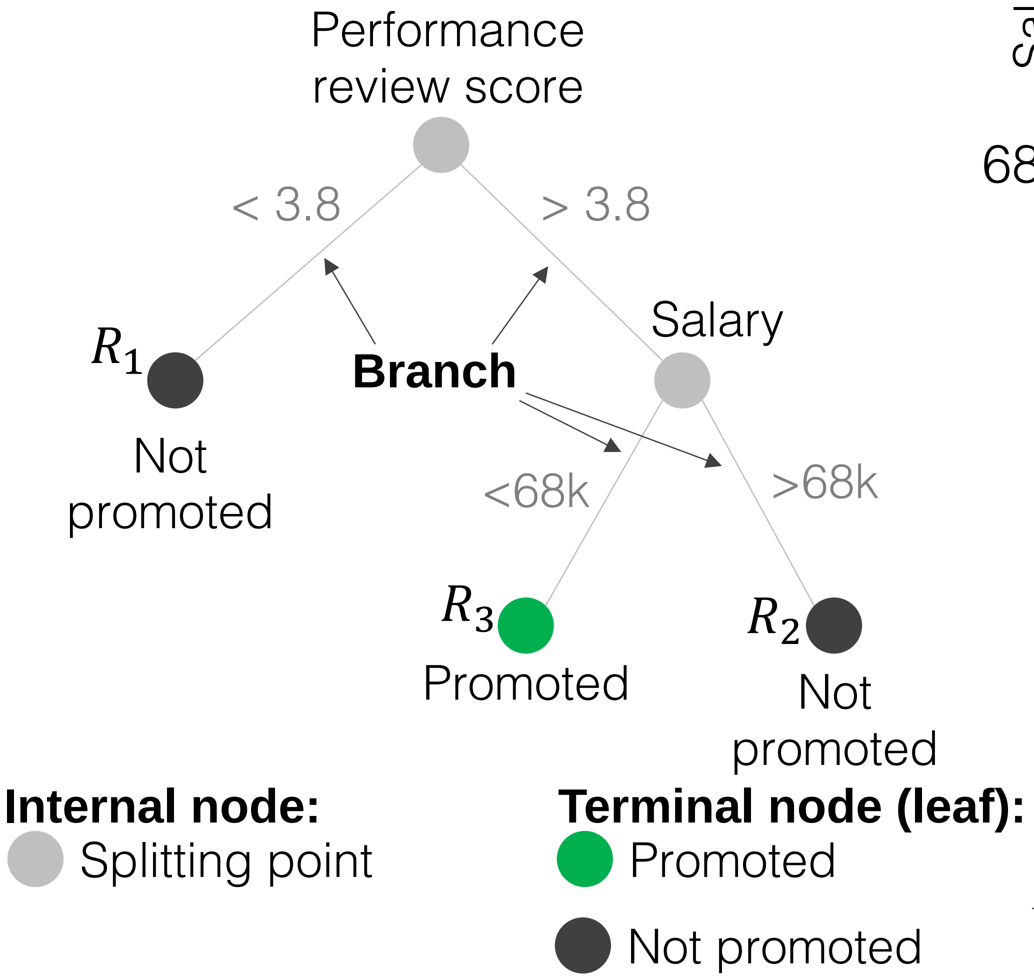
Tree representation:

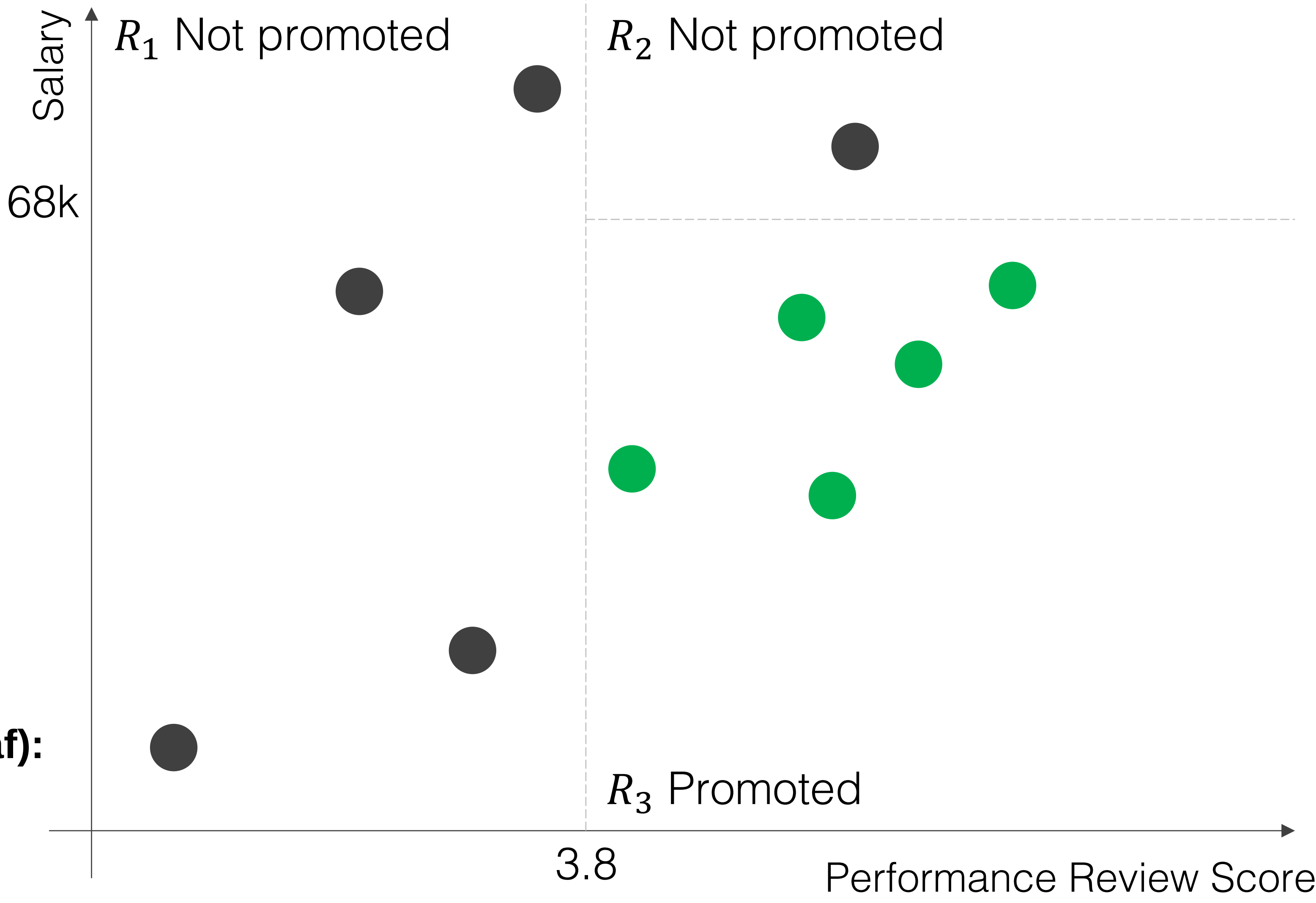


# Classification and Regression Trees (CART)

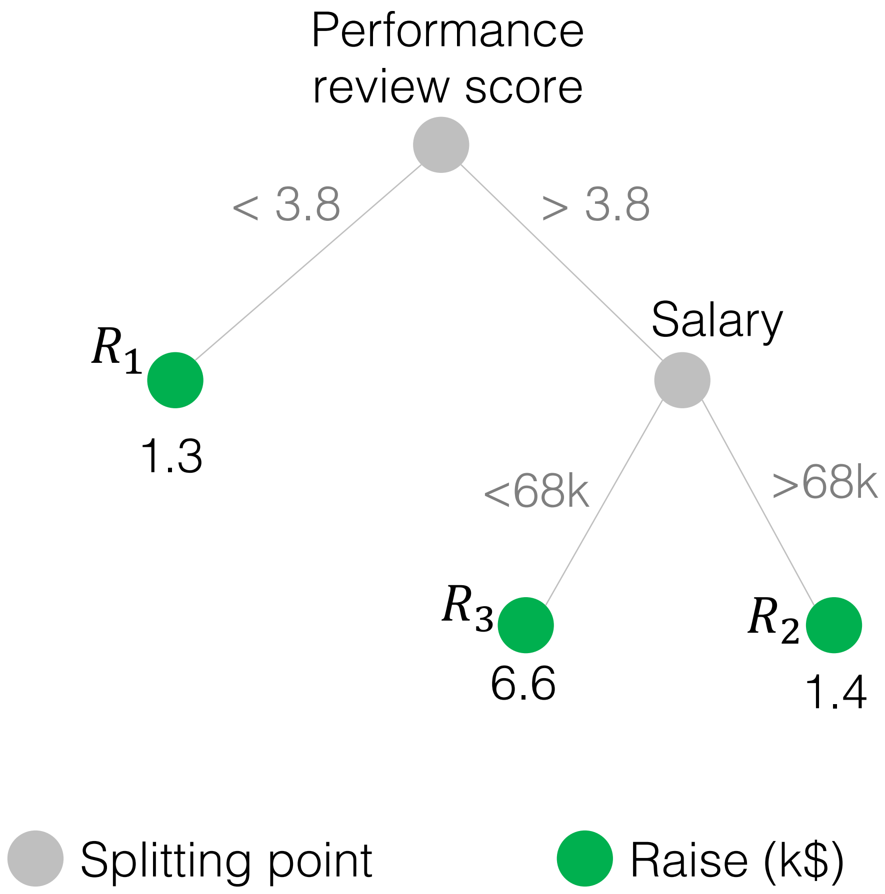
Tree representation:

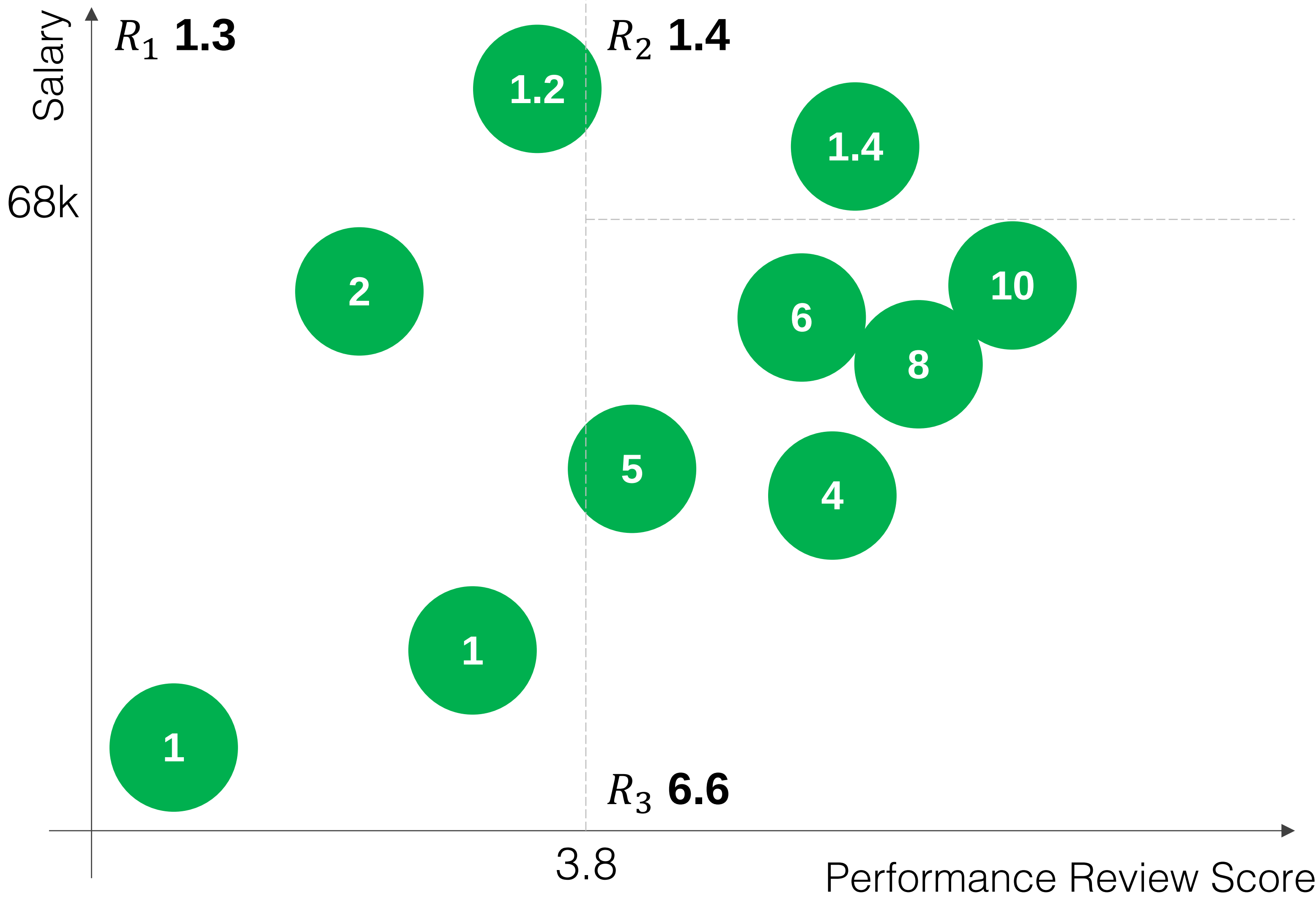


# The Regression Setting

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 $R_r$ is the set of samples in region r)

## Classification

**Misclassification rate**

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

**Gini impurity**

Measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled

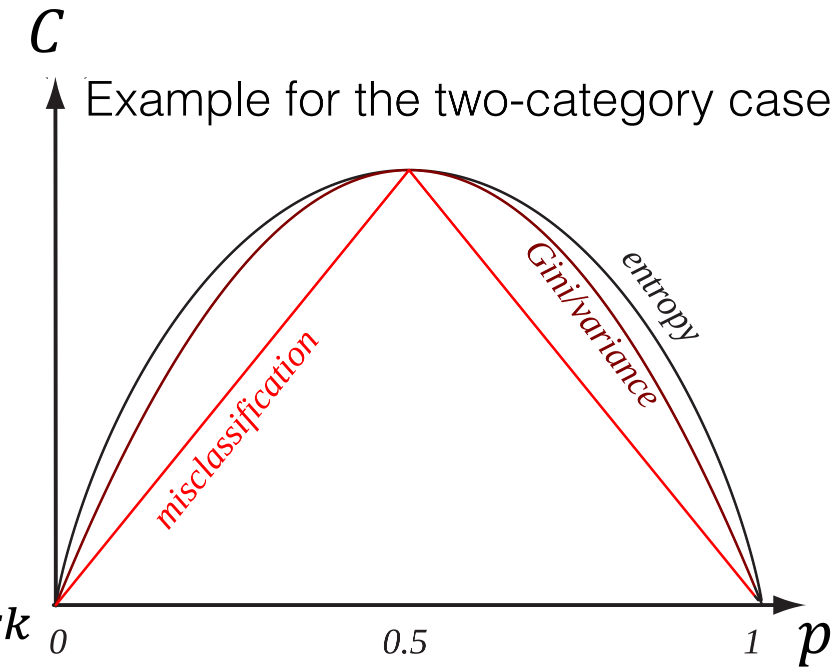
$$Q_{Gini}(r) = \sum_{k=1}^{K} \hat{p}_{rk}(1 - \hat{p}_{rk})$$

**Cross-entropy**

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

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

Example for the two-category case

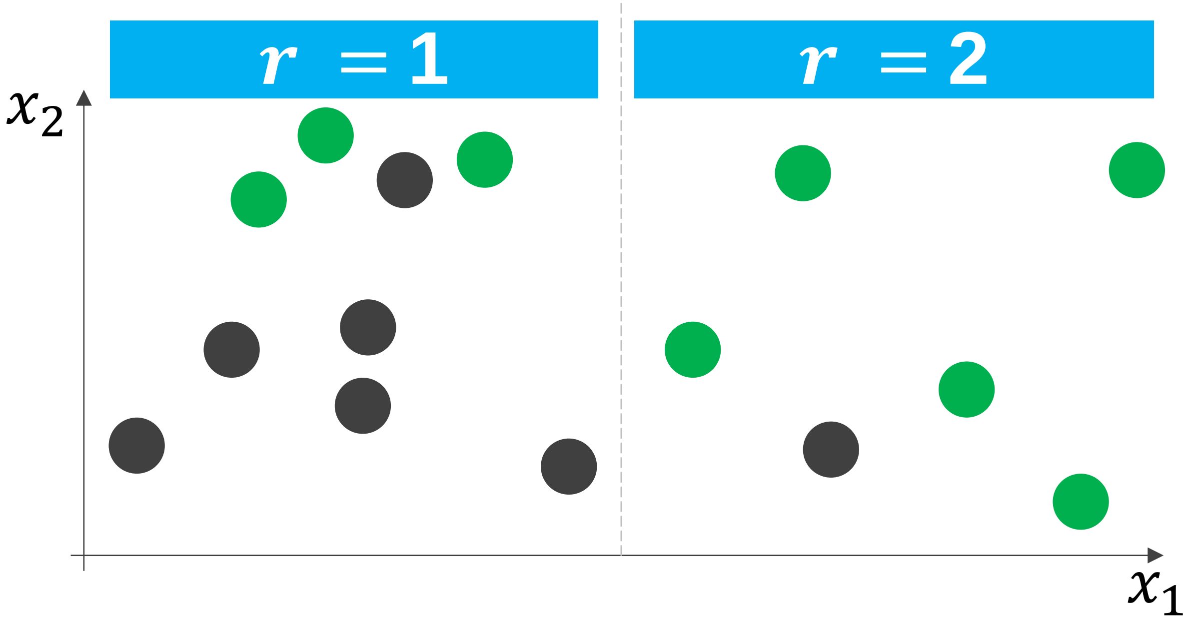



Duda, Hart, and Stork., Pattern Classification

# How to measure quality of split for classification?

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

Class 1 ●
Class 2 ●




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

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

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

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

## For each region:

| | 1 | 2 |
|---|---|---|
| **Misclassification rate** $Q_{Misclass}(r) = 1 - \max_k (\hat{p}_{rk})$ | 0.333 | 0.167 |
| **Gini impurity** $Q_{Gini}(r) = \sum_{k=1}^{K} \hat{p}_{rk}(1 - \hat{p}_{rk})$ | 0.444 | 0.278 |
| **Cross-entropy** $Q_{entropy}(r) = -\sum_{k=1}^{K} \hat{p}_{rk} \log \hat{p}_{rk}$ | 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 class of observations in each region (leads to complete overfit)

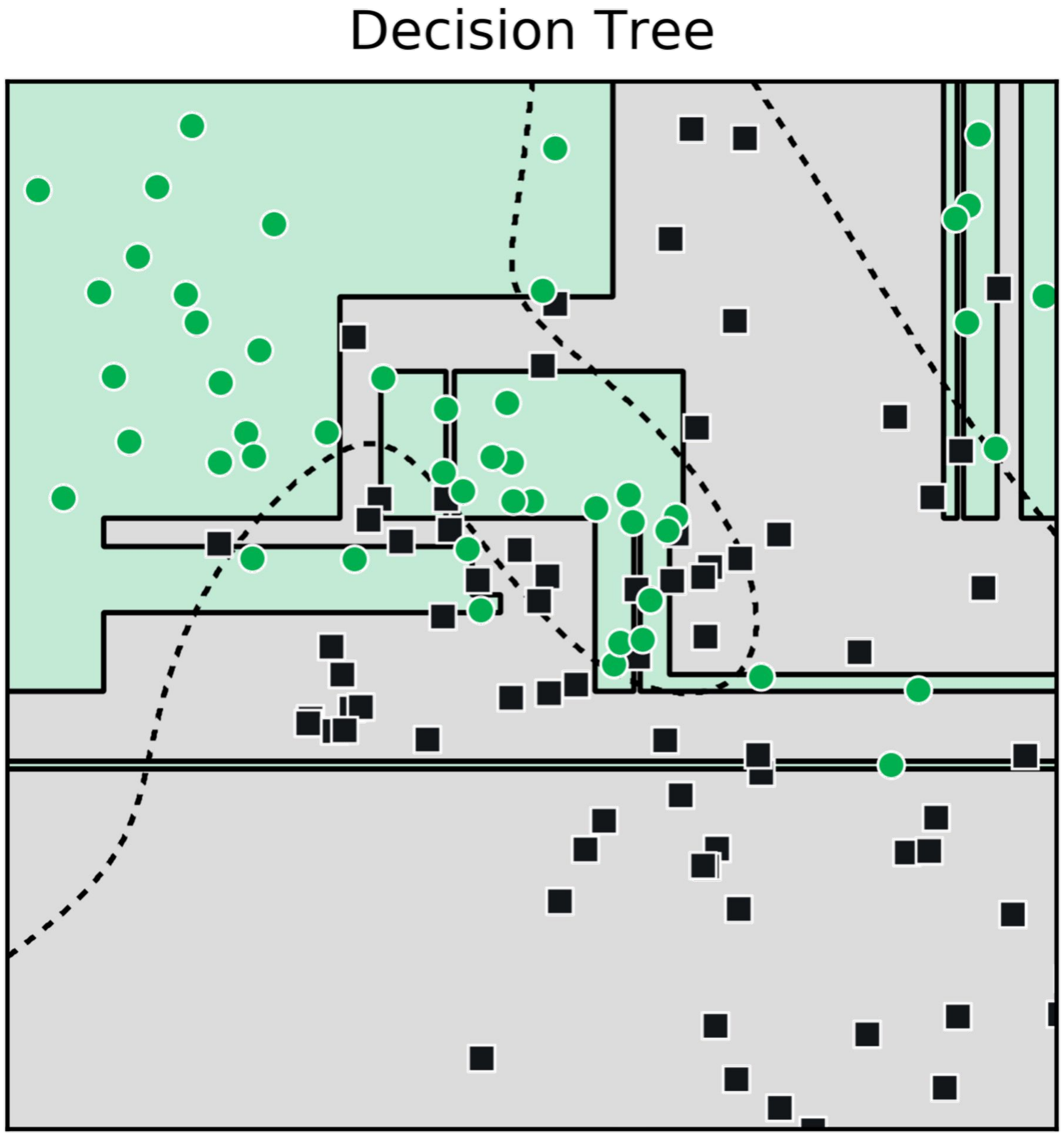
**Pruning** the tree reduces this overfit (removing splits after the tree is formed)

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

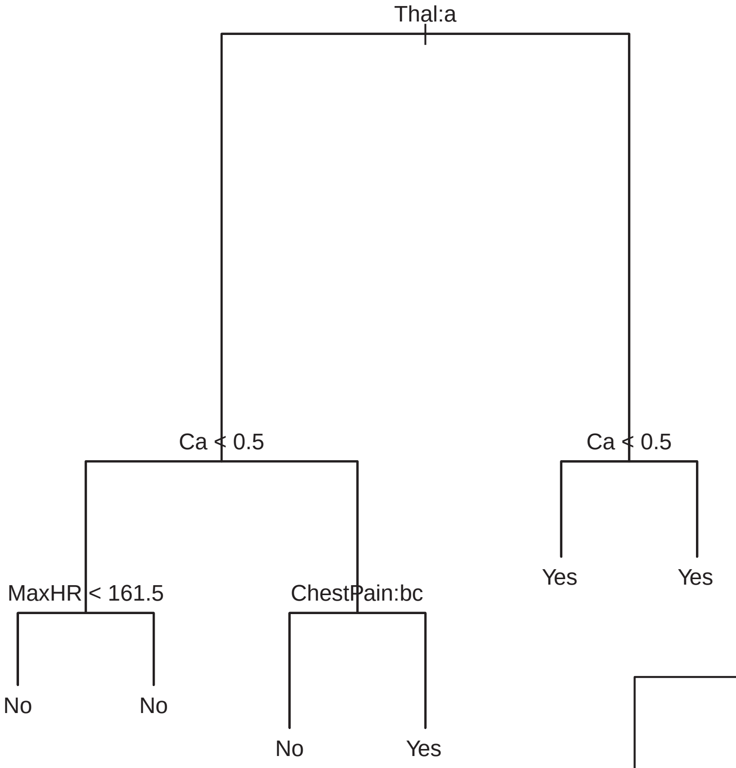
$$C_{Prune} = \sum_{j=1}^{T} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2 + \alpha T$$

penalty on number of terminal nodes

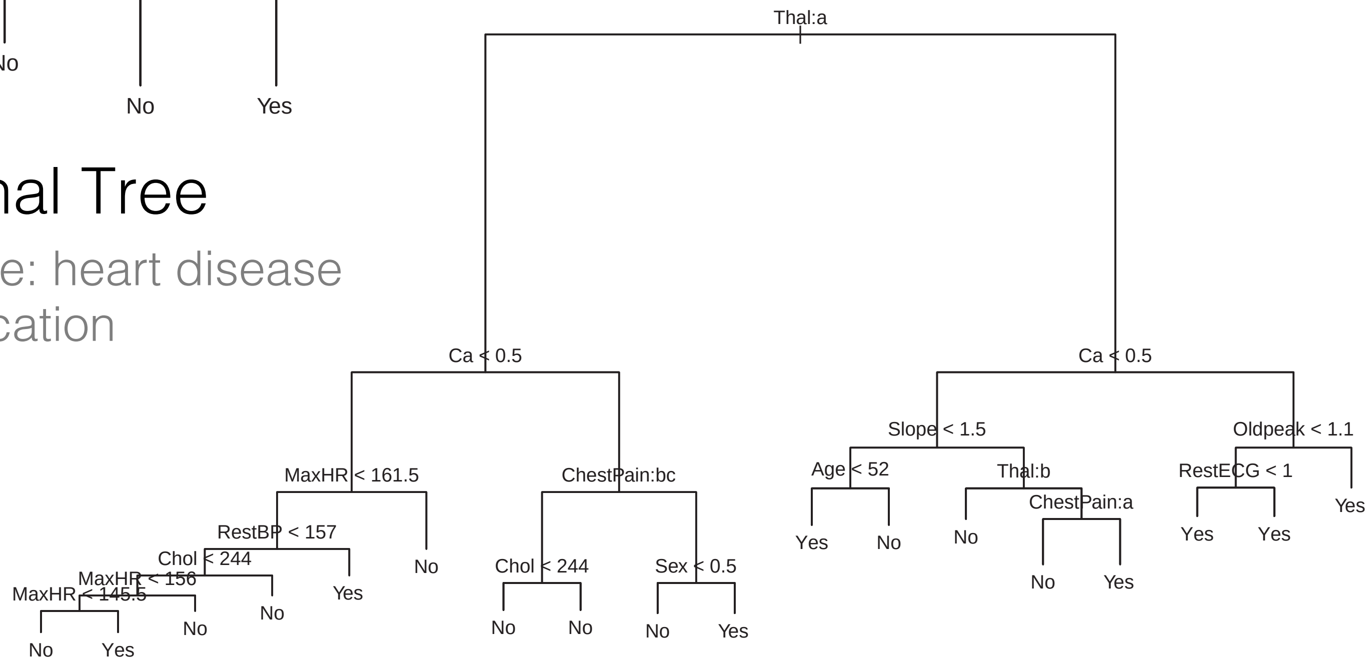
number of terminal nodes



# Pruning example



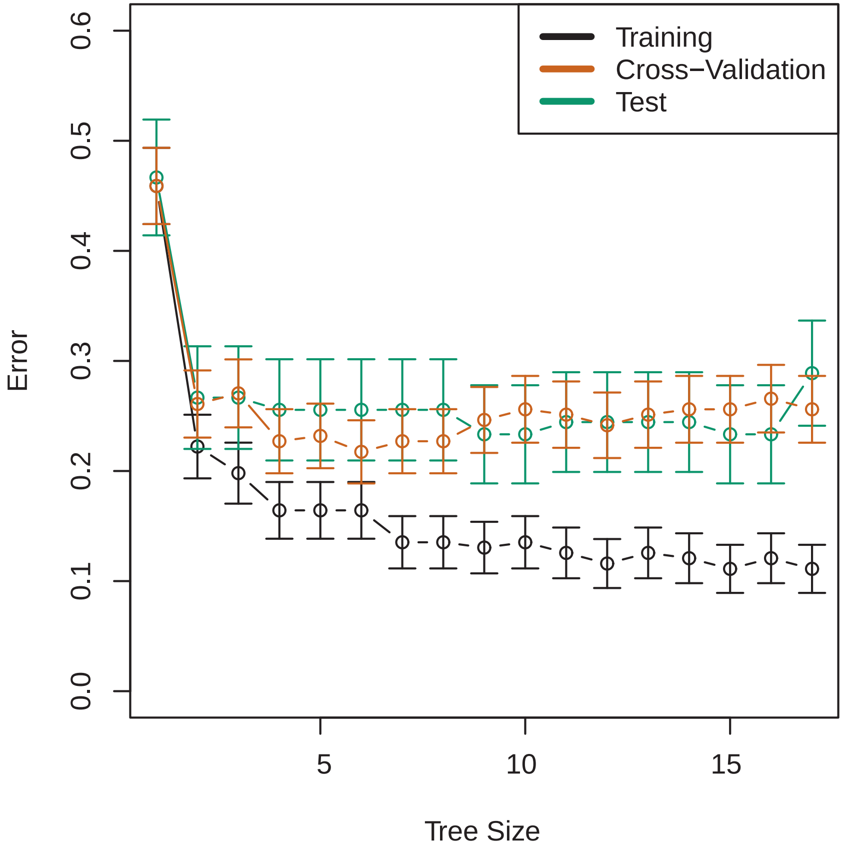

Pruned Tree




Original Tree

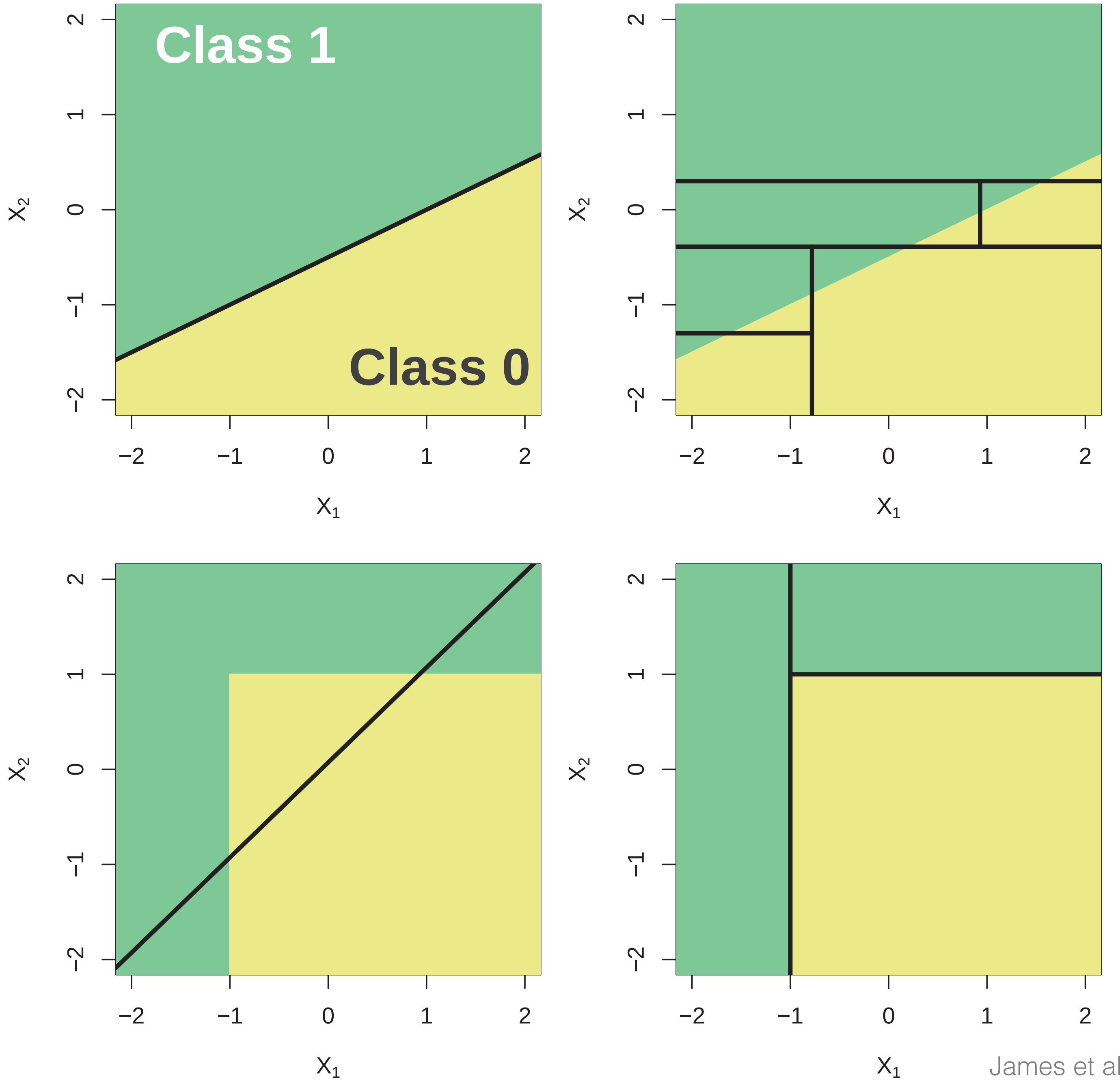
Example: heart disease classification

Performance




James et al., An Introduction to Statistical Learning

# Linear model




# Classification Tree

Struggle when the boundary is not parallel to an axis

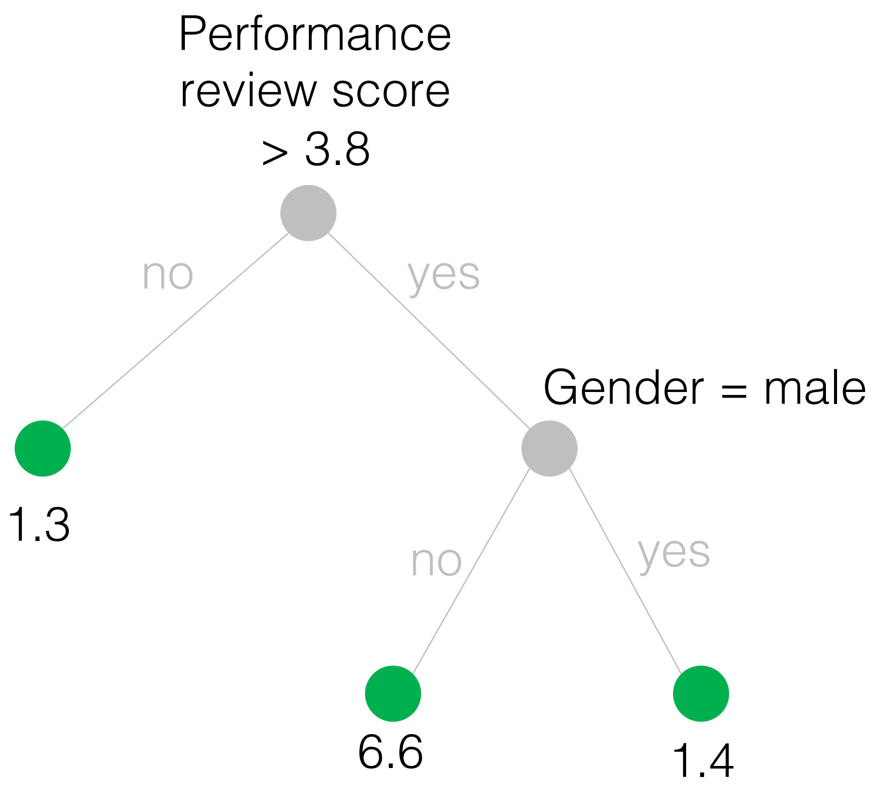
…nonlinear feature transforms could help…

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 other methods

Tend to overfit
(have high variance)

# Ensemble learning

Combining models to improve performance beyond any individual model alone

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

# Reducing Variance or Bias through ensembles

| | **Bagging** | **Boosting** |
|---|---|---|
| Models in ensemble: | high variance, low bias<br>(i.e. overfit models) | high bias, low variance<br>(i.e. underfit models, “weak learners”) |
| 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**

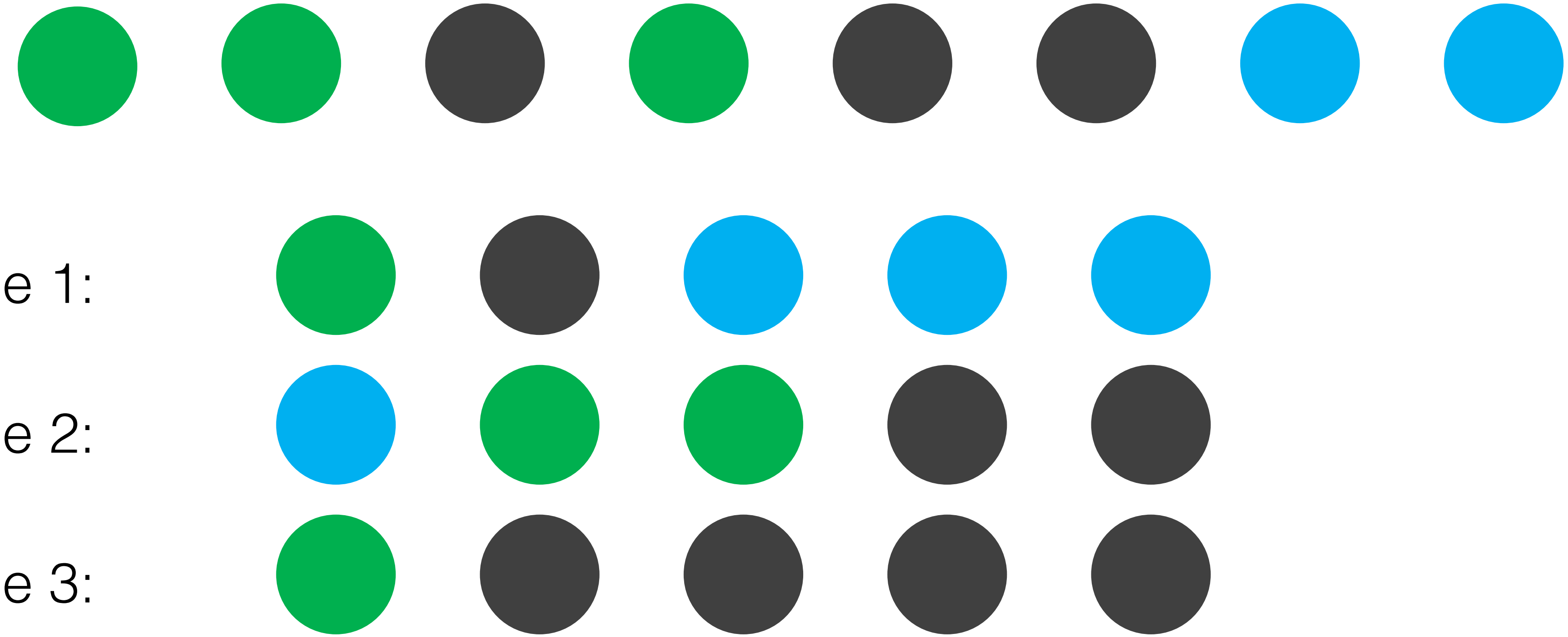
Recall bootstrap sampling (sampling with replacement):

Original Data:

Bootstrapped sample 1:

Bootstrapped sample 2:

Bootstrapped sample 3:



# Bagging

Decision trees are a popular base model

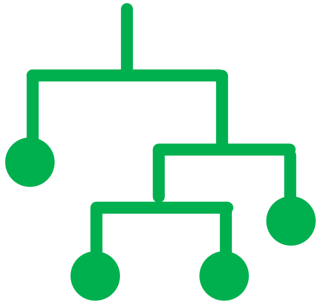
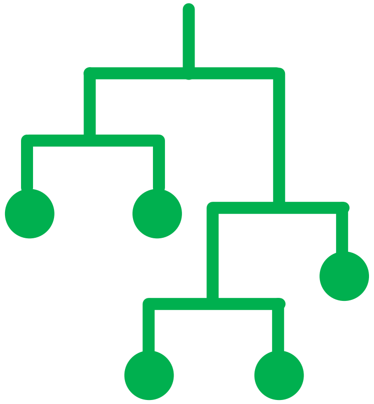
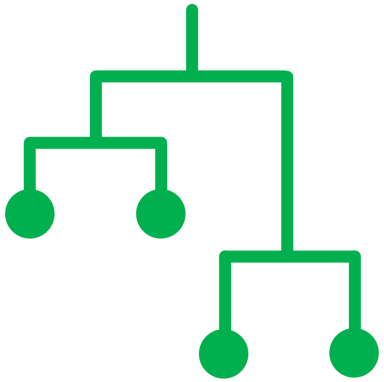
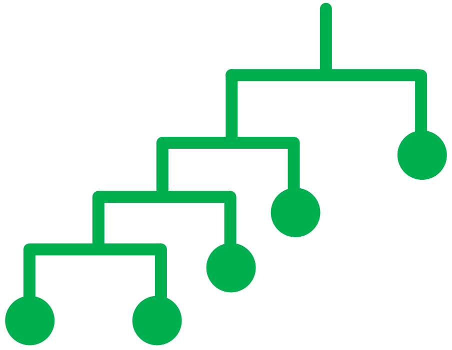
Bootstrap aggregation

1. Create a random bootstrap sample from the training data
2. Train a model on that bootstrap sample and call it $\hat{f}_b(\boldsymbol{x})$
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:

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

(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: |  |  |  |  |

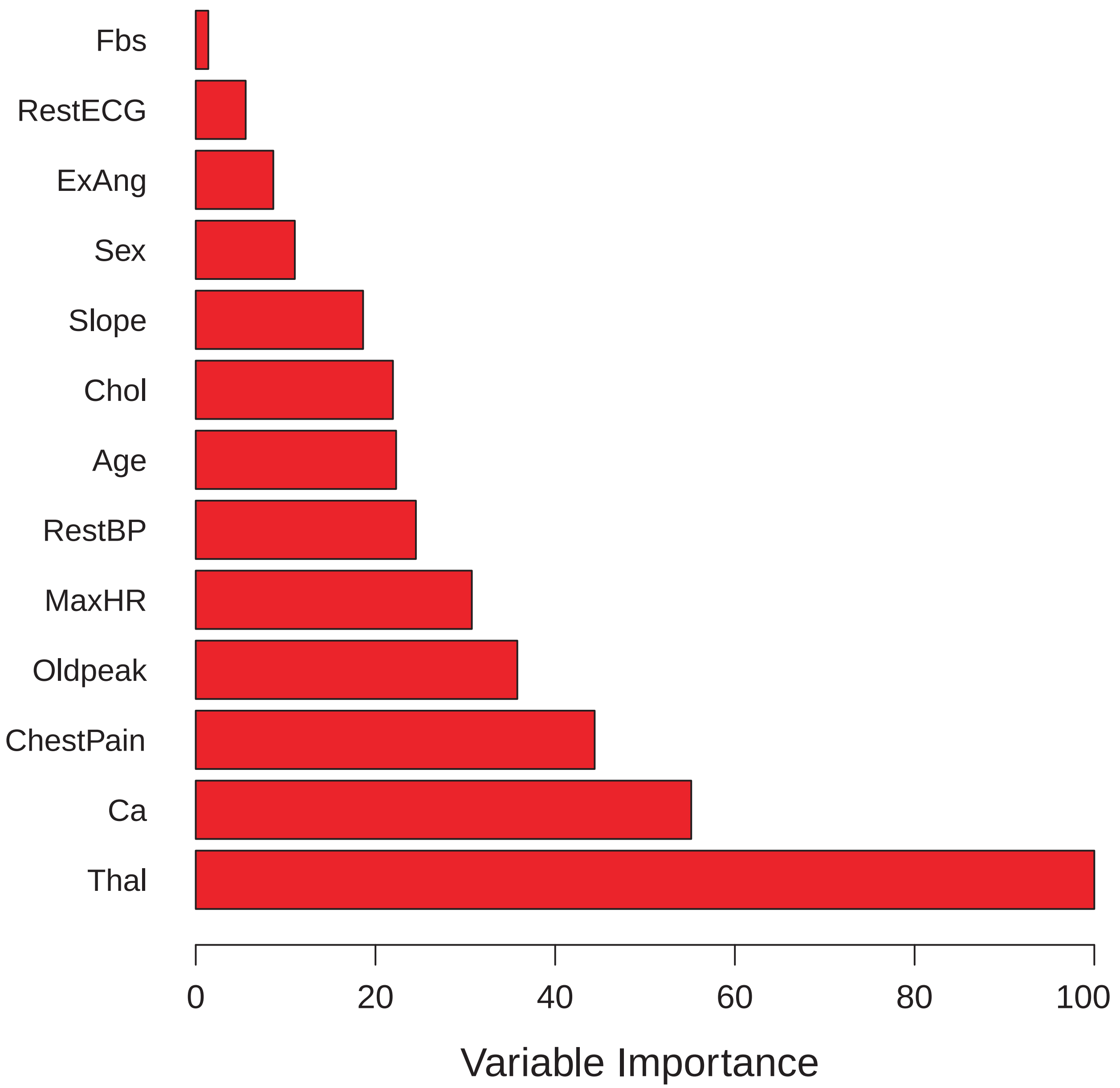
# Variable Importance

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

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

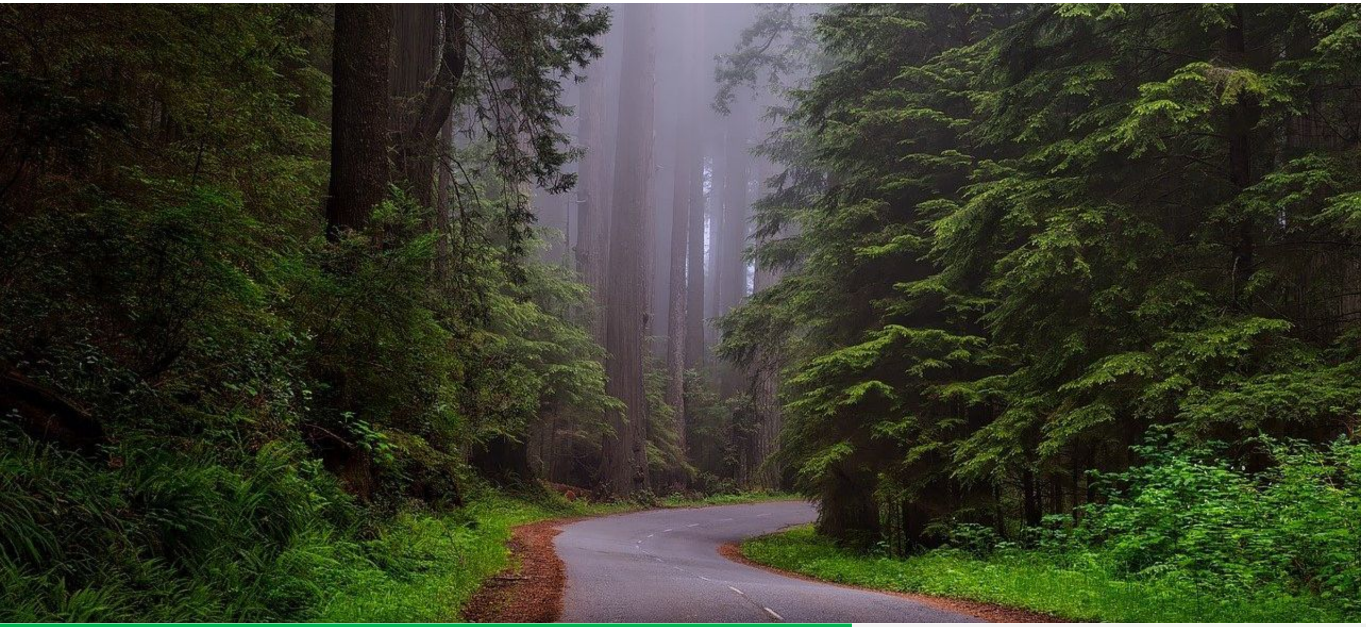
1. Calculate the total amount the error (or impurity) decreased by splitting on each feature.

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.

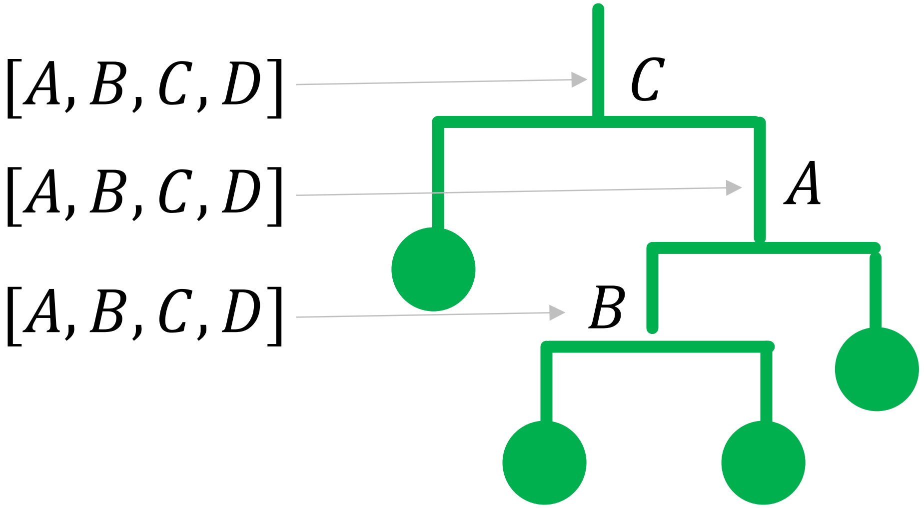
Each time a split is considered, a **random subset of $m$ features** is selected as candidates from the full set of $p$ features

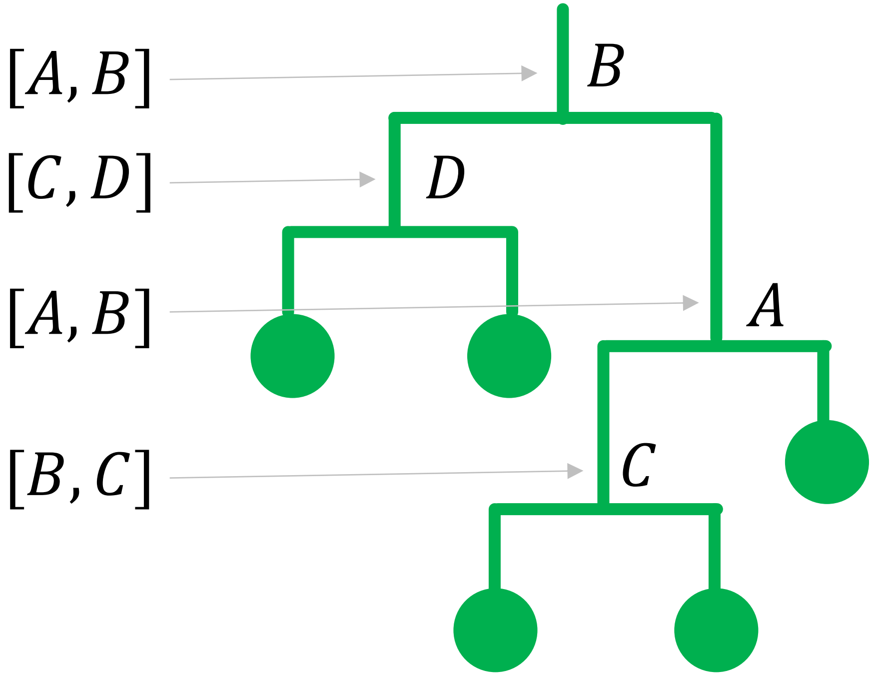
Typically chose: $m = \sqrt{p}$

(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:



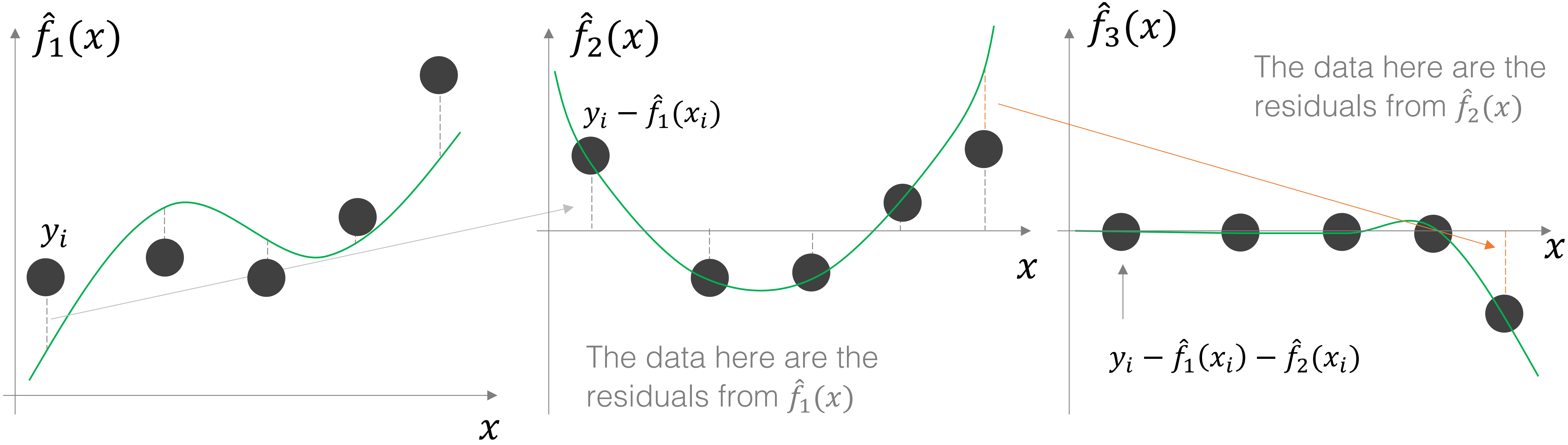
# Boosting

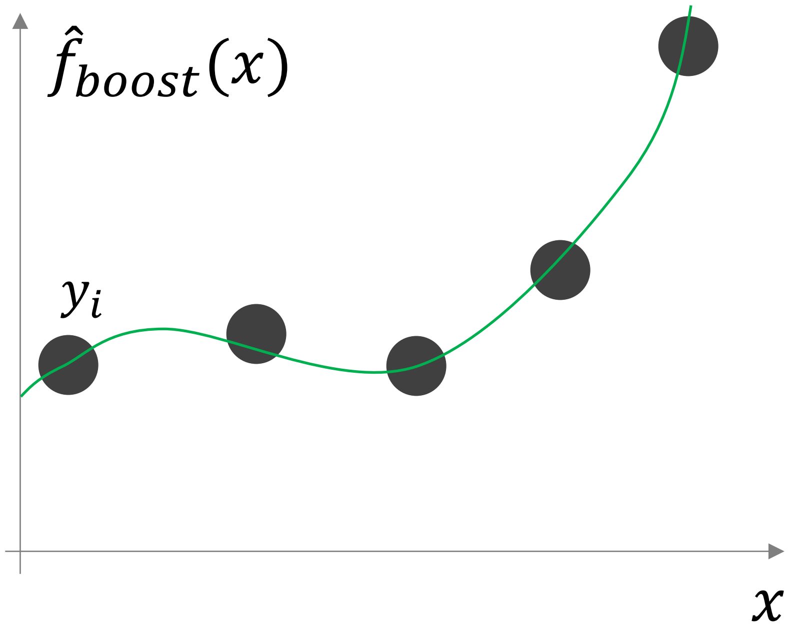


**Bagging** created trees that were designed to be as independent as possible

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

# Boosting




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 for regression trees

1. Select the number of models to train, $B$, and learning rate $\lambda$

$\lambda$ 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

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

Often this is just a small number of splits (a stump)

4. Update $\hat{f}(\boldsymbol{x}) \leftarrow \hat{f}(\boldsymbol{x}) + \lambda \hat{f}_b(\boldsymbol{x})$

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

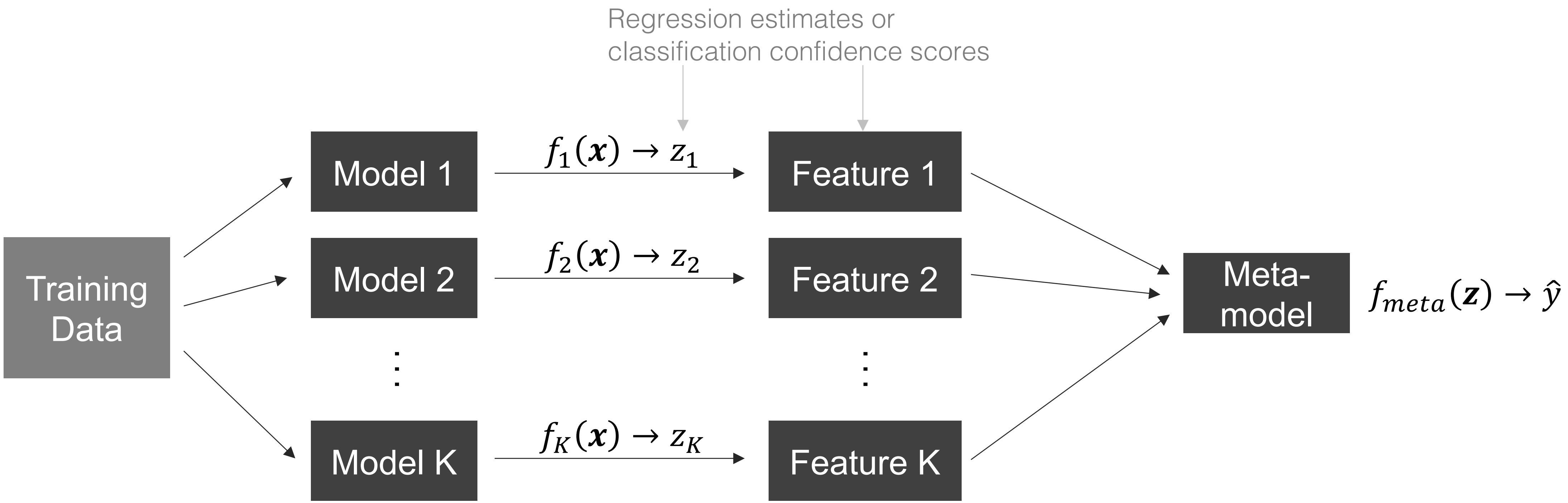
Repeat $B$ times

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

# 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**



# 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 numerous machine learning techniques, often CART

Appropriate for:
- Classification
- Regression