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

3 How to make decisions using models

Regression

MSE, explained variance, R^2

Supervised Learning Techniques

Covered so far

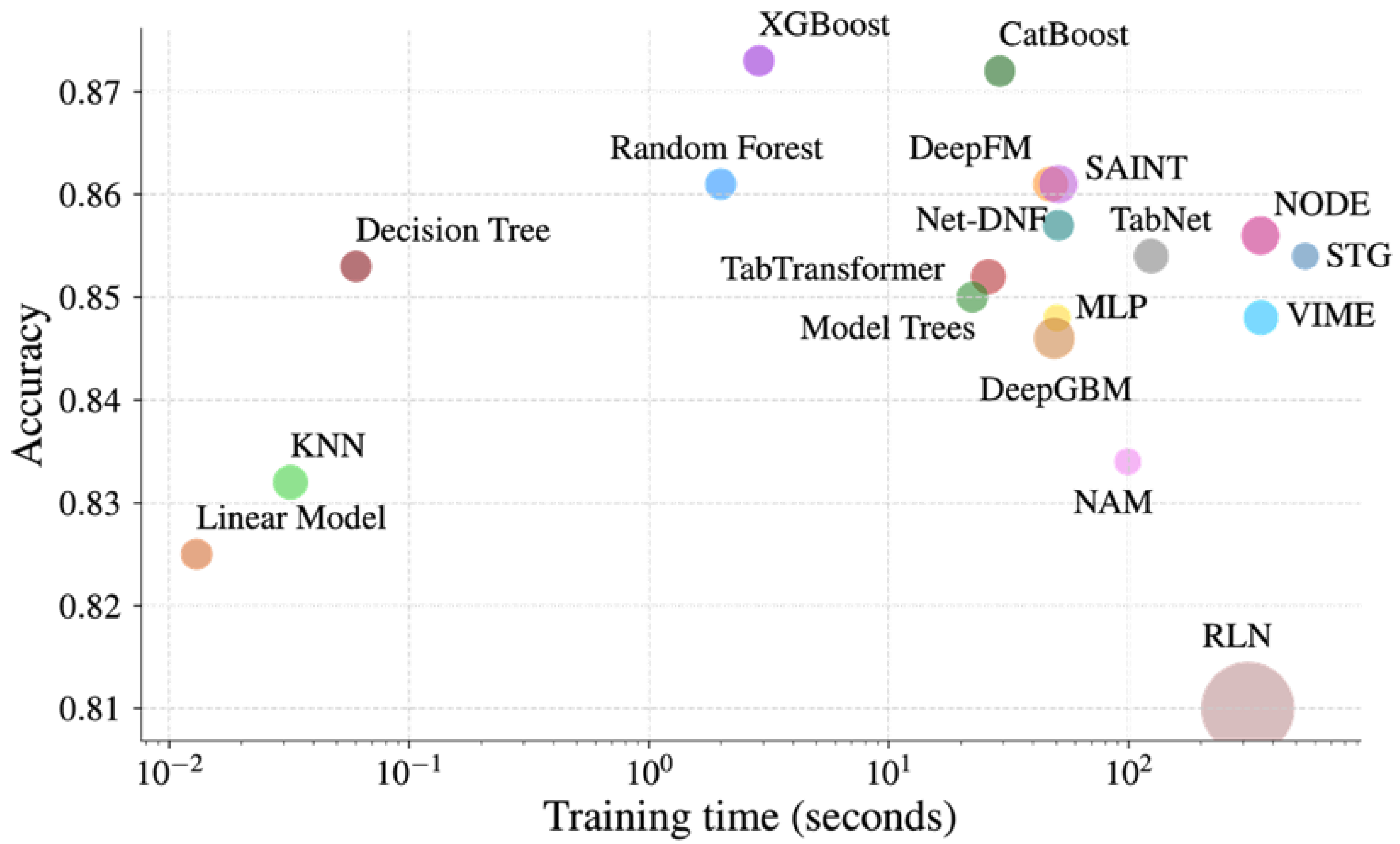
- Linear Regression
- K-Nearest Neighbors
- Logistic Regression
- Linear/Quadratic Discriminant Analysis
- Naïve Bayes

- Decision Trees and Random Forests
- Ensemble methods (bagging, boosting, stacking)

Decision Tree Ensembles

"**gradient-boosted tree ensembles** still mostly outperform deep learning models on supervised learning tasks [on **heterogeneous tabular data**]"

a data set with a fixed number of features that are either continuous or categorical



Results on Adult Income dataset from UCI repository. Task: predict whether income exceeds $50K/yr based on census data

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M. and Kasneci, G., 2022. Deep neural networks and tabular data: A survey. IEEE Transactions on Neural Networks and Learning Systems.

	Benchmark datasets:	HELOC		Adult		HIGGS		Covertype		Cal. Housing
		Acc ↑	AUC ↑	Acc ↑	AUC ↑	Acc ↑	AUC ↑	Acc ↑	AUC ↑	MSE ↓
Gradient-boosted decision tree ensembles	Linear Model	73.0±0.0	80.1±0.1	82.5±0.2	85.4±0.2	64.1±0.0	68.4±0.0	72.4±0.0	92.8±0.0	0.528±0.008
	KNN [65]	72.2±0.0	79.0±0.1	83.2±0.2	87.5±0.2	62.3±0.1	67.1±0.0	70.2±0.1	90.1±0.2	0.421±0.009
	Decision Tree [197]	80.3±0.0	89.3±0.1	85.3±0.2	89.8±0.1	71.3±0.0	78.7±0.0	79.1±0.0	95.0±0.0	0.404±0.007
	Random Forest [198]	82.1±0.2	90.0±0.2	86.1±0.2	91.7±0.2	71.9±0.0	79.7±0.0	78.1±0.1	96.1±0.0	0.272±0.006
	XGBoost [53]	83.5±0.2	92.2±0.0	87.3±0.2	92.8±0.1	77.6±0.0	85.9±0.0	**97.3±0.0**	**99.9±0.0**	0.206±0.005
	LightGBM [78]	83.5±0.1	92.3±0.0	**87.4±0.2**	**92.9±0.1**	77.1±0.0	85.5±0.0	93.5±0.0	99.7±0.0	**0.195±0.005**
	CatBoost [79]	**83.6±0.3**	**92.4±0.1**	87.2±0.2	92.8±0.1	77.5±0.0	85.8±0.0	96.4±0.0	99.8±0.0	0.196±0.004
	Model Trees [199]	82.6±0.2	91.5±0.0	85.0±0.2	90.4±0.1	69.8±0.0	76.7±0.0	-	-	0.385±0.019
Deep Learning	MLP [200]	73.2±0.3	80.3±0.1	84.8±0.1	90.3±0.2	77.1±0.0	85.6±0.0	91.0±0.4	76.1±3.0	0.263±0.008
	DeepFM [15]	73.6±0.2	80.4±0.1	86.1±0.2	91.7±0.1	76.9±0.0	83.4±0.0	-	-	0.260±0.006
	DeepGBM [70]	78.0±0.4	84.1±0.1	84.6±0.3	90.8±0.1	74.5±0.0	83.0±0.0	-	-	0.856±0.065
	RLN [72]	73.2±0.4	80.1±0.4	81.0±1.6	75.9±8.2	71.8±0.2	79.4±0.2	77.2±1.5	92.0±0.9	0.348±0.013
	TabNet [5]	81.0±0.1	90.0±0.1	85.4±0.2	91.1±0.1	76.5±1.3	84.9±1.4	93.1±0.2	99.4±0.0	0.346±0.007
	VIME [88]	72.7±0.0	79.2±0.0	84.8±0.2	90.5±0.2	76.9±0.2	85.5±0.1	90.9±0.1	82.9±0.7	0.275±0.007
	TabTransformer [98]	73.3±0.1	80.1±0.2	85.2±0.2	90.6±0.2	73.8±0.0	81.9±0.0	76.5±0.3	72.9±2.3	0.451±0.014
	NODE [6]	79.8±0.2	87.5±0.2	85.6±0.3	91.1±0.2	76.9±0.1	85.4±0.1	89.9±0.1	98.7±0.0	0.276±0.005
	Net-DNF [57]	82.6±0.4	91.5±0.2	85.7±0.2	91.3±0.1	76.6±0.1	85.1±0.1	94.2±0.1	99.1±0.0	-
	STG [201]	73.1±0.1	80.0±0.1	85.4±0.1	90.9±0.1	73.9±0.1	81.9±0.1	81.8±0.3	96.2±0.0	0.285±0.006
	NAM [202]	73.3±0.1	80.7±0.3	83.4±0.1	86.6±0.1	53.9±0.6	55.0±1.2	-	-	0.725±0.022
	SAINT [9]	82.1±0.3	90.7±0.2	86.1±0.3	91.6±0.2	**79.8±0.0**	**88.3±0.0**	96.3±0.1	99.8±0.0	0.226±0.004

Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M. and Kasneci, G., 2022. Deep neural networks and tabular data: A survey. IEEE Transactions on Neural Networks and Learning Systems.

Parametric vs Nonparametric techniques

Parametric Models

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

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

Non-parametric Models

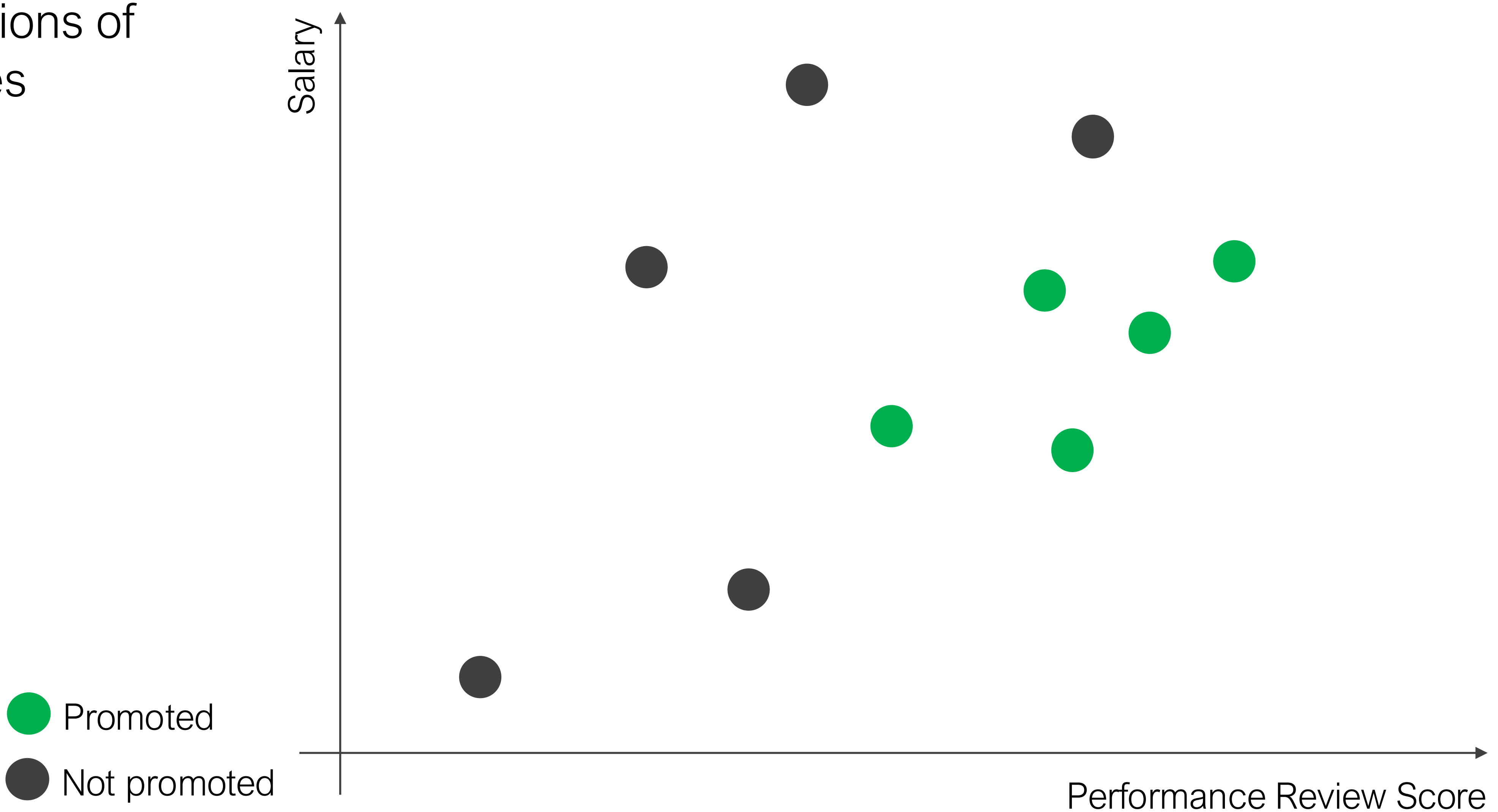
Complexity of the model grows with the size of the training data

- K-Nearest Neighbors
- Decision Trees
- Random forests
- Gradient boosted decision trees

Classification and Regression Trees (CART)

Classification trees = decision trees

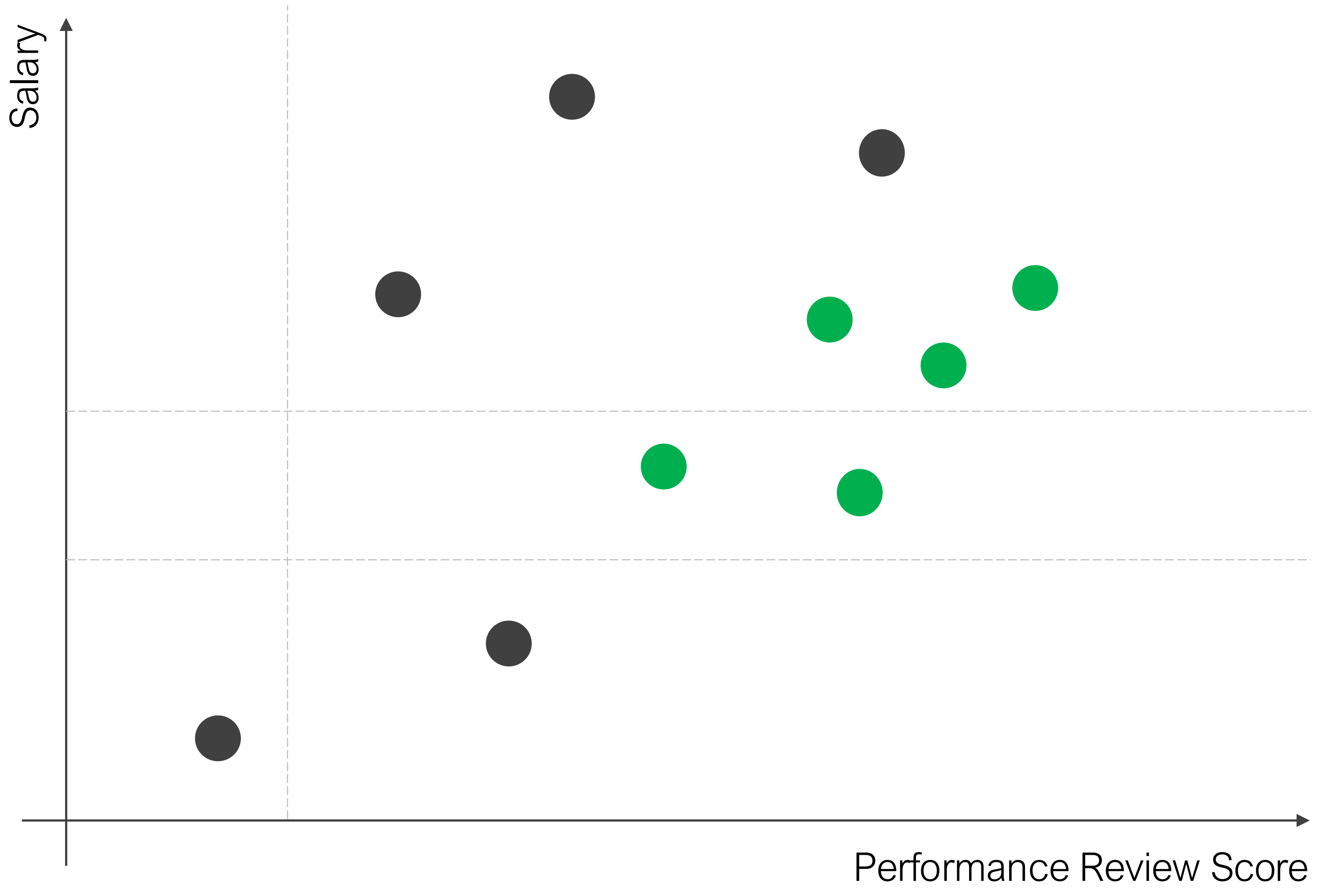
Predicting promotions of salaried employees



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



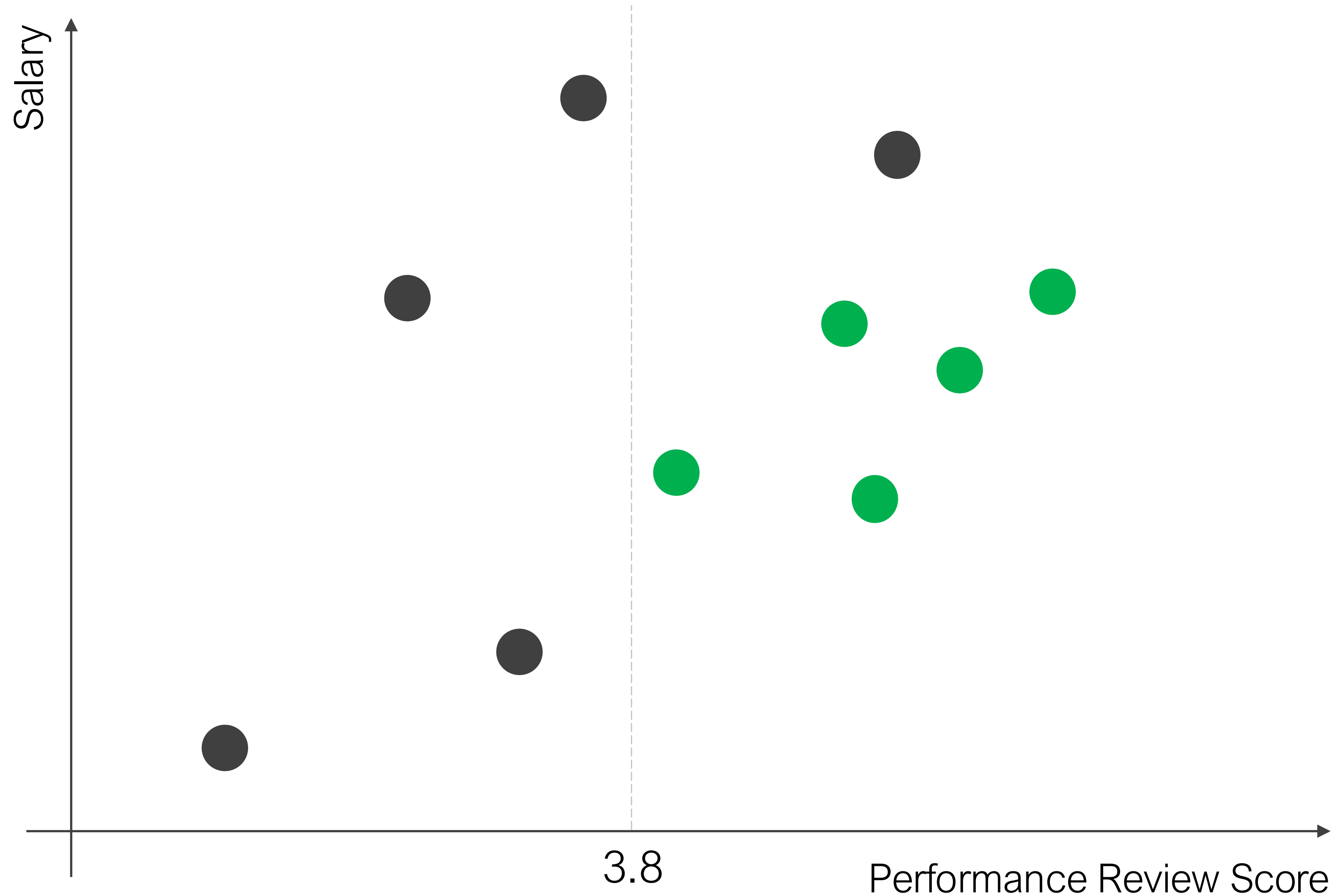
● Promoted

● 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



● Promoted

● Not promoted

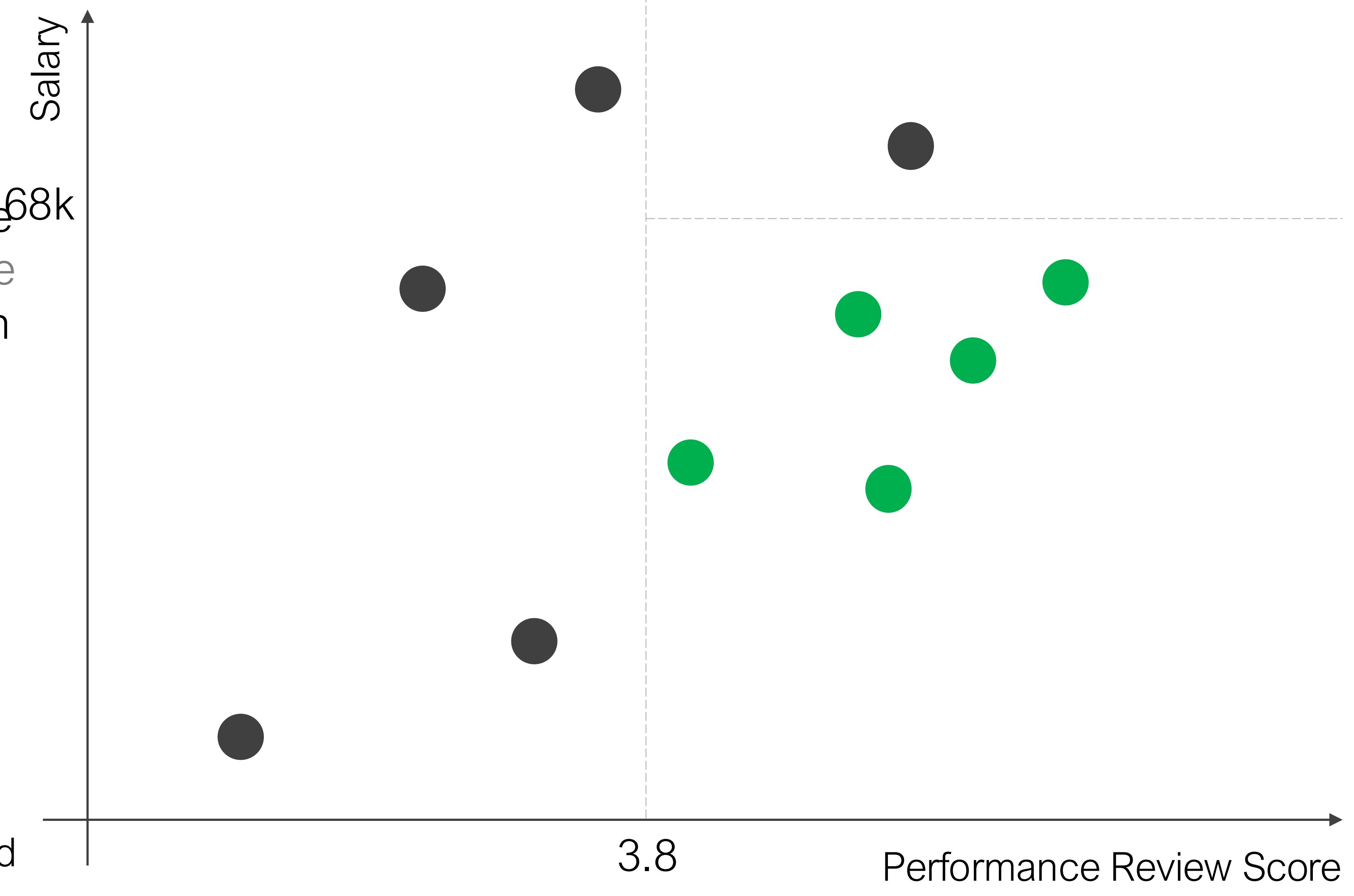
Classification and Regression Trees (CART)

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

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

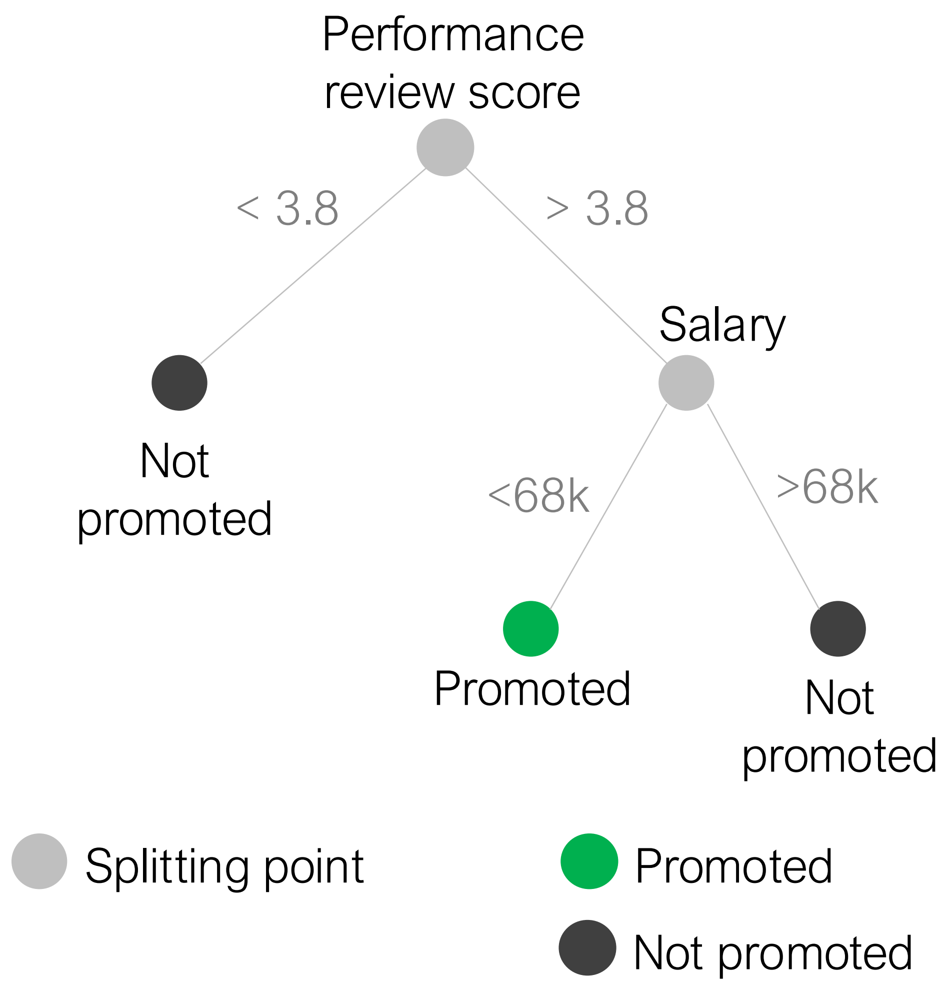
Greedy, recursive binary tree

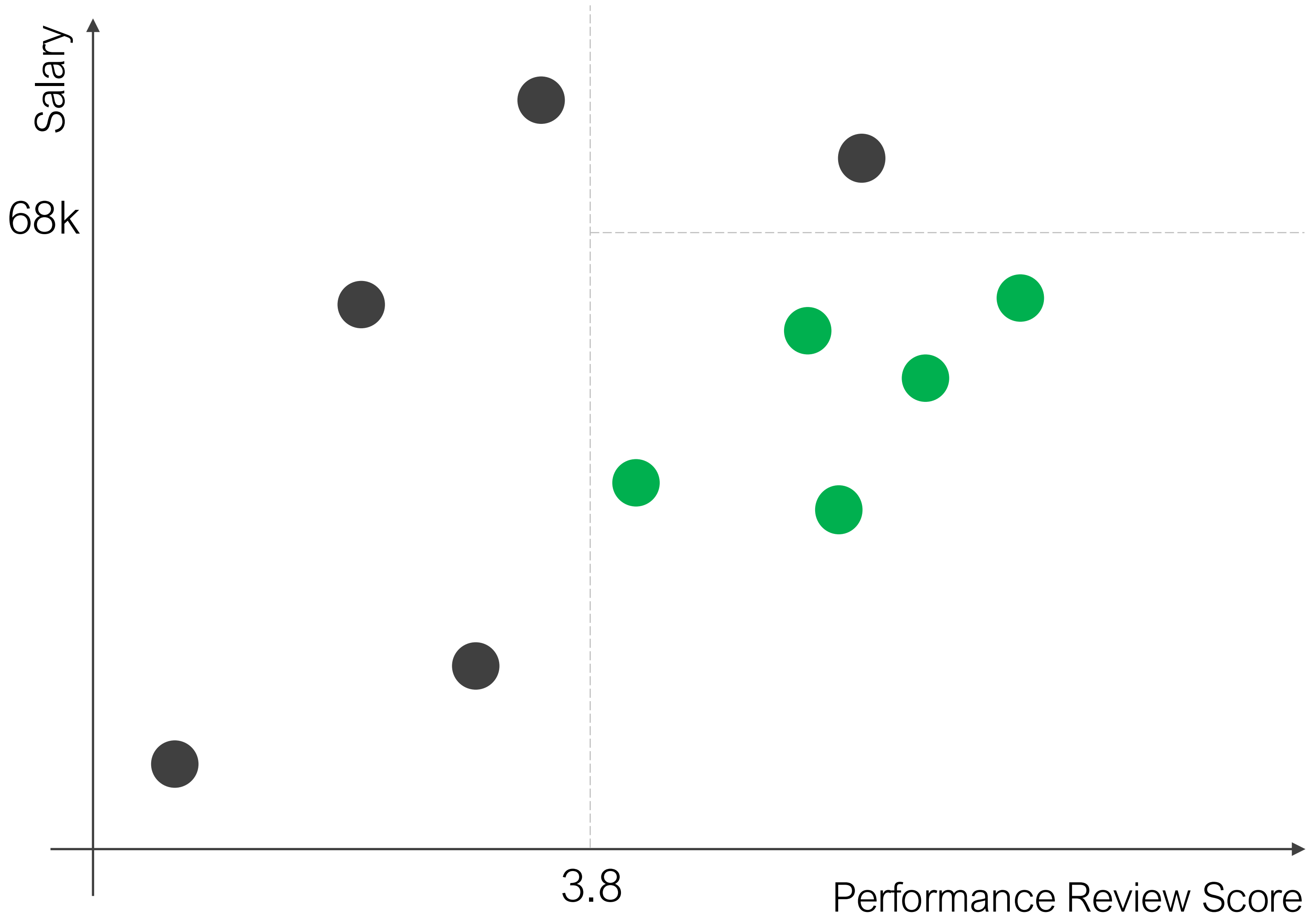


● Promoted
● Not promoted

Classification and Regression Trees (CART)

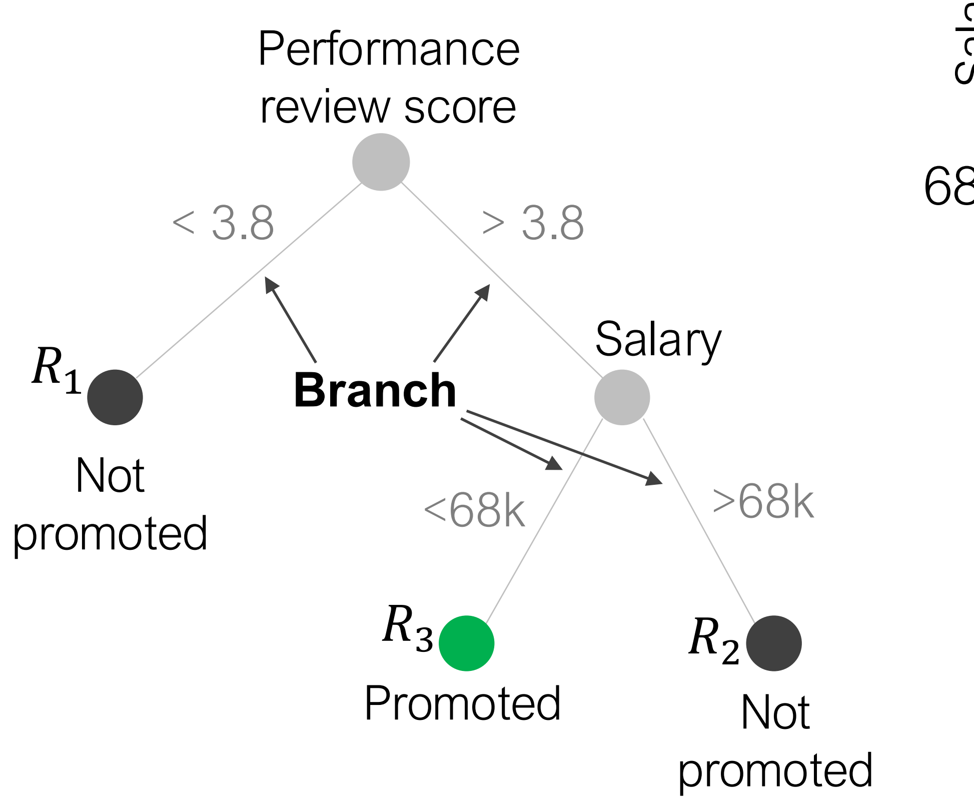
Tree representation:

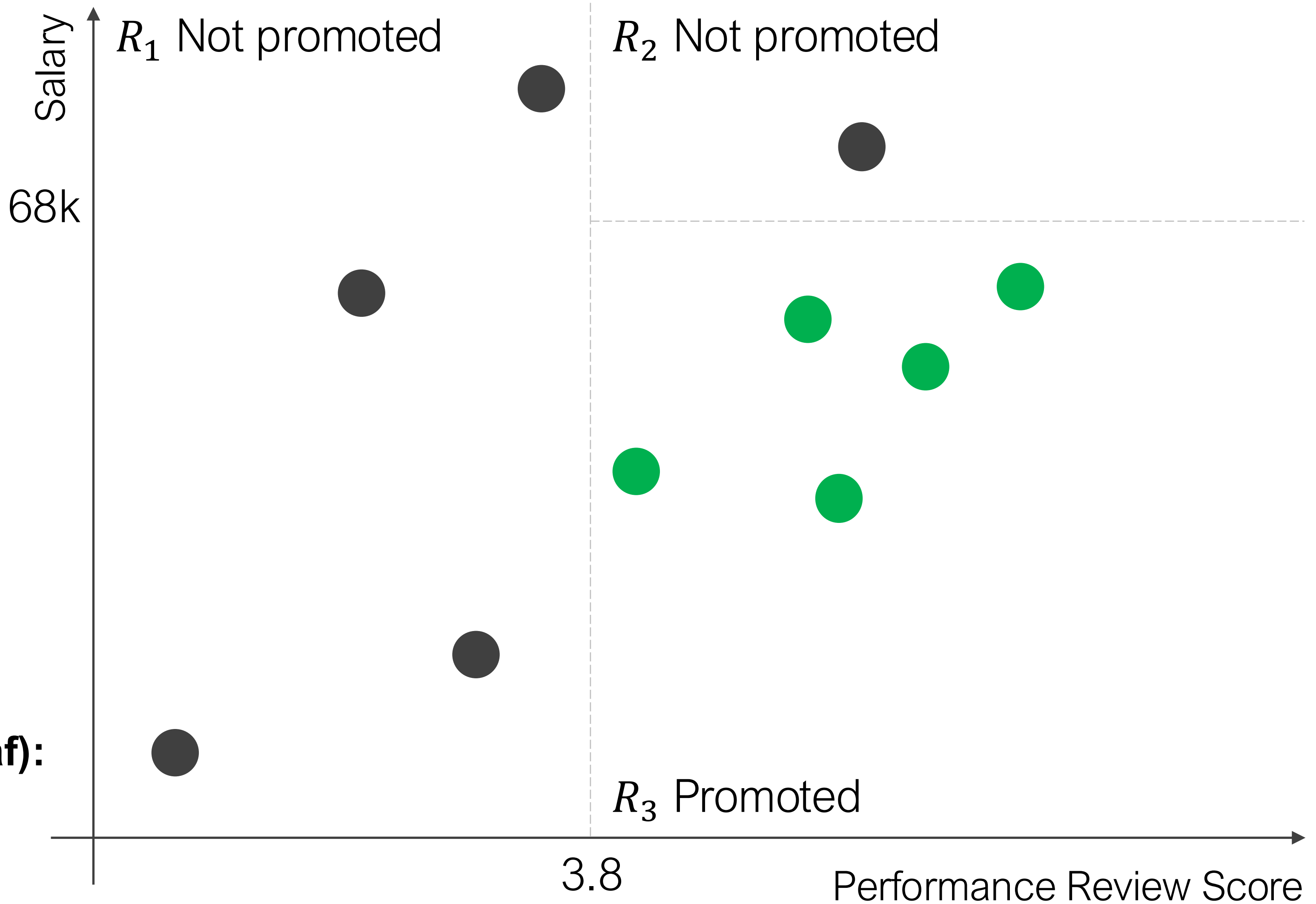




Classification and Regression Trees (CART)

Tree representation:



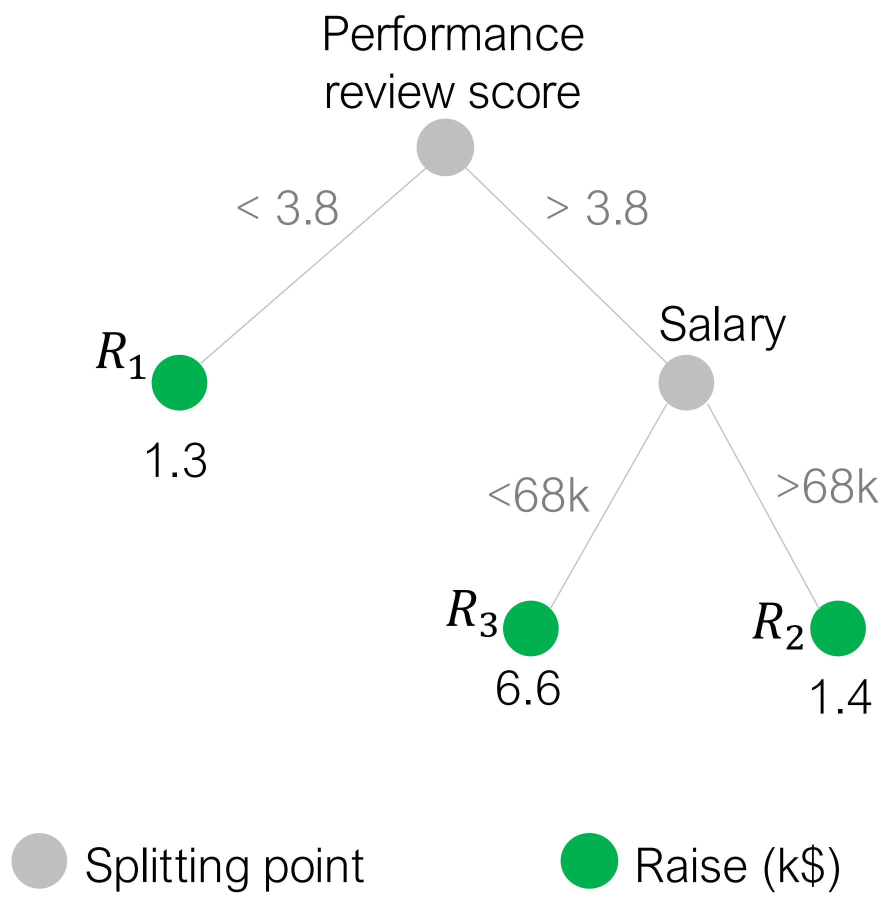


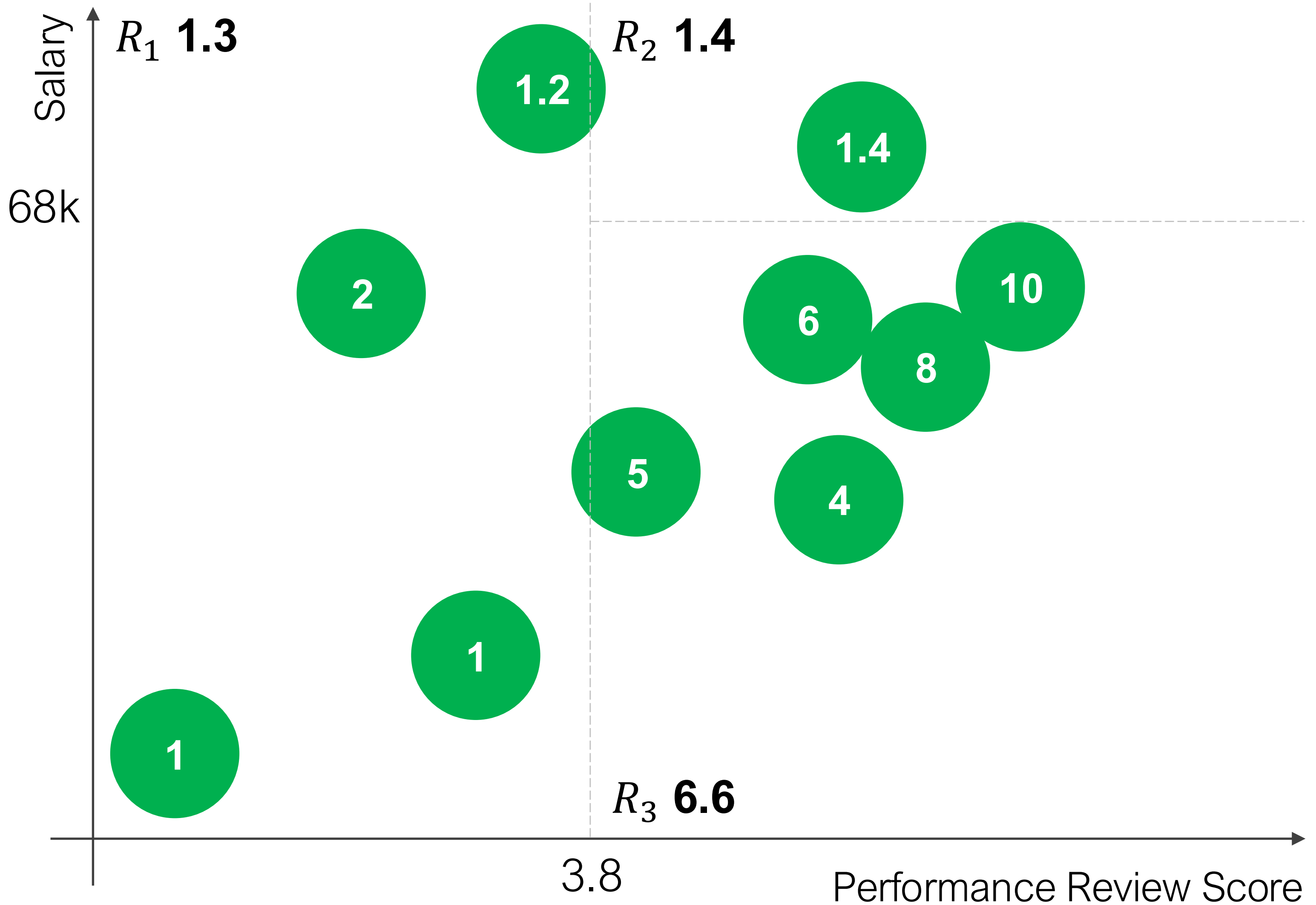
Internal node:
Splitting point

Terminal node (leaf):
● Promoted
● Not promoted

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

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

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

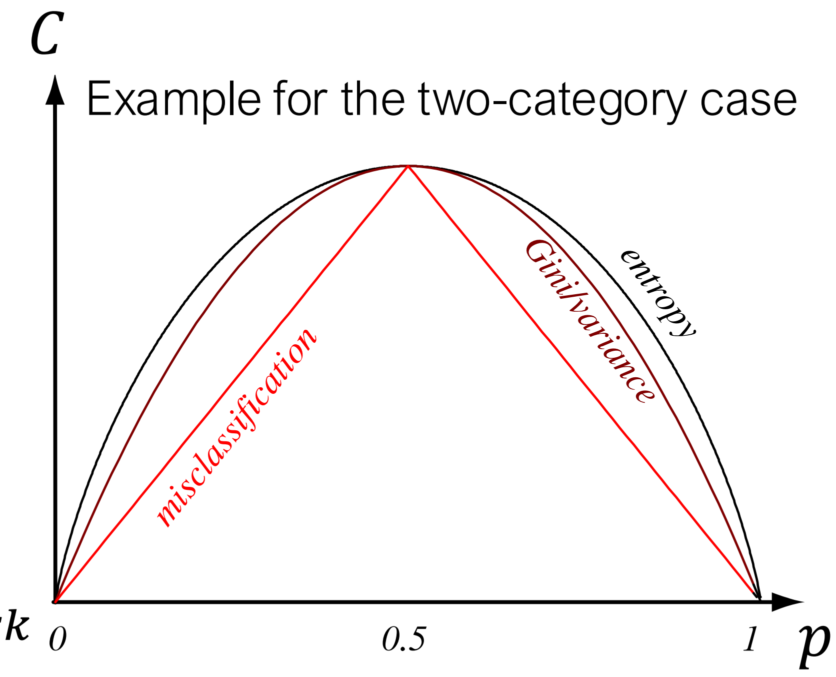
Gini impurity

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

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

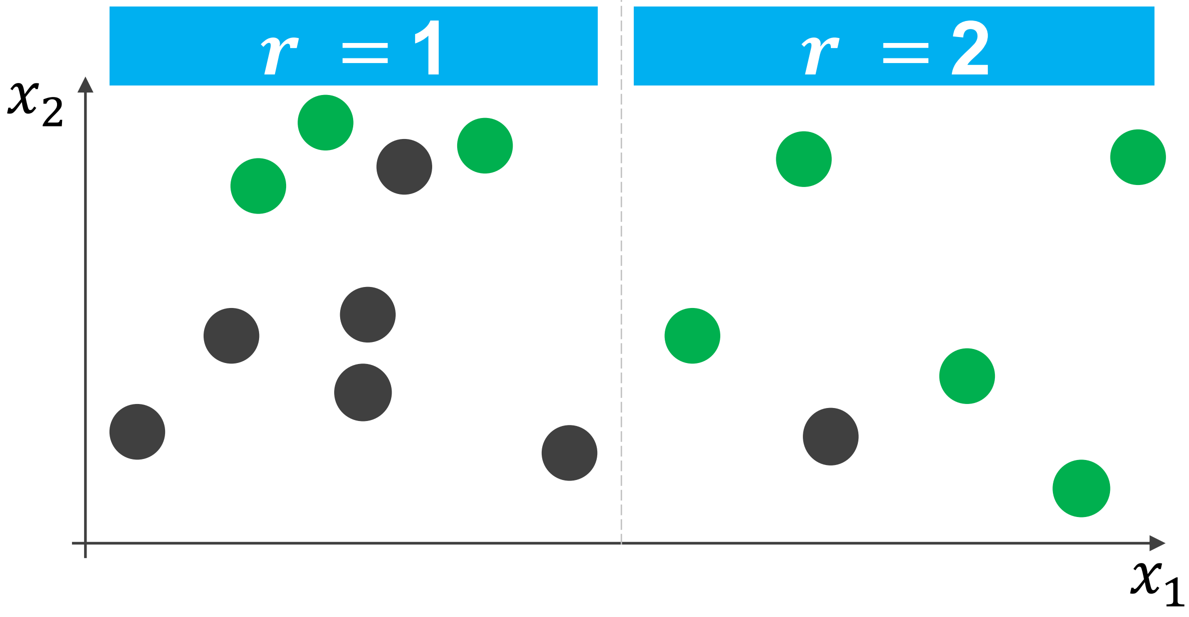


Duda, Hart, and Stork., Pattern Classification

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	⚫



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

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

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

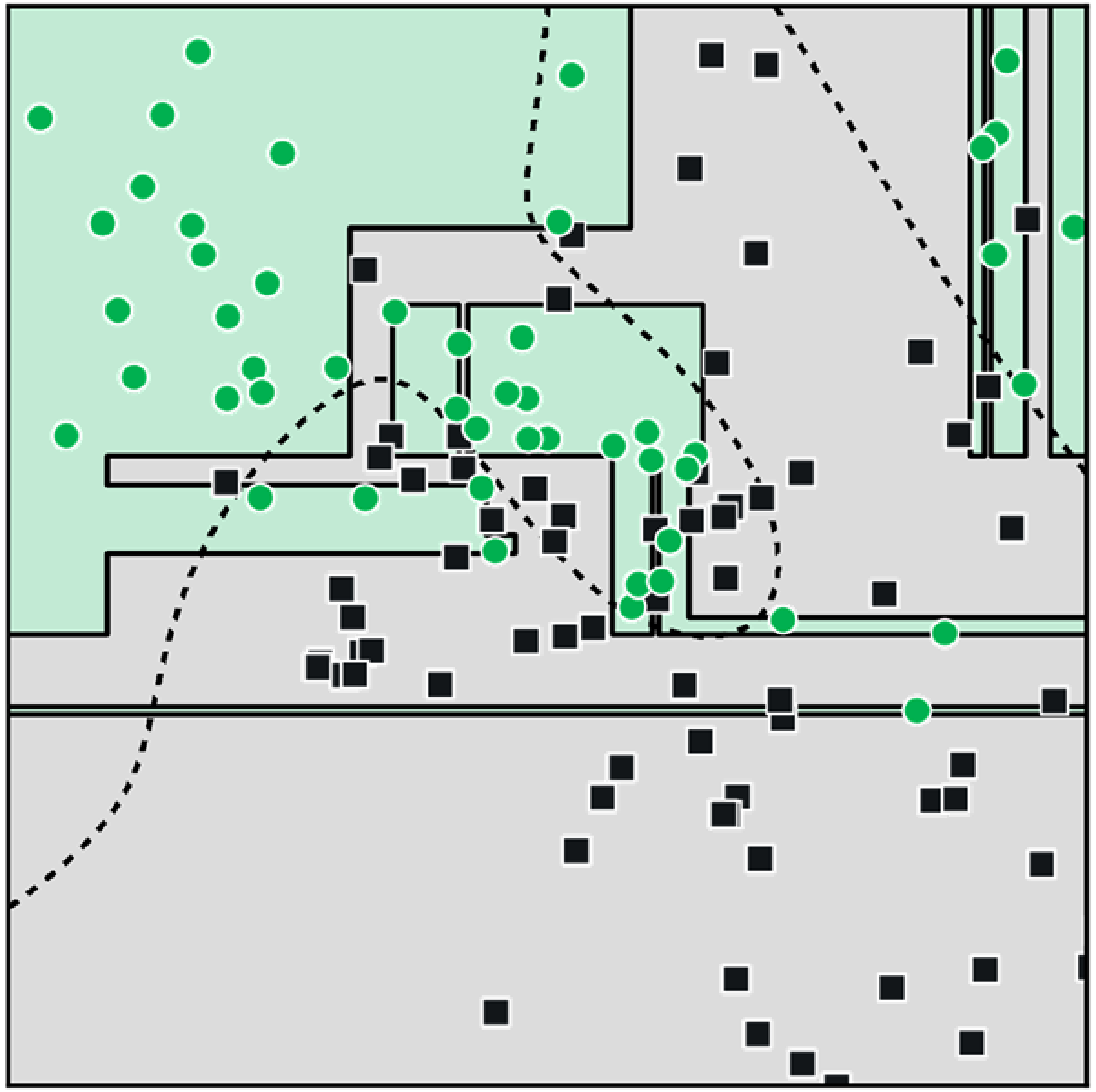
Consider the stopping rule: stop splitting once there is only 1 class of observations in each region (leads to complete overfit)

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 (regression example):

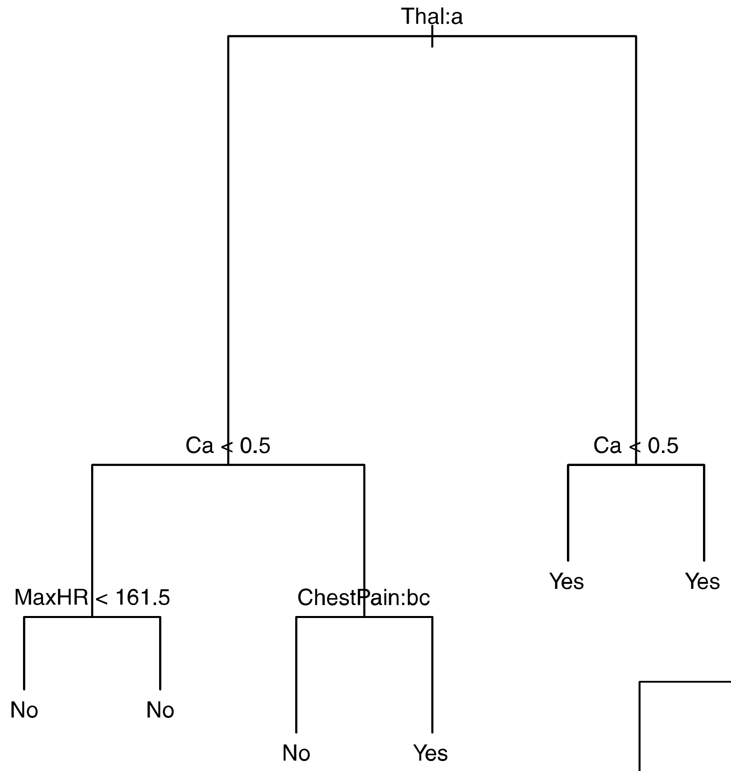
$$C_{Prune} = \sum_{j=1}^{T} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j} \right)^2 + \underset{\text{penalty on number of terminal nodes}}{\alpha} \underset{\text{number of terminal nodes}}{T}$$

Decision Tree



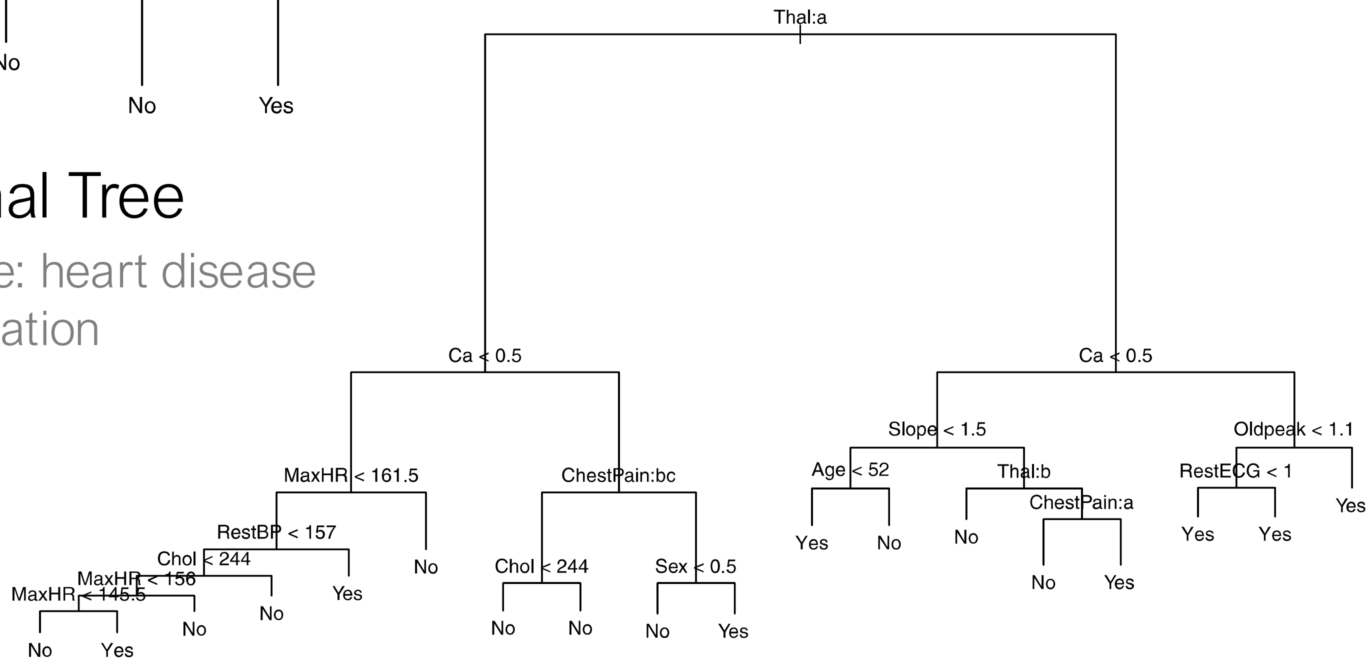
Pruning example

Pruned Tree

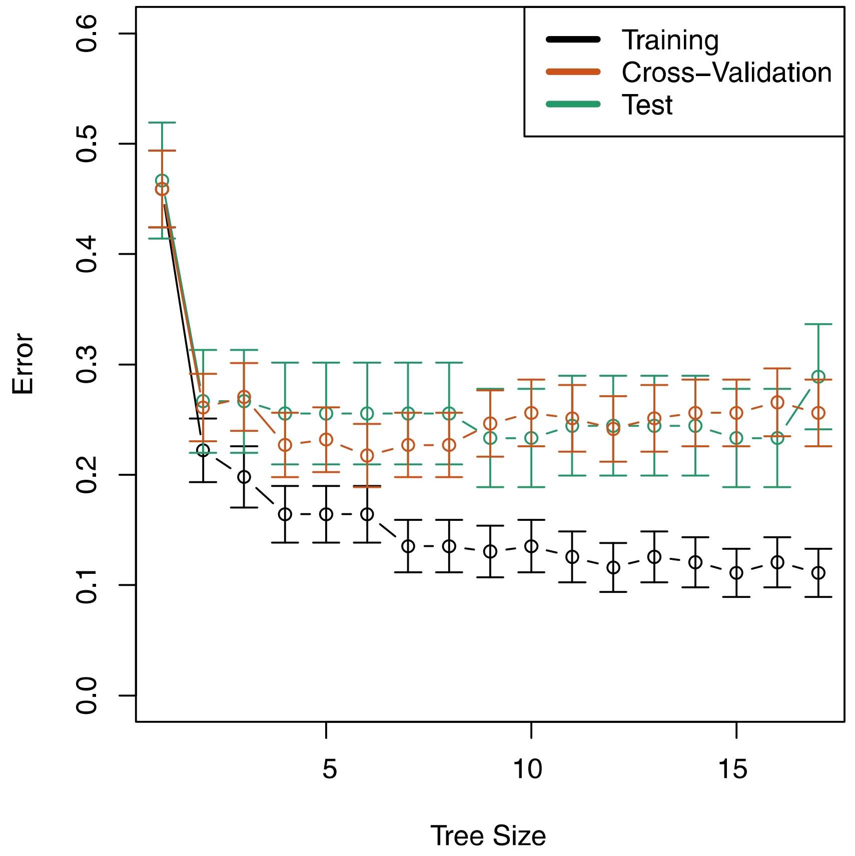


Original Tree

Example: heart disease classification

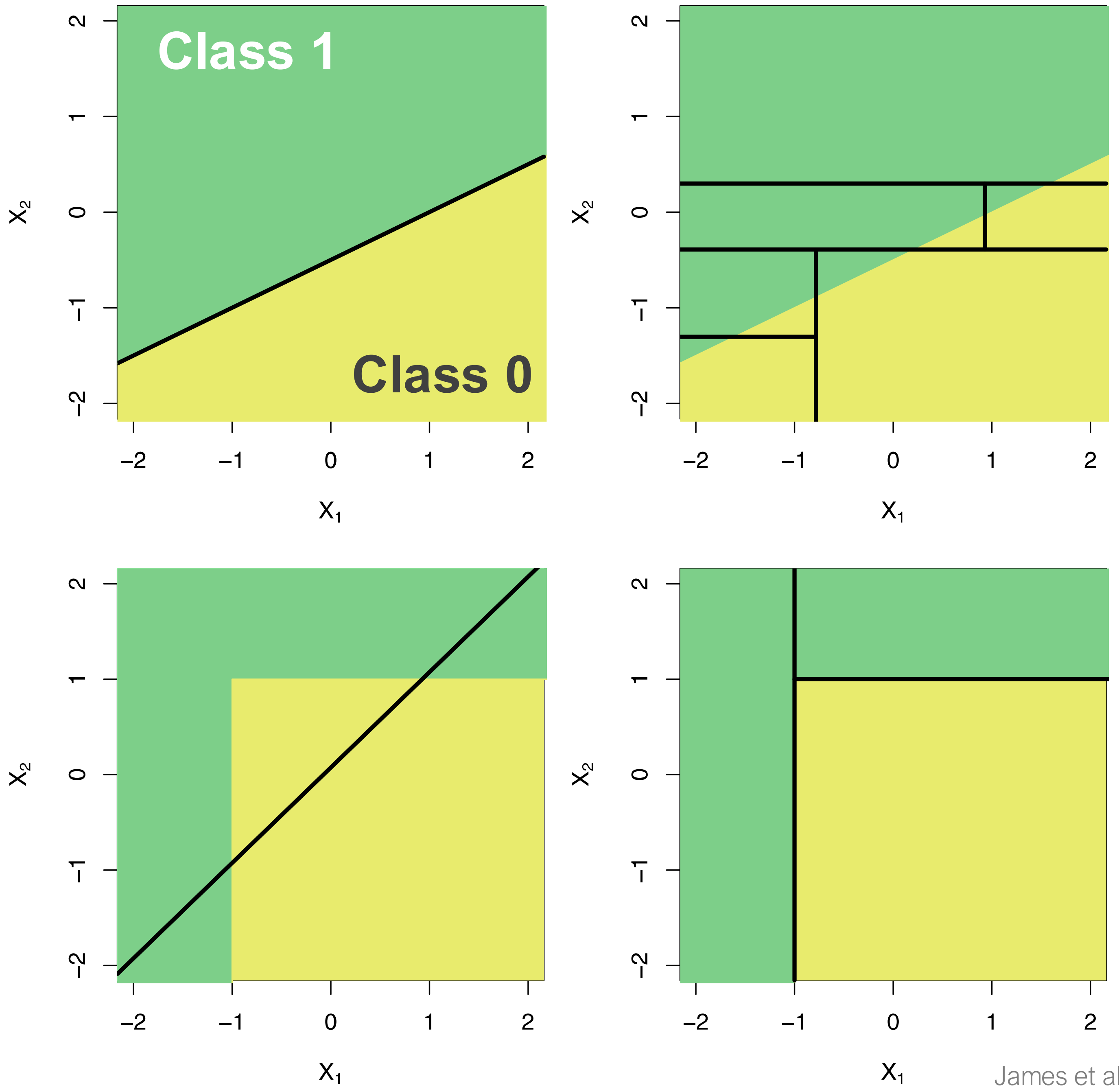


Performance



James et al., An Introduction to Statistical Learning

Linear model



Classification Tree

Struggle when the boundary is not parallel to an axis

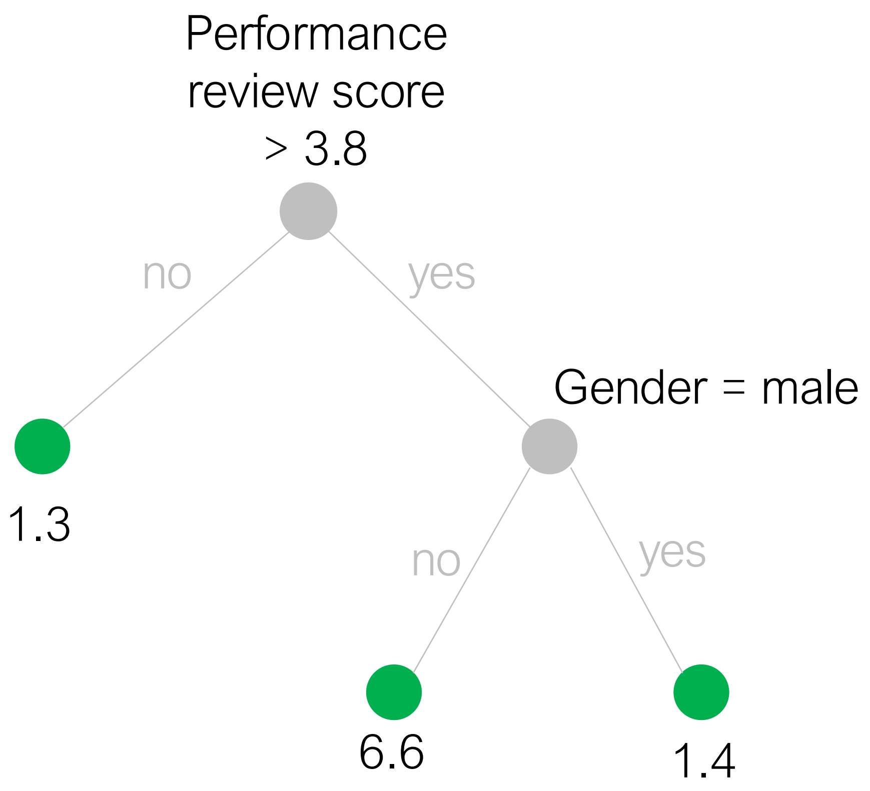
…nonlinear feature transforms could help…

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 other methods

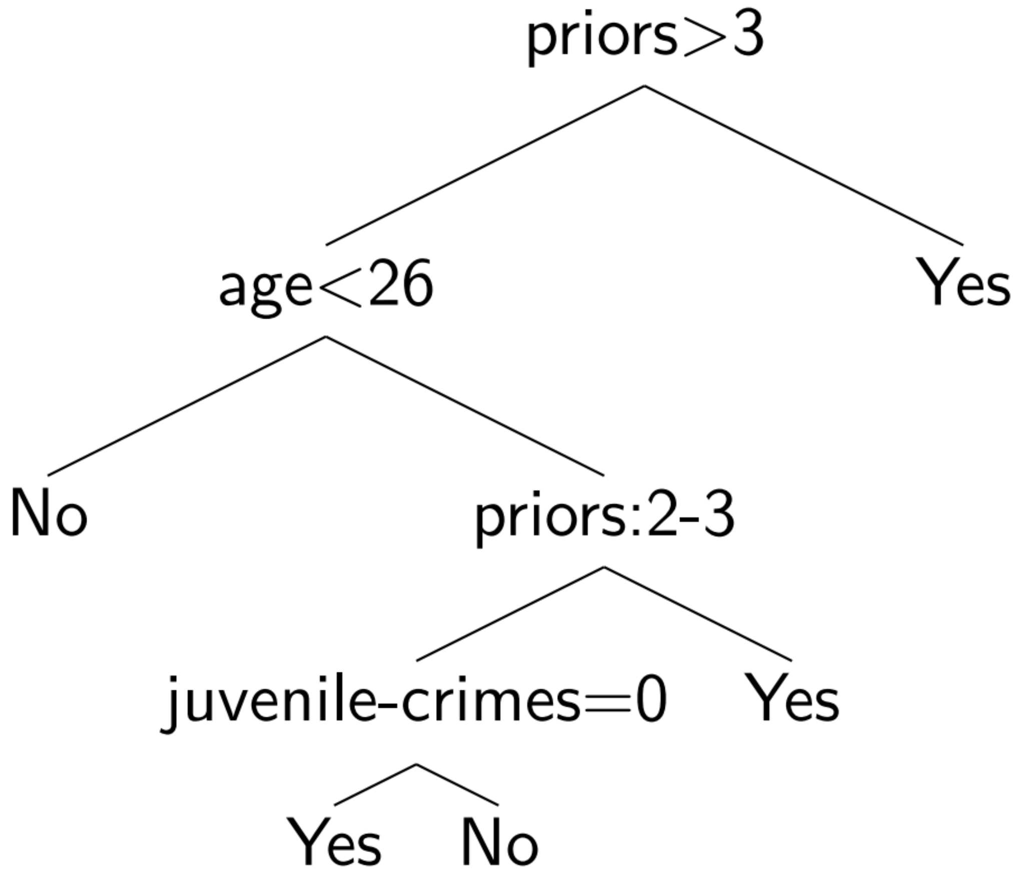
Tend to overfit (have high variance)

Optimal Sparse Decision Trees

Searches the entire space of possible tree structures to find the **global optimum** given constraints
(as opposed to CART's greedy search)

Sparse and **highly interpretable**

Can be computationally expensive



Hu, X., Rudin, C. and Seltzer, M., 2019. Optimal sparse decision trees. *Advances in neural information processing systems*, *32*.

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

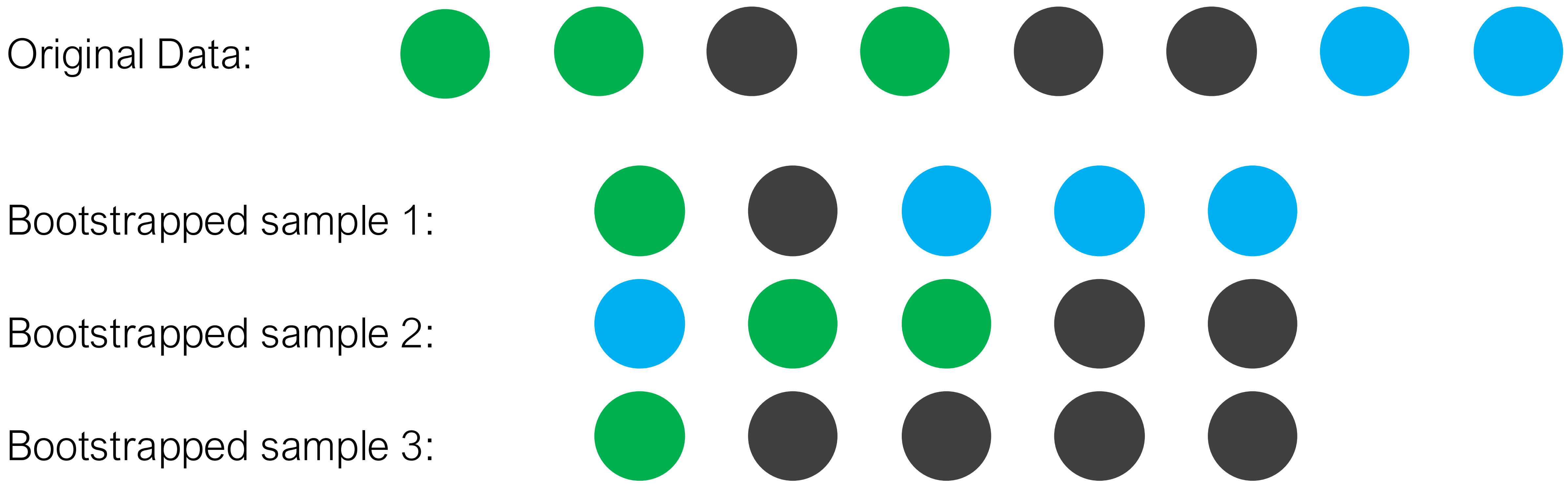
	Bagging	**Boosting**
Models in ensemble:	high variance, low bias (i.e. overfit models)	high bias, low variance (i.e. underfit models, “weak learners”)
Effect of aggregating:	Reduce variance through averaging output	Reduce bias through sequentially fitting models to previous model errors

Bagging

Bootstrap aggregation

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

Recall bootstrap sampling (sampling with replacement):



Bagging

Decision trees are a popular base model

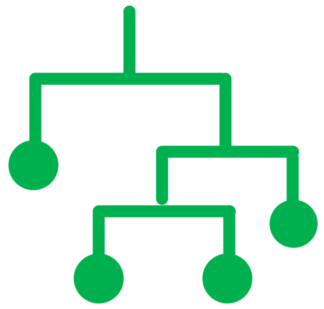
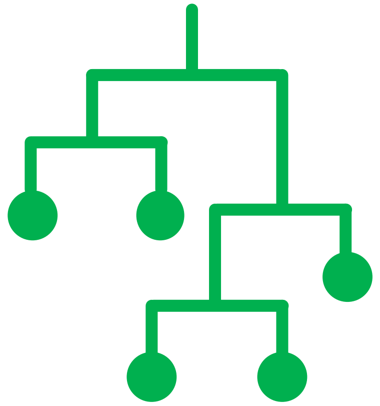
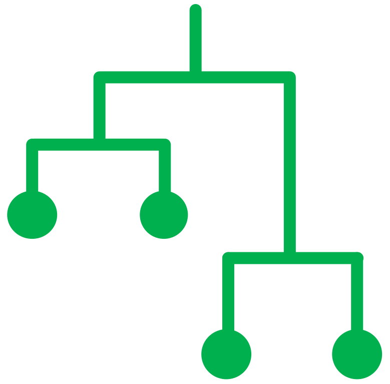
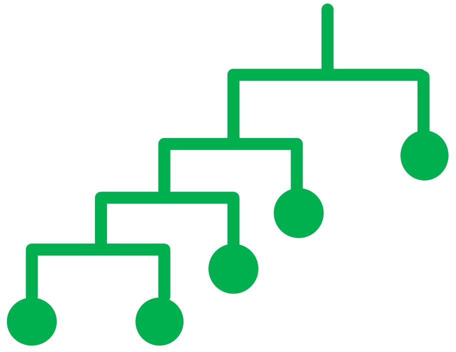
Bootstrap aggregation

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:				

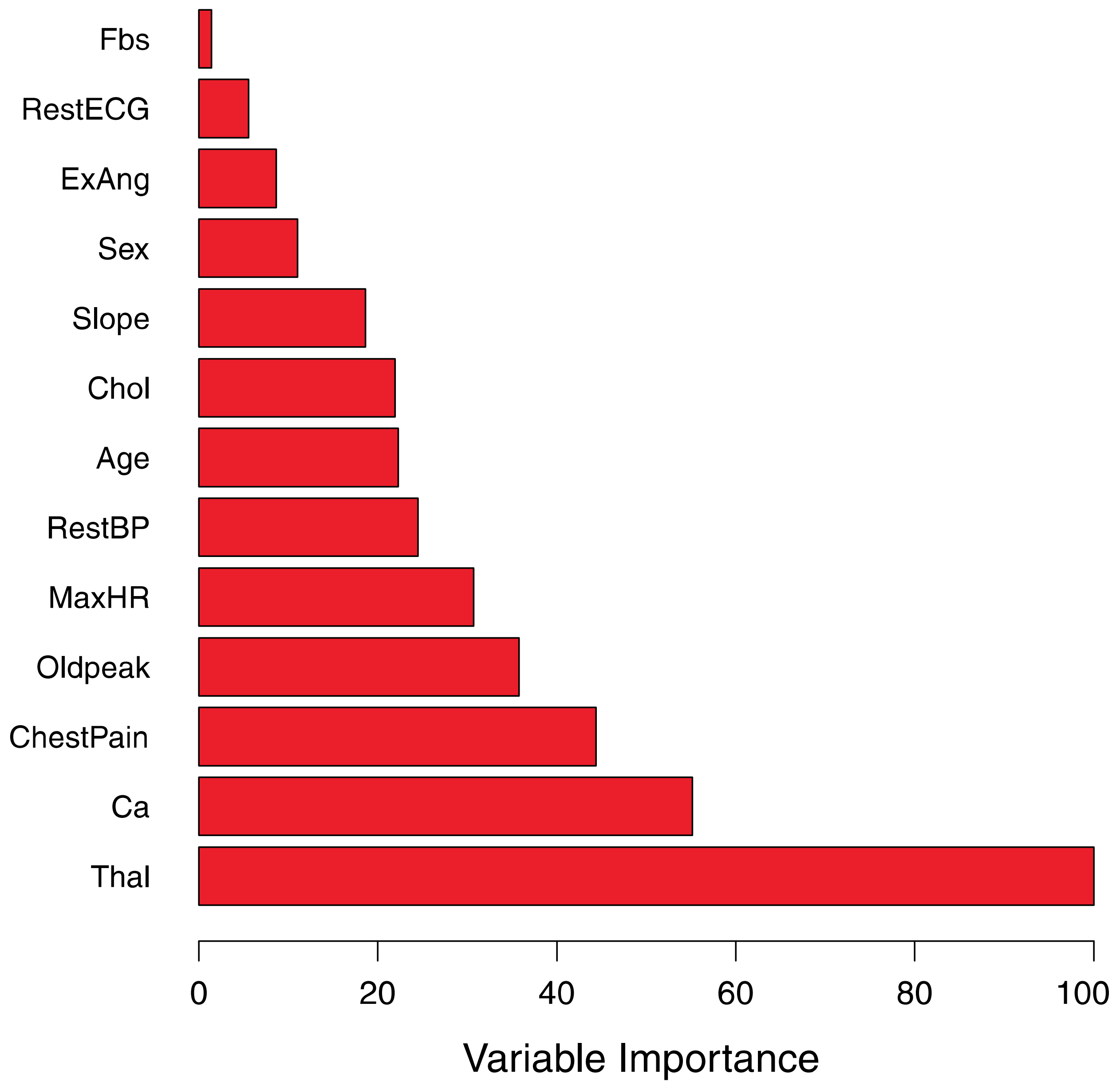
Variable Importance

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

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

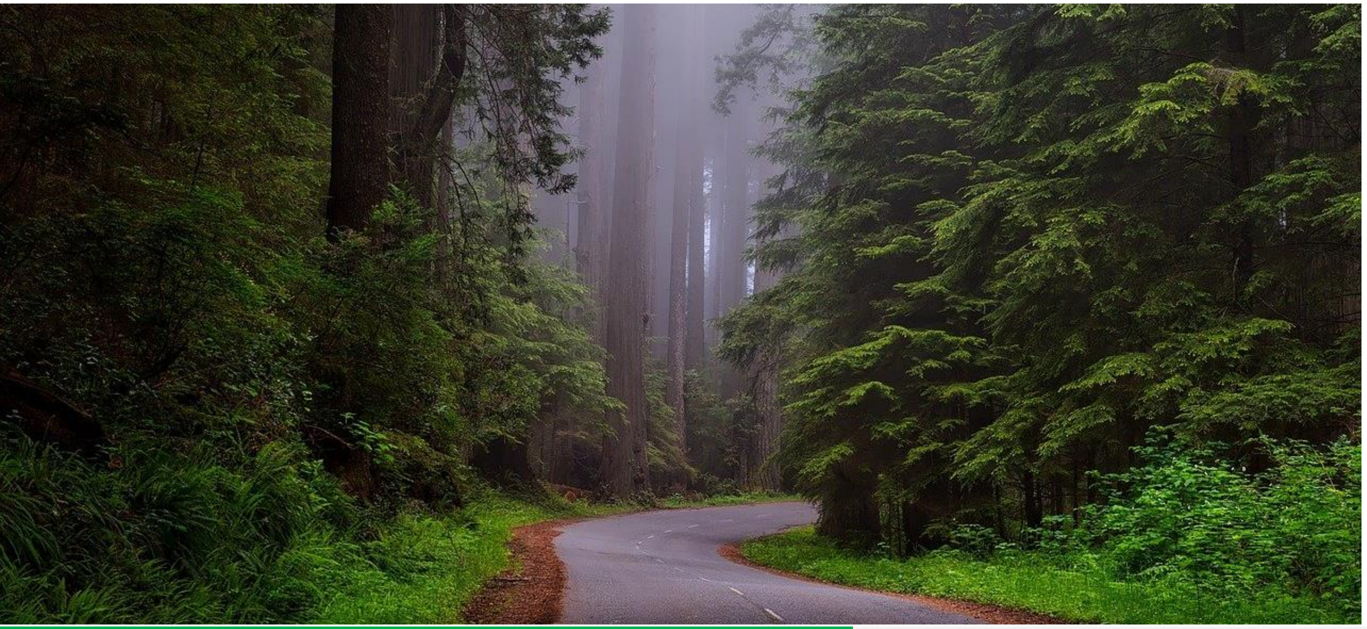
1. Calculate the total amount the error (or impurity) decreased by splitting on each feature.

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.

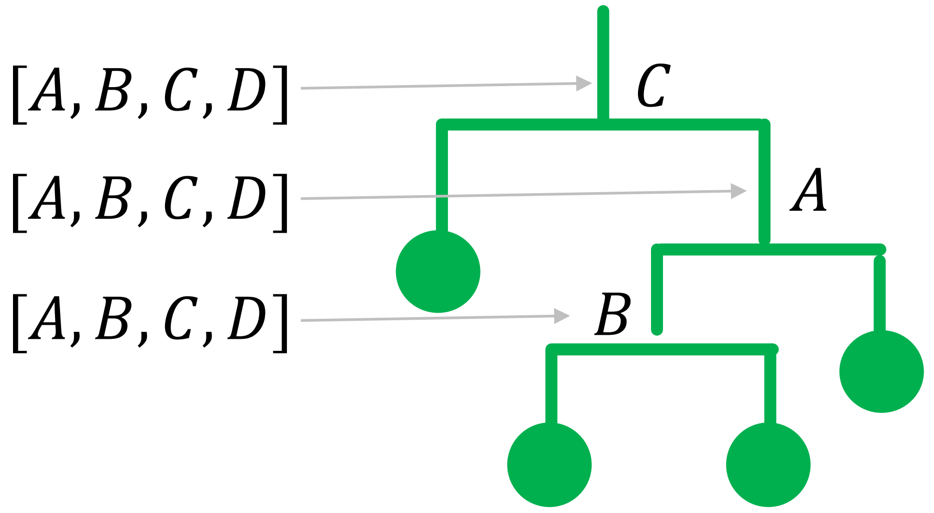
Each time a split is considered, a **random subset of m features** is selected as candidates from the full set of p features

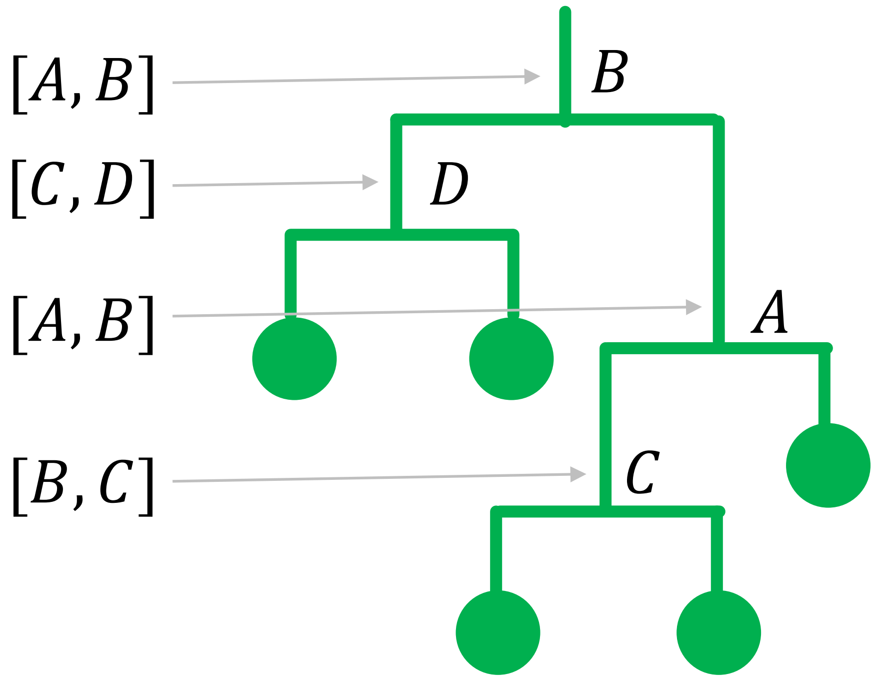
Typically chose: $m = \sqrt{p}$

(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:	$[A, B, C, D]$	$[A, B, C, D]$

eature options for each split:

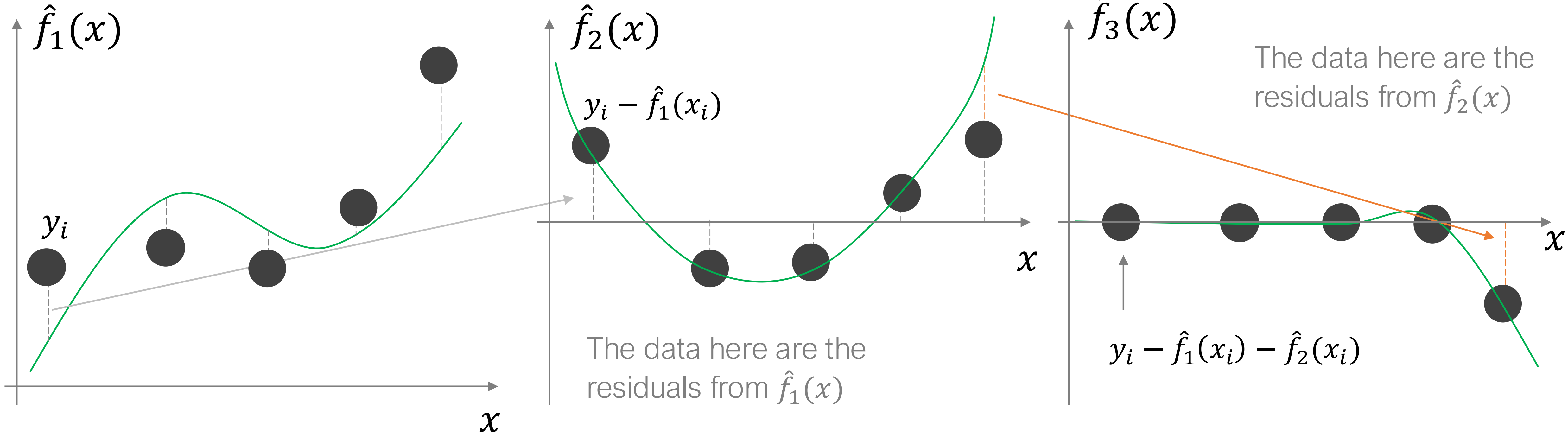


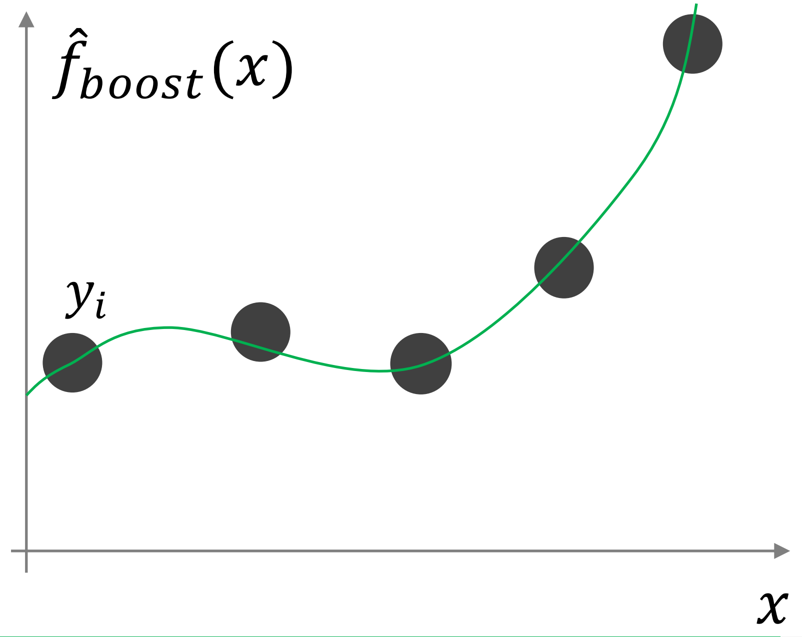


Boosting

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





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
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)
4. Update $\hat{f}(\boldsymbol{x}) \leftarrow \hat{f}(\boldsymbol{x}) + \lambda \hat{f}_b(\boldsymbol{x})$
5. Update the residuals $r_i \leftarrow r_i - \lambda \hat{f}_b(\boldsymbol{x}_i)$
6. Output the boosted model: $\hat{f}(\boldsymbol{x}) = \sum_{b=1}^{B} \lambda \hat{f}_b(\boldsymbol{x})$

Repeat B times

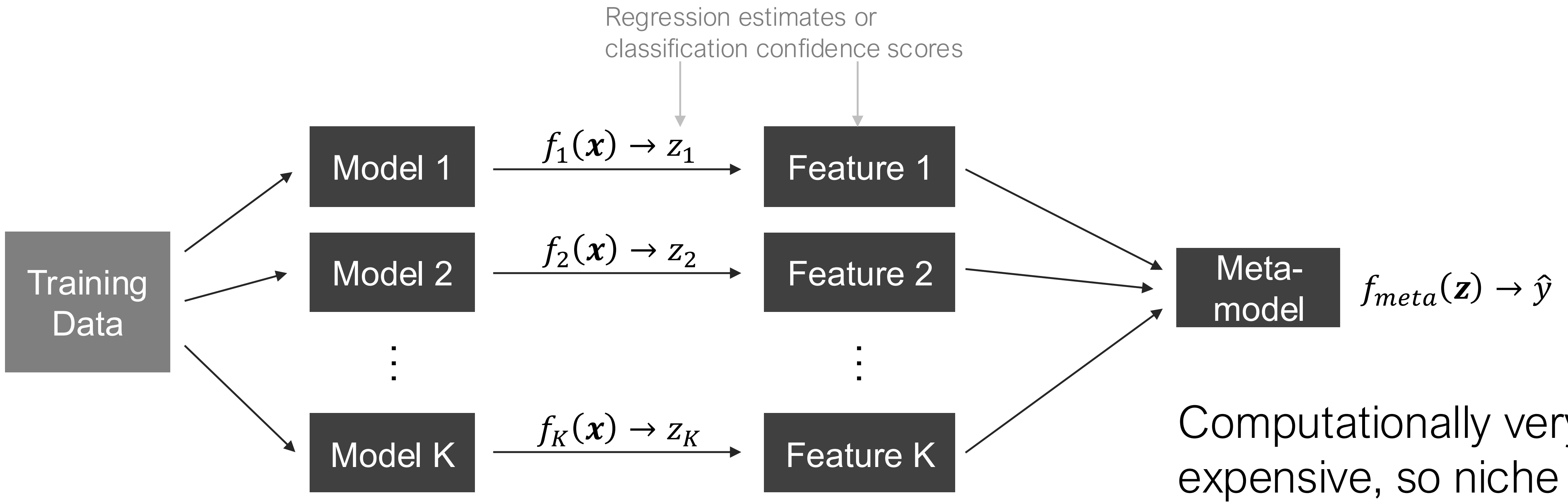
λ slows down the learning process to avoid overfitting

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

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**


Regression estimates or classification confidence scores
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
$f_{meta}(\boldsymbol{z}) \rightarrow \hat{y}$

Computationally very expensive, so niche use cases

Supervised Learning Techniques

Covered so far

⚫		Linear Regression
⚫	🟢	K-Nearest Neighbors
	🟢	Logistic Regression
	🟢	Linear/Quadratic Discriminant Analysis
	🟢	Naïve Bayes
⚫	🟢	Decision Trees
⚫	🟢	Random Forests
⚫	🟢	Gradient Boosted Decision Trees

Appropriate for:

🟢 Classification

⚫ Regression

Ensemble approaches, including bagging, boosting, and stacking, can be used with numerous machine learning techniques, often CART