Data Mining Techniques

CS 6220 - Section 3 - Fall 2016

Lecture 7: Classification 4

Jan-Willem van de Meent
(*credit*: Yijun Zhao, Carla Brodley, Cheng Li, Hastie)

Homework and Project

- Project teams are **due today**
- Homework 2 is **out today**
- Homework 1 is **due on Friday**
  - *Simplification for exercise 2b: You may use expressions derived in Bishop 2.3.1:*

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \quad (2.113)$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}) \quad (2.114)$$

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\boldsymbol{\Lambda}^{-1}\mathbf{A}^{\mathrm{T}}) \quad (2.115)$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\Sigma}\{\mathbf{A}^{\mathrm{T}}\mathbf{L}(\mathbf{y} - \mathbf{b}) + \boldsymbol{\Lambda}\boldsymbol{\mu}\}, \boldsymbol{\Sigma}) \quad (2.116)$$

Decision Trees

Example: Play Golf?

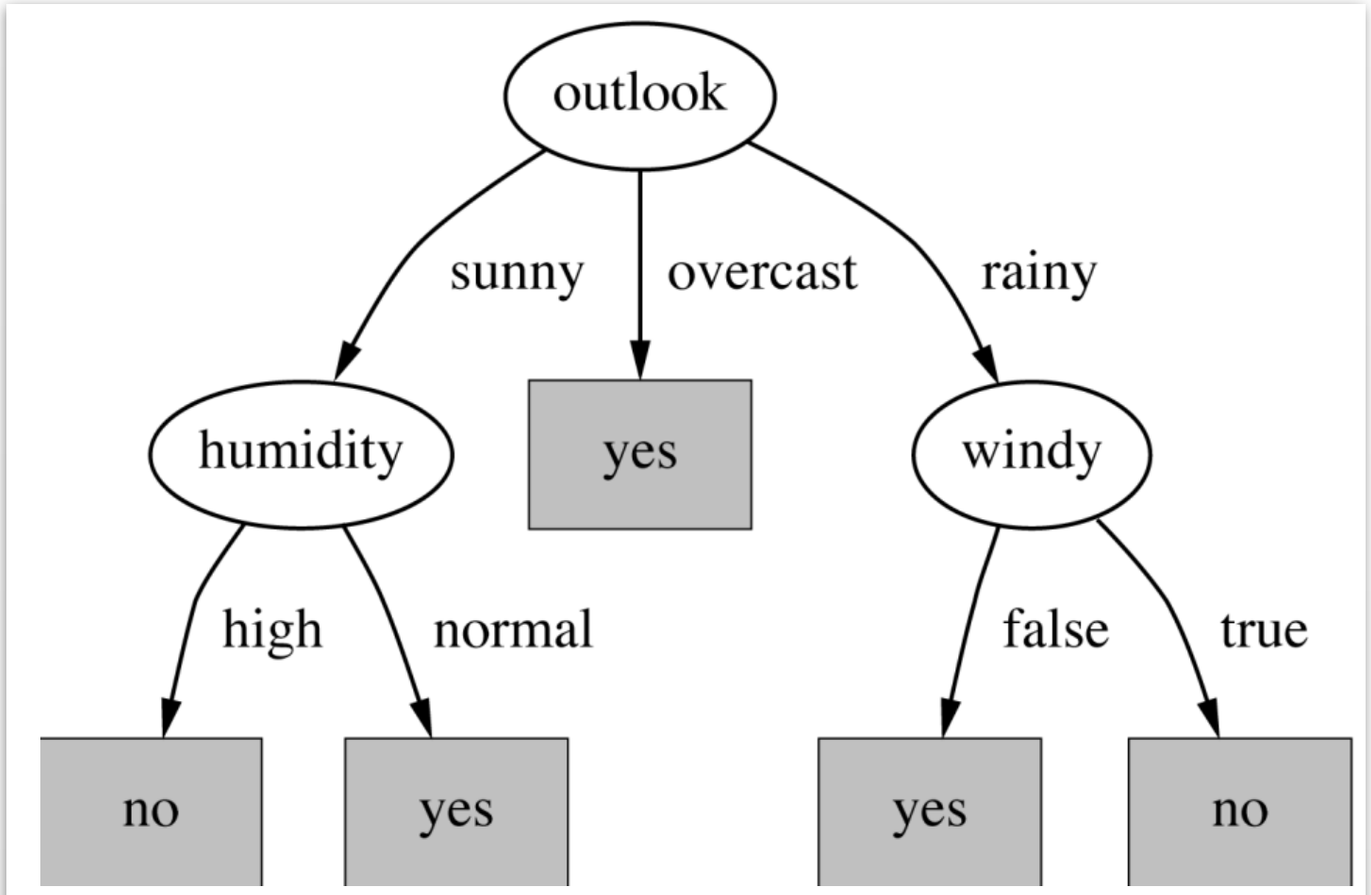
Outlook	Temp	Humidity	Windy	Play
Sunny	Hot	High	False	No
Sunny	Hot	High	True	No
Overcast	Hot	High	False	Yes
Rainy	Mild	High	False	Yes
Rainy	Cool	Normal	False	Yes
Rainy	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Sunny	Mild	High	False	No
Sunny	Cool	Normal	False	Yes
Rainy	Mild	Normal	False	Yes
Sunny	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Rainy	Mild	High	True	No

- Two classes
  - Yes
  - No

- Attributes
  - Outlook
  - Temperature
  - Humidity
  - Windy

Decision Tree

a.k.a. Classification and Regression Tree (CART)

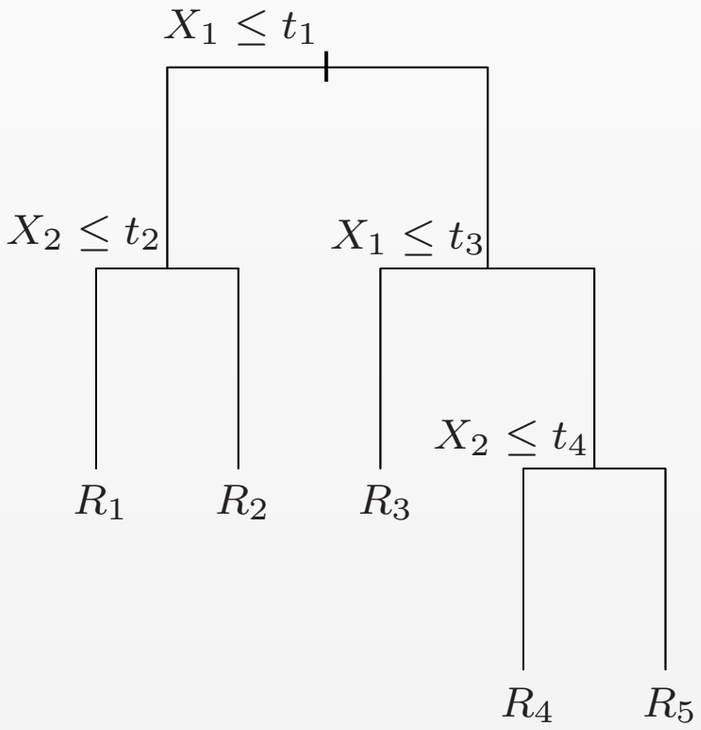


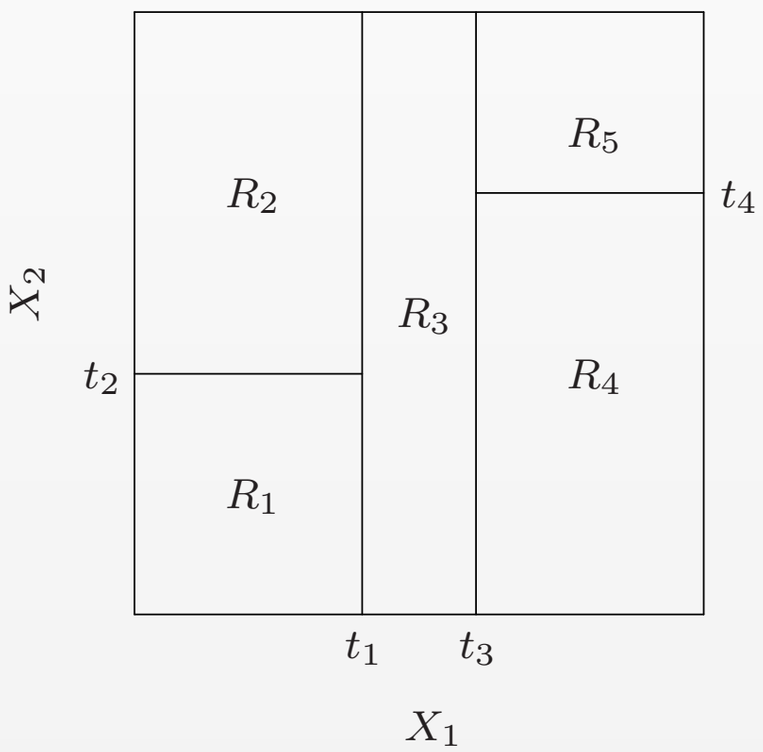
Which class?

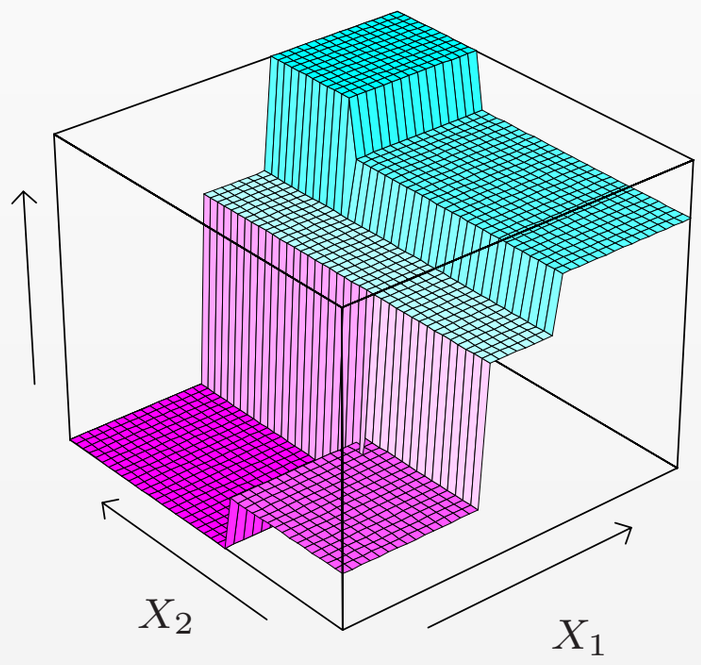
outlook=*sunny*, temp=*hot*, humidity=*high*, wind=*strong*

Decision Tree

a.k.a. Classification and Regression Tree (CART)







Learning a Binary Tree

Recursive Algorithm (Discrete Features)

- Create **root** node
- ***If*** all examples are **1** or **0**,
  - ***Return* root** with **label = 1** or **label = 0**
- ***If*** attributes are empty,
  - ***Return* root** with **label = [*majority*]**
- ***Else begin***
  - Split on examples on feature ***d*** that ***best**** predicts class
  - ***For*** each value ***v*** for feature ***d***
    - Create a **branch** with test $v = x_d$
    - ***If*** set of examples with $v = x_d$ is empty, set **label = [*majority*]**
    - ***Else*** learn subtree for examples with $v = x_d$

Learning a Binary Tree

Recursive Algorithm (Discrete Features)

- Create **root** node
- ***If*** all examples are **1** or **0**,
  - ***Return* root** with **label = 1** or **label = 0**
- ***If*** attributes are empty,
  - ***Return* root** with **label = [*majority*]**
- ***Else begin***
  - Split on examples on feature ***d*** that ***best**** predicts class
  - ***For*** each value ***v*** for feature ***d***
    - Create a **branch** with test $v = x_d$
    - ***If*** set of examples with $v = x_d$ is empty, set **label = [*majority*]**
    - ***Else*** learn subtree for examples with $v = x_d$

What if features are real-valued?

Learning a Binary Tree

Recursive Algorithm (Continuous Features)

- Create **root** node
- ***If*** all **examples** are **1** or **0**,
  - ***Return*** **root** with **label = 1** or **label = 0**
- ***If*** attributes are empty,
  - ***Return*** **root** with **label = [*mean*]** of **examples**
- ***Else begin***
  - Find split $v <= x_d$ that ***best**** predicts class
  - ***For*** **subset** of **examples** matching $v <= x_d$ and $v > x_d$
    - Create a **branch** corresponding to **subset**
    - ***If*** set **subset** is empty, set **label = [*mean*]** of **examples**
    - ***Else*** learn subtree for **subset**

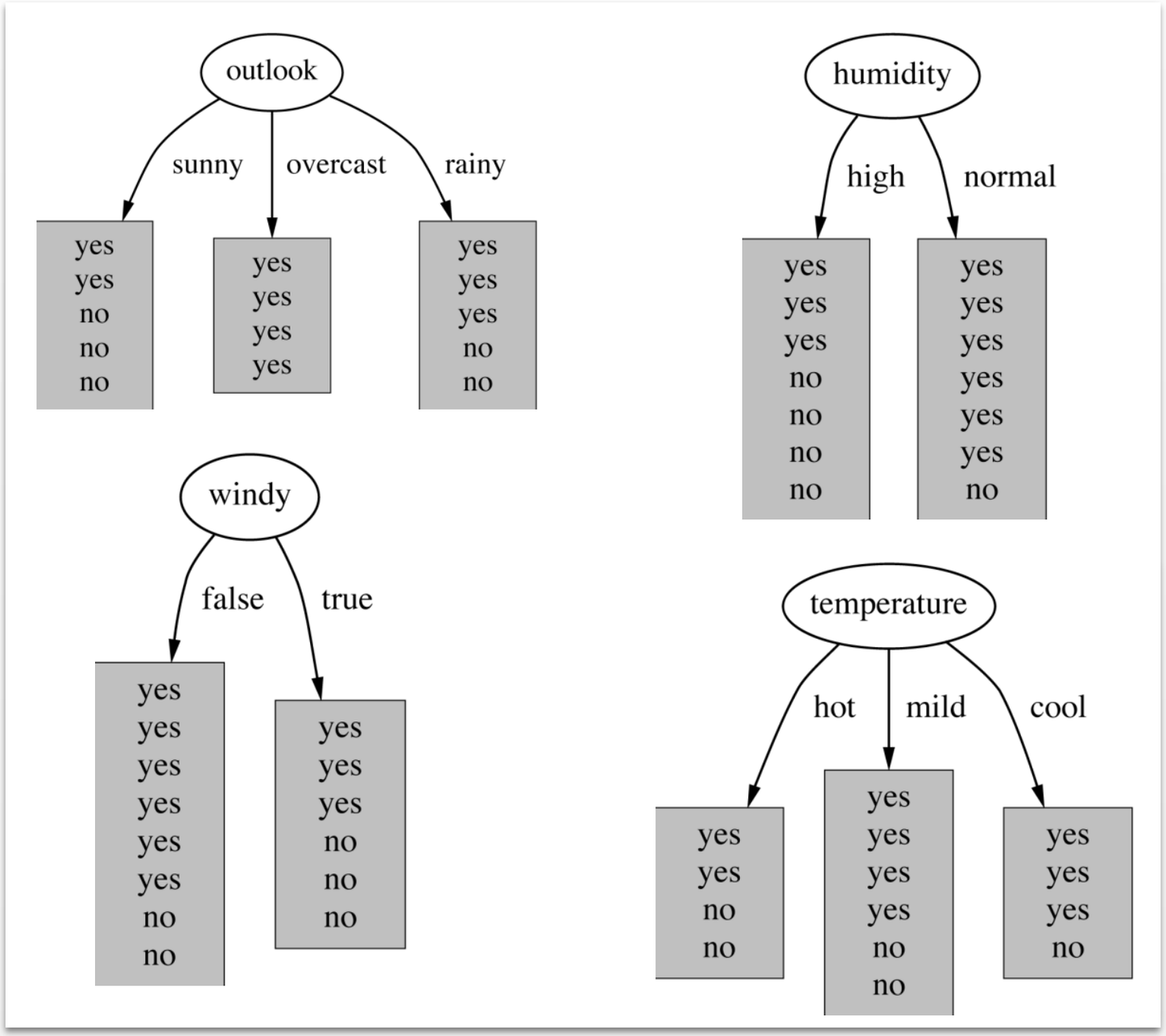
Learning a Binary Tree

Recursive Algorithm (Continuous Features)

- Create **root** node
- ***If*** all **examples** are **1** or **0**,
  - ***Return*** **root** with **label = 1** or **label = 0**
- ***If*** attributes are empty,
  - ***Return*** **root** with **label = [*mean*]**
- ***Else begin***
  - Find split $v <= x_d$ that ***best**** predicts class
  - ***For*** **subset** of **examples** matching $v <= x_d$ and $v > x_d$
    - Create a **branch** corresponding to **subset**
    - ***If*** set **subset** is empty, set **label = [*mean*]** of **examples**
    - ***Else*** learn subtree for **subset**

How can we find the best split?

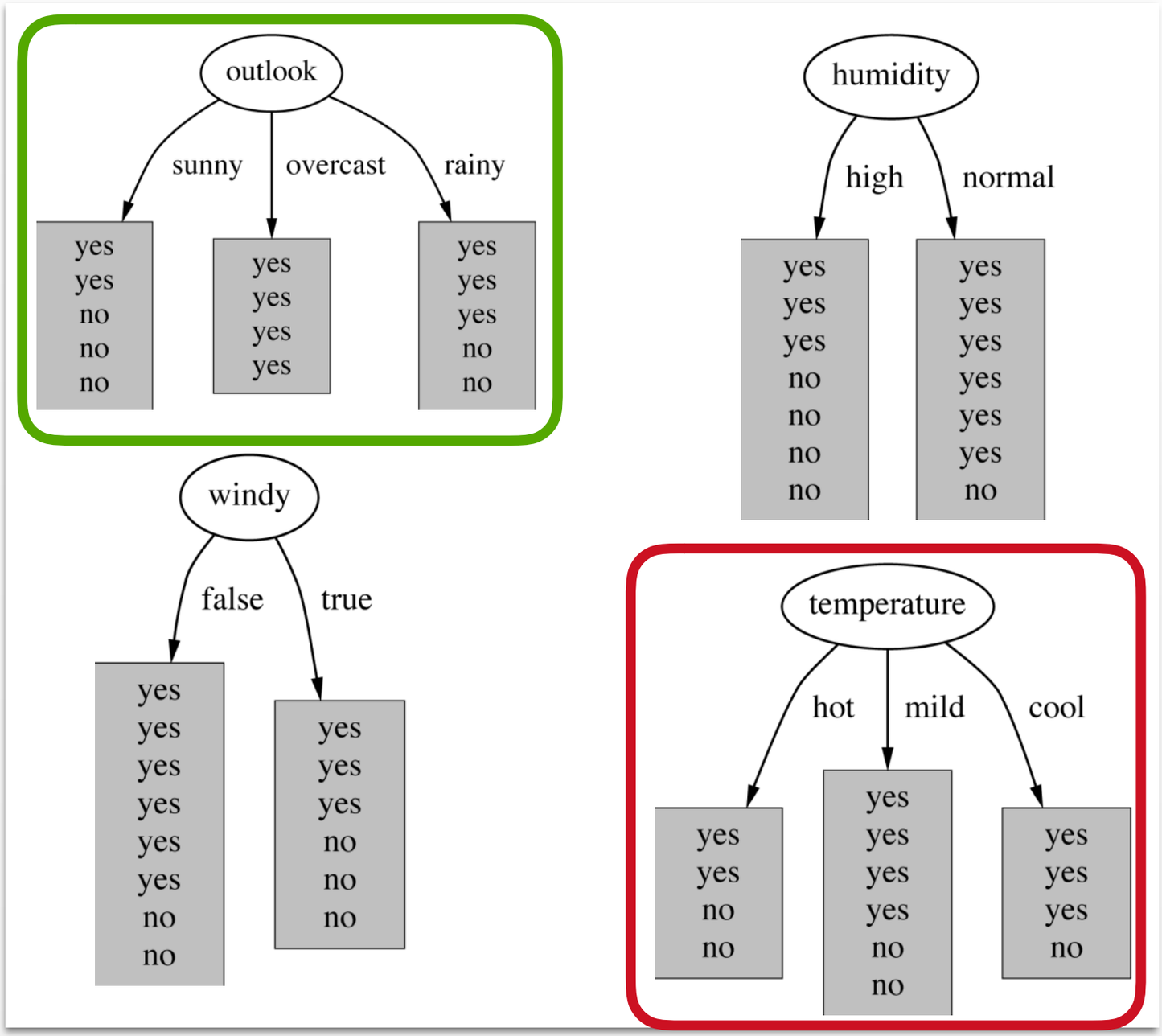
Choosing the best split



Intuition

- What is a ***good*** split?
- What is a ***bad*** split?

Choosing the best split



Intuition

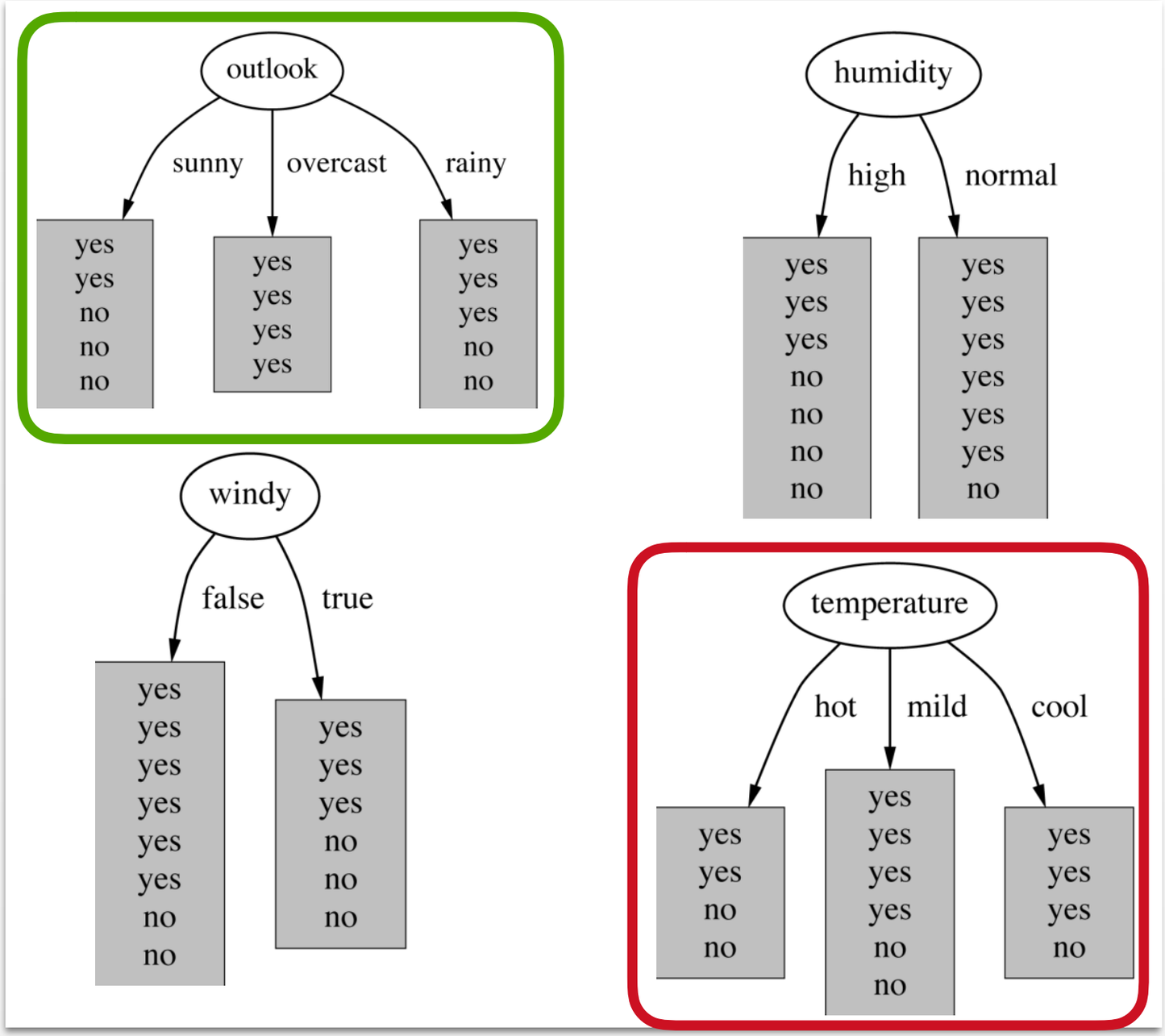
- What is a ***good*** split?
- What is a ***bad*** split?

-> Find “pure” **subsets**

Commonly Used Metrics

- Entropy
- Gini Index

Choosing the best split



Intuition

- What is a ***good*** split?
- What is a ***bad*** split?

-> Find “pure” **subsets**

Commonly Used Metrics

- **Entropy**
- Gini Index

Entropy of a Discrete Variable

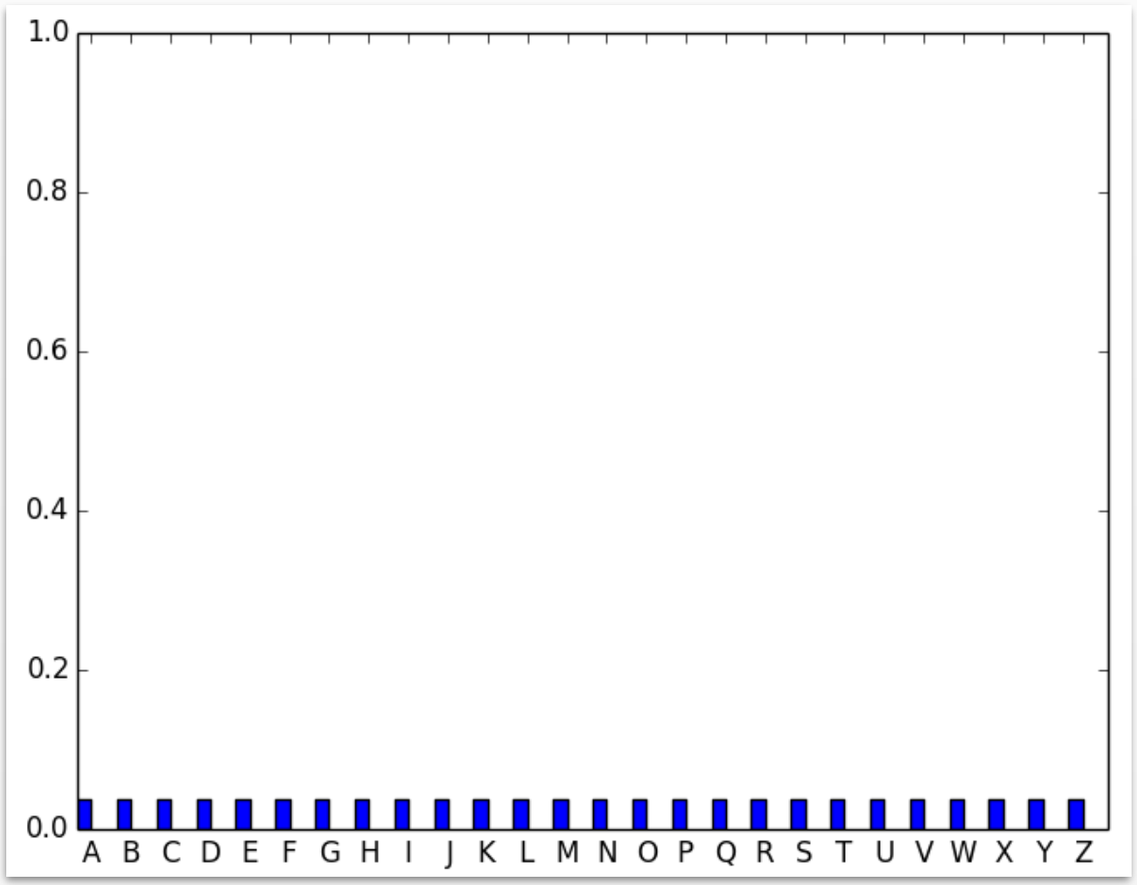
Entropy of a random variable X
(measured in "bits")

$$H(X) = -\sum_{i=1}^{n} (p(x_i) \log_2 (p(x_i)))$$

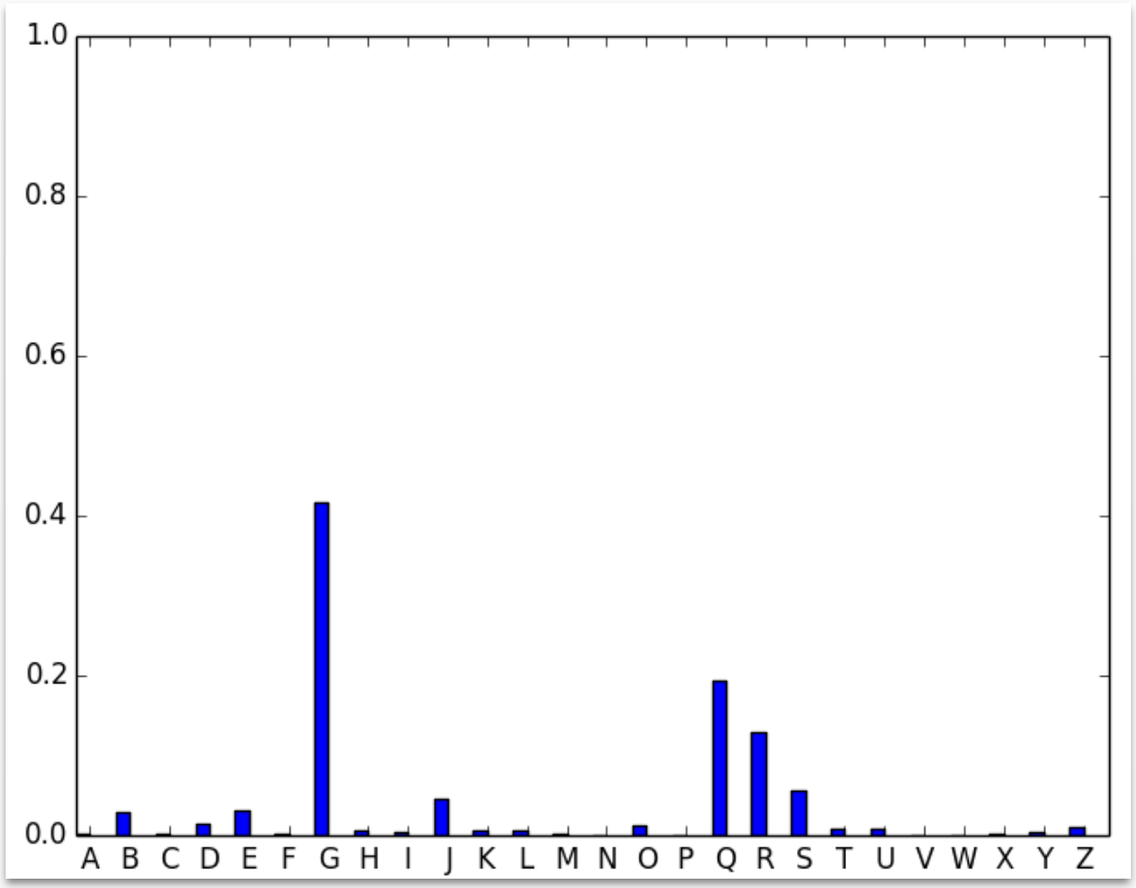
where $p(x_i)$ is the probability
that $X = x_1, \ldots, x_n$

Entropy of a Discrete Variable

High Entropy



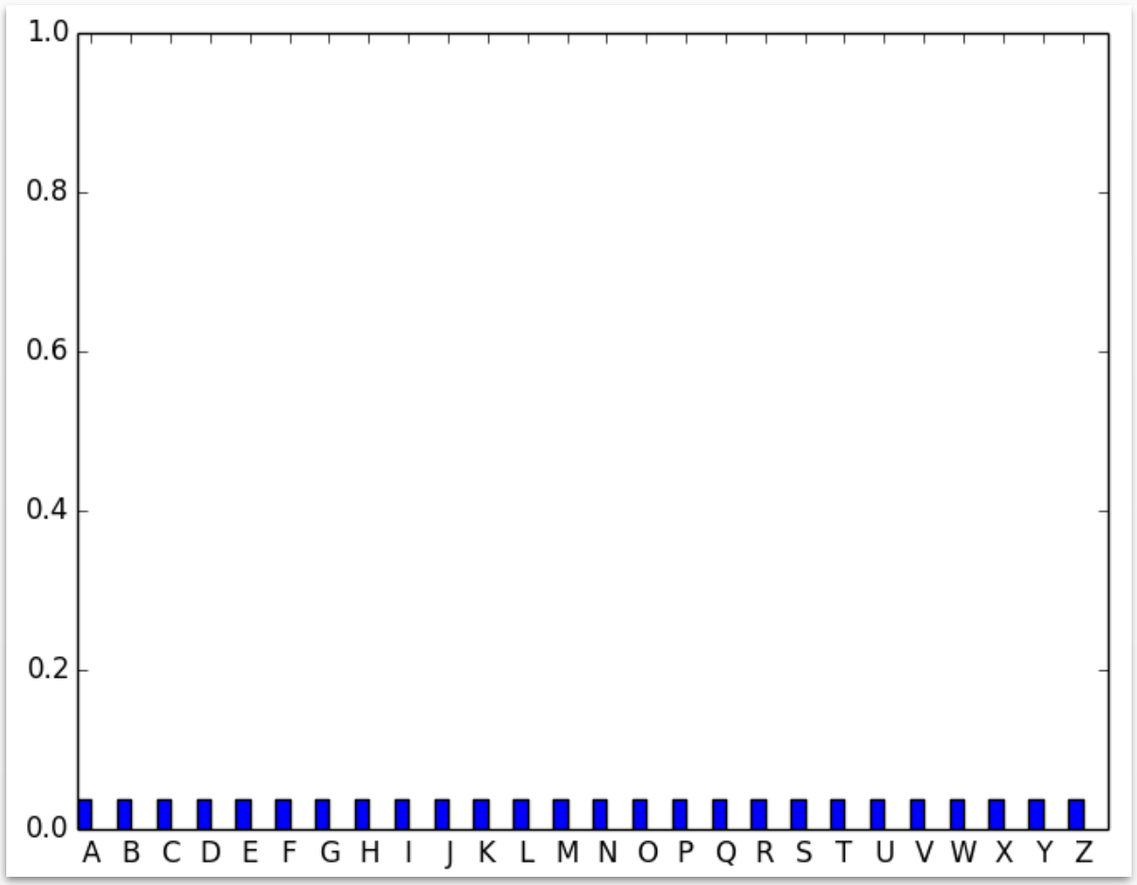
Low Entropy



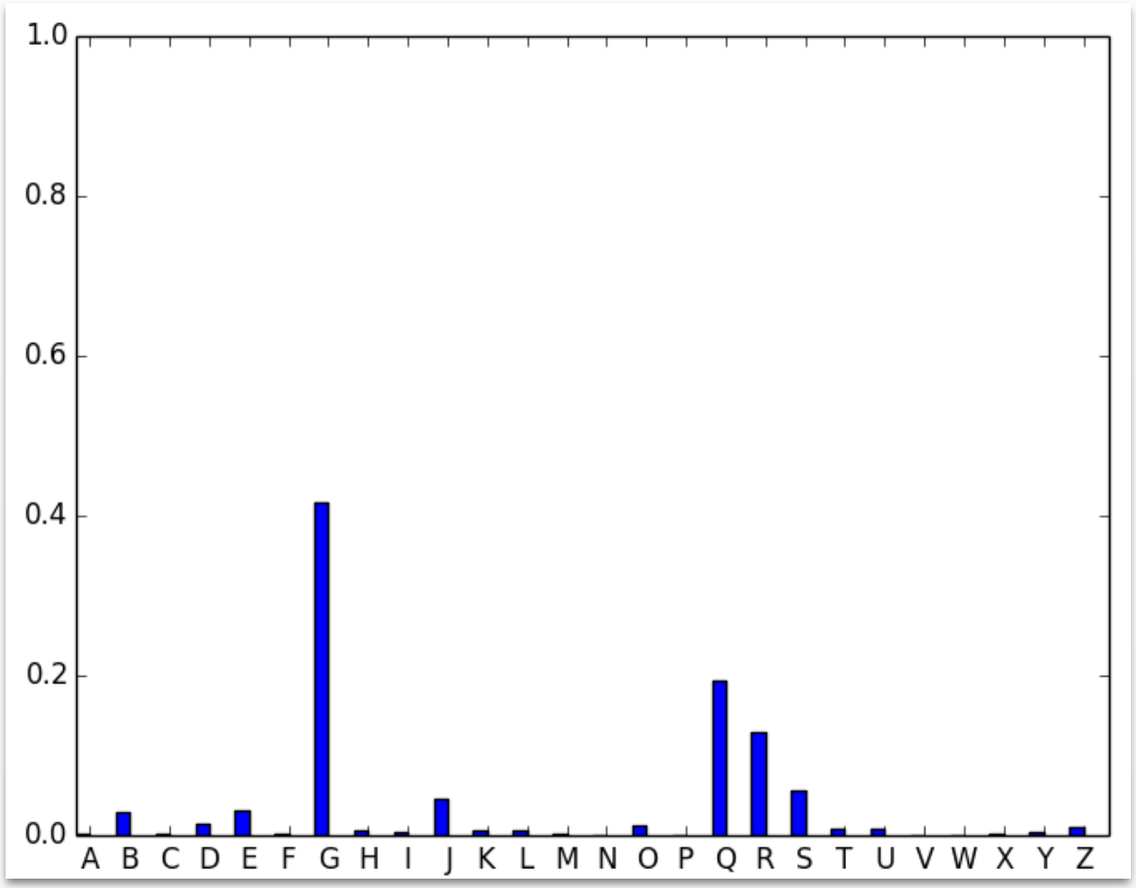
$$H(X) = -\sum_{i=1}^{n}(p(x_i)\log_2(p(x_i)))$$

Entropy of a Discrete Variable

High Entropy



Low Entropy



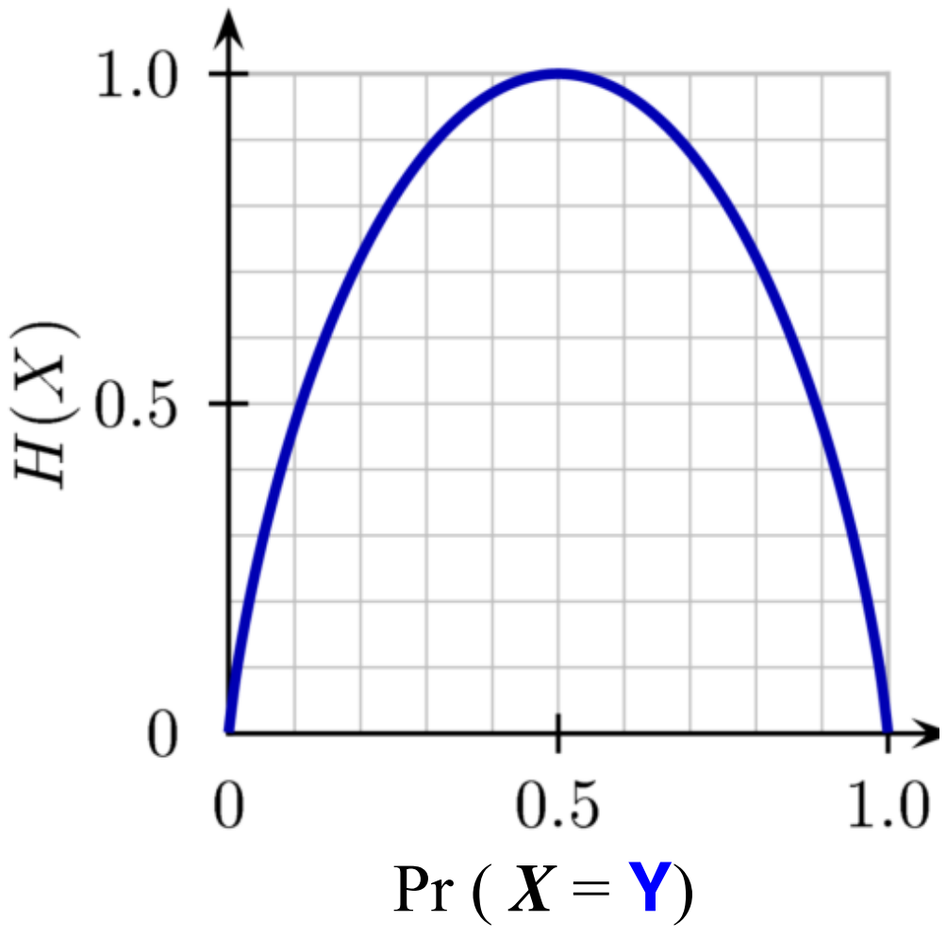
$$H(X) = -\sum_{i=1}^{n}(p(x_i)\log_2(p(x_i)))$$

Best and worst case?

Entropy of a Binary Variable

Consider a binary variable $X = \{\mathbf{Y}, \mathbf{N}\}$, $n = 2$

$$H(X) = -\sum_{i=1}^{n} p(\mathbf{Y}) \log p(\mathbf{Y}) - p(\mathbf{N}) \log p(\mathbf{N})$$

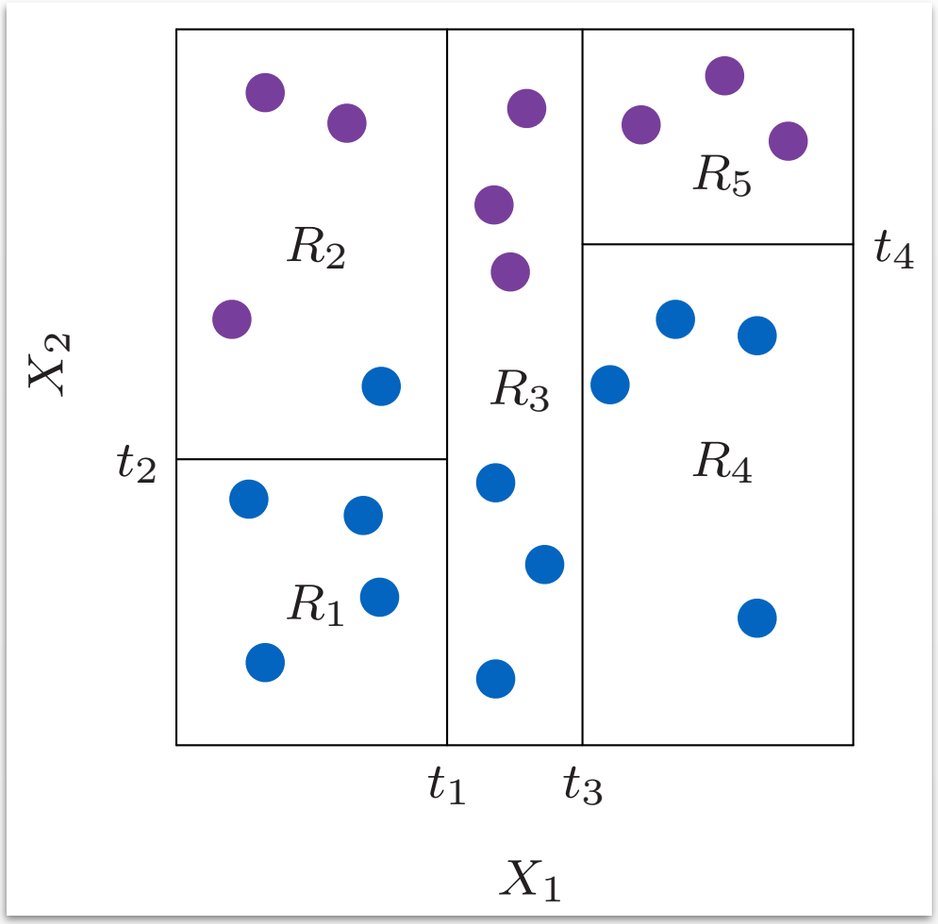


$\mathrm{p}(\mathbf{Y})=0 \Rightarrow H(X) = 0$

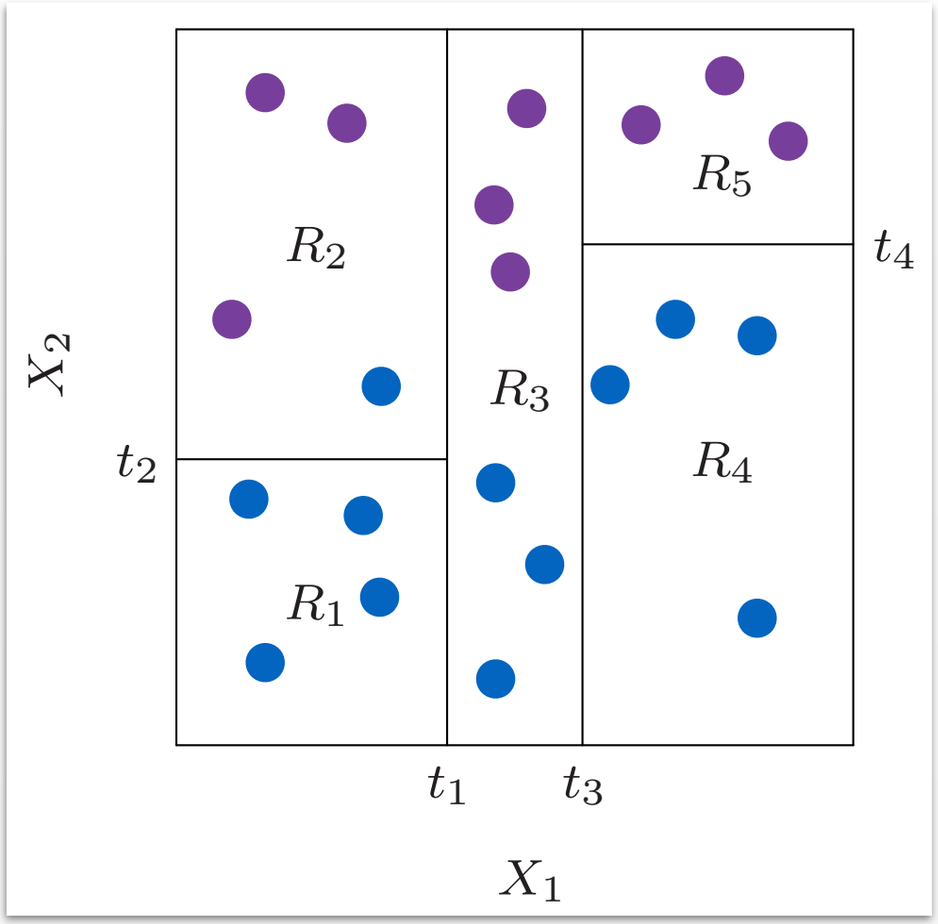
$\mathrm{p}(\mathbf{Y})=\frac{1}{2} \Rightarrow H(X) = 1$

$\mathrm{p}(\mathbf{Y})=\frac{1}{3} \Rightarrow H(X) = ?$

Information Gain of a Split



Information Gain of a Split



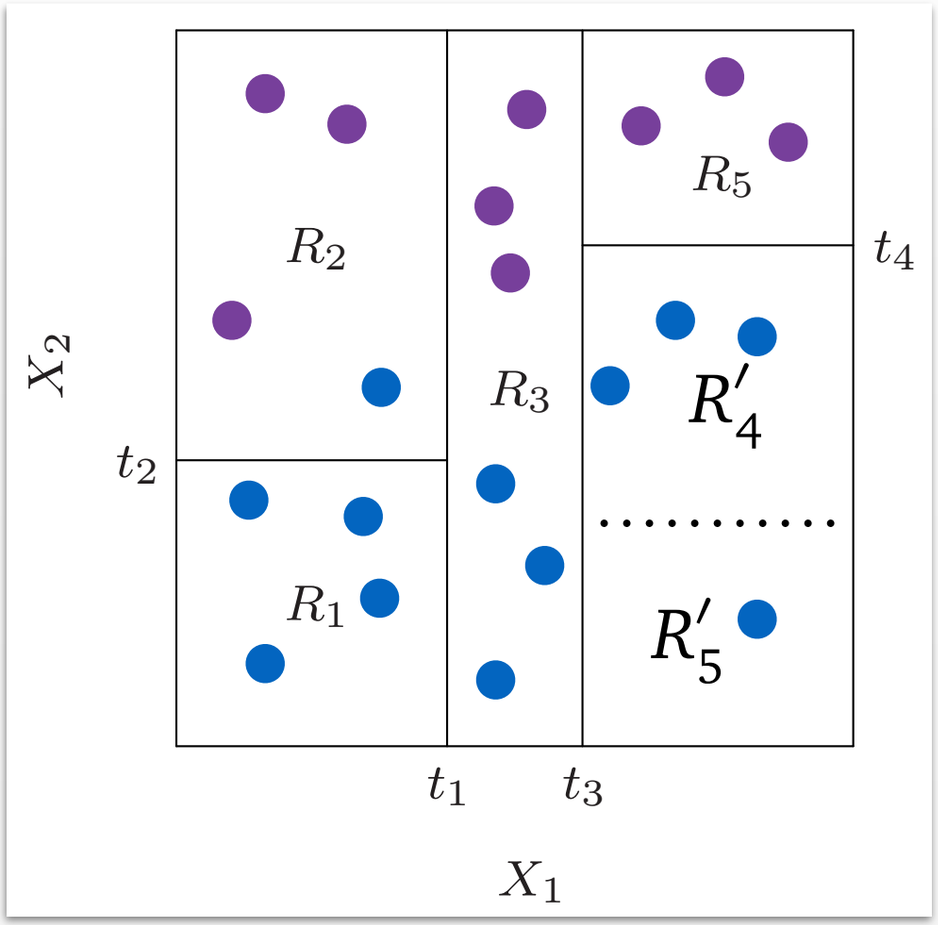
Label distribution for R_m

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\boldsymbol{x}_i \in R_m} I(y_i = k)$$

Entropy for R_m

$$H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Information Gain of a Split



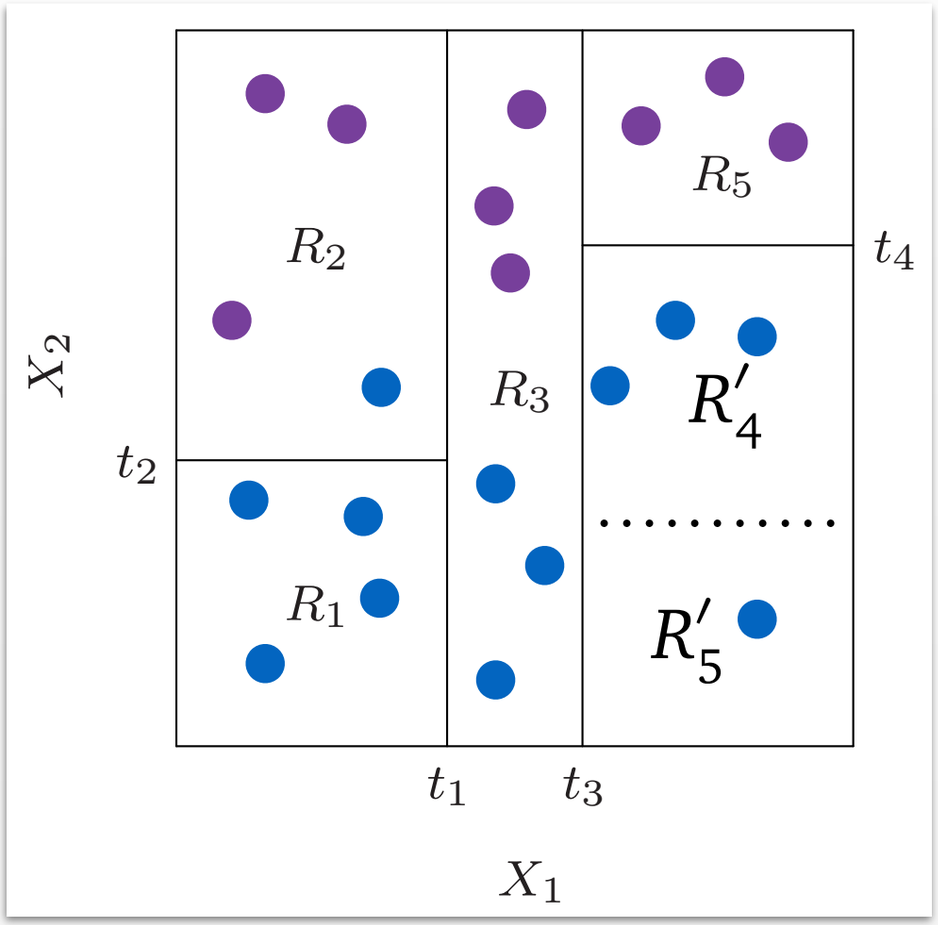
Label distribution for R_m

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\boldsymbol{x}_i \in R_m} I(y_i = k)$$

Entropy for R_m

$$H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Information Gain of a Split



Label distribution for R_m

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\boldsymbol{x}_i \in R_m} I(y_i = k)$$

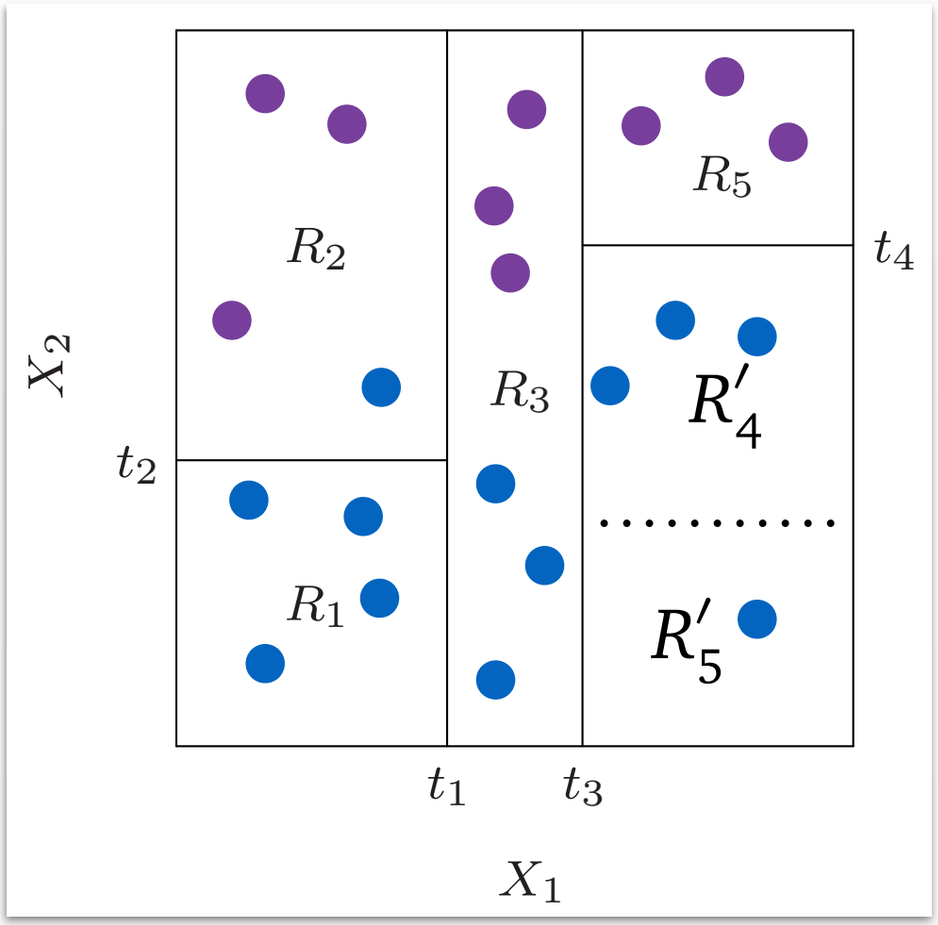
Entropy for R_m

$$H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Information Gain for split $R_m \rightarrow \{R'_m, R'_{m+1}\}$

$$H = \frac{1}{N} \sum_{m} N_m H_m$$

Information Gain of a Split



Label distribution for R_m

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\boldsymbol{x}_i \in R_m} I(y_i = k)$$

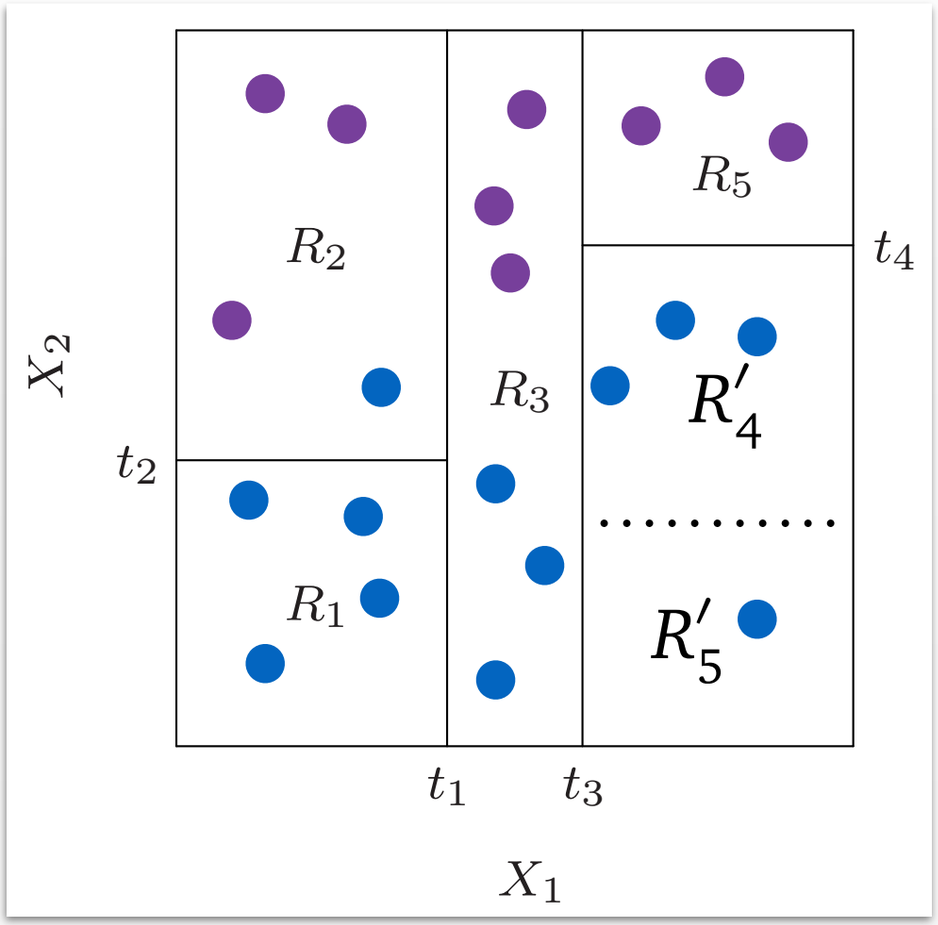
Entropy for R_m

$$H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Information Gain for split $R_m \rightarrow \{R'_m, R'_{m+1}\}$

$$H = \frac{1}{N} \sum_m N_m H_m \qquad IG = H - H'$$

Information Gain of a Split



Label distribution for R_m

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\boldsymbol{x}_i \in R_m} I(y_i = k)$$

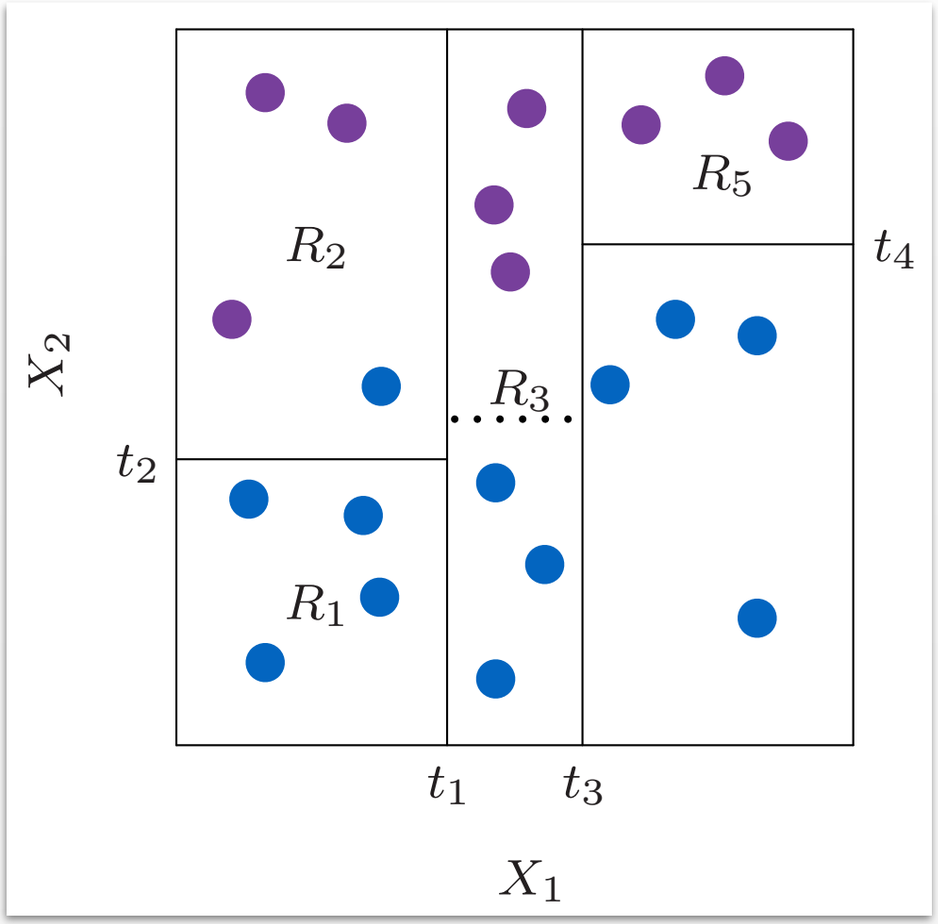
Entropy for R_m

$$H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Information Gain for split $R_m \rightarrow \{R'_m, R'_{m+1}\}$

$$H = \frac{1}{N} \sum_m N_m H_m \qquad IG = \frac{1}{N} \left(N_m H_m - (N'_m H'_m + N'_{m+1} H'_{m+1}) \right)$$

Information Gain of a Split



Label distribution for R_m

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\boldsymbol{x}_i \in R_m} I(y_i = k)$$

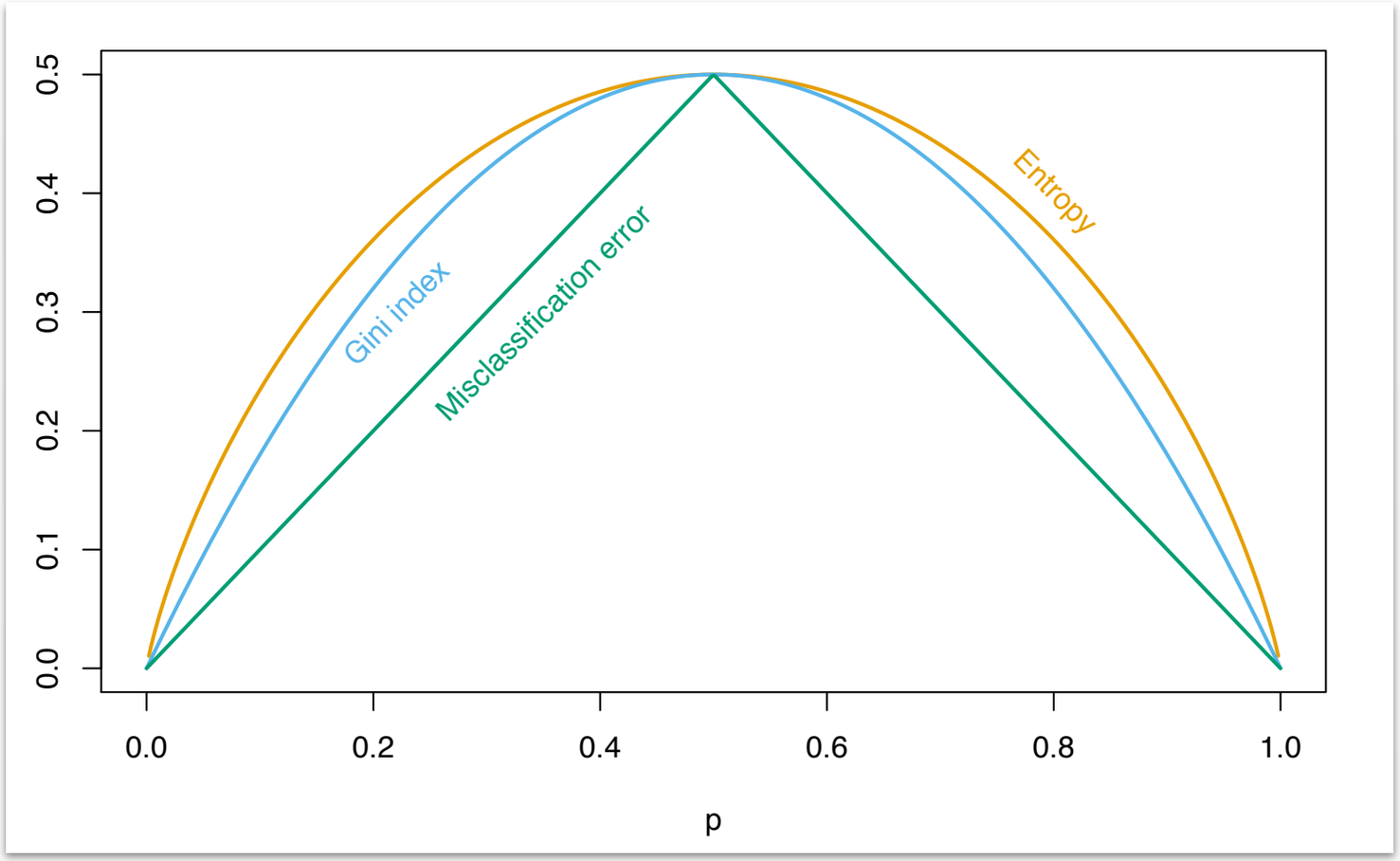
Entropy for R_m

$$H_m = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

Information Gain for split $R_m \rightarrow \{R'_m, R'_{m+1}\}$

$$H = \frac{1}{N} \sum_m N_m H_m \qquad IG = \frac{1}{N} \left(N_m H_m - (N'_m H'_m + N'_{m+1} H'_{m+1}) \right)$$

Impurity Measures

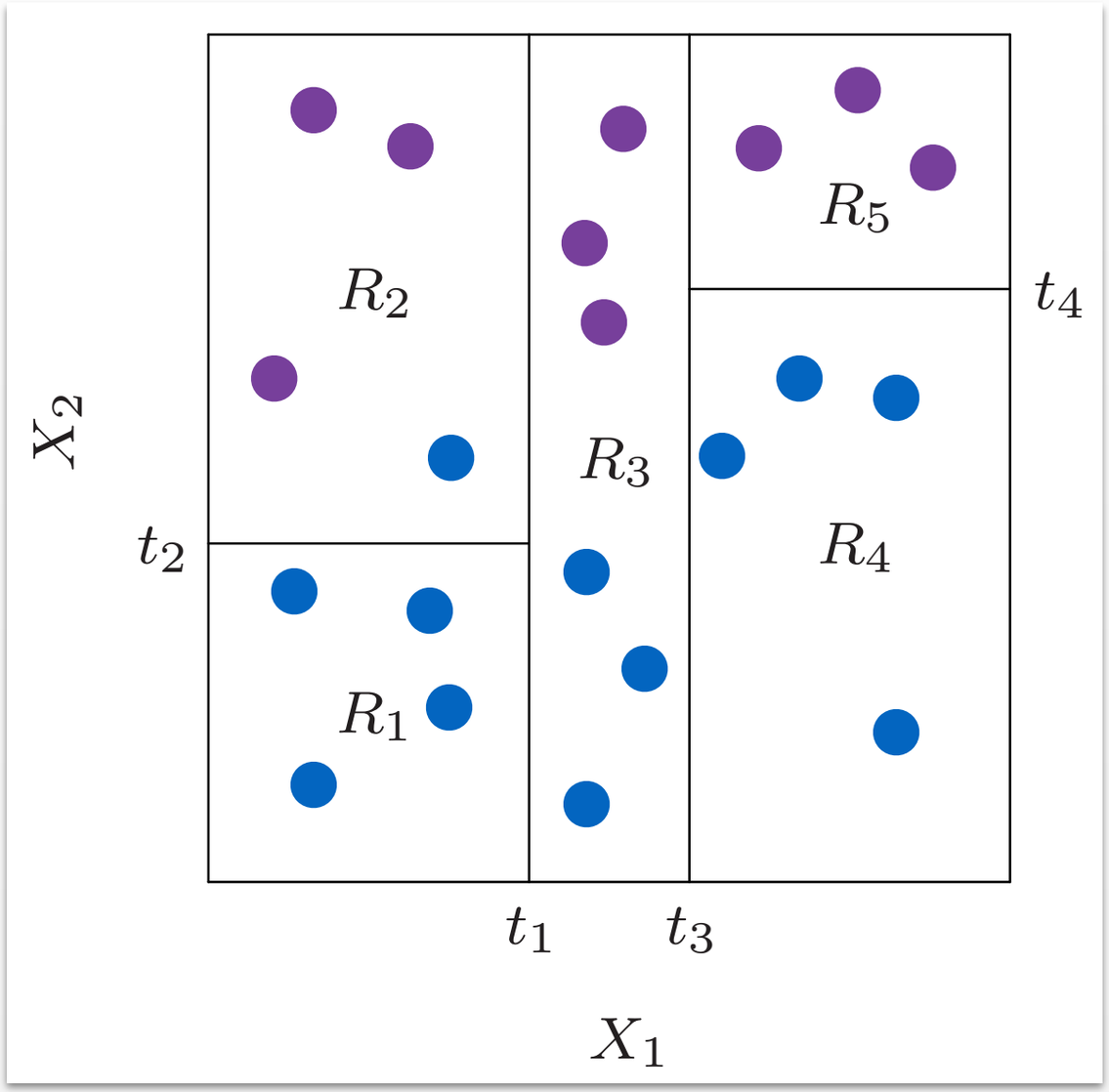


Misclassification error: $\frac{1}{N_m}\sum_{i\in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}.$

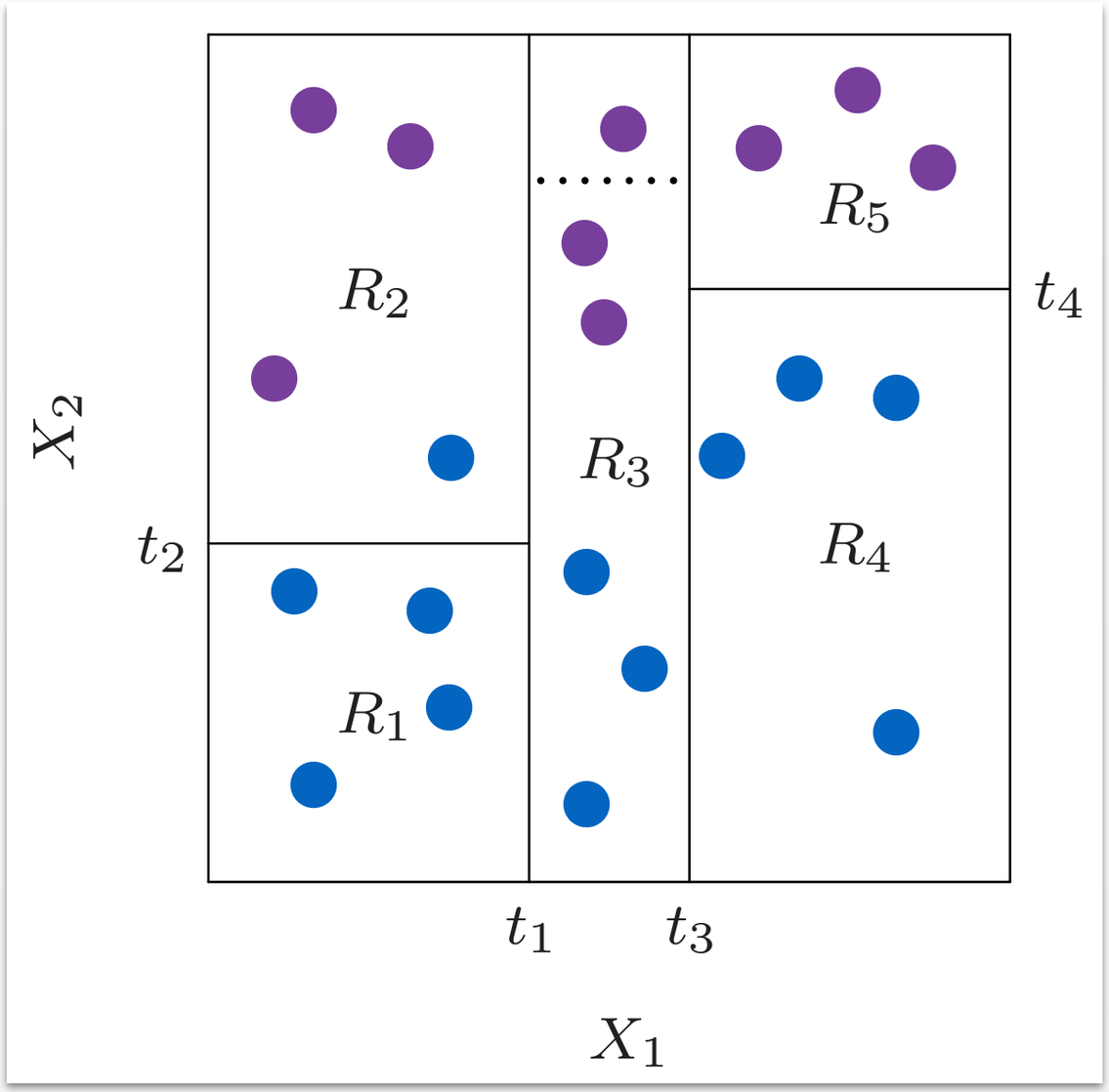
Gini index: $\sum_{k\neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}).$

Cross-entropy or deviance: $-\sum_{k=1}^{K} \hat{p}_{mk}\log\hat{p}_{mk}.$

Can we evaluate all splits?

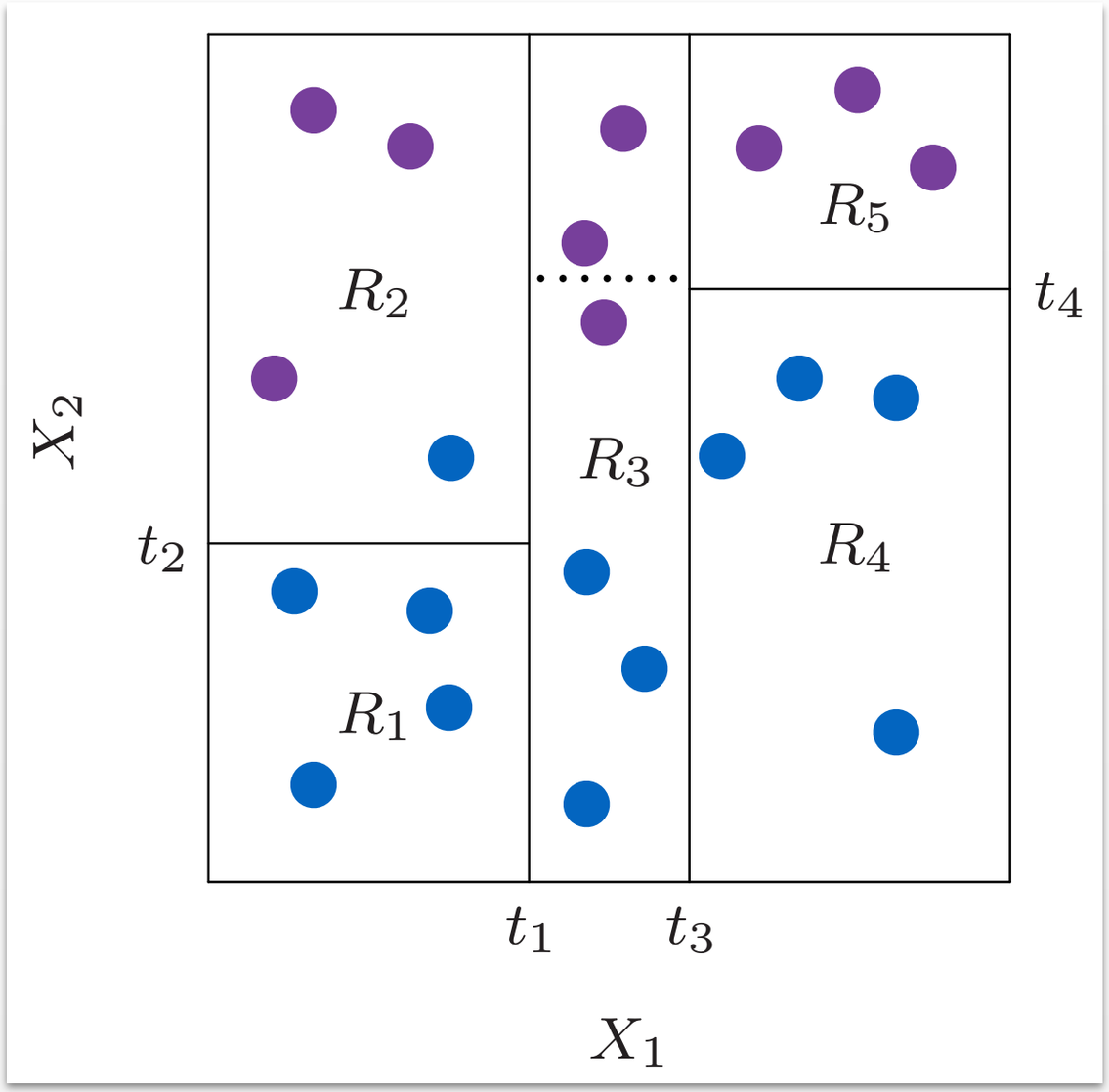


Can we evaluate all splits?



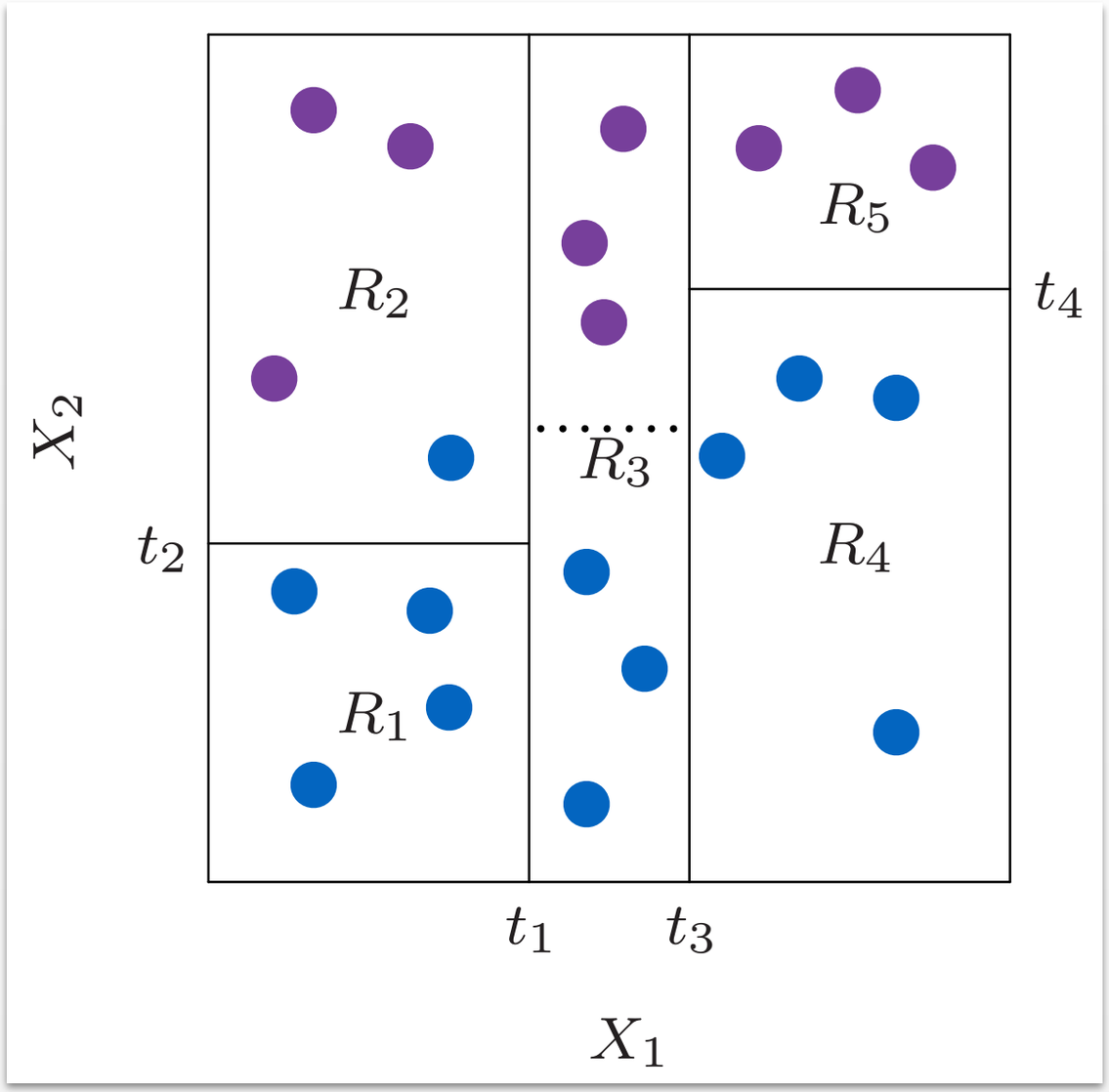
Yes! Evaluate all N_m - 1 midpoints between examples

Can we evaluate all splits?



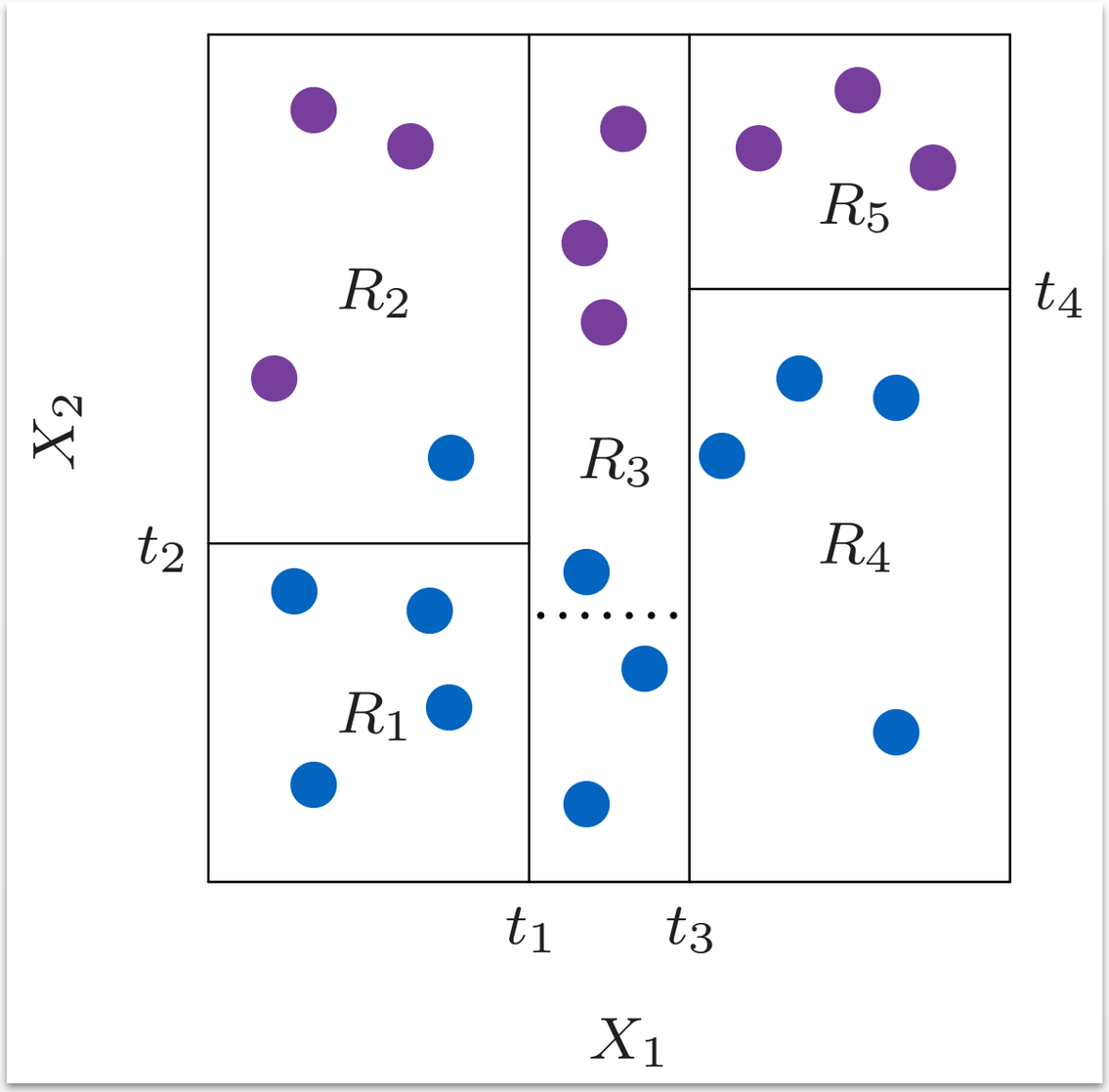
Yes! Evaluate all N_m - 1 midpoints between examples

Can we evaluate all splits?



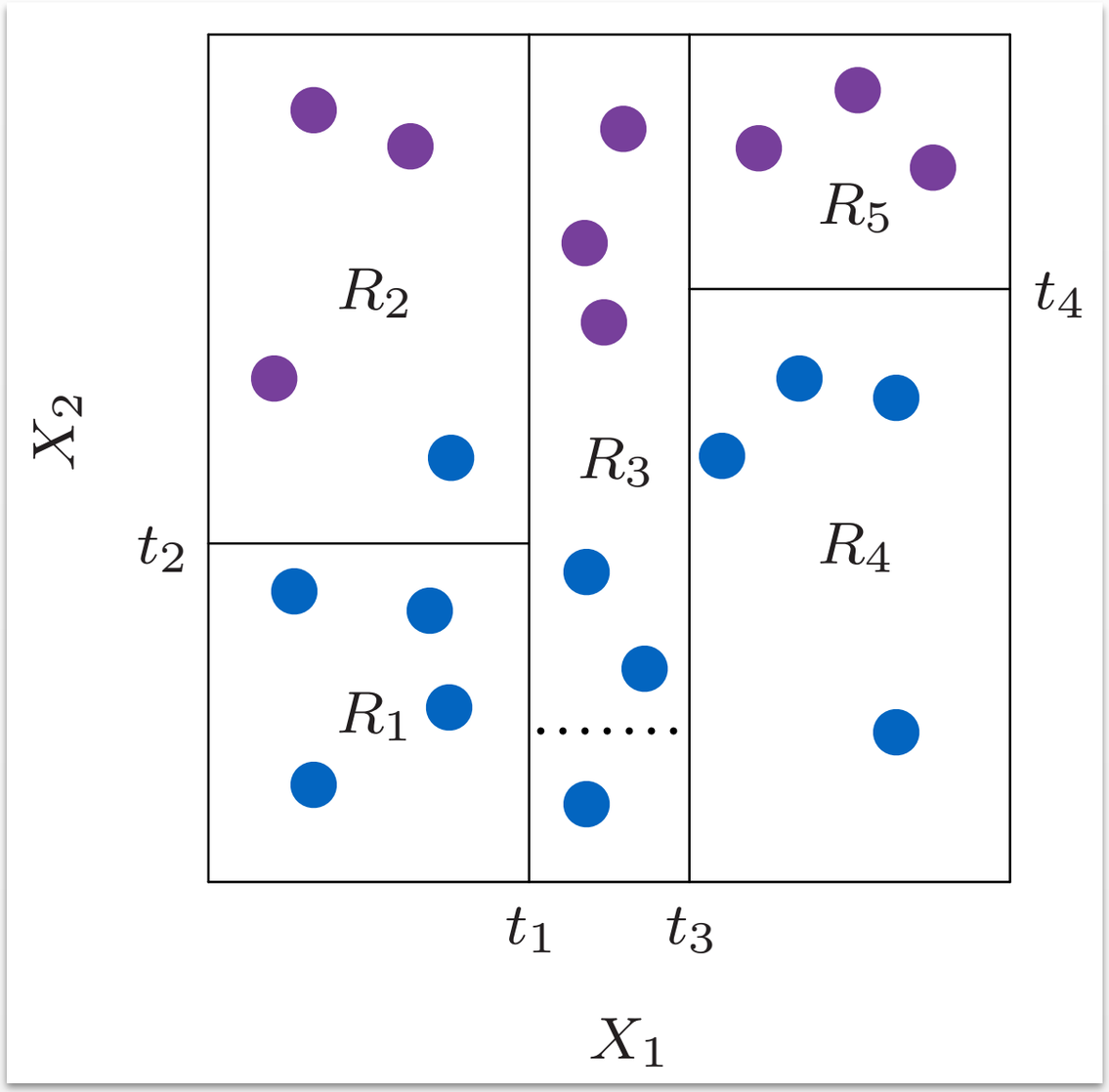
Yes! Evaluate all N_m - 1 midpoints between examples

Can we evaluate all splits?



Yes! Evaluate all N_m - 1 midpoints between examples

Can we evaluate all splits?



Yes! Evaluate all N_m - 1 midpoints between examples

Choosing Tree Depth

- *Naive Solution:* Keep splitting until you have one item in each bin?
- Why might this be a good / bad idea?

Review: Bias-Variance Trade-off

Maximum likelihood estimator

$$\hat{f} := \underset{\tilde{f}}{\operatorname{argmax}}\, p(\boldsymbol{y} \mid \tilde{f})$$

Bias-variance decomposition

(*expected value over possible data points*)

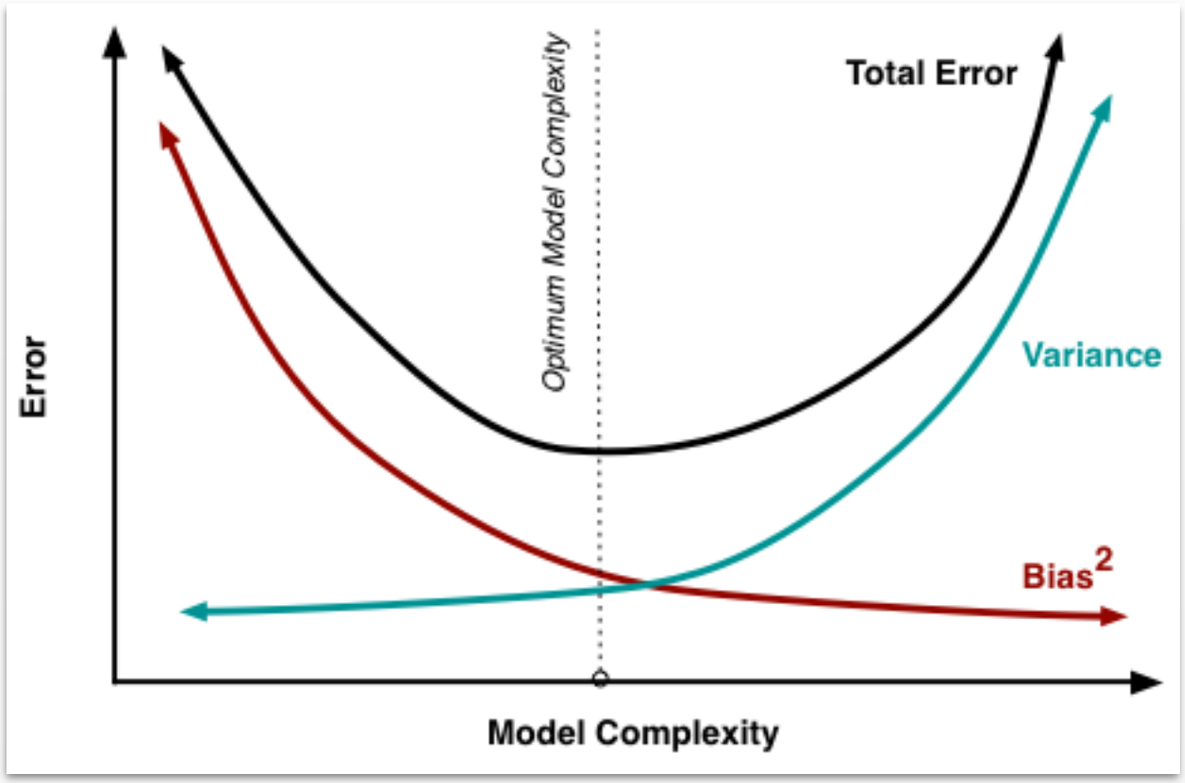
$$\mathbb{E}[(y - \hat{f}(\boldsymbol{x}))^2] = \operatorname{Bias}[\hat{f}(\boldsymbol{x})]^2 + \operatorname{Var}[\hat{f}(\boldsymbol{x})] + \sigma^2$$

$$\operatorname{Bias}[\hat{f}(\boldsymbol{x})] = \mathbb{E}[\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x})]$$

$$\operatorname{Var}[\hat{f}(\boldsymbol{x})] = \mathbb{E}[\hat{f}(\boldsymbol{x})^2] - \mathbb{E}[\hat{f}(\boldsymbol{x})]^2$$

$$\sigma^2 = \mathbb{E}[y^2] - \mathbb{E}[f(\boldsymbol{x})]^2$$

Bias and Variance for Trees



- *Low depth*
  Low **variance**, high **bias** (underfitting)
- *High depth*
  High **variance**, low **bias** (overfitting)

Regularize cost by tree size

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m H_m(T) + \alpha|T|$$

Ensemble Methods

Idea: Combine Many Classifiers

Additive Model

$$\boldsymbol{y} = \alpha_0 + \sum_{j=1}^{p} \alpha_j f_j(\boldsymbol{x}_j) + \epsilon$$

Weighted Majority Vote

$$\boldsymbol{y}^* = \mathrm{Sign}[\alpha_0 + \sum_{j=1}^{p} \alpha_j f_j(\boldsymbol{x}_j^*)]$$

Two Commonly Used Techniques

- *Bagging:* Train classifiers on different subsets of the the data
- *Boosting:* Train next classifier to improve prediction on previously misclassified examples

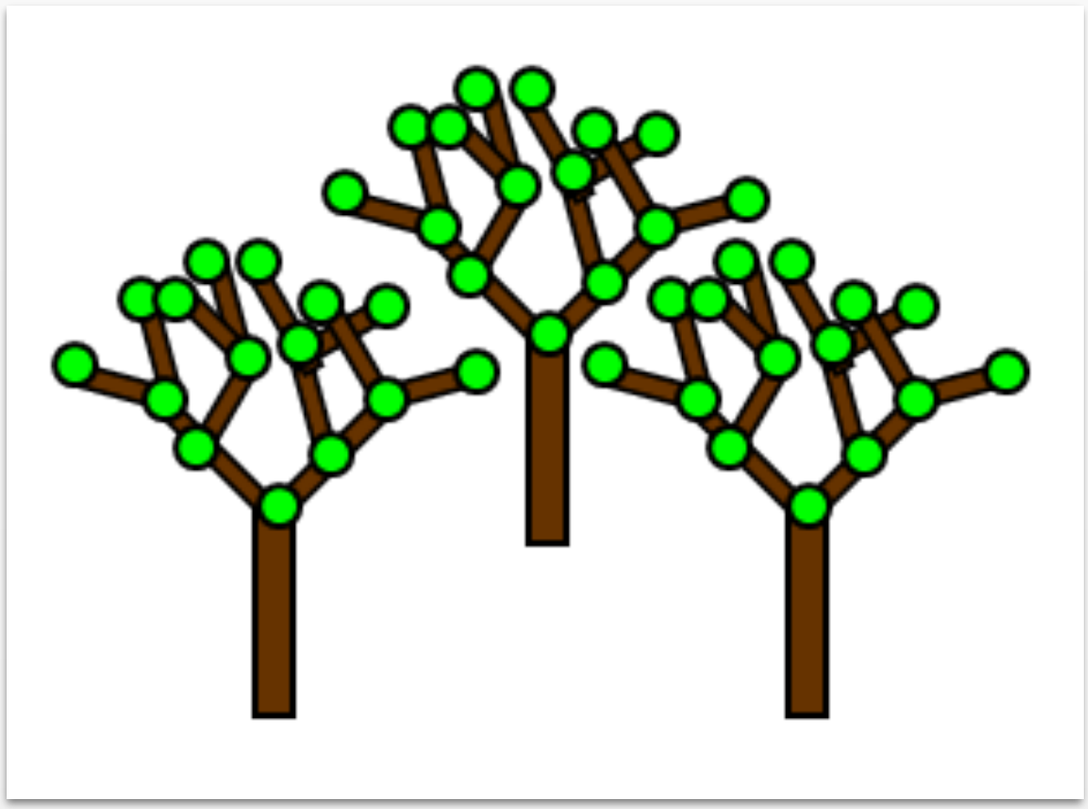
Bootstrap Aggregation (Bagging)

$$\hat{f}_{\mathrm{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$

- ***For*** b = 1 to B
  - Construct a bootstrap sample
    $\boldsymbol{x}_b, \boldsymbol{y}_b = \{(x_{b1}, y_{b1}), \ldots, (x_{bn}, y_{bn})\}$
    by sampling at random from **x**,**y** *with replacement*
  - Train a classifier f^{*b} on $\boldsymbol{x}_b$, $\boldsymbol{y}_b$

Random Forests

$$\hat{f}_{\mathrm{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x).$$



- ***For*** b = 1 to B
  - Select *N* datapoints at random *with* replacement
  - Select *M* features at random *without* replacement
  - Train classifier on subset of data and features

Random Forests

One of the best general-purpose classifiers
(performance is not sensitive to choice of B, N, and M)

Table 3. Normalized scores of each learning algorithm by problem (averaged over eight metrics)

MODEL	CAL	COVT	ADULT	LTR.P1	LTR.P2	MEDIS	SLAC	HS	MG	CALHOUS	COD	BACT	MEAN
BST-DT	PLT	**.938**	.857	**.959**	**.976**	.700	.869	**.933**	.855	**.974**	**.915**	.878*	**.896***
RF	PLT	.876	.930	.897	.941	**.810**	.907*	.884	.883	.937	.903*	.847	.892
BAG-DT	–	.878	.944*	.883	.911	.762	.898*	.856	**.898**	.948	.856	**.926**	.887*
BST-DT	ISO	.922*	.865	.901*	.969	.692*	.878	.927	.845	.965	.912*	.861	.885*
RF	–	.876	.946*	.883	.922	.785	.912*	.871	.891*	.941	.874	.824	.884
BAG-DT	PLT	.873	.931	.877	.920	.752	.885	.863	.884	.944	.865	.912*	.882
RF	ISO	.865	.934	.851	.935	.767*	**.920**	.877	.876	.933	.897*	.821	.880
BAG-DT	ISO	.867	.933	.840	.915	.749	.897	.856	.884	.940	.859	.907*	.877
SVM	PLT	.765	.886	.936	.962	.733	.866	.913*	.816	.897	.900*	.807	.862
ANN	–	.764	.884	.913	.901	.791*	.881	.932*	.859	.923	.667	.882	.854
SVM	ISO	.758	.882	.899	.954	.693*	.878	.907	.827	.897	.900*	.778	.852
ANN	PLT	.766	.872	.898	.894	.775	.871	.929*	.846	.919	.665	.871	.846
ANN	ISO	.767	.882	.821	.891	.785*	.895	.926*	.841	.915	.672	.862	.842
BST-DT	–	.874	.842	.875	.913	.523	.807	.860	.785	.933	.835	.858	.828
KNN	PLT	.819	.785	.920	.937	.626	.777	.803	.844	.827	.774	.855	.815
KNN	–	.807	.780	.912	.936	.598	.800	.801	.853	.827	.748	.852	.810
KNN	ISO	.814	.784	.879	.935	.633	.791	.794	.832	.824	.777	.833	.809
BST-STMP	PLT	.644	**.949**	.767	.688	.723	.806	.800	.862	.923	.622	.915*	.791
SVM	–	.696	.819	.731	.860	.600	.859	.788	.776	.833	.864	.763	.781
BST-STMP	ISO	.639	.941	.700	.681	.711	.807	.793	.862	.912	.632	.902*	.780
BST-STMP	–	.605	.865	.540	.615	.624	.779	.683	.799	.817	.581	.906*	.710
DT	ISO	.671	.869	.729	.760	.424	.777	.622	.815	.832	.415	.884	.709
DT	–	.652	.872	.723	.763	.449	.769	.609	.829	.831	.389	.899*	.708
DT	PLT	.661	.863	.734	.756	.416	.779	.607	.822	.826	.407	.890*	.706
LR	–	.625	.886	.195	.448	.777*	.852	.675	.849	.838	.647	.905*	.700
LR	ISO	.616	.881	.229	.440	.763*	.834	.659	.827	.833	.636	.889*	.692
LR	PLT	.610	.870	.185	.446	.738	.835	.667	.823	.832	.633	.895	.685
NB	ISO	.574	.904	.674	.557	.709	.724	.205	.687	.758	.633	.770	.654
NB	PLT	.572	.892	.648	.561	.694	.732	.213	.690	.755	.632	.756	.650
NB	–	.552	.843	.534	.556	.011	.714	-.654	.655	.759	.636	.688	.481

R Caruana et al, "An Empirical Comparison of Supervised Learning Algorithms", ICML 2006

Gradient Boosting

- ***For*** $m = 1$ to M
  - Define weighted classifier

$$F_m(\boldsymbol{x}) = \sum_{j=0}^{m} \alpha_j f_j(\boldsymbol{x})$$

  - Define residuals

$$r_n = y_n - F_m(\boldsymbol{x}_n)$$

  - Train next classifier on residuals

$$f_m = \underset{f}{\operatorname{argmin}} \sum_{n=1}^{N} (r_n - f(\boldsymbol{x}_n))^2$$

  - Optimize weight

$$\alpha_m = \underset{\alpha}{\operatorname{argmin}} (y_n - F_m(\boldsymbol{x}_n) - \alpha f_m(\boldsymbol{x}_n))^2$$

Gradient Boosting

- ***For*** $m = 1$ to M
  - Define weighted classifier

$$F_m(\boldsymbol{x}) = \sum_{j=0}^{m} \alpha_j f_j(\boldsymbol{x})$$

  - Define residuals

$$r_n = -\left[\frac{\partial L(y_n, F(\boldsymbol{x_n}))}{\partial F(\boldsymbol{x}_n)}\right]_{F(\boldsymbol{x}_n)=F_m(\boldsymbol{x}_n)}$$

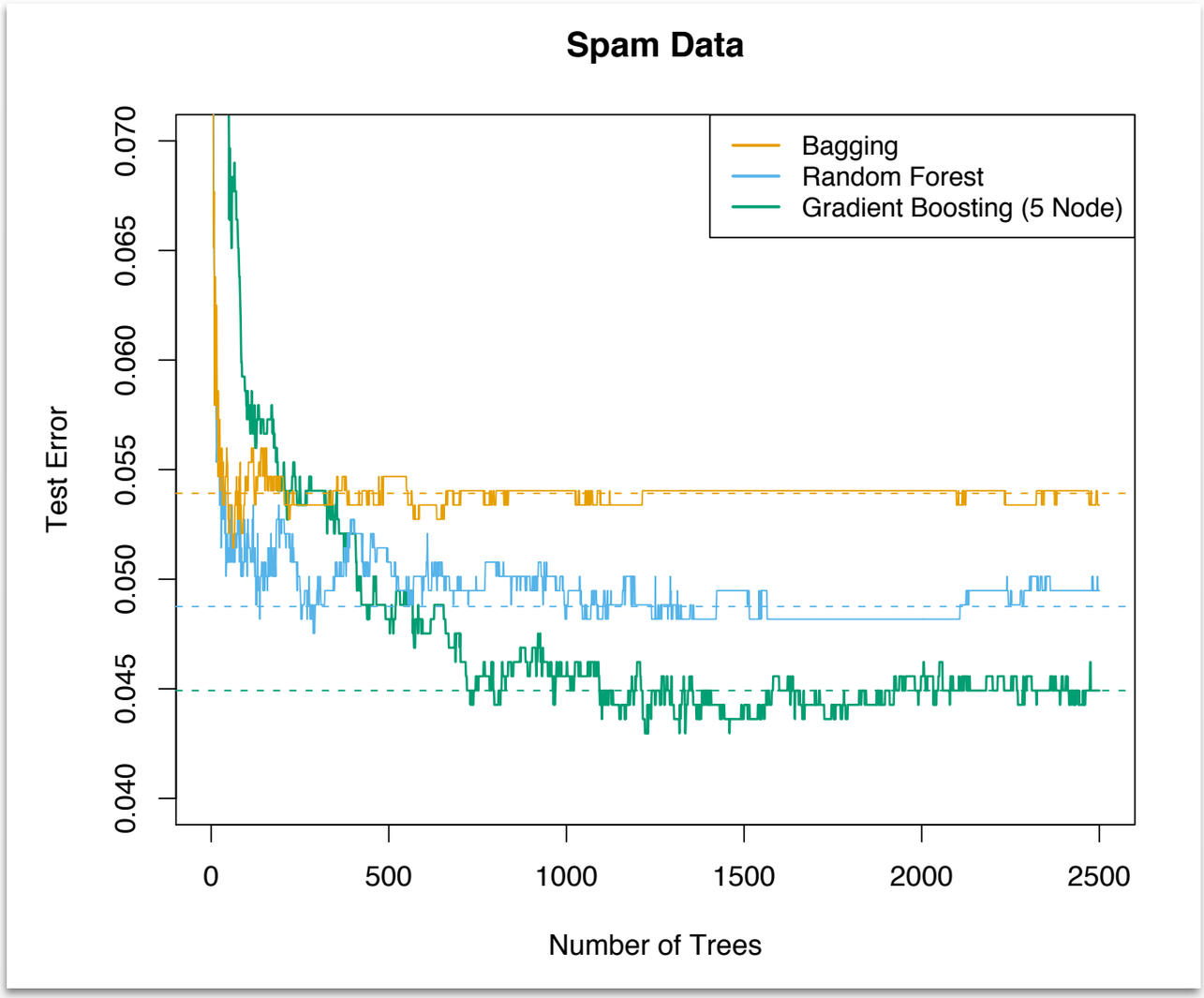
  - Train next classifier on residuals

$$f_m = \underset{f}{\operatorname{argmin}} \sum_{n=1}^{N} L(r_n, f(\boldsymbol{x}_n))$$

  - Optimize weight

$$\alpha_m = \underset{\alpha}{\operatorname{argmin}} \sum_{n=1}^{N} L(y_n, F_m(\boldsymbol{x}_n) + \alpha f_m(\boldsymbol{x}_n))$$

Random Forests vs Gradient Boosting



Gradient boosting often yields slightly better performance

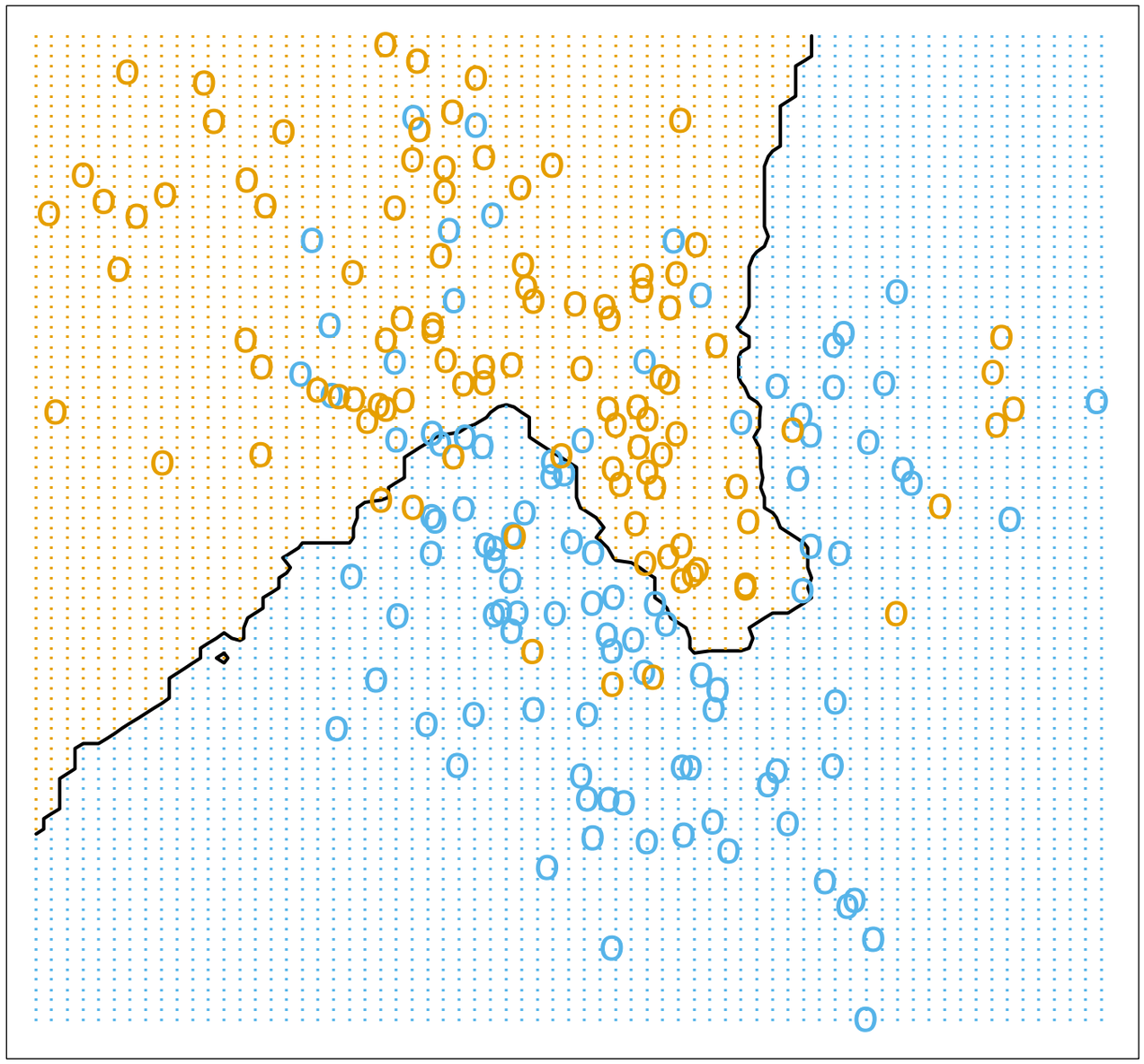
Random Forests

- Combine *full trees* (*high* bias, *low* variance)
- Train on *subset* of data and features
- Amenable to distributed computation

Gradient Boosting

- Combine *stumps* (*low* bias, *high* variance)
- Train on *all* of data and features
- Serial computation

k-Nearest Neighbors



15 Nearest Neighbors

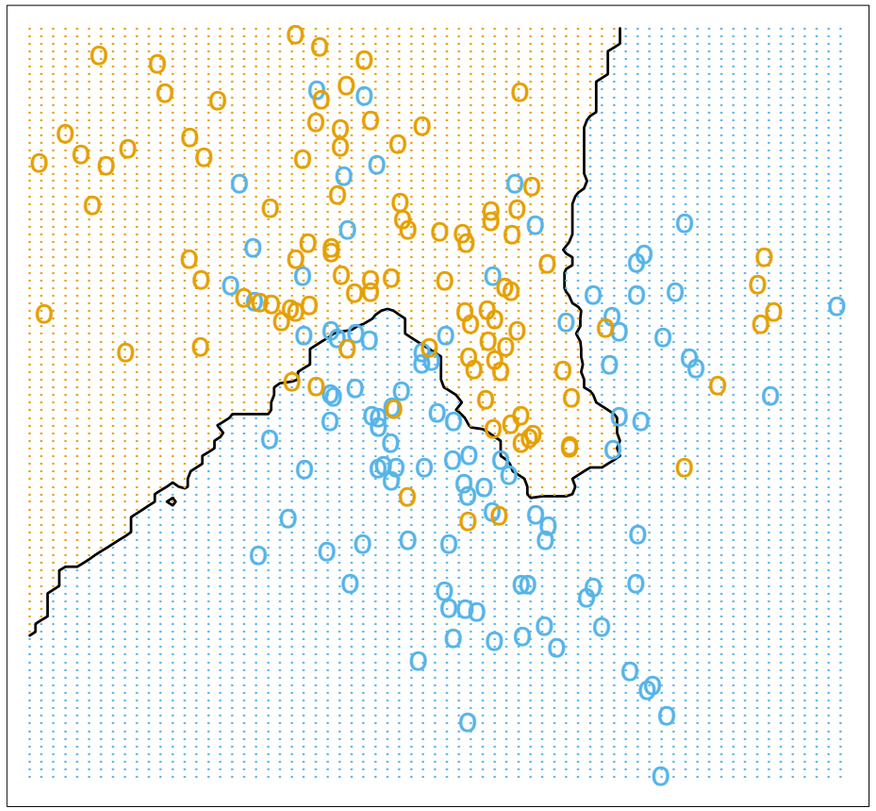
Algorithm

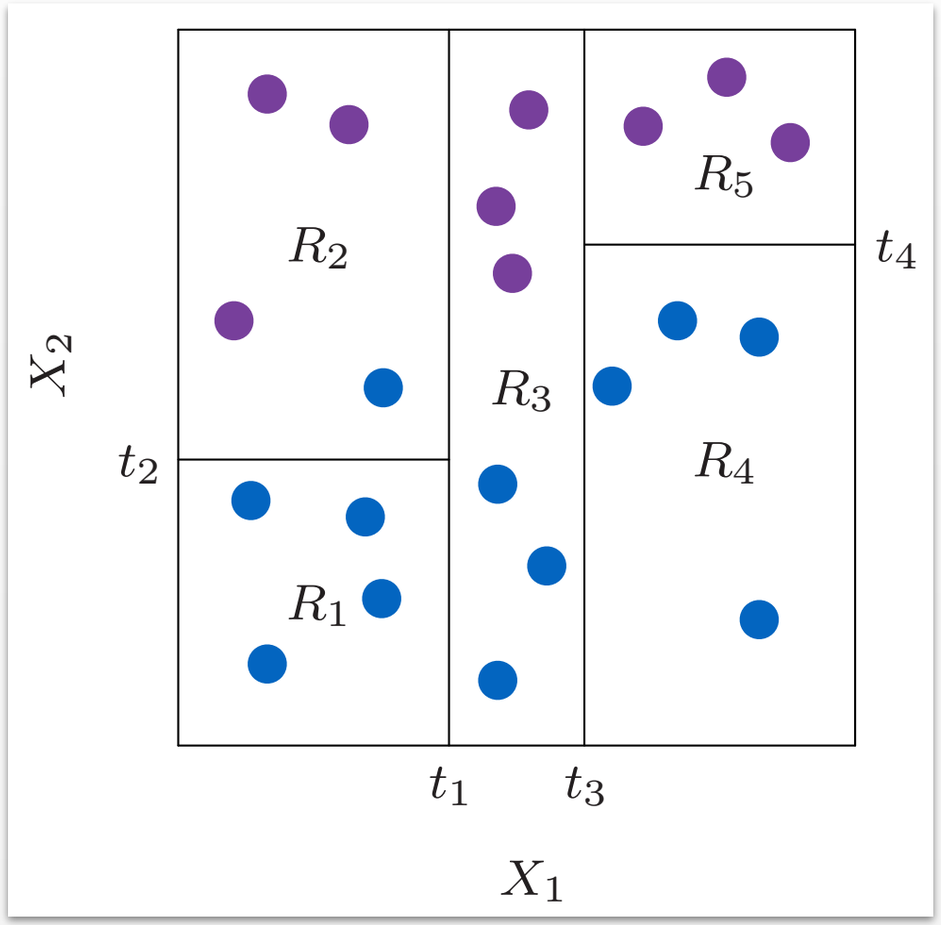
- Find k nearest points to $\boldsymbol{x}^*$
- Predict y^* according to majority vote

Properties

- *Lazy*
  No learned parameters.
- *Instance Learner*
  Needs all data at test time.

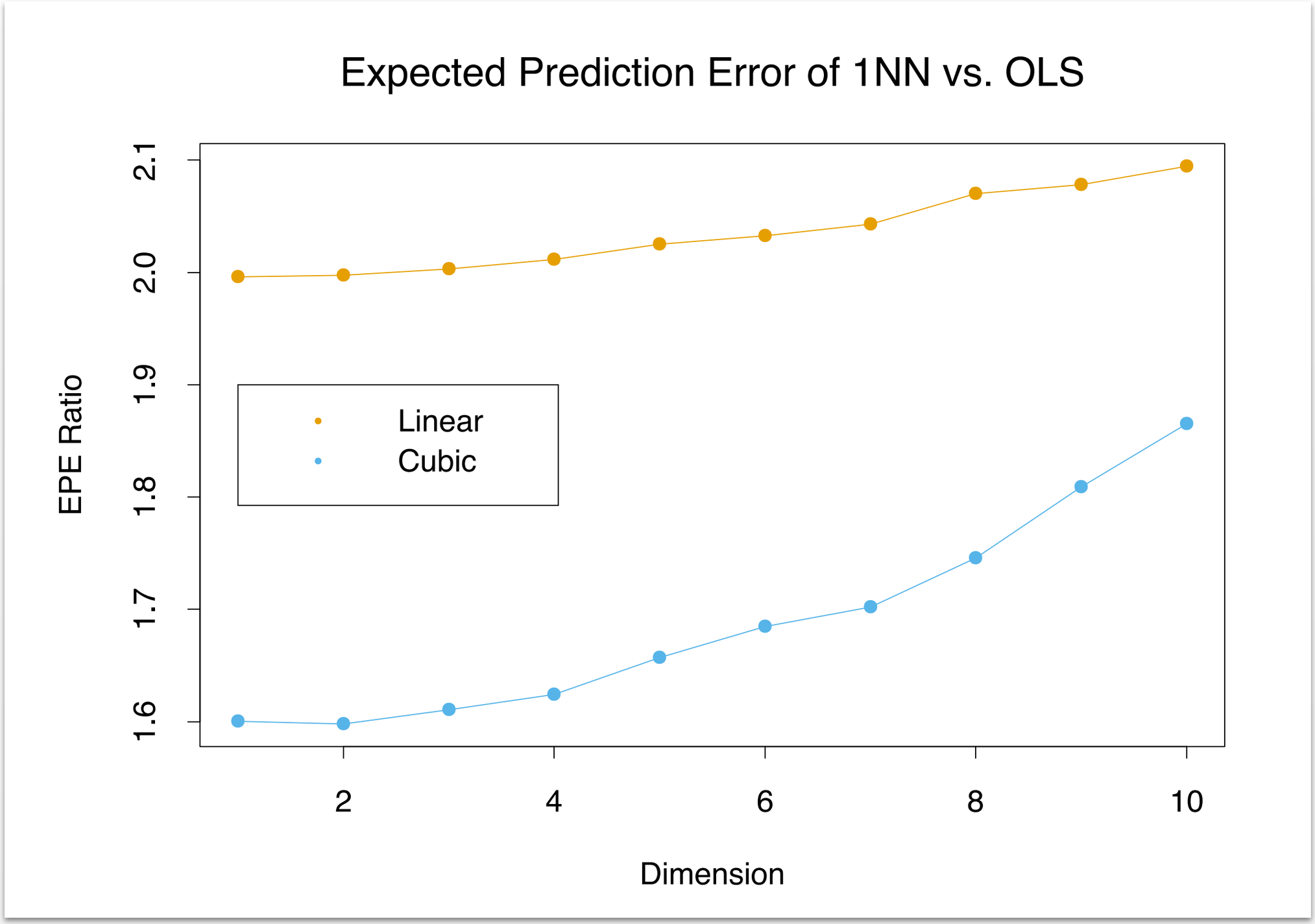
Decision Trees vs Nearest Neighbors



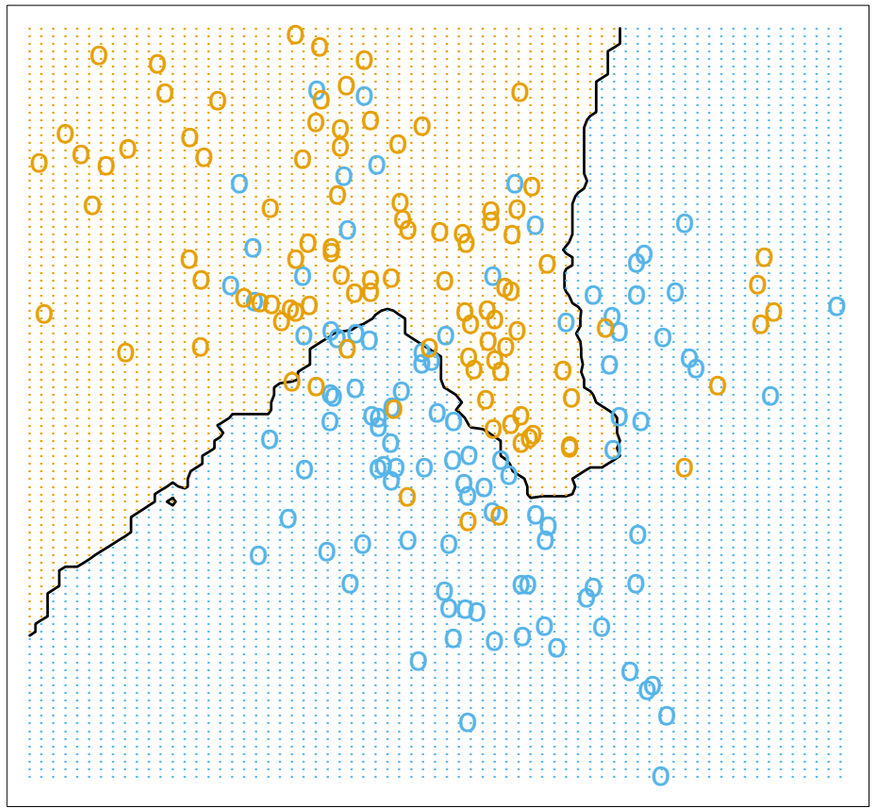


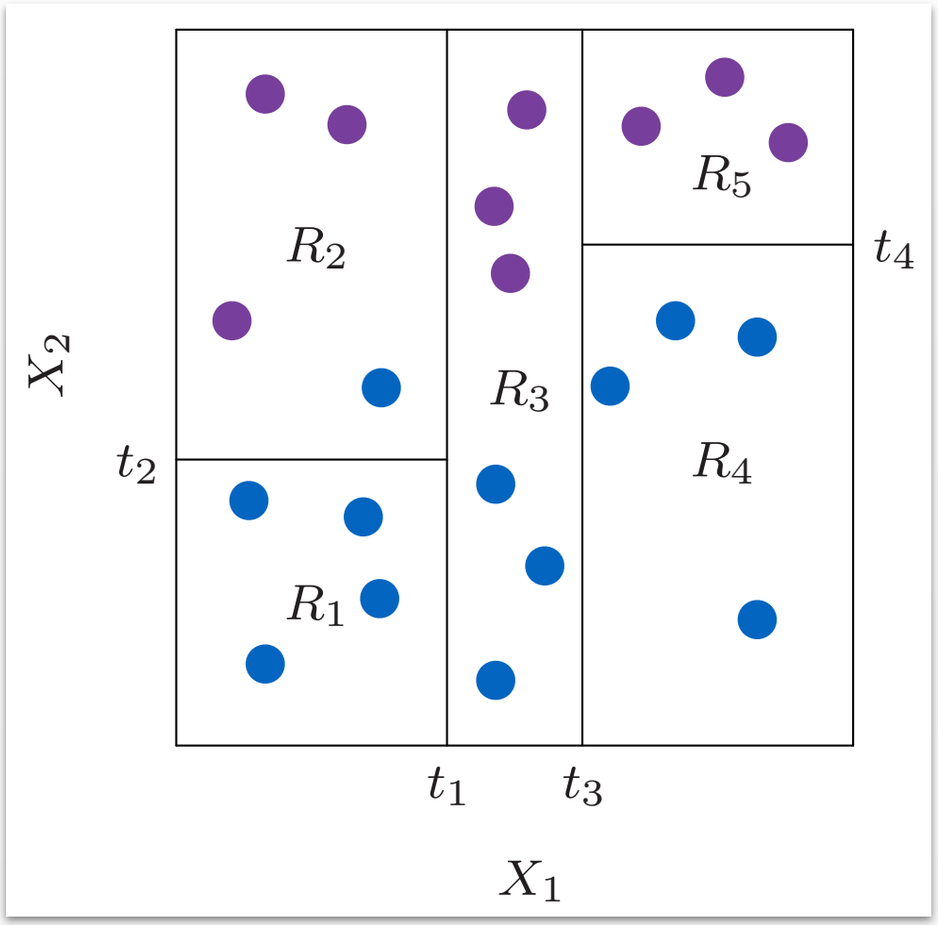
How are these similar?

Curse of Dimensionality



Random Forests vs Nearest Neighbors





Why might random forests perform better than decision trees and/or nearest neighbors?