Trees and Ensembles

Jan Drchal

Artificial Intelligence Center
Faculty of Electrical Engineering
Czech Technical University in Prague

AiCENTER

Outline

Topics covered in the lecture:

- Decision and Regression Trees
- Ensembles: Bagging
- Random Forests
- Ensembles: Boosting
- Gradient Boosting Trees

Decision and Regression Trees

- Supervised machine learning model
- Interpretable
- Supports both classification (decision trees) and regression (regression trees)
- Binary/multi-valued/continuous inputs
- Can deal with missing values
- Fast training and prediction

Decision Trees

Decision Tree Example

- Will John play tennis?

Training examples: **9 yes / 5 no**

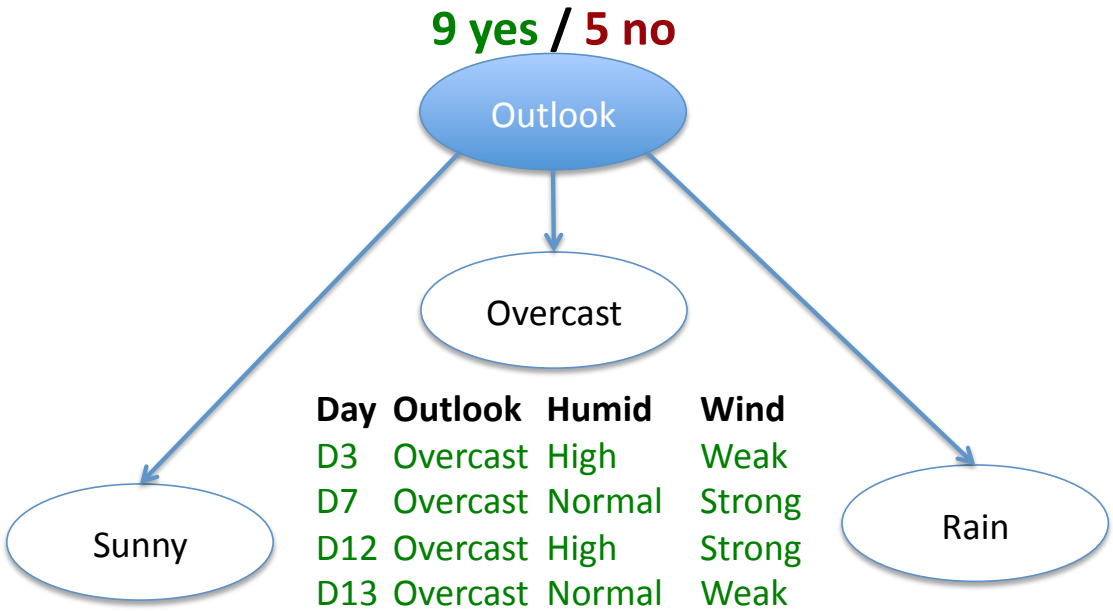
Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

New data:

D15	Rain	High	Weak	?

Example and figures by Victor Lavrenko

Decision Tree Example



Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

4 yes / 0 no
pure subset

Day	Outlook	Humid	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

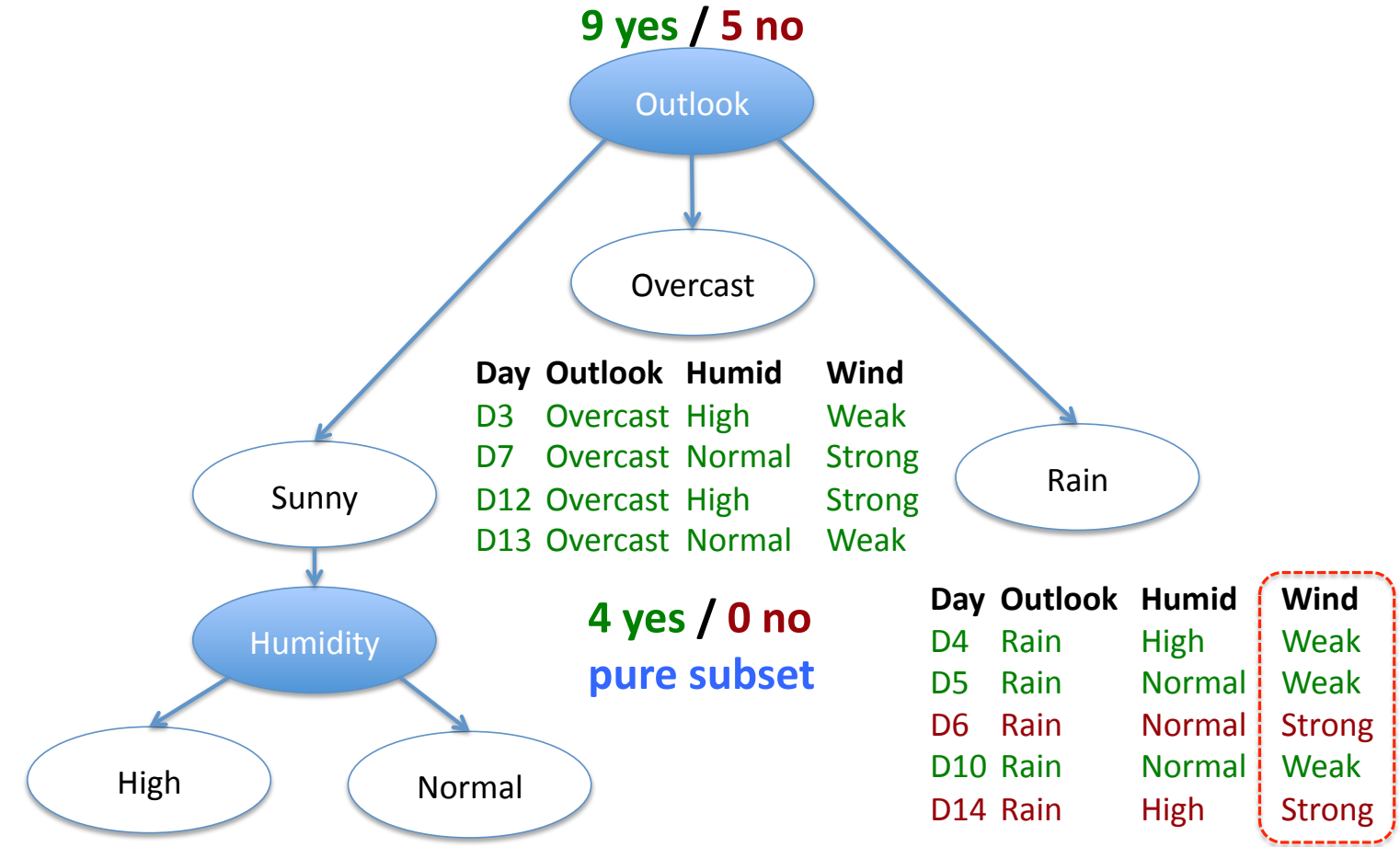
2 yes / 3 no
split further

Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

3 yes / 2 no
split further

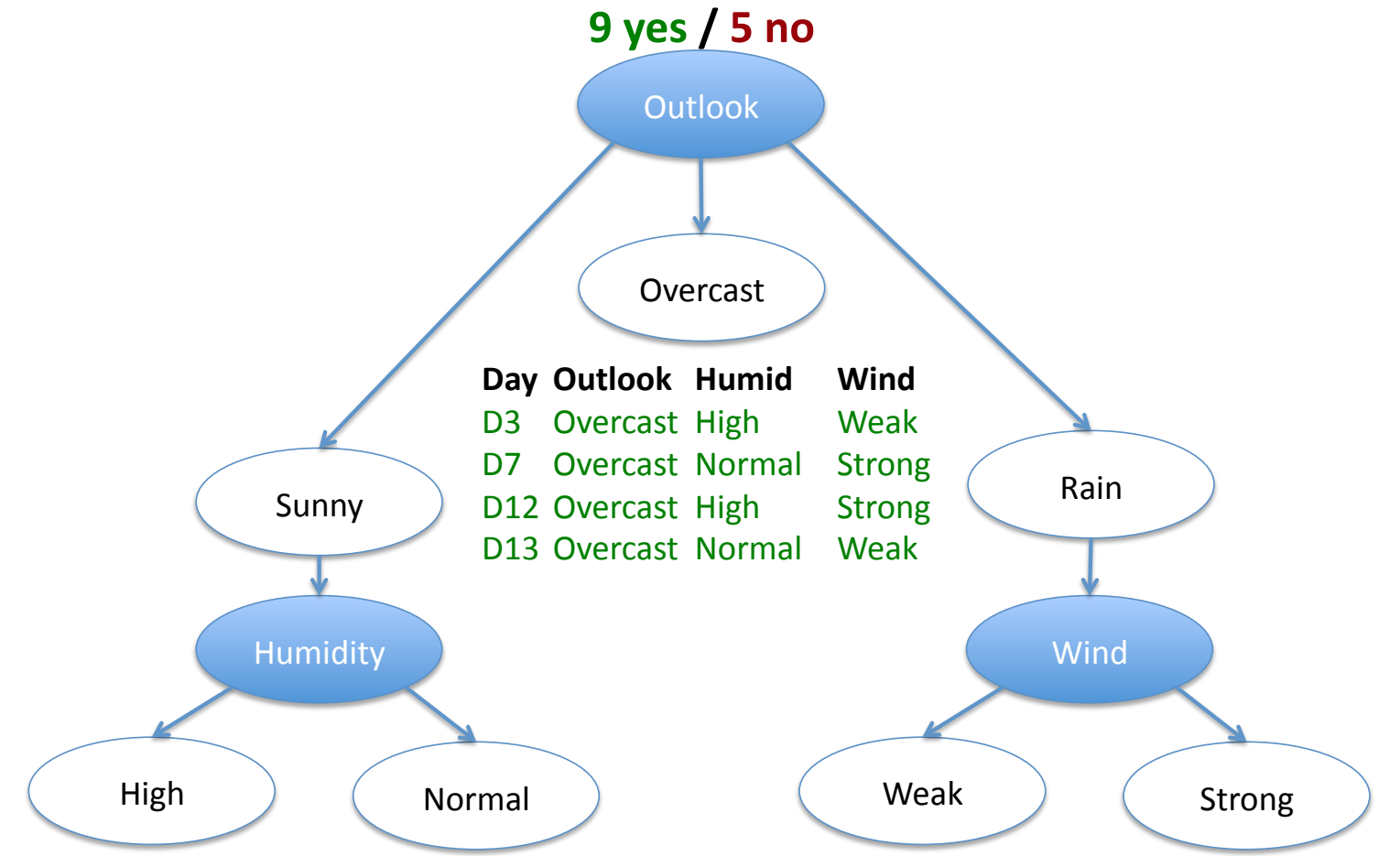
Example and figures by Victor Lavrenko

Decision Tree Example



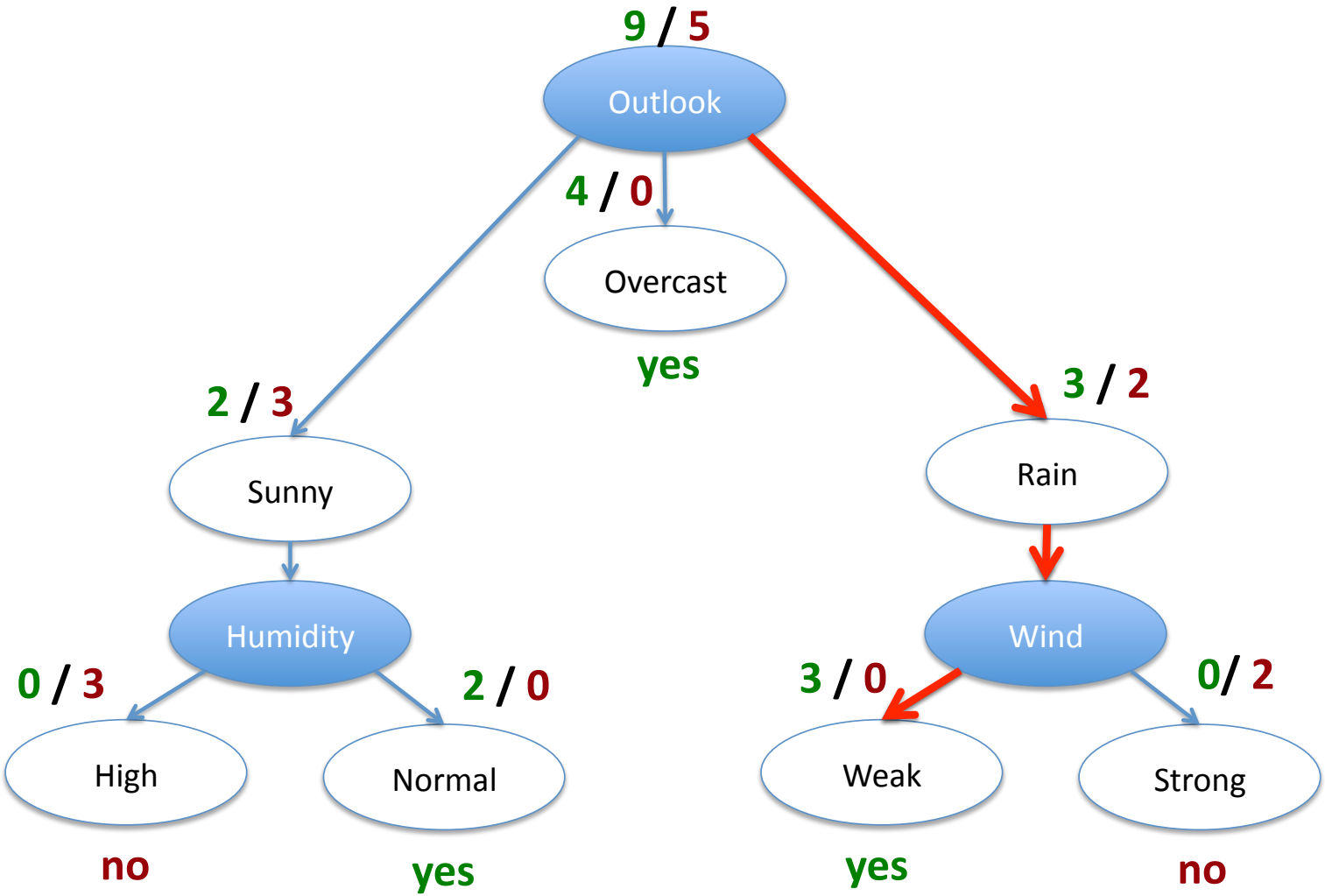
Example and figures by Victor Lavrenko

Decision Tree Example



Example and figures by Victor Lavrenko

Decision Tree Example



New data:

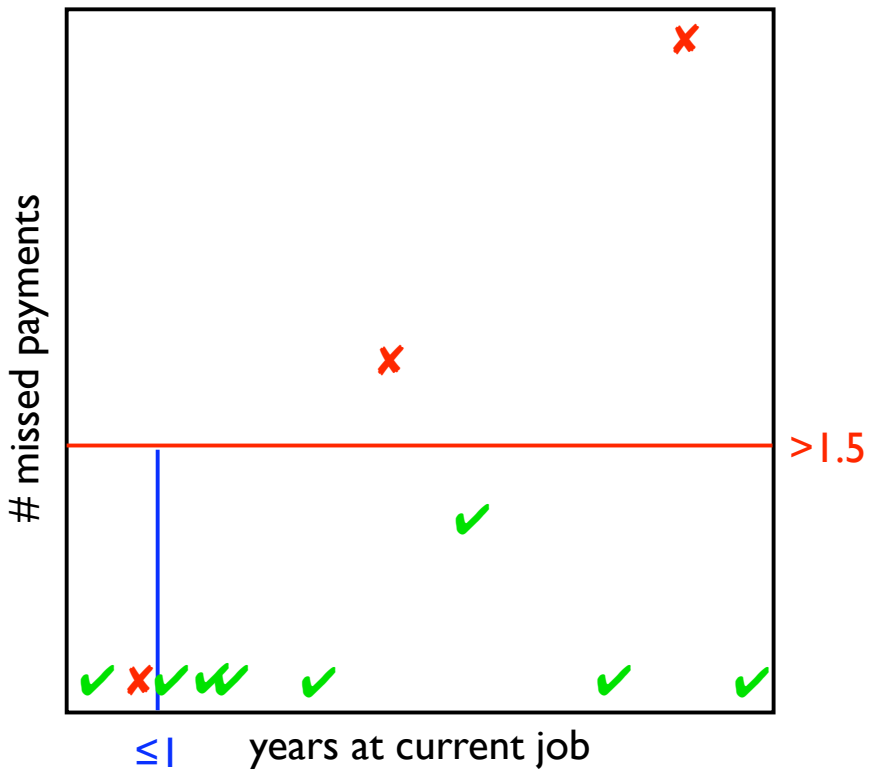
Day	Outlook	Humid	Wind	
D15	Rain	High	Weak	➔ Yes

Example and figures by Victor Lavrenko

Continuous Inputs

Predicting credit risk

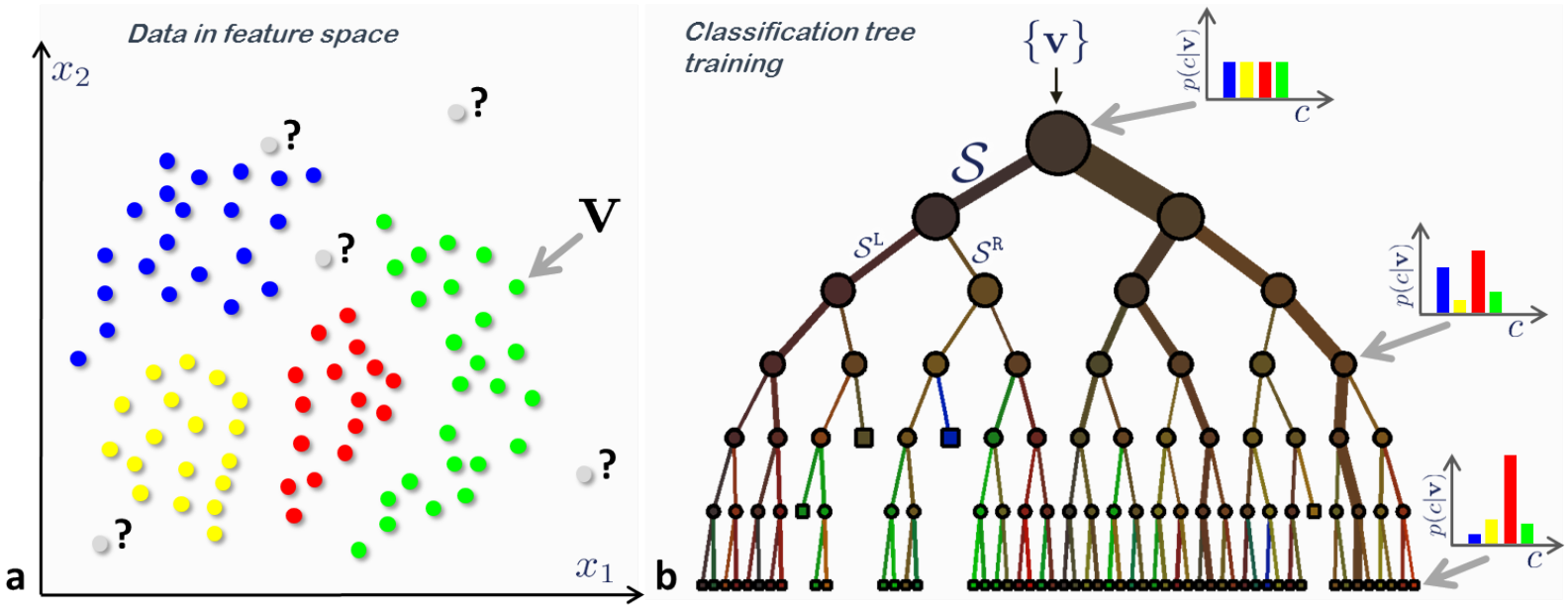
years at current job	# missed payments	defaulted?
7	0	N
0.75	0	Y
3	0	N
9	0	N
4	2	Y
0.25	0	N
5	1	N
8	4	Y
1.0	0	N
1.75	0	N



Example and figure by Michael S. Lewicki

Output Empirical Distribution

- We can output whole distribution instead of just the prevalent class



Criminisi et al.: *Decision Forests for Classification, Regression, Density Estimation, Manifold Learning and Semi-Supervised Learning*, 2011

Why Greedy Learning?

- How many distinct decision trees with n Boolean attributes for binary classification?
  - at least as many as boolean functions of n attributes
  - $=$ number of distinct truth tables with 2^n rows: 2^{2^n}
  - For 6 Boolean attributes at least 18,446,744,073,709,551,616 trees!
- Learning is NP-complete: [Hyafil and Rivest 1976]
- $\Rightarrow$ we need heuristics $\Rightarrow$ greedy approach
- Recursively choose the "most important" attribute to find a small tree consistent with the training data
- Split points:
  - nominal attribute: try all possibilities
  - ordinal/continuous attribute: try attribute values based on all training data samples or their subset

Entropy

- Measure of unpredictability used by information theory
- Lossless compression $\Rightarrow$ compressed information has more entropy per character
- Entropy of a random variable Y with possible values $\{y_1, y_2, \ldots, y_n\}$:

$$H(Y) = -\sum_{i=1}^{n} \mathbb{P}(Y = y_i) \log_2 \mathbb{P}(Y = y_i)$$

- Tossing a fair coin:

$$\begin{aligned} H(Y) &= -\mathbb{P}(\text{head}) \log_2 \mathbb{P}(\text{head}) - \mathbb{P}(\text{tail}) \log_2 \mathbb{P}(\text{tail}) \\ &= -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1 \text{ bit} \end{aligned}$$

- Two-heads coin: $H(Y) = 0$ bits

Entropy Example

H_1 = 4.76 bits/char

1. *Zero-order approximation.* (The symbols are independent and equiprobable.)

   XFOML RXKHRJFFJUJ ZLPWCFWKCYJ

   FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

H_2 = 4.03 bits/char

2. *First-order approximation.* (The symbols are independent. Frequency of letters matches English text.)

   OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI

   ALHENHTTPA OOBTTVA NAH BRL

   •
   •
   •

H_2 = 2.8 bits/char

5. *Fourth-order approximation.* (The frequency of quadruplets of letters matches English text. Each letter depends on the previous three letters. This sentence is from Lucky's book, *Silicon Dreams* [183].)

   THE GENERATED JOB PROVIDUAL BETTER TRAND THE

   DISPLAYED CODE, ABOVERY UPONDULTS WELL THE

   CODERST IN THESTICAL IT DO HOCK BOTHE MERG.

Figure by Michael S. Lewicki

Entropy of Classification

- We can use entropy as an impurity measure
- $\mathbb{P}(\text{def} = \text{Y}) = \frac{3}{10}$
- $\mathbb{P}(\text{def} = \text{N}) = \frac{7}{10}$
- Entropy:

$$H(\text{def}) = -\sum_{y \in \{\text{Y},\text{N}\}}^{n} \mathbb{P}(\text{def} = y) \log_2 \mathbb{P}(\text{def} = y) =$$

$$= -\frac{3}{10} \log_2 \frac{3}{10} - \frac{7}{10} \log_2 \frac{7}{10} \approx 0.8813$$

- We get zero entropy for a pure dataset

Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

Example and figure by Michael S. Lewicki

Conditional Entropy

- Conditional entropy is the amount of uncertainty remaining about Y after X is known
- We first define the *specific conditional entropy*:

$$H(Y|X=x) = -\sum_{y} \mathbb{P}(Y=y|X=x) \log \mathbb{P}(Y=y|X=x)$$

- The *conditional entropy* is then:

$$H(Y|X) = \mathbb{E}_x(H(Y|X=x)) = \sum_{x} \mathbb{P}(X=x)\, H(Y|X=x)$$

Mutual Information (Information Gain)

- Mutual information is a symmetric measure:

$$I(Y;X) = \sum_y \sum_x \mathbb{P}(X = x, Y = y) \log \left(\frac{\mathbb{P}(X = x, Y = y)}{\mathbb{P}(X = x)\mathbb{P}(Y = y)} \right)$$
$$= H(X) - H(X|Y) = H(Y) - H(Y|X)$$

- It quantifies an information gain for a random variable when other random variable gets involved

Maximizing Information Gain

- Consider the splitting attribute j and the split point s, we get a pair of half-planes $R_L(j, s)$ and $R_R(j, s)$
- We seek for j and s maximizing the information gain:

$$I_s(Y; X_j) = H(Y) - H_s(Y|X_j)$$

where for ordinal attributes we have:

$$H_s(Y|X_j) = \mathbb{P}(X_j \leq s)\, H(Y|X_j \leq s) + \mathbb{P}(X_j > s)\, H(Y|X_j > s)$$

while for the nominal attributes:

$$H_s(Y|X_j) = \mathbb{P}(X_j = s)\, H(Y|X_j = s) + \mathbb{P}(X_j \neq s)\, H(Y|X_j \neq s)$$

Maximizing Information Gain Example

- $H(\text{def}|\text{2yrs} = \text{Y}) = -\frac{1}{4}\log_2\frac{1}{4} - \frac{3}{4}\log_2\frac{3}{4} \approx 0.8113$
- $H(\text{def}|\text{2yrs} = \text{N}) = -\frac{2}{6}\log_2\frac{2}{6} - \frac{4}{6}\log_2\frac{4}{6} \approx 0.9183$
- $H(\text{def}|\text{2yrs}) \approx \frac{4}{10} \times 0.8113 + \frac{6}{10} \times 0.9183 \approx 0.8755$

- $H(\text{def}|\text{miss} = \text{Y}) = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \approx 0.9183$
- $H(\text{def}|\text{miss} = \text{N}) = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} \approx 0.5917$
- $H(\text{def}|\text{miss}) \approx \frac{3}{10} \times 0.9183 + \frac{7}{10} \times 0.5917 \approx 0.69$

- $H(\text{def}) - H(\text{def}|\text{2yrs}) \approx 0.8813 - 0.8755 = 0.0058$
- $H(\text{def}) - H(\text{def}|\text{miss}) \approx 0.8813 - 0.69 = \mathbf{0.1913}$

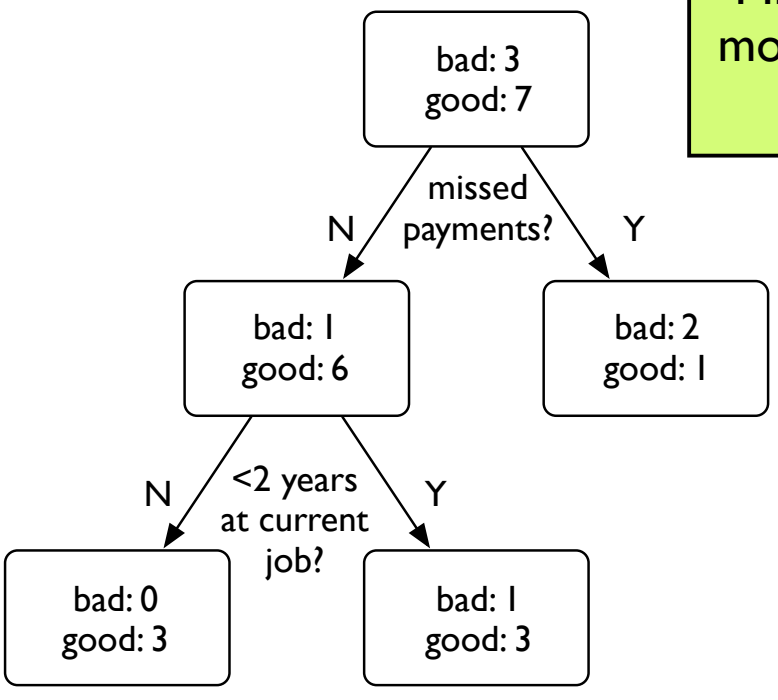
Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

Example and figure by Michael S. Lewicki

Maximizing Information Gain Example (contd.)

- $I(\text{def}; \text{2yrs}) = H(\text{def}) - H(\text{def}|\text{2yrs}) \approx 0.0058$
- $I(\text{def}; \text{miss}) = H(\text{def}) - H(\text{def}|\text{miss}) \approx \mathbf{0.1913}$



Missed payments are the most informative attribute about defaulting.

Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

Example and figure by Michael S. Lewicki

Tree Learning Algorithm

BUILD-TREE(S)

1 $i = \text{IMPURITY}(S)$
2 $\hat{i}, \hat{j}, \hat{s}, \hat{S}_L, \hat{S}_R = 0, 0, 0, \emptyset, \emptyset$ // current best kept in these
3 **for** $j \in \{1, \ldots, p\}$ // iterate over attributes
4 **for** $s \in \text{SPLIT-POINTS}(S, j)$ // iterate over all split points
5 $S_L, S_R = \text{SPLIT}(S, j, s)$
6 $i_L = \text{IMPURITY}(S_L)$
7 $i_R = \text{IMPURITY}(S_R)$
8 **if** $i_L + i_R < \hat{i}$ and $|S_L| > 0$ and $|S_R| > 0$
9 $\hat{i}, \hat{j}, \hat{s}, \hat{S}_L, \hat{S}_R = (i_L + i_R), j, s, S_L, S_R$
10 **if** $\hat{i} < i$
11 $N_L = \text{BUILD-TREE}(\hat{S}_L)$
12 $N_R = \text{BUILD-TREE}(\hat{S}_R)$
13 **return** $\text{DECISION-NODE}(\hat{j}, \hat{s}, N_L, N_R)$
14 **else return** $\text{LEAF-NODE}(S)$

When to Stop Splitting?

- Split while impurity decreases
  - no assurance of zero impurity at leafs (e.g., for two samples $\boldsymbol{x}_i = \boldsymbol{x}_j$, $y_i \neq y_j$)
  - when all leaves are pure then tree becomes a lookup table $\Rightarrow$ **overfitting!**
- Check generalization error using validation set, stop when validation error starts to increase
- Use threshold β: stop splitting when maximum possible gain drops below β
  - uses all training data unlike the previous approach
  - leaves at different depths: adapts to complexity in input distribution
  - drawback: hard to set β

When to Stop Splitting? (contd.)

- Stop when the node represents less than n (e.g., 10) samples or less then a percentage of total samples (e.g., 5%)
- Trade complexity for test accuracy, minimize:

$$\alpha \cdot \mathit{size} + \sum_{l \in \mathit{leaves}} \text{IMPURITY}(S_l)$$

  where $\alpha > 0$ and *size* can be the number of tree nodes or links
- Check statistical significance of the impurity reduction, e.g. using chi-squared test:
  - when a candidate split does not reduce the impurity *significantly*, splitting is stopped
  - does a candidate split significantly differ from a random split?

Pruning

- The previously stopping methods may stop tree growth prematurely due to the greedy approach
- Pruning: reduce a fully grown tree starting at leaves
- All pairs of sibling leaf nodes are considered for *merge*
- Any pair whose elimination yields a satisfactory (small) increase in impurity is eliminated
- Computationally costly but preferred for smaller problems
- Rule pruning: simplifying rules defined by conjunction of tests on a way from the root to leaves $\Rightarrow$ better interpretability

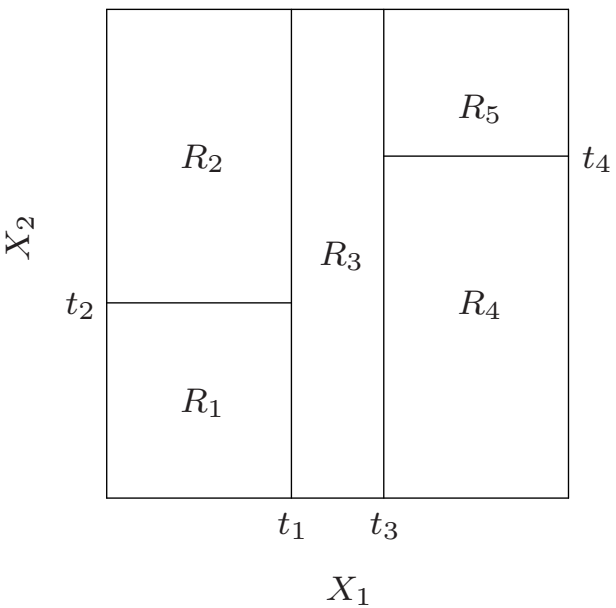
Decision Tree Methods

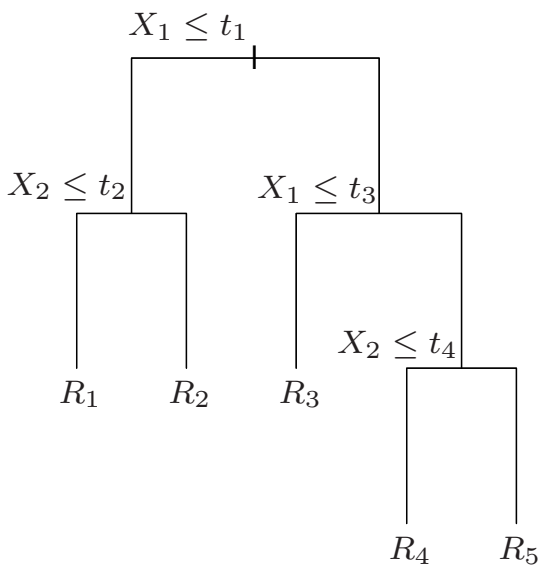
- CART (Classification And Regression Trees): described in previous slides (some extensions beyond were shown)
- ID3 (Interactive Dichotomizer 3)
  - Quinlan: *Induction of Decision Trees*, 1986
  - nominal (unordered inputs), uses binning for continuous variables
  - multiway
  - depth $\leq$ number of input variables
  - no pruning originally
- C4.5 (Quinlan)
  - multiway for nominal data
  - pruning based on statistical significance tests
  - missing features different that CART p. 412
  - rule-based prunning
- C5.0 (Quinlan): patented, faster, less memory, boosting support
- CHAID (CHi-square Automatic Interaction Detector)

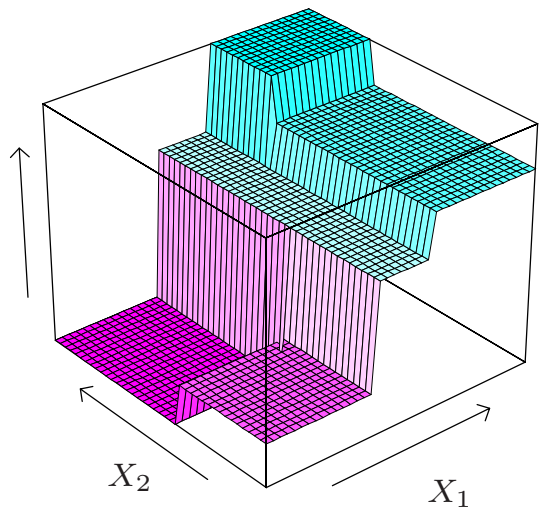
Regression Trees

Regression Trees

- Nodes at the same level correspond to mutually exclusive subsets of the original training data as well as mutually exclusive subsets of the input space $\mathcal{X}$
- Inner node further splits its subset







Regression Trees (contd.)

- Training set: $\mathcal{T}^m = \{(\boldsymbol{x}_i, y_i) \mid i = 1, \ldots, m\}$, $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$
- Input space split into regions defined in leaves: $R_r,\ r \in \{1, \ldots, M\}$
- We can model *region responses* by constants $c_r,\ r \in \{1, \ldots, M\}$ but other possibilities, e.g., linear regression are possible
- Prediction:

$$h(\boldsymbol{x}) = \sum_{r=1}^{M} c_r \mathbb{I}\{\boldsymbol{x} \in R_r\}$$

- For sum of squares *loss function* $\sum_{i=1}^{m}(y_i - h(\boldsymbol{x}_i))^2$ we set the responses to be the averages over regions:

$$\hat{c}_r = \frac{1}{|S_r|} \sum_{(\boldsymbol{x}_i, y_i) \in S_r} y_i$$

where $S_r = \{(\boldsymbol{x}_i, y_i) : (\boldsymbol{x}_i, y_i) \in \mathcal{T}^m \wedge \boldsymbol{x}_i \in R_r\}$

Ensembles

Ensemble Methods

- Inspired in *Wisdom of the crowd*
  - (weighted) averaging or taking majority vote
  - cancelling effect of noise of individual opinions,
  - examples: politics, trial by jury (vs. trial by judge), sports (figure skating, gymnastics), Wikipedia, Quora, Stack Overflow, ...
- Learning and aggregating multiple predictors
- Ensemble may be built using single or different types of predictors

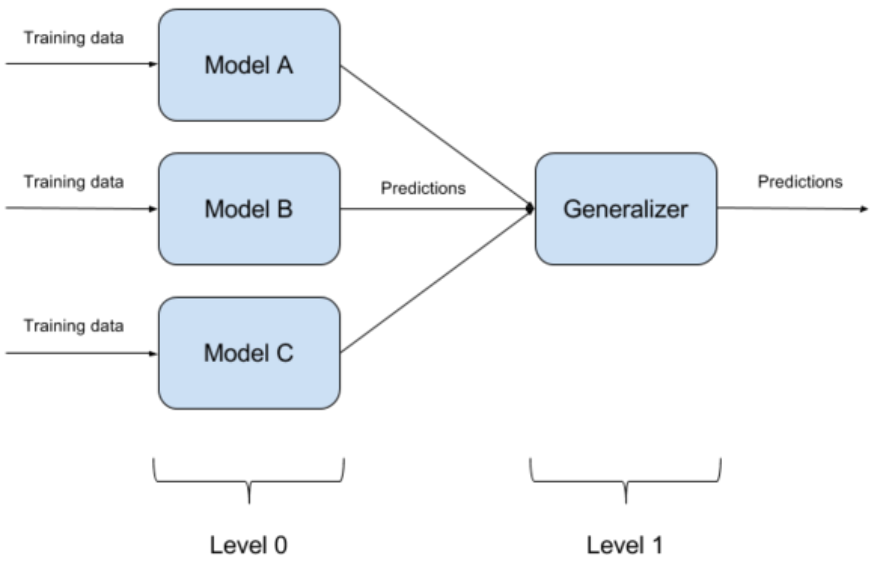
Wikimedia Commons

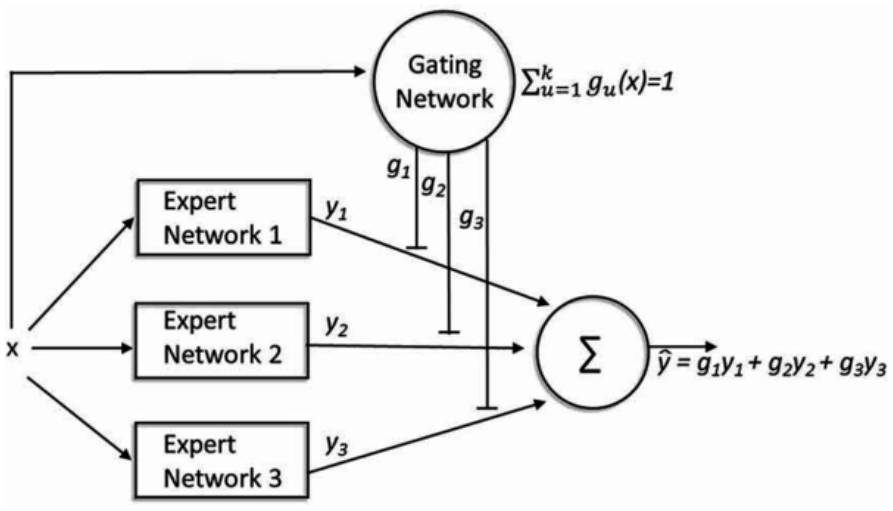
Ensembling Approaches

- Bagging (Bootstrap AGGregatING):
  - sample different training sets from the original training set
  - train *high variance low bias* predictors based on these sets and average them
  - exploits independence between predictors
- Boosting:
  - sequentially train *low variance high bias* predictors
  - subsequent predictors learn to fix the mistakes of the previous ones
  - exploits dependence between learners

Stacking and Mixture of Experts

- Combine *base-learners* with *meta-learner*





https://www.commonlounge.com/discussion/9331c0d004704e89bd4d1da08fd7c7bc

Prediction Problem: Expected Risk and Error Decomposition

Expected risk for data generated by $p(x, y)$:

$$R(h) = \mathbb{E}_{(x,y)\sim p}\Big[\ell(y, h(x))\Big]$$

- The best attainable (Bayes) risk is $R^* = \inf_{h\in\mathcal{Y}^{\mathcal{X}}} R(h)$
- The best predictor in $\mathcal{H}$ is $h_{\mathcal{H}} \in \operatorname{Argmin}_{h\in\mathcal{H}} R(h)$
- The predictor $h_m = A(\mathcal{T}^m)$ learned from $\mathcal{T}^m$ has risk $R(h_m)$

Excess error measures deviation of the learned predictor from the best one:

$$\underbrace{\Big(R(h_m) - R^*\Big)}_{\text{excess error}} = \underbrace{\Big(R(h_m) - R(h_{\mathcal{H}})\Big)}_{\text{estimation error}} + \underbrace{\Big(R(h_{\mathcal{H}}) - R^*\Big)}_{\text{approximation error}}$$

Predictors Averaged over Datasets

- Let us define a model averaged over all possible datasets:

$$g_m(x) = \mathbb{E}_{\mathcal{T}^m}\Big[h_m(x)\Big]$$

- Unlike individual h_m models, g_m has an access to the whole $p(x, y)$
- Note: in general $g_m \neq h_{\mathcal{H}}$ due to training algorithm A involved in h_m.
- Also: g_m can't be actually evaluated for infinite number of $\mathcal{T}^m$ datasets

Bias-Variance Decomposition for Regression

- Consider a regression problem with data generated as follows:

$$y = h^*(x) + \epsilon$$

  where ϵ is noise: $\mathbb{E}[\epsilon] = 0$ and $\mathrm{Var}(\epsilon) = \sigma^2$, e.g., $\epsilon \sim \mathcal{N}(0, \sigma^2)$

- Use squared loss:

$$\ell(y, h(x)) = \Big(h(x) - y\Big)^2$$

- The optimal predictor $h^*(x)$ has a nonzero risk (for $\sigma^2 > 0$):

$$R^* = \mathbb{E}_{x,y}\left[\Big(h^*(x) - y\Big)^2\right] = \mathbb{E}_{\epsilon}\left[\epsilon^2\right] = \mathrm{Var}(\epsilon) = \sigma^2$$

Bias-Variance Decomposition for Regression 2

- The expected risk for h_m can be decomposed:

$$\begin{aligned}
\mathbb{E}_{\mathcal{T}^m}\Big[R(h_m)\Big] &= \mathbb{E}_{x,y,\mathcal{T}^m}\left[\Big(h_m(x) - y\Big)^2\right] \\
&= \cdots \\
&= \underbrace{\mathbb{E}_{x,\mathcal{T}^m}\left[\Big(h_m(x) - g_m(x)\Big)^2\right]}_{\text{variance}} + \\
&+ \underbrace{\mathbb{E}_{x}\left[\Big(g_m(x) - h^*(x)\Big)^2\right]}_{\text{bias}^2} + \underbrace{\sigma^2}_{\text{noise}}
\end{aligned}$$

- The error splits into three terms
  - **variance**: difference of h_m from the averaged predictor g_m,
  - **bias**^{2}: difference of the averaged predictor g_m from the optimal one,
  - **noise**: irreducible determined by data

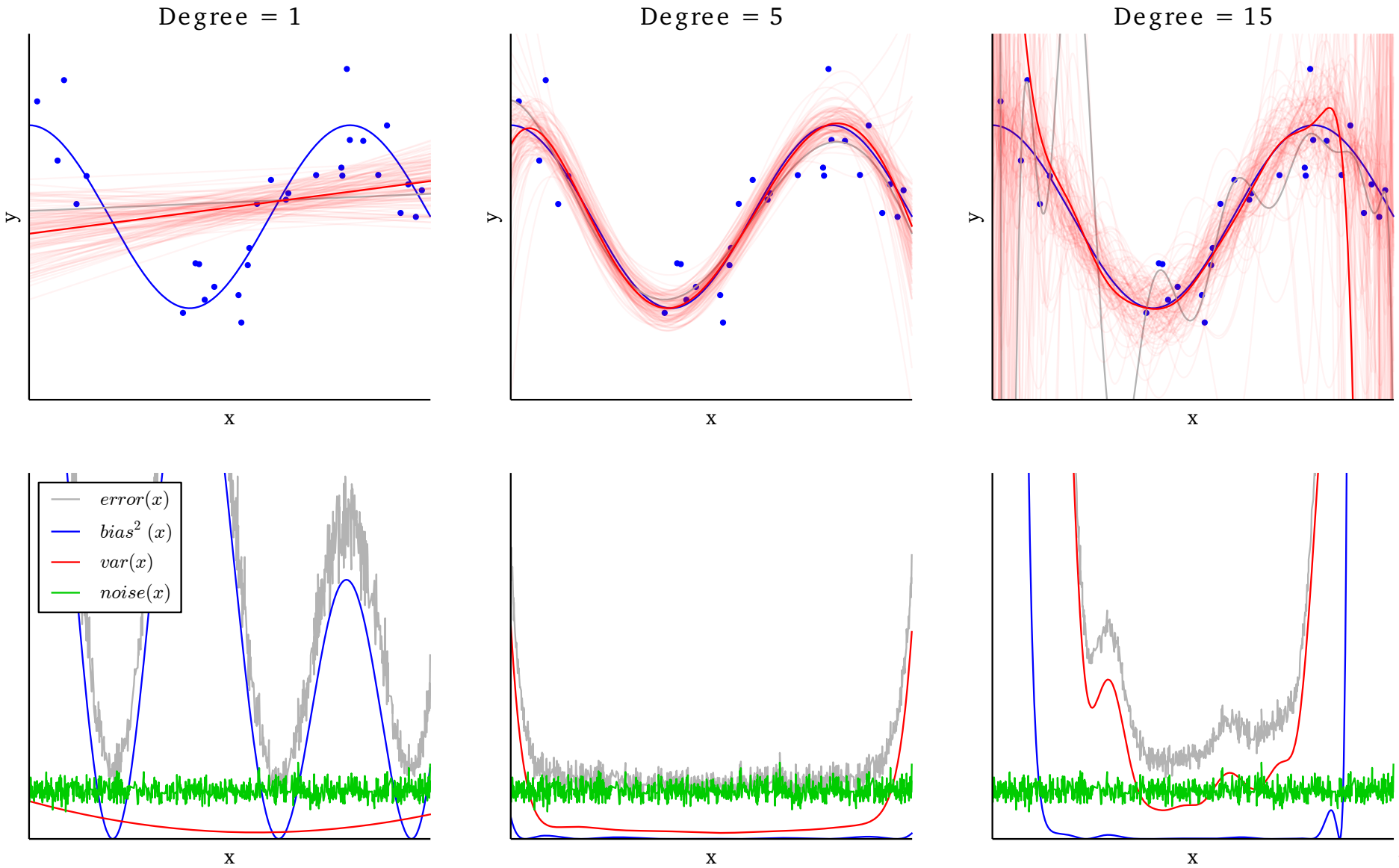
Pointwise Bias-Variance

We can express the bias and variance as function of x by not integrating over in expected values

$$\begin{aligned}
\mathbb{E}_{y|x,\mathcal{T}^m}\Big[\ell(y, h_m(x))\Big] &= \mathbb{E}_{y|x,\mathcal{T}^m}\Big[\Big(h_m(x) - y\Big)^2\Big] \\
&= \underbrace{\mathrm{Var}_{\mathcal{T}^m}\Big(h_m(x)\Big)}_{\text{variance(x)}} + \\
&+ \underbrace{\Big(g_m(x) - h^*(x)\Big)^2}_{\text{bias(x)}^2} + \underbrace{\sigma(x)^2}_{\text{noise}}
\end{aligned}$$

Bias-Variance: Example

- Polynomial regression with a varying degree of polynomial



Gilles Louppe: *Understanding Random Forests: From Theory to Practice*, 2014

Bias and Variance of Decision Trees

- Small changes of training data lead to big differences in final trees
- Decision trees grown deep enough have typically:
  - low bias
  - high variance

  $\Rightarrow$ overfitting
- Idea: *average multiple models* to reduce variance while (happily) not increasing bias much

Averaging Models

- Define *bagging model* b as an average of K *component models*:

$$b(x) = \frac{1}{K} \sum_{i=1}^{K} h_m^{(i)}(x)$$

  trained using a set of i.i.d. datasets of size m: $\mathcal{D}^m = \{\mathcal{T}_1^m, \ldots, \mathcal{T}_K^m\}$ so $h_m^{(1)}(x)$ is trained using $\mathcal{T}_1^m$, $h_m^{(2)}(x)$ using $\mathcal{T}_2^m$, etc.

- Note that $b(x)$ approximates the *averaging model*:

$$g_m(x) = \mathbb{E}_{\mathcal{T}^m}\Big[h_m(x)\Big]$$

Averaging Models: Bias

- Bias remains unchanged when compared to a single model:

$$
\begin{aligned}
\text{bias}(x)^2 &= \Big(g_m(x) - h^*(x)\Big)^2 \\
&= \Big(\mathbb{E}_{\mathcal{D}^m}\Big[b(x)\Big] - h^*(x)\Big)^2 \\
&= \left(\mathbb{E}_{\mathcal{D}^m}\left[\frac{1}{K}\sum_{i=1}^{K} h_m^{(i)}(x)\right] - h^*(x)\right)^2 \\
&= \left(\frac{1}{K}\sum_{i=1}^{K} \mathbb{E}_{\mathcal{T}_i^m}\Big[h_m^{(i)}(x)\Big] - h^*(x)\right)^2 \\
&= \Big(\mathbb{E}_{\mathcal{T}^m}\Big[h_m(x)\Big] - h^*(x)\Big)^2
\end{aligned}
$$

where $\mathbb{E}_{\mathcal{T}^m}\Big[h_m(x)\Big]$ corresponds to the averaging model as defined for any particular component model $h_m(x)$

Averaging Models: Variance

- For uncorrelated component models $h_m^{(i)}(x)$:

$$\mathrm{Var}_{\mathcal{D}^m}\Big(b(x)\Big) = \mathrm{Var}_{\mathcal{D}^m}\left(\frac{1}{K}\sum_{i=1}^{K} h_m^{(i)}(x)\right)$$

$$= \frac{1}{K^2}\sum_{i=1}^{K}\mathrm{Var}_{\mathcal{T}_i^m}\Big(h_m^{(i)}(x)\Big) = \frac{1}{K}\mathrm{Var}_{\mathcal{T}^m}\Big(h_m(x)\Big)$$

  which is a great improvement based on the very strong assumption
- There is no improvement for maximum correlation, i.e., for all component models equal: $h_m^{(i)}(x) = h_m(x)$ for $i = 1, \ldots, K$, we get:

$$\mathrm{Var}_{\mathcal{D}^m}\Big(b(x)\Big) = \mathrm{Var}_{\mathcal{D}^m}\left(\frac{1}{K}\sum_{i=1}^{K} h_m^{(i)}(x)\right) = \mathrm{Var}_{\mathcal{T}^m}\Big(h_m(x)\Big)$$

  $\Rightarrow$ we need to train uncorrelated (diverse) component models while keeping their bias reasonably low

Bootstrapping

- In practice we have only a single training dataset $\mathcal{T}^m$
- Bootstrapping is a method producing datasets $\mathcal{T}_i^m$ for $i = 1, \ldots K$ by sampling $\mathcal{T}^m$ uniformly with *replacement*
- Bootstrap datasets have the same size as the original dataset $|\mathcal{T}_i^m| = |\mathcal{T}^m|$
- $\mathcal{T}_i^m$ is expected to have the fraction $1 - \frac{1}{e} \approx 63.2\%$ of unique samples from $\mathcal{T}^m$, others are duplicates

Bagging

- Bagging = Bootstrap AGGregating [Breiman 1994]:
  1. Use bootstrapping to generate K datasets
  2. Train a model $h_m^{(i)}(x)$ on each dataset $\mathcal{T}_i^m$
  3. Average the models getting the bagging model $b(x)$
- When decision trees are used as the models $\Rightarrow$ random forests
- Low bias is achieved by growing the trees to maximal depth
- Trees are decorrelated by:
  - training each tree on a different bootstrap dataset
  - randomization of split attribute selection

Random Forest Algorithm

1. For $i = 1 \ldots K$:
   - 1.1 draw a bootstrap dataset $\mathcal{T}_i^m$ from $\mathcal{T}^m$, $|\mathcal{T}_i^m| = |\mathcal{T}^m| = m$
   - 1.2 grow a tree $h_m^{(i)}$ using $\mathcal{T}_i^m$ by recursively repeating the following, until the minimum node size n_{min} is reached:
     - 1.2.1 select k attributes at random from the p attributes
     - 1.2.2 pick the best attribute and split-point among the k
     - 1.2.3 split the node into two daughter nodes
2. Output ensemble of trees $b(x)$ averaging $h_m^{(i)}(x)$ (regression) or selecting a majority vote (classification)

- Node size n_{min} is the number of the training dataset samples associated with the node, limits tree depth

Random Forest Summary

- Easy to use method: robust w.r.t. parameter settings (K, node size)
- While *statistical consistency* is proven for decision trees (both regression and classification) we have only proofs for simplified versions of random forests [Breiman, 1984]
- Related methods: boosted trees

Boosting

- Sequentially train weak learners/predictors *low variance high bias*
- Subsequent predictors fix the mistakes of the previous ones reducing bias
- Methods discussed here:
  - Forward Stagewise Additive Modeling
  - Gradient Boosting Machine
  - Gradient Boosted Trees

Forward Stagewise Additive Modeling (FSAM)

1. Initialize $f_0(x) = 0$
2. For $k = 1$ to K:
   2.1 Find
   $$(\beta_k, \theta_k) = \underset{\beta,\theta}{\operatorname{argmin}} \sum_{i=1}^{m} \ell\Big(y_i, f_{k-1}(x_i) + \beta b(x_i; \theta)\Big)$$
   where $b(x_i; \theta_k)$ is the basis function and β_k the corresponding coefficient
   2.2 Set $f_k(x) = f_{k-1}(x) + \beta_k b(x; \theta_k)$
3. Return $h_m(x) = f_K(x)$

FSAM and Gradient Descent

- FSAM update looks very similar to the gradient descent one:

$$f_k(x) = f_{k-1}(x) + \beta_k b(x; \theta_k)$$

- Just think of
  - $\beta_k \approx$ step size (learning rate)
  - $b(x_i; \theta_k) \approx$ the negative of gradient

FSAM for Squared Loss

- Again consider regression with the squared loss:

$$\ell(y, f(x)) = (y - f(x))^2$$

- For FSAM we get:

$$\begin{aligned}\ell(y_i, f_k(x_i)) &= \ell(y_i, f_{k-1}(x_i) + \beta_k b(x_i; \theta_k)) \\ &= (y_i - f_{k-1}(x_i) - \beta_k b(x_i; \theta_k))^2 \\ &= (r_{ik} - \beta_k b(x_i; \theta_k))^2\end{aligned}$$

  where $r_{ik} = y_i - f_{k-1}(x_i)$ is the *residual* of the current model for the i-th sample

- The task of FSAM is to fit the model $\beta_k b(x_i; \theta_k)$ to match the residuals
- The method is sometimes called the *least-squares boosting*

Gradient Boosting for Regression

- In case of regression with squared loss we minimize:

$$\mathcal{L} = \sum_{i=1}^{m} \ell(y_i, f(x_i)) = \sum_{i=1}^{m} \frac{1}{2}(y_i - f(x_i))^2,$$

  which is same as minimization of the empirical risk
- We can treat $f(x_1), f(x_2), \ldots, f(x_m)$ as parameters and take derivatives:

$$\frac{\partial \mathcal{L}}{\partial f(x_i)} = \frac{\partial \left(\sum_{j=1}^{m} \ell(y_j, f(x_j)) \right)}{\partial f(x_i)} = \frac{\partial \ell(y_i, f(x_i))}{\partial f(x_i)}$$
$$= f(x_i) - y_i = -r_i$$

- The *least-squares boosting* hence takes steps in the negative gradient direction where $r_i = -\frac{\partial \mathcal{L}}{\partial f(x_i)}$
- This approach can be generalized for any differentiable loss function!

Gradient Boosting Machine

1. Initialize $f_0(x) = 0$ or $f_0(x) = \operatorname{argmin}_\gamma \sum_{i=1}^{m} \ell(y_i, \gamma)$
2. For $k = 1$ to K:
   2.1 Compute:
   $$\boldsymbol{g}_k = \left[\frac{\partial \ell(y_i, f_{k-1}(x_i))}{\partial f_{k-1}(x_i)} \right]_{i=1}^{m}$$
   2.2 Fit a regression model $b(\cdot; \theta)$ to $-\boldsymbol{g}_k$ using squared loss:
   $$\theta_k = \operatorname*{argmin}_\theta \sum_{i=1}^{m} [(-\boldsymbol{g}_k)_i - b(x_i; \theta)]^2$$
   2.3 Choose a fixed step size $\beta_k = \beta > 0$ or use line search:
   $$\beta_k = \operatorname*{argmin}_{\beta > 0} \sum_{i=1}^{m} \ell\Big(y_i, f_{k-1}(x_i) + \beta b(x_i; \theta_k)\Big)$$
   2.4 Set $f_k(x) = f_{k-1}(x) + \beta_k b(x; \theta_k)$
3. Return $h_m(x) = f_K(x)$

Multinominal Classification: Gradient Boosting Machine

- Train one GBM per target class
- Use softmax to get probability distribution
- Use multinominal cross-entropy as the loss

Gradient Boosted Trees

- Gradient Boosting Tree is GBM where all weak learners are decision or regression trees
- Use limit on depth/number of leaves/node size for the weak learners $\Rightarrow$ high bias
- Often single-level tree: *decision stump*
- Meta-parameters such as K (number of trees) and β (learning rate) have to be found using cross validation
- Model is built sequentially (unlike random forests)
- Highly optimized algorithms based on Gradient Boosting Trees:
  - XGBoost, LightGBM
  - parallelization, scalability, regularization


AiCENTER

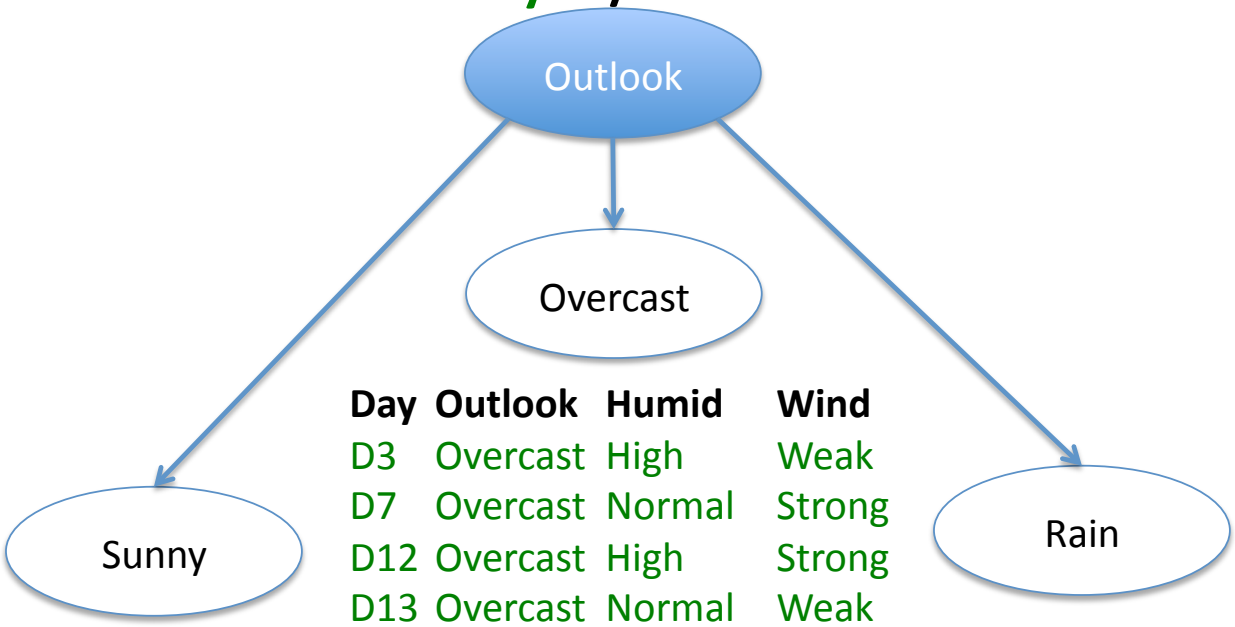
Training examples: **9 yes / 5 no**

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No

New data:

D15	Rain	High	Weak	?

9 yes / 5 no



Day	Outlook	Humid	Wind
D3	Overcast	High	Weak
D7	Overcast	Normal	Strong
D12	Overcast	High	Strong
D13	Overcast	Normal	Weak

4 yes / 0 no

pure subset

Day	Outlook	Humid	Wind
D1	Sunny	High	Weak
D2	Sunny	High	Strong
D8	Sunny	High	Weak
D9	Sunny	Normal	Weak
D11	Sunny	Normal	Strong

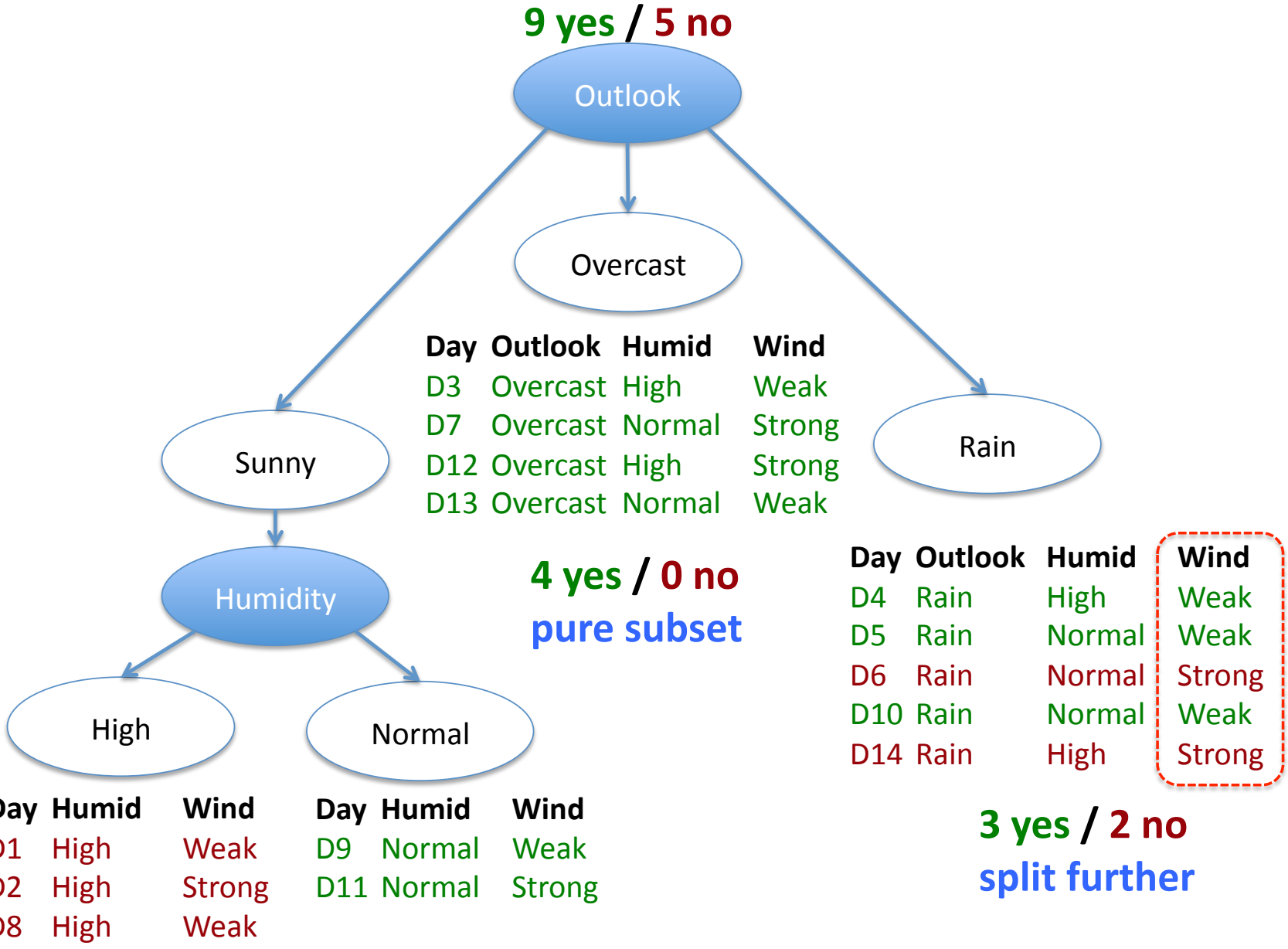
2 yes / 3 no

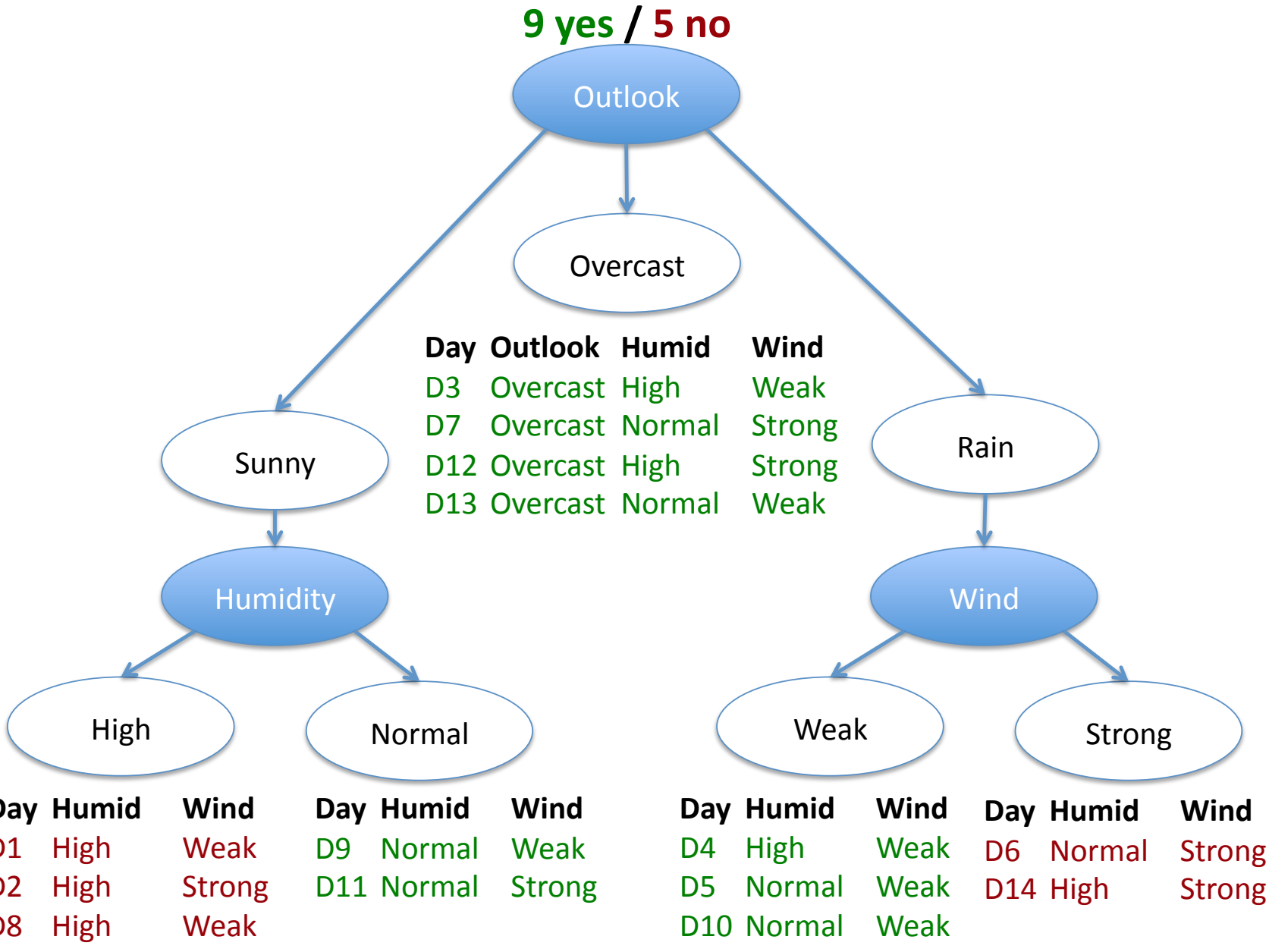
split further

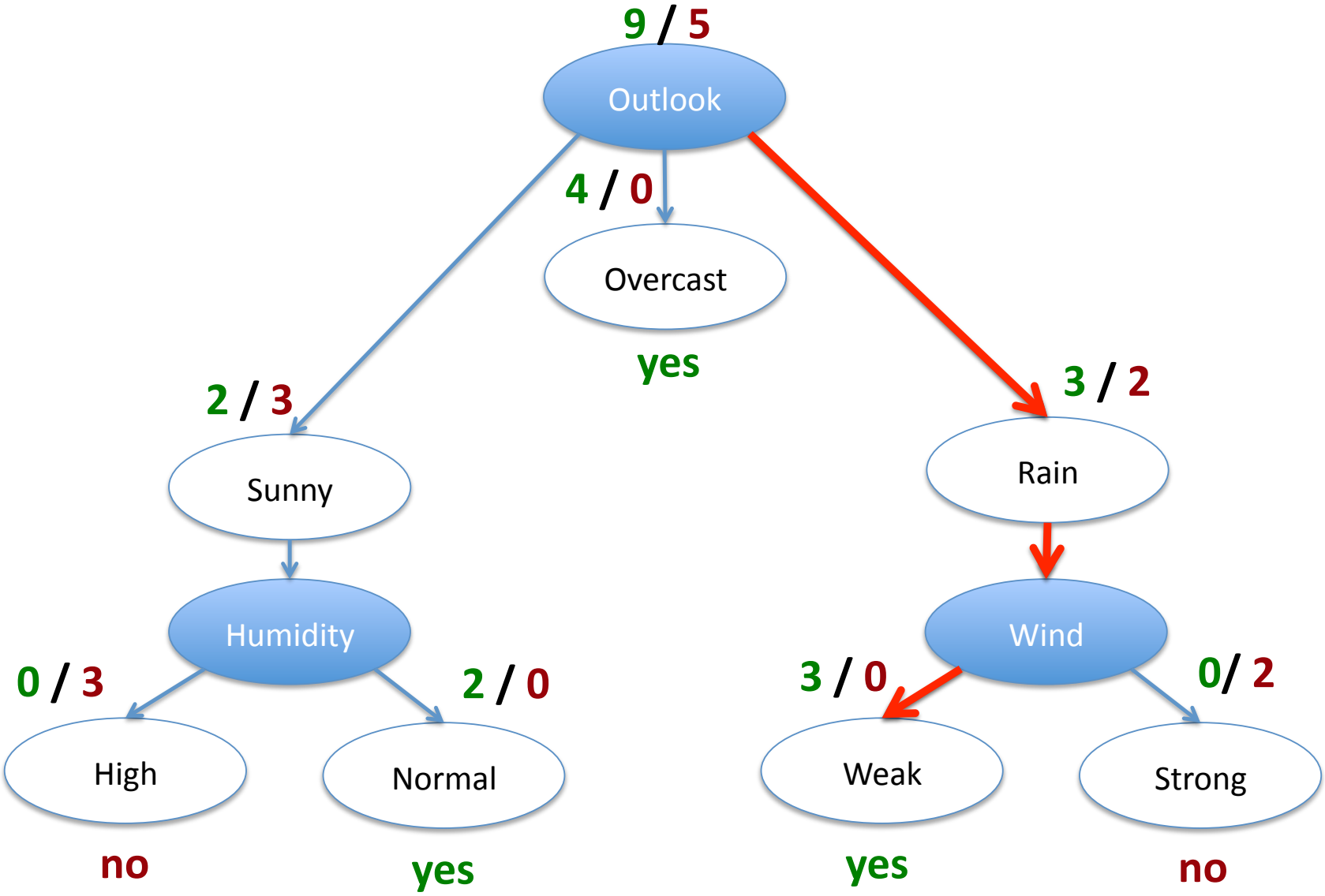
Day	Outlook	Humid	Wind
D4	Rain	High	Weak
D5	Rain	Normal	Weak
D6	Rain	Normal	Strong
D10	Rain	Normal	Weak
D14	Rain	High	Strong

3 yes / 2 no

split further


9 yes / 5 no
Outlook
Overcast
Sunny
Rain
Humidity
High
Normal
Day Outlook Humid Wind
D3 Overcast High Weak
D7 Overcast Normal Strong
D12 Overcast High Strong
D13 Overcast Normal Weak
4 yes / 0 no
pure subset
Day Outlook Humid Wind
D4 Rain High Weak
D5 Rain Normal Weak
D6 Rain Normal Strong
D10 Rain Normal Weak
D14 Rain High Strong
3 yes / 2 no
split further
Day Humid Wind
D1 High Weak
D2 High Strong
D8 High Weak
Day Humid Wind
D9 Normal Weak
D11 Normal Strong


9 yes / 5 no
Outlook
Overcast
Sunny
Rain
Humidity
Wind
High
Normal
Weak
Strong
Day Outlook Humid Wind
D3 Overcast High Weak
D7 Overcast Normal Strong
D12 Overcast High Strong
D13 Overcast Normal Weak
Day Humid Wind
D1 High Weak
D2 High Strong
D8 High Weak
Day Humid Wind
D9 Normal Weak
D11 Normal Strong
Day Humid Wind
D4 High Weak
D5 Normal Weak
D10 Normal Weak
Day Humid Wind
D6 Normal Strong
D14 High Strong

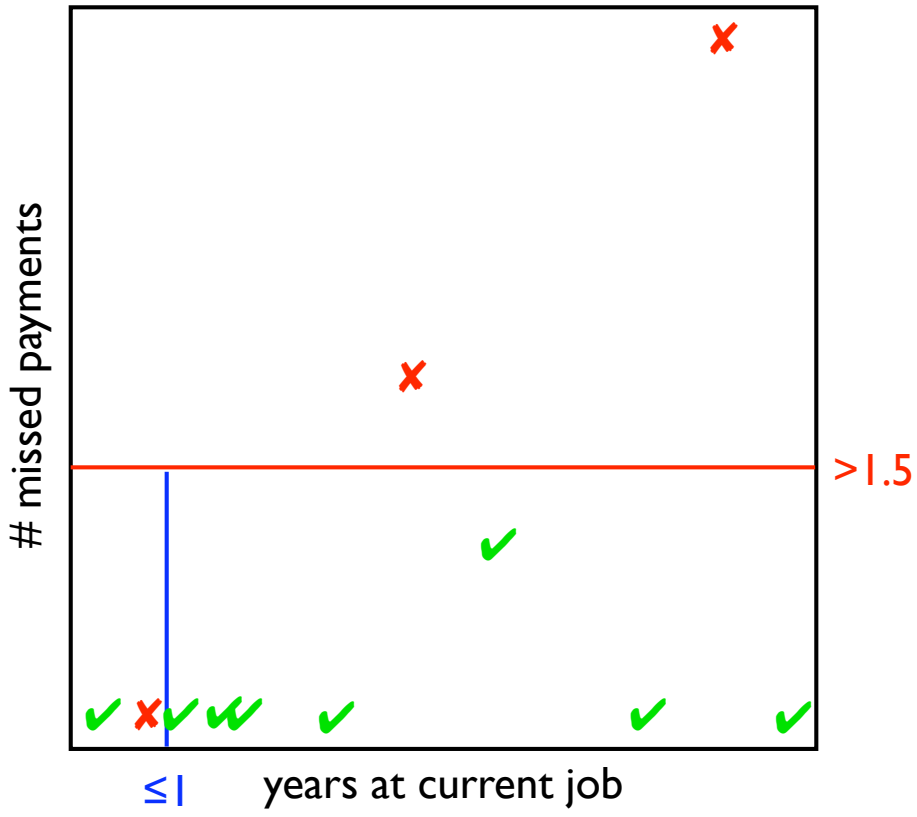


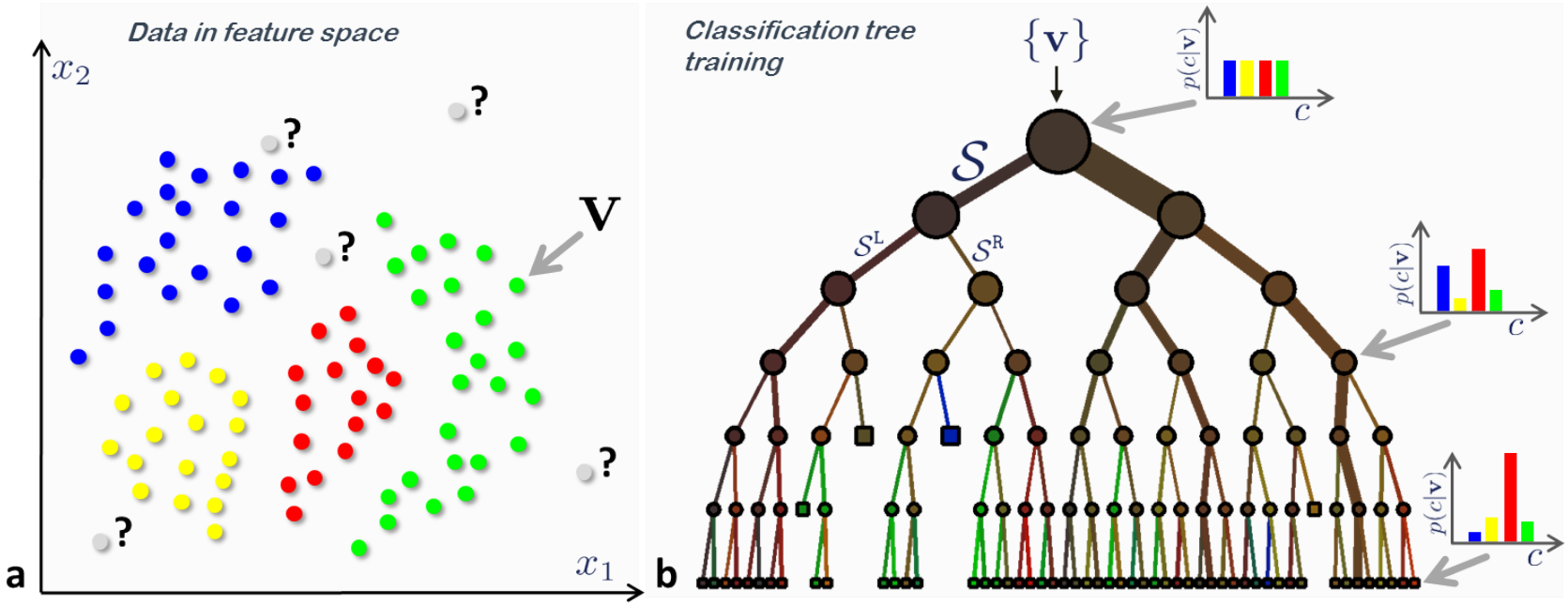
New data:

Day	Outlook	Humid	Wind	
D15	Rain	High	Weak	→ Yes

Predicting credit risk

years at current job	# missed payments	defaulted?
7	0	N
0.75	0	Y
3	0	N
9	0	N
4	2	Y
0.25	0	N
5	1	N
8	4	Y
1.0	0	N
1.75	0	N




Data in feature space
x_2
x_1
?
?
?
?
?
V
a
Classification tree training
$\{\mathbf{v}\}$
$\mathcal{S}$
$\mathcal{S}^L$
$\mathcal{S}^R$
$p(c|\mathbf{v})$
c
b

H_1 = 4.76 bits/char

1. *Zero-order approximation.* (The symbols are independent and equiprobable.)

   XFOML RXKHRJFFJUJ ZLPWCFWKCYJ

   FFJEYVKCQSGXYD QPAAMKBZAACIBZLHJQD

H_2 = 4.03 bits/char

2. *First-order approximation.* (The symbols are independent. Frequency of letters matches English text.)

   OCRO HLI RGWR NMIELWIS EU LL NBNESEBYA TH EEI

   ALHENHTTPA OOBTTVA NAH BRL

   •
   •
   •

H_2 = 2.8 bits/char

5. *Fourth-order approximation.* (The frequency of quadruplets of letters matches English text. Each letter depends on the previous three letters. This sentence is from Lucky's book, *Silicon Dreams* [183].)

   THE GENERATED JOB PROVIDUAL BETTER TRAND THE

   DISPLAYED CODE, ABOVERY UPONDULTS WELL THE

   CODERST IN THESTICAL IT DO HOCK BOTHE MERG.

Predicting credit risk

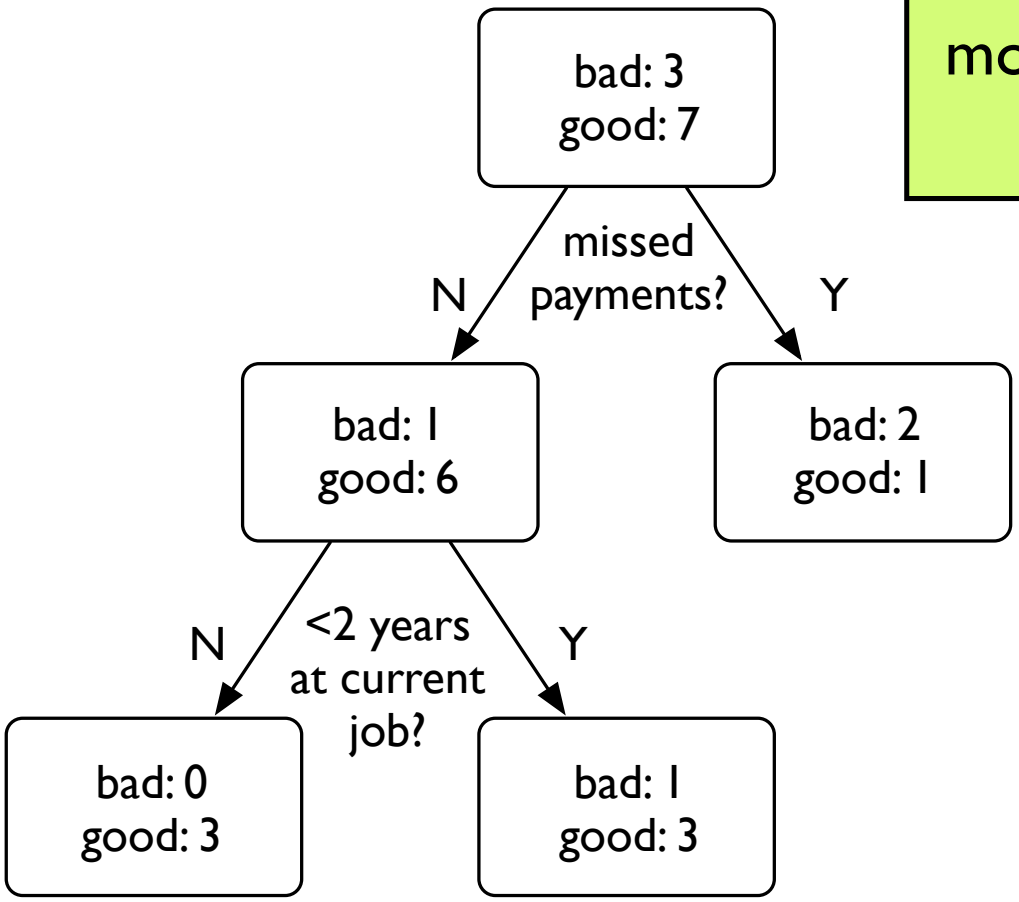
<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

Predicting credit risk

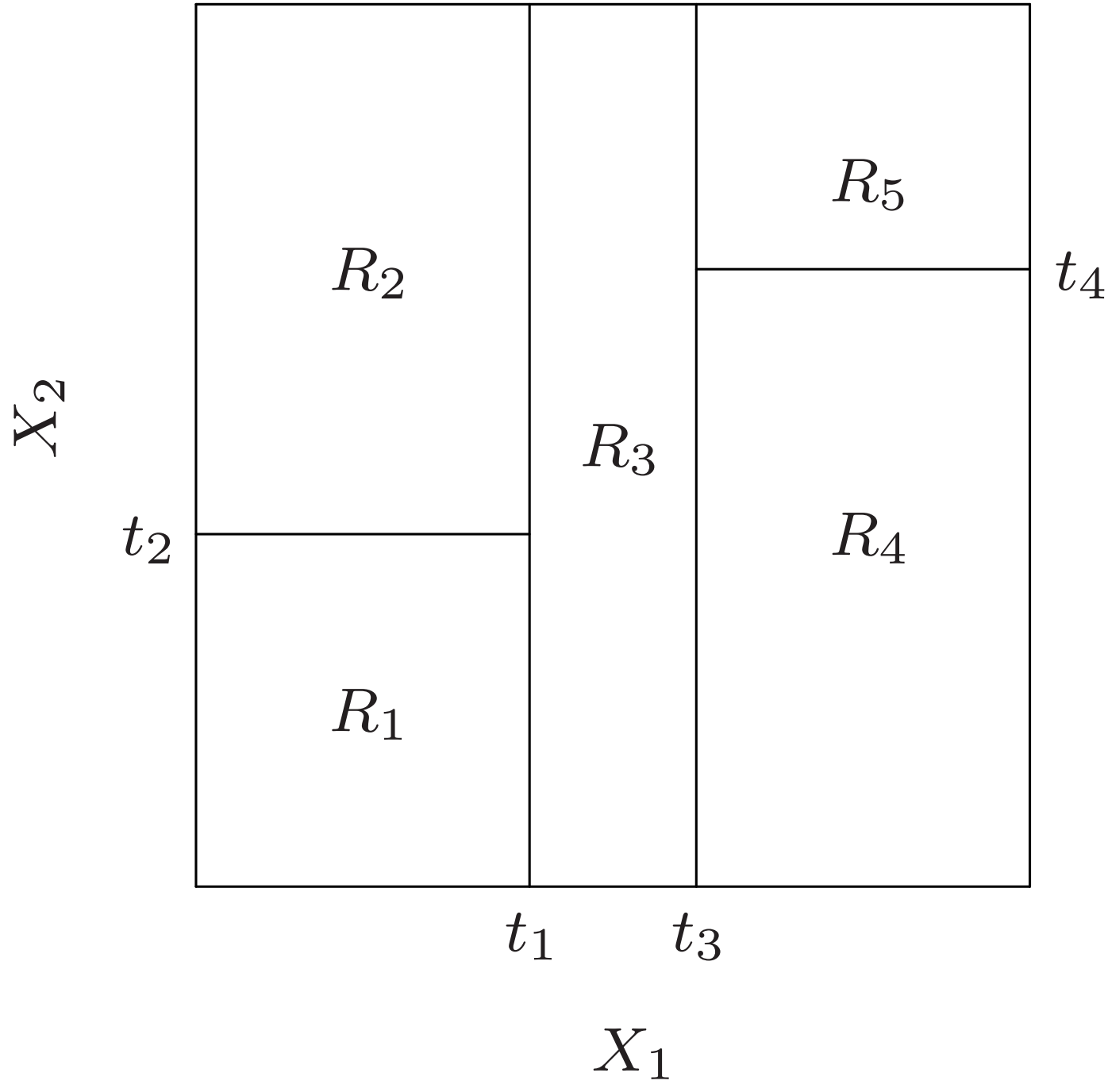
<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N

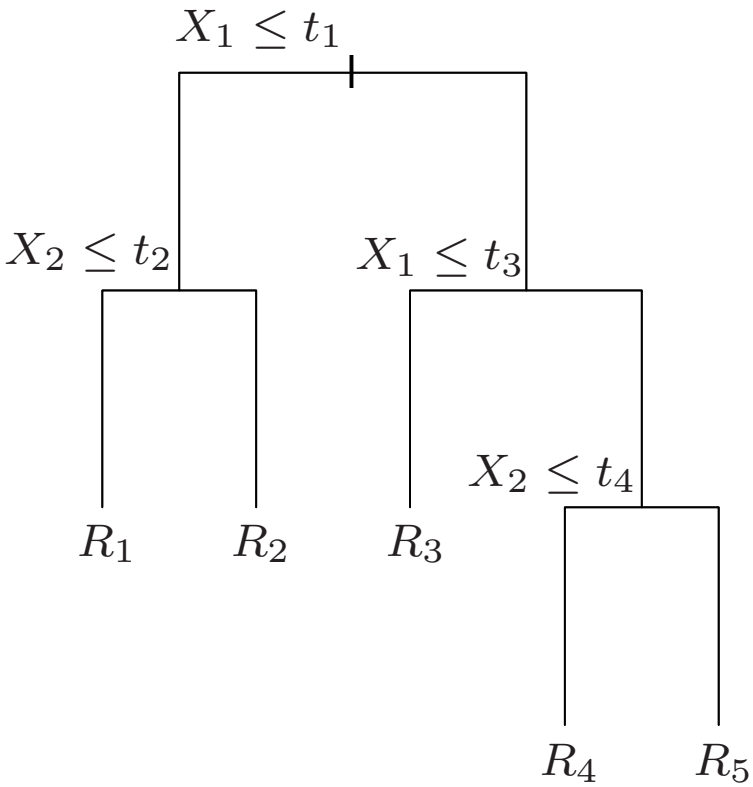
Predicting credit risk

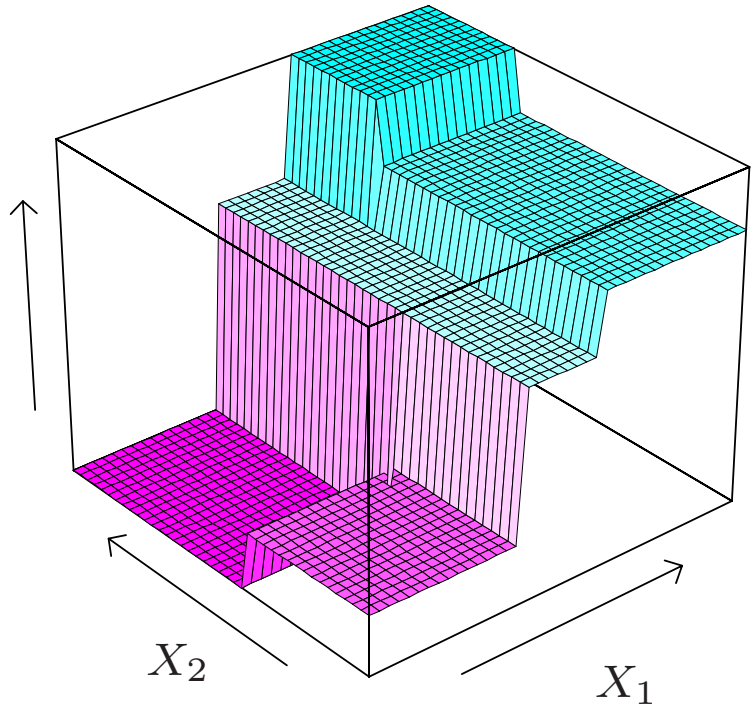
<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N



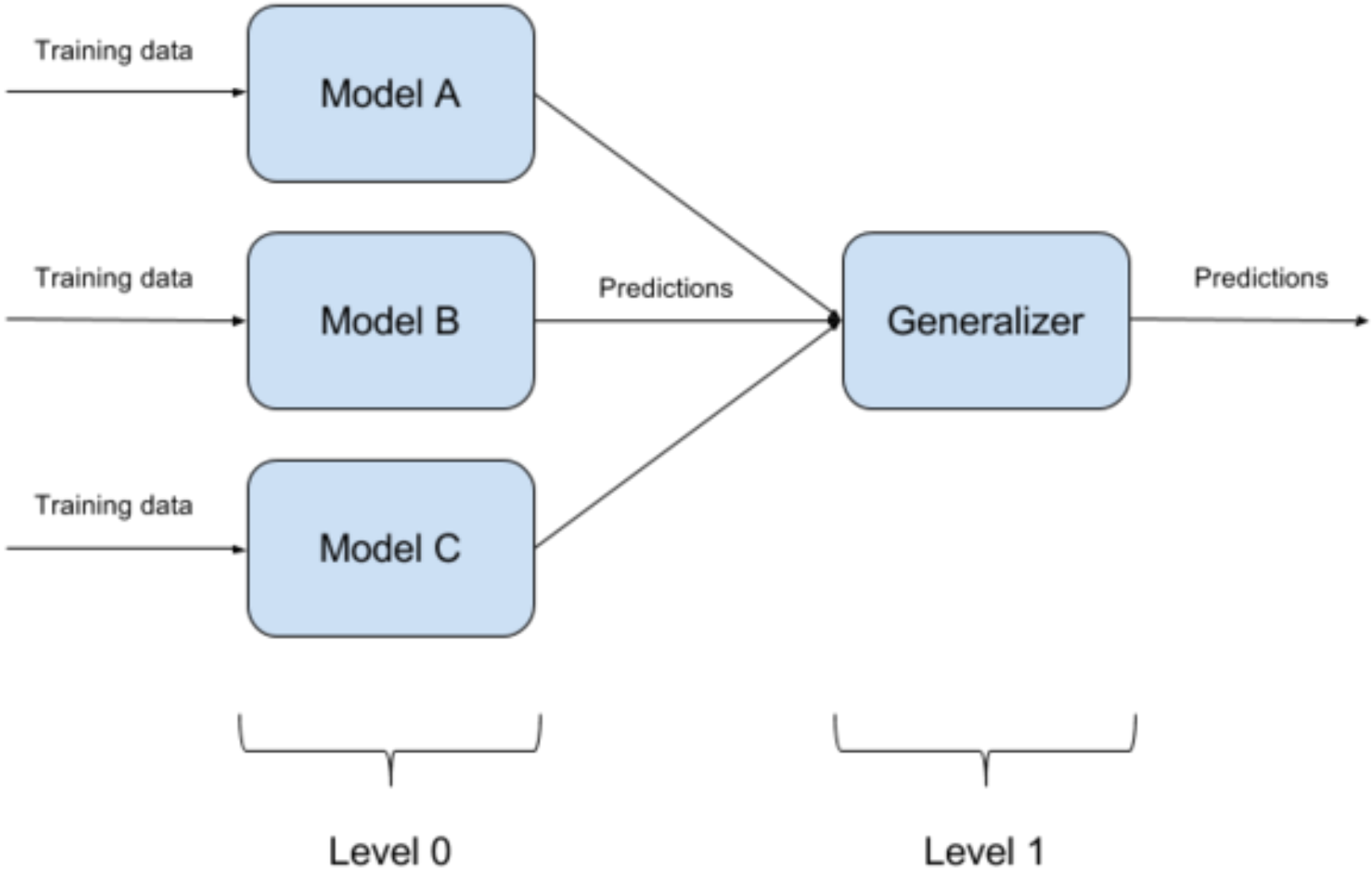
Missed payments are the most informative attribute about defaulting.

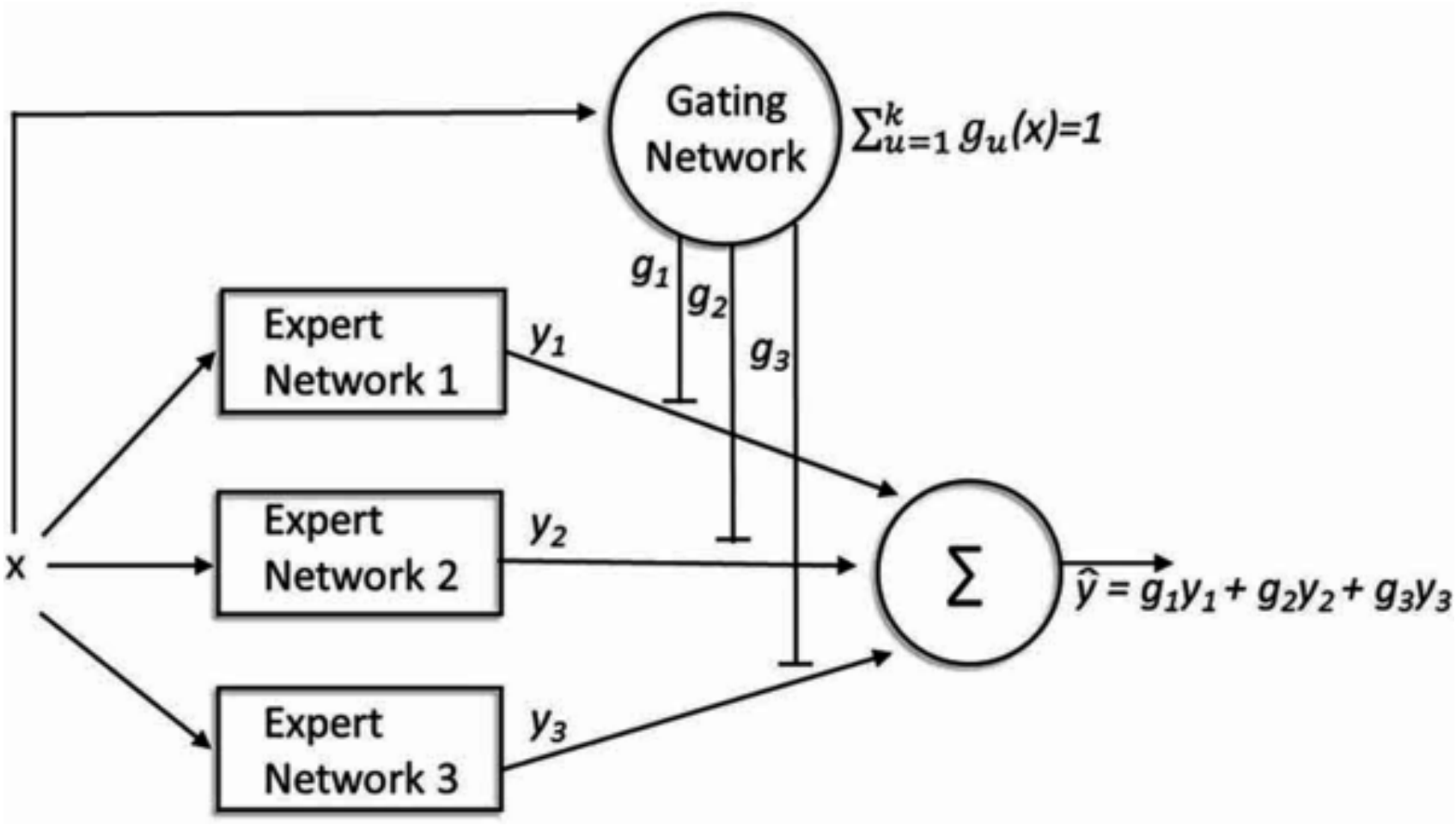

R_5
R_2
t_4
X_2
R_3
t_2
R_4
R_1
t_1
t_3
X_1

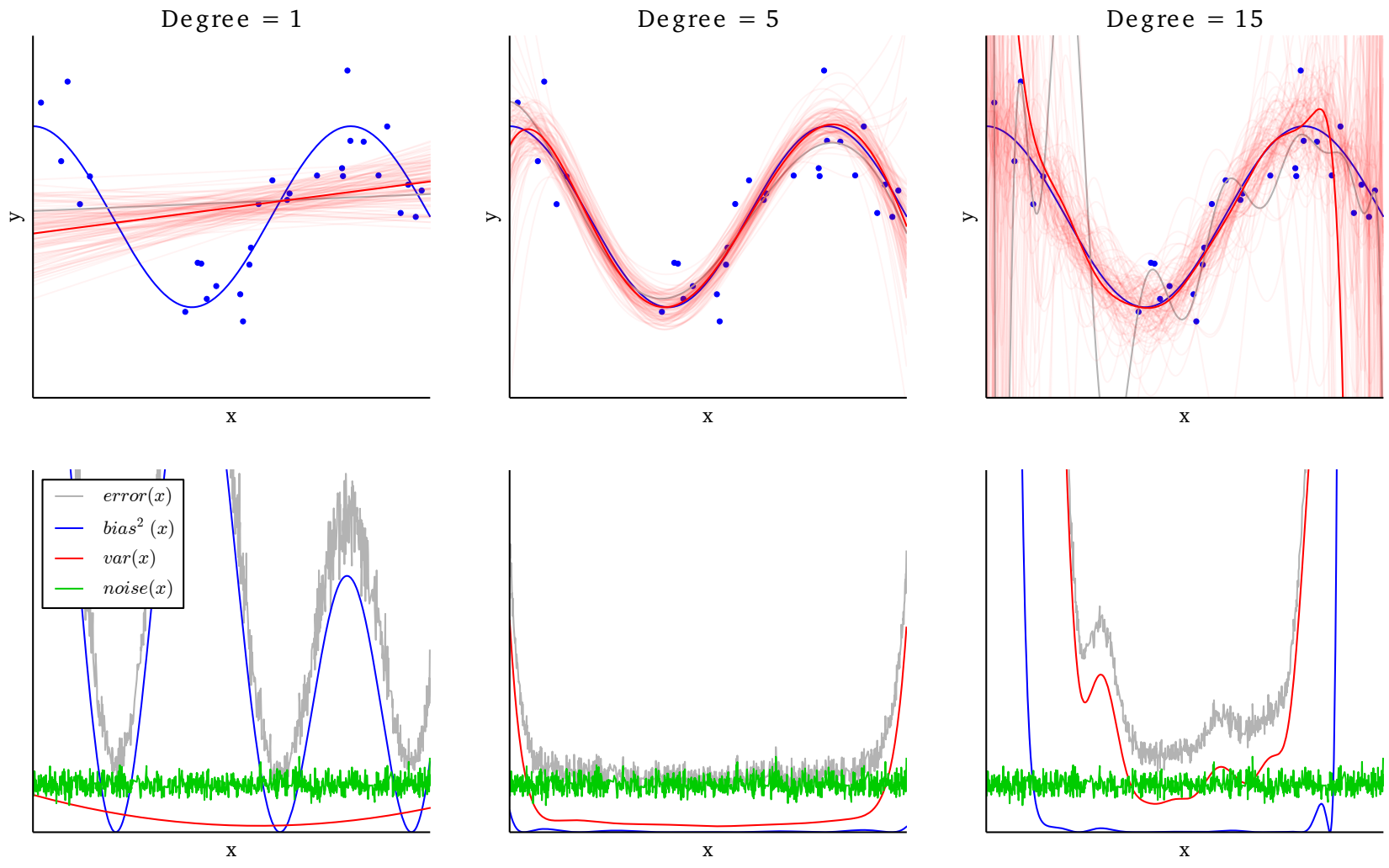

$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
R_1
R_2
R_3
R_4
R_5


X_2
X_1


3
3
5
3
5


Training data
Model A
Training data
Model B
Predictions
Generalizer
Predictions
Training data
Model C
Level 0
Level 1


Gating Network
$\sum_{u=1}^{k} g_u(x)=1$
g_1
g_2
g_3
Expert Network 1
y_1
Expert Network 2
y_2
Expert Network 3
y_3
x
$\sum$
$\hat{y} = g_1y_1 + g_2y_2 + g_3y_3$


Degree = 1
Degree = 5
Degree = 15
y
x
error(x)
bias² (x)
var(x)
noise(x)