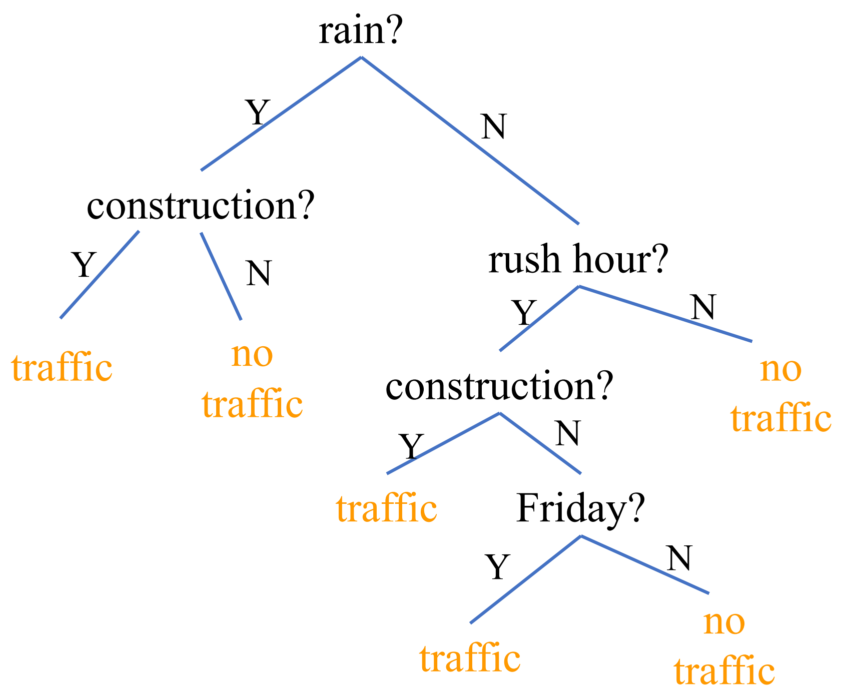
Decision Trees Part 1

Cynthia Rudin

Machine Learning Class

Duke University

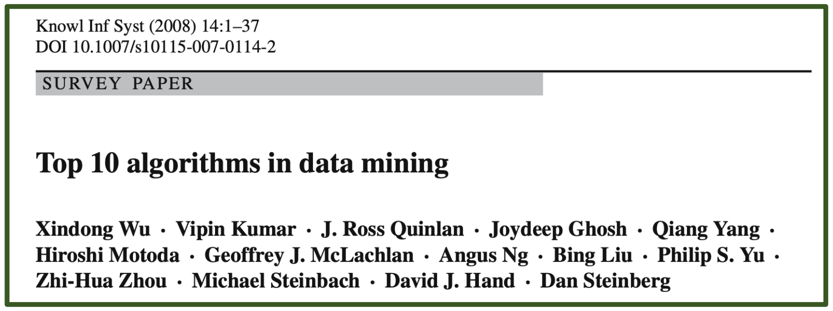
Will I run into traffic on my way home from work?



Why trees?

- interpretable/intuitive, popular in medical applications because they mimic the way people like to reason
- model discrete outcomes nicely
- powerful, nonlinear, can be as complex as you need them
- C4.5 and CART – both in "top 10" from 2008

(Quinlan, 1993) (Breiman et al., 1984)


Knowl Inf Syst (2008) 14:1–37
DOI 10.1007/s10115-007-0114-2

SURVEY PAPER

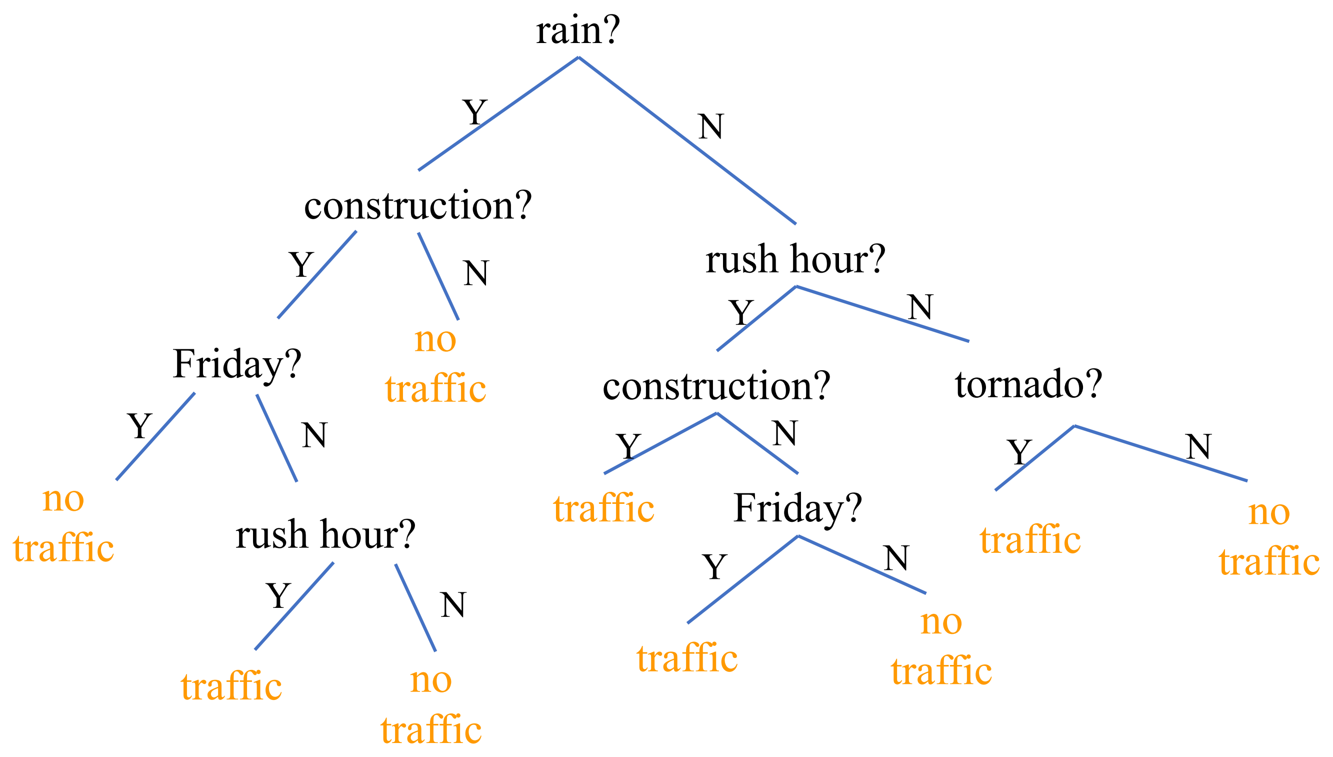
Top 10 algorithms in data mining

Xindong Wu · Vipin Kumar · J. Ross Quinlan · Joydeep Ghosh · Qiang Yang · Hiroshi Motoda · Geoffrey J. McLachlan · Angus Ng · Bing Liu · Philip S. Yu · Zhi-Hua Zhou · Michael Steinbach · David J. Hand · Dan Steinberg

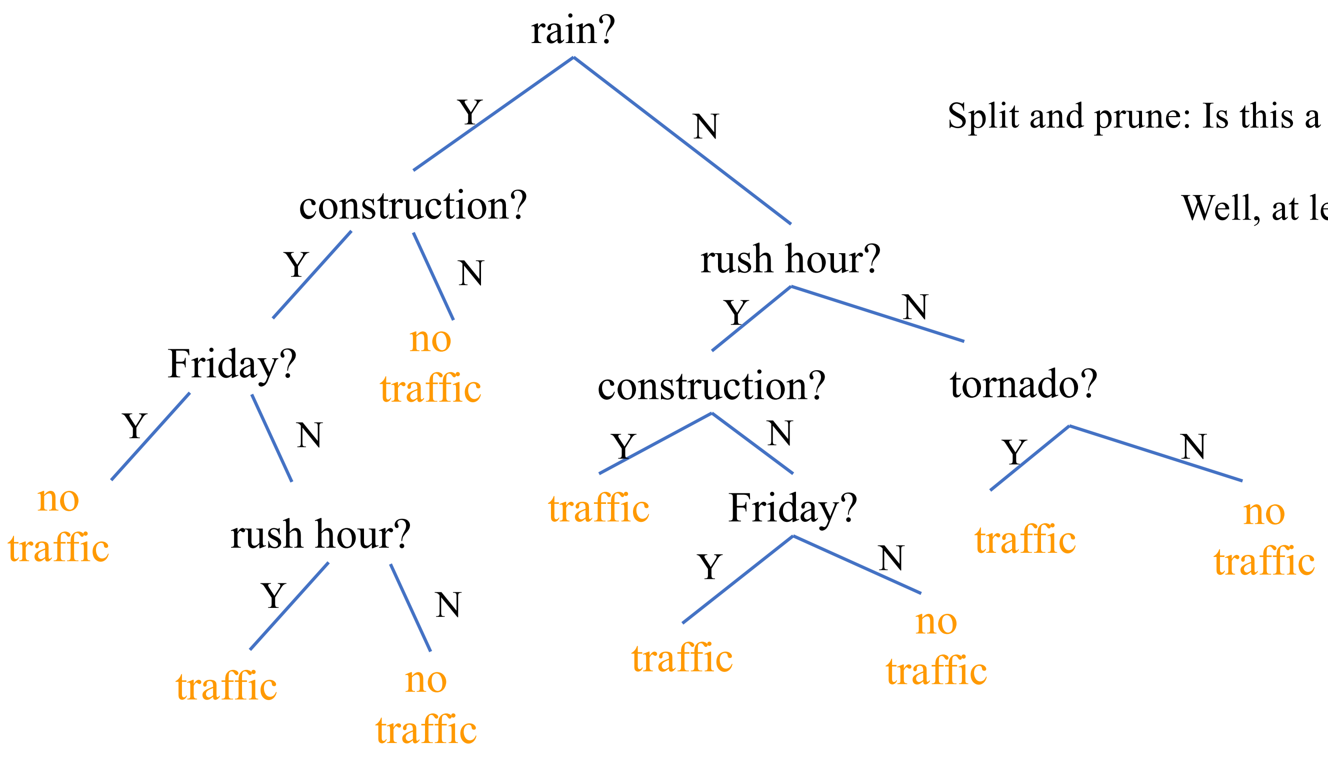
Some real examples (from Russell & Norvig, Mitchell)

- BP's GasOIL system for separating gas and oil on offshore platforms: C4.5 replaced a hand-designed rule system with 2500 rules. It saved BP millions. (1986)
- Learning to fly a Cessna on a flight simulator by watching human experts fly the simulator (1992)

Will I run into traffic on my way home from work?



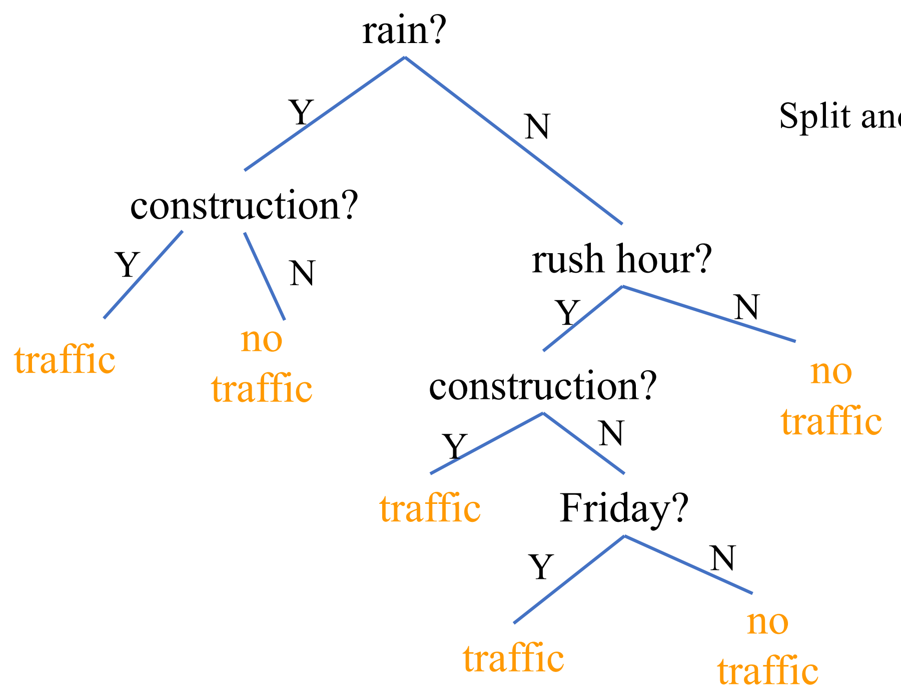
Will I run into traffic on my way home from work?



Split and prune: Is this a good way to build a tree?

Well, at least it's fast…

Will I run into traffic on my way home from work?



Split and prune: Is this a good way to build a tree?

Well, at least it’s fast…

Example (adapted from Russell & Norvig):

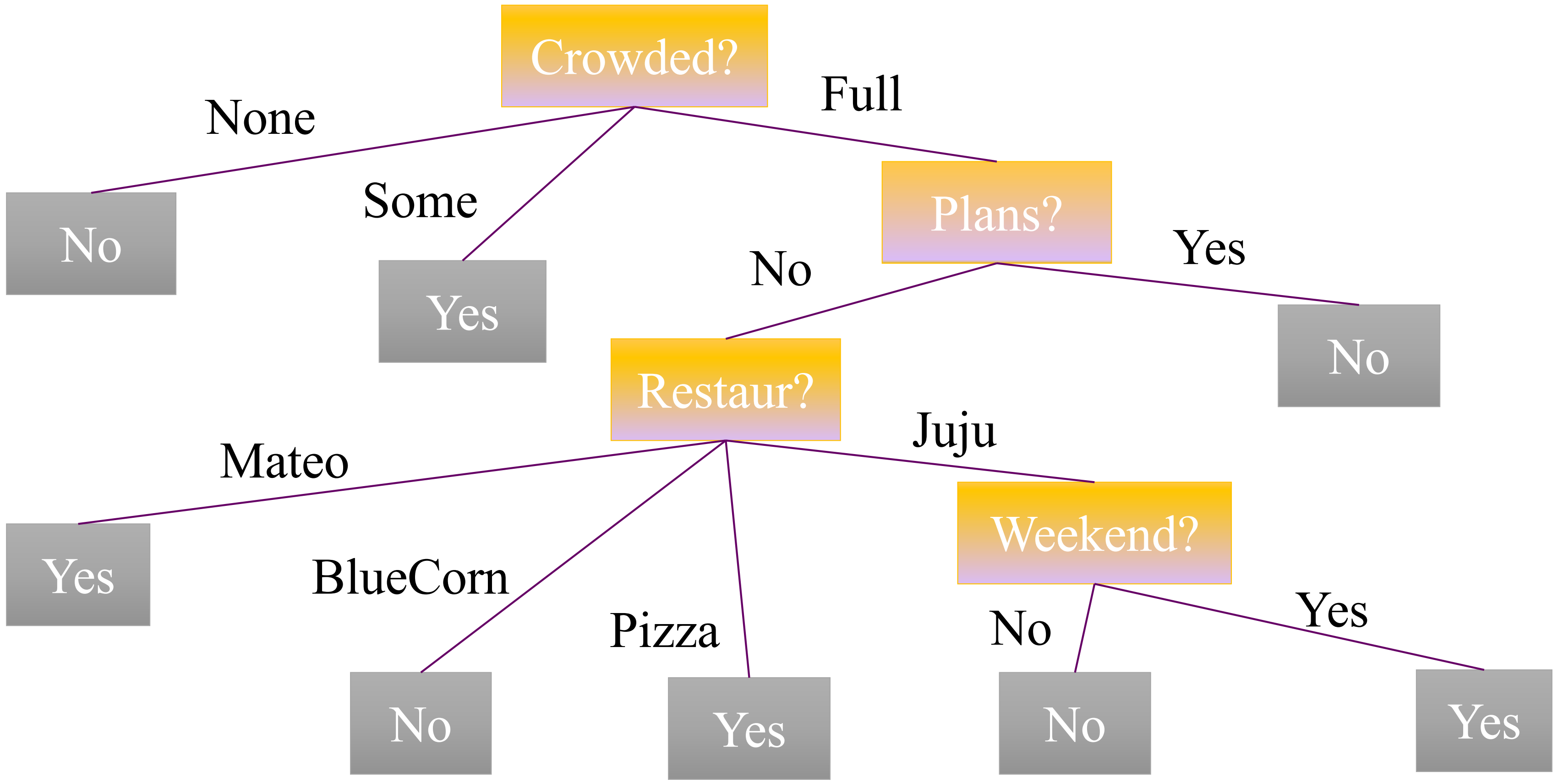
Will the customer wait for a table at a restaurant?

- OthOptions: Other options, True if there are restaurants nearby.
- Weekend: This is true if it is Friday, Saturday or Sunday.
- WaitArea: Does it have a bar or other nice waiting area to wait in?
- Plans: Does the customer have plans just after dinner?
- Price: This is either $, $$, $$$, or $$$$
- Precip: Is it raining or snowing?
- Restaur:
  - Mateo (fancy),
  - Juju (nice),
  - Blue Corn Mexican Café (casual),
  - Pompieri Pizza (very casual)
- Wait: Wait time estimate: 0-5 min, 6-15 min, 16-30 min, or 30+
- Crowded: Whether there are other customers (no, some, or full)

Dataset

	OthOptions	Weekend	WaitArea	Plans	Price	Precip	Restaur	Wait	Crowded	Stay?
x_1	Yes	No	No	Yes	$$$	No	Mateo	0-5	some	Yes
x_2	Yes	No	No	Yes	$	No	Juju	16-30	full	No
x_3	No	No	Yes	No	$	No	Pizza	0-5	some	Yes
x_4	Yes	Yes	No	Yes	$	No	Juju	6-15	full	Yes
x_5	Yes	Yes	No	No	$$$	No	Mateo	30+	full	No
x_6	No	No	Yes	Yes	$$	Yes	BlueCorn	0-5	some	Yes
x_7	No	No	Yes	No	$	Yes	Pizza	0-5	none	No
x_8	No	No	No	Yes	$$	Yes	Juju	0-5	some	Yes
x_9	No	Yes	Yes	No	$	Yes	Pizza	30+	full	No
x_{10}	Yes	Yes	Yes	Yes	$$$	No	BlueCorn	6-15	full	No
x_{11}	No	No	No	No	$	No	Juju	0-5	none	No
x_{12}	Yes	Yes	Yes	Yes	$	No	Pizza	16-30	full	Yes

Will the customer wait for a table at a restaurant?

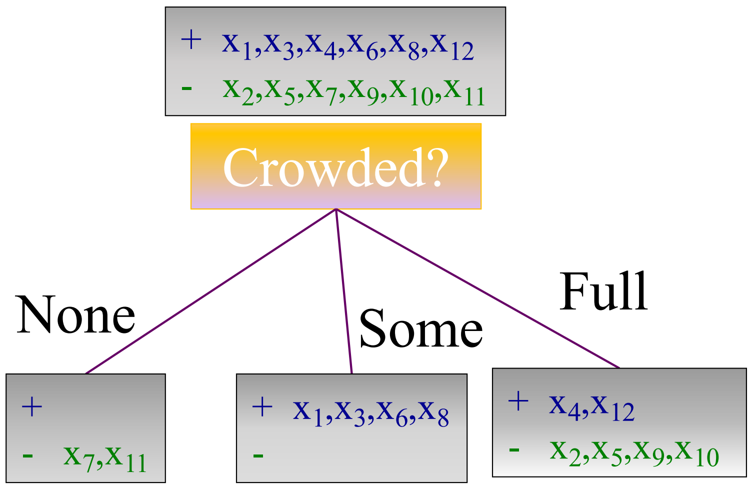


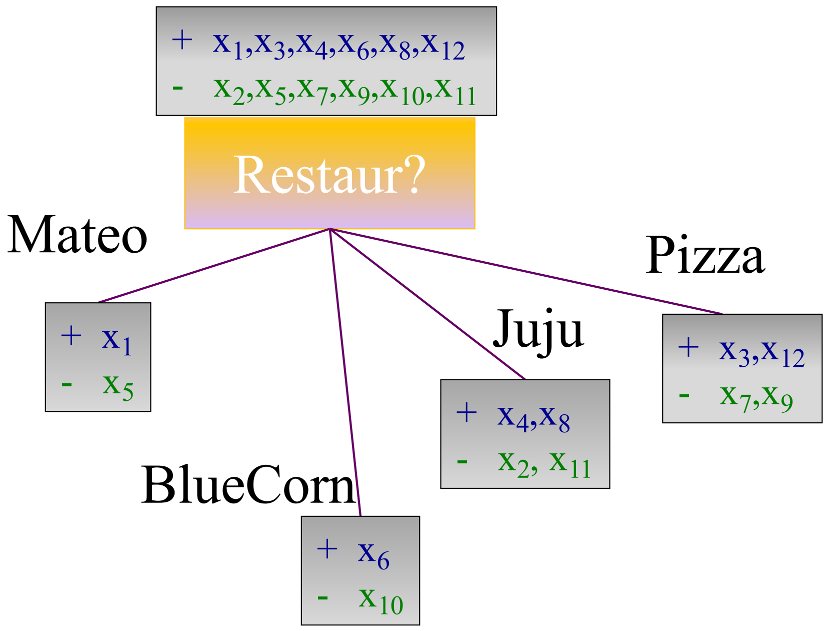
Greedy tree induction:

- Start at the top of the tree.
- Grow it by “splitting” features one by one. To split, look at how “impure” the node is.
- Assign leaf nodes the majority vote in the leaf.
- At the end, go back and prune leaves to reduce overfitting.

Which of these two features should we split on?

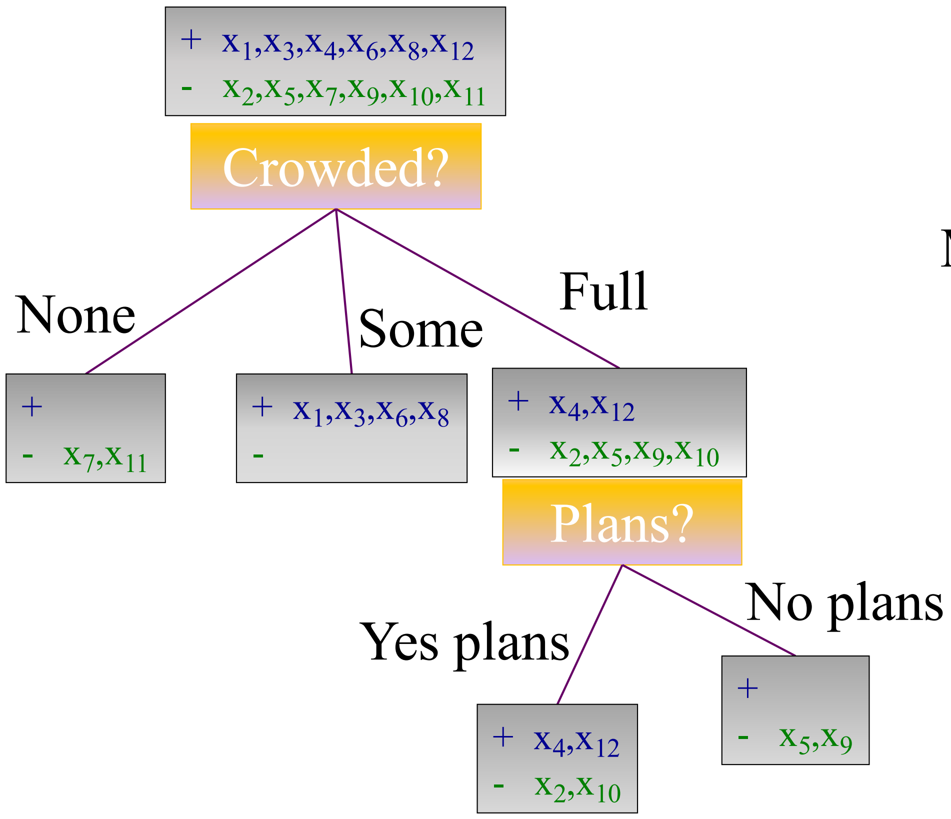
Which gives the most information about whether the customer will wait?

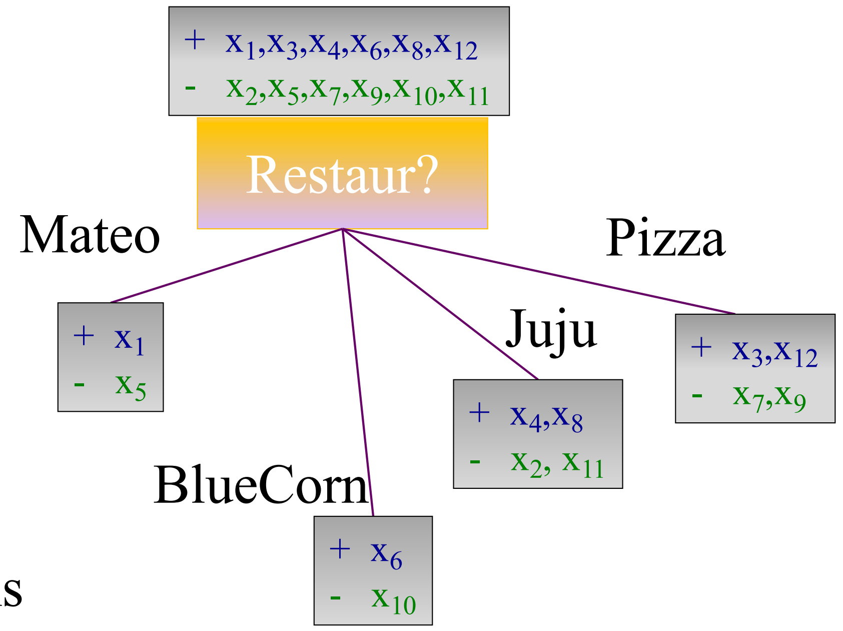




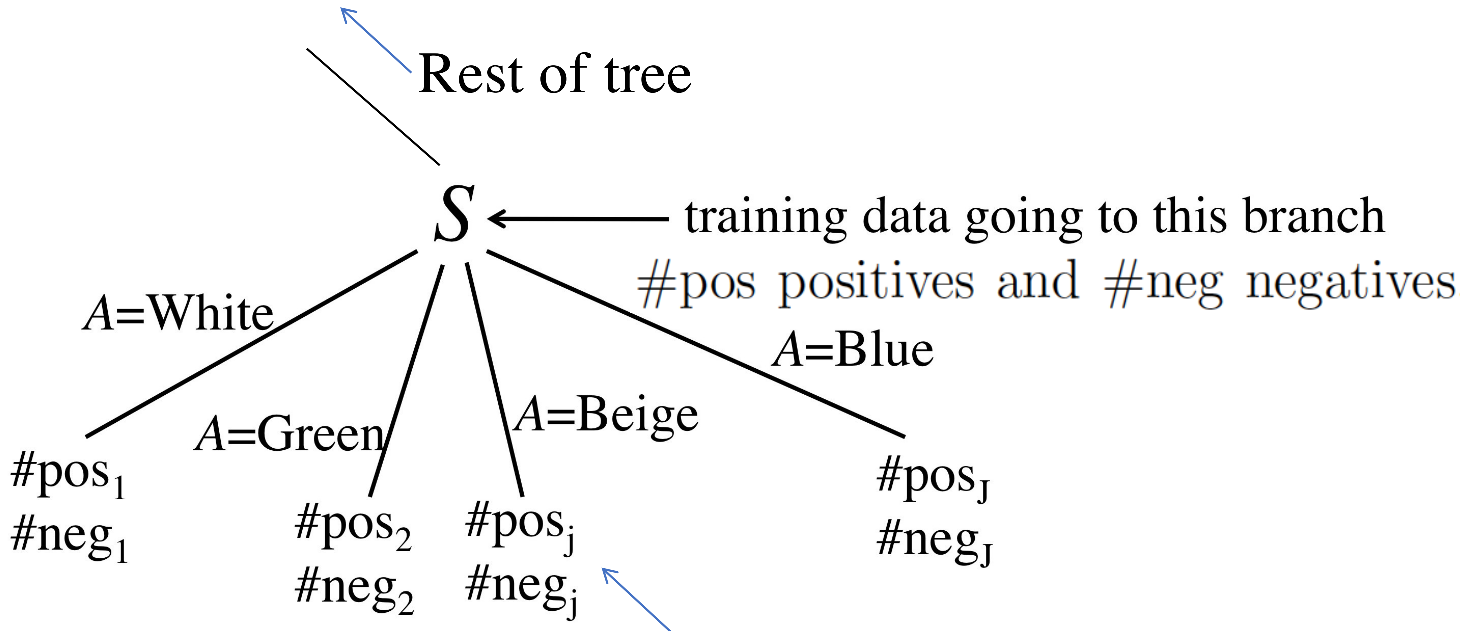
Which of these two features should we split on?

Which gives the most information about whether the customer will wait?





Splitting Criteria for Decision Trees: Information Gain


Rest of tree
S
training data going to this branch
#pos positives and #neg negatives
A=White
A=Green
A=Beige
A=Blue
$\#\text{pos}_1$ $\#\text{neg}_1$
$\#\text{pos}_2$ $\#\text{neg}_2$
$\#\text{pos}_j$ $\#\text{neg}_j$
$\#\text{pos}_J$ $\#\text{neg}_J$

The training probabilities in branch j are:

$$\left[\frac{\#\text{pos}_j}{\#\text{pos}_j + \#\text{neg}_j}, \frac{\#\text{neg}_j}{\#\text{pos}_j + \#\text{neg}_j}\right]$$

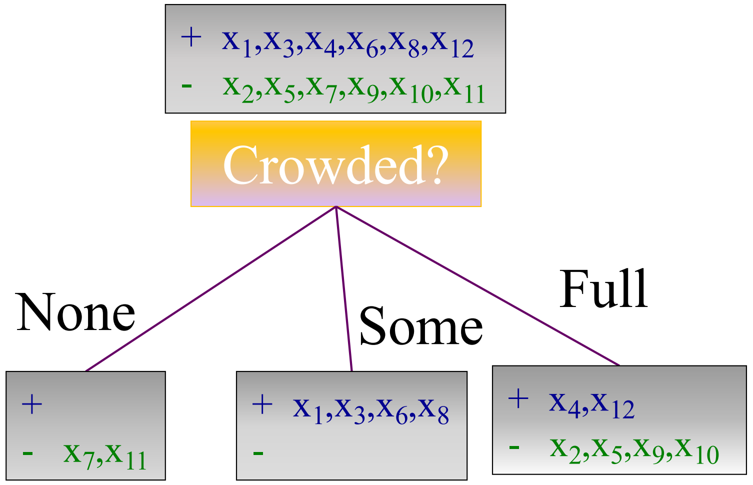
[.01,.99] is good
[.50,.50] is bad

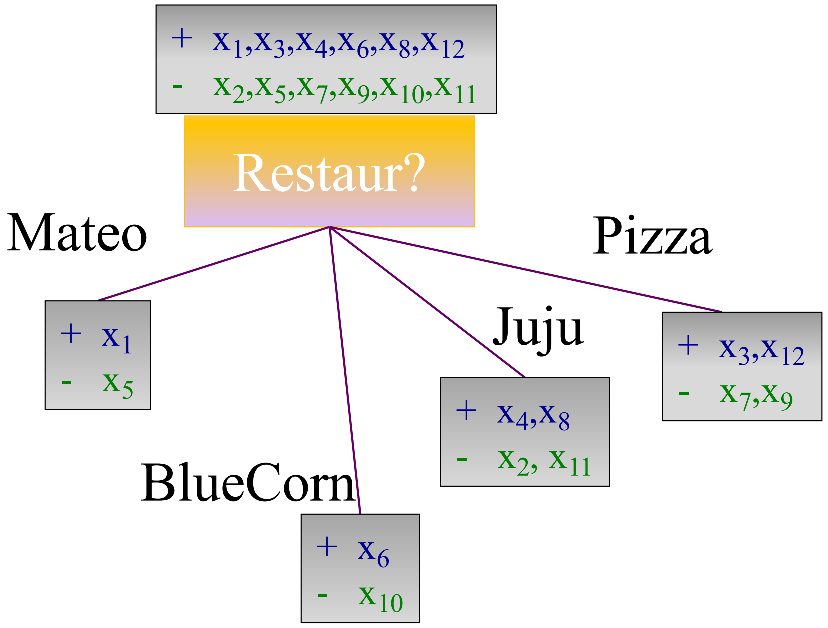
A Splitting Criteria for Decision Trees: Information Gain

$$\begin{aligned}\text{Gain}(S, A) &= \text{expected reduction in entropy due to branching on attribute } A \\ &= H\left(\left[\frac{\#\text{pos}}{\#\text{pos} + \#\text{neg}}, \frac{\#\text{neg}}{\#\text{pos} + \#\text{neg}}\right]\right) \\ &\qquad - \sum_{j=1}^{J} \frac{\#\text{pos}_j + \#\text{neg}_j}{\#\text{pos} + \#\text{neg}} H\left[\frac{\#\text{pos}_j}{\#\text{pos}_j + \#\text{neg}_j}, \frac{\#\text{neg}_j}{\#\text{pos}_j + \#\text{neg}_j}\right].\end{aligned}$$

Which of these two features should we split on?

Which gives the most information about whether the customer will wait?





A Splitting Criteria for Decision Trees: Information Gain

$$\begin{aligned}\mathrm{Gain}(S,A) &= \text{expected reduction in entropy due to branching on attribute } A\\ &= \text{original entropy} - \text{entropy after branching}\\ &= H\left(\left[\frac{\#\mathrm{pos}}{\#\mathrm{pos}+\#\mathrm{neg}}, \frac{\#\mathrm{neg}}{\#\mathrm{pos}+\#\mathrm{neg}}\right]\right)\\ &\qquad -\sum_{j=1}^{J}\frac{\#\mathrm{pos}_j+\#\mathrm{neg}_j}{\#\mathrm{pos}+\#\mathrm{neg}} H\left[\frac{\#\mathrm{pos}_j}{\#\mathrm{pos}_j+\#\mathrm{neg}_j}, \frac{\#\mathrm{neg}_j}{\#\mathrm{pos}_j+\#\mathrm{neg}_j}\right].\end{aligned}$$

(entropy of whole dataset) (entropy after splitting)

$$\begin{aligned}\text{Gain(S,Crowded)} &= H\left(\left[\frac{1}{2},\frac{1}{2}\right]\right) - \left[\frac{2}{12}H([0,1]) + \frac{4}{12}H([1,0]) + \frac{6}{12}H\left(\left[\frac{2}{6},\frac{4}{6}\right]\right)\right]\\ &\approx 0.541 \text{ bits.}\end{aligned}$$

Splitting Criteria for Decision Trees: Information Gain

$$\text{Gain(S,Restaur)} = 1 - \left[\frac{2}{12}H\left(\left[\frac{1}{2},\frac{1}{2}\right]\right) + \frac{2}{12}H\left(\left[\frac{1}{2},\frac{1}{2}\right]\right) + \frac{4}{12}H\left(\left[\frac{2}{4},\frac{2}{4}\right]\right) + \frac{4}{12}H\left(\left[\frac{2}{4},\frac{2}{4}\right]\right)\right] \approx 0 \text{ bits.}$$

$$\text{Gain(S,Crowded)} = H\left(\left[\frac{1}{2},\frac{1}{2}\right]\right) - \left[\frac{2}{12}H([0,1]) + \frac{4}{12}H([1,0]) + \frac{6}{12}H\left(\left[\frac{2}{6},\frac{4}{6}\right]\right)\right] \boxed{\approx 0.541 \text{ bits}}.$$

So far…

- I’ve discussed Information Gain, which is the splitting criteria for C4.5.

(Quinlan, 1993)

- I still need to discuss:
    - other possible splitting criteria
    - pruning criteria
    - CART
    - Some uses for greedy tree-splitting algorithms
    - What happens if you don’t want to be greedy?

Decision Trees Part 2

Cynthia Rudin

Machine Learning Class

Duke University

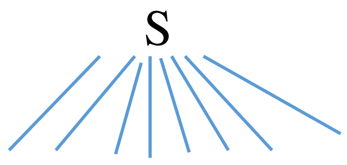
Where we left off:

- Information gain as the splitting criteria for C4.5 (Quinlan, 1993)
    - Chooses a split to reduce the entropy as much as possible

A Splitting Criterion for Decision Trees: Information Gain

$$\begin{aligned}
\text{Gain}(S, A) &= \text{expected reduction in entropy due to branching on attribute } A \\
&= \text{original entropy} - \text{entropy after branching} \\
&= H\left(\left[\frac{\#\text{pos}}{\#\text{pos}+\#\text{neg}}, \frac{\#\text{neg}}{\#\text{pos}+\#\text{neg}}\right]\right) \\
&\quad - \sum_{j=1}^{J} \frac{\#\text{pos}_j + \#\text{neg}_j}{\#\text{pos}+\#\text{neg}} H\left[\frac{\#\text{pos}_j}{\#\text{pos}_j+\#\text{neg}_j}, \frac{\#\text{neg}_j}{\#\text{pos}_j+\#\text{neg}_j}\right].
\end{aligned}$$

Encourages too many branches?



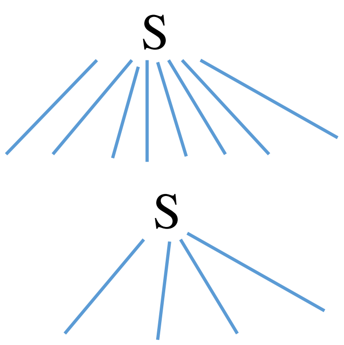
Another splitting criterion: Information Gain Ratio (Quinlan, 1986)

$$\mathrm{Gain}(S, A) \quad \leftarrow \quad \text{want large}$$

$|S_j|$ is the amount of data in branch j

$$\mathrm{SplitInfo}(S, A) = -\sum_{j=1}^{J} \underbrace{\frac{|S_j|}{|S|}}_{\text{Want large}} \log\left(\frac{|S_j|}{|S|}\right)$$

Want large



More splitting criteria for binary splitting

Other replacements for H([p,1-p])

- Gini Index (used by CART, Breiman, 1984)

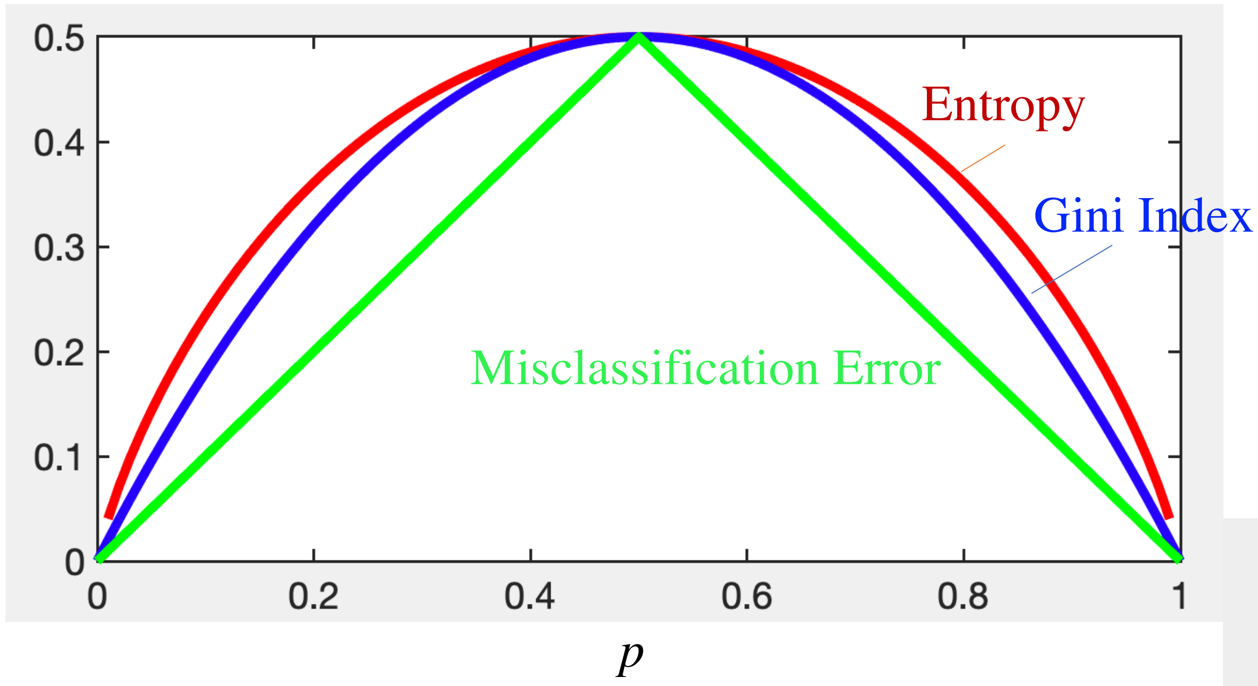
  Gini index = $2p(1-p)$ = 2 x variance of Bernoulli

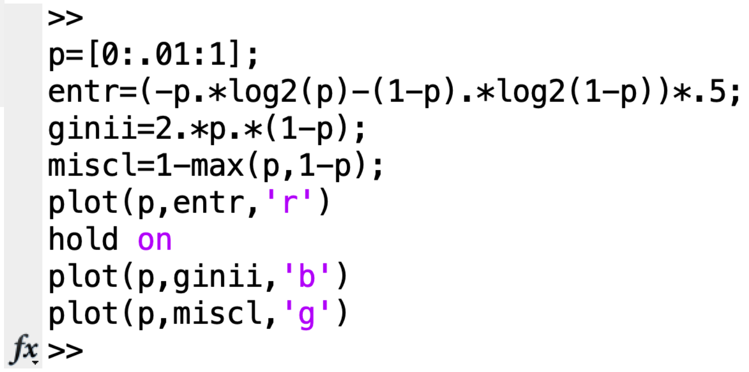
- Misclassification error $1-\max(p,1-p)$

Classify according to majority vote: if $p \leq .5$ vote no, otherwise yes

6 yes, 4 no - predict yes	made 4/10 errors	$p = .6$	$1-\max(p,1-p) = 1-\max(.6,.4)=1-.6=.4$
2 yes, 8 no - vote no	made 2/10 errors	$p = .2$	$1-\max(p,1-p) = 1-\max(.2,.8)=1-.8=.2$
7 yes, 3 no - vote yes	made 3/10 errors	$p = .7$	$1-\max(p,1-p) = 1-\max(.7,.3)=1-.7=.3$

More splitting criteria for binary splitting





```
>>
p=[0:.01:1];
entr=(-p.*log2(p)-(1-p).*log2(1-p))*.5;
ginii=2.*p.*(1-p);
miscl=1-max(p,1-p);
plot(p,entr,'r')
hold on
plot(p,ginii,'b')
plot(p,miscl,'g')
fx >>
```

Keep splitting until:

- All observations in each leaf have the same class
- No more features to split (you split on all of them already)

This is likely to overfit

Decision Trees Part 3

Cynthia Rudin

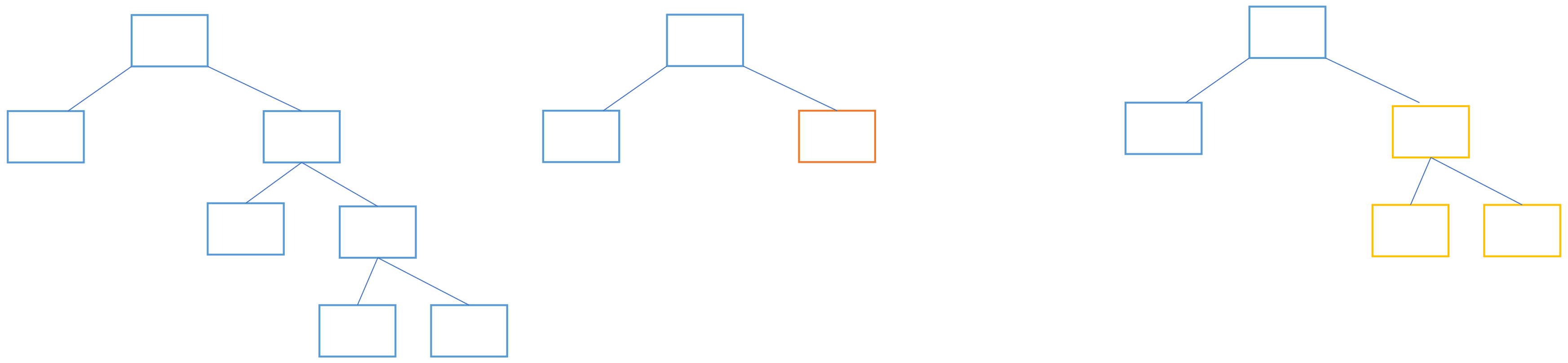
Machine Learning Class

Duke University

Pruning

C4.5 recursively makes choices for pruning:

- Option 1: leaving the tree as is
- Option 2: collapse that part of the tree into a leaf. The leaf has the most frequent label in the data S going to that part of the tree.
- Option 3: replace that part of the tree with one of its subtrees, corresponding to the most common branch in the split



Pruning

Which of the three options?

C4.5 computes upper bounds on the probability of error for each option. It uses standard upper confidence bounds on probabilities from the binomial distribution with $\alpha=.25$.

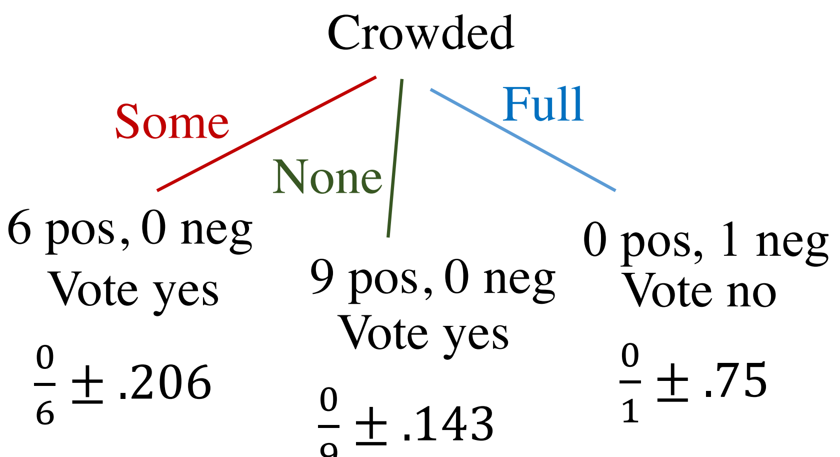
- Prob of error for Option 1 $\leq$ UpperBound1
- Prob of error for Option 2 $\leq$ UpperBound2
- Prob of error for Option 3 $\leq$ UpperBound3

	Crowded	Price	Stay?
x_1	Some	$$	Yes
x_2	Some	$$$	Yes
x_3	None	$$	Yes
x_4	Some	$$	Yes
x_5	None	$$	Yes
x_6	None	$$$$	Yes
x_7	None	$$	Yes
x_8	None	$	Yes
x_9	Some	$$	Yes
x_{10}	None	$$	Yes
x_{11}	Some	$	Yes
x_{12}	Full	$$	No
x_{13}	None	$$	Yes
x_{14}	None	$	Yes
x_{15}	Some	$$$	Yes
x_{16}	None	$	Yes

New dataset (from the Kranf site)

Consider split on Crowded

Option 1: Leave the tree as it is



```
>> binocdf(0,6,.206)

ans =

    0.2506
```

```
>> binocdf(0,9,.143)

ans =

    0.2494
```

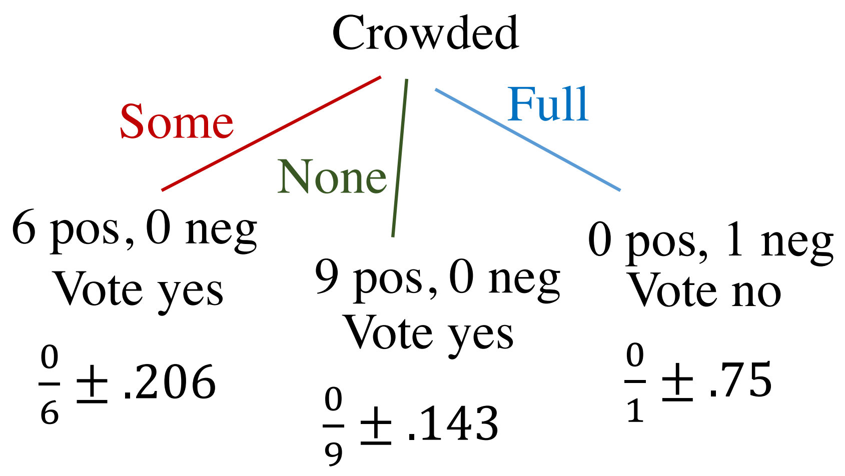
```
>> binocdf(0,1,.75)

ans =

    0.2500
```

	Crowded	Price	Stay?
x_1	Some	$$	Yes
x_2	Some	$$$	Yes
x_3	None	$$	Yes
x_4	Some	$$	Yes
x_5	None	$$	Yes
x_6	None	$$$$	Yes
x_7	None	$$	Yes
x_8	None	$	Yes
x_9	Some	$$	Yes
x_{10}	None	$$	Yes
x_{11}	Some	$	Yes
x_{12}	Full	$$	No
x_{13}	None	$$	Yes
x_{14}	None	$	Yes
x_{15}	Some	$$$	Yes
x_{16}	None	$	Yes

New dataset (from the Kranf site)

Consider split on Crowded

Option 1: Leave the tree as it is



UpperBound1 = average upper bound over leaves

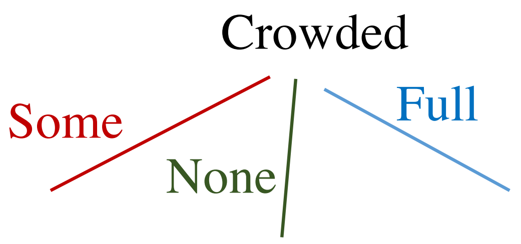
$$= \frac{1}{16}(6\times.206 \;+\; 9\times.143 \;+\; 1\times.75\,)$$

	Crowded	Price	Stay?
x_1	Some	$$	Yes
x_2	Some	$$$	Yes
x_3	None	$$	Yes
x_4	Some	$$	Yes
x_5	None	$$	Yes
x_6	None	$$$$	Yes
x_7	None	$$	Yes
x_8	None	$	Yes
x_9	Some	$$	Yes
x_{10}	None	$$	Yes
x_{11}	Some	$	Yes
x_{12}	Full	$$	No
x_{13}	None	$$	Yes
x_{14}	None	$	Yes
x_{15}	Some	$$$	Yes
x_{16}	None	$	Yes

New dataset (from the Kranf site)

Consider split on Crowded

Option 2 & 3: Prune the tree to a leaf



	Crowded	Price	Stay?
x_1	Some	$$	Yes
x_2	Some	$$$	Yes
x_3	None	$$	Yes
x_4	Some	$$	Yes
x_5	None	$$	Yes
x_6	None	$$$$	Yes
x_7	None	$$	Yes
x_8	None	$	Yes
x_9	Some	$$	Yes
x_{10}	None	$$	Yes
x_{11}	Some	$	Yes
x_{12}	Full	$$	No
x_{13}	None	$$	Yes
x_{14}	None	$	Yes
x_{15}	Some	$$$	Yes
x_{16}	None	$	Yes

New dataset

Consider split on Crowded

Option 2 & 3: Prune the tree to a leaf

Crowded

15 pos, 1 neg

.157

```
>> binocdf(1,16,.157)

ans =

    0.2589
```

UpperBound2 = average upper bound over leaves

$$= \frac{1}{16}(16\times.157) = .157$$

UpperBound1 = average upper bound over leaves

$$= \frac{1}{16}(6\times.206 \; + \; 9\times.143 \; + \; 1\times.75\,) = .204$$

	Crowded	Price	Stay?
x_1	Some	$$	Yes
x_2	Some	$$$	Yes
x_3	None	$$	Yes
x_4	Some	$$	Yes
x_5	None	$$	Yes
x_6	None	$$$$	Yes
x_7	None	$$	Yes
x_8	None	$	Yes
x_9	Some	$$	Yes
x_{10}	None	$$	Yes
x_{11}	Some	$	Yes
x_{12}	Full	$$	No
x_{13}	None	$$	Yes
x_{14}	None	$	Yes
x_{15}	Some	$$$	Yes
x_{16}	None	$	Yes

New dataset

Consider split on Crowded

Option 2 & 3: Prune the tree to a leaf

UpperBound2 = average upper bound over leaves

$$= \frac{1}{16}(16\times.157) = \boxed{.157}$$

UpperBound1 = average upper bound over leaves

$$= \frac{1}{16}(6\times.206 + 9\times.143 + 1\times.75) = .204$$

Other questions

Q. Where did α=.25 come from?

Q. How do you know which subtrees to consider?

Q. Can I change it?

Q. Are the trees “optimal”?

Decision Trees Part 4

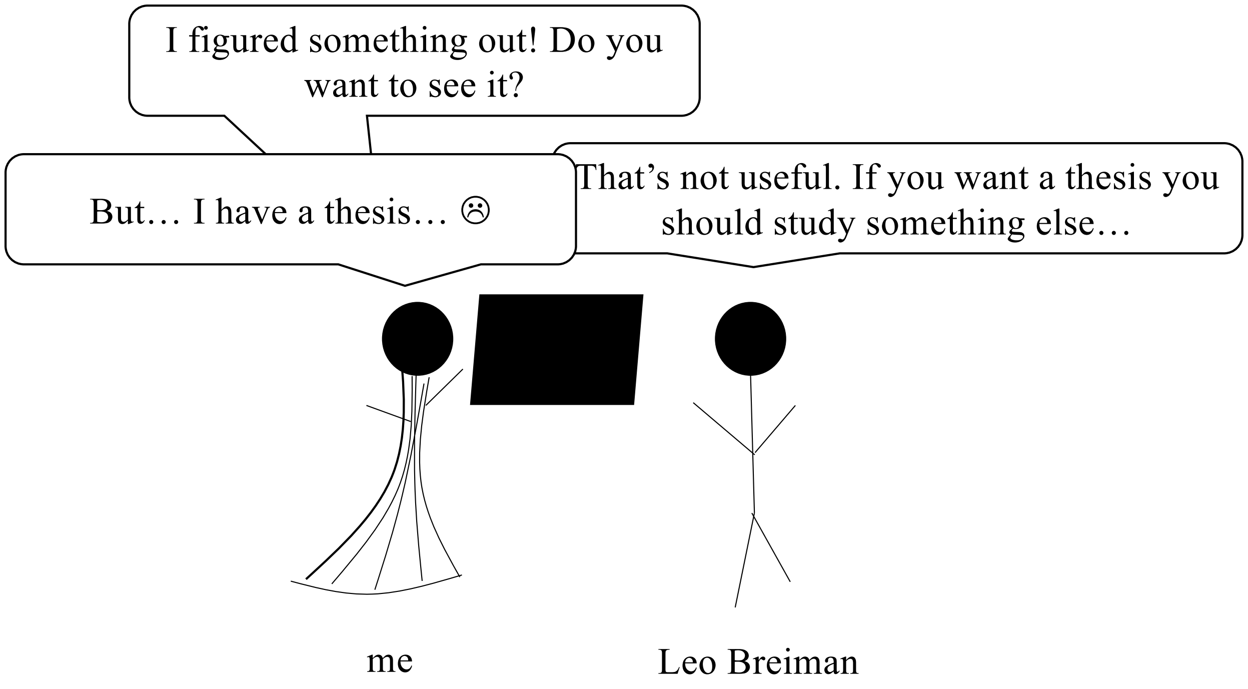
Cynthia Rudin

Machine Learning Class

Duke University

CART – Classification and Regression Trees

(Breiman, Friedman, Olshen, Stone, 1984)


I figured something out! Do you want to see it?
That's not useful. If you want a thesis you should study something else…
But… I have a thesis… ☹
me
Leo Breiman

CART – *Classification* and Regression Trees

(Breiman, Friedman, Olshen, Stone, 1984)

- Does only binary splits, not multiway splits.
- Uses the Gini index for splitting.
- Uses Minimum Cost Complexity for pruning

Each subtree is assigned a cost. Choose the subtree with the lowest cost

$$\text{cost(subtree)} = \sum_{\text{leaves } j} \overset{\substack{\text{Misclassification} \\ \text{Error} \\ \downarrow}}{\sum_{x_i \in \text{leaf } j}} \mathbf{1}_{[y_i \neq \text{leaf's class}]} + \overset{\substack{\text{Regularization} \\ \downarrow}}{C\,[\#\text{leaves in subtree}]}$$

Each new leaf costs C.

That means each new leaf is worth the same as C misclassified points.

CART – Classification and *Regression* Trees

(Breiman, Friedman, Olshen, Stone, 1984)

- For regression, assign f (x) to be constant in each leaf.
- Choose value of f (x) to minimize the squared loss:

$$R^{\text{train}}(f) = \sum_i (y_i - f(x_i))^2$$

$$= \sum_{\text{leaves } j} \sum_{i \in \text{leaf } j} (y_i - f(x_i))^2$$ (Group terms by leaf)

$$= \sum_{\text{leaves } j} \sum_{i \in \text{leaf } j} (y_i - f_j)^2 =: \sum_{\text{leaves } j} R_j^{\text{train}}(f_j).$$

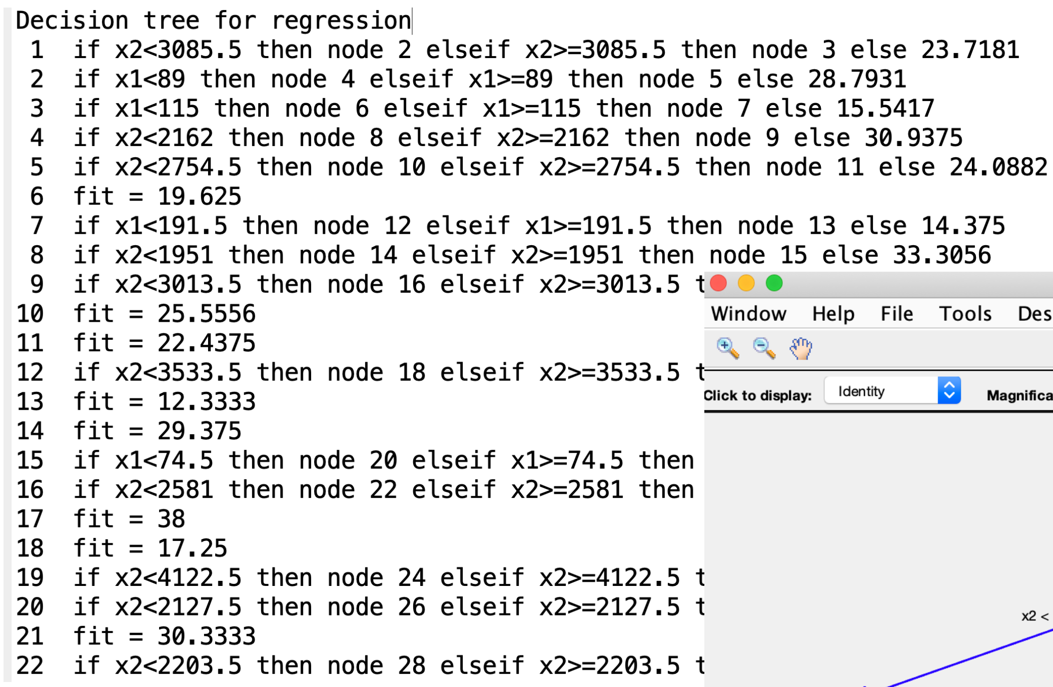
(Constant f in leaf)

Choose f_j to minimize $R_j^{\text{train}}(f_j)$

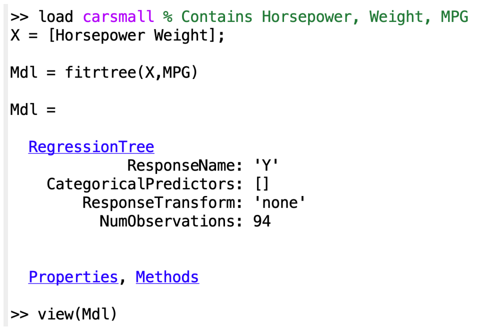
$$\begin{aligned}
0 &= \frac{d}{d\tilde{f}} \sum_{i \in \text{leaf } j} (y_i - \tilde{f})^2 \Big|_{\tilde{f}=f_j} \\
&= -2 \sum_i (y_i - \tilde{f}) \Big|_{\tilde{f}=f_j} = -2 \left(\left(\sum_i y_i \right) - |S_j| \tilde{f} \right) \Big|_{\tilde{f}=f_j} \\
f_j &= \frac{1}{|S_j|} \sum_{i \in \text{leaf } j} y_i = \bar{y}_{S_j},
\end{aligned}$$

where $\bar{y}_{S_j}$ is the sample average of the labels for leaf j's examples.

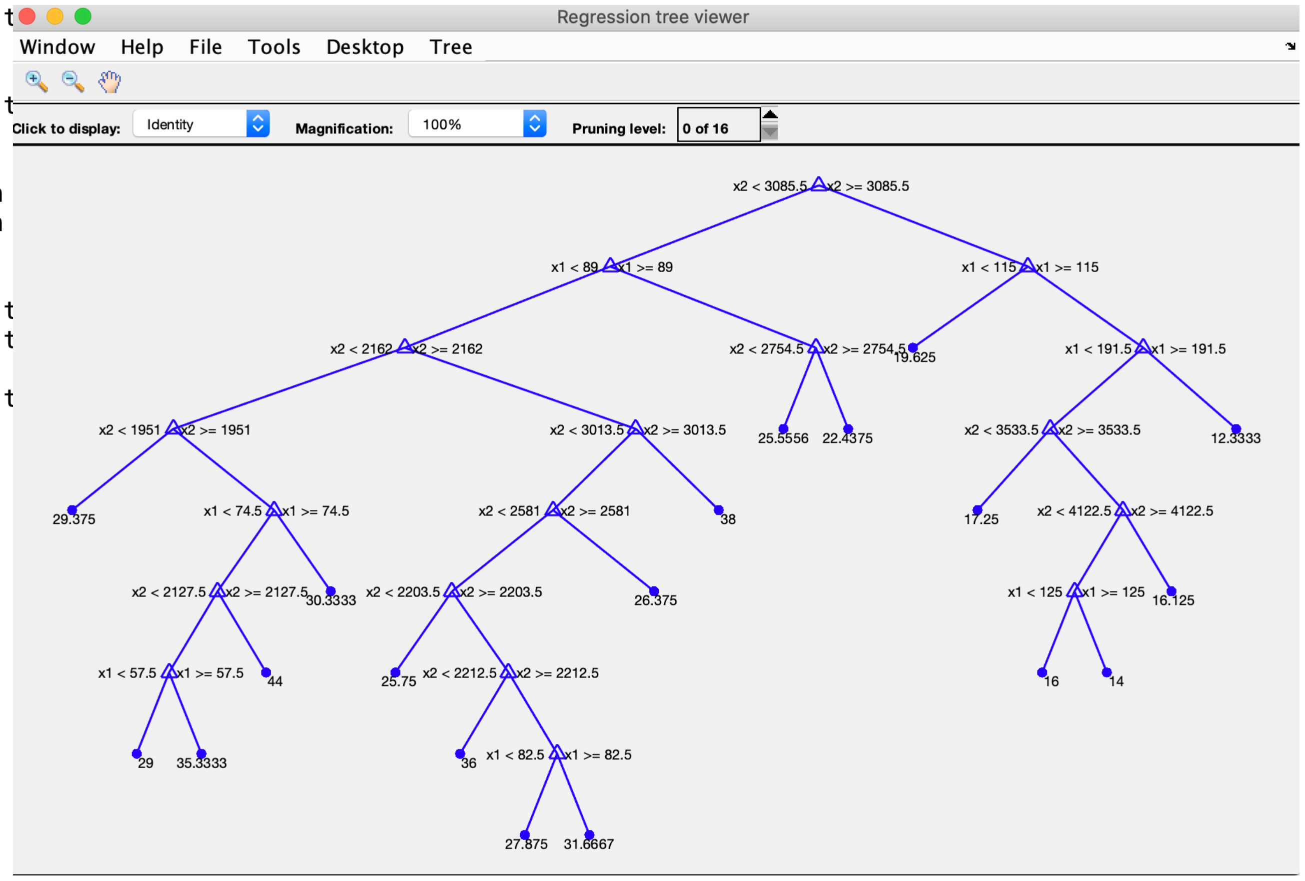
Choose f_j to minimize $R_j^{\text{train}}(f_j)$

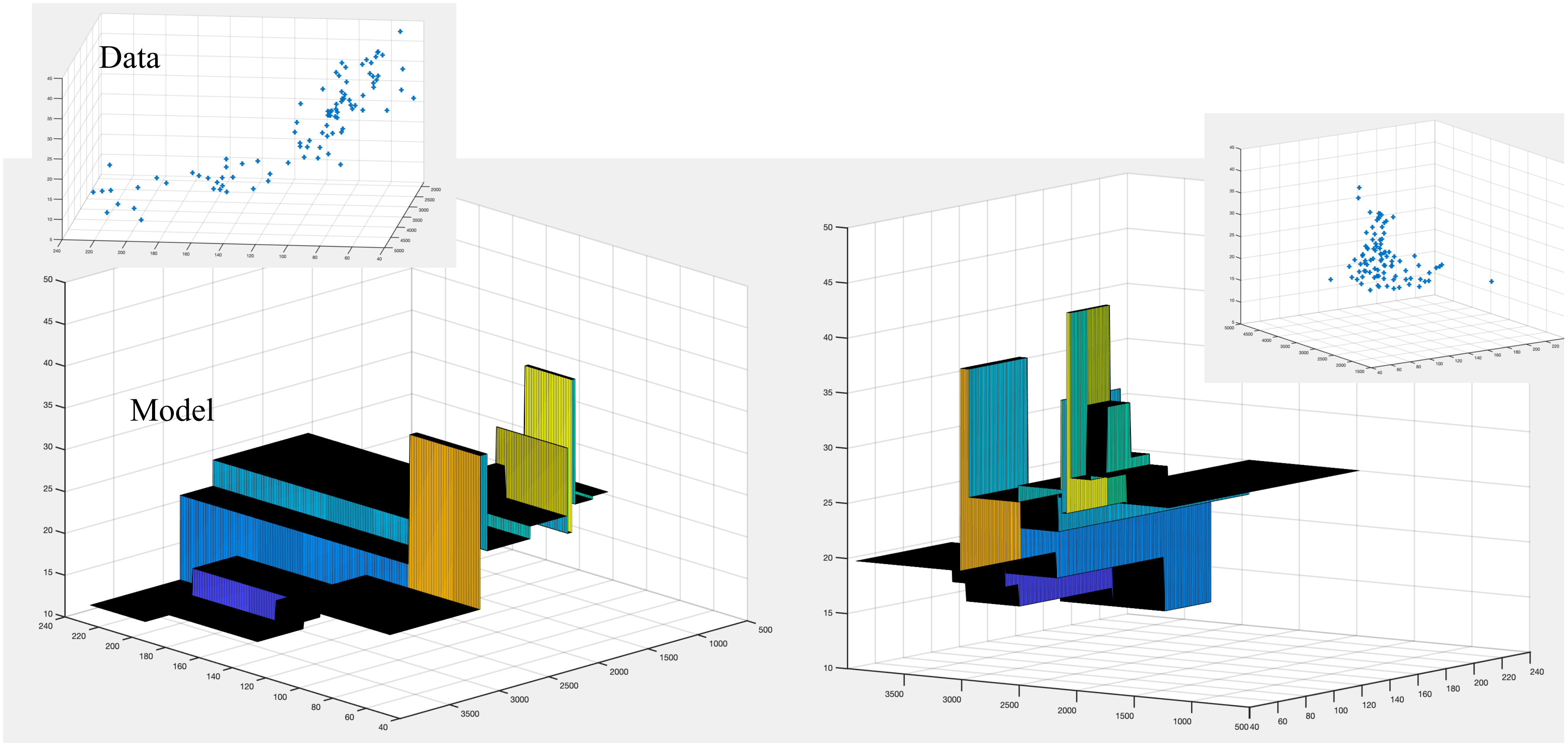


```
Decision tree for regression
 1  if x2<3085.5 then node 2 elseif x2>=3085.5 then node 3 else 23.7181
 2  if x1<89 then node 4 elseif x1>=89 then node 5 else 28.7931
 3  if x1<115 then node 6 elseif x1>=115 then node 7 else 15.5417
 4  if x2<2162 then node 8 elseif x2>=2162 then node 9 else 30.9375
 5  if x2<2754.5 then node 10 elseif x2>=2754.5 then node 11 else 24.0882
 6  fit = 19.625
 7  if x1<191.5 then node 12 elseif x1>=191.5 then node 13 else 14.375
 8  if x2<1951 then node 14 elseif x2>=1951 then node 15 else 33.3056
 9  if x2<3013.5 then node 16 elseif x2>=3013.5 t
10  fit = 25.5556
11  fit = 22.4375
12  if x2<3533.5 then node 18 elseif x2>=3533.5 t
13  fit = 12.3333
14  fit = 29.375
15  if x1<74.5 then node 20 elseif x1>=74.5 then
16  if x2<2581 then node 22 elseif x2>=2581 then
17  fit = 38
18  fit = 17.25
19  if x2<4122.5 then node 24 elseif x2>=4122.5 t
20  if x2<2127.5 then node 26 elseif x2>=2127.5 t
21  fit = 30.3333
22  if x2<2203.5 then node 28 elseif x2>=2203.5 t
```



```
>> load carsmall % Contains Horsepower, Weight, MPG
X = [Horsepower Weight];

Mdl = fitrtree(X,MPG)

Mdl =

  RegressionTree
           ResponseName: 'Y'
  CategoricalPredictors: []
      ResponseTransform: 'none'
        NumObservations: 94


  Properties, Methods

>> view(Mdl)
```




Data
Model

For real-valued features, CART chooses feature j to split, split point s, values C_1 and C_2 for leaves all at the same time.

$$\min_{\substack{j,\,s \\ \text{for each feature } j \text{ do} \\ \text{a linesearch over } s}} \left[\min_{C_1} \sum_{x_i \in \{\text{leaf} | x^{(j)} \leq s\}} (y_i - C_1)^2 + \min_{C_2} \sum_{x_i \in \{\text{leaf} | x^{(j)} > s\}} (y_i - C_2)^2 \right]$$

split point

split point

For real-valued features, CART chooses
feature j to split, split point s, values C_1 and C_2 for leaves
all at the same time.

$$\min_{\substack{j,\,s \\ \text{for each feature } j \text{ do} \\ \text{a linesearch over } s}} \left[\underbrace{\min_{C_1} \sum_{x_i \in \{\text{leaf}|x^{(j)} \le s\}} (y_i - C_1)^2}_{C_1 = \bar{y}_{\{\text{leaf}|x^{(j)} \le s\}}} + \underbrace{\min_{C_2} \sum_{x_i \in \{\text{leaf}|x^{(j)} > s\}} (y_i - C_2)^2}_{C_2 = \bar{y}_{\{\text{leaf}|x^{(j)} > s\}}} \right]$$

For regression, CART also does cost-complexity pruning

$$\text{cost} = \sum_{\text{leaves } j} \sum_{x_i \in S_j} (y_i - \bar{y}_{S_j})^2 + C[\# \text{ leaves in tree}]$$

Some Perspective

- We have learned several splitting and pruning procedures. We know how CART and C4.5 work.

- Why are we studying decision trees again?
  1) If you combine many decision trees, amazing results!
  2) Decision trees are interpretable (well, CART is, C4.5 not so much)

Greedy is *not* the only way to train an interpretable decision tree.

Decision Trees Part 5

Cynthia Rudin

Machine Learning Class

Duke University

Advantages of Decision Tree Methods CART and C4.5 (and friends)

- Interpretable (at least CART)
- Handles nonlinearities
- Greedy, so very fast
- Can easily handle imbalanced data by reweighting the points.

Disadvantages of CART and C4.5

- Greedy, less accurate than other methods
- Heuristic algorithms – not elegant
- No proof of optimality
- No proof of nearness to optimality
- Tend to do poorly for imbalanced data, even after adjusting the parameters.
- CART tends to be more interpretable but less accurate than C4.5, but C4.5 produces completely uninterpretable models by default.

Modern Decision Tree Methods

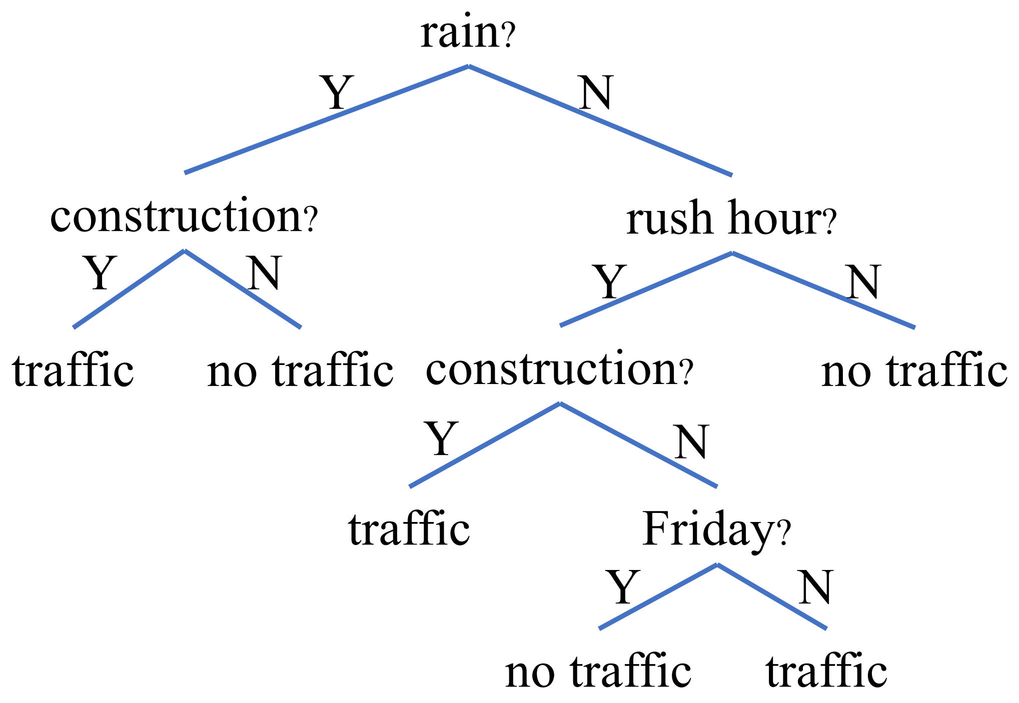
- Aim for full minimization of the Cost Complexity objective

$$\min_{\text{tree}} \hat{L}(\text{tree}, \{(x_i, y_i)\}_i) \text{ where}$$

$$\hat{L}(\text{tree}, \{(x_i, y_i)\}_i) = \underbrace{\frac{1}{n}\sum_{i=1}^{n} 1_{[\text{tree}(x_i) \neq y_i]}}_{\text{Misclassification Error}} + \underbrace{\lambda(\#\text{ leaves in tree})}_{\text{Sparsity}}$$

- Most recent method is called GOSDT – Generalized and Scalable Optimal Sparse Decision Trees (Lin et al, ICML 2020) – beyond scope of course

Modern Decision Tree Methods



Modern Decision Tree Methods

- Aim for full minimization of the Cost Complexity objective

$$\min_{\text{tree}} \hat{L}(\text{tree}, \{(x_i, y_i)\}_i) \text{ where}$$

$$\hat{L}(\text{tree}, \{(x_i, y_i)\}_i) = \underset{\text{Generalized Objectives}\ \Downarrow}{\ell(\text{tree}, \{(x_i, y_i)\}_i)} + \underbrace{\lambda(\#\text{ leaves in tree})}_{\text{Sparsity}}$$

- Any loss function monotonically increasing in FP and FN
  - Balanced accuracy, weighted accuracy, F-1, precision, …
- Rank statistics
  - AUC and partial AUC under the ROC convex hull
- Can prove optimality or nearness to optimality for not-huge datasets

Modern Decision Tree Methods

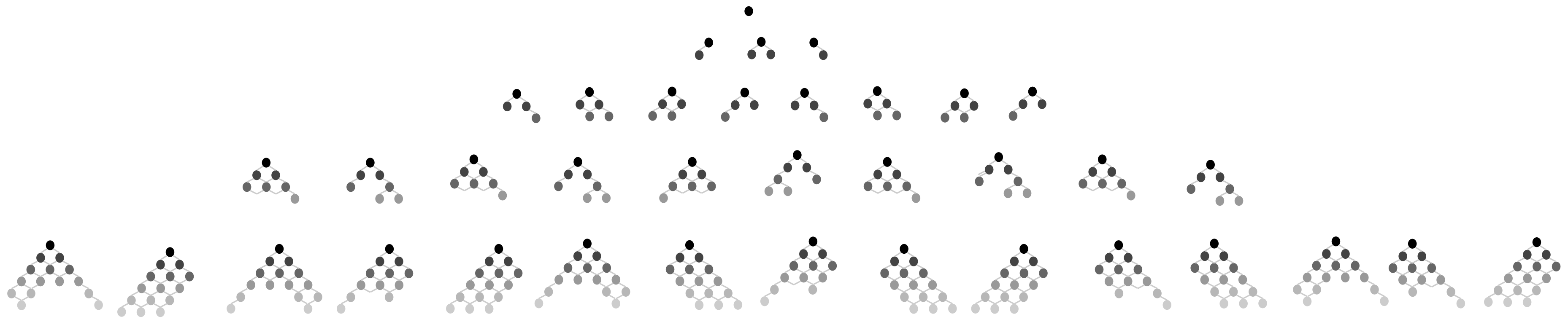
- Not greedy splitting and pruning
- Sophisticated pairing of theoretical bounds for reducing the search space with techniques for careful storage+lookup of previously-solved subproblems.

(Dynamic programming + analytical bounds)

Generalized and Scalable Optimal Sparse Decision Trees

Analytical Bounds

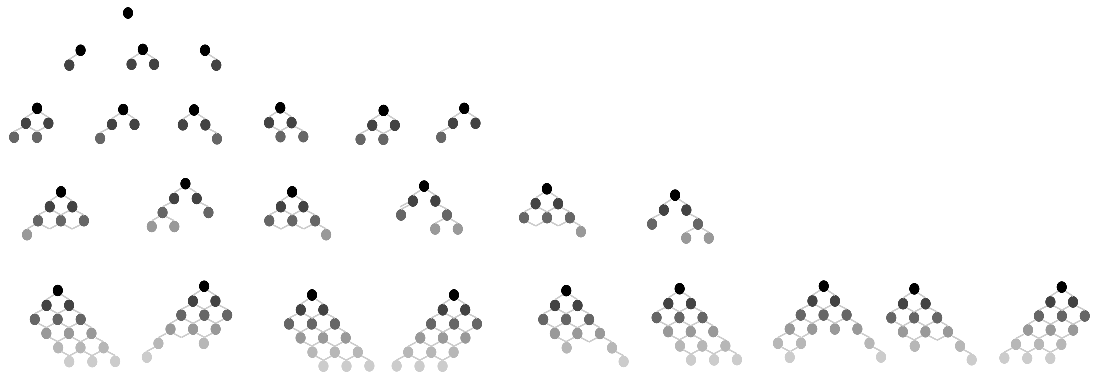
Several theorems show that some partial trees can never be extended to form optimal trees.



Generalized and Scalable Optimal Sparse Decision Trees

Analytical Bounds

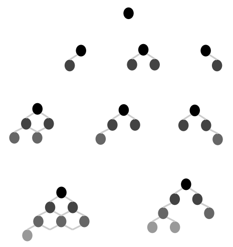
Several theorems show that some partial trees can never be extended to form optimal trees.



Generalized and Scalable Optimal Sparse Decision Trees

Analytical Bounds

Several theorems show that some partial trees can never be extended to form optimal trees.



$$R(\text{tree}) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[y_i \neq \text{tree}(\mathbf{x}_i)]} + C(\#\text{leaves in tree})$$

Theorem (Basic Size Bound).
If we have previously encountered a tree with objective R^c, then any optimal tree, $\text{tree}^* \in \arg\min_{\text{tree}} R(\text{tree})$, obeys

$$\#\text{leaves in tree}^* \leq \lfloor R^c/C \rfloor.$$

$$\begin{aligned}
R(\text{tree}^*) &\leq R^c \quad \text{(definition of tree}^* \text{ as optimal)}\\
\frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{[y_i \neq \text{tree}^*(\mathbf{x}_i)]} + C(\#\text{leaves in tree}^*) &\leq R^c\\
C(\#\text{leaves in tree}^*) &\leq R^c\\
\#\text{leaves in tree}^* &\leq \lfloor R^c/C \rfloor
\end{aligned}$$

Many other bounds

- Theorems prove certain splits are not possible because they won't lead to accurate enough trees
- We can sometimes prove that one partial tree can never be extended to form a better tree than one we have already seen.
- There are many isomorphisms of the same tree. We need only work with one of them.
- Trees that are very similar to each other have similar objective values.

Generalized and Scalable Optimal Sparse Decision Trees

Advantages of GOSDT

- Interpretable, simple models that generalize well
- Handles nonlinearities
- Handles wide variety of objectives, even custom objectives
- Handles imbalanced data
- Proof of optimality, or closeness to optimality
- No splitting and pruning heuristics

Disadvantages of GOSDT

- Can’t prove optimality for big datasets… yet