Decision Trees

Pradeep Ravikumar

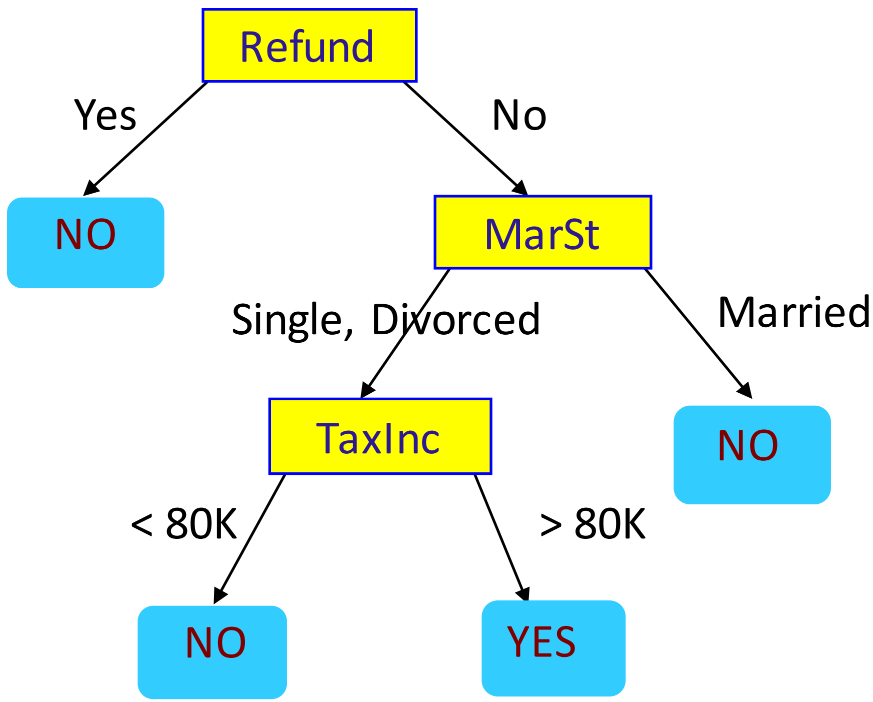
Co-instructor: Ziv Bar-Joseph

Machine Learning 10-701

Representation

- Question: What function does a decision tree represent?
  - Recall that in linear regression, we used a linear function of the input to predict the output

Decision Tree for Tax Fraud Detection



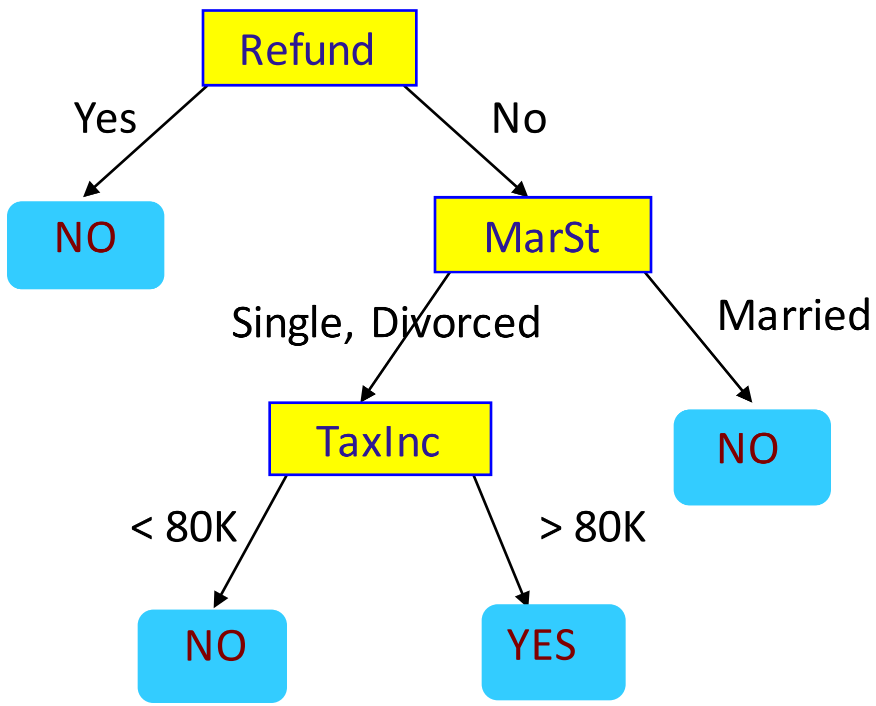
X_1	X_2	X_3	Y
Refund	**Marital Status**	**Taxable Income**	**Cheat**

- Each internal node: test one feature X_i
- Each branch from a node: selects some value for X_i
- Each leaf node: prediction for Y

Prediction

- Question: Given a decision tree, how do we assign a label to a test point?

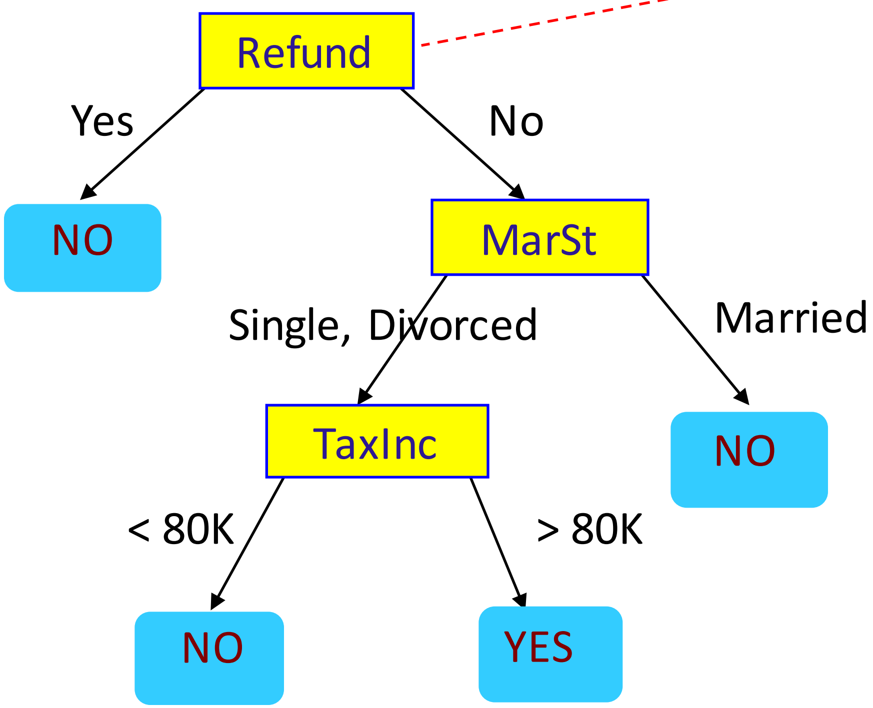
Decision Tree for Tax Fraud Detection



Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

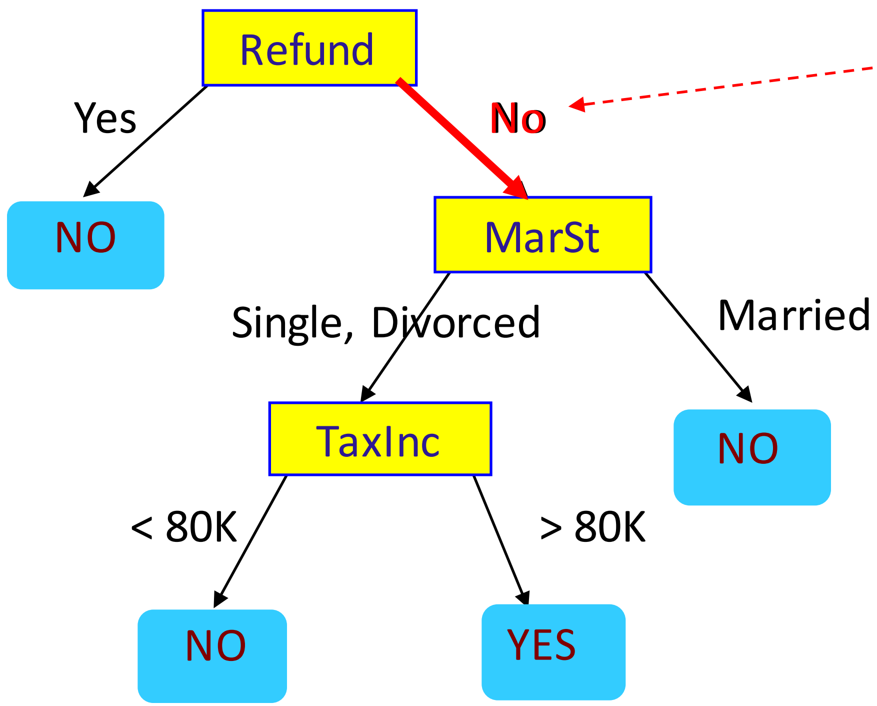
Decision Tree for Tax Fraud Detection



Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

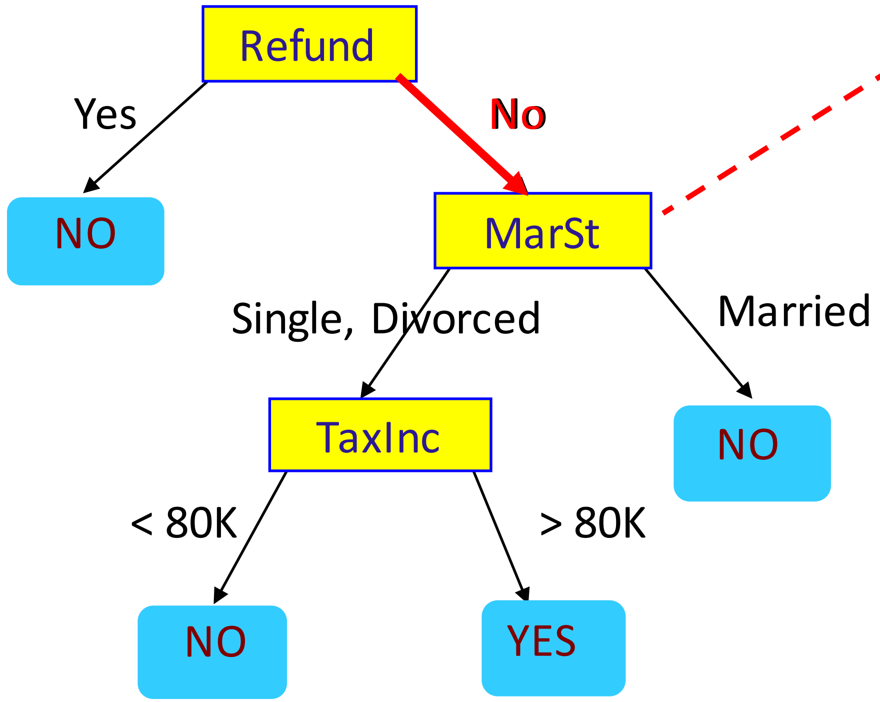
Decision Tree for Tax Fraud Detection



Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

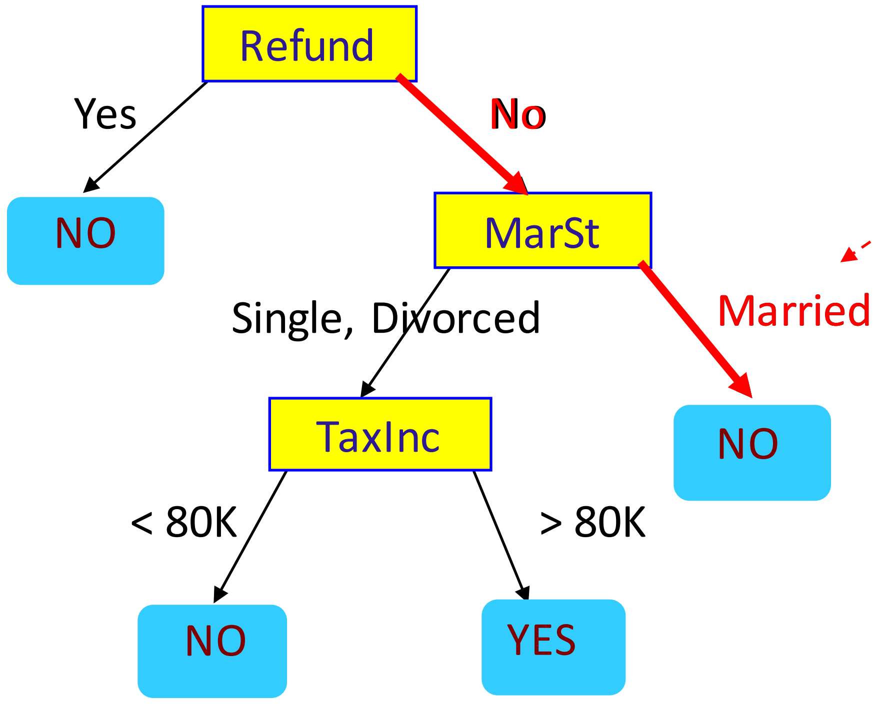
Decision Tree for Tax Fraud Detection



Query Data

X_1 Refund	X_2 Marital Status	X_3 Taxable Income	Y Cheat
No	Married	80K	?

Decision Tree for Tax Fraud Detection



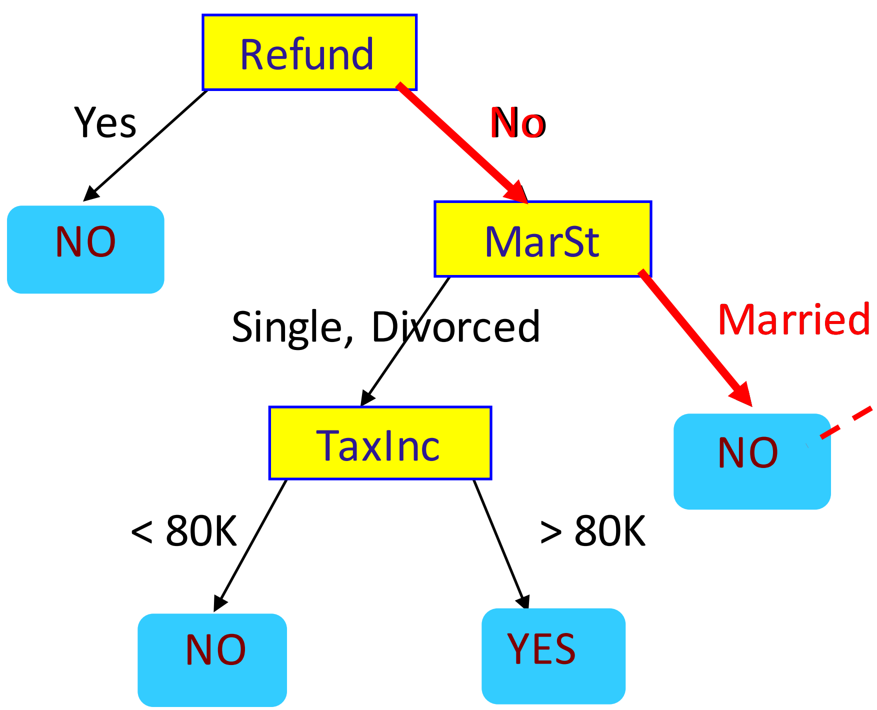
Query Data

Refund (X_1)	Marital Status (X_2)	Taxable Income (X_3)	Cheat (Y)
No	Married	80K	?

Decision Tree for Tax Fraud Detection

Query Data

X_1	X_2	X_3	Y
Refund	**Marital Status**	**Taxable Income**	**Cheat**
No	Married	80K	?



Assign Cheat to "No"

So far…

- What function does a decision tree represent
- Given a decision tree, how do we assign label to a test point

Now …

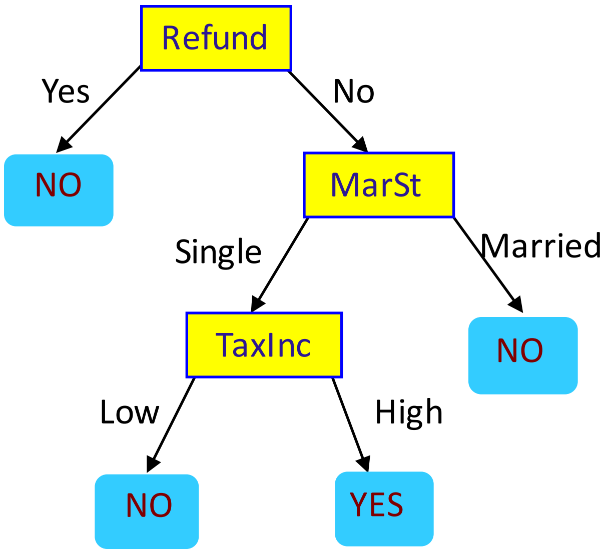
- How do we learn a decision tree from training data?

How to learn a decision tree

- Top-down induction [ID3]

Main loop:

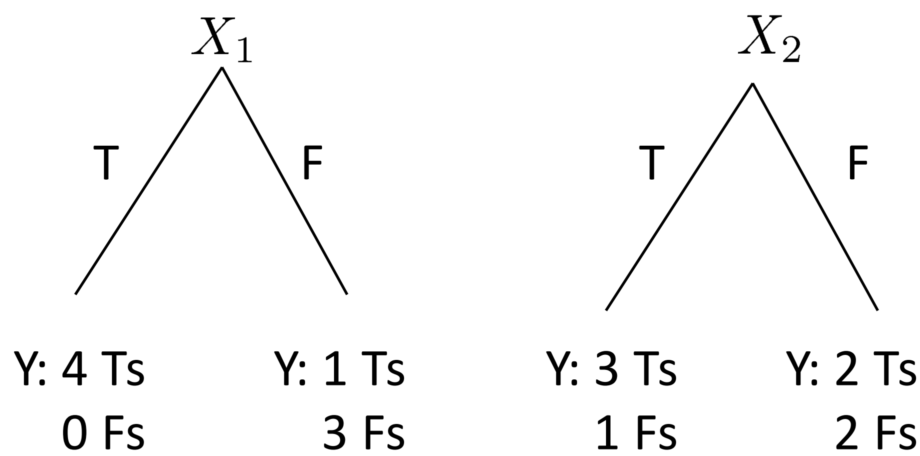
1. $X \leftarrow$ the "best" decision feature for next *node*
2. Assign X as decision feature for *node*
3. For each value of X, create new descendant of *node* (Discrete features)
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes (steps 1-5) after removing current feature



6. When all features exhausted, assign majority label to the leaf node

Which feature is best?

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F



Absolutely sure

Kind of sure

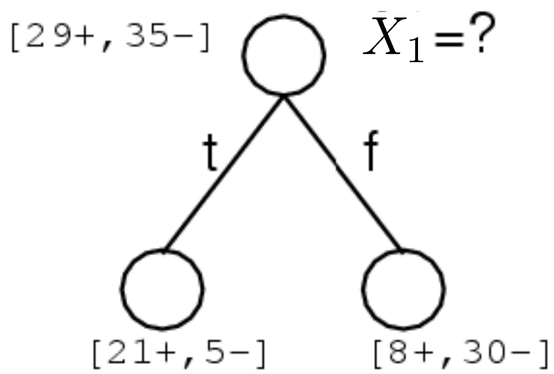
Kind of sure

Absolutely unsure

Good split if we are more certain about classification after split –

Uniform distribution of labels is bad

Which feature is best?



Pick the attribute/feature which yields maximum information gain:

$$\arg\max_i I(Y, X_i) = \arg\max_i [H(Y) - H(Y|X_i)]$$

$H(Y)$ – entropy of Y $\quad$ $H(Y|X_i)$ – conditional entropy of Y

Andrew Moore's Entropy in a Nutshell

Low Entropy

High Entropy

..the values (locations of soup) sampled entirely from within the soup bowl

..the values (locations of soup) unpredictable... almost uniformly sampled throughout our dining room

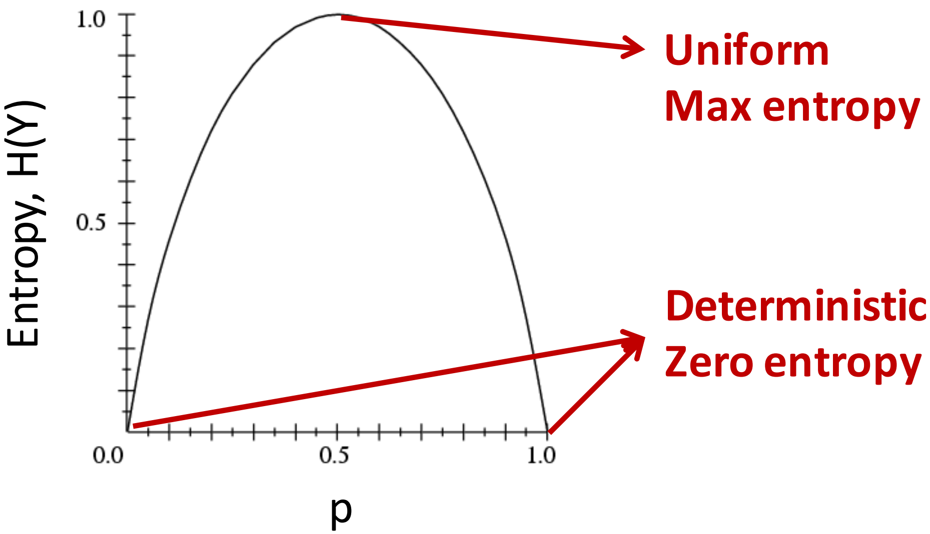
Entropy

- Entropy of a random variable Y

$$H(Y) = -\sum_{y} P(Y = y) \log_2 P(Y = y)$$

More uncertainty, more entropy!

Y ~ Bernoulli(p)



<u>**Information Theory interpretation**</u>: $H(Y)$ is the expected number of bits needed to encode a randomly drawn value of Y (under most efficient code)

Information Gain

- Advantage of attribute = decrease in uncertainty
  - Entropy of Y before split

$$H(Y) = -\sum_y P(Y=y)\log_2 P(Y=y)$$

  - Entropy of Y after splitting based on X_i
    - Weight by probability of following each branch

$$\begin{aligned} H(Y \mid X_i) &= \sum_x P(X_i = x) H(Y \mid X_i = x) \\ &= -\sum_x P(X_i = x) \sum_y P(Y = y \mid X_i = x) \log_2 P(Y = y \mid X_i = x) \end{aligned}$$

- Information gain is difference

$$I(Y, X_i) = H(Y) - H(Y \mid X_i)$$

Max Information gain = min conditional entropy

Which feature is best to split?

Pick the attribute/feature which yields maximum information gain:

$$\arg\max_i I(Y, X_i) = \arg\max_i [H(Y) - H(Y|X_i)]$$

$$= \arg\min_i H(Y|X_i)$$

Entropy of Y $\quad H(Y) = -\sum_y P(Y=y)\log_2 P(Y=y)$

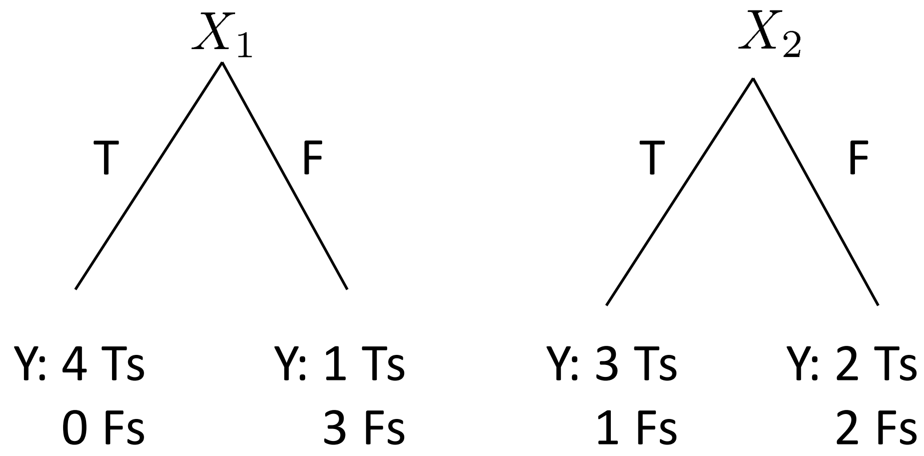
Conditional entropy of Y $\quad H(Y \mid X_i) = \sum_x P(X_i = x) H(Y \mid X_i = x)$

Feature which yields maximum reduction in entropy (uncertainty) provides maximum information about Y

Information Gain

$$H(Y \mid X_i) \;=\; -\sum_x P(X_i = x) \sum_y P(Y = y \mid X_i = x) \log_2 P(Y = y \mid X_i = x)$$

X_1	X_2	Y
T	T	T
T	F	T
T	T	T
T	F	T
F	T	T
F	F	F
F	T	F
F	F	F



$$\widehat{H}(Y|X_1) = -\frac{1}{2}[1 \log_2 1 + 0 \log_2 0] - \frac{1}{2}[\frac{1}{4}\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}]$$

$$\widehat{H}(Y|X_2) = -\frac{1}{2}[\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}] - \underbrace{\frac{1}{2}[\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}]}_{>0}$$

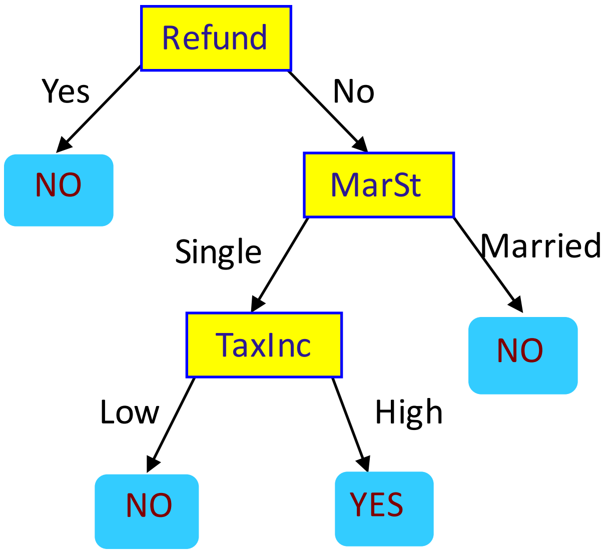
$$\widehat{H}(Y|X_1) < \widehat{H}(Y|X_2)$$

How to learn a decision tree

- Top-down induction [ID3]

Main loop:

1. $X \leftarrow$ the "best" decision feature for next *node*
2. Assign X as decision feature for *node*
3. For each value of X, create new descendant of *node* (Discrete features)
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes (steps 1-5) after removing current feature



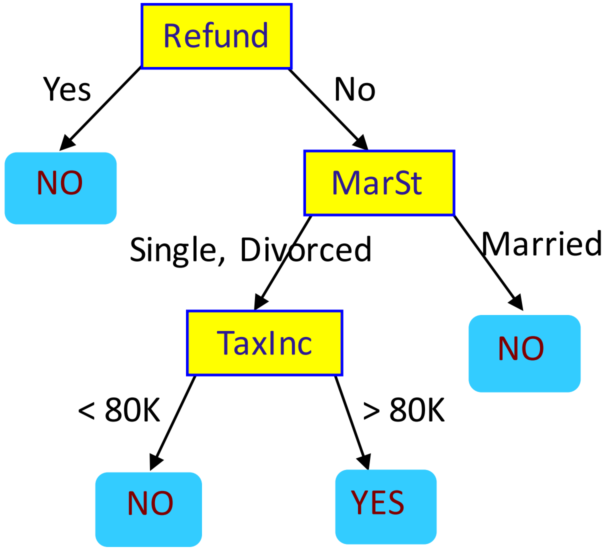
6. When all features exhausted, assign majority label to the leaf node

How to learn a decision tree

- Top-down induction [ID3, C4.5, C5, ...]

Main loop: C4.5

1. $X \leftarrow$ the "best" decision feature for next *node*
2. Assign X as decision feature for *node*
3. For "best" split of X, create new descendants of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes
6. Prune back tree to reduce overfitting
7. Assign majority label to the leaf node



Handling continuous features (C4.5)

Convert continuous features into discrete by setting a threshold.

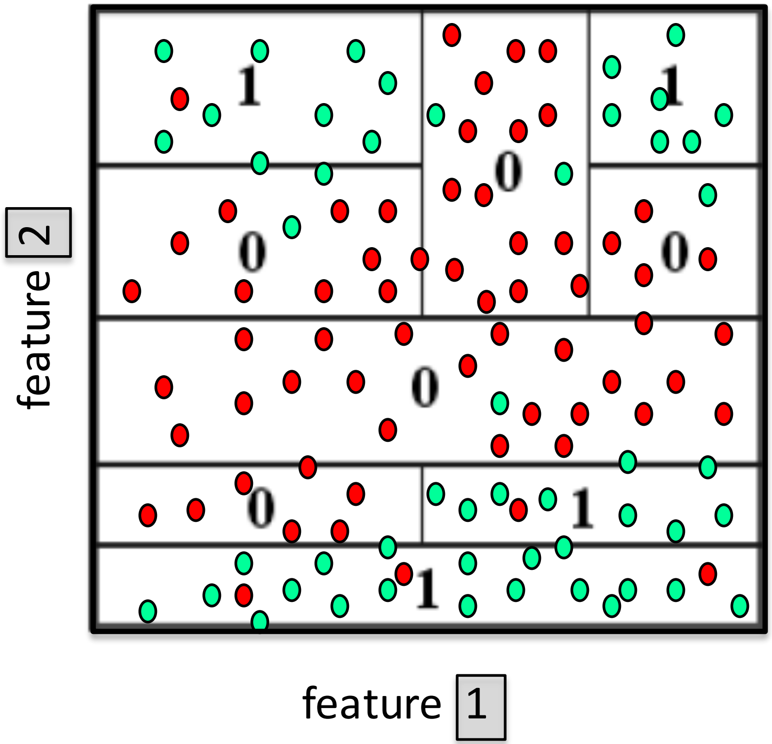
What threshold to pick?

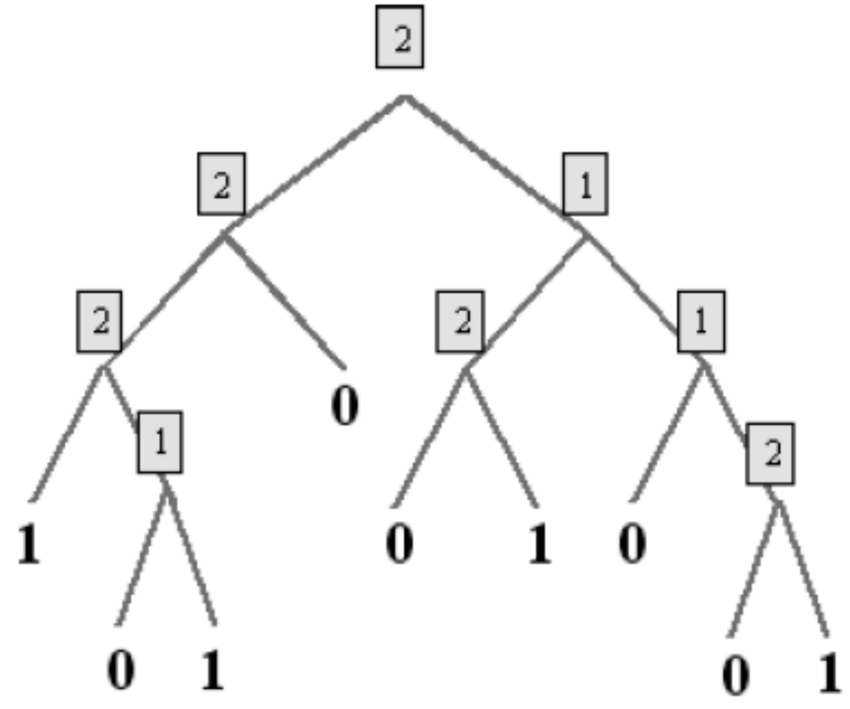
Search for best one as per information gain. Infinitely many??

Don't need to search over more than ~ n (number of training data),e.g. say X_1 takes values $x_1^{(1)}, x_1^{(2)}, \ldots, x_1^{(n)}$ in the training set. Then possible thresholds are

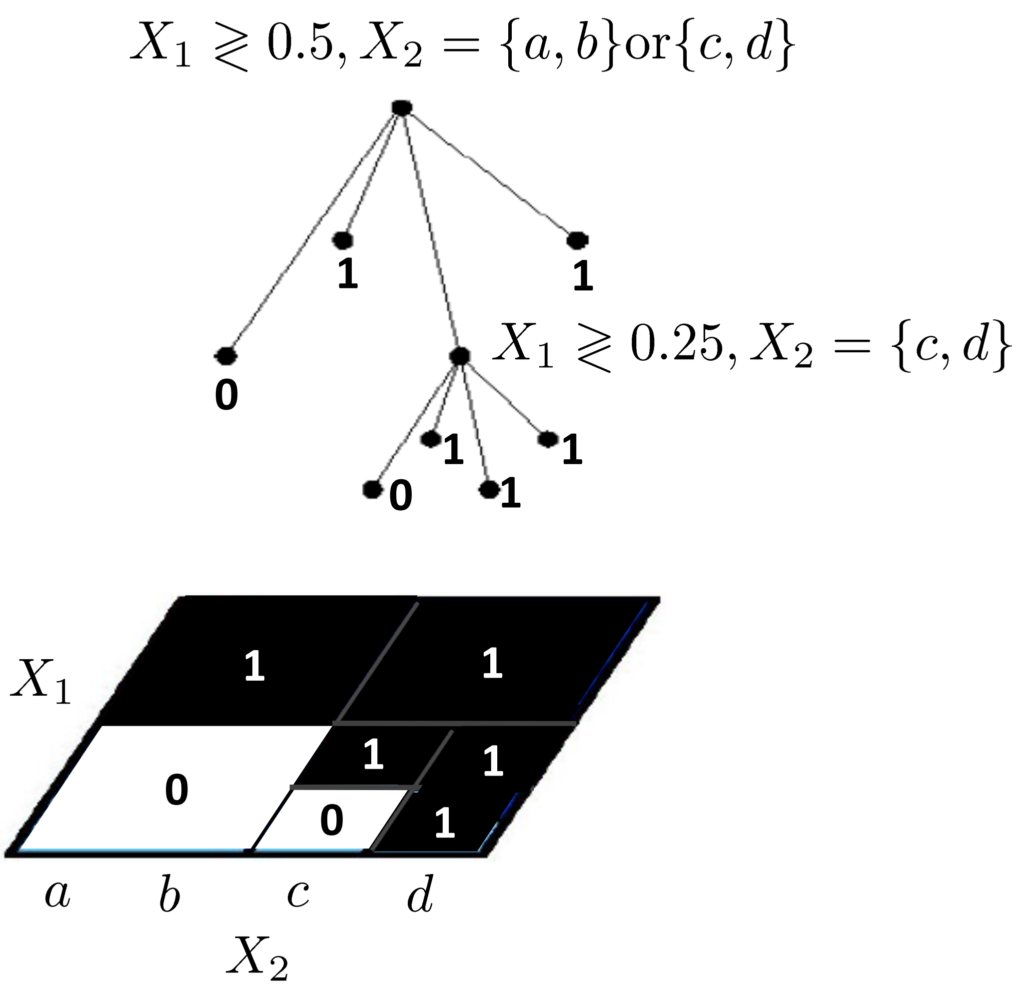
$$[x_1^{(1)} + x_1^{(2)}]/2,\ [x_1^{(2)} + x_1^{(3)}]/2,\ \ldots,\ [x_1^{(n-1)} + x_1^{(n)}]/2$$

Dyadic decision trees
(split on mid-points of features)





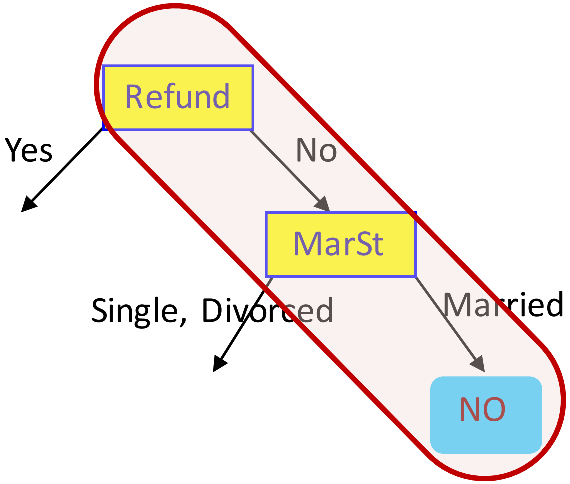
Decision Tree more generally…



- Features can be discrete, continuous or categorical
- Each internal node: test some set of features $\{X_i\}$
- Each branch from a node: selects a set of value for $\{X_i\}$
- Each leaf node: prediction for Y

When to Stop?

- Many strategies for picking simpler trees:
  - Pre-pruning
    - Fixed depth (e.g. ID3)
    - Fixed number of leaves
  - Post-pruning
    - Chi-square test
      - Convert decision tree to a set of rules
      - Eliminate variable values in rules which are independent of label (using chi-square test for independence)
      - Simplify rule set by eliminating unnecessary rules
  - Information Criteria: MDL(Minimum Description Length)



Information Criteria

- Penalize complex models by introducing cost

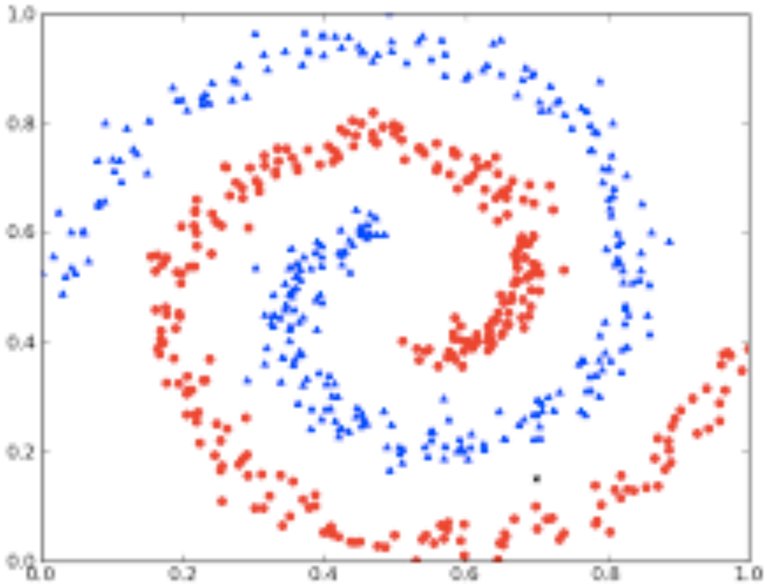
$$\hat{f} = \arg\min_T \underbrace{\left\{ \frac{1}{n} \sum_{i=1}^{n} \text{loss}(\hat{f}_T(X_i), Y_i)\right.}_{\text{log likelihood}} + \underbrace{\left.\text{pen}(T) \right\}}_{\text{cost}}$$

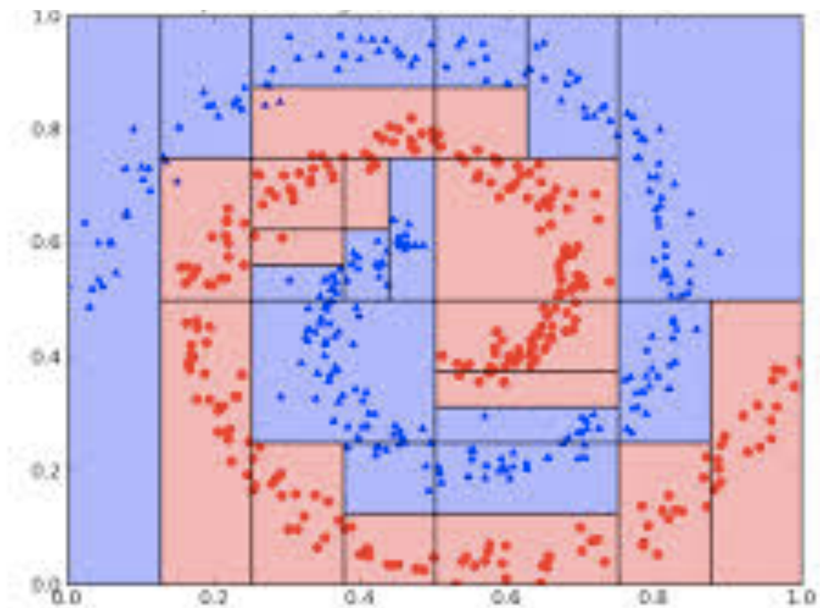
$$\begin{aligned} \text{loss}(\hat{f}_T(X_i), Y_i) &= (\hat{f}_T(X_i) - Y_i)^2 && \text{regression} \\ &= \mathbf{1}_{\hat{f}_T(X_i) \neq Y_i} && \text{classification} \end{aligned}$$

$\text{pen}(T) \propto |T|$ penalize trees with more leaves

CART – optimization can be solved by dynamic programming

Example of 2-feature decision tree classifier



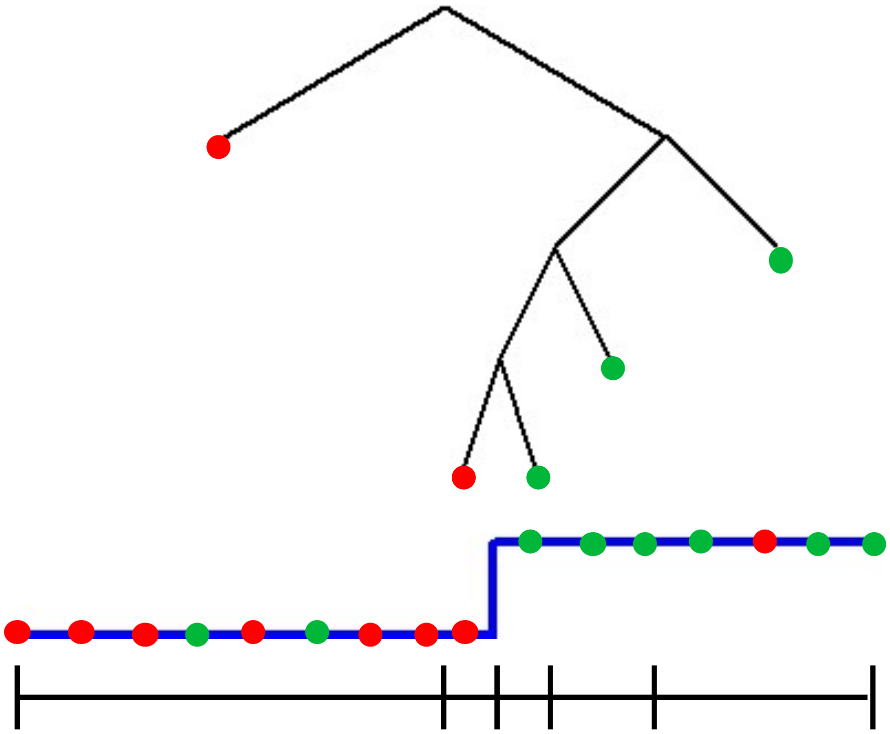


cs.uchicago.edu

How to assign label to each leaf

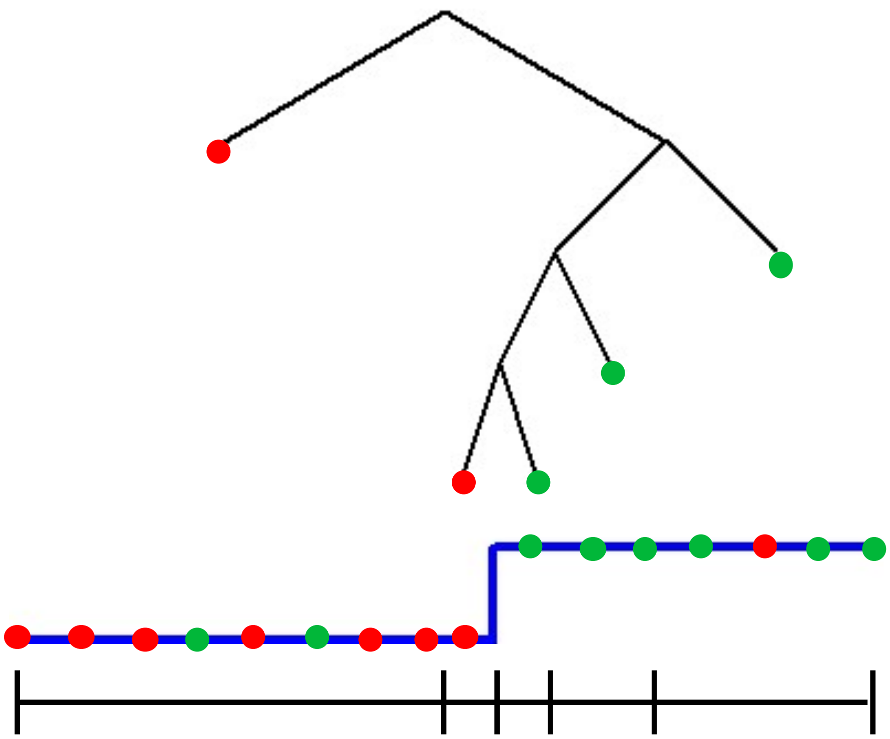
Classification – Majority vote

Regression – ?

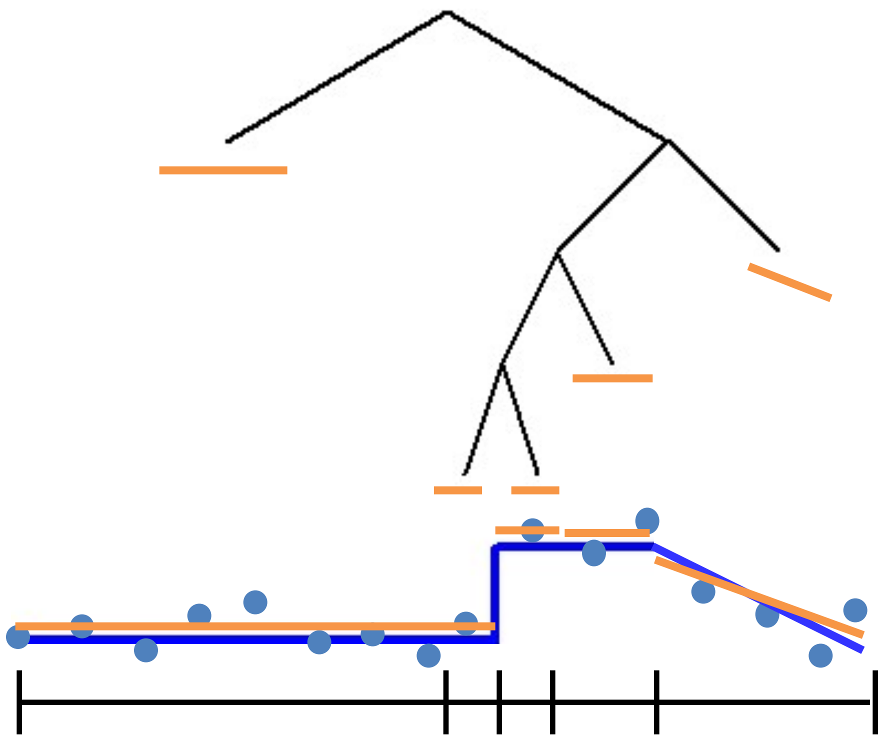


How to assign label to each leaf

Classification – Majority vote



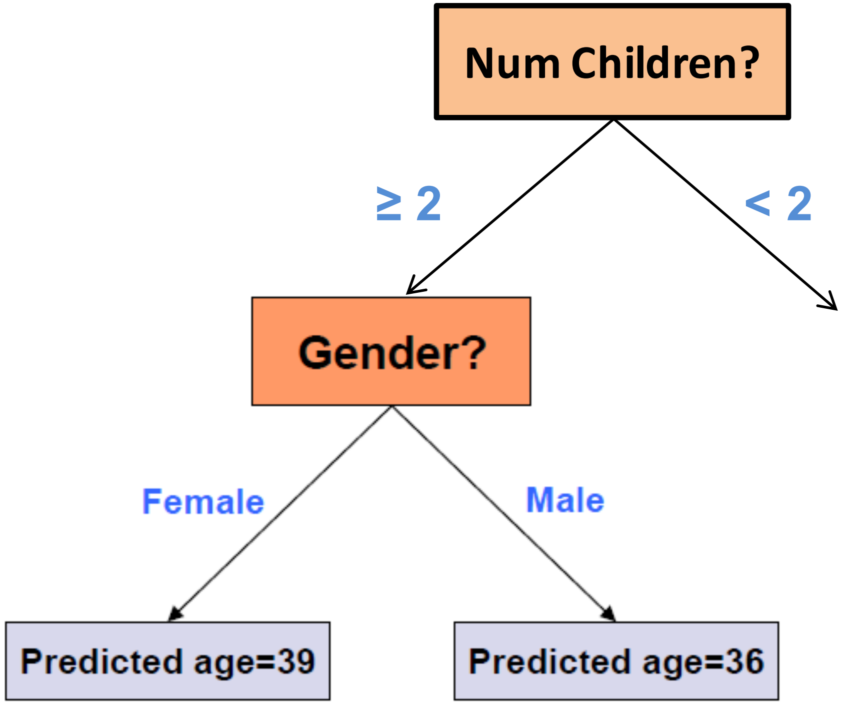
Regression – Constant/ Linear/Poly fit



Regression trees

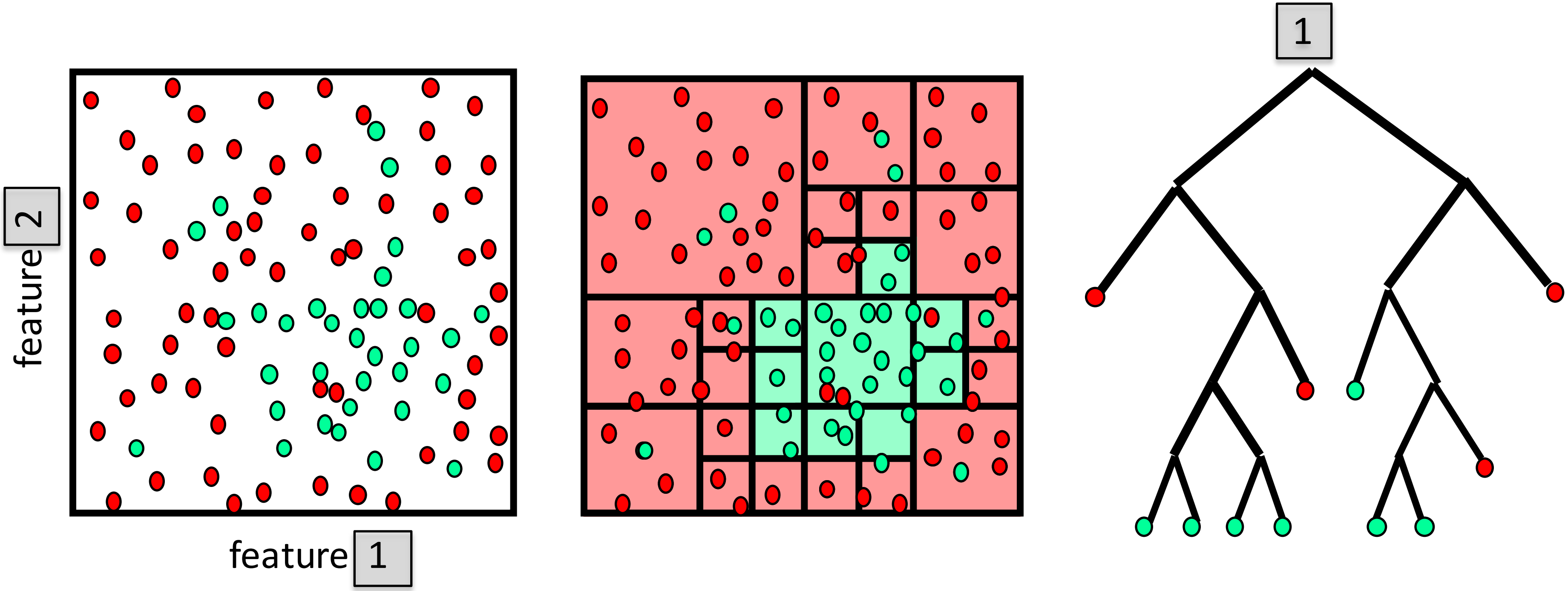
$X^{(1)}$ .... $X^{(p)}$ Y

Gender	Rich?	Num. Children	# travel per yr.	Age
F	No	2	5	38
M	No	0	2	25
M	Yes	1	0	72
:	:	:	:	:



Average (fit a constant) using training data at the leaves

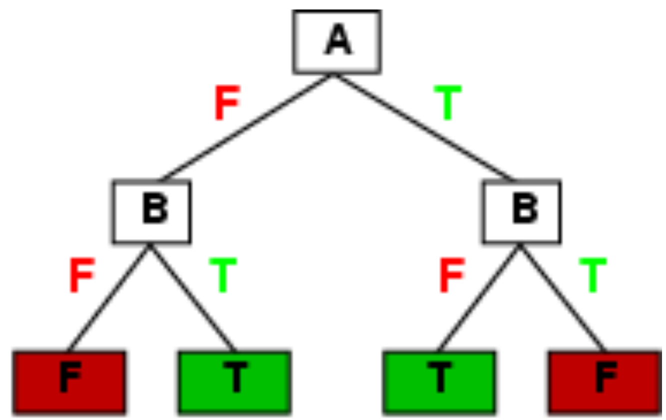
Example of decision tree classifier
with dyadic splits (mid-point of feature)



Expressiveness of Decision Trees

- Decision trees in general (without pruning) can express any function of the input features.
- E.g., for Boolean functions, truth table row → path to leaf:

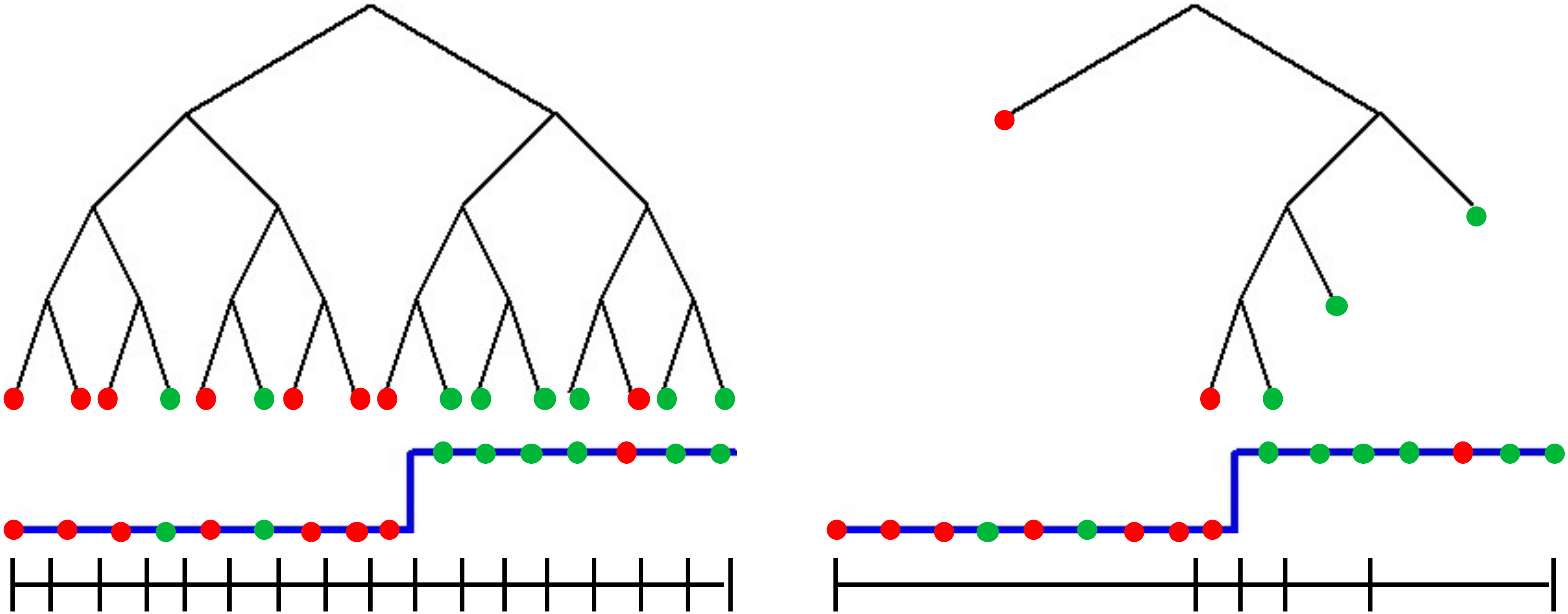
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- There is a decision tree which perfectly classifies a training set with one path to leaf for each example - overfitting
- But it won't generalize well to new examples - prefer to find more compact decision trees

Decision Trees - Overfitting

One training example per leaf – overfits, need compact/pruned decision tree



What you should know

- Decision trees are one of the most popular data mining tools
  - Interpretability
  - Ease of implementation
  - Good performance in practice (for small dimensions)
- Information gain to select attributes (ID3, C4.5,…)
- Can be used for classification, regression and density estimation too
- Decision trees will overfit!!!
  - Must use tricks to find “simple trees”, e.g.,
    - Pre-Pruning: Fixed depth/Fixed number of leaves
    - Post-Pruning: Chi-square test of independence
    - Complexity Penalized/MDL model selection