10-301/601: Introduction to Machine Learning Lecture 2 – Decision Trees

Henry Chai

5/18/22

Front Matter

- Announcements:
  - HW1 released 5/17 due 5/24 at 1 PM
  - Recitation 1 on 5/19: review of prerequisite material
  - General advice for the summer:
    - Start HWs early!
    - Go to office hours! Starting today, 5/18
- Recommended Readings:
  - Daumé III, Chapter 1: Decision Trees

Our second Machine Learning Classifier

- A **classifier** is a function that takes feature values as input and outputs a label
- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict the majority vote

Family History	Resting Blood Pressure	Cholesterol	Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

Notation

- Feature space, $\mathcal{X}$
- Label space, $\mathcal{Y}$
- (Unknown) Target function, $c^*: \mathcal{X} \to \mathcal{Y}$
- Training dataset:

$$\mathcal{D} = \left\{\left(\boldsymbol{x}^{(1)}, c^*\left(\boldsymbol{x}^{(1)}\right) = y^{(1)}\right), \left(\boldsymbol{x}^{(2)}, y^{(2)}\right) \dots, \left(\boldsymbol{x}^{(N)}, y^{(N)}\right)\right\}$$

- Data point:

$$\left(\boldsymbol{x}^{(n)}, y^{(n)}\right) = \left(x_1^{(n)}, x_2^{(n)}, \dots, x_D^{(n)}, y^{(n)}\right)$$

- Classifier, $h : \mathcal{X} \to \mathcal{Y}$
- Goal: find a classifier, h, that best approximates c^*

Evaluation

- Loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
  - Defines how "bad" predictions, $\hat{y} = h(\boldsymbol{x})$, are compared to the true labels, $y = c^*(\boldsymbol{x})$
  - Common choices

1. Squared loss (for regression): $\ell(y, \hat{y}) = (y - \hat{y})^2$
2. Binary or 0-1 loss (for classification):

$$\ell(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{otherwise} \end{cases}$$

Evaluation

- Loss function, $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$
  - Defines how "bad" predictions, $\hat{y} = h(\boldsymbol{x})$, are compared to the true labels, $y = c^*(\boldsymbol{x})$
  - Common choices

1. Squared loss (for regression): $\ell(y, \hat{y}) = (y - \hat{y})^2$
2. Binary or 0-1 loss (for classification):

$$\ell(y, \hat{y}) = \mathbb{1}(y \neq \hat{y})$$

- Error rate:

$$err(h, \mathcal{D}) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\left(y^{(n)} \neq \hat{y}^{(n)}\right)$$

Notation: Example

- Memorizer: if a set of features exists in the **training** dataset, predict its corresponding label; otherwise, predict the majority vote

	x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?	$\hat{y}$ Predictions
	Yes	Low	Normal	No	No
$\boldsymbol{x}^{(2)}$	No	Medium	Normal	No	No
	No	Low	Abnormal	Yes	Yes
	Yes	Medium	Normal	Yes	Yes
	Yes	High	Abnormal	Yes	Yes

- $N = 5$ and $D = 3$
- $\boldsymbol{x}^{(2)} = \left(x_1^{(2)} = \text{"No"}, x_2^{(2)} = \text{"Medium"}, x_3^{(2)} = \text{"Normal"}\right)$

Our second Machine Learning Classifier

- Memorizer:

```
def train(𝒟):
    store 𝒟
def majority_vote(𝒟):
    return mode(y^(1), y^(2), …, y^(N))
def predict(x′):
    if ∃ x^(n) ∈ 𝒟 s.t. x′ = x^(n):
        return y^(n)
    else
        return majority_vote(𝒟)
```

Our third Machine Learning Classifier

- Alright, let's actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

- Decision stump: based on a single feature, x_d, predict the most common label in the training dataset among all data points that have the same value for x_d

Our third Machine Learning Classifier: Example

- Alright, let's actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

- Decision stump on x_1:

$$h(\boldsymbol{x}') = h(x_1', \dots, x_D') = \begin{cases} ??? & \text{if } x_1' = \text{"Yes"} \\ ??? & \text{otherwise} \end{cases}$$

Our third Machine Learning Classifier: Example

- Alright, let's actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

- Decision stump on x_1:

$$h(\boldsymbol{x}') = h(x_1', \dots, x_D') = \begin{cases} \text{"Yes" if } x_1' = \text{"Yes"} \\ ??? \text{ otherwise} \end{cases}$$

Our third Machine Learning Classifier: Example

- Alright, let's actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

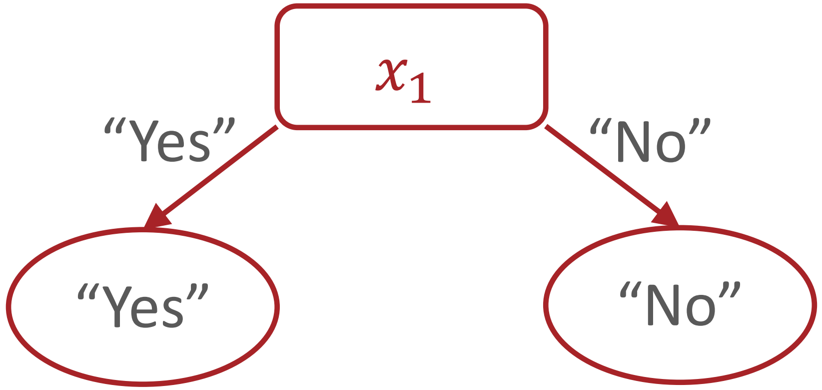
- Decision stump on x_1:

$$h(\boldsymbol{x}') = h(x_1', \dots, x_D') = \begin{cases} \text{“Yes” if } x_1' = \text{“Yes”} \\ \text{“No” otherwise} \end{cases}$$

Our third Machine Learning Classifier: Example

- Alright, let's actually (try to) extract a pattern from the data

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?	$\hat{y}$ Predictions
Yes	Low	Normal	No	Yes
No	Medium	Normal	No	No
No	Low	Abnormal	Yes	No
Yes	Medium	Normal	Yes	Yes
Yes	High	Abnormal	Yes	Yes



Decision Stumps: Pseudocode

def train($\mathcal{D}$):

1. pick a feature, x_d
2. split $\mathcal{D}$ according to x_d

 for v in $V(x_d)$, all possible values of x_d:

$$\mathcal{D}_v = \left\{\left(x^{(i)}, y^{(i)}\right) \in \mathcal{D} \mid x_d^{(i)} = v\right\}$$

3. Compute the majority vote for each split

 for v in $V(x_d)$, all possible values of x_d:

 $\hat{y}_v =$ majority_vote($\mathcal{D}_v$)

def predict($\boldsymbol{x}'$):

 for v in $V(x_d)$, all possible values of x_d:

 if $\boldsymbol{x}' = v$: return $\hat{y}_v$

Decision Stumps: Questions

1. How can we pick which feature to split on?

Lecture 2 Polls

0 done

0 underway

Which feature do you think we should split on for this data set?

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes

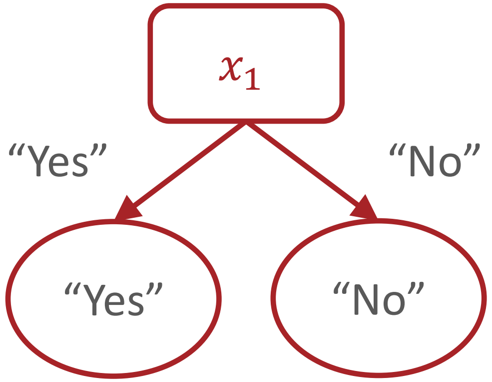
- x_1
- x_2
- x_3

Splitting Criterion

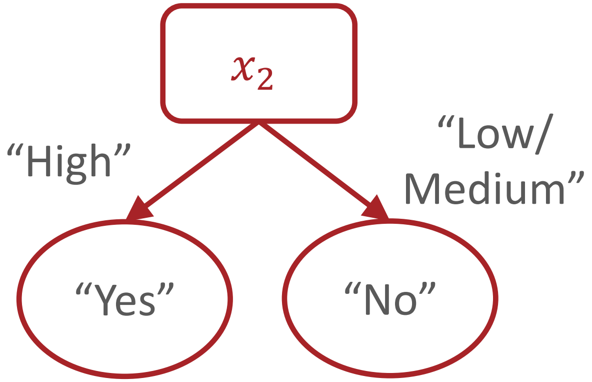
- A **splitting criterion** is a function that measures how good or useful splitting on a particular feature is *for a specified dataset*
- Insight: use the feature that optimizes the splitting criterion for our decision stump.

Training error rate as a Splitting Criterion

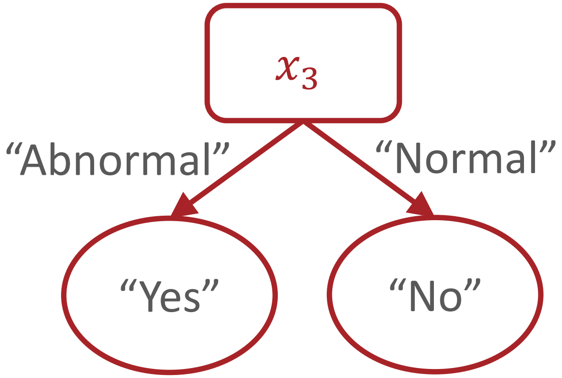
x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes



Training error rate: 2/5



Training error rate: 2/5

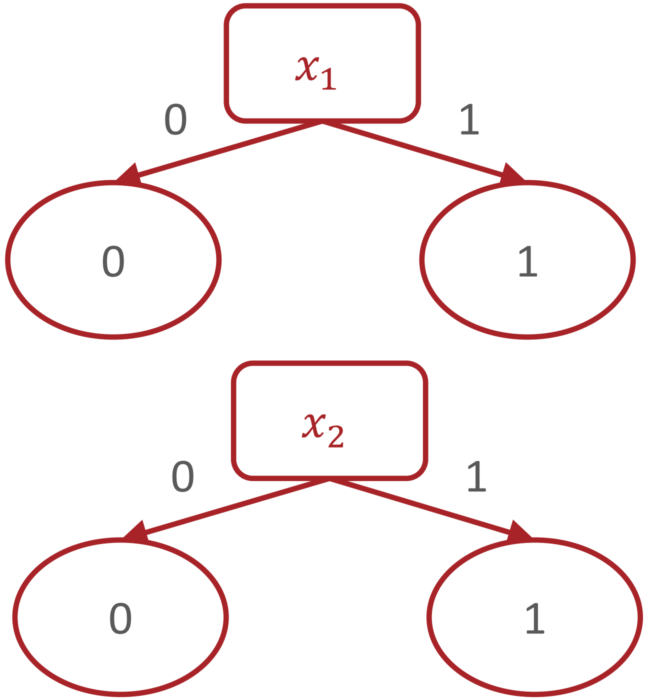


Training error rate: 1/5

Training error rate as a Splitting Criterion?

x_1	x_2	y
1	0	0
1	0	0
1	0	1
1	0	1
1	1	1
1	1	1
1	1	1
1	1	1

- Which feature would you split on using training error rate as the splitting criterion?



Training error rate: 2/8

Splitting Criterion

- A **splitting criterion** is a function that measures how good or useful splitting on a particular feature is *for a specified dataset*
- Insight: use the feature that optimizes the splitting criterion for our decision stump.
- Potential splitting criteria:
  - Training error rate (minimize)
  - Gini impurity (minimize) → CART algorithm
  - Mutual information (maximize) → ID3 algorithm

Splitting Criterion

- A **splitting criterion** is a function that measures how good or useful splitting on a particular feature is *for a specified dataset*
- Insight: use the feature that optimizes the splitting criterion for our decision stump.
- Potential splitting criteria:
  - Training error rate (minimize)
  - Gini impurity (minimize) → CART algorithm
  - <u>Mutual information</u> (maximize) → ID3 algorithm

Entropy

- Entropy describes the purity or uniformity of a collection of values: the lower the entropy, the more pure

$$H(S) = -\sum_{v \in V(S)} \frac{|S_v|}{|S|} \log_2\left(\frac{|S_v|}{|S|}\right)$$

where S is a collection of values,

$V(S)$ is the set of unique values in S

S_v is the collection of elements in S with value v

- If all the elements in S are the same, then

$$\mathrm{H}(S) = -1\log_2(1) = 0$$

Entropy

- Entropy describes the purity or uniformity of a collection of values: the lower the entropy, the more pure

$$H(S) = -\sum_{v \in V(S)} \frac{|S_v|}{|S|} \log_2\left(\frac{|S_v|}{|S|}\right)$$

where S is a collection of values,

$V(S)$ is the set of unique values in S

S_v is the collection of elements in S with value v

- If S is split fifty-fifty between two values, then

$$\mathrm{H}(S) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = -\log_2\left(\frac{1}{2}\right) = 1$$

Mutual Information

- Mutual information describes how much information or clarity a particular feature provides about the label

$$I(x_d, Y) = H(Y) - \sum_{v \in V(x_d)} (f_v)\left(H\left(Y_{x_d=v}\right)\right)$$

where x_d is a feature

Y is the collection of all labels

$V(x_d)$ is the set of unique values of x_d

f_v is the fraction of inputs where $x_d = v$

$Y_{x_d=v}$ is the collection of labels where $x_d = v$

Mutual Information: Example

x_d	y
1	1
1	1
0	0
0	0

$$I(x_d, Y) = H(Y) - \sum_{v \in V(x_d)} (f_v)\left(H\left(Y_{x_d=v}\right)\right)$$

$$= 1 - \frac{1}{2} H\left(Y_{x_d=0}\right) - \frac{1}{2} H\left(Y_{x_d=1}\right)$$

$$= 1 - \frac{1}{2}(0) - \frac{1}{2}(0) = 1$$

Mutual Information: Example

x_d	y
1	1
0	1
1	0
0	0

$$I(x_d, Y) = H(Y) - \sum_{v \in V(x_d)} (f_v)\left(H\left(Y_{x_d=v}\right)\right)$$

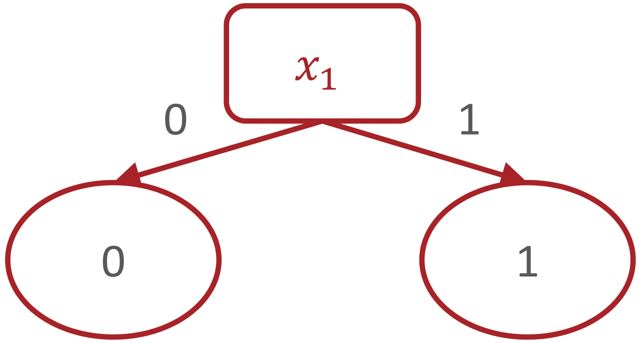
$$= 1 - \frac{1}{2} H\left(Y_{x_d=0}\right) - \frac{1}{2} H\left(Y_{x_d=1}\right)$$

$$= 1 - \frac{1}{2}(1) - \frac{1}{2}(1) = 0$$

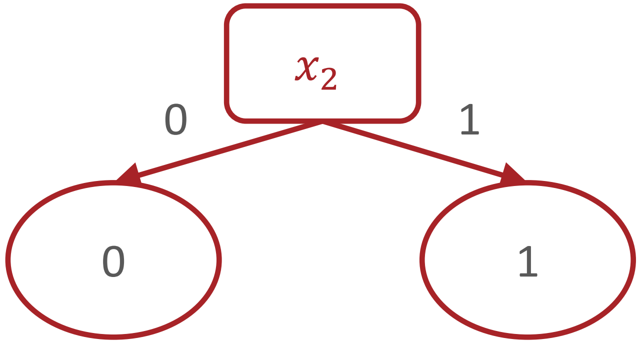
Mutual Information as a Splitting Criterion

x_1	x_2	y
1	0	0
1	0	0
1	0	1
1	0	1
1	1	1
1	1	1
1	1	1
1	1	1

- Which feature would you split on using mutual information as the splitting criterion?



Mutual Information: 0

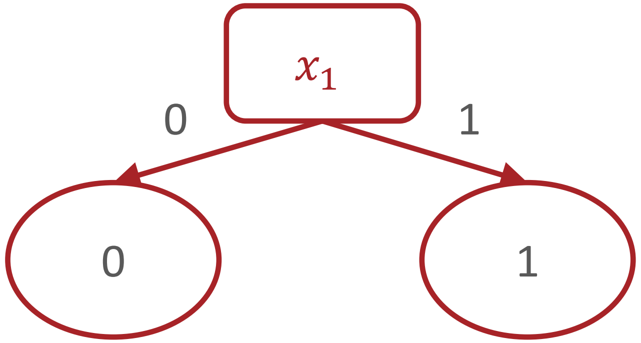


Mutual Information: $H(Y) - \frac{1}{2}H\left(Y_{x_2=0}\right) - \frac{1}{2}H\left(Y_{x_2=1}\right)$

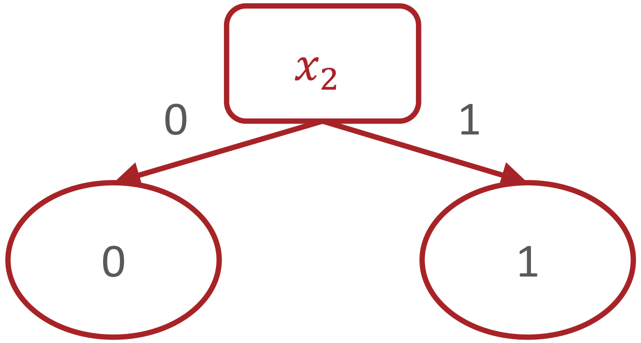
Mutual Information as a Splitting Criterion

x_1	x_2	y
1	0	0
1	0	0
1	0	1
1	0	1
1	1	1
1	1	1
1	1	1
1	1	1

- Which feature would you split on using mutual information as the splitting criterion?



Mutual Information: 0



Mutual Information: $-\frac{2}{8}\log_2\frac{2}{8}-\frac{6}{8}\log_2\frac{6}{8}-\frac{1}{2}(1)-\frac{1}{2}(0)\approx 0.31$

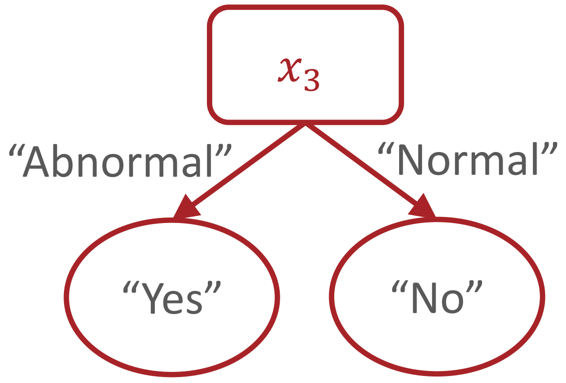
Decision Stumps: Questions

1. How can we pick which feature to split on?

2. Why stop at just one feature?

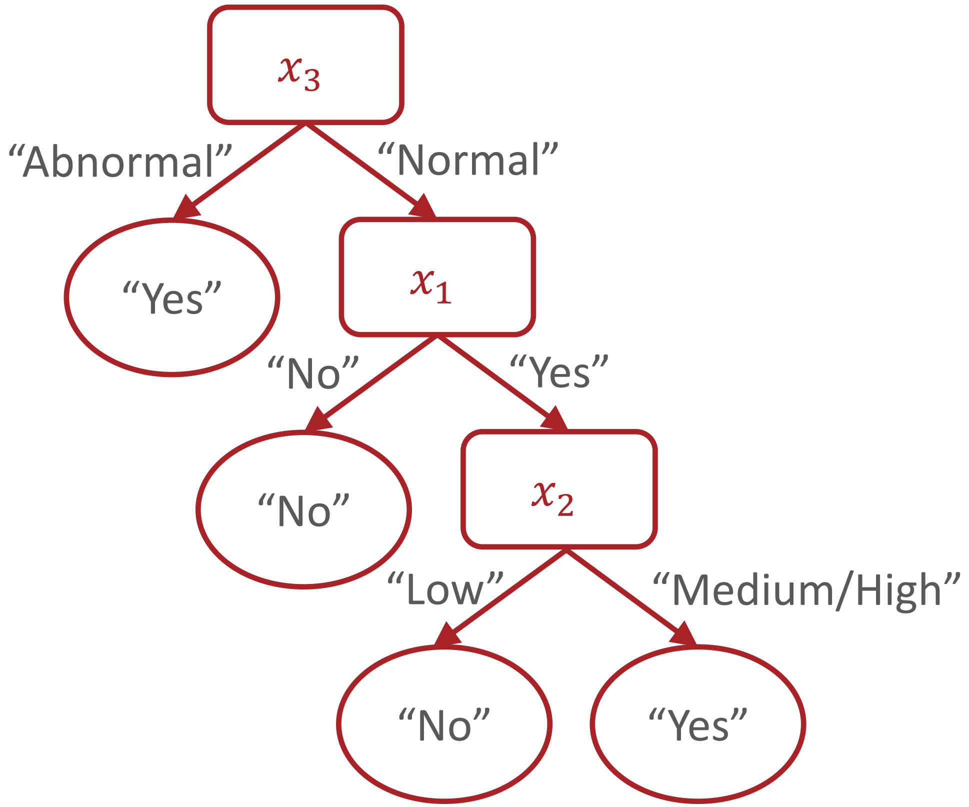
From Decision Stump …

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes



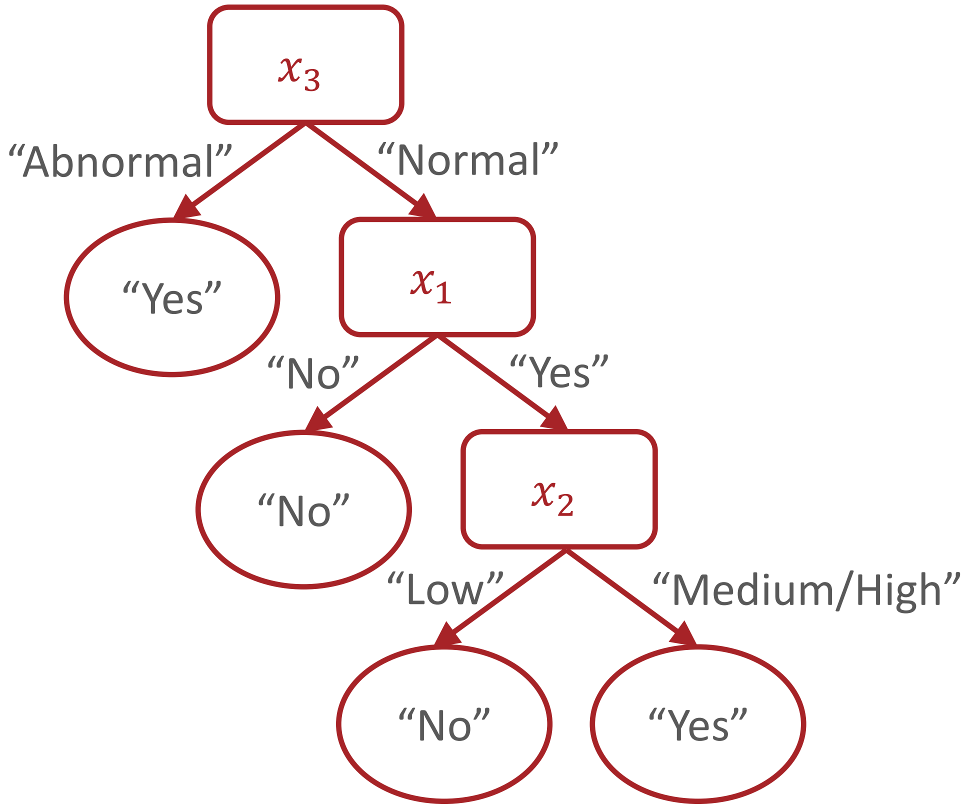
From Decision Stump to Decision Tree

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes



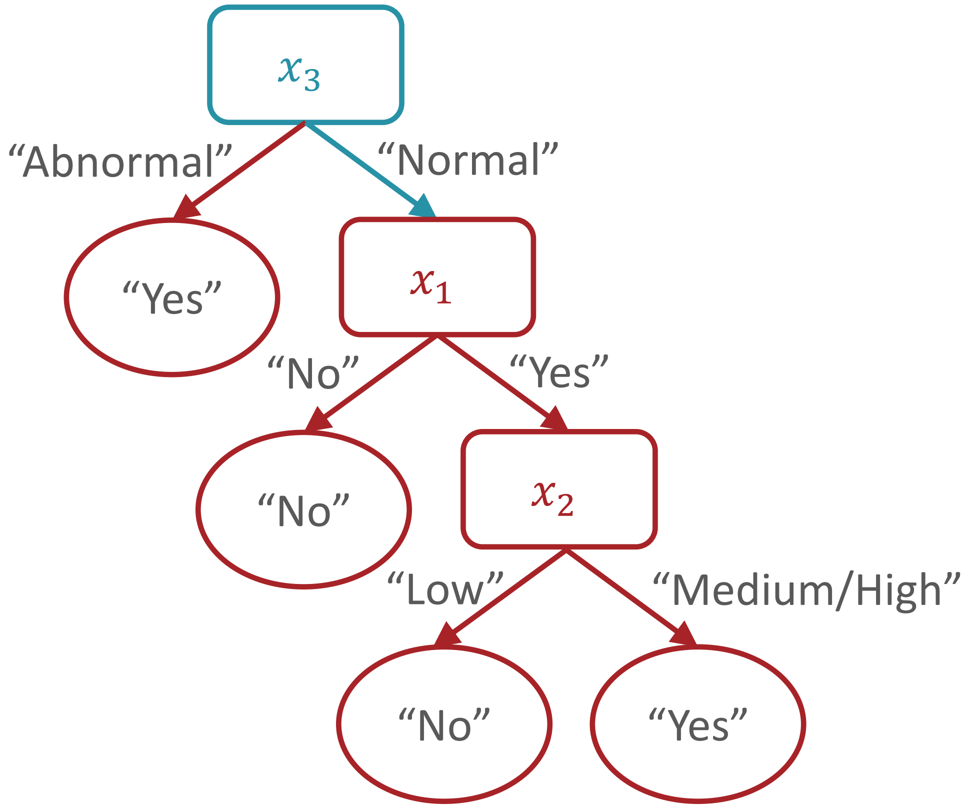
From Decision Stump to Decision Tree

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes
No	High	Normal	No



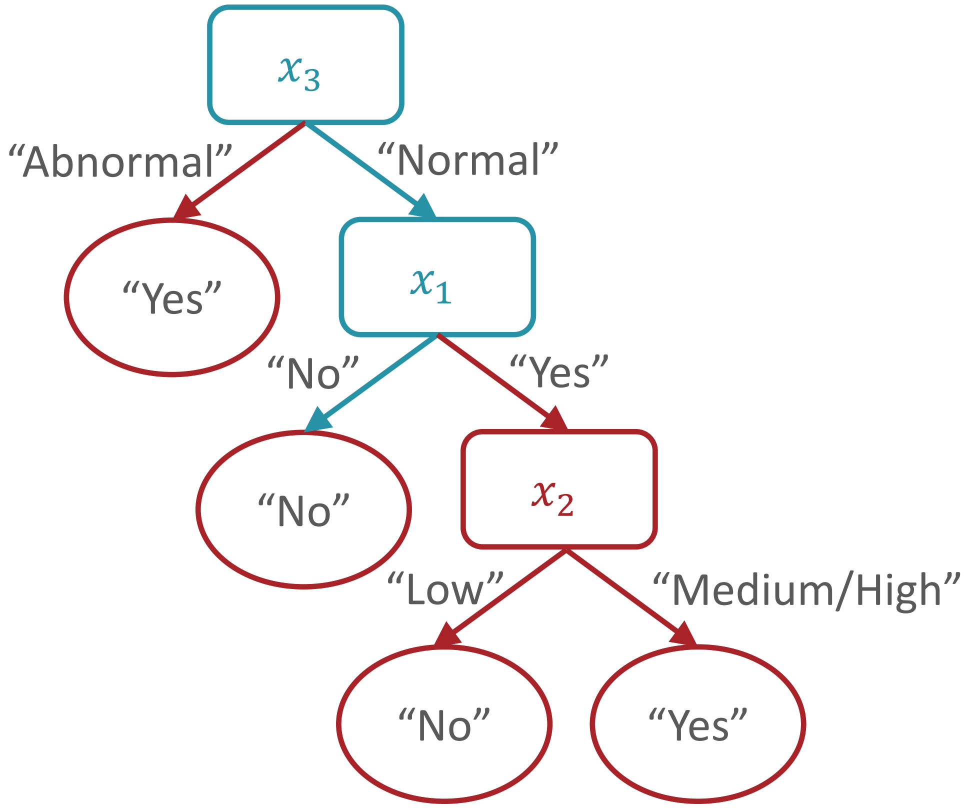
From Decision Stump to Decision Tree

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes
No	High	Normal	No



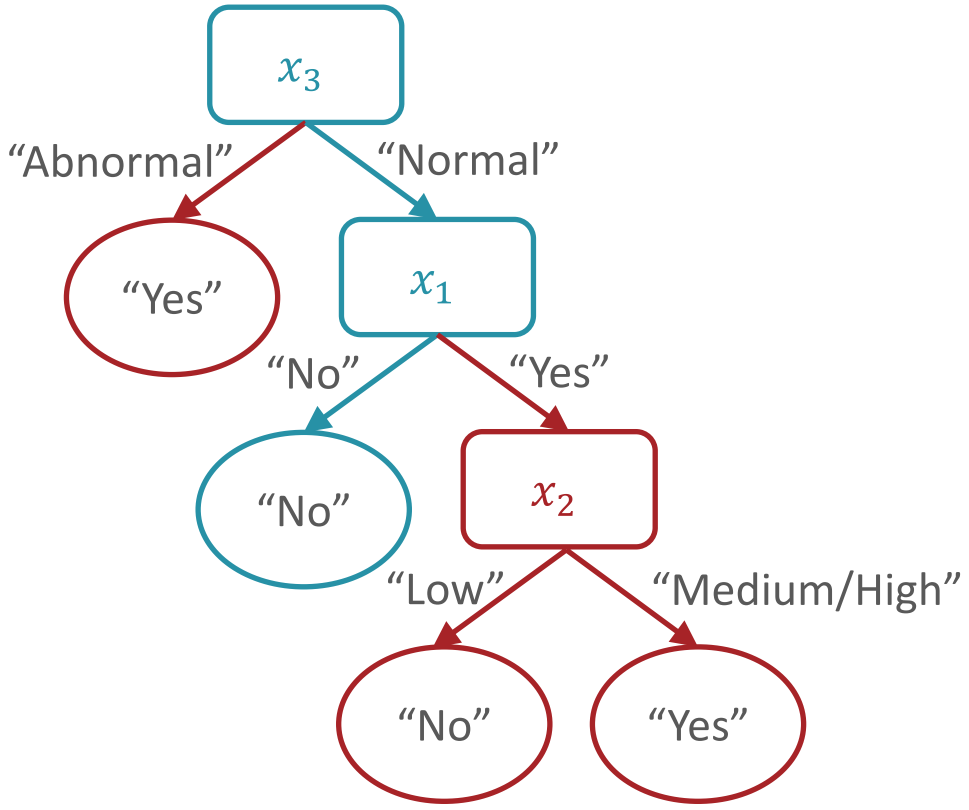
From Decision Stump to Decision Tree

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes
No	High	Normal	No



From Decision Stump to Decision Tree

x_1 Family History	x_2 Resting Blood Pressure	x_3 Cholesterol	y Heart Disease?
Yes	Low	Normal	No
No	Medium	Normal	No
No	Low	Abnormal	Yes
Yes	Medium	Normal	Yes
Yes	High	Abnormal	Yes
No	High	Normal	No



Decision Tree: Example

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```


Figure courtesy of Tom Mitchell

Decision Tree: Pseudocode

```
def predict(x'):
 - walk from root node to a leaf node
   while(true):
      if current node is internal (non-leaf):
           check the associated attribute, x_d
           go down branch according to x'_d
      if current node is a leaf node:
           return label stored at that leaf
```

Decision Tree: Pseudocode

```
def train(𝒟):
    store root = tree_recurse(𝒟)
def tree_recurse(𝒟'):
    q = new node()
    base case – if (SOME CONDITION):
    recursion – else:
        find best attribute to split on, x_d
        q.split = x_d
        for v in V(x_d), all possible values of x_d:
            𝒟_v = {(x^(i), y^(i)) ∈ 𝒟 | x_d^(i) = v}
            q.children(v) = tree_recurse(𝒟_v)
    return q
```

$$\mathcal{D}_v = \left\{ \left(x^{(i)}, y^{(i)} \right) \in \mathcal{D} \mid x_d^{(i)} = v \right\}$$

Decision Tree: Pseudocode

```
def train(𝒟):
    store root = tree_recurse(𝒟)
def tree_recurse(𝒟′):
    q = new node()
    base case – if (𝒟′ is empty OR
        all labels in 𝒟′ are the same OR
        all features in 𝒟′ are identical OR
        some other stopping criterion):
        q.label = majority_vote(𝒟′)


    recursion – else:
    return q
```

Decision Trees: Pros & Cons

- Pros
  - Interpretable
  - Efficient (computational cost and storage)
  - Can be used for classification and regression tasks
  - Compatible with categorical and real-valued features
- Cons
  - Learned greedily: each split only considers the immediate impact on the splitting criterion
    - Not guaranteed to find the smallest (fewest number of splits) tree that achieves a training error rate of 0.
  - Liable to overfit!

Decision Trees: Inductive Bias

- The **inductive bias** of a machine learning algorithm is the principal by which it generalizes to unseen examples
- What is the inductive bias of the ID3 algorithm?
  - Try to find the smallest tree that achieves a training error rate of 0 with high mutual information features at the top
- Occam's razor: try to find the "simplest" (e.g., smallest decision tree) classifier that explains the training dataset

Overfitting

- Overfitting occurs when the classifier (or model)…
  - is too complex
  - fits noise or “outliers” in the training dataset as opposed to the actual pattern of interest
  - doesn’t have enough inductive bias pushing it to generalize
- Underfitting occurs when the classifier (or model)…
  - is too simple
  - can’t capture the actual pattern of interest in the training dataset
  - has too much inductive bias

Overfitting in Decision Trees

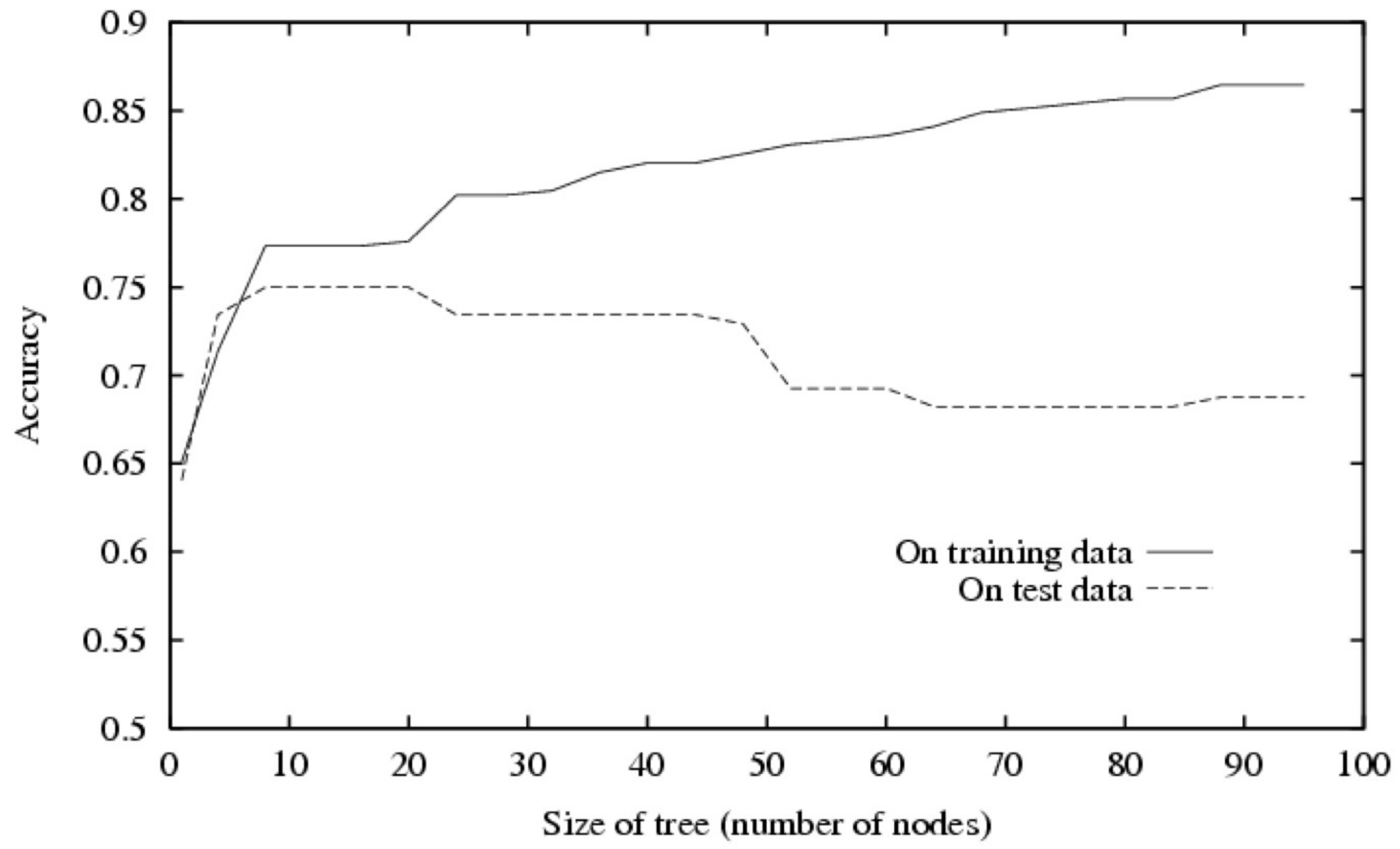


Figure courtesy of Tom Mitchell

Combatting Overfitting in Decision Trees

- Heuristics:
  - Do not split leaves past a fixed depth, δ
  - Do not split leaves with fewer than c data points
  - Do not split leaves where the maximal information gain is less than τ
  - Take a majority vote in impure leaves

Combatting Overfitting in Decision Trees

- Pruning:
  - First, learn a decision tree
  - Then, evaluate each split using a "validation" dataset by comparing the validation error rate with and without that split
  - Greedily remove the split that most decreases the validation error rate
  - Stop if no split is removed

Pruning Decision Trees

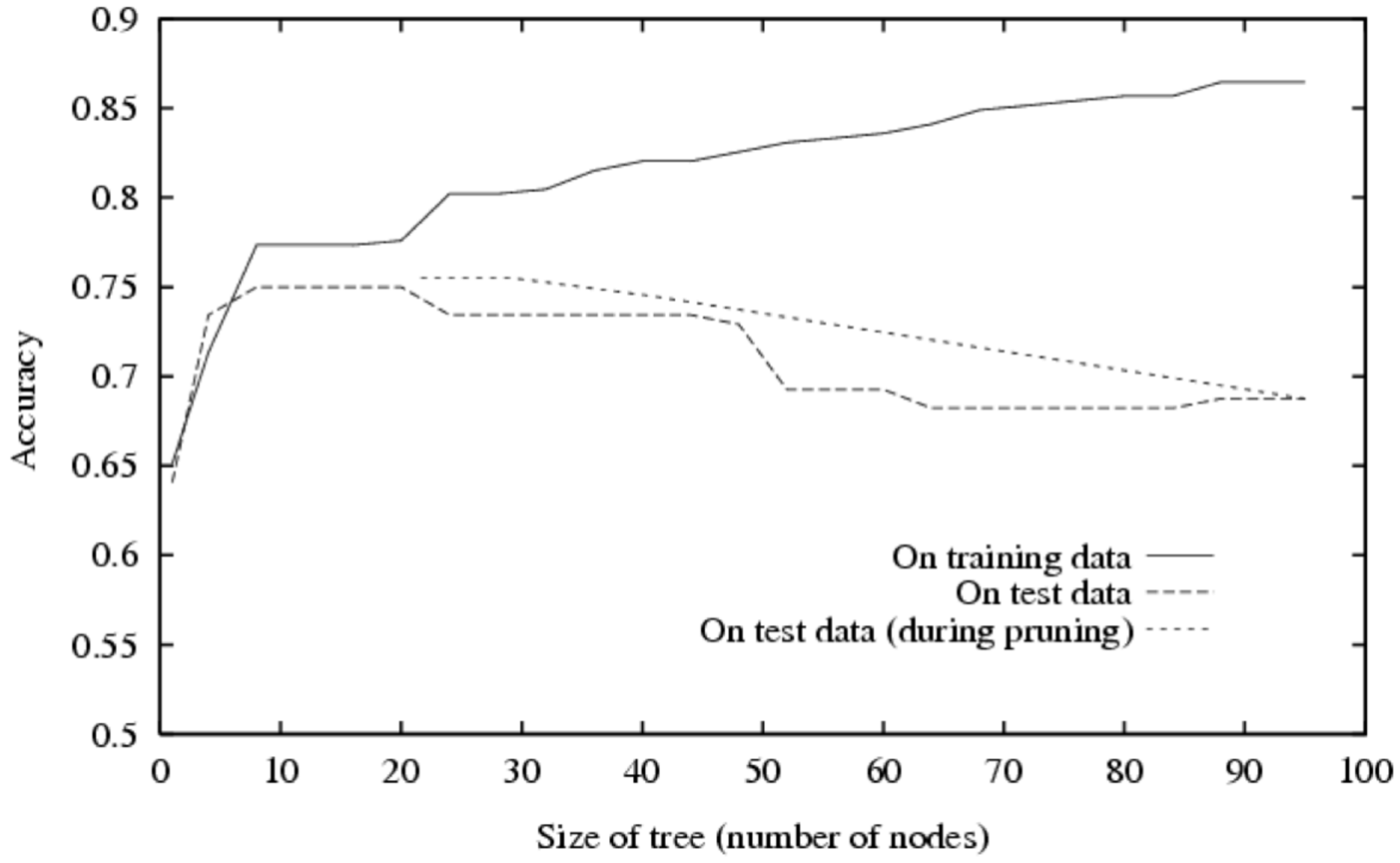


Figure courtesy of Tom Mitchell

Key Takeaways

- Mutual information as a splitting criterion for decision stumps/trees
- Decision tree algorithm via recursion
- Inductive bias of decision trees
- Overfitting vs. Underfitting
- How to combat overfitting in decision trees