Introduction to Artificial Intelligence (AI)

Computer Science cpsc502, Lecture 15

Nov, 1, 2011

Slide credit: C. Conati, S. Thrun, P. Norvig, Wikipedia

Supervised ML: Formal Specification

ONE EXAMPLE

$$
\begin{array}{ccccc}
\boxed{X_{11} & X_{12} & \cdots & X_{1N} & \to Y_1} \\
X_{21} & \cdots & & X_{2N} & \to Y_2 \\
\vdots & & & & \\
X_{M1} & \cdots & & X_{MN} & \to Y_M
\end{array}
$$

N features

M examples

$f(\bar{X}) \to Y$

Example Classification Data

	Y	X_1	X_2	X_3	X_4
	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

$f(\text{Known}, \text{new}, \text{long}, \text{work})$?

Today Nov 1

- **Supervised Machine Learning**
  - Naïve Bayes
  - Markov-Chains
  - Decision Trees Classification y discrete
  - Regression y continuous
  - Logistic Regression $y \rightsquigarrow z \in [0,1]$
- **Key Concepts**
  - Over-fitting
  - Evaluation

Example Classification Data

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

We want to classify new examples on property ***Action*** based on the examples' *Author, Thread, Length*, and *Where.*

Learning task

- **Inductive inference**
  - Given a set of examples of

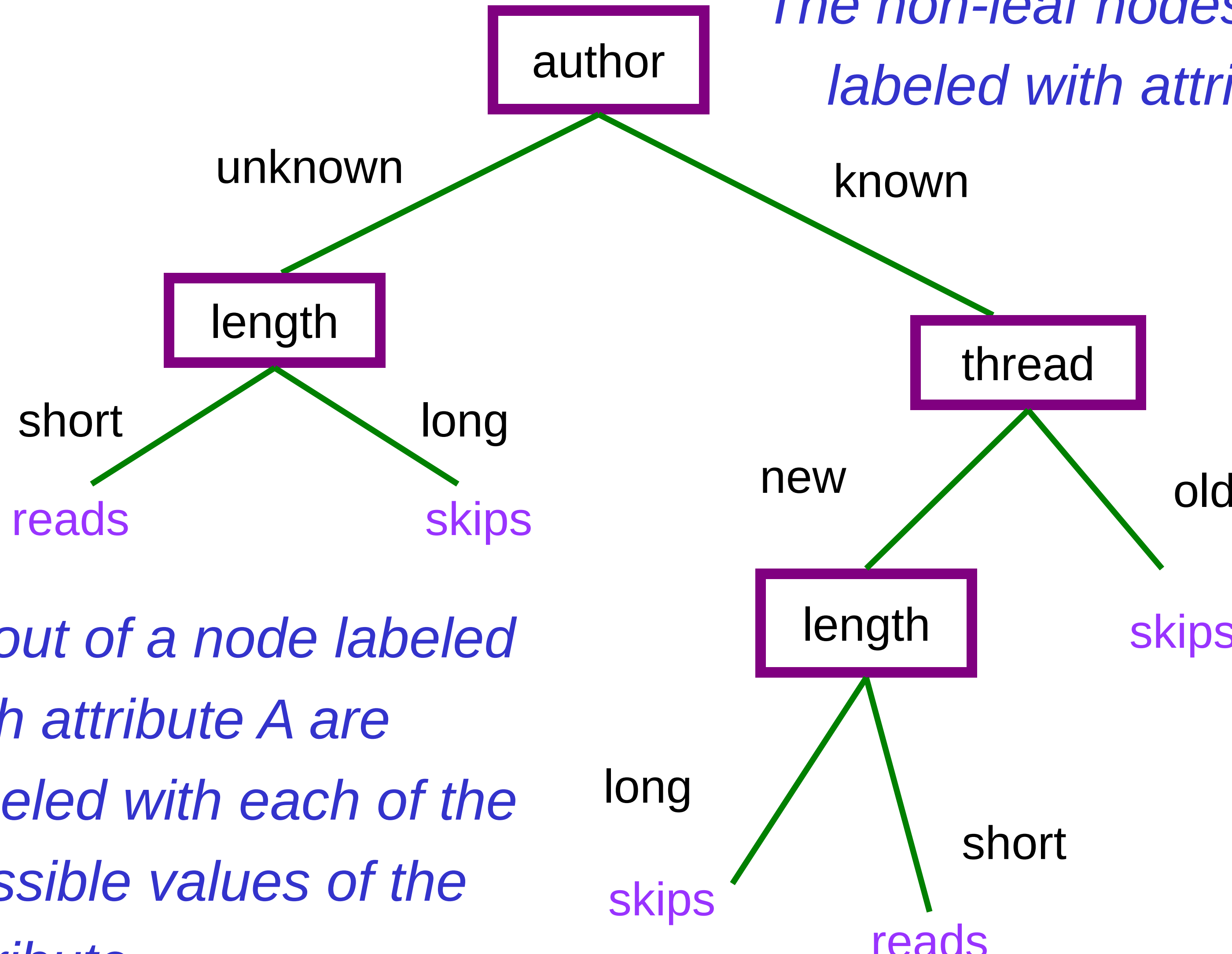
    f(author,thread, length, where) = {reads,skips}

  - Find a function *h(author,thread, length, where)* that approximates *f*

Example Decision Tree

The non-leaf nodes are labeled with attributes.

y ~ Action



Arcs out of a node labeled with attribute A are labeled with each of the possible values of the attribute

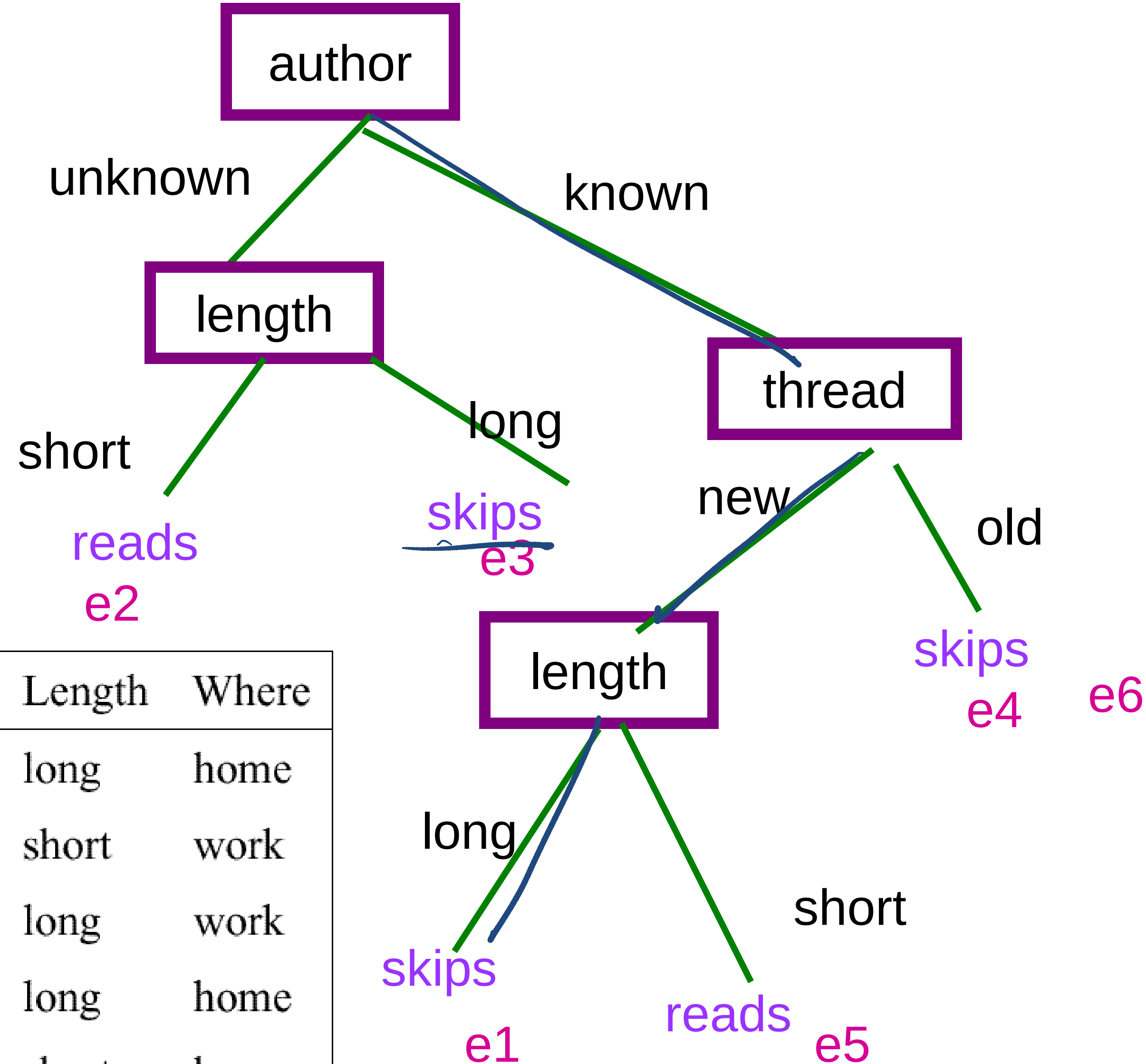
The leaves of the tree are labeled with classifications.

DT as classifiers

To classify an example, filter in down the tree

- For each attribute of the example, follow the branch corresponding to that attribute’s value.
- When a leaf is reached, the example is classified as the label for that leaf.

DT as classifiers



	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

DT Applications

- **DT are often the first method tried in many areas of industry and commerce**, when task involves learning from a data set of examples
- Main reason: **the output is easy to interpret by humans**

Learning Decision Trees

Method for supervised classification (we will assume attributes with finite discrete values)

- Representation is a **decision tree**.

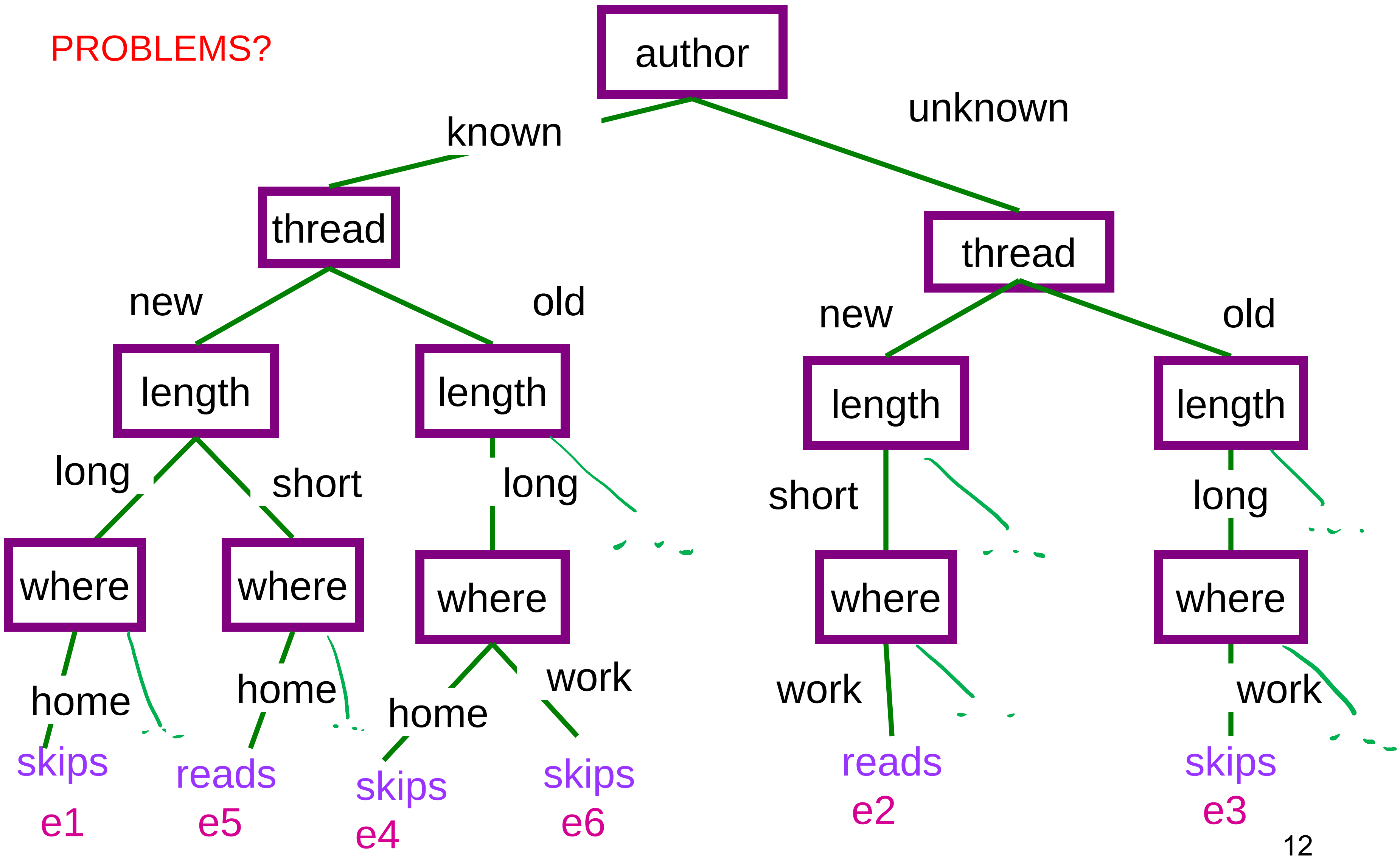
- Bias is towards simple decision trees.

- Search through the space of decision trees, from simple decision trees to more complex ones.

Trivially Correct Decision Tree

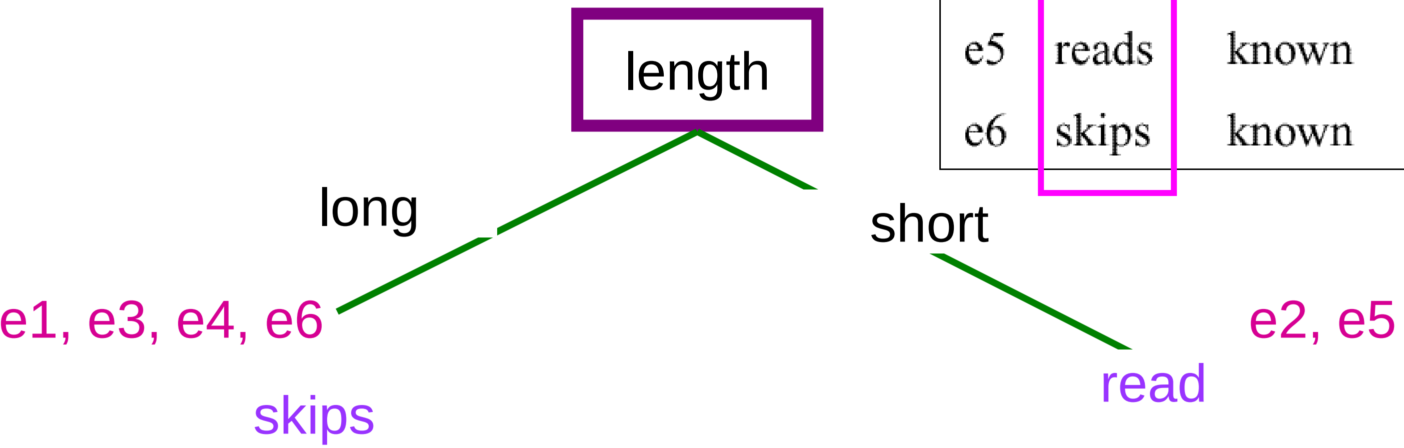
Complete

PROBLEMS?



Example Decision Tree (2)

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

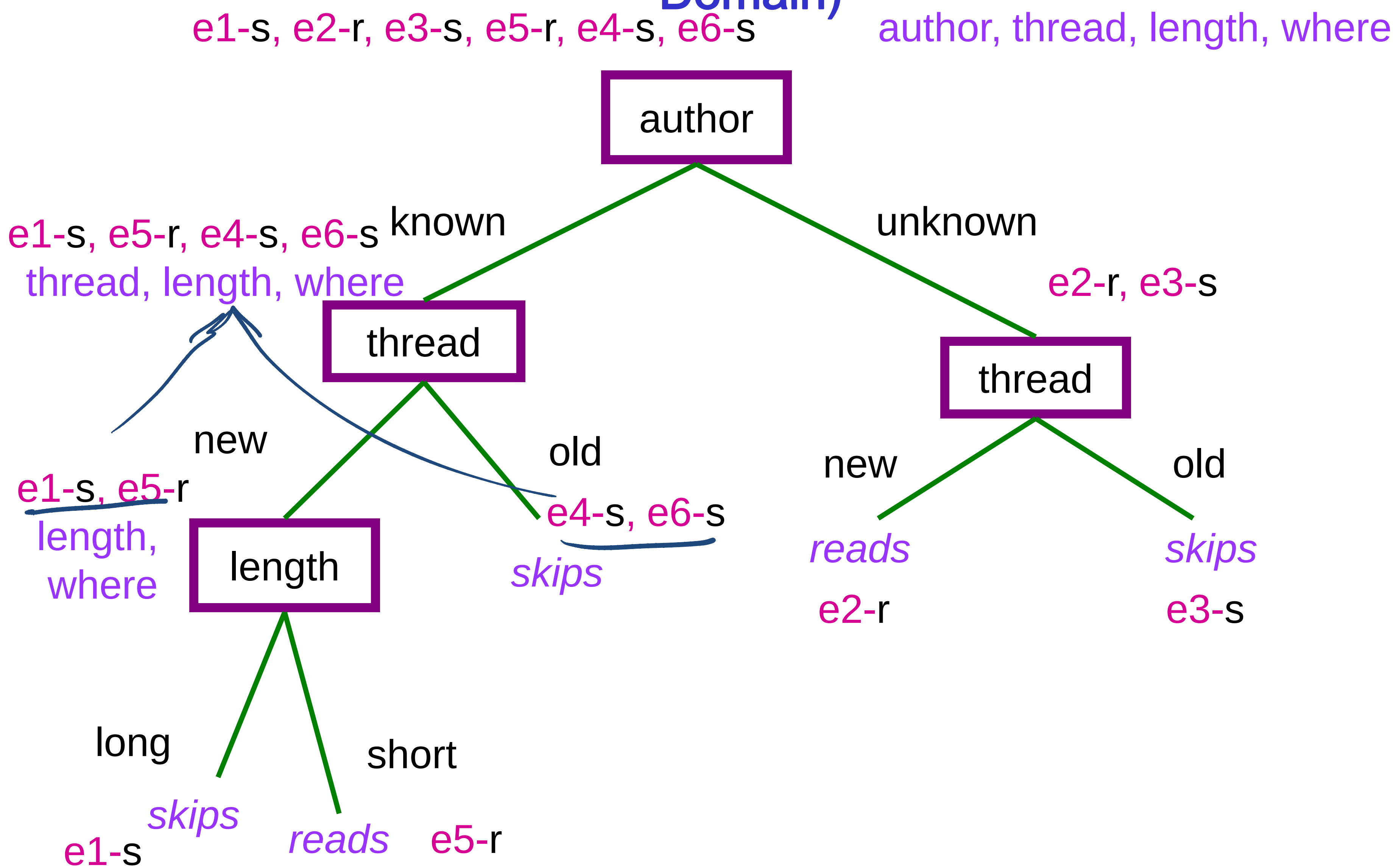


But this tree also classifies my examples correctly.

Searching for a Good Decision Tree

- **The input is**
  - a target attribute for which we want to build a classifier,
  - a set of examples
  - a set of attributes.
- **Stop if all examples have the same classification (good ending).**
  - Plus some other stopping conditions for not so good endings
- **Otherwise, choose an attribute to split on (greedy, or myopic step)**
  - for each value of this attribute, build a sub-tree for those examples with this attribute value

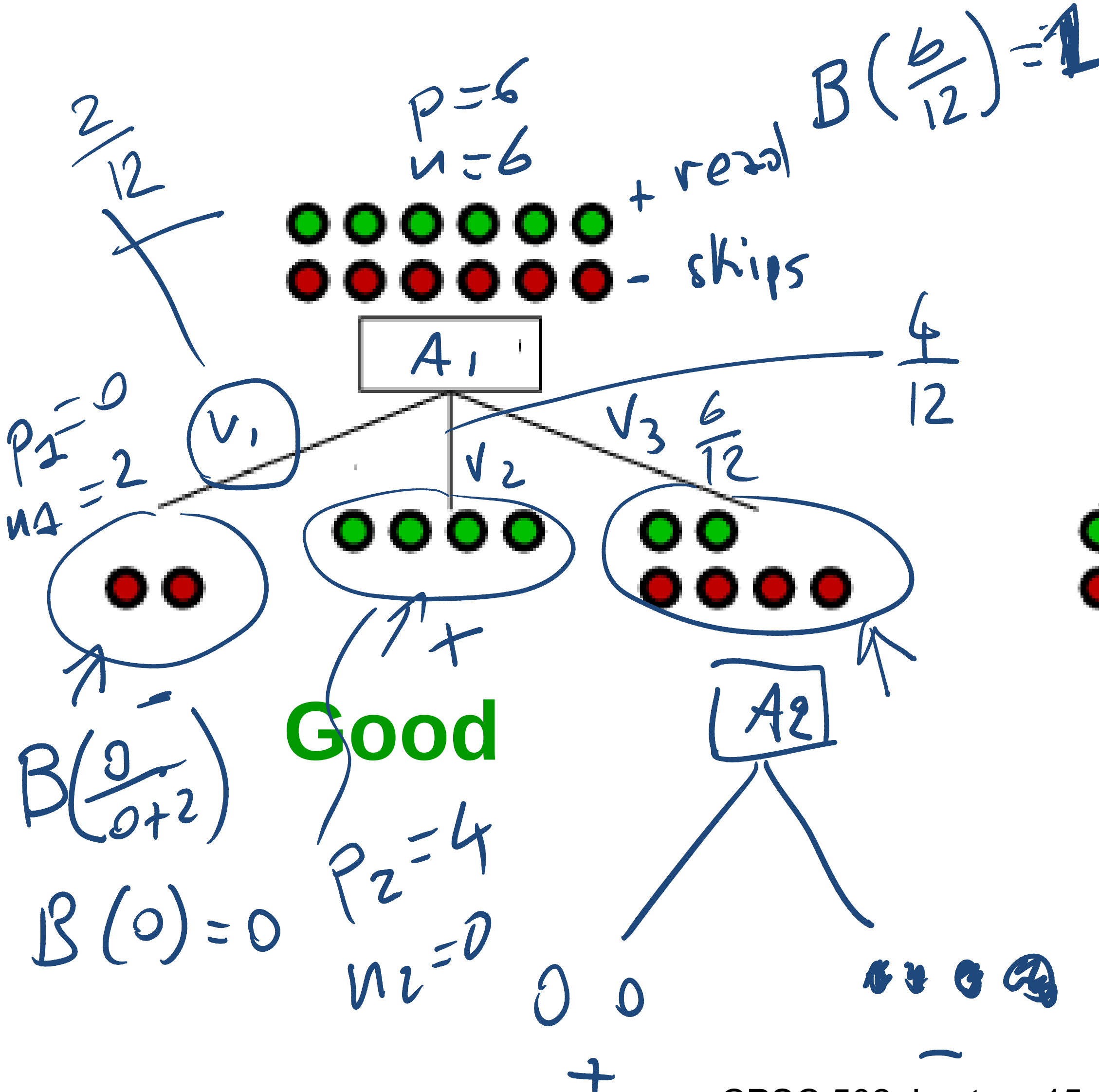
Building a Decision Tree (Newsgroup Read Domain)

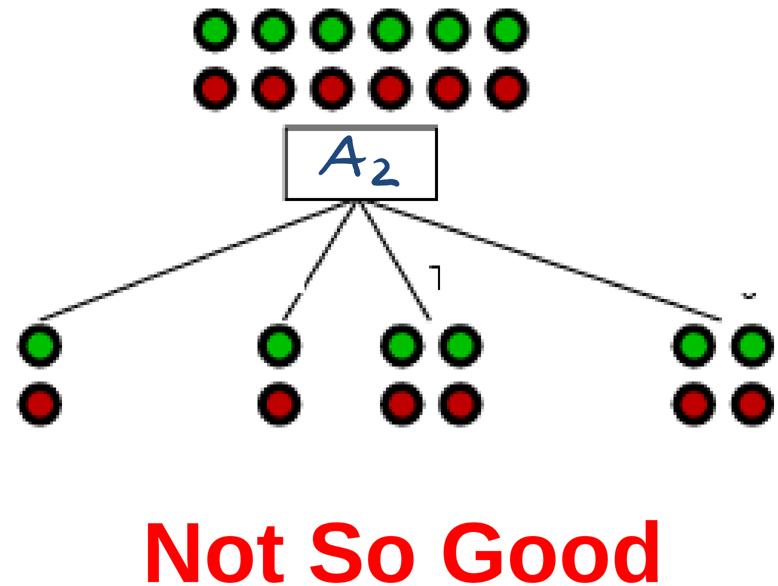


Choosing a good split

- **Goal:** try to minimize the depth of the tree
- Split on attributes that move as much as possible toward an exact classification of the examples
- **Ideal split** divides examples into sets, with the same classification
- **Bad split** leaves about the same proportion of examples in the different classes

Choosing a Good Attribute (binary case)





Information Gain (1)

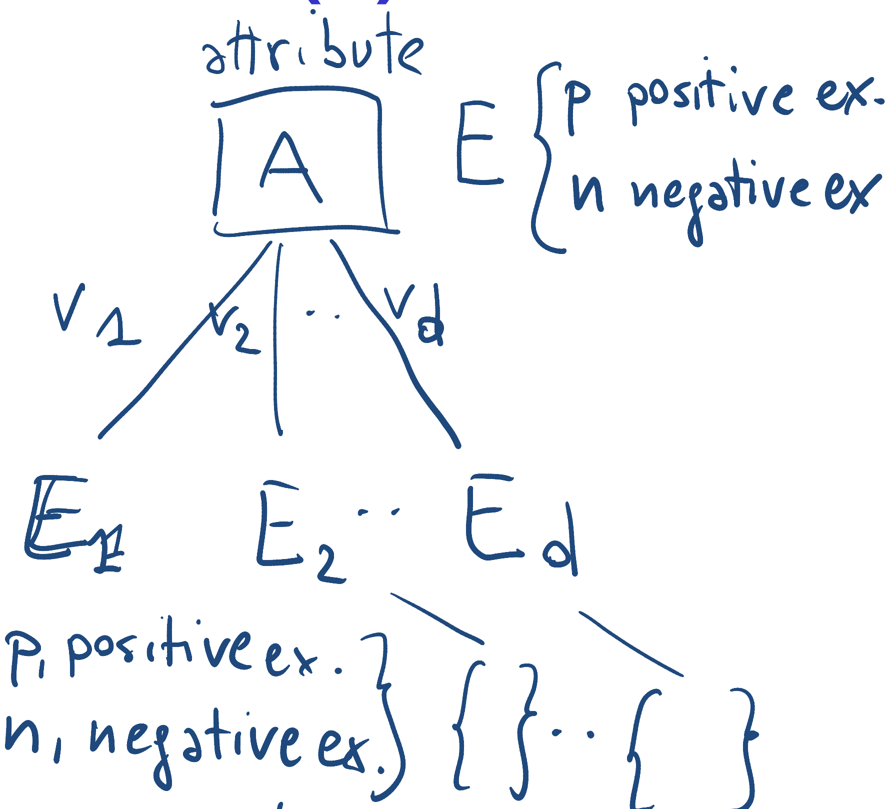
Entropy of $E = B\left(\frac{p}{n+p}\right)$

Entropy of each $E_k = B\left(\frac{p_k}{n_k+p_k}\right)$

Prob. of ex. having value for A equal to $v_k = \frac{p_k+n_k}{p+n}$

E_k

E



$$E = \bigcup_{k=1}^{d} E_k$$

Information Gain (2): Expected Reduction in Entropy

$$Entropy\ \ of\ E = B(\frac{p}{p+n})$$ a

Entropy of split on attribute A

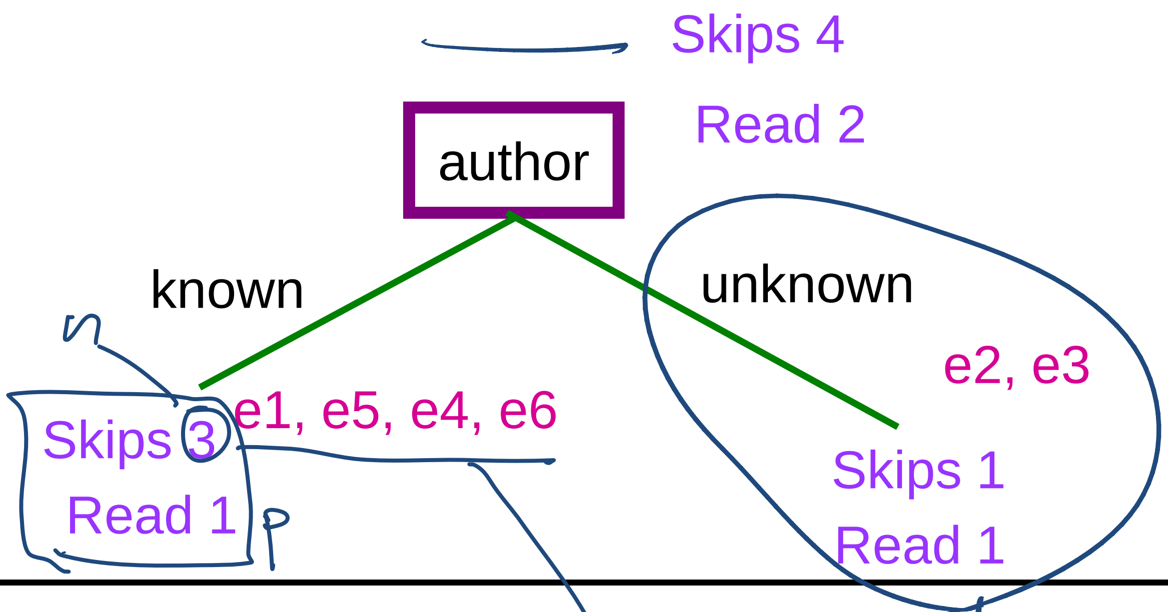
$$remainder(A) = \sum_{k=1}^{d} \boxed{\frac{p_k + n_k}{p+n}} \boxed{B(\frac{p_k}{p_k + n_k})}$$ b

$$Gain(A) = B(\frac{p}{p+n}) - remainder(A)$$

- Chose the attribute with the highest Gain

Example: possible splits

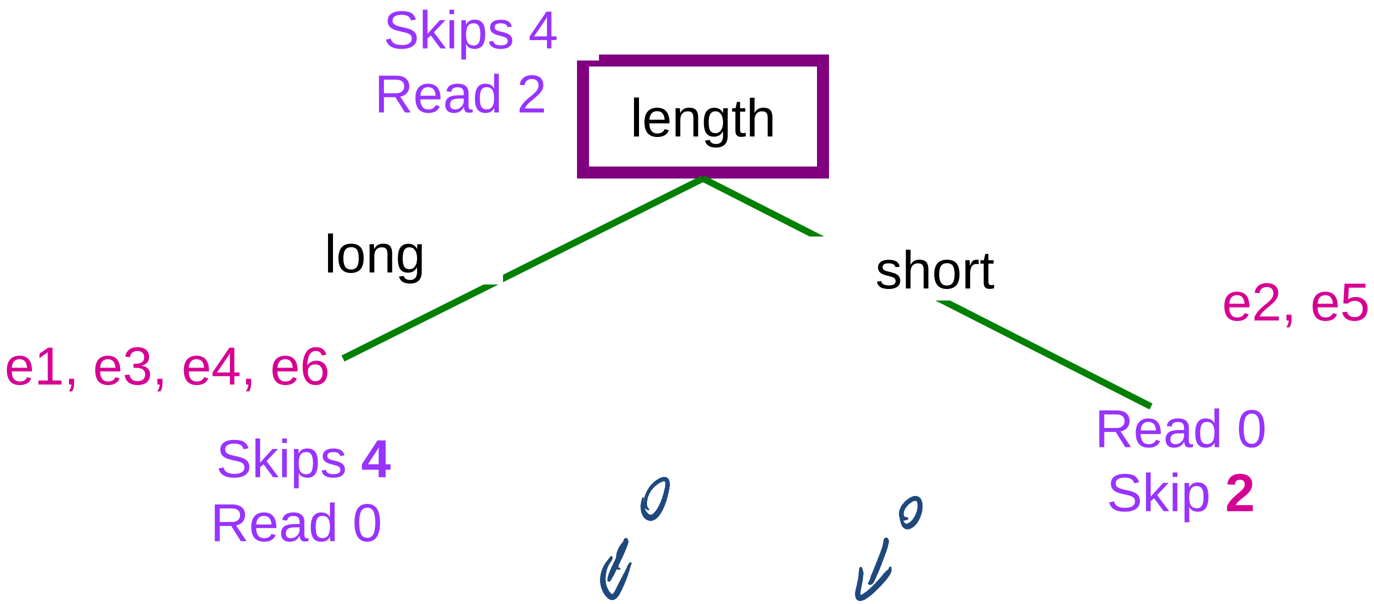
For the initial training set $B(4/6)$ = 0.92 bit



$$Reminder(author) = \boxed{\boxed{4/6} * B(1/4)} + \boxed{2/6 * B(1/2)}$$
$$= 2/3*0.811+1/3 =$$
$$Gain(author) = 0.92 - 0.875 = 0.054$$

Example: possible splits

For the initial training set I(4/6, 2/6) = 0.92 bit



*Reminder (length) = 4/6 * B(1) + 2/6 * B(1)=0*

Gain(length)=0.92

Drawback of Information Gain

Tends to favor attributes with many different values

- Can fit the data better than spitting on attributes with fewer values

Imagine extreme case of using "*message id-number*" in the newsgroup reading example

- Every example may have a different value on this attribute
- Splitting on it would give highest information gain, even if it is unlikely that this attribute is relevant for the user's reading decision

Alternative measures (e.g. *gain ratio*)

Expressiveness of Decision Trees

They can represent any discrete function, an consequently any Boolean function

2^{2^n}

How many ?

n boolean vars

2^n configurations

x_1	...	x_n	$f(x_1 \dots x_n)$
1 1 1 1 1 1 1 1		1	1 0 - -
1 1 1 1 1 1 1		0	1 1 - -
⋮			1 1 - -
			1 1 - -
			1 1
			1 1

2^n

Handling Overfitting

- This occurs with noise and correlations in the available examplesthat are not reflected in the data as a whole.

- One technique to handle overfitting: *decision tree pruning*
  - Statistical techniques to evaluate when the **gain** on the attribute selected by the splitting technique is *large enough* to be relevant

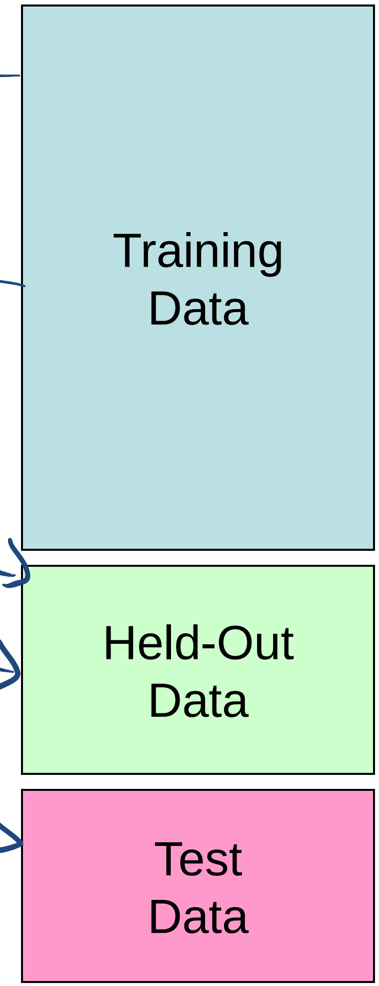
- Generic techniques to test ML algorithms

How to Learn

- Data: labeled instances, e.g. emails marked spam/ham
  - Training set
  - Held out (validation) set
  - Test set
- Experimentation cycle
  - Learn model on training set
  - Tune it on held-out set
  - Compute accuracy on test set
  - Very important: never "peek" at the test set!
- Evaluation
  - Accuracy: fraction of instances predicted correctly
- Overfitting and generalization
  - Want a classifier which does well on *test* data
  - Overfitting: fitting the training data very closely, but not generalizing well to test data

split$_1$ DT_1

split$_2$ DT_2



Cross-Validation

- Partition the training set into k sets
- Run the algorithm *k* times, each time (*fold*) using one of the *k* sets as the test test, and the rest as training set
- Report algorithm performance as the average performance (e.g. accuracy) over the k different folds

➢ **Useful to select different candidate algorithms/models**

- E.g. a DT built using information gain vs. some other measure for splitting
- Once the algorithm/model type is selected via cross-validation, return the model trained on *all available data*

Other Issues in DT Learning

- Attributes with continuous and integer values (e.g. *Cost* in $)
  - Important because many real world applications deal with continuous values
  - Methods for finding the *split point* that gives the highest information gain (e.g. Cost > 50$)
  - Still the most expensive part of using DT in real-world applications
- **Continue-valued output attribute (e.g. predicted cost in $): *Regression Tree***
  - Splitting may stop before classifying all examples

STOP HERE

  - Leaves with unclassified examples use a linear function of a subset of the attributes to classify them via linear regression
  - Tricky part: decide when to stop splitting and start linear regression

TODO for this Thurs

- Read 7.5, 7.6 and 11.1, 11.2
- Assignment 3-Part1 due