Artificial Intelligence Foundations and Applications

Machine Learning – Part 3

Decision Tree

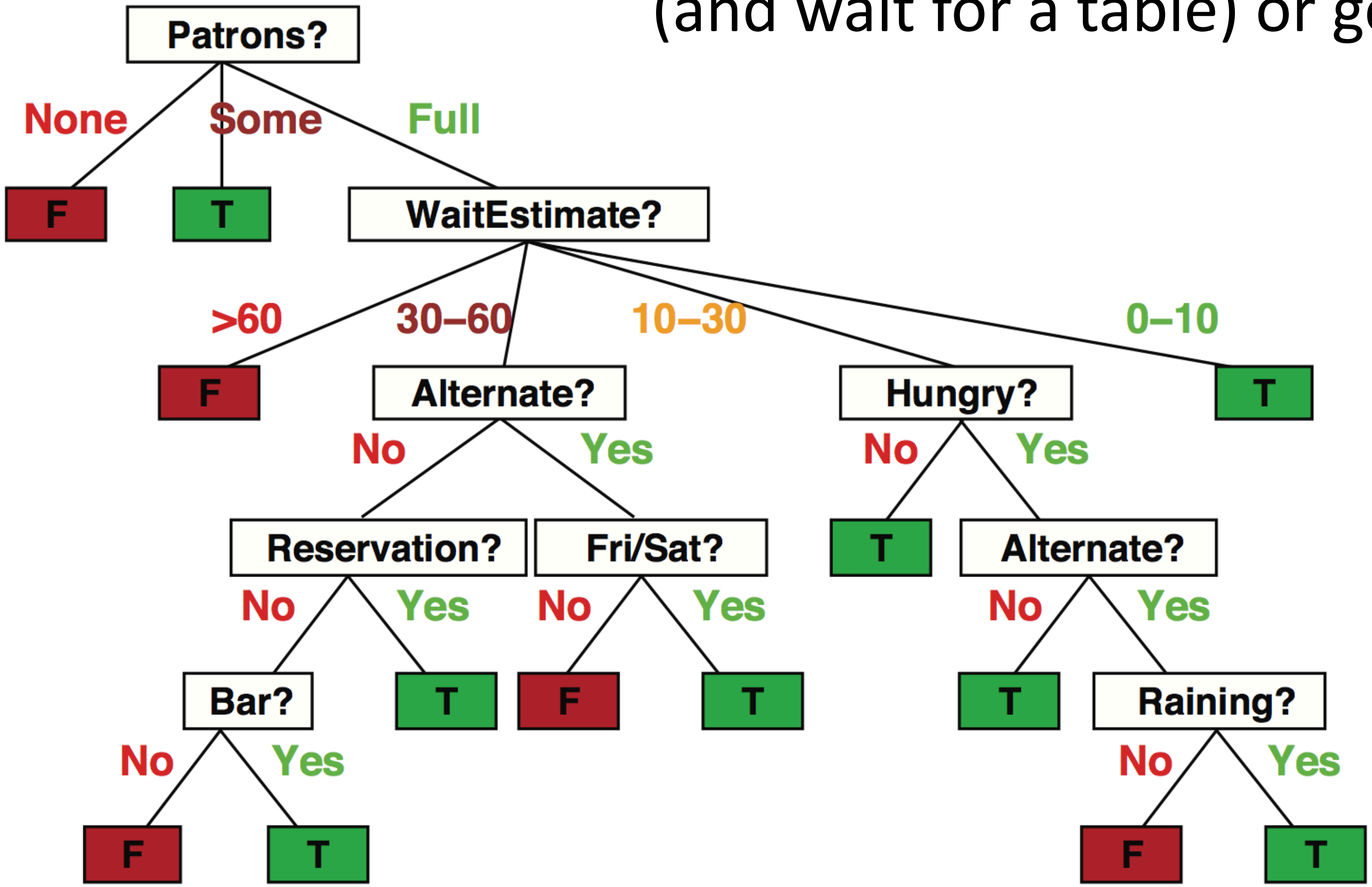
Sudeshna Sarkar

7 Nov 2022

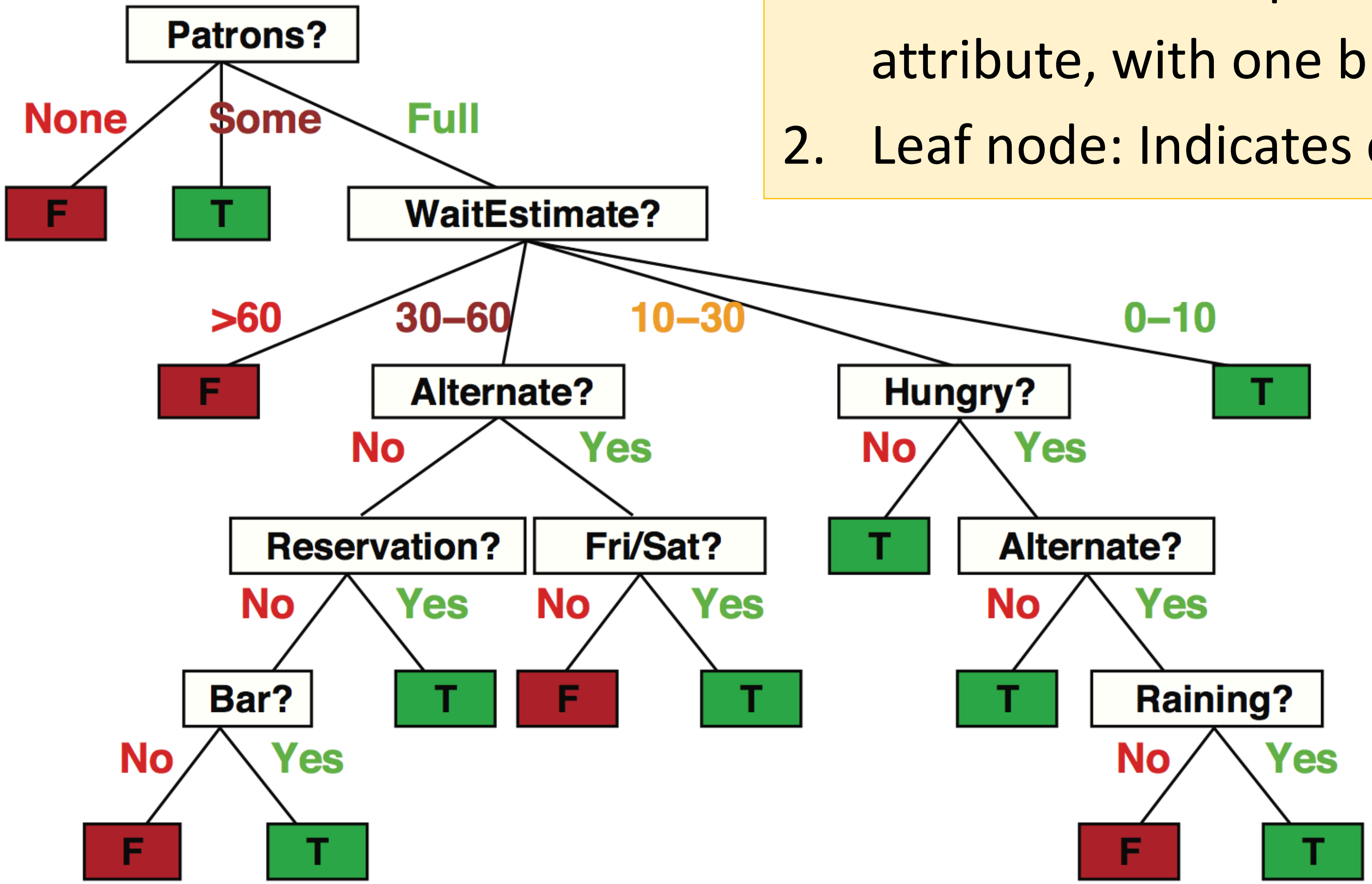
Centre of Excellence in Artificial Intelligence, IIT Kharagpur

Decision trees

I’ve just arrived at a restaurant: should I stay (and wait for a table) or go elsewhere?



Slide credit: ai.berkeley.edu



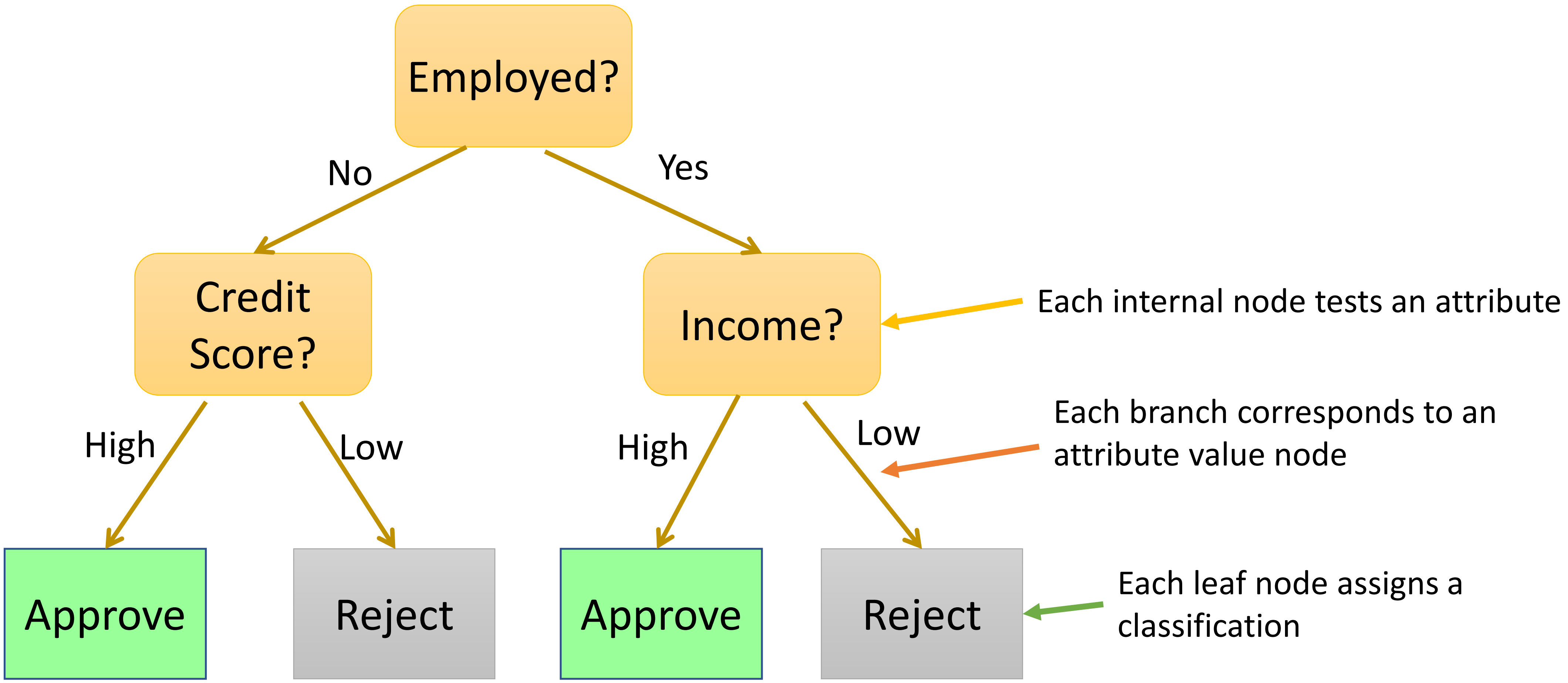
A decision tree is a classifier in the form of a tree structure with two types of nodes:

1. Decision node: Specifies a choice or test of some attribute, with one branch for each outcome
2. Leaf node: Indicates classification of an example

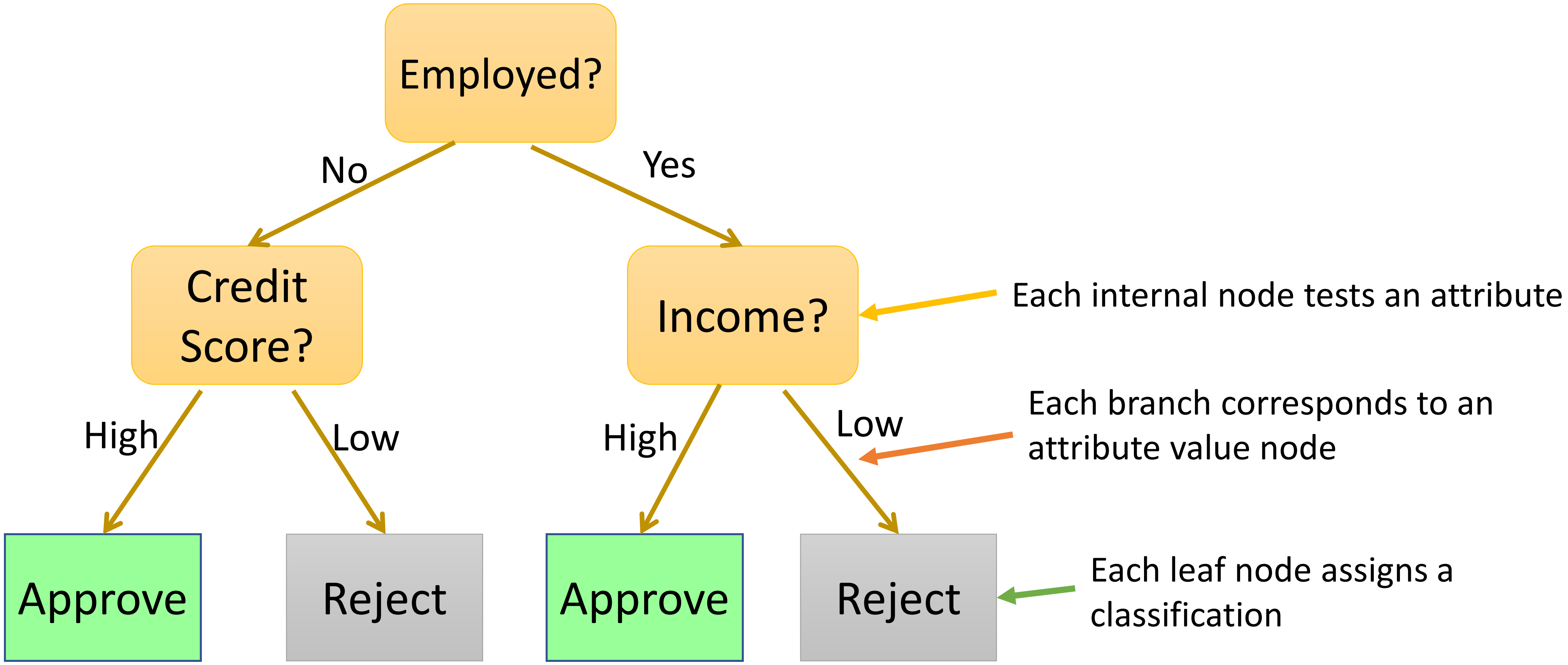
Decision tree ***partitions*** the input space, assigns a label to each partition

Slide credit: ai.berkeley.edu

Decision Tree for Whether to approve a loan



A decision trees represent disjunctions of conjunctions



$$(\text{Employed?=No}) \wedge (\text{Credit Score=High})$$
$$\vee \; (\text{Employed?=Yes}) \wedge (\text{Income=High})$$

Issues

- Given some training examples, what decision tree should be generated?
- One proposal: prefer the smallest tree that is consistent with the data (Bias)

- Possible method:
  - search the space of decision trees for the smallest decision tree that fits the data

Finding a minimal decision tree consistent with a set of data is NP-hard.

Example Data

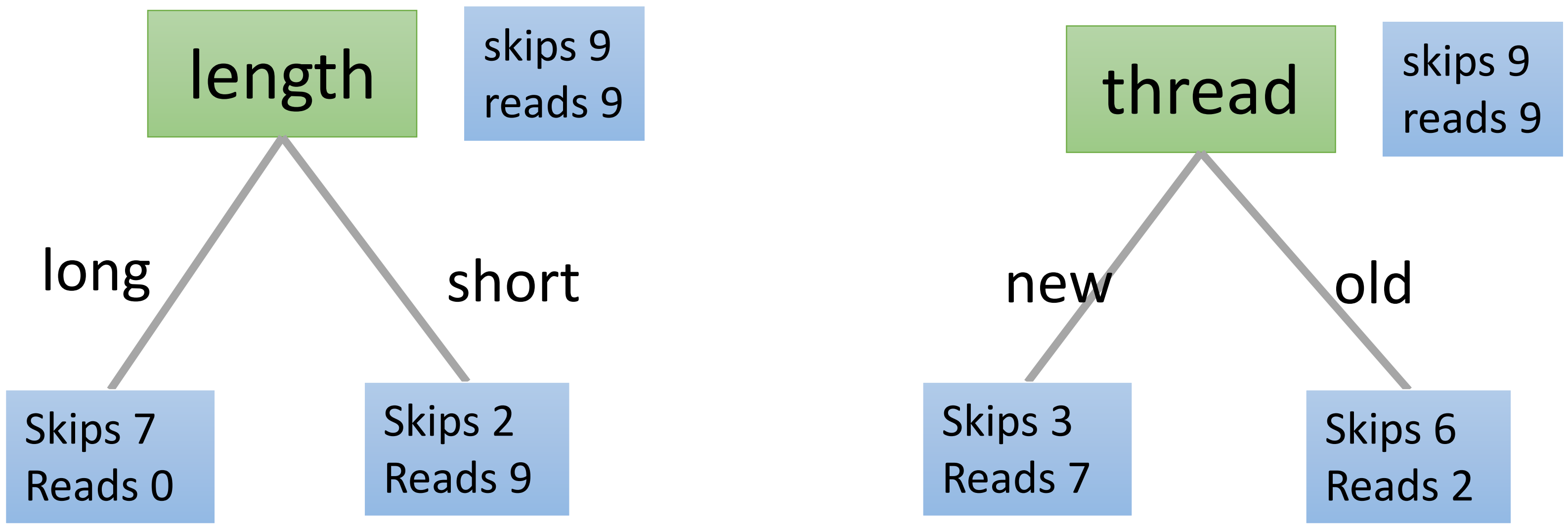
Training Examples:

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

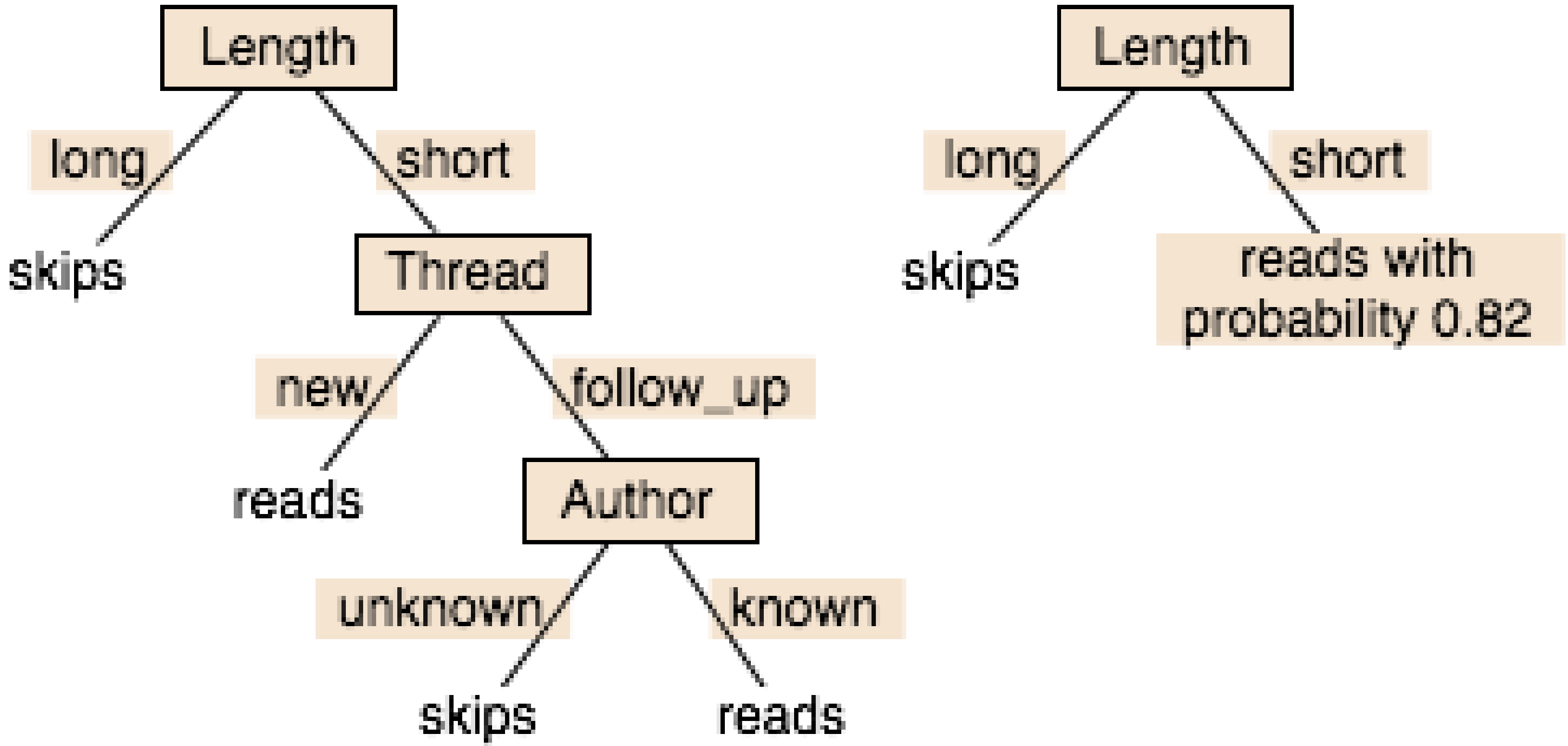
New Examples:

e7	known	new	short	work	???
e8	unknown	new	short	work	???

Which is a better split?



Which DT do you prefer?



Building a Decision Tree

```
Function BuildTree(D,A)
  # D: dataset at current node, A: current set of attributes
  If empty(A) or all labels in D are the same
      # Leaf node
      class = most common class in D
  else
       # Internal node
       a ⇐ bestAttribute(D,A)
       LeftNode = BuildTree(D(a=1), A \ {a})
       RightNode = BuildTree(D(a=0), A \ {a})
  end
end
```

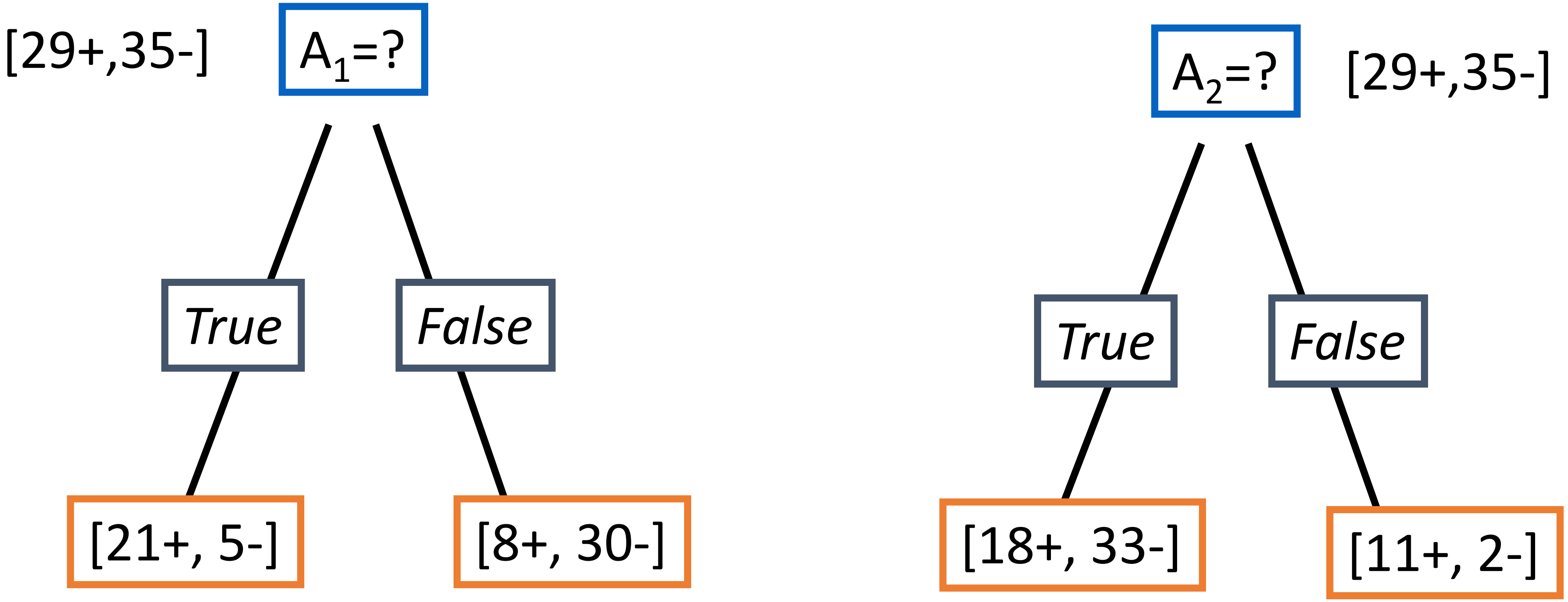
```
Function BuildTree(D,A)
   # D: dataset at current node, A: current set of attributes
   If empty(A) or all labels in D are the same
      # Leaf node
      class = most common class in D
   else
      # Internal node
      a ⇐ bestAttribute(D,A)
      LeftNode = BuildTree(D(a=1), A \ {a})
      RightNode = BuildTree(D(a=0), A \ {a})
   end
end
```

Choices

1. When to stop
   - no more input features
   - all examples are classified the same
   - too few examples to make an informative split

2. Which test to split on
   - split gives smallest error.

Which Attribute is "best"?



Information gain

measures how well a given attribute separates the training examples according to their target classification

Gain is a measure of how much we can reduce uncertainty

Entropy

A measure for
- uncertainty
- purity
- information content

S is a sample of training examples

p_+ is the proportion of positive examples in S

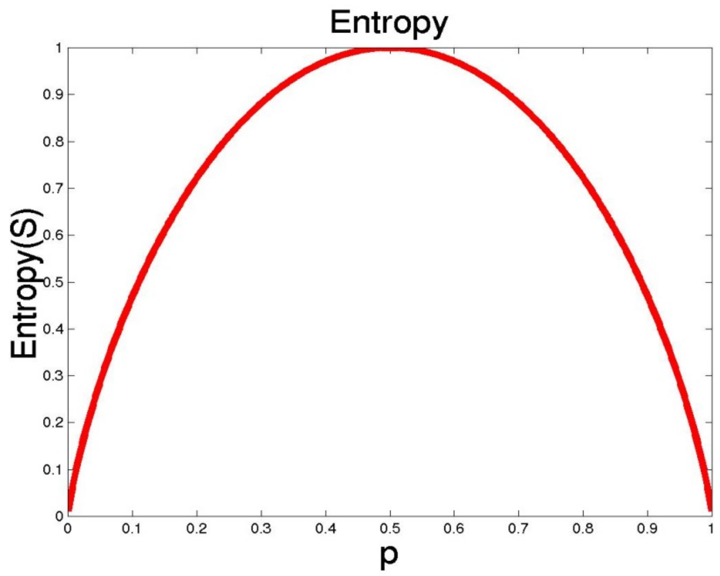
p_- is the proportion of negative examples in S

- Entropy of S: average optimal number of bits to encode information about certainty/uncertainty about S

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

In general, when p_i is the fraction of examples labeled i:

$$Entropy(S[p_1, p_2, \ldots, pk]) = -\sum_1^k p_i \log(p_i)$$



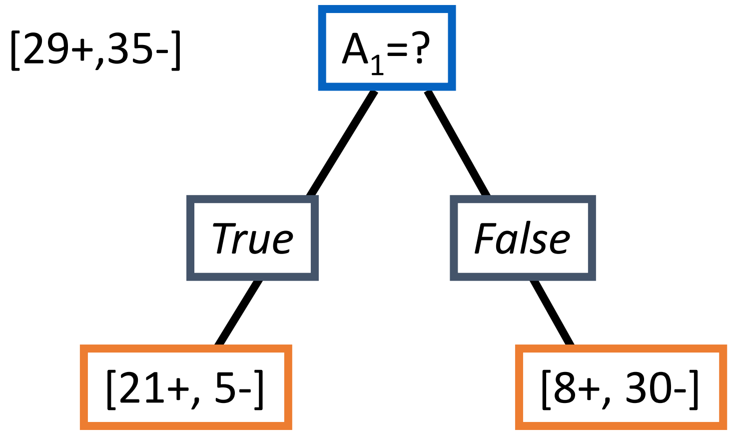
Information Gain

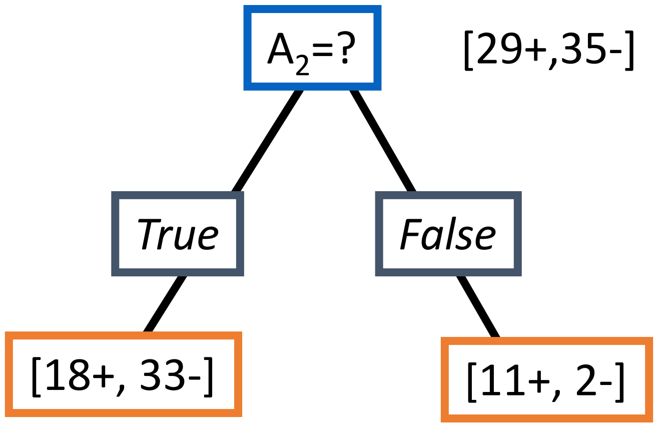
Gain(S,A): expected reduction in entropy due to sorting S on attribute A

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in \text{values}(A)} |S_v|/|S| \ \text{Entropy}(S_v)$$

S_v is the subset of S for which attribute A has value v, and

$$\text{Entropy}([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$$





Information Gain Computation

$\text{Entropy}([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$

$\text{Entropy}([21+,5-]) = 0.71$

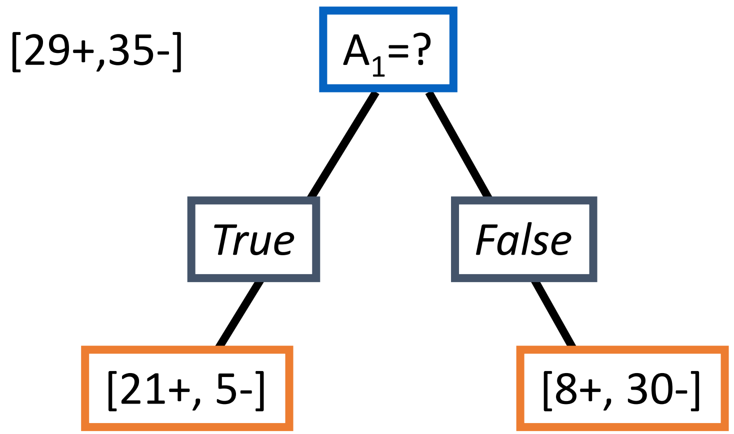
$\text{Entropy}([8+,30-]) = 0.74$

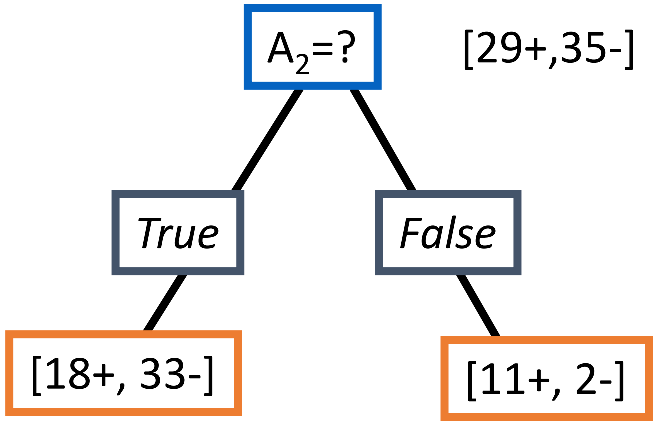
$\text{Gain}(S,A_1)=\text{Entropy}(S)$
$\quad -26/64*\text{Entropy}([21+,5-])$
$\quad -38/64*\text{Entropy}([8+,30-])$
$\quad =0.27$

$\text{Entropy}([18+,33-]) = 0.94$

$\text{Entropy}([8+,30-]) = 0.62$

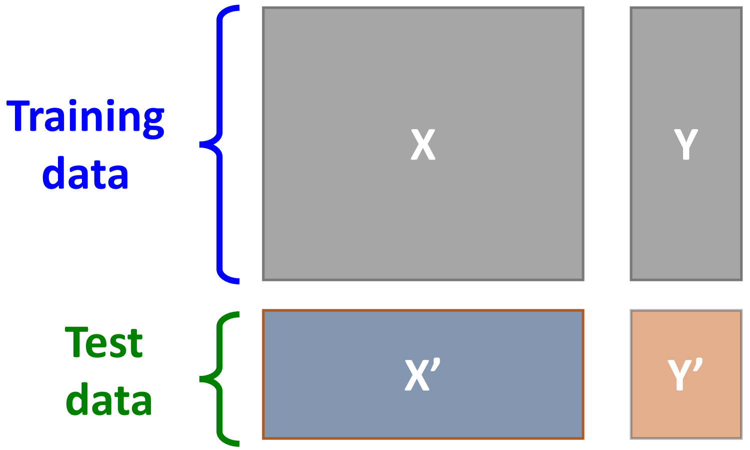
$\text{Gain}(S,A_2)=\text{Entropy}(S)$
$\quad -51/64*\text{Entropy}([18+,33-])$
$\quad -13/64*\text{Entropy}([11+,2-])$
$\quad =0.12$



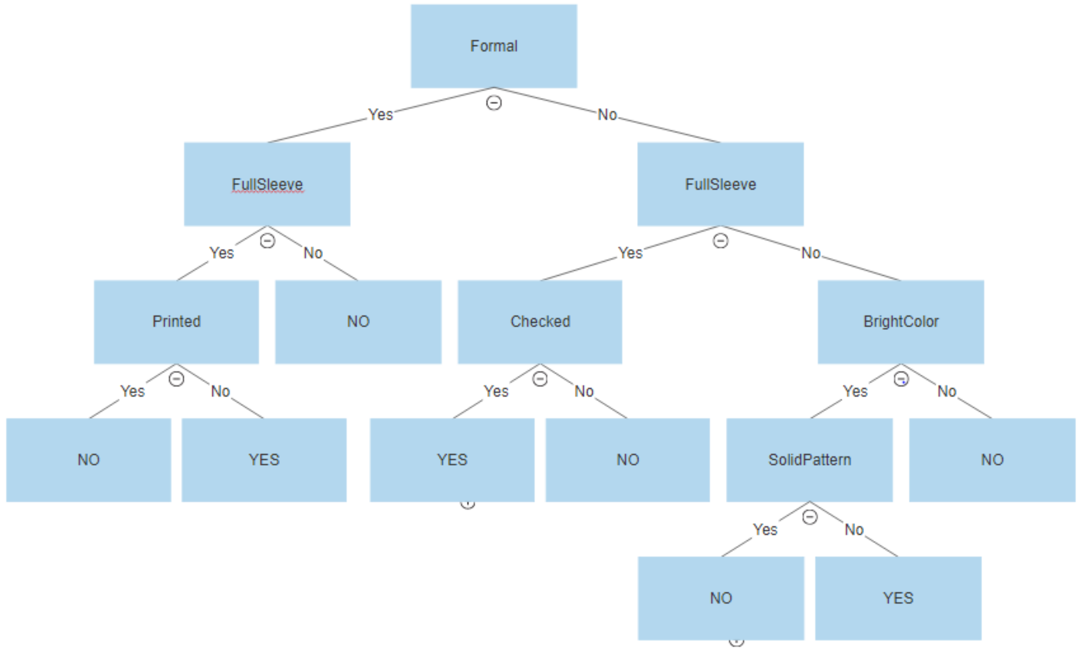


Validation

- Divide your data randomly into training and *test* data.
- Build your best model based on the training data only.
- Apply your model to the test data.
- Does your model predict y’ for the test data as well as it predicted y for the training data?

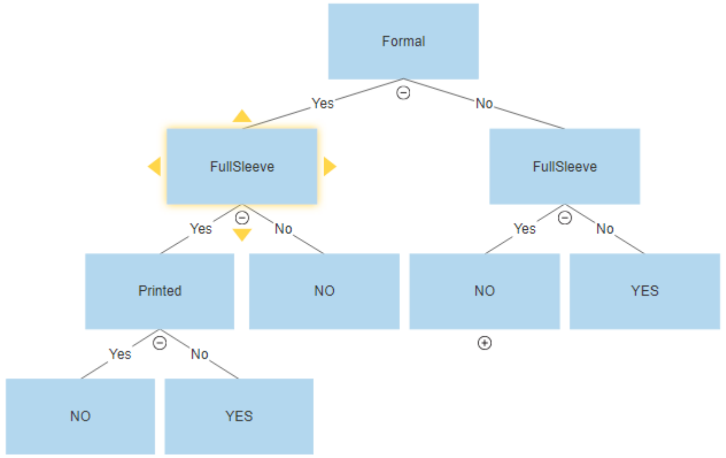


Which Decision Tree?



Training Error = 0.05

Test Error = 0.2



Training Error = 0.1

Test Error = 0.15

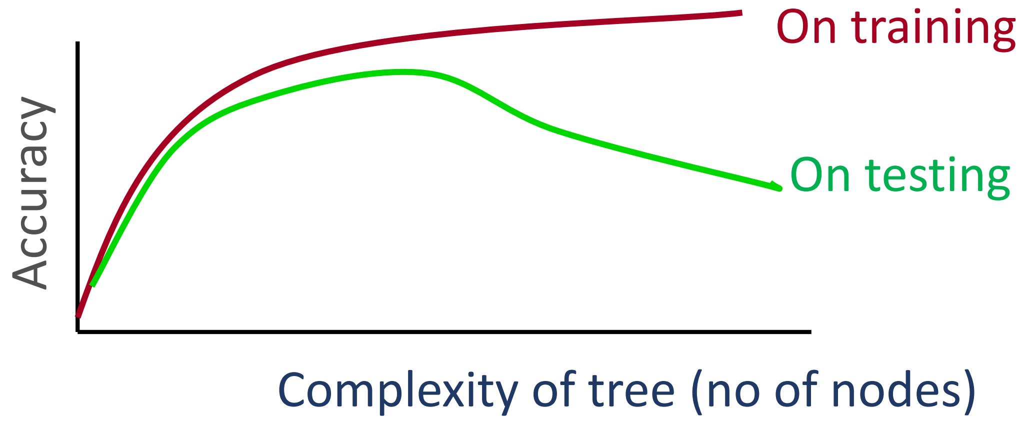
Overfitting

Overfitting :

- Fit the training data too well
- But fail to generalize to new examples

Causes

- Noise
- Irrelevant Features
- Insufficient Data



Overfitting results in decision trees that are more complex than necessary

Overfitting

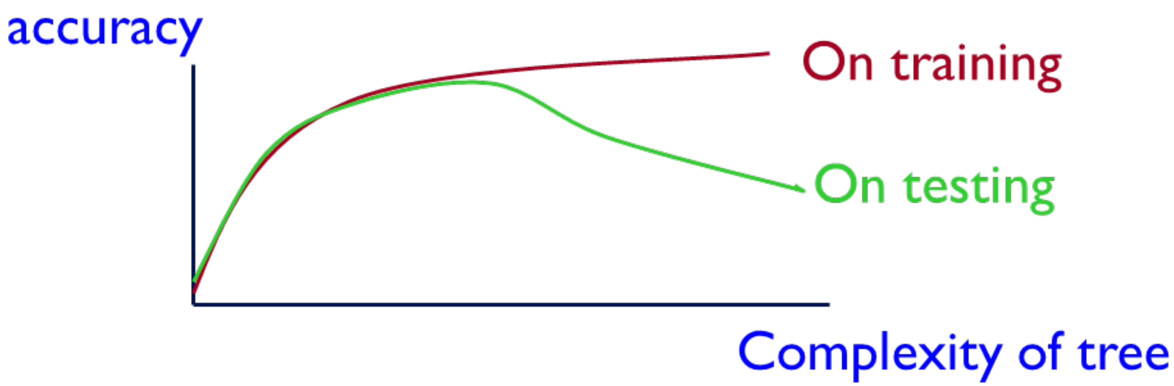
A hypothesis h is said to overfit the training data if there is another hypothesis h' such that h has smaller error than h' on the training data but h has larger error on the test data than h' .

How to create a classification decision tree

- Greedy Splitting : Grow the tree
- Stopping Criterion: when the number of samples in a leaf is small enough.
- Pruning The Tree: remove unnecessary leaves to
  - make it more efficient and
  - solve overfitting problems.

Overfitting in Decision Trees

- Your model shows much greater loss on the test data than on the training data.
- Example: a decision tree with so many levels that the typical leaf is reached by only one member of the training set.

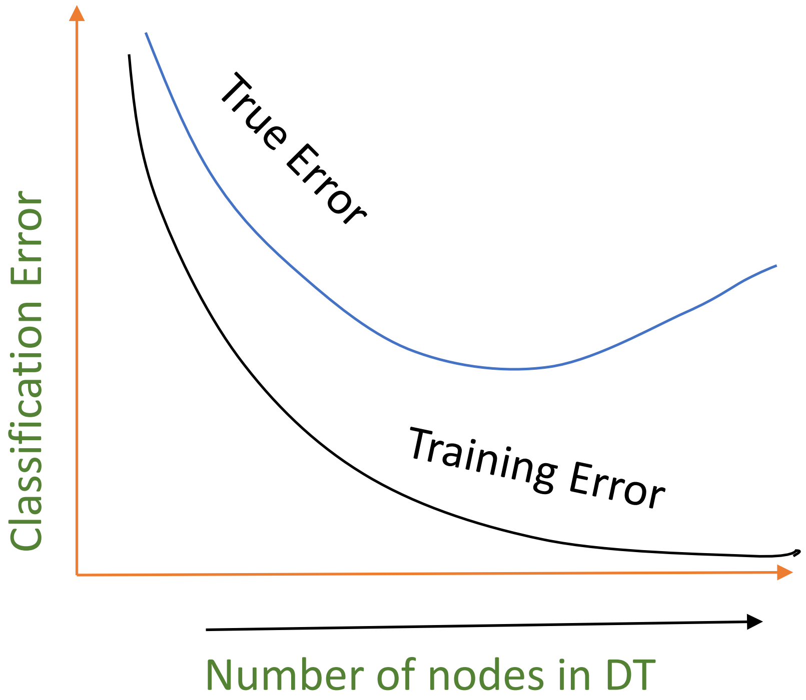


Avoid Overfitting

- How can we avoid overfitting a decision tree?
  - Prepruning: Stop growing when data split not statistically significant
  - Postpruning: Grow full tree then remove nodes

Pre-Pruning (Early Stopping)

- Early Stopping: Stop the learning algorithm before tree becomes too complex



Stopping conditions:

- Do not split a node which contains too few instances
- Stop if expanding the current node does not improve impurity measures significantly (e.g., Gini or information gain)
- Limit tree depth

Reduced-error Pruning

Partition data into train set and validation set

- Build a tree using the train set.
- Until accuracy on validation set decreases, do:

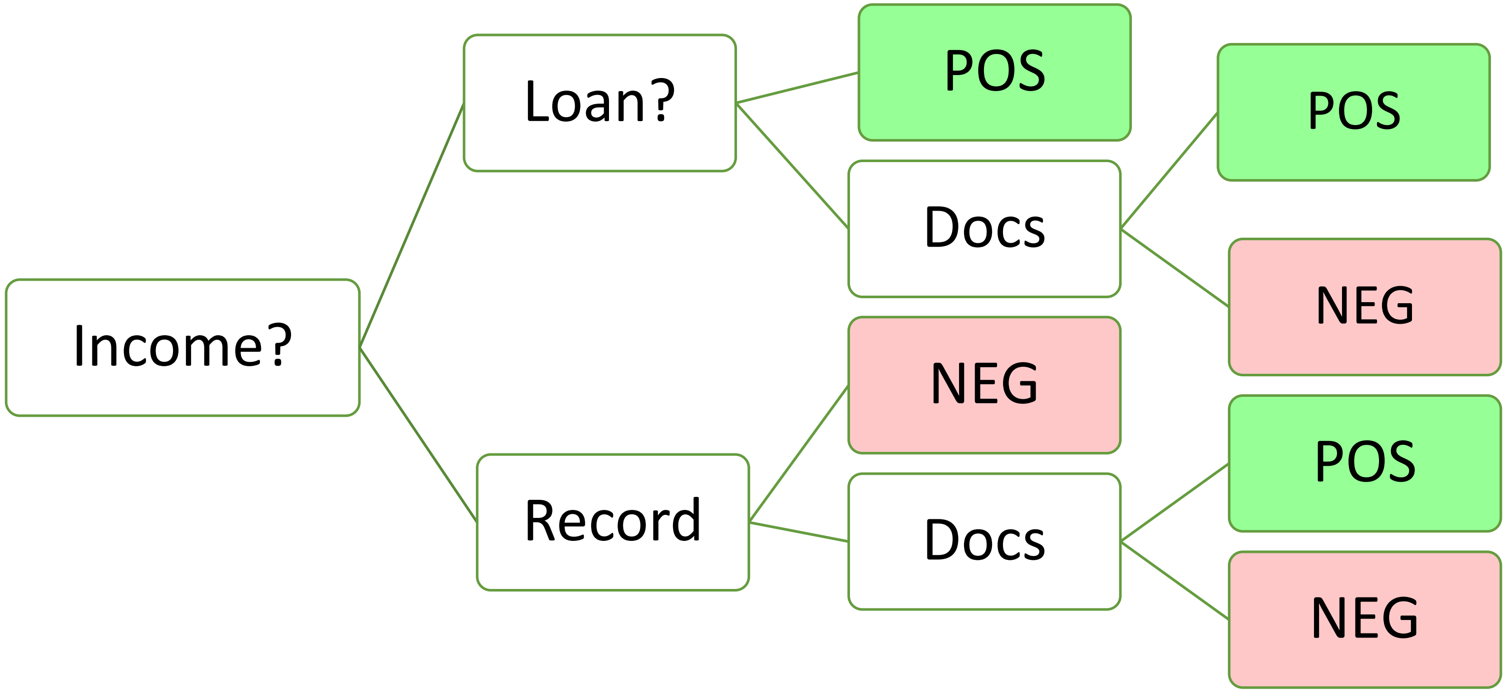
  For each non-leaf node in the tree

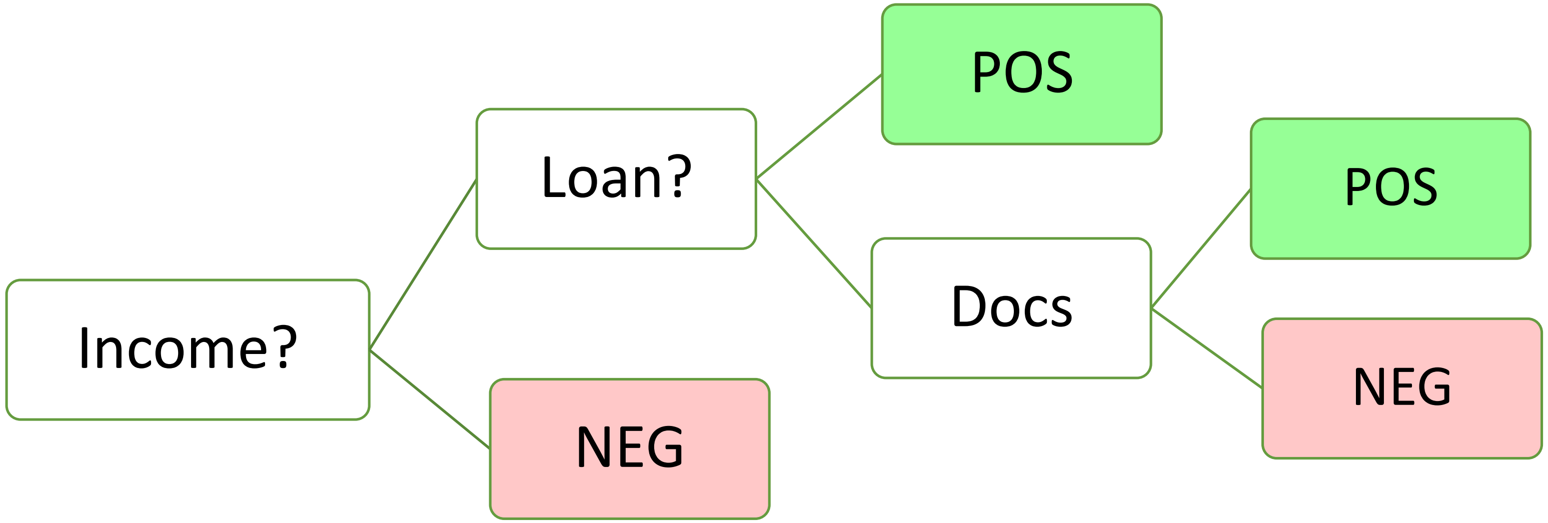
    Temporarily prune the tree below; replace it by majority vote

    Test the accuracy of the hypothesis on the validation set

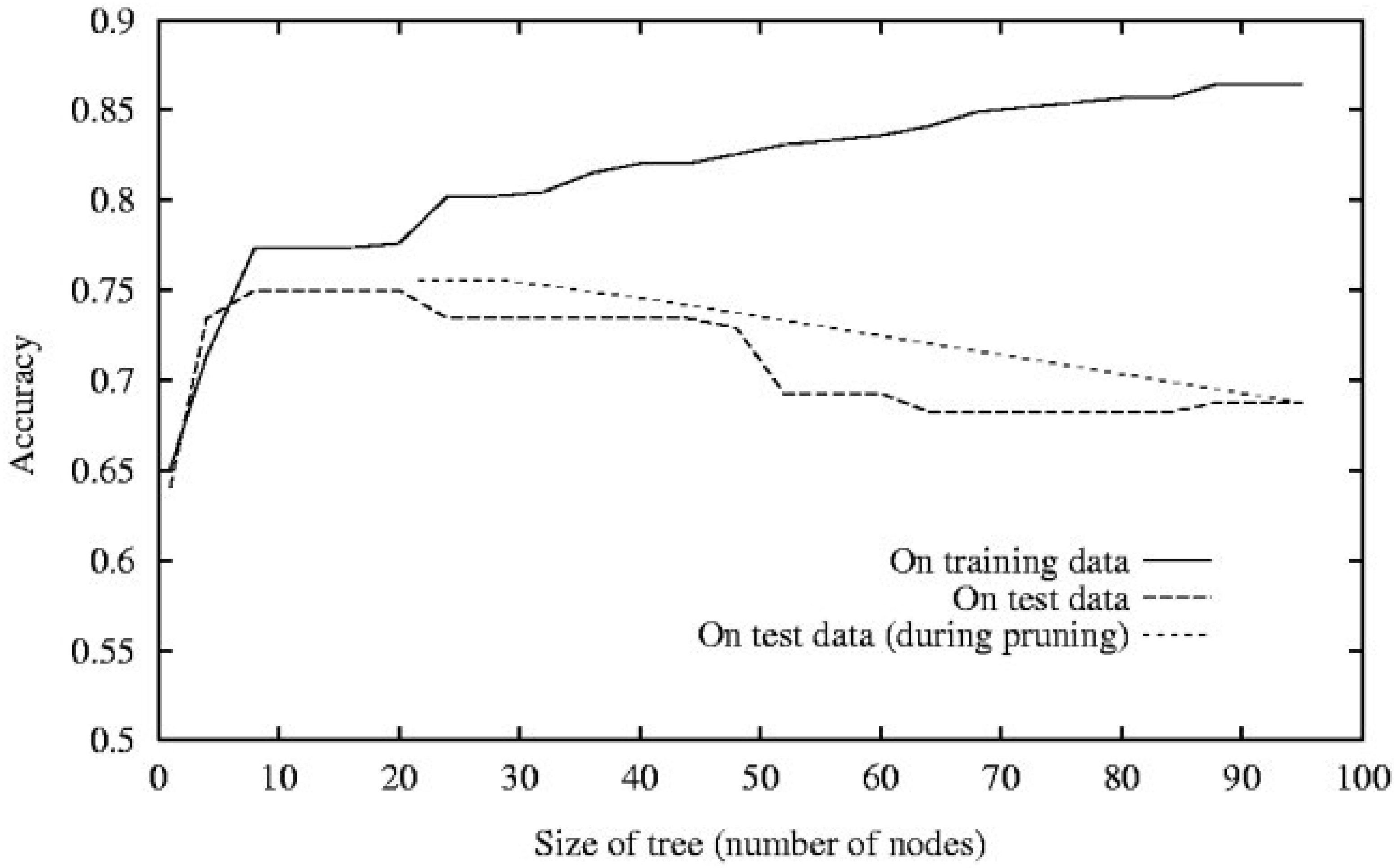
    Permanently prune the node with the greatest increase in accuracy on the validation test.

Tree Pruning

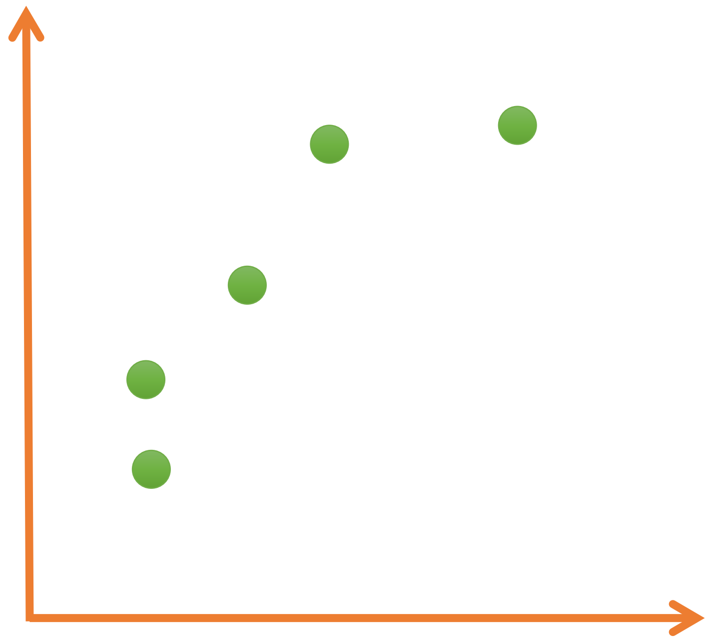




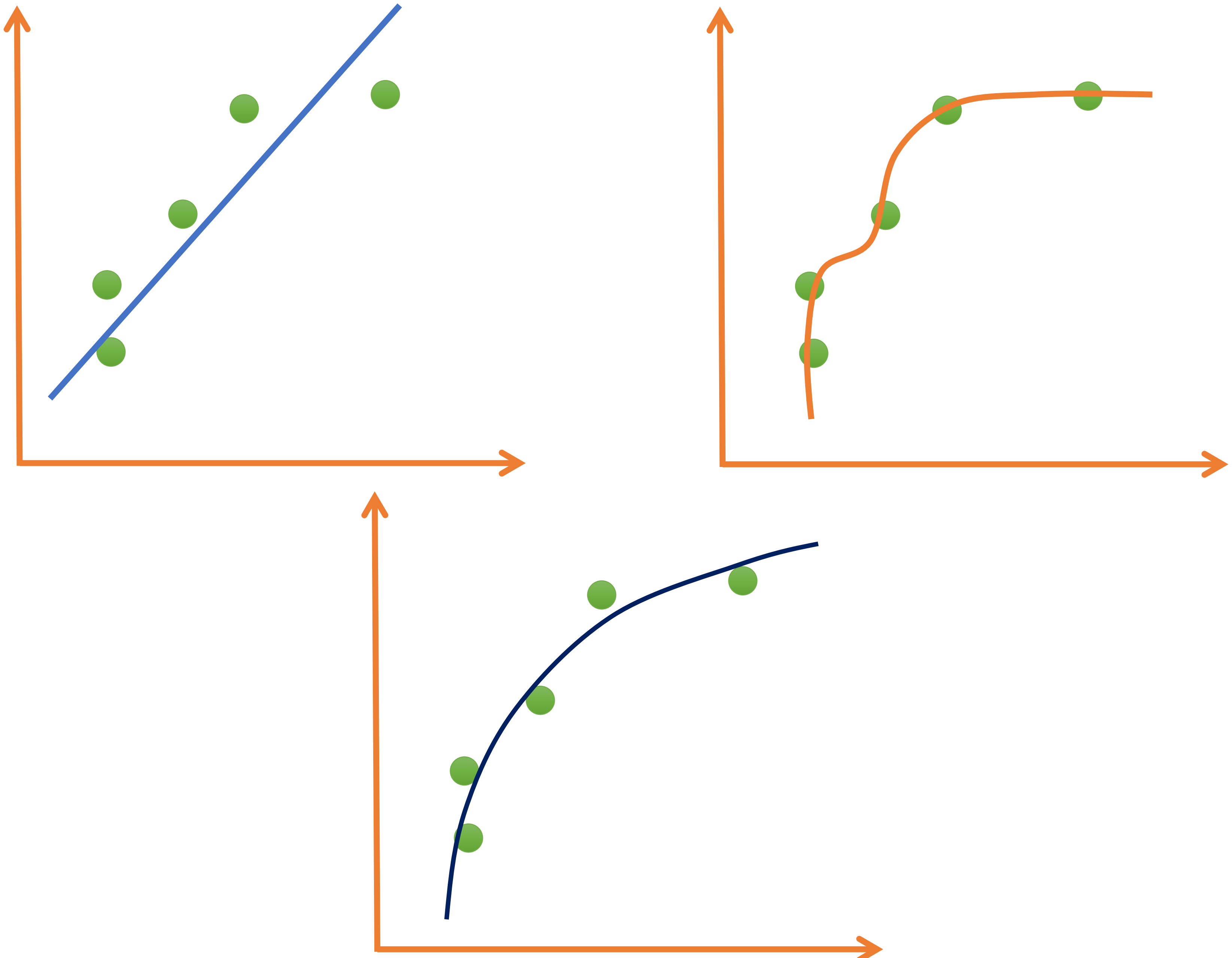
Reduced Error Pruning



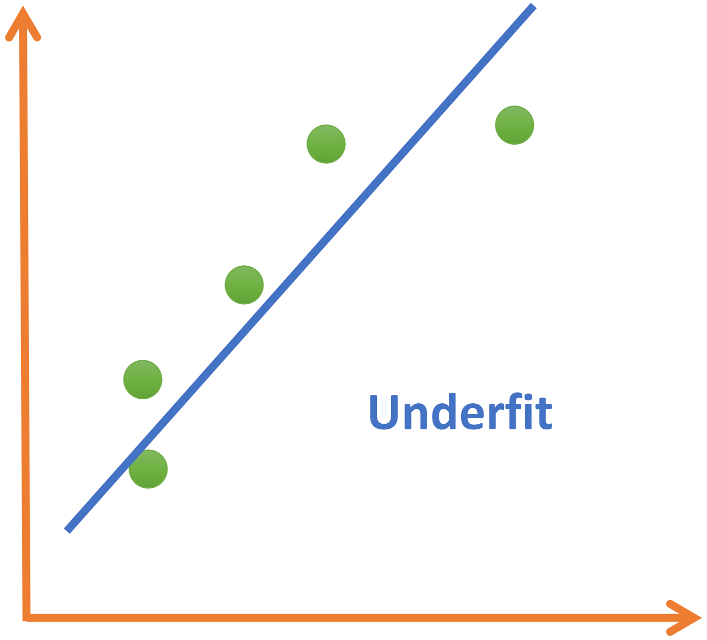
Regression

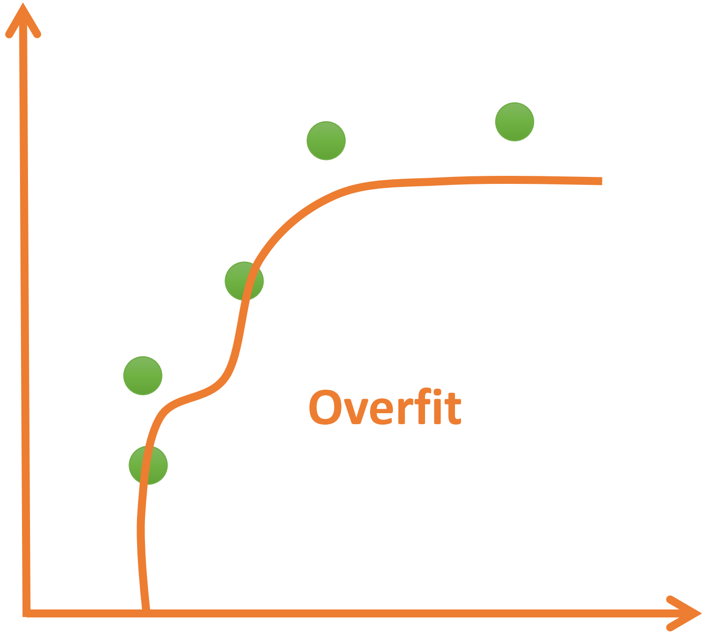


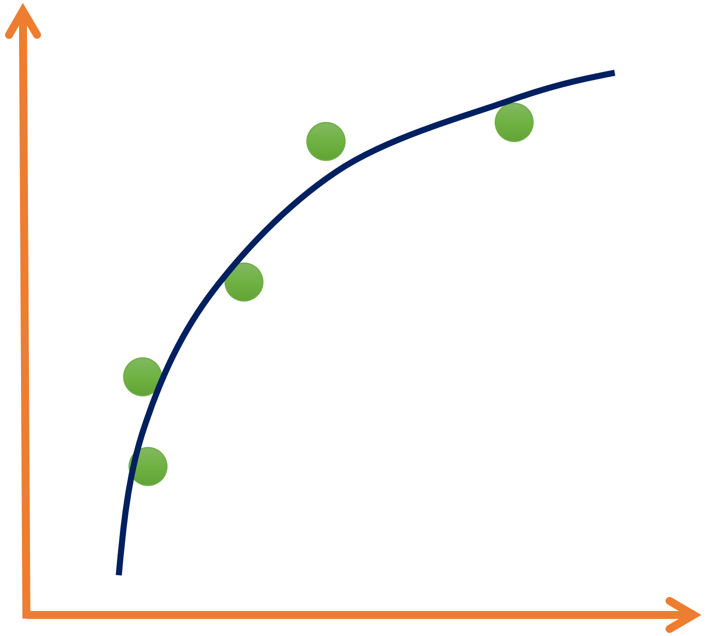
Regression



Regression







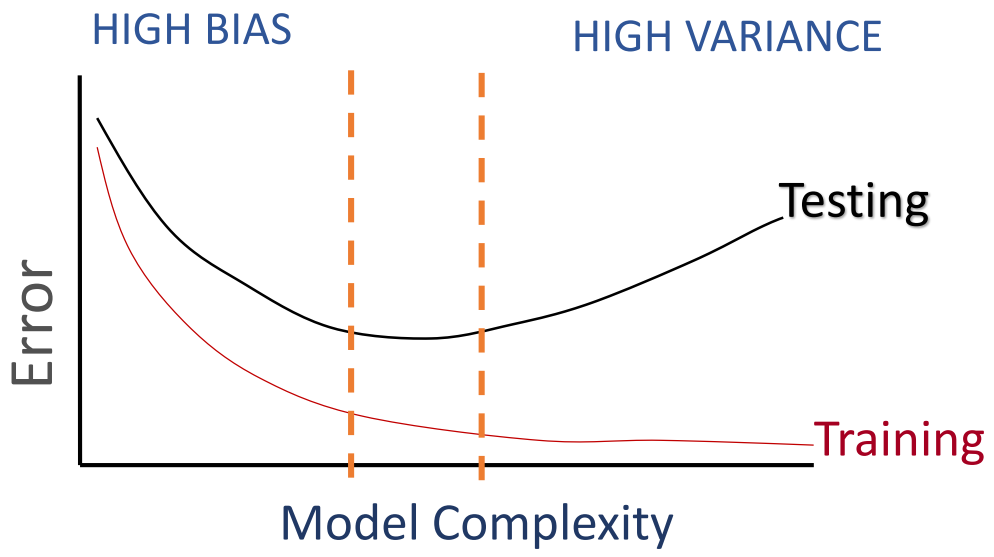
Overfitting vs Underfitting

Underfitting

- Not able to capture the concept
  - Features don’t capture concept
  - Model is not powerful.

Overfitting

- Fitting the data too well



BIAS

- Error caused because the model can not represent the concept
- Bias is the expected difference between the model prediction and the true y's.
- Higher Bias:
  - Decision tree of lower depth
  - Linear functions
  - Important features missing

if we train models $f_D(X)$ on many training sets D, bias is the expected difference between their predictions and the true y's.

$$Bias = \mathrm{E}[f_D(X) - y]$$

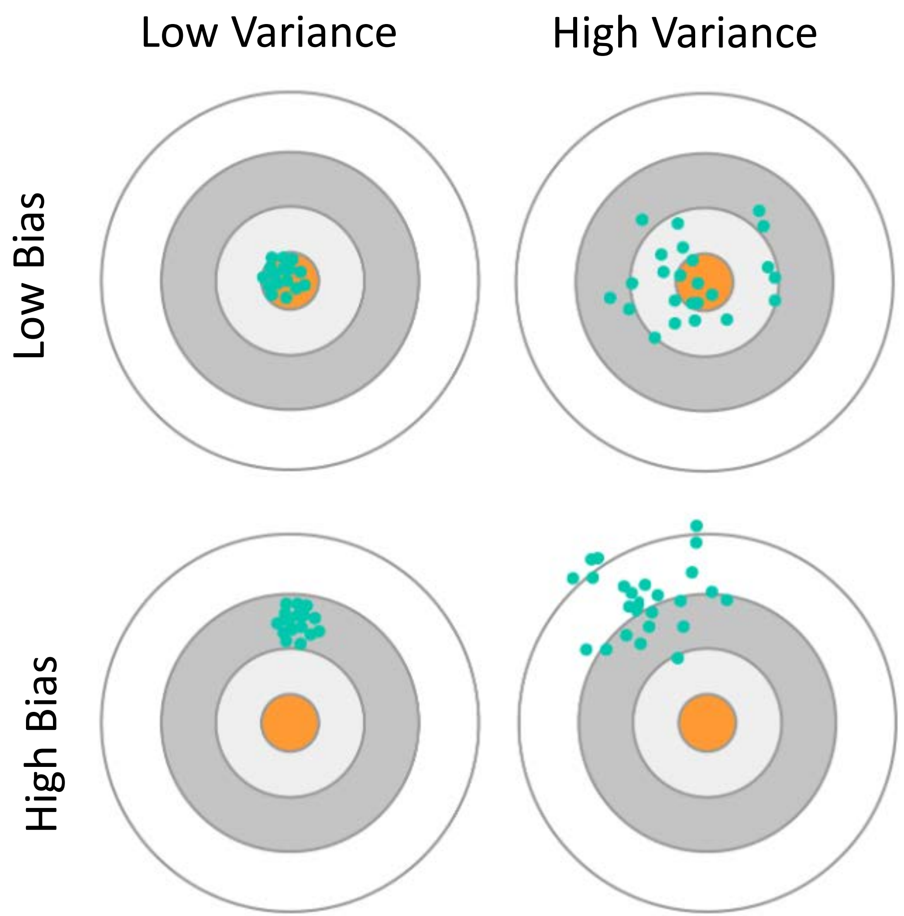
VARIANCE

- Error caused because the learned model reacts to small changes (noise) in the training data
- High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs
- Higher Variance
  - Decision tree with large no of nodes
  - High degree polynomials
  - Many features

if we train models $f_D(X)$ on many training sets D, variance is the variance of the estimates:

$$Variance = \mathrm{E}\left[\left(f_D(X) - \bar{f}(X)\right)^2\right]$$

Bias and Variance

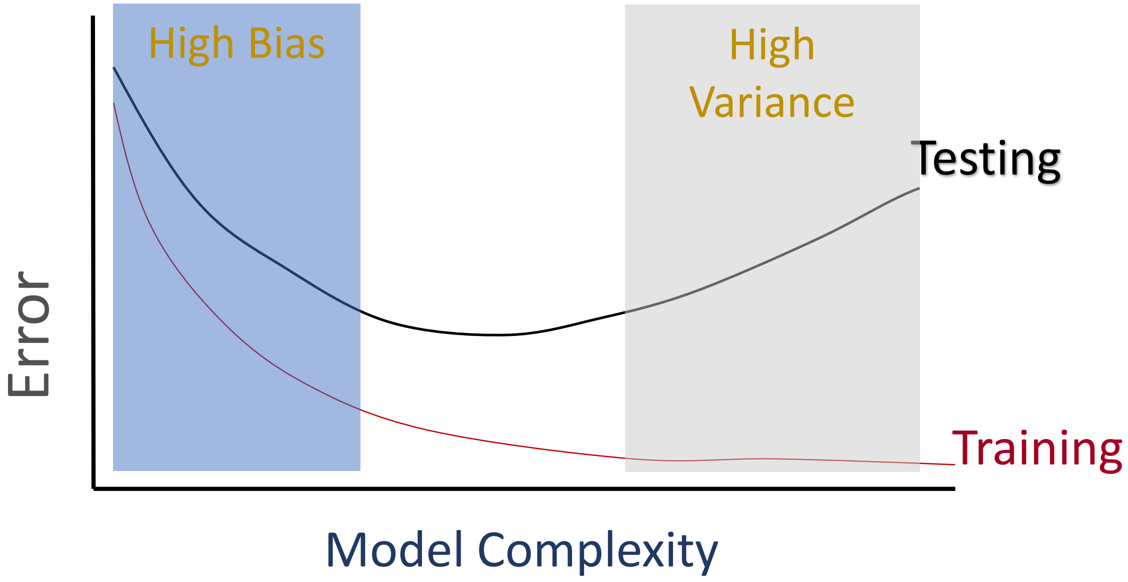


Bias and Variance Tradeoff

There is usually a bias-variance tradeoff caused by model complexity.

Complex models often have lower bias, but higher variance.

Simple models often have higher bias, but lower variance.

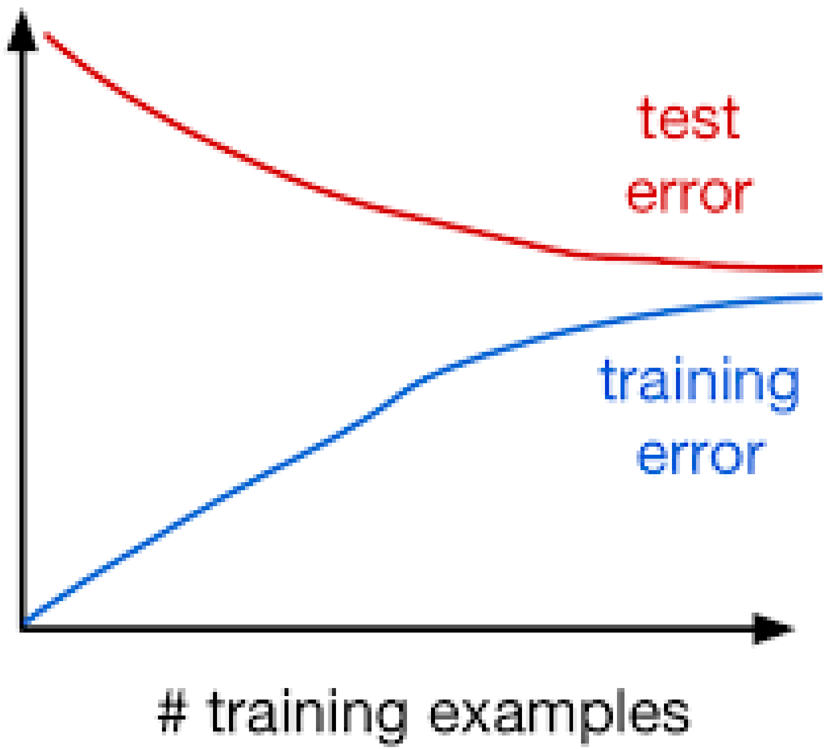


Trade-Offs

- There is a trade-off between these factors:
  - Complexity of Model $c(H)$
  - Training set size, *m,*
  - Generalization error, *E* on new data

1. As *m* *increases*, *E* decreases
2. As *c* (H) *increases*, first *E* *decreases* and then *E* *increases*
3. As *c* (H) *increases*, the training error *decreases* for some time and then stays constant (frequently at 0)

As m increases, E decreases



Model complexity

2. As c (H) increases, first E decreases and then E increases
3. As *c* (H) *increases*, the training error *decreases* for some time and then stays constant (frequently at 0)