



# Overfitting

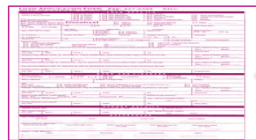


# in decision trees

Emily Fox & Carlos Guestrin  
Machine Learning Specialization  
University of Washington

# Review of loan default prediction

Loan  
Applications

A pink-outlined rectangular box representing a loan application form.A blue-outlined rectangular box representing a loan application form.A green-outlined rectangular box representing a loan application form.

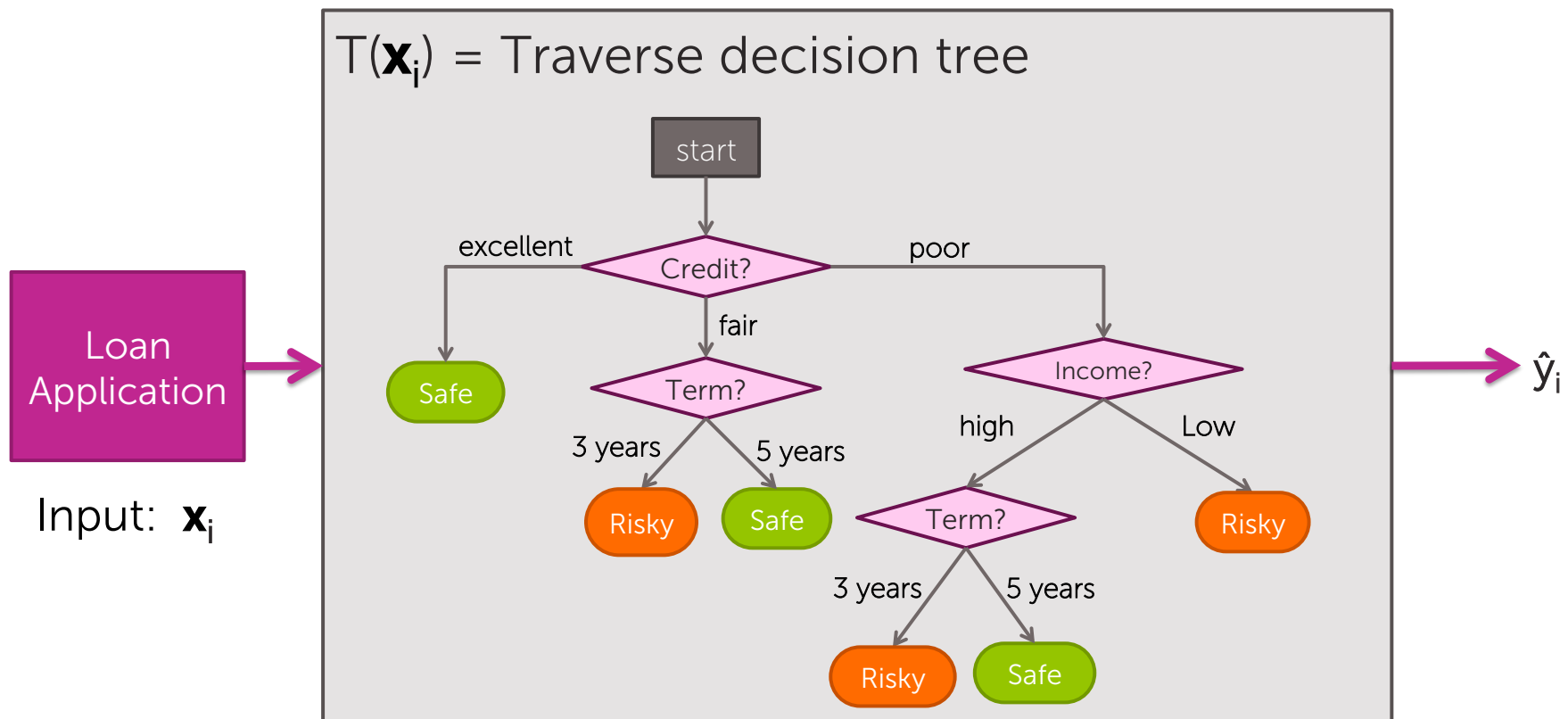
Intelligent loan application  
review system

Safe  
✓

Risky  
✗

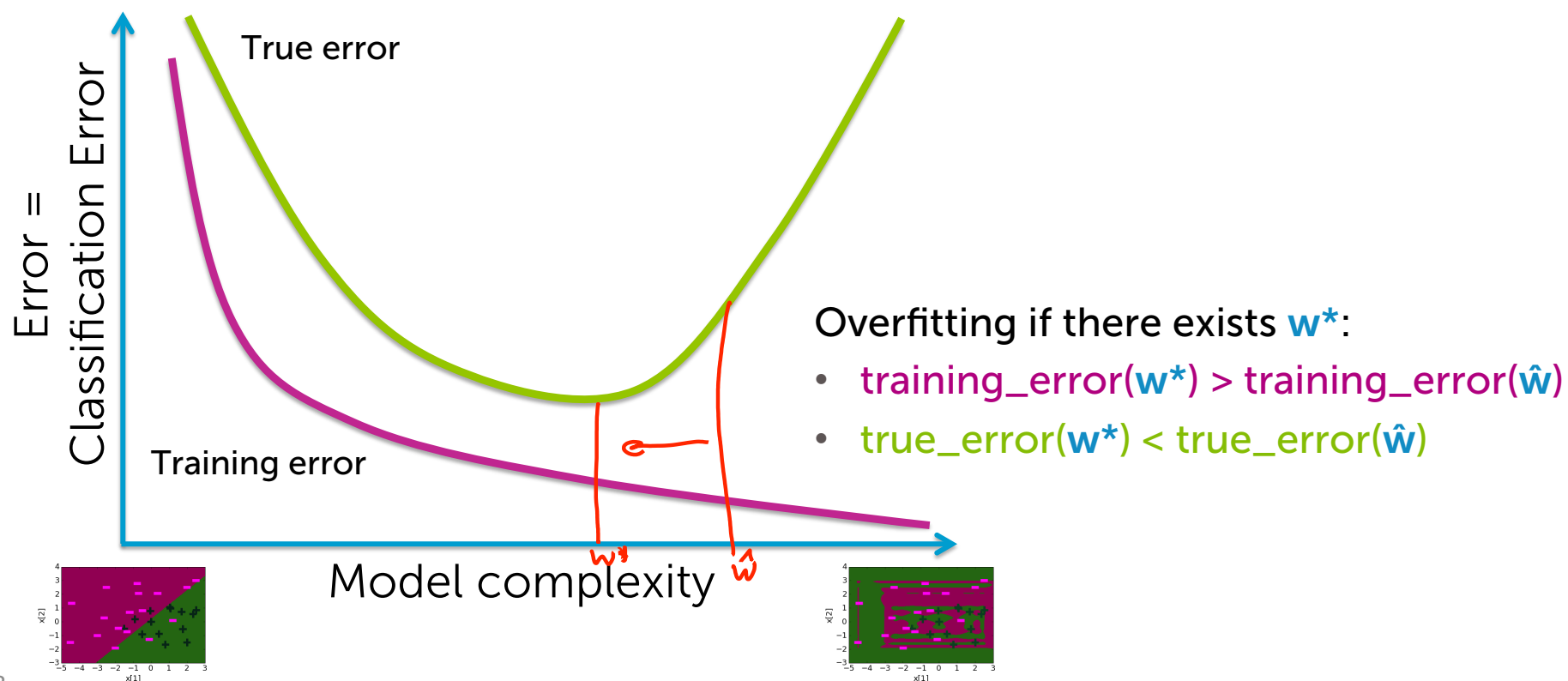
Risky  
✗

# Decision tree review

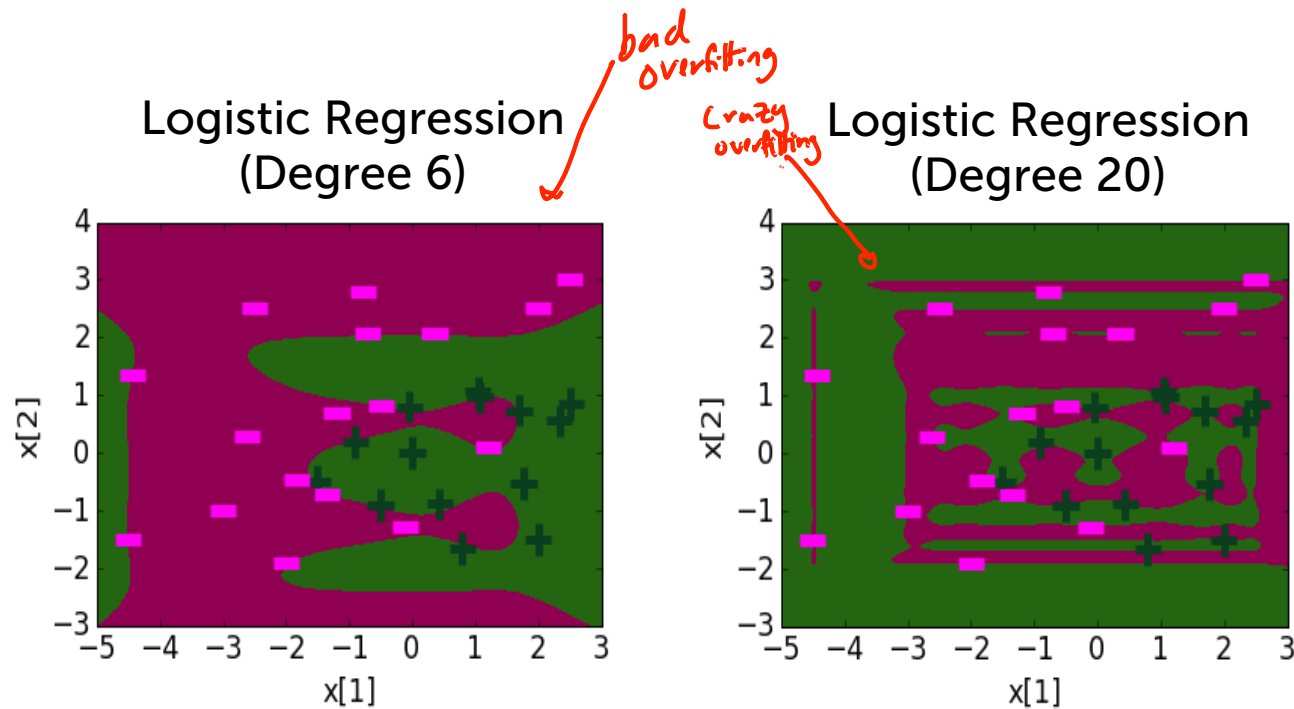


# Overfitting review

# Overfitting in logistic regression

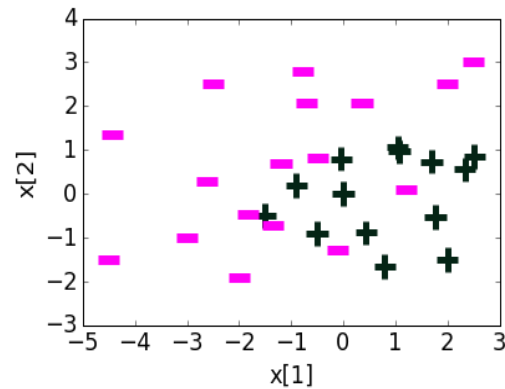


# Overfitting → Overconfident predictions

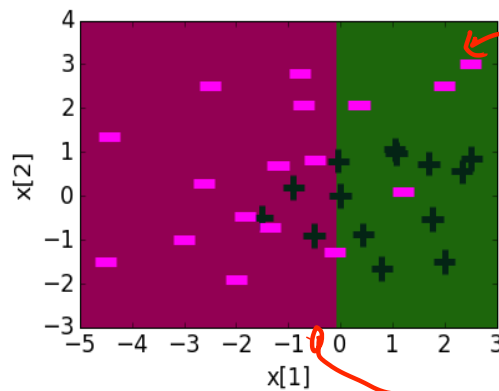
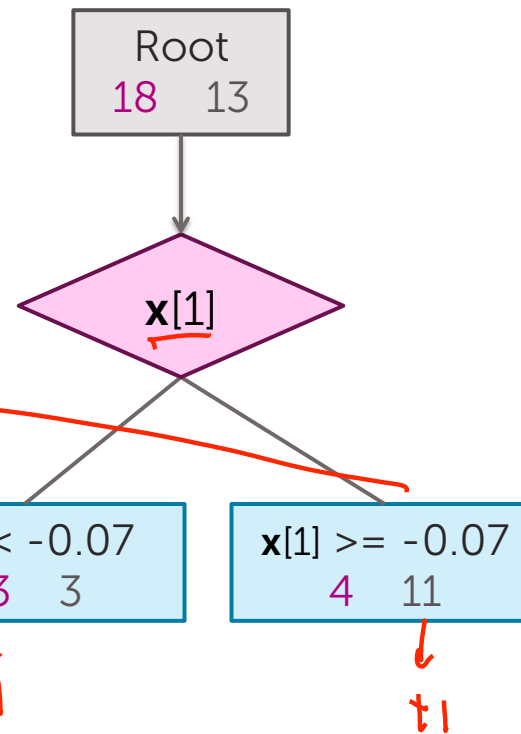


# Overfitting in decision trees

# Decision stump (Depth 1): Split on $x[1]$



y values  
- +

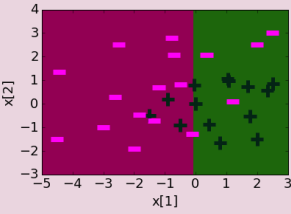
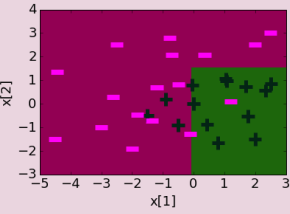
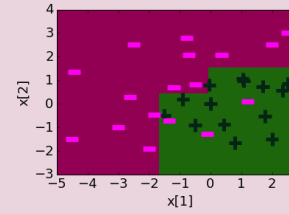
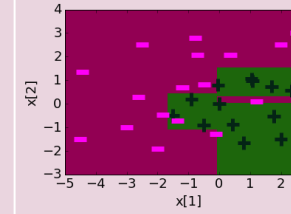
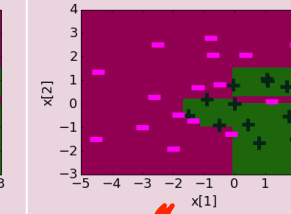




# What happens when we increase depth?

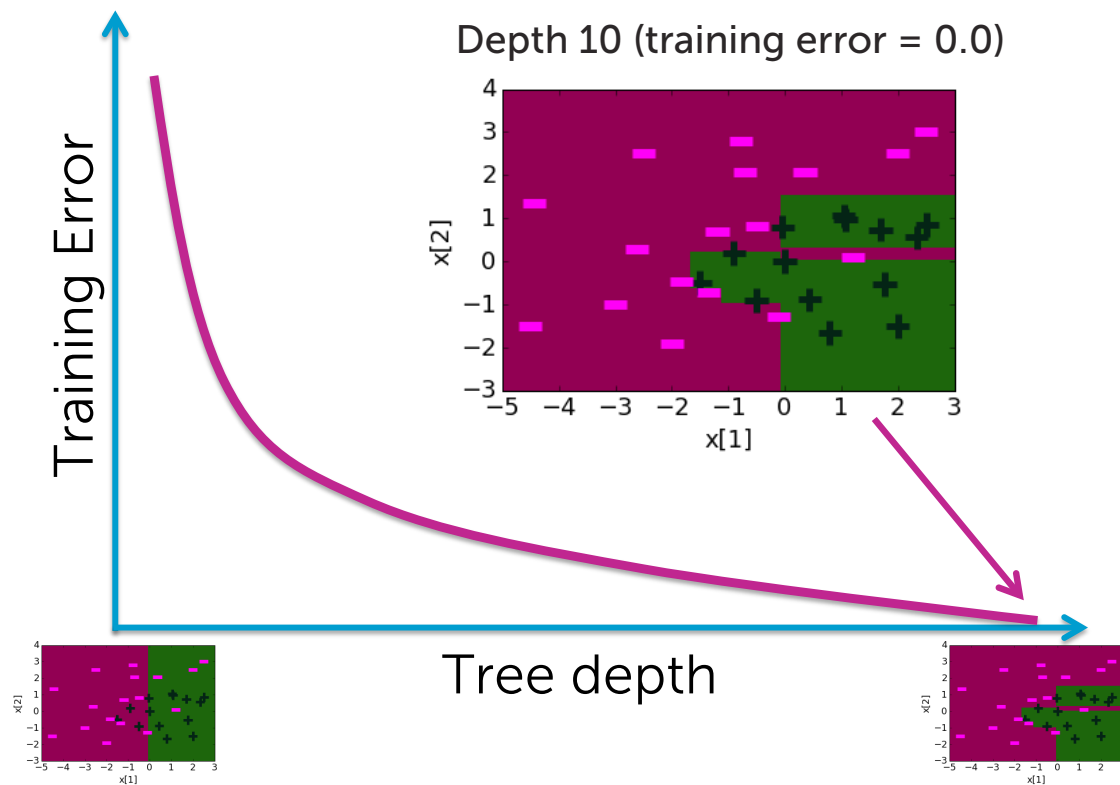
Training error reduces with depth

*Big warning!!*

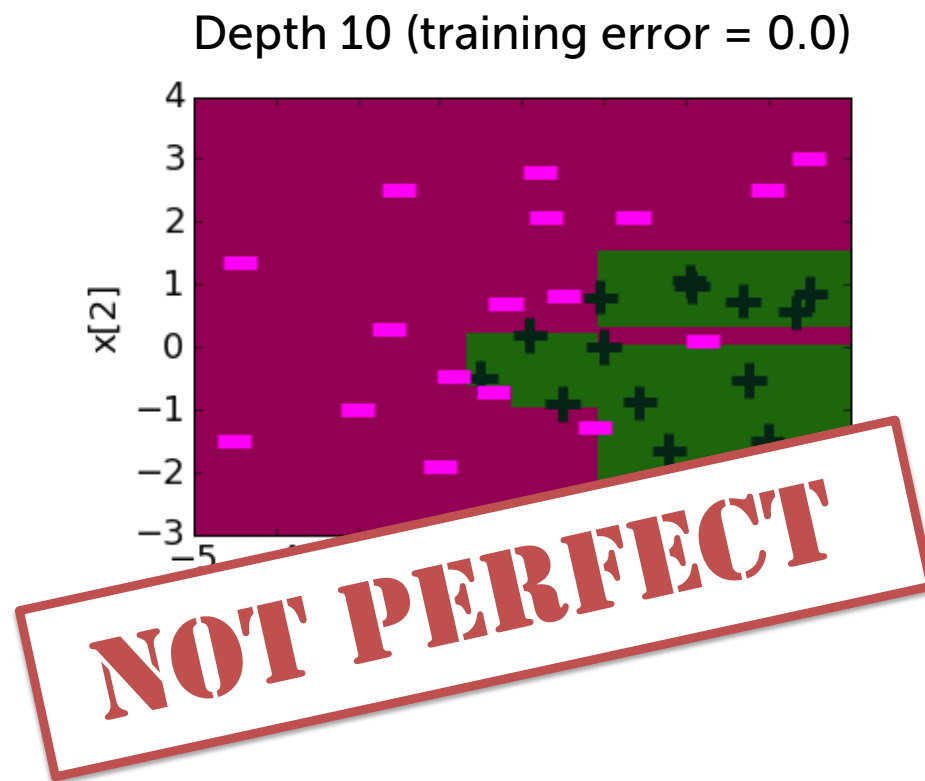
Tree depth	depth = 1	depth = 2	depth = 3	depth = 5	depth = 10
<u>Training error</u>	<u>0.22</u>	<u>0.13</u>	<u>0.10</u>	0.03	<u>0.00</u>
Decision boundary					

*complexity of decision boundary*

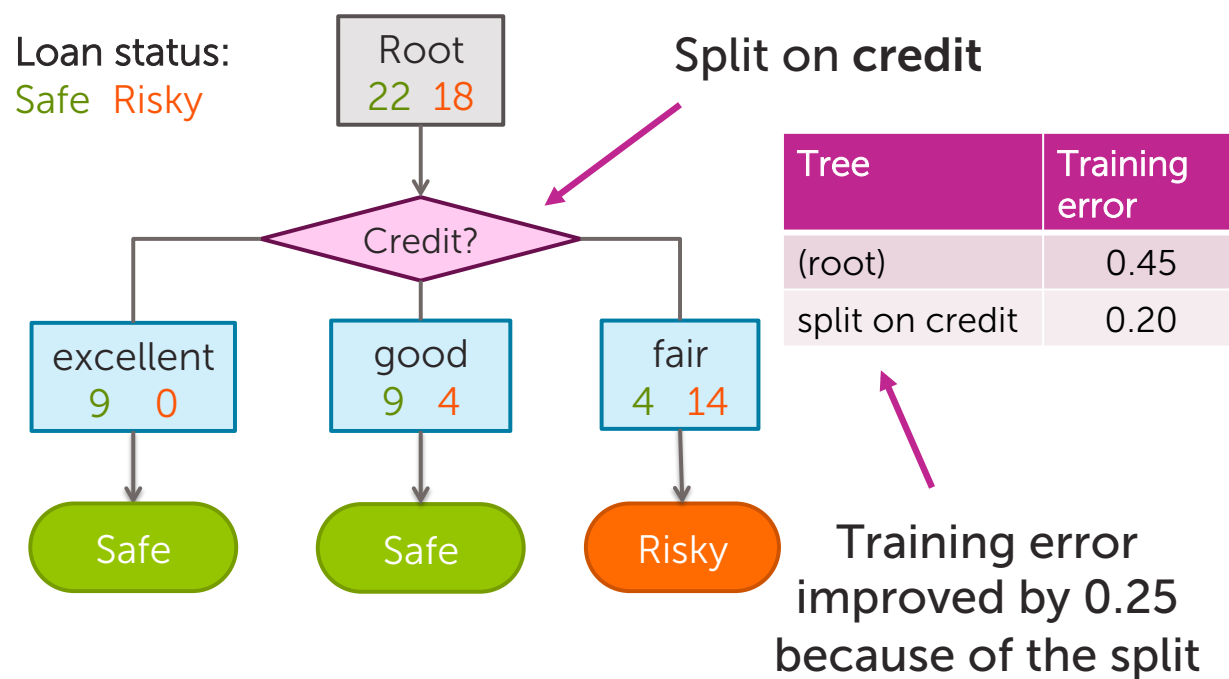
# Deeper trees → lower training error



# Training error = 0: Is this model perfect?



# Why training error reduces with depth?

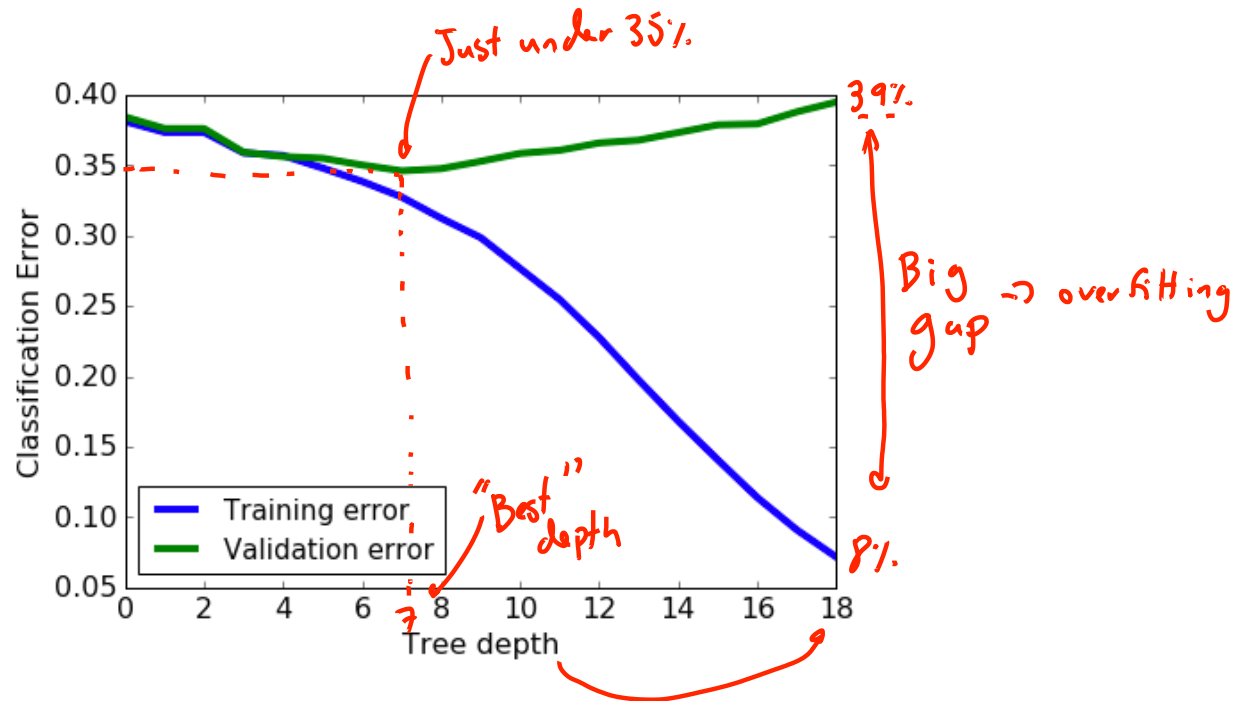


# Feature split selection algorithm

- Given a subset of data  $M$  (a node in a tree)
- For each feature  $h_i(\mathbf{x})$ :
  1. Split data of  $M$  according to feature  $h_i(\mathbf{x})$
  2. Compute classification error split
- Chose feature  $h^*(\mathbf{x})$  with lowest classification error

By design, each split  
reduces training error

# Decision trees overfitting on loan data



Principle of Occam's razor:  
*Simpler trees are better*

# Principle of Occam's Razor



*"Among competing hypotheses, the one with fewest assumptions should be selected",  
William of Occam, 13<sup>th</sup> Century*

Symptoms:  $S_1$  and  $S_2$

Diagnosis 1: 2 diseases

Two diseases  $D_1$  and  $D_2$  where  
 $D_1$  explains  $S_1$ ,  $D_2$  explains  $S_2$

OR

**SIMPLER**

Diagnosis 2: 1 disease

Disease  $D_3$  explains both  
symptoms  $S_1$  and  $S_2$



# Occam's Razor for decision trees

*When two trees have similar classification error on the validation set, pick the simpler one*

Complexity	Train error	Validation error
Simple	0.23	0.24
<b>Moderate</b>	0.12	0.15
<b>Complex</b>	0.07	0.15
Super complex	0	0.18

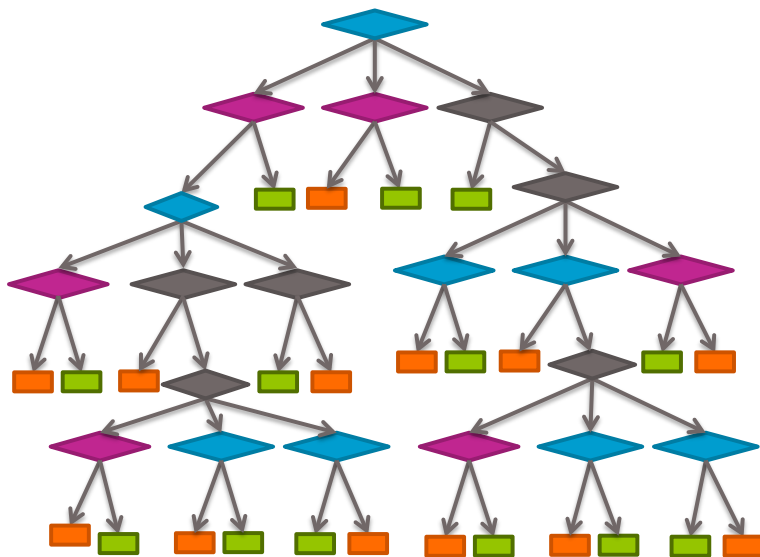
Same validation error

pick

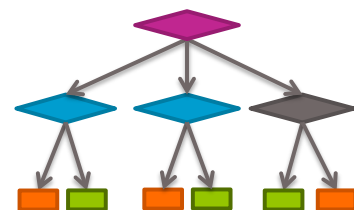
bad!

Overfit

# Which tree is simpler?



**OR**

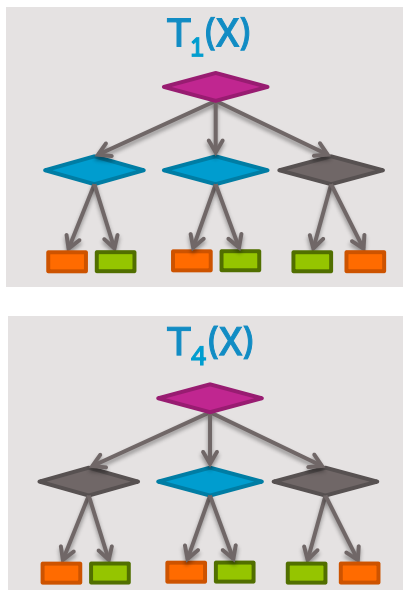


## SIMPLER

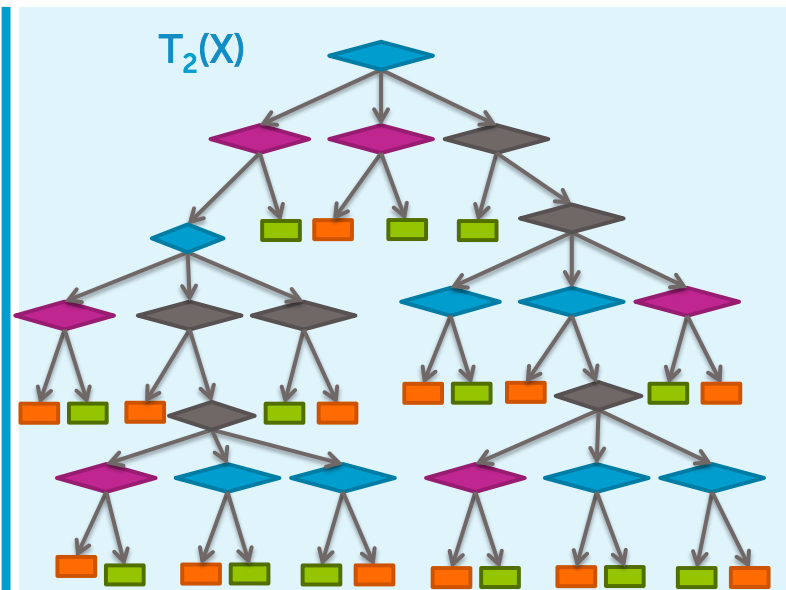
# Modified tree learning problem

Find a “**simple**” decision tree with low classification error

Simple trees



Complex trees





# How do we pick simpler trees?

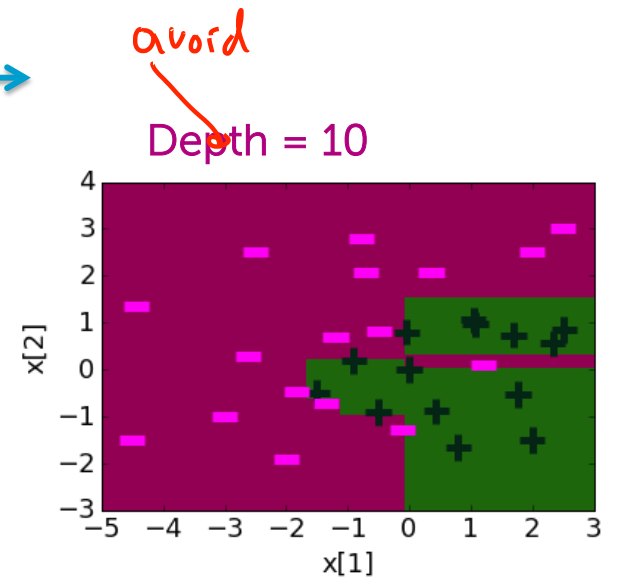
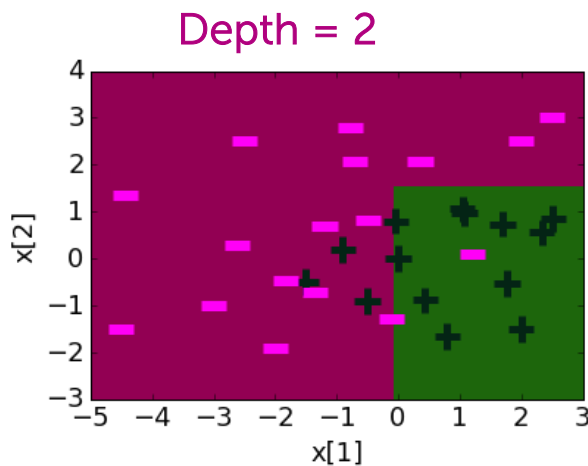
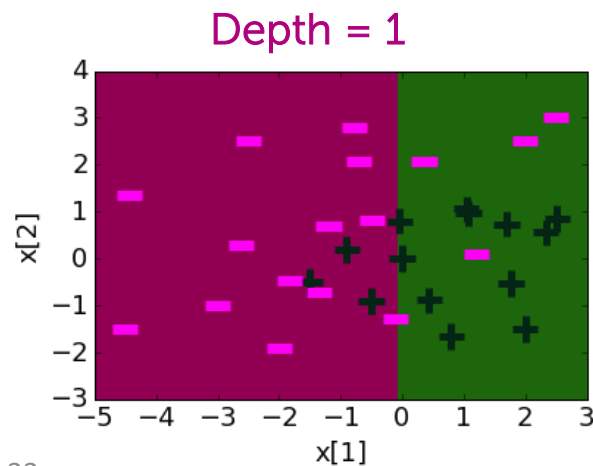
1. **Early Stopping:** Stop learning algorithm **before** tree become too complex
2. **Pruning:** Simplify tree **after** learning algorithm terminates



# *Early stopping for learning decision trees*

# Deeper trees → Increasing complexity

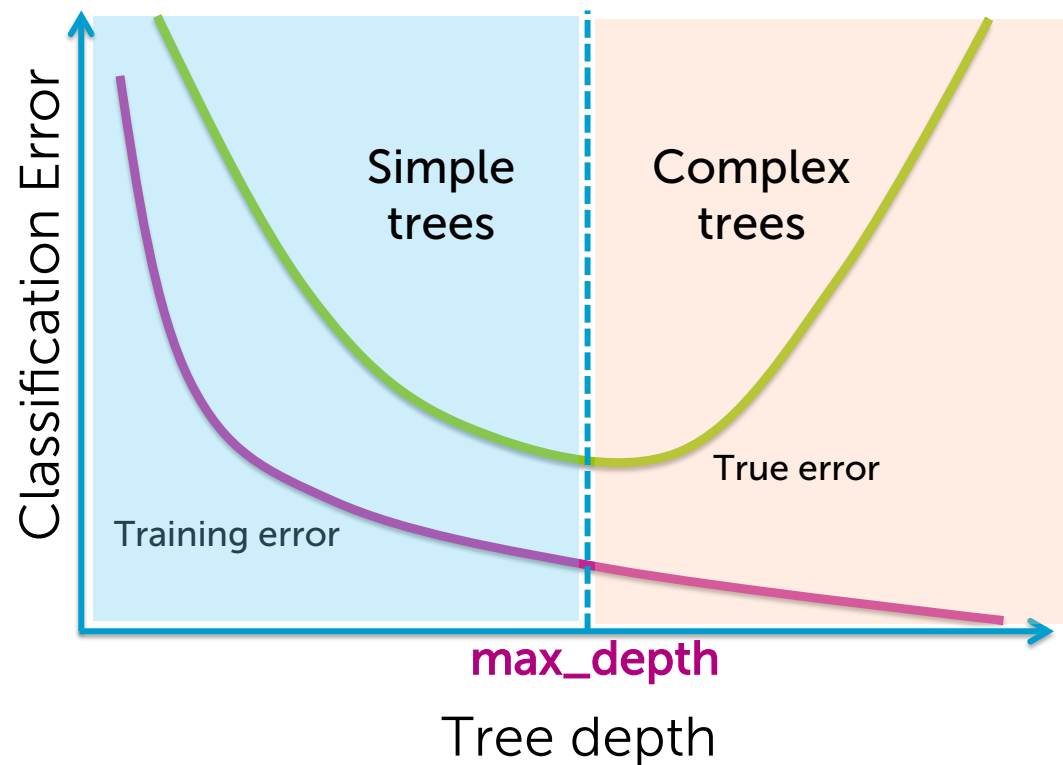
Model complexity increases with depth





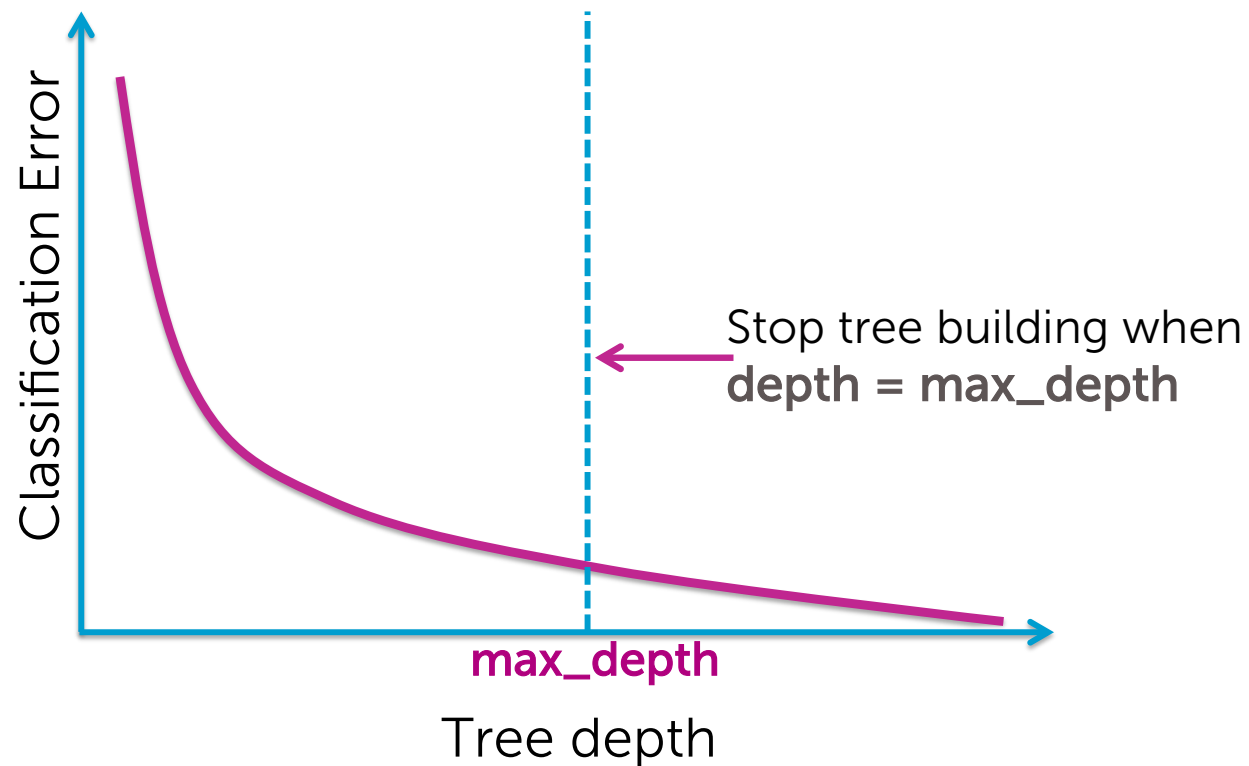
Early stopping condition 1:  
*Limit the depth of a tree*

# Restrict tree learning to shallow trees?

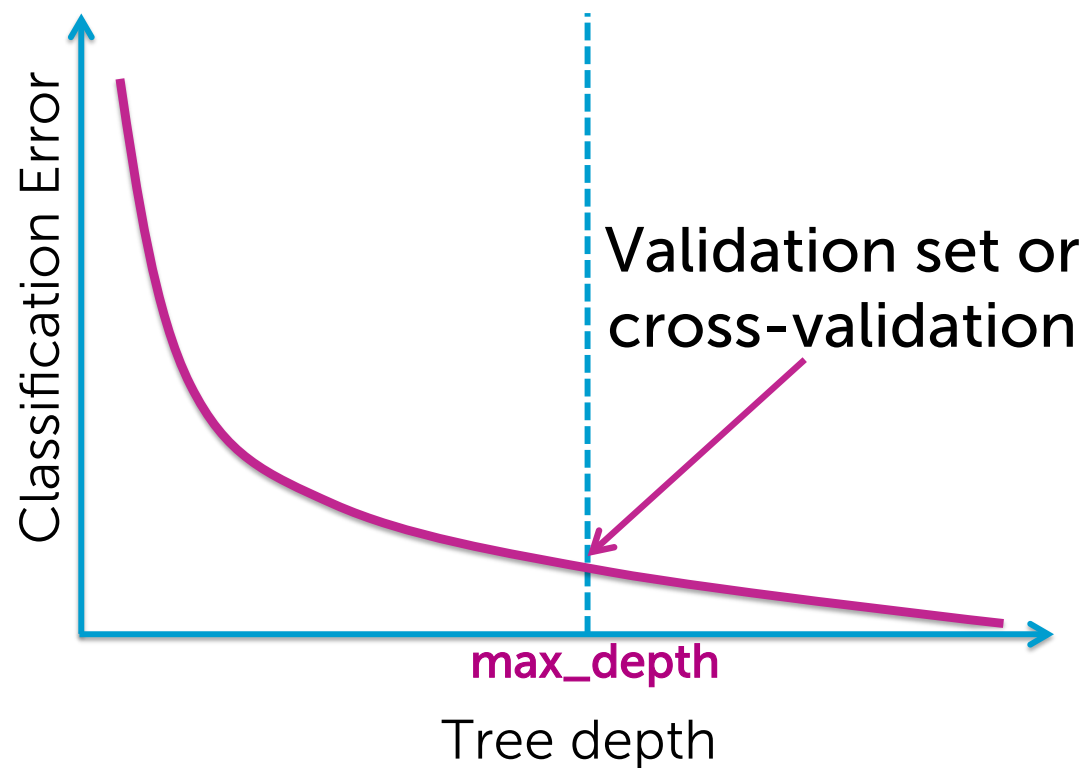




## Early stopping condition 1: Limit depth of tree



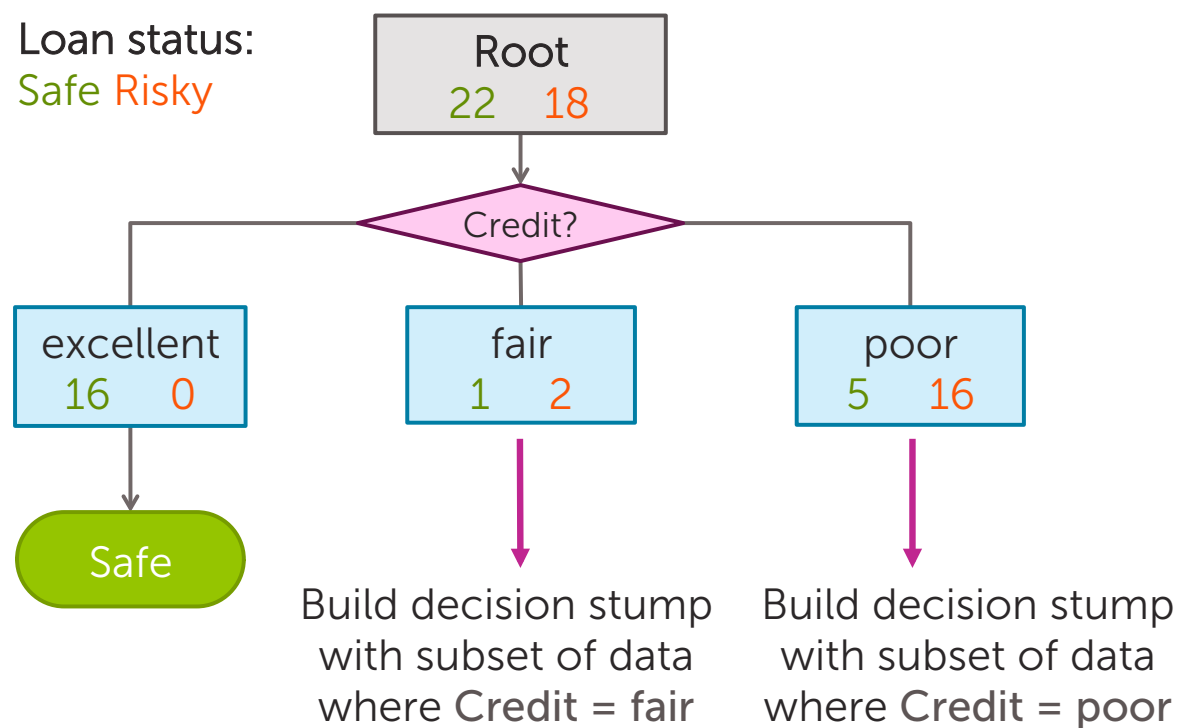
## Picking value for `max_depth`???



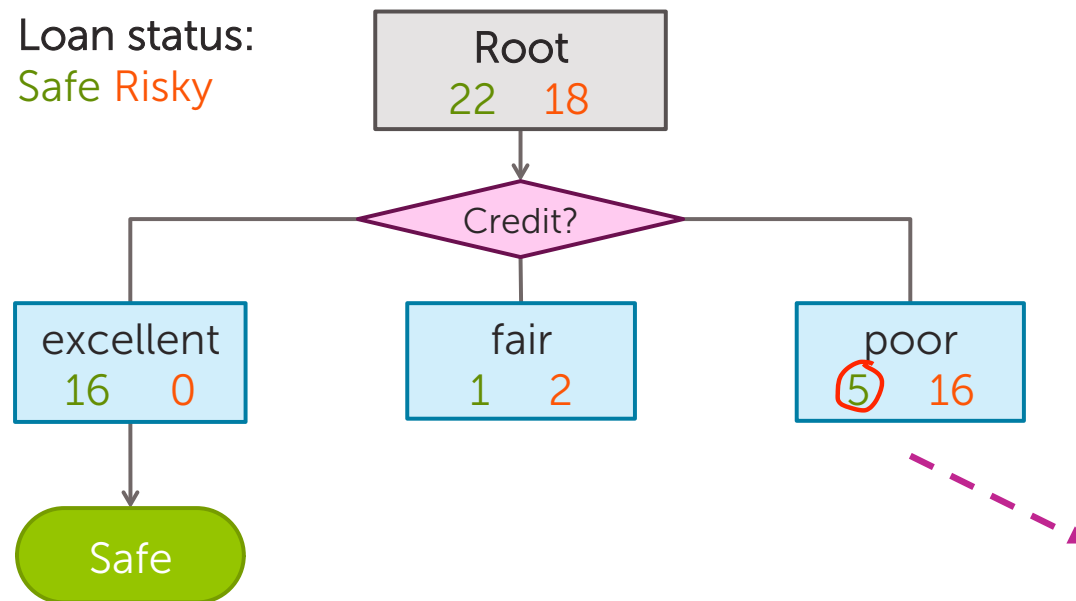


Early stopping condition 2:  
*Use classification error to  
limit depth of tree*

# Decision tree recursion review



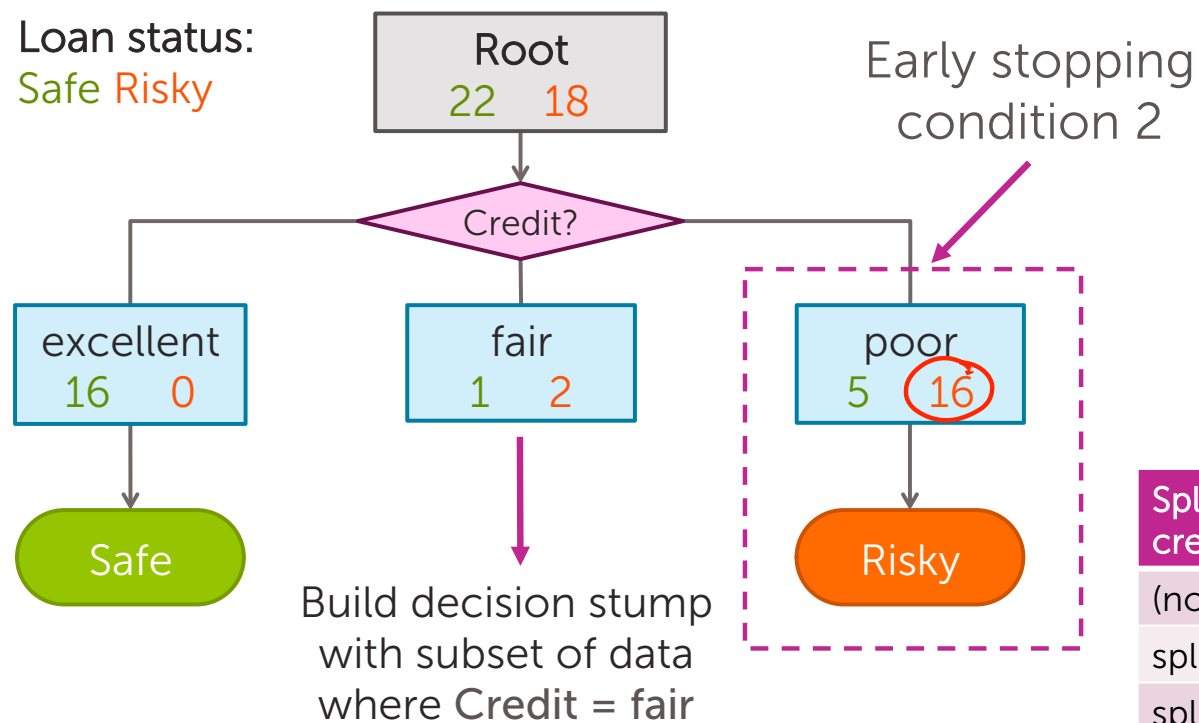
# Split selection for credit=poor



No split improves  
classification error  
→ Stop!

Splits for credit=poor	Classification error
(no split)	<u>0.24</u>
split on term	<u>0.24</u>
split on income	<u>0.24</u>

## Early stopping condition 2: No split improves classification error



Splits for credit=poor	Classification error
(no split)	0.24
split on term	0.24
split on income	0.24



## Practical notes about stopping when classification error doesn't decrease

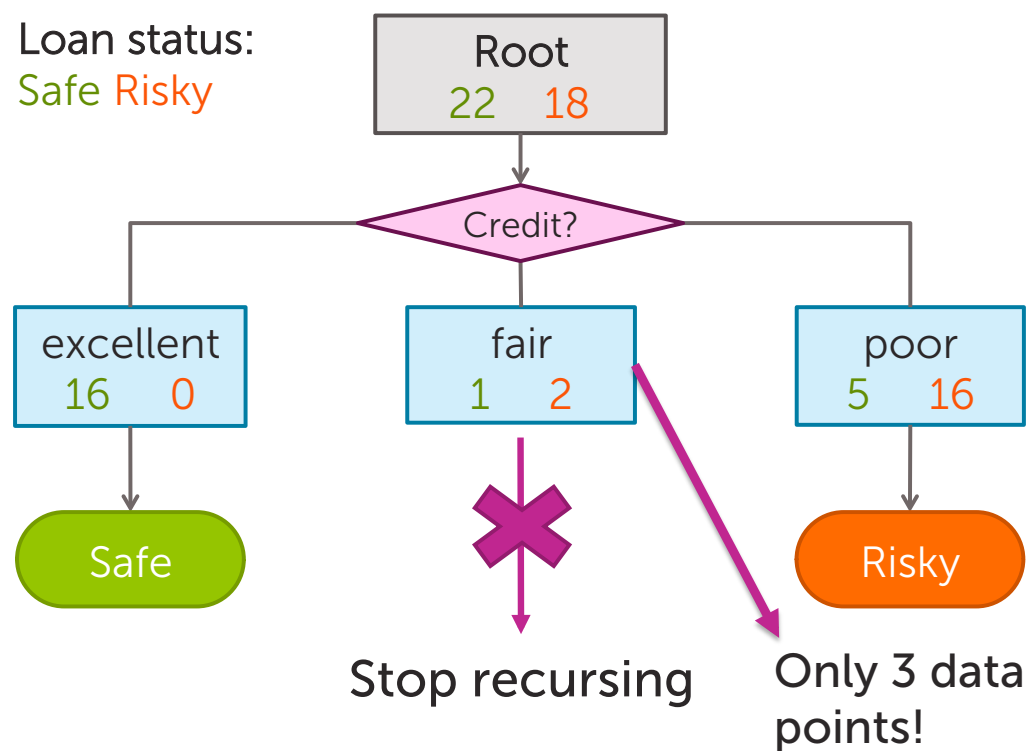
1. Typically, add magic parameter  $\epsilon$ 
  - Stop if error doesn't decrease by more than  $\epsilon$
2. Some pitfalls to this rule (see pruning section)
3. Very useful in practice



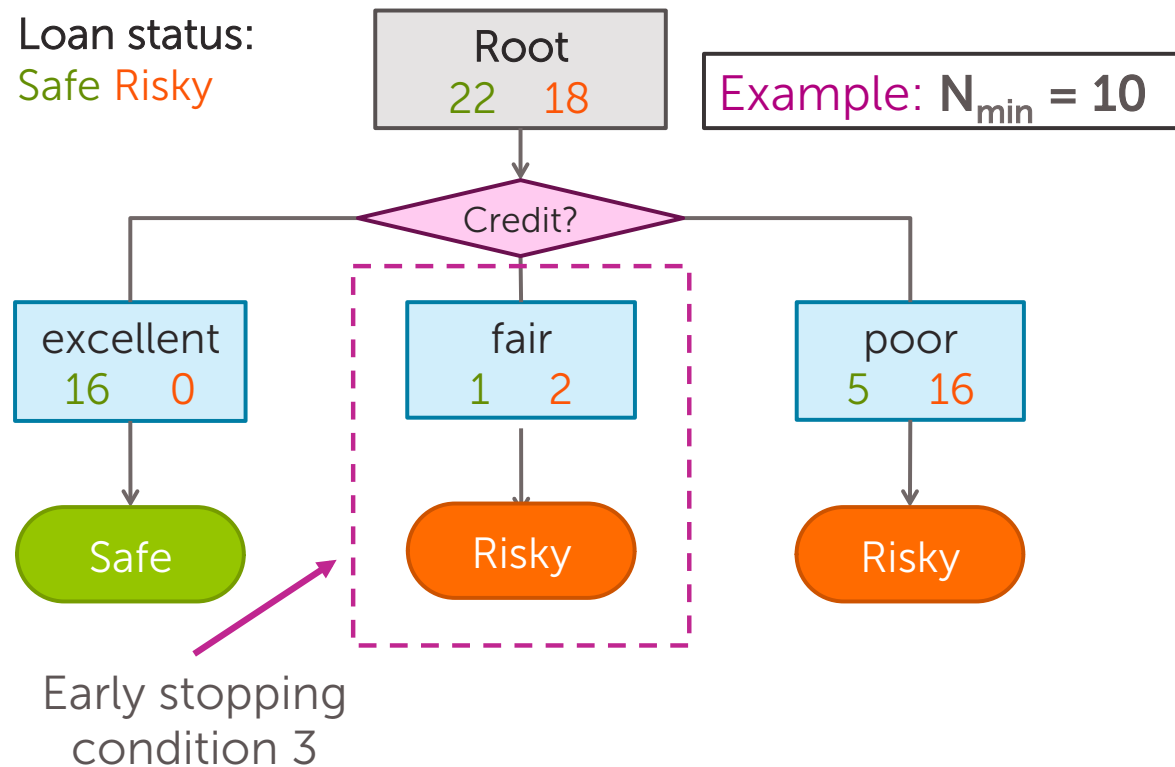
Early stopping condition 3:  
*Stop if number of data points  
contained in a node is too small*



# Can we trust nodes with very few points?



## Early stopping condition 3: Stop when data points in a node $\leq N_{\min}$





## Summary of decision trees with early stopping



# Early stopping: Summary

1. **Limit tree depth:** Stop splitting after a certain depth
2. **Classification error:** Do not consider any split that does not cause a sufficient decrease in classification error
3. **Minimum node "size":** Do not split an intermediate node which contains too few data points

# Greedy decision tree learning

- **Step 1:** Start with an empty tree
- **Step 2:** Select a feature to split data
- For each split of the tree:

- **Step 3:** If nothing more to, make predictions ← Majority

- **Step 4:** Otherwise, go to **Step 2** & continue (recurse) on this split

Previous module

Stopping conditions 1 & 2  
or  
Early stopping conditions 1, 2 & 3

Recursion

# Overfitting in Decision Trees: *Pruning*

**OPTIONAL**



# Stopping condition summary

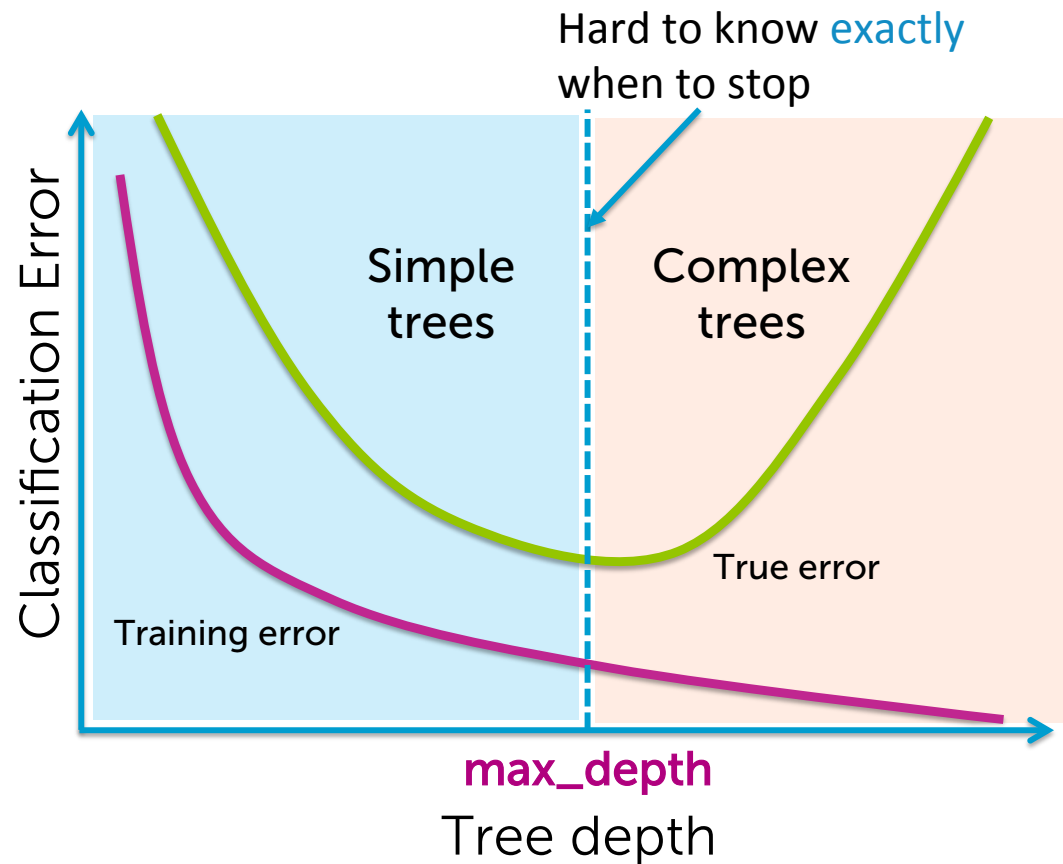
- Stopping condition:
  1. All examples have the same target value
  2. No more features to split on
- Early stopping conditions:
  1. Limit tree depth
  2. Do not consider splits that do not cause a sufficient decrease in classification error
  3. Do not split an intermediate node which contains too few data points



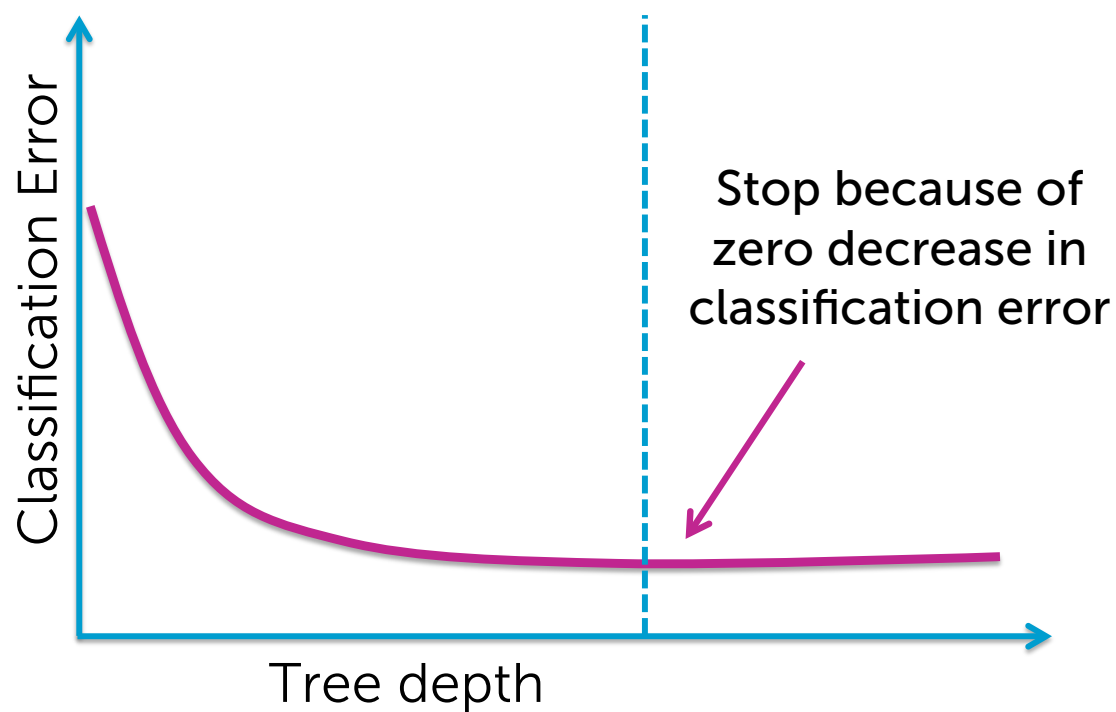
## Exploring some challenges with early stopping conditions



# Challenge with early stopping condition 1



# Is early stopping condition 2 a good idea?



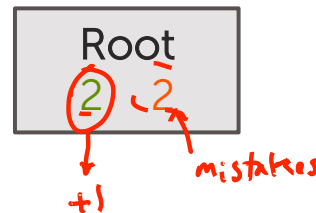
## Early stopping condition 2:

Don't stop if error doesn't decrease???

$$y = \mathbf{x}[1] \text{ xor } \mathbf{x}[2]$$

$\mathbf{x}[1]$	$\mathbf{x}[2]$	$y$
False	False	False
False	True	True
True	False	True
True	True	False

y values  
True False



$$\text{Error} = \frac{2}{4} = 0.5$$

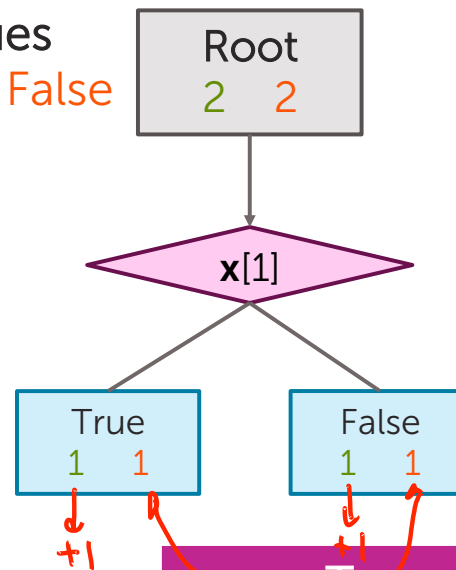
Tree	Classification error
(root)	0.5

# Consider split on $x[1]$

$$y = x[1] \text{ xor } x[2]$$

$x[1]$	$x[2]$	$y$
False	False	False
False	True	True
True	False	True
True	True	False

y values  
True False



$$\text{Error} = \frac{1+1}{4} = 0.5$$

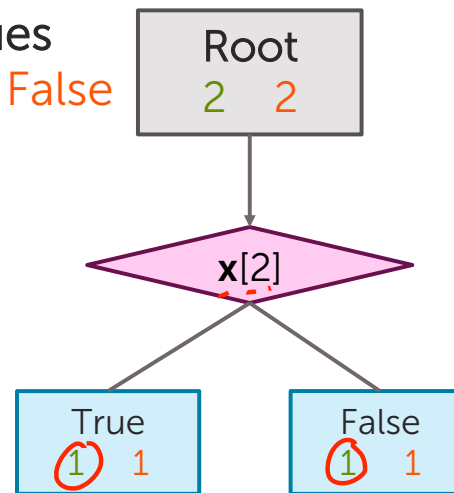
Tree	Classification error
(root)	0.5
Split on $x[1]$	0.5

# Consider split on $x[2]$

$$y = x[1] \text{ xor } x[2]$$

$x[1]$	$x[2]$	$y$
False	False	False
False	True	True
True	False	True
True	True	False

y values  
True False



$$\text{Error} = \frac{1 + 1}{4} = 0.5$$

Neither features  
improve training error...  
Stop now???

Tree	Classification error
(root)	0.5
Split on $x[1]$	0.5
Split on $x[2]$	0.5

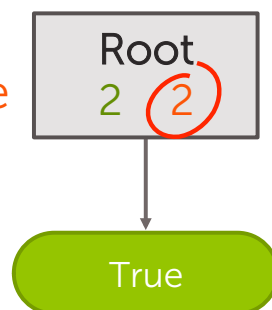
Same error

# Final tree with early stopping condition 2

$$y = \mathbf{x}[1] \text{ xor } \mathbf{x}[2]$$

$\mathbf{x}[1]$	$\mathbf{x}[2]$	$y$
False	False	False
False	True	True
True	False	True
True	True	False

y values  
True False



Tree	Classification error
with early stopping condition 2	0.5

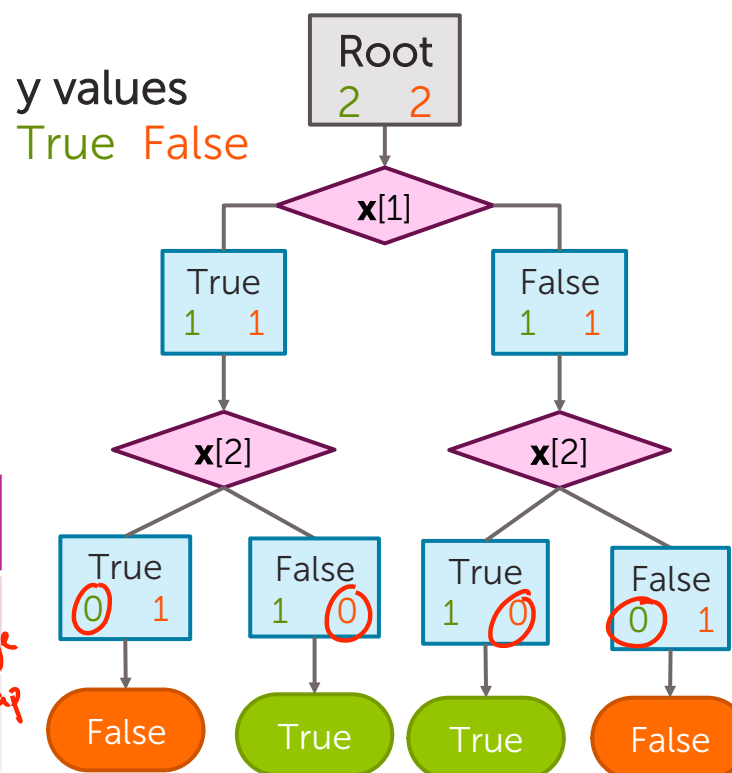
# Without early stopping condition 2

$$y = \mathbf{x}[1] \text{ xor } \mathbf{x}[2]$$

$\mathbf{x}[1]$	$\mathbf{x}[2]$	$y$
False	False	False
False	True	True
True	False	True
True	True	False

Tree	Classification error
with early stopping condition 2	0.5
without early stopping condition 2	<u>0.0</u>

*huge gap*





## Early stopping condition 2: Pros and Cons

- Pros:
  - A reasonable heuristic for early stopping to avoid useless splits
- Cons:
  - Too short sighted: We may miss out on “good” splits may occur right after “useless” splits





# Tree pruning



# Two approaches to picking simpler trees

1. **Early Stopping:** Stop the learning algorithm **before** the tree becomes too complex

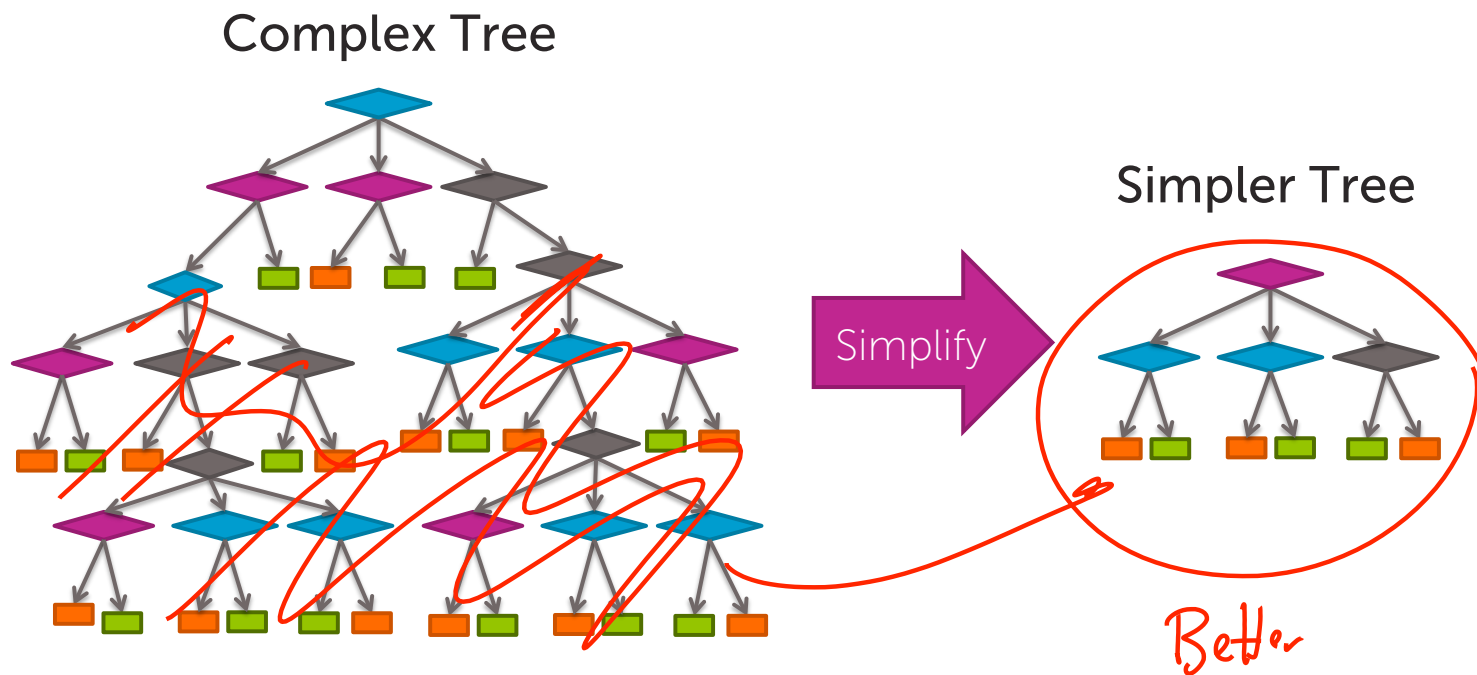
2. **Pruning:** Simplify the tree **after** the learning algorithm terminates



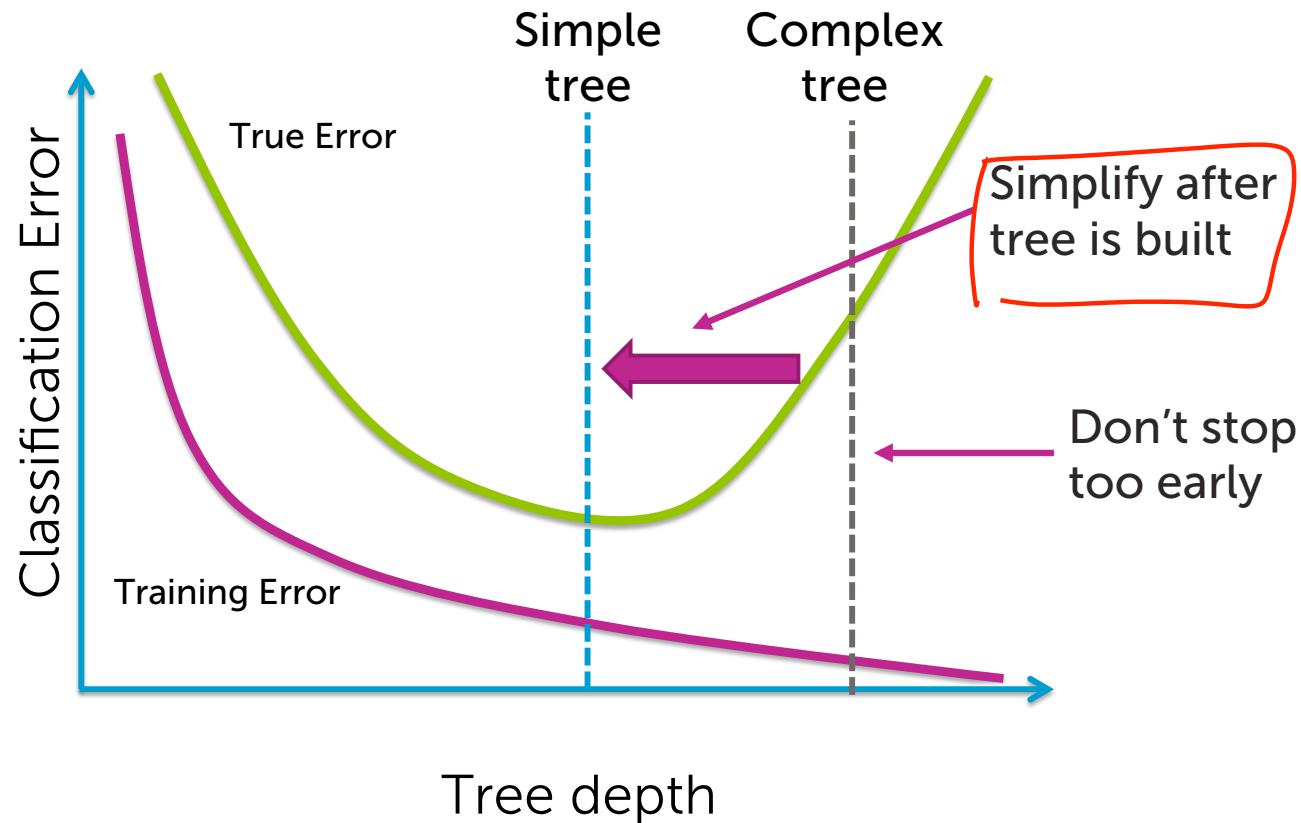
Complements early stopping

# Pruning: *Intuition*

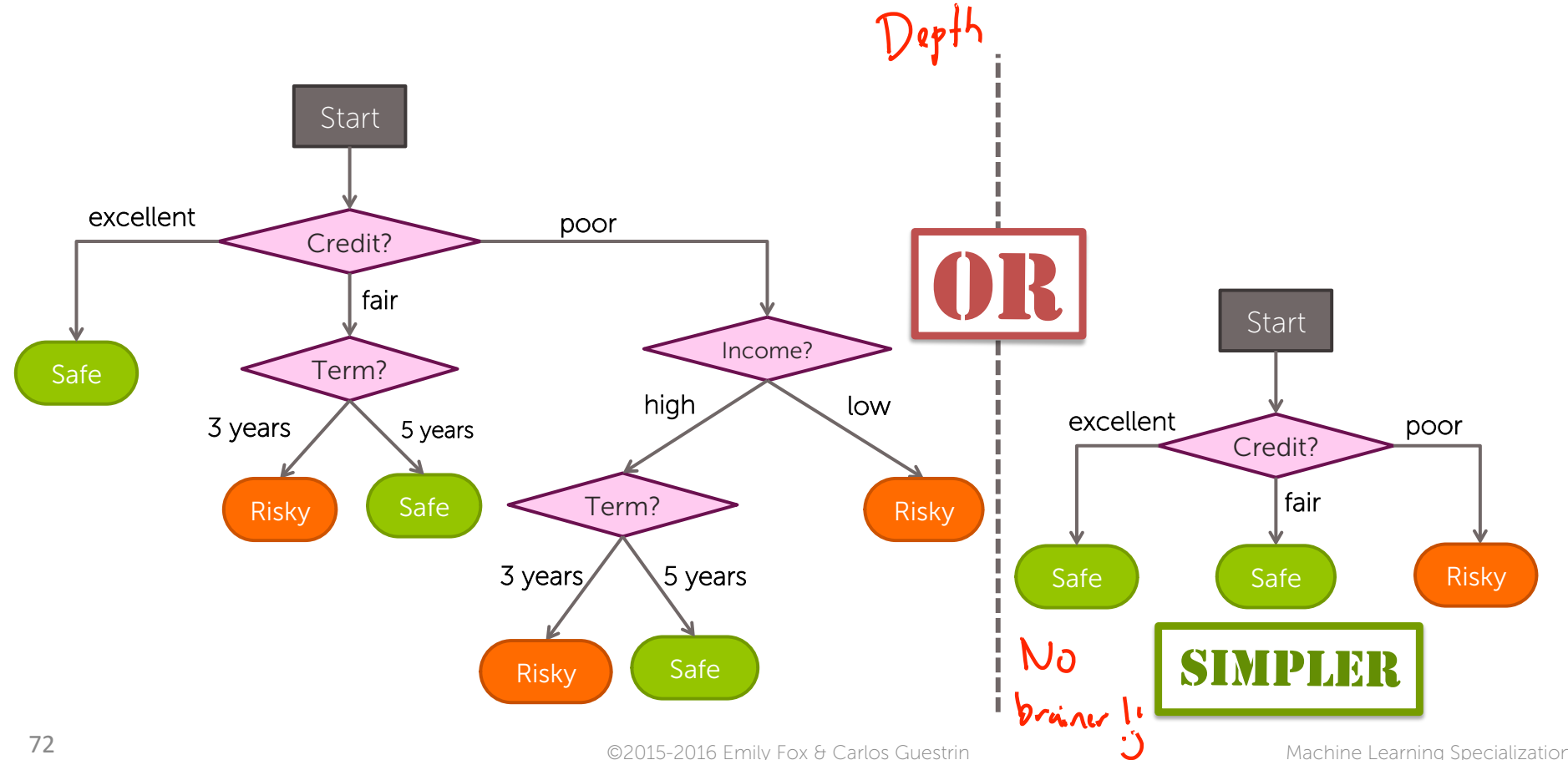
Train a complex tree, simplify later



# Pruning motivation

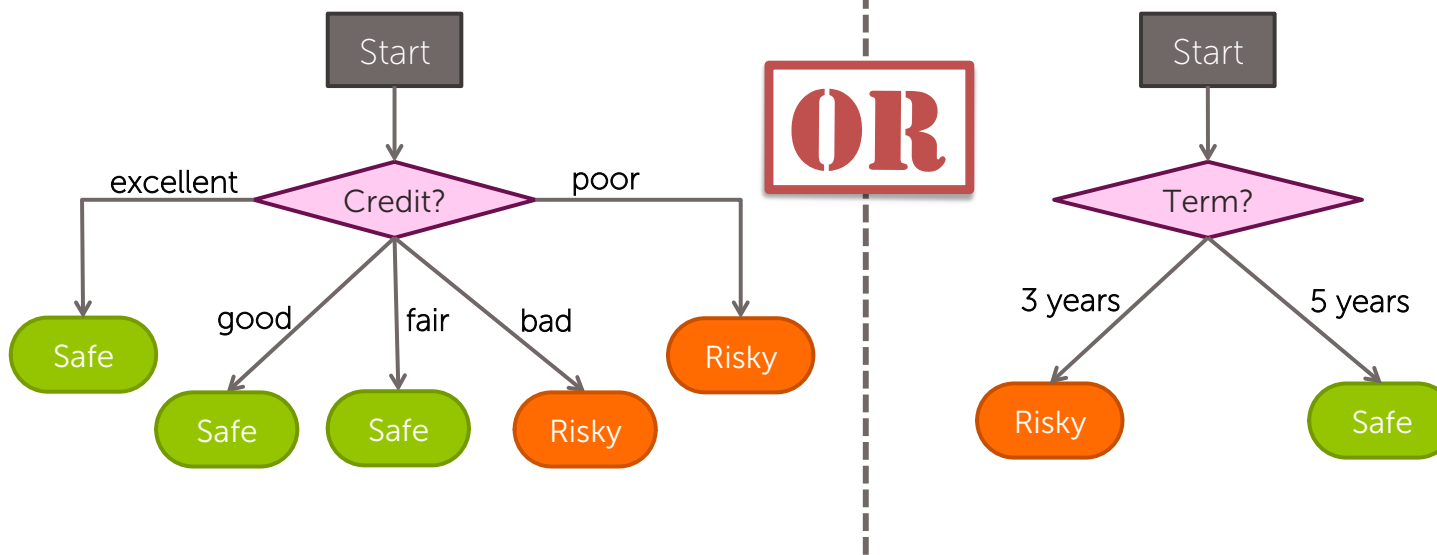


# Example 1: Which tree is simpler?

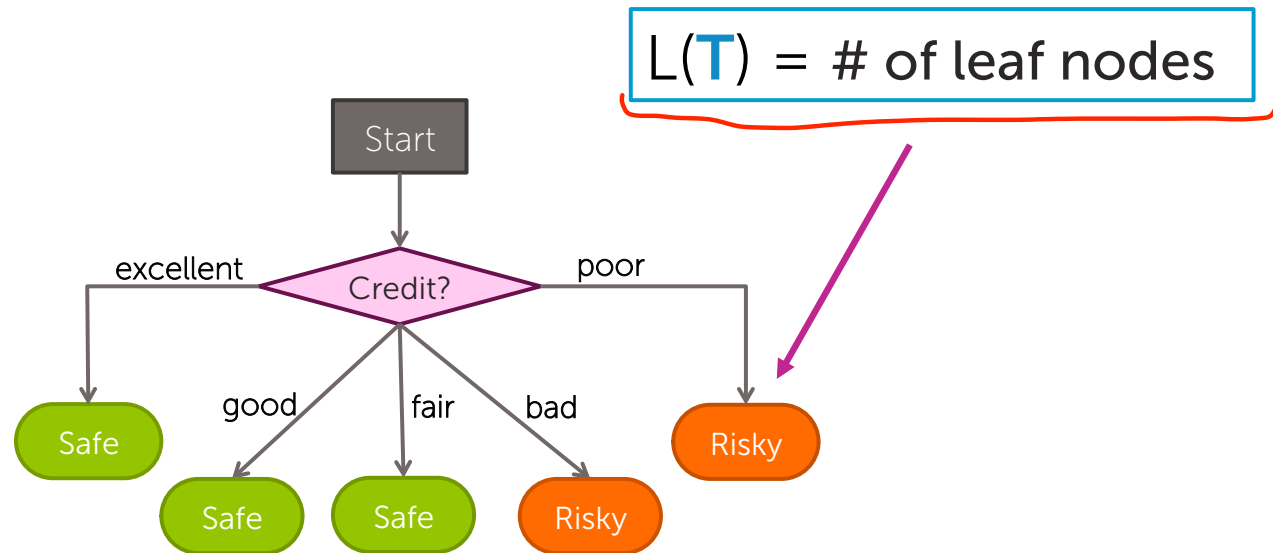


## Example 2: Which tree is simpler???

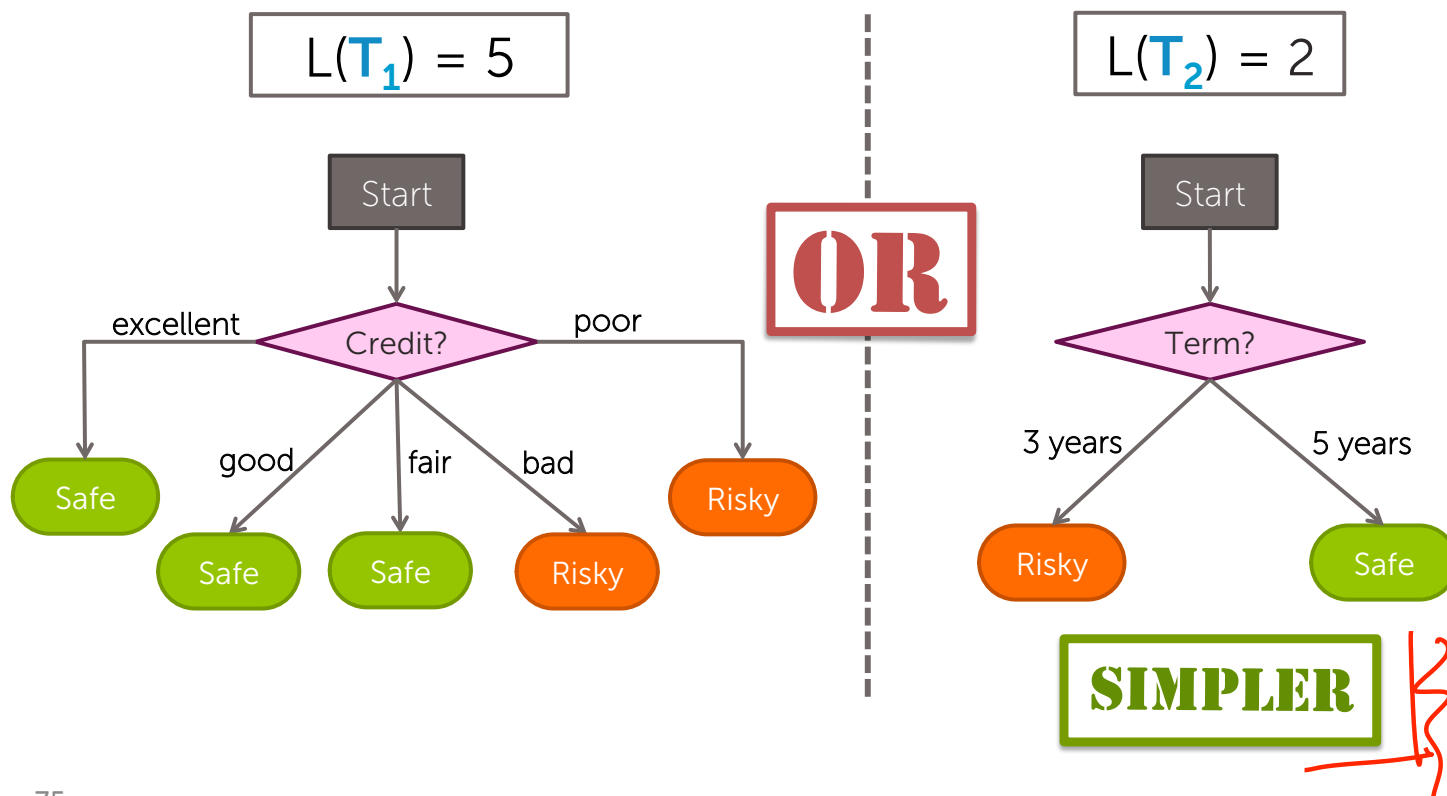
*Same Depth:  
Simpler?*



# Simple measure of complexity of tree



# Which tree has lower $L(T)$ ?

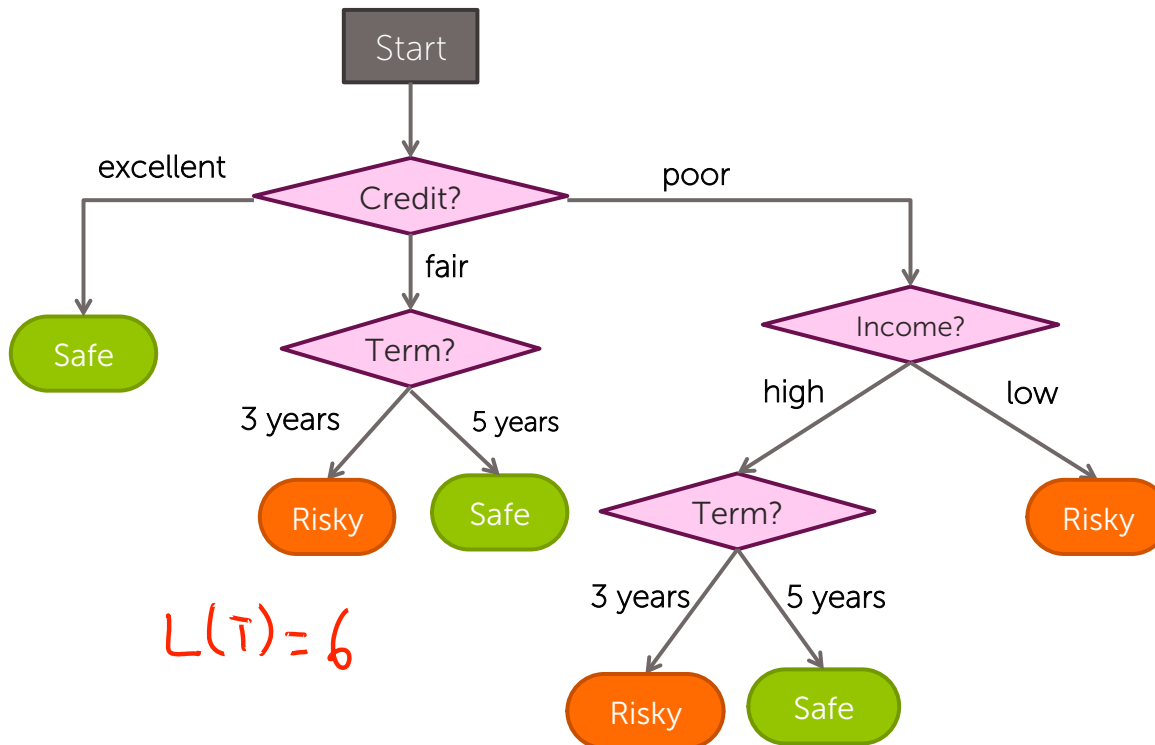




# Balance simplicity & predictive power

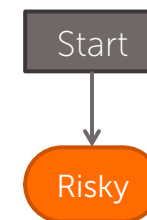
Too complex, risk of overfitting

*solution in between*



$$L(\bar{T}) = 6$$

Too simple, high classification error

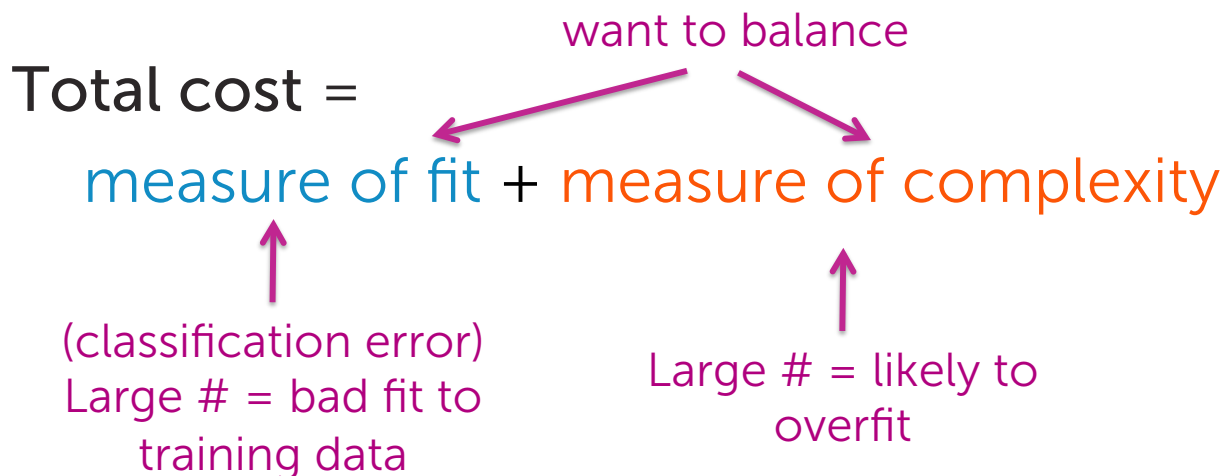


$$L(T) = 1$$

# Desired total quality format

Want to balance:

- i. How well tree fits data
- ii. Complexity of tree



## Consider specific total cost

$$\text{Total cost} = \underbrace{\text{classification error}}_{\text{Error}(\mathbf{T})} + \underbrace{\text{number of leaf nodes}}_{L(\mathbf{T})}$$

# Balancing fit and complexity

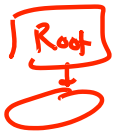
$$\text{Total cost } C(\mathbf{T}) = \text{Error}(\mathbf{T}) + \lambda L(\mathbf{T})$$

tuning parameter

If  $\lambda = 0$ :

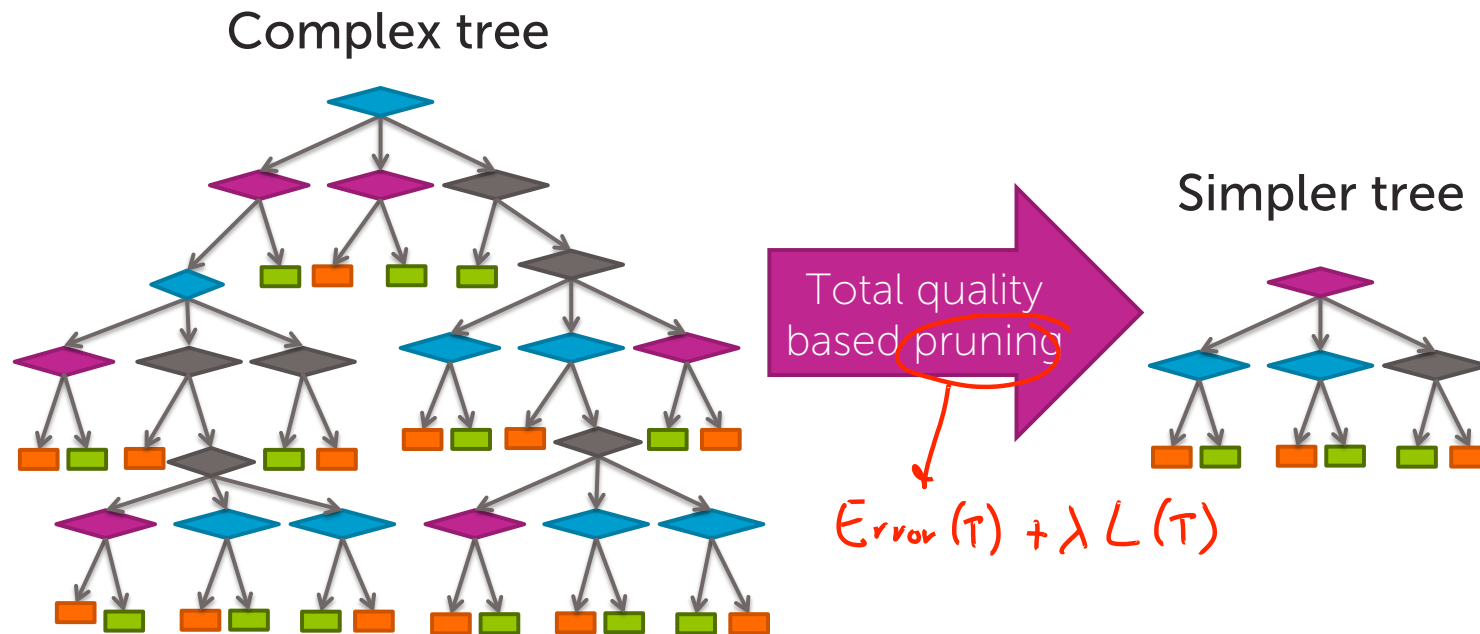
Standard decision tree learning

If  $\lambda = \infty$ :

$\infty$  penalty  $\rightarrow$    $\rightarrow \hat{y} = \text{Majority class}$

If  $\lambda$  in between: Balance fit & complexity

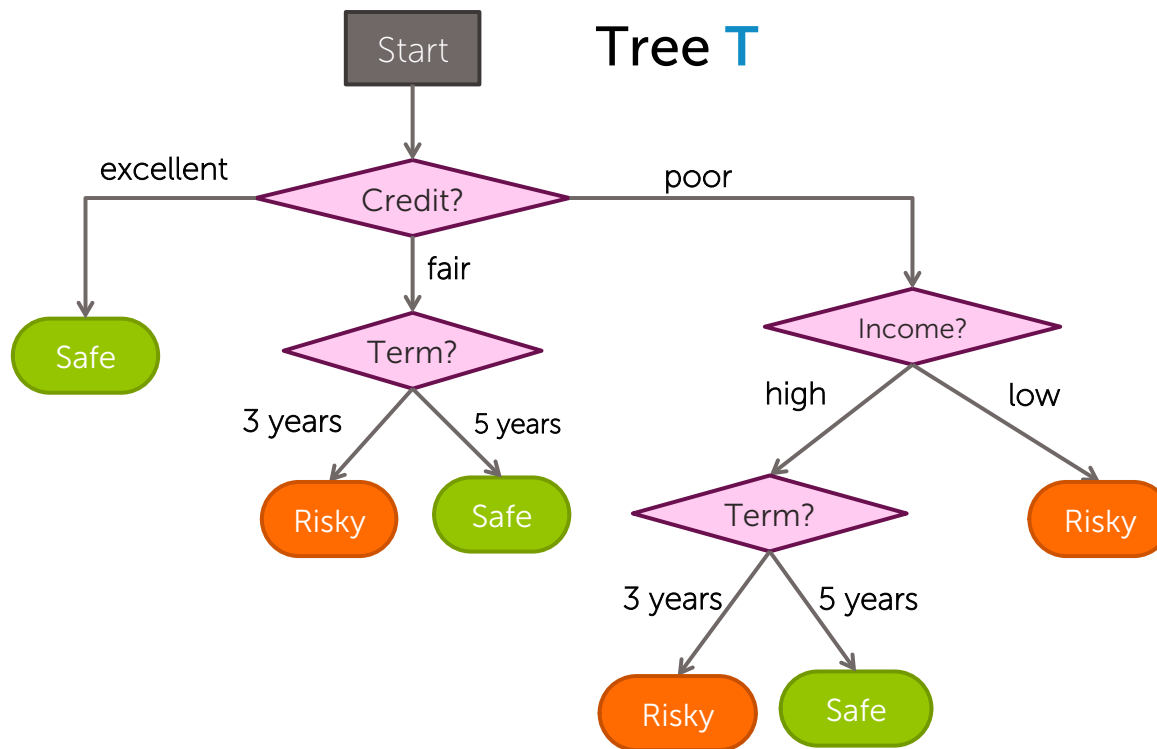
# Use total cost to simplify trees



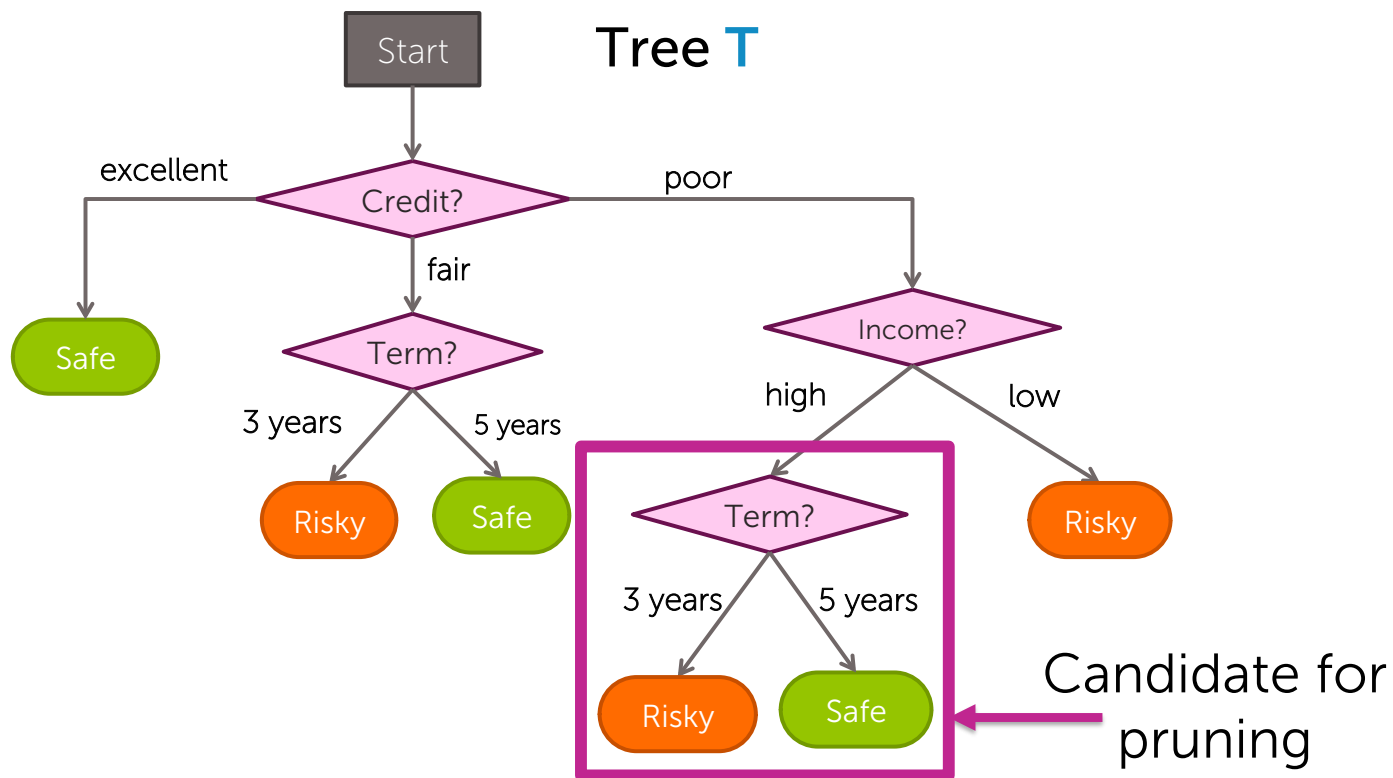


# Tree pruning algorithm

# Pruning Intuition

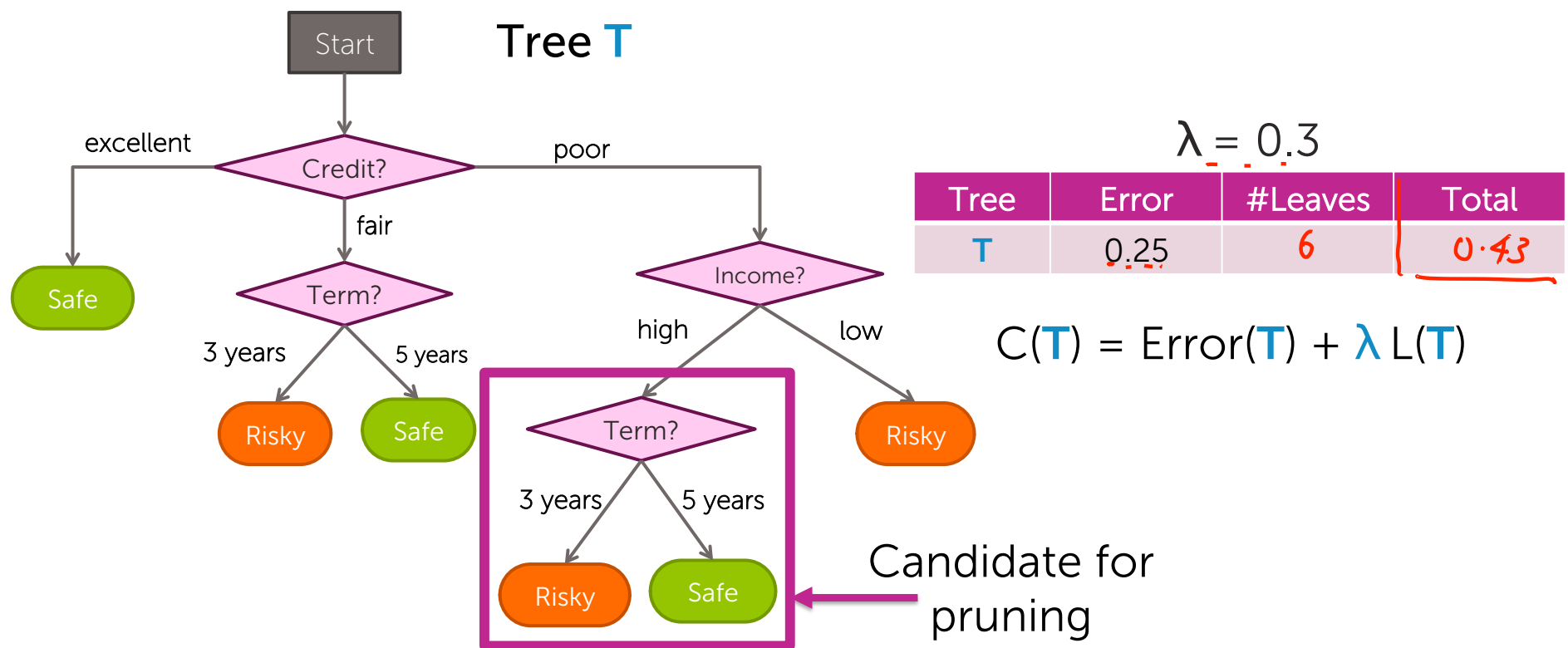


# Step 1: Consider a split

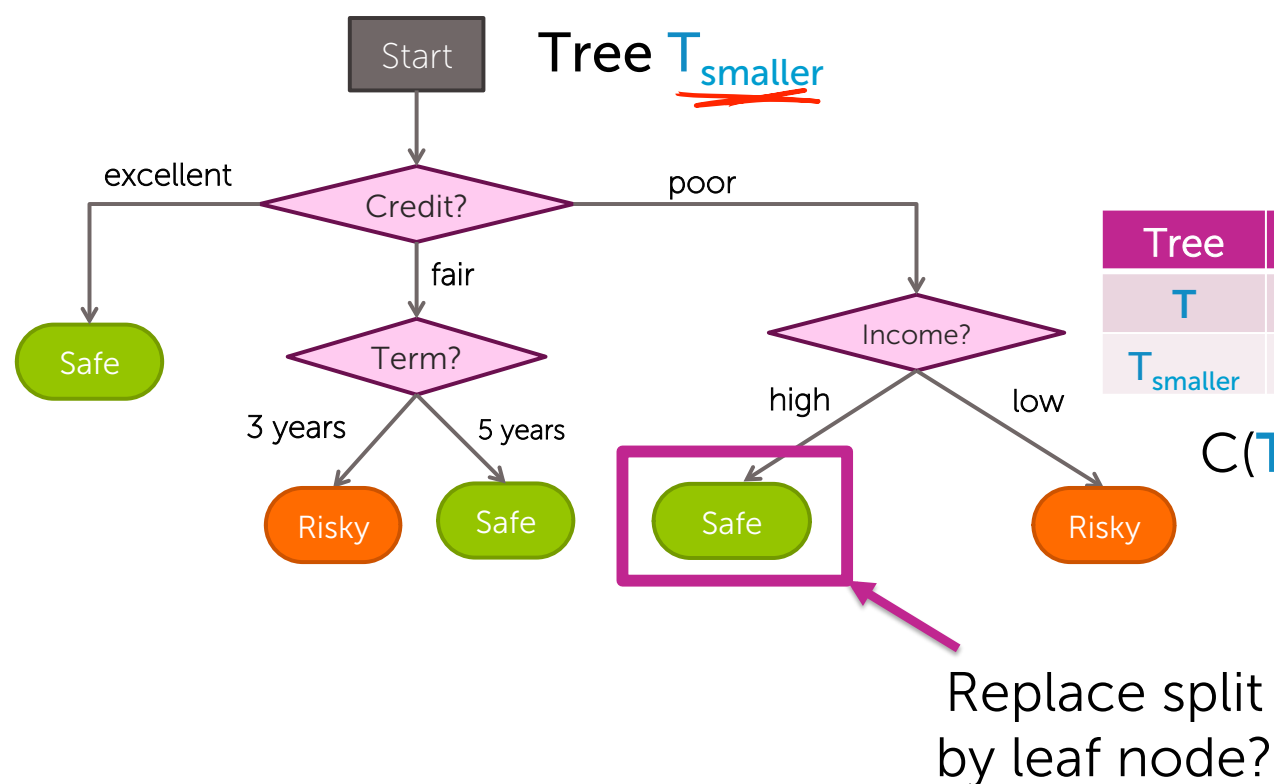




## Step 2: Compute total cost $C(\mathbf{T})$ of split



## Step 2: “Undo” the splits on $T_{\text{smaller}}$

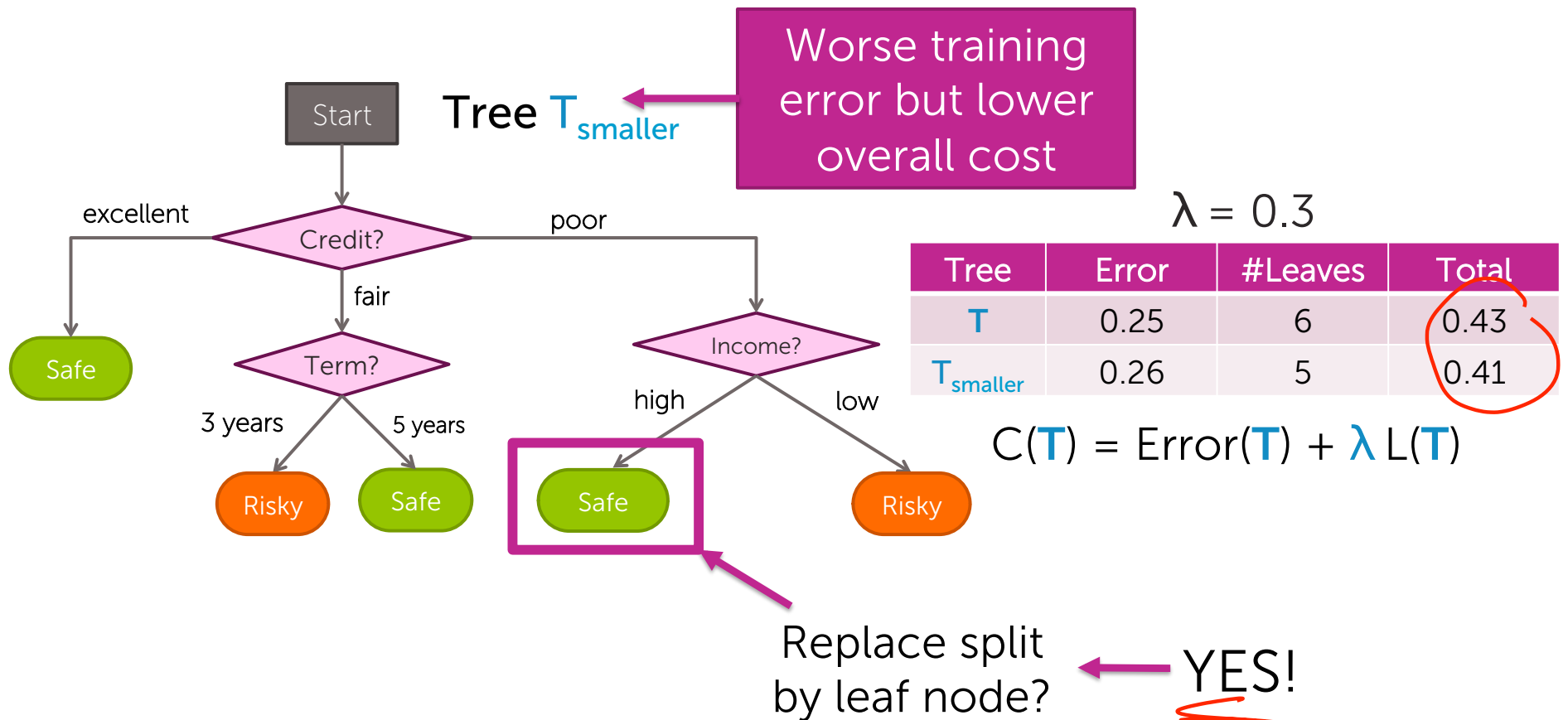


$\lambda = 0.3$

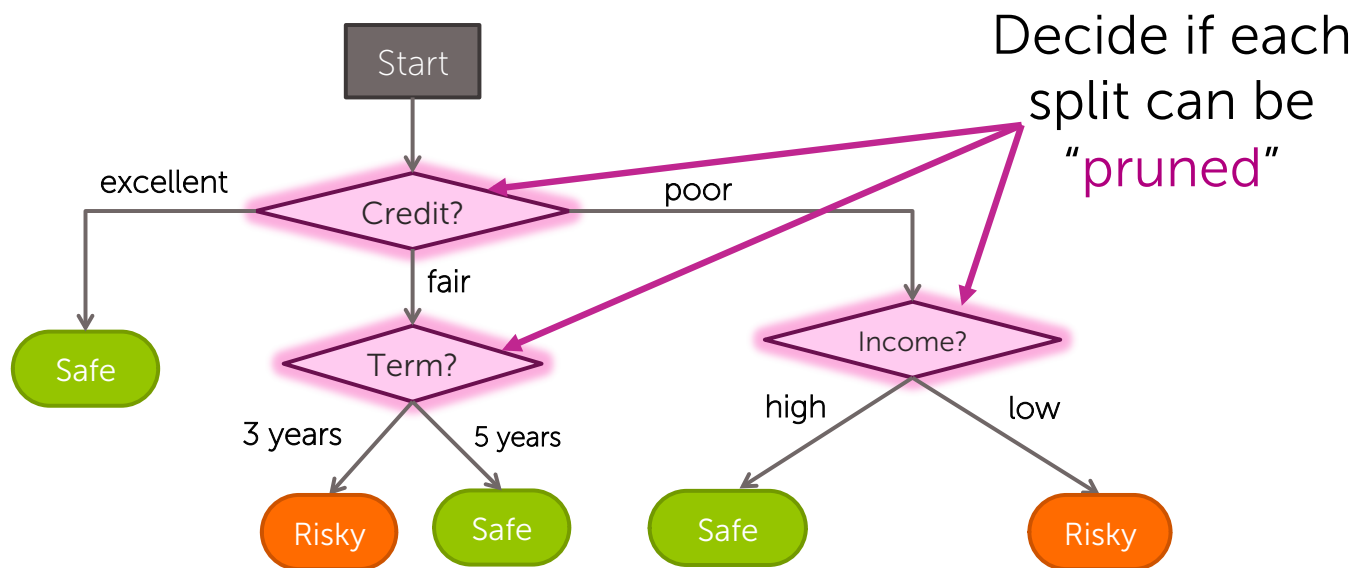
Tree	Error	#Leaves	Total
$T$	0.25	6	0.43
$T_{\text{smaller}}$	0.26	5	0.41

$$C(T) = \text{Error}(T) + \lambda L(T)$$

Prune if total cost is lower:  $C(T_{\text{smaller}}) \leq C(T)$



## Step 5: Repeat Steps 1-4 for every split



# Decision tree pruning algorithm

- Start at bottom of tree  $T$  and traverse up, apply *prune\_split* to each decision node  $M$
- *prune\_split*( $T, M$ ):
  1. Compute total cost of tree  $T$  using
$$C(T) = \text{Error}(T) + \lambda L(T)$$
  2. Let  $T_{\text{smaller}}$  be tree after pruning subtree below  $M$
  3. Compute total cost complexity of  $T_{\text{smaller}}$ 
$$C(T_{\text{smaller}}) = \text{Error}(T_{\text{smaller}}) + \lambda L(T_{\text{smaller}})$$
  4. If  $C(T_{\text{smaller}}) < C(T)$ , prune to  $T_{\text{smaller}}$



# Summary of overfitting in decision trees



# What you can do now...

- Identify when overfitting in decision trees
- Prevent overfitting with early stopping
  - Limit tree depth
  - Do not consider splits that do not reduce classification error
  - Do not split intermediate nodes with only few points
- Prevent overfitting by pruning complex trees
  - Use a total cost formula that balances classification error and tree complexity
  - Use total cost to merge potentially complex trees into simpler ones

---

# Thank you to Dr. Krishna Sridhar



Dr. Krishna Sridhar  
Staff Data Scientist, Dato, Inc.