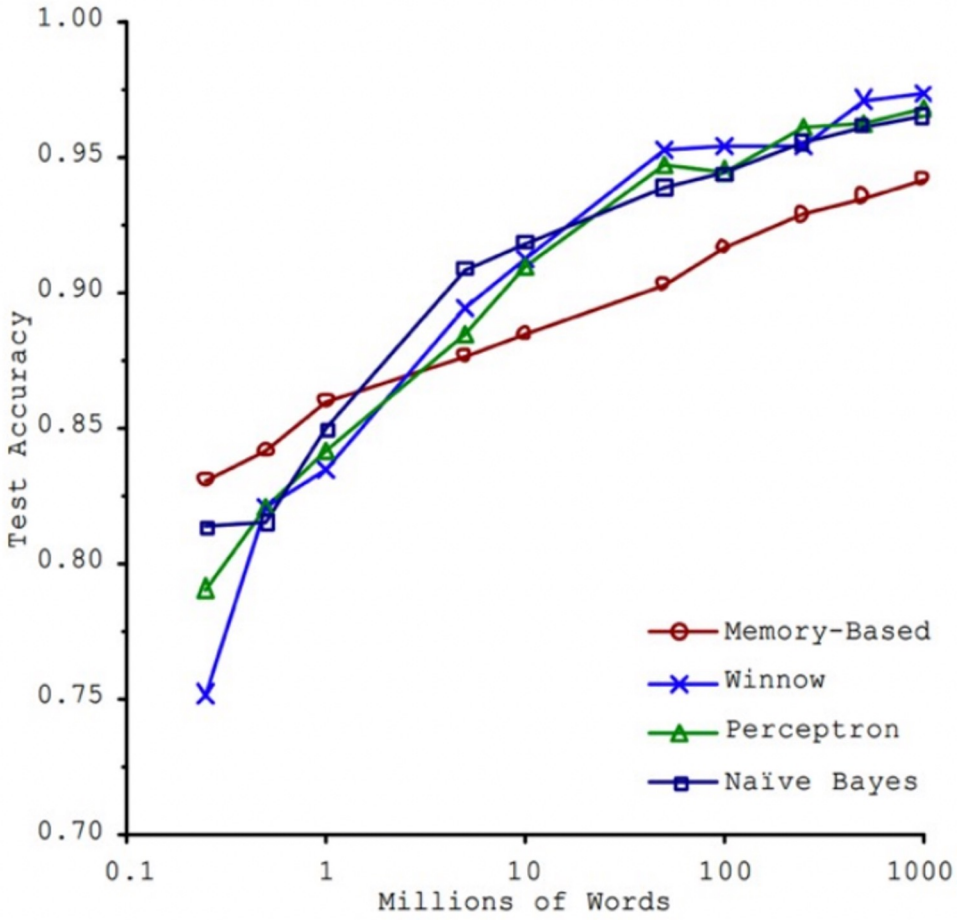
Decision Trees

Why Large-Scale ML?

- Brawn or Brains?
  - In 2001, Microsoft researchers ran a test to evaluate 4 of different approaches to ML-based language translation
- Findings:
  - Size of the dataset used to train the model mattered more than the model itself
  - As the dataset grew large, performance difference between the models became small

Performance of different language translation models



Why Large-Scale ML?

The Unreasonable Effectiveness of Data

- In 2017, Google revisited the same type of experiment with the latest Deep Learning models in computer vision

Findings:

- Performance increases logarithmically based on volume of training data
- Complexity of modern ML models (i.e., deep neural nets) allows for even further performance gains

Why worry about Non – Deep Models?

- Classical tasks in NLP and Vision are getting commoditized (you take pretrained model and fine tune it), but there are many other unique ML tasks.
- Deep models are often hard to scale and require lots and lots of data. Traditional models allow you to encode prior knowledge better and give you more control.
- Personally, if I am working on a well understood problem I'd use deep learning, but if I am the first person to work on a new problem/classifier I'd use techniques we'll discuss here.

Predicting potential loan defaults

Credit history explained

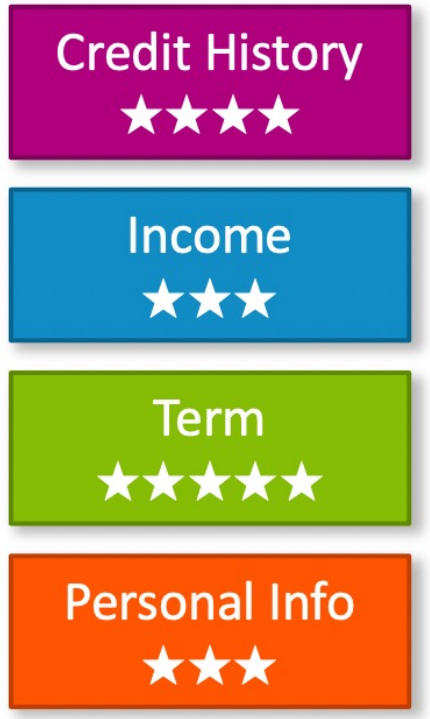
Did I pay previous loans on time?

Example: excellent, good, or fair

Income

What's my income?

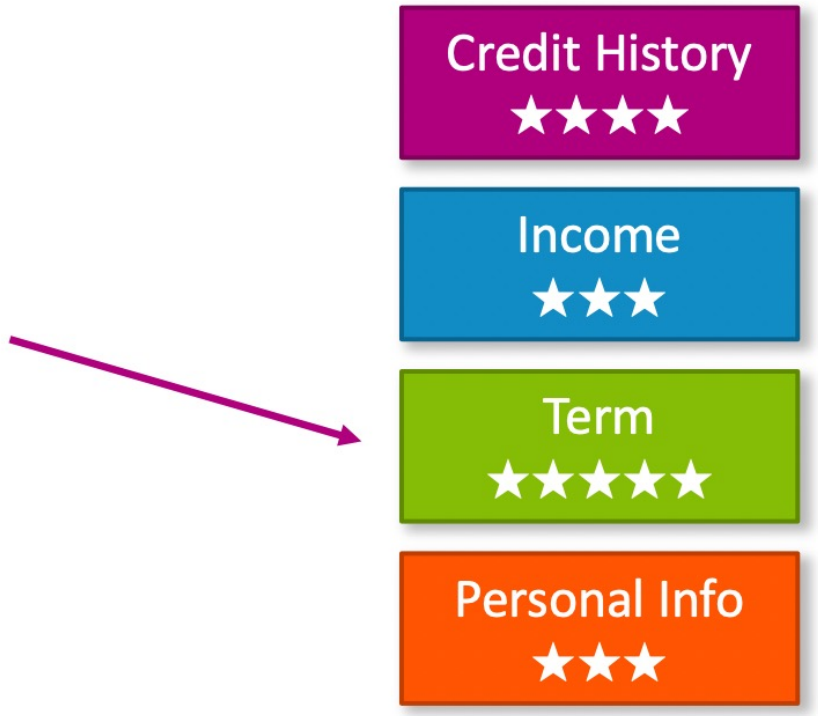
Example:
$80K per year



Loan terms

How soon do I need to pay the loan?

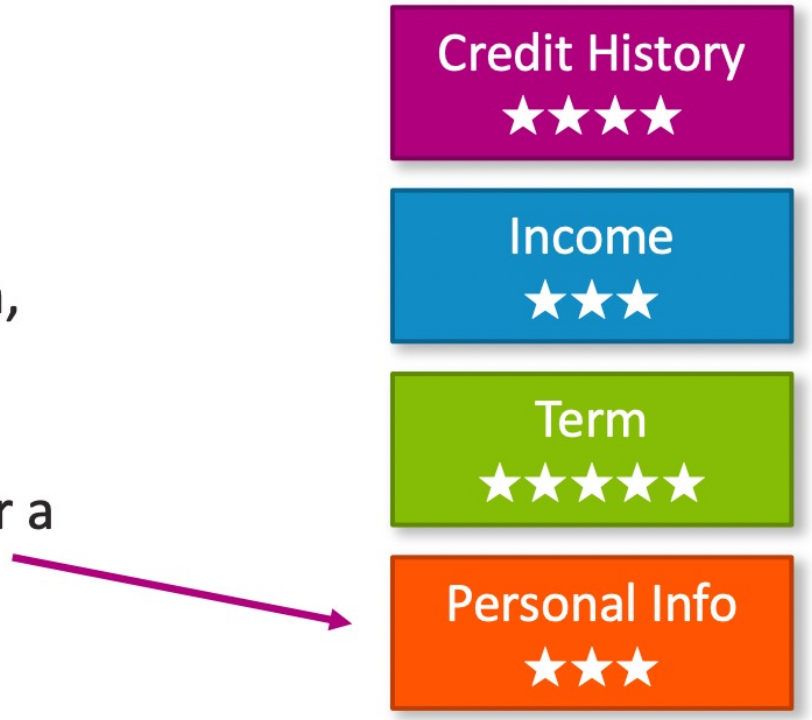
Example: 3 years, 5 years,…



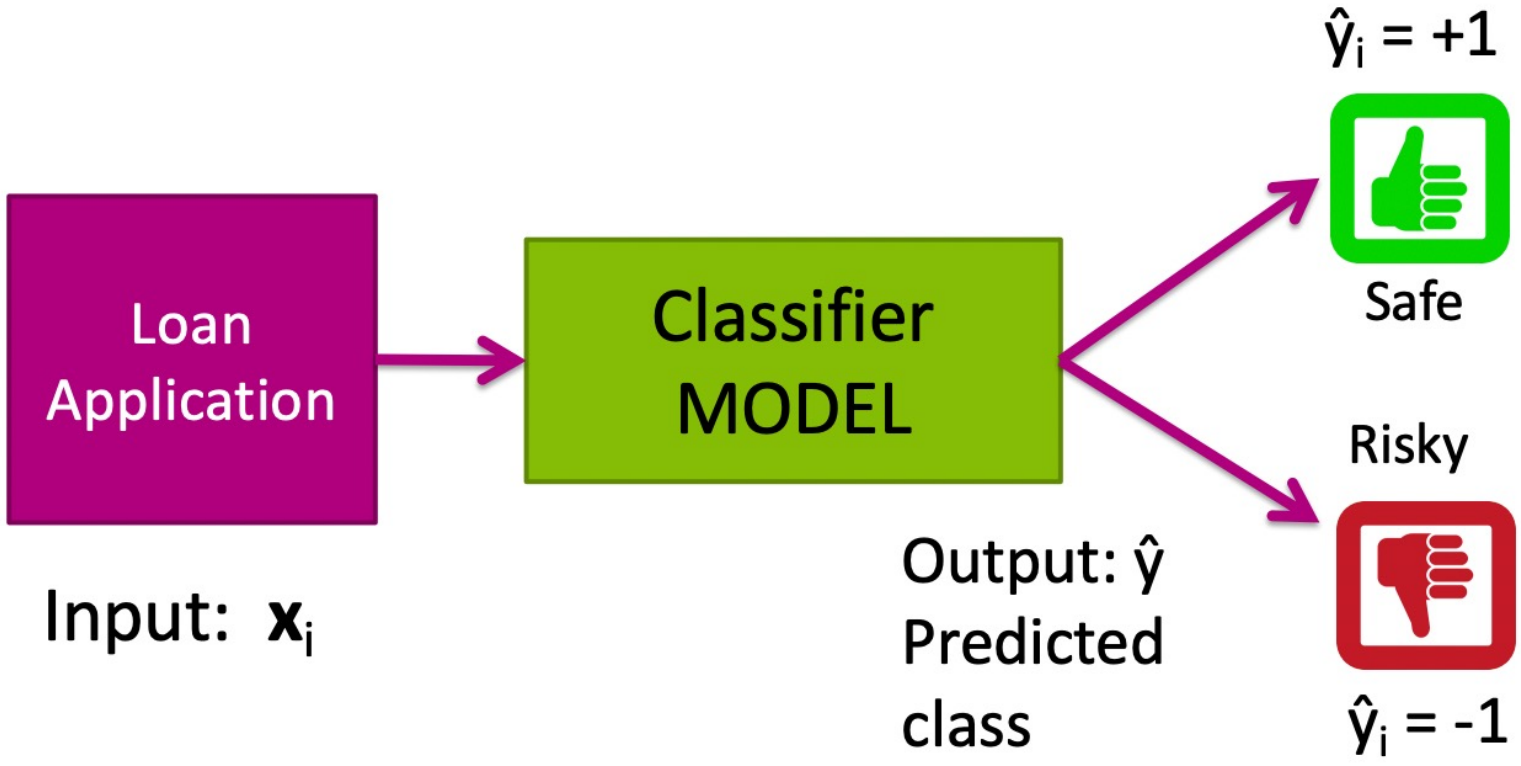
Personal information

Age, reason for the loan, marital status,…

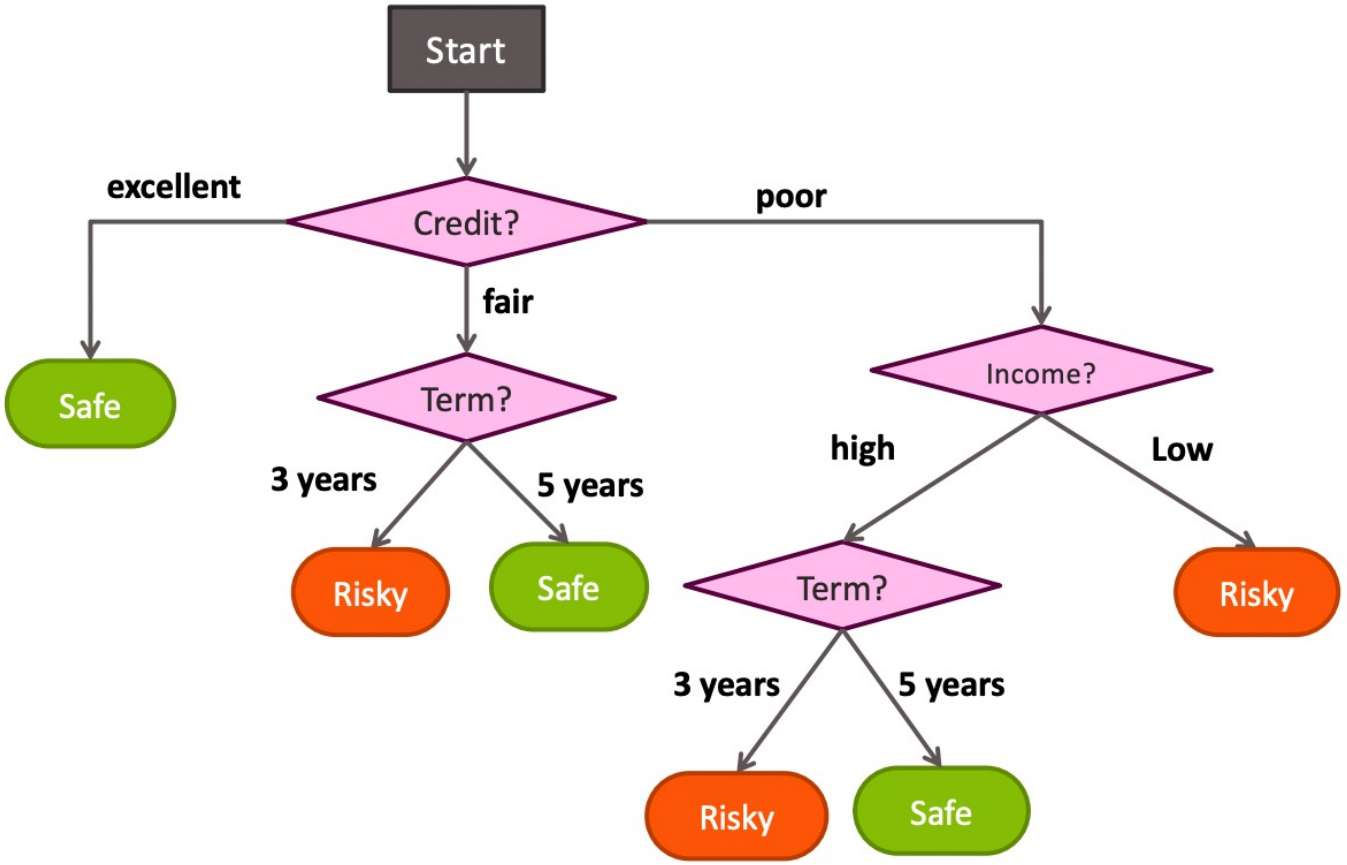
Example: Home loan for a married couple



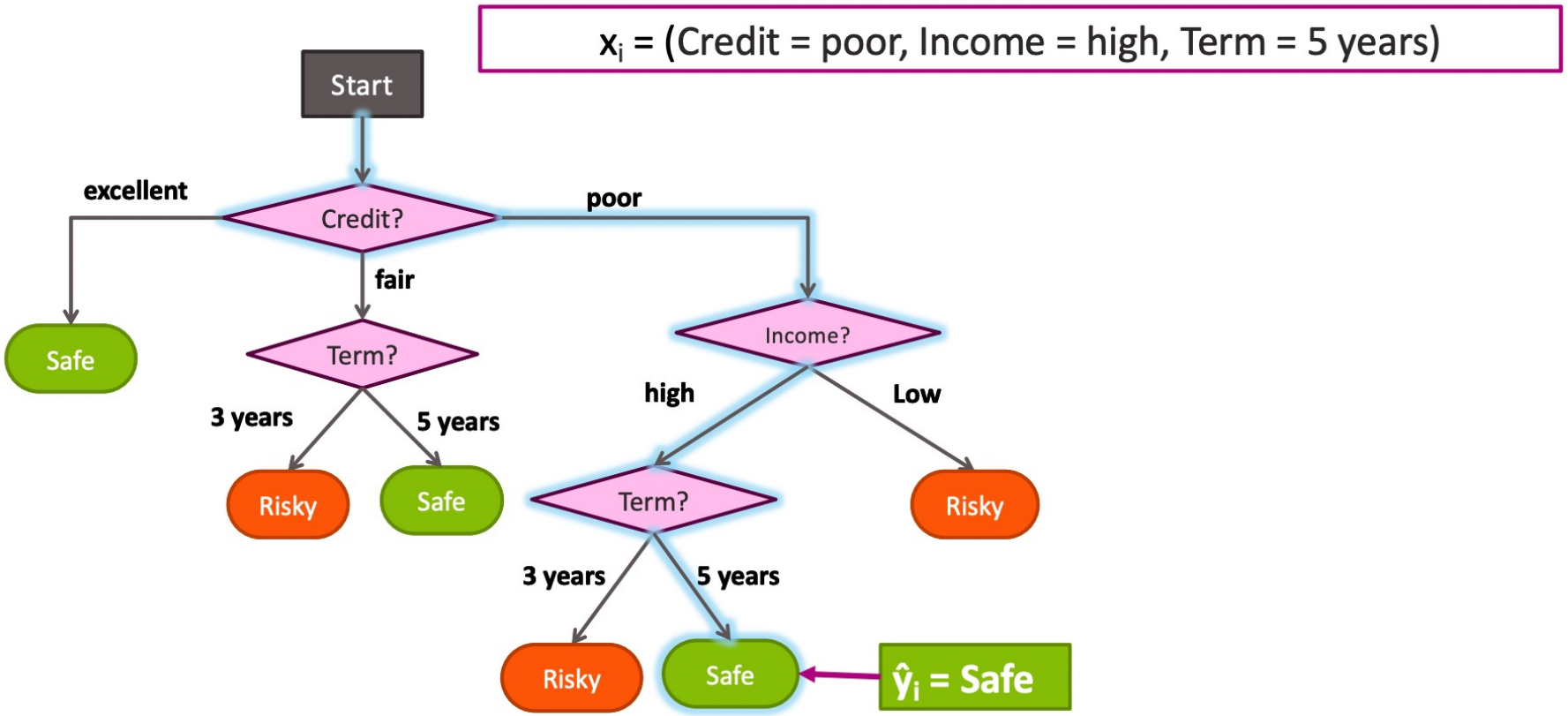
Classifier review



This module … decision trees



Scoring a loan application



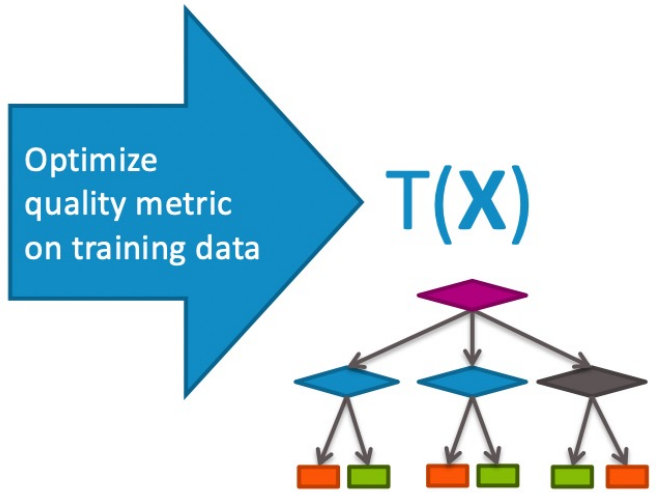
Preface: Decision Trees

- Decision trees are part of ML since 1980s
  - Introduced by Leo Breiman in 1984
  - Notable algorithms: ID3, C4.5
- More recent innovations include:
  - Boosted decision trees (gradient boosted DT)
  - Random forest
- Even though DTs are old, hand-engineered and heuristic, they are a method of choice for tabular data and for Kaggle competitions!

Decision tree learning problem

Training data: N observations (x_i, y_i)

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe



Quality metric: Classification error

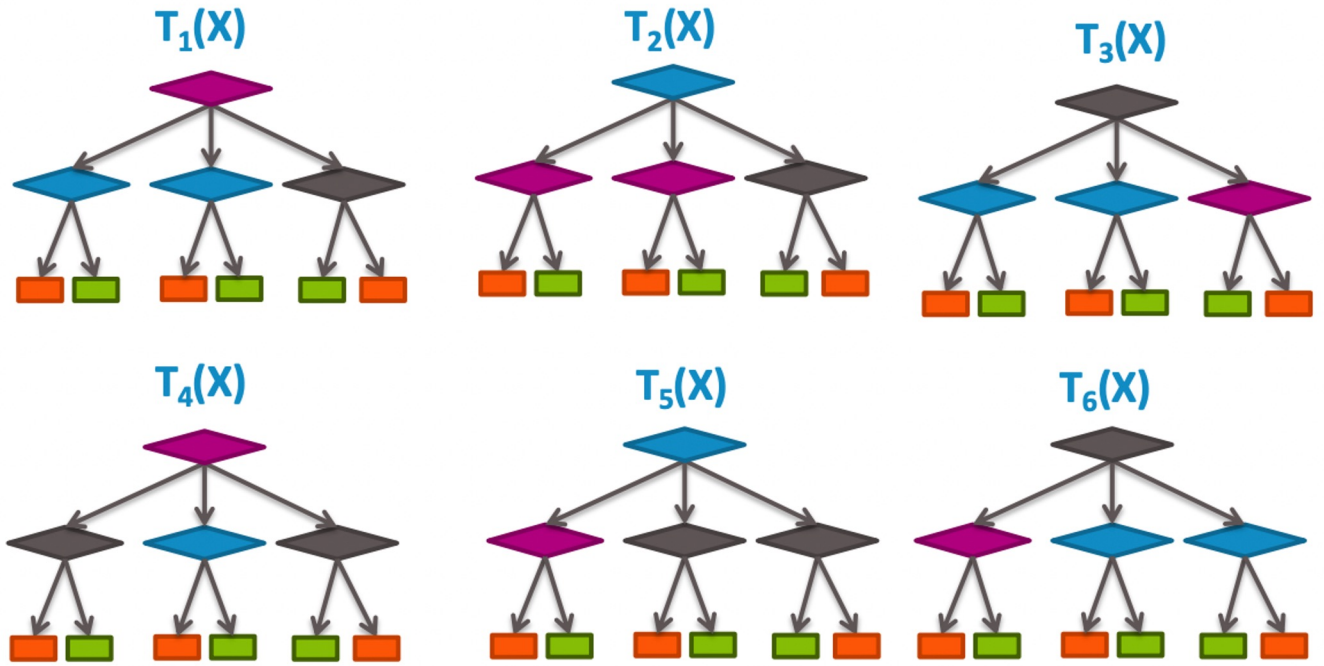
- Error measures fraction of mistakes

$$\text{Error} = \frac{\text{\# incorrect predictions}}{\text{\# examples}}$$

  - Best possible value : 0.0
  - Worst possible value: 1.0

How do we find the best tree?

Exponentially large number of possible trees makes decision tree learning hard!



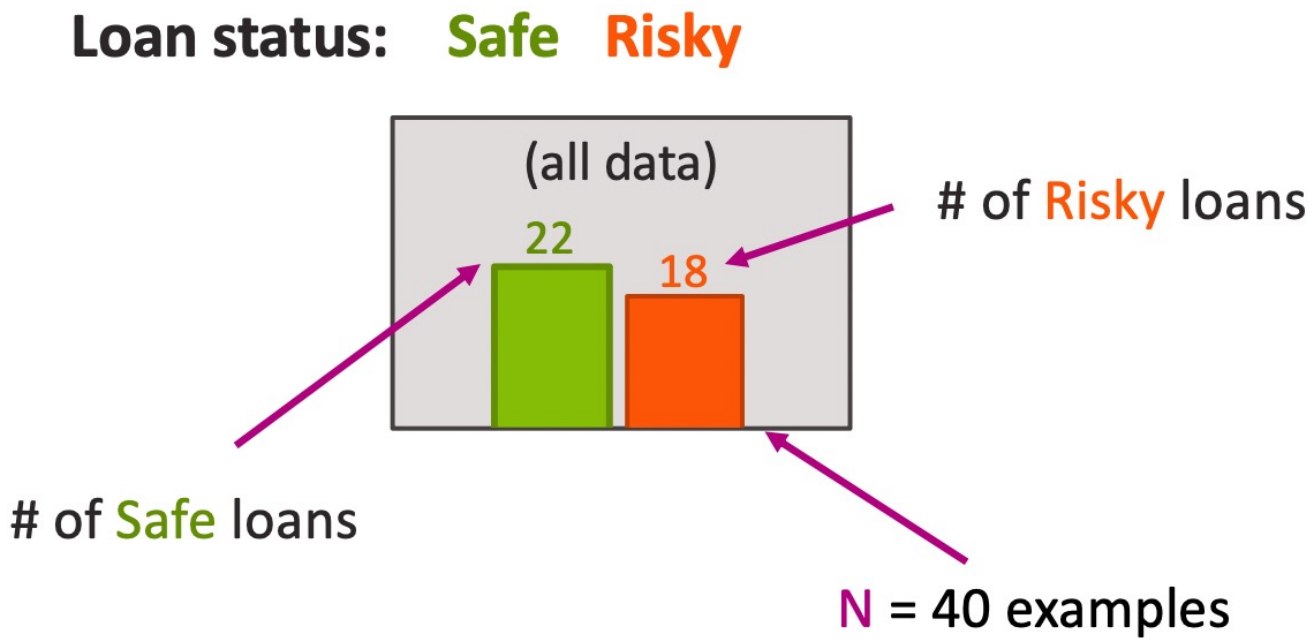
Greedy decision tree learning

Our training data table

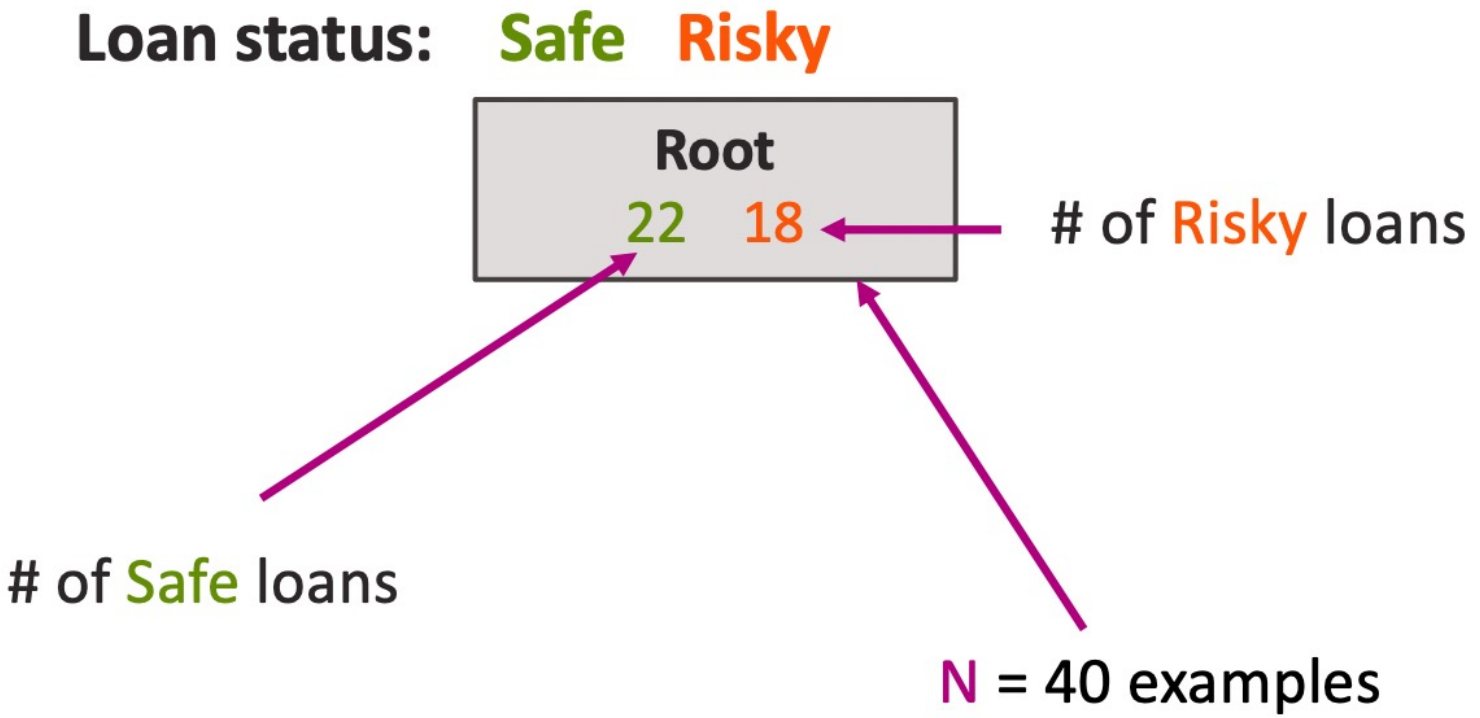
Assume N = 40, 3 features

Credit	Term	Income	y
excellent	3 yrs	high	safe
fair	5 yrs	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

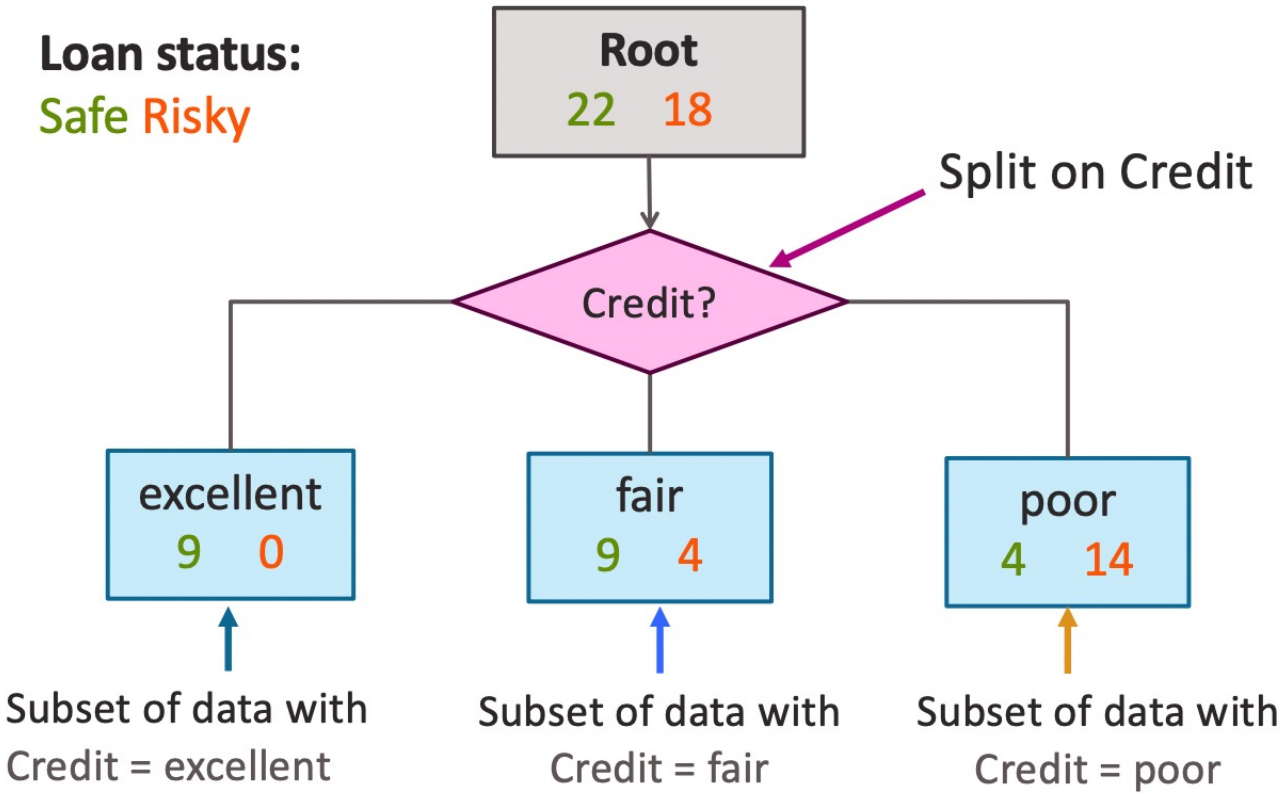
Start with all the data



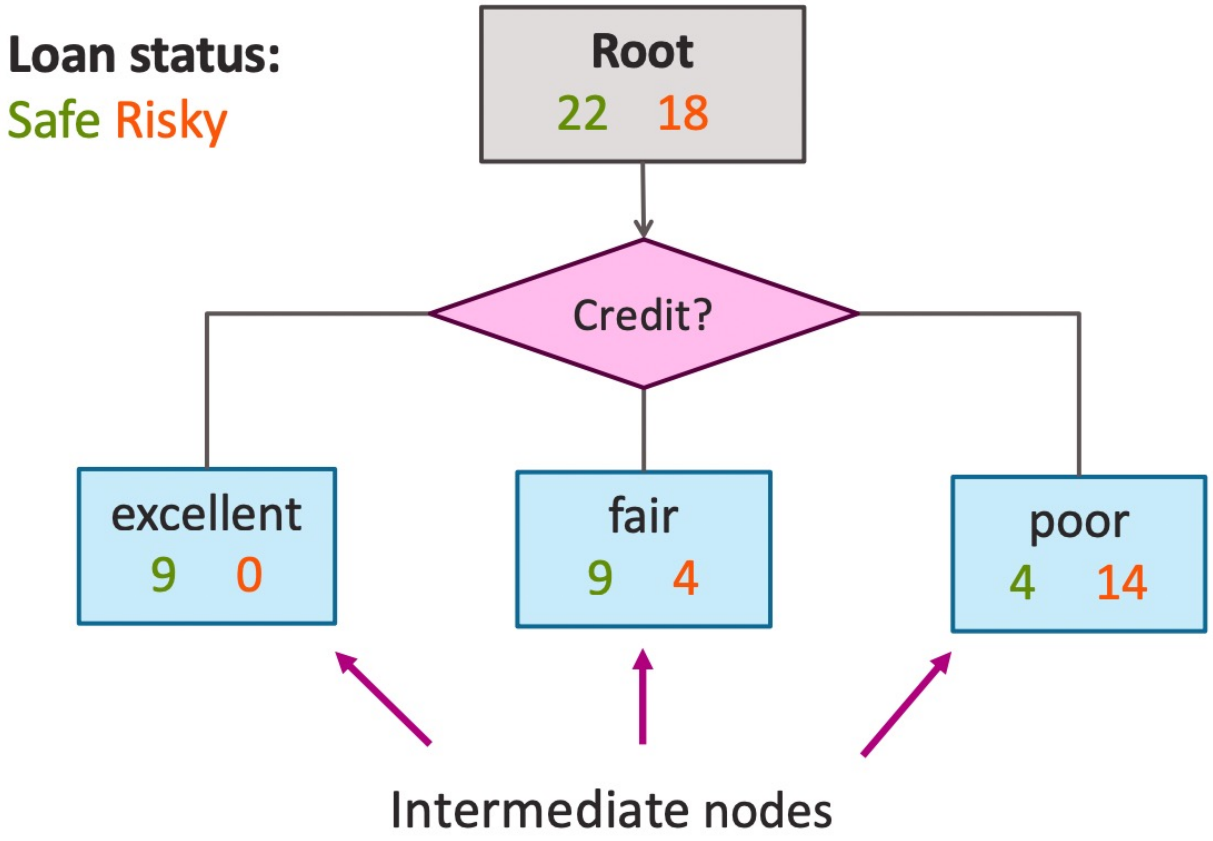
Compact visual notation: Root node



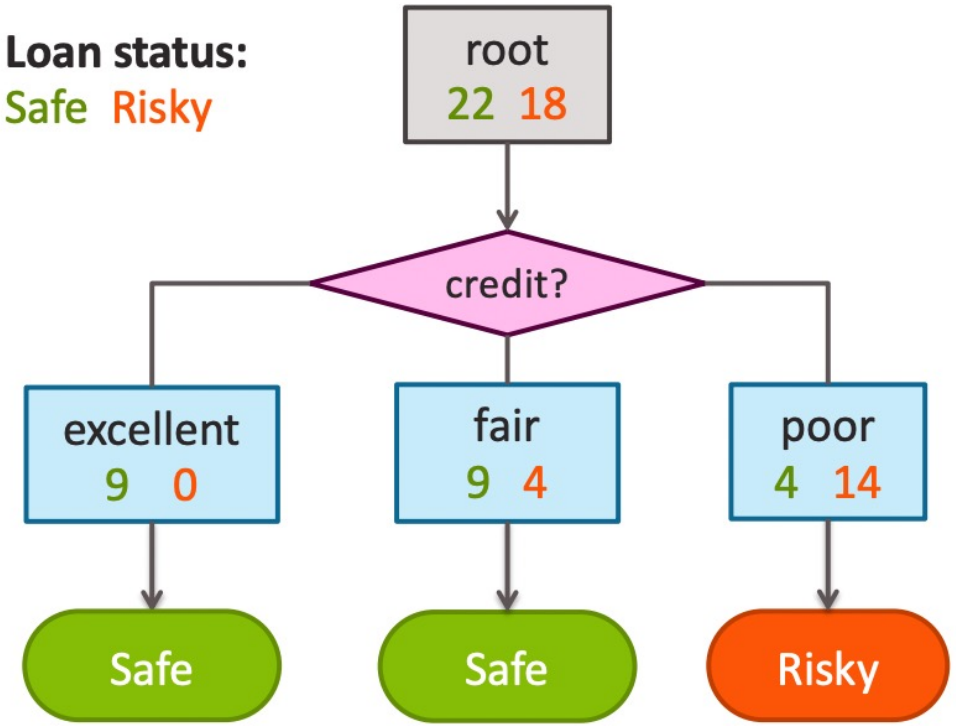
Decision stump: Single level tree



Visual notation: Intermediate nodes

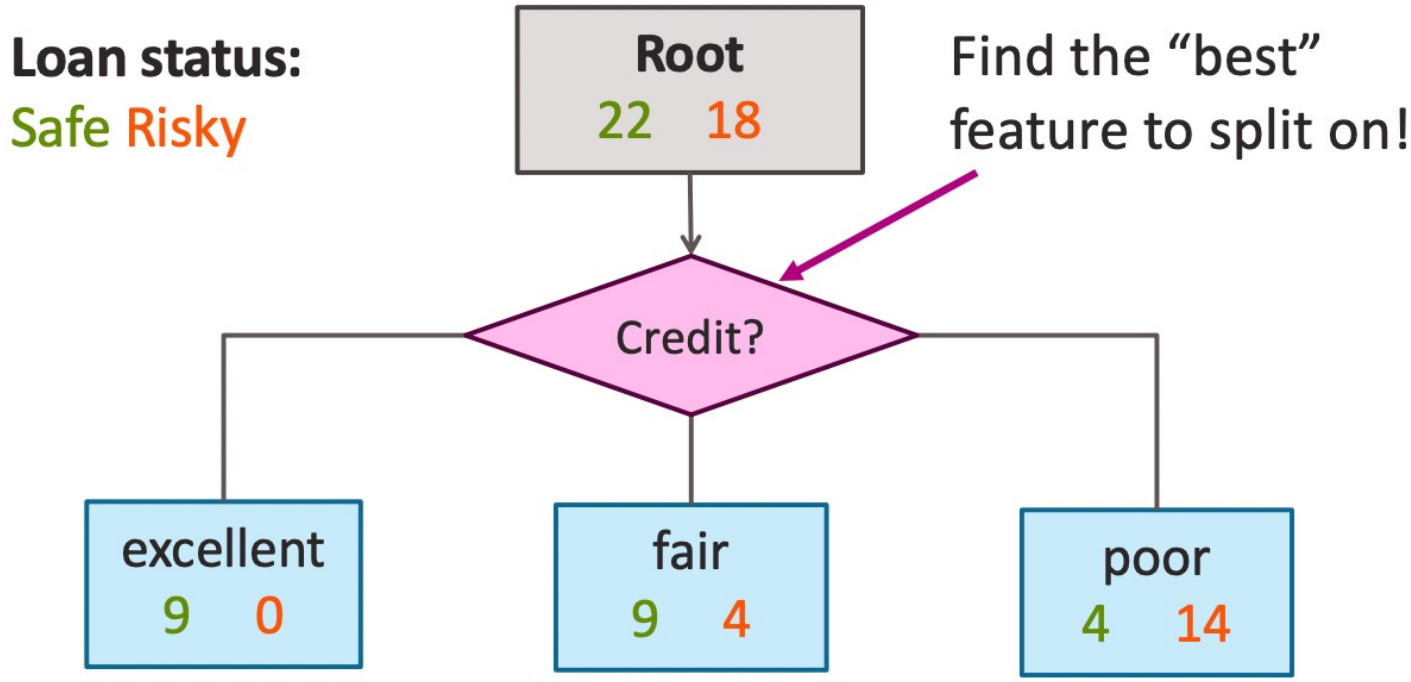


Making predictions with a decision stump

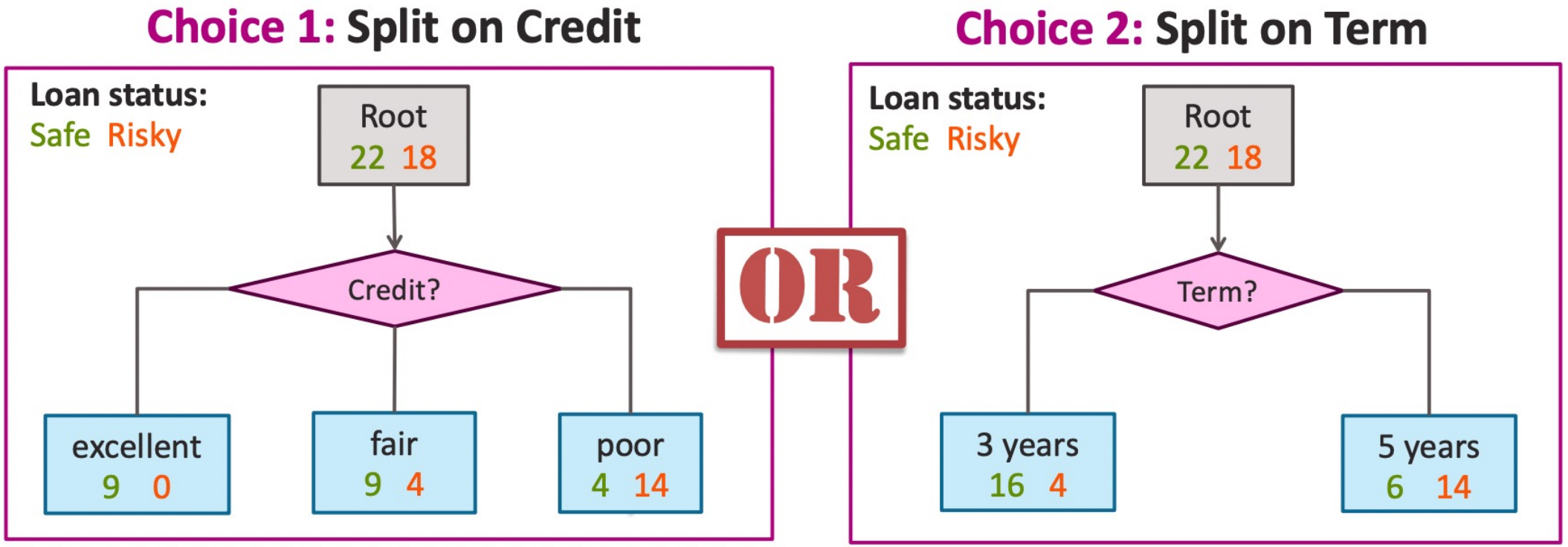


For each intermediate node,
set $\hat{y}$ = majority value

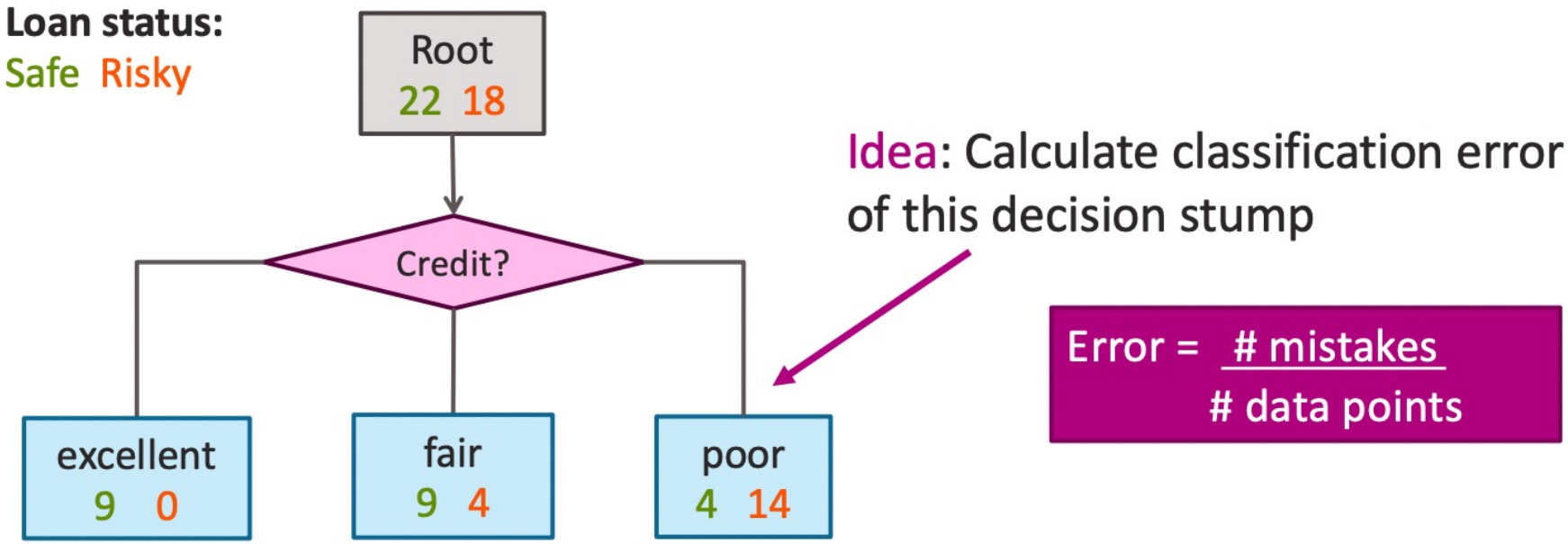
How do we learn a decision stump?



How do we select the best feature?

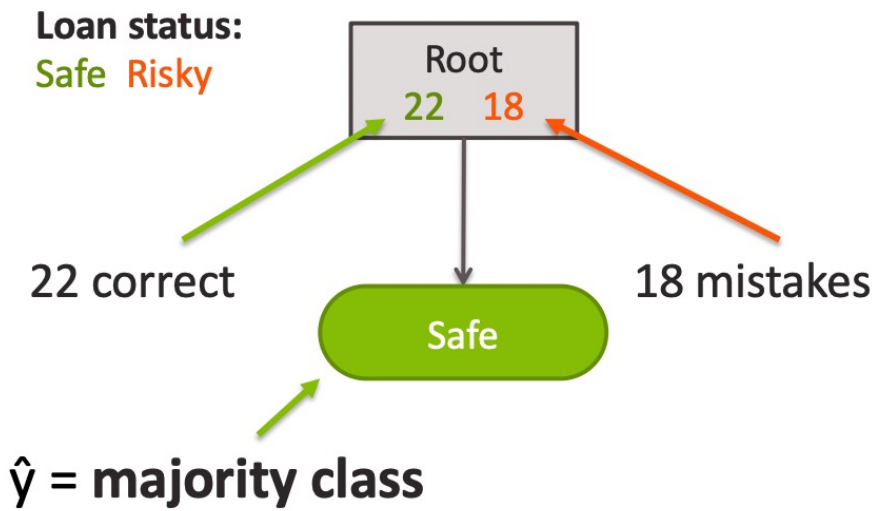


How do we measure effectiveness of a split?



Calculating classification error

- Step 1: $\hat{y}$ = class of majority of data in node
- Step 2: Calculate classification error of predicting $\hat{y}$ for this data

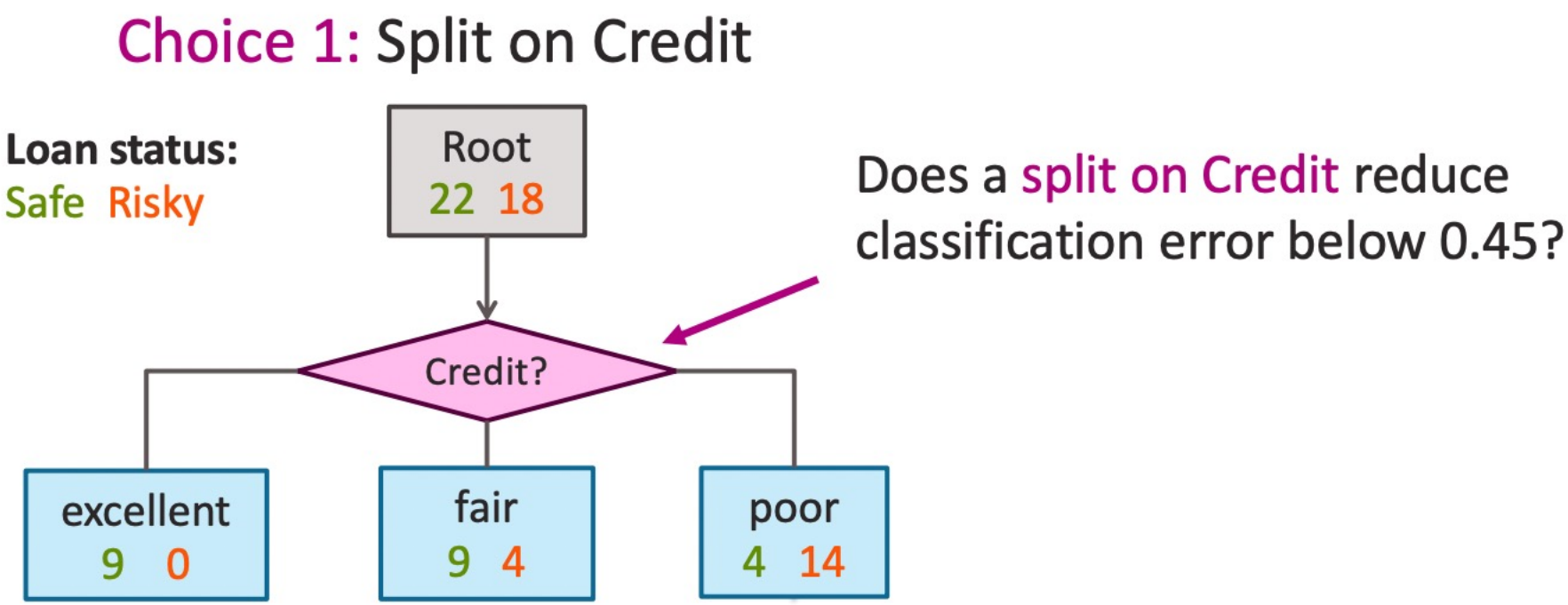


Error = _____

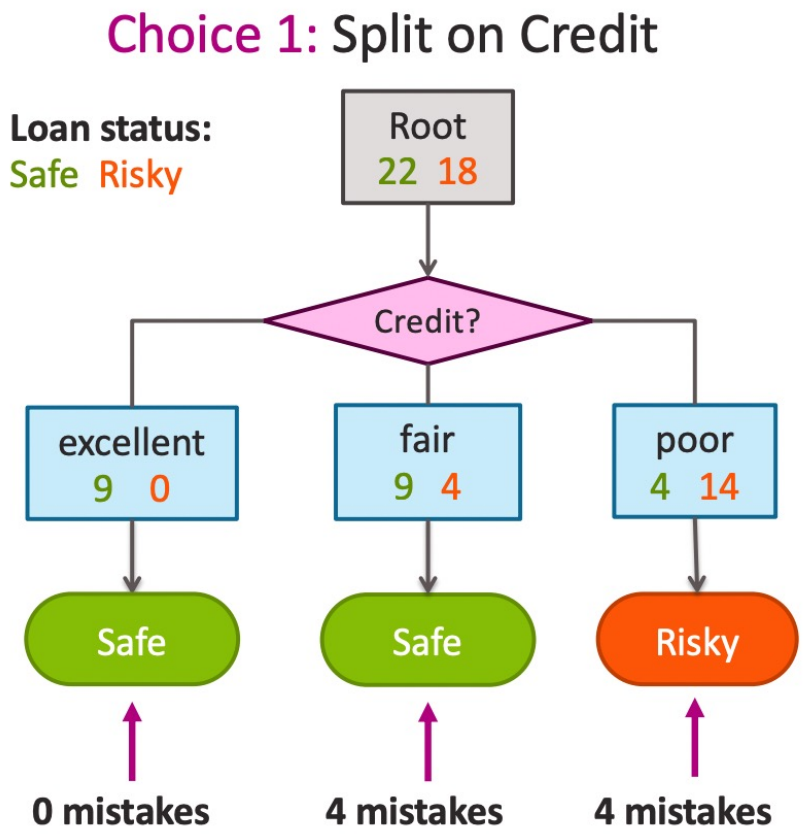
=

Tree	Classification error
(root)	0.45

Choice 1: Split on Credit history?



Split on Credit: Classification error

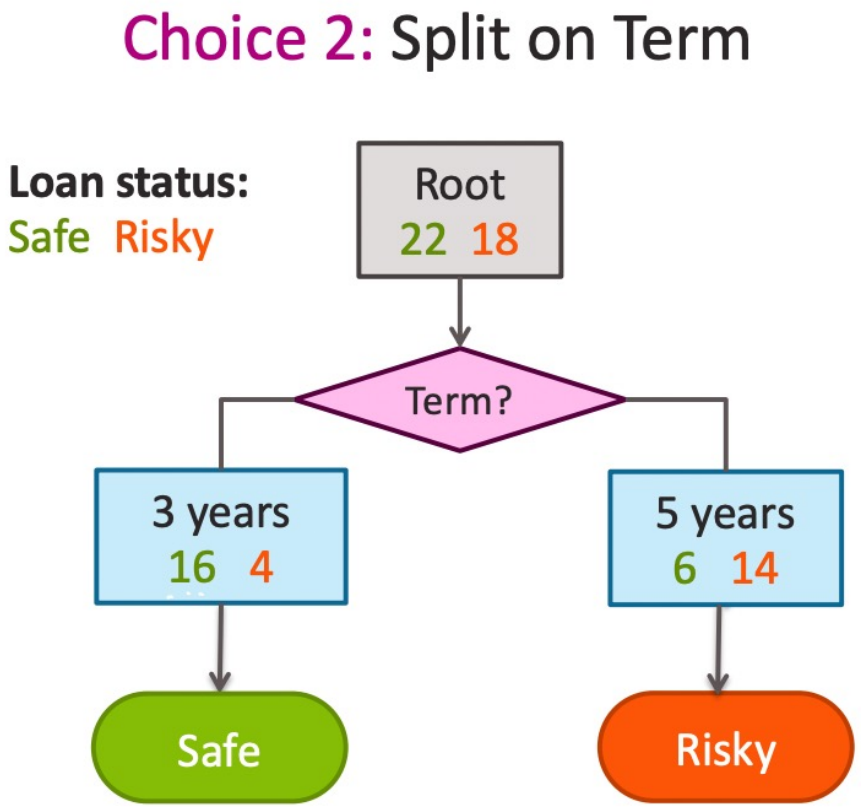


Error = _____

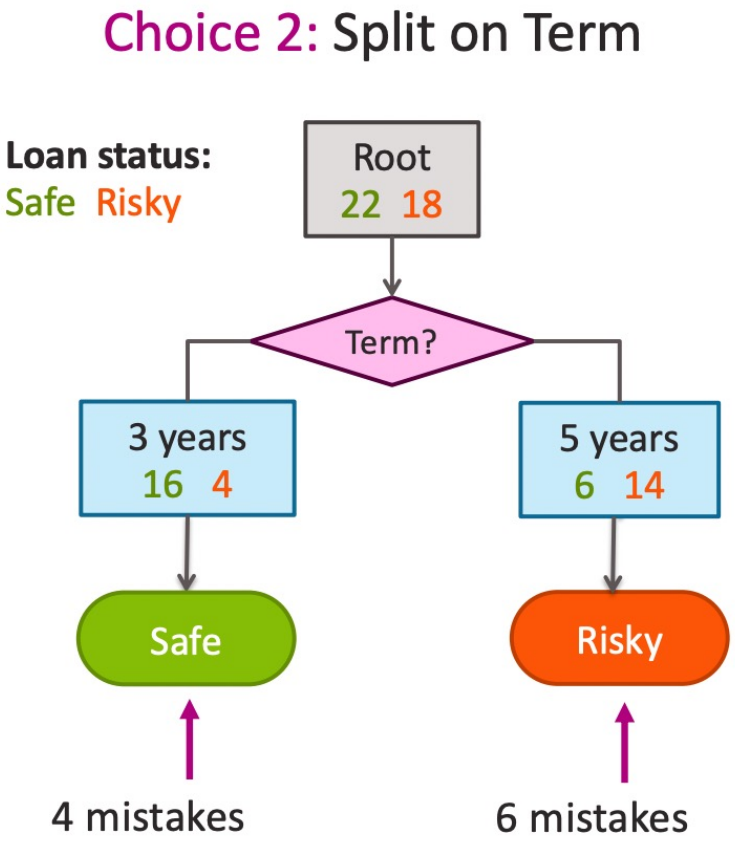
=

Tree	Classification error
(root)	0.45
Split on credit	0.2

Choice 2: Split on Term?



Evaluating the split on Term

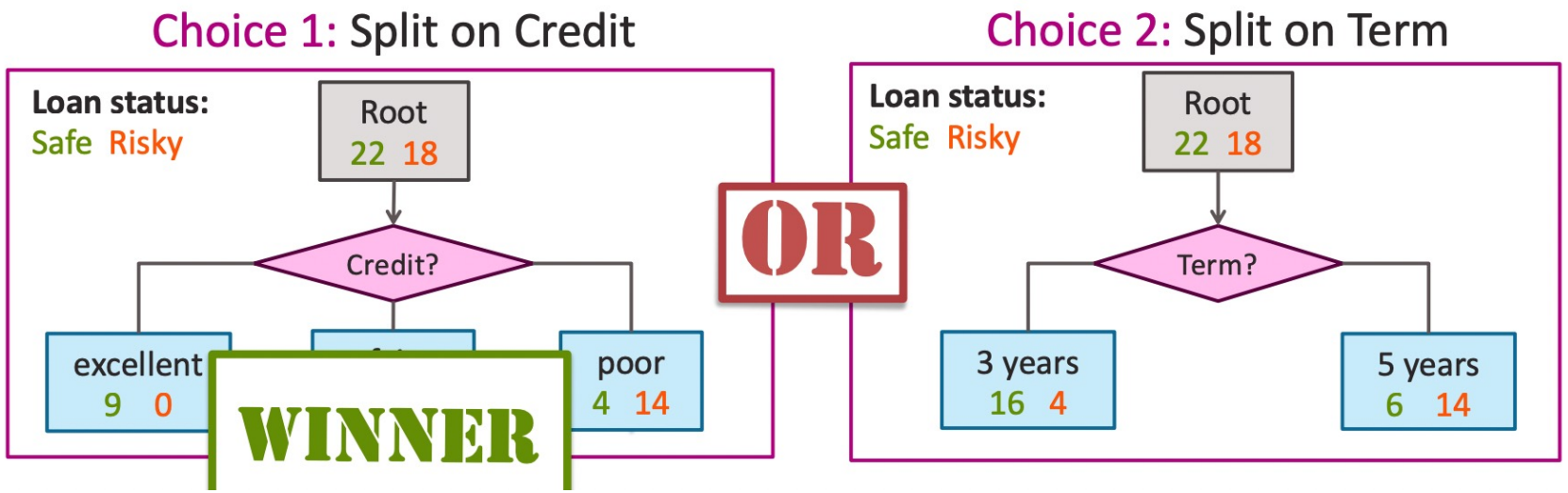


Error = _____

=

Tree	Classification error
(root)	0.45
Split on credit	0.2
Split on term	0.25

Choice 1 vs Choice 2: Comparing split on Credit vs Term



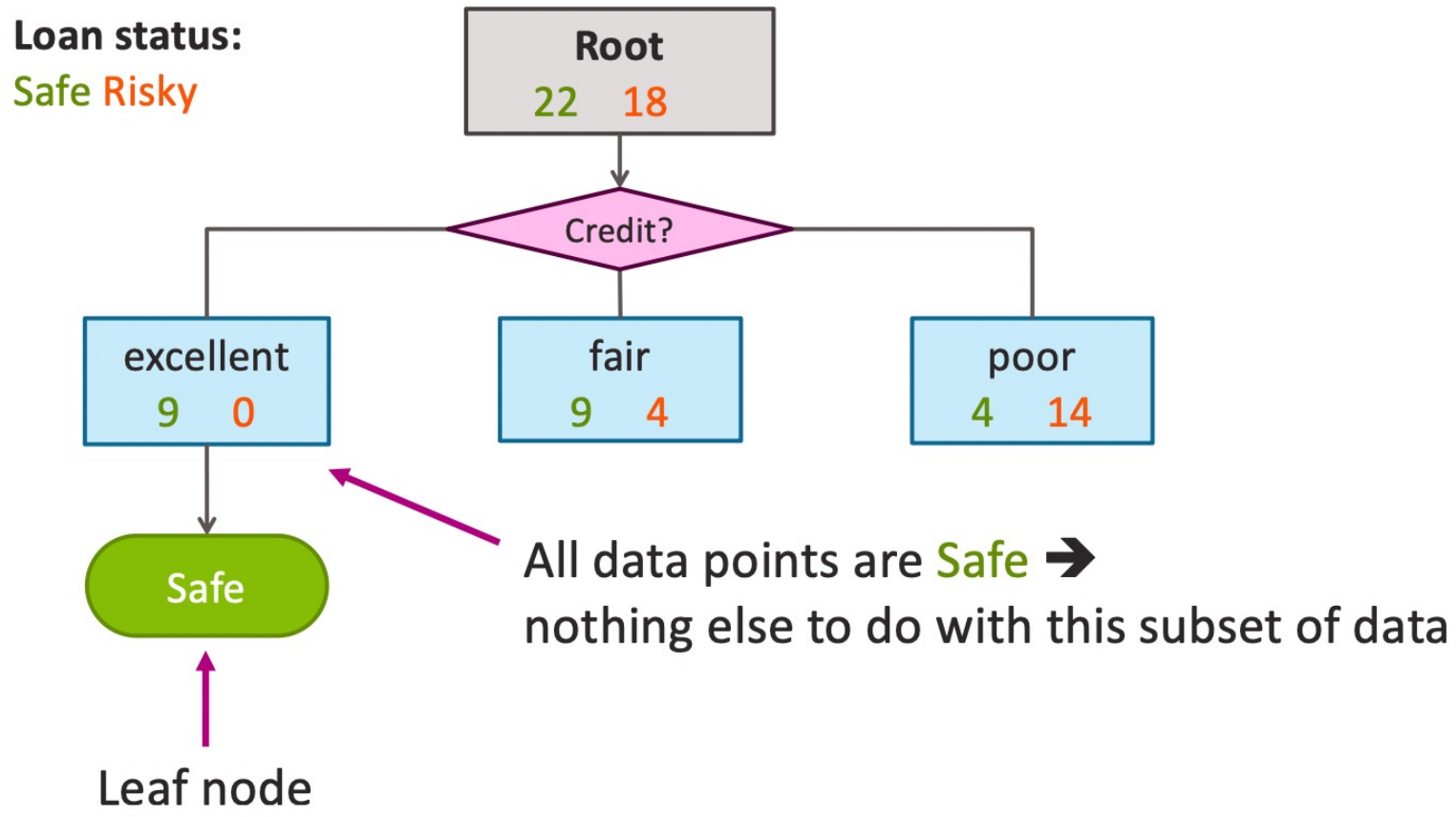
Tree	Classification error
(root)	0.45
split on credit	0.2
split on loan term	0.25

Feature split selection algorithm

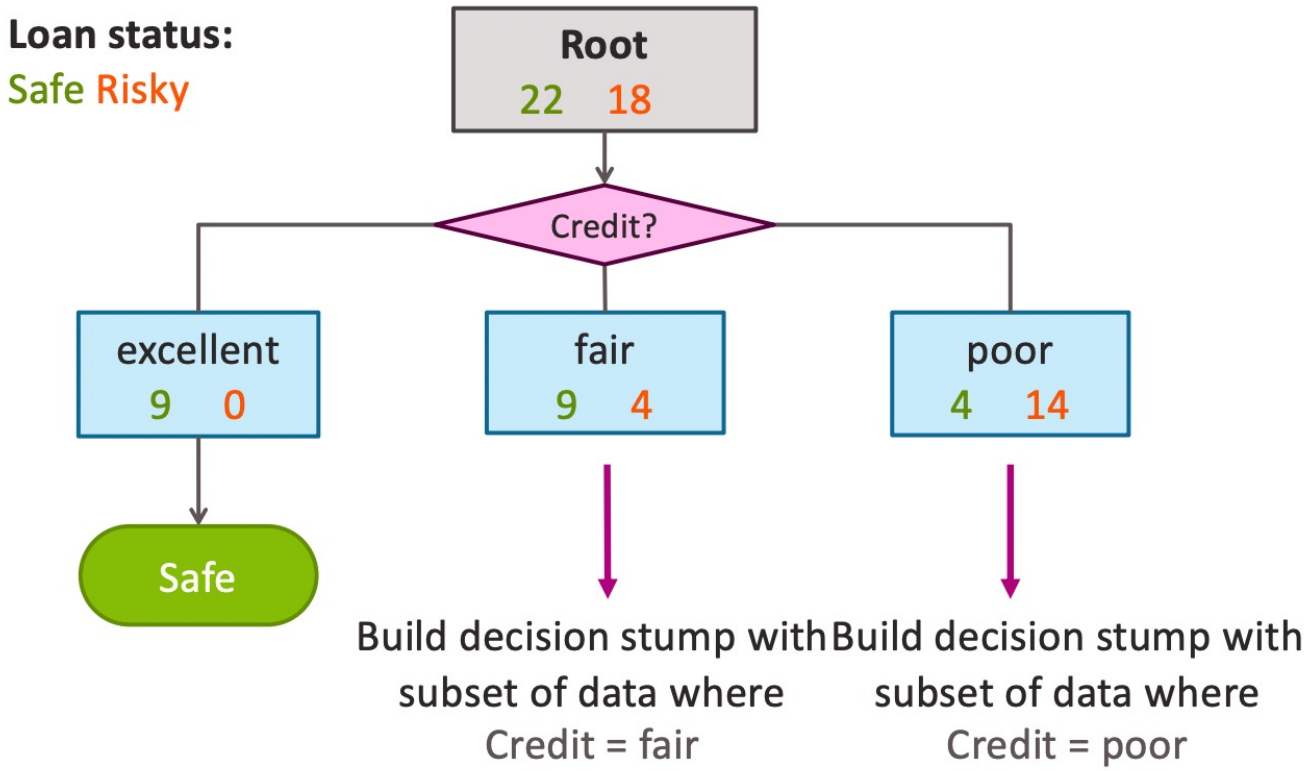
- Given a subset of data M (a node in a tree)
- For each feature hi (x):
    1. Split data of M according to feature hi (x)
    2. Compute classification error of split
- Chose feature h*(x) with lowest classification error

Recursion & Stopping conditions

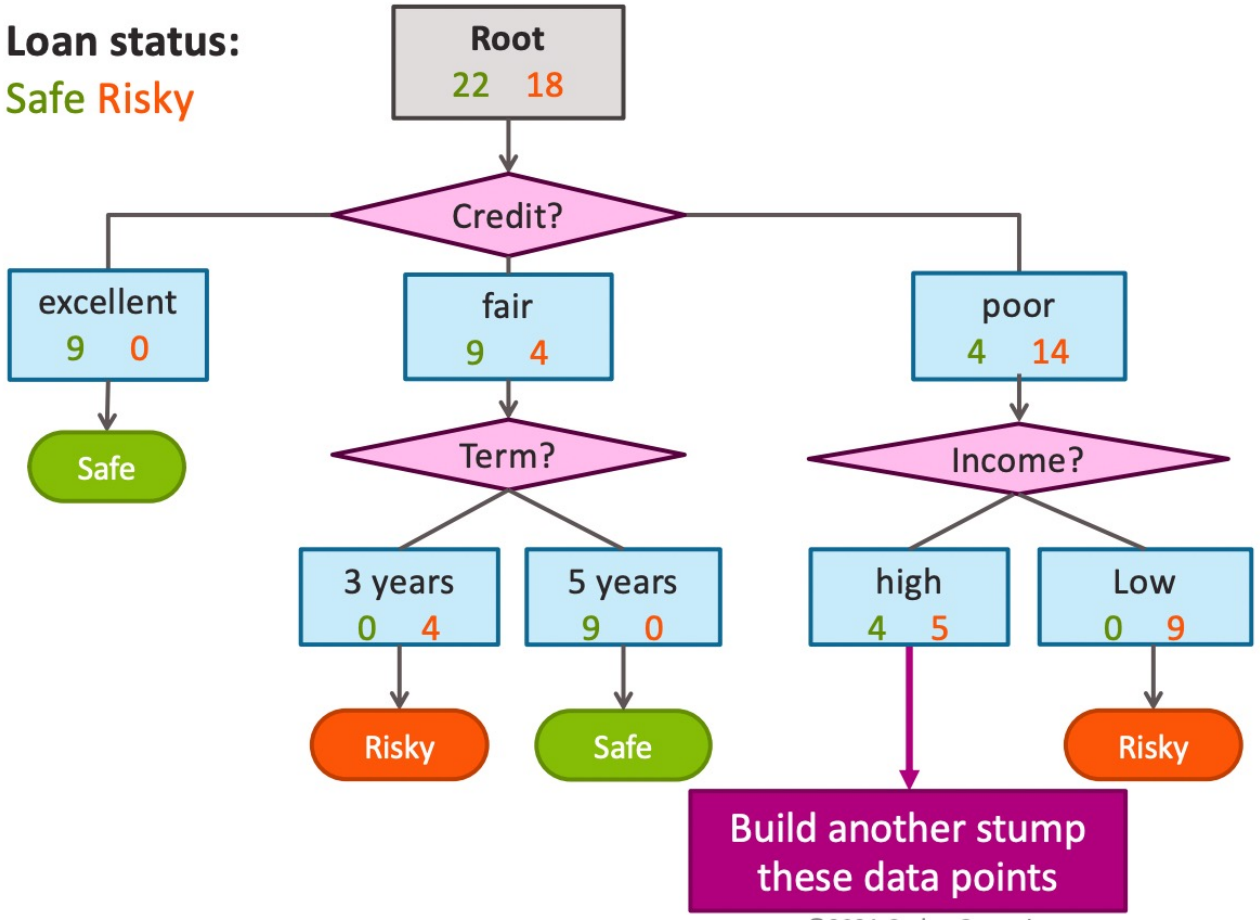
We've learned a decision stump, what next?



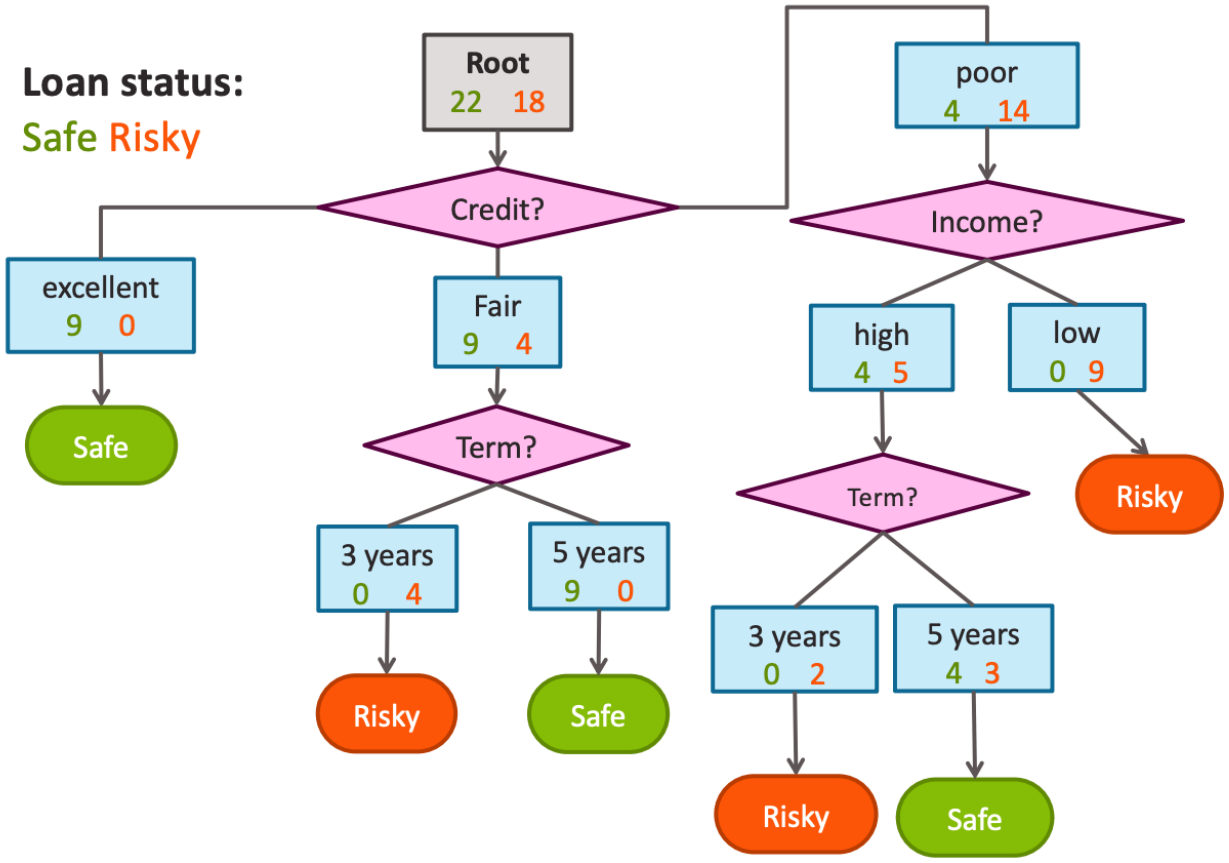
Tree learning = Recursive stump learning



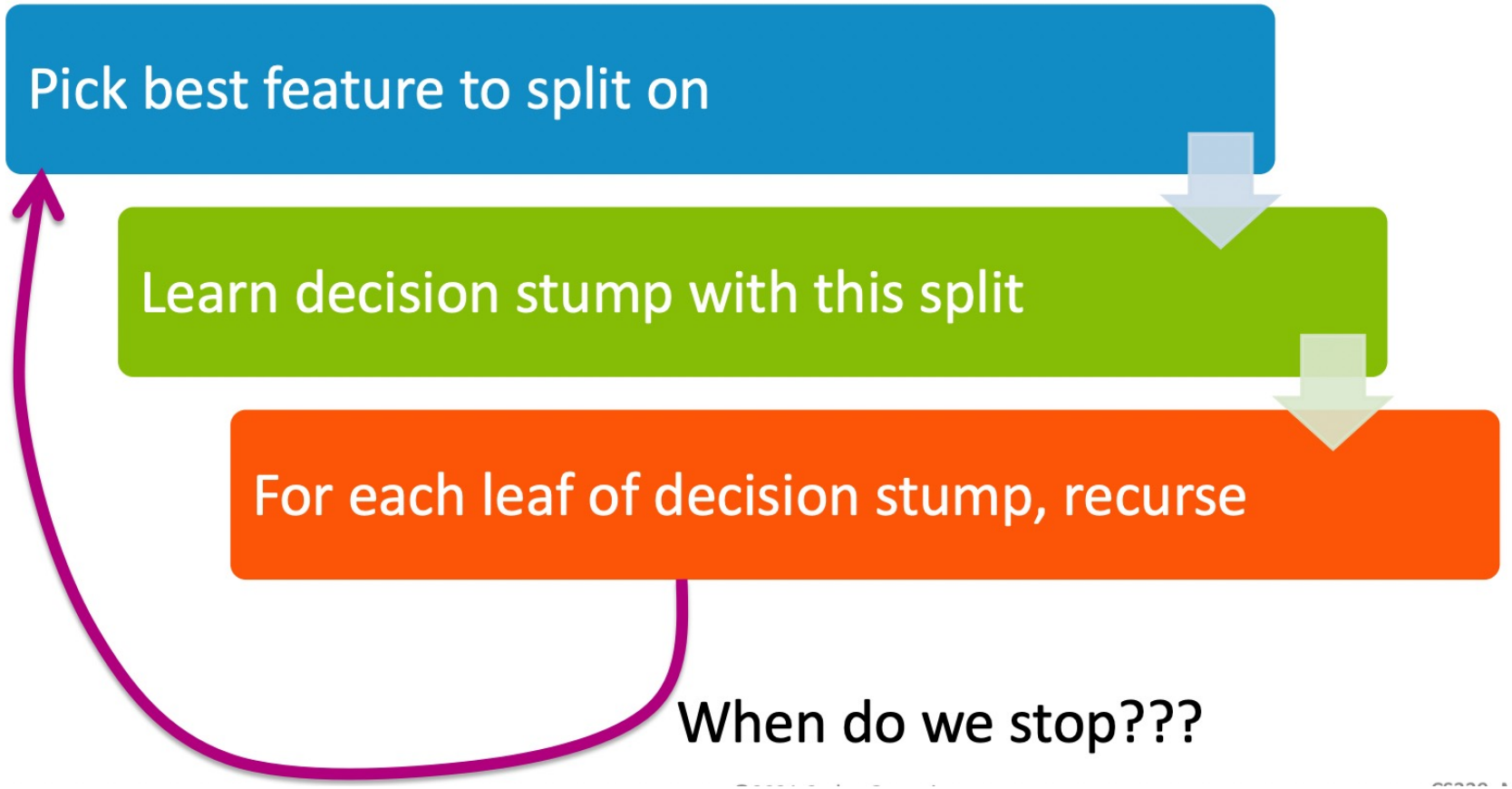
Second level



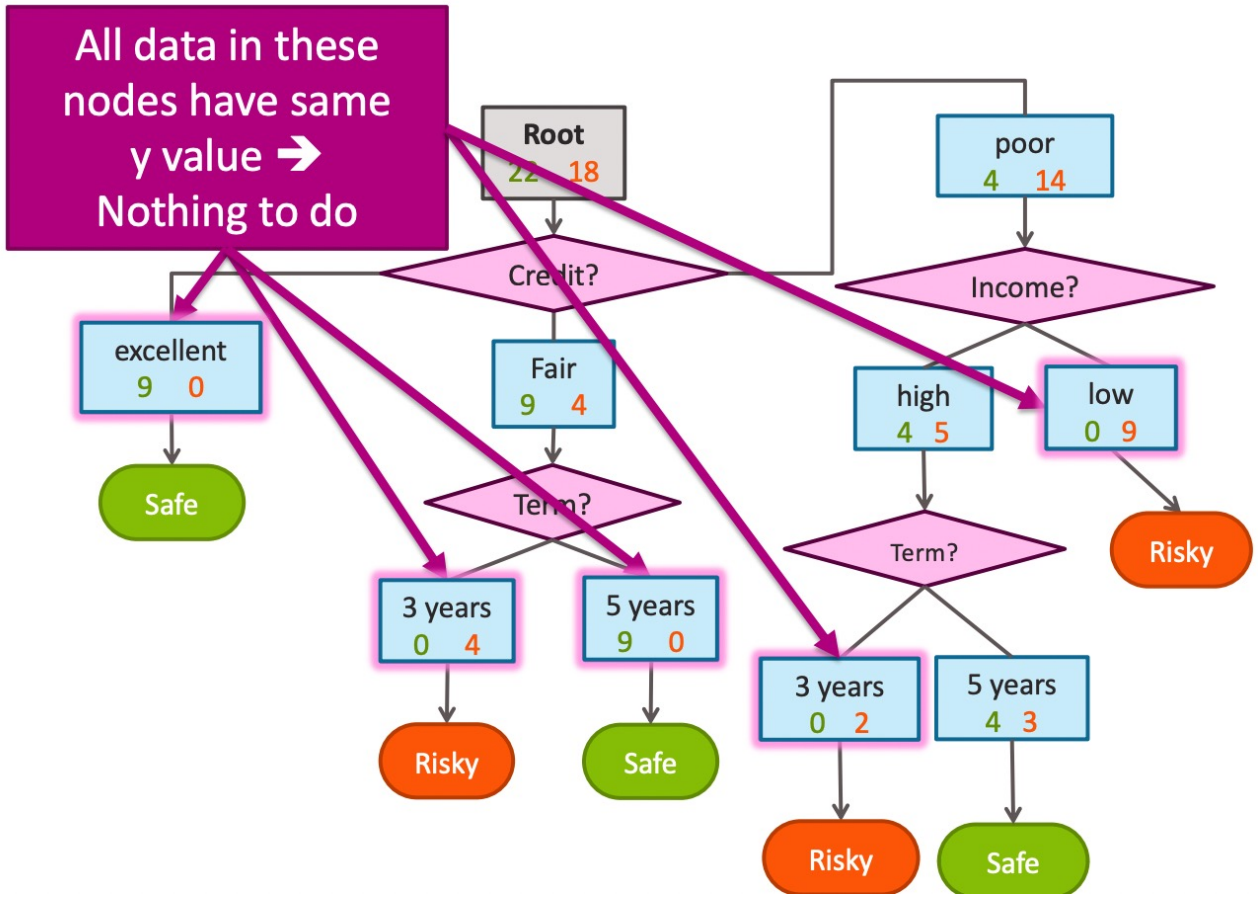
Final decision tree



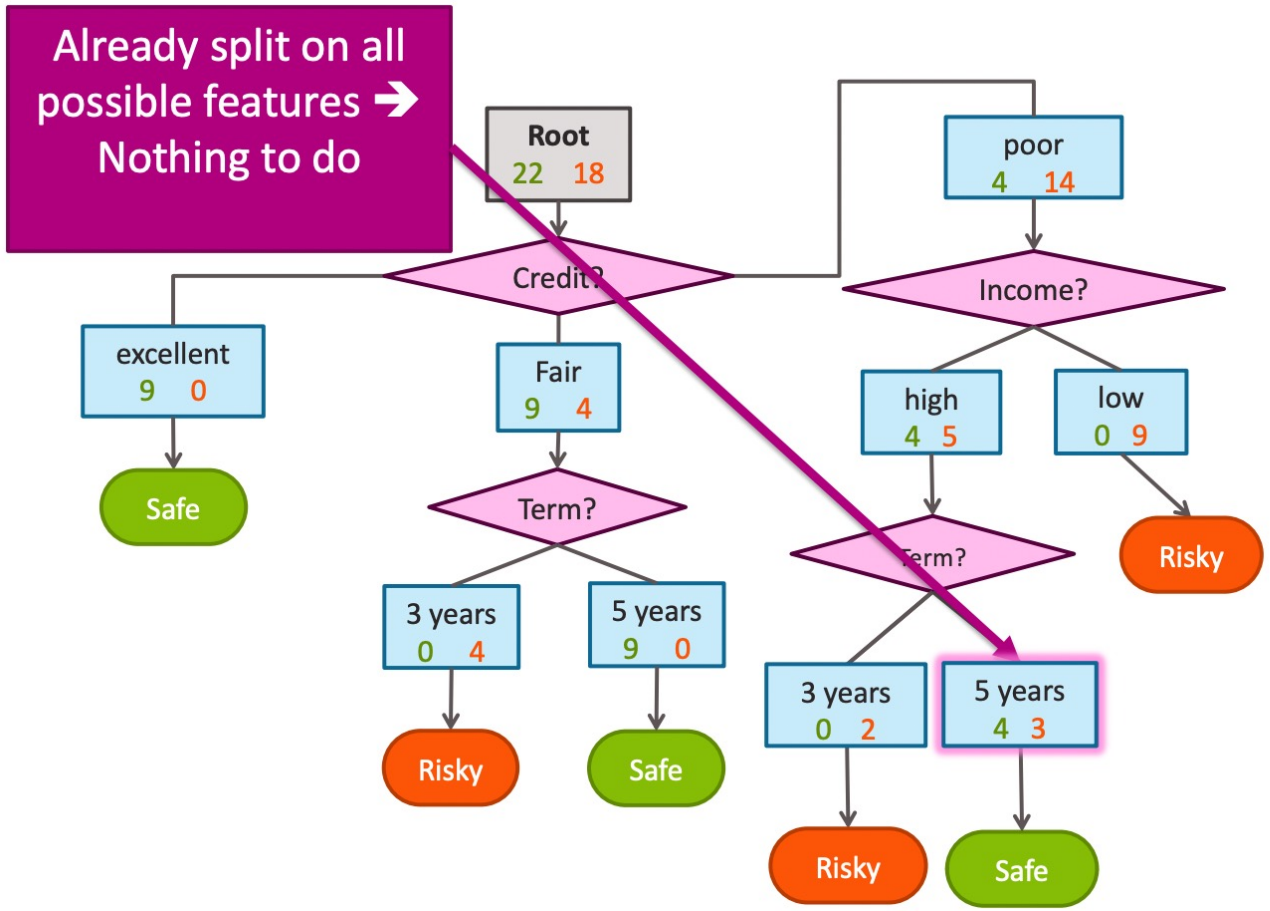
Simple greedy decision tree learning



Stopping condition 1: All data agrees on Y

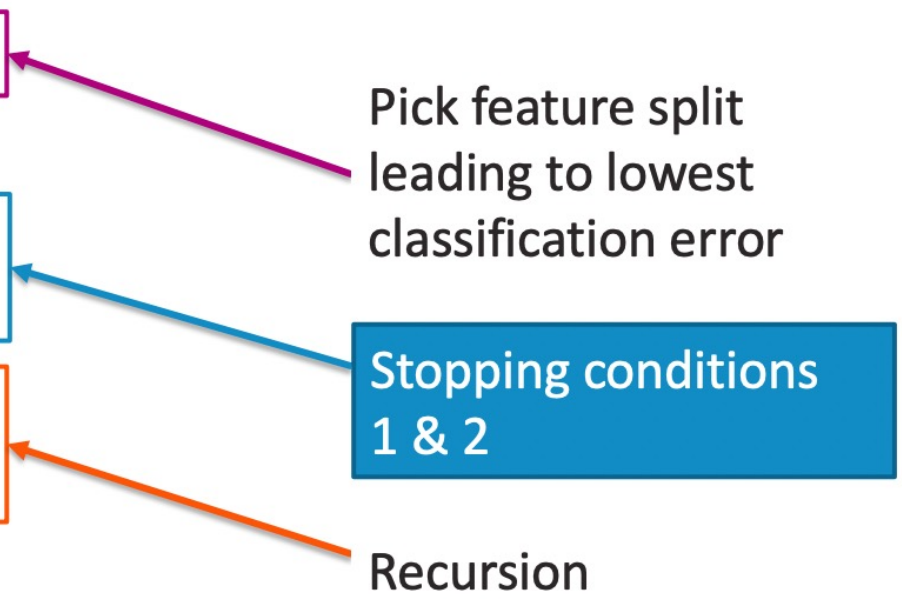


Stopping condition 2: Already split on all features



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



Is this a good idea?

Stopping condition 3:

y = x[1] xor x[2]

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

y values
True False

Root
2 2

Error = _____

=

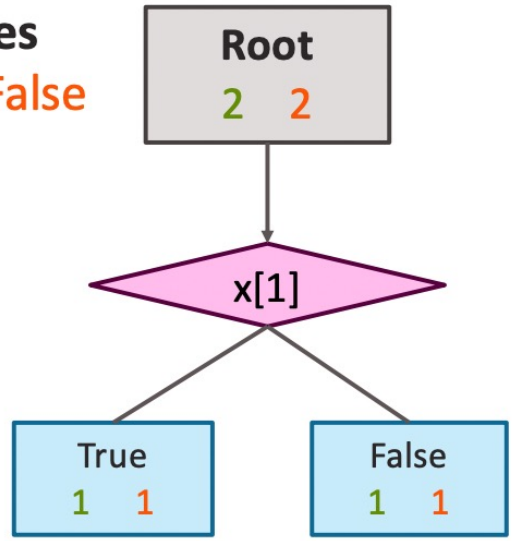
Tree	Classification error
(root)	0.5

Consider split on x[1]

y = x[1] xor x[2]

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

y values
True False



Error = _____.

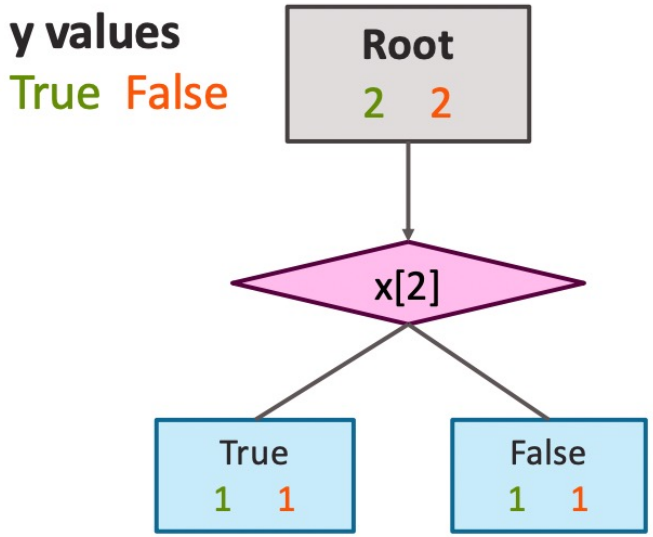
=

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

Consider split on x[2]

y = x[1] xor x[2]

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



Error = $\frac{1+1}{2+2}$ = 0.5

Neither features improve training error… Stop now???

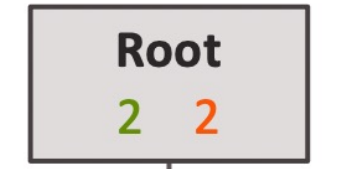
Tree	Classification error
(root)	0.5
Split on x[1]	0.5
Split on x[2]	0.5

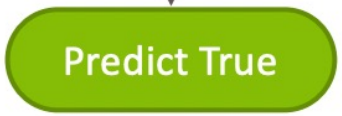
Final tree with stopping condition 3

y = x[1] xor x[2]

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

y values
True False





Tree	Classification error
with stopping condition 3	0.5

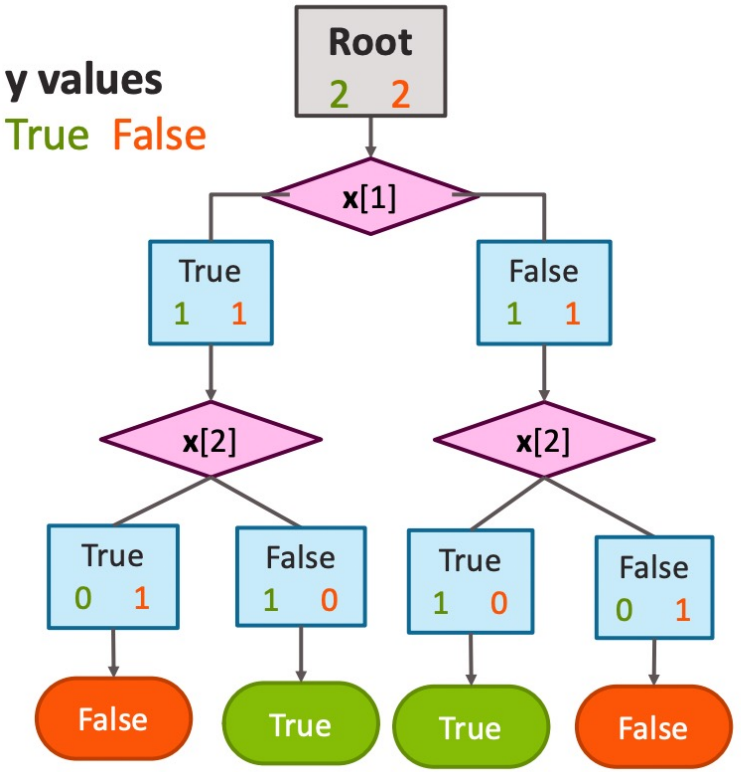
Without stopping condition 3

Condition 3 (stopping when training error doesn't' improve) **is not recommended!**

y = x[1] xor x[2]

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

Tree	Classification error
with stopping condition 3	0.5
without stopping condition 3	

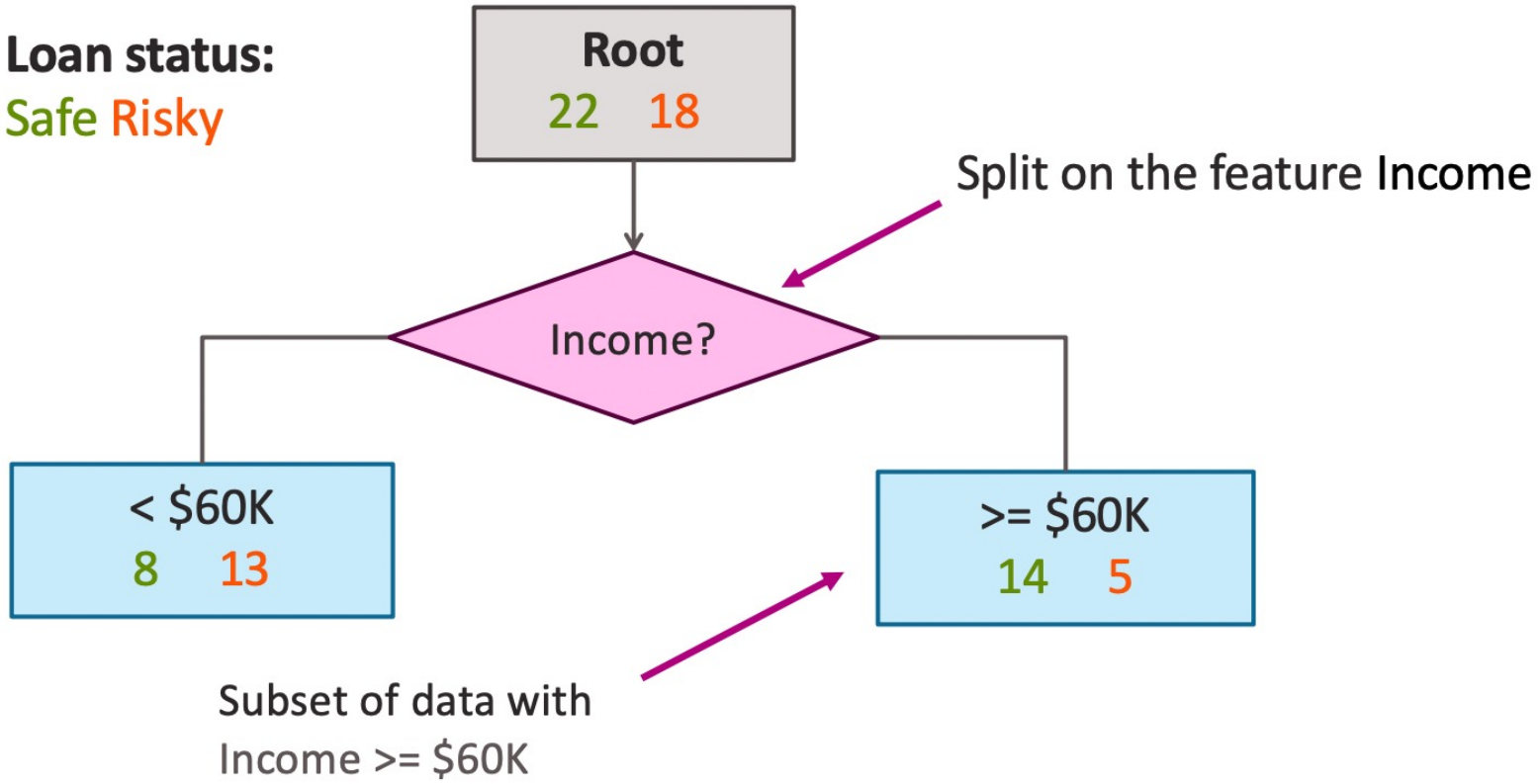


Decision tree learning: Real valued features

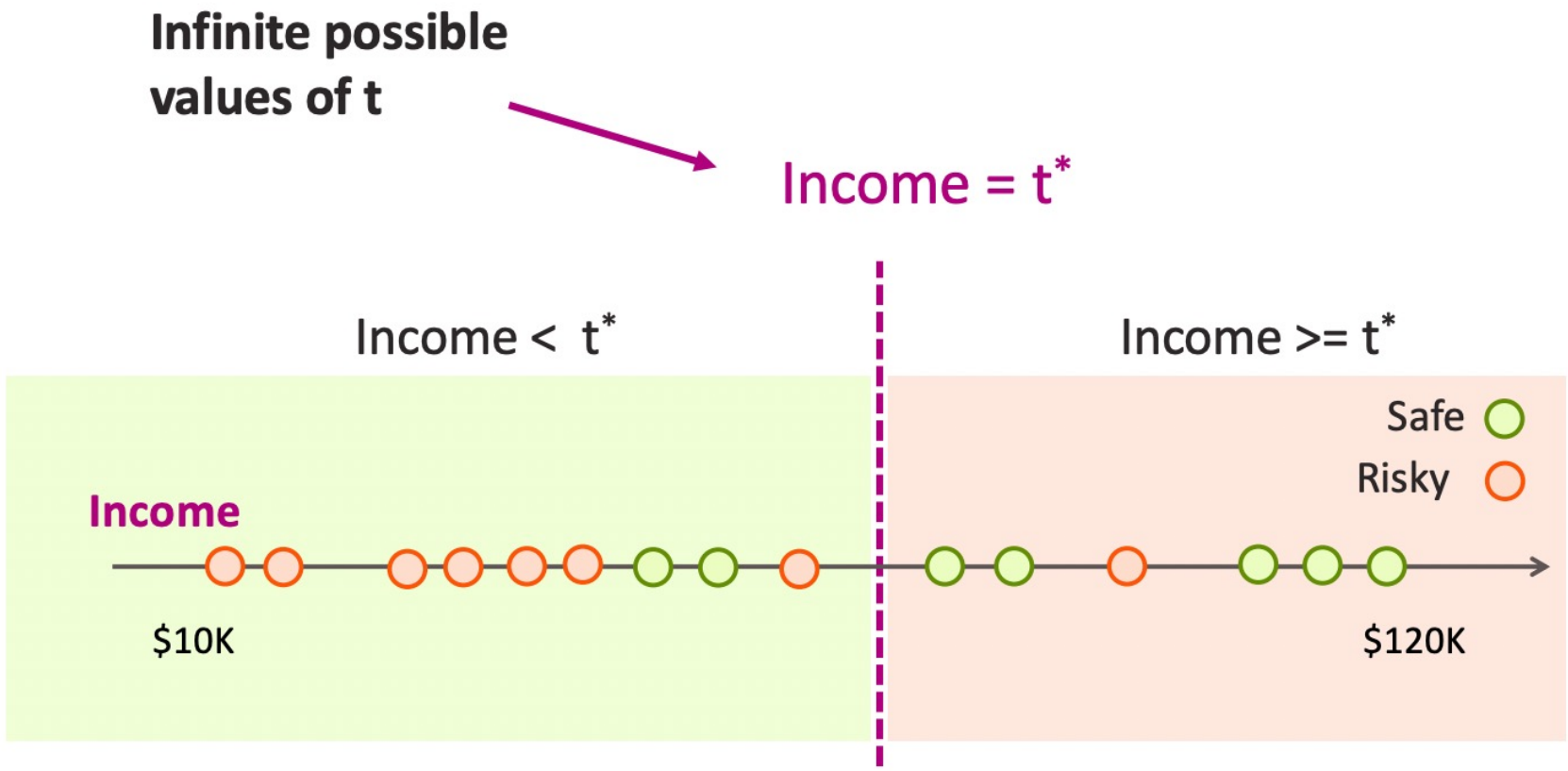
How do we use real values inputs?

Income	Credit	Term	y
$105 K	excellent	3 yrs	Safe
$112 K	good	5 yrs	Risky
$73 K	fair	3 yrs	Safe
$69 K	excellent	5 yrs	Safe
$217 K	excellent	3 yrs	Risky
$120 K	good	5 yrs	Safe
$64 K	fair	3 yrs	Risky
$340 K	excellent	5 yrs	Safe
$60 K	good	3 yrs	Risky

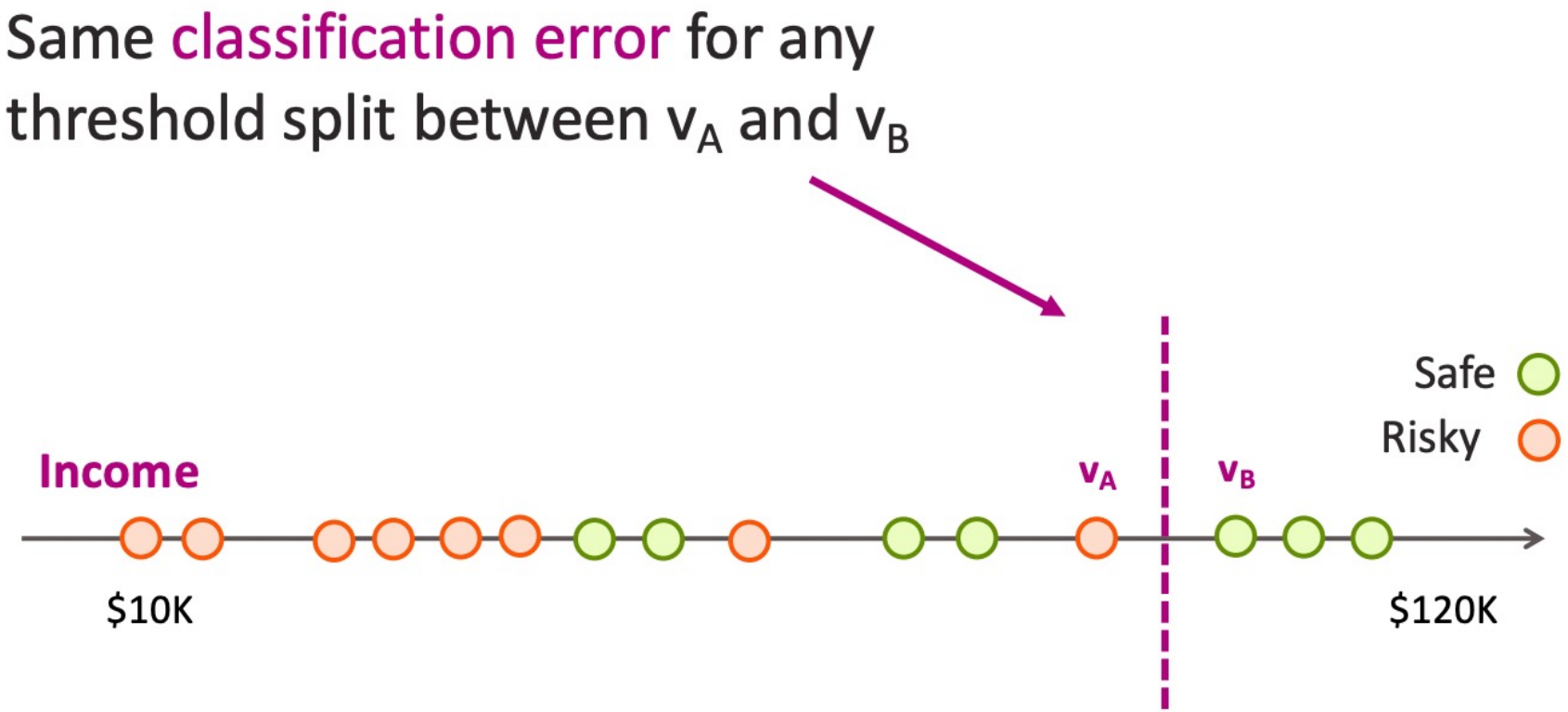
Threshold split



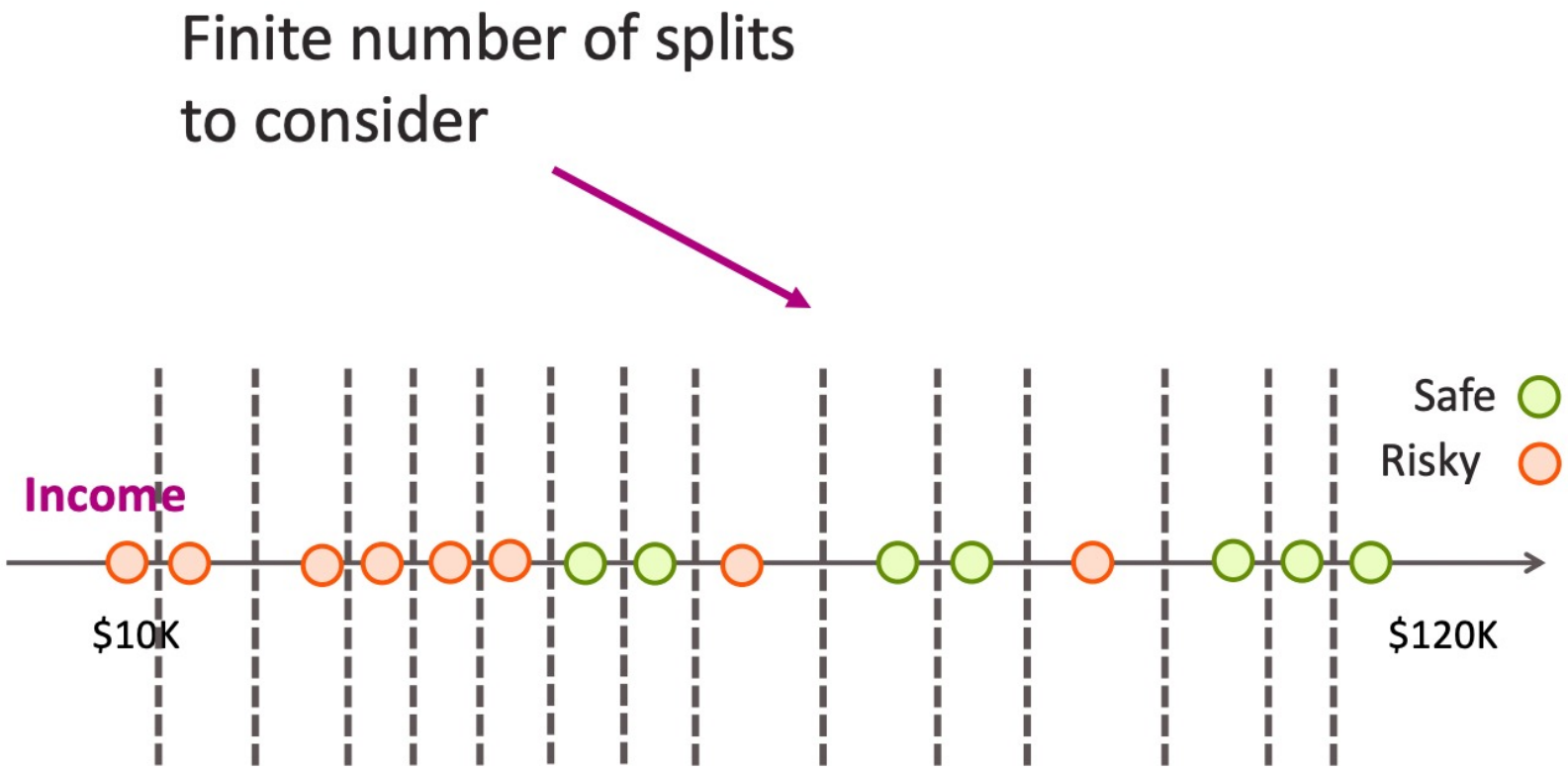
Finding the best threshold split



Consider a threshold between points



Only need to consider mid-points



Threshold split selection algorithm

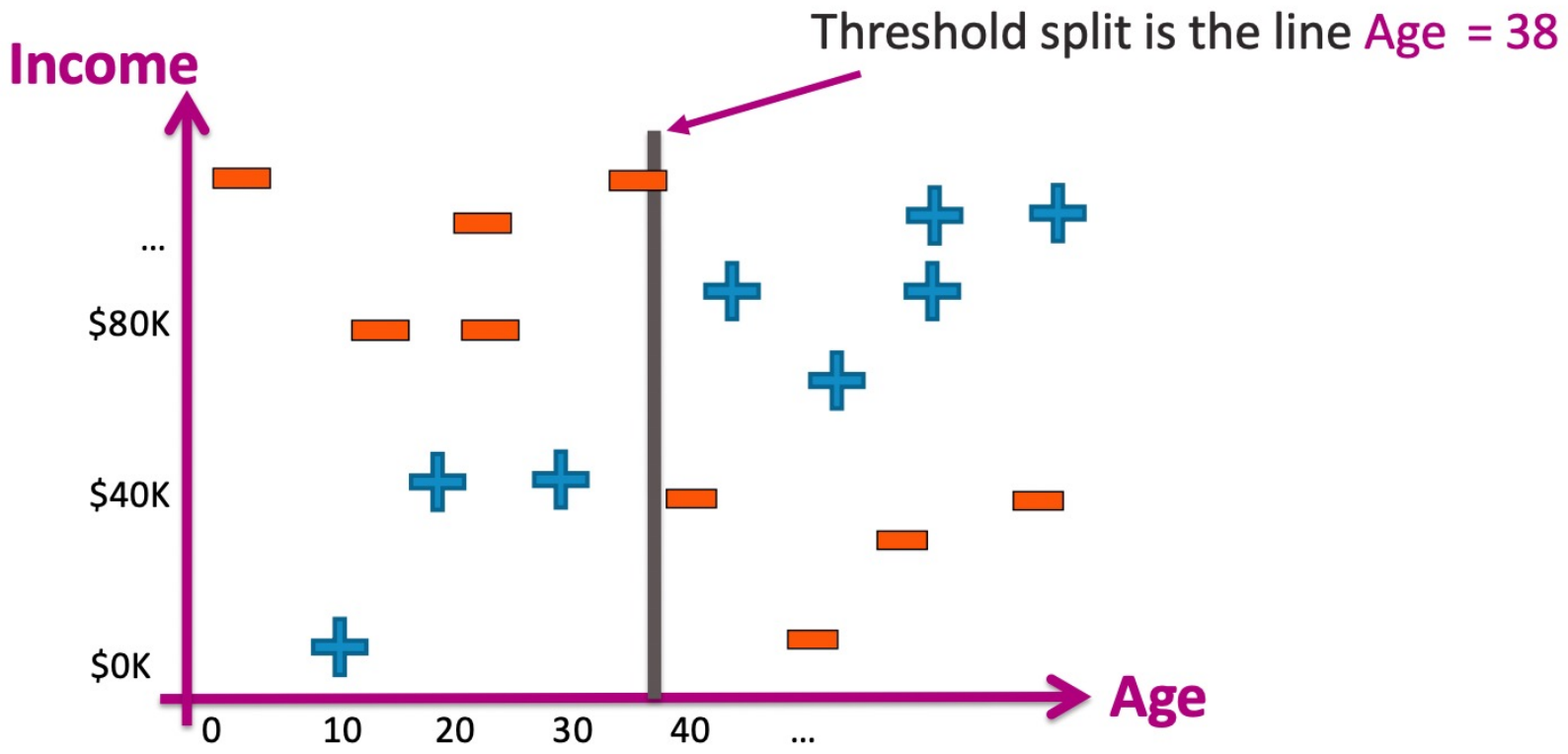
- Step 1: Sort the values of a feature $h_j(x)$:

  Let $\{v_1, v_2, v_3, \ldots v_N\}$ denote sorted values

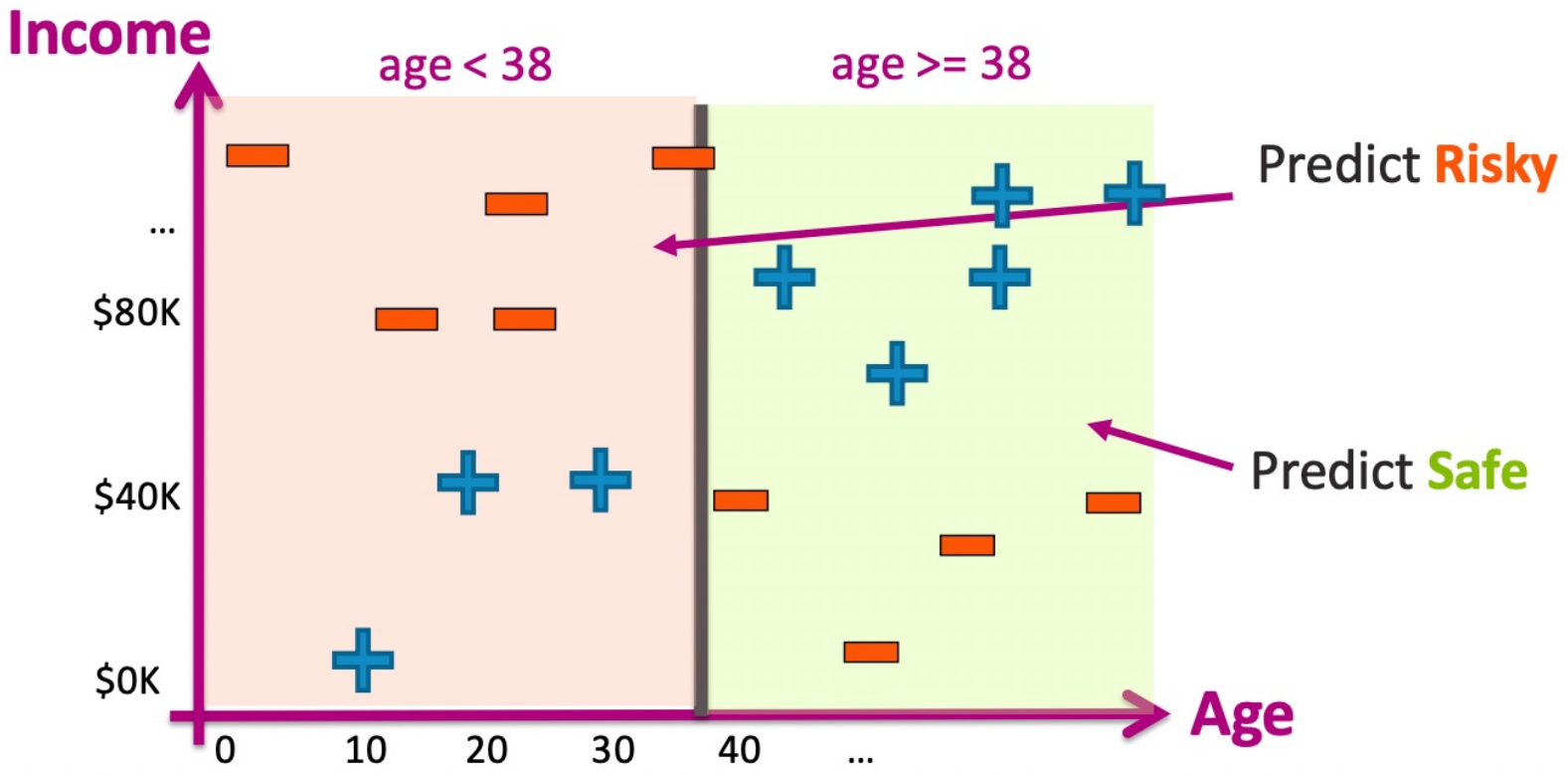
- Step 2:

  - For i = 1, 2, ... N-1
    - Consider split $t_i = (v_i + v_{i+1}) / 2$
    - Compute classification error for threshold split $h_j(x) >= t_i$
  - Choose the t* with the lowest classification error

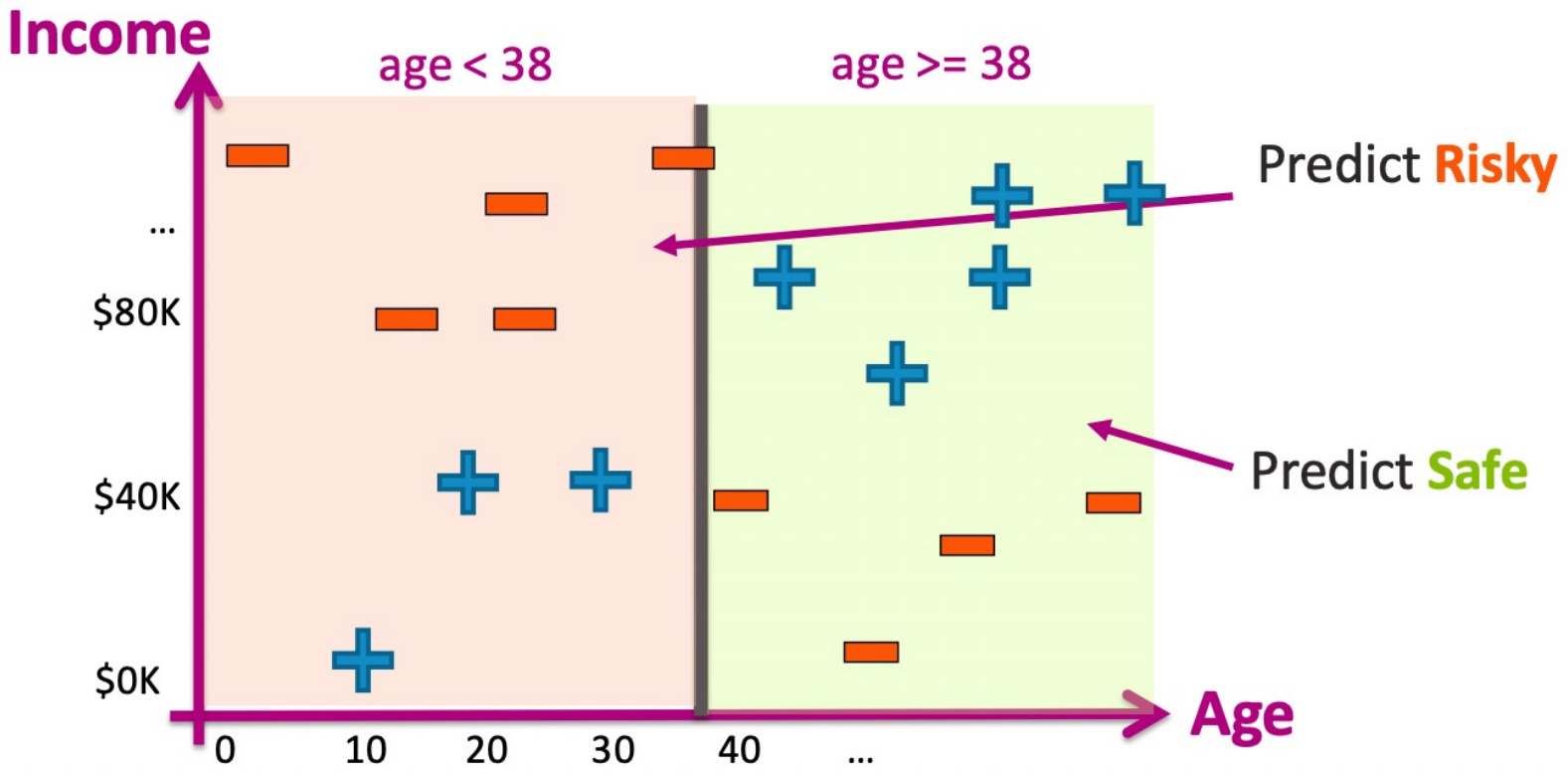
Visualizing the threshold split



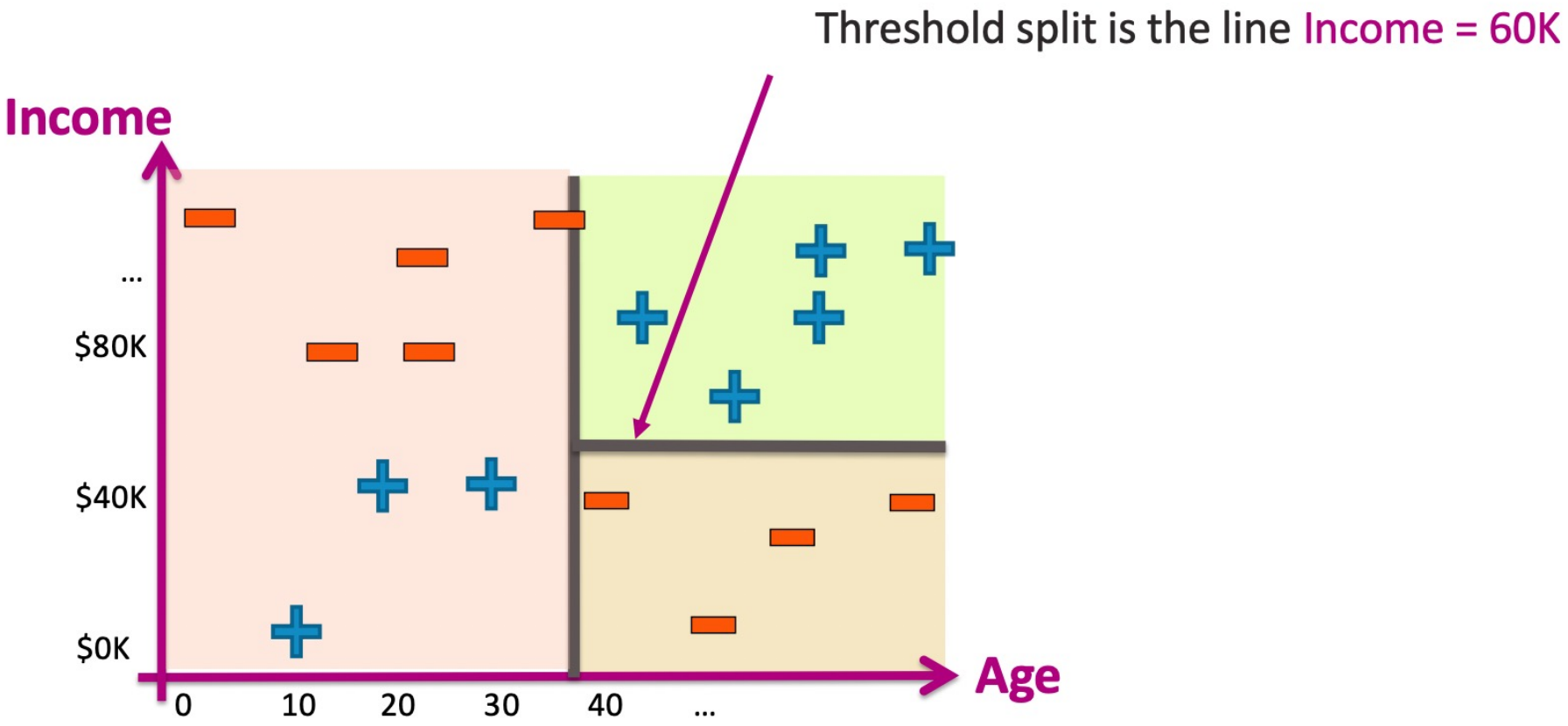
Split on Age >= 38



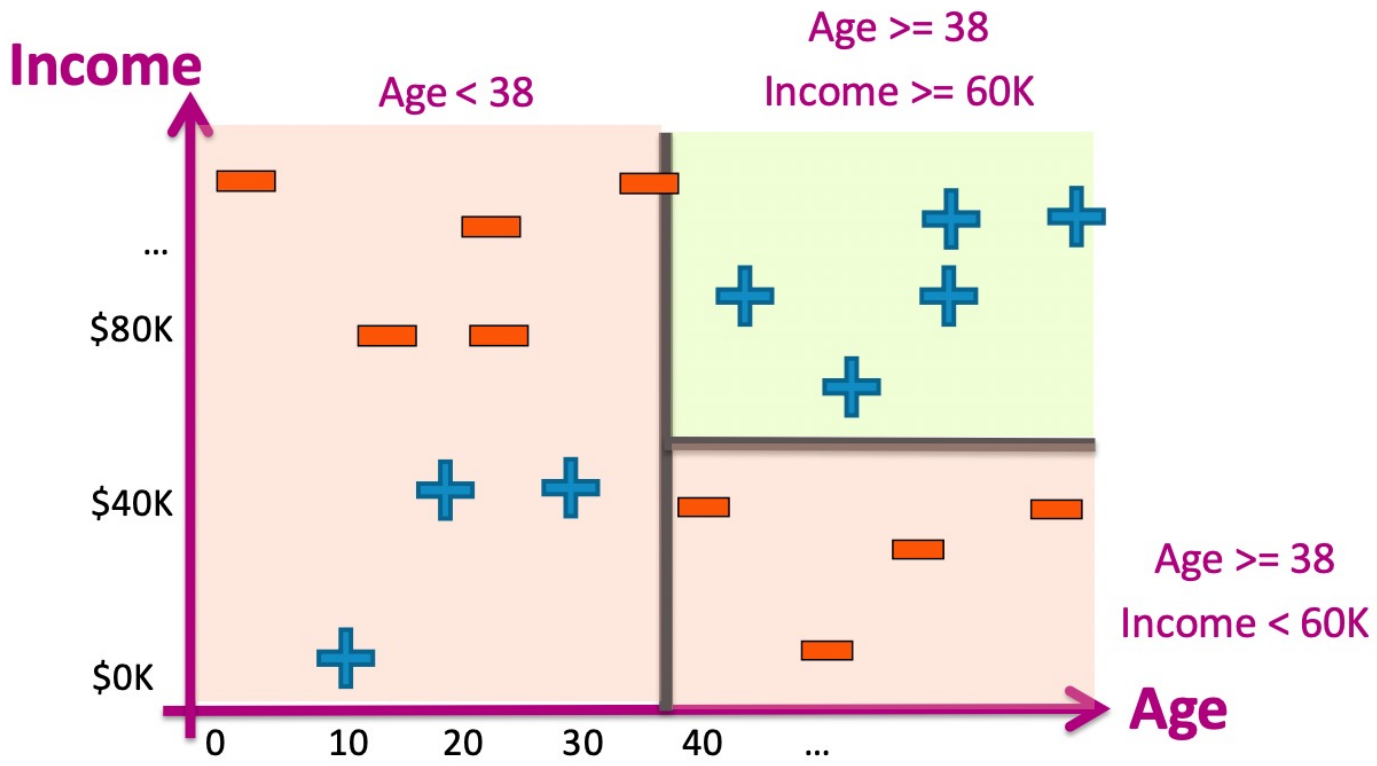
Split on Age >= 38



Depth 2: Split on Income >= $60K



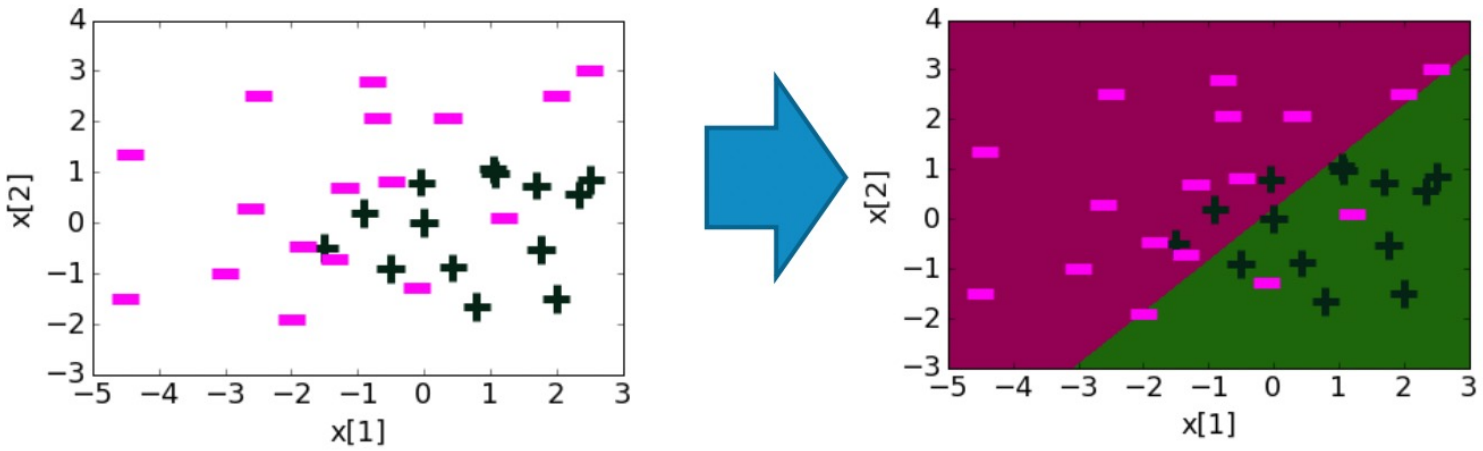
Each split partitions the 2-D space



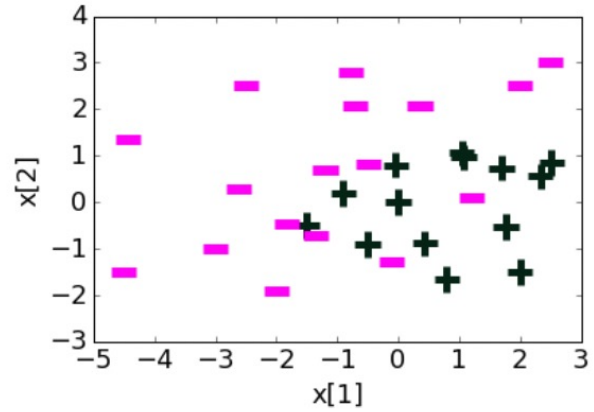
Decision trees vs regression:

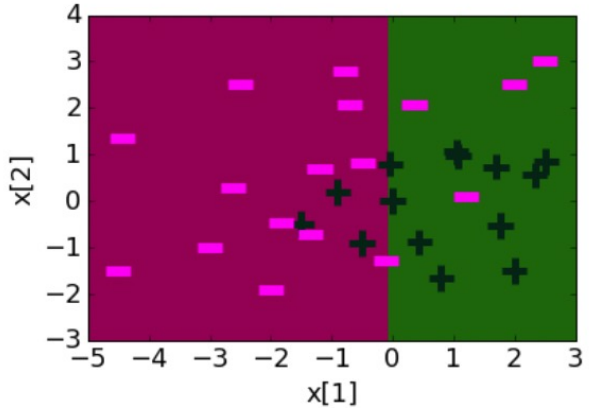
Regression

Feature	Value	Weight Learned
$h_0(x)$	1	0.22
$h_1(x)$	x[1]	1.12
$h_2(x)$	x[2]	-1.07



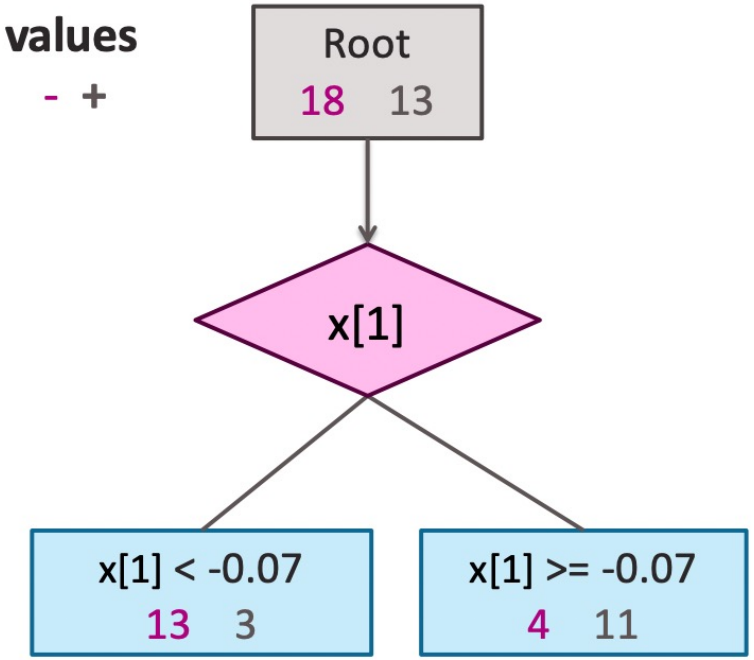
Depth 1: Split on x[1]



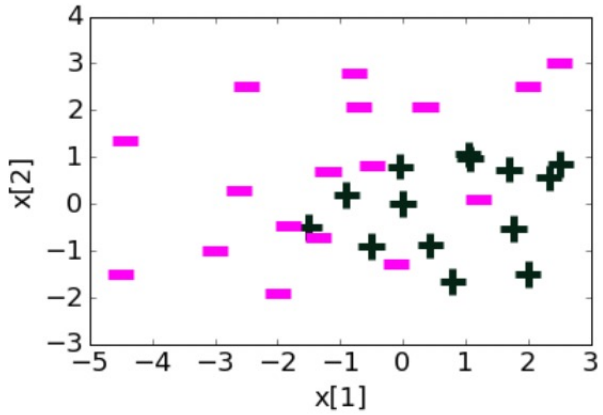


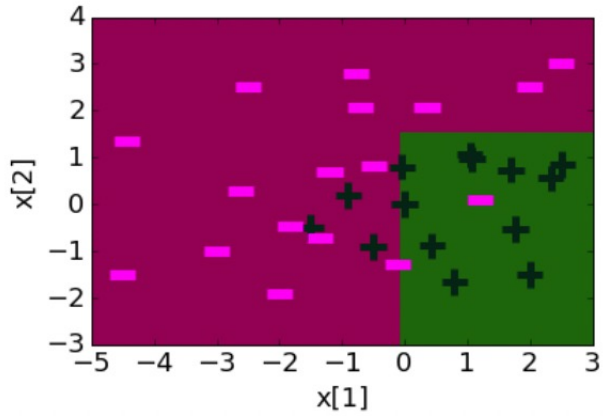
y values

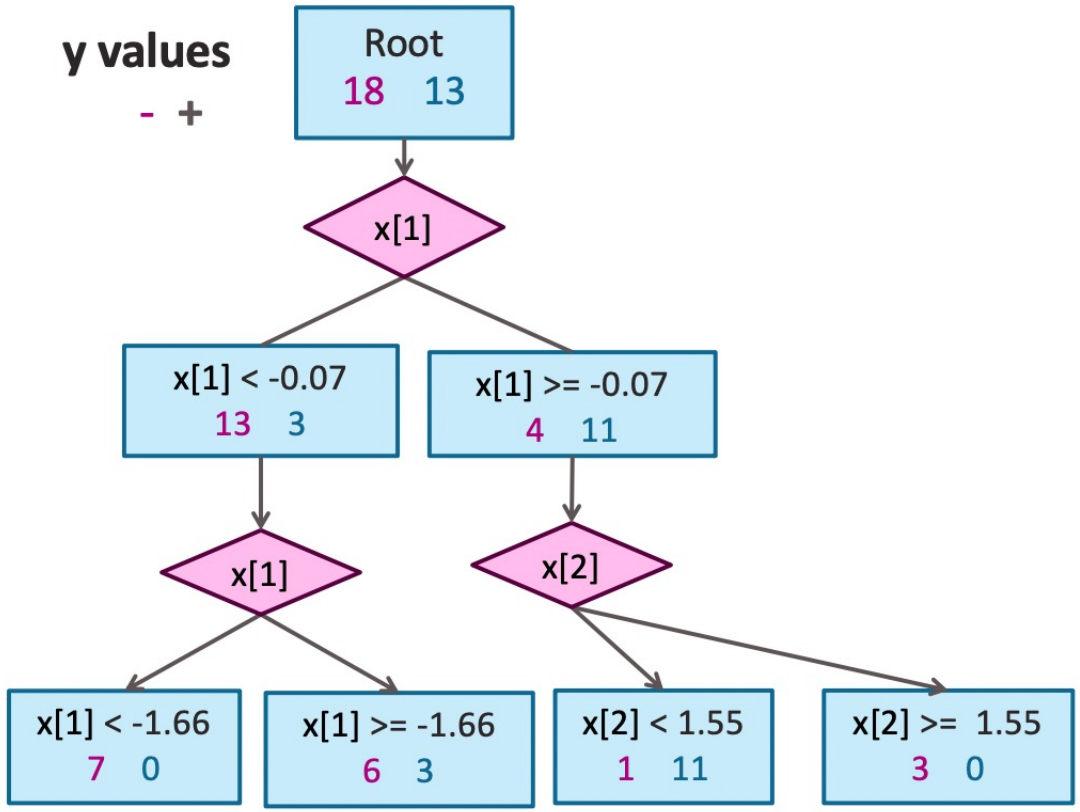
\- +



Depth 2

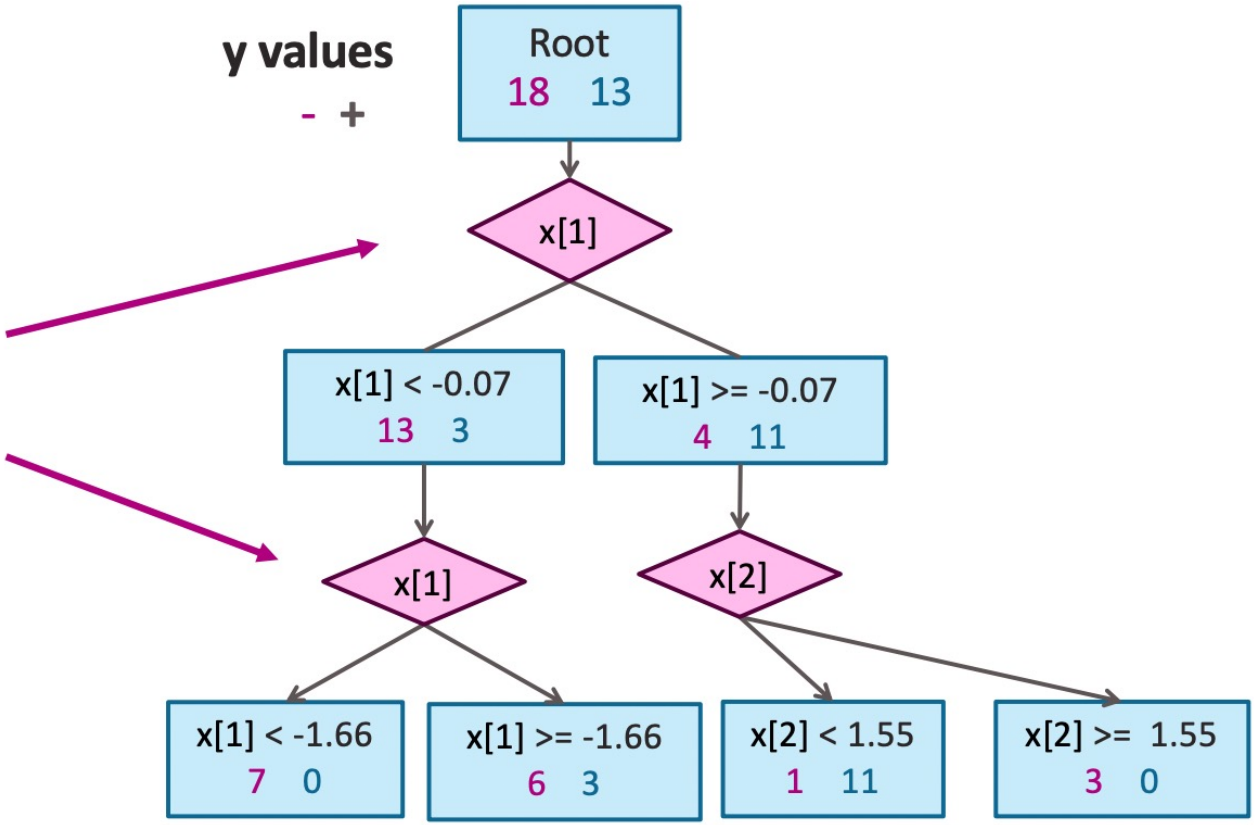

x[2]
x[1]


x[2]
x[1]

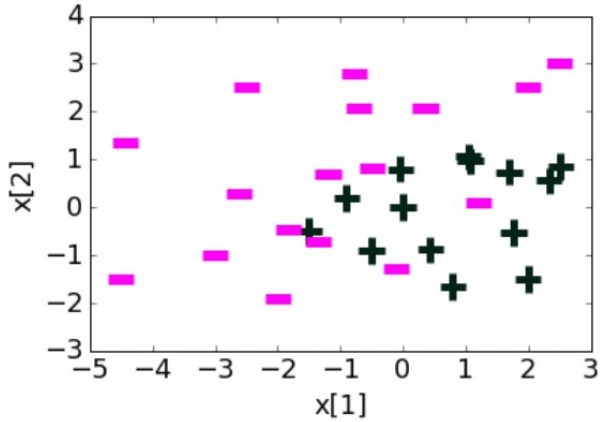

y values
- +
Root
18 13
x[1]
x[1] < -0.07
13 3
x[1] >= -0.07
4 11
x[1]
x[2]
x[1] < -1.66
7 0
x[1] >= -1.66
6 3
x[2] < 1.55
1 11
x[2] >= 1.55
3 0

Threshold split caveat

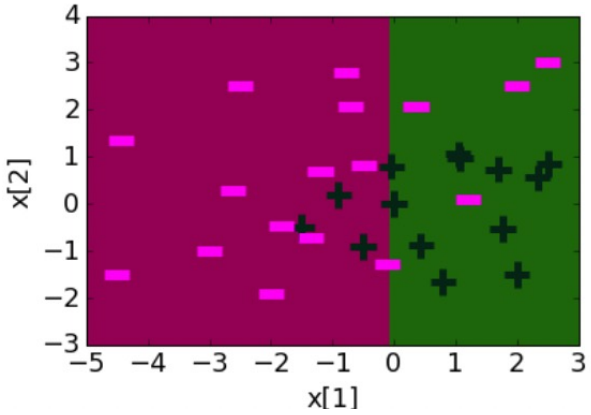
For threshold splits, same feature can be used multiple times



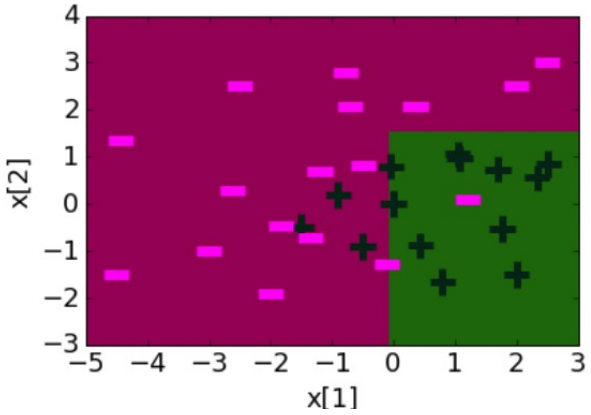
Decision boundaries



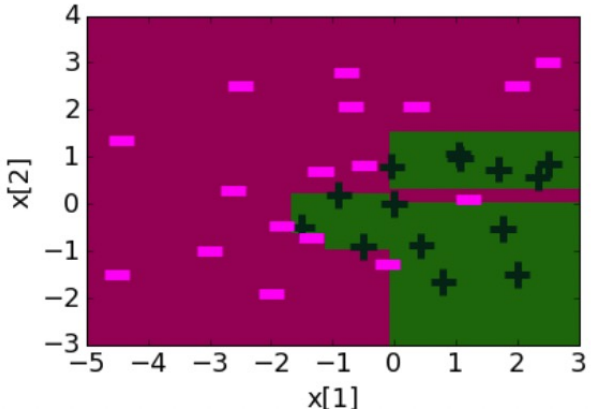
Depth 1



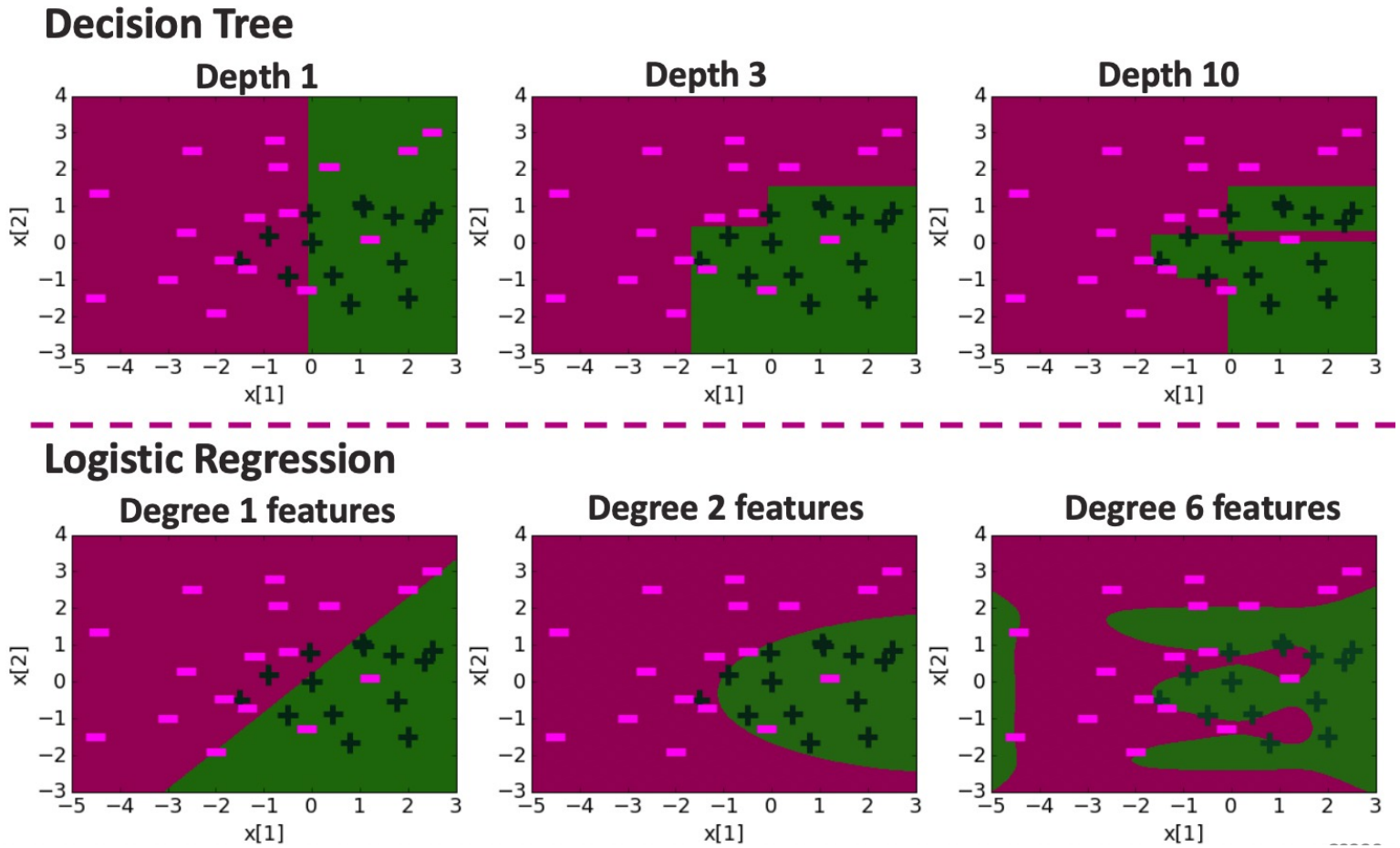
Depth 2



Depth 10



Comparing decision boundaries



Summary of decision trees

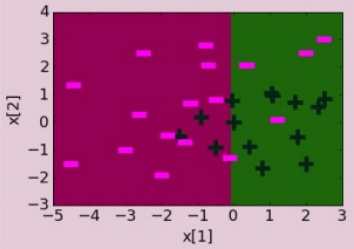
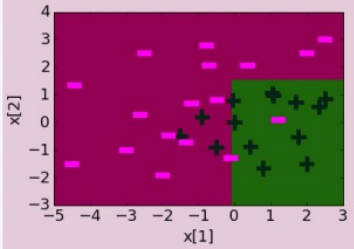
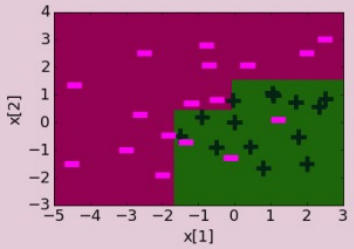
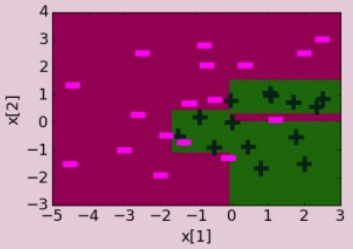
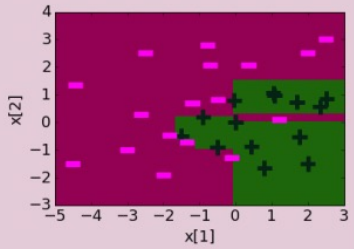
What you can do now

- Define a decision tree classifier
- Interpret the output of a decision trees
- Learn a decision tree classifier using greedy algorithm
- Traverse a decision tree to make predictions
    - Majority class predictions
- Tackle continuous and discrete features

Overfitting in decision trees

What happens when we increase depth?

Training error reduces with depth

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

Two approaches to picking simpler trees

1. Early Stopping:

   Stop the learning algorithm before tree becomes too complex

2. Pruning:

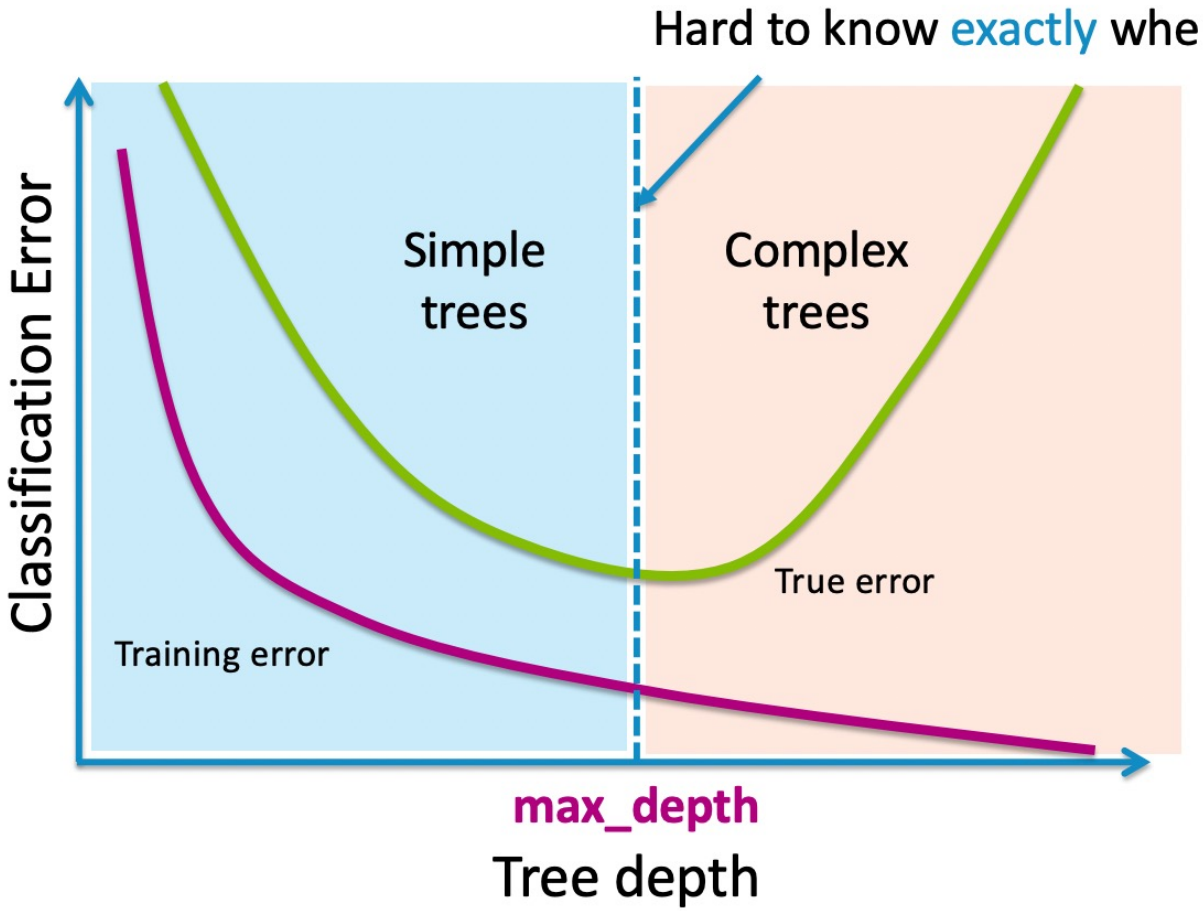
   Simplify the tree after the learning algorithm terminates

Technique 1: Early stopping

- Stopping conditions (recap):
    1. All examples have the same target value
    2. No more features to split on
- Early stopping conditions:

1. Limit tree depth (choose max depth using validation set)
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

Challenge with early stopping condition 1

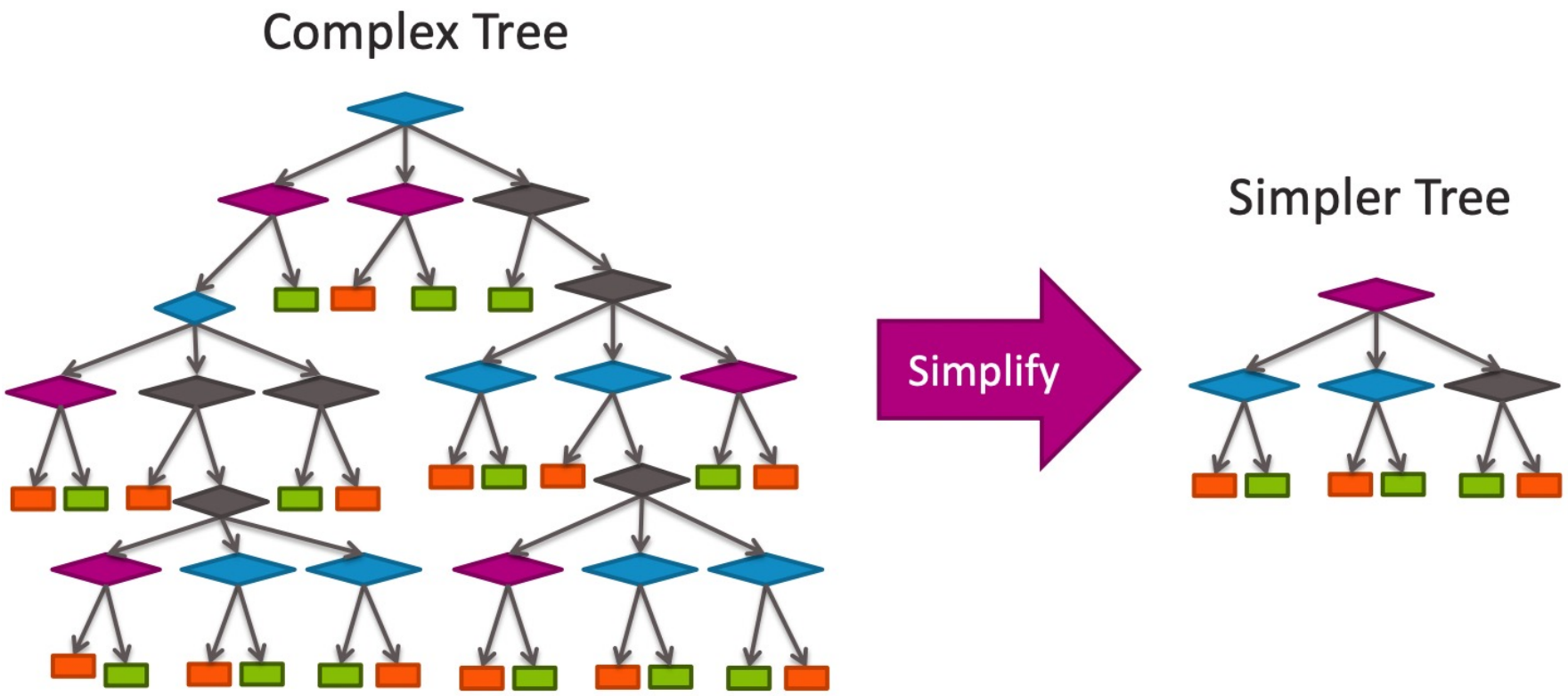


Also, might want some branches of tree to go deeper while others remain shallow

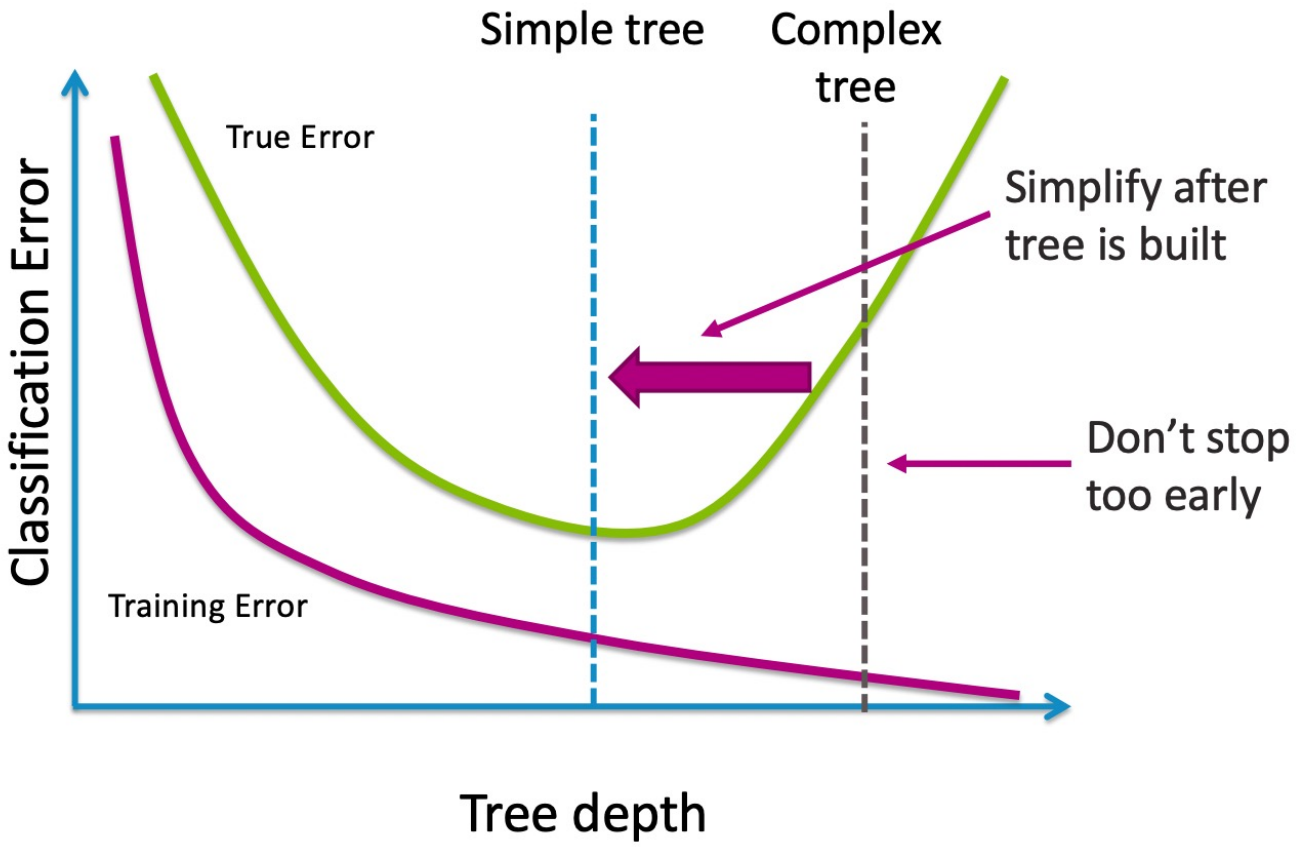
Early stopping condition 2: Pros and Cons

- 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

Pruning: Intuition



Pruning motivation

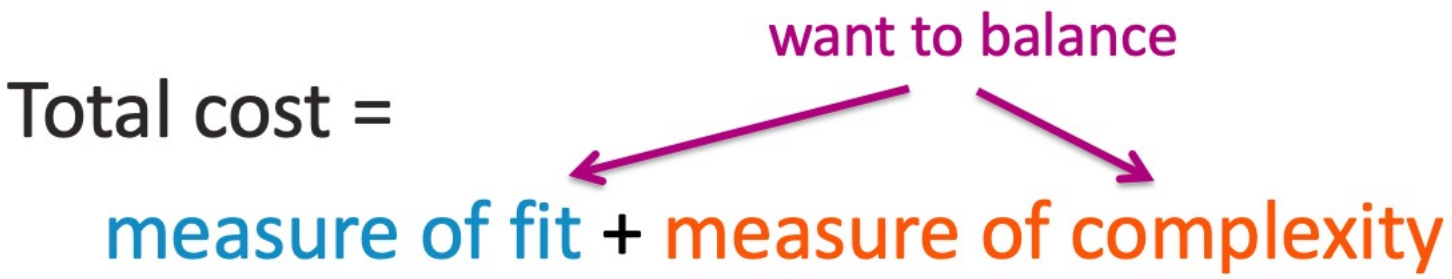


Scoring trees: Desired total quality format

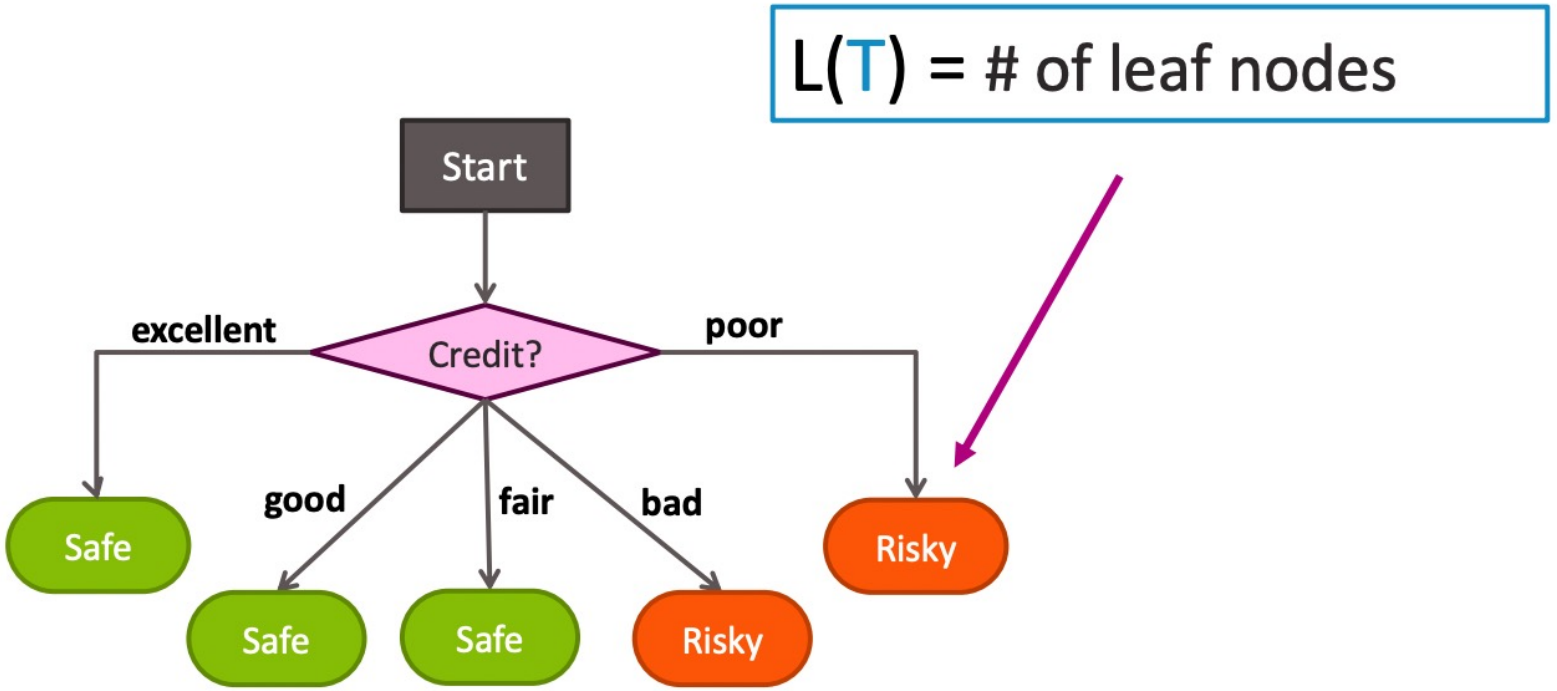
Want to balance:

i. How well tree fits data

ii. Complexity of tree

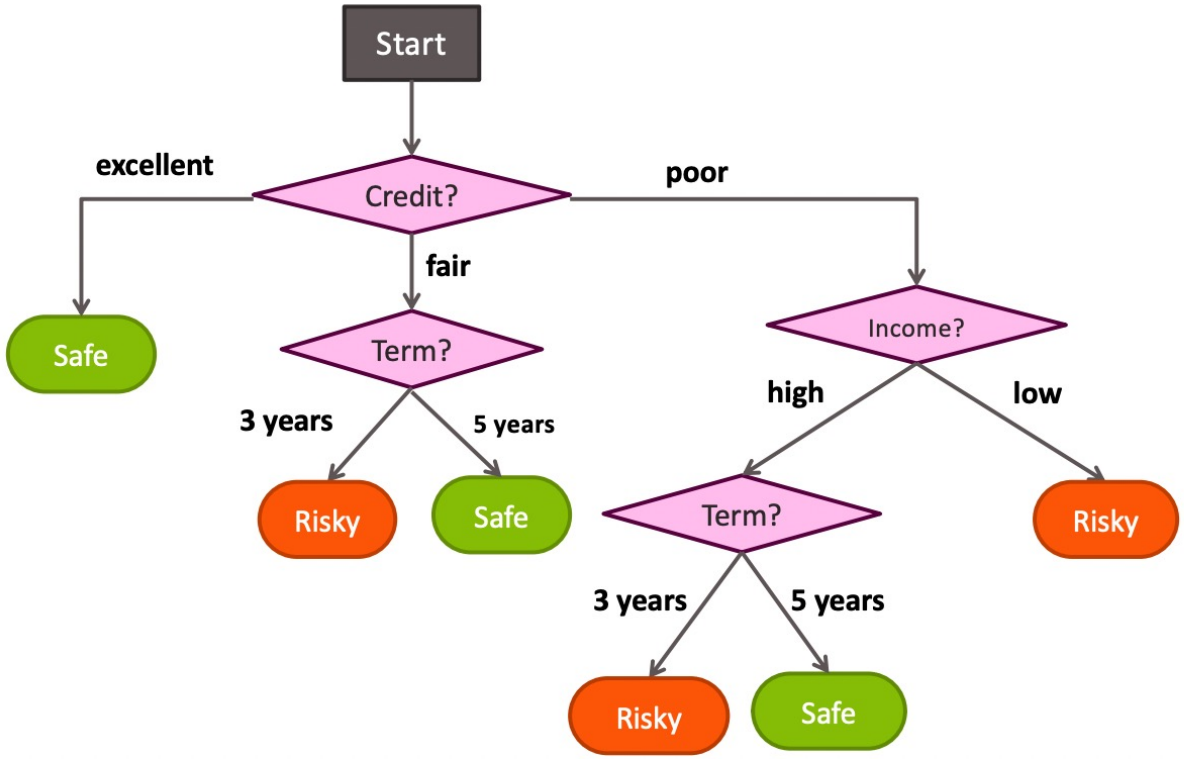


Simple measure of complexity of tree

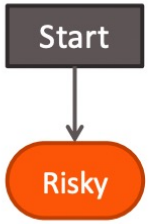


Balance simplicity & predictive power

Too complex, risk of overfitting



Too simple, high classification error



Balancing fit and complexity

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

tuning parameter

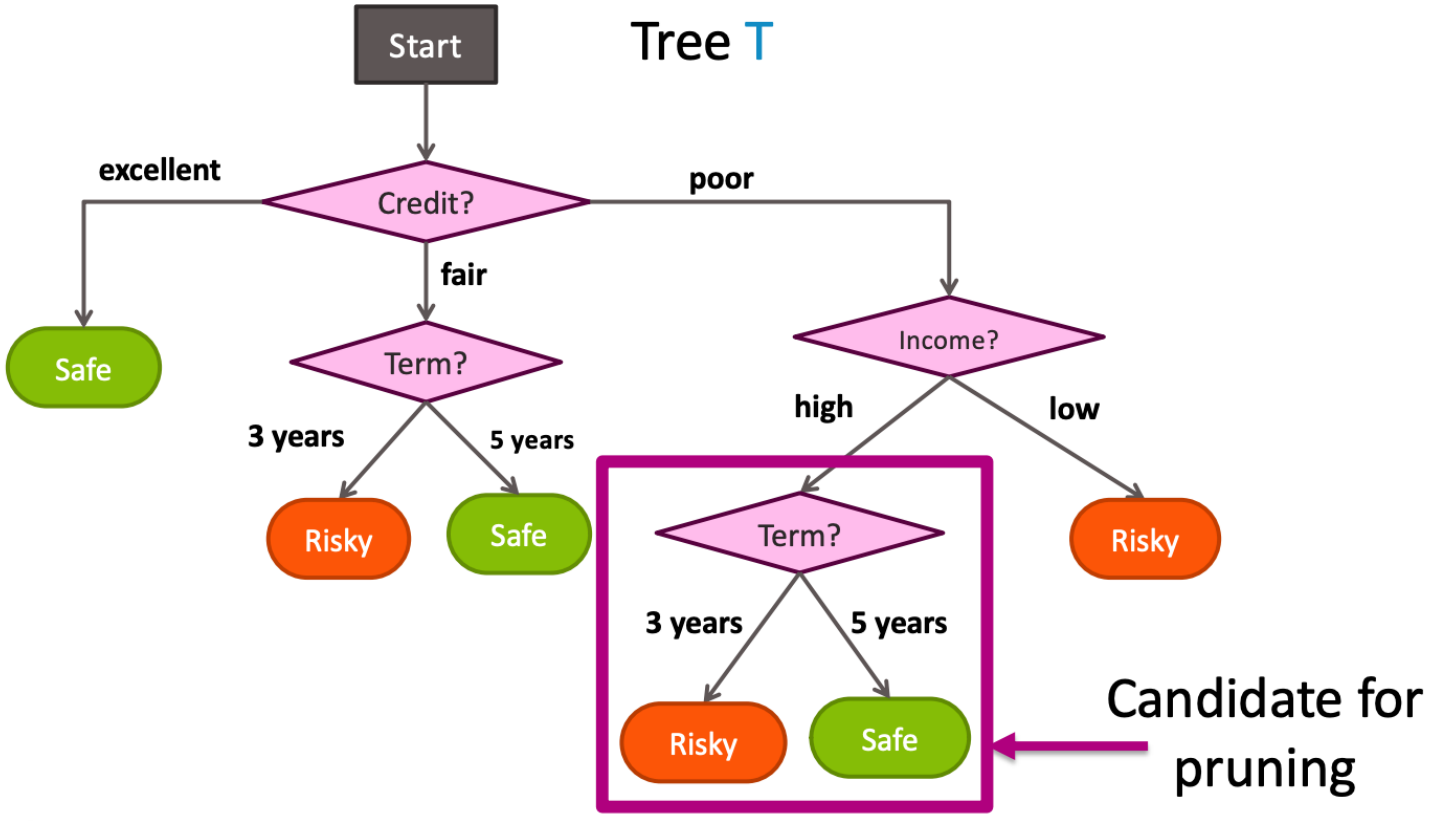
If $\lambda=0$:

If $\lambda=\infty$:

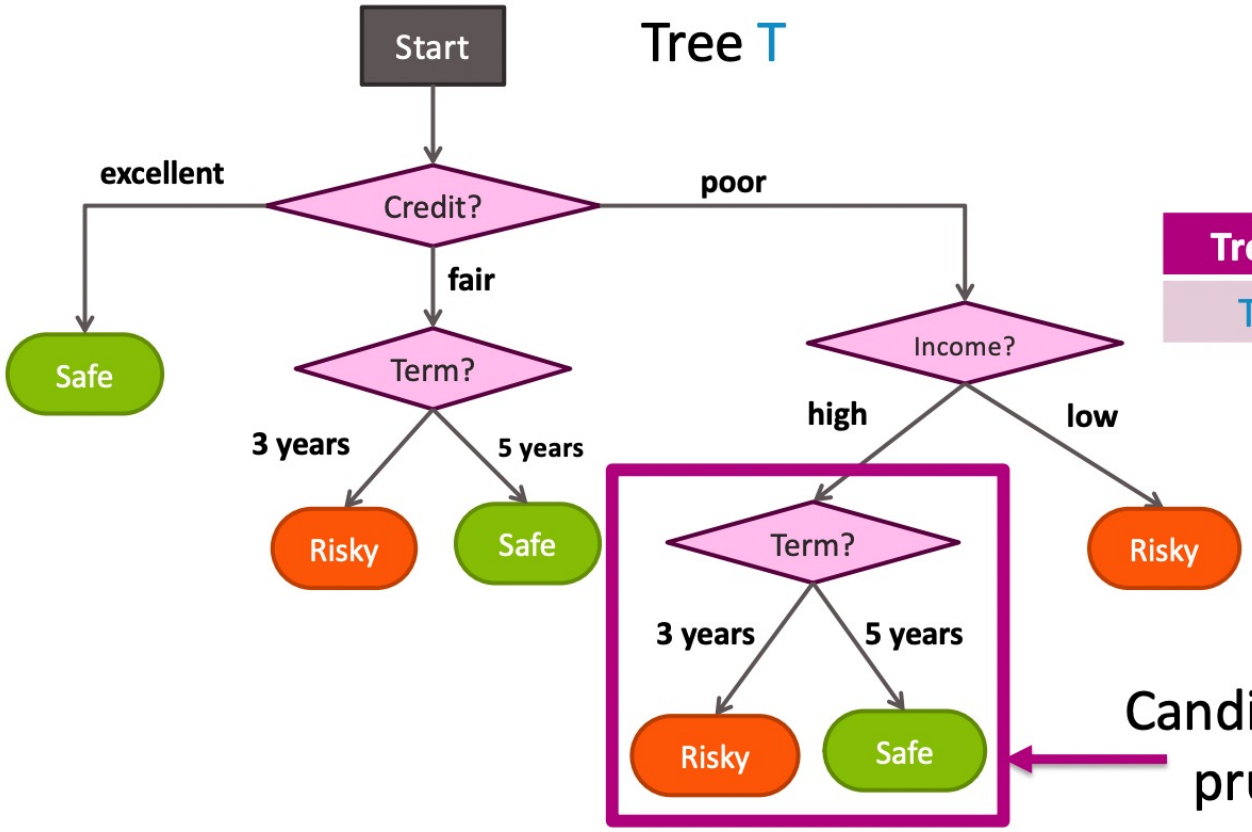
If λ in between:

Tree pruning algorithm

Step 1: Consider a split



Step 2: Compute total cost C(T) of split



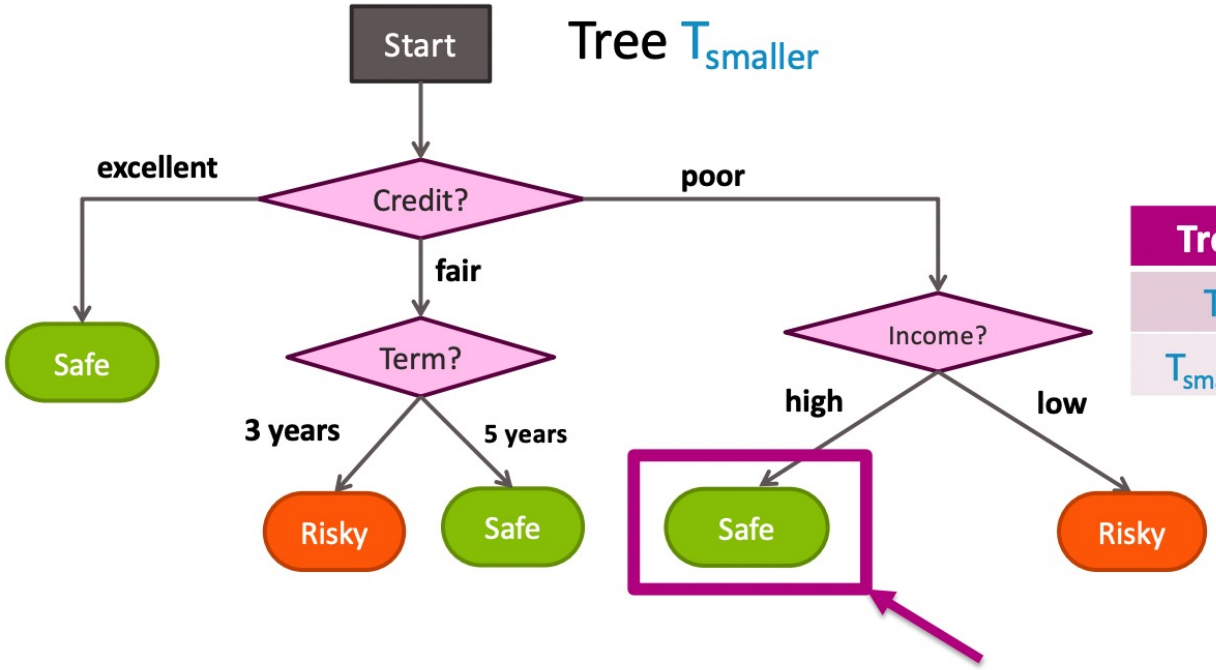
$\lambda = 0.3$

Tree	Error	#Leaves	Total
T	0.25	6	0.43

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

Candidate for pruning

Step 2: "Undo" the splits on $T_{smaller}$



Tree $T_{smaller}$

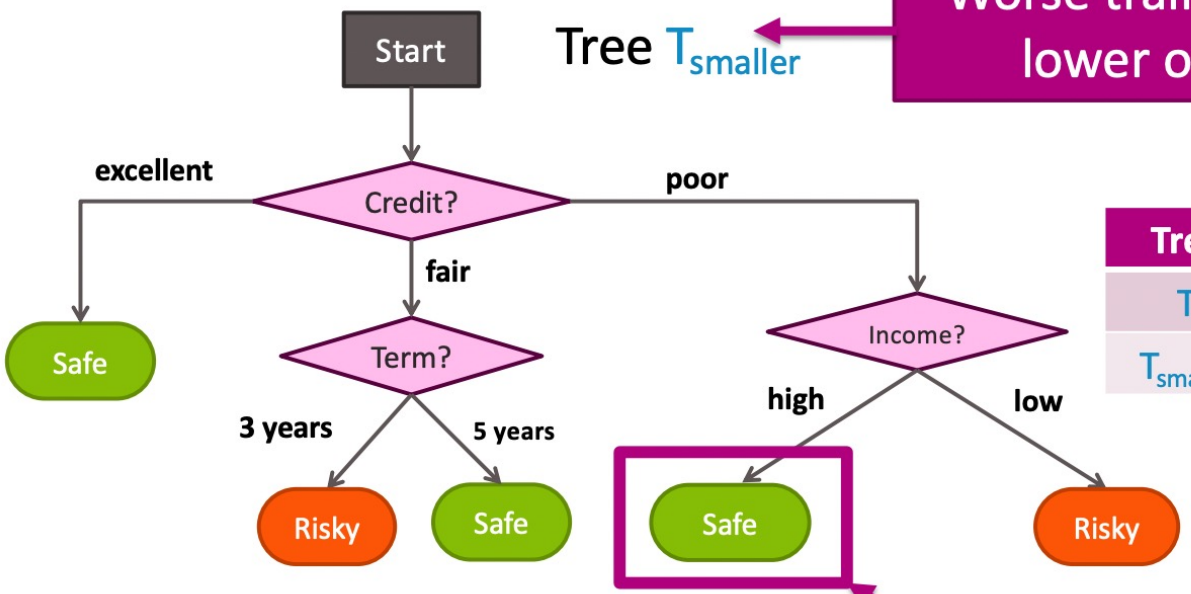
$\lambda = 0.3$

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

$C(T) = Error(T) + \lambda L(T)$

Replace split by leaf node?

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



Tree $T_{smaller}$

Worse training error but lower overall cost

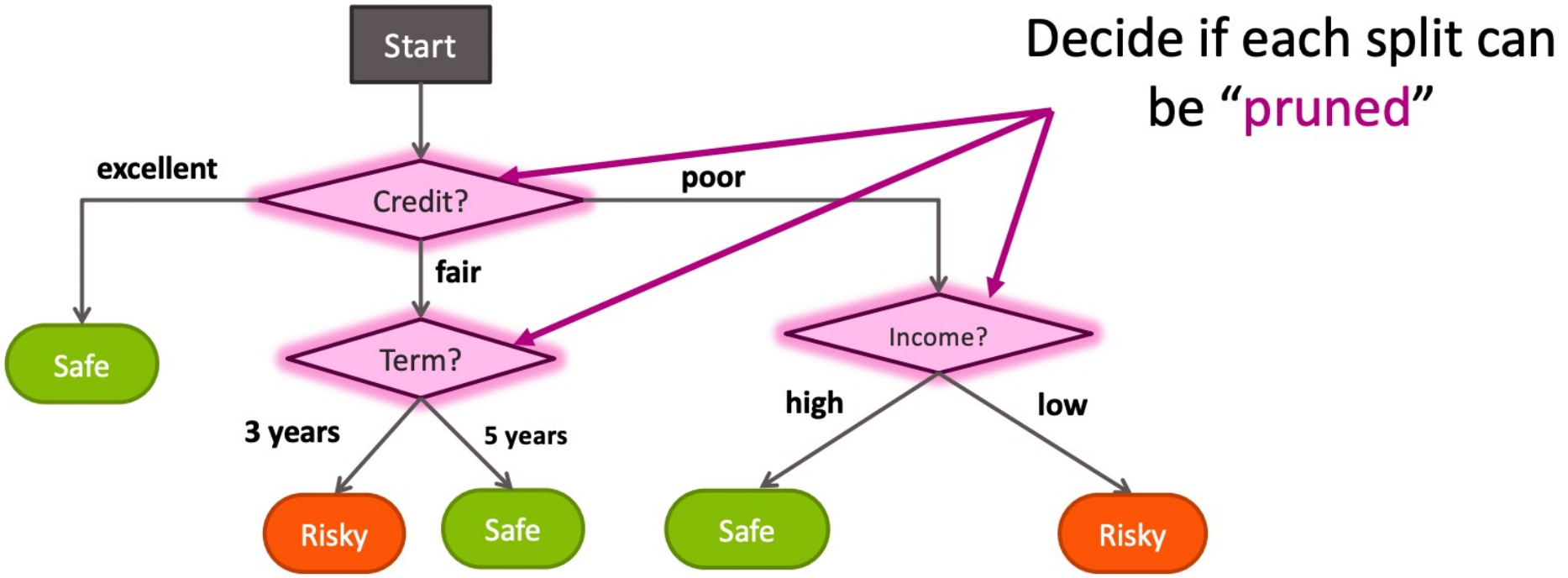
$\lambda = 0.3$

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

$C(T) = Error(T) + \lambda L(T)$

Replace split by leaf node? ← YES!

Step 5: Repeat Steps 1-4 for every split



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