CSE/STAT 416

Naïve Bayes and Decision Trees

Pre-Class Videos

Tanmay Shah
Paul G. Allen School of Computer Science & Engineering
University of Washington

July 10, 2024

Probability Classifier

Idea: Estimate probabilities $\hat{P}(y|x)$ and use those for prediction

Probability Classifier

Input x: Sentence from review

Estimate class probability $\hat{P}(y = +1|x)$

If $\hat{P}(y = +1|x) > 0.5$:

- $\hat{y} = +1$

Else:

- $\hat{y} = -1$

Notes:

Estimating the probability improves **interpretability**

Interpreting Score

$$Score(x_i) = w^T h(x_i)$$

$-\infty$ | $\hat{y}_i = -1$ | 0 | $\hat{y}_i = +1$ | ∞

Very sure $\hat{y}_i = -1$

Not sure if $\hat{y}_i = -1$ *or* $\hat{y}_i = +1$

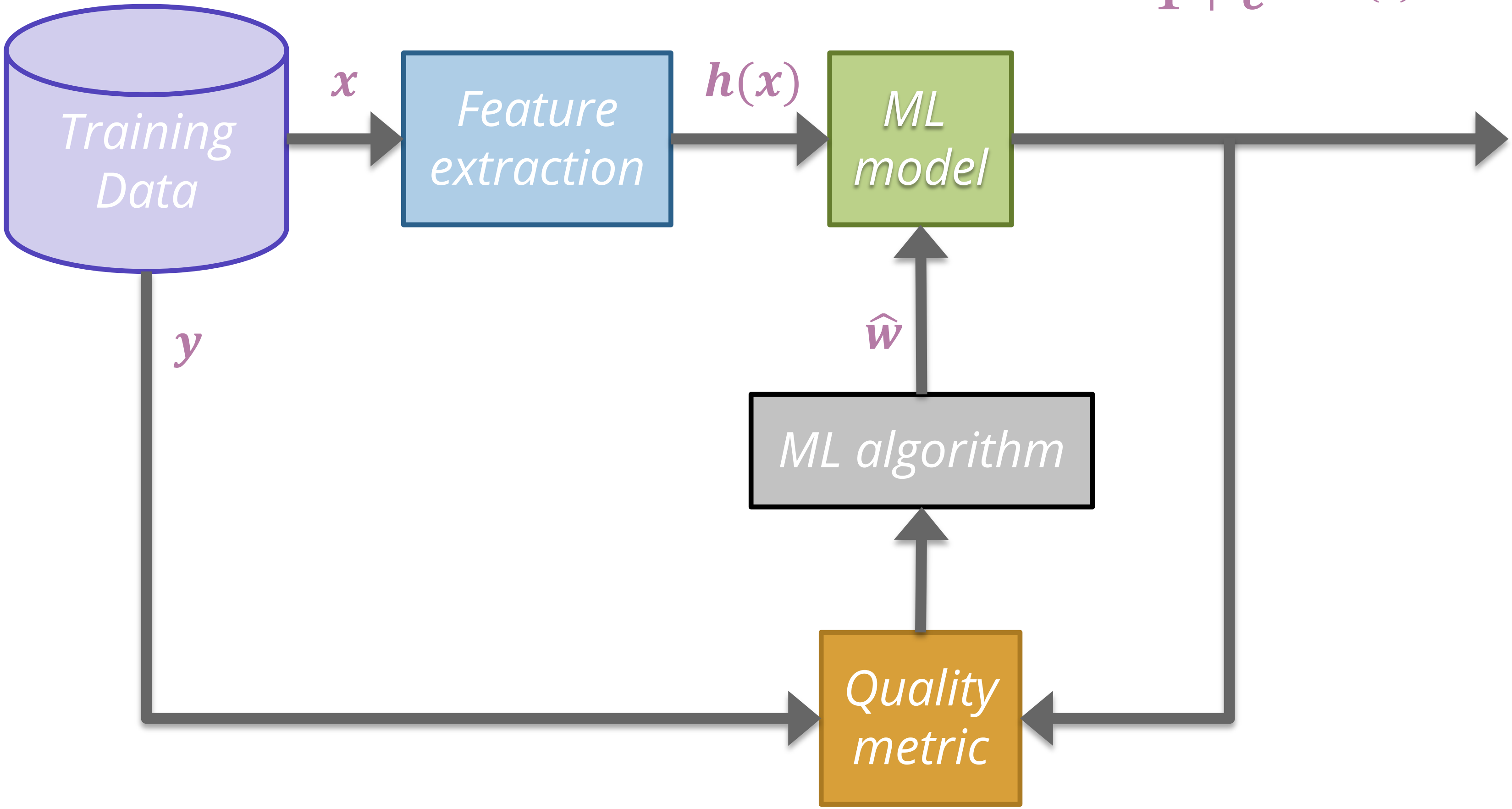
Very sure $\hat{y}_i = +1$

$\hat{P}(y_i = +1|x_i) = 0$ $\quad \hat{P}(y_i = +1|x_i) = 0.5$ $\quad \hat{P}(y_i = +1|x_i) = 1$

0 $\quad 1$

$\hat{P}(y = +1\,|x)$

$$\hat{P}(y=+1|x,\hat{w}) = sigmoid\left(\hat{w}^T h(x)\right) = \frac{1}{1+e^{-\hat{w}^T h(x)}}$$



Naïve Bayes

Idea: Naïve Bayes

$x = \text{"The sushi \& everything else was awesome!"}$

$P(y = +1 \mid x = \text{"The sushi \& everything else was awesome!"})?$

$P(y = -1 \mid x = \text{"The sushi \& everything else was awesome!"})?$

Idea: Select the class that is the most likely!

Bayes Rule:

$$P(y = +1|x) = \frac{P(x|y = +1)P(y = +1)}{P(x)}$$

Example

$$\frac{P(\text{"The sushi \&everything else was awesome!"} \mid y = +1)\, P(y = +1)}{P(\text{"The sushi \& everything else was awesome!"})}$$

Since we're just trying to find out which class has the greater probability, we can discard the divisor.

Naïve Assumption

Idea: Select the class with the highest probability!

Problem: We have not seen the sentence before.

Assumption: Words are independent from each other.

$x = \text{“}The\ sushi\ \&\ everything\ else\ was\ awesome!\text{”}$

$$\frac{P(\text{“}The\ sushi\ \&\ everything\ else\ was\ awesome!\text{”} | y = +1)\ P(y = +1)}{P(\text{“}The\ sushi\ \&\ everything\ else\ was\ awesome!\text{”})}$$

$$\begin{aligned} &P(\text{“}The\ sushi\ \&everything\ else\ was\ awesome!\text{”}\ |y = +1) \\ &= P(\text{The} \mid \text{y=+1}) * \text{P}(sushi\ |y = +1) * P(\&|y = +1) \\ &\quad * P(everything|y = +1) * P(else|y = +1) * P(was|y = +1) \\ &\quad * P(awesome|y = +1) \end{aligned}$$

Compute Probabilities

How do we compute something like

$$P(y = +1)?$$

How do we compute something like

$$P(\text{“}awesome\text{”}\ | y = +1)?$$

Zeros

If a feature is missing in a class everything becomes zero.

$$P(\text{“}The\ sushi\ \&everything\ else\ was\ awesome!\text{”}\ |y = +1)$$
$$= P(\text{The} \mid \text{y=+1}) * \text{P}(sushi\ |y = +1) * P(\&|y = +1)$$
$$* P(everything|y = +1) * P(else|y = +1) * P(was|y = +1)$$
$$* P(awesome|y = +1)$$

Solutions?

Take the log (product becomes a sum).

- Generally define $\log(0) = 0$ in these contexts

Laplacian Smoothing (adding a constant to avoid multiplying by zero)

Compare Models

Logistic Regression:

$$P(y = +1|x, w) = \frac{1}{1 + e^{-w^T h(x)}}$$

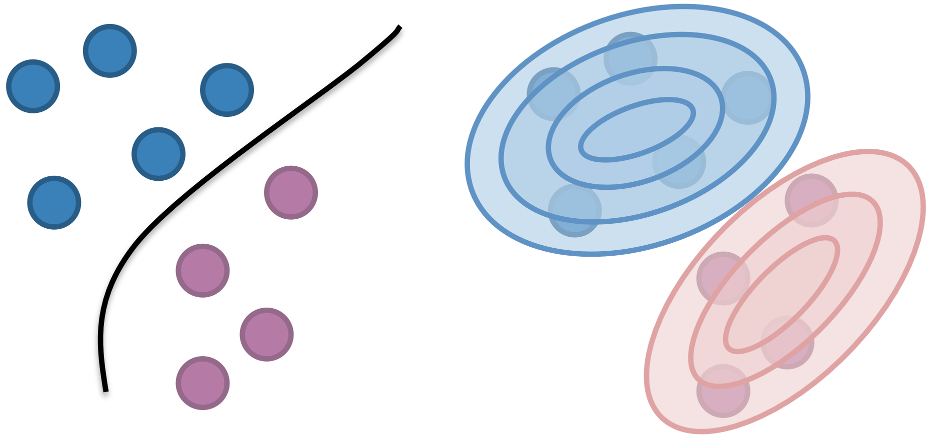
Naïve Bayes:

$$P(y|x_1, x_2, \ldots, x_d) = \prod_{j=1}^{d} P(x_j|y)\ P(y)$$

Compare Models

Generative: defines a model for generating x (e.g. Naïve Bayes)

Discriminative: only cares about defining and optimizing a decision boundary (e.g. Logistic Regression)



CSE/STAT 416

Naïve Bayes and Decision Trees

Tanmay Shah
Paul G. Allen School of Computer Science & Engineering
University of Washington

July 10, 2024

❓ **Questions?** Raise hand or **sli.do #cs416**
💬 **Before Class:** Pro-rain or anti-rain person?
🎵 **Listening to:** Alvvays

Administrivia

- Midterm next week
  - Released Monday July 15, 8 AM
  - Due Wednesday July 17, 11:59 PM
  - Timed (3 hr)
  - Open book, individual submissions only

Compare Models

Logistic Regression:

$$P(y = +1|x, w) = \frac{1}{1 + e^{-w^T h(x)}}$$

Naïve Bayes:

$$P(y|x_1, x_2, \dots, x_d) = \prod_{j=1}^{d} P(x_j|y)\ P(y)$$

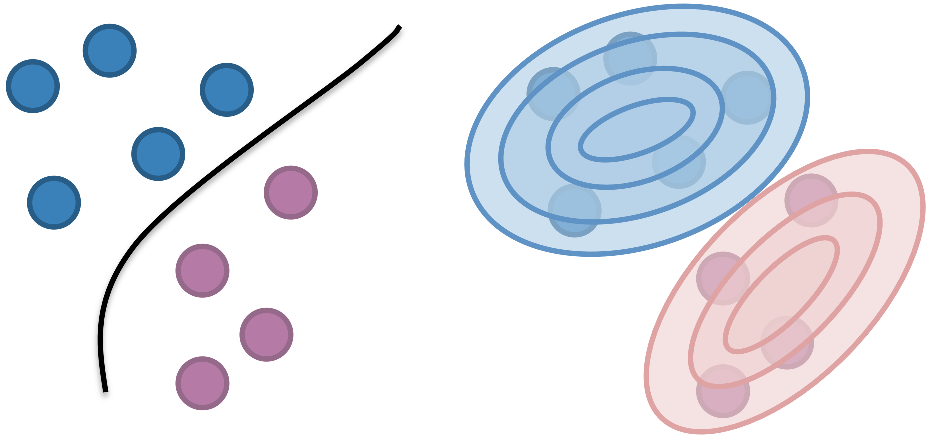
Based on counts of words/classes

- Laplace Smoothing

Compare Models

Generative: defines a model for generating x (e.g. Naïve Bayes)

Discriminative: only cares about defining and optimizing a decision boundary (e.g. Logistic Regression)



slido

Group

2 min

sli.do #cs416

Recap: What is the predicted class for this sentence assuming we have the following training set (no Laplace Smoothing).
“he is not cool”

Sentence	Label
this dog is cute	Positive
he does not like dogs	Negative
he is not bad he is cool	Positive

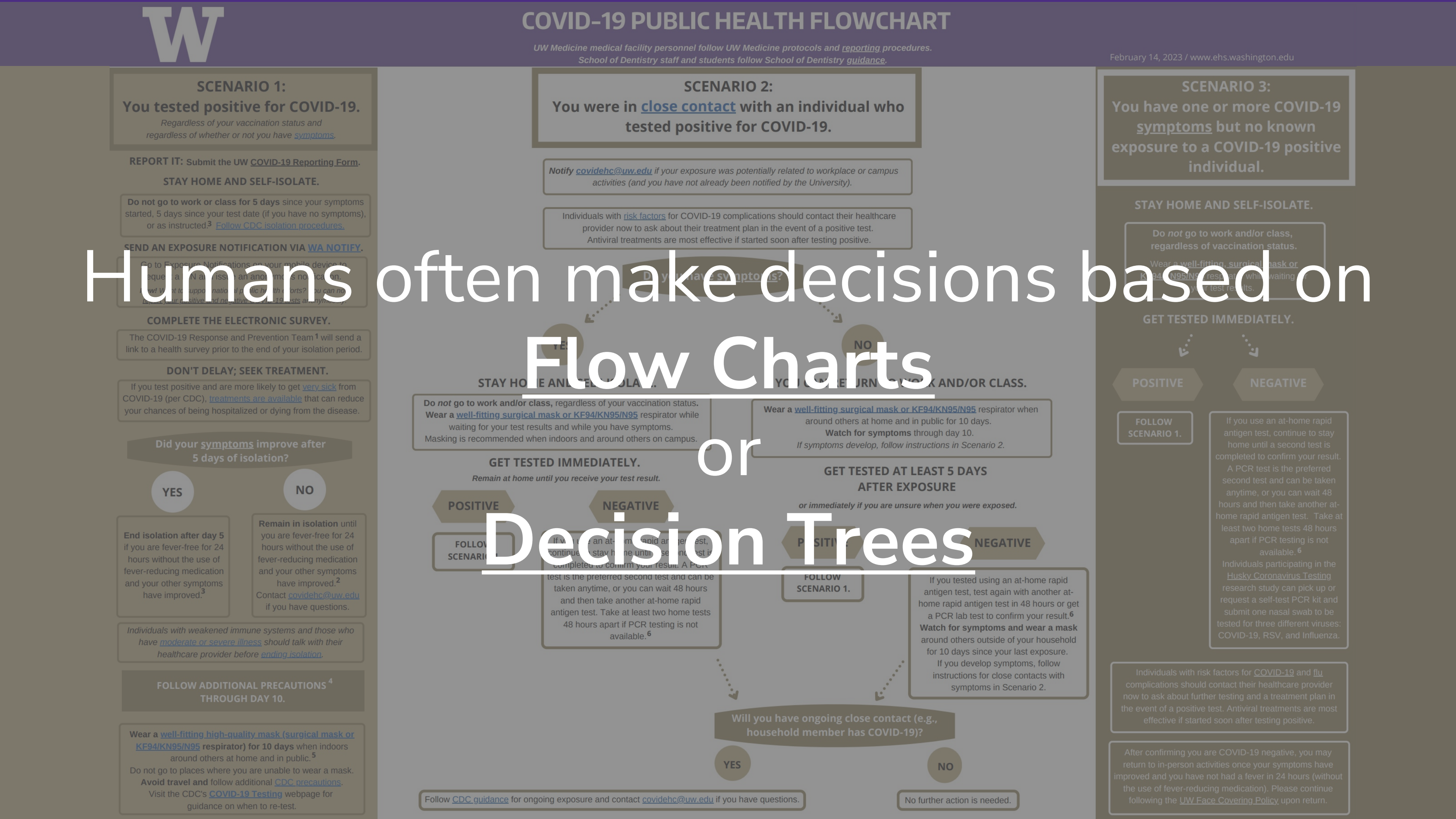
Decision Trees

Humans often make decisions based on

Flow Charts

or

Decision Trees



Parametric vs. Non-Parametric Methods

Parametric Methods: make assumptions about the data distribution

- Linear Regression ⇒ assume the data is linear
- Logistic Regression ⇒ assume probability has the shape of of a logistic curve and linear decision boundary
- Those assumptions result in a *parameterized* function family. Our learning task is to learn the parameters.

Non-Parametric Methods: (mostly) don't make assumptions about the data distribution

- Decision Trees, k-NN (soon)
- We're still learning something, but not the parameters to a function family that we're assuming describes the data.
- Useful when you don't want to (or can't) make assumptions about the data distribution.

XOR

A line might not always support our decisions.

What makes a loan risky?

I want to buy a new house!

Loan Application

Credit History
★★★★

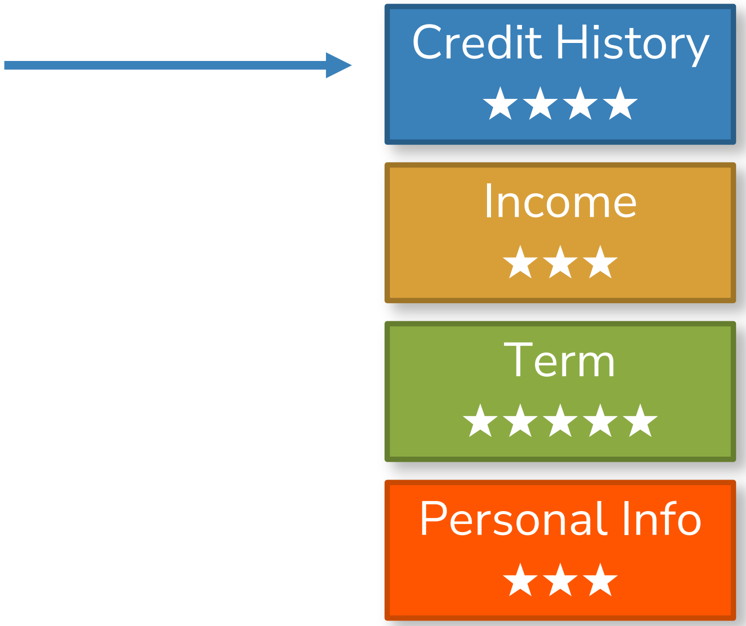
Income
★★★

Term
★★★★★

Personal Info
★★★

Credit history explained

Did I pay previous loans on time?



Example:

excellent, good, or fair

Income

What's my income?

Example:
$80K per year

Credit History
★★★★

Income
★★★

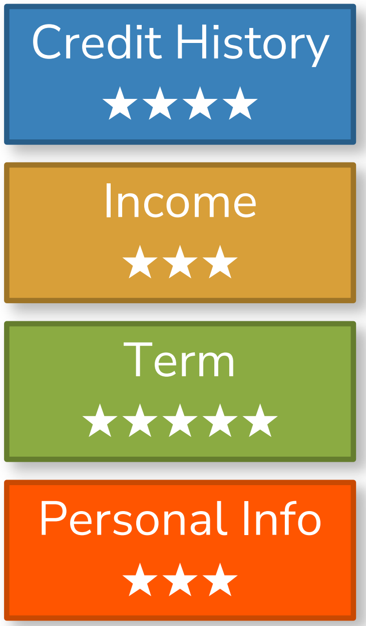
Term
★★★★★

Personal Info
★★★

Loan terms

How soon do I need to pay the loan?

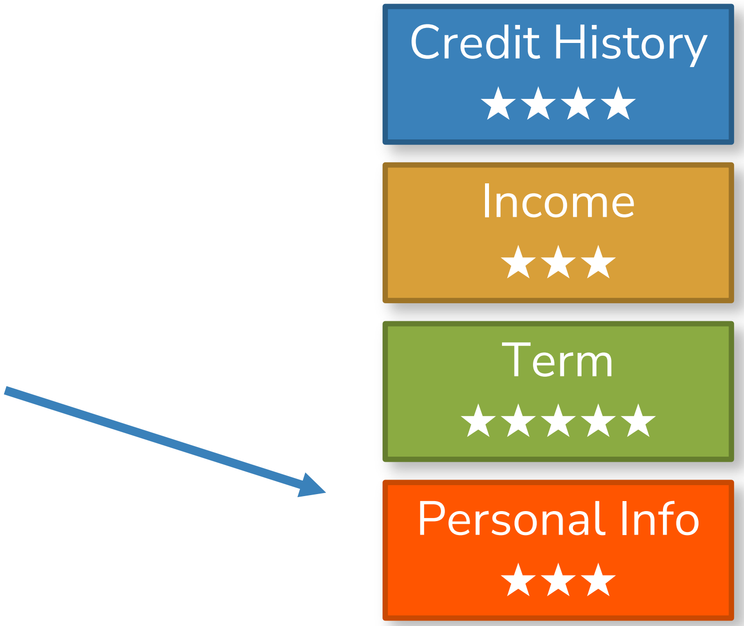
Example: 3 years, 5 years,...



Personal information

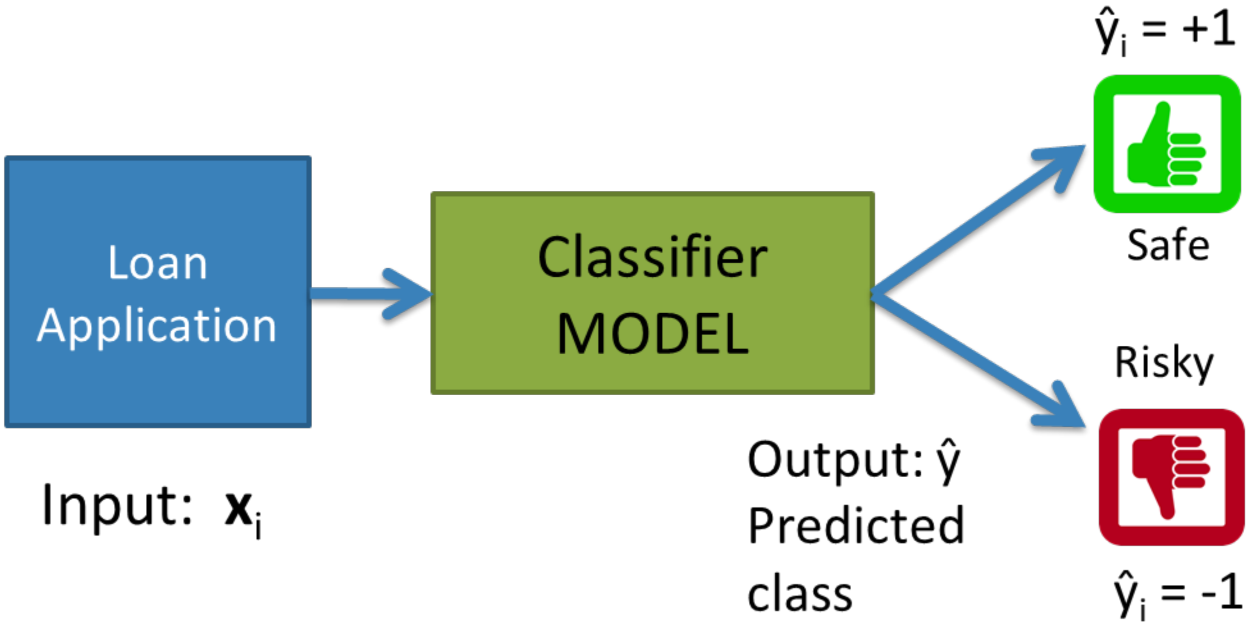
Age, reason for the loan, marital status,...

Example: Home loan for a married couple



Intelligent application

Classifier review



Input: $\mathbf{x}_i$

Output: $\hat{y}$
Predicted class

Setup

Data (N observations, 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	safe
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

Evaluation: classification error

Many possible decisions: number of trees grows exponentially!

Poll Everywhere

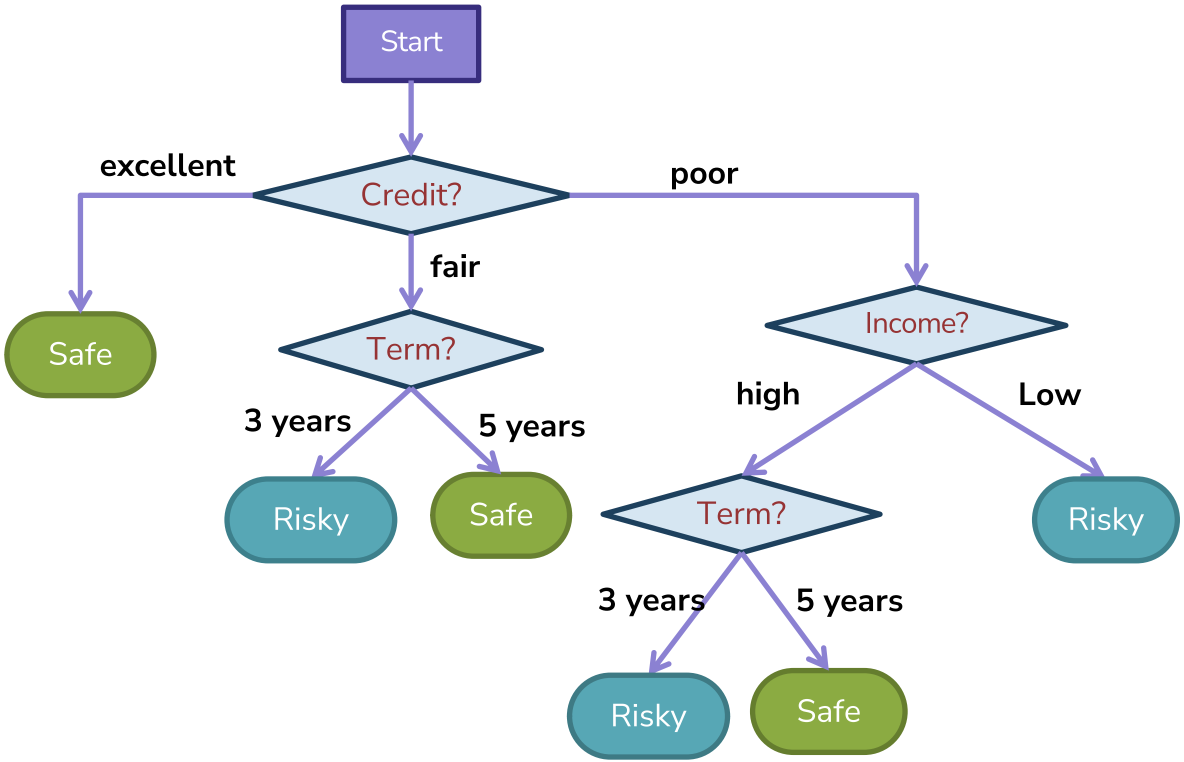
Think

2 min

With our discussion of bias and fairness from last week, discuss the potential biases and fairness concerns that might be present in our dataset about loan safety.

p No poll 5

Decision Trees



- **Branch/Internal node:** splits into possible values of a feature
- **Leaf node:** final decision (the class value)

Brain Break

Growing Trees

Visual Notation

Loan status: Safe Risky

Root
6 3

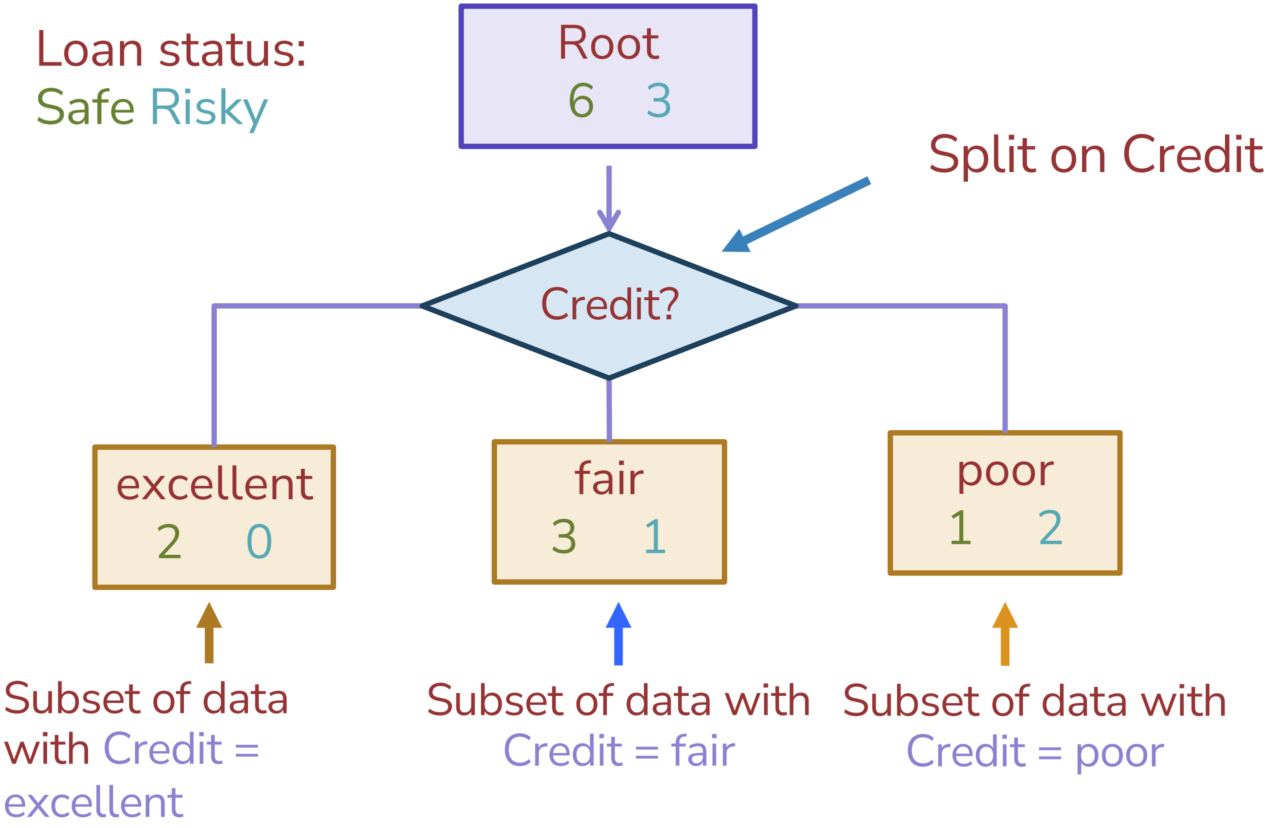
# of Risky loans

# of Safe loans

N = 9 examples

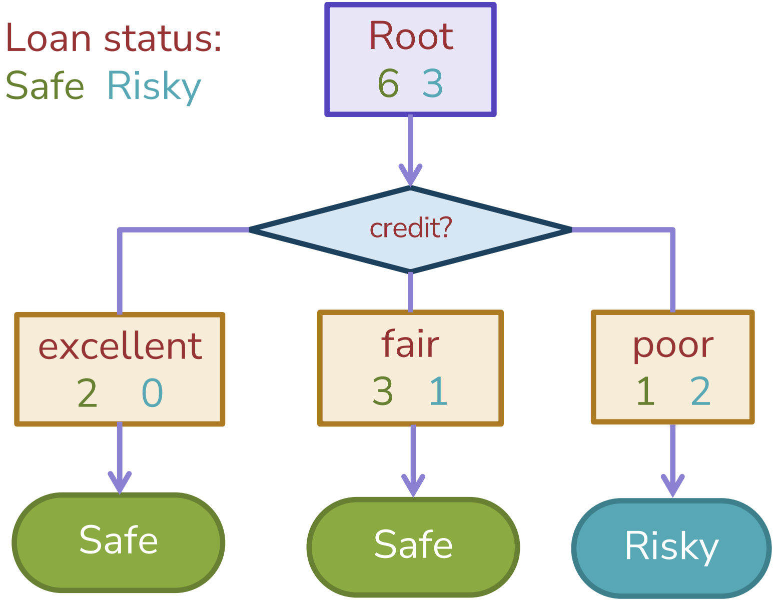
Decision stump: 1 level

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	safe
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe



Making predictions

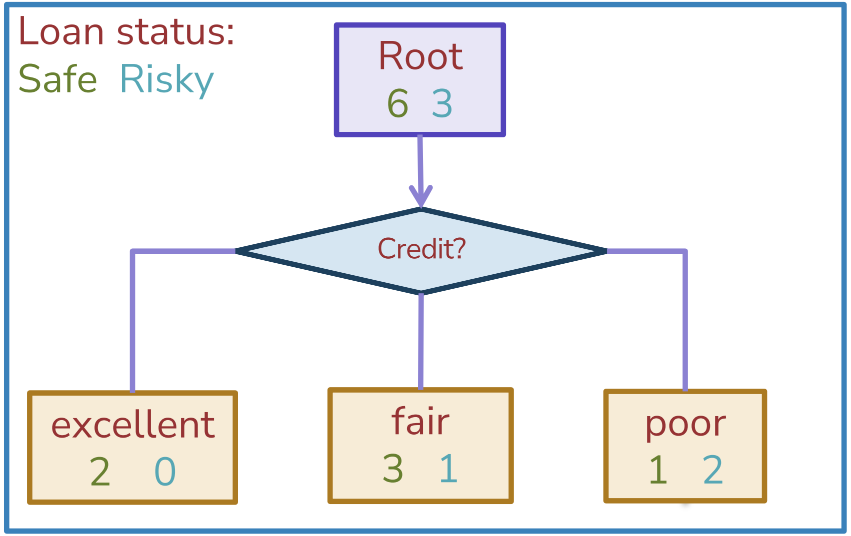
For each leaf node, set $\hat{y}$ = majority value



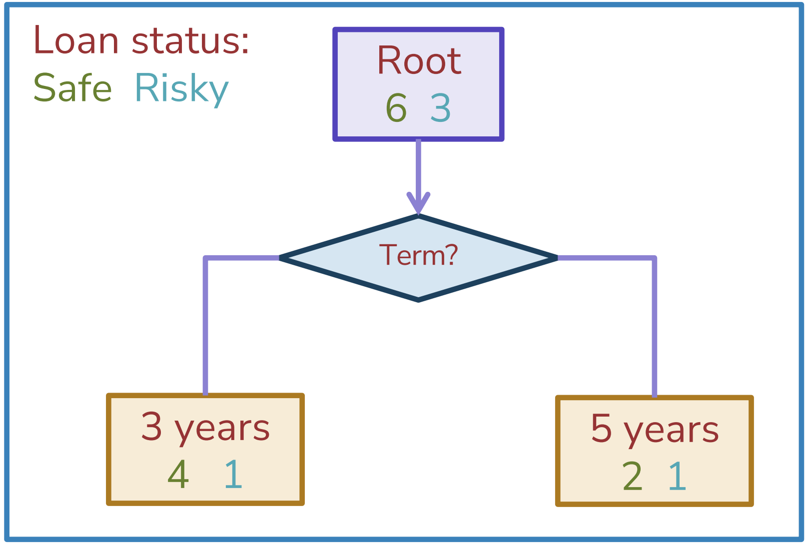
How do we select the best feature?

- Select the split with lowest classification error

Choice 1: Split on Credit



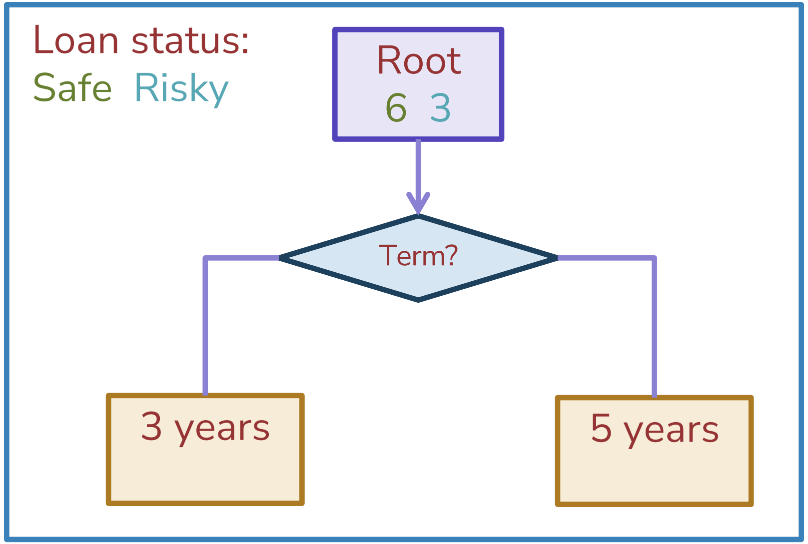
Choice 2: Split on Term



Calculate the node values.

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	safe
fair	5 yrs	low	safe
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe

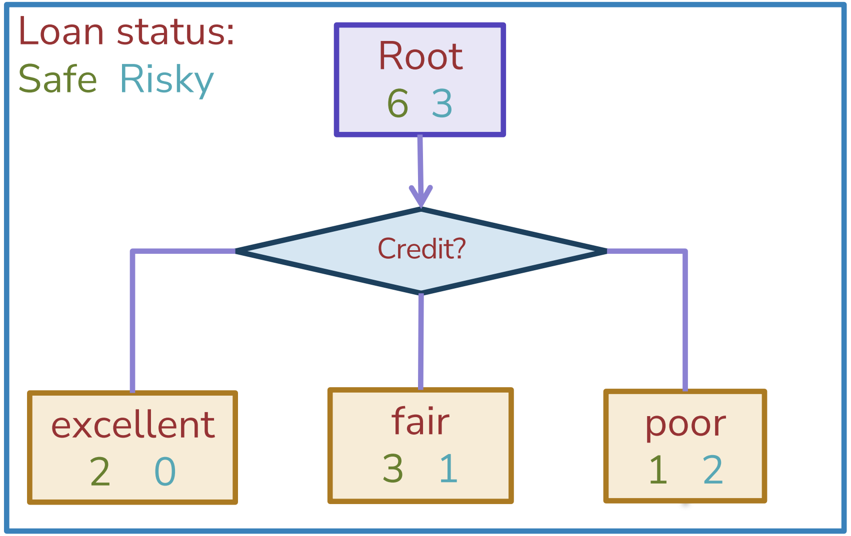
Choice 2: Split on Term



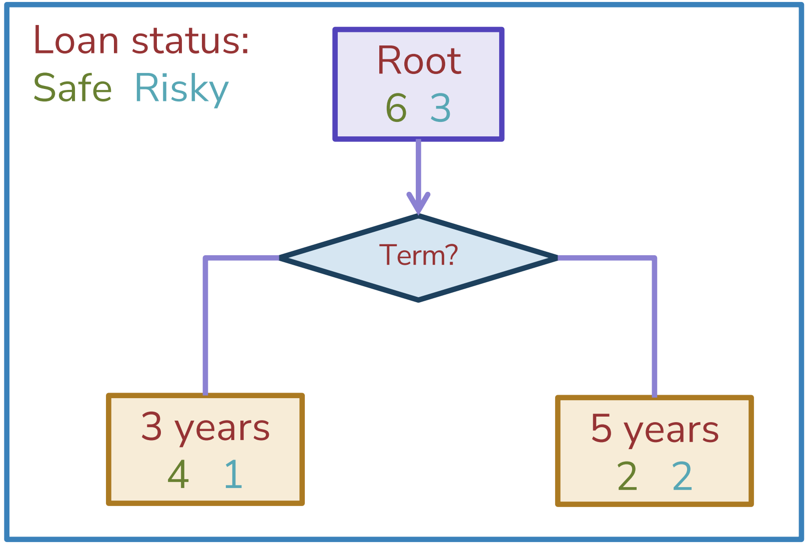
How do we select the best feature?

Select the split with lowest classification error

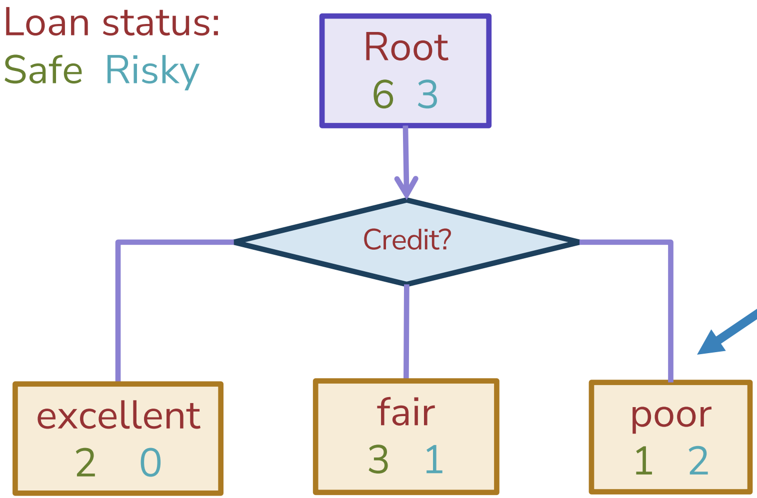
Choice 1: Split on Credit



Choice 2: Split on Term



How do we measure effectiveness of a split?



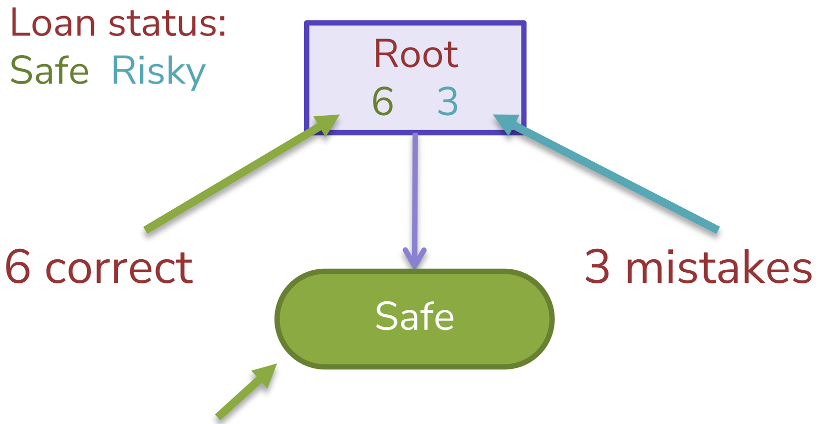
Idea: Calculate classification error of this decision stump

$$\text{Error} = \frac{\text{\# mistakes}}{\text{\# data points}}$$

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



$\hat{y}$ = majority class

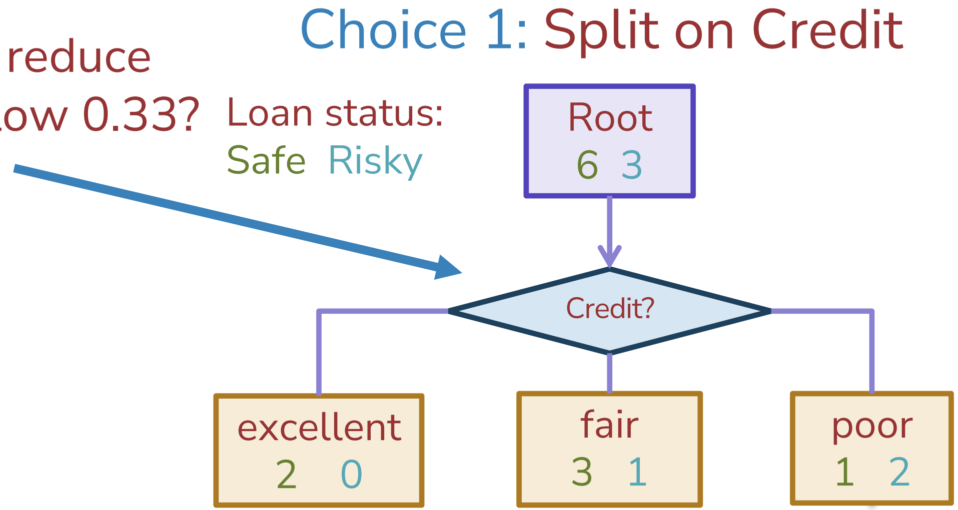
Error = ______

=

Tree	Classification error
(root)	0.33

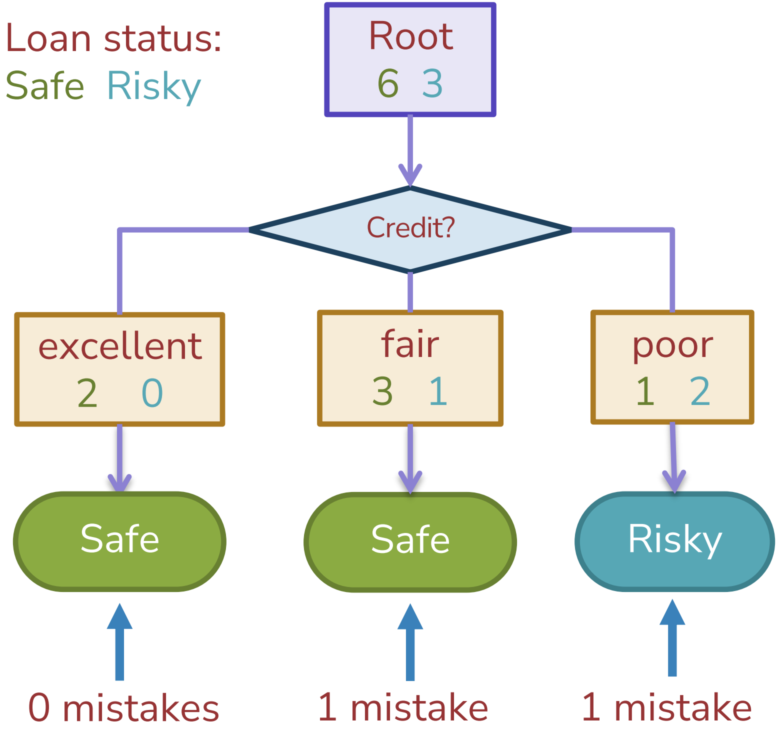
Choice 1: Split on Credit history?

Does a split on Credit reduce classification error below 0.33?



Split on Credit: Classification error

Choice 1: Split on Credit

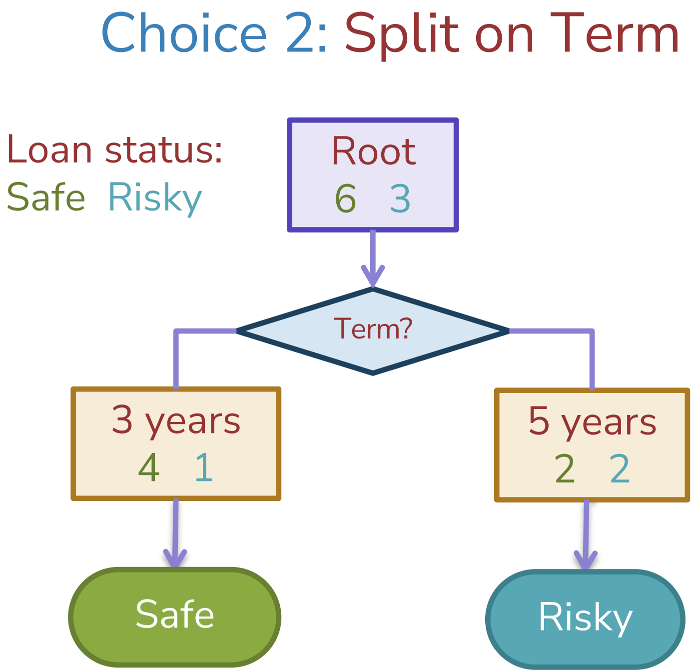


Error = ______

=

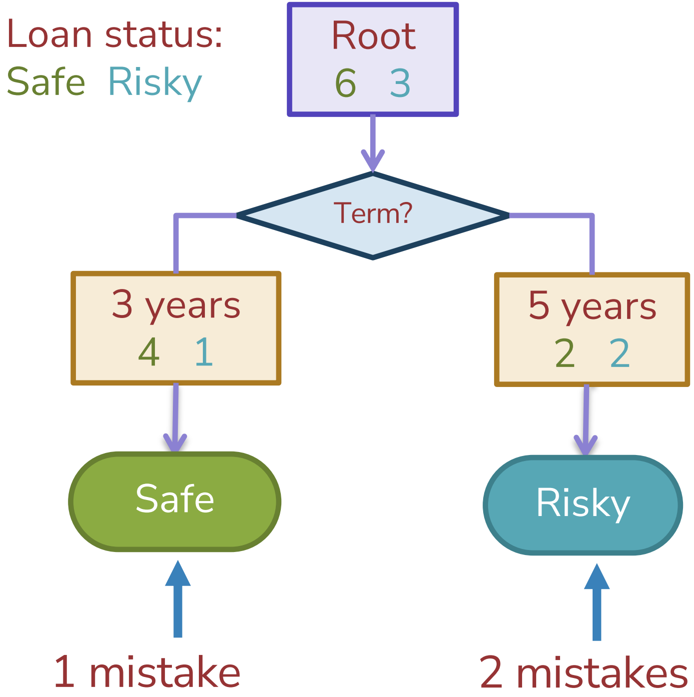
Tree	Classification error
(root)	0.33
Split on credit	0.22

Choice 2: Split on Term?



Evaluating the split on Term

Choice 2: Split on Term



Error = ______

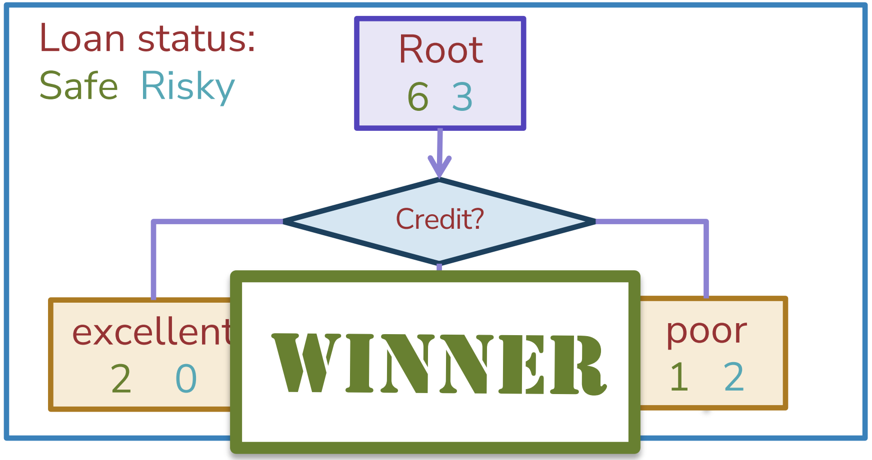
=

Tree	Classification error
(root)	0.33
Split on credit	0.22
Split on term	0.33

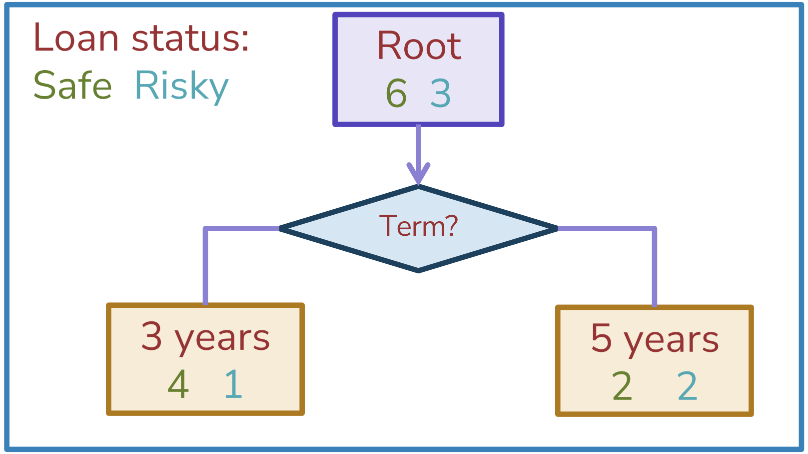
Choice 1 vs Choice 2: Comparing split on credit vs term

Tree	Classification error
(root)	0.33
split on credit	0.22
split on loan term	0.33

Choice 1: Split on Credit



Choice 2: Split on Term



Split Selection

Split(node)

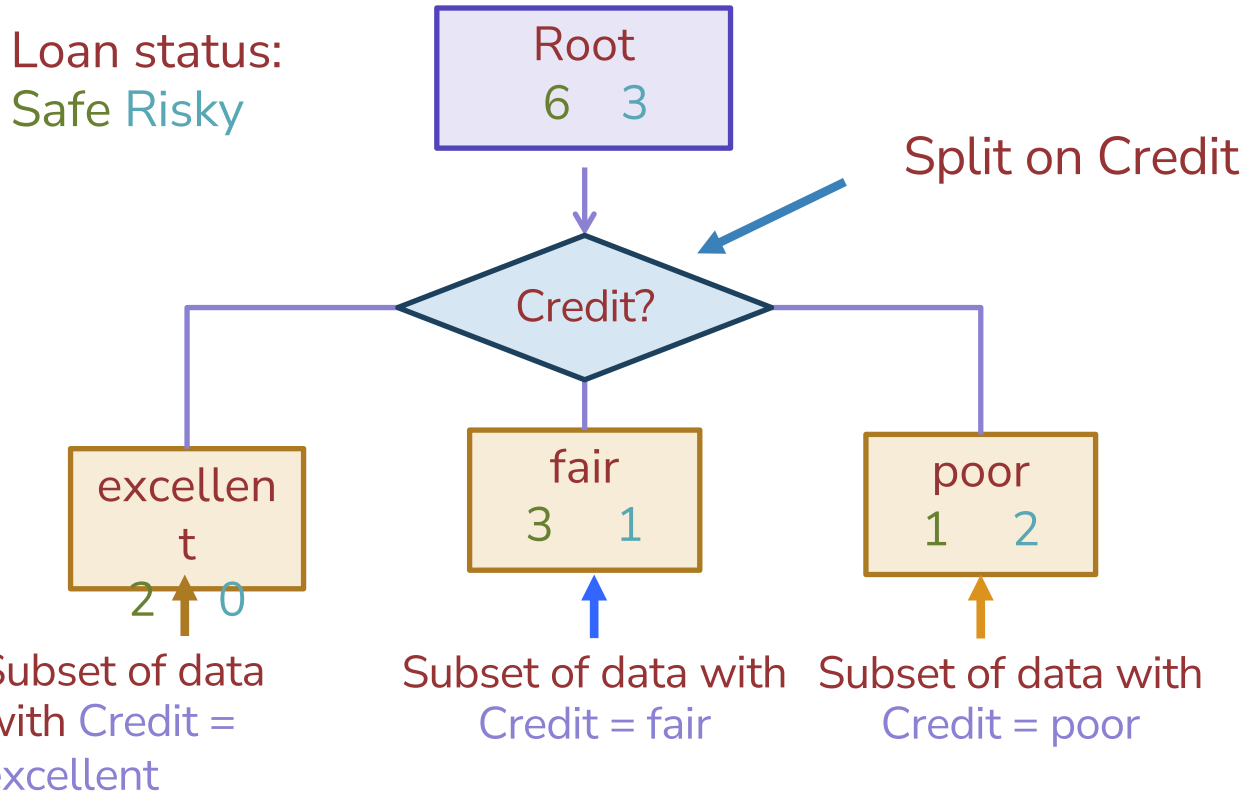
- Given M, the subset of training data at a node
- For each (remaining) feature $h_j(x)$:
  - Split data M on feature $h_j(x)$
  - Compute the classification error for the split
- Chose feature $h_j^*(x)$ with the lowest classification error

Greedy & Recursive Algorithm

BuildTree(node)

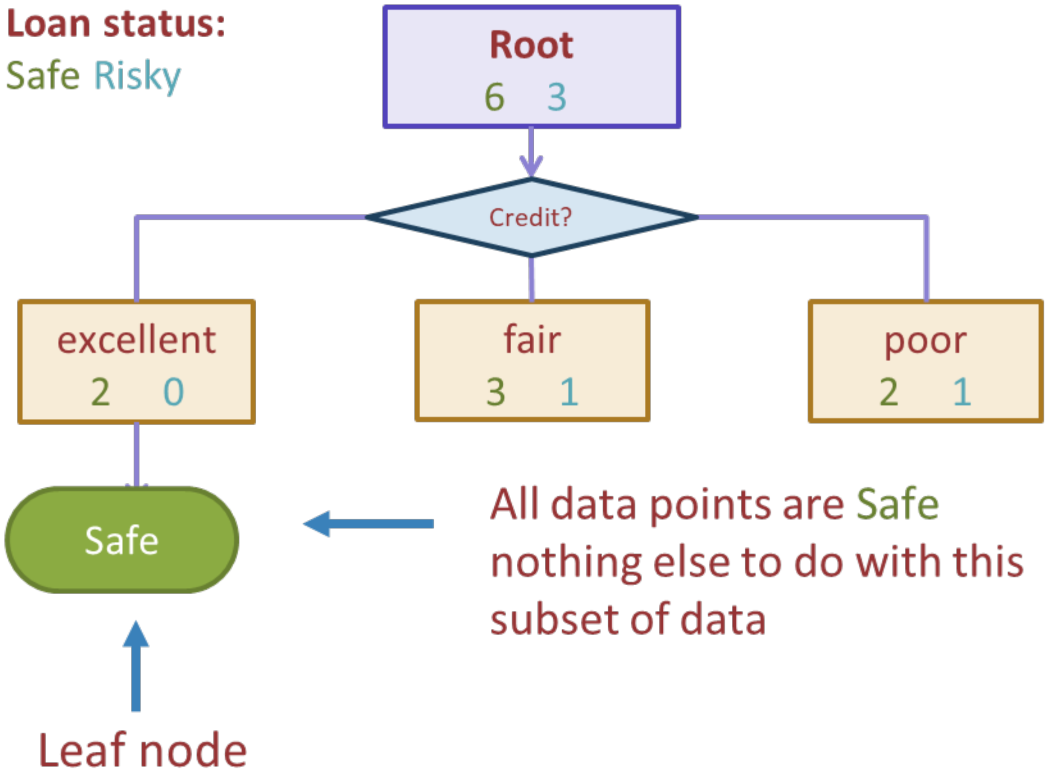
- If termination criterion is met:
  - Stop
- Else:
  - Split(node)
  - For child in node:
    - BuildTree(child)

Decision stump: 1 level

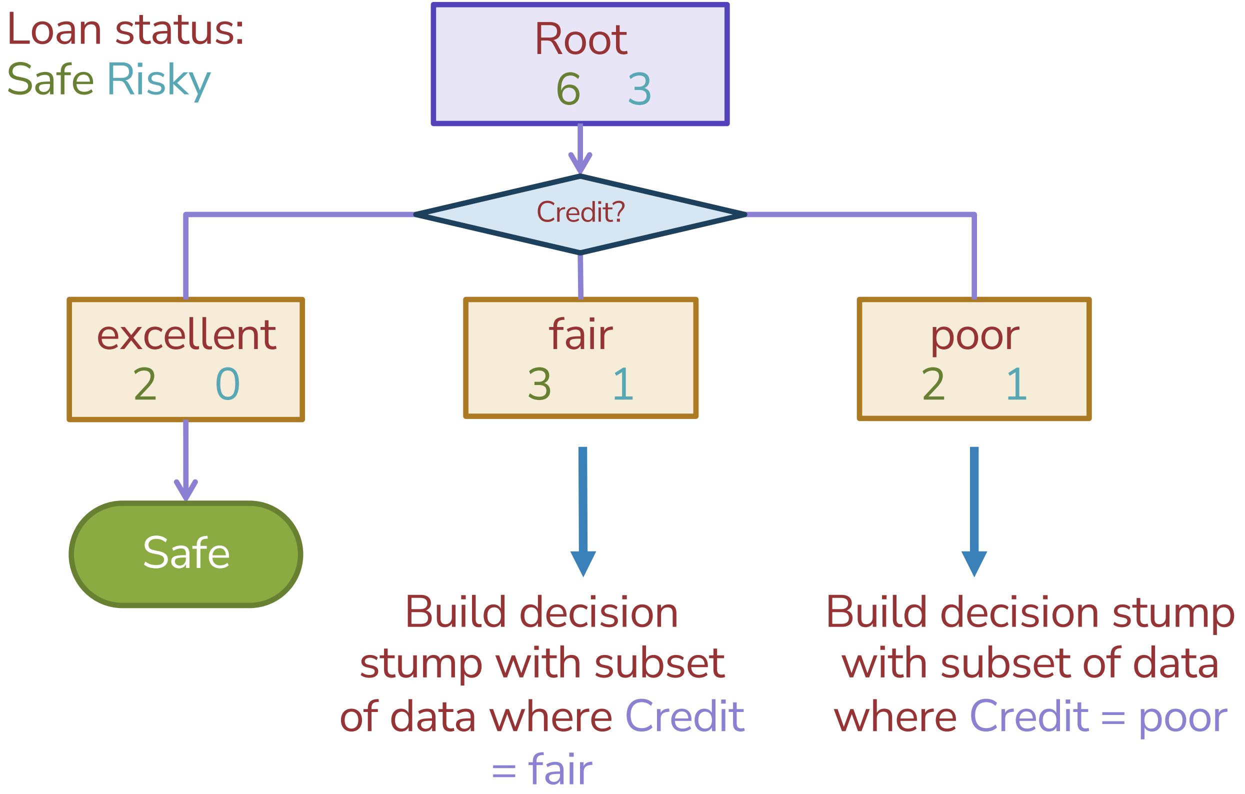


Stopping

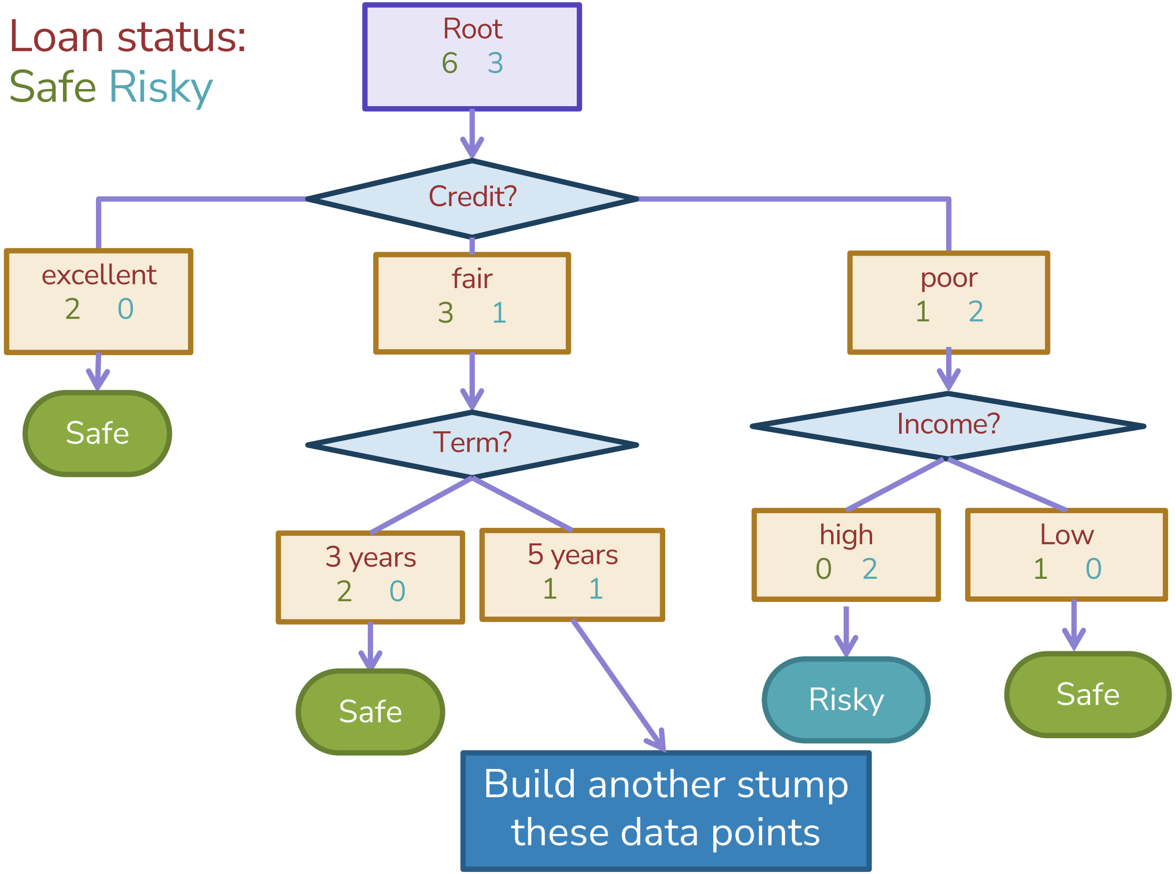
For now: Stop when all points are in one class



Tree learning = Recursive stump learning



Second level



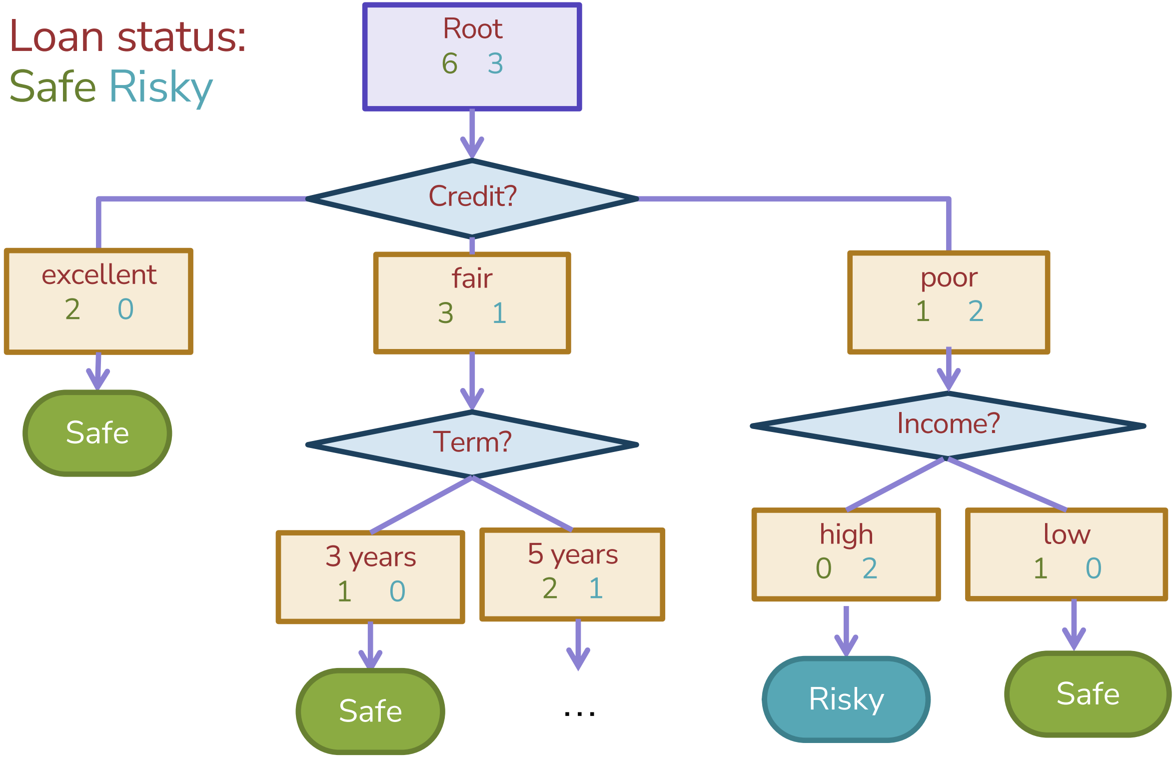
slido

Think

1 min

What predictions **should** the below decision tree output for the following datapoints?

Credit	Term	Income
excellent	5 yrs	high
fair	3 yrs	low
poor	5 yrs	(missing)



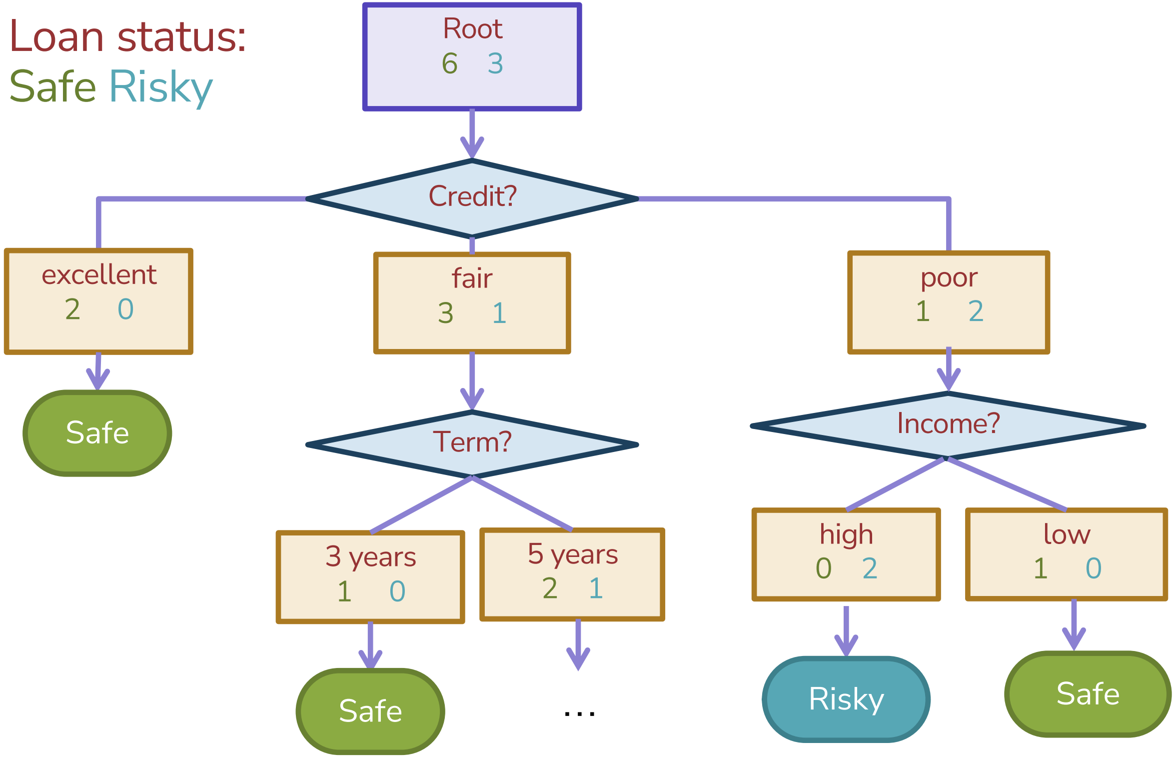
slido

Group

2 min

What predictions **should** the below decision tree output for the following datapoints?

Credit	Term	Income
excellent	5 yrs	high
fair	3 yrs	low
poor	5 yrs	(missing)

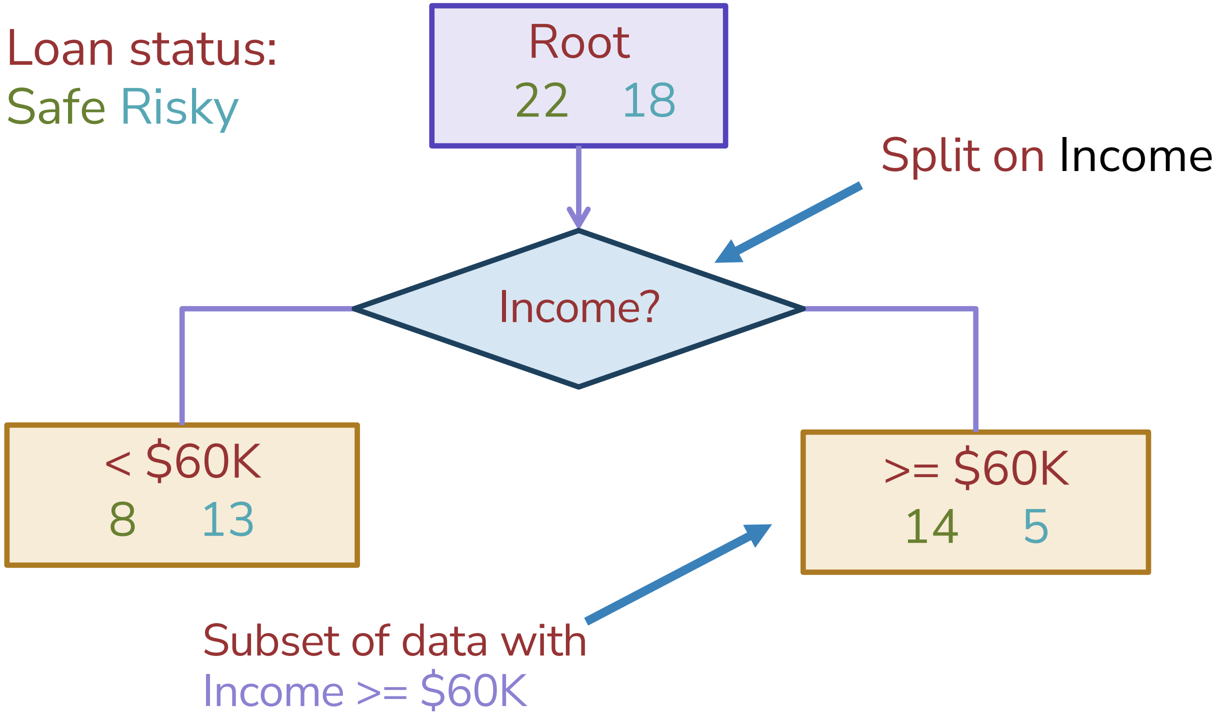


Brain Break

Real valued features

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

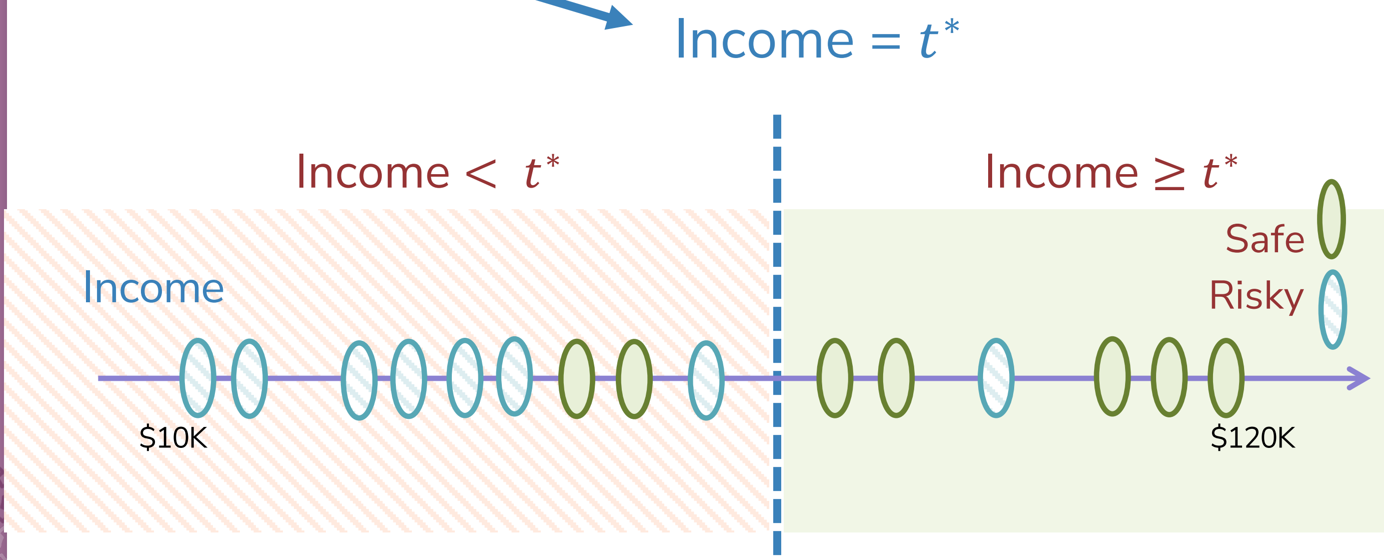
Threshold split



Best threshold?

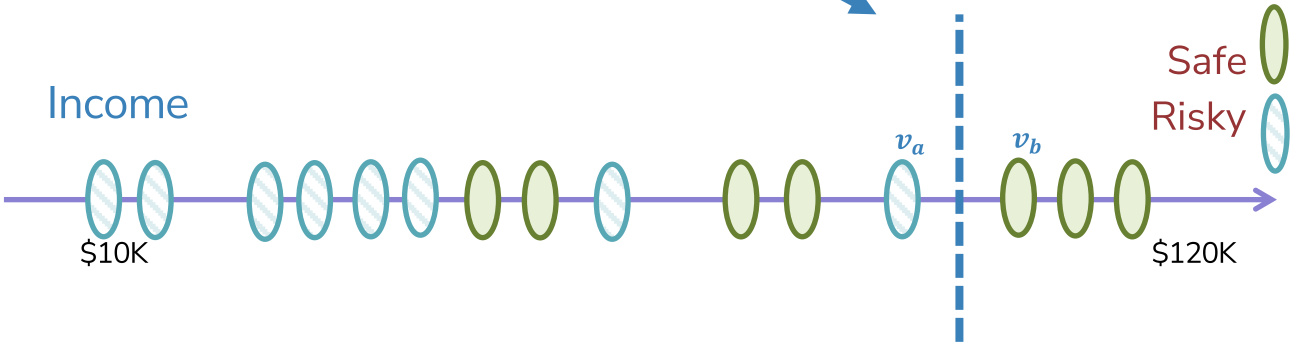
Similar to our simple, threshold model when discussing Fairness!

Infinite possible values of $\boldsymbol{t}$



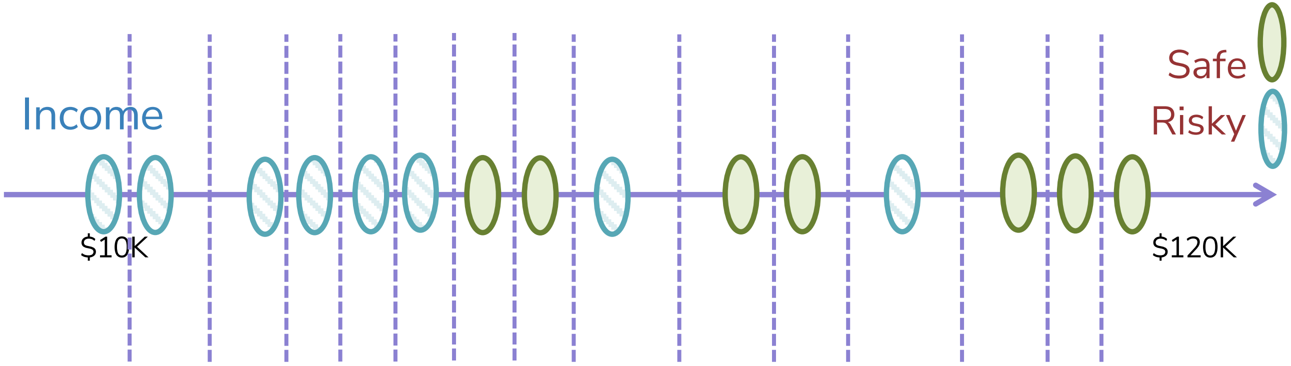
Threshold between points

Same classification error for any threshold split between v_a and v_b



Only need to consider mid-points

Finite number of splits to consider



Threshold split selection algorithm

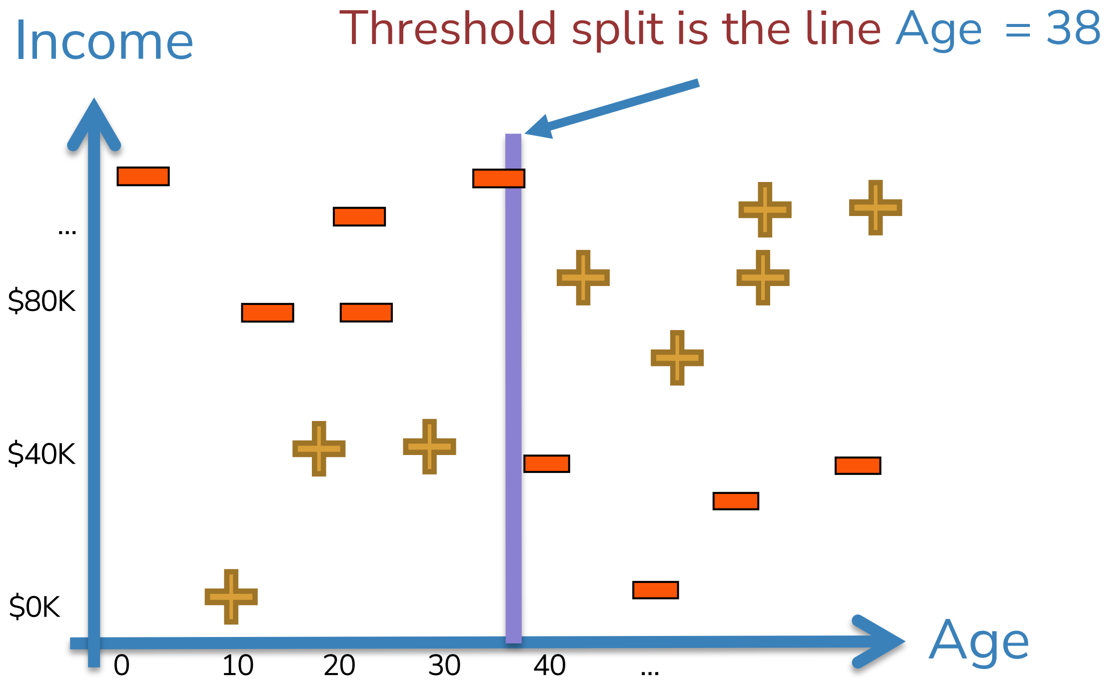
Step 1: Sort the values of a feature $h_j(x)$:

Let $[v_1, v_2, \ldots, v_N]$ denote sorted values

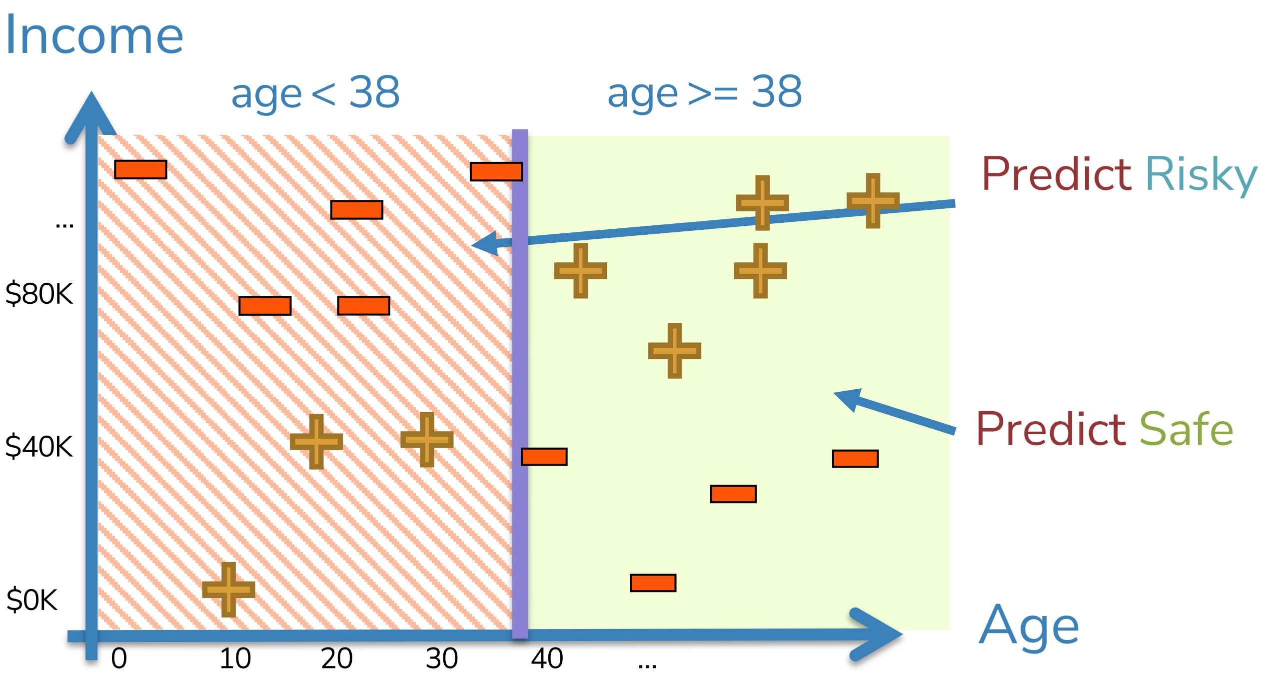
Step 2:

- For $i = [1, \ldots, N - 1]$
  - Consider split $t_i = \frac{v_i + v_{i+1}}{2}$
  - Compute classification error for threshold split $h_j(x) \geq t_i$
- Chose the $\boldsymbol{t}^*$ with the lowest class. error

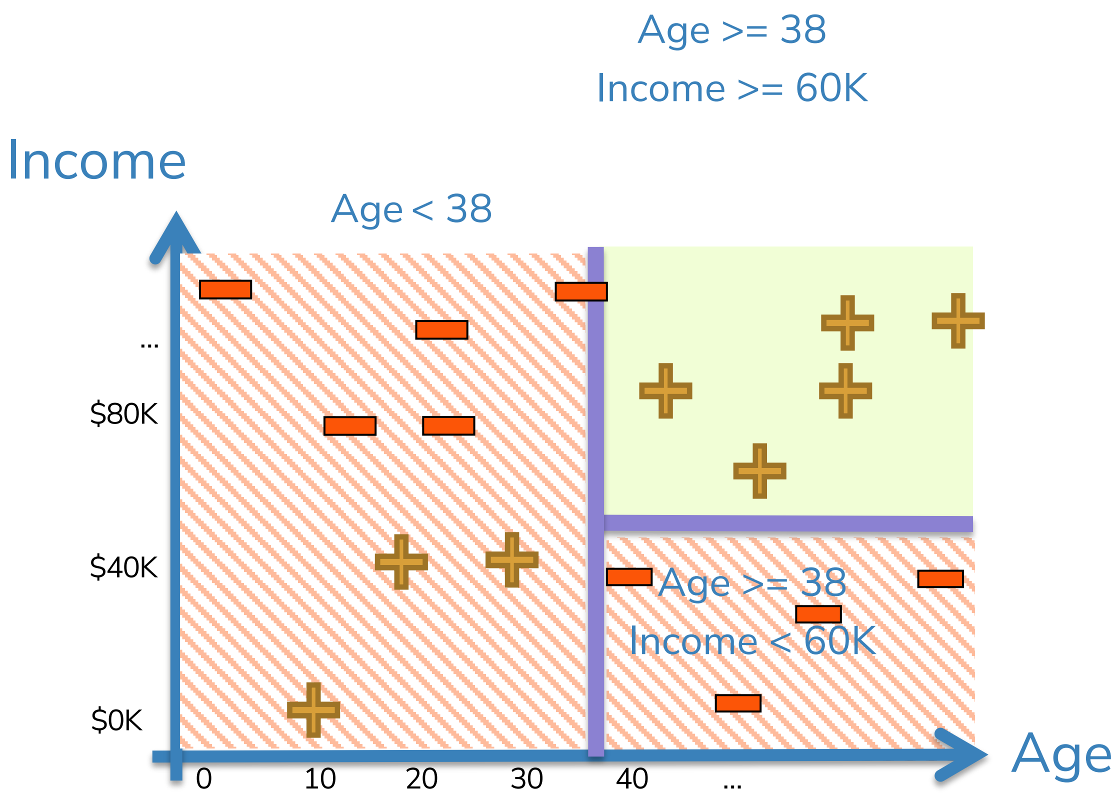
Visualizing the threshold split



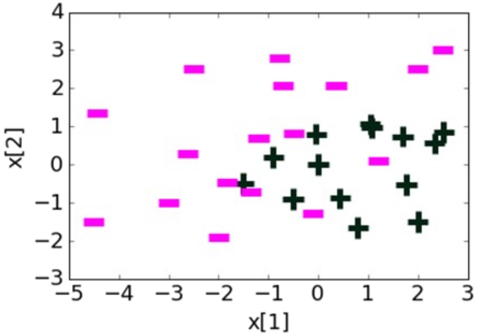
Split on Age >= 38

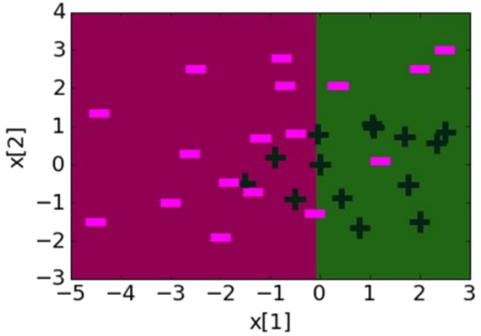


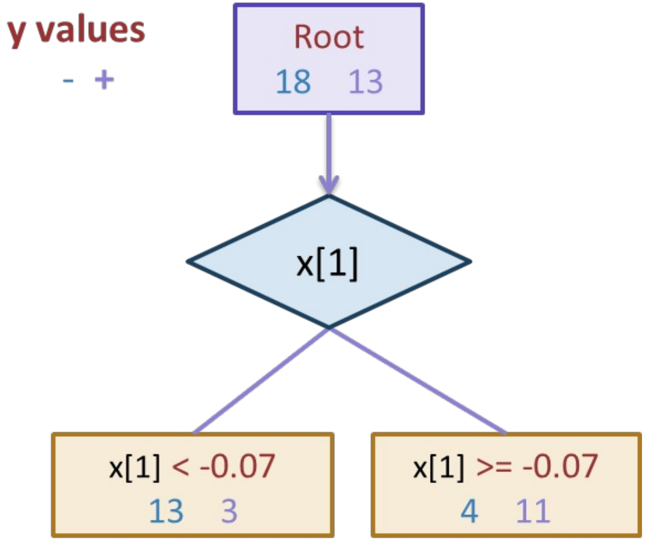
Each split partitions the 2-D space



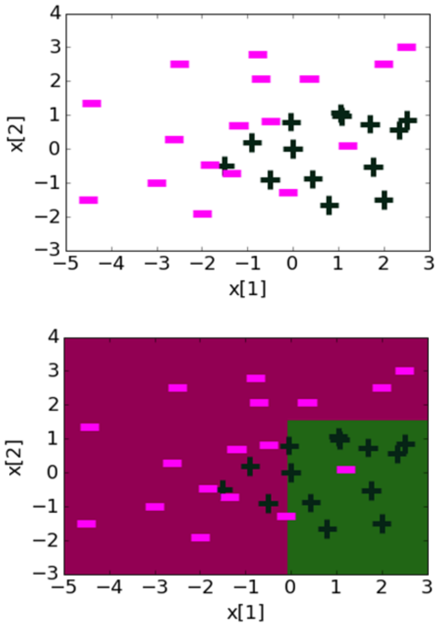
Depth 1:
Split on x[1]



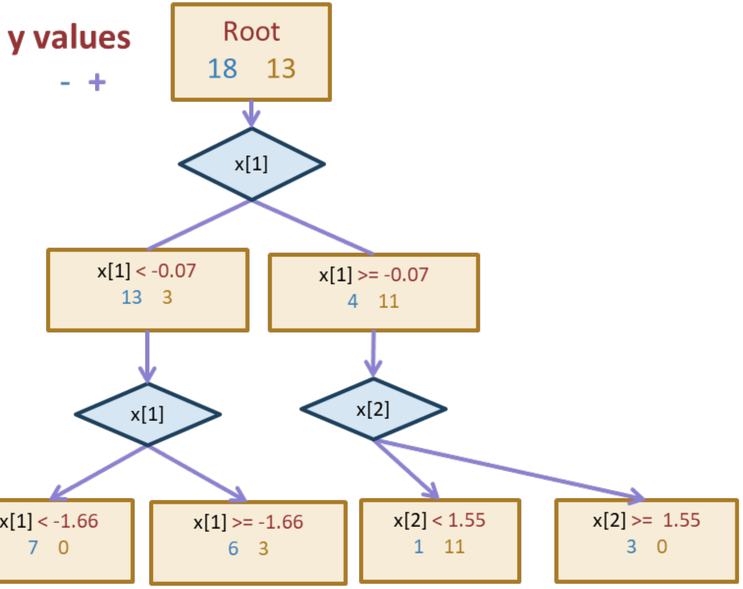




Depth 2

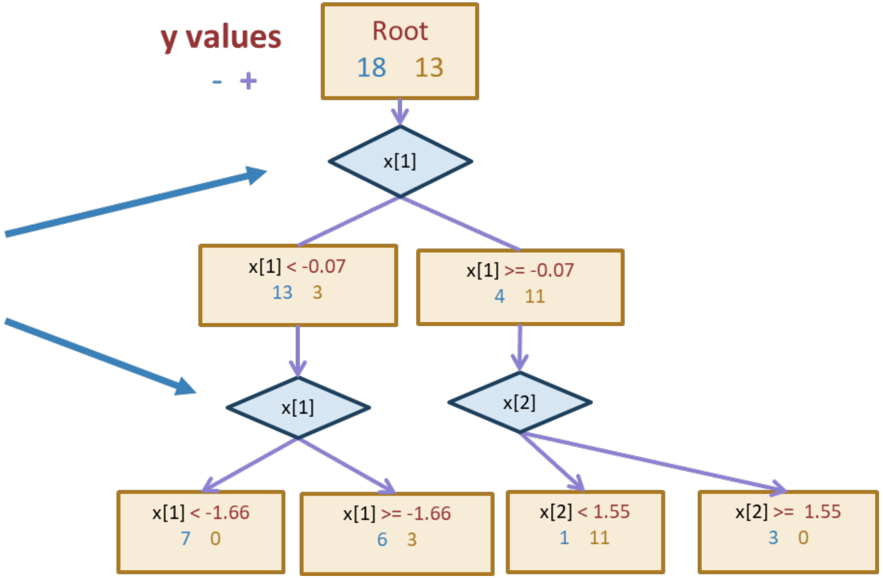


x[1]
x[2]



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

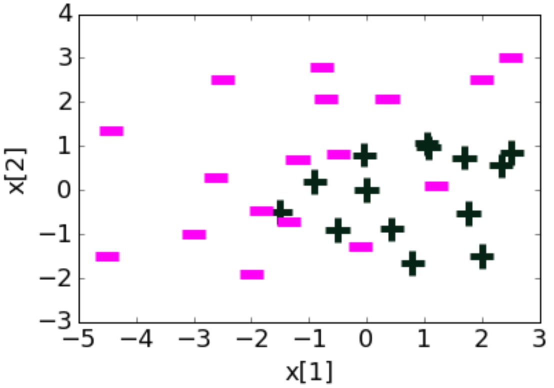
Threshold split caveat

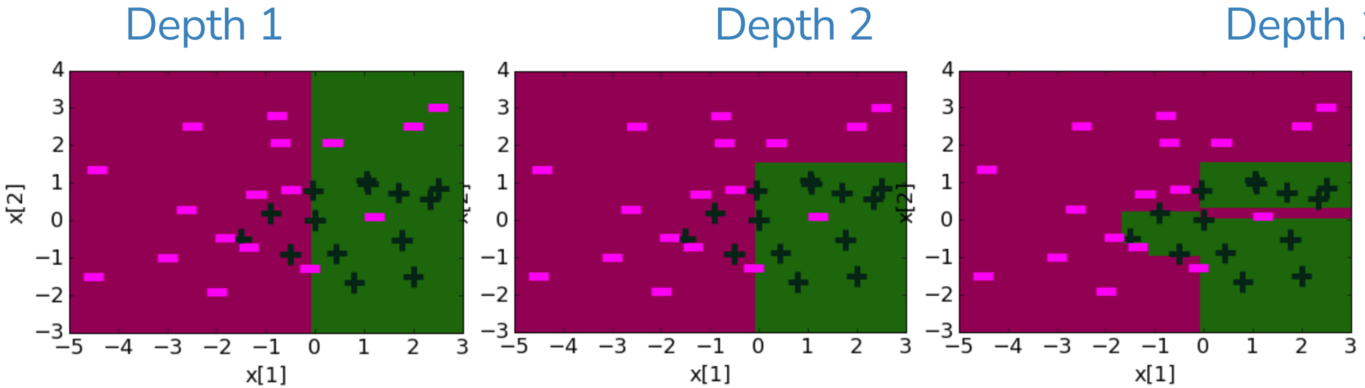


For threshold splits, same feature can be used multiple times

Decision boundaries

Decision boundaries can be complex!





Overfitting

Deep decision trees are prone to overfitting

- Decision boundaries are interpretable but not stable
- Small change in the dataset leads to big difference in the outcome

Overcoming Overfitting:

- Stop when tree reaches certain height (e.g., 4 levels)
- Stop when leaf has ≤ some num of points (e.g., 20 pts)
  - Will be the stopping condition for HW
- Stop if split won't significantly decrease error by more than some amount (e.g., 10%)

Other methods include growing full tree and pruning back

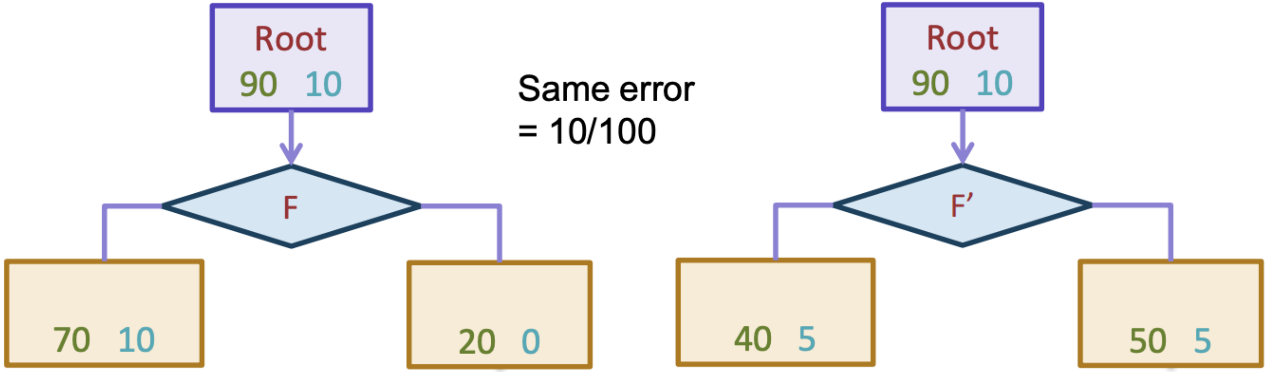
Fine-tune hyperparameters with validation set or CV

In Practice

Trees can be used for classification or regression (CART)

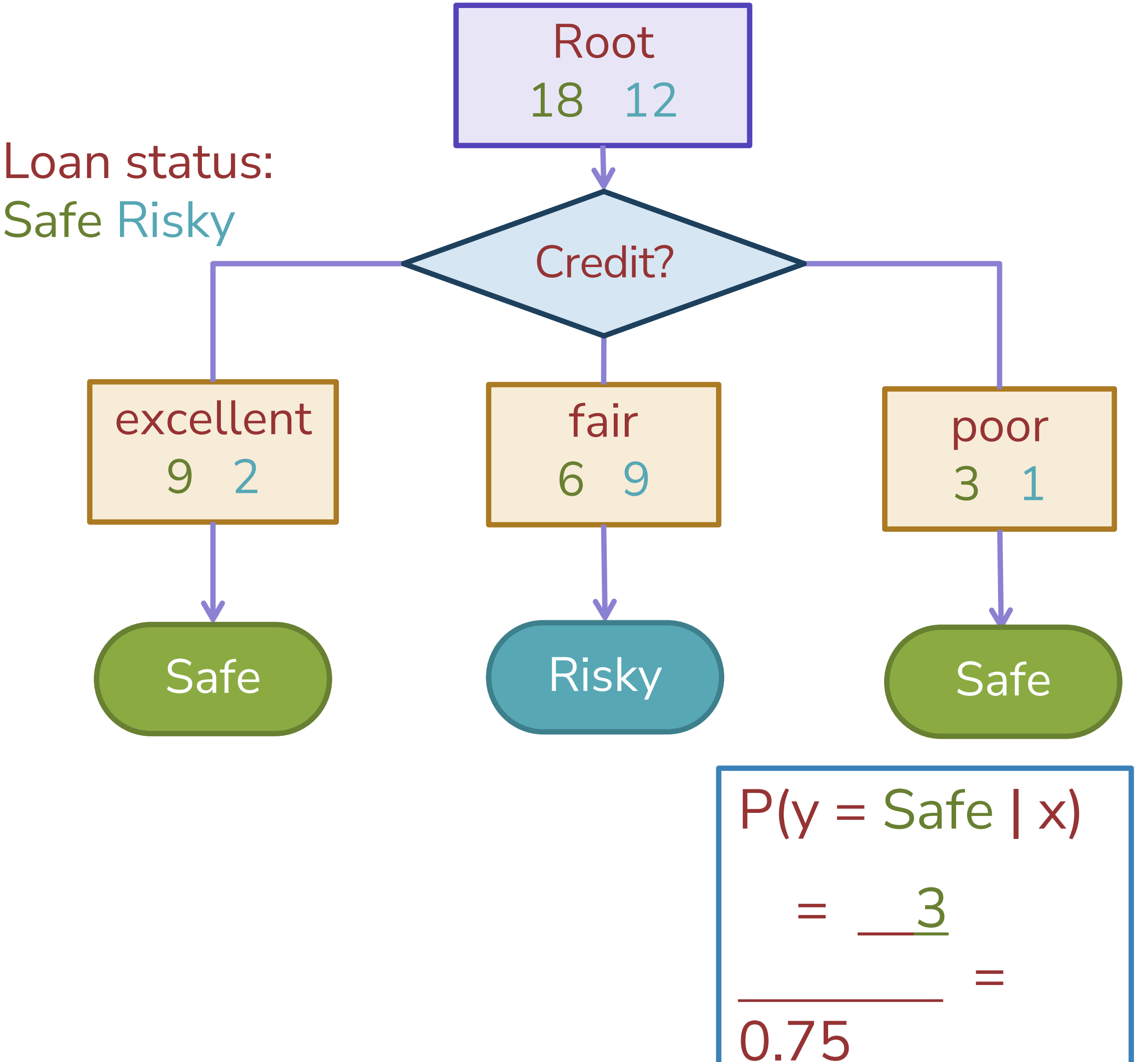
- Classification: Predict majority class for root node
- Regression: Predict average label for root node

In practice, we don't minimize classification error but instead some more complex metric to measure quality of split such as **Gini Impurity** or **Information Gain** (not covered in 416)

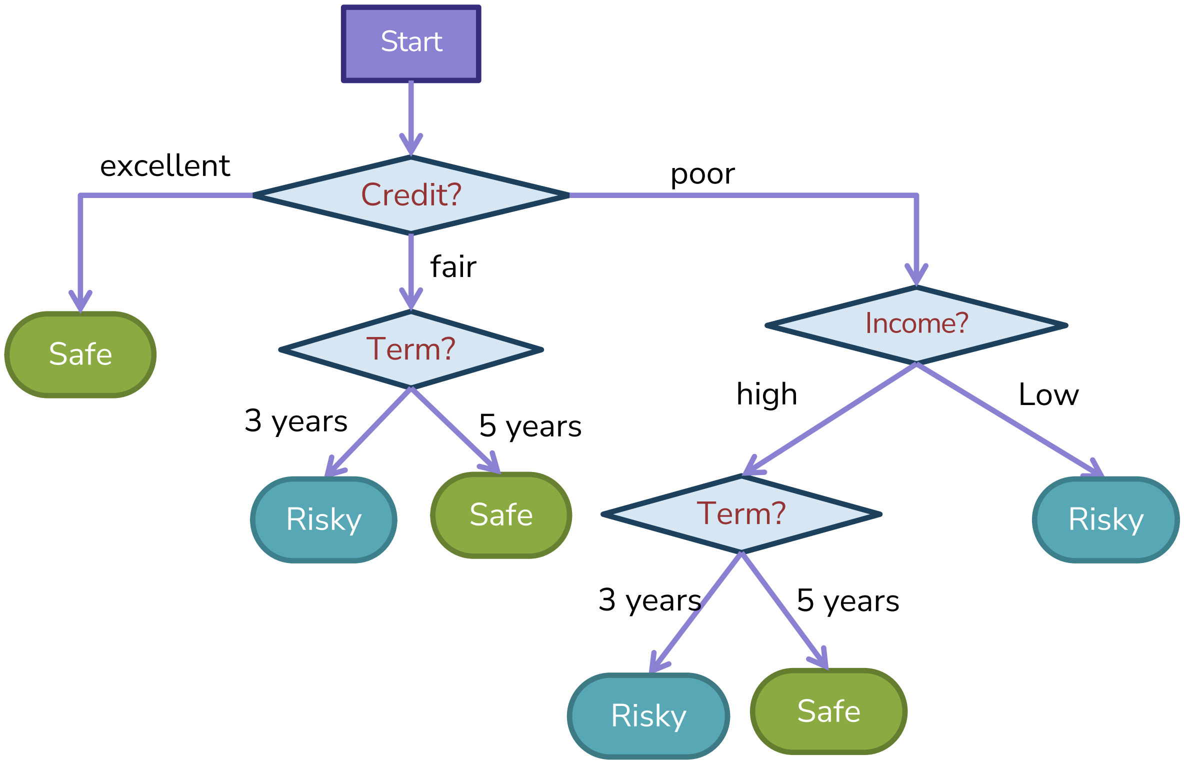


Can also be used to predict probabilities

Predicting probabilities



Decision Trees Overview



- **Branch/Internal node:** splits into possible values of a feature
- **Leaf node:** final decision (the class value)

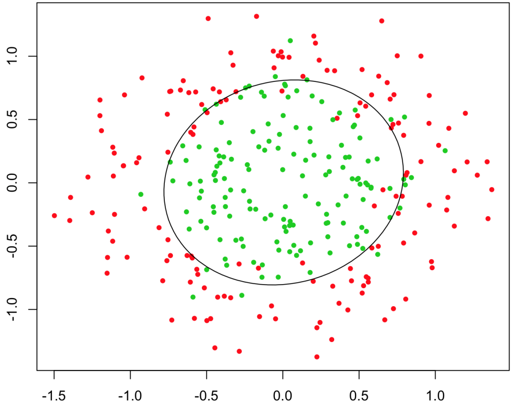
Pros/Cons Decision Tree

Pros:

- Easy to interpret
- Handles numeric and categorical variables without preprocessing*
    - In theory, scikit-learn still requires preprocessing
- No normalization required as it uses rule-based approach
- Can create non-linear decision boundaries
- Can readily do multi-class classification (unlike Logistic Regression)

Cons:

- Deep decision trees are prone to overfitting
- Only allows axis-parallel decision boundaries



Ensemble Method

Instead of switching to a brand new type of model that is more powerful than trees, what if we instead tried to make the tree into a more powerful model.

What if we could combine many weaker models in such a way to make a more powerful model?

A **model ensemble** is a collection of (generally weak) models that are combined in such a way to create a more powerful model.

There are two common ways this is done with trees

- Random Forest (Bagging) [next pre-lecture video]
- AdaBoost (Boosting)

AdaBoost

Boosting

Background

A **weak learner** is a model that only does slightly better than random guessing.

Kearns and Valiant (1988, 1989):

> "Can a set of weak learners create a single strong learner?"

Schapire (1990)

> "Yes!"

AdaBoost Overview

AdaBoost is a model similar to Random Forest (an ensemble of decision trees) with three notable differences that impact how we train it quite severely.

Instead of using high depth trees that will overfit, we limit ourselves to **decision stumps**.

Instead of doing majority voting, each model in the ensemble gets a weight and we take a **weighted majority vote**

$$\hat{y} = \hat{F}(x) = sign\left(\sum_{t=1}^{T} \hat{w}_t \hat{f}_t(x)\right)$$

Instead of doing random sampling with replacement, we **use the whole dataset and assign each datapoint a weight**, where high-weight datapoints were frequently misclassified by earlier models in the ensemble.

Poll Everywhere

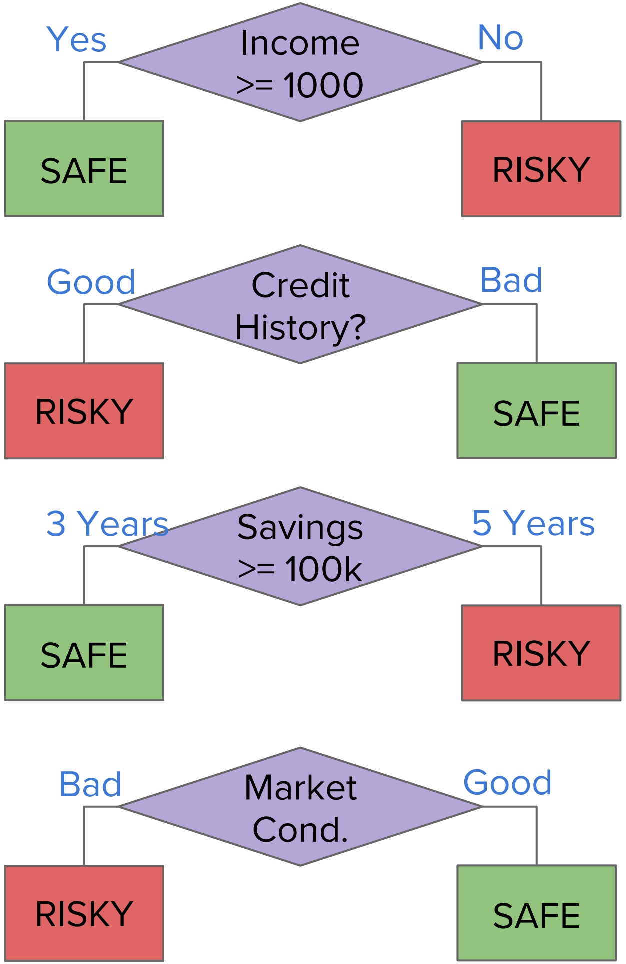
Think

2 min
sli.do #cs416

Recall the prediction rule for **weighted majority vote.**

$$\hat{y} = \hat{F}(x) = sign\left(\sum_{t=1}^{T} \hat{w}_t \hat{f}_t(x)\right)$$

What label will AdaBoost predict with these trees and weights?



$\hat{f}_1(x) = +1$

$\hat{f}_2(x) = -1$

$\hat{f}_3(x) = -1$

$\hat{f}_4(x) = +1$

Weight	Value
$\hat{w}_1$	2
$\hat{w}_2$	-1
$\hat{w}_3$	1.5
$\hat{w}_4$	0

Training AdaBoost

With AdaBoost, training is going to look very different.

We train each model **in succession**, where we use the errors of the previous model to affect how we learn the next one.

To do this, we will need to keep track of two types of weights

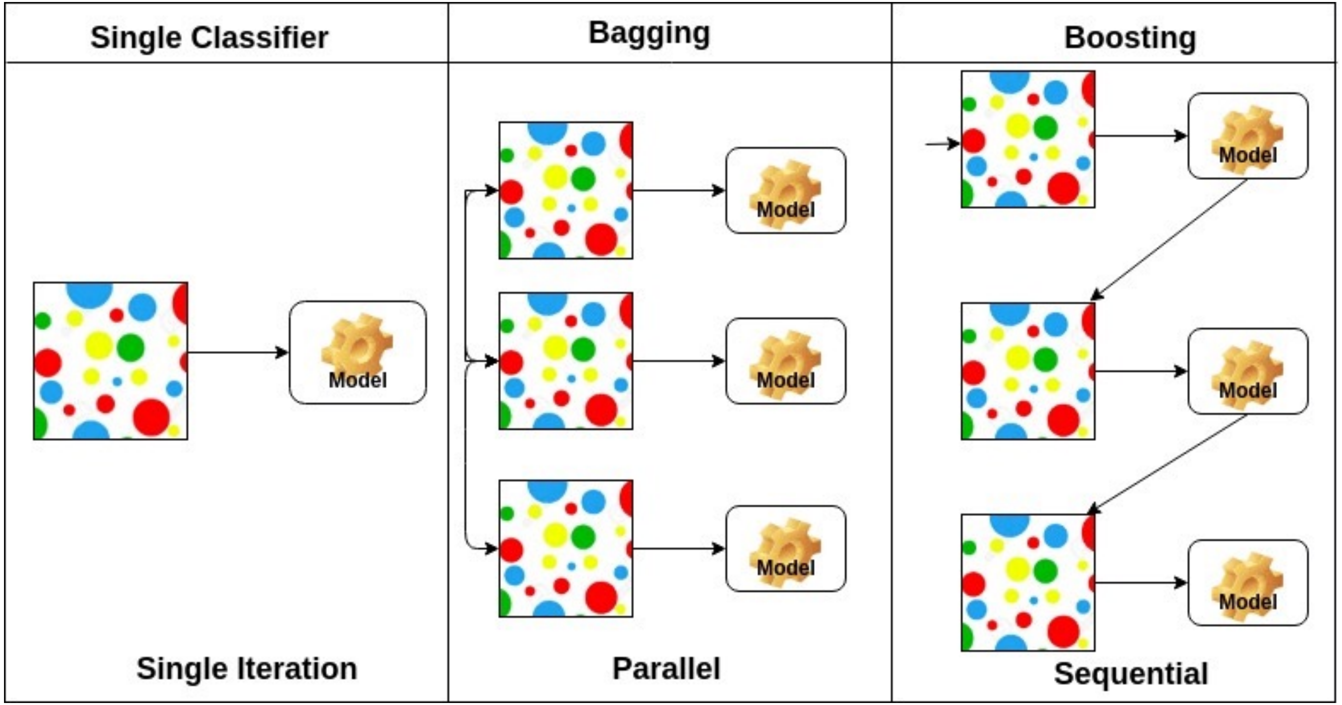
The first are the $\hat{w}_t$ that we will use as the end result to weight each model.

- **Intuition**: An accurate model within the ensemble should have a high weight

We will also introduce a weight α_i for each example in the dataset that we update each time we train a new model

- **Intuition**: We want to put more weight on examples that seem hard to classify correctly

Boosting (AdaBoost) vs. Bagging (Random Forrest)



AdaBoost Ada Glance

Train

for t in $[1, 2, \dots, T]$:

- Learn $\hat{f}_t(x)$ based on data weights $\alpha_{i,t}$
- Compute model weight $\hat{w}_t$
- Compute data weights $\alpha_{i,t+1}$

Predict

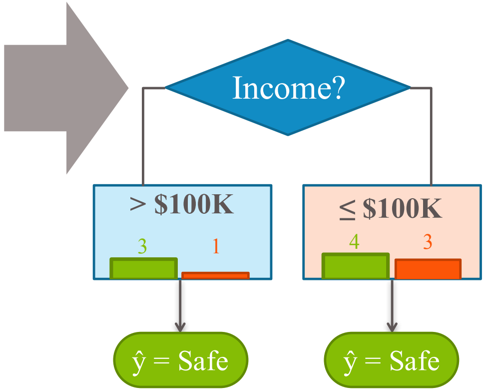
$$\hat{y} = \hat{F}(x) = sign\left(\sum_{t=1}^{T} \hat{w}_t \hat{f}_t(x)\right)$$

Weighted Data α_i

Start with a dataset and train our first model (a decision stump)

For all the things it gets wrong, increase the weight of that example. For each one that's right, decrease its weight.

Credit	Income	y
A	$130K	Safe
B	$80K	Risky
C	$110K	Risky
A	$110K	Safe
A	$90K	Safe
B	$120K	Safe
C	$30K	Risky
C	$60K	Risky
B	$95K	Safe
A	$60K	Safe
A	$98K	Safe



Credit	Income	y	Weight α
A	$130K	Safe	0.5
B	$80K	Risky	1.5
C	$110K	Risky	1.2
A	$110K	Safe	0.8
A	$90K	Safe	0.6
B	$120K	Safe	0.7
C	$30K	Risky	3
C	$60K	Risky	2
B	$95K	Safe	0.8
A	$60K	Safe	0.7
A	$98K	Safe	0.9

Learning w/ Weighted Data

Before, when we learned decision trees we found the split that minimized classification error.

Now, we want to minimize **weighted classification error**

$$WeightedError(f_t) = \frac{\sum_{i=1}^{n} \alpha_{i,t} \mathbb{I}\{\hat{f}_t(x_i) \neq y_i\}}{\sum_{i=1}^{n} \alpha_{i,t}}$$

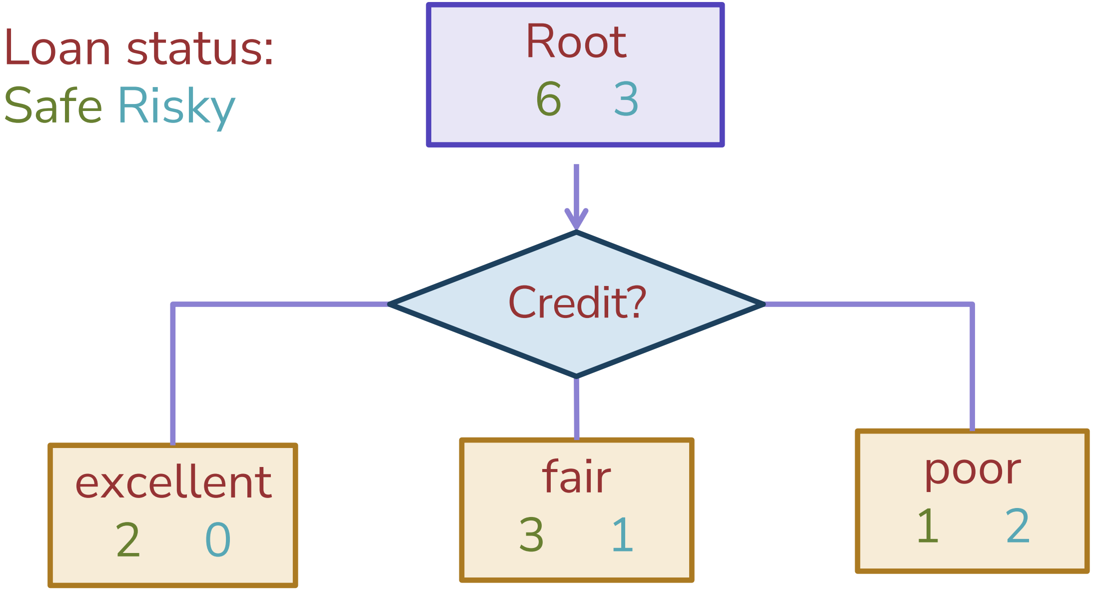
If an example x_2 has weight $\alpha_2 = 3$, this means getting that example wrong is the same as getting 3 examples wrong!

This will most likely change which split is optimal!

Learning w/ Weighted Data

Credit	y	weight
excellent	safe	1.2
fair	risky	3.0
fair	safe	0.5
poor	risky	0.9
excellent	safe	0.9
fair	safe	0.7
poor	risky	1.0
poor	safe	2.1
fair	safe	1.2

We also set leaf node predictions to be the **class with larger total weight**, not the class with more instances.



Poll Everywhere

Think

2 min

pollev.com/cs416

Consider the following weighted dataset, what is the weighted classification error of the optimal decision stump (just one split)?

We want to use the TumorSize and IsSmoker to predict if a patient's tumor is malignant.

TumorSize	IsSmoker	Malignant	Weight
Small	No	No	0.5
Small	Yes	Yes	1.2
Large	No	No	0.3
Large	Yes	Yes	0.5
Small	Yes	No	3.3

Poll Everywhere

Think

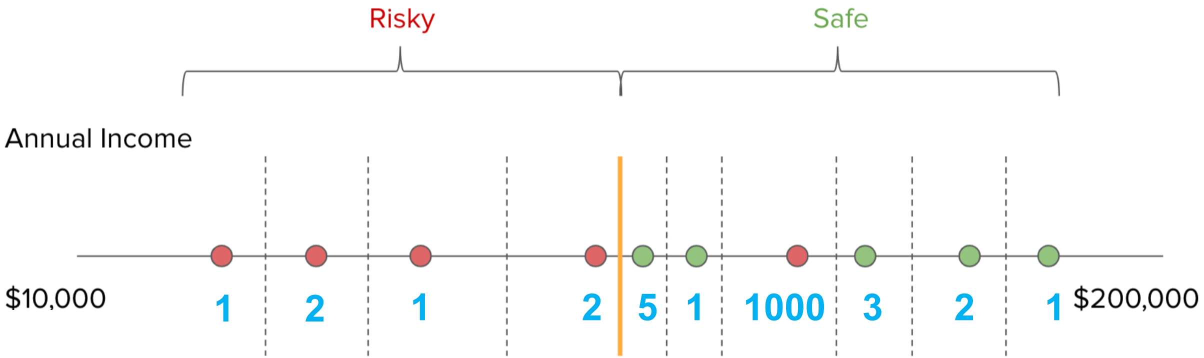
0 min

pollev.com/cs416

TumorSize	IsSmoker	Malignant	Weight
Small	No	No	0.5
Small	Yes	Yes	1.2
Large	No	No	0.3
Large	Yes	Yes	0.5
Small	Yes	No	3.3

Real Valued Features

The algorithm is more or less the same, but now we need to account for weights



Recap

What you can do now:

- Define the assumptions and modeling for Naïve Bayes
- 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
- Decision Tree pros/cons
- Ensemble methods
- AdaBoost intro