CSE/STAT 416

Naïve Bayes and Decision Trees

Pemi Nguyen
Paul G. Allen School of Computer Science & Engineering
University of Washington

April 18, 2022

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 $\hat{y}_i = -1 \; or \; \hat{y}_i = +1$

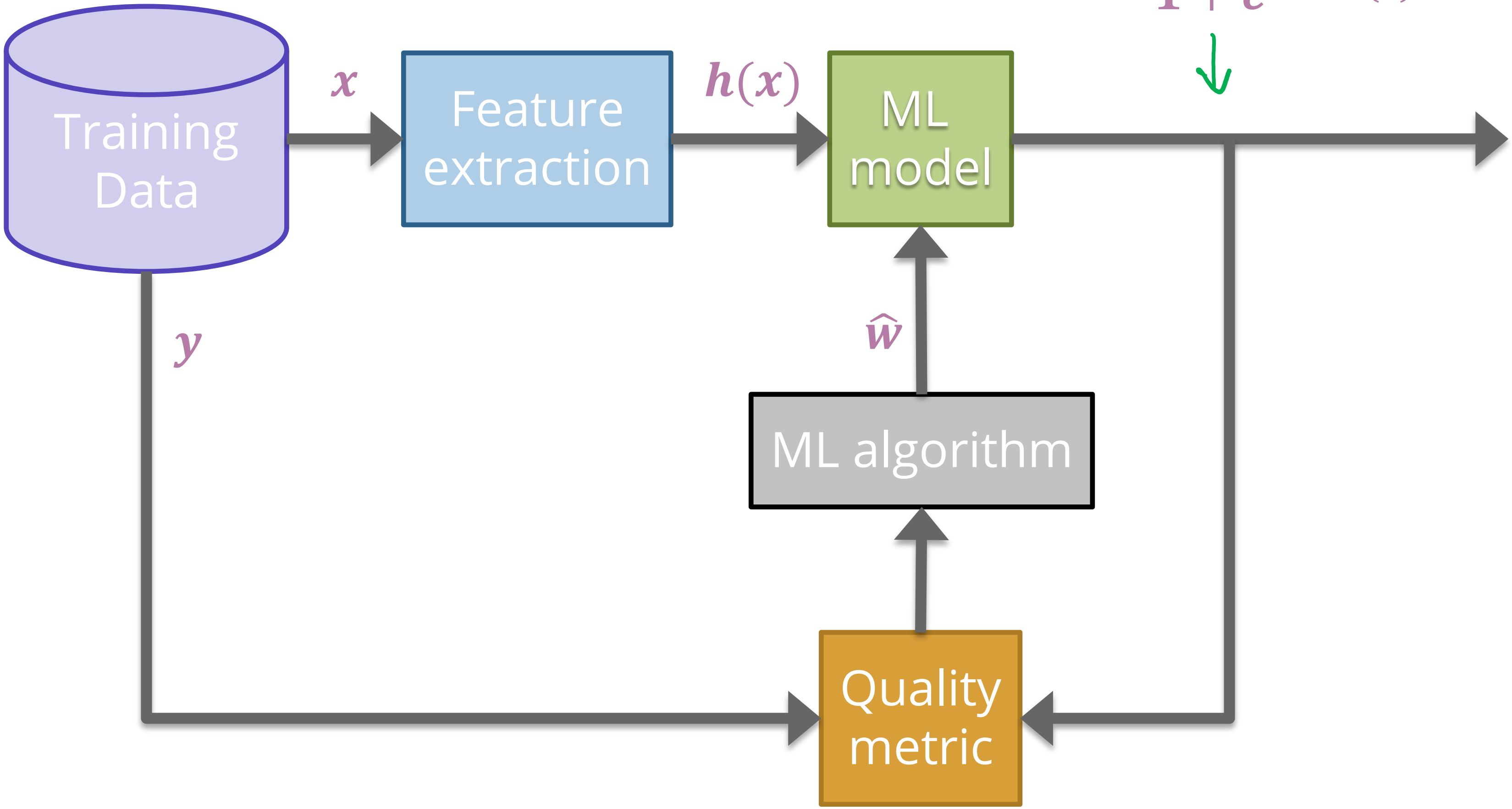
Very sure $\hat{y}_i = +1$

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

0 | 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

$x =$ *"The sushi & everything else was awesome!"*

$P\left(y = +1 \,\middle|\, x = \text{"The sushi \& everything else was awesome!"}\right)?$

$P\left(y = -1 \,\middle|\, x = \text{"The sushi \& everything else was awesome!"}\right)?$

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

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

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.

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

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

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

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.

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

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:

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

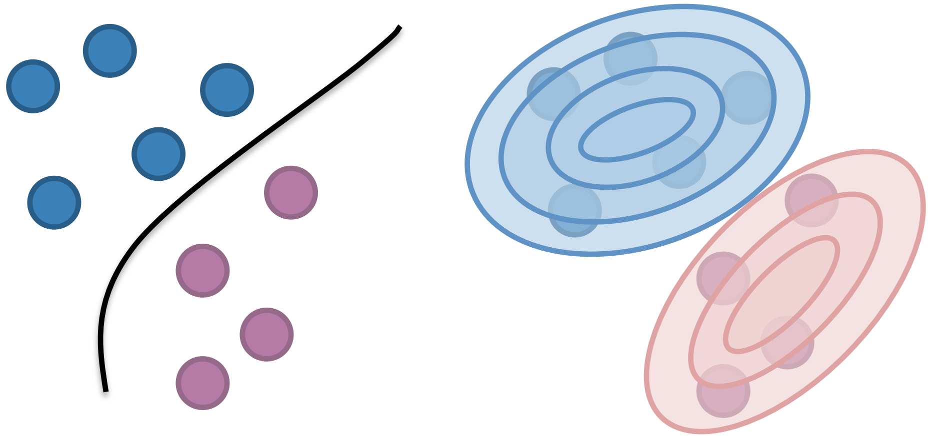
Naïve Bayes:

$$P(y|\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{xd}) = \prod_{j=1}^{d} P(x_j|y)\ P(y)$$

Compare Models

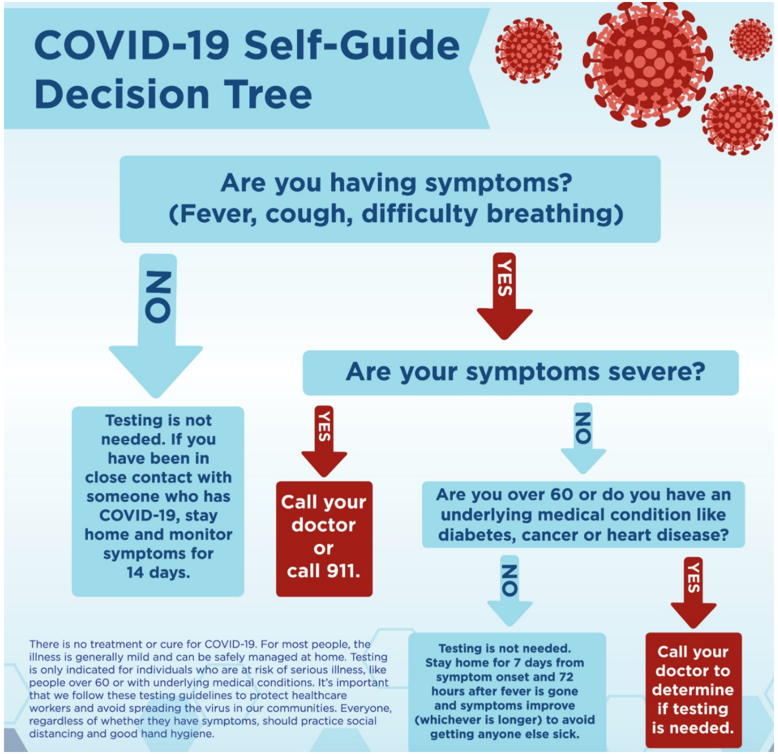
Generative: defines a distribution for generating x (e.g. Naïve Bayes)

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



Decision Trees

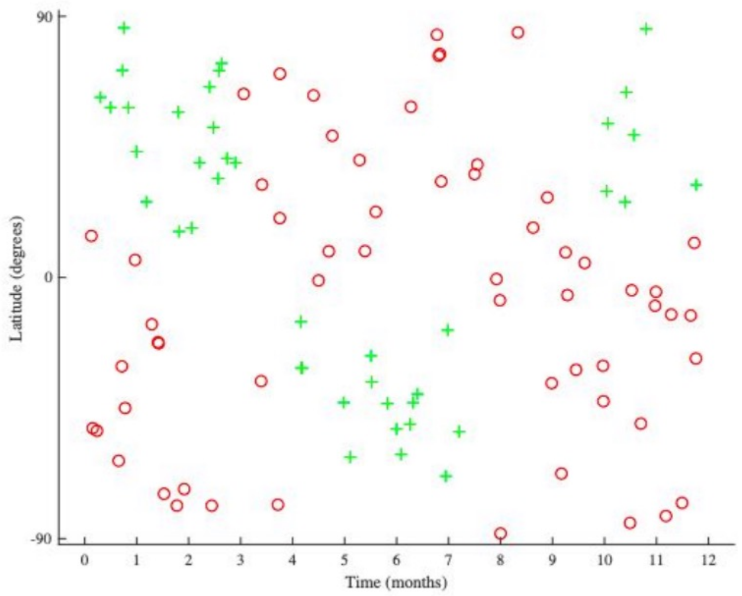
How do we make decisions?



https://www.holzer.org/coronavirus-covid-19-updates/

Non-linear decision boundaries

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 ★★★

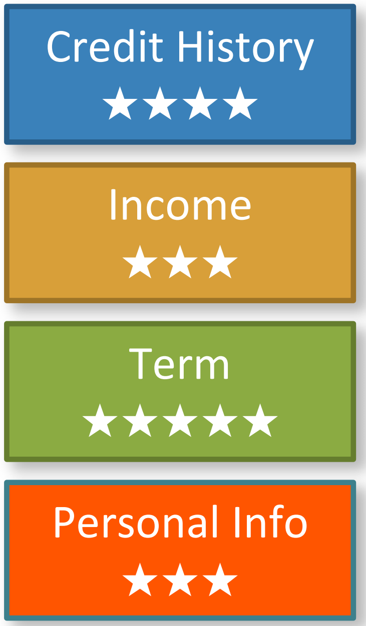
Term ★★★★★

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,…

Credit History
★★★★

Income
★★★

Term
★★★★★

Personal Info
★★★

Personal information

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

Example: Home loan for a married couple

Credit History ★★★★

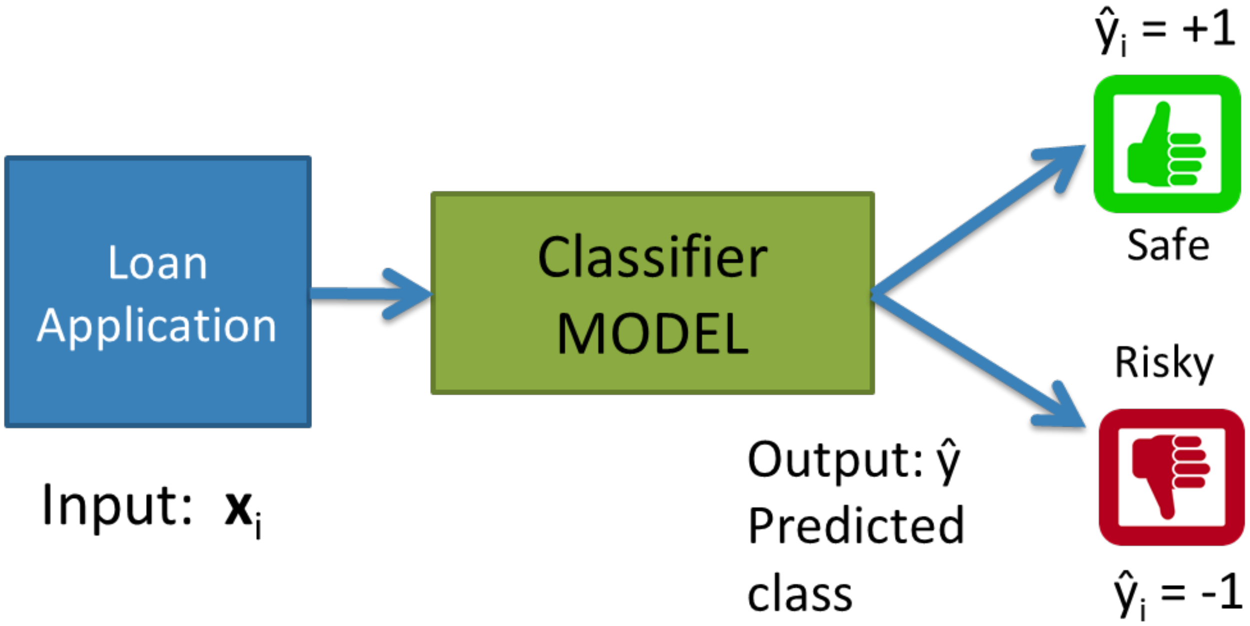
Income ★★★

Term ★★★★★

Personal Info ★★★

Intelligent application

Classifier review



Setup

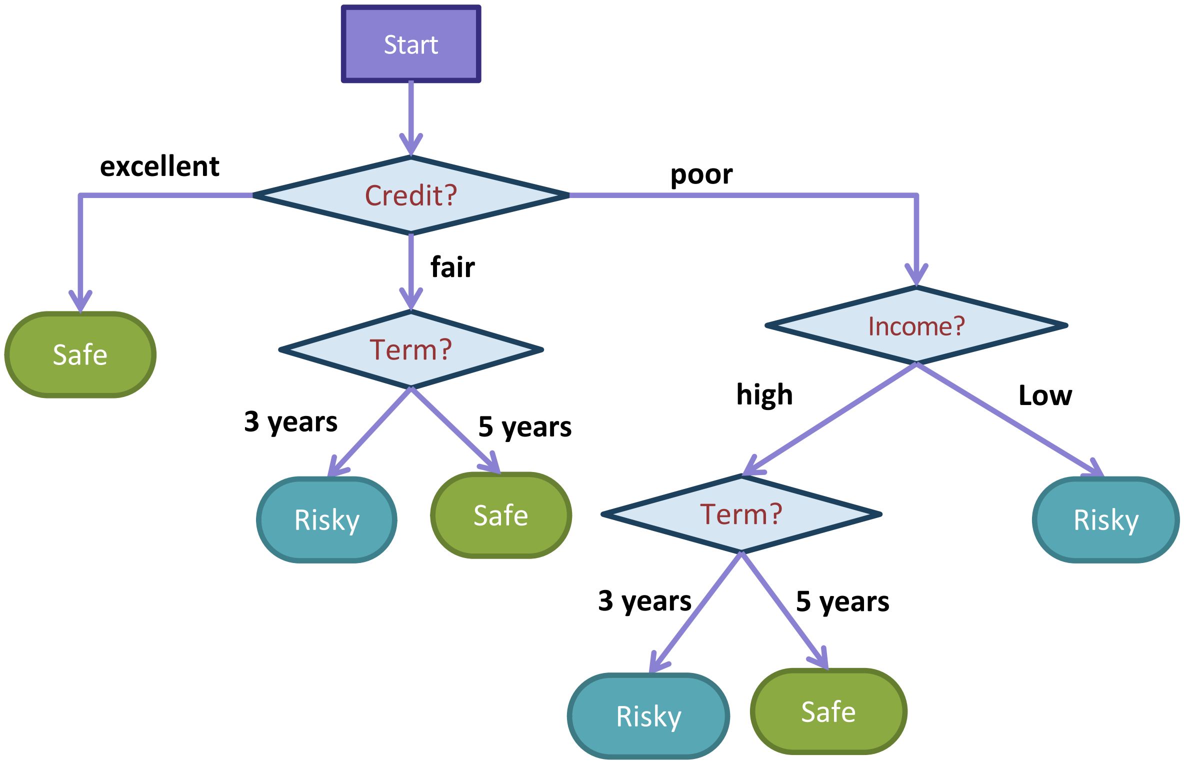
Data (N observations, 3 features)

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!

Decision Trees

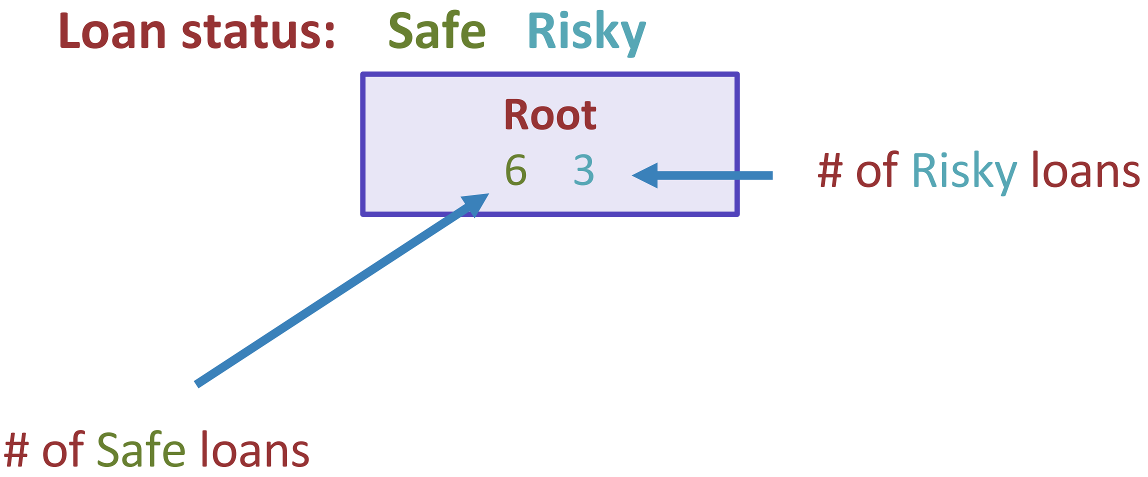


- **Branch/Internal node:** splits into possible values of a feature
- **Leaf node:** final decision (the class value)

Growing Trees

- Grow the trees using a greedy approach
- What do we need?

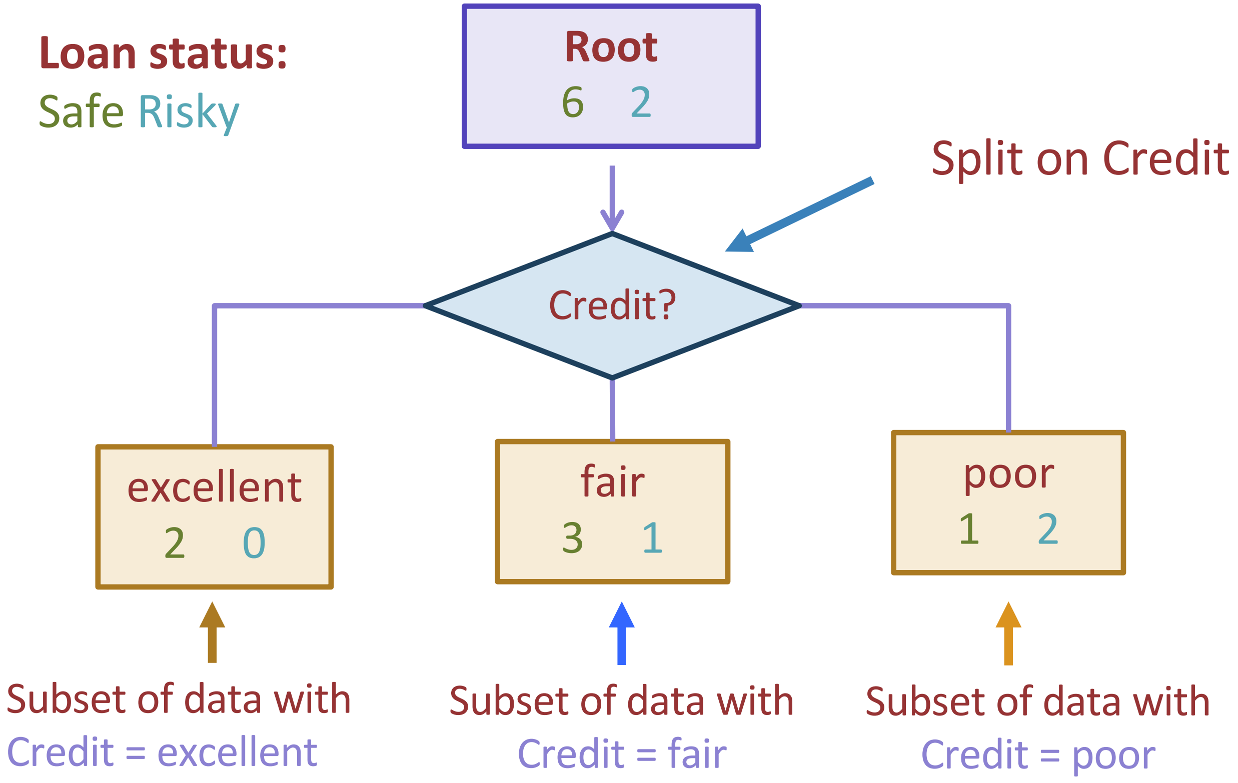
Visual Notation



N = 9 examples

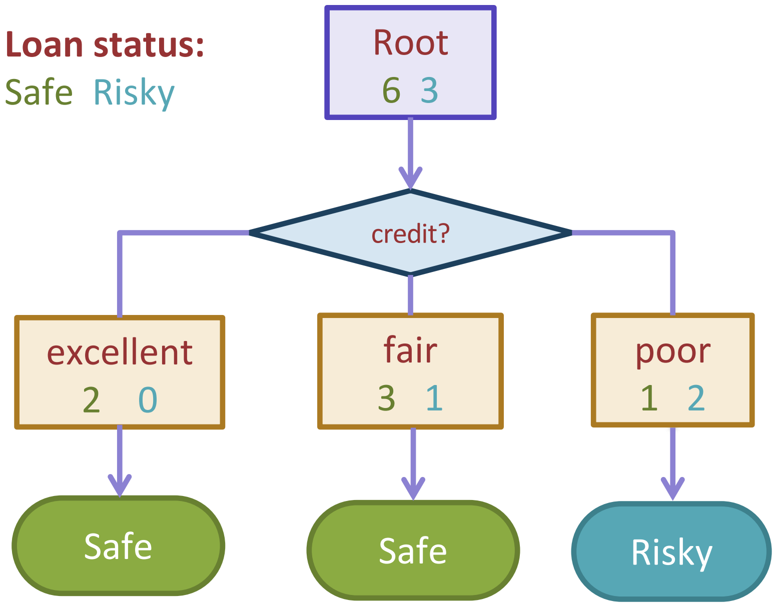
Decision stump: 1 level

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

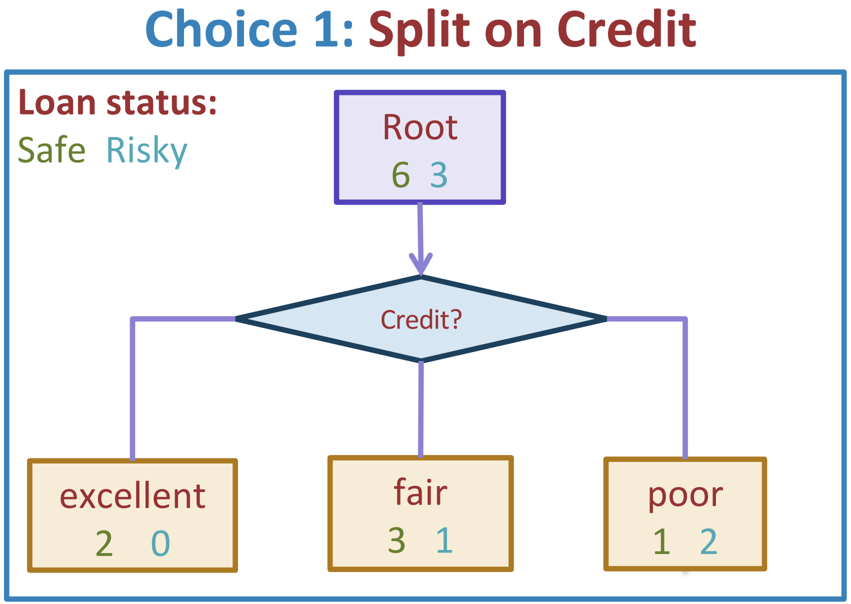
For each leaf node, set $\hat{y}$ = majority value

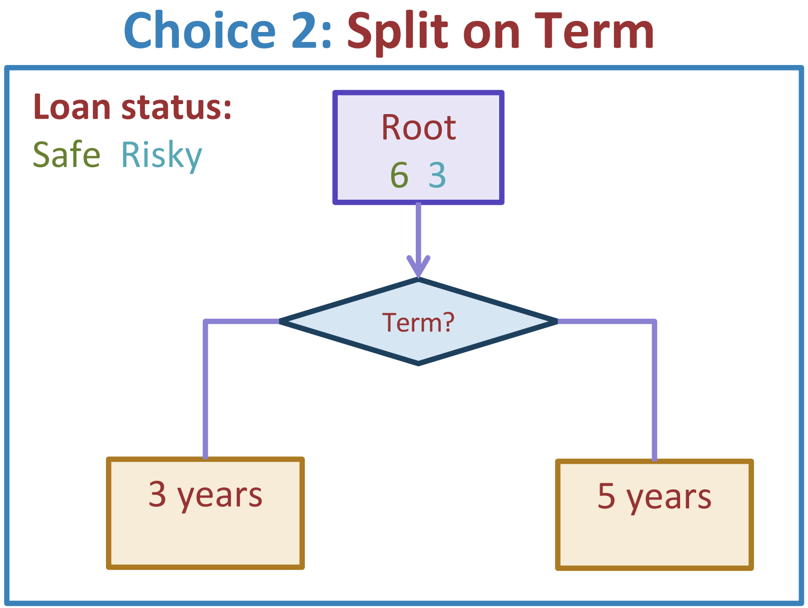


How do we select the best feature?

.

* Select the split with lowest classification error

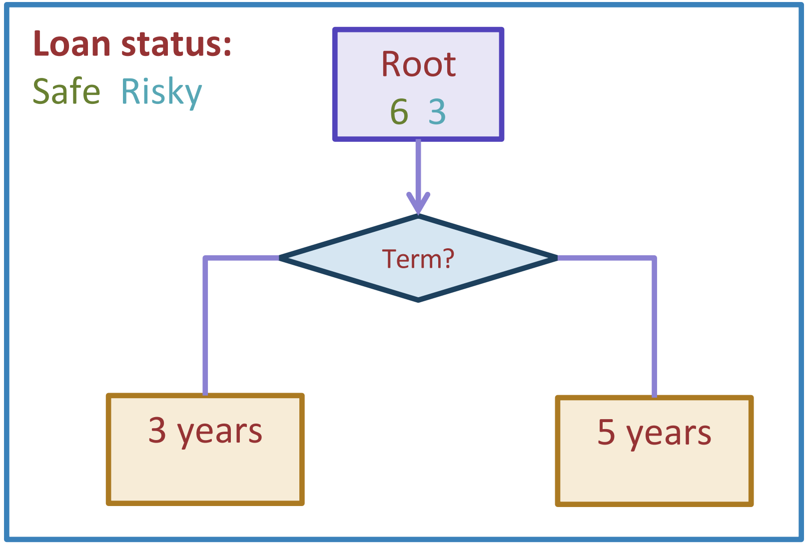




Calculate the node values.

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

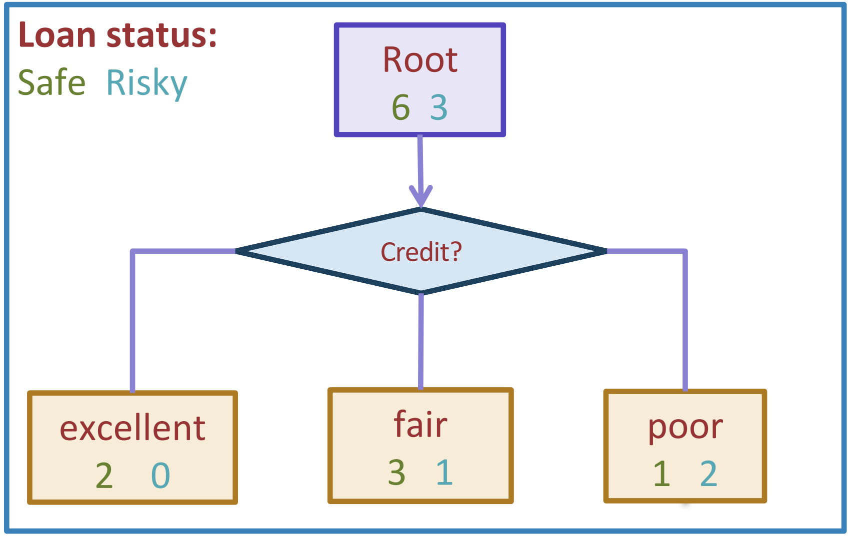
Choice 2: Split on Term



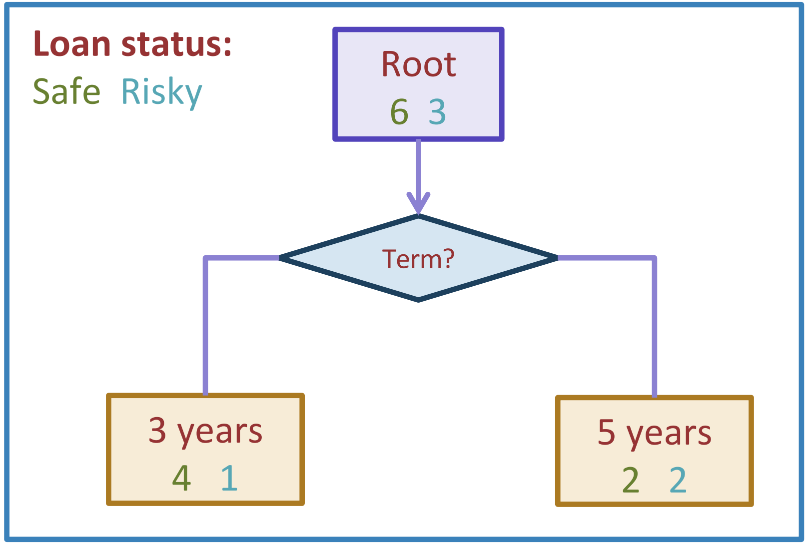
How do we select the best feature?

Select the split with lowest classification error

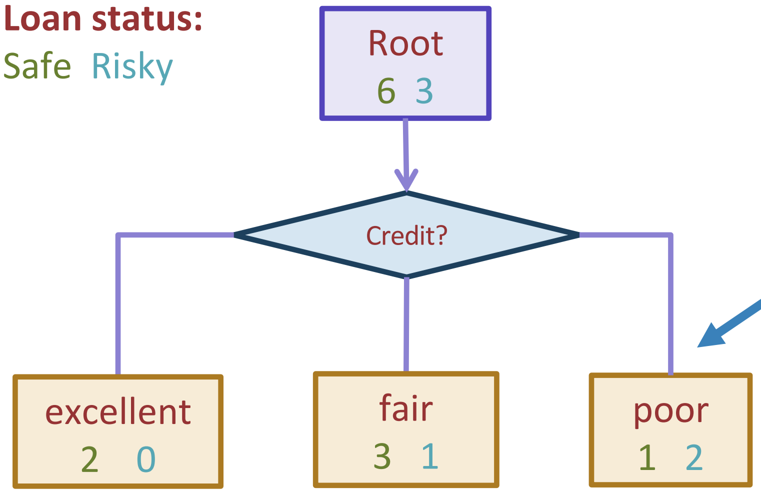
Choice 1: Split on Credit



Choice 2: Split on Term



How do we measure effectiveness of a split?



Idea: Calculate classification error of this decision stump

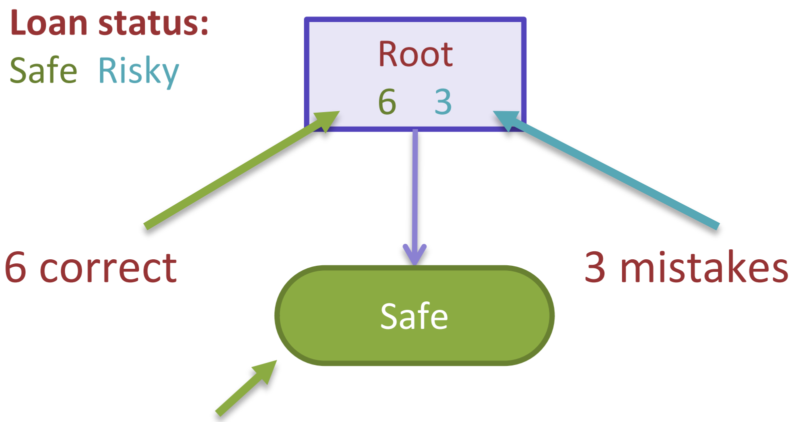
Error at a node

$$= \frac{\text{\# mistakes in each child node}}{\text{\# data points at the node}}$$

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**

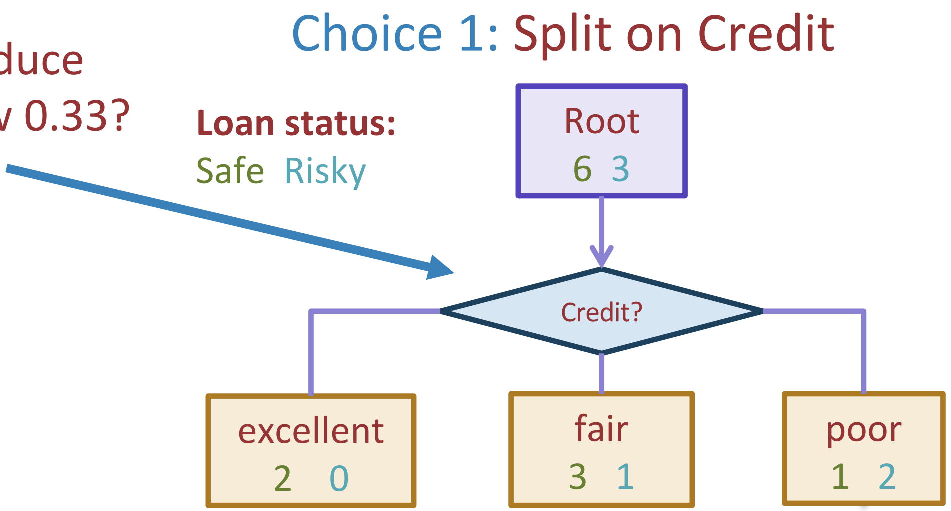
$$\text{Error} = \frac{3}{9}$$

$$= 0.33$$

Tree	Classification error
(root)	0.33

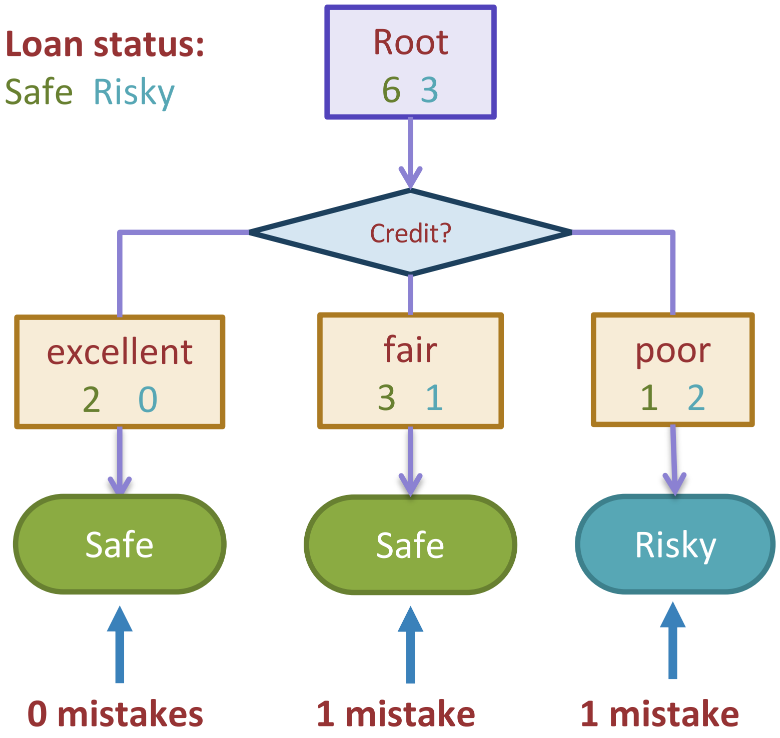
Choice 1: Split on Credit history?

Does a split on Credit reduce classification error below 0.33?



Split on Credit: Classification error

Choice 1: Split on Credit

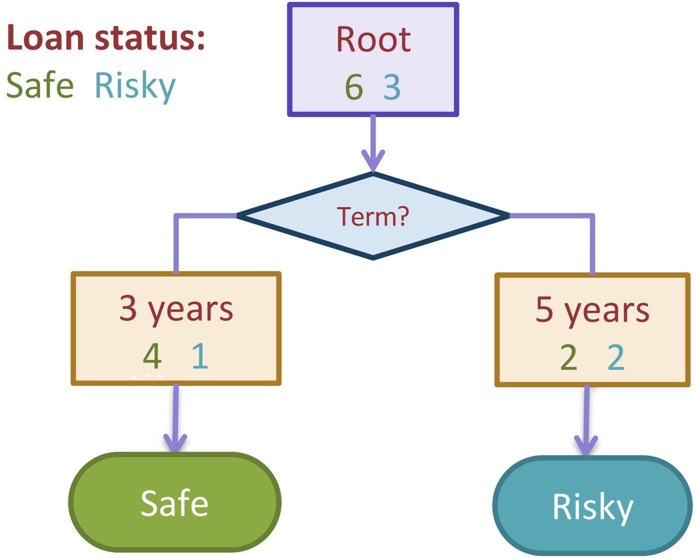


$$\text{Error} = \frac{0+1+1}{9} = \frac{2}{9}$$
$$= 0.22$$

Tree	Classification error
(root)	0.33
Split on credit	0.22

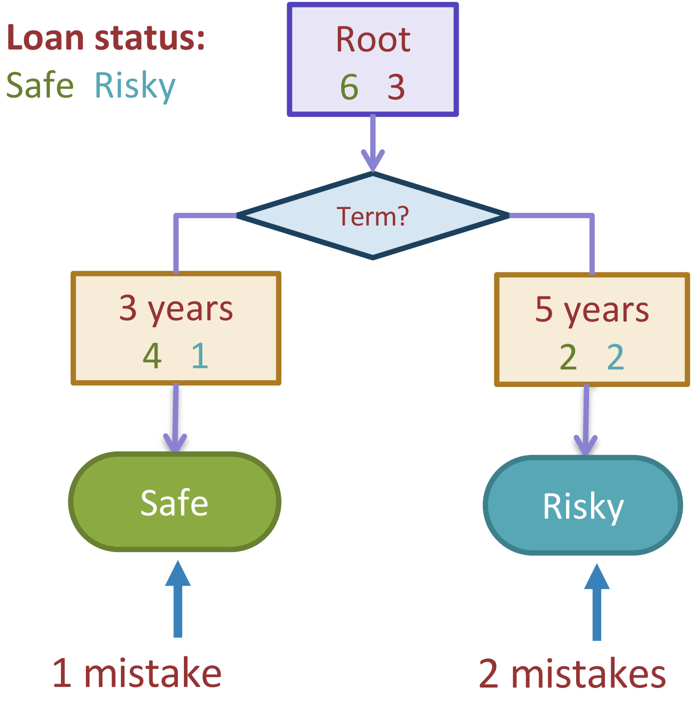
Choice 2: Split on Term?

Choice 2: Split on Term



Evaluating the split on Term

Choice 2: Split on Term



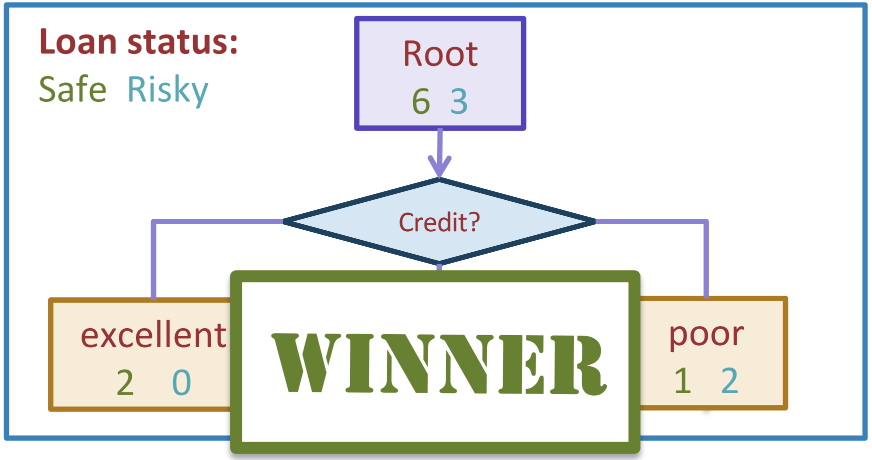
$$\text{Error} = \frac{1+2}{9} = \frac{3}{9}$$
$$= 0.33$$

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

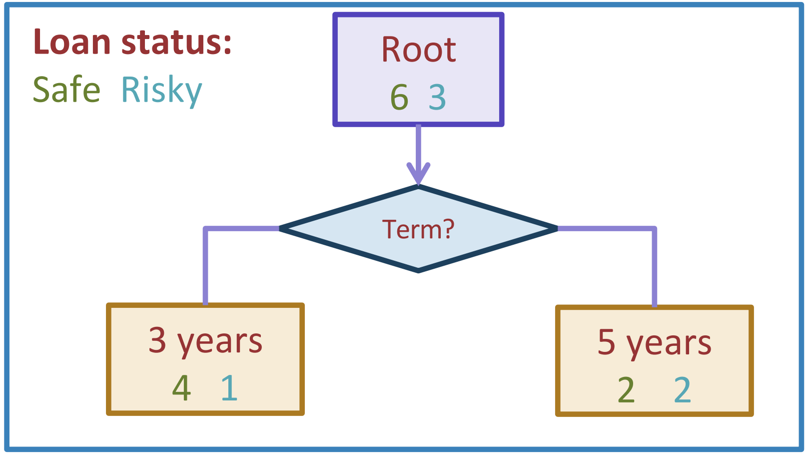
Choice 1 vs Choice 2: Comparing split on credit vs term

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)

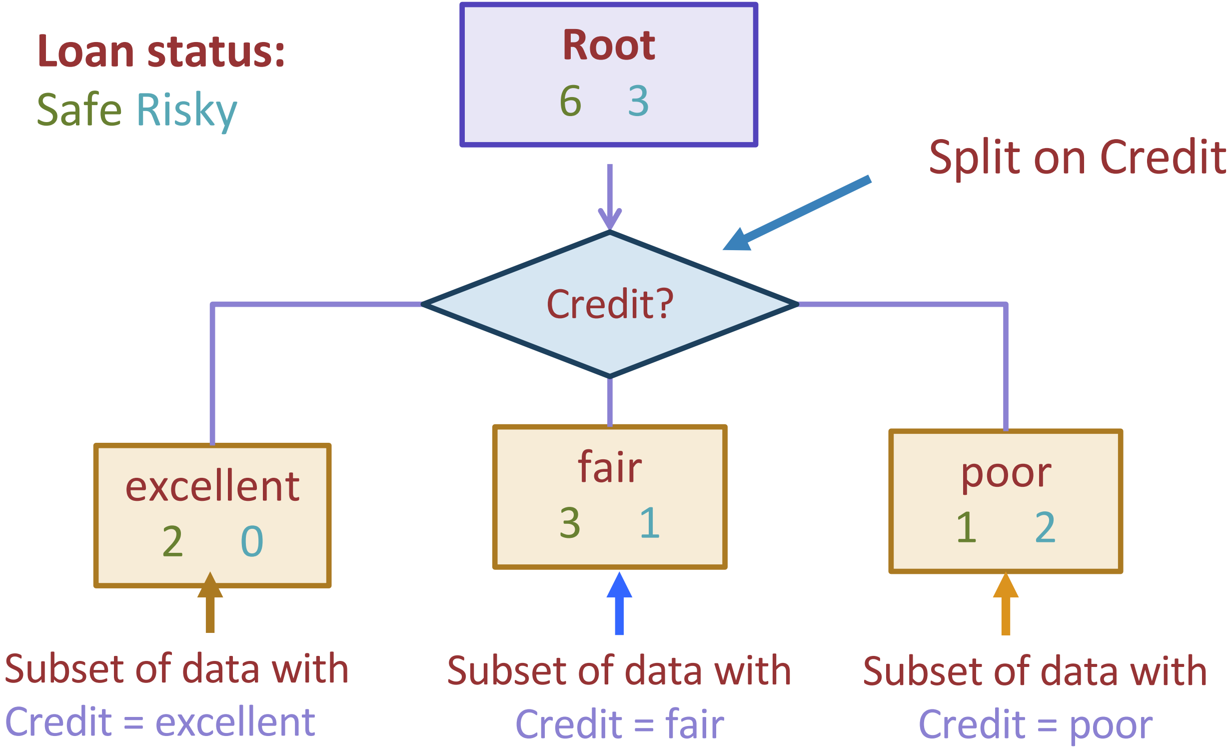
- Given a subset of data M in node
- For each feature h_i :
  - Compute classification error for a split of M according to feature h_i :
- Chose feature $h^*(x)$ with lowest classification error and expand the tree to include the children of current node after the split

Greedy & Recursive Algorithm

BuildTree(node)

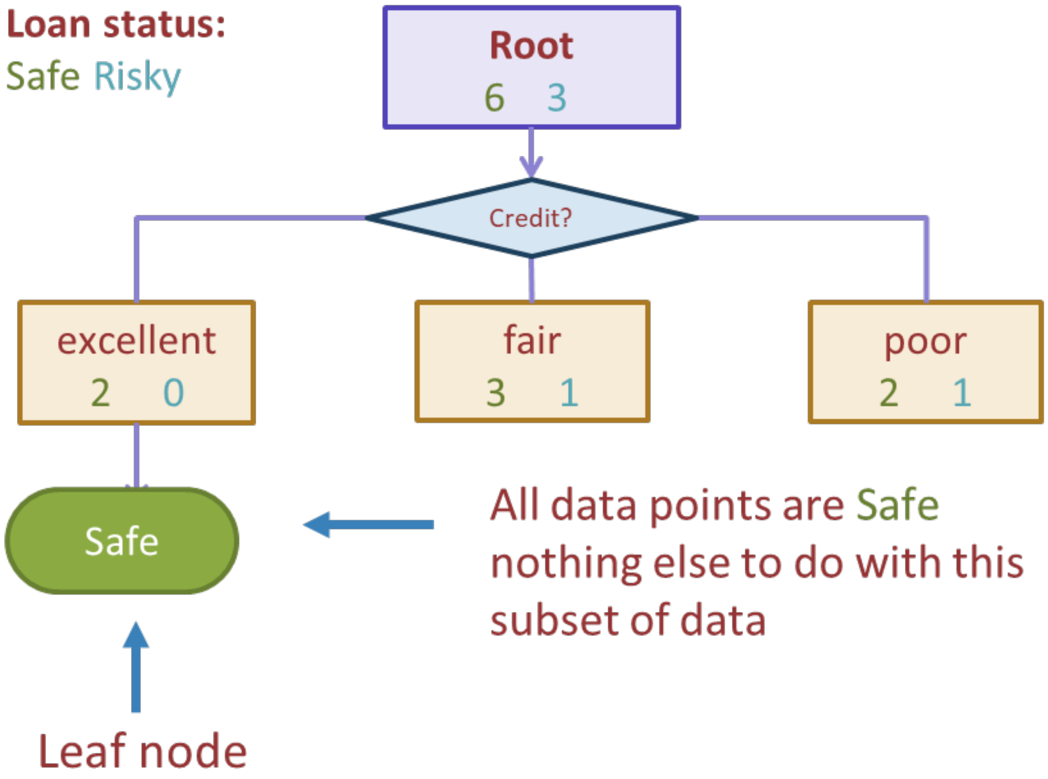
- If the number of datapoints at the current node or the classification error is within a certain threshold:
  - Stop
- Else:
  - Split(node)
  - For child in node:
    - BuildTree(child)
- Decision Tree algorithm is **greedy**: It aims to optimize the classification error at each node. As a result, the final result won't be globally optimal, but it guarantees computational efficiency
- Decision Tree algorithm is **recursive**: From the current node, if we decide to further expanding the tree, we will repeat the same operations in the child nodes.

Decision tree expansion: 1 level

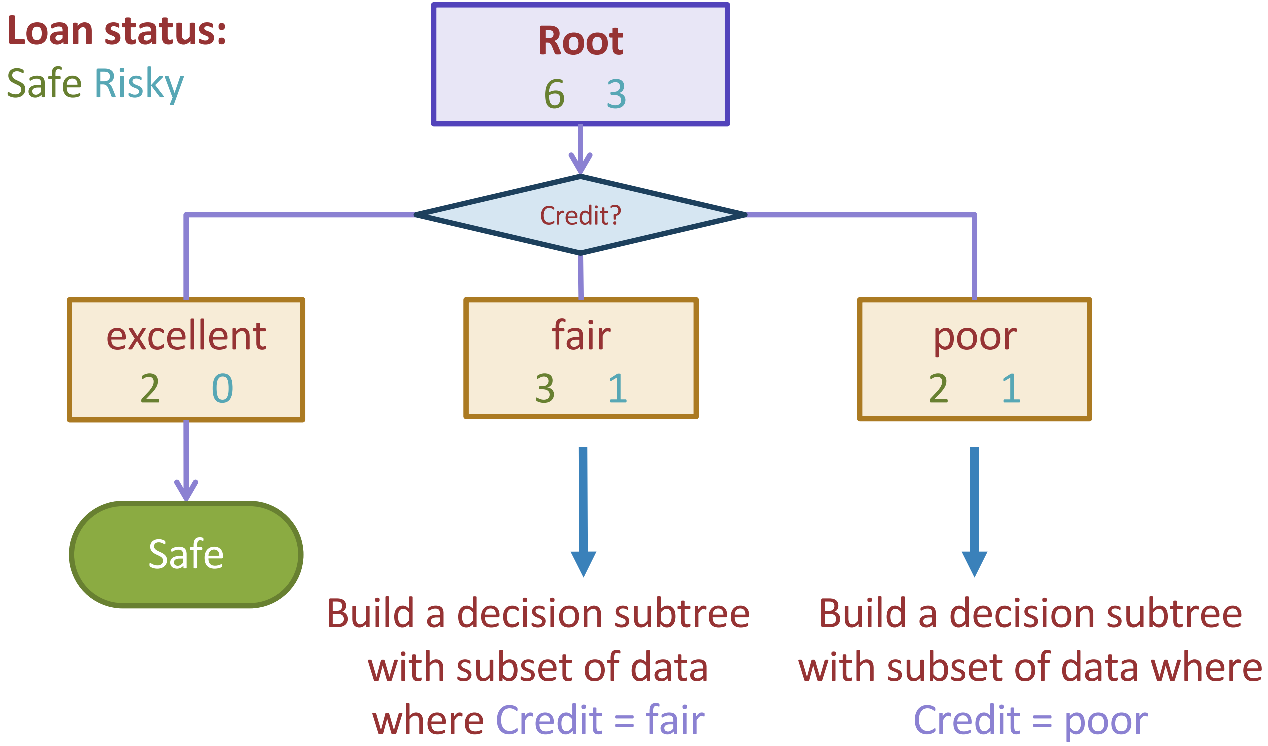


Stopping

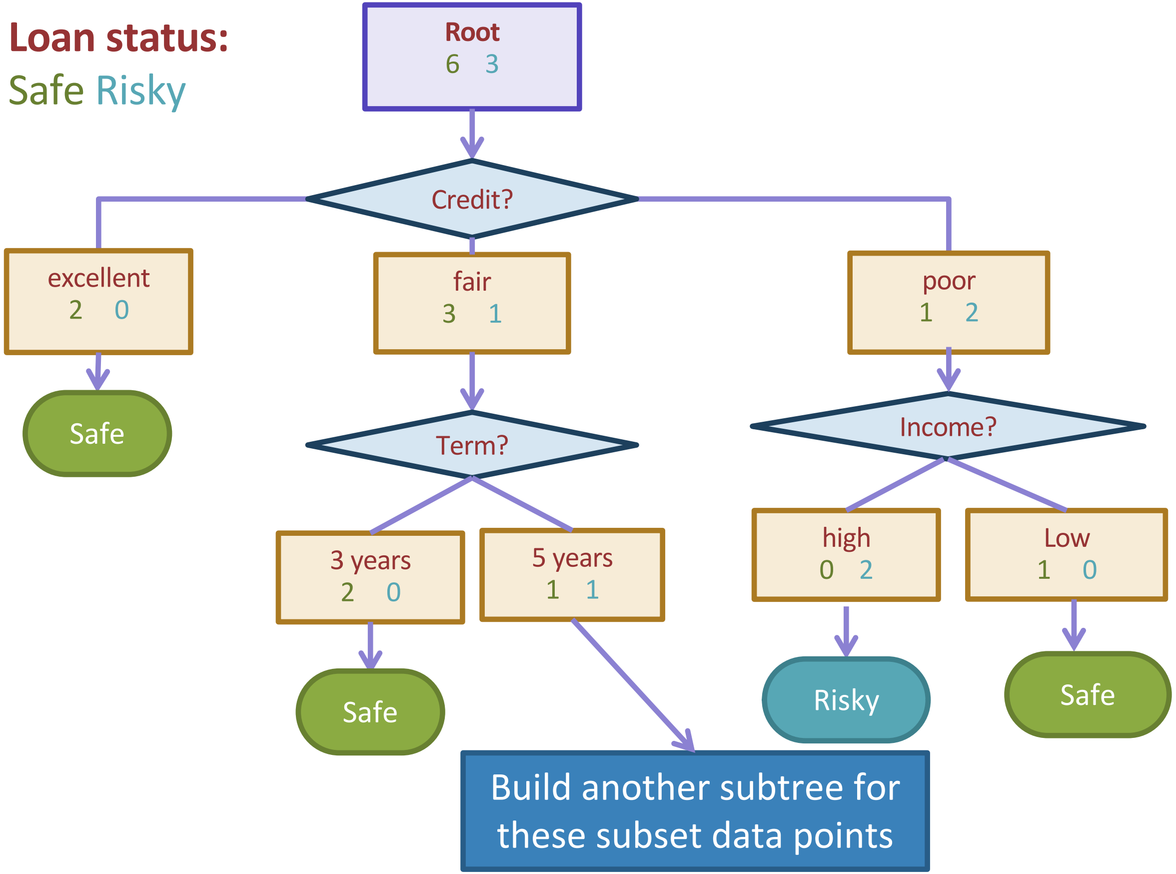
- Stop if all points are in one class



Expanding the trees by recursing on children nodes



Second level



Different data types that decision trees support

Income	Credit	Home ownership	y
$105 K	excellent	Rent	Safe
$112 K	good	Own	Risky
$73 K	fair	Rent	Safe
$69 K	excellent	Mortgage	Safe
$217 K	excellent	Own	Risky
$120 K	good	Mortgage	Safe
$64 K	fair	Own	Risky
$340 K	excellent	Own	Safe
$60 K	good	Other	Risky

Numeric features vs Categorical features

We have been used to numerical features (such as number of bedrooms / bathrooms, income, are).

However, in practice, data comes from different forms.

- There are three main data types:
    - **Numeric**
    - **Categorical**: data that takes on a number of fixed possible values
        - **Ordinal**: data that have ordered categories
          E.g: credit quality (good / fair / bad)
        - **Nominal**: data that doesn't have ordered categories
          E.g: home ownership (rent / own / mortgage)

- Reminder about the extra credit: Zip code is a categorically nominal variable

Transforming categorical features

Depending on implementations, decision trees might not need you to transform data of categorical types into numeric values

However, in models that use differentiable loss functions (like Linear Regression / Logistic Regression, some forms of decision trees), you need to transform categorical data

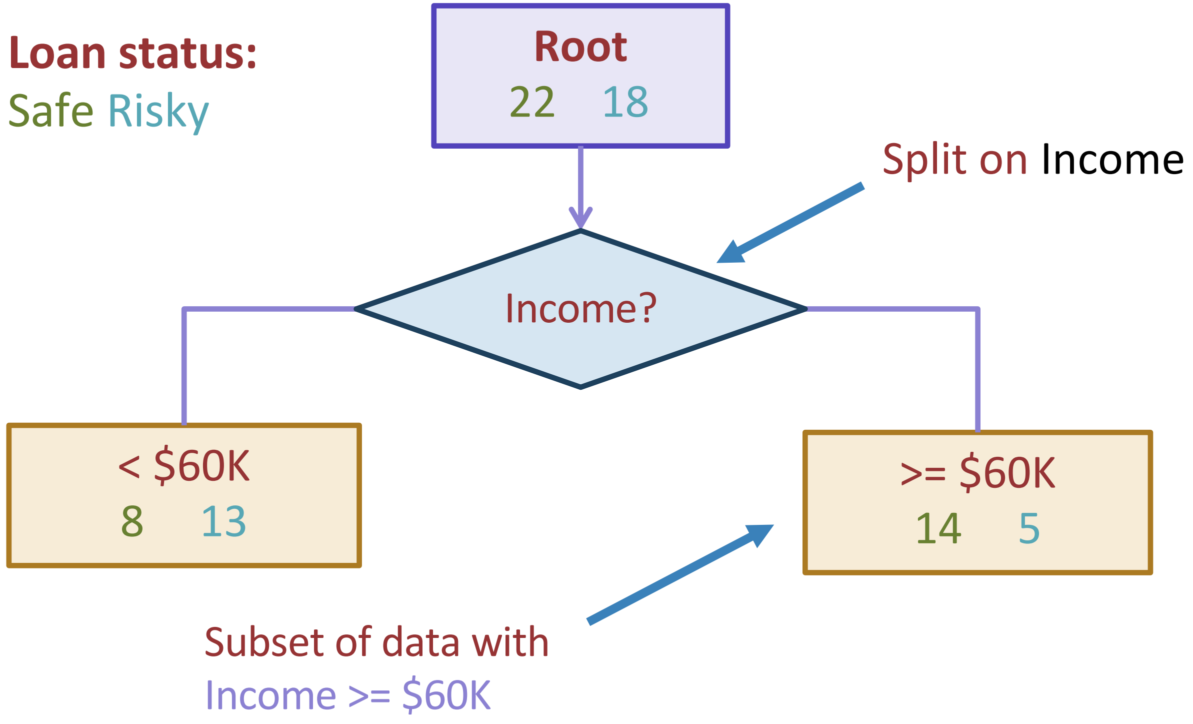
- **Ordinal**: Transform into numbers of a certain order
  e.g: For 3 categories (bad / fair / good)
  bad = 0, fair = 1, good = 2
- **Nominal**: Using one-hot encoding

id	color
1	red
2	blue
3	green
4	blue

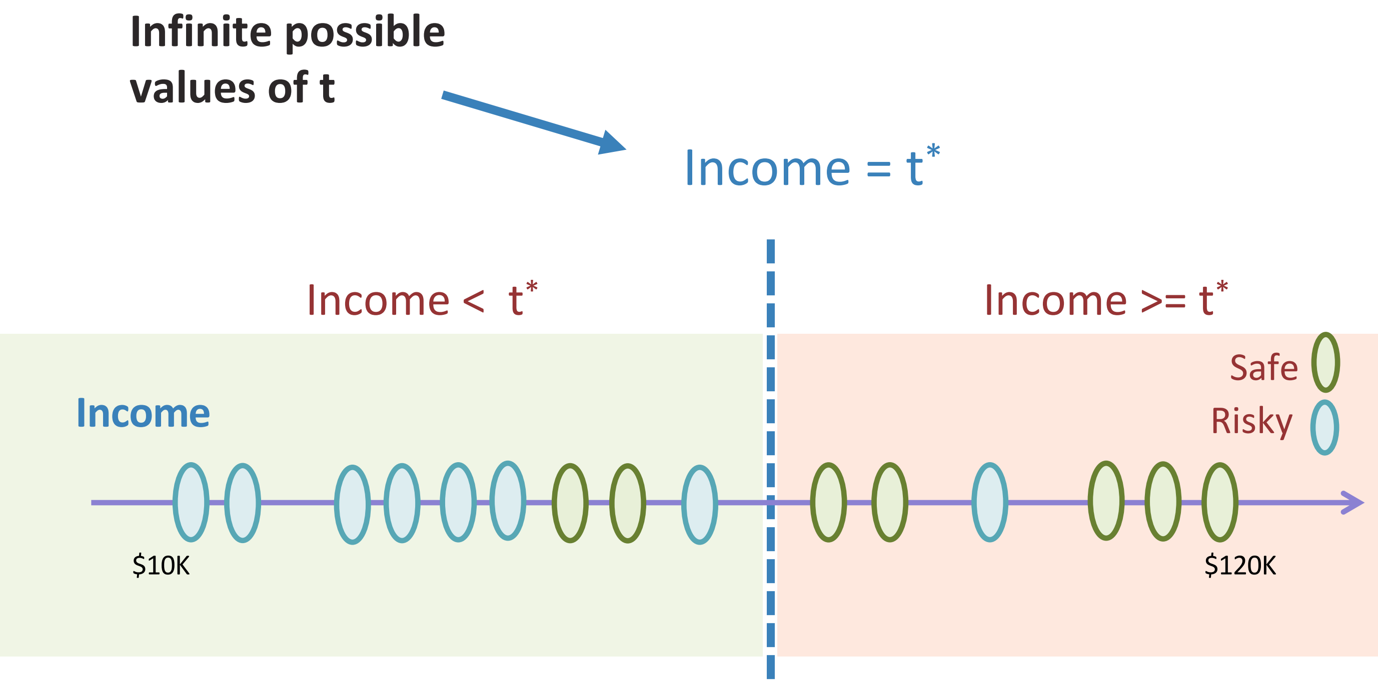
One Hot Encoding

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Threshold split for numeric features

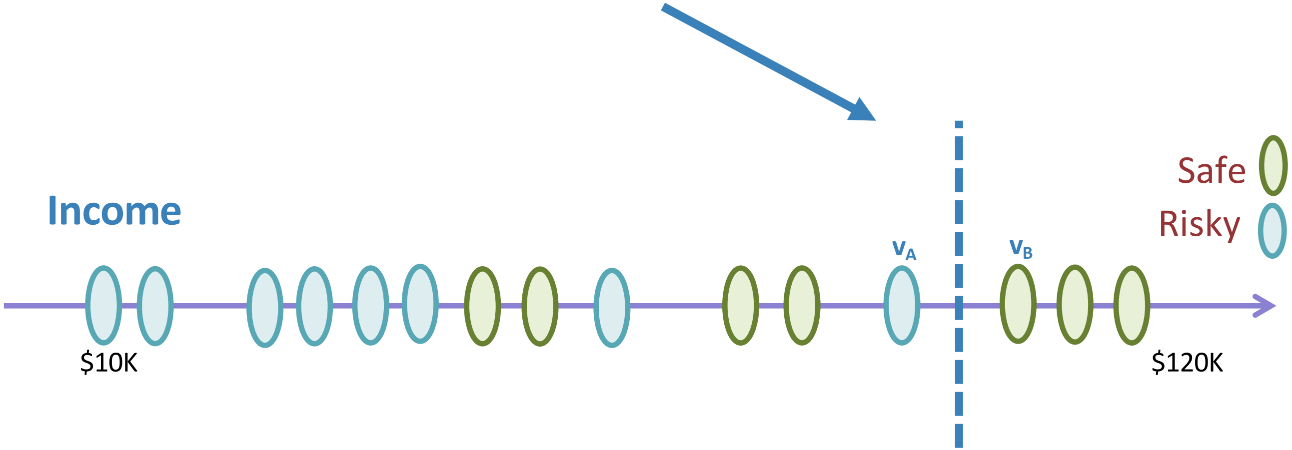


Best threshold?



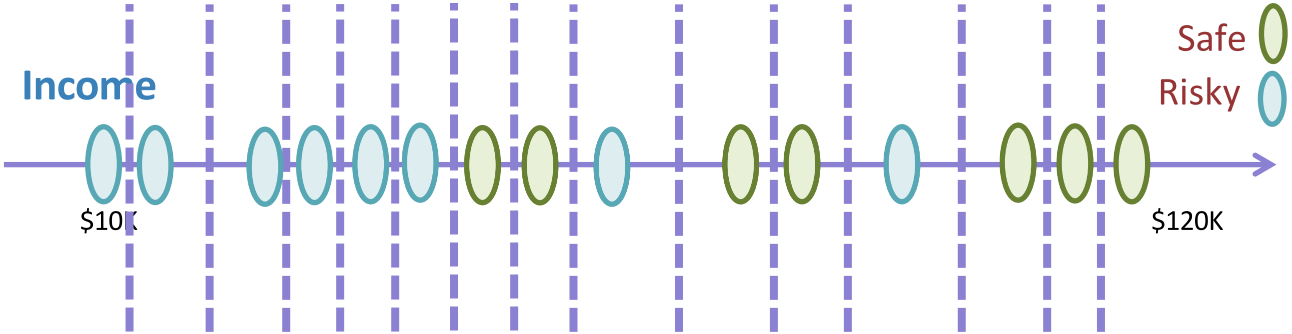
Threshold between points

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



Splitting for numeric features

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

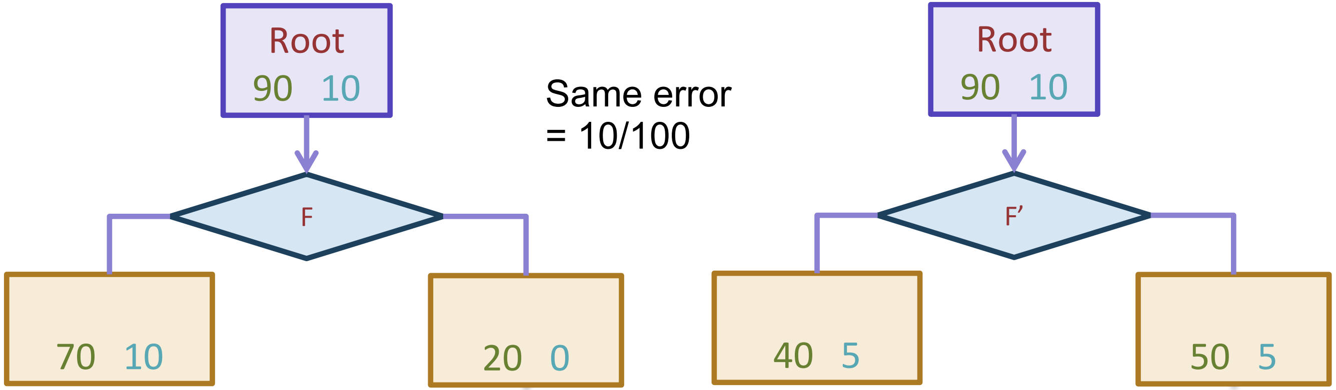
Let $\{\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}, \dots \mathbf{v_N}\}$ denote sorted values

Step 2:

- For i = 1 … N-1
  - Consider split $t_i = (\mathbf{v_i} + \mathbf{v_{i+1}}) / 2$
  - Compute classification error
- Chose the $\mathbf{t}^*$ with the lowest classification error

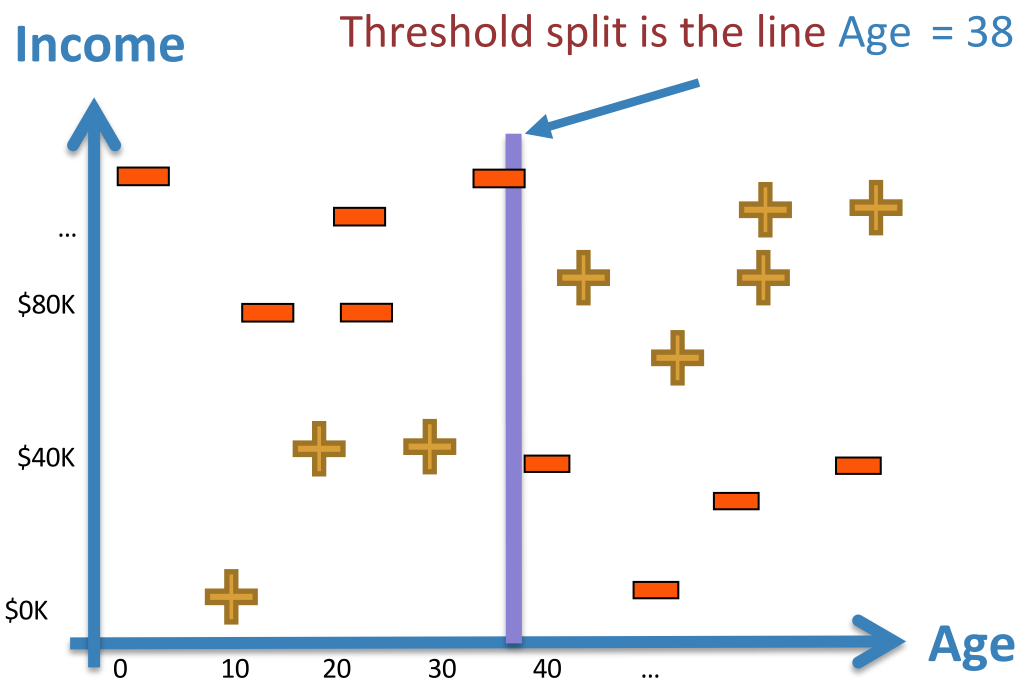
Issue with using classification error for splitting

Earlier, we learn how to learn decision trees based on classification error. However, this metric is not sensitive enough if two different splits give the same classification error.

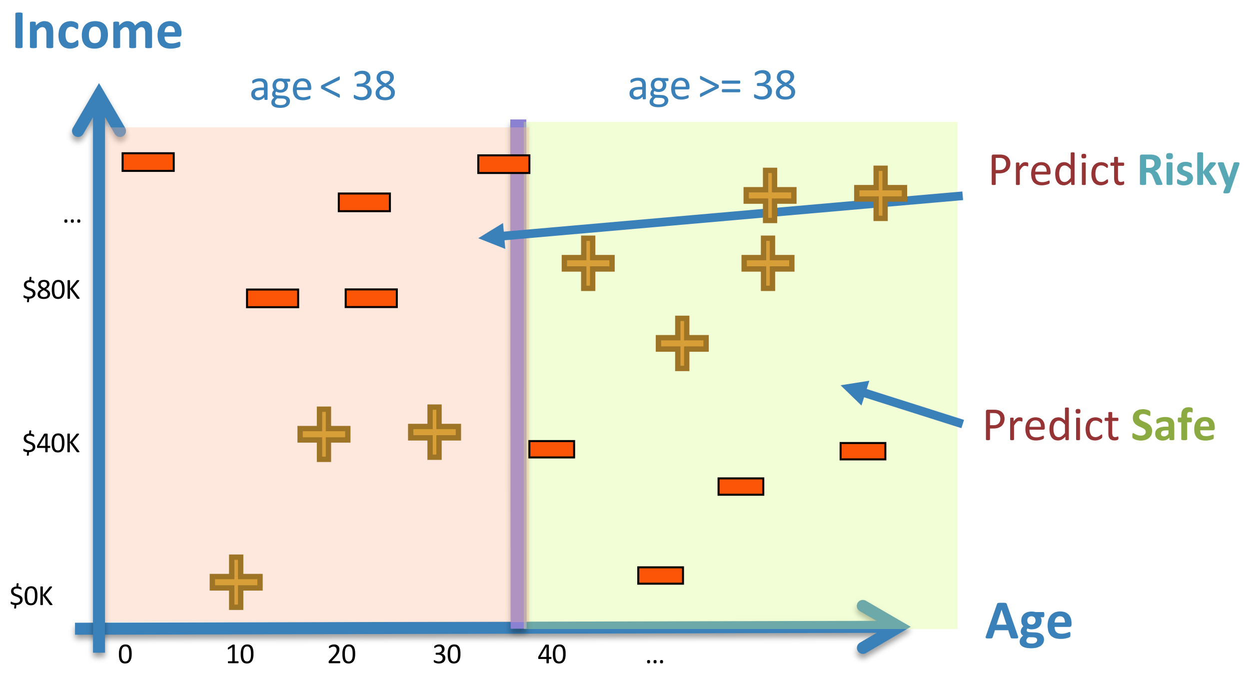


In practice, people prefer using continuous loss functions for splitting decision (two algorithms like ID3 – entropy loss or CART – Gini impurity loss).

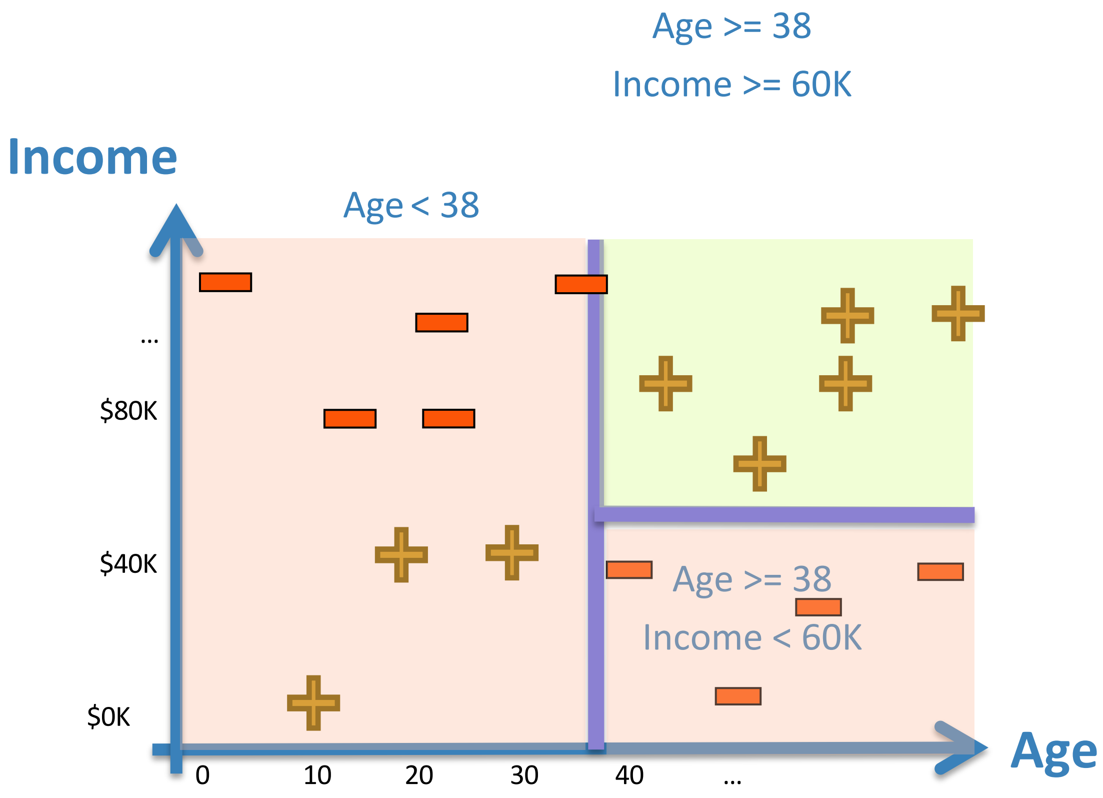
Visualizing the threshold split



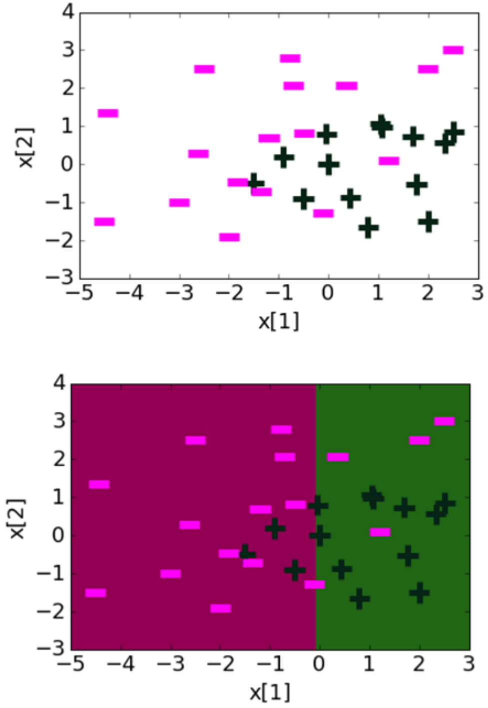
Split on Age >= 38

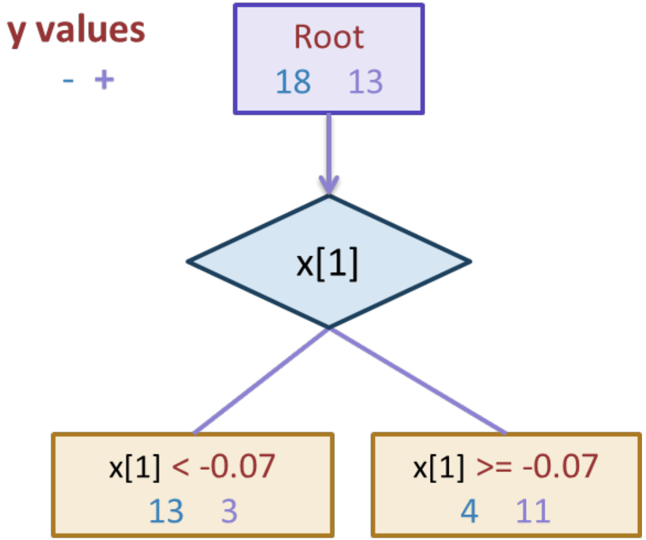


Each split partitions the 2-D space

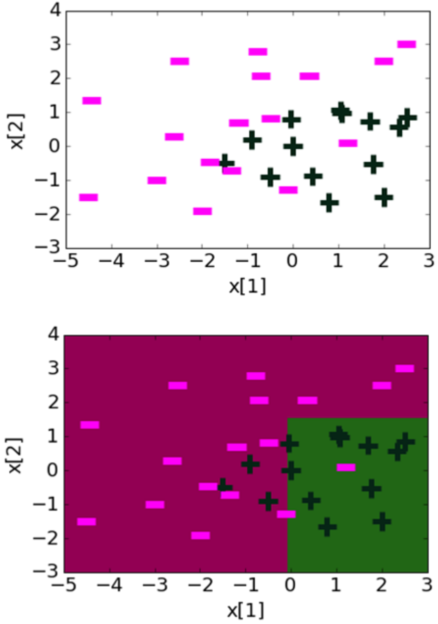


Depth 1:
Split on x[1]

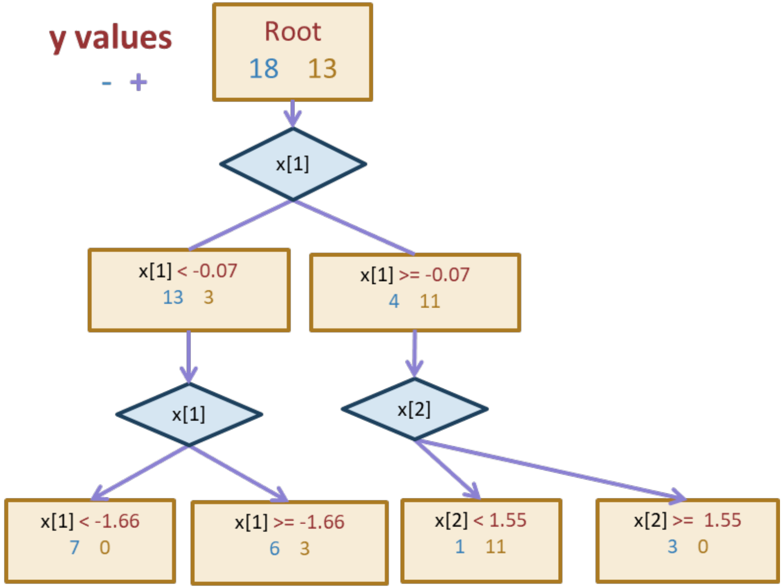




Depth 2



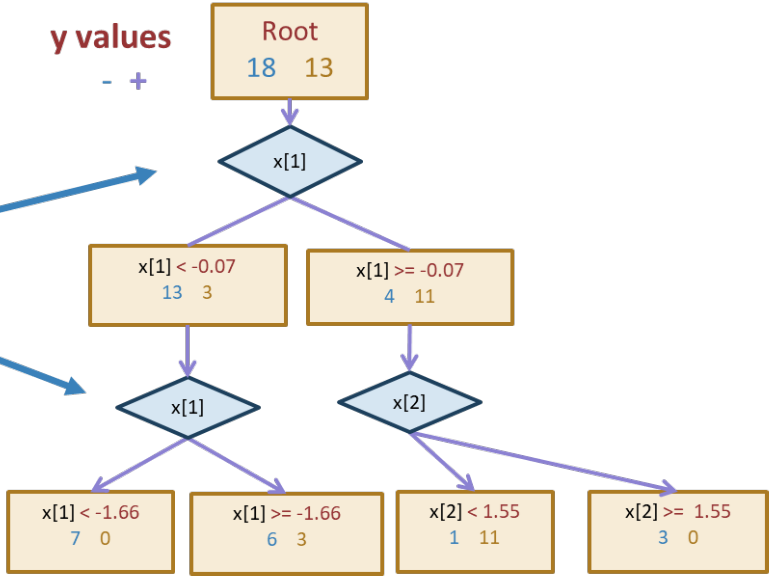
x[2]
x[1]



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

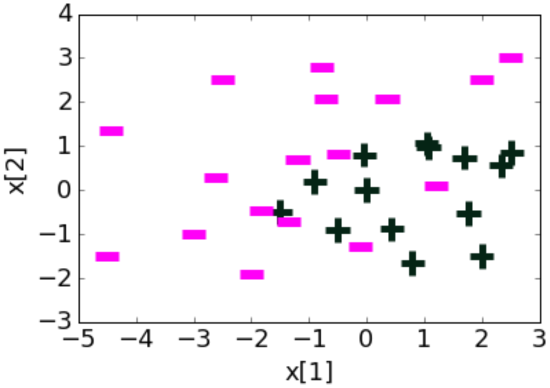
Same feature can be used to split multiple times

For threshold splits, same feature can be used multiple times

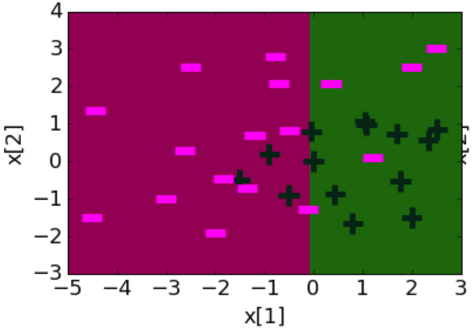


Decision boundaries at different depths

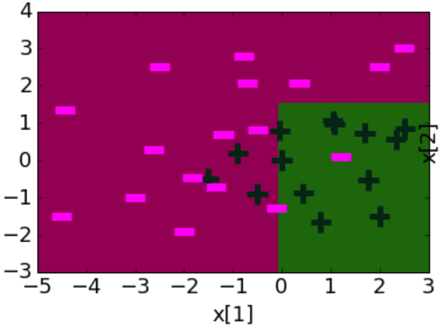
Decision boundaries can be complex!



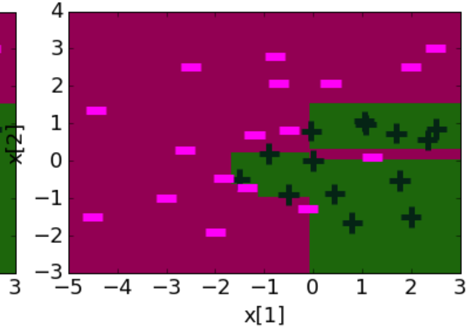
Depth 1



Depth 2



Depth 10



Advantages of Decision Tree

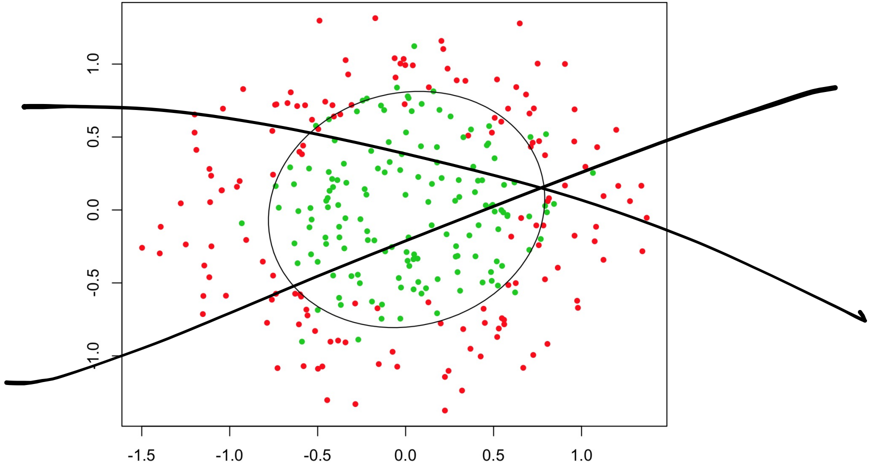
Advantages:

- Easy to interpret
- Can handle both continuous and categorical variables without preprocessing
- No normalization required as it uses rule-based approach
- Can create non-linear decision boundaries
- Can handle missing values

Disadvantages of Decision Tree

Disadvantages:

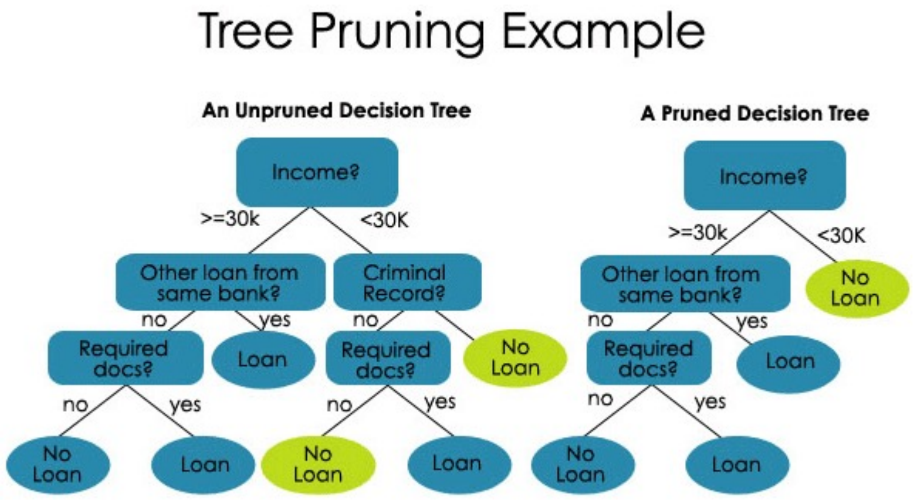
- Deep decision trees are prone to overfitting.
  - Decision boundaries are interpretable but not stable, because adding new datapoints will caused the trees to be regenerated.
  - Not suitable for large datasets due to the growing complexity
- Only allows axis-parallel rectangular decision boundaries



Overfitting prevention

Overcoming Overfitting:

- Set minimum number of data points in a node to split
- Early stopping
  - Fixed length depth
  - Maximum number of nodes
  - Stop if error does not considerably decrease
- Pruning



Recap

What you can do now:

- Define a decision tree classifier
- Interpret the output of a decision trees
- Learn a decision tree classifier using greedy & recursive algorithm
- Advantages and Disadvantages of a decision tree
- Understand different data types and necessary preprocessing steps
- Ways to overcome overfitting in decision trees