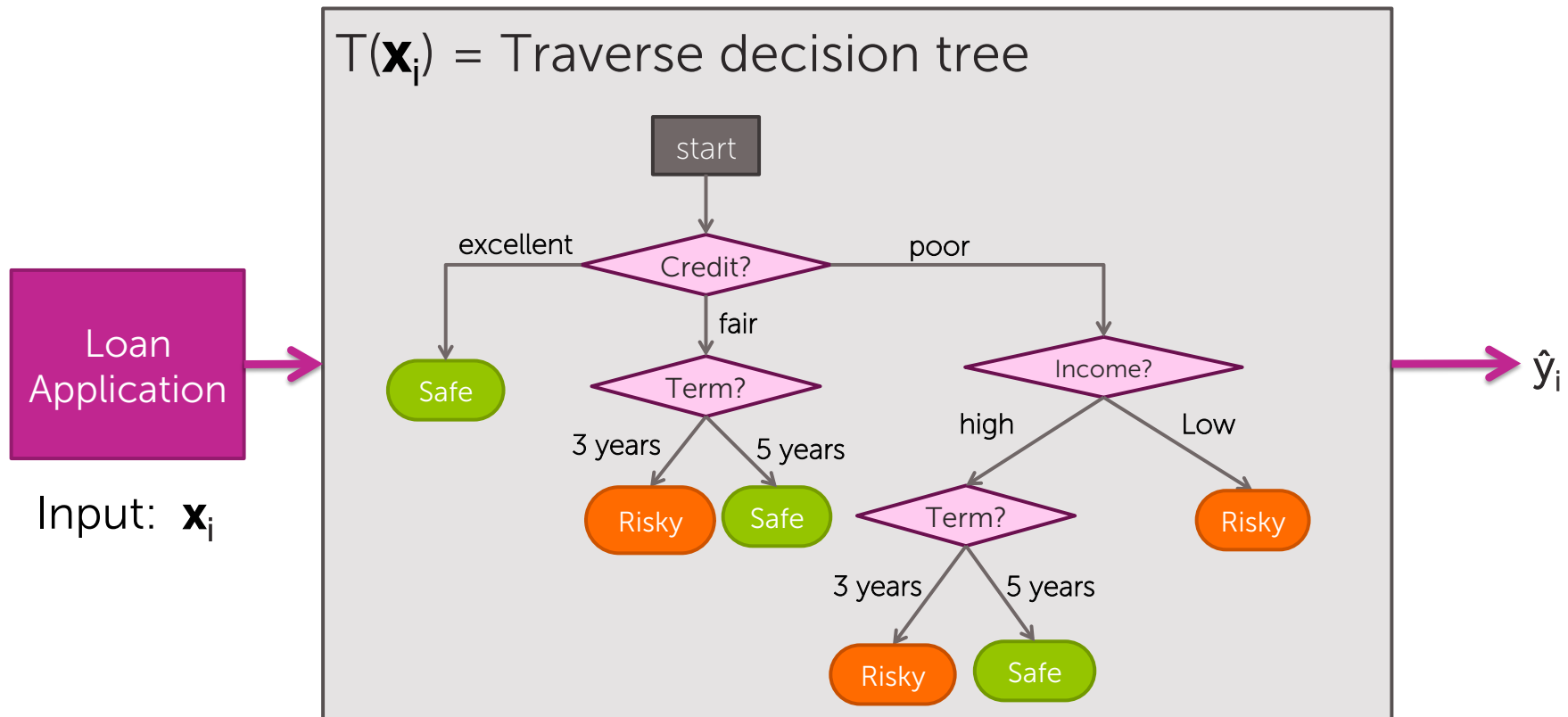




# Handling missing data

Emily Fox & Carlos Guestrin  
Machine Learning Specialization  
University of Washington

# Decision tree review



## So far: data always completely observed

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

Known x and y  
values for all  
data points

# Missing data

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

Loan application  
may be  
3 or 5 years

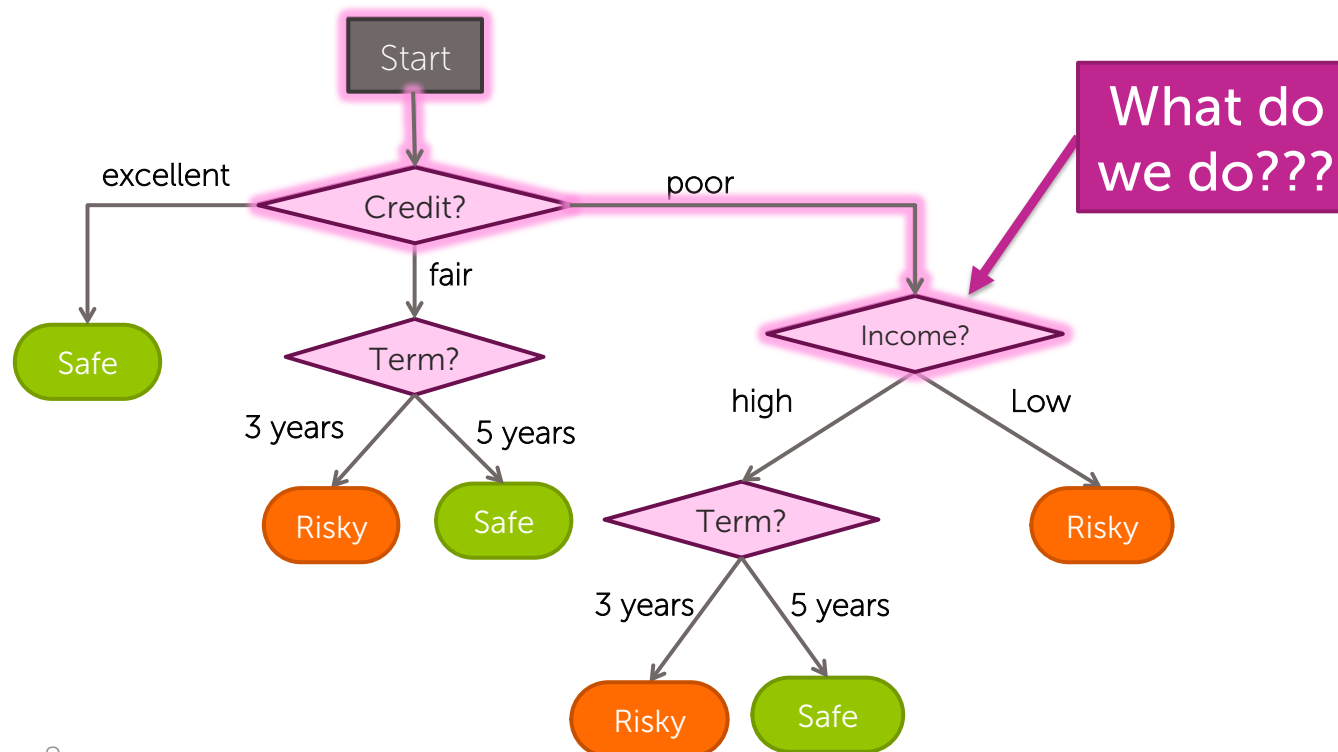


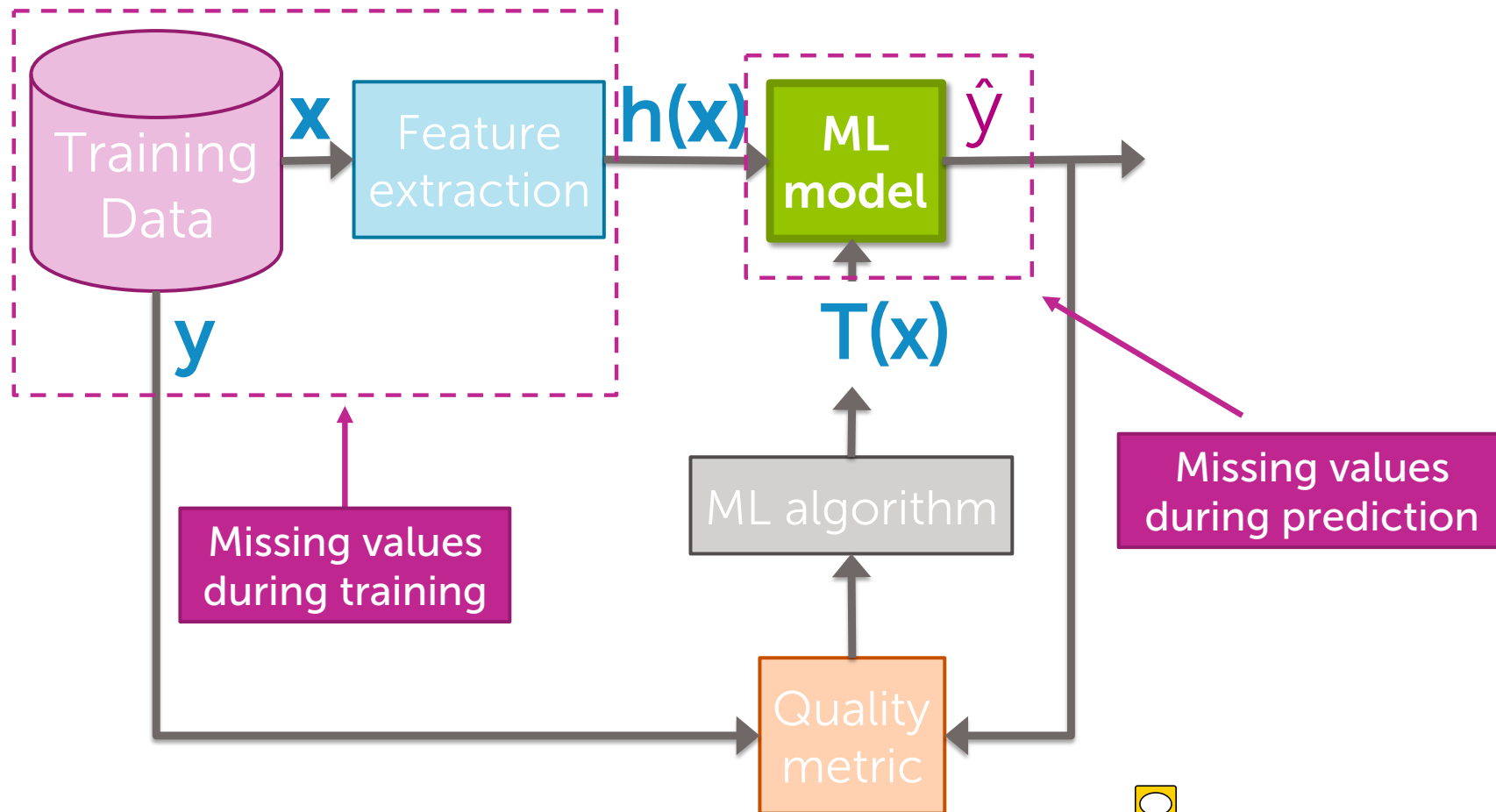
# Missing values impact **training** and **predictions**

1. **Training data:** Contains “unknown” values
2. **Predictions:** Input at prediction time contains “unknown” values

# Missing values during prediction

$\mathbf{x}_i = (\text{Credit} = \text{poor}, \text{Income} = ?, \text{Term} = 5 \text{ years})$





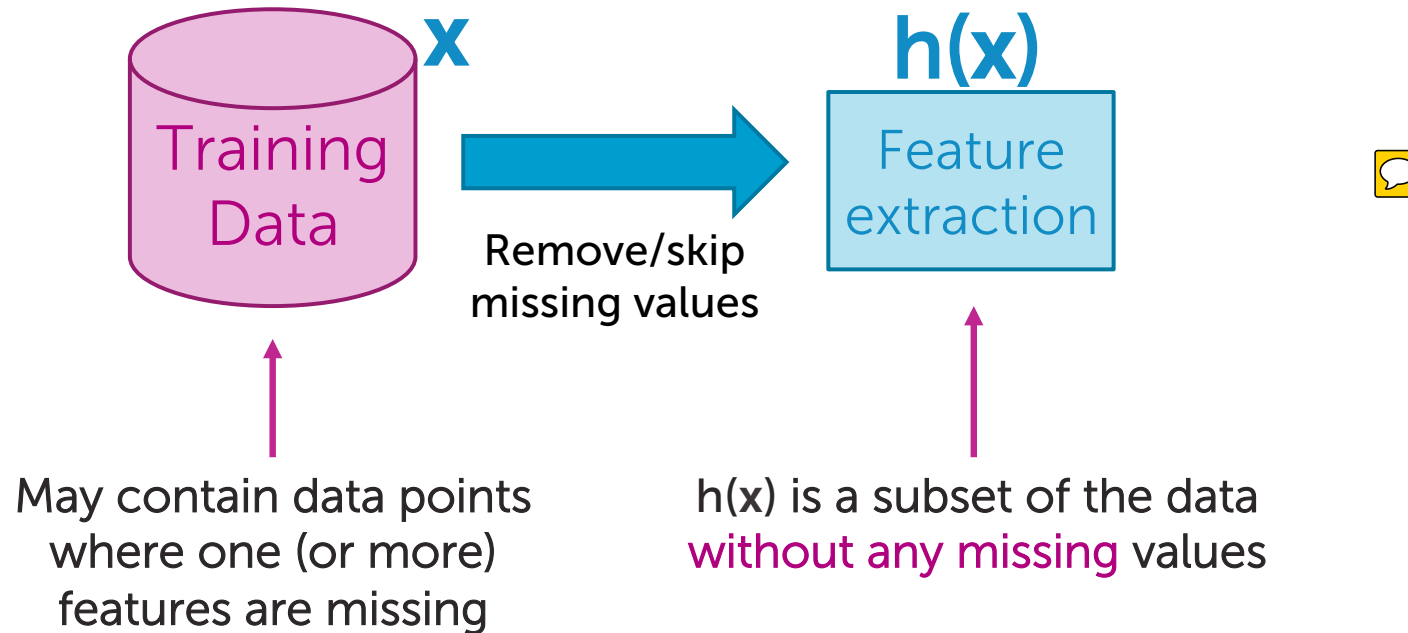


# Handling missing data

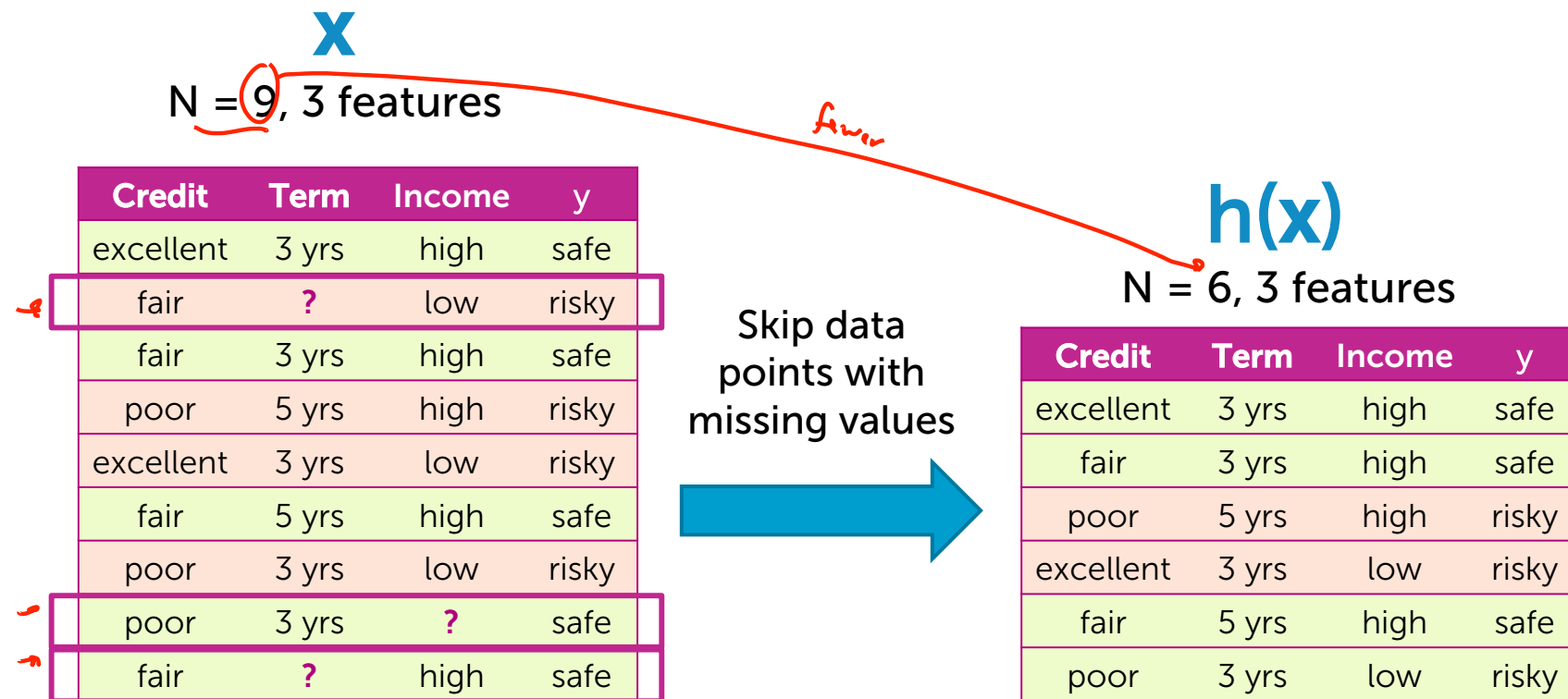
## *Strategy 1: Purification by skipping*



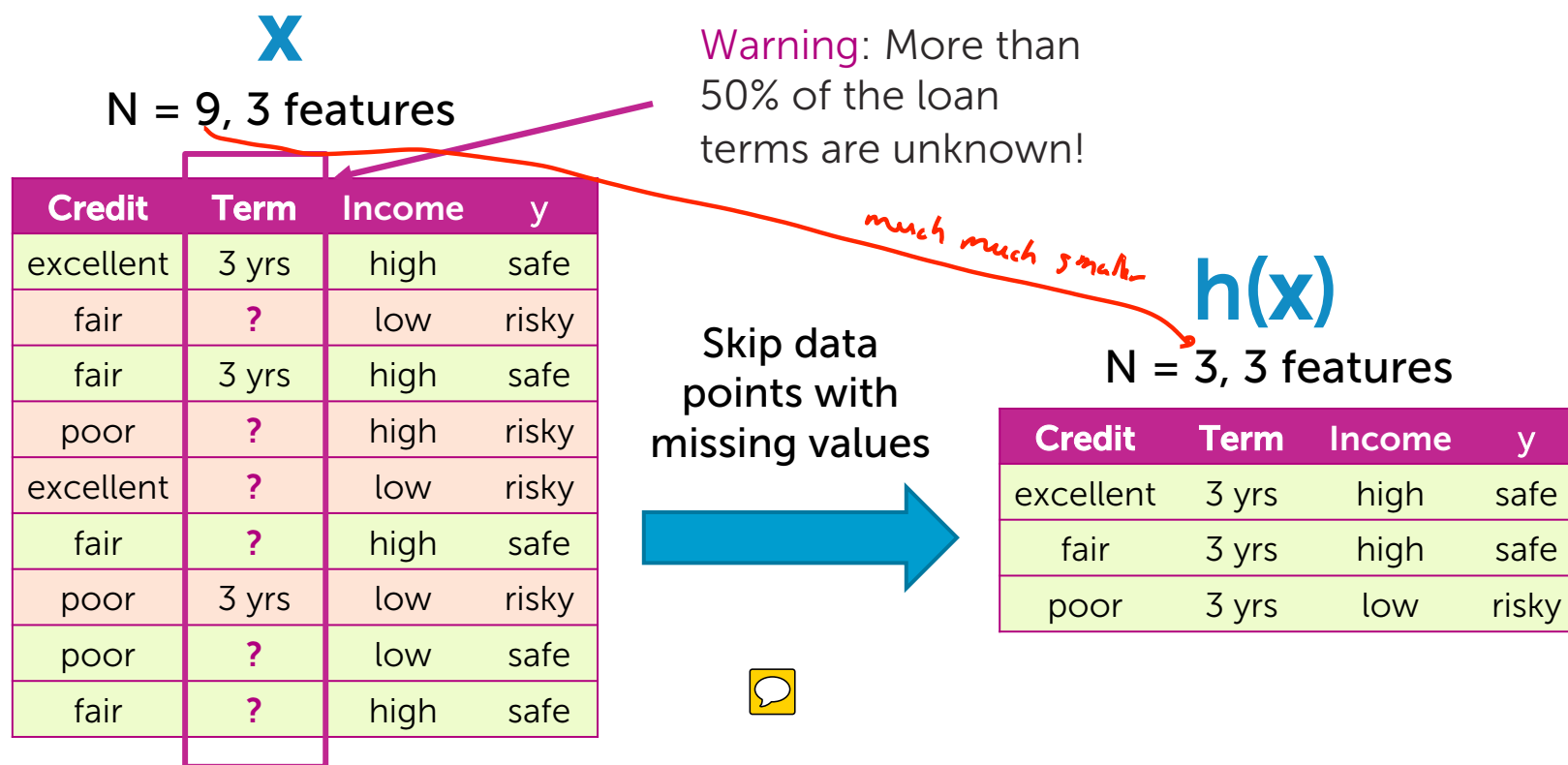
# Idea 1: Purification by skipping/removing



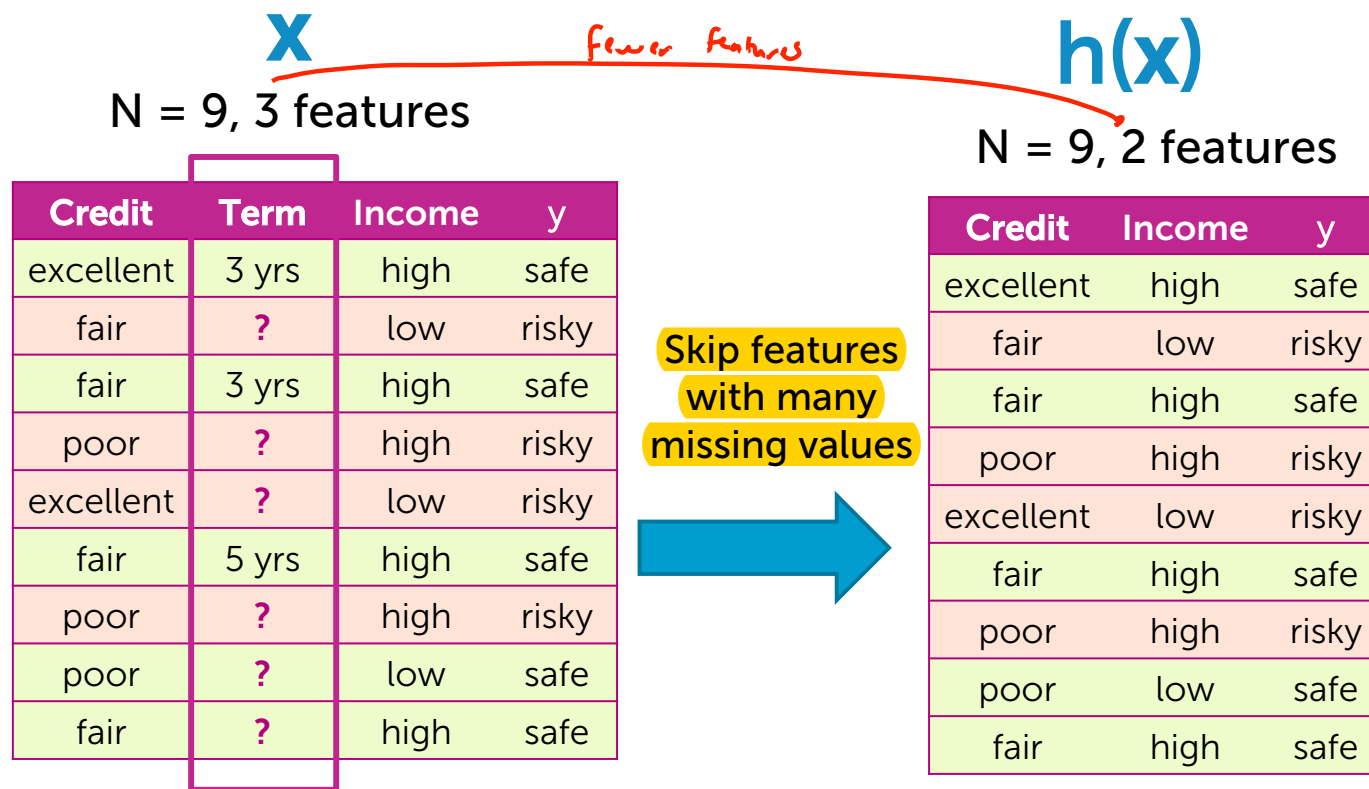
# Idea 1: Skip data points with missing values



# The challenge with Idea 1



## Idea 2: Skip features with missing values





# Missing value skipping: Ideas 1 & 2

Idea 1: Skip data points where any feature contains a missing value

- Make sure only a few data points are skipped

Idea 2: Skip an entire feature if it's missing for many data points

- Make sure only a few features are skipped

---

# Missing value skipping: Pros and Cons

## Pros

- Easy to understand and implement
- Can be applied to any model (decision trees, logistic regression, linear regression,...)

## Cons

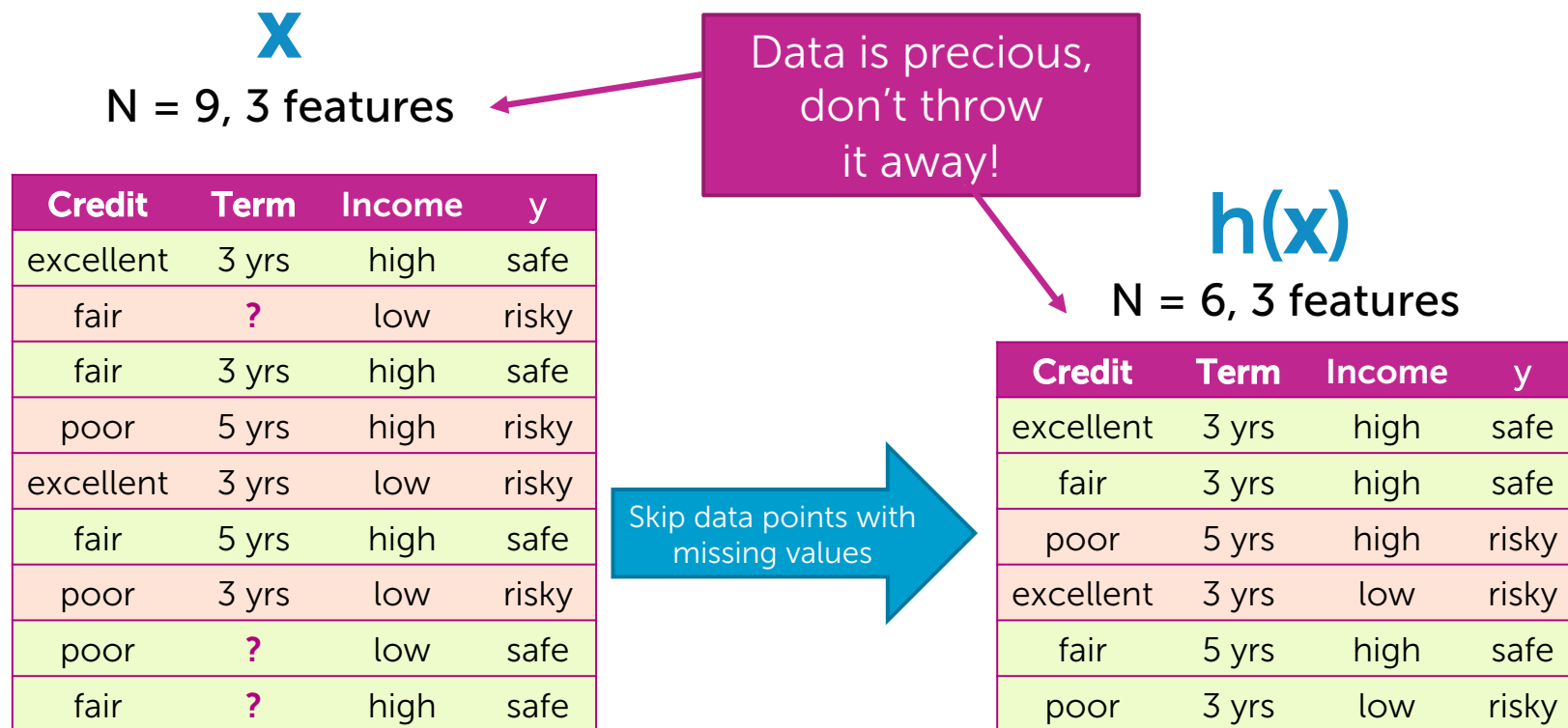
- Removing data points and features may remove important information from data
- Unclear when it's better to remove data points versus features
- Doesn't help if data is missing at prediction time



# Handling missing data

## *Strategy 2: Prification by imputing*

# Main drawback of skipping strategy



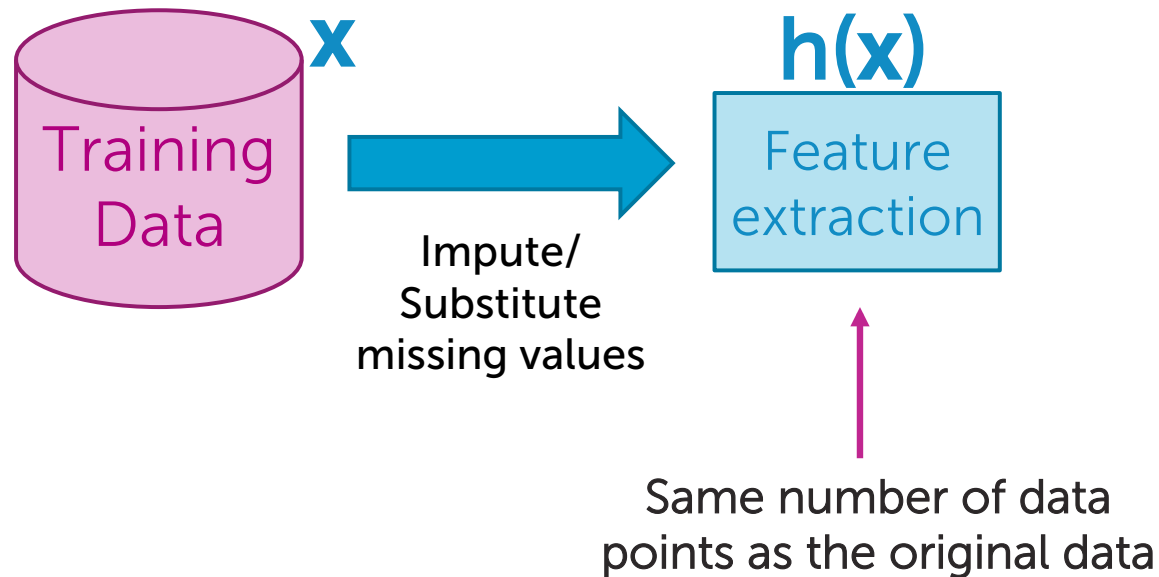


# Can we keep all the data?

credit	term	income	y
excellent	3 yrs	high	safe
fair	?	low	risky
fair	3 yrs	high	safe
poor	5 yrs	high	risky
excellent	3 yrs	low	risky
fair	5 yrs	high	safe
poor	3 yrs	high	risky
poor	?	low	safe
fair	?	high	safe

Use other data points in  $\mathbf{x}$  to "guess" the "?"

## Idea 2: Purification by imputing



## Idea 2: Imputation/Substitution

N = 9, 3 features

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

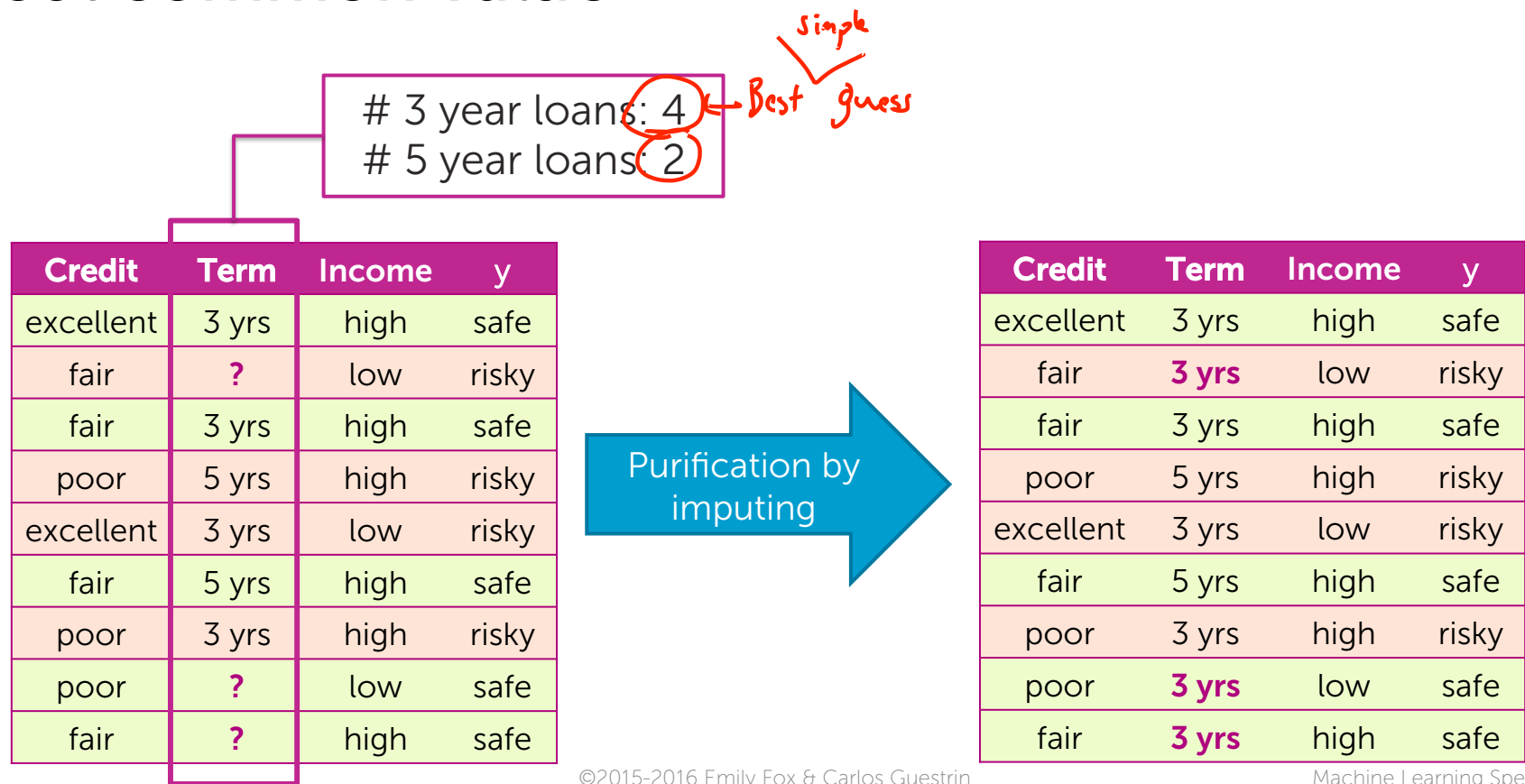
Fill in each missing value with a calculated guess



N = 9, 3 features

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

## Example: Replace ? with most common value



# Common (simple) rules for purification by imputation

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

Impute each feature with missing values:

1. Categorical features use mode: Most popular value (mode) of non-missing  $x_i$
2. Numerical features use average or median: Average or median value of non-missing  $x_i$

Many advanced methods exist,  
e.g., expectation-maximization (EM) algorithm





# Missing value imputation: Pros and Cons

## Pros

- Easy to understand and implement
- Can be applied to any model  
(decision trees, logistic regression, linear regression,...)
- Can be used at prediction time: use same imputation rules

## Cons

- May result in systematic errors

**Example:** Feature “age” missing in all banks in Washington by state law

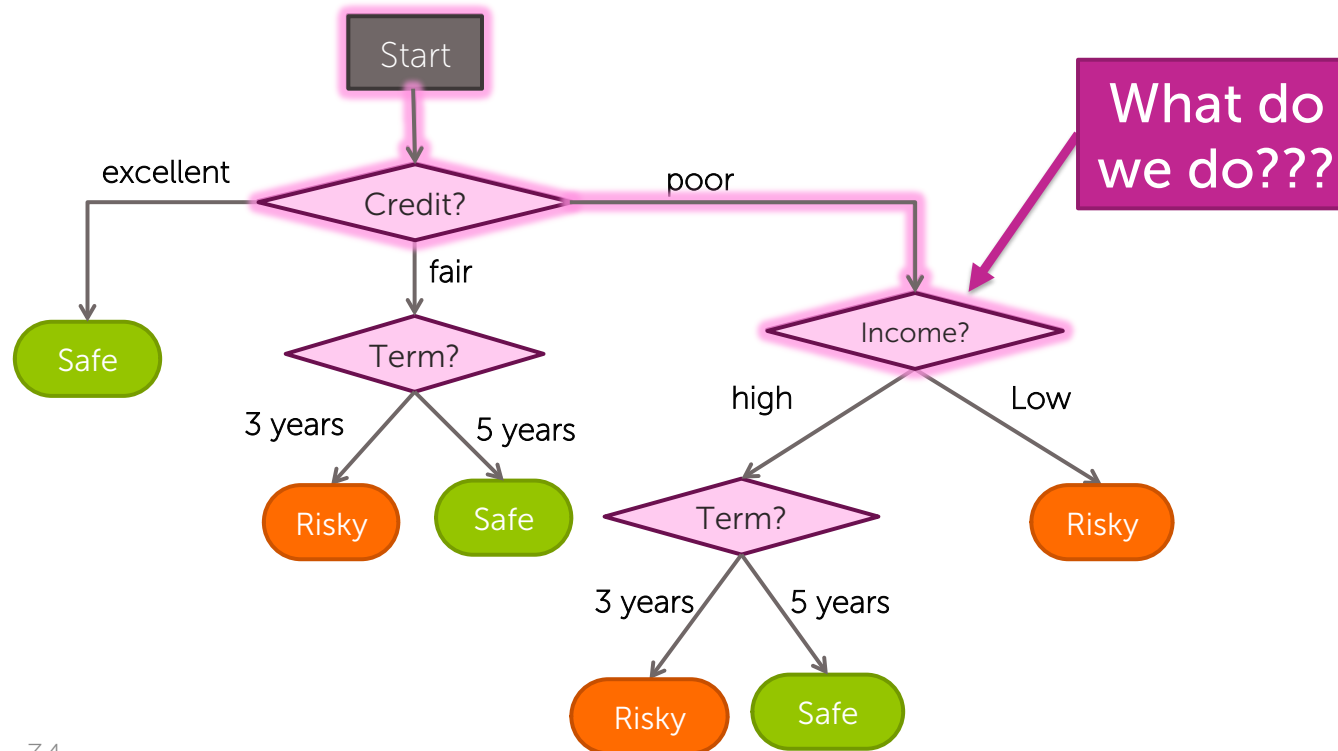
# Handling missing data

*Strategy 3: Adapt learning algorithm  
to be robust to missing values*



# Missing values during prediction: *revisited*

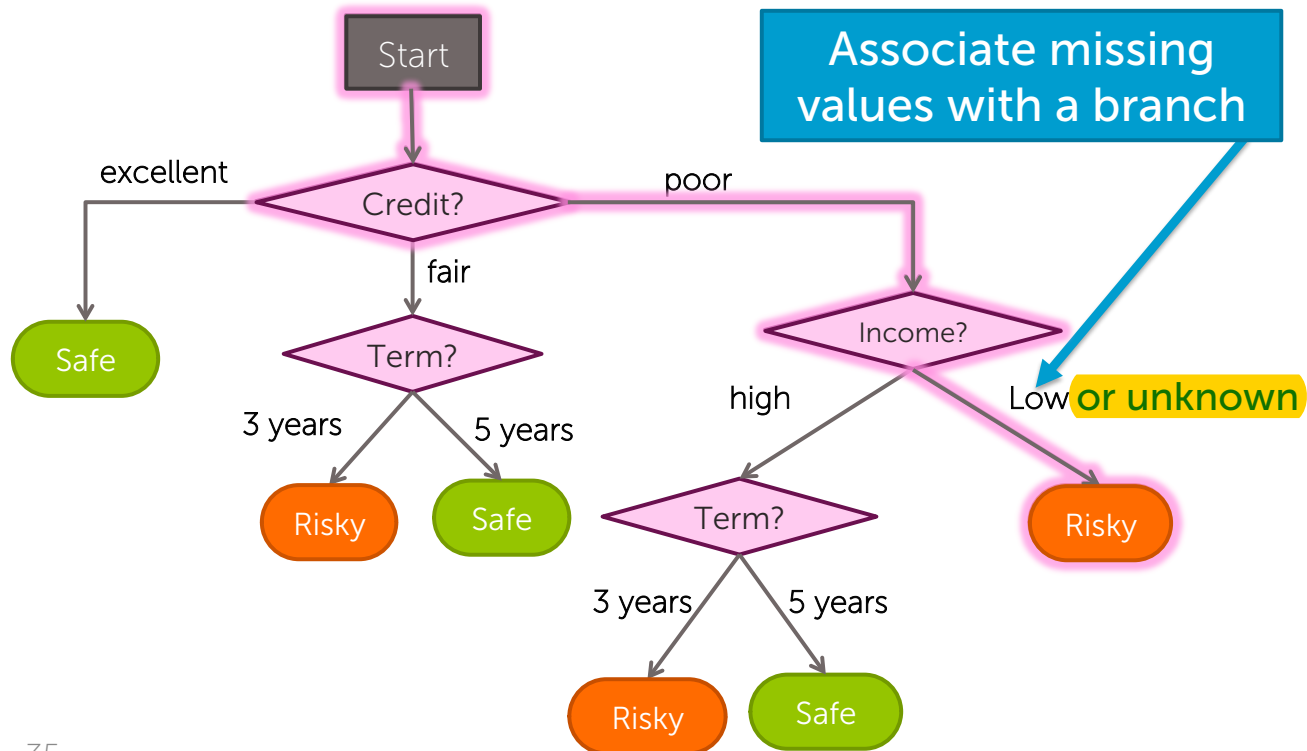
$\mathbf{x}_i = (\text{Credit} = \text{poor}, \text{Income} = ?, \text{Term} = 5 \text{ years})$



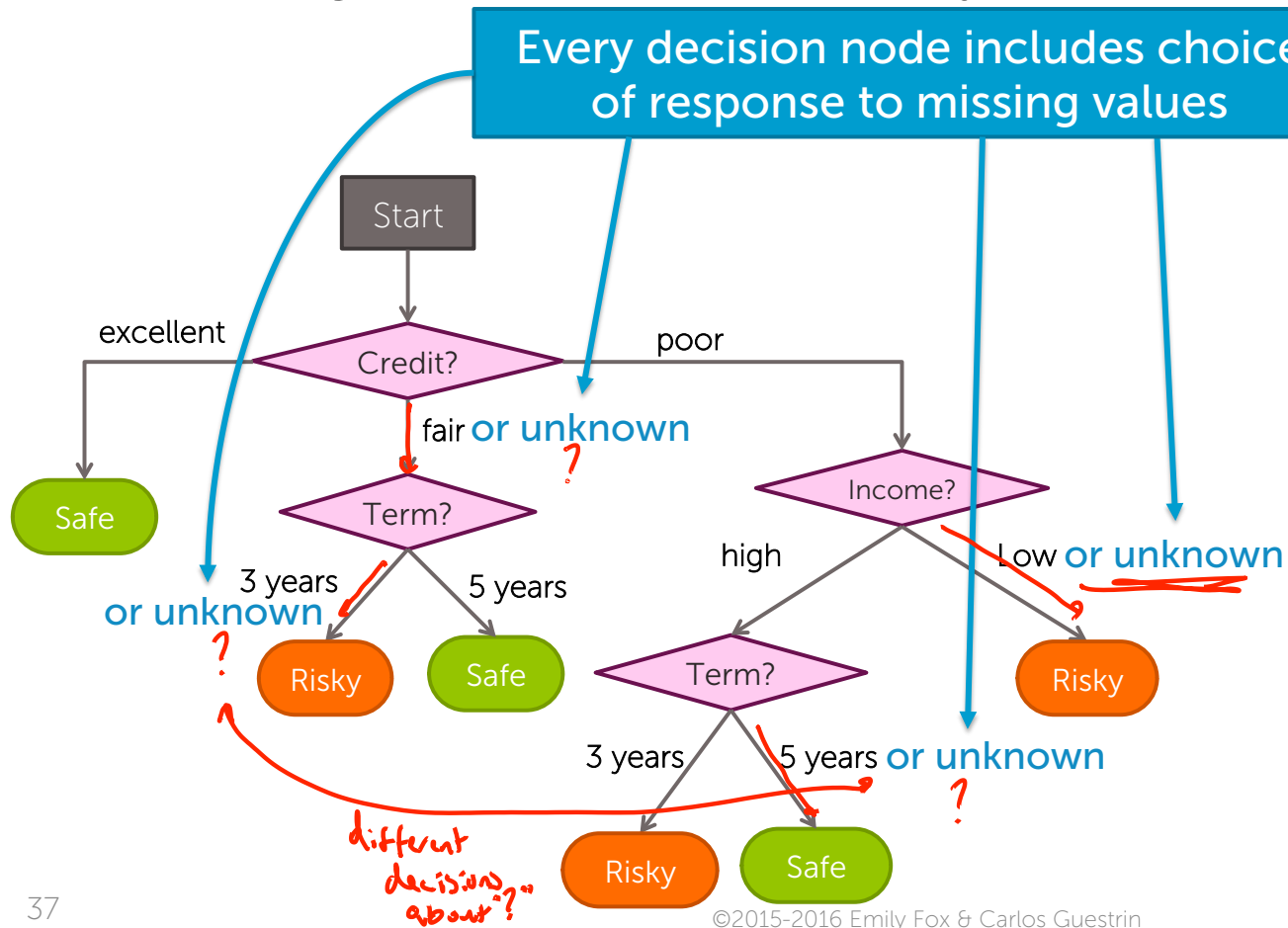


# Add missing values to the tree definition

$\mathbf{x}_i = (\text{Credit} = \text{poor}, \text{Income} = ?, \text{Term} = 5 \text{ years})$

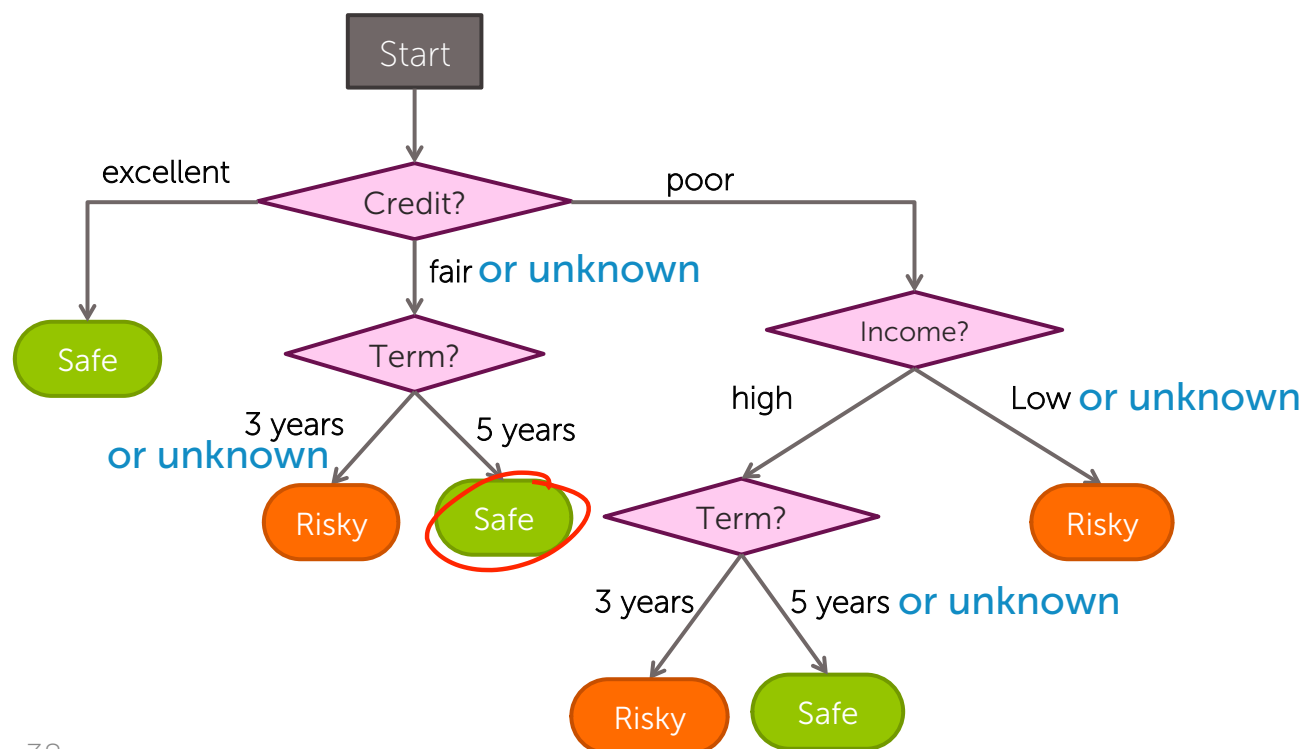


## Add missing value choice to every decision node



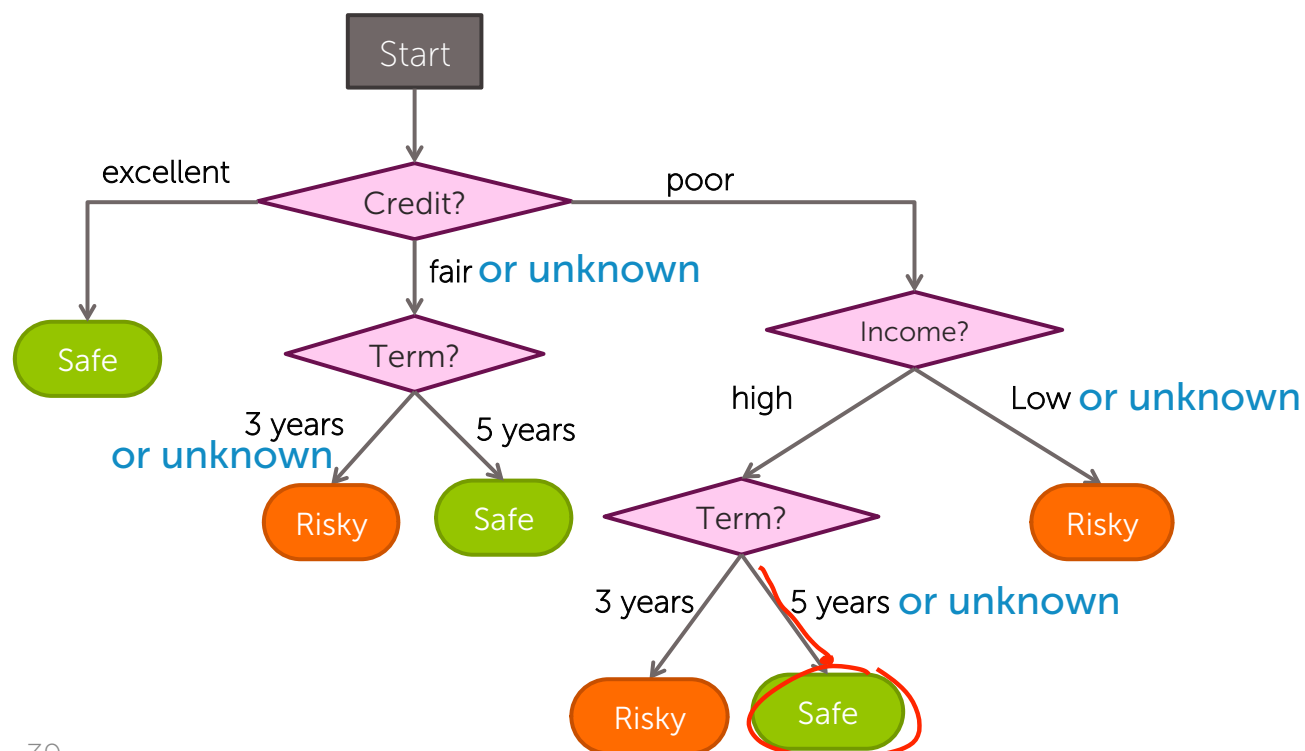
## Prediction with missing values becomes simple

$\mathbf{x}_i = (\text{Credit} = ?, \text{Income} = \text{high}, \text{Term} = 5 \text{ years})$



## Prediction with missing values becomes simple

$\mathbf{x}_i = (\text{Credit} = \text{poor}, \text{Income} = \text{high}, \text{Term} = ?)$



# Explicitly handling missing data by learning algorithm: Pros and Cons

## Pros

- Addresses training and prediction time
- **More accurate predictions**



## Cons

- Requires modification of learning algorithm
  - **Very simple for decision trees**

## Feature split selection with missing data



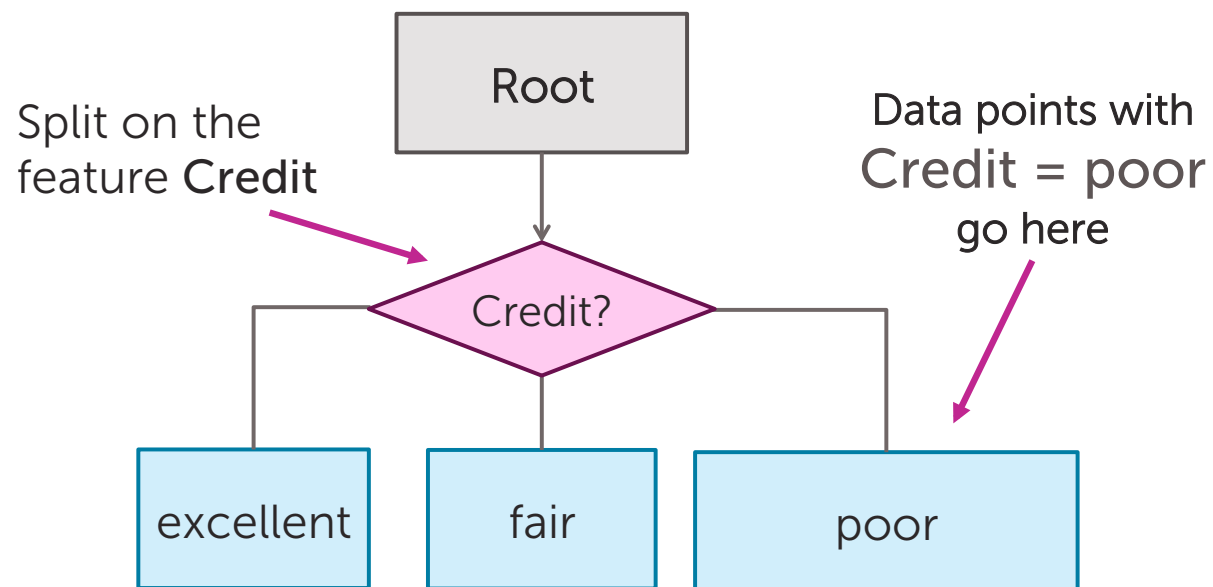
# Greedy decision tree learning

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

Pick feature split  
leading to lowest  
classification error

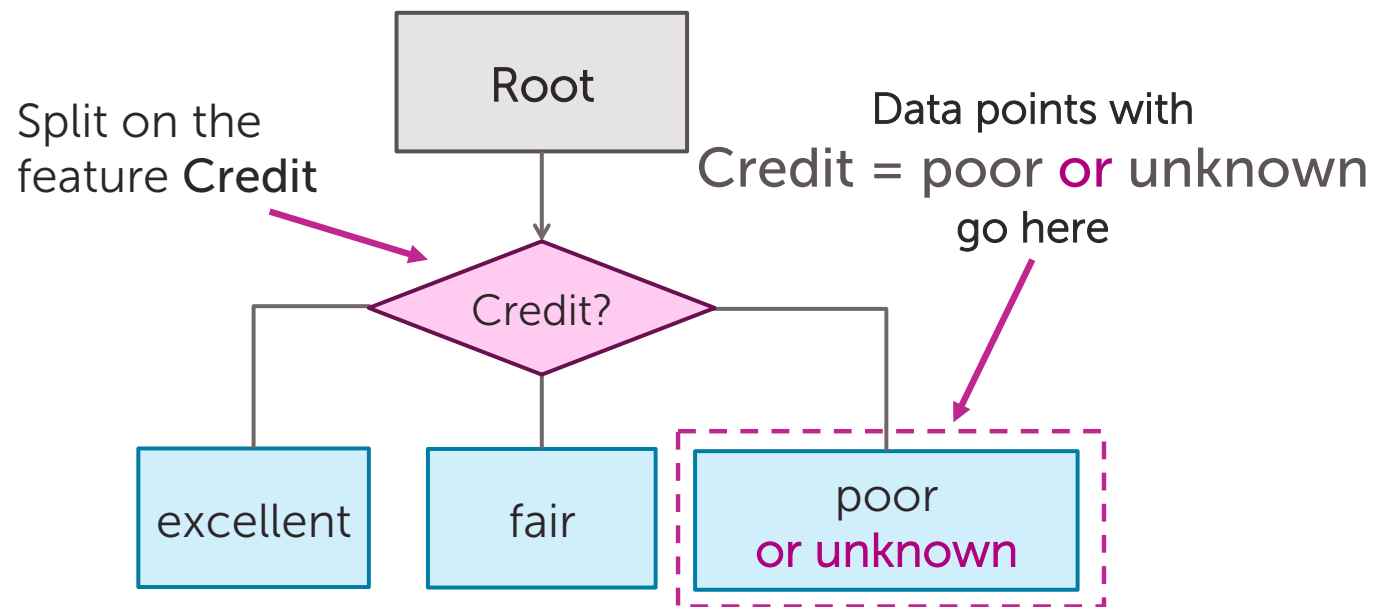
Must select feature &  
branch for missing values!

# Feature split (without missing values)

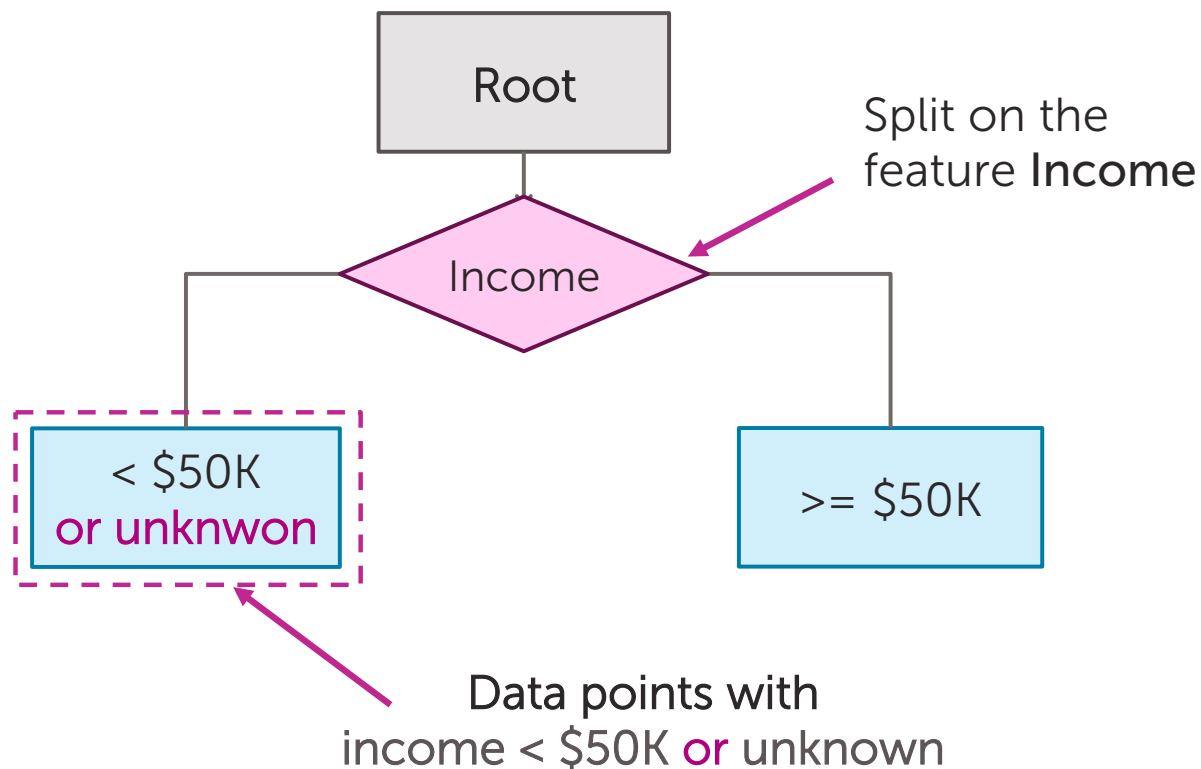




# Feature split (**with** missing values)



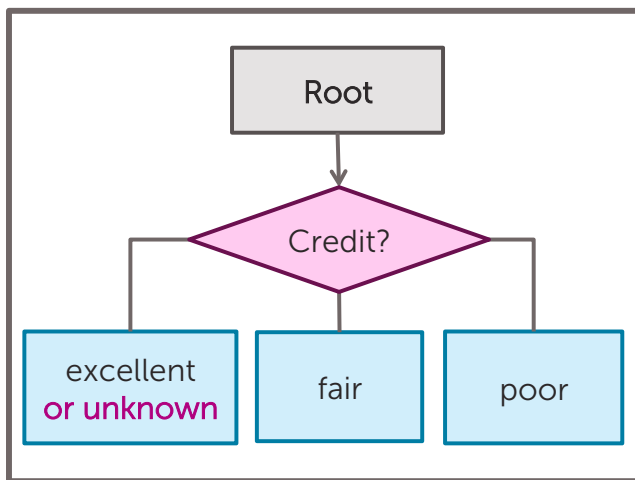
# Missing value handling in threshold splits



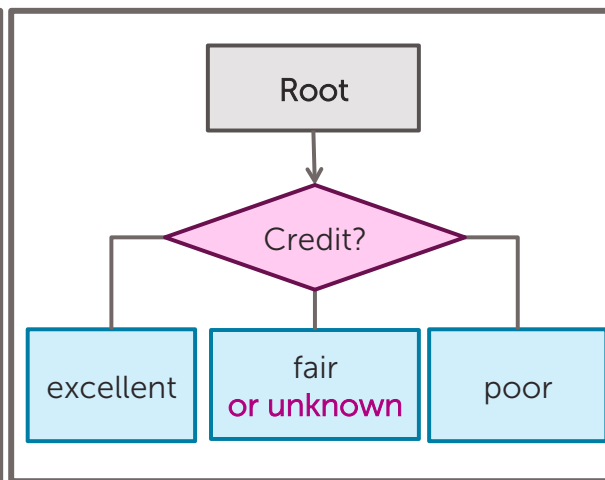
## Should **missing** go left, right, or middle?

Choose branch that leads to  
lowest classification error!

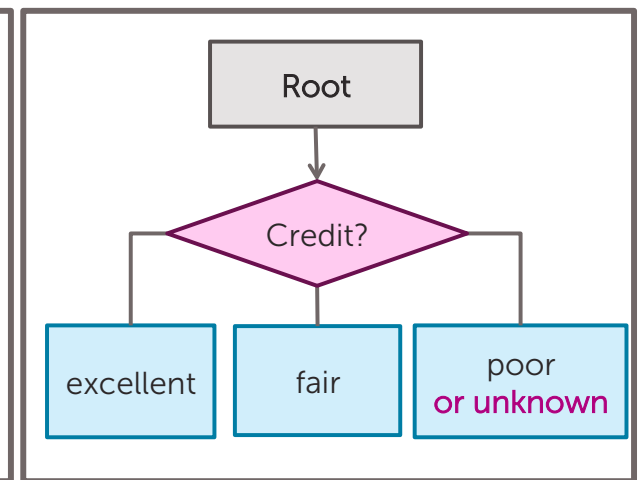
Choice 1: Missing values go  
with Credit=excellent



Choice 2: Missing values  
go with Credit=fair



Choice 3: Missing values  
go with Credit=poor

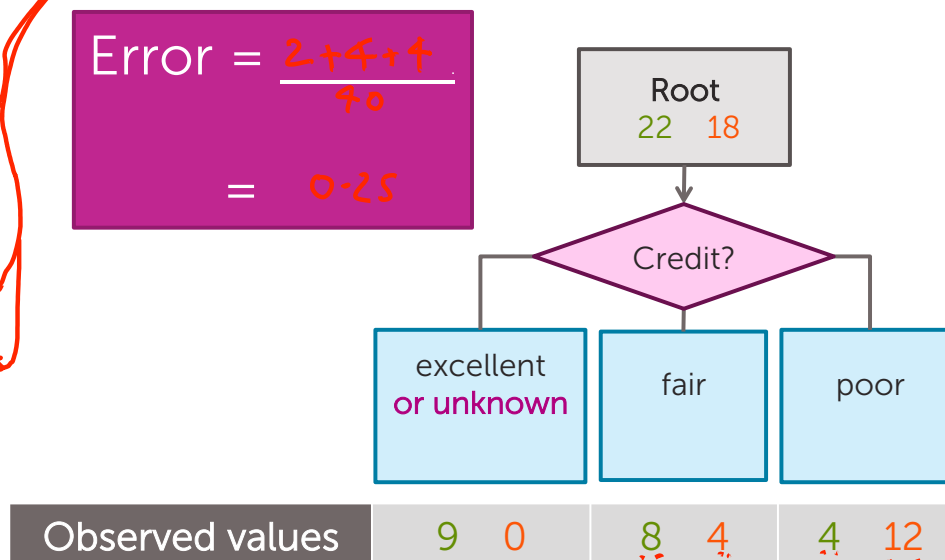


# Computing classification error of decision stump with missing data

N = 40, 3 features

Credit	Term	Income	y
excellent	3 yrs	high	safe
?	5 yrs	low	<u>risky</u>
fair	3 yrs	high	safe
poor	5 yrs	high	risky
?	3 yrs	low	<u>risky</u>
?	5 yrs	low	<u>safe</u>
poor	3 yrs	high	risky
poor	5 yrs	low	safe
fair	3 yrs	high	safe
...	...	...	...

$$\text{Error} = \frac{2+4+4}{40} = 0.25$$

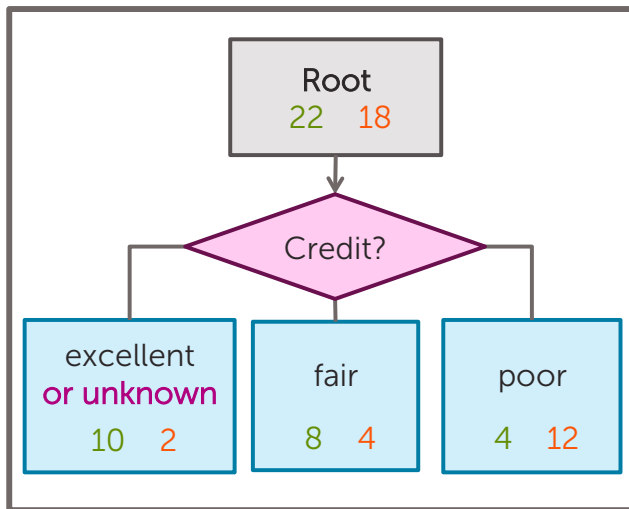


0 0 0

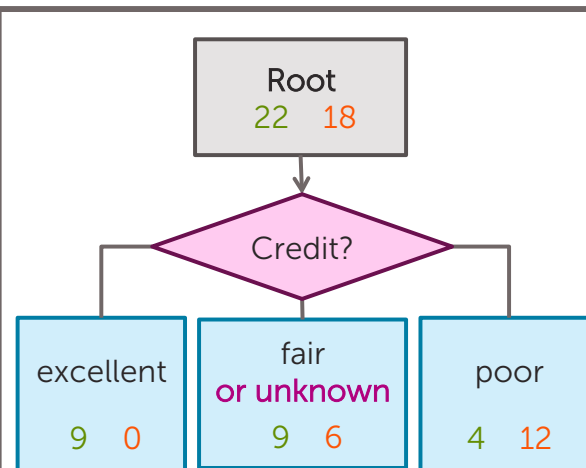
# Use classification error to decide

Best choice  $\rightarrow$  assign "unknown"  
to Credit = poor

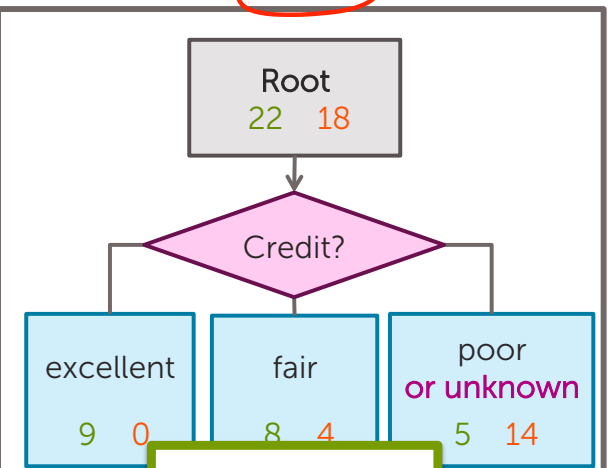
Choice 1: error = 0.25



Choice 2: error = 0.25



Choice 3: error = 0.225



**WINNER**

# Feature split selection algorithm with missing value handling

- Given a subset of data  $M$  (a node in a tree)
- For each feature  $h_i(x)$ :
  1. Split data points of  $M$  where  $h_i(x)$  is *not* “unknown” according to feature  $h_i(x)$
  2. Consider assigning data points with “unknown” value for  $h_i(x)$  to each branch
    - A. Compute classification error split & branch assignment of “unknown” values
- Chose feature  $h^*(x)$  & branch assignment of “unknown” with lowest classification error

# Summary of handling missing data



# What you can do now...

Describe common ways to handling missing data:

1. Skip all rows with any missing values
2. Skip features with many missing values
3. Impute missing values using other data points

Modify learning algorithm (**decision trees**) to handle missing data:

1. Missing values get added to one branch of split
2. Use classification error to determine where missing values go



---

# Thank you to Dr. Krishna Sridhar



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