# Decision Tree Classification

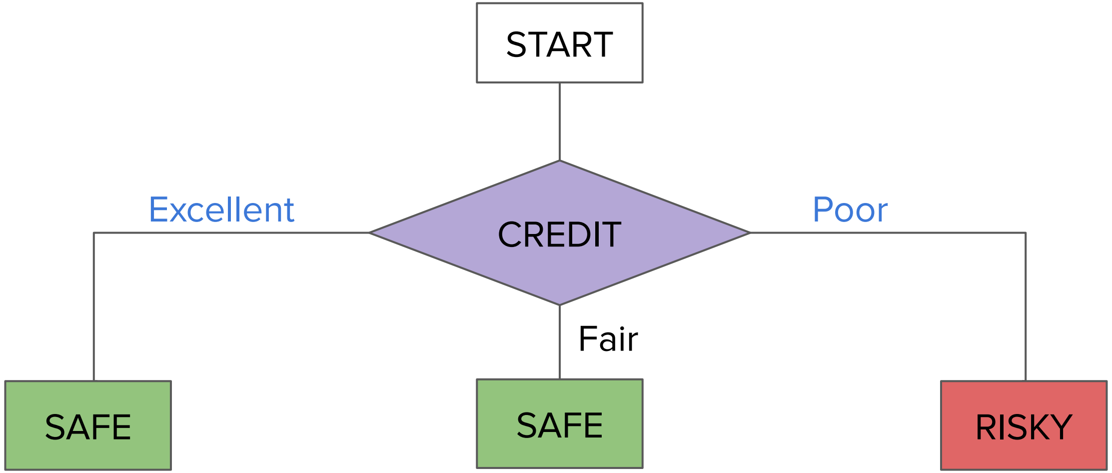
Guest Lecturer: Joshua Ervin

# Example: Predicting potential loan defaults

- Data: discrete for now (e.g. credit rating: excellent, fair, poor)
- Goal: Given a new loan application, predict whether or not the applicant will default on their loan:

| Credit | Term | Income | Y |
|---|---|---|---|
| excellent | 3 years | high | safe |
| fair | 5 years | low | risky |
| fair | 3 years | high | risky |
| poor | 3 years | high | risky |

# Decision Tree



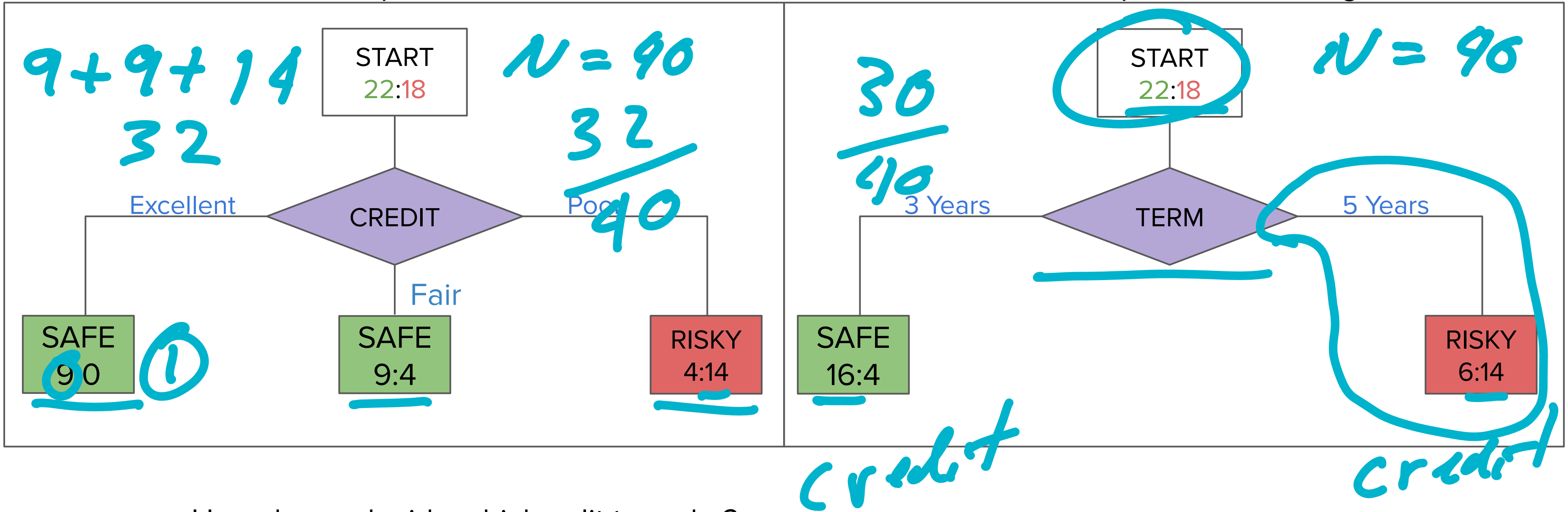

- Internal Node: A node that tests a feature
- Branch: Splits input data based on the value of a feature
- Leaf: Assigns a class to data (i.e. SAFE, RISKY)

pollev.com/cse416

# Decision Stumps

Choice 1: Split on Credit ✓

Choice 2: Split on Term Length




- How do we decide which split to make?
- Always pick the split which maximizes accuracy

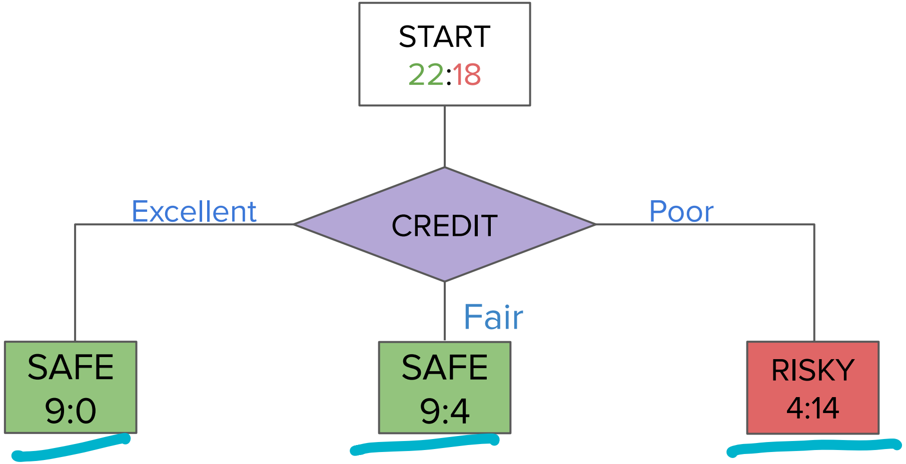
$$accuracy = \frac{\#correct\ predictions}{\#data\ points}$$

# Greedy Algorithm for Growing a Decision Tree

- Start with a single root node
- Repeat while the stopping rule is not met
  - Choose a feature x[i] to split that maximizes classification accuracy
- Stopping Rule:
  - 1) Do not branch if all data has the same label (pure)
  - 2) We have already split on that feature before

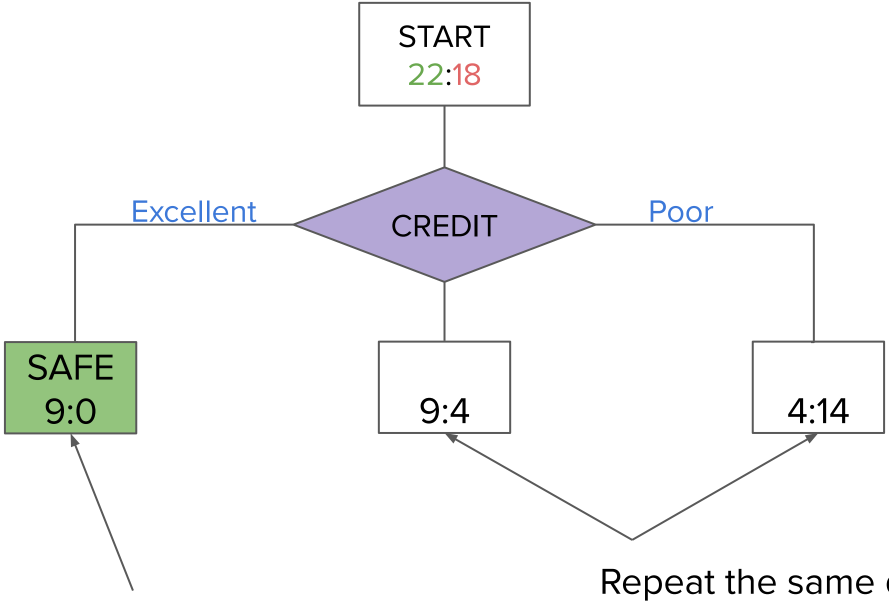
# Greedy Algorithm

Classification Accuracy: 80%



# Greedy Algorithm

Classification Accuracy:

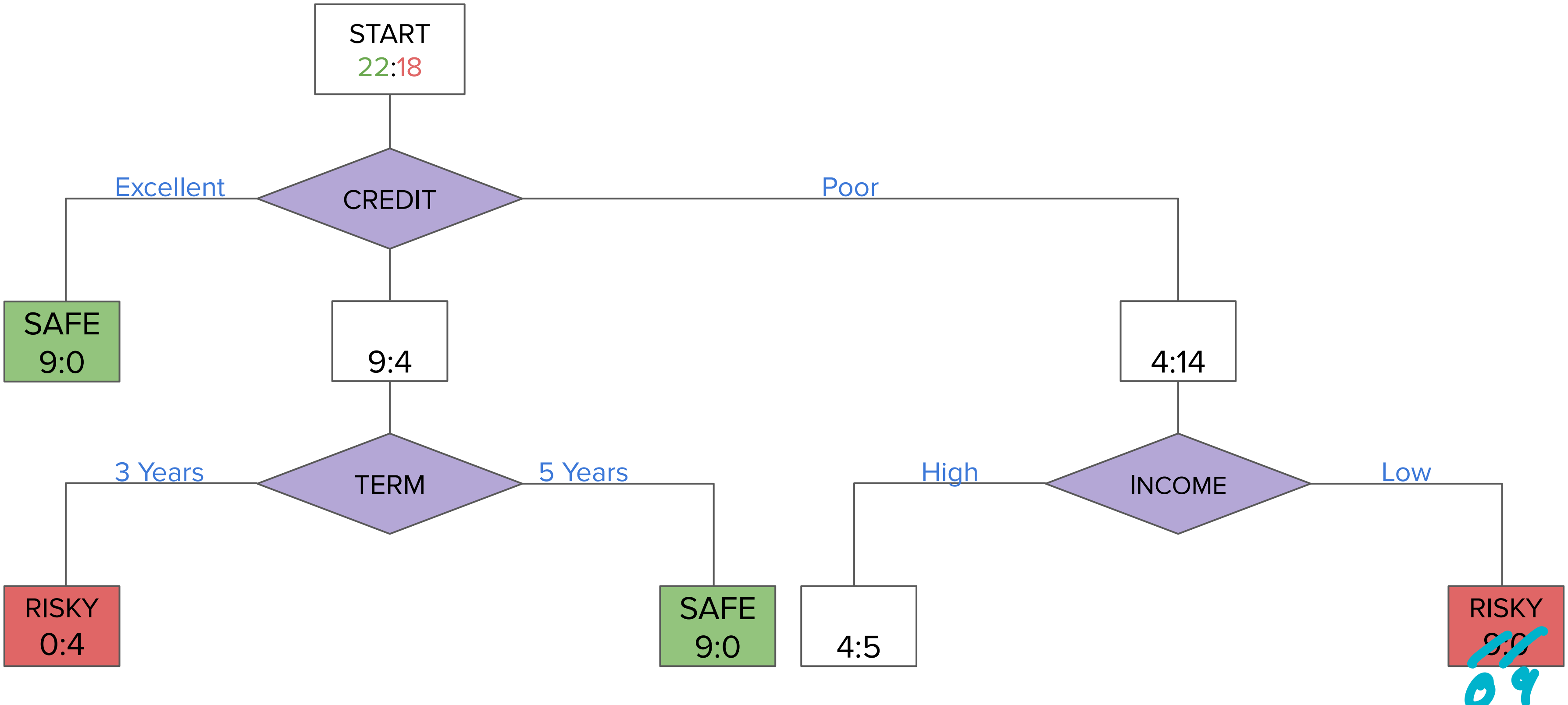



Why is it ok to stop here?

Repeat the same decision tree stump-building process with this subset of data.

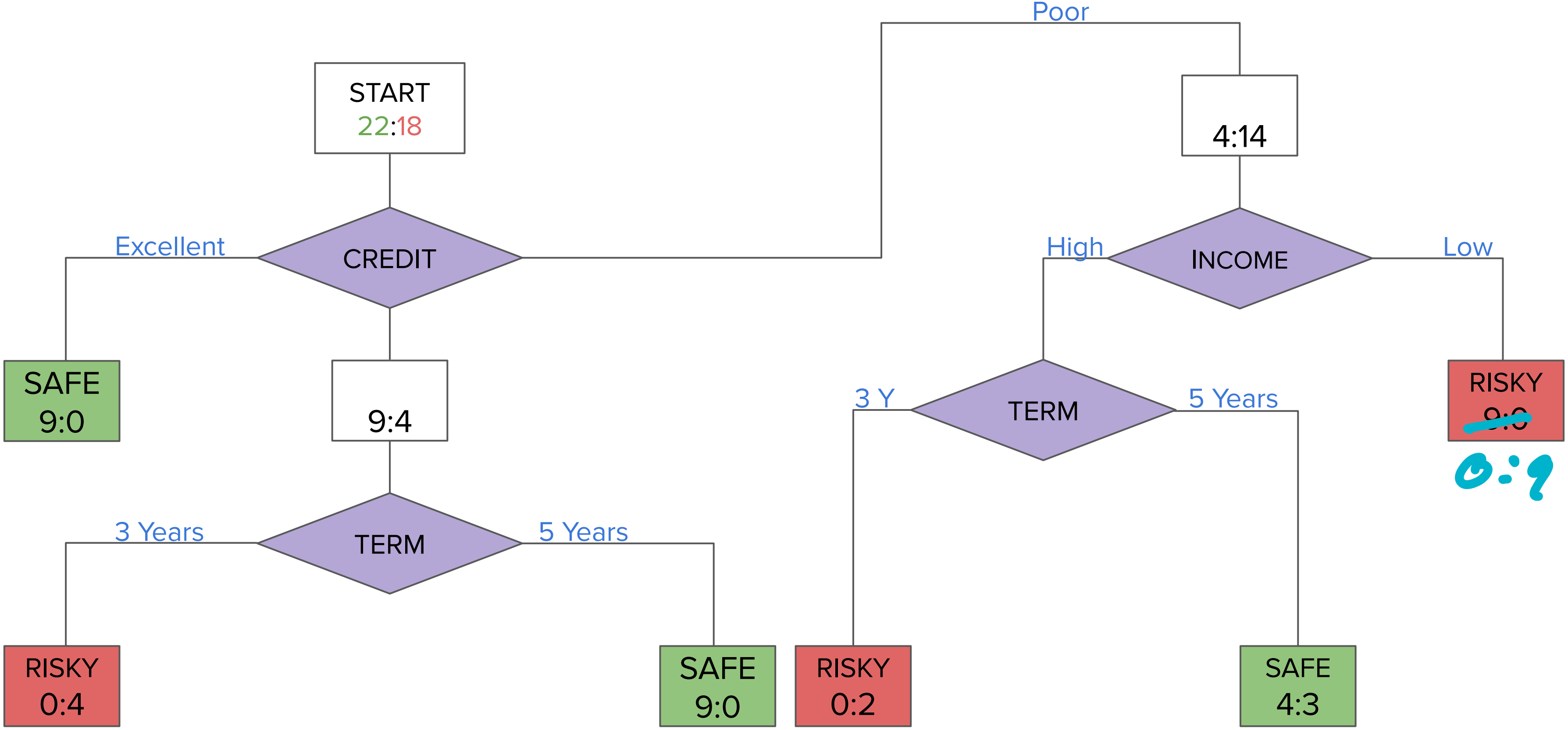
# Greedy Algorithm

Classification Accuracy:



# Greedy Algorithm

Classification Accuracy: 92.5%



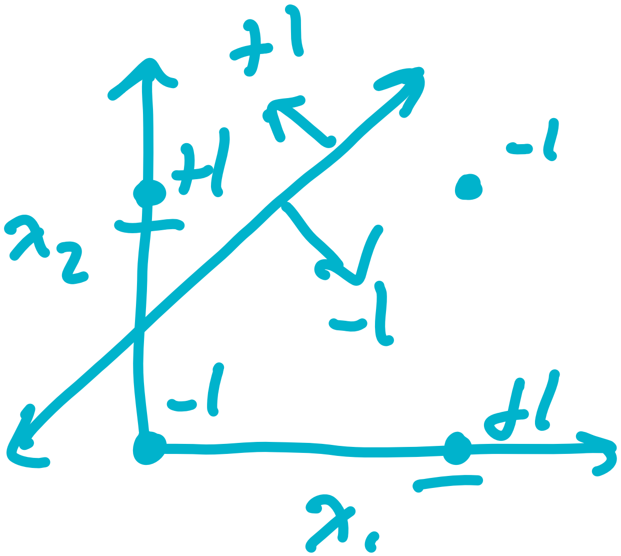
# Early Stopping Rules

"exactly one of ..."

- Stopping Rules:
  - 1) Do not branch if all data has the same label (pure)
  - 2) We have already split on that feature before
  - **3*) If adding a branch does not increase accuracy, should we still branch?**

XOR

| x[1] | x[2] | y |
|---|---|---|
| -1 | -1 | -1 |
| -1 | +1 | +1 |
| +1 | -1 | +1 |
| +1 | +1 | -1 |



# XOR: Root

| x[1] | x[2] | y |
|---|---|---|
| -1 | -1 | -1 |
| -1 | +1 | +1 |
| +1 | -1 | +1 |
| +1 | +1 | -1 |

KEY
+1:-1

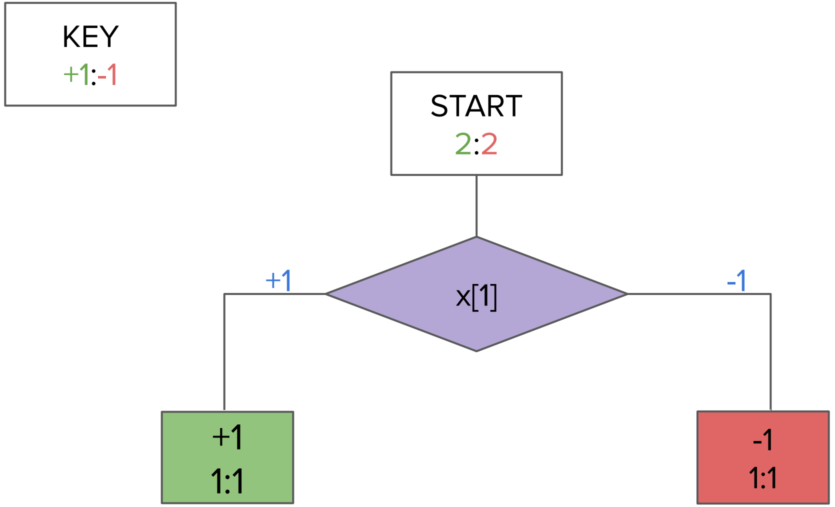
+1
2:2

| # Levels | Accuracy |
|---|---|
| 0 | 50% |
| 1 | ? |
| 2 | ? |

# XOR: 1 Split

| x[1] | x[2] | y |
|---|---|---|
| -1 | -1 | -1 |
| -1 | +1 | +1 |
| +1 | -1 | +1 |
| +1 | +1 | -1 |

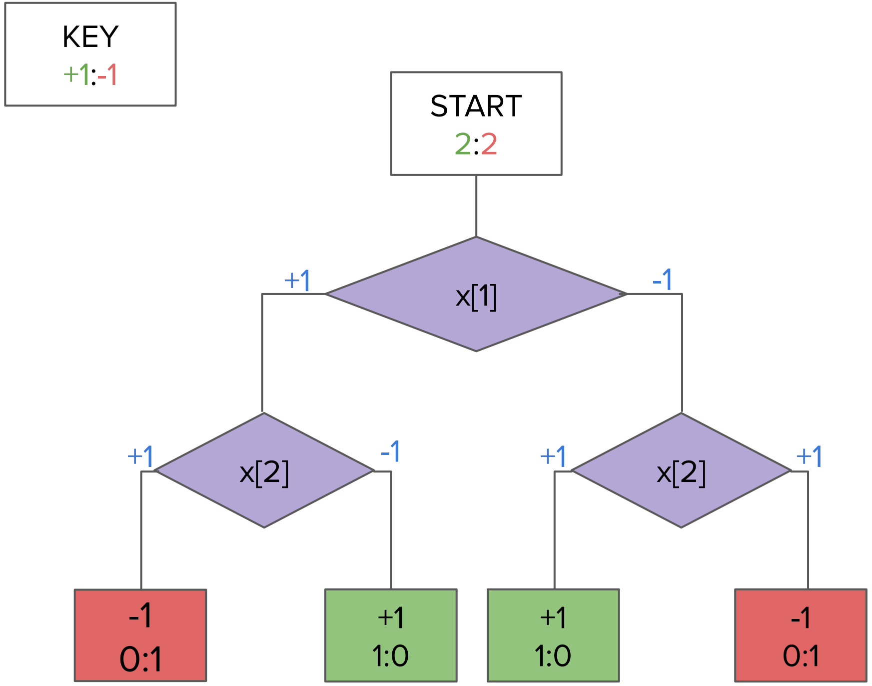
| # Levels | Accuracy |
|---|---|
| 0 | 50% |
| 1 | 50% |
| 2 | ? |



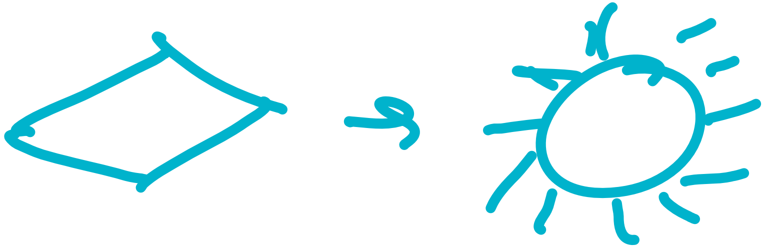
# XOR: 2 Splits

| x[1] | x[2] | y |
|---|---|---|
| -1 | -1 | -1 |
| -1 | +1 | +1 |
| +1 | -1 | +1 |
| +1 | +1 | -1 |

| # Levels | Accuracy |
|---|---|
| 0 | 50% |
| 1 | 50% |
| 2 | 100% |



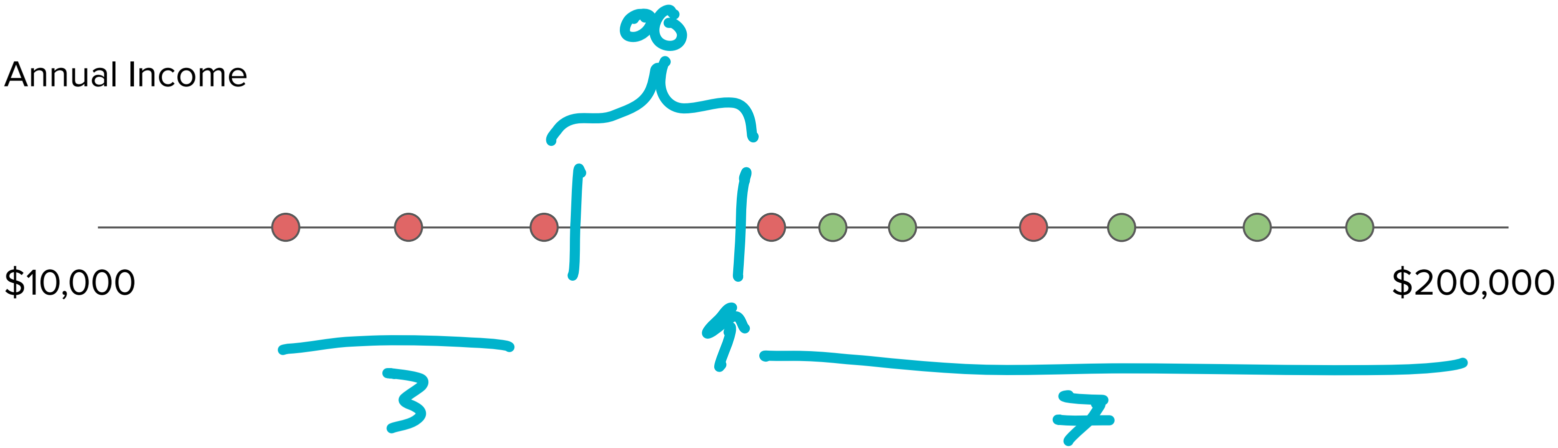
# Real Valued Data



- We've been making an assumption so far that our data takes on discrete values.
- How do we know here to split our data? There are an infinite number of possible splits we can make.

| Credit | Term | Income | Y |
|---|---|---|---|
| excellent | 3 years | $105,000.00 | safe |
| fair | 5 years | $63,000.00 | risky |
| fair | 3 years | $85.000.00 | risky |
| poor | 3 years | $99,000.00 | risky |

# Real Valued Data

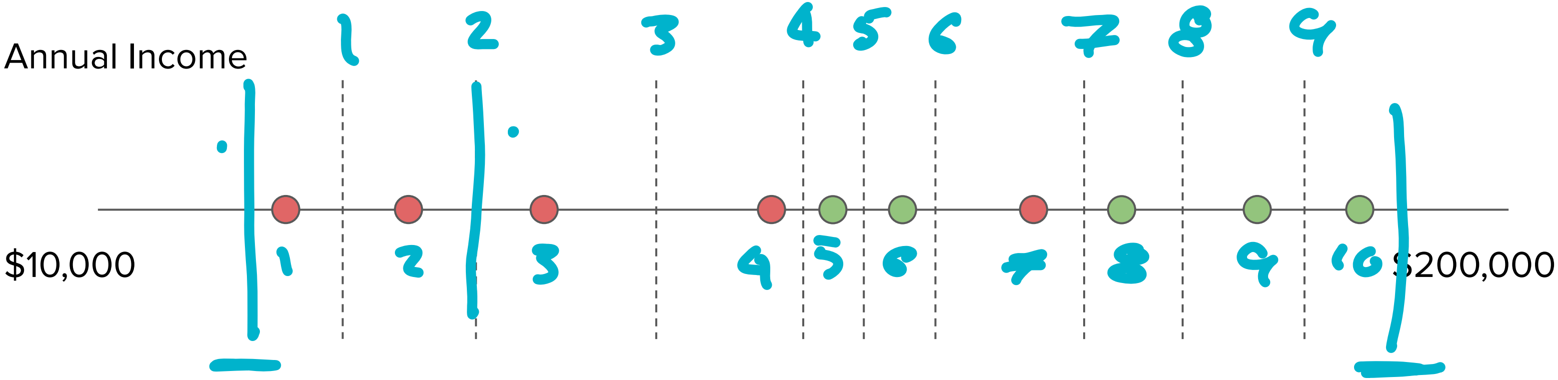



**How do we know where to split our data?**

# Real Valued Data

N data points

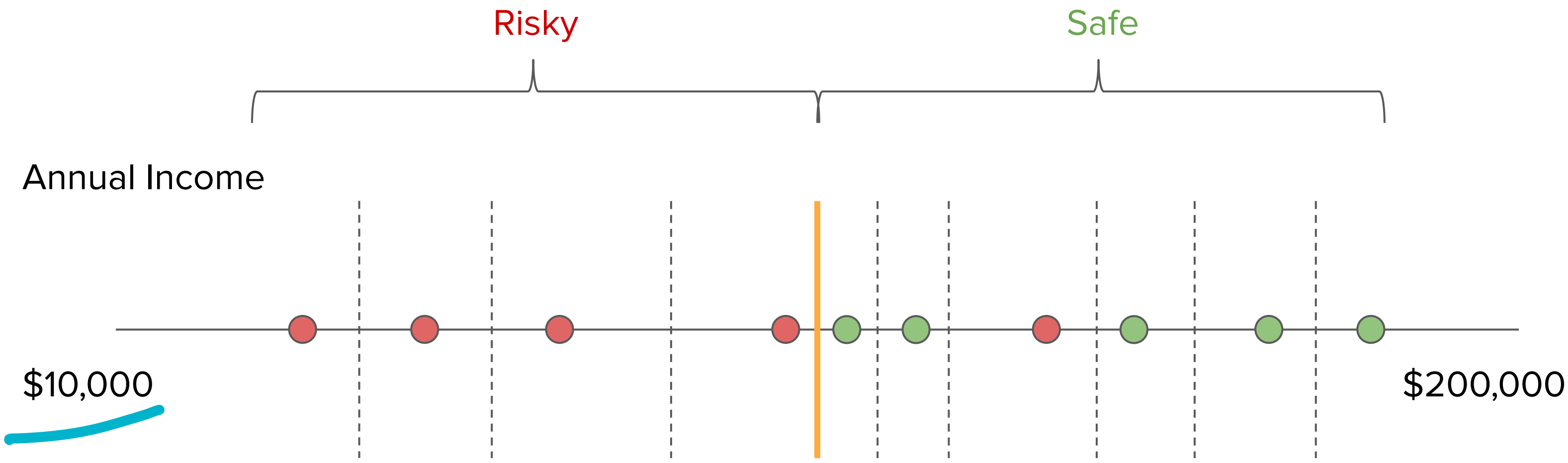
$< N-1$




**Key Insight: Sort the data and split halfway between each pair of adjacent points. There will always be a finite number of splits. How many splits are there?**

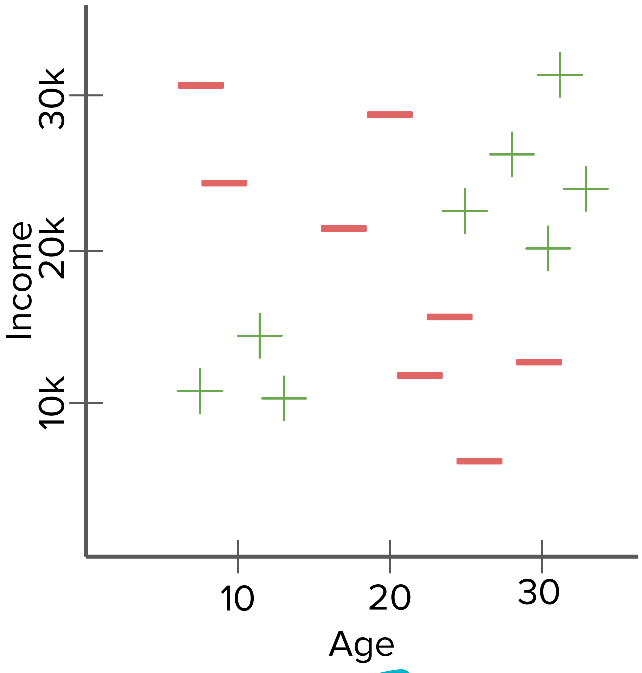
$O(n \log n)$

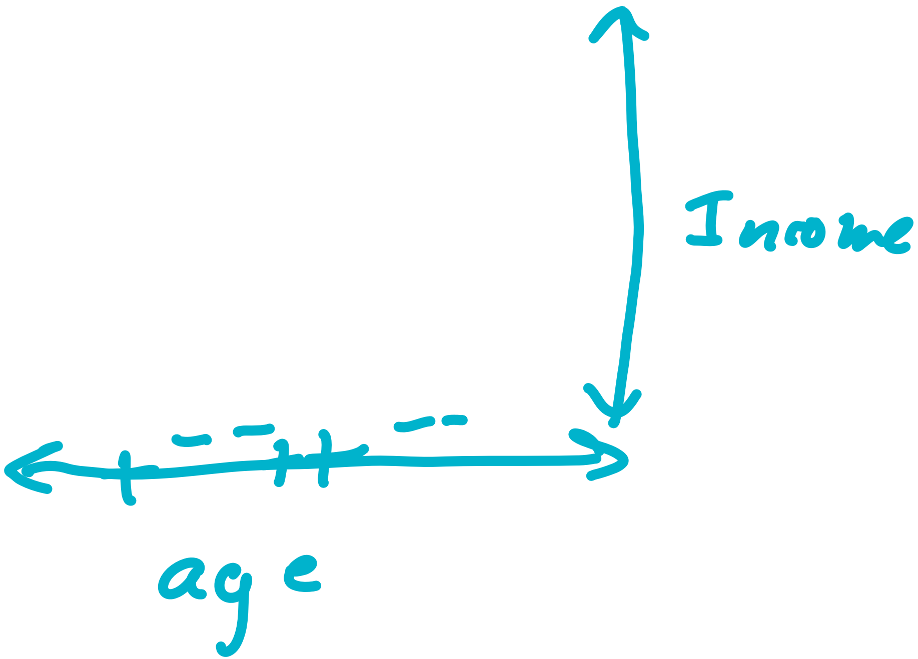
# Real Valued Data



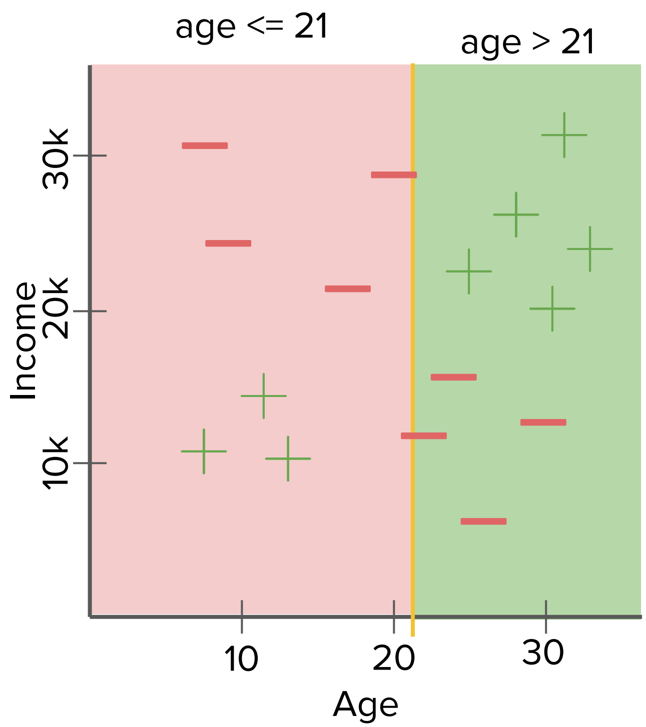

**Which split is best? Pick the one that maximizes accuracy.**

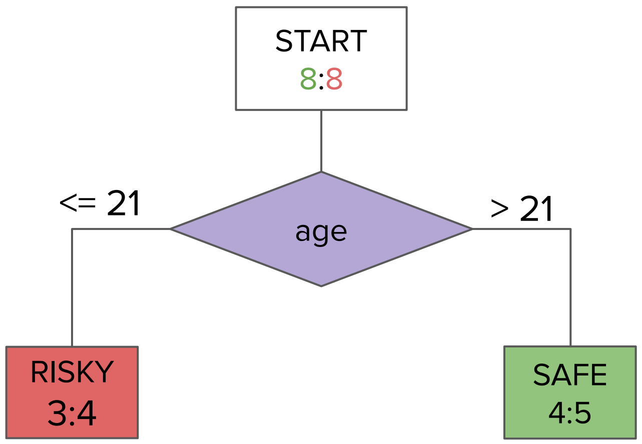
# Real Valued Data



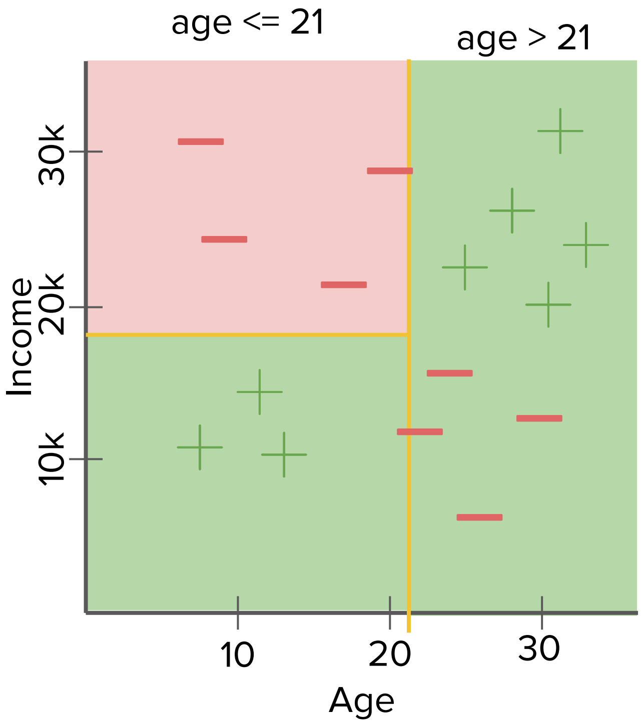



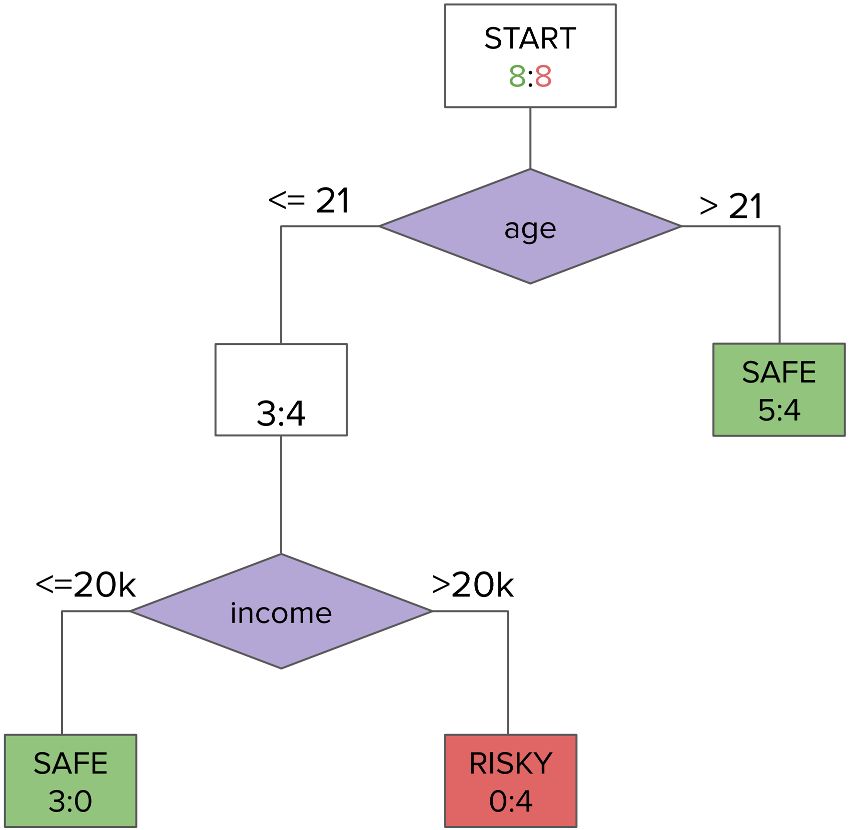
# Real Valued Data



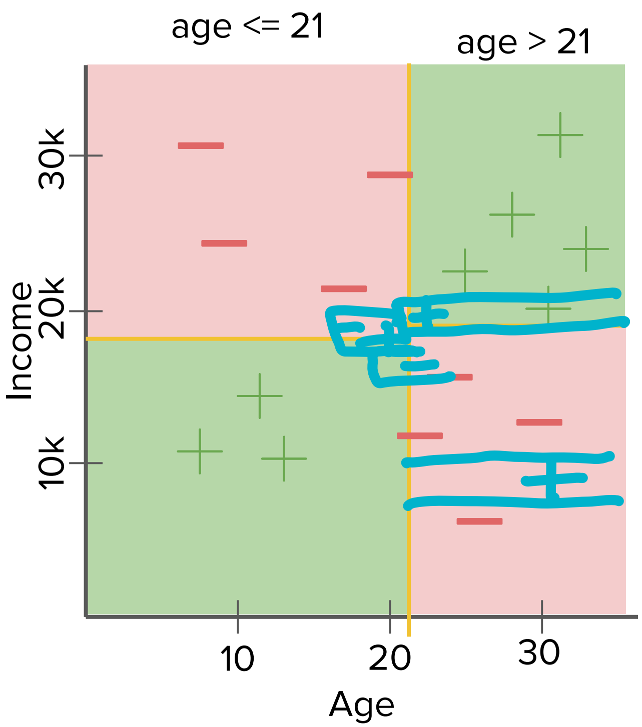



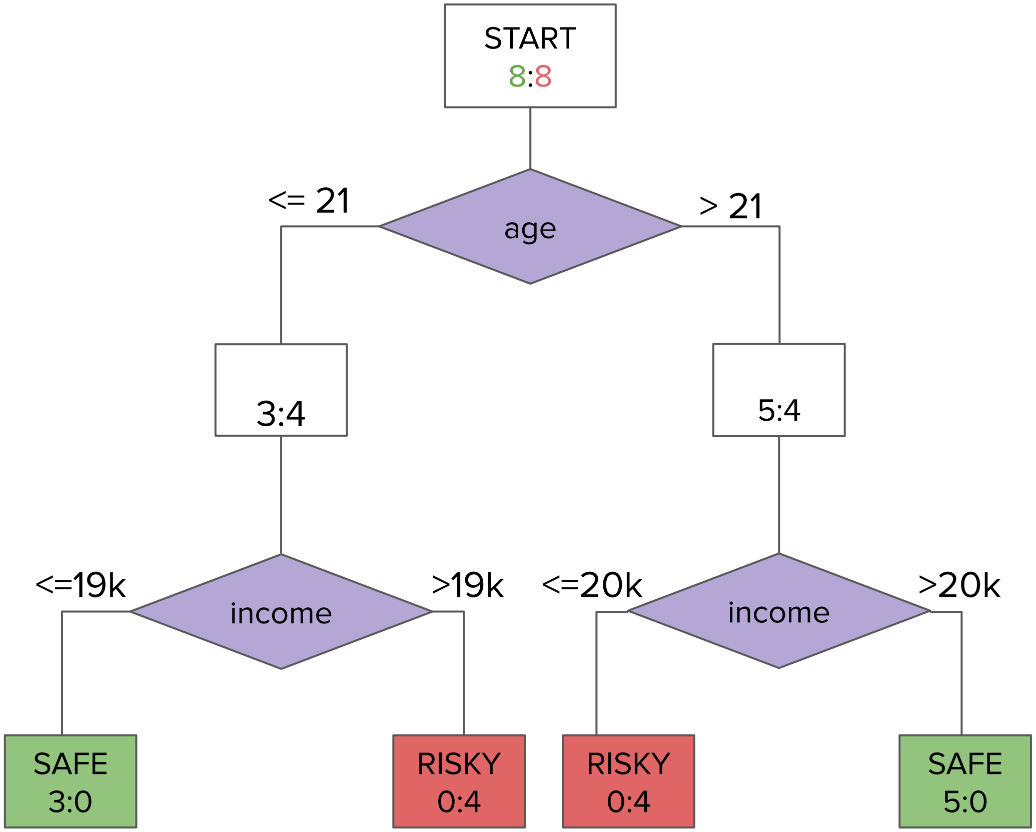
# Real Valued Data



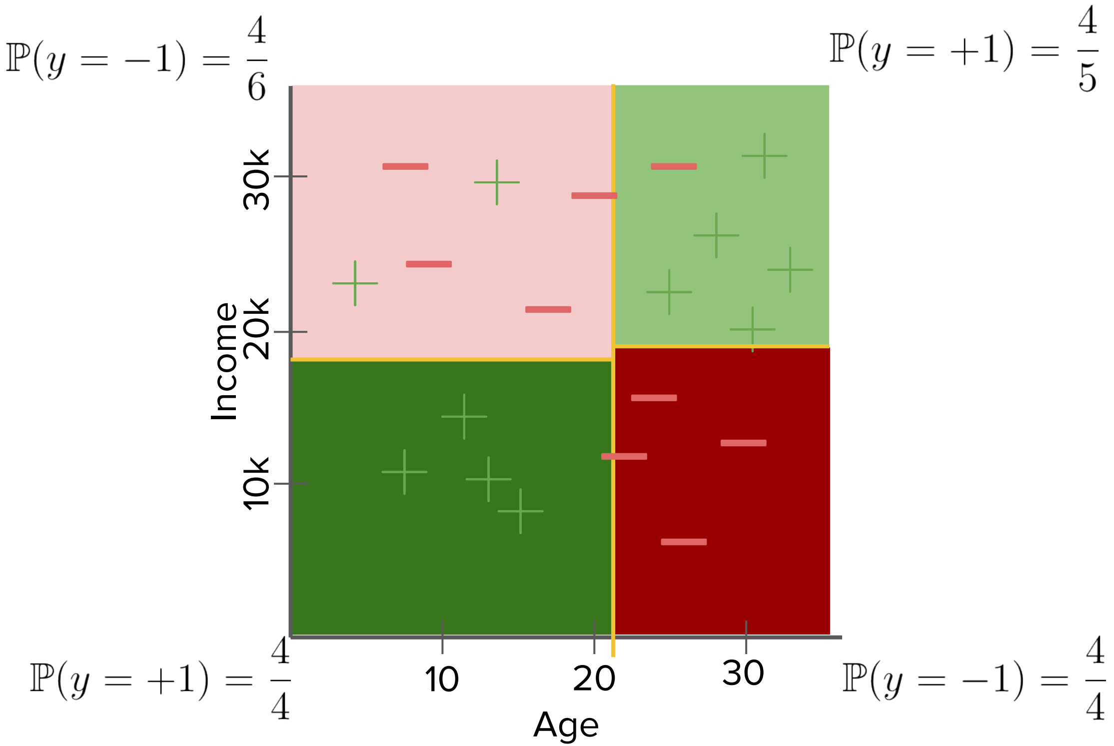



# Real Valued Data



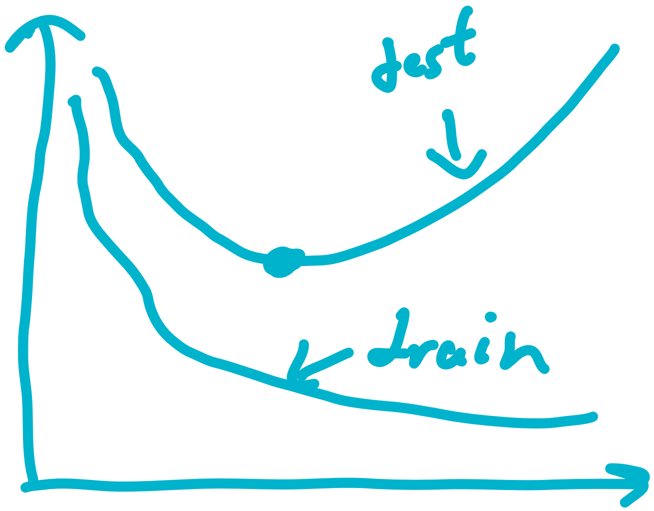



# Probabilistic Prediction

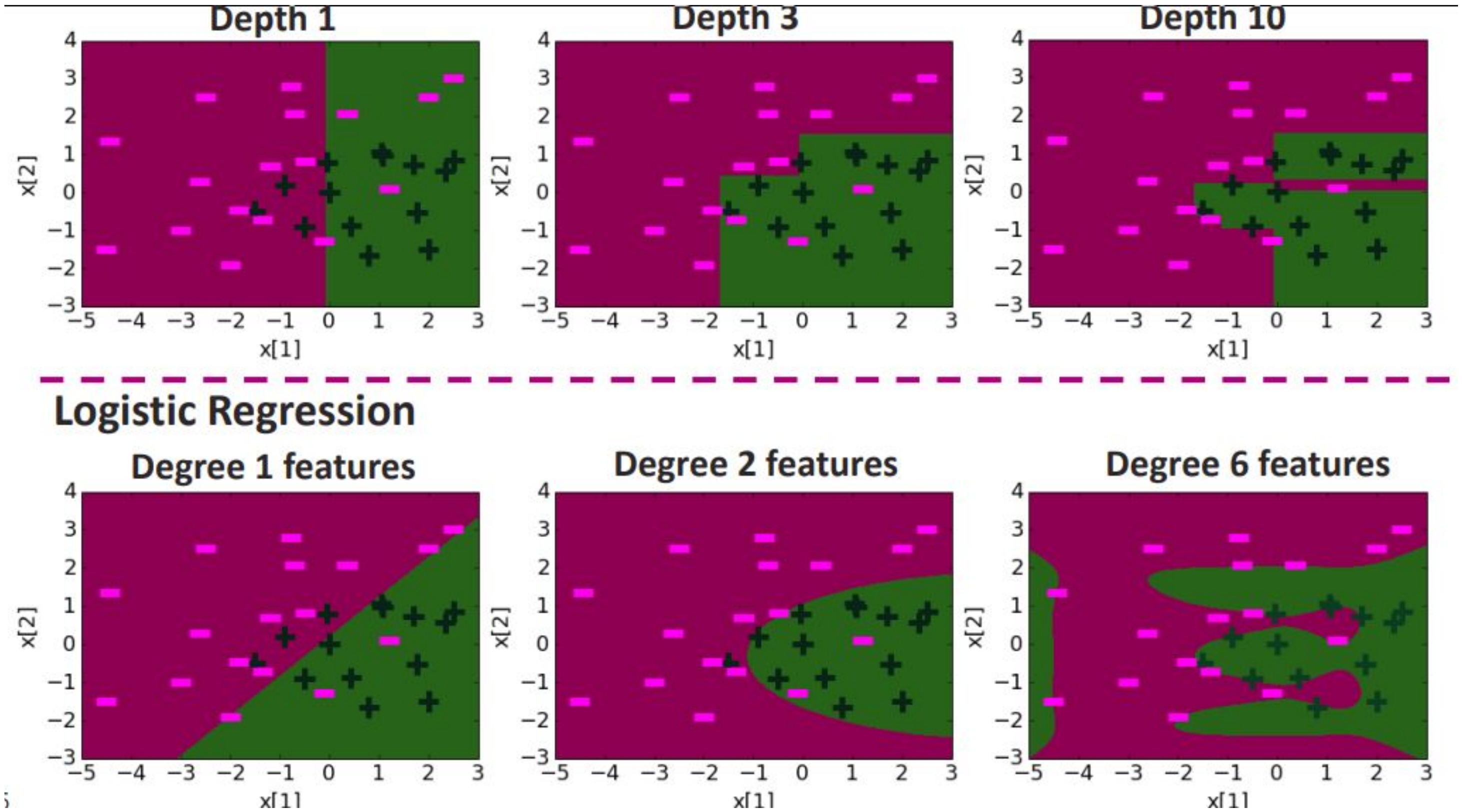


# Overfitting

- Similar to regression, training error monotonically non-increases with model complexity.
- Model complexity with decision trees is commonly measured in the depth of the tree.
- Two methods for preventing overfitting:
    - 1) Early stopping
        - Stop the tree before it can get too complex
    - 2) Pruning
        - Create a complex tre and make it more simple



# Overfitting

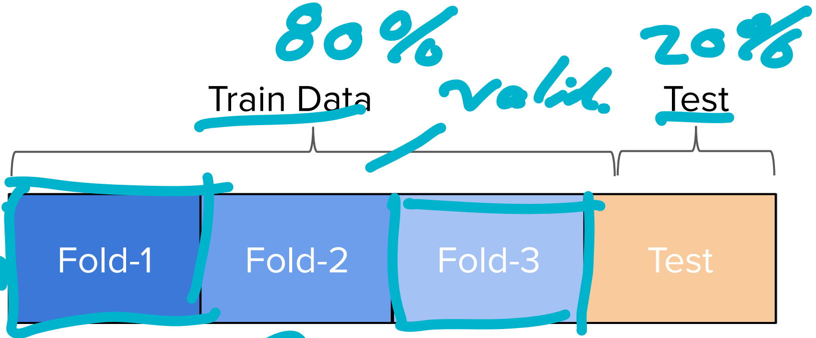


# Overfitting: Early Stopping

- Stopping Rules:
    - 1) All data in the subset have the same label
    - 2) No more features left to split
- Early Stopping Rule
    - Only grow up to a max depth hyperparameter (choose via validation)
        - Can be difficult to know the depth.
        - Oftentimes the correct tree is one that is imbalanced
    - Don’t split if there is not a sufficient decrease in error
        - Problem: difficult to classify XOR problems

/cse416

# Exercise: Overfitting and cross validation



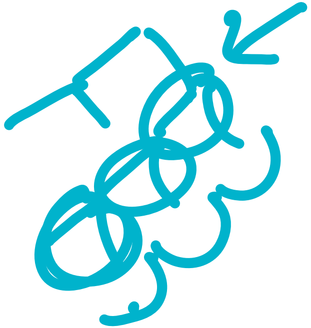

| Max Height | Fold-1 Error | Fold-2 Error | Fold-3 Error | Test Error |
|---|---|---|---|---|
| 5 | 10.3 | 14.2 | 12.5 | 14.5 |
| 10 | 5.6 | 4.3 | 7.3 | 8.7 |
| 15 | 3.1 | 10.4 | 8.8 | 6.9 |

```
cross-validation(data d, folds k):
  fold_1, fold_k = split_data(d, k)

  for each model m:
    for i from 1 to k:
      model = train_model(m, fold -i)
      err = error(model, fold_i)
    avg_err = average err over folds
    keep track of m with smallest avg_err

  return m with smallest avg_err
```

≈12 err validation
≈5.7 validation
≈7.9 validation

# Overfitting: Pruning

- Basic Idea: Train a tall, overfit model and then simplify it.
- Pruning is defined by a quality metric that balances classification error and model complexity.



$$Loss(T) = Error(T) + \underbrace{\lambda r(T)}_{\text{\# leaves in model}}$$

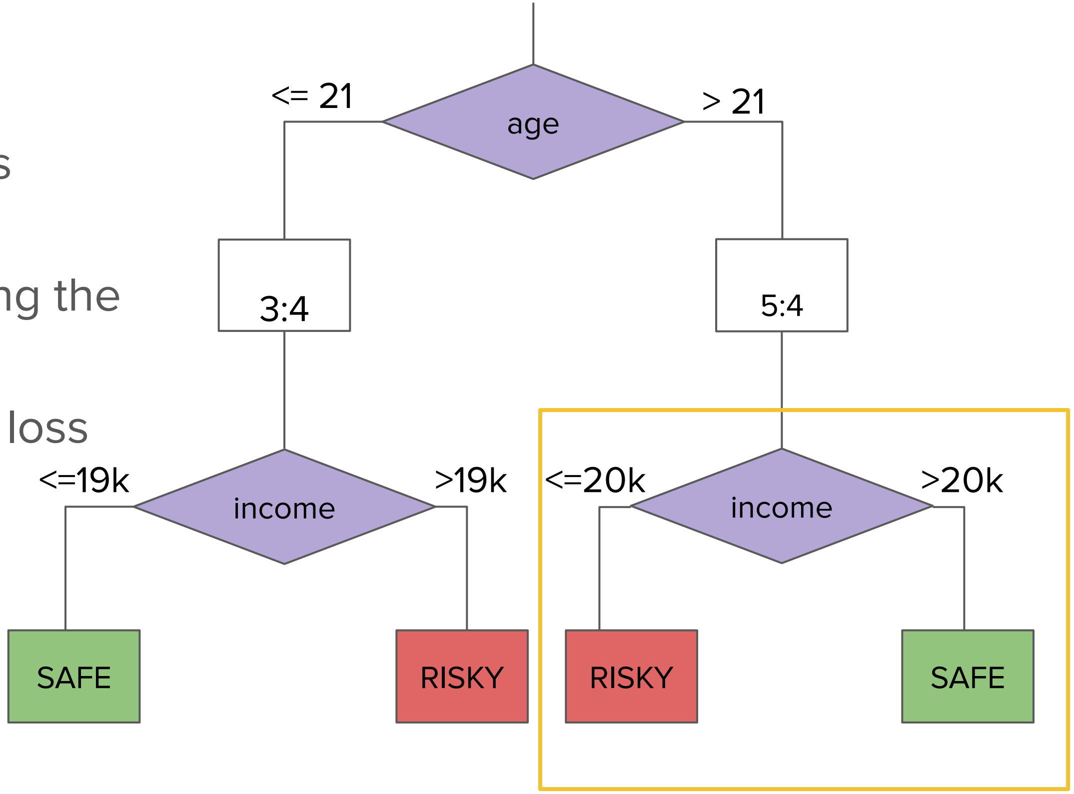
$\text{Total}(T) = \text{Error}(T) + \lambda \,\#\,\text{leaves}(T)$

# Pruning Algorithm

1. Consider some arbitrary split
2. Compute the error if the split is taken away
3. Compute the penalty of keeping the split
4. Pick whichever one minimizes loss
5. Repeat 1-4 for all splits

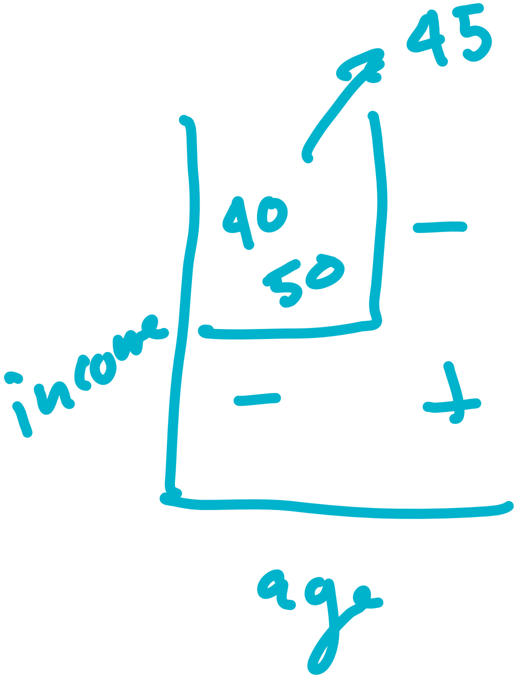
| Tree | Error | # Leaves | Total |
|---:|---:|---:|---:|
| T | 0.25 | 4 | 0.43 |
| T’ | 0.26 | 3 | 0.41 |

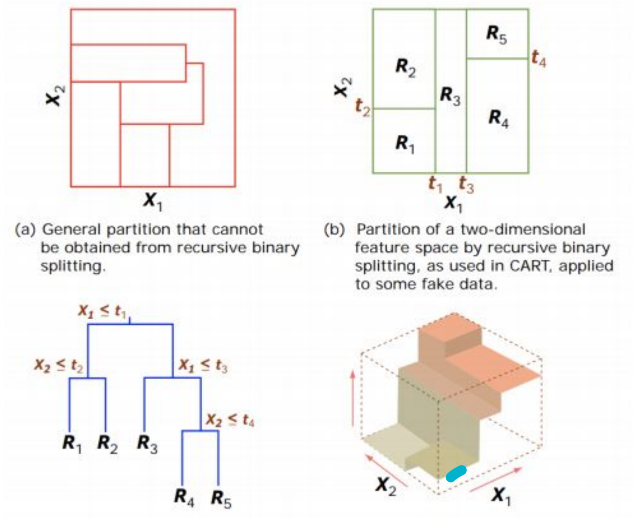
$\lambda = 003$



# Decision Trees for Regression

- Error measured by mean squared error
- Prediction is the mean value of all partitions in the sample




(a) General partition that cannot be obtained from recursive binary splitting.

(b) Partition of a two-dimensional feature space by recursive binary splitting, as used in CART, applied to some fake data.

(c) Tree corresponding to the partition in the top right panel.

(d) A perspective plot of the prediction surface.