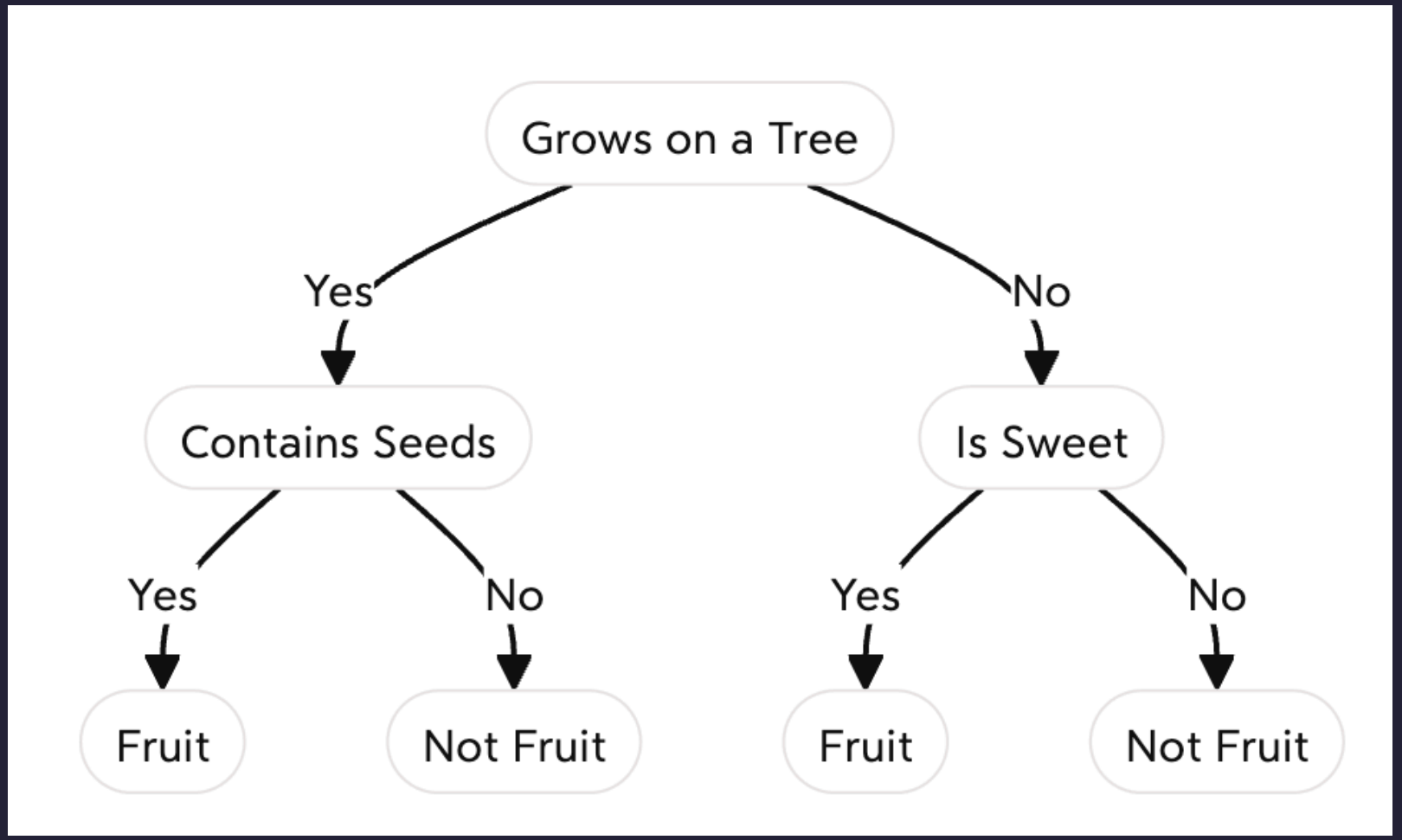
Decision trees

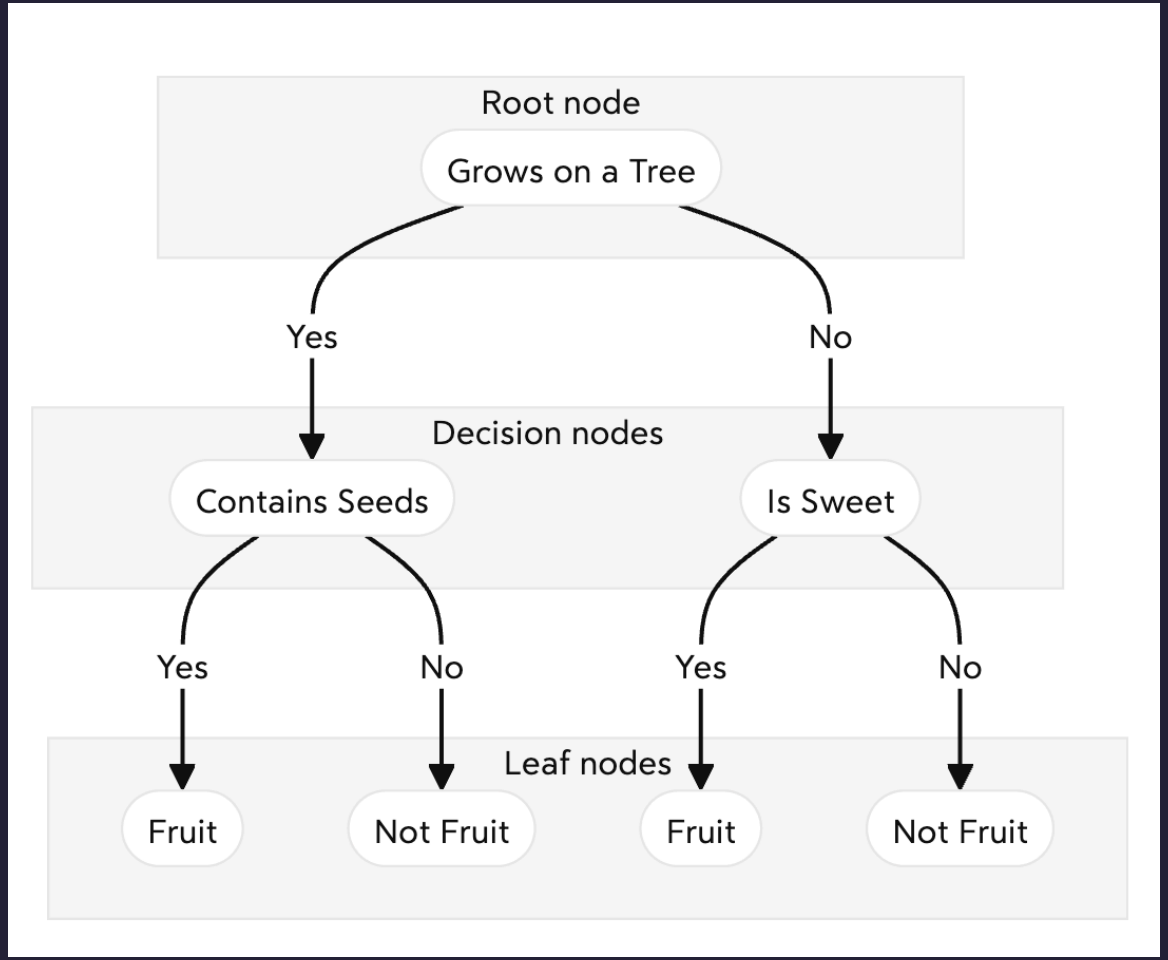
Is it a fruit?

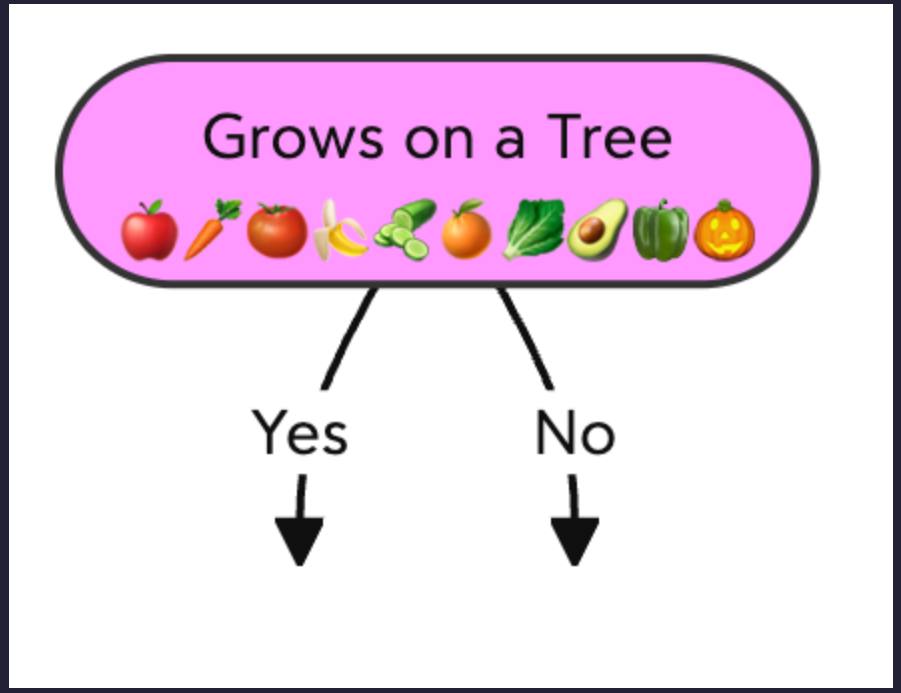
Name	Grows on a Tree	Contains Seeds	Is Sweet	Label
Apple	Yes	Yes	Yes	Fruit
Carrot	No	No	No	Not Fruit
Tomato	No	Yes	No	Fruit
Banana	Yes	Yes	Yes	Fruit
Cucumber	No	Yes	No	Not Fruit
Orange	Yes	Yes	Yes	Fruit
Lettuce	No	No	No	Not Fruit
Avocado	Yes	Yes	No	Fruit
Bell Pepper	No	Yes	No	Not Fruit
Pumpkin	No	Yes	Yes	Not Fruit

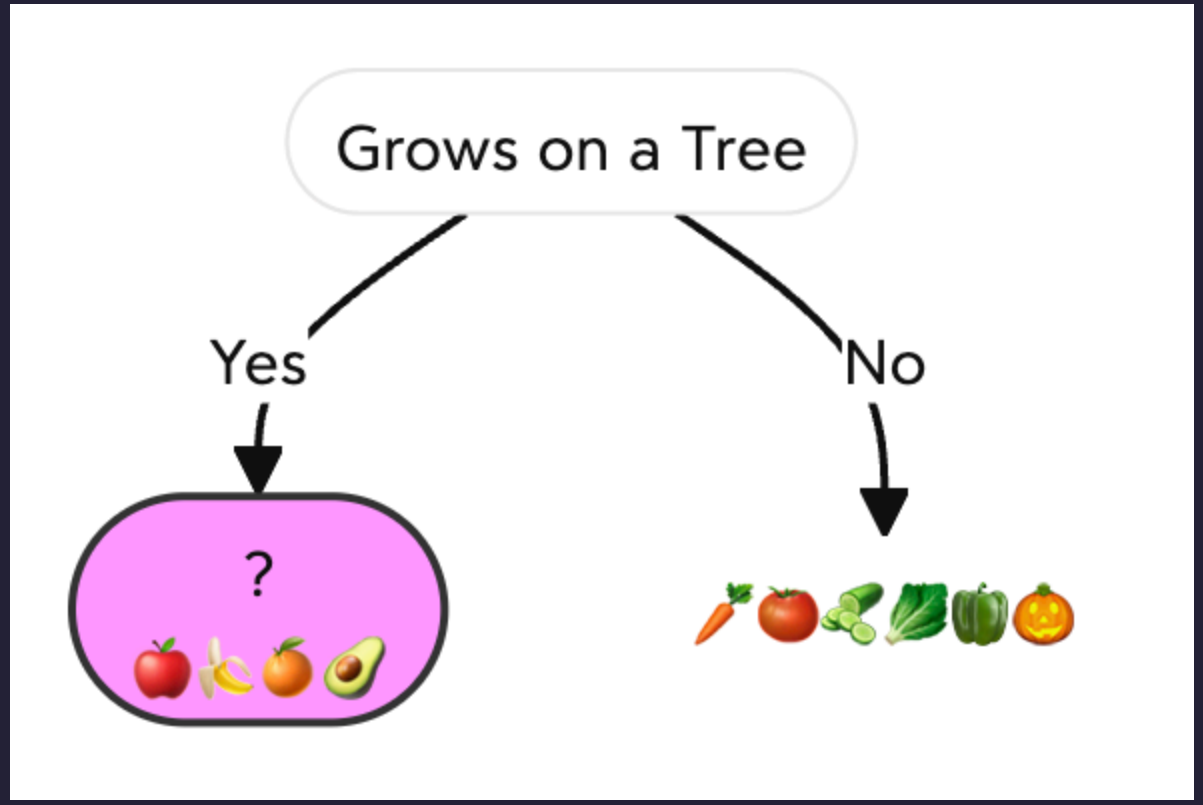

Grows on a Tree
Yes
No
Contains Seeds
Is Sweet
Yes
No
Yes
No
Fruit
Not Fruit
Fruit
Not Fruit

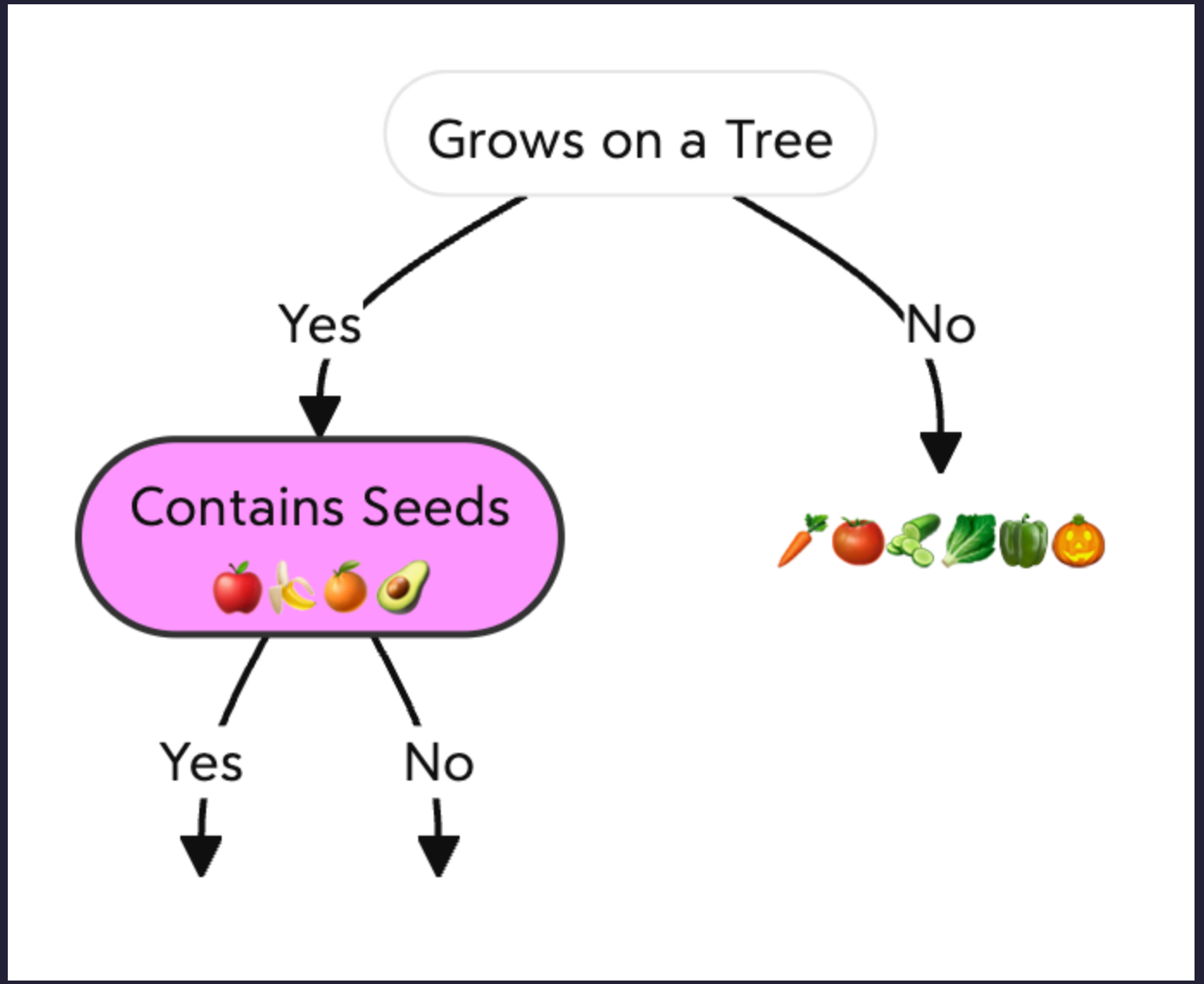
Decision trees

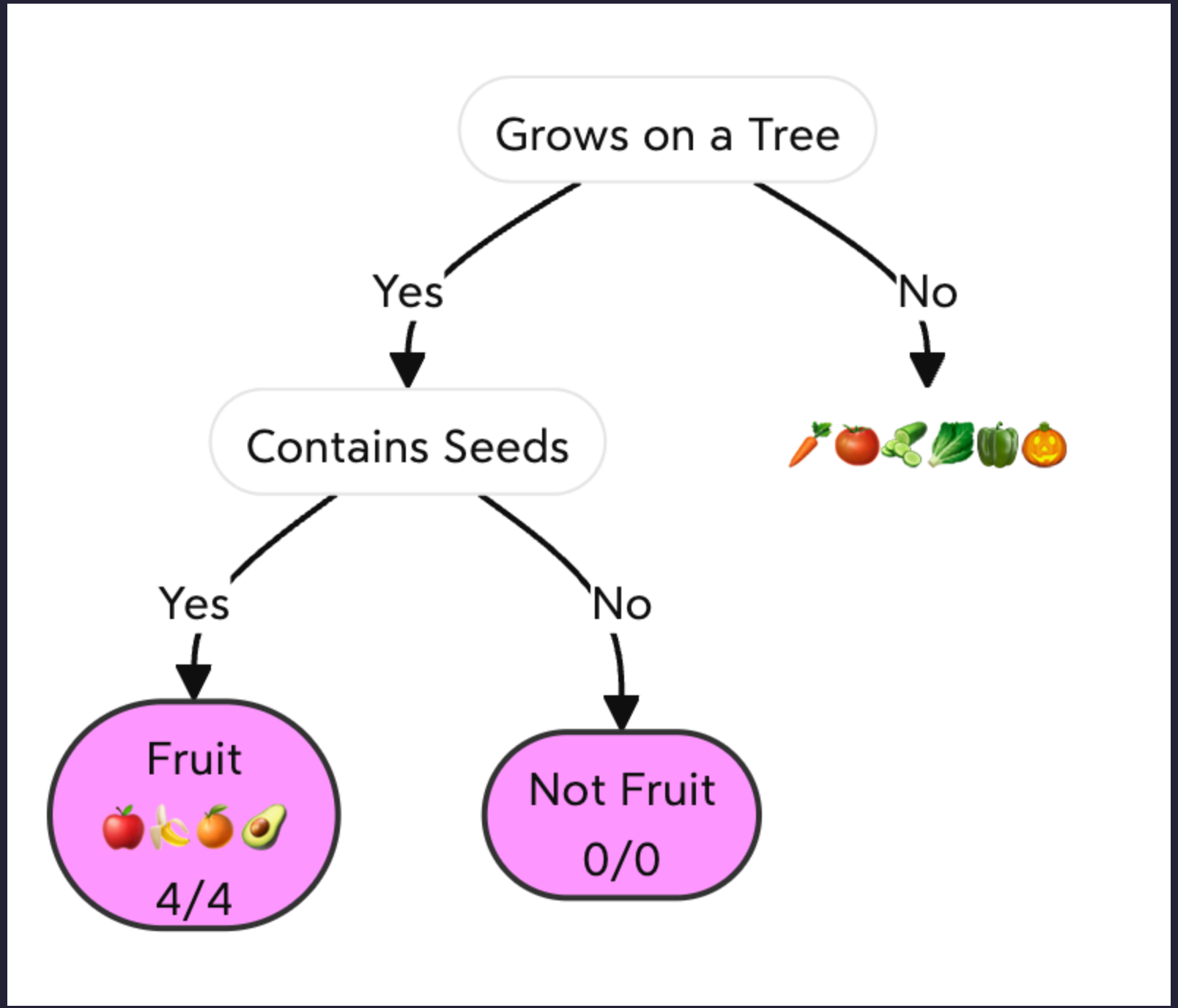
- **Root node**
  Represents the entire dataset
- **Decision nodes**
  Decision points based on feature values
- **Leaf nodes**
  Terminal nodes representing class labels

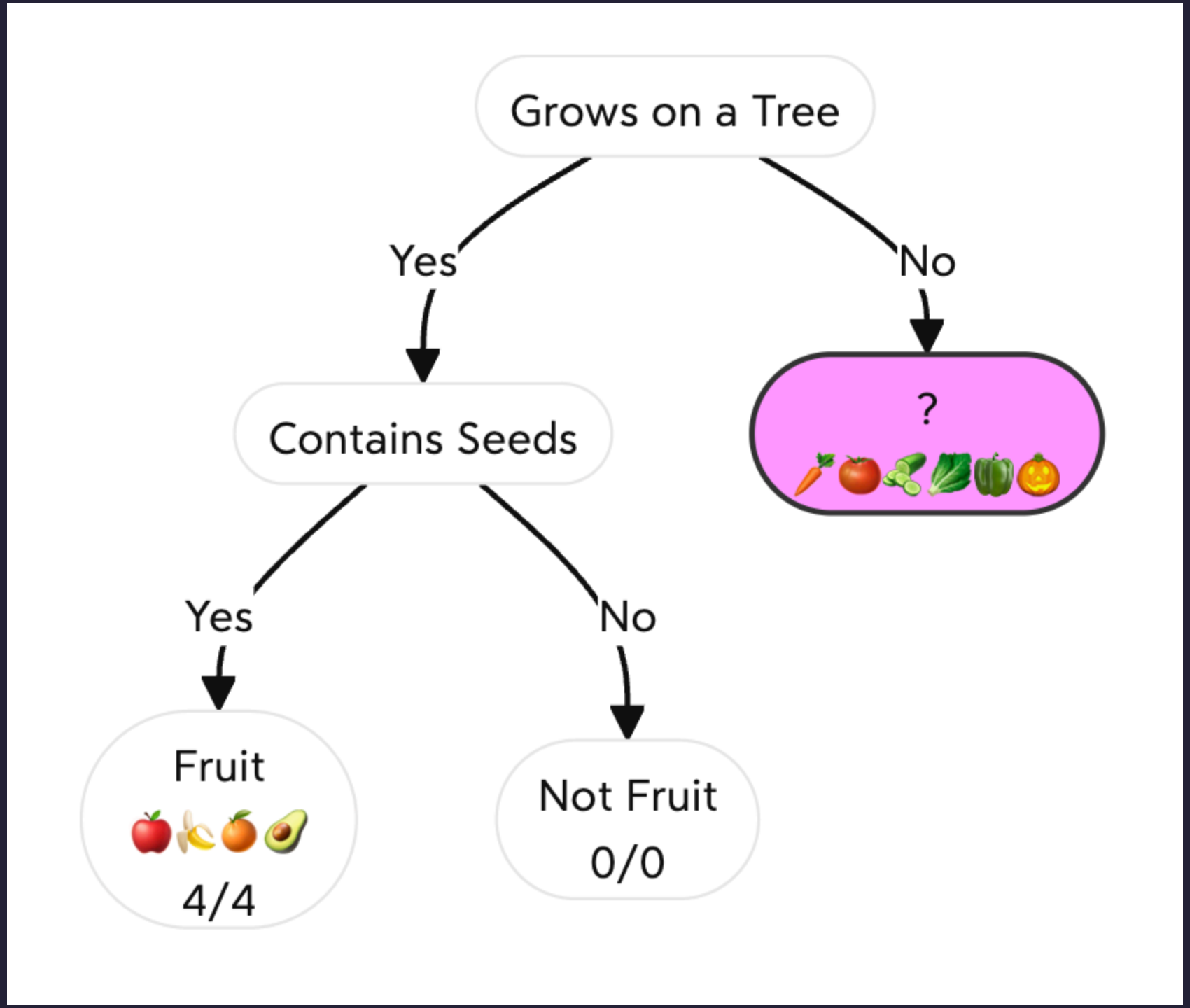


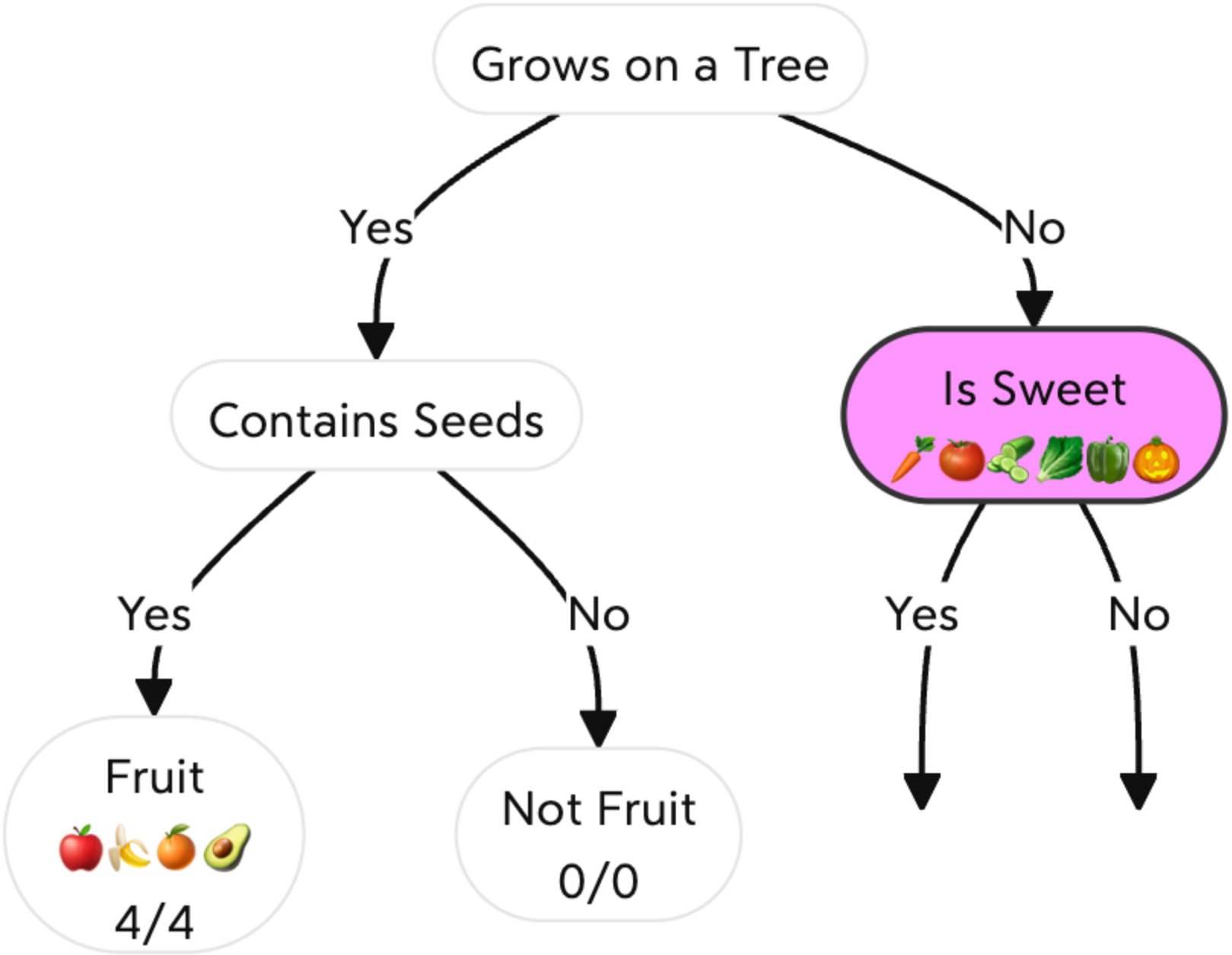

Grows on a Tree
Yes
No

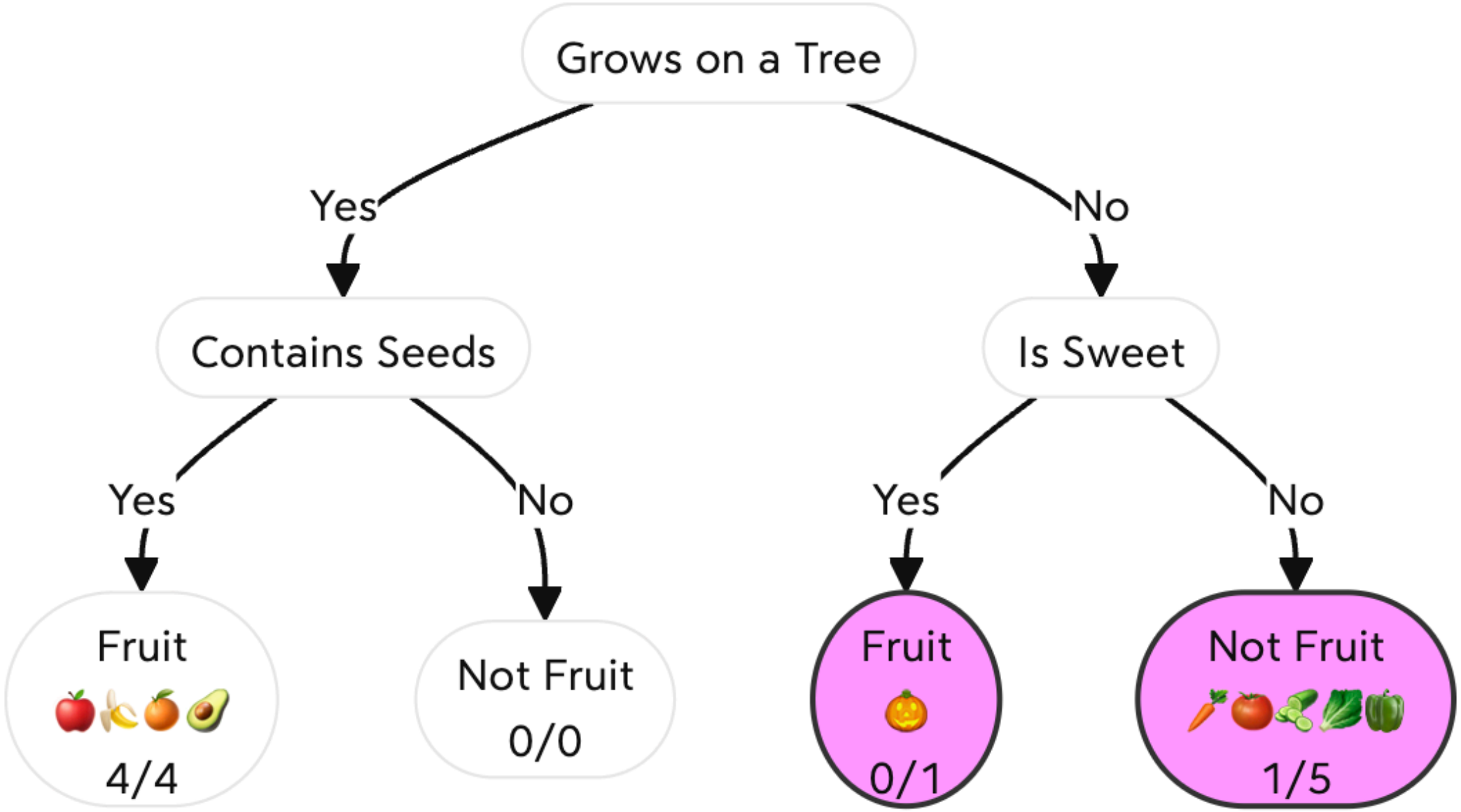

Grows on a Tree
Yes
No
?


Grows on a Tree
Yes
No
Contains Seeds
🍎🍌🍊🥑
Yes
No
🥕🍅🥒🥬🫑🎃

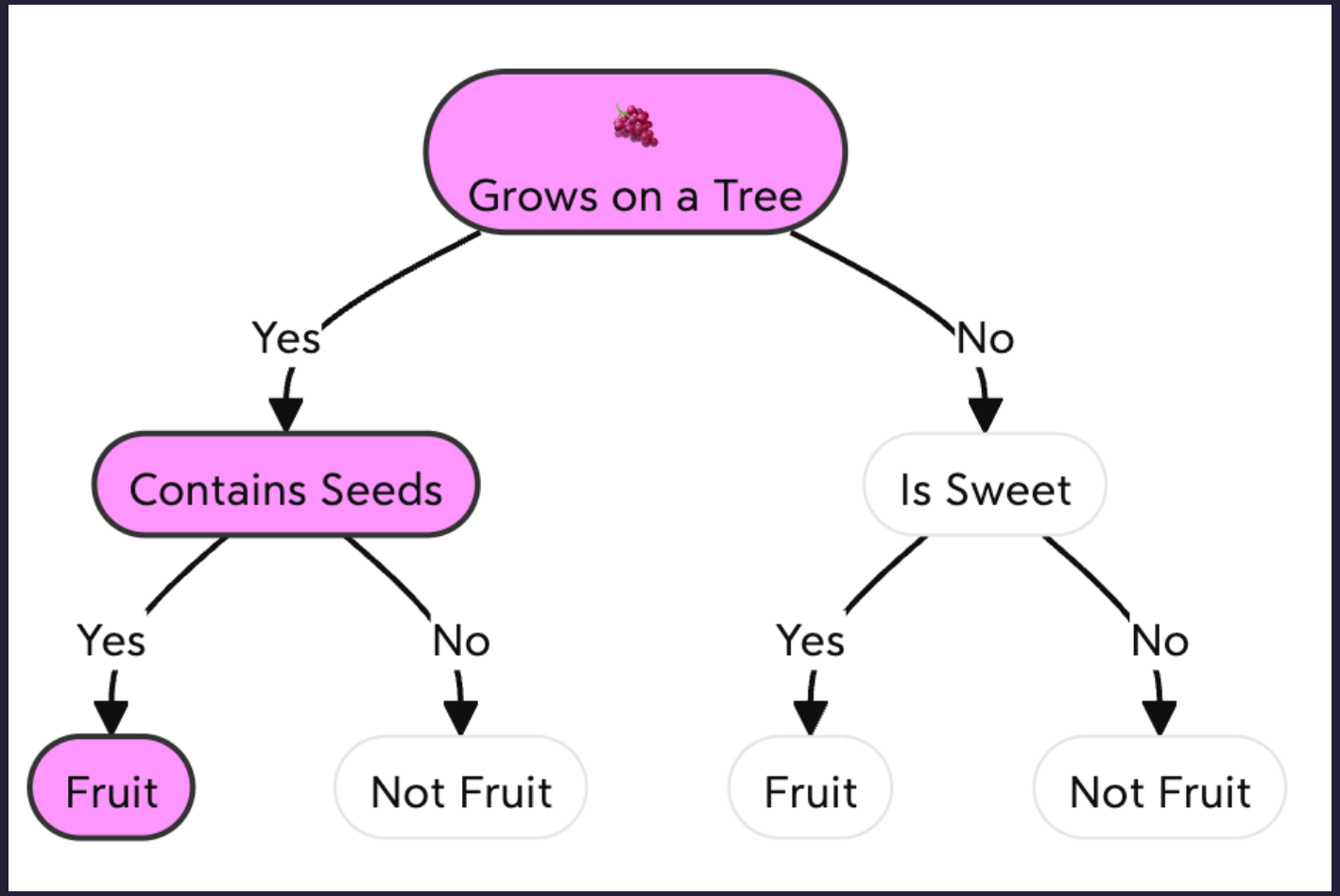

Grows on a Tree
Yes
No
Contains Seeds
🥕🍅🥒🥬🫑🎃
Yes
No
Fruit
🍎🍌🍊🥑
4/4
Not Fruit
0/0


Grows on a Tree
Yes
No
Contains Seeds
?
Yes
No
Fruit
4/4
Not Fruit
0/0


Grows on a Tree
Yes
No
Contains Seeds
Is Sweet
🥕🍅🥒🥬🫑🎃
Yes
No
Yes
No
Fruit
🍎🍌🍊🥑
4/4
Not Fruit
0/0


Grows on a Tree
Yes
No
Contains Seeds
Is Sweet
Yes
No
Yes
No
Fruit
🍎🍌🍊🥑
4/4
Not Fruit
0/0
Fruit
🎃
0/1
Not Fruit
🥕🍅🥦🥬🫑
1/5

Prediction for new example: Grapes 🍇

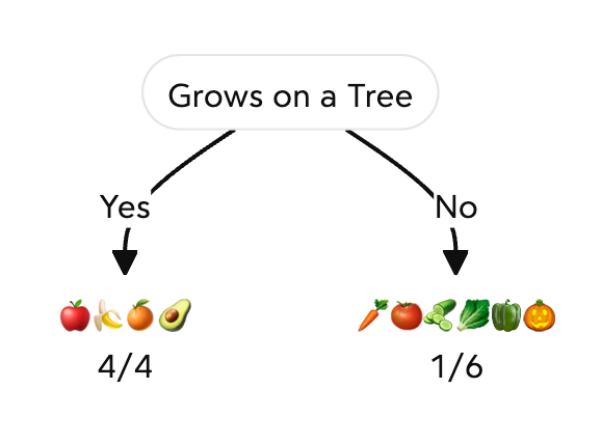


Decision tree learning

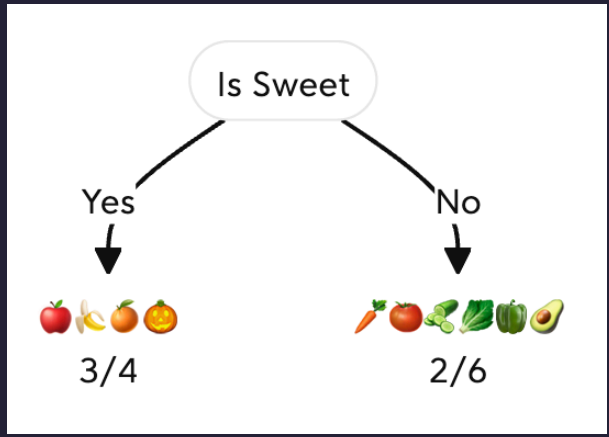
Decision 1: Which feature to split on?

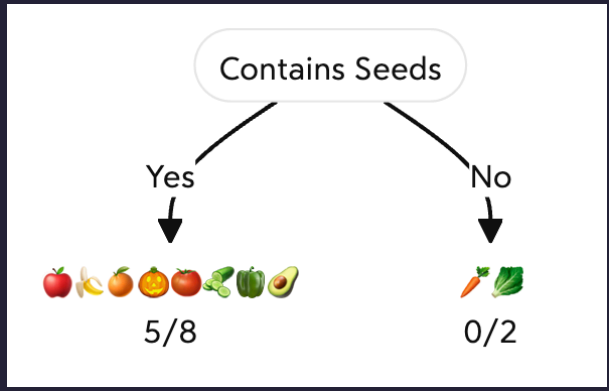
Decision 2: When to stop splitting?

Decision 1: Which feature to split on?



👍





👎

- **Maximize purity (or minimize impurity)**

Measuring impurity - Entropy

$$H(p_1) = -p_1 \log_2(p_1) - (1 - p_1) \log_2(1 - p_1)$$

$p_1 = 1,\ p_1 = 0 \Rightarrow H(1) = 0$

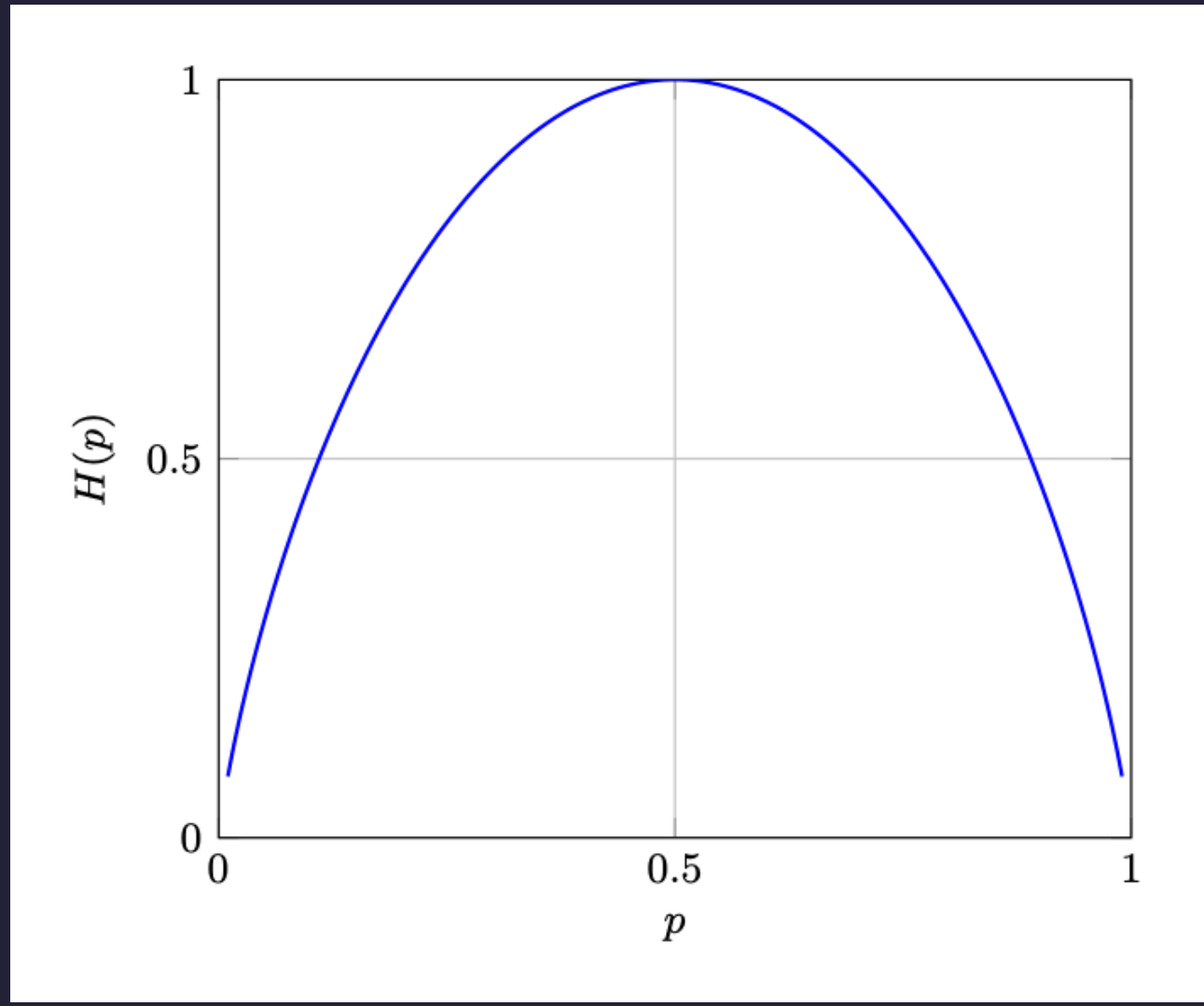
no uncertainty, perfect information (pure)

$p_1 = 1/2 \Rightarrow H(0.5) = 1$

most uncertain, no information

Note:

$0 * log_2(0) = 0$



Measuring impurity - Entropy

🍎🍌🍊🥑 4/4:

$p_1 = 1,\ h(p_1) = 0$

🥕🍅🥒🥬🫑🎃 1/6:

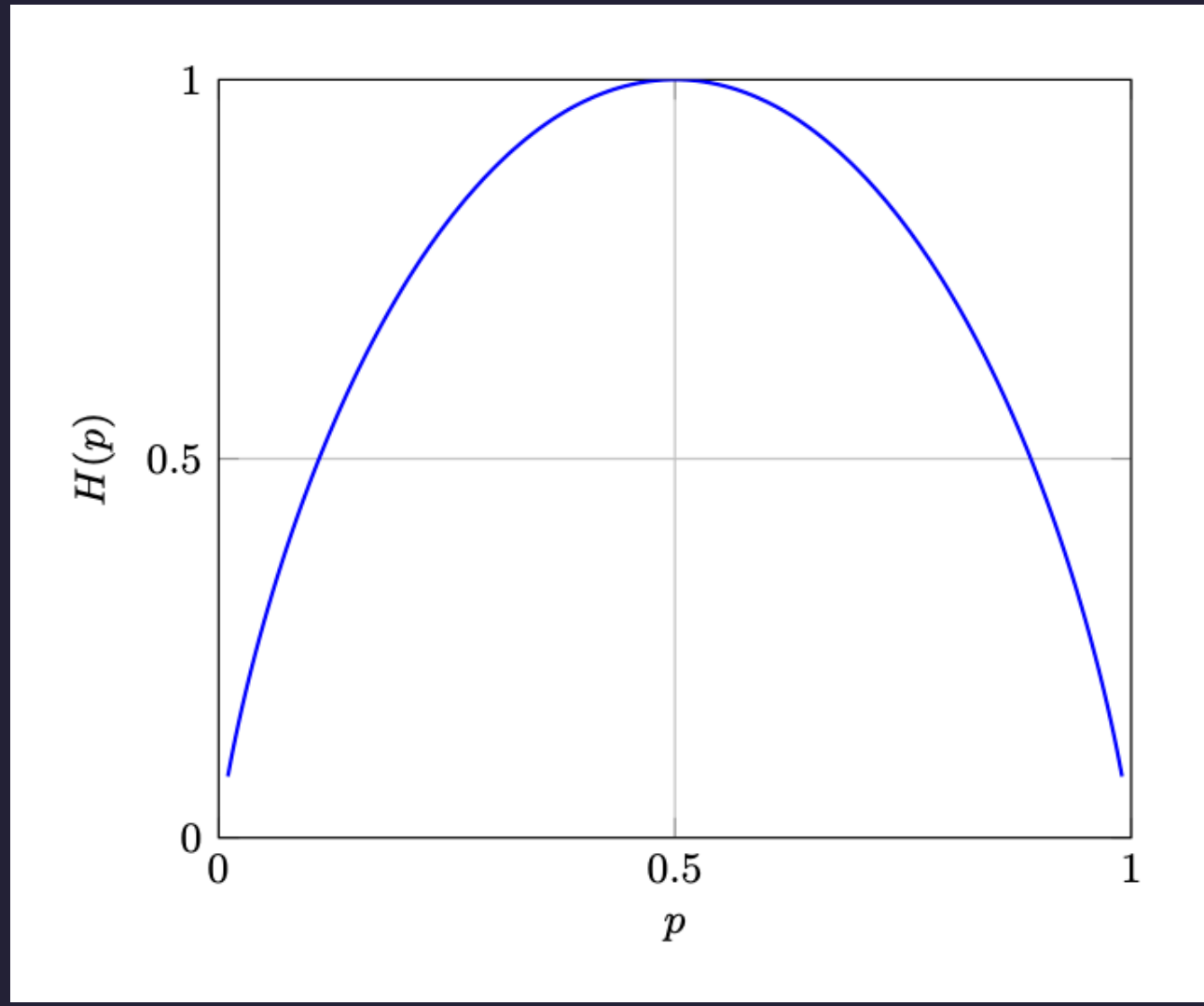
$p_1 = 1/6,\ h(p_1) = 0.65$

🍎🍌🍊🎃 3/4:

$p_1 = 3/4,\ h(p_1) = 0.81$

🥕🍅🥒🥬🫑🥑 2/6:

$p_1 = 2/6,\ h(p_1) = 0.92$



Information gain

Measure of the reduction in entropy (uncertainty) after splitting the dataset on an attribute

$$IG(D, A) = H(D) - H(D|A) = H(D) - \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i)$$

- D is the dataset
  - $|D|$ is the number of examples in D
- $H(D)$ is the entropy of the dataset
  - $H(D) = -p_1 \log_2(p_1) - (1 - p_1) \log_2(1 - p_1)$
  - p_1 is the proportion of positive examples in D
- D_i is the subset of D after splitting on attribute A into n branches
- $H(D|A)$ is the entropy of the dataset after splitting

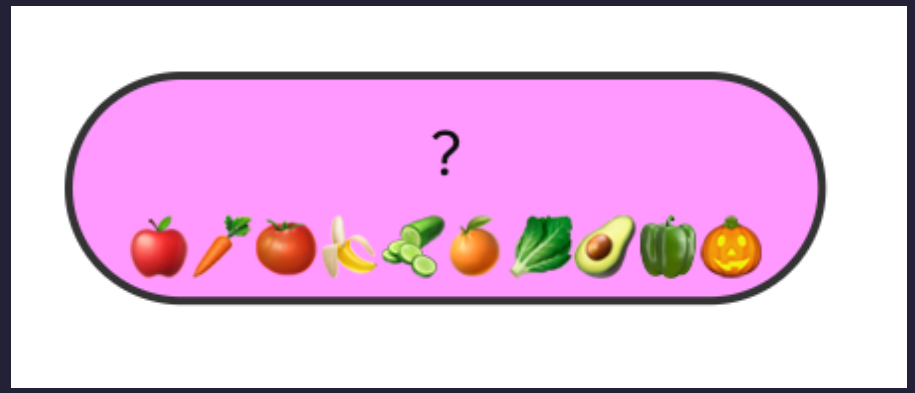
Decision 1: Which feature to split on?

Dataset D:

🍎🍌🍊🥑🥕🍅🥒🥬🫑🎃 **5/10**

Attributes:

- Grows on a Tree
- Is Sweet
- Contains Seeds



Information gain: "Grows on a Tree"

Dataset D:

🍎🍌🍊🥑🥕🍅🥒🥬🫑🎃 5/10

Attribute A: "Grows on a Tree"

Dataset after splitting on attribute A

- Left $D1$: 🍎🍌🍊🥑 4/4
- Right $D2$: 🥕🍅🥒🥬🫑🎃 1/6

Information Gain:

$$IG(D, A) = H(D) - H(D|A)$$

$$H(D) = -\frac{1}{2}\log_2(\frac{1}{2}) - (1 - \frac{1}{2})\log_2(1 - \frac{1}{2}) = 1$$

$$H(D_1) = -\log_2(1) - (1 - 1)\log_2(1 - 1) = 0$$

$$H(D_2) = -\frac{1}{6}\log_2(\frac{1}{6}) - (1 - \frac{1}{6})\log_2(1 - \frac{1}{6}) = 0.65$$

$$H(D|A) = \frac{|D_1|}{|D|}H(D_1) + \frac{|D_2|}{|D|}H(D_2)$$

$$= \frac{4}{10} * 0 + \frac{6}{10} * 0.65 = 0.39$$

$$IG(D, A) = 1 - 0.39 = 0.61$$

Information gain: "Is Sweet"

Dataset D:

🍎🍌🍊🥑🥕🍅🥒🥬🫑🎃 **5/10**

Attribute A: "Is Sweet"

Dataset after splitting on attribute A

- Left $D1$: 🍎🍌🍊🎃 **3/4**
- Right $D2$: 🥑🥕🍅🥒🥬🫑 **2/6**

Information Gain:

$$IG(D, A) = H(D) - H(D|A)$$

$$H(D) = 1$$
$$H(D_1) = 0.81$$
$$H(D_2) = 0.92$$

$$H(D|A) = \frac{4}{10} * 0.81 + \frac{6}{10} * 0.92 = 0.87$$

$$IG(D, A) = 1 - 0.87 = 0.13$$

Information gain: "Contains Seeds"

Dataset D:

🍎🍌🍊🥑🥕🍅🥒🥬🫑🎃 5/10

Attribute A: "Contains Seeds"

Dataset after splitting on attribute A

- Left $D1$: 🍎🍌🍊🥑🍅🥒🫑🎃 5/8
- Right $D2$: 🥕🥬 0/2

Information Gain:

$$IG(D, A) = H(D) - H(D|A)$$

$$H(D) = 1$$

$$H(D_1) = 0.95$$

$$H(D_2) = 0$$

$$H(D|A) = \frac{8}{10} * 0.95 + \frac{2}{10} * 0 = 0.76$$

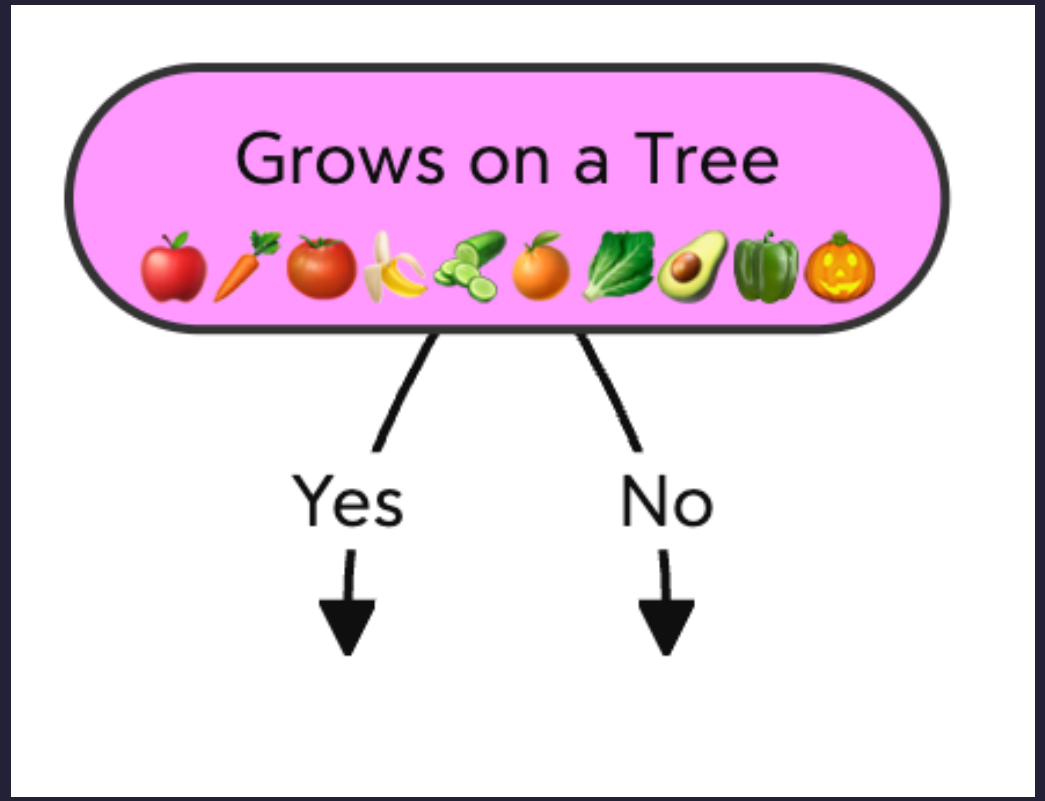
$$IG(D, A) = 1 - 0.76 = 0.24$$

Decision 1: Which feature to split on?

Pick the feature with the highest information gain

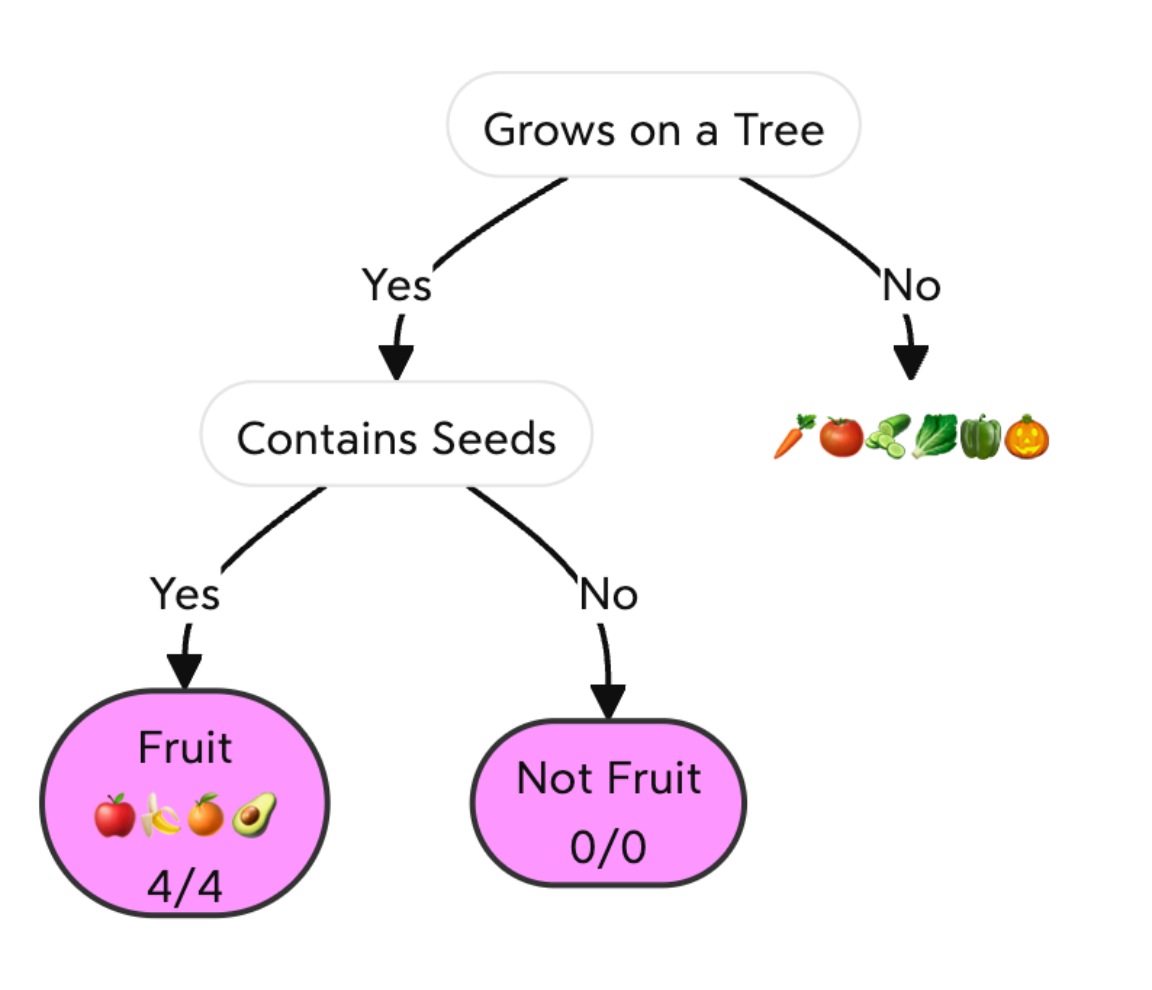
- **Grows on a Tree: 0.61**
- Is Sweet: 0.13
- Contains Seeds: 0.24

Split on that feature and repeat the process for each subbranch



Decision 2: When to stop splitting?

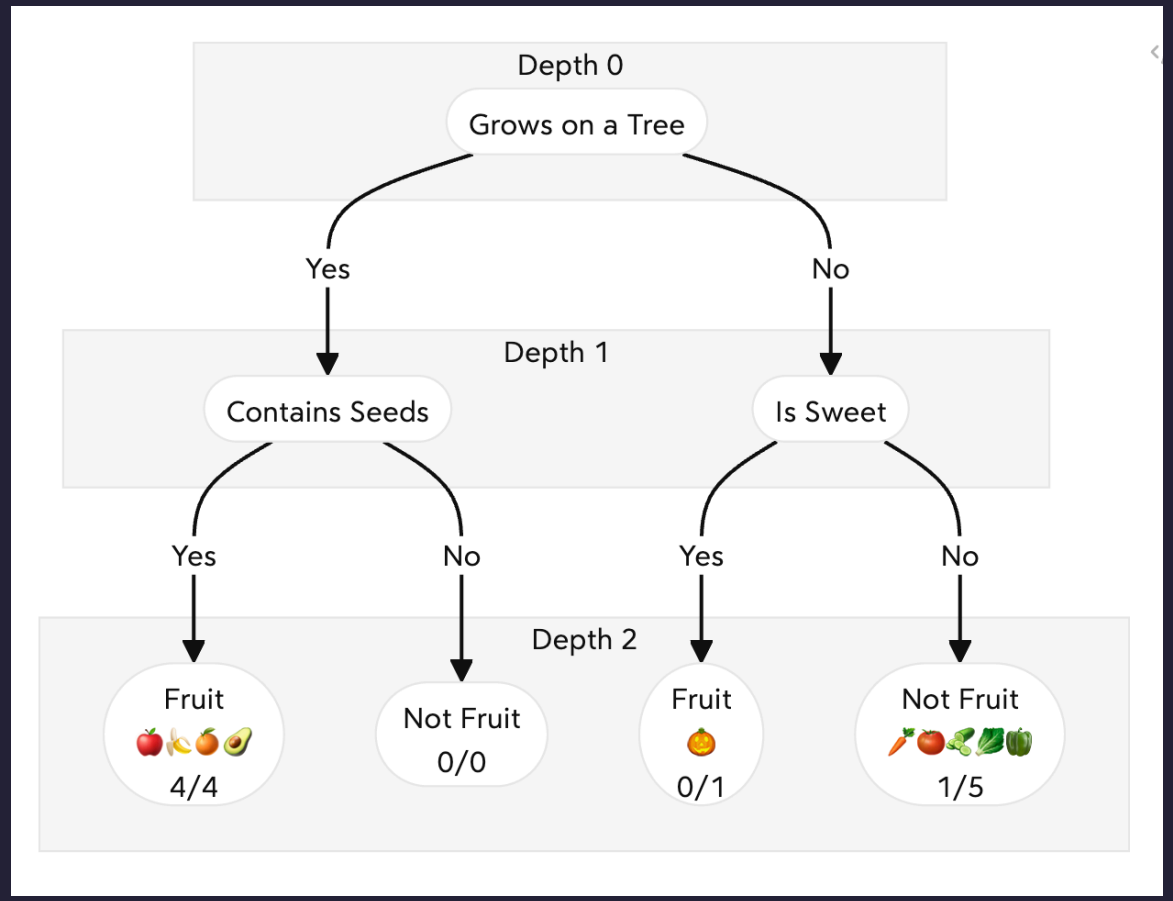
- **Perfect classification**: When a node is 100% one class



Stopping criteria

To prevent overfitting, we can stop splitting when:

- **Max depth**: splitting a node will result in the tree exceeding a maximum depth
- **Information gain**: information gain from additional splits is less than threshold
- **Minimum samples**: number of examples in a node is below a threshold



Decision tree learning

- Start with all examples at the root node
- Calculate information gain for all possible features, and pick the one with the highest information gain
- Split dataset according to selected feature, and create left and right branches of the tree
- Keep repeating splitting process until stopping criteria is met

Extensions to non-binary inputs and outputs

Categorical features

Sweetness: Low, Medium, High

Name	Sweetness
Apple	Medium
Carrot	Low
Tomato	Low
Banana	High

One-hot encoding for each category

Name	sweet_L	sweet_M	sweet_H
Apple	0	1	0
Carrot	1	0	0
Tomato	1	0	0
Banana	0	0	1

Continuous features

Sweetness: 0-100

Name	Sweetness
Apple	90
Carrot	10
Tomato	40
Banana	85

One-hot encoding for each midpoint

Name	sweet_87.5	sweet_62.5	sweet_25
Apple	1	1	1
Carrot	0	0	0
Tomato	0	0	1
Banana	0	1	1

Sort the values and find all the midpoints

- 87.5 (Apple-Banana), 62.5 (Banana-Tomato), 25 (Carrot-Tomato), ...

Calculate the information gain for each midpoint

Choose the midpoint with the highest information gain

Regression trees

Decision trees used for predicting continuous outcomes

Differences from Classification Trees:

- Leaf nodes contain continuous values (instead of class labels)
- Splitting criteria based on variance reduction (instead of entropy)

How sweet is it?

Name	Grows on a Tree	Contains Seeds	*Sweetness*
Apple	Yes	Yes	90
Carrot	No	No	10
Tomato	No	Yes	40
Banana	Yes	Yes	85
Cucumber	No	Yes	30
Orange	Yes	Yes	95
Lettuce	No	No	5
Avocado	Yes	Yes	50
Bell Pepper	No	Yes	25
Pumpkin	No	Yes	60

Prediction

Mean of the values in the leaf node

Prediction at the leaf node (left):

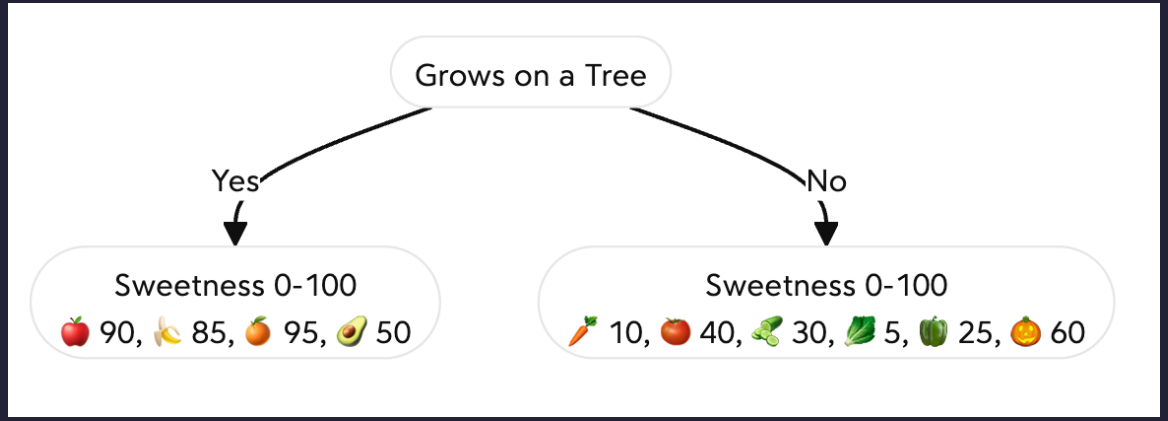
`np.mean([90,85,95,50]) = 80`

Prediction at the leaf node (right):

`np.mean([10,40,30,5,25,60]) = 27.5`

Prediction for new example 🍇

- Grows on a Tree: Yes
- 80



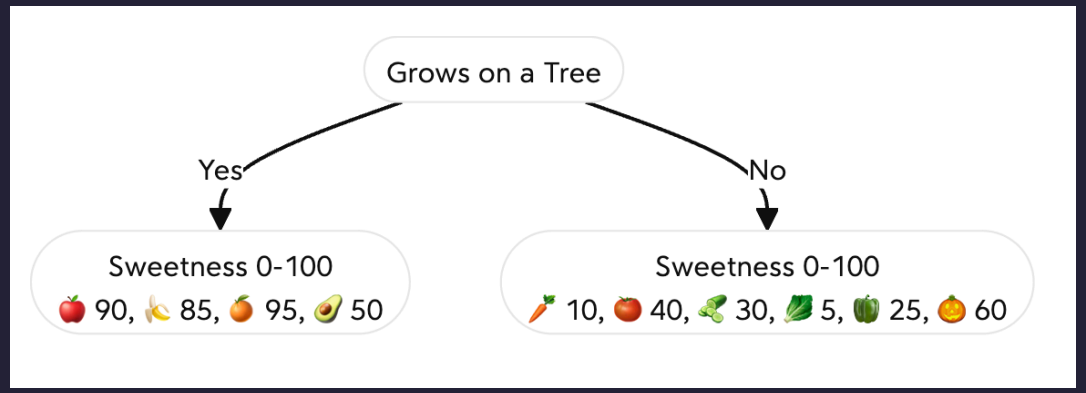
Variance as a measure of impurity

Variance:

$$\text{Var} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

Information Gain: Reduction in variance

$$IG(D, A) = \text{Var}(D) - \text{Var}(D|A)$$



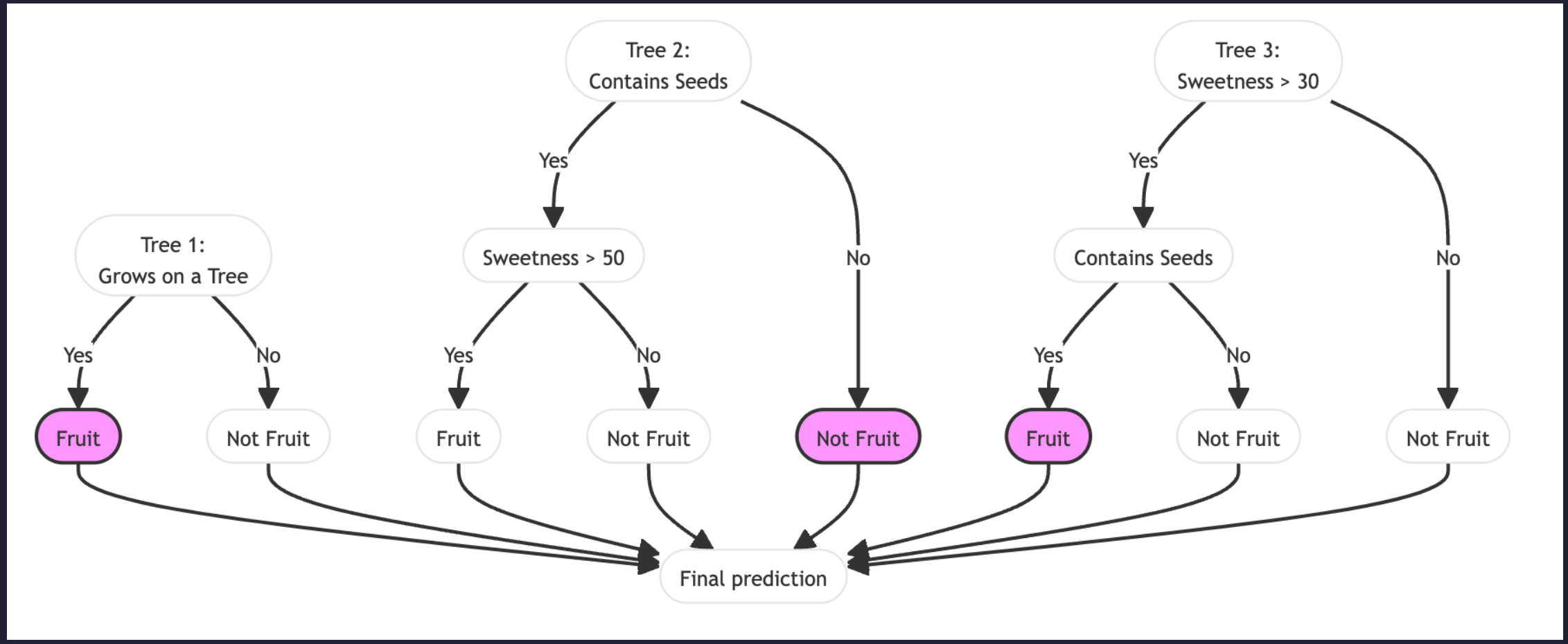
$$Var(D) = 876.6$$
$$Var(D_1, A) = 312.5$$
$$Var(D_2, A) = 339.26$$
$$Var(D|A) = \frac{4}{10} * 312.5 + \frac{6}{10} * 339.26$$
$$= 328.5$$
$$IG(D, A) = 876.6 - 328.5 = 548.1$$

Tree ensembles

Overfitting in decision trees

- **Decision trees can always classify training data *perfectly***
  - Keep splitting until each leaf node is one class (one value or one example)
- **Prone to overfitting**
- **Sensitive to noise in the data**
  - Small changes in the data can lead to different trees

Multiple decision trees: Tree ensembles



Multiple decision trees: Tree ensembles

- Methods that grow and combine multiple decision trees
- Reduces overfitting and sensitivity to noise
- Ensemble learning types:
  - **Bagging**: Builds multiple trees **independently**
  - **Boosting**: Builds trees **sequentially**, each focusing on correcting errors of the previous one

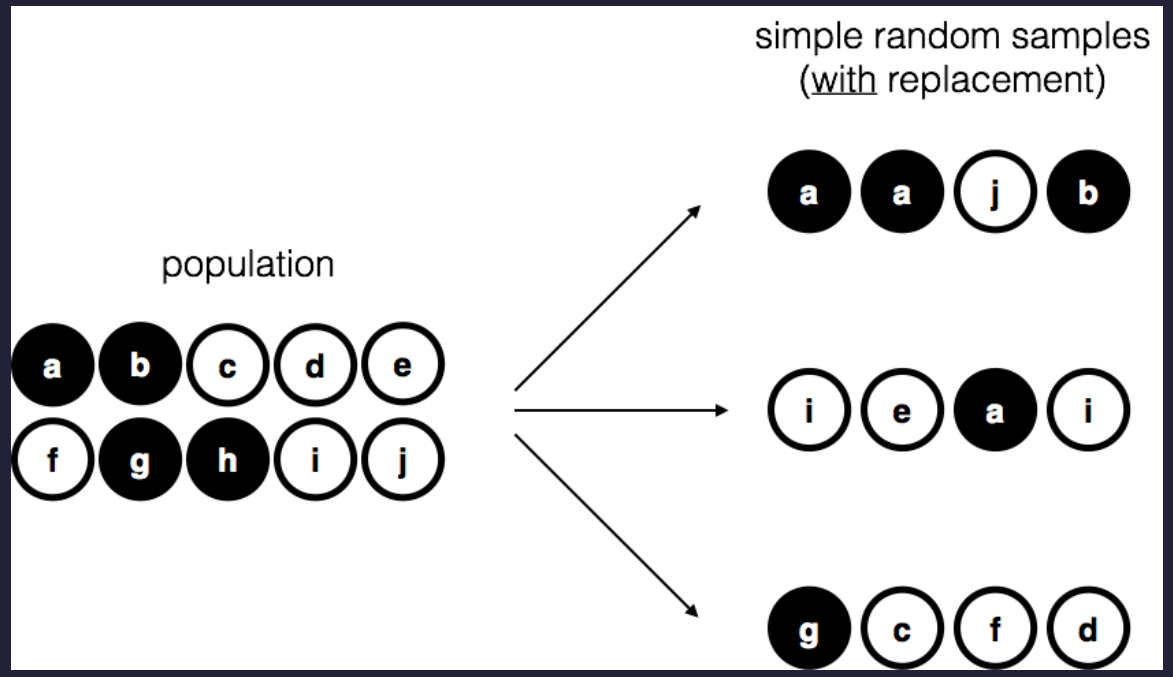
Sampling with replacement

Process:

- Given a dataset of size m
- Pick one, put it back, pick another
- Repeat the process m times

Creates multiple datasets with some randomness

- Some examples may be repeated, some may be left out



Random forest

Bagging:

- Given a dataset of size m, use sampling with replacement to create a new training set of size m
- Train a decision tree on the new dataset
- Repeat the process to create multiple trees

Randomized feature selection:

- At each split, choose a random subset of features to consider
- To reduce correlation between trees

Aggregation:

- Average or majority vote of predictions from all trees (unweighted)

AdaBoost

Adaptive Boosting:

- Given a dataset of size m, use sampling with replacement to create a new training set of size m
  - **higher weights to misclassified examples from previous trees**
- Train a decision tree on the new dataset
- Repeat the process to create multiple trees

Aggregation:

- Average or majority vote of predictions from all trees (weighted)

XGBoost

Gradient Boosting:

- Train a decision tree on the dataset
- **Fit a decision tree to the residuals (errors) of the previous tree**
- Update the residuals based on the new tree
- Repeat the process to create multiple trees

Aggregation:

- Sum of predictions from all trees (weighted)

💻 Decision trees