

Decision Trees

By Kayla Thomas

Thanks to: Ryan Henning



- Decision Trees
- Entropy
- Information Gain
- Recursion
- How to build a tree

Historical log of times I played tennis:

Temp	Outlook	Humidity	Windy	Played
Hot	Sunny	High	False	No
Hot	Sunny	High	True	No
Hot	Overcast	High	False	Yes
Cool	Rain	Normal	False	Yes
Cool	Overcast	Normal	True	Yes
Mild	Sunny	High	False	No
Cool	Sunny	Normal	False	Yes
Mild	Rain	Normal	False	Yes
Mild	Sunny	Normal	True	Yes
Mild	Overcast	High	True	Yes
Hot	Overcast	Normal	False	Yes
Mild	Rain	High	True	No
Cool	Rain	Normal	True	No
Mild	Rain	High	False	Yes

```
def will_play(temp, outlook, humidity,\n              windy):\n\n    if outlook == 'sunny':\n        if humidity == 'normal':\n            return True\n        else: # humidity == 'high'\n            return False\n\n    elif outlook == 'overcast':\n        return True\n\n    else: # outlook == 'rain'\n        if windy == True:\n            return False\n        else: # windy == False:\n            return True
```

DON'T WRITE CODE LIKE THIS!!!! AHHH!!! %#%#@#%

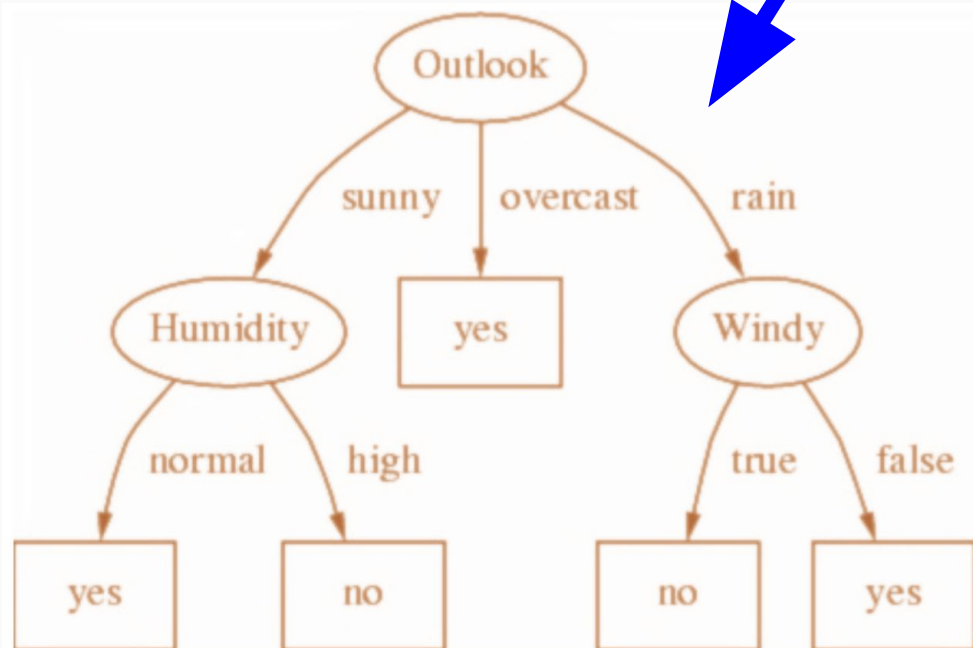
```
def will_play(temp, outlook, humidity,\n              windy):
```

```
    if outlook == 'sunny':\n        if humidity == 'normal':\n            return True\n        else: # humidity == 'high'\n            return False
```

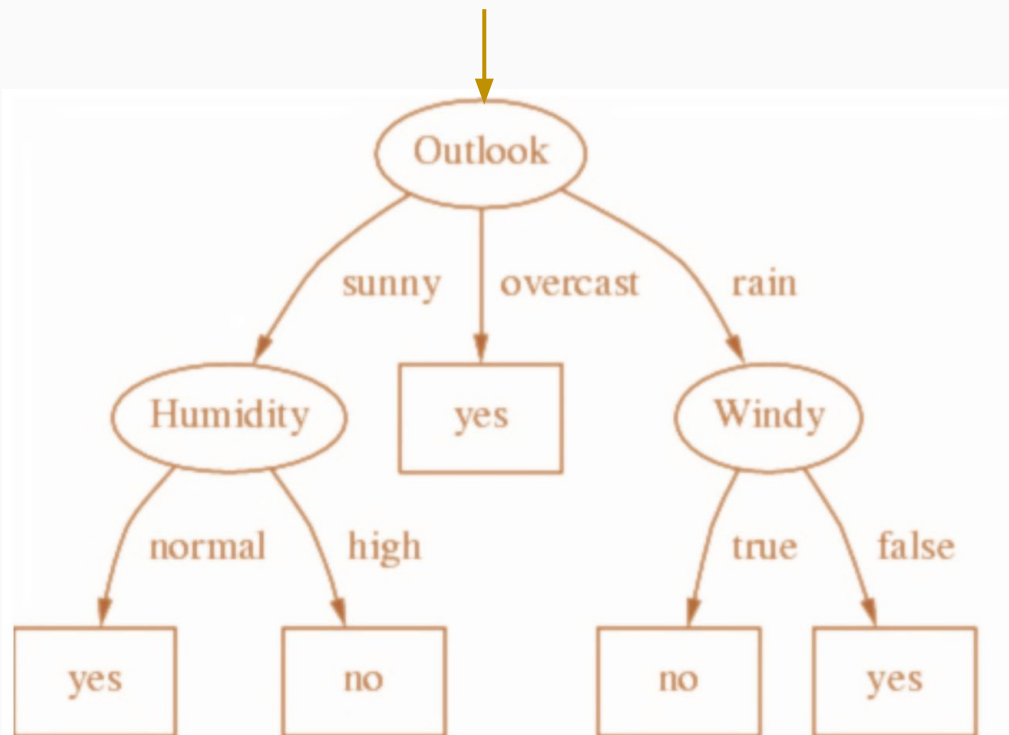
```
    elif outlook == 'overcast':\n        return True
```

```
    else: # outlook == 'rain'\n        if windy == True:\n            return False\n        else: # windy == False:\n            return True
```

Instead, let's write an algorithm to build a **Decision Tree** for us, based on the training data we have.



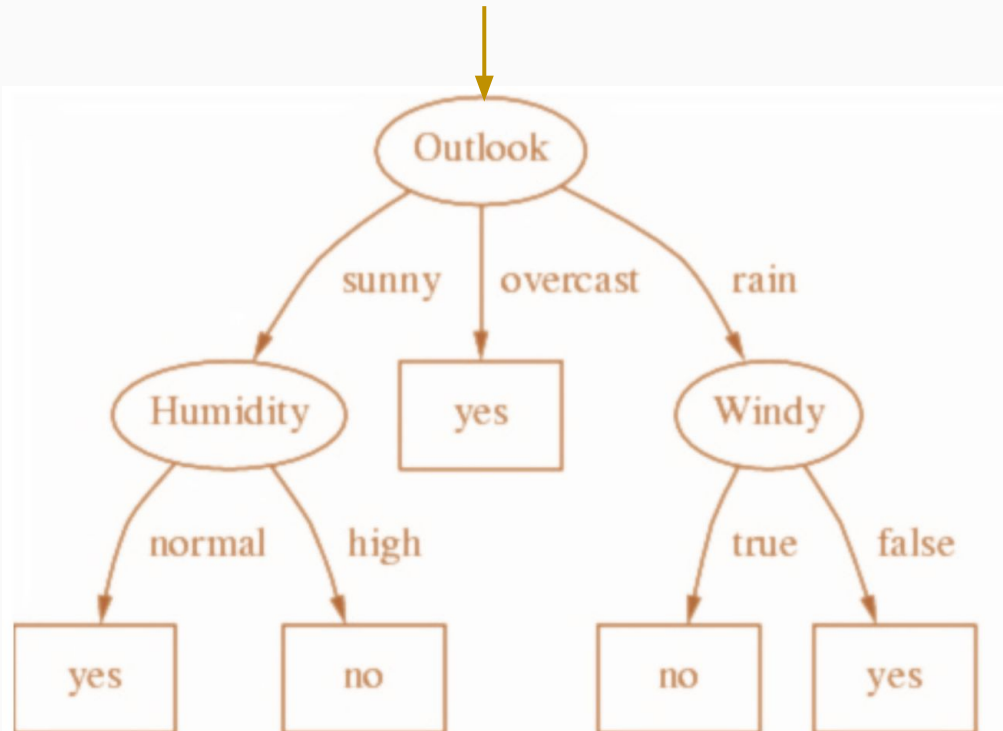
Will I play tennis?



Benefits:

- non-parametric, non-linear
- can be used for classification and for regression
- real and/or categorical features
- easy to interpret
- computationally cheap prediction
- handles missing values and outliers
- can handle irrelevant features

Will I play tennis?



Drawbacks:

- expensive to train
- greedy algorithm (local maxima)
- easily overfits
- right-angle decision boundaries only

But how can we build one of these from training data?

Shannon Entropy

discrete random
variable

information content
of X

number of bits needed to
encode each X event

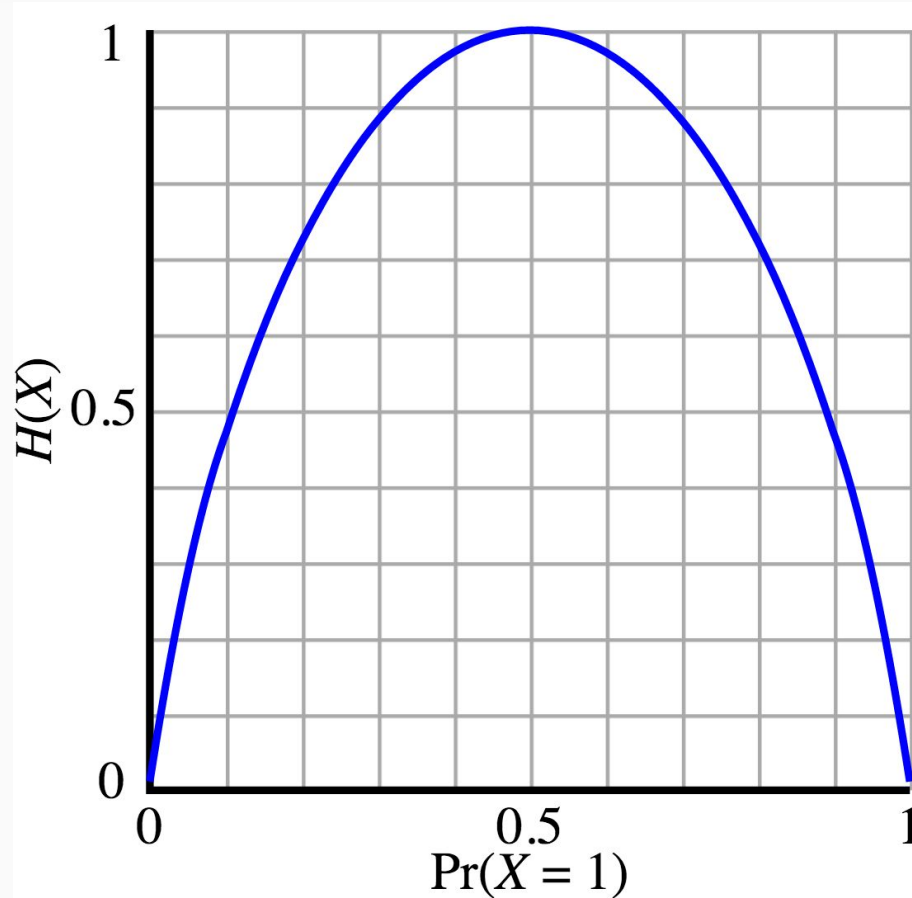
$$H(X) = E[I(X)] = E[\log_2(\frac{1}{P(X)})]$$

$$= -E[\log_2(P(X))]$$

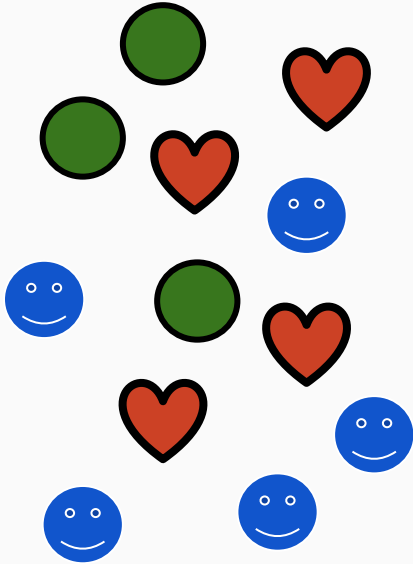
$$H(X) = - \sum_i p_i \log_2(p_i)$$

probability of
each possible
discrete outcome

iterate over pmf



We can measure the diversity of a set using Shannon Entropy (H) if we interpret the frequency of elements in the set as probabilities.



Estimate:

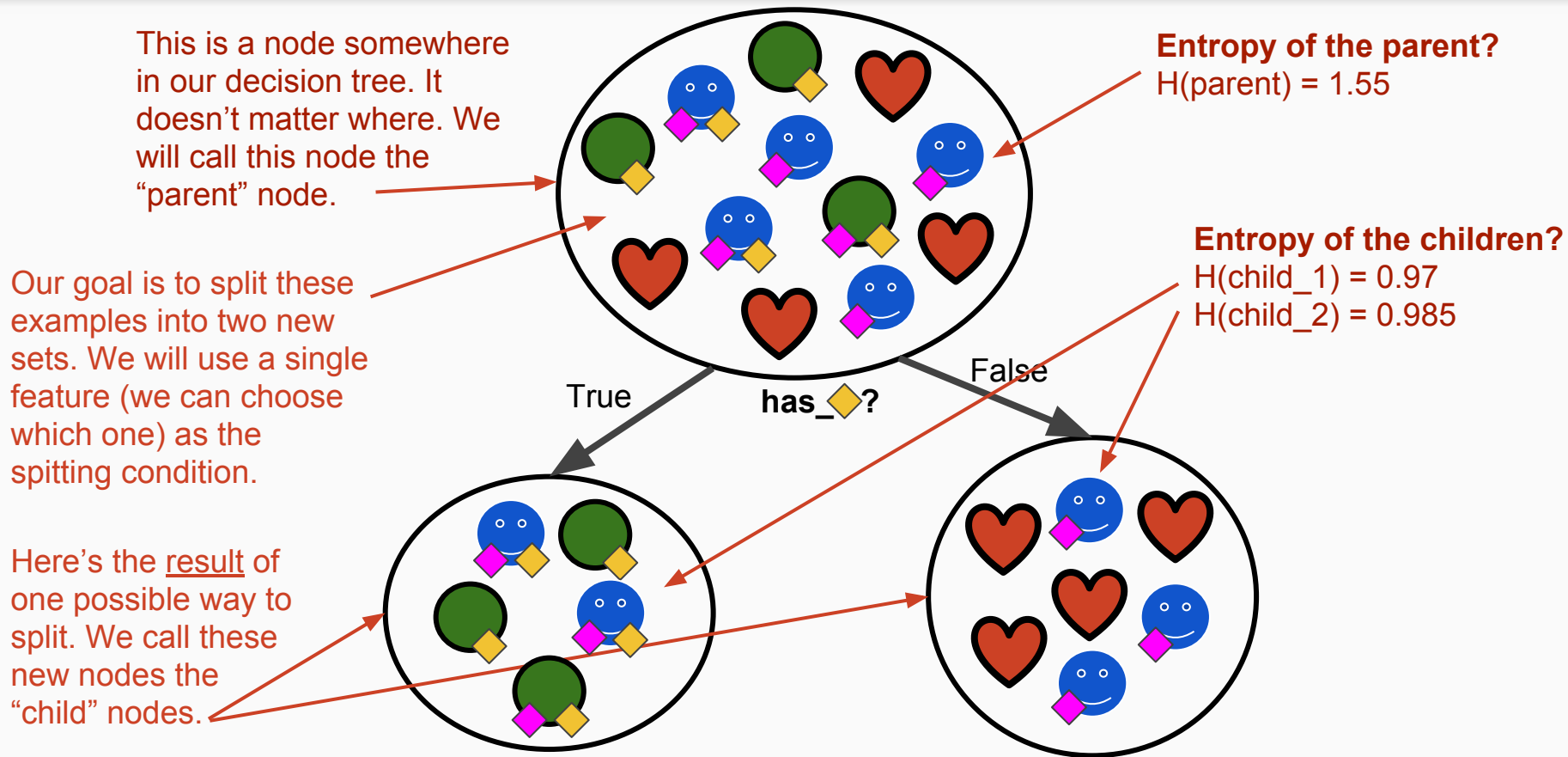
$$P(\text{green circle}) = 3/12 = 0.25$$

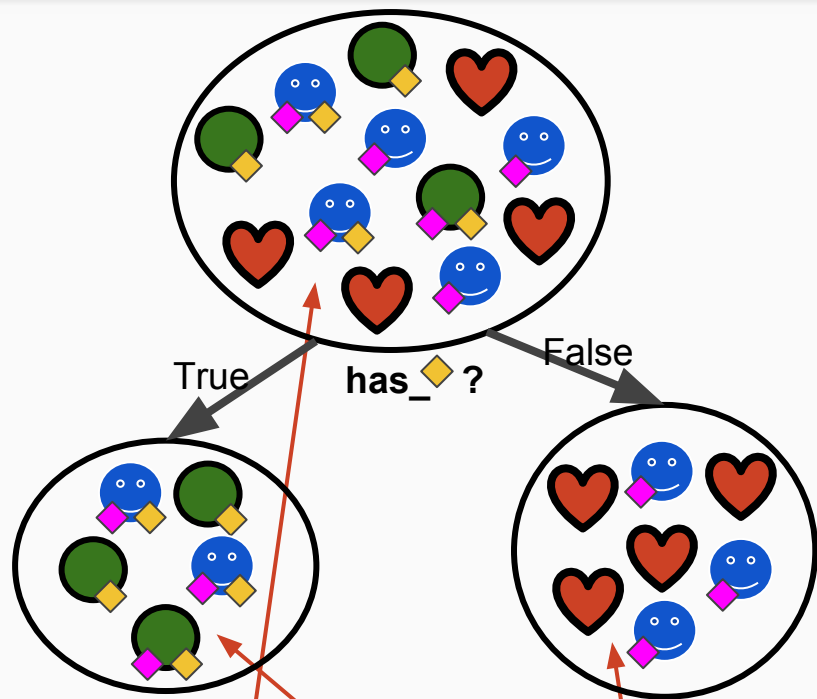
$$P(\text{red heart}) = 4/12 = 0.33$$

$$P(\text{blue smiley}) = 5/12 = 0.42$$

$$H = 1.55$$

One level in a decision tree:





Information gain
from this split

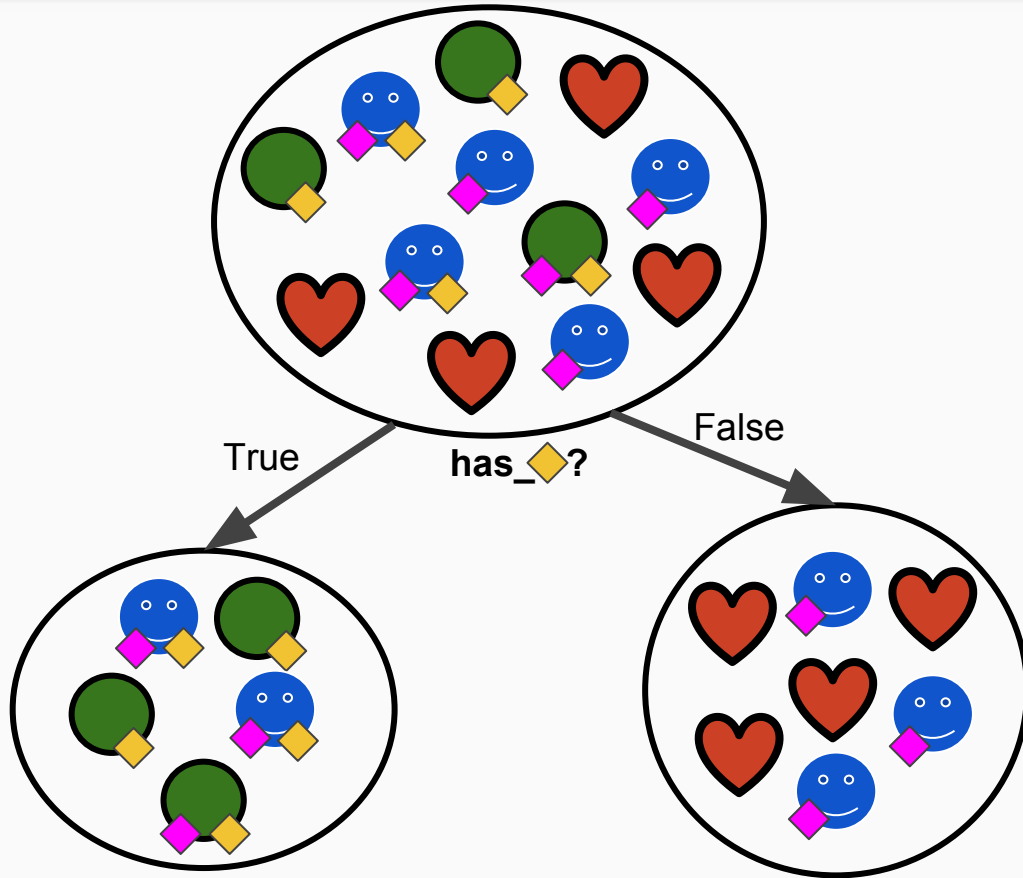
the set of
children

$$IG(S, C) = H(S) - \sum_{C_i \in C} \frac{|C_i|}{|S|} H(C_i)$$

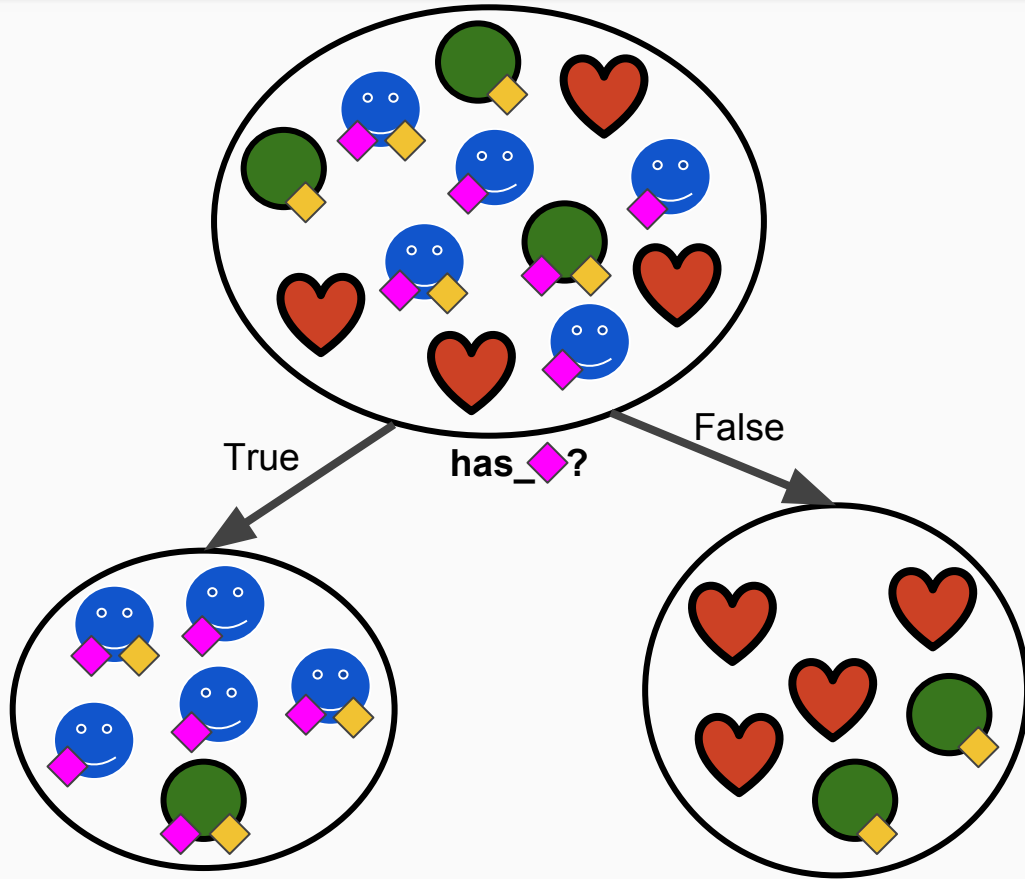
the parent's set
of examples

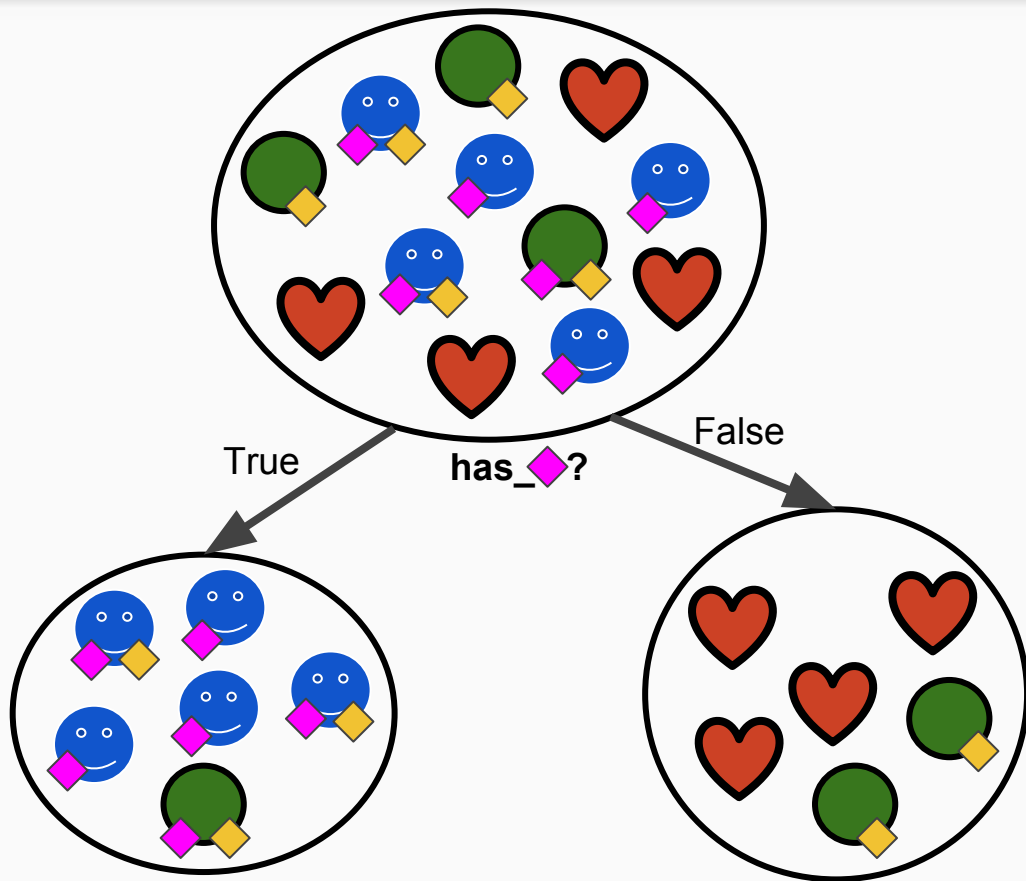
the set of
examples in
each child

$$IG(\text{parent}, \{\text{child}_1, \text{child}_2\}) = 1.55 - 5/12 * 0.97 - 7/12 * 0.985 = 0.57$$



Information Gain = 0.57





Information Gain = 0.765

MORE THAN
THE LAST
SPLIT. THIS IS
GOOD!

Splitting Algorithm:

Possible Splits:

Consider all binary splits based on a single feature:

- if the feature is categorical, split on value or not value.
- if the feature is numeric, split at a threshold: $>\text{threshold}$ or $\leq \text{threshold}$

Splitting Algorithm:

1. Calculate the information gain for all possible splits.
2. Commit to the split that has the highest information gain.

Recursion

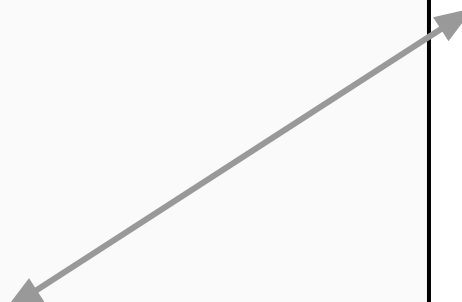
What is this function?

$$f(x) = \prod_{i=1}^x i$$

Is this an equivalent function?

$$f(x) = \begin{cases} 1, & \text{if } x \leq 1 \\ x f(x-1), & \text{otherwise} \end{cases}$$

```
def f(x):  
    '''  
    This function returns x!.  
    >>> f(5)  
    120  
    '''  
    if x <= 1:  
        return 1  
    else:  
        return x * f(x-1)  
  
if __name__ == '__main__':  
    import doctest  
    doctest.testmod()
```



How to build a decision tree (pseudocode):

```
function BuildTree:
```

```
    If every item in the dataset is in the same class  
    or there is no feature left to split the data:
```

```
        return a leaf node with the class label
```

```
    Else:
```

```
        find the best feature and value to split the data
```

```
        split the dataset
```

```
        create a node
```

```
        for each split
```

```
            call BuildTree and add the result as a child of the node
```

```
        return node
```


The Gini Index

A measure of impurity: the probability of a misclassification if a random sample drawn from the set is classified according to the distribution of classes in the set

Scikit-learn doesn't use *Shannon Entropy Diversity* by default. It uses the *Gini Index*:

$$\text{Gini}(S) = 1 - \sum_{i \in S} p_i^2$$

Information gain using the *Gini Index*:

$$\text{IG}(S, C) = \text{Gini}(S) - \sum_{C_i \in C} \frac{|C_i|}{|S|} \text{Gini}(C_i)$$

Regression Trees

Targets are real values... so...

now we can't use Information Gain or Gini Index for splitting! What do we do?

Use *variance*! Cool, now we can train.

How do we predict?

Either predict the mean value of the leaf, or do linear regression within the leaf!

Overfitting is likely if you build your tree all the way until every leaf is pure.

Prepruning ideas (prune while you build the tree):

- **leaf size:** stop splitting when #examples gets small enough
- **depth:** stop splitting at a certain depth
- **purity:** stop splitting if enough of the examples are the same class
- **gain threshold:** stop splitting when the information gain becomes too small

Postpruning ideas (prune after you've finished building the tree):

- merge leaves if doing so decreases test-set error
- Basically grow the tree large then punish trees with more terminal nodes
- (see pair.md for details)

Algorithm Names:

The details of training a decision tree vary... each specific algorithm has a name. Here are a few you'll often see:

- **ID3:** category features only, information gain, multi-way splits, ...
- **C4.5:** continuous and categorical features, information gain, missing data okay, pruning, ...
- **CART:** continuous and categorical features and targets, gini index, binary splits only, ...
- SciKit-learn uses an optimized version of CART -important to note that CART uses only Binary splits (for categorical: Value or not value, continuous: $>\text{threshold}$, or $\leq \text{threshold}$)

Summary

- Trees are easy to explain often even easier than a linear regression
- Mirrors human decision making
- Trees can be displayed graphically which makes them easy to interpret especially for non-experts
- Handles numeric and categorical features
- Alone decision trees are not as accurate at predicting, but when combined in ensemble methods trees performance can be greatly improved