Decision Trees

By Kayla Thomas
Thanks to: Ryan Henning
Edit by Thomas Duffy 1-21-20

galvanize

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

Historical log of times I played tennis:

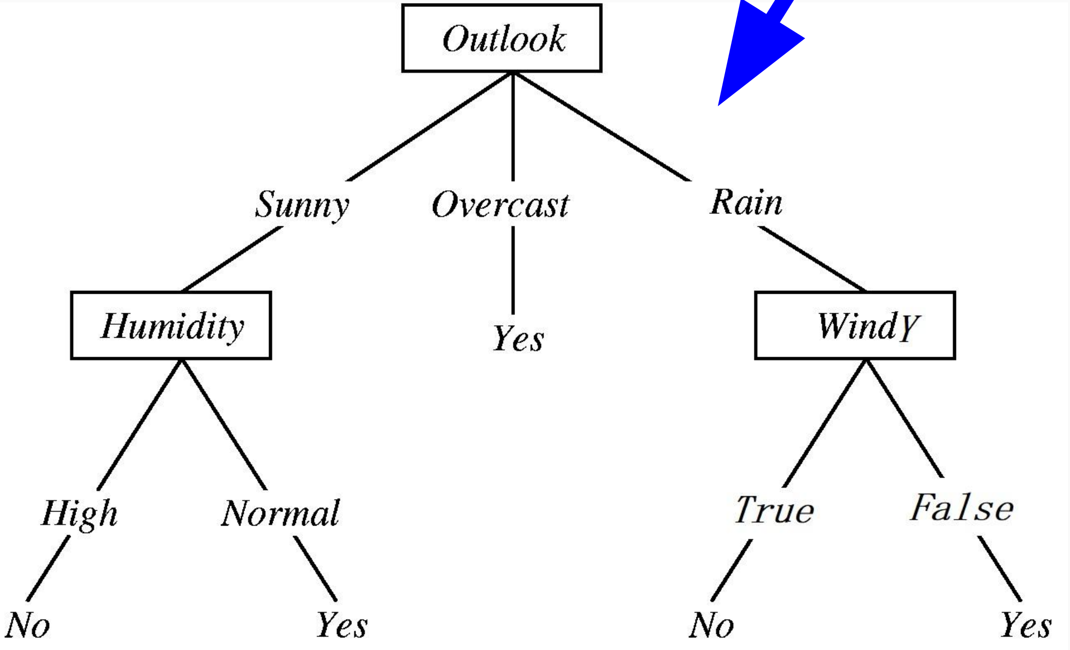
Temp	Outlook	Humidity	Windy	Played
Hot	Sunny	High	False	No
Hot	Sunny	High	True	No
Hot	Overcast	High	False	Yes
Cool	Rain	Normal	False	Yes
Cool	Overcast	Normal	True	Yes
Mild	Sunny	High	False	No
Cool	Sunny	Normal	False	Yes
Mild	Rain	Normal	False	Yes
Mild	Sunny	Normal	True	Yes
Mild	Overcast	High	True	Yes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

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

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

    elif outlook == 'overcast':
        return True

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

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

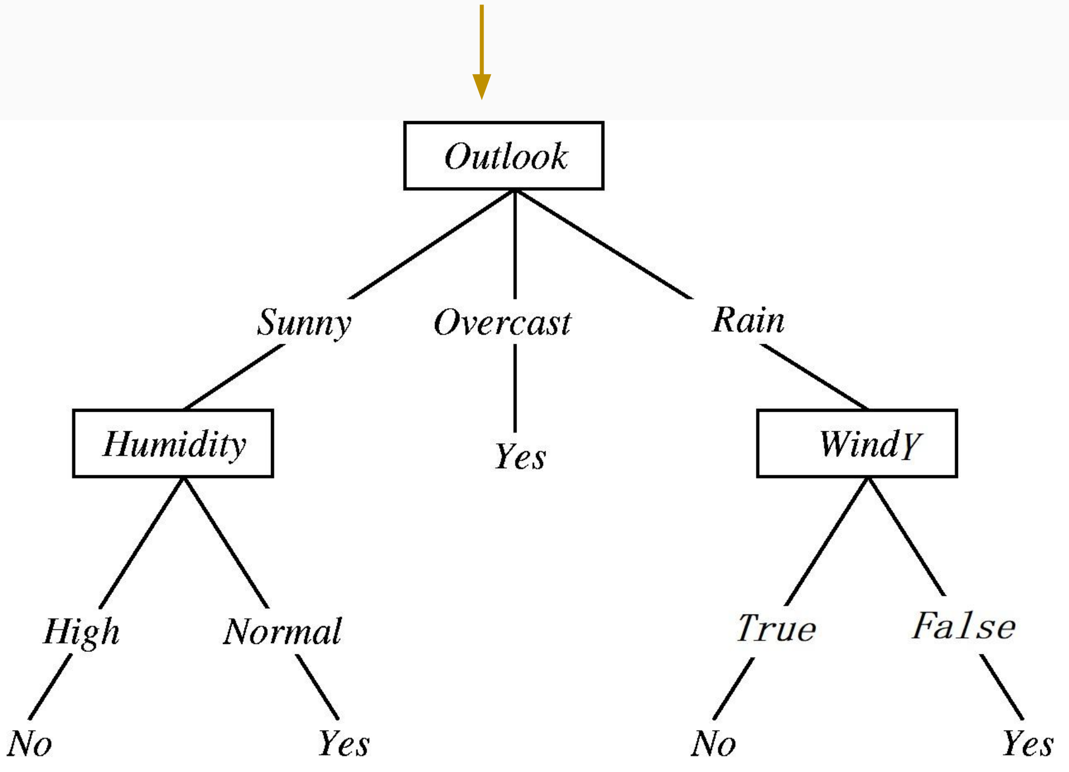
```
def will_play(temp, outlook, humidity,\
              windy):

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

    elif outlook == 'overcast':
        return True

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

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



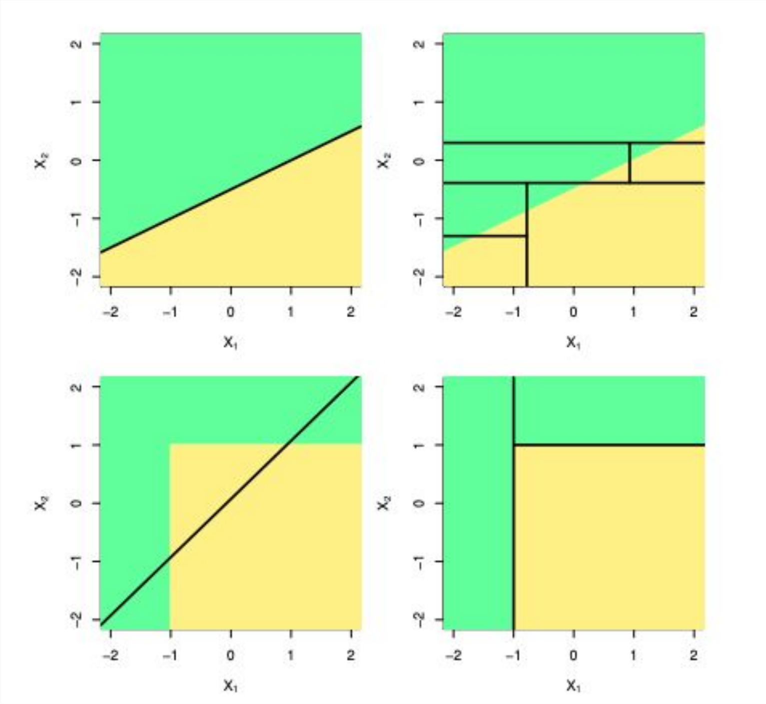
Benefits of a decision tree:

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



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?

Entropy (a measure of information in an event stream)

galvanize

Shannon Entropy

discrete random variable

information content of X

number of bits needed to encode each X event

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

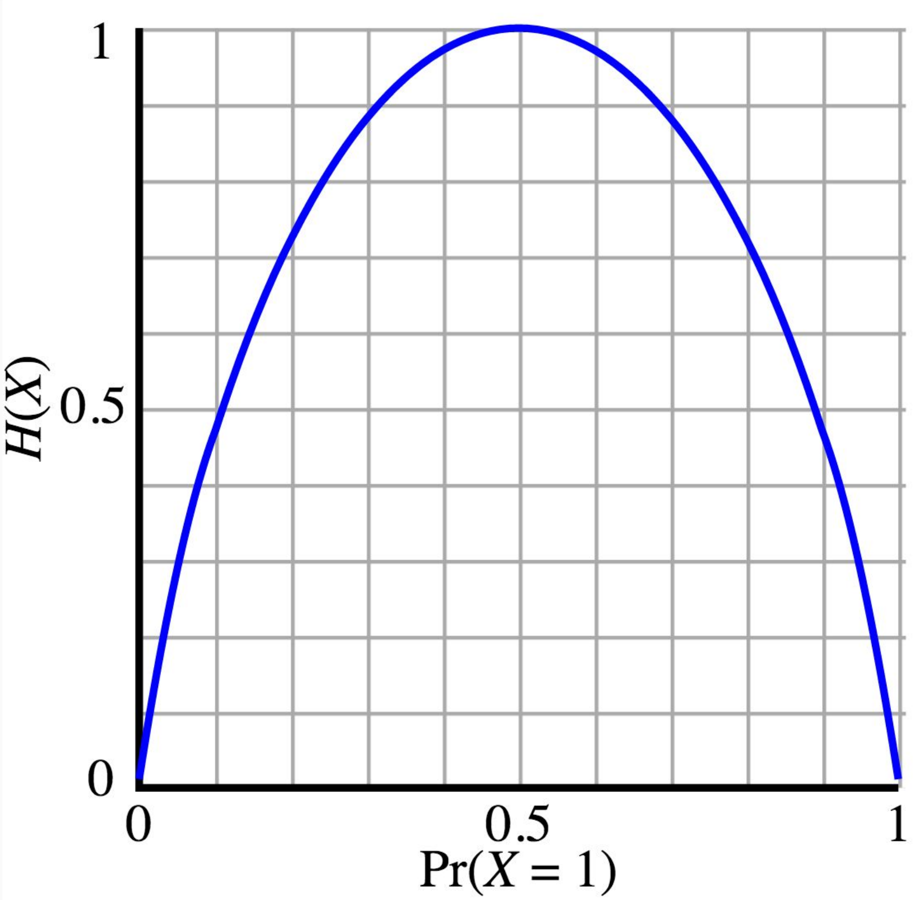
$$= -E[log_2(P(X))]$$

probability of each possible discrete outcome

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

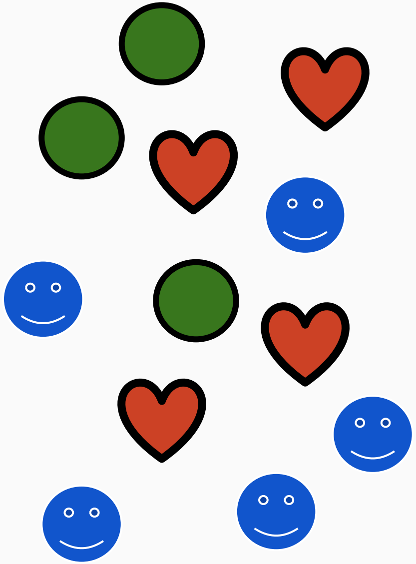
iterate over pmf

Entropy graph of a Bernoulli random variable X

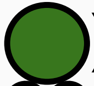


Shannon Entropy Diversity Index (aka, the Shannon Index)

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() = 3/12 = 0.25

P() = 4/12 = 0.33

P() = 5/12 = 0.42

H = 1.55

Shannon Entropy in Python

Now lets go over this in Python:

```
import math
prob_circle = 3/12
prob_heart = 4/12
prob_smile = 5/12

H = 0
for probability in [prob_circle,prob_heart,prob_smile]:
    H+=(probability*math.log(probability,2))
H*=-1
print(H)
```

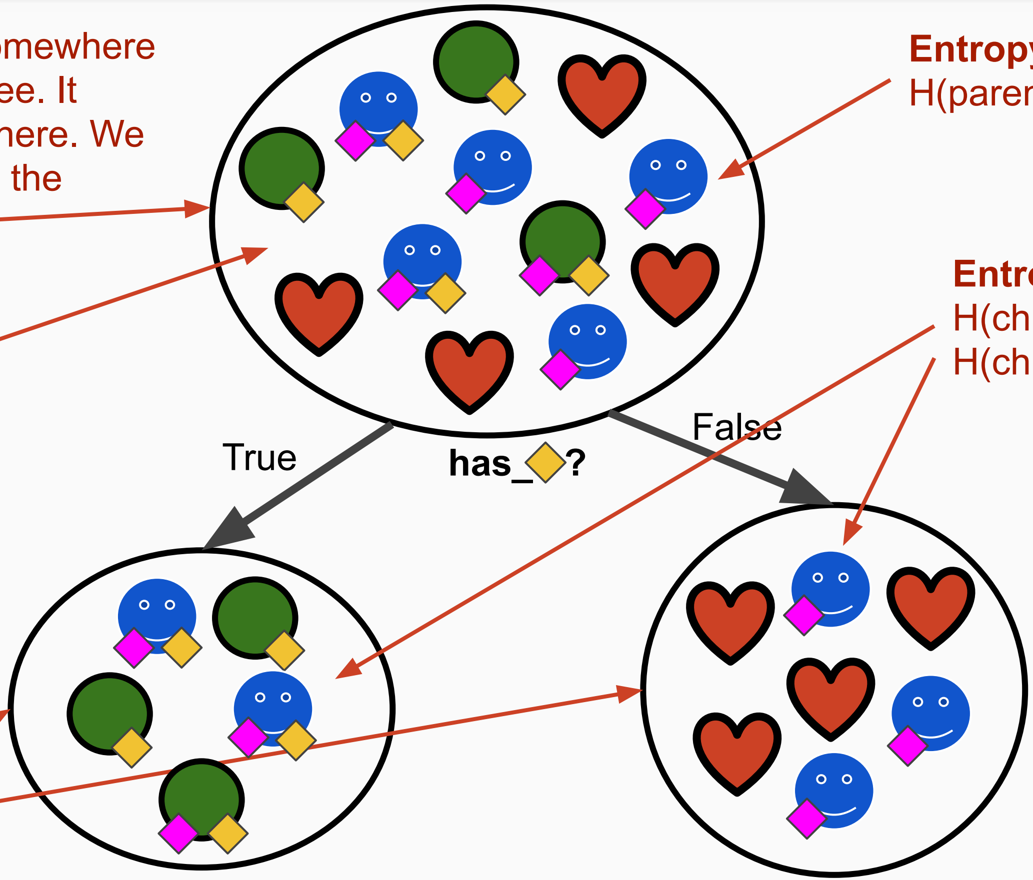
H=1.55

One level in a decision tree:

This is a node somewhere in our decision tree. It doesn't matter where. We will call this node the "parent" node.

Our goal is to split these examples into two new sets. We will use a single feature (we can choose which one) as the spitting condition.

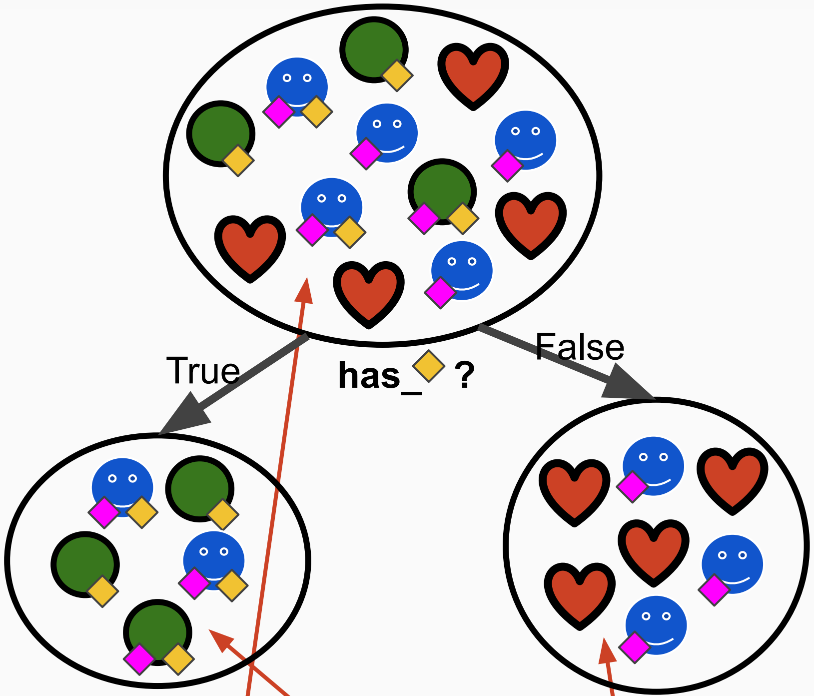
Here's the <u>result</u> of one possible way to split. We call these new nodes the "child" nodes.



Entropy of the parent?
H(parent) = 1.55

Entropy of the children?
H(child_1) = 0.97
H(child_2) = 0.985

Information Gain (using Shannon Entropy Diversity Index)



Information gain from this split

the set of children

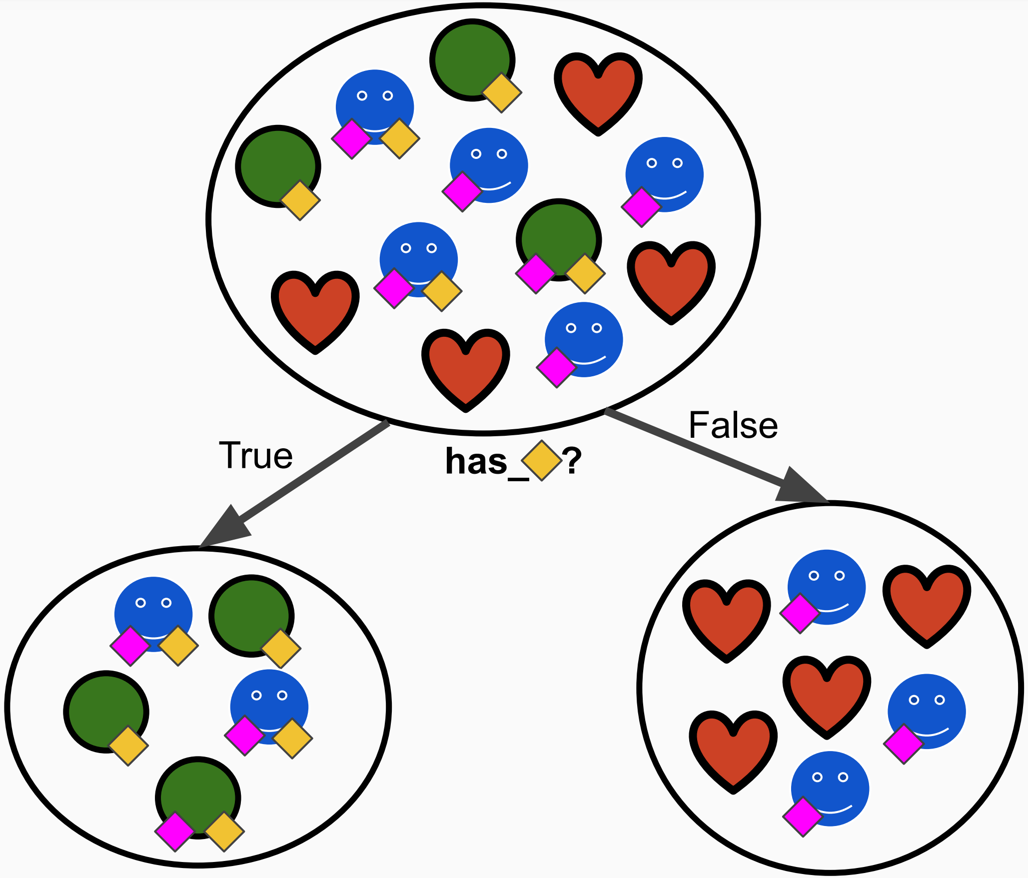
$$\mathrm{IG}(S, C) = H(S) - \sum_{C_i \in C} \frac{|C_i|}{|S|} H(C_i)$$

the parent's set of examples

the set of examples in each child

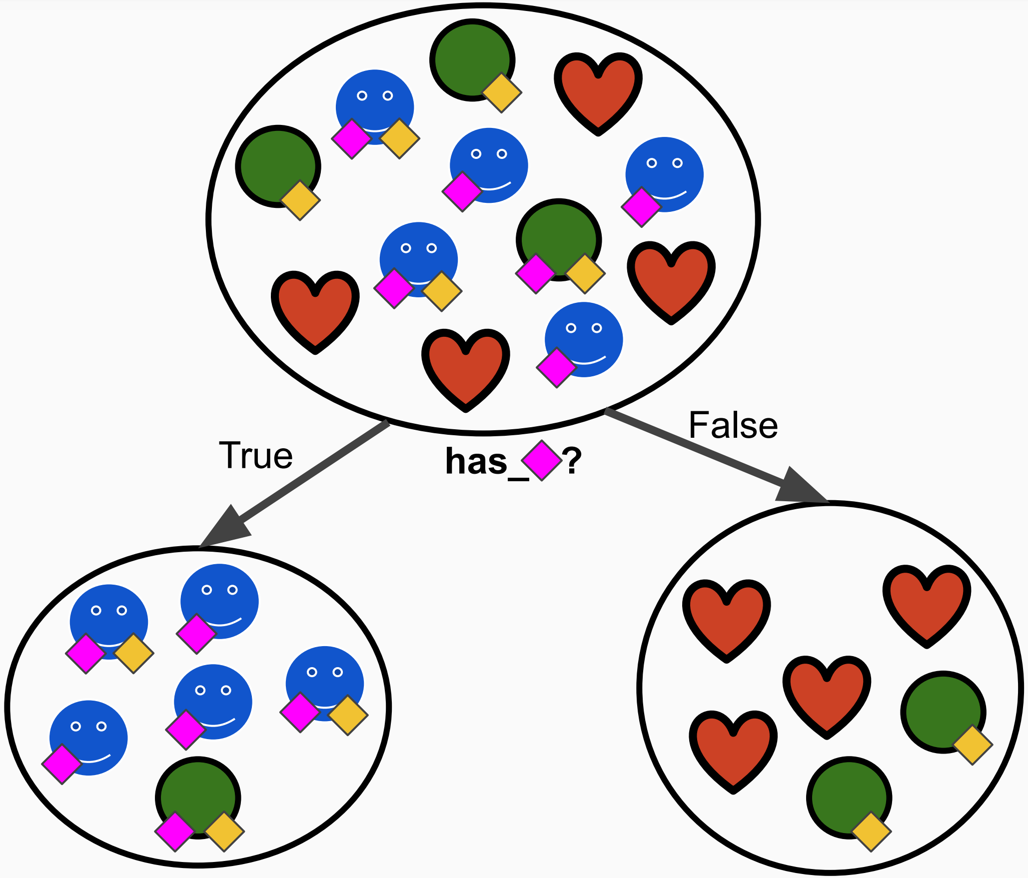
$$\mathrm{IG}(\text{parent}, \{\text{child\_1}, \text{child\_2}\}) = 1.55 - 5/12 * 0.97 - 7/12 * 0.985 = 0.57$$

Information Gain

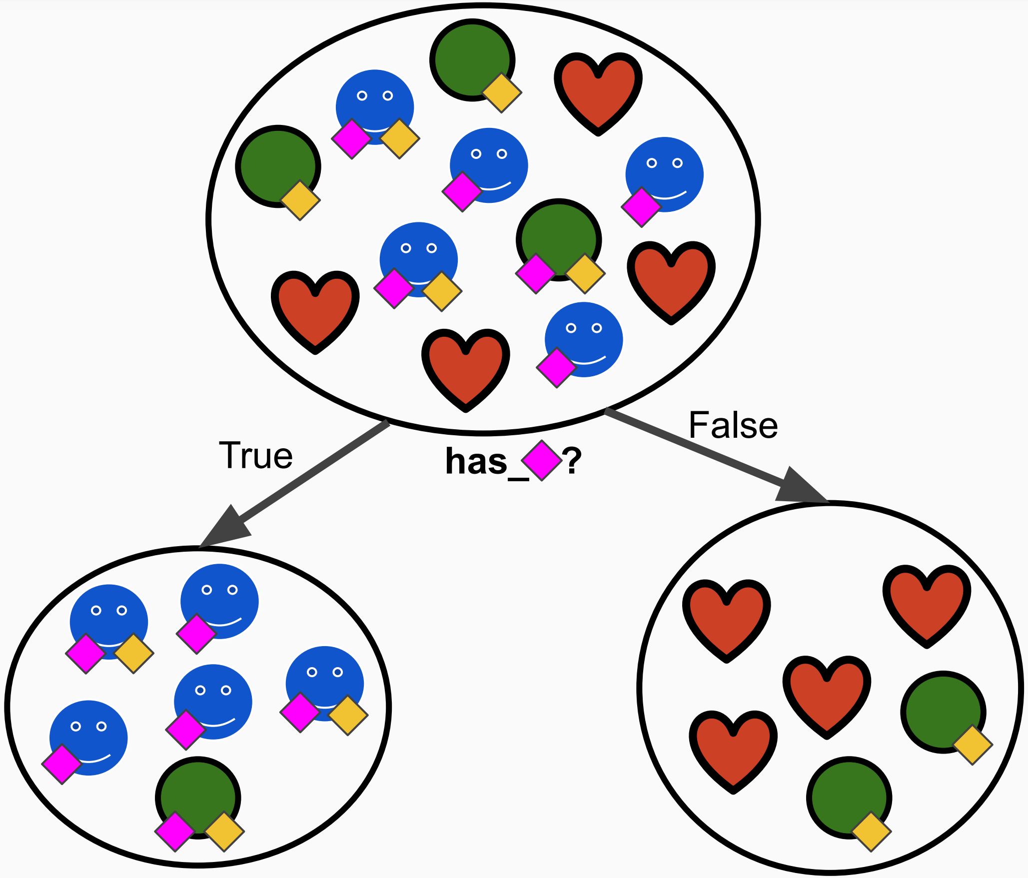


Information Gain = 0.57

Information Gain



Information Gain



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: >threshold or <=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?

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

```python
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()
```

Recursion cont.

So based off of the last slide let's use the example f(x) with x=5

1. So first we get 5*f(5-1)
2. Next we get 5*(4*f(4-1))
3. Then we get 5*(4*(3*f(3-1)))
4. Then we get 5*(4*(3*(2*f(2-1))))
5. Finally we get 5*(4*(3*(2*(1))))
6. This will equal 120 which is the same as 5! (factorial)

Recursion gif

Recursion Factorial Gif

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*:

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

Information gain using the *Gini Index*:

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

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.

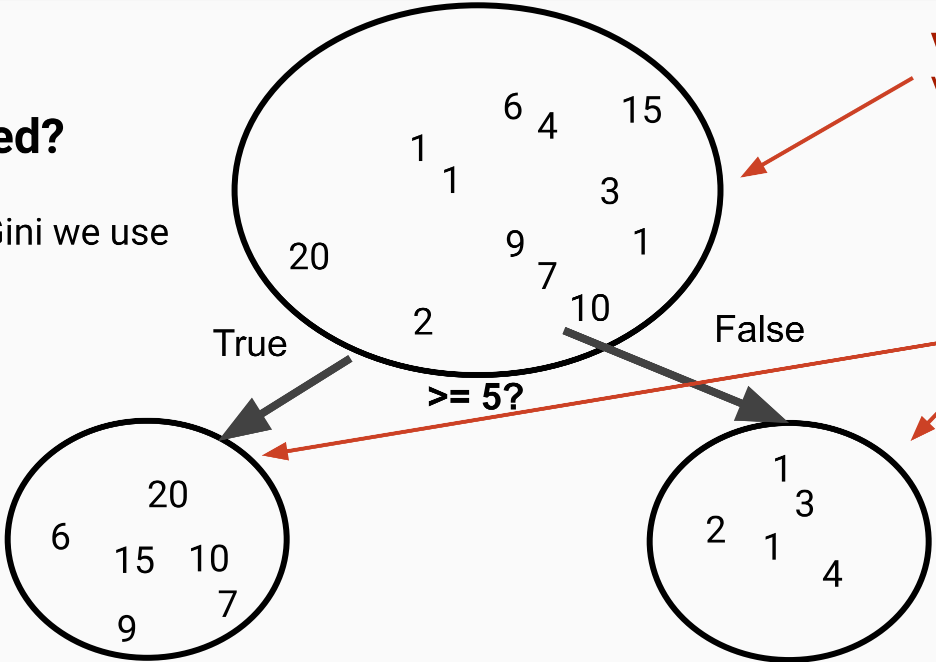
How do we predict?

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

Regression Tree examples

Information Gained?

Instead of Entropy or Gini we use Variance!

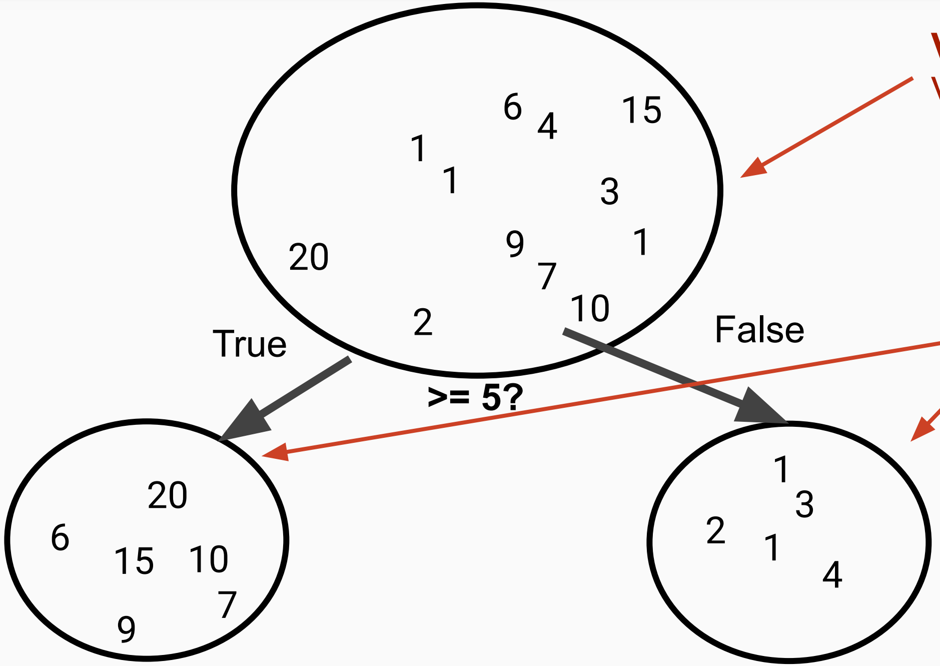


Variance of the parent?
Variance(P) = 33.537

Variance of the children?
Variance(C1) = 23.805
Variance(C2) = 1.359

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

Regression Tree examples



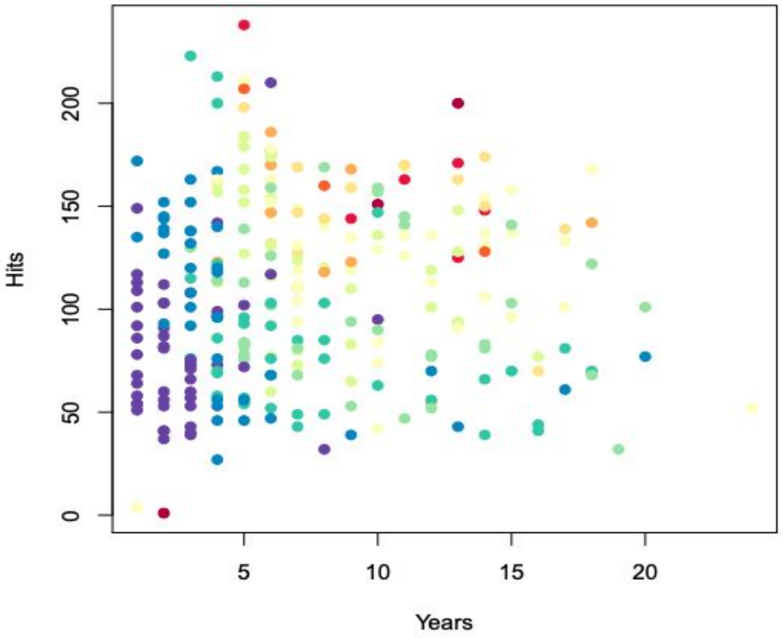
Variance of the parent?
Variance(P) = 33.537

Variance of the children?
Variance(C1) = 23.805
Variance(C2) = 1.359

$$IG(S,C) = 33.537 - \left(\frac{6}{11}(23.805) + \frac{5}{11}(1.359)\right) = 19.934$$

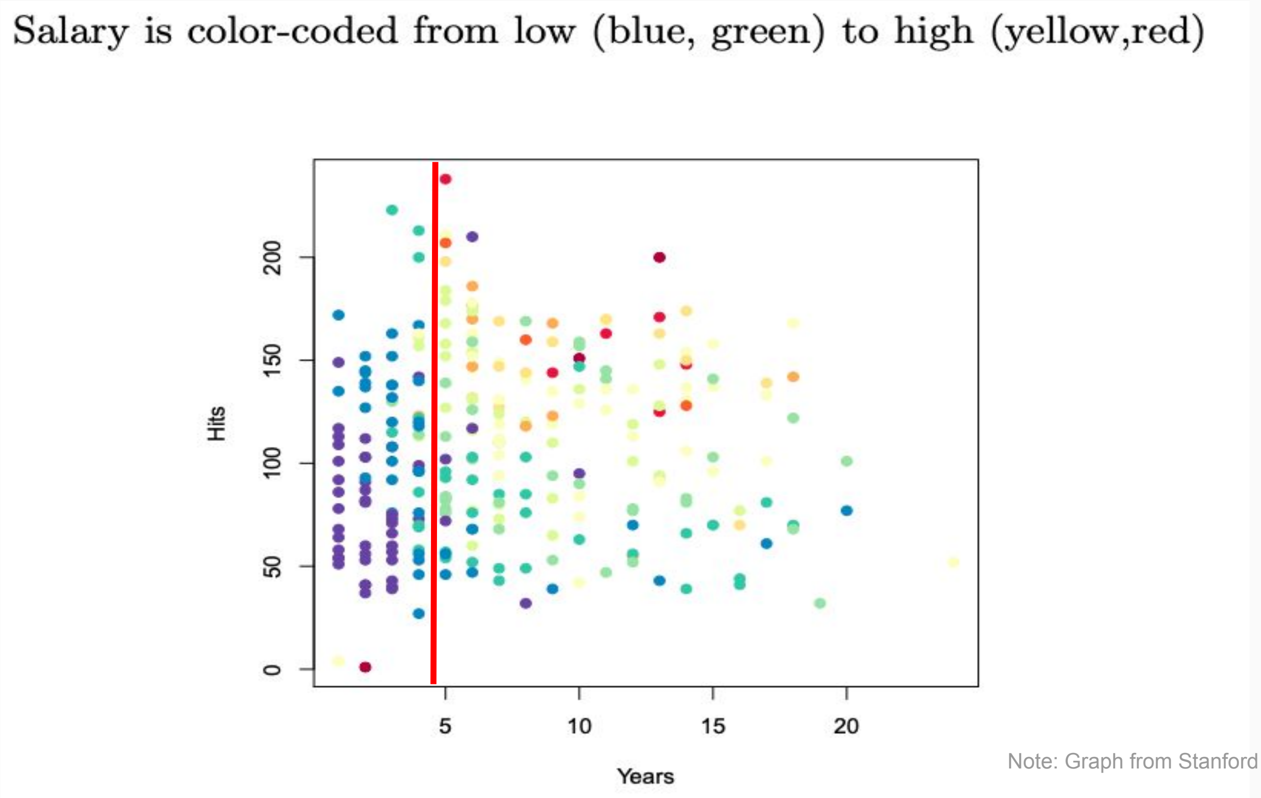
Baseball salary data: How would you stratify it?

Salary is color-coded from low (blue, green) to high (yellow,red)



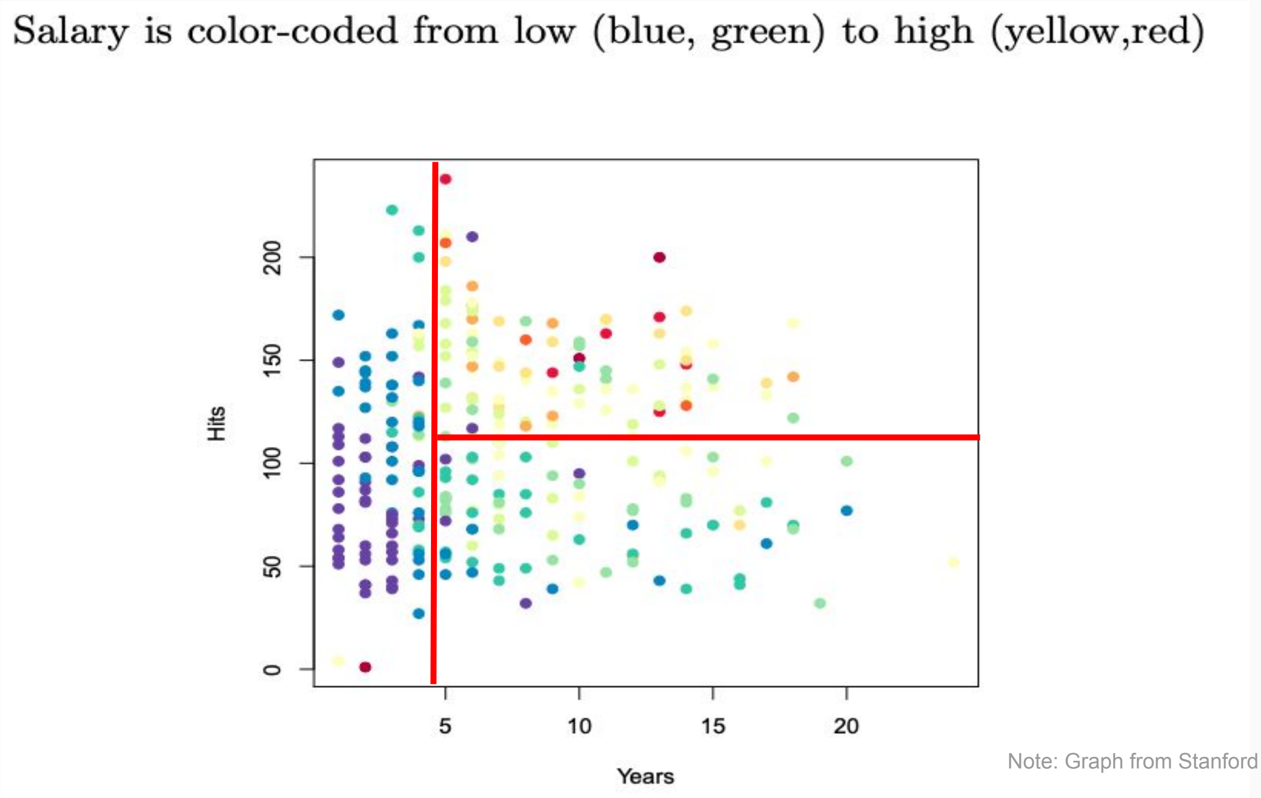
Note: Graph from Stanford - Statistical learning

Baseball salary data: How would you stratify it?



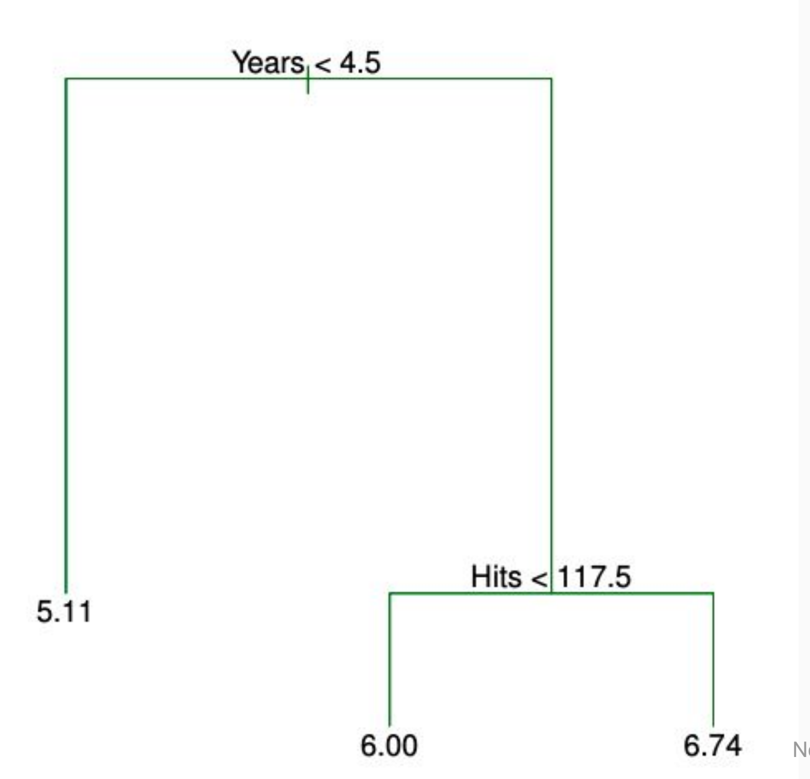
Note: Graph from Stanford - Statistical learning

Baseball salary data: How would you stratify it?



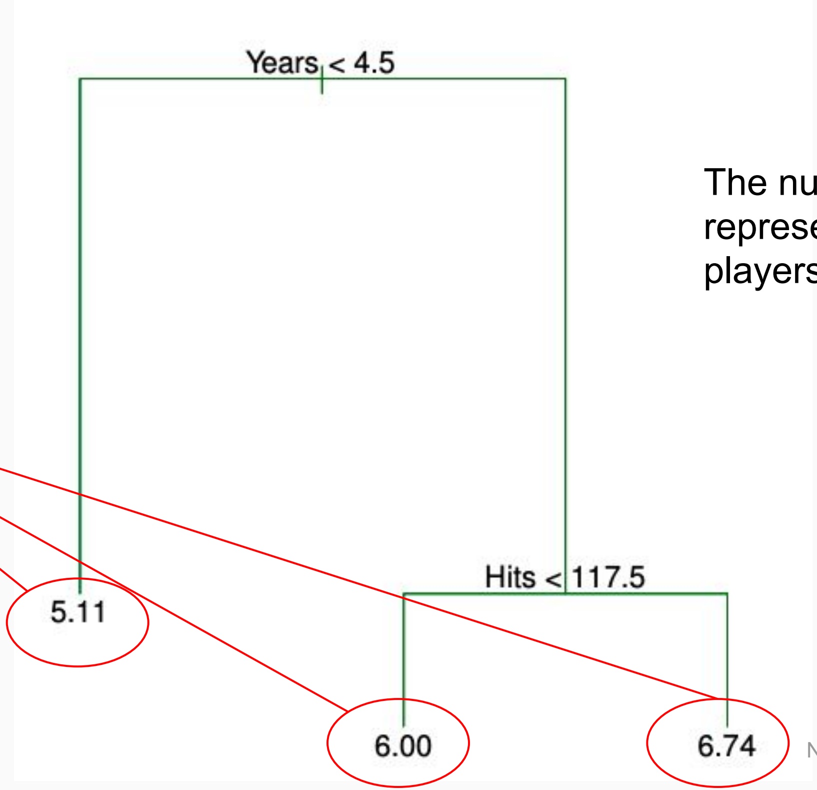
Note: Graph from Stanford - Statistical learning

Decision tree for baseball salary data



Note: Graph from Stanford - Statistical learning

Decision tree for baseball salary data



Terminal Nodes
(aka leaves)

The number in the terminal nodes represents the mean (log salary) of players that fall into each.

Note: Graph from Stanford - Statistical learning

The “Pretty” version of this:

Decision Tree Visualization

Pruning (Remember Hyper Parameters?)

galvanize

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

Pre Pruning ideas - setting limitations on the building of the decision 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

Post Pruning - Let the tree grow large and prune it back:

- Cost complexity pruning - punish trees with more terminal nodes

Let us take a look at these in scikit learn! Classifier Regressor

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: >threshold, or <=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

Questions!

1. What’s the benefits of Decision Trees?
2. What's the drawbacks of Decision Trees?
3. How can we quantify the “randomness” or diversity of a node?
4. How can we tell if a split is a good split?
   a. With classification/regression?
5. Summarize recursion at a high level.
6. How do you make sure your tree does not overfit?

What questions do you have?