



# Decision and Regression Trees

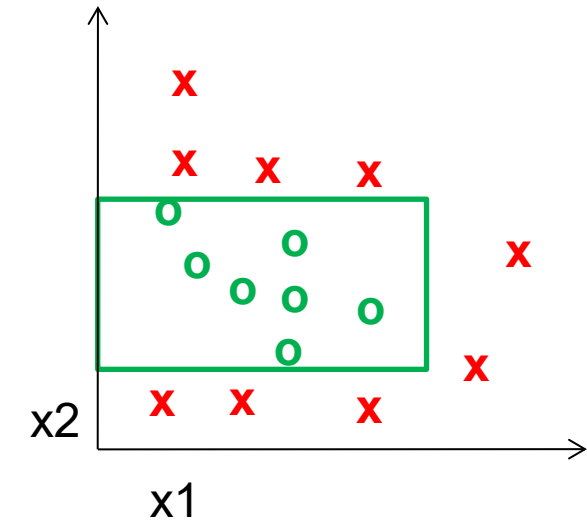
Applied Machine Learning  
Derek Hoiem

Dall-E: A dirt road splits around a large gnarly tree, fractal art

So far, we've seen two main choices for how to use features

1. Nearest neighbor uses all the features jointly to find similar examples
2. Linear models make predictions out of weighted sums of the features

If you wanted to give someone a rule to split the 'o' from the 'x', what other idea might you try?

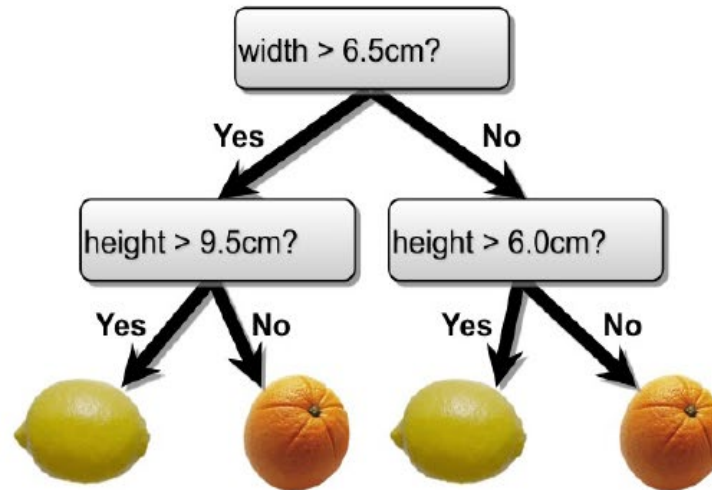
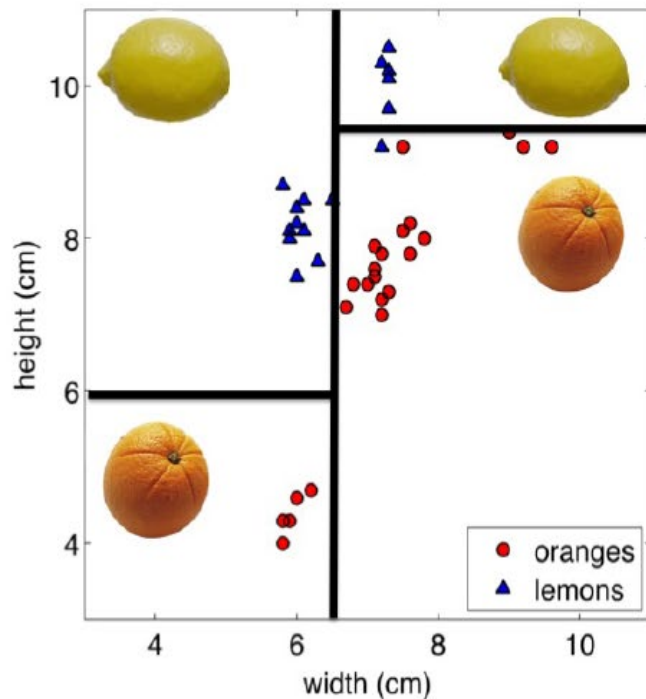


```
If  $x_2 < 0.6$  and  $x_2 > 0.2$  and  $x_1 < 0.7$ , 'o'  
Else 'x'
```

Can we learn these kinds of rules automatically?

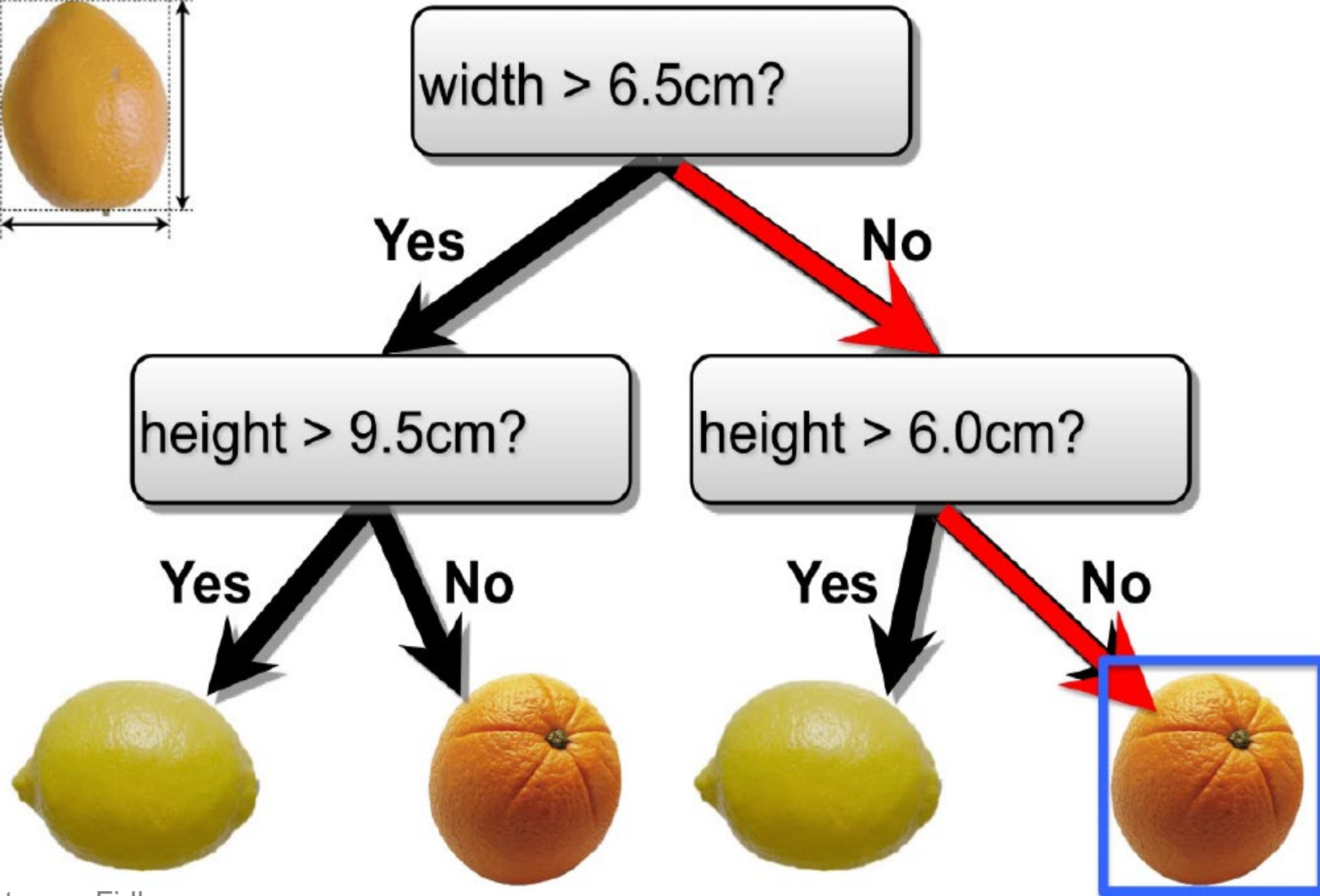
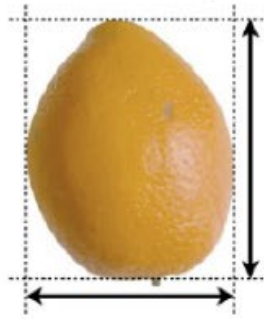
# Decision trees

- Training: Iteratively choose the attribute and split value that best separates the classes for the data in the current node
- Combines feature selection/modeling with prediction



# Decision Tree Classification

Test example



# Example with discrete inputs

| Example  | Input Attributes |            |            |            |            |              |             |            |             |            | Goal                  |
|----------|------------------|------------|------------|------------|------------|--------------|-------------|------------|-------------|------------|-----------------------|
|          | <i>Alt</i>       | <i>Bar</i> | <i>Fri</i> | <i>Hun</i> | <i>Pat</i> | <i>Price</i> | <i>Rain</i> | <i>Res</i> | <i>Type</i> | <i>Est</i> | <i>WillWait</i>       |
| $x_1$    | Yes              | No         | No         | Yes        | Some       | \$\$\$       | No          | Yes        | French      | 0-10       | $y_1 = \text{Yes}$    |
| $x_2$    | Yes              | No         | No         | Yes        | Full       | \$           | No          | No         | Thai        | 30-60      | $y_2 = \text{No}$     |
| $x_3$    | No               | Yes        | No         | No         | Some       | \$           | No          | No         | Burger      | 0-10       | $y_3 = \text{Yes}$    |
| $x_4$    | Yes              | No         | Yes        | Yes        | Full       | \$           | Yes         | No         | Thai        | 10-30      | $y_4 = \text{Yes}$    |
| $x_5$    | Yes              | No         | Yes        | No         | Full       | \$\$\$       | No          | Yes        | French      | >60        | $y_5 = \text{No}$     |
| $x_6$    | No               | Yes        | No         | Yes        | Some       | \$\$         | Yes         | Yes        | Italian     | 0-10       | $y_6 = \text{Yes}$    |
| $x_7$    | No               | Yes        | No         | No         | None       | \$           | Yes         | No         | Burger      | 0-10       | $y_7 = \text{No}$     |
| $x_8$    | No               | No         | No         | Yes        | Some       | \$\$         | Yes         | Yes        | Thai        | 0-10       | $y_8 = \text{Yes}$    |
| $x_9$    | No               | Yes        | Yes        | No         | Full       | \$           | Yes         | No         | Burger      | >60        | $y_9 = \text{No}$     |
| $x_{10}$ | Yes              | Yes        | Yes        | Yes        | Full       | \$\$\$       | No          | Yes        | Italian     | 10-30      | $y_{10} = \text{No}$  |
| $x_{11}$ | No               | No         | No         | No         | None       | \$           | No          | No         | Thai        | 0-10       | $y_{11} = \text{No}$  |
| $x_{12}$ | Yes              | Yes        | Yes        | Yes        | Full       | \$           | No          | No         | Burger      | 30-60      | $y_{12} = \text{Yes}$ |

Attributes:

|     |                                                                                   |
|-----|-----------------------------------------------------------------------------------|
| 1.  | Alternate: whether there is a suitable alternative restaurant nearby.             |
| 2.  | Bar: whether the restaurant has a comfortable bar area to wait in.                |
| 3.  | Fri/Sat: true on Fridays and Saturdays.                                           |
| 4.  | Hungry: whether we are hungry.                                                    |
| 5.  | Patrons: how many people are in the restaurant (values are None, Some, and Full). |
| 6.  | Price: the restaurant's price range (\$, \$\$, \$\$\$).                           |
| 7.  | Raining: whether it is raining outside.                                           |
| 8.  | Reservation: whether we made a reservation.                                       |
| 9.  | Type: the kind of restaurant (French, Italian, Thai or Burger).                   |
| 10. | WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).   |

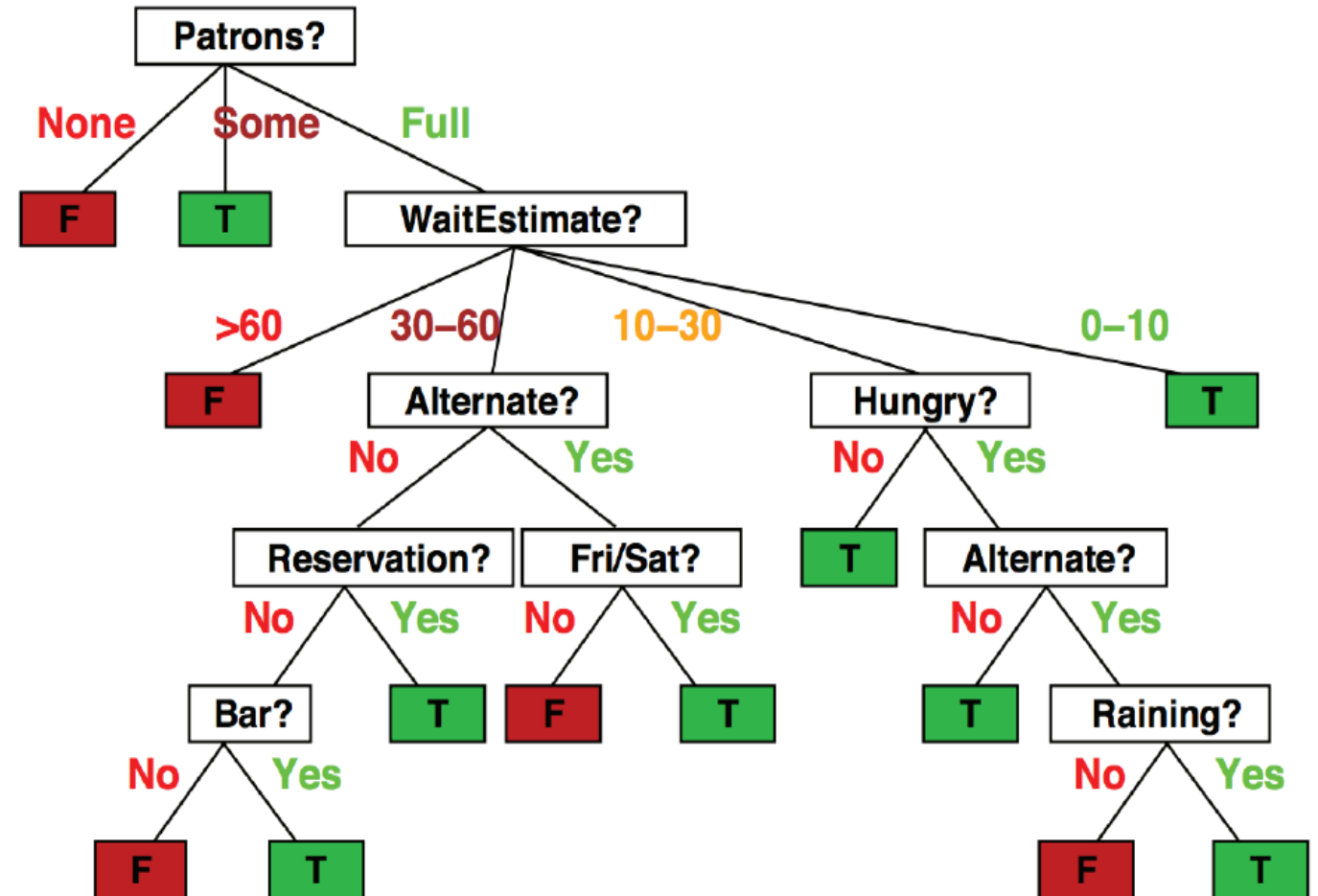
# Example with discrete inputs

- The tree to decide whether to wait (T) or not (F)

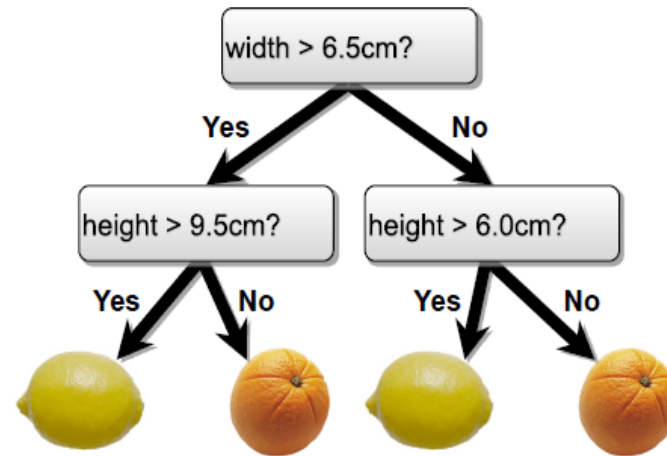
| Example  | Input Attributes |            |            |            |            |              |             |            |             |            | Goal<br><i>WillWait</i> |
|----------|------------------|------------|------------|------------|------------|--------------|-------------|------------|-------------|------------|-------------------------|
|          | <i>Alt</i>       | <i>Bar</i> | <i>Fri</i> | <i>Hun</i> | <i>Pat</i> | <i>Price</i> | <i>Rain</i> | <i>Res</i> | <i>Type</i> | <i>Est</i> |                         |
| $x_1$    | Yes              | No         | No         | Yes        | Some       | \$\$\$       | No          | Yes        | French      | 0-10       | $y_1 = \text{Yes}$      |
| $x_2$    | Yes              | No         | No         | Yes        | Full       | \$           | No          | No         | Thai        | 30-60      | $y_2 = \text{No}$       |
| $x_3$    | No               | Yes        | No         | No         | Some       | \$           | No          | No         | Burger      | 0-10       | $y_3 = \text{Yes}$      |
| $x_4$    | Yes              | No         | Yes        | Yes        | Full       | \$           | Yes         | No         | Thai        | 10-30      | $y_4 = \text{Yes}$      |
| $x_5$    | Yes              | No         | Yes        | No         | Full       | \$\$\$       | No          | Yes        | French      | >60        | $y_5 = \text{No}$       |
| $x_6$    | No               | Yes        | No         | Yes        | Some       | \$\$         | Yes         | Yes        | Italian     | 0-10       | $y_6 = \text{Yes}$      |
| $x_7$    | No               | Yes        | No         | No         | None       | \$           | Yes         | No         | Burger      | 0-10       | $y_7 = \text{No}$       |
| $x_8$    | No               | No         | No         | Yes        | Some       | \$\$         | Yes         | Yes        | Thai        | 0-10       | $y_8 = \text{Yes}$      |
| $x_9$    | No               | Yes        | Yes        | No         | Full       | \$           | Yes         | No         | Burger      | >60        | $y_9 = \text{No}$       |
| $x_{10}$ | Yes              | Yes        | Yes        | Yes        | Full       | \$\$\$       | No          | Yes        | Italian     | 10-30      | $y_{10} = \text{No}$    |
| $x_{11}$ | No               | No         | No         | No         | None       | \$           | No          | No         | Thai        | 0-10       | $y_{11} = \text{No}$    |
| $x_{12}$ | Yes              | Yes        | Yes        | Yes        | Full       | \$           | No          | No         | Burger      | 30-60      | $y_{12} = \text{Yes}$   |

Attributes:

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range (\$, \$\$, \$\$\$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai or Burger).
10. WaitEstimate: the wait estimated by the host (0-10 minutes, 10-30, 30-60, >60).



# Decision Trees



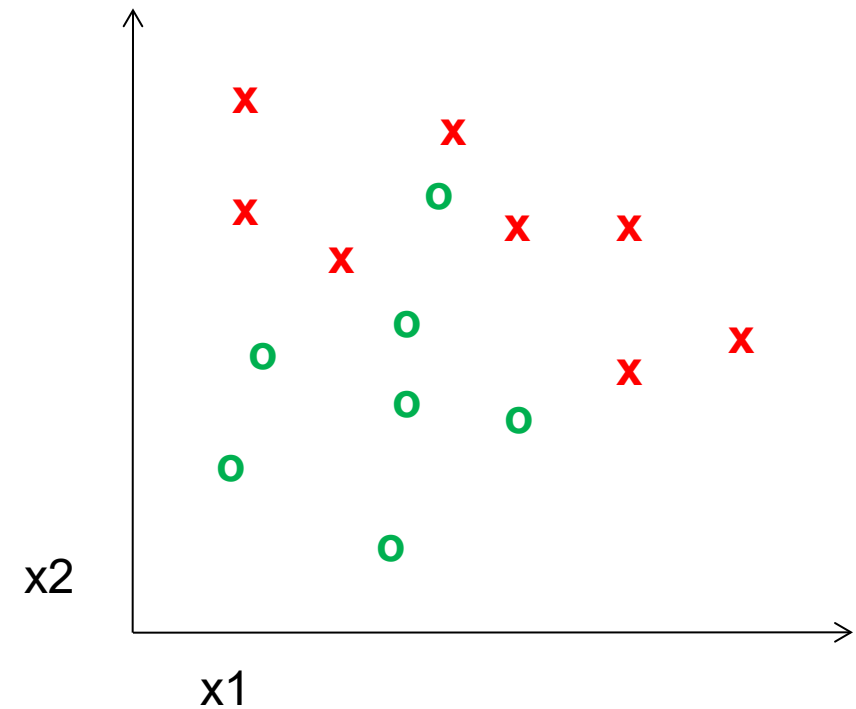
- Internal nodes **test attributes**
- Branching is determined by **attribute value**
- Leaf nodes are **outputs** (class assignments)

# Decision tree algorithm

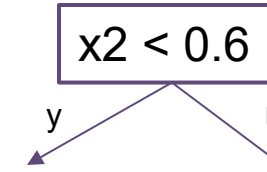
## Training

Recursively, for each node in tree:

1. If labels in the node are mixed:
  - a. Choose attribute and split values based on data that reaches each node
  - b. Branch and create 2 (or more) nodes
2. Return



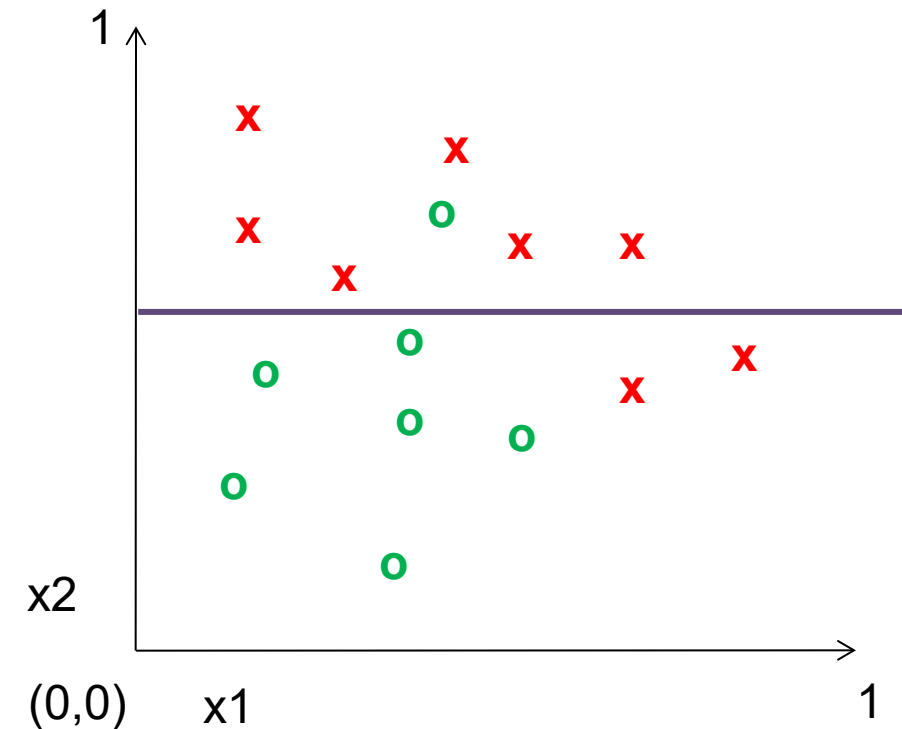
# Decision tree algorithm



## Training

Recursively, for each node in tree:

1. If labels in the node are mixed:
  - a. Choose attribute and split values based on data that reaches each node
  - b. Branch and create 2 (or more) nodes
2. Return

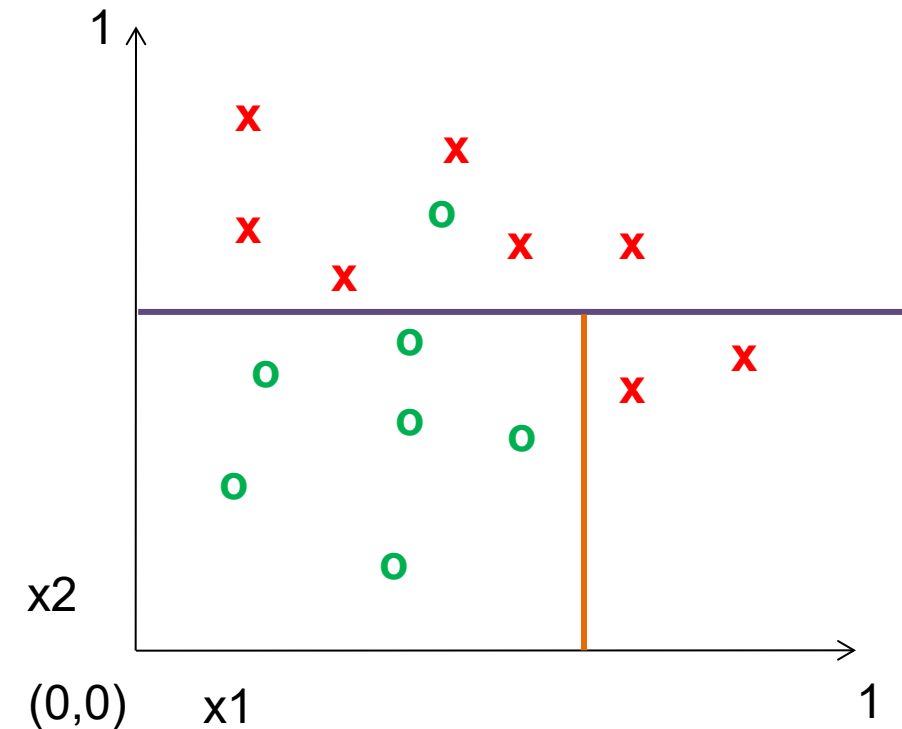
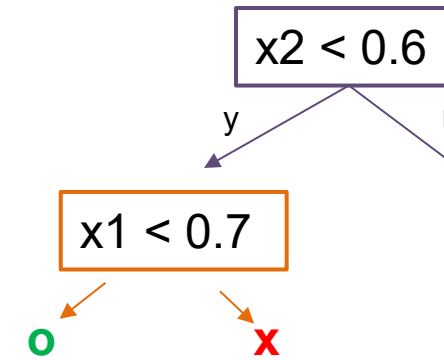


# Decision tree algorithm

## Training

Recursively, for each node in tree:

1. If labels in the node are mixed:
  - a. Choose attribute and split values based on data that reaches each node
  - b. Branch and create 2 (or more) nodes
2. Return

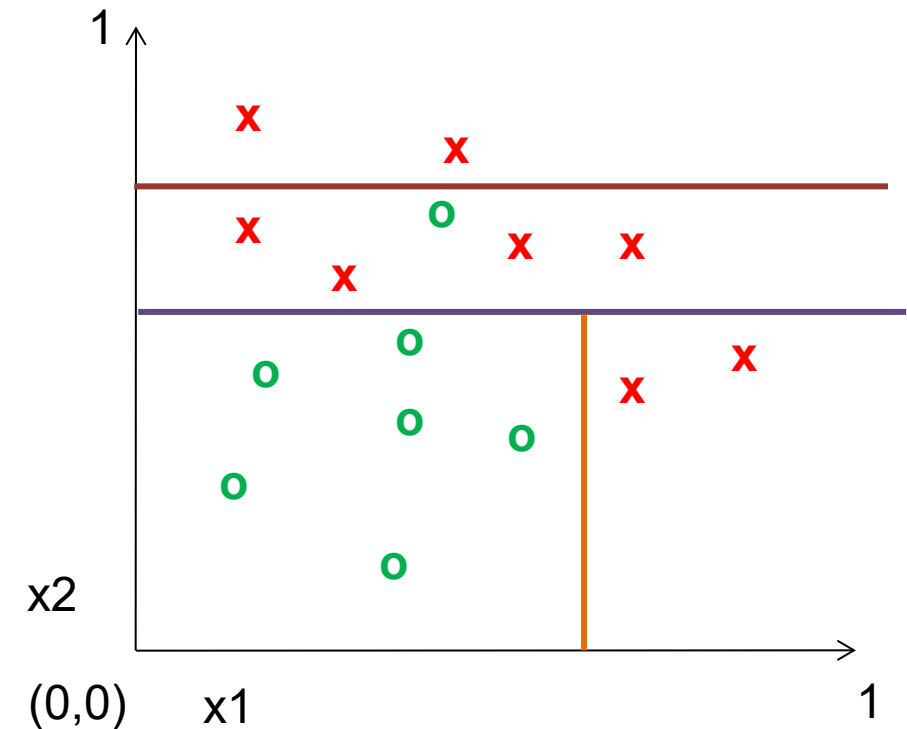
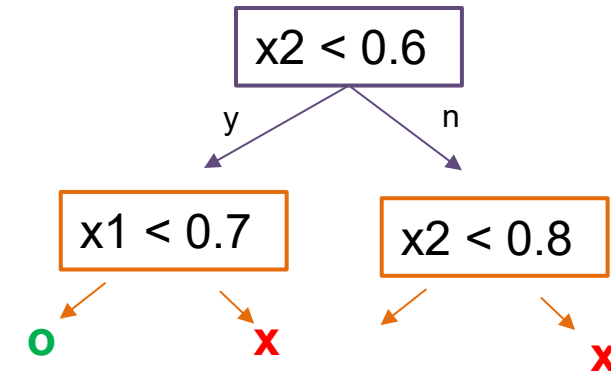


# Decision tree algorithm

## Training

Recursively, for each node in tree:

1. If labels in the node are mixed:
  - a. Choose attribute and split values based on data that reaches each node
  - b. Branch and create 2 (or more) nodes
2. Return

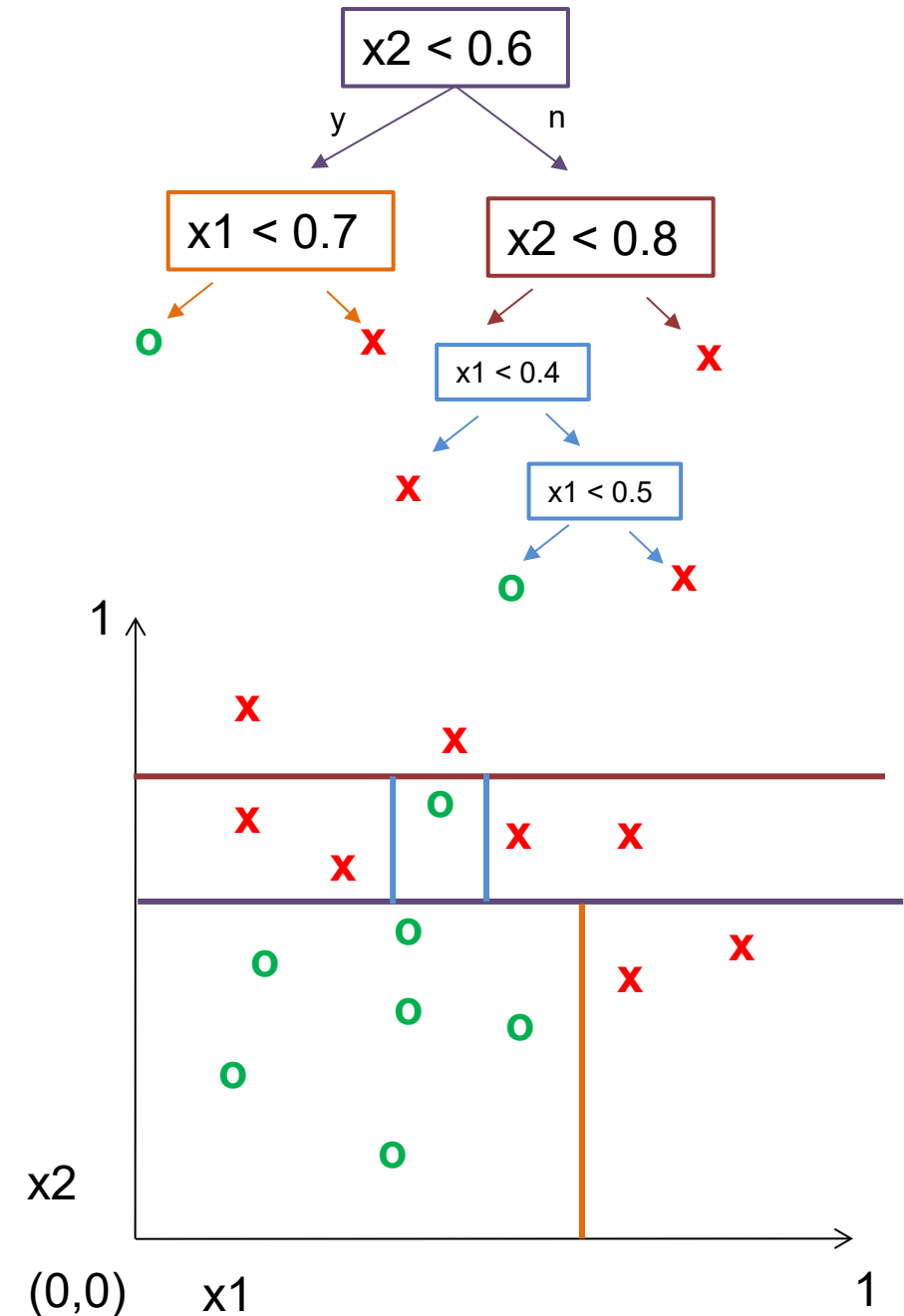


# Decision tree algorithm

## Training

Recursively, for each node in tree:

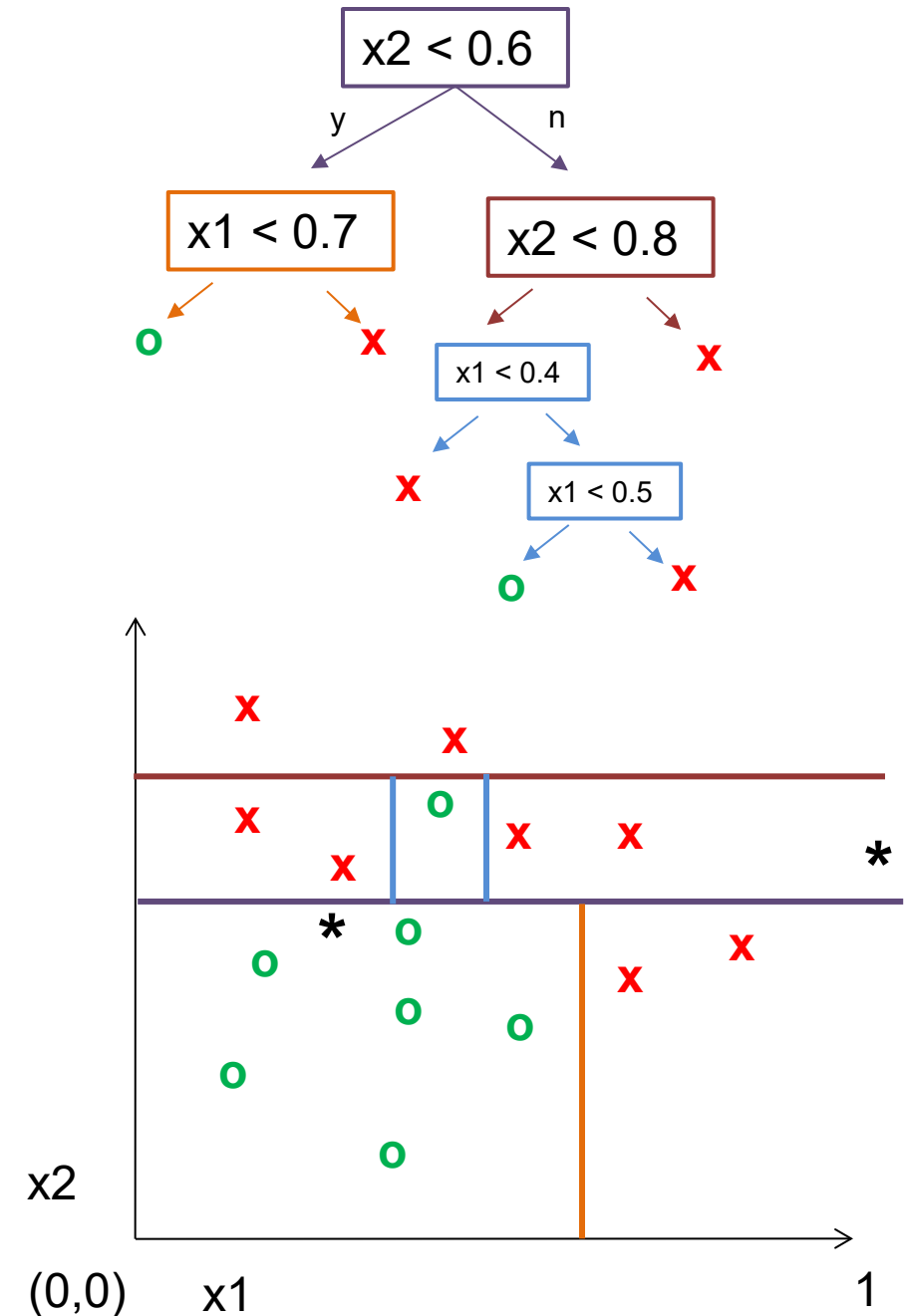
1. If labels in the node are mixed:
  - a. Choose attribute and split values based on data that reaches each node
  - b. Branch and create 2 (or more) nodes
2. Return



# Decision tree algorithm

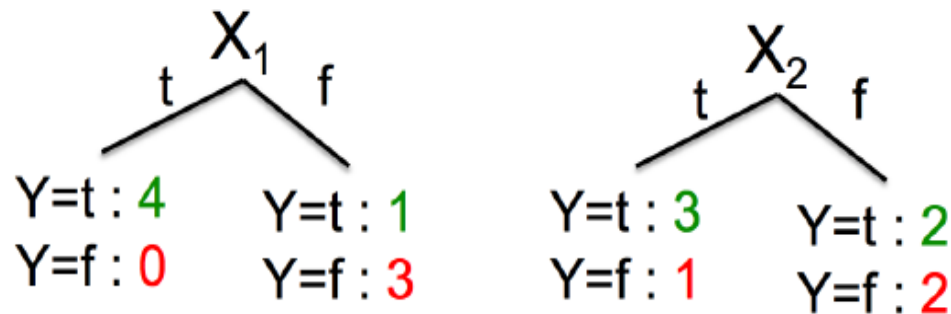
## Prediction

1. Check conditions to descend tree
2. Return label of leaf node



# How do you choose what/where to split?

- Which attribute is better to split on,  $X_1$  or  $X_2$ ?



| $X_1$ | $X_2$ | Y |
|-------|-------|---|
| T     | T     | T |
| T     | F     | T |
| T     | T     | T |
| T     | F     | T |
| F     | T     | T |
| F     | F     | F |
| F     | T     | F |
| F     | F     | F |

**Idea:** Use counts at leaves to define probability distributions, so we can measure uncertainty

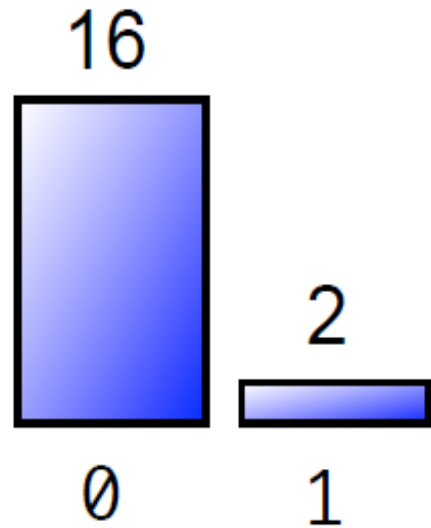
# Quantifying Uncertainty: Coin Flip Example

Sequence 1:

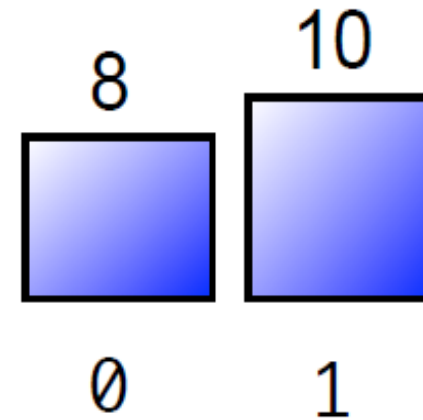
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 ... ?

Sequence 2:

0 1 0 1 0 1 1 1 0 1 0 0 1 1 0 1 0 1 ... ?



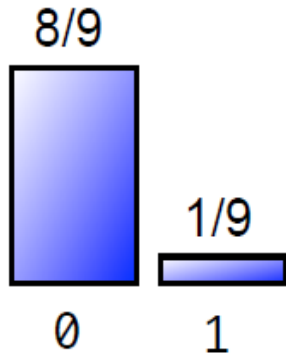
versus



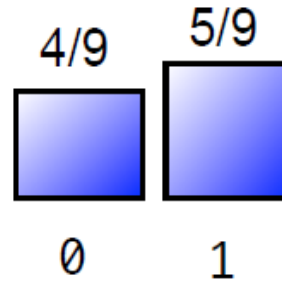
# Quantifying Uncertainty: Coin Flip Example

Entropy  $H$ :

$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x)$$



$$-\frac{8}{9} \log_2 \frac{8}{9} - \frac{1}{9} \log_2 \frac{1}{9} \approx \frac{1}{2}$$



$$-\frac{4}{9} \log_2 \frac{4}{9} - \frac{5}{9} \log_2 \frac{5}{9} \approx 0.99$$

**Why does entropy have that equation?**

- $p(x)$  is the probability that  $X$  has value  $x$
- $-\log_2 p(x)$  is the number of binary search steps needed to identify value  $x$
- Entropy is expectation of the number of binary search steps (i.e. bits) needed to identify the value of  $X$

- How surprised are we by a new value in the sequence?
- How much information does it convey?

# Entropy Explanation

Q: Why is entropy  $H(X) = -\sum_x p(x)\log_2 p(x)$  ?

A: Entropy = average number of bits needed to encode X

E.g.

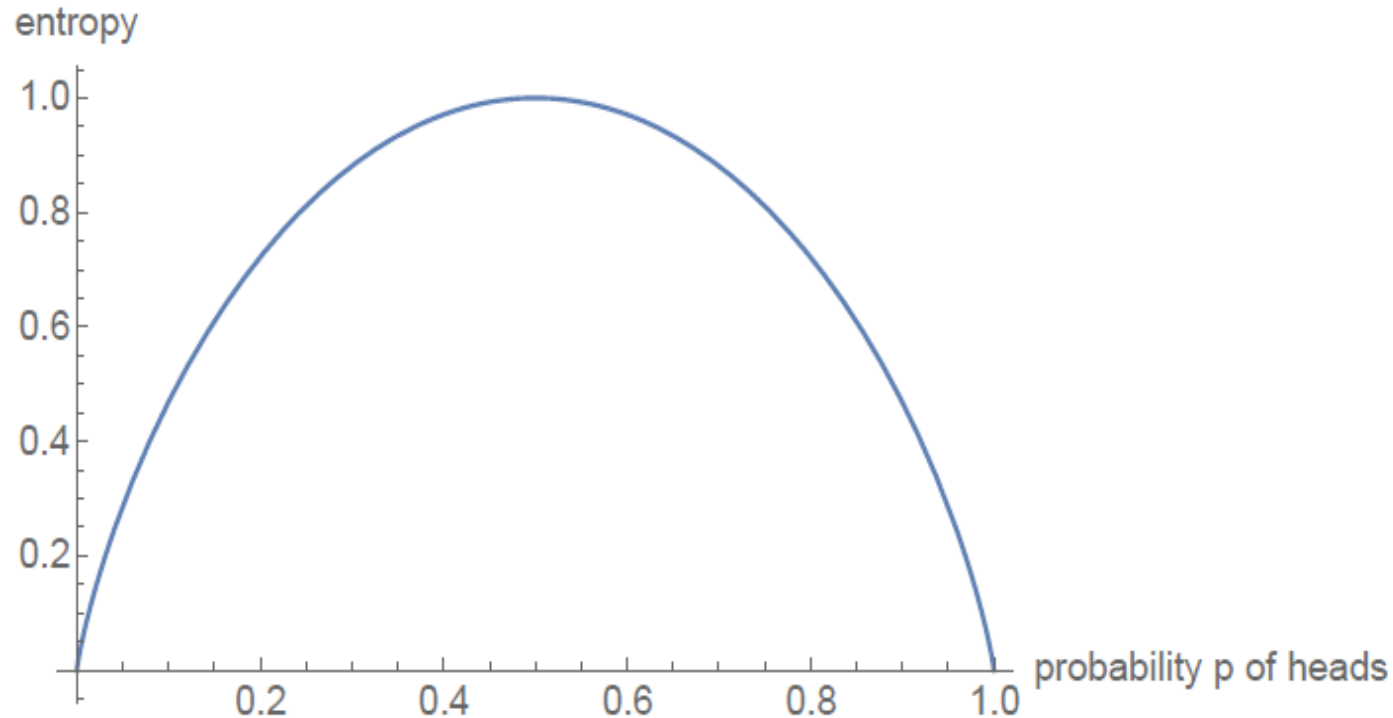
|           |     |     |    |     |
|-----------|-----|-----|----|-----|
| X=        | 1   | 2   | 3  | 4   |
| P(X)      | 50% | 25% | 0% | 25% |
| Encoding  | 0   | 1-0 |    | 1-1 |
| Bits Used | 1   | 2   | 0  | 2   |

Expected bits =  $0.5 * 1 + 0.25 * 2 + 0 * 0 + 0.25 * 2 = 1.5$

Entropy =  $-0.5 * \log_2 0.5 - 0.25 * \log_2 0.25 - 0 - 0.25 * \log_2 0.25 = 1.5$

# Quantifying Uncertainty: Coin Flip Example

$$\text{Entropy: } H(X) = - \sum_{x \in X} p(x) \log_2 p(x)$$



# Entropy of a Joint Distribution

- Example:  $X = \{\text{Raining}, \text{Not raining}\}$ ,  $Y = \{\text{Cloudy}, \text{Not cloudy}\}$

|             | Cloudy | Not Cloudy |
|-------------|--------|------------|
| Raining     | 24/100 | 1/100      |
| Not Raining | 25/100 | 50/100     |

$$\begin{aligned} H(X, Y) &= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(x, y) \\ &= -\frac{24}{100} \log_2 \frac{24}{100} - \frac{1}{100} \log_2 \frac{1}{100} - \frac{25}{100} \log_2 \frac{25}{100} - \frac{50}{100} \log_2 \frac{50}{100} \\ &\approx 1.56 \text{ bits} \end{aligned}$$

# Specific Conditional Entropy

$$Y = \{\text{Cloudy}, \text{Not cloudy}\}$$

$$X = \{\text{Raining}, \text{Not raining}\}$$

|             | Cloudy | Not Cloudy |
|-------------|--------|------------|
| Raining     | 24/100 | 1/100      |
| Not Raining | 25/100 | 50/100     |

What is the entropy of cloudiness  $Y$ , **given that it is raining**?

$$\begin{aligned} H(Y|X = x) &= - \sum_{y \in Y} p(y|x) \log_2 p(y|x) \\ &= -\frac{24}{25} \log_2 \frac{24}{25} - \frac{1}{25} \log_2 \frac{1}{25} \\ &\approx 0.24 \text{ bits} \end{aligned}$$

We used:  $p(y|x) = \frac{p(x,y)}{p(x)}$ , and  $p(x) = \sum_y p(x,y)$  (sum in a row)

# Conditional Entropy

|             | Cloudy | Not Cloudy |
|-------------|--------|------------|
| Raining     | 24/100 | 1/100      |
| Not Raining | 25/100 | 50/100     |

- The expected conditional entropy:

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= - \sum_{x \in X} \sum_{y \in Y} p(x, y) \log_2 p(y|x) \end{aligned}$$

# Conditional Entropy

$Y = \{\text{Cloudy}, \text{Not cloudy}\}$

$X = \{\text{Raining}, \text{Not raining}\}$

|             | Cloudy | Not Cloudy |
|-------------|--------|------------|
| Raining     | 24/100 | 1/100      |
| Not Raining | 25/100 | 50/100     |

- What is the entropy of cloudiness, given the knowledge of whether or not it is raining?

$$\begin{aligned} H(Y|X) &= \sum_{x \in X} p(x) H(Y|X = x) \\ &= \frac{1}{4} H(\text{cloudiness} | \text{is raining}) + \frac{3}{4} H(\text{cloudiness} | \text{not raining}) \\ &\quad \text{0.242} \qquad \qquad \text{0.918} \\ &\approx 0.749 \text{ bits} \end{aligned}$$

# Come useful properties of conditional entropy

- $H$  is always non-negative
- Chain rule:  $H(X, Y) = H(X|Y) + H(Y) = H(Y|X) + H(X)$
- If  $X$  and  $Y$  independent, then  $X$  doesn't tell us anything about  $Y$ :

$$H(Y|X) = H(Y)$$

- But  $Y$  tells us everything about  $Y$ :  $H(Y|Y) = 0$
- By knowing  $X$ , we can only decrease uncertainty about  $Y$ :

$$H(Y|X) \leq H(Y)$$

# Information Gain

|     |             | $Y$    |            |
|-----|-------------|--------|------------|
|     |             | Cloudy | Not Cloudy |
| $X$ | Raining     | 24/100 | 1/100      |
|     | Not Raining | 25/100 | 50/100     |

How much information about cloudiness do we get by discovering whether it is raining?

$$\begin{aligned} IG(Y|X) &= H(Y) - H(Y|X) \\ &\approx 1.0 - 0.75 \\ &\approx 0.25 \text{ bits} \end{aligned}$$

Also called **information gain** in  $Y$  due to  $X$

- If  $X$  is completely uninformative about  $Y$ :  $IG(Y|X) = 0$
- If  $X$  is completely informative about  $Y$ :  $IG(Y|X) = H(Y)$

How can we use this to construct our decision tree?

# Terminology Recap

- Entropy,  $H(X)$ : measures uncertainty of  $X$
- Specific conditional entropy,  $H(X|Y = y)$ : measures uncertainty of  $X$  if  $Y$  is known to have a particular value
- Conditional entropy  $H(X|Y)$ : measures expected uncertainty of  $X$  if I know  $Y$
- Information gain  $I(X|Y)$ : measures how much knowing  $Y$  would reduce my uncertainty in  $X$

$$H(X) = - \sum_x P(X = x) \log_2 P(X = x)$$

$$H(X|Y = y) = - \sum_x P(X = x|Y = y) \log_2 P(X = x|Y = y)$$

$$H(X|Y) = \sum_y H(X|Y = y) P(Y = y)$$

$$I(X|Y) = H(X) - H(X|Y)$$

# Constructing decision tree

## Training

Recursively, for each node in tree:

1. If labels in the node are mixed:

- a. Choose attribute and split values based on data that reaches each node
- b. Branch and create 2 (or more) nodes



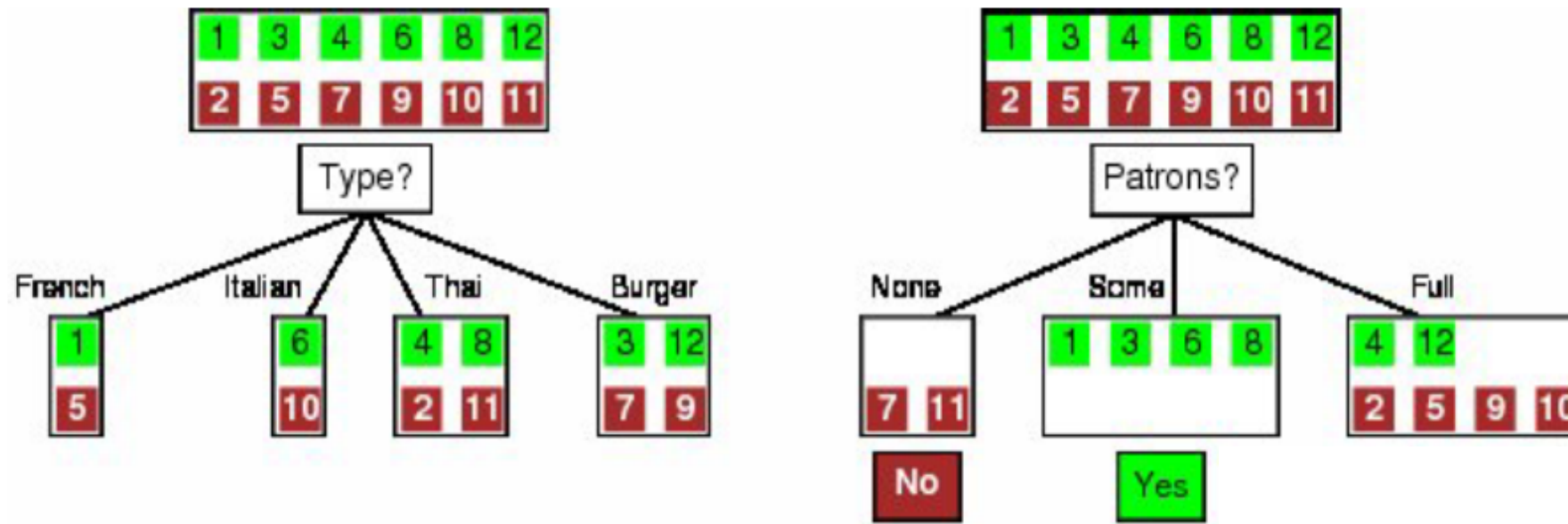
2. Return

1. Measure information gain
  - For each discrete attribute: compute information gain of split
  - For each continuous attribute: select most informative threshold and compute its information gain. Can be done efficiently based on sorted values.
2. Select attribute / threshold with highest information gain

Q1-Q3

<https://tinyurl.com/441AML-L12>





Q3 answer explained

$$IG(Y) = H(Y) - H(Y|X)$$

$$IG(type) = 1 - \left[ \frac{2}{12}H(Y|Fr.) + \frac{2}{12}H(Y|It.) + \frac{4}{12}H(Y|Thai) + \frac{4}{12}H(Y|Bur.) \right] = 0$$

$$IG(Patrons) = 1 - \left[ \frac{2}{12}H(0, 1) + \frac{4}{12}H(1, 0) + \frac{6}{12}H\left(\frac{2}{6}, \frac{4}{6}\right) \right] \approx 0.541$$

# What if you need to predict a continuous value?

- Regression Tree

- Same idea, but choose splits to minimize sum squared error

$$\sum_{n \in \text{node}} (f_{\text{node}}(x_n) - y_n)^2$$

- $f_{\text{node}}(x_n)$  typically returns the mean prediction value of data points in the leaf node containing  $x_n$

- What are we minimizing?

Q4

<https://tinyurl.com/441AML-L12>



# What does a regression tree minimize?

Conditional variance  $\text{Var}(y|x)$  according to clustering given by tree:

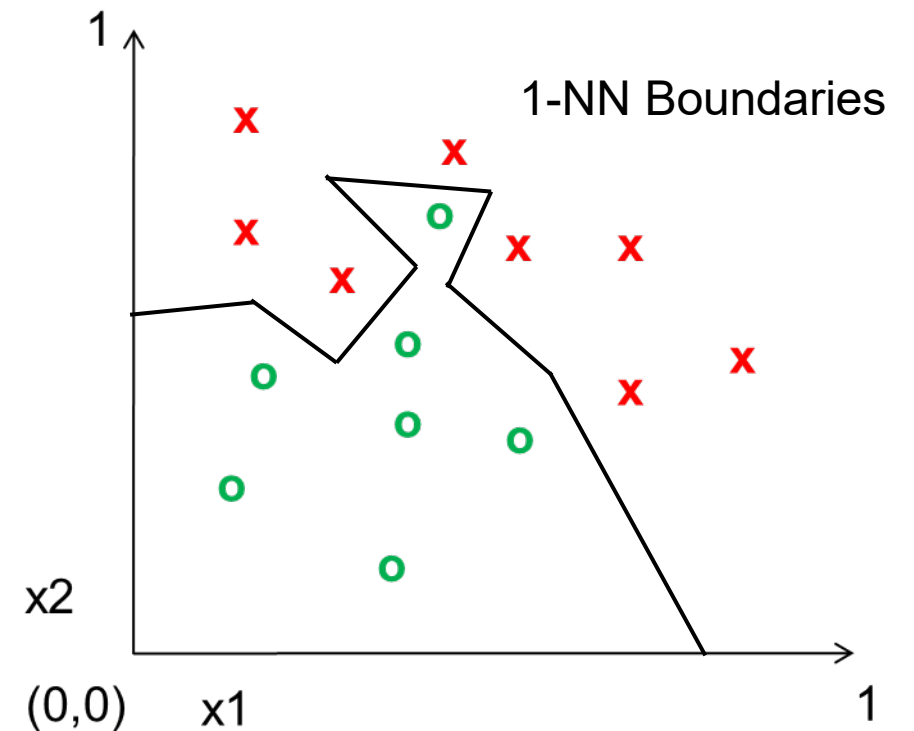
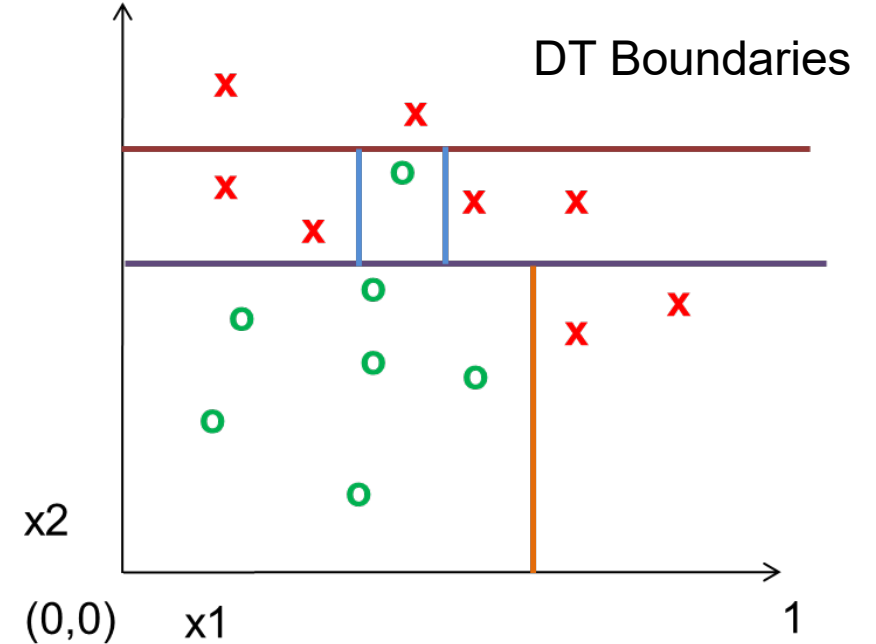
$$\sum_n (f(x_n) - y_n)^2 \text{ where } f(x_n) \text{ is mean of } y_i \text{'s with corresponding } x_i \text{ in the same node as } x_n$$

# Variants

- Different splitting criteria, e.g. Gini index:  $1 - \sum_i p_i^2$  (very similar result, a little faster to compute)
- Most commonly, split on one attribute at a time
  - In case of continuous vector data, can also split on linear projections of features
- Can stop early
  - when leaf node contains fewer than  $N_{\min}$  points
  - when max tree depth is reached
- Can also predict multiple continuous values or multiple classes

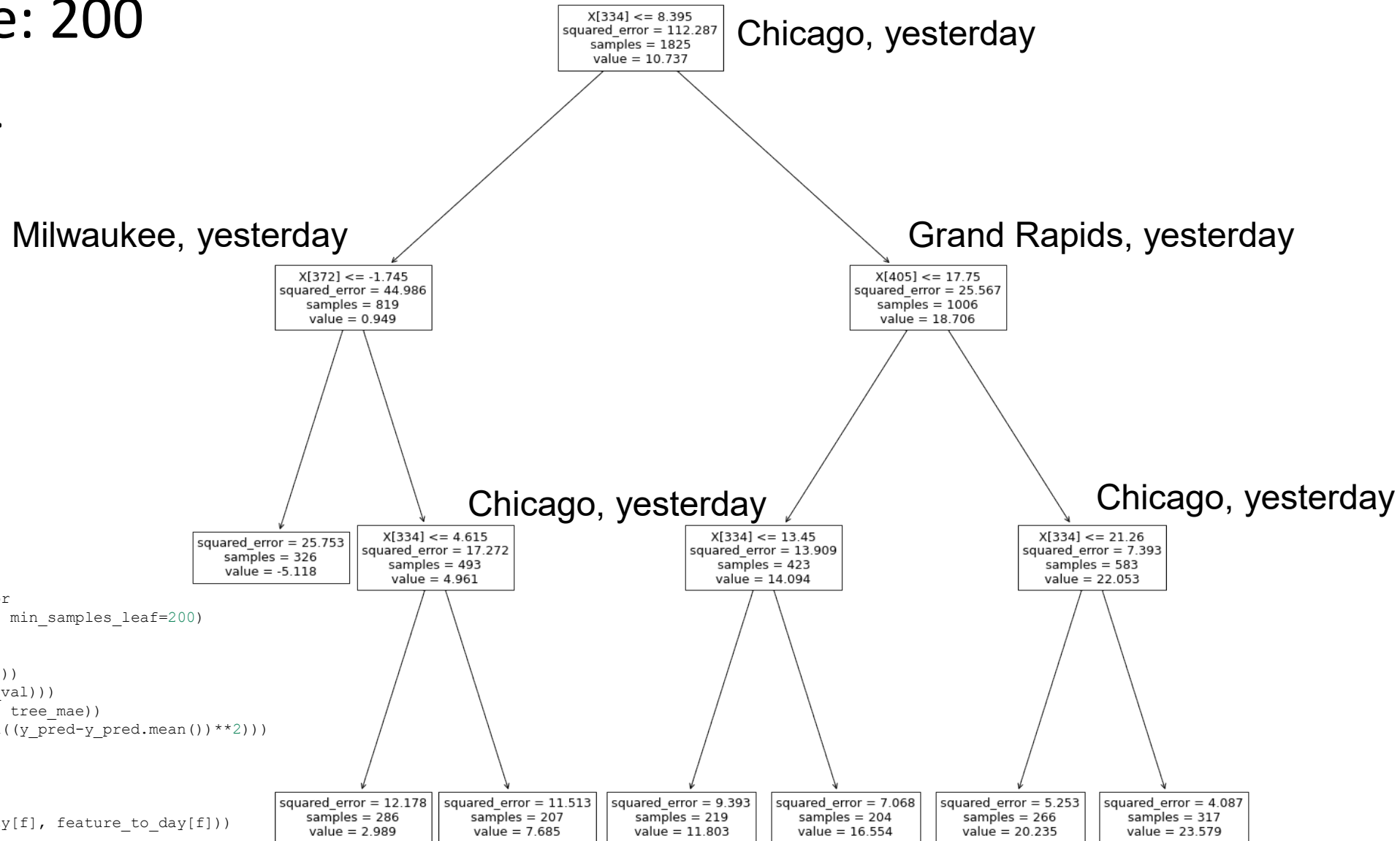
# Decision Tree vs. 1-NN

- Both have piecewise-linear decisions
- Decision tree is typically “axis-aligned”
- Decision tree has ability for early stopping to improve generalization
- True power of decision trees arrives with ensembles (lots of small or randomized trees)



# Regression Tree for Temperature Prediction

- Min leaf size: 200
- RMSE= 3.42
- $R^2=0.88$



```
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor(random_state=0, min_samples_leaf=200)
model.fit(x_train, y_train)
y_pred = model.predict(x_val)
tree_rmse = np.sqrt(np.mean((y_pred-y_val)**2))
tree_mae = np.sqrt(np.median(np.abs(y_pred-y_val)))
print('LR: RMSE={}, MAE={}'.format(tree_rmse, tree_mae))
print('R^2: {}'.format(1-tree_rmse**2/np.mean((y_pred-y_pred.mean())**2)))
plt.figure(figsize=(20,20))
tree.plot_tree(model)
plt.show()
for f in [334, 372, 405]:
    print('{}: {}, {}'.format(f, feature_to_city[f], feature_to_day[f]))
```

```
from sklearn import tree
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor(random_state=0, min_samples_leaf=200)
model.fit(x_train, y_train)
y_pred = model.predict(x_val)
tree_rmse = np.sqrt(np.mean((y_pred-y_val)**2))
tree_mae = np.sqrt(np.median(np.abs(y_pred-y_val)))
print('LR: RMSE={}, MAE={}'.format(tree_rmse, tree_mae))
print('R^2: {}'.format(1-tree_rmse**2/np.mean((y_pred-
y_pred.mean())**2)))
plt.figure(figsize=(20,20))
tree.plot_tree(model)
plt.show()
for f in [334, 372, 405]:
    print('{}: {}, {}'.format(f, feature_to_city[f], feature_to_day[f]
    ))
```

# Classification/Regression Trees Summary

- Key Assumptions
  - Samples with similar features have similar predictions
- Model Parameters
  - Tree structure with split criteria at each internal node and prediction at each leaf node
- Designs
  - Limits on tree growth
  - What kinds of splits are considered
  - Criterion for choosing attribute/split (e.g. gini impurity score is another common choice)
- When to Use
  - Want an explainable decision function (e.g. for medical diagnosis)
  - As part of an ensemble (as we'll see Thursday)
- When Not to Use
  - One tree is not a great performer, but a forest is

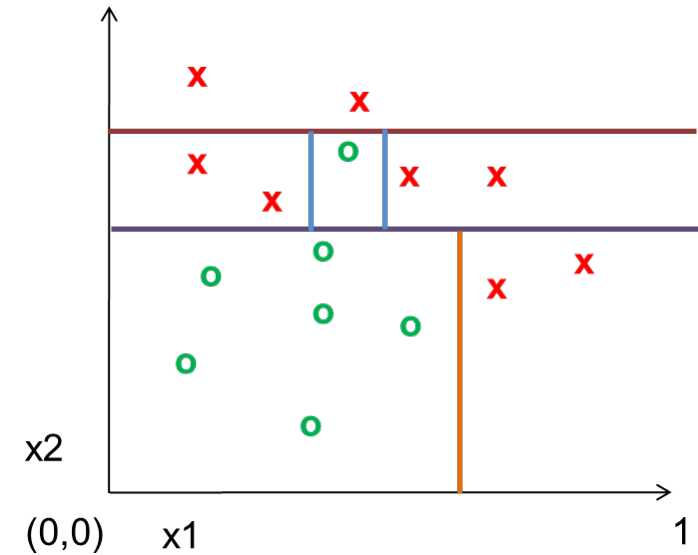
Q5-Q8

<https://tinyurl.com/441AML-L12>



# Things to remember

- Decision/regression trees learn to split up the feature space into partitions with similar values
- Entropy is a measure of uncertainty
- Information gain measures how much particular knowledge reduces prediction uncertainty



$$H(X) = - \sum_{x \in X} p(x) \log_2 p(x)$$

$$IG(Y|X) = H(Y) - H(Y|X)$$

# Thursday

- Ensembles: model averaging and forests