

Machine Learning with Decision Trees

ECS 170

University of California, Davis

Gabriel Simmons

(adapted from slides by Michael Livanos)

Approaches to Intelligence

Intelligence as *search through a state space*

We're intelligent if we can navigate to goal states efficiently

Intelligence as *game-playing*

We're intelligent if we can navigate to goal states efficiently...

...in adversarial environments

Approaches to Intelligence

Intelligence as...

Function Approximation:

We're intelligent if we can replicate a data-generating process, based on observations of its outputs

Discovering Structure:

We're intelligent if we can find a “simple explanation” for seemingly-complex phenomena

Prediction:

We're intelligent if we can say something true beyond what's obvious from direct observation

Approaches to Intelligence

Intelligence as...

Function Approximation:

We're intelligent if we can replicate a data-generating process, based on observations of its outputs

Discovering Structure:

We're intelligent if we can find a “simple explanation” for seemingly-complex phenomena

Prediction:

We're intelligent if we can say something true beyond what's obvious from direct observation

Machine Learning

Goal: develop a model of some process based on data

Machine Learning

Goal: develop a model of some process based on observations of the process

observations == data

Types of Machine Learning

ML

Supervised ML

have labels

Unsupervised ML

no labels

Reinforcement Learning

have a
function that
“labels”
world states

Types of Machine Learning

ML

**Supervised
ML**

have labels

**Unsupervised
ML**

no labels

**Reinforcement
Learning**

have a
function that
“labels”
world states

Fast Forward



it's the afternoon of your graduation

ECS 170 is just a faint memory



your friends and family have waited for *three hours* to see you walk



(you forgot to shake the Chancellor's hand, but it's ok)

now your friends and
family are hungry...

Fast Forward

you take everyone to your favorite restaurant in Davis

... but there's a line

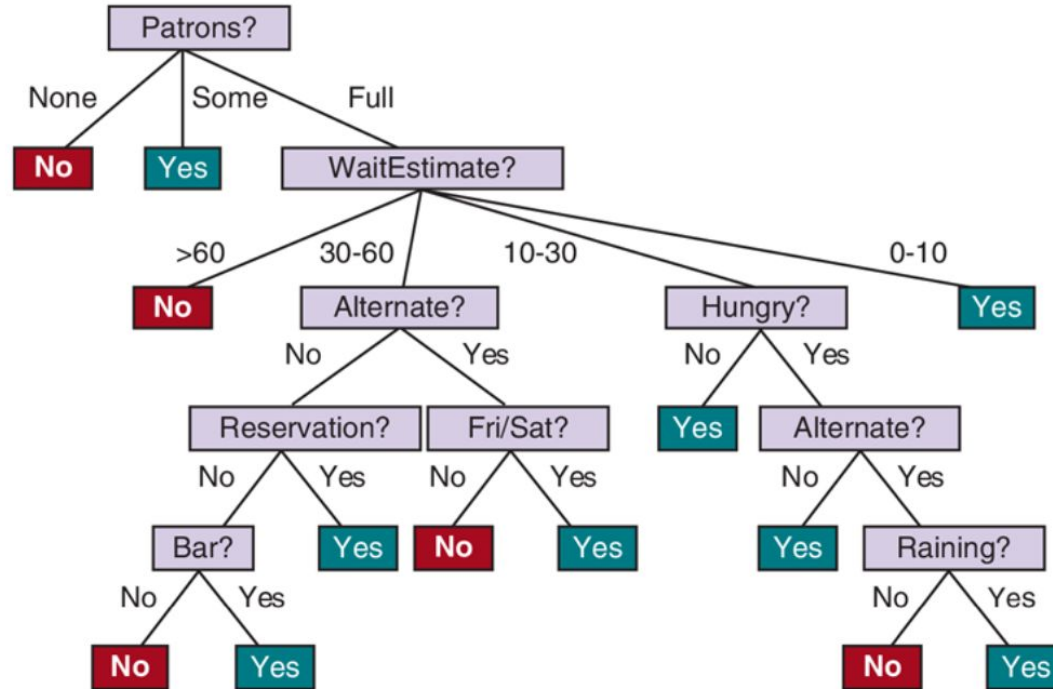
... do you wait?



... do you wait?

... do you wait?

Figure 19.3



A decision tree for deciding whether to wait for a table.

The Decision Tree

A Simple Supervised Machine Learning Model

map a vector of attribute values (inputs) to a single value (output)

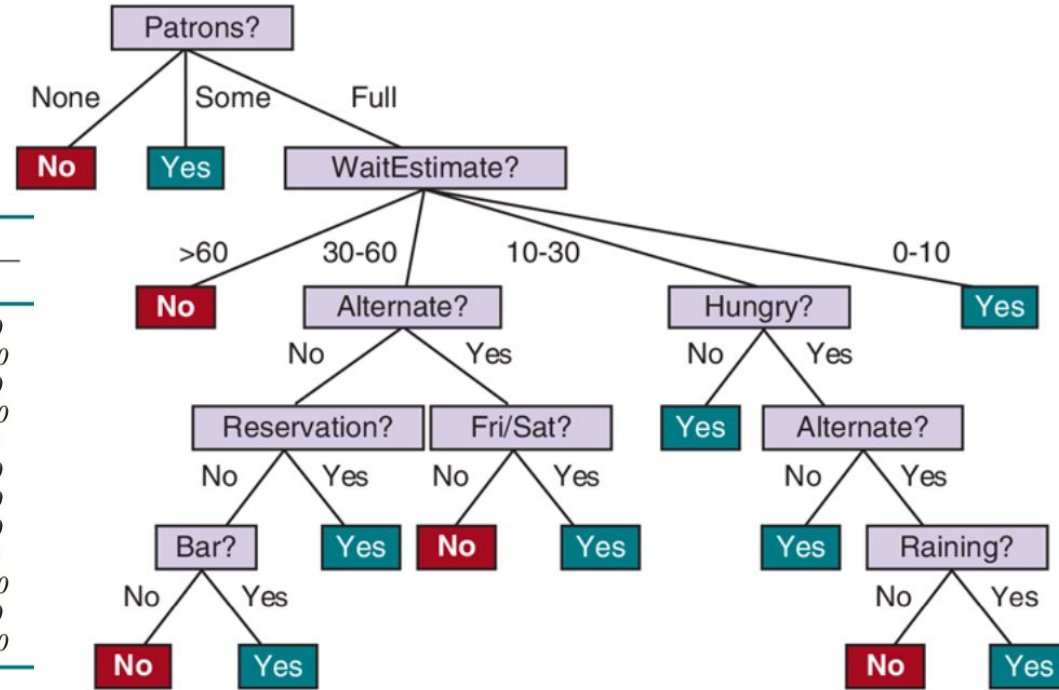
“attribute values” == “features”

Example	Input Attributes										Output
	<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	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	$y_1 = \text{Yes}$
x_2	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	$y_2 = \text{No}$
x_3	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	$y_3 = \text{Yes}$
x_4	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	$y_4 = \text{Yes}$
x_5	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	$y_5 = \text{No}$
x_6	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	$y_6 = \text{Yes}$
x_7	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	$y_7 = \text{No}$
x_8	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	$y_8 = \text{Yes}$
x_9	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	$y_9 = \text{No}$
x_{10}	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	$y_{10} = \text{No}$
x_{11}	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	$y_{11} = \text{No}$
x_{12}	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	$y_{12} = \text{Yes}$

Figure 19.3

Output class x1? x2?

Example	Input Attributes									
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est
x ₁	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10
x ₂	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60
x ₃	No	Yes	No	No	Some	\$	No	No	Burger	0-10
x ₄	Yes	No	Yes	Yes	Full	\$	Yes	No	Thai	10-30
x ₅	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60
x ₆	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10
x ₇	No	Yes	No	No	None	\$	Yes	No	Burger	0-10
x ₈	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10
x ₉	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60
x ₁₀	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30
x ₁₁	No	No	No	No	None	\$	No	No	Thai	0-10
x ₁₂	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60



A decision tree for deciding whether to wait for a table.

Example	Input Attributes										Output
	<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>
x₁	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	<i>y₁ = Yes</i>
x₂	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	<i>y₂ = No</i>
x₃	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y₃ = Yes</i>
x₄	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	<i>y₄ = Yes</i>
x₅	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	<i>y₅ = No</i>
x₆	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	<i>y₆ = Yes</i>
x₇	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	<i>y₇ = No</i>
x₈	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	<i>y₈ = Yes</i>
x₉	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	<i>y₉ = No</i>
x₁₀	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	<i>y₁₀ = No</i>
x₁₁	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	<i>y₁₁ = No</i>
x₁₂	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	<i>y₁₂ = Yes</i>

Machine Learning

Goal: develop a model of some process based on observations of the process

Can we model whether potential customers will wait for a seat or not?

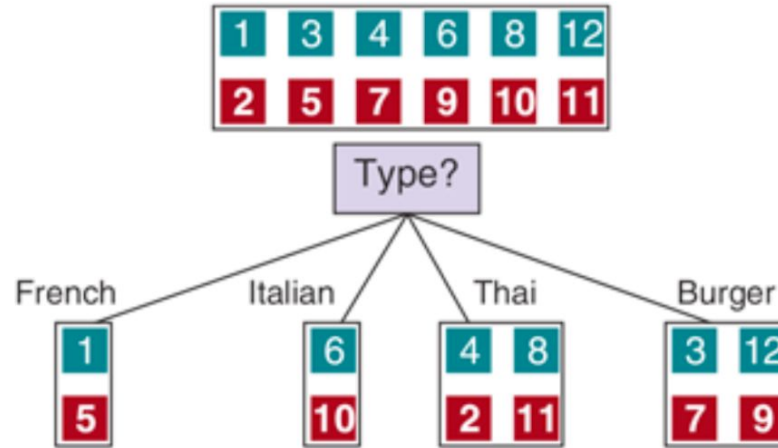
Decision Tree: Core Idea

Select an “informative” feature

Split the data based on feature values

Repeat until the data is separated by output label

What feature should we use?



After splitting on **Type**, our best guess is still 50/50

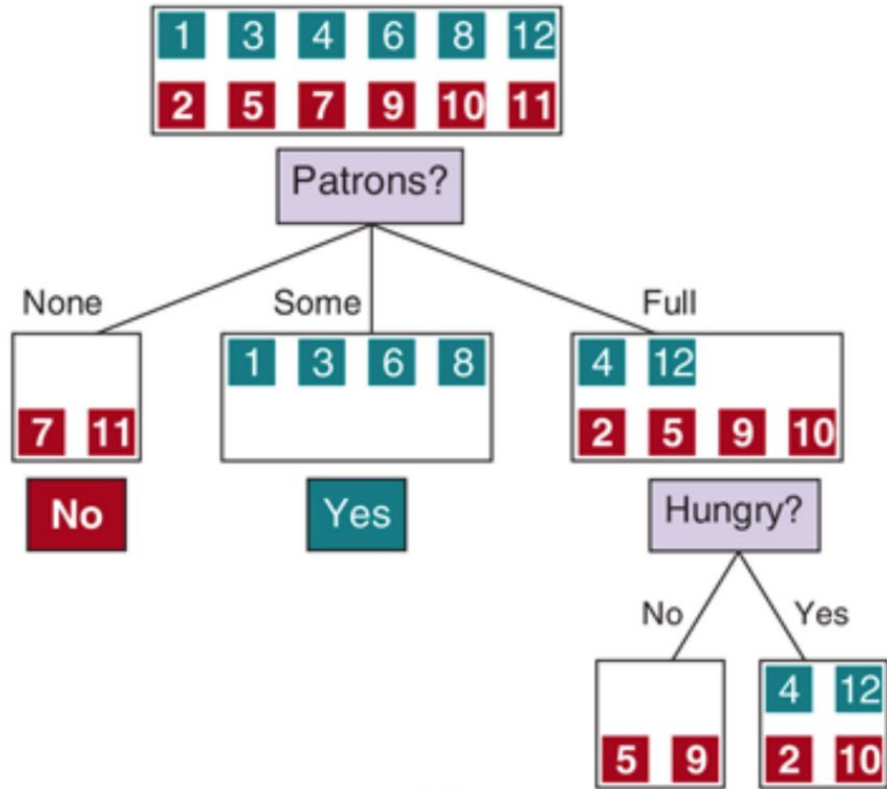
What feature should we use?

If we split on **Patrons**, we can be more confident

When there are some patrons, people always wait

When there are no patrons, people never wait

Informative features partition the data into homogeneous subsets



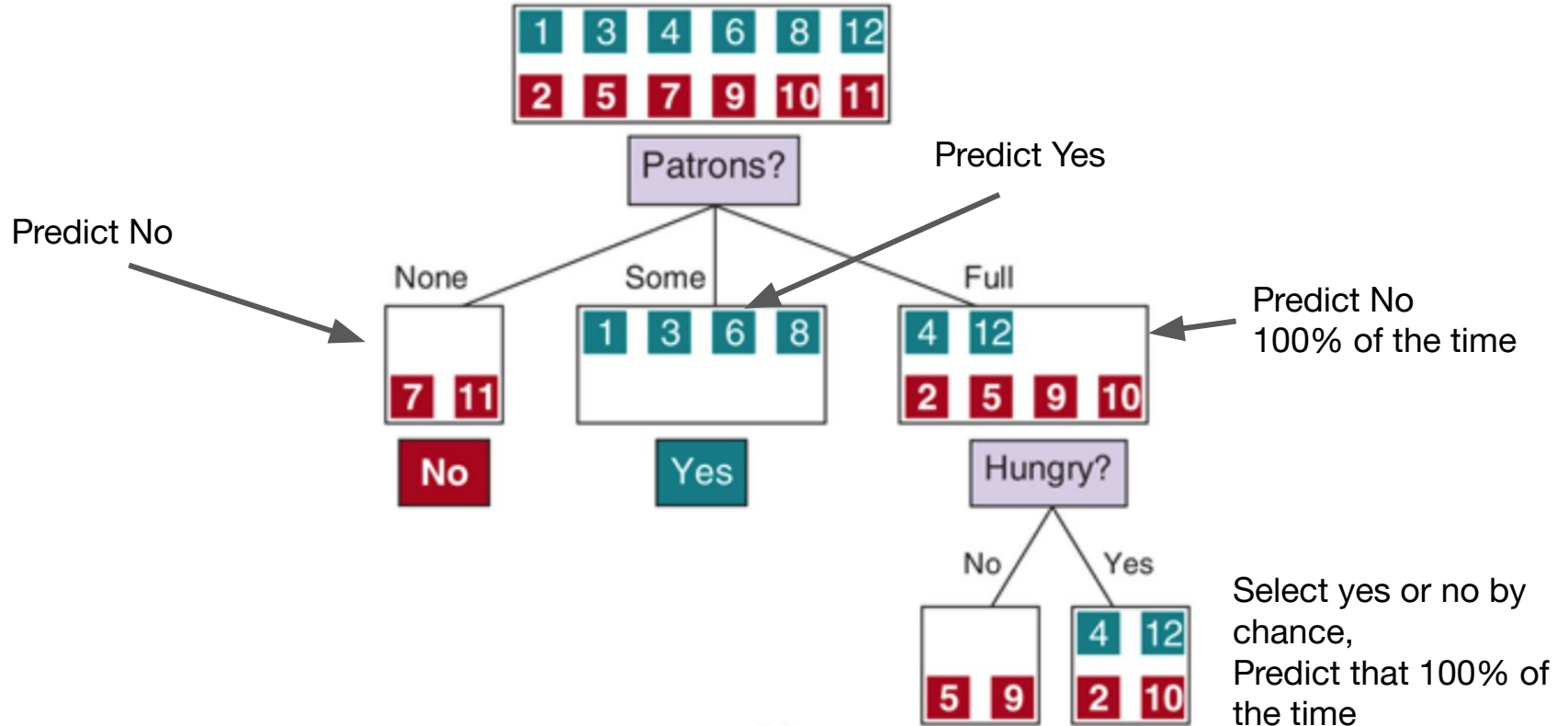
Decision Tree Algorithm

```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree

if examples is empty then return PLURALITY-VALUE(parent_examples)
else if all examples have the same classification then return the classification
else if attributes is empty then return PLURALITY-VALUE(examples)
else
     $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$ 
    tree  $\leftarrow$  a new decision tree with root test A
    for each value v of A do
        exs  $\leftarrow \{e : e \in \text{examples} \text{ and } e.A = v\}$ 
        subtree  $\leftarrow$  LEARN-DECISION-TREE(exs, attributes – A, examples)
        add a branch to tree with label (A = v) and subtree subtree
    return tree
```

The decision tree learning algorithm. The function IMPORTANCE is described in [Section 19.3.3](#). The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

What to do with leaf nodes?



Decision Tree Algorithm

```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree  
  
  if examples is empty then return PLURALITY-VALUE(parent_examples)  
  else if all examples have the same classification then return the classification  
  else if attributes is empty then return PLURALITY-VALUE(examples)  
  else  
     $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$   
    tree  $\leftarrow$  a new decision tree with root test A  
    for each value v of A do  
       $\text{exs} \leftarrow \{e : e \in \text{examples} \text{ and } e.A = v\}$   
      subtree  $\leftarrow$  LEARN-DECISION-TREE(exs, attributes – A, examples)  
      add a branch to tree with label (A = v) and subtree subtree  
  return tree
```

The decision tree learning algorithm. The function IMPORTANCE is described in [Section 19.3.3](#). The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

Determining Feature Importance

Entropy:
$$H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = - \sum_k P(v_k) \log_2 P(v_k).$$

entropy is a measure of uncertainty about a random variable v_k

more certainty == less entropy

more informative features help us become more certain (decrease entropy)

Determining Feature Importance

It will help to define $B(q)$ as the entropy of a Boolean random variable that is true with probability q :

$$B(q) = -(q \log_2 q + (1 - q) \log_2 (1 - q)).$$

$$H(\textit{Output}) = B\left(\frac{p}{p + n}\right)$$

Determining Feature Importance

An attribute A with d distinct values divides the training set E into subsets E_1, \dots, E_d . Each subset E_k has p_k positive examples and n_k negative examples, so if we go along that branch, we will need an additional $B(p_k / (p_k + n_k))$ bits of information to answer the question. A randomly chosen example from the training set has the k th value for the attribute (i.e., is in E_k with probability $(p_k + n_k) / (p + n)$), so the expected entropy remaining after testing attribute A is

$$\text{Remainder}(A) = \sum_{k=1}^d \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right).$$

Determining Feature Importance

An attribute A with d distinct values divides the training set E into subsets E_1, \dots, E_d . Each subset E_k has p_k positive examples and n_k negative examples, so if we go along that branch, we will need an additional $B(p_k / (p_k + n_k))$ bits of information to answer the question. A randomly chosen example from the training set has the k th value for the attribute (i.e., is in E_k with probability $(p_k + n_k) / (p + n)$), so the expected entropy remaining after testing attribute A is

$$\text{Remainder}(A) = \sum_{k=1}^d \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right).$$

The **information gain** from the attribute test on A is the expected reduction in entropy:

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A).$$

Information gain

In fact $Gain(A)$ is just what we need to implement the IMPORTANCE function. Returning to the attributes considered in **Figure 19.4**, we have

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

$$Gain(Type) = 1 - \left[\frac{2}{12} B\left(\frac{1}{2}\right) + \frac{2}{12} B\left(\frac{1}{2}\right) + \frac{4}{12} B\left(\frac{2}{4}\right) + \frac{4}{12} B\left(\frac{2}{4}\right) \right] = 0 \text{ bits},$$

Decision Tree Algorithm

```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree  
  
  if examples is empty then return PLURALITY-VALUE(parent_examples)  
  else if all examples have the same classification then return the classification  
  else if attributes is empty then return PLURALITY-VALUE(examples)  
  else  
     $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$   
    tree  $\leftarrow$  a new decision tree with root test A  
    for each value v of A do  
       $\text{exs} \leftarrow \{e : e \in \text{examples} \text{ and } e.A = v\}$   
      subtree  $\leftarrow$  LEARN-DECISION-TREE(exs, attributes – A, examples)  
      add a branch to tree with label (A = v) and subtree subtree  
  return tree
```

The decision tree learning algorithm. The function IMPORTANCE is described in [Section 19.3.3](#). The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

Decision Tree Algorithm

```
function LEARN-DECISION-TREE(examples, attributes, parent_examples) returns a tree  
  
  if examples is empty then return PLURALITY-VALUE(parent_examples)  
  else if all examples have the same classification then return the classification  
  else if attributes is empty then return PLURALITY-VALUE(examples)  
  else  
     $A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$   
    tree  $\leftarrow$  a new decision tree with root test A  
    for each value v of A do  
       $\text{exs} \leftarrow \{e : e \in \text{examples} \text{ and } e.A = v\}$   
      subtree  $\leftarrow$  LEARN-DECISION-TREE(exs, attributes – A, examples)  
      add a branch to tree with label (A = v) and subtree subtree  
  return tree
```

The decision tree learning algorithm. The function IMPORTANCE is described in [Section 19.3.3](#). The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

Overfitting

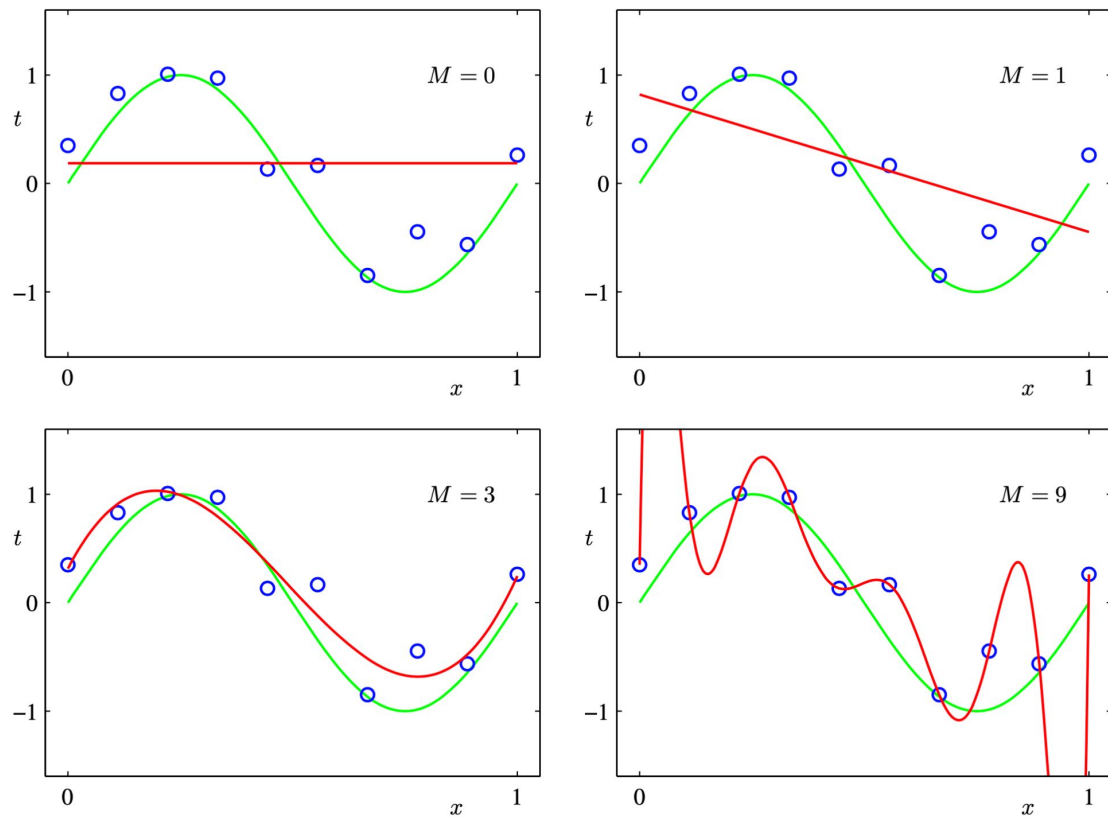
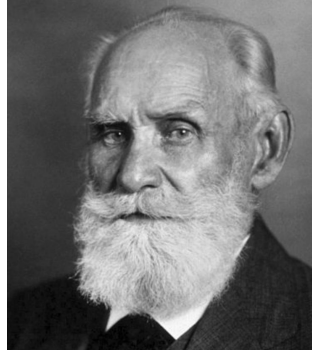
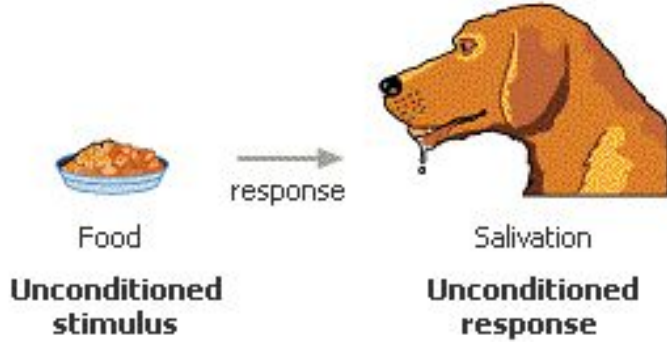


Figure 1.4 Plots of polynomials having various orders M , shown as red curves, fitted to the data set shown in Figure 1.2.

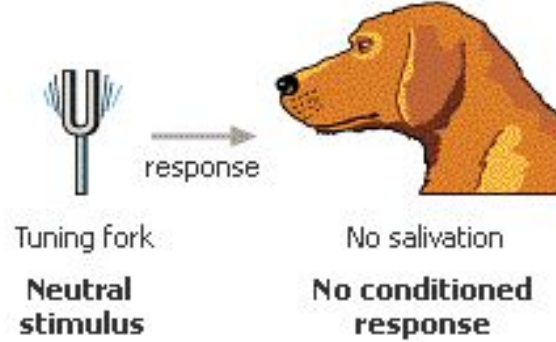
Pavlov's Dogs



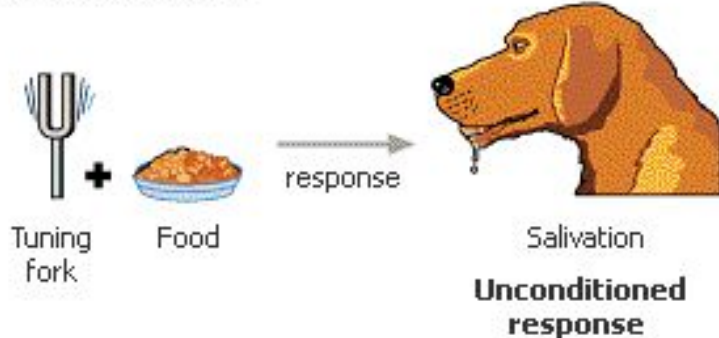
1. Before conditioning



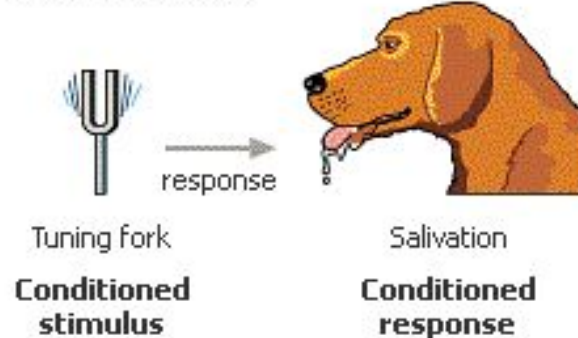
2. Before conditioning



3. During conditioning



4. After conditioning



Pavlov's Dogs



Intelligent?

Overfitting

We're attempting to model a data-generating process...

... but we only have a finite number of observations.

some correlations in our dataset are reliable...

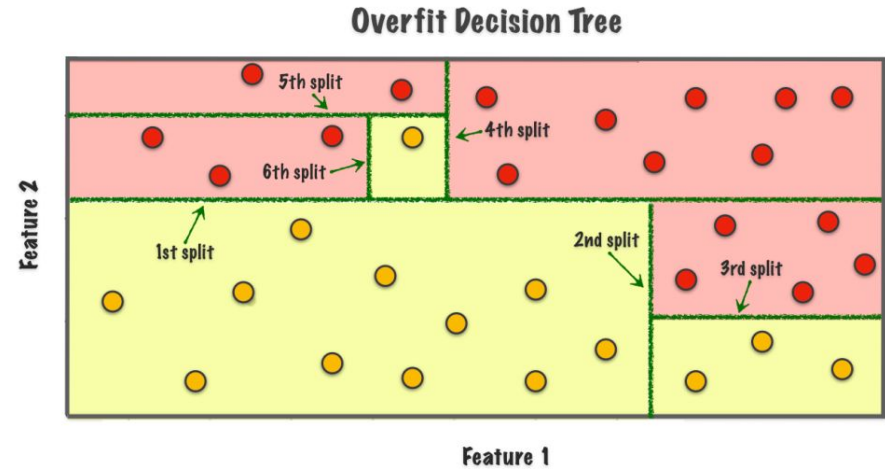
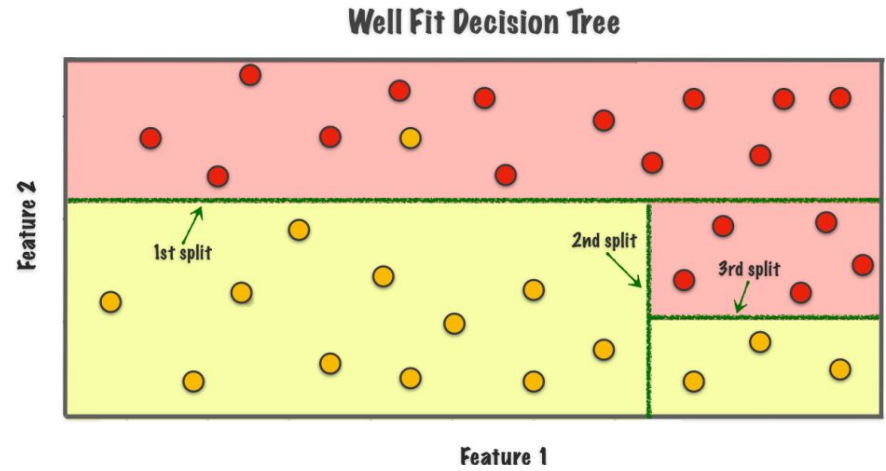
other correlations are spurious.

Overfitting

We want to learn the **signal** - the true relationships between variables

And ignore the **noise** - relationships that only appear in our data by coincidence

Overfitting



Preventing Overfitting – Regularization

Regularization techniques combat overfitting

What does it mean to be regular?



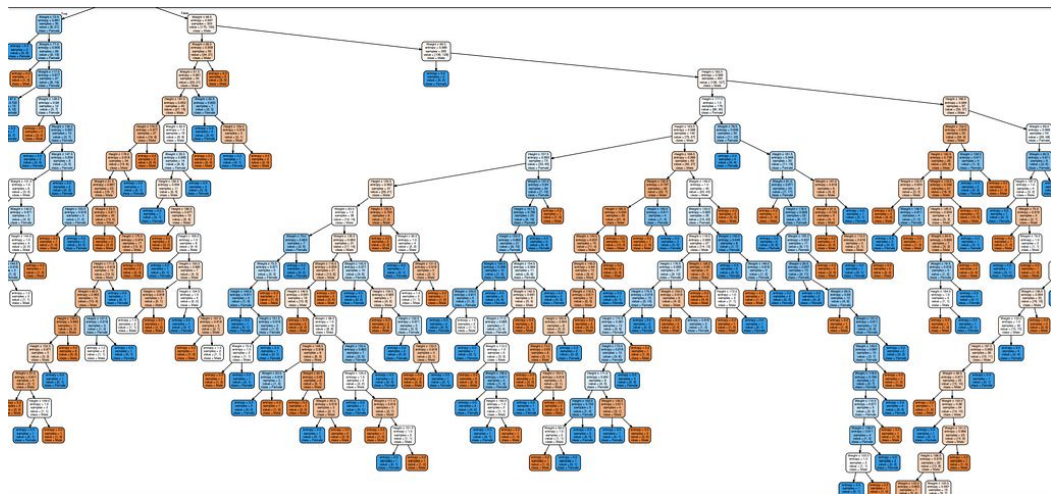
I'm gonna play video games for a living

You're never going to get into Johns Hopkins like this...



<https://towardsdatascience.com/an-introduction-to-decision-trees-with-python-and-scikit-learn-1a5ba6fc204f>

<https://towardsai.net/p/programming/decision-trees-explained-with-a-practical-example-fe47872d3b53>



Regularization for Decision Trees – Chi-Squared Pruning

Key Idea: remove decisions that only lead to small information gain

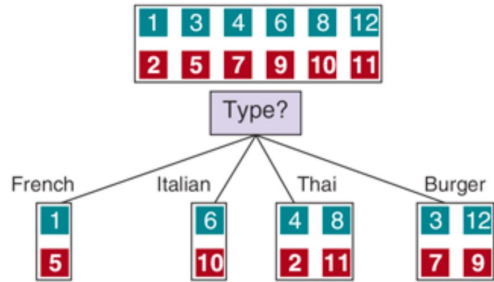
How much information gain do we need to keep the split?

Test for statistical significance

Hypothesis Testing:

Null Hypothesis: feature is uninformative, information gain is zero

Proportion in nodes
after splitting is the
same as before



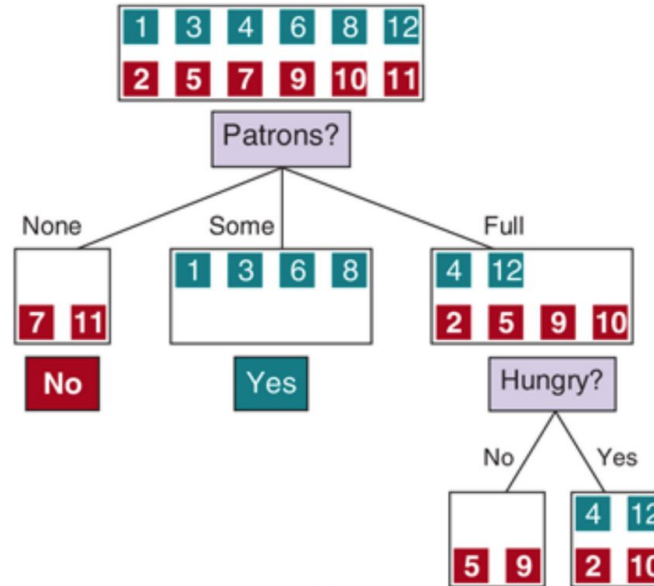
No information gain

Hypothesis Testing:

Null Hypothesis: feature is uninformative, information gain is zero

Proportion in nodes after splitting is different, more homogeneous

Higher information gain



Hypothesis Testing:

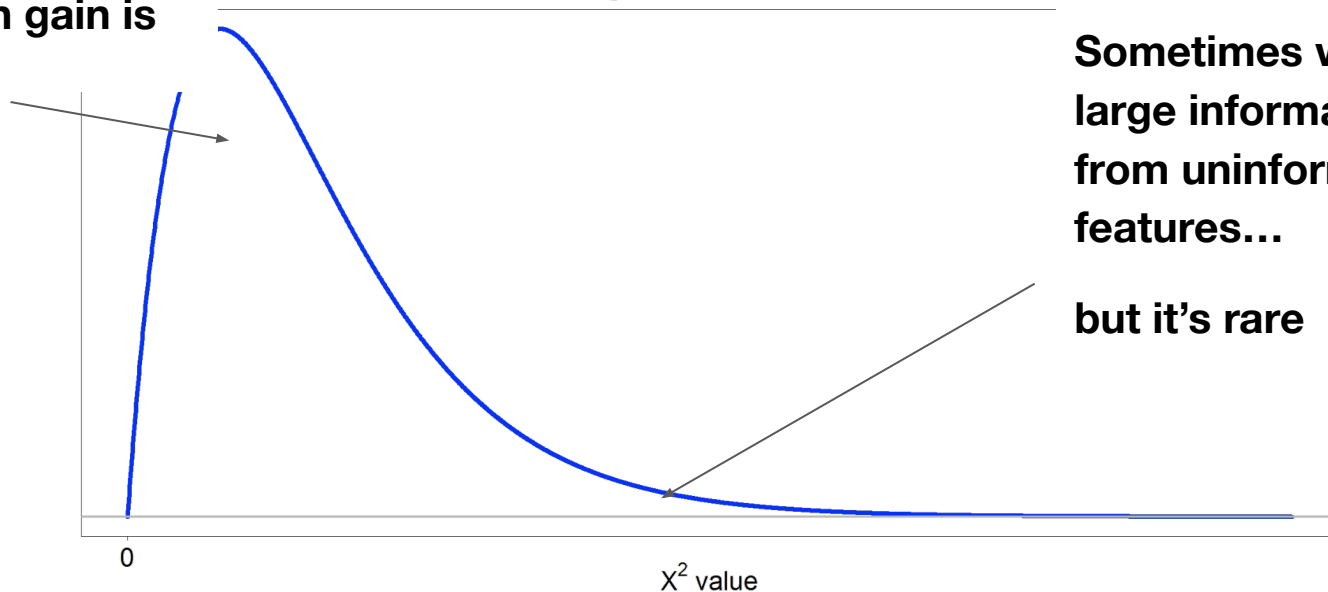
Null Distribution: Construct a distribution of the outcomes we would see by chance for uninformative features

**Most of the time,
information gain is
low**

A chi-squared distribution

**Sometimes we get a
large information gain
from uninformative
features...**

but it's rare



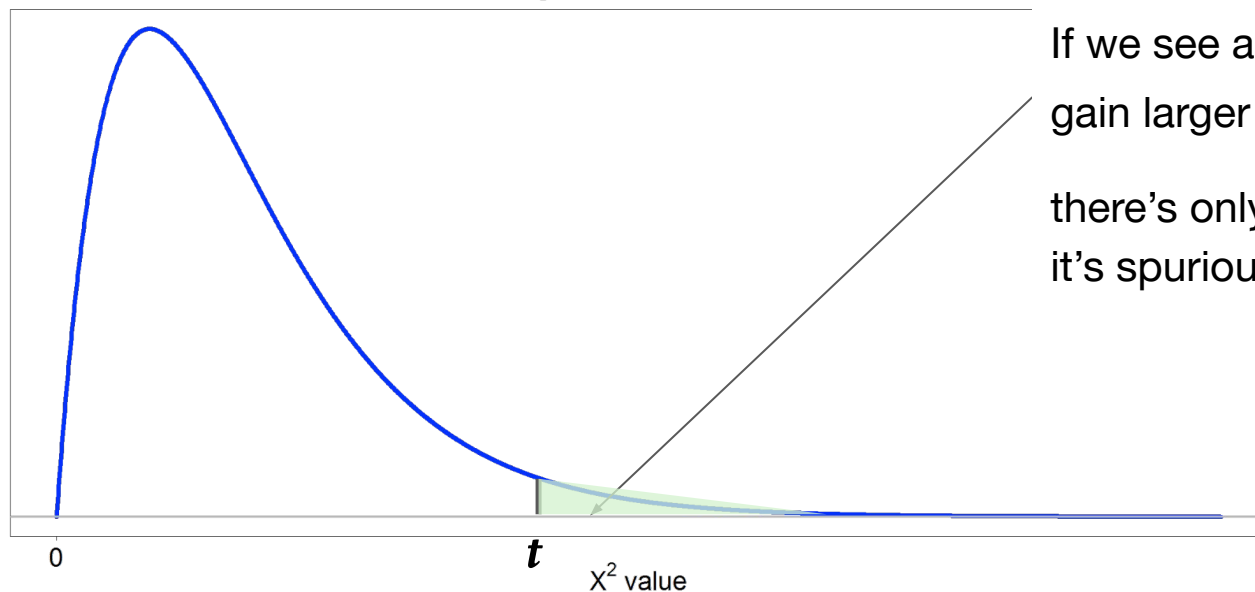
Hypothesis Testing:

Null Distribution: Construct a distribution of the outcomes we would see by chance for uninformative features

Set a threshold t on the information gain

A chi-squared distribution

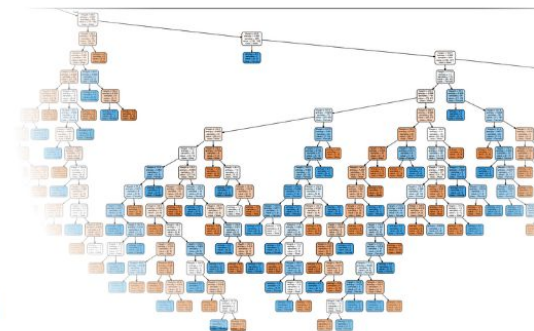
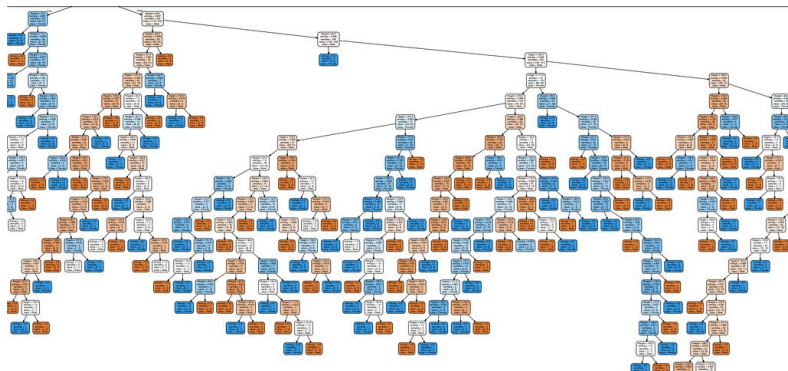
Area = 0.05



Regularization for Decision Trees – Chi-Squared Pruning

Step 1: Fit the tree until all the leaf nodes are homogeneous

Step 2: For each split, check whether it **significantly** reduces our prediction uncertainty using chi2 test, remove if not



Pruning vs. Early Stopping

Why fit the whole tree first?

Why not just check information gain along the way?

Sometimes features are only informative in combination

Boolean Math: XOR (\oplus)

X	Y	$X \oplus Y$
0	0	0
0	1	1
1	0	1
1	1	0

Our goal in machine learning is to select a hypothesis that will optimally fit future examples. To make that precise we need to define “future example” and “optimal fit.”

First we will make the assumption that the future examples will be like the past. We call this the **stationarity** assumption; without it, all bets are off. We assume that each example E_j has the same prior probability distribution:

$$\mathbf{P} (E_j) = \mathbf{P} (E_{j+1}) = \mathbf{P} (E_{j+2}) = \cdots,$$

Stationarity

and is independent of the previous examples:

$$\mathbf{P} (E_j) = \mathbf{P} (E_j | E_{j-1}, E_{j-2}, \dots) .$$

Examples that satisfy these equations are *independent and identically distributed* or **i.i.d.**.

I.i.d.

Selecting Models

Assume that the data distribution is stationary and samples are independent -> IID assumption

Define the “best model” to be the model that minimizes **error rate** on future unseen data

Error rate says how often our prediction does not match the truth

Estimate the error rate by testing our model on a set of examples not seen during training - the “test set”

Hyperparameters

If we are only going to create one hypothesis, then this approach is sufficient. But often we will end up creating multiple hypotheses: we might want to compare two completely different machine learning models, or we might want to adjust the various “knobs” within one model. For example, we could try different thresholds for χ^2 pruning of decision trees, or different degrees for polynomials. We call these “knobs” **hyperparameters**—parameters of the model class, not of the individual model.

Hyperparameters

Selecting Models

Assume that the data distribution is stationary and samples are independent -> IID assumption

Define the “best model” to be the model that minimizes **error rate** on future unseen data

Error rate says how often our prediction does not match the truth

Estimate the error rate by testing our model on a set of examples not seen during training or hyperparameter tuning - the “test set”

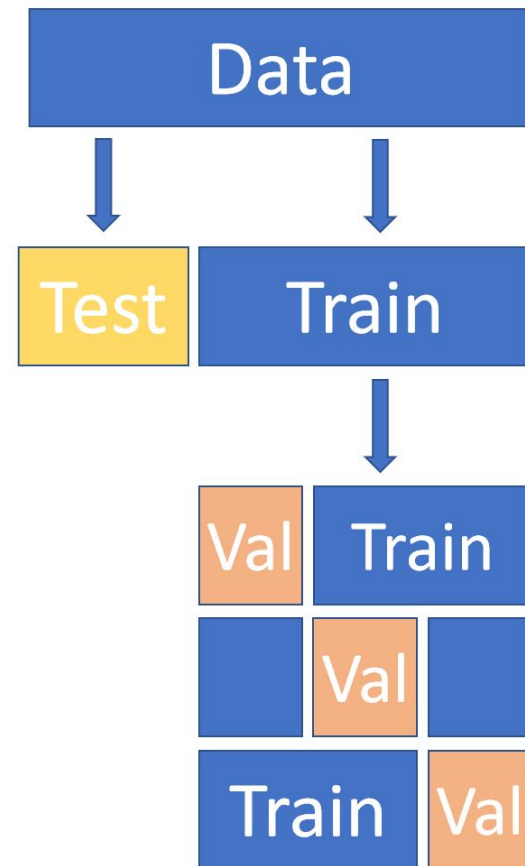
Splitting the Data

Split the data into train and test


For development, split train further into train and validation

When the train dataset is small, perform validation multiple times

(cross-validation)



Model Selection and Optimization

In **Figure 19.1**  (page 654) we saw a linear function underfit the data set, and a high-degree polynomial overfit the data. We can think of the task of finding a good hypothesis as two subtasks: **model selection**⁴ chooses a good hypothesis space, and **optimization** (also called **training**) finds the best hypothesis within that space.

⁴ Although the name “model selection” is in common use, a better name would have been “model *class* selection” or “hypothesis space selection.” The word “model” has been used in the literature to refer to three different levels of specificity: a broad hypothesis space (like “polynomials”), a hypothesis space with hyperparameters filled in (like “degree-2 polynomials”), and a specific hypothesis with all parameters filled in (like $5x^2 + 3x - 2$).