Decision Trees

Kaiqi Zhao
The University of Auckland

Slides are partially based on the materials from the University of British Columbia

Content

- Machine Learning Example
- Decision Trees
- Hypothesis Space
- Decision Tree Learning
- Supervised Learning
- Unsupervised Learning

Example: Should we wait for a table?

- You want to figure out whether you should wait for a table or not in a restaurant.

	Alt	**Bar**	**Fri**	**Patrons**	**Price**	**Res**	**Type**	**Est**	**WillWait**
x_1	Yes	No	No	Some	$$$	Yes	French	Short	Yes
x_2	Yes	No	No	Full	$	No	Thai	Mid	No
x_3	No	Yes	No	Some	$	No	Burger	Short	Yes
x_4	Yes	No	Yes	Full	$	No	Thai	Mid	Yes
x_5	Yes	No	Yes	Full	$$$	Yes	French	Long	No
x_6	No	Yes	No	Some	$$	Yes	Italian	Short	Yes

[Example adapted from the AI book of Stuart J. Russell and Peter Norvig]

Alt: Is there any other suitable restaurants nearby?
Bar: Is there any bar area to wait in?
Fri: Is it Friday or Saturday?
Patrons: How many people are there (Some, Full)

Price: price range ($, $$, $$$)
Res: reservation of table?
Type: the type of restaurant (French, Thai, Burger, Italian)
Est: estimated waiting time (Short, Mid, Long)

Example: Should we wait for a table?

- You want to figure out whether you should wait for a table or not in a restaurant.

	Alt	**Bar**	**Fri**	**Patrons**	**Price**	**Res**	**Type**	**Est**	**WillWait**
$\boldsymbol{x_1}$	Yes	No	No	Some	$$$	Yes	French	Short	Yes
$\boldsymbol{x_2}$	Yes	No	No	Full	$	No	Thai	Mid	No
$\boldsymbol{x_3}$	No	Yes	No	Some	$	No	Burger	Short	Yes
$\boldsymbol{x_4}$	Yes	No	Yes	Full	$	No	Thai	Mid	Yes
$\boldsymbol{x_5}$	Yes	No	Yes	Full	$$$	Yes	French	Long	No
$\boldsymbol{x_6}$	No	Yes	No	Some	$$	Yes	Italian	Short	Yes

[Example adapted from the AI book of Stuart J. Russell and Peter Norvig]

- How can we find the pattern and determine what leads to the decision?
  - Can you look at one attribute at a time?
  - Should we look at all attributes?

A Naïve Model

Example	Features (attributes)								Class labels
	Alt	Bar	Fri	Patrons	Price	Res	Type	Est	WillWait
x_1	Yes	No	No	Some	$$$	Yes	French	Short	Yes
x_2	Yes	No	No	Full	$	No	Thai	Mid	No
x_3	No	Yes	No	Some	$	No	Burger	Short	Yes
x_4	Yes	No	Yes	Full	$	No	Thai	Mid	Yes
x_5	Yes	No	Yes	Full	$$$	Yes	French	Long	No
x_6	No	Yes	No	Some	$$	Yes	Italian	Short	Yes

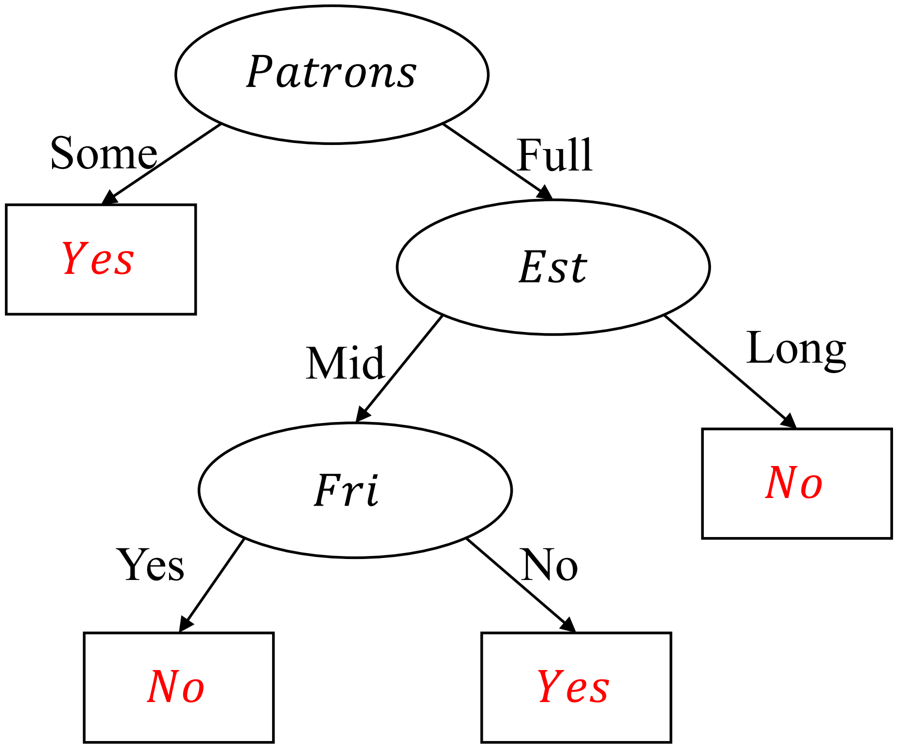
- A very naïve model – always predict one class label
  - Count how many times each label occurred in the data (4 Yes vs. 2 No)
  - Always predict the most common label, e.g., Yes.
- Problems: Not accurate! Features are not considered!
- How to leverage the features?

Decision Trees

Decision Trees

- Decision trees are simple models consisting of
  - A nested sequence of "if-else" decisions based on the features (splitting rules)
  - A class label as a return value at the end of each sequence

Can draw sequences of decisions as a tree



Example decision tree

if $patrons = Some$ **then**
    return Yes
Else if $patrons = Full$ **then**
    if $Est = Long$ **then**
        return No
    else
        if $Fri = Yes$ **then**
            return No
        else
            return Yes
    end
end

Another Example - Food Allergies

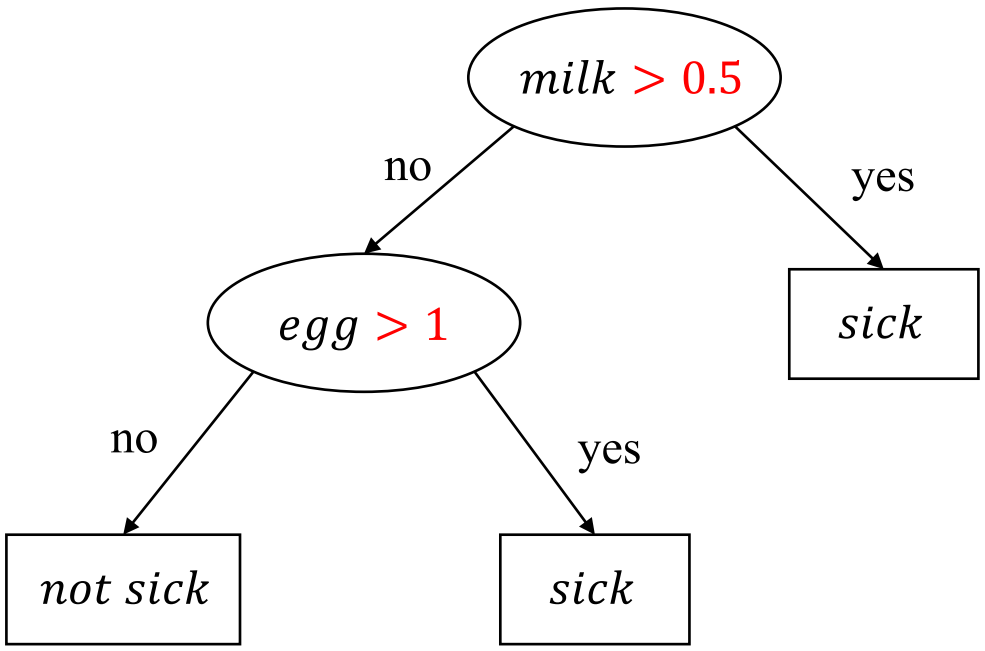
- The previous restaurant example has only **discrete features**, how about **real-value features**?
- If you frequently start getting an upset stomach and suspect an adult-onset food allergy. To solve the mystery, you start a food journal

Egg	Milk	Fish	Wheat	Shellfish	Peanuts	…	Sick?
0.0	0.7	0.0	0.3	0.0	0.00	…	1
0.3	0.7	0.0	0.6	0.0	0.01	…	1
0.0	0.0	0.0	0.8	0.0	0.00	…	0
0.3	0.7	1.2	0.0	0.1	0.01	…	1
0.3	0.0	1.2	0.3	0.1	0.01	…	1

Another Example - Food Allergies

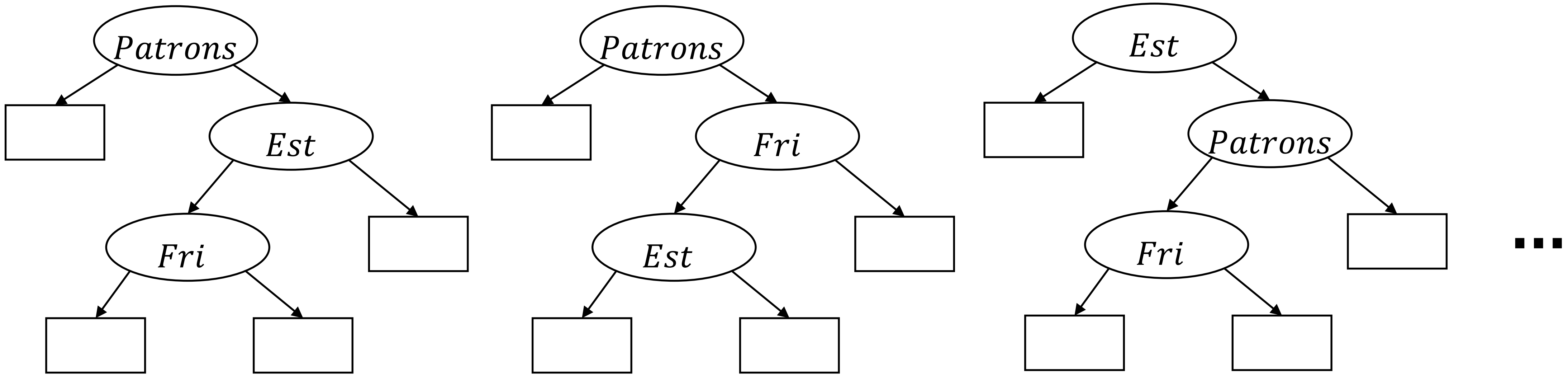
- The splitting rule of the continuous feature contains a threshold.

```
if milk > 0.5 then
    return sick
else
    if egg > 1 then
        return sick
    else
        return not sick
    end
end
```



How do we find the best decision tree?

- The number of possible decision trees is exponential!!!



- The number of possible orders to select the attributes is already exponential!
- Learning the smallest decision tree is an NP-hard problem (Hyafil & Rivest '76)

Hypothesis Space

Hypothesis Space - Learning as Search

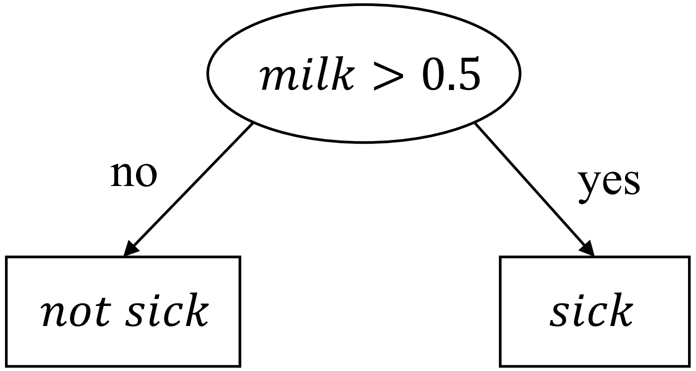
- Learning can be defined as searching the best hypothesis (e.g., decision tree) for all observed data
  - For decision trees, the hypothesis space are all possible decision trees that can be generated for a data set
  - The learner searches through the space and returns the best hypothesis, for decision trees, the tree that potentially best predicts new data

- For a small space, it is possible to test all hypotheses
- When the hypothesis space is large, how do we search the space to find the "best" decision tree?

Decision Tree Learning

Decision Stumps

- The simplest case - "**decision stump**"
  - A decision tree with only ONE splitting rule based on thresholding ONE feature



- How do we find the best "rule" (feature, threshold, and leaf labels)?
  1. Define a 'score' for the rules
  2. Search for the rule with the best score
- What would you suggest as a score?

Learning a Decision Stump: Accuracy Score

- The most intuitive score: classification accuracy
  - How many examples are labelled correctly?

- Computing classification accuracy for $(egg > 1)$:
  - Find most common labels if we use this rule:
    - When (egg > 1), we were "sick" 2 times out of 2
    - When (egg ≤ 1), we were "not sick" 3 times out of 4
  - Compute accuracy:
    - The accuracy ("score") of the rule (egg > 1) is 5 times out of 6

Egg	Milk	Fish	…	Sick?
1	0.7	0.0	…	1
2	0.7	0.0	…	1
0	0.0	1.2	…	0
0	0.7	1.2	…	0
2	0.0	1.3	…	1
0	0.0	0.0	…	0

Learning a Decision Stump: Example

- Search for the decision stump maximizing classification score:
  - **Baseline rule** – predict the most common label: this gets $3/6$ accuracy
  - If $(milk > 0)$ predict "sick" $(2/3)$ else predict "not sick" $(2/3)$: $4/6$ accuracy
  - If $(fish > 0)$ predict "not sick" $(2/3)$ else predict "sick" $(2/3)$: $4/6$ accuracy
  - If $(fish > 1.2)$ predict "sick" $(1/1)$ else predict "not sick" $(3/5)$: $4/6$ accuracy
  - If $(egg > 0)$ predict "sick" $(3/3)$ else predict "not sick" $(3/3)$: $6/6$ accuracy
  - If $(egg > 1)$ predict "sick" $(2/2)$ else predict "not sick" $(3/4)$: $5/6$ accuracy
- Highest-scoring rule: $(egg > 0)$, then "sick", else "not sick"
- Questions:
  - Do we need to test the rule $(egg > 3)$?
  - Do we need to test the rule $(egg > 0.5)$?
  - Do we need to test the rule $(egg < 1)$?

Egg	Milk	Fish	…	Sick?
1	0.7	0.0	…	1
2	0.7	0.0	…	1
0	0.0	1.2	…	0
0	0.7	1.2	…	0
2	0.0	1.3	…	1
0	0.0	0.0	…	0

- We only need to test feature thresholds that happen in the data!

Decision Tree Learning

- Decision stumps have only ONE rule based on only ONE feature
  - Very limited class of models: usually not very accurate for most tasks

- Decision tree learning
  - Recursive stump learning with greedy choice

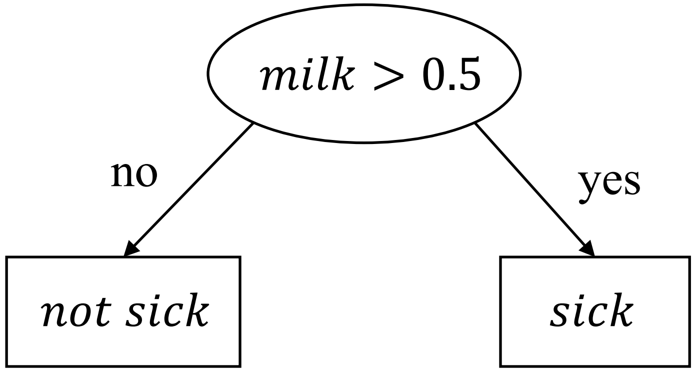
THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Example of Greedy Recursive Splitting

Start with the full data set

Egg	Milk	…	Sick?
0	0.7	…	1
1	0.7	…	1
0	0.0	…	0
1	0.6	…	1
1	0.0	…	0
2	0.6	…	1
0	1.0	…	1
2	0.0	…	1
0	0.3	…	0
1	0.6	…	0
2	0.0	…	1

Find the decision stump with the best score



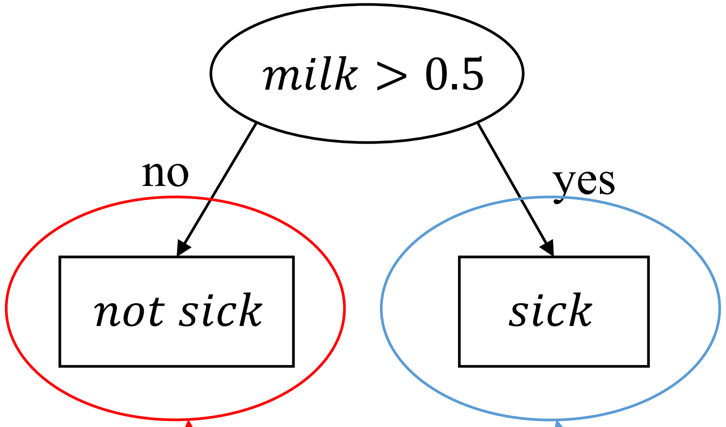
Split into two smaller data sets based on stump

Egg	Milk	…	Sick?
0	0.0	…	0
1	0.0	…	0
2	0.0	…	1
0	0.3	…	0
2	0.0	…	1

Egg	Milk	…	Sick?
0	0.7	…	1
1	0.7	…	1
1	0.6	…	1
2	0.6	…	1
0	1.0	…	1
1	0.6	…	0

Greedy Recursive Splitting

We now have a decision stump and two data sets

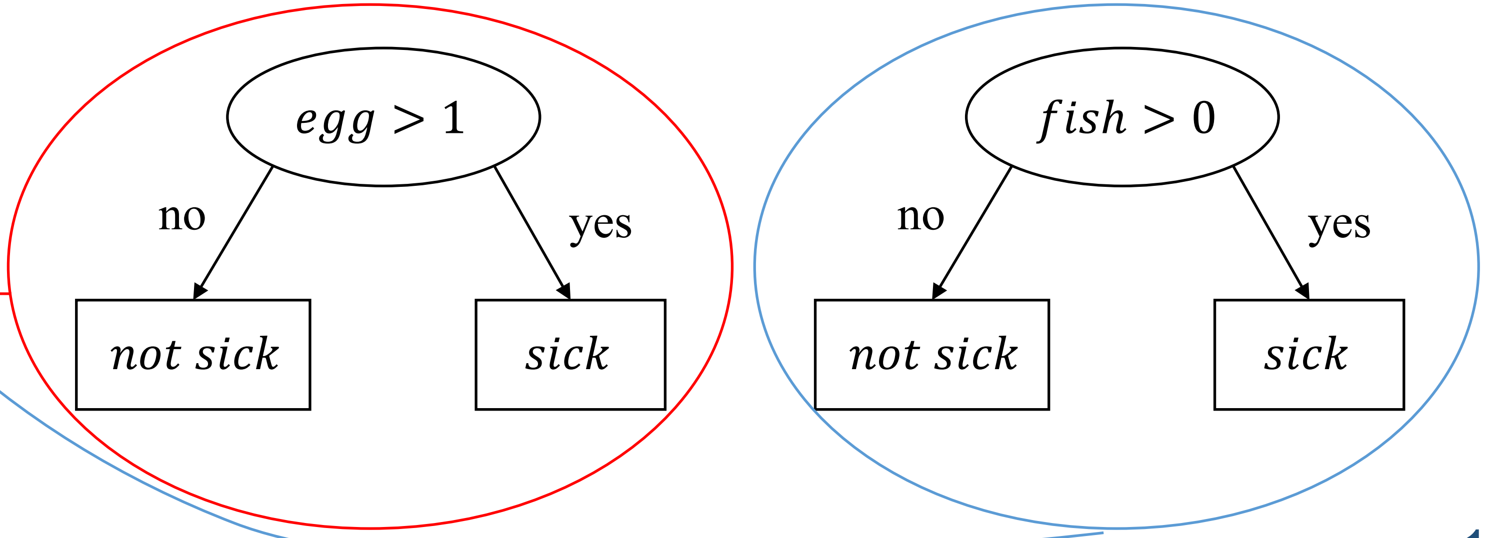


$milk \leq 0.5$

Egg	Milk	…	Sick?
0	0.0	…	0
1	0.0	…	0
2	0.0	…	1
0	0.3	…	0
2	0.0	…	1

$milk > 0.5$

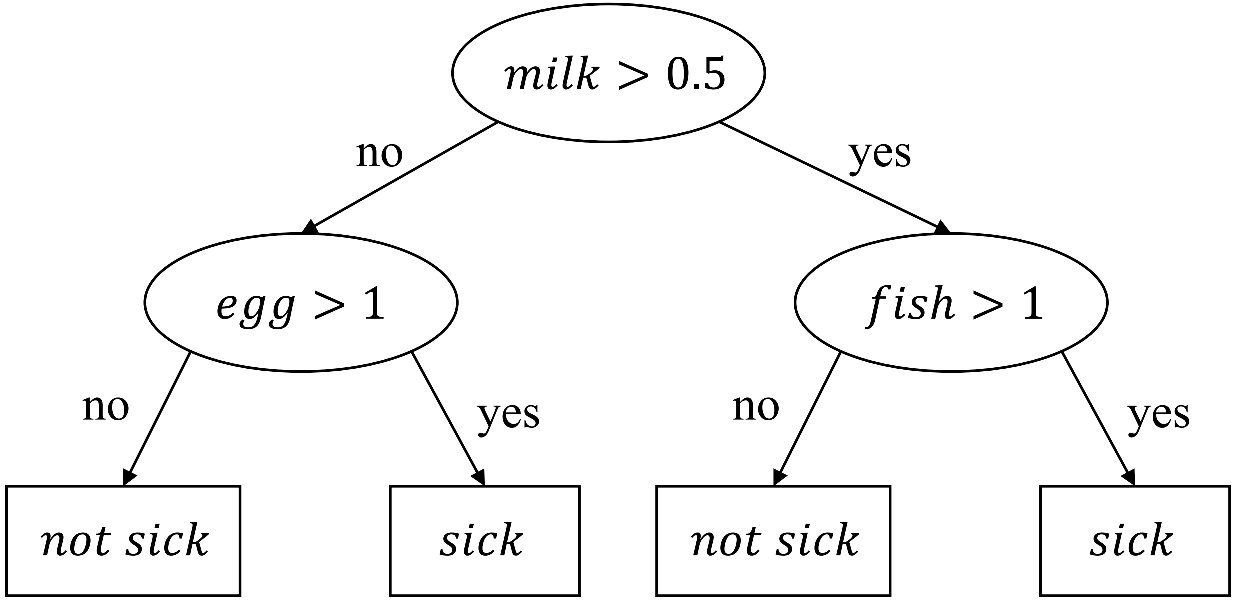
Egg	Milk	…	Sick?
0	0.7	…	1
1	0.7	…	1
1	0.6	…	1
2	0.6	…	1
0	1.0	…	1
1	0.6	…	0



Fit a decision stump to each leaf's data

Then add these stumps to the tree

Greedy Recursive Splitting

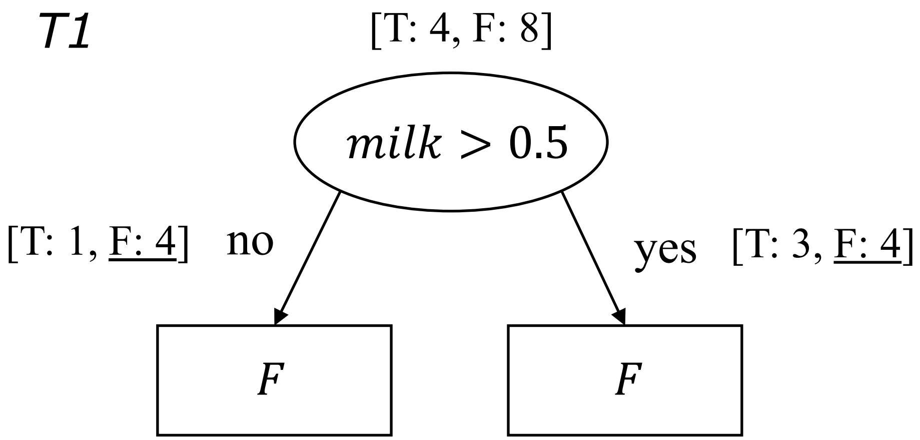


- When to stop splitting?
  1. Can’t split a leaf node further, e.g., only 1 example in the leaf node
  2. Leaves have only one label
  3. User-defined maximum depth → Should work, but how to determine the max depth?
  4. Should we stop when accuracy doesn’t increase → You might get a shallow tree with low accuracy!
- Multiple criteria can be applied, e.g., 1, 2 and 3

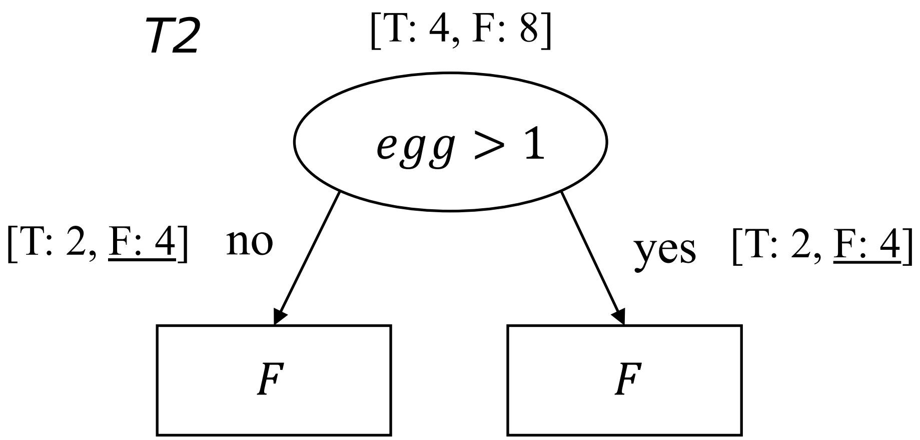
Information Gain

Revisit Accuracy Score…

- Question: Is the accuracy score a good way to choose the right feature to split on?
  - Consider a decision stump (T - Sick, F - Not Sick)



Accuracy before splitting (baseline rule): 2/3
Accuracy of this rule: 2/3



Accuracy before splitting (baseline rule): 2/3
Accuracy of this rule: 2/3

- Both *T1* and *T2* can’t improve accuracy!!
- However, *T1* seems to be a better choice because at least the left branch is more predictable

Choosing a Good Splitting Rule

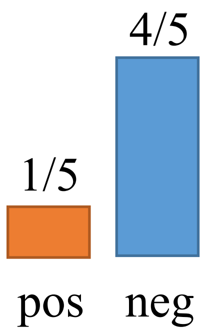
- Intuitively, we want each splitting rule to best distinguish the class labels.
  - The best case is to have only one class in each resulting branch (deterministic)
  - The worst case is all classes are equally probable in each branch (random)

- The more deterministic the better!
- How to quantify the uncertainty?

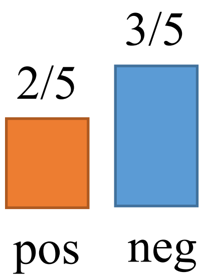
Quantifying Uncertainty

- Entropy: $H(X) = -\sum_{x \in X} p(X) \log_2 p(X)$
  - Example: $X = \{pos, neg\}$

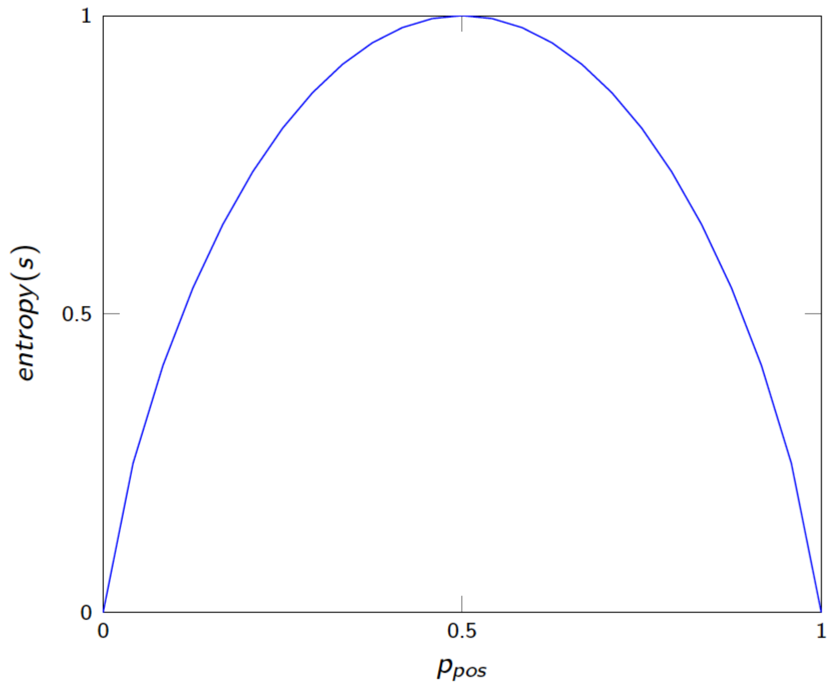
$$H(X) = -p_{pos} \log_2 p_{pos} - p_{neg} \log_2 p_{neg}$$



$$-\frac{1}{5}\log_2\frac{1}{5} - \frac{4}{5}\log_2\frac{4}{5} \approx 0.72$$



$$-\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} \approx 0.97$$



- High Entropy – more uncertain, e.g., uniform distribution has the highest entropy
- Low Entropy – more certain, e.g., $p_{pos} = 1$ or $p_{neg} = 1$

Information Gain

- Idea: choose the split that decreases the entropy of labels the most
  - The decrease of entropy: “Information Gain” (IG)

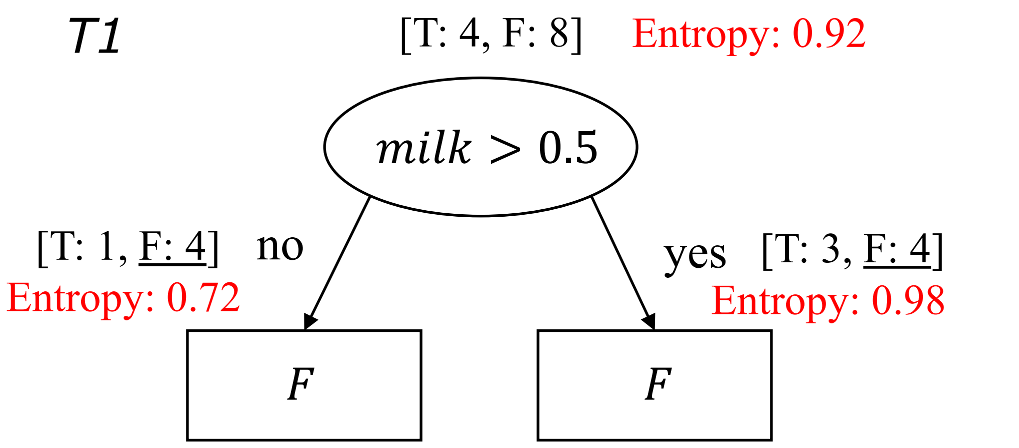
$$IG(S, A) = \underbrace{H(S)}_{\text{entropy before split}} - \boxed{\sum_{branch \in B} \overbrace{\frac{|S_{branch}|}{|S|}}^{\text{fraction of examples in each branch}} \cdot \underbrace{H(S_{branch})}_{\text{entropy of each branch}}}$$

The expected entropy of all branches

- Information gain for baseline rule (majority class) is 0
- Information gain is large if labels are much "more predictable" ("less random") in next layer

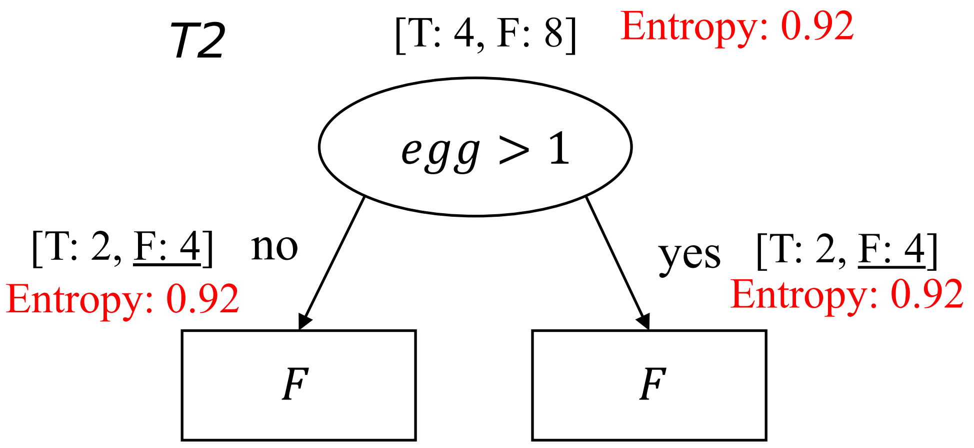
Information Gain: Example

- Consider a decision stump (T - Sick, F - Not Sick)
  - The two examples below have the same accuracy
  - How about entropy?



Entropy before splitting (baseline rule): 0.92

IG of this rule: 0.92 – (5/12)*0.72 – (7/12)*0.98=0.048

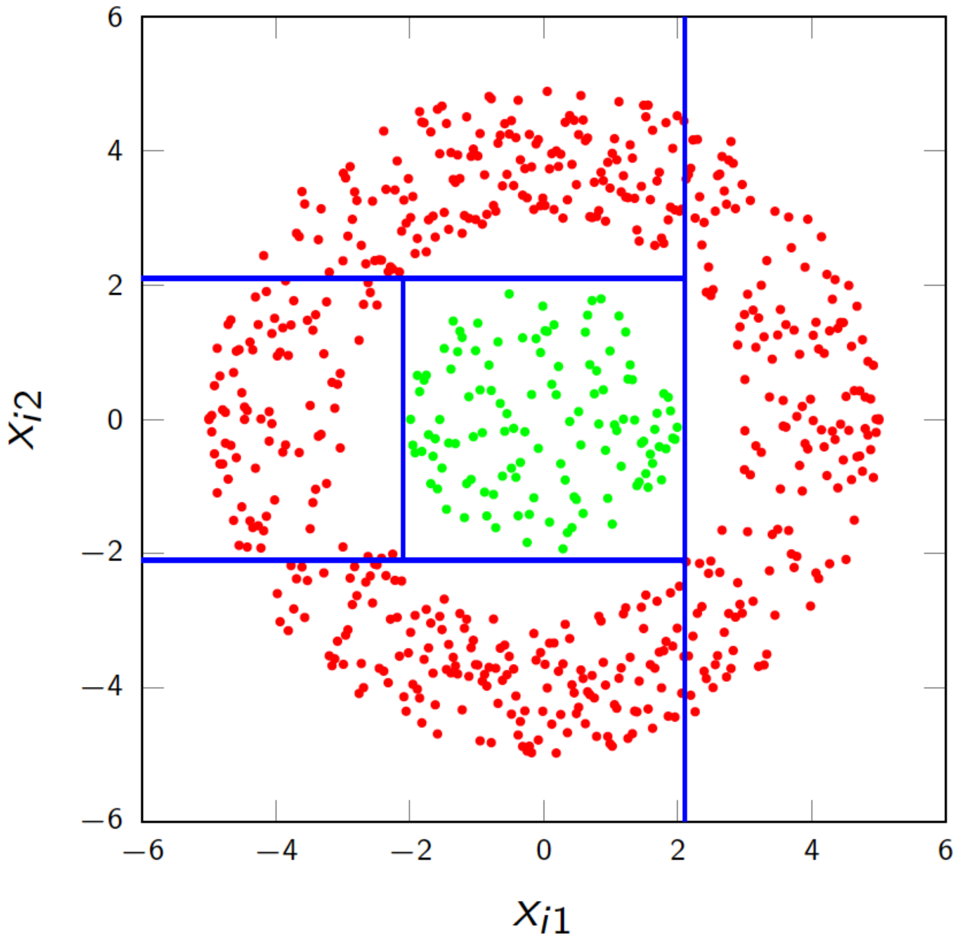


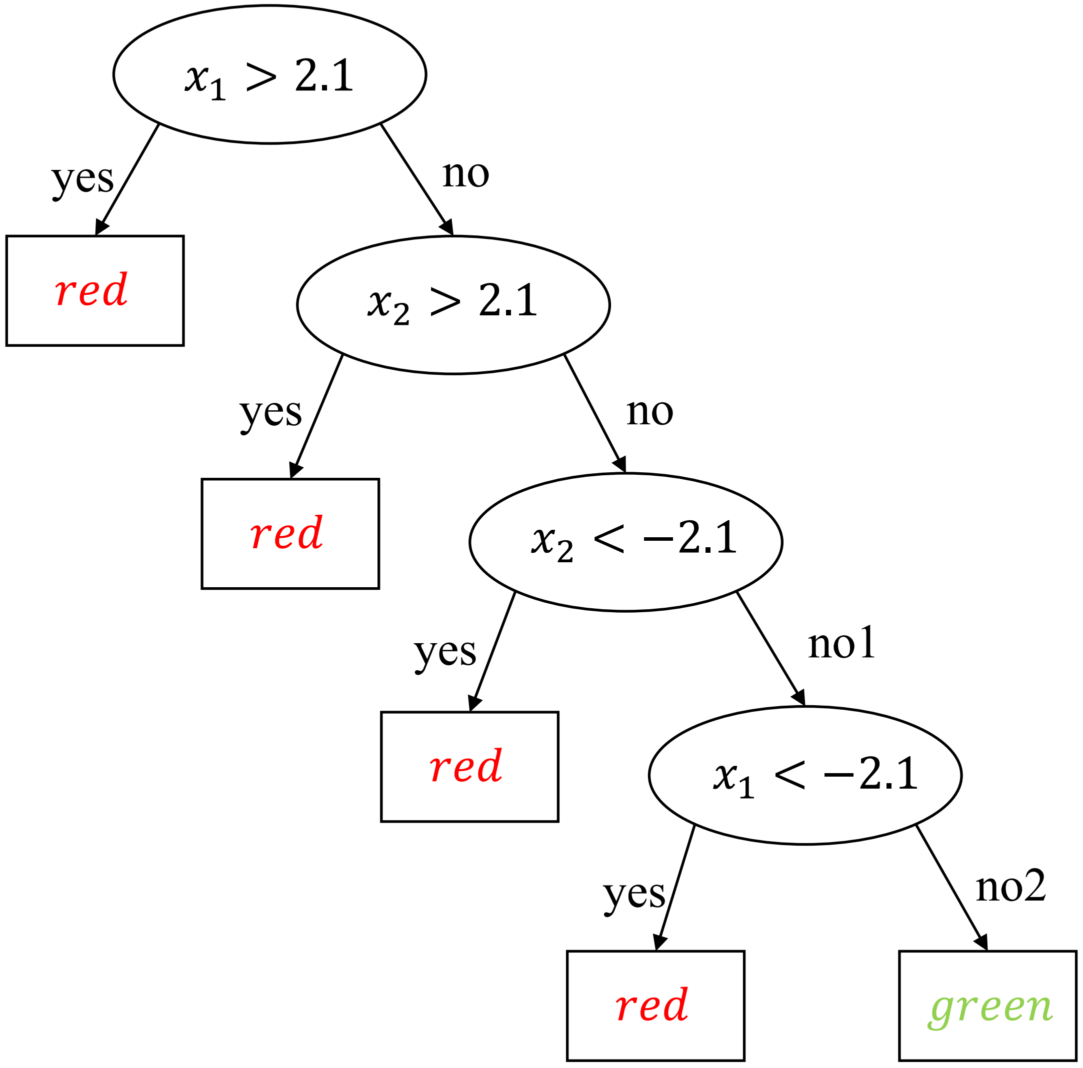
Entropy before splitting (baseline rule): 0.92

IG of this rule: 0.92 – 0.5*0.92– 0.5*0.92=0

- *T1* has a positive information gain because its left branch is more predictable for the negative examples

Decision tree as feature space partitioning





A decision tree partitions the feature space along feature axis!

Decision Trees

- Advantages:
  - Easy to implement
  - Interpretable
  - Learning is fast prediction is very fast
  - Can elegantly handle a small number missing values during training
- Disadvantages
  - Hard to find optimal set of rules
  - Greedy splitting often not accurate, requires very deep trees

Supervised Learning

Supervised Learning

- We can formulate this as a supervised learning problem

Example	Features									Class labels
	Alt	**Bar**	**Fri**	**Patrons**	**Price**	**Res**	**Type**	**Est**		**WillWait**
x_1	Yes	No	No	Some	$$$	Yes	French	Short	$\Rightarrow$	Yes
x_2	Yes	No	No	Full	$	No	Thai	Mid		No
x_3	No	Yes	No	Some	$	No	Burger	Short		Yes
x_4	Yes	No	Yes	Full	$	No	Thai	Mid		Yes
…	…	…	…	…	…	…	…	…		…

- The input for an **example** (e.g., x_1) is a set of **features** (Alt, Bar, …)
- The output is a target **class label** (Yes or No)
- **Supervised learning:**
  - Use data to find a model that outputs the right label based on the features
  - The model should be able to predict with arbitrary **new feature combinations**.

Supervised Learning Formulation

$$X = \begin{array}{c} \begin{array}{cccccccc} \textbf{Alt} & \textbf{Bar} & \textbf{Fri} & \textbf{Patrons} & \textbf{Price} & \textbf{Res} & \textbf{Type} & \textbf{Est} \end{array} \\ \underbrace{\left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 3 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 3 & 1 & 0 & 2 \\ 0 & 1 & 0 & 0 & 2 & 1 & 3 & 0 \end{array}\right]}_{d} \end{array} \left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\\1\end{array}}\right\} n \qquad y = \begin{array}{c} \textbf{WillWait} \\ \boxed{\begin{array}{c} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{array}} \end{array}$$

- If the attributes are categorical, we can represent the input as a matrix by encoding the input attributes into numeric values:
  - Binary values: Alt: (Yes → 1, No → 0), Bar: (Yes → 1, No → 0)
  - Multiple outcomes: Price: ($→1, $$→2,$$$→3), Type: (Frech→0, Thai→1, Burger→2, Italian→3)

Supervised Learning Formulation

$$X = \begin{array}{c} \begin{array}{cccccccc} \textbf{Alt} & \textbf{Bar} & \textbf{Fri} & \textbf{Patrons} & \textbf{Price} & \textbf{Res} & \textbf{Type} & \textbf{Est} \end{array} \\ \underbrace{\left[\begin{array}{cccccccc} 1 & 0 & 0 & 0 & 3 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 2 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 3 & 1 & 0 & 2 \\ 0 & 1 & 0 & 0 & 2 & 1 & 3 & 0 \end{array}\right]}_{d} \end{array} \left.\vphantom{\begin{array}{c}1\\1\\1\\1\\1\\1\end{array}}\right\} n \qquad y = \begin{array}{c} \textbf{WillWait} \\ \boxed{\begin{array}{c} 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{array}} \end{array}$$

- Feature matrix X has rows as examples, columns as features
  - x_{ij} is the j-th feature for the i-th example (e.g., x_{12} is the Bar attribute of the first example)
  - x_i is the list of all features for the i-th example
- Label vector y contains the labels of the examples
  - y_i is the label of the i-th example (1 for “wait", 0 for “do not wait")

Supervised Learning Notation

$$X = \begin{array}{c} \begin{array}{ccccccc} \text{Egg} & \text{Milk} & \text{Fish} & \text{Wheat} & \text{Shellfish} & \text{Peanuts} & \dots \end{array} \\ \underbrace{\left.\begin{bmatrix} 0.0 & 0.7 & 0.0 & 0.3 & 0.0 & 0.00 & \dots \\ 0.3 & 0.7 & 0.0 & 0.6 & 0.0 & 0.01 & \dots \\ 0.0 & 0.0 & 0.0 & 0.8 & 0.0 & 0.00 & \dots \\ 0.3 & 0.7 & 1.2 & 0.0 & 0.1 & 0.01 & \dots \\ 0.3 & 0.0 & 1.2 & 0.3 & 0.1 & 0.01 & \dots \end{bmatrix}\right\} n}_{d} \end{array} \qquad y = \begin{array}{c} \text{Sick?} \\ \hline 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{array}$$

- Training phase
  - Use X and y to find a model (like a decision stump)
- Prediction phase
  - Given an example x_i , use model to predict a label $\hat{y}_i$ ("sick" or "not sick")
- Training error
  - Fraction of times our prediction $\hat{y}_i$ does not equal the true y_i label

Supervised Learning

- General supervised learning problem:
  - Take features of examples and corresponding labels as inputs
  - Find a model that can accurately predict the labels of new examples
- This is the most successful or widely used machine learning technique
  - Spam filtering, optical character recognition, speech recognition, classifying tumours, etc.
- We've talked about categorical labels in the decision tree examples, which is called classification. The model is called a classifier.
- When the labels are real-value numbers, the problem is called regression. The model is called a regressor.

Unsupervised Learning

Unsupervised Learning

- Supervised learning:
  - We have features x_i and class labels y_i
  - Find a mapping from x_i to y_i
- Unsupervised learning
  - We only have x_i values, but no explicit labels
  - Understand the underlying patterns (e.g., clusters)

- Some unsupervised learning tasks
  - Clustering: What types of x_i are there?
  - Association rules: Which x_j occur together?
  - Outlier detection: Is this a 'normal' x_i ?
  - Similarity search: Which examples look like this x_i ?
  - Latent-factors: What 'parts' are the x_i made from?
  - Ranking: Which are the most important x_i?

Summary

- **Decision trees**: predicting a label using a sequence of simple rules
- **Decision stumps**: simple decision trees with only one splitting rule
- **Greedy recursive splitting**: uses a sequence of decision stumps to grow a tree
  - Very fast and interpretable, but not always the most accurate
- **Information gain**: splitting score based on decreasing entropy
- **Supervised learning** v.s. **Unsupervised Learning**
  - Supervised learning: finding a mapping from input features to class labels
  - Unsupervised learning: finding patterns within data without explicit labels