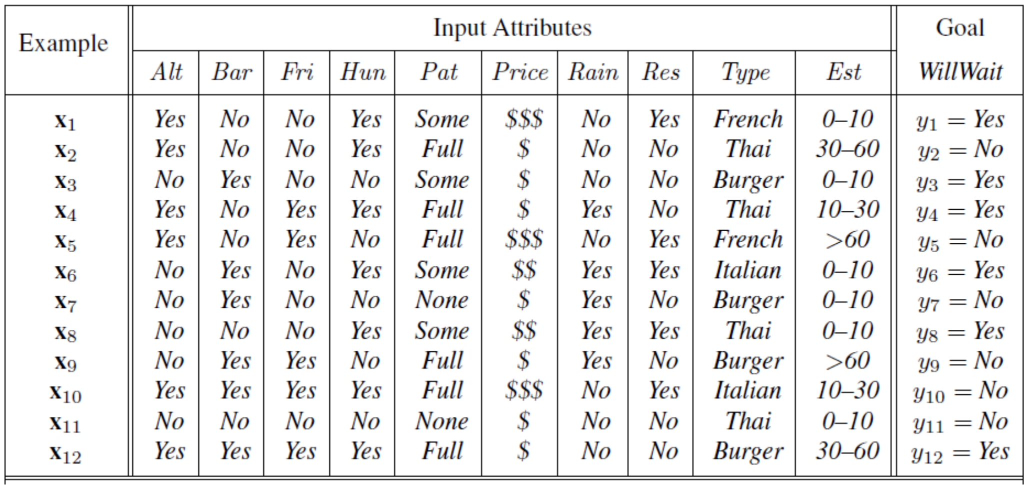
CS 6140: Machine Learning **Spring 2018**
College of Computer and Information Science
Northeastern University
Lecture 1 **January, 8**
Instructor: Bilal Ahmed **Scribe: Bilal Ahmed & Virgil Pavlu**

Decision Trees

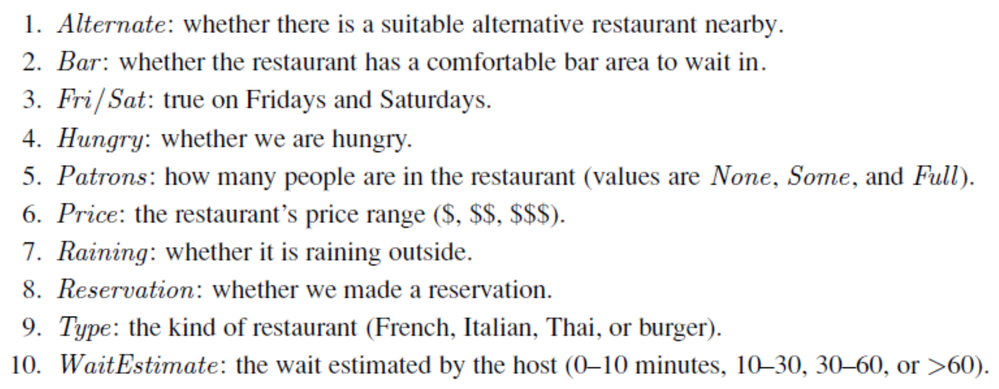
Assume we are given the following data:



Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

Figure 1: Training data for the *wait-for-table* classification task.

The task at hand is to develop an application that advises whether we should wait for a table at a restaurant or not. To this end we are going to assume that the given data represents the true concept *WillWait* and we will further assume that all future data will be described by the same input attributes *aka* features. A description of these features is provided in Figure-2.



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, or >60).

Figure 2: The available input attributes (features) and their brief descriptions.

We can view this problem as playing a *20 questions* game, where every question we ask:

- should help us in narrowing down the value of the target *WillWait*
- depends on the previous questions that we may have already asked

Here, features represent questions and the answer is the specific feature value for a data instance.
What happens when we ask a question?

- Depending upon the number of possible answers the data is *split* into multiple subsets
- If a subset has the same value for the target concept (e.g., *WillWait*) then we have our answer
- If the subset has different values for the target concept then we need to ask more questions

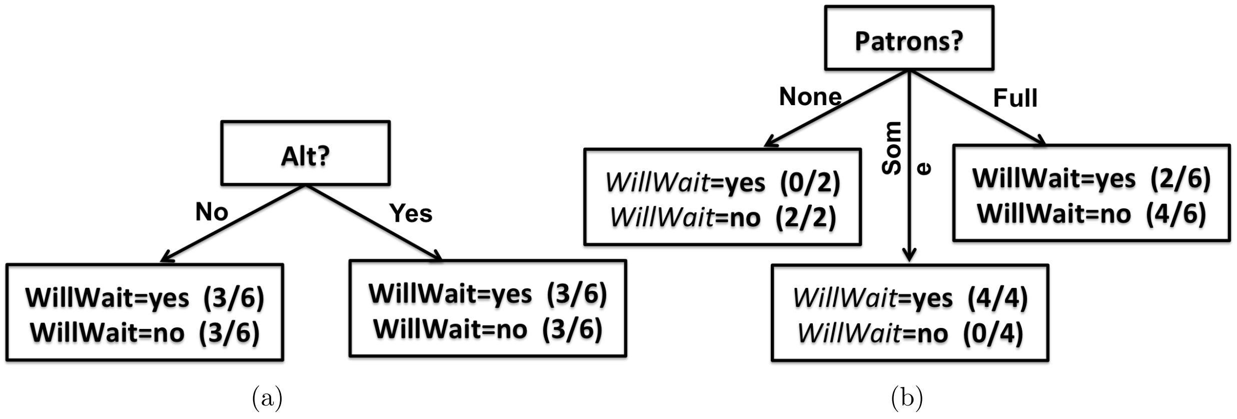


Figure 3: Possible splitting of data based on different features. The root shows the feature we are testing, and its child nodes show how the data is split.(a) Split using the *Alternate?* feature, (b) using the *Patrons?* feature.

Figure-3 shows what happens when we ask two different questions. For instance in the case of Figure-3a, we have selected to ask the question whether an alternate restaurant is available nearby or not, in this case we can see that the question is not informative in regards to answering our question because no matter what the answer is we have a fifty-fifty chance of being correct. This is as good as deciding randomly (flipping a fair coin).
However, if we ask the question about the number of patrons in the restaurant (Figure-3b) then for two out of the three possible answers we can predict the target concept with 100% confidence. If we keep asking questions till we reach nodes in which we can make the predictions with acceptable chances of error or run out of questions we will get a decision tree built using the training data. One possible decision tree for the *WillWait* problem is shown in Figure-4.

1 Decision Trees

A decision tree can be viewed as a function that maps a vector valued input to a single output or "decision" value. The output/decision value is real-valued or continuous when the tree is used for regression and is discrete valued in the case of a classification tree.

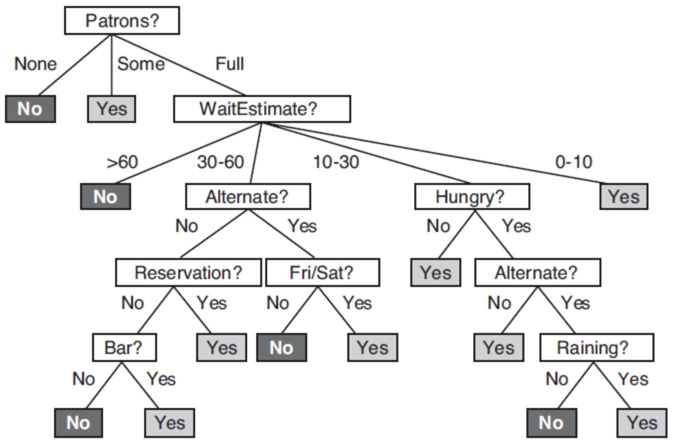


Figure 4: A decision tree for the *WillWait* task

1.0.1 How does it work?

- A decision tree maps the input to the final decision value by performing a sequence of tests on the different feature values.
- For a given data instance, each internal node of the tree tests a single feature value (later we will discuss how this can be generalized to utilize multiple features) to select one of its child nodes.
- This process continues till we reach a leaf node which assigns the final decision value to the instance.

Predicting the label for a data instance using a decision tree: Given an instance:

$$x_1 = (\text{'Yes'},\text{'No'},\text{'Yes'},\text{'Yes'},\text{'Full'},\text{'\$\$\$'},\text{'No'},\text{'No'},\text{'French'},10-30)$$

and the decision tree in Figure-4, we can predict the label as:

- Start at the root and follow the branch corresponding to the value of the feature that the root tests (*Patrons*). Since the given instance has a value of 'full' for this feature we follow the corresponding branch and reach the next node which tests for the *WaitEstimate* feature.
- Repeat this process till a leaf node is reached. For the given instance this results in a decision of 'No'.

Remark: It should be noted here that by changing the order in which the features are tested we can end up with an entirely different tree which maybe shorter or longer than the one in Figure-4. Similarly, not all features are required to be used in the tree, for example the tree does not test the *Type* or *Price* features.