Decision Tree

Decision Tree

- **Decision tree is one of the most popular classification type of algorithm**

- **Classification is done by tree like structures that have different test criteria for a variable at each of the nodes**

- **New leaves are generated based on the results of the tests at the nodes**

- **Decision Tree is a supervised learning system in which classification rules are constructed from the decision tree**

Decision Tree

ID3, stands for "Inductive Dichotomizer", this is a non-incremental algorithm, which means that it derives its classes from a fixed set of training data

Incremental algorithms can revise developed concepts if necessary, using a new sample, but the decision trees ID3 algorithm in contrast, is an inductive type

In inductive type algorithms specific classes once created, are expected to work for all future instances or data with the same dimensions

ID3 is a greedy algorithm, which constructs a decision tree in the top down recursive manner, and never checks back on its previous decisions

Concept Learning Systems

ID3 is an improved version of a Concept Learning System (CLS)

CLS Algorithm:

Step 1:

If

All instances in training dataset, C are positive

then

Create YES node and halt

If

All instances in training dataset, C are negative

then

create NO node and halt

Otherwise select a feature, F with values $v_1, v_2, \ldots v_n$ and create a decision node.

Step 2:

Partition the training instances in C into subset C_1, C_2, …, C_n according to the value of V

Step 3:

Apply the algorithm recursively to each of the sets C_i

Decision Trees

- Decision trees represent attribute-based descriptions of concepts:
  - We assume a universe of objects of interest (e.g. the set of all animals)
  - A “concept” is defined by the subset of objects which satisfy it (e.g. the concept of a “fox”, as defined by the set of all foxes)
  - Each object has a number of attributes, or properties (e.g. “colour”, “has_fur”, “eats_meat”, etc.)
  - The decision tree defines the concept in terms of the values of these attributes that an example must have
- Often, decision trees define Boolean Functions: Input to the tree is an object described by a set of properties; output is “Yes/No”

Decision Trees

- **Each internal node in the tree corresponds to a test of the value of one attribute, with branches to each of the possible values of that test.**
- **Each leaf node specifies the Boolean output of the tree if that point is reached.**
- **The aim is to learn a definition of the *goal predicate* for the problem at hand.**
  - **The goal predicate can be seen as a logical rule defining the concept – equivalent to the decision tree.**

Example – Restaurant Problem

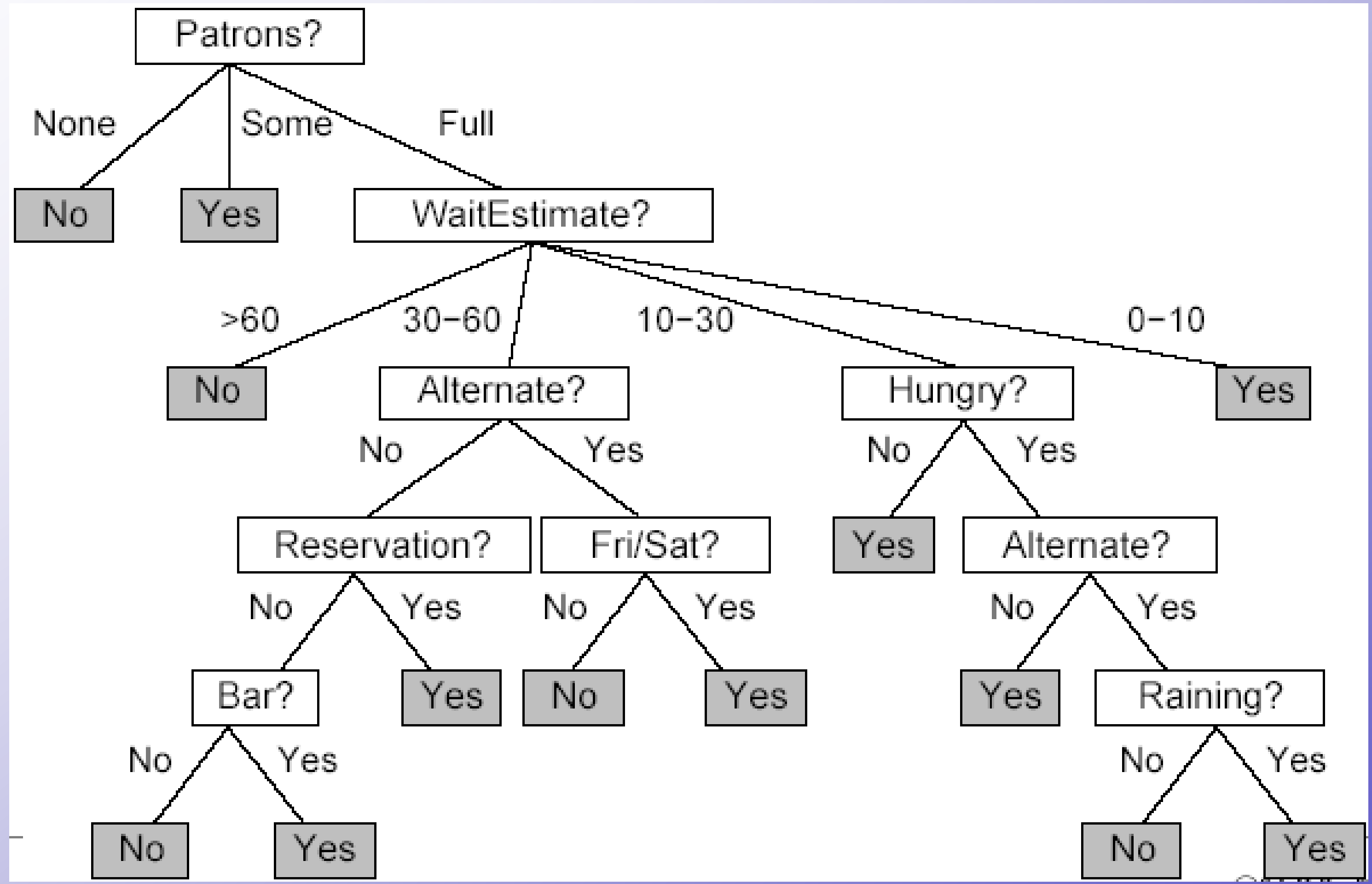
- **Should we wait for a table at a restaurant?**
- **Input Attributes:**
  - **Alternate – Suitable alternative restaurant nearby?**
  - **Bar – Is there one to wait in?**
  - **Fri/Sat – Is it Friday or Saturday?**
  - **Hungry – Are we?**
  - **Patrons – How many (None, Some, Full)?**
  - **Price – How pricey is this place?**
  - **Raining – Is it?**
  - **Reservation – Have we made one?**
  - **Type – Of food (French, Italian, Burger, . . .).**
  - **WaitEstimate – How long, estimated by the host.**
- **Determine output attribute: *WillWait*.**

Some Training Examples

Example	Attributes										Goal WillWait?
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	
X1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0-10	**Yes**
X2	Yes	No	No	Yes	Full	$	No	No	Thai	30-60	**No**
X3	No	Yes	No	No	Some	$	No	No	Burger	0-10	**Yes**
X4	Yes	No	Yes	Yes	Full	$	No	No	Thai	10-30	**Yes**
X5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	**No**
X6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0-10	**Yes**
X7	No	Yes	No	No	None	$	Yes	No	Burger	0-10	**No**
X8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0-10	**Yes**
X9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	**No**
X10	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10-30	**No**
X11	No	No	No	No	None	$	No	No	Thai	0-10	**No**
X12	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30-60	**Yes**

- *We need a collection of example objects (rows in a table), each with values for all the attributes and the values of the "output class"*
- *Positive (Yes) and Negative (No) examples are needed*

Possible Decision Tree



How to Construct a Decision Tree

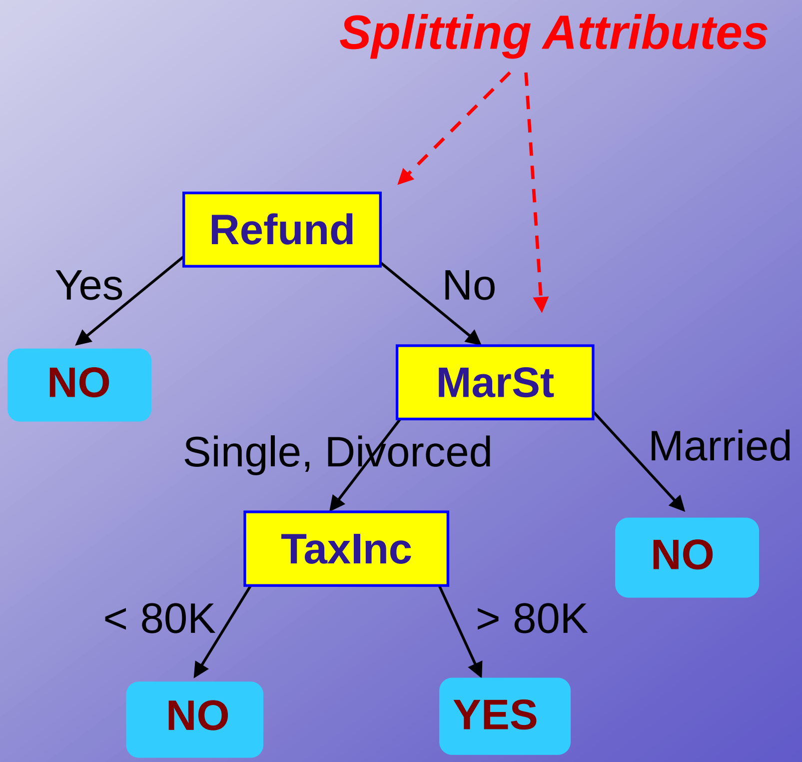
- Given a table of positive & negative examples, each with an output class, how do we produce a decision tree for this training data set?
- Observe that each non-leaf node of the tree gives a test on one of the attributes in the dataset.
- Any example will satisfy only one of the options given.
- Therefore, the tree splits the training dataset between the branches of the tree according to attribute values.
- So, we start at the root node with all the training examples, and choose an attribute to split on.
- Each of the daughter nodes gives a subset of the training set.
- Then we can do exactly the same thing with each daughter node, until the examples remaining are all “Yes” or all “No”, or we run out of attributes.

Example of a Decision Tree

categorical categorical continuous class

Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

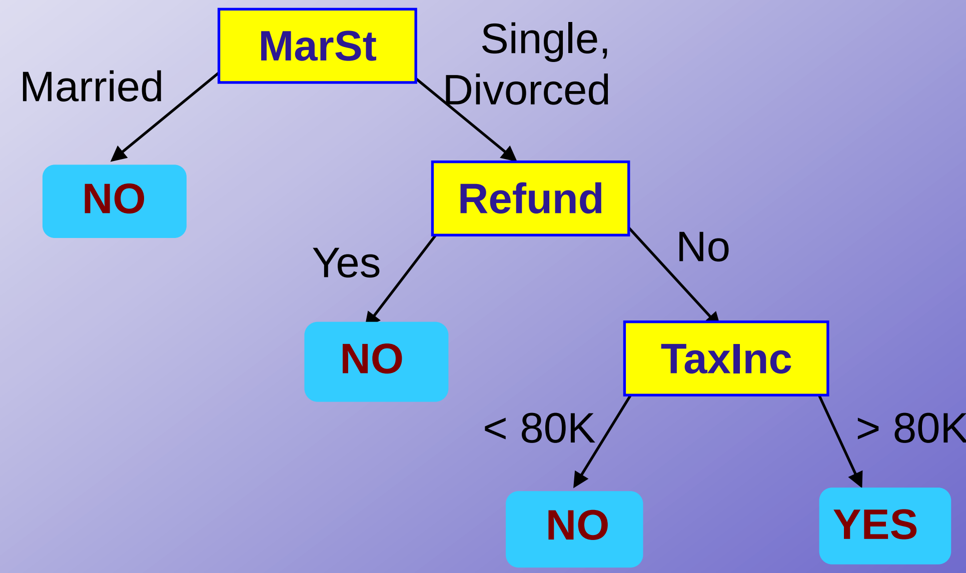
Training Data



Model: Decision Tree

Another Example of Decision Tree

	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



There could be more than one tree that fits the same data!

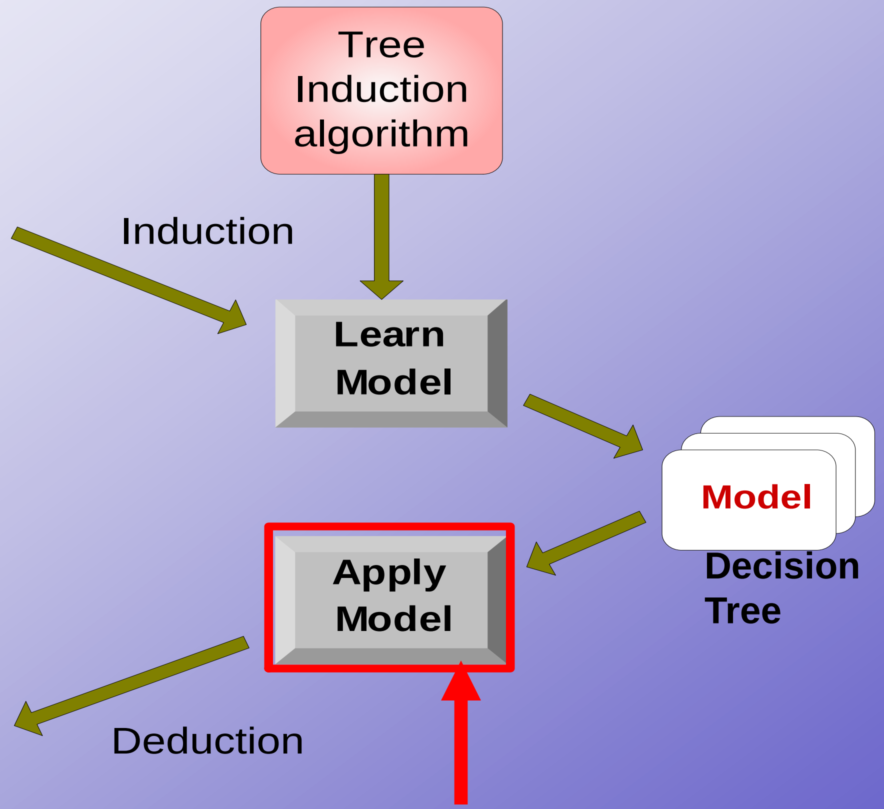
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

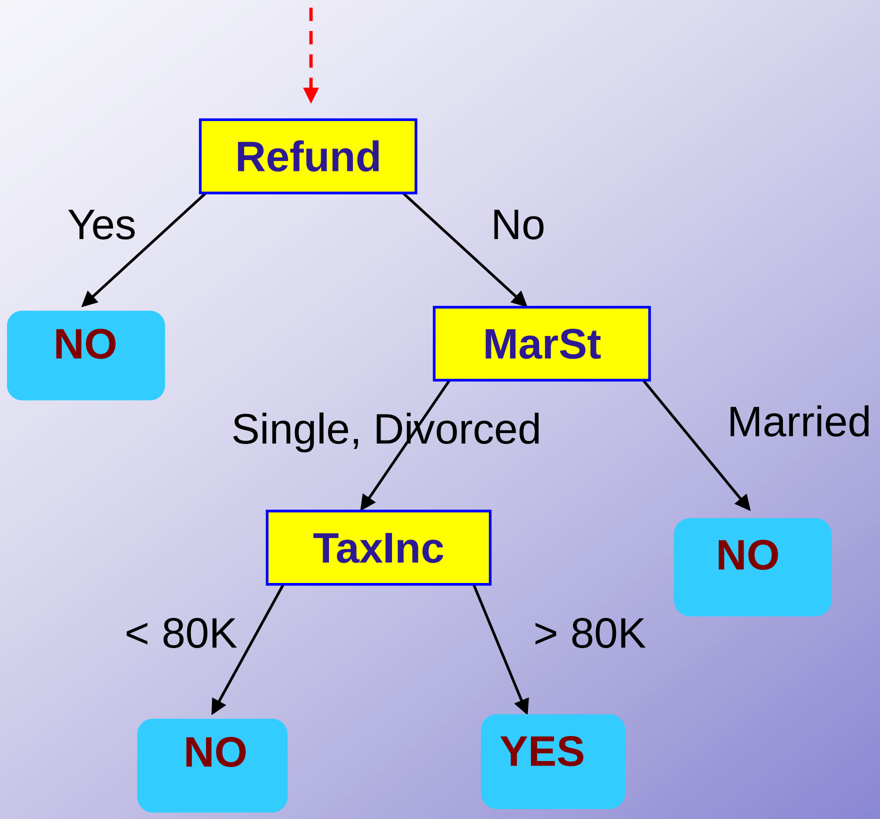
Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Apply Model to Test Data

Start from the root of tree.



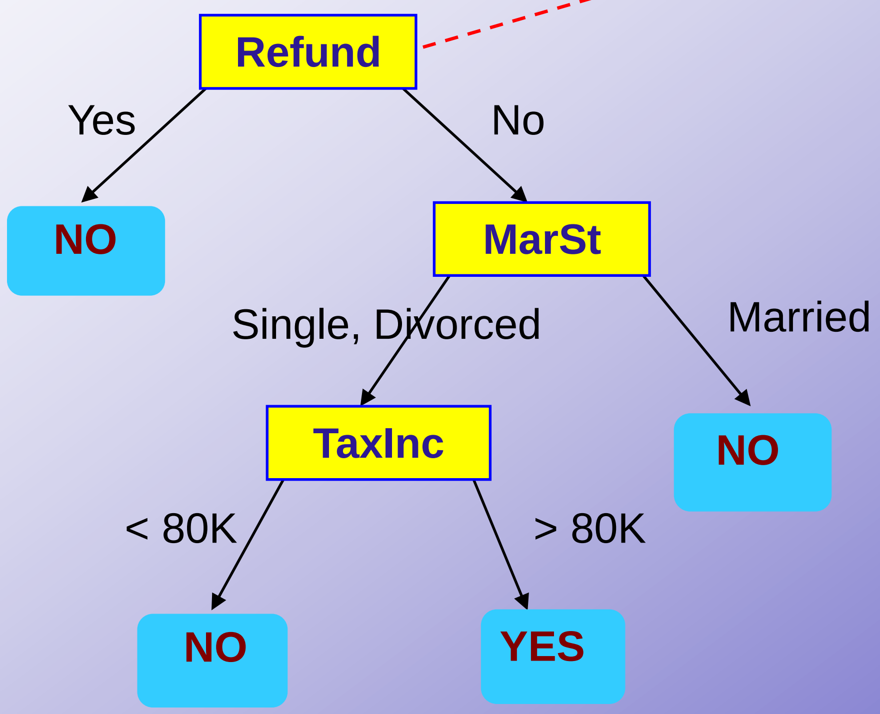
Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Test Data

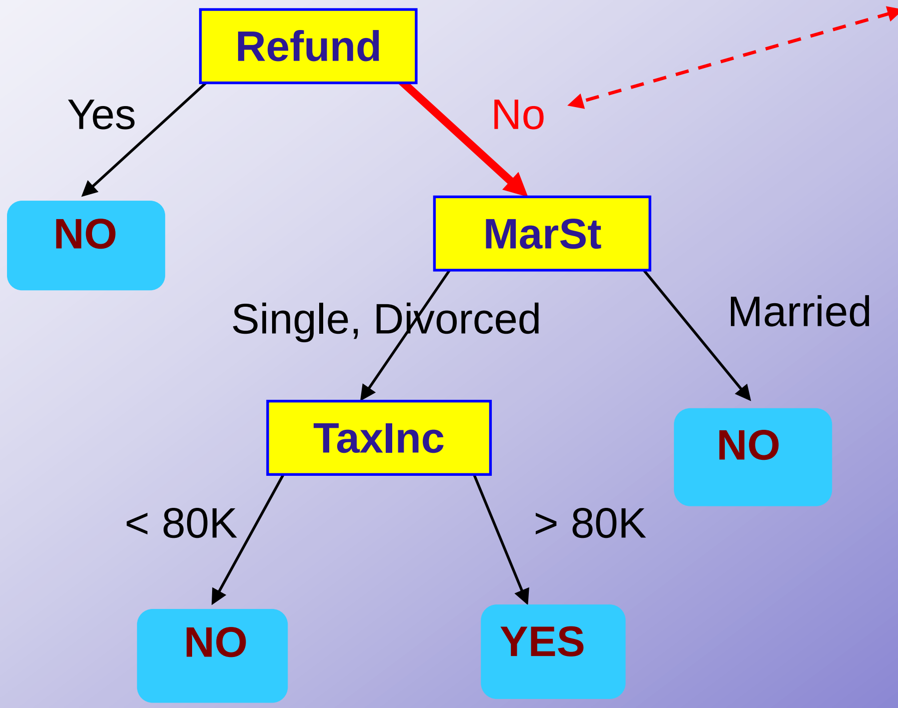
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**



Apply Model to Test Data

Test Data

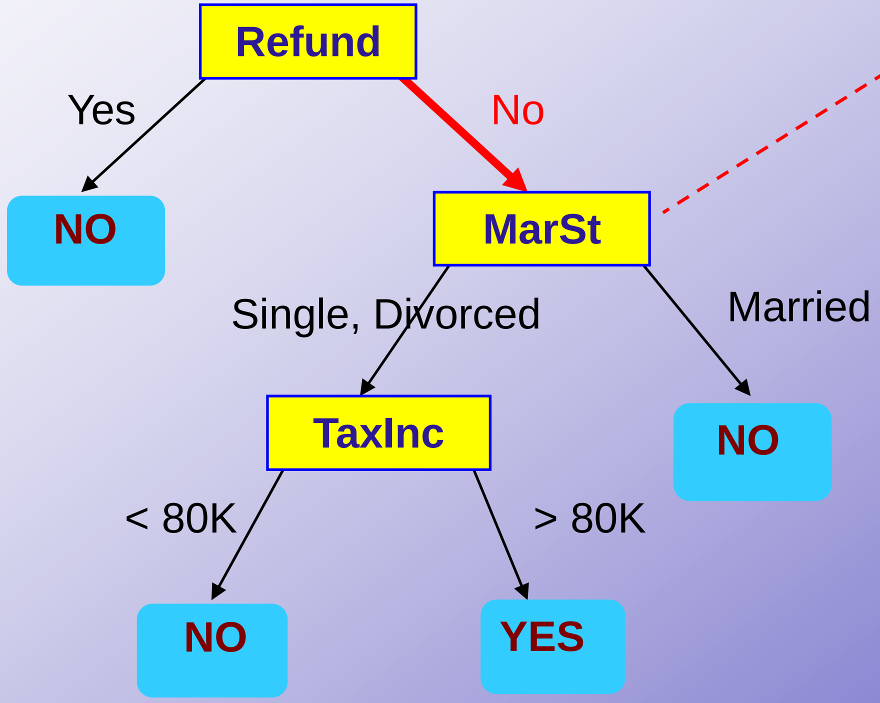
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	**?**



Apply Model to Test Data

Test Data

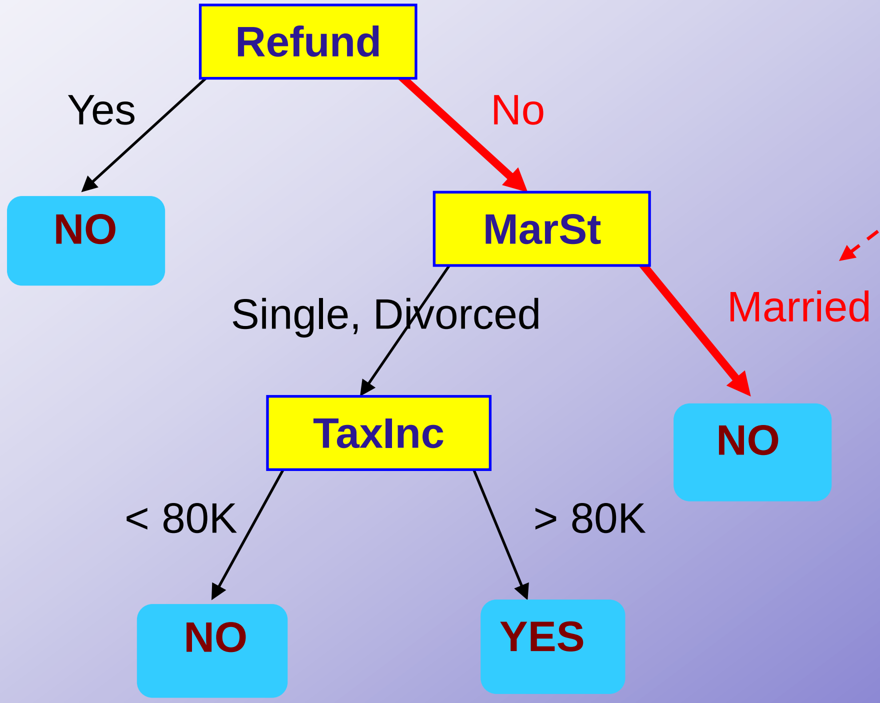
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

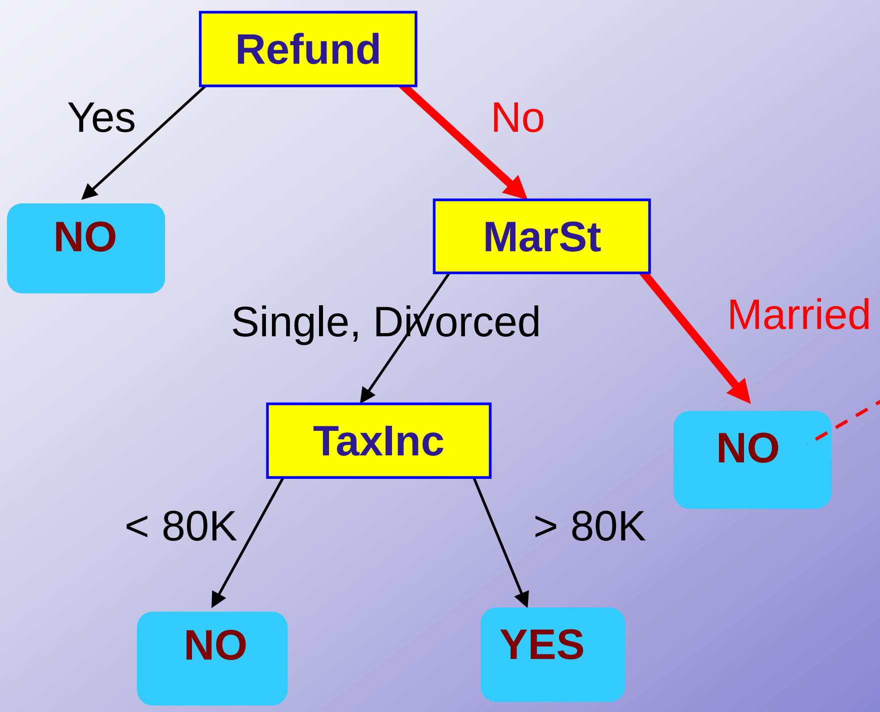
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Apply Model to Test Data

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

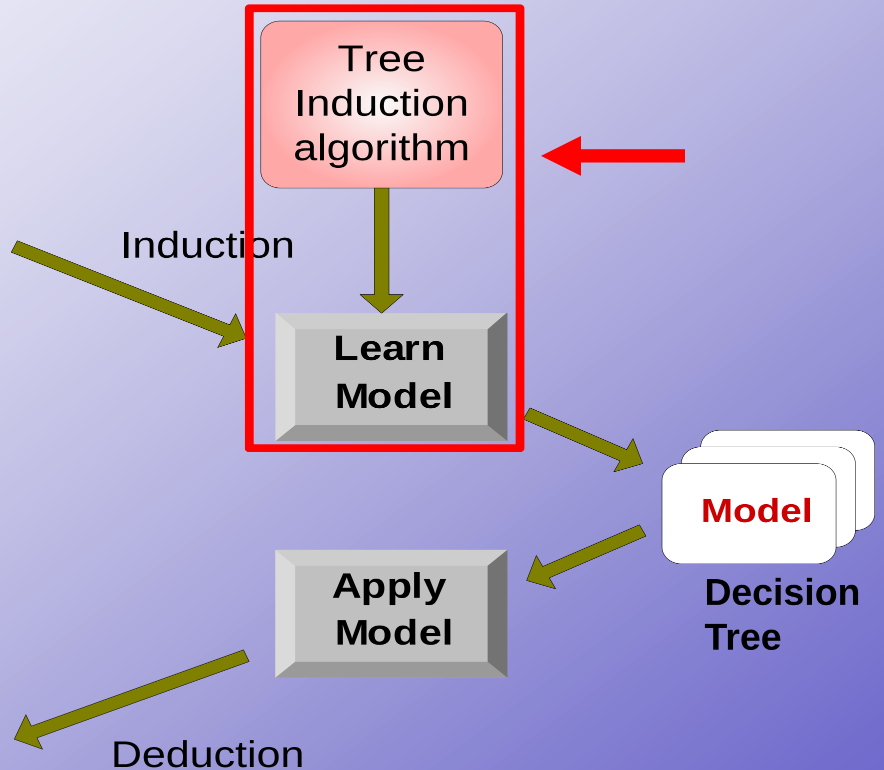


Assign Cheat to “No”

Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

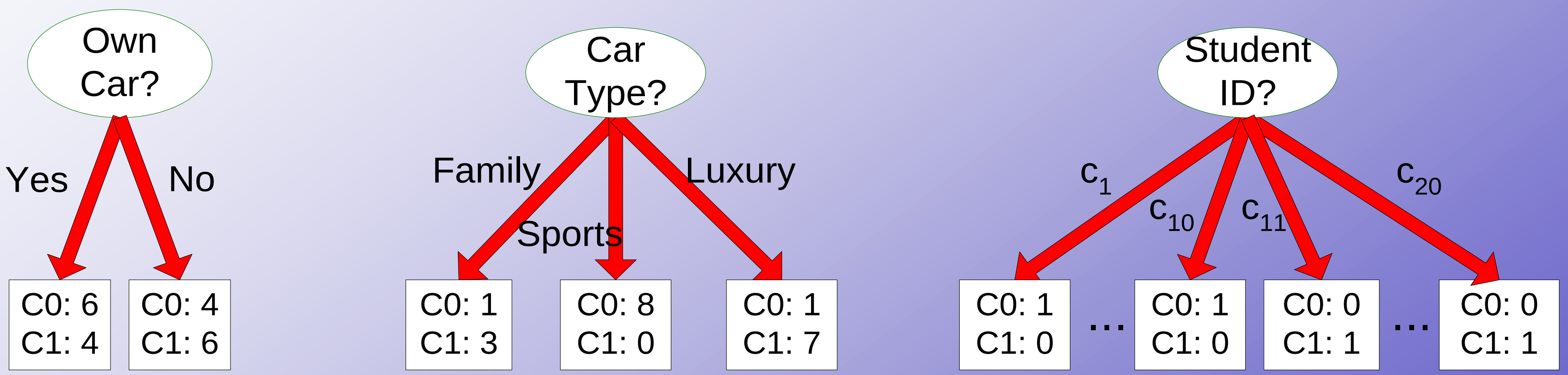


Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set

How to determine the Best Split

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

How to determine the Best Split

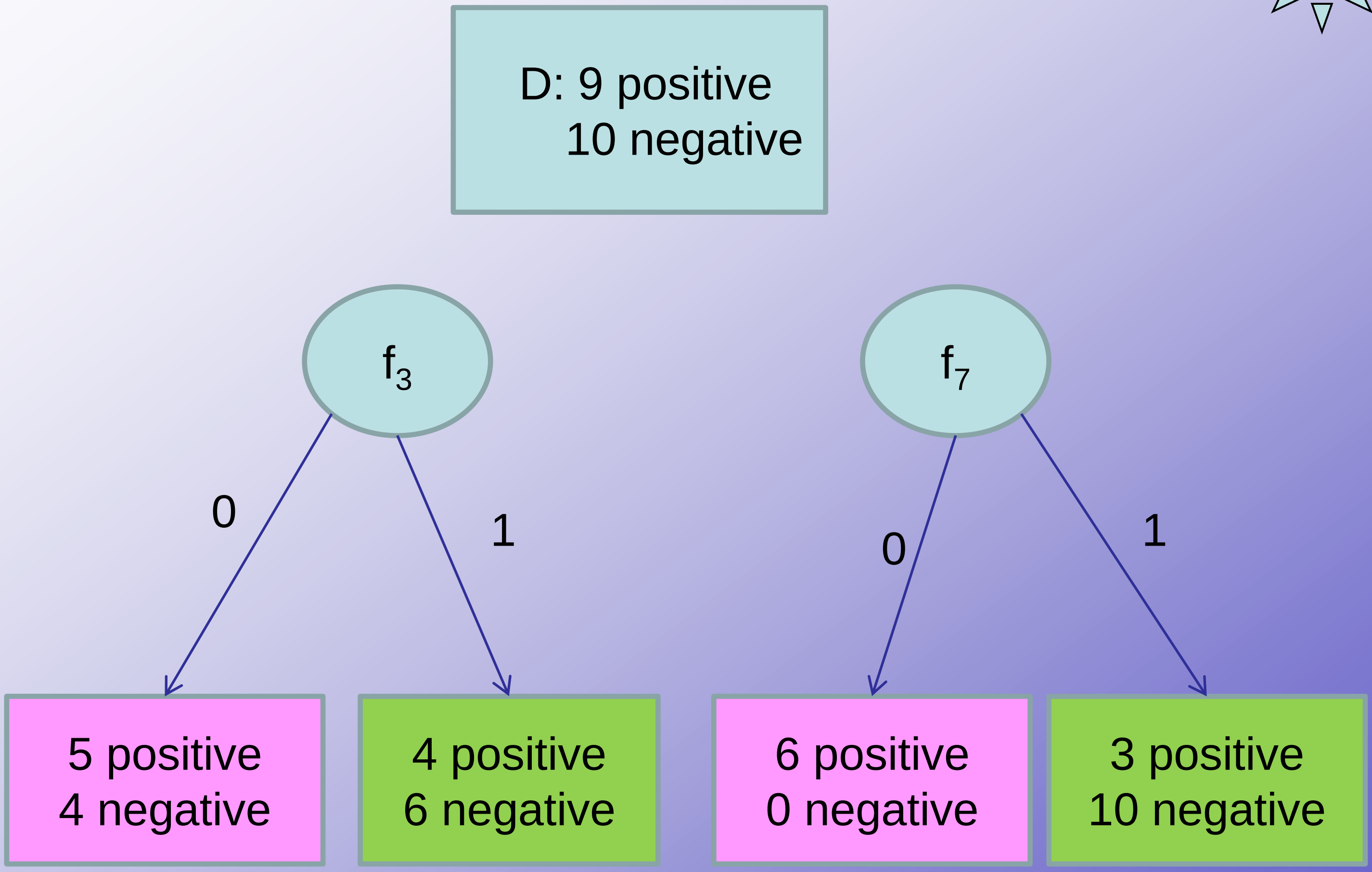
- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

C0: 5	C0: 9
C1: 5	C1: 1
Non-homogeneous,	**Homogeneous,**
High degree of impurity	**Low degree of impurity**

Measures of Node Impurity

- Gini Index
- Entropy
- Misclassification error

Let’s Split

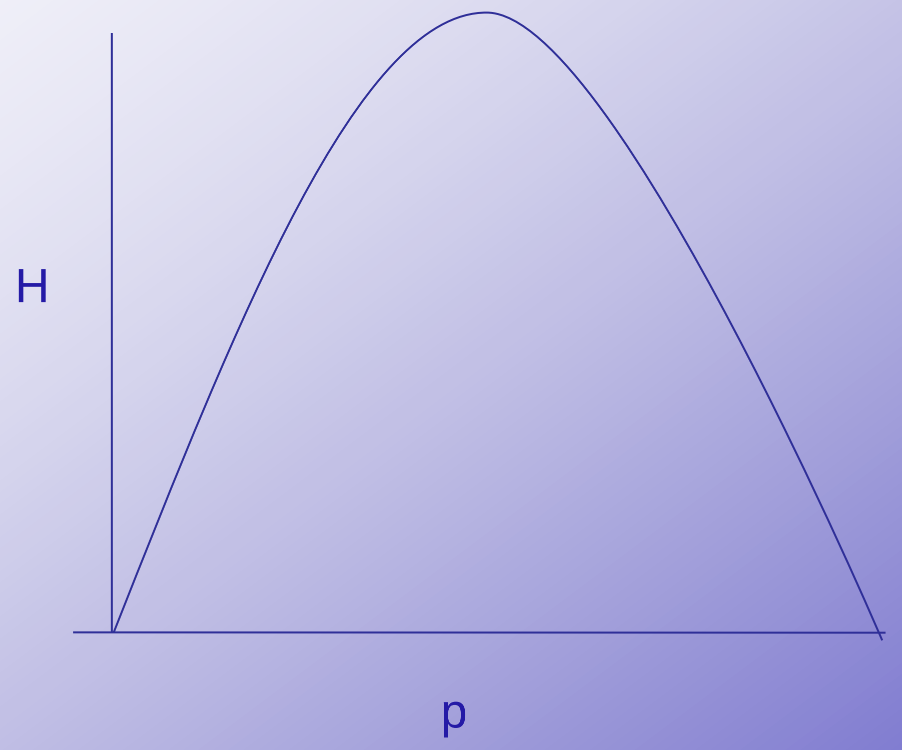


Selecting f_3 doesn’t help much as mixture is still with almost the same ratio compared with f_7 which seems to be a good selection

Entropy

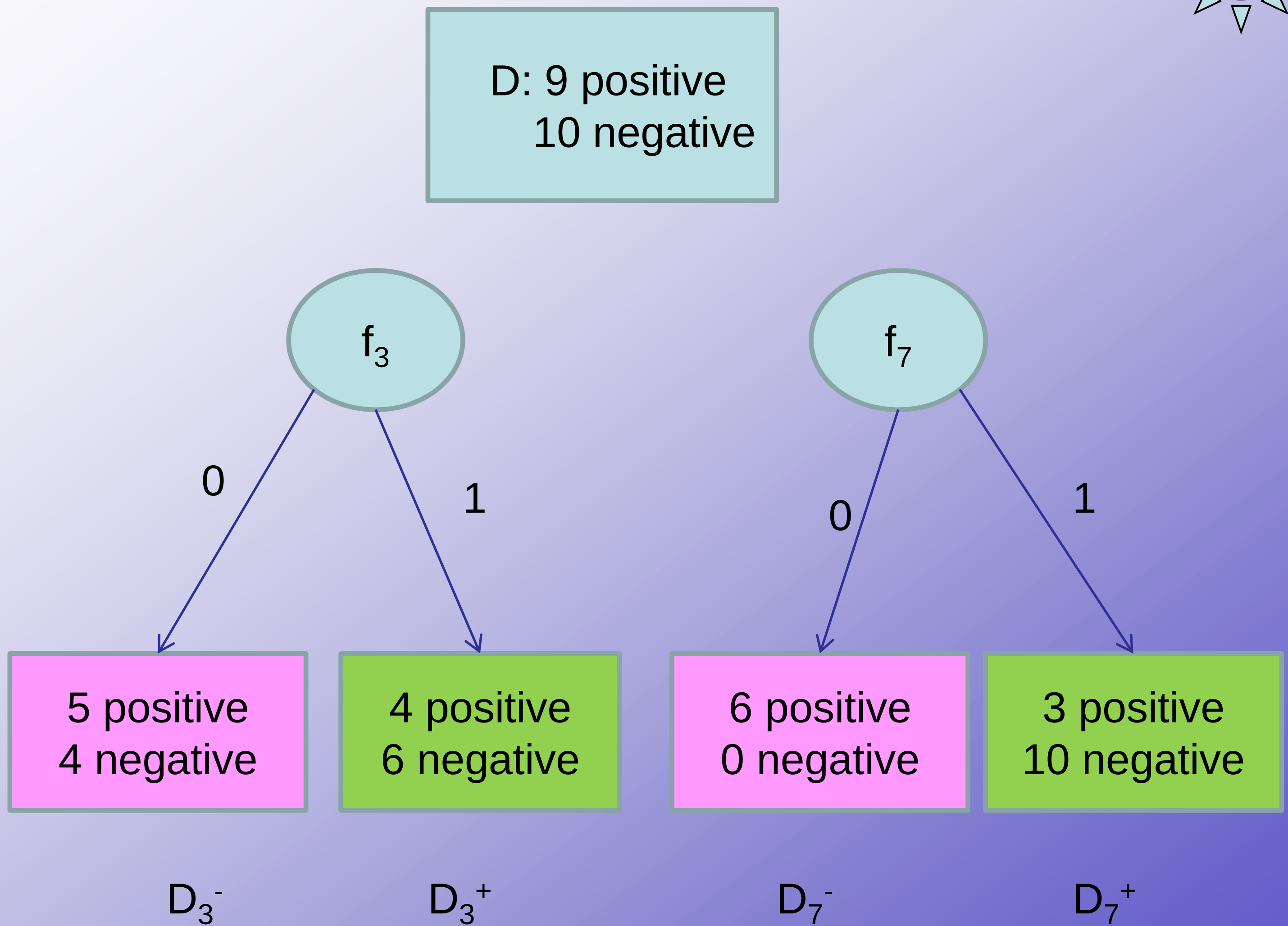
p := proportion of positive examples in data set

$$H = -p \log_2 p - (1-p) \log_2 (1-p)$$

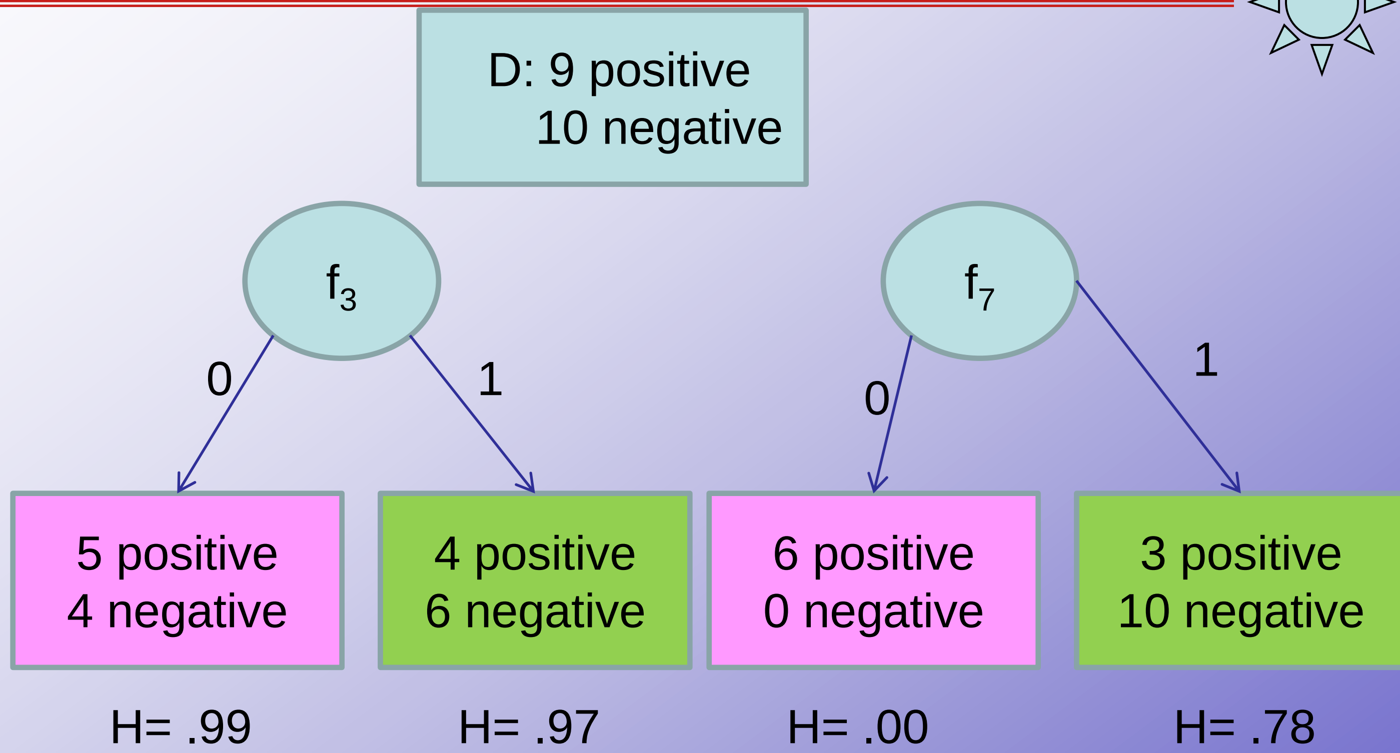


$0 \log_2 0 = 0$

Let’s Split


D: 9 positive
10 negative
f_3
0
1
f_7
0
1
5 positive
4 negative
4 positive
6 negative
6 positive
0 negative
3 positive
10 negative
D_3^-
D_3^+
D_7^-
D_7^+

Let's Split

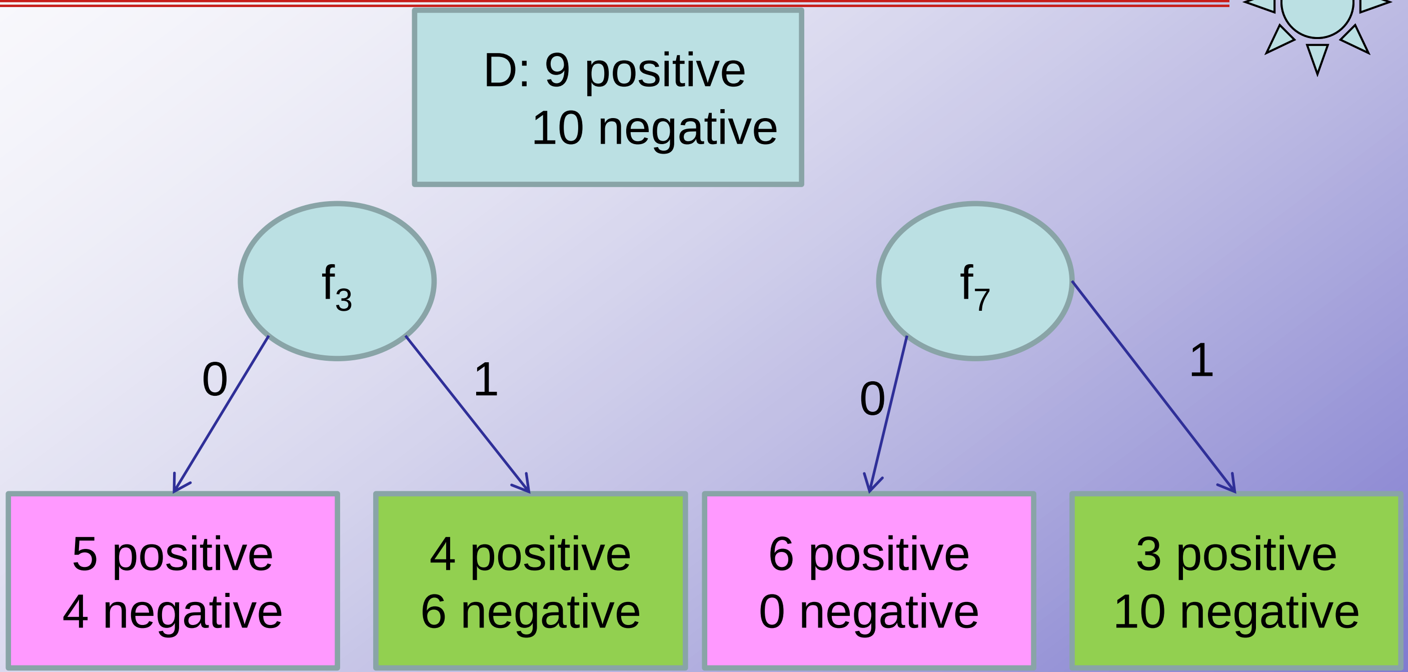


$$AE\ (j) = P_j\ H\ (D_j^{+}) + (1-p_j)\ H(D_j^{-})$$

% of D with f_j=1

subset of D with f_j=1

Let’s Split



H= .99 H= .97 H= .00 H= .78

$$AE = (9/19) * .99 + (10/19) * .97 = .98$$

$$AE = (6/19) * .00 + (13/19) * .78 = .53$$

Algoritm

```
BuildTree(Data)
  if all elements of Data have the same y value, then
      MakeLeafNode(y)
  else
      feature := PickBestFeature(Data)
      MakeInternalNode(feature,
          BuildTree(SelectFalse(Data,Feature)),
          BuildTree(SelectTrue(Data,Feature)))
```

- Best feature minimizes average entropy of data in the children

Stopping

•Stop recursion if data contains only multiple instances of the same x with different y values

❖Make leaf node with output equal to the y value that occurs in the majority of the cases in the data

•Consider stopping to avoid over-fitting when:

❖Entropy of a data set is below some threshold
❖Number of elements in a data set is below threshold
❖Best next split does not decrease average entropy

Example

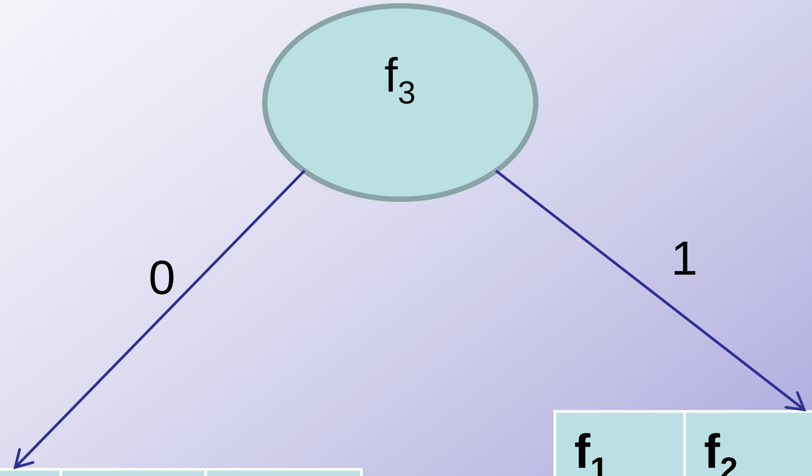
Entropy for each feature

- $AE_1 = .92$
- $AE_2 = .92$
- $AE_3 = .81$
- $AE_4 = 1$

The best feature for split is f_3

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
1	0	0	1	0
0	1	1	1	0

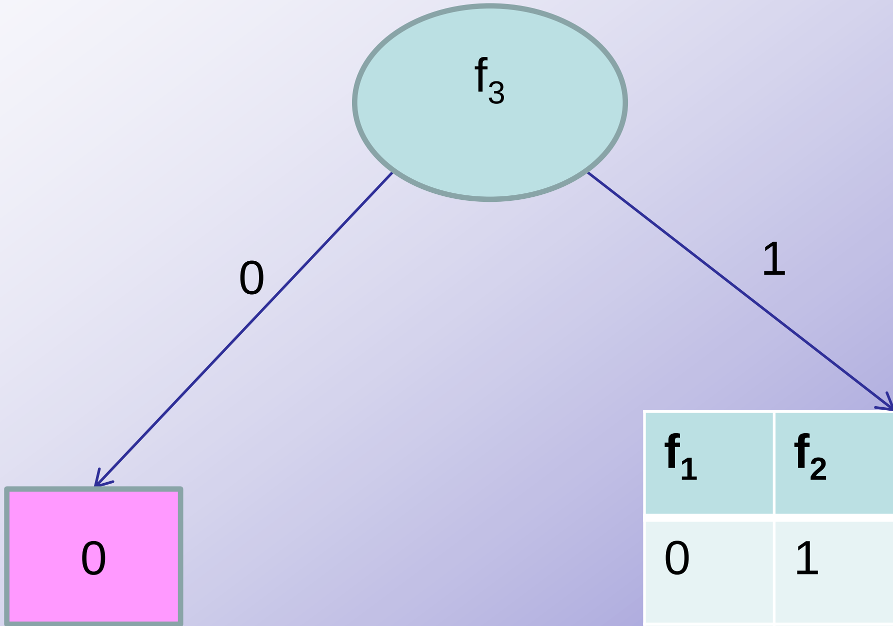
Simulation



f_1	f_2	f_3	f_4	y
1	0	0	1	0

f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
0	1	1	1	0

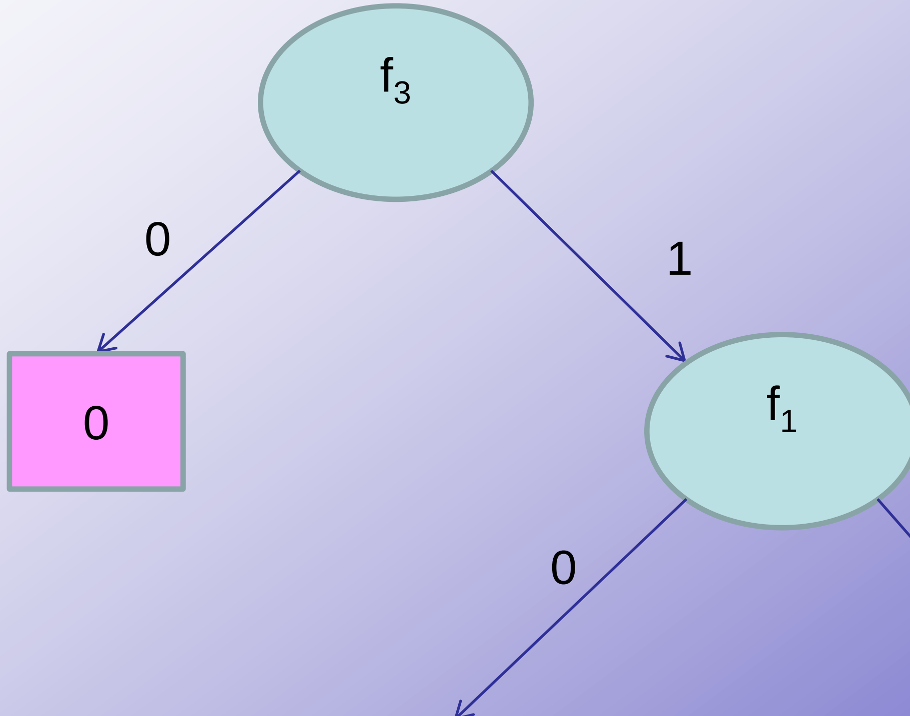
Simulation



f_1	f_2	f_3	f_4	y
0	1	1	0	0
1	0	1	1	1
1	1	1	0	1
0	0	1	1	1
0	1	1	1	0

- $AE_1 = .55$
- $AE_2 = .55$
- $AE_4 = .95$
- Arbitrarily decide to split on f_1

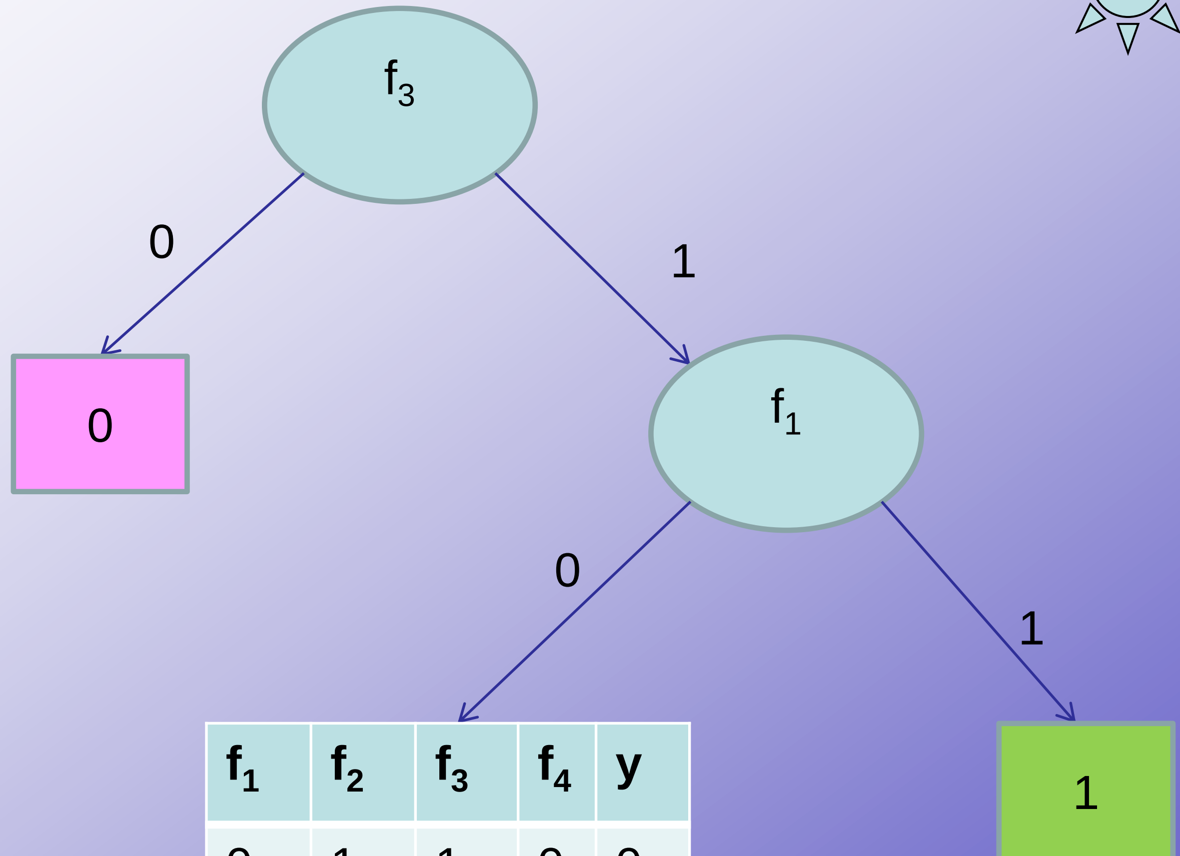
Simulation



f_1	f_2	f_3	f_4	y
0	1	1	0	0
0	0	1	1	1
0	1	1	1	0

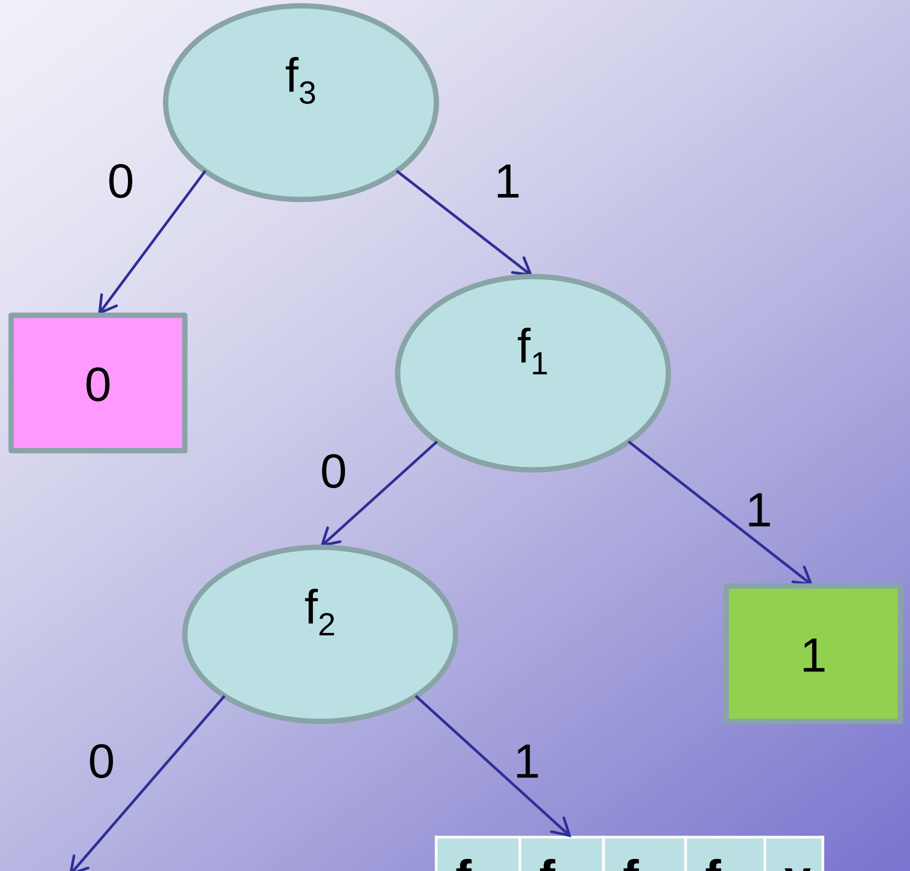
f_1	f_2	f_3	f_4	y
1	0	1	1	1
1	1	1	0	1

Simulation



f_1	f_2	f_3	f_4	y
0	1	1	0	0
0	0	1	1	1
0	1	1	1	0

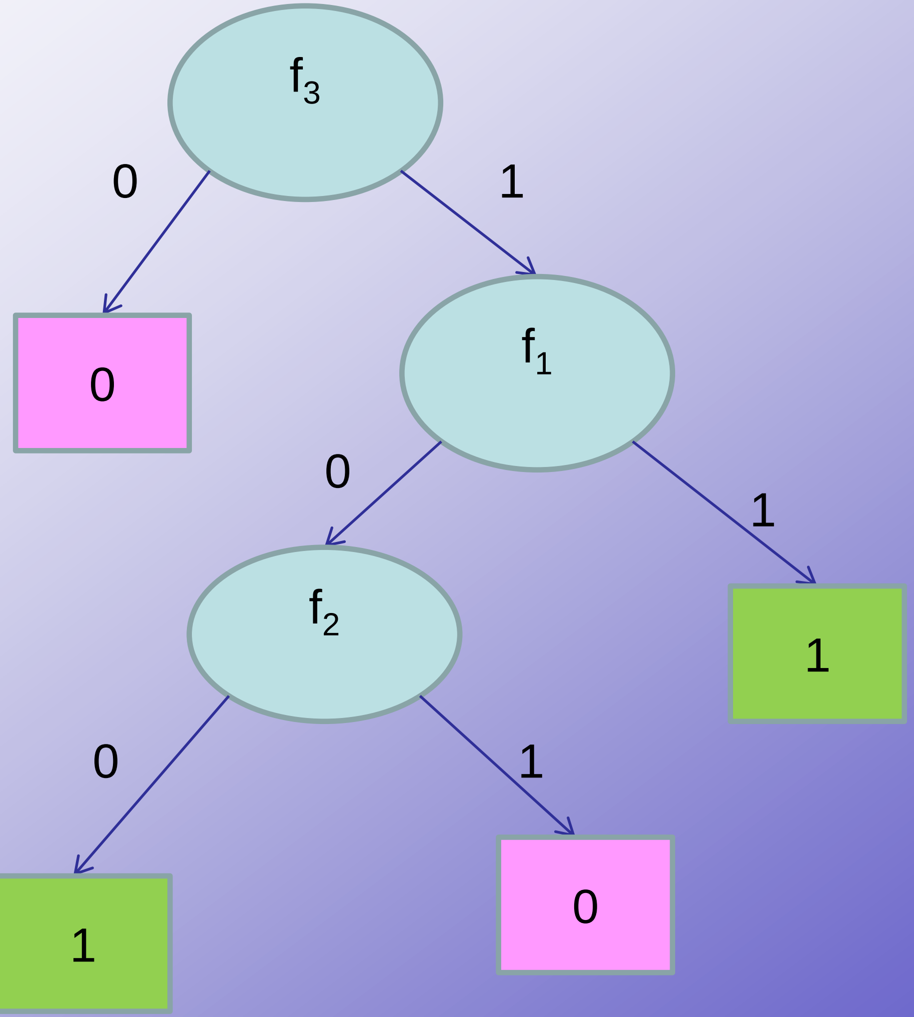
Simulation



f1	f2	f3	f4	y
0	0	1	1	1

f1	f2	f3	f4	y
0	1	1	0	0
0	1	1	1	0

Simulation



Pruning

•Best way to avoid over-fitting and not get tricked by short-term lack of progress

•Grow tree as far as possible

❖Leaves are uniform or contain a single X

•Prune the tree until it performs well on held-out data

•Amount of pruning is like epsilon in DNF algorithm

Pre pruning

Post Pruning

ML

•Making useful predictions in (usually corporate) applications

•Decision trees very popular because :

❖Easy to implement

❖Efficient (even on huge data sets)

❖Easy for humans to understand resulting hypotheses