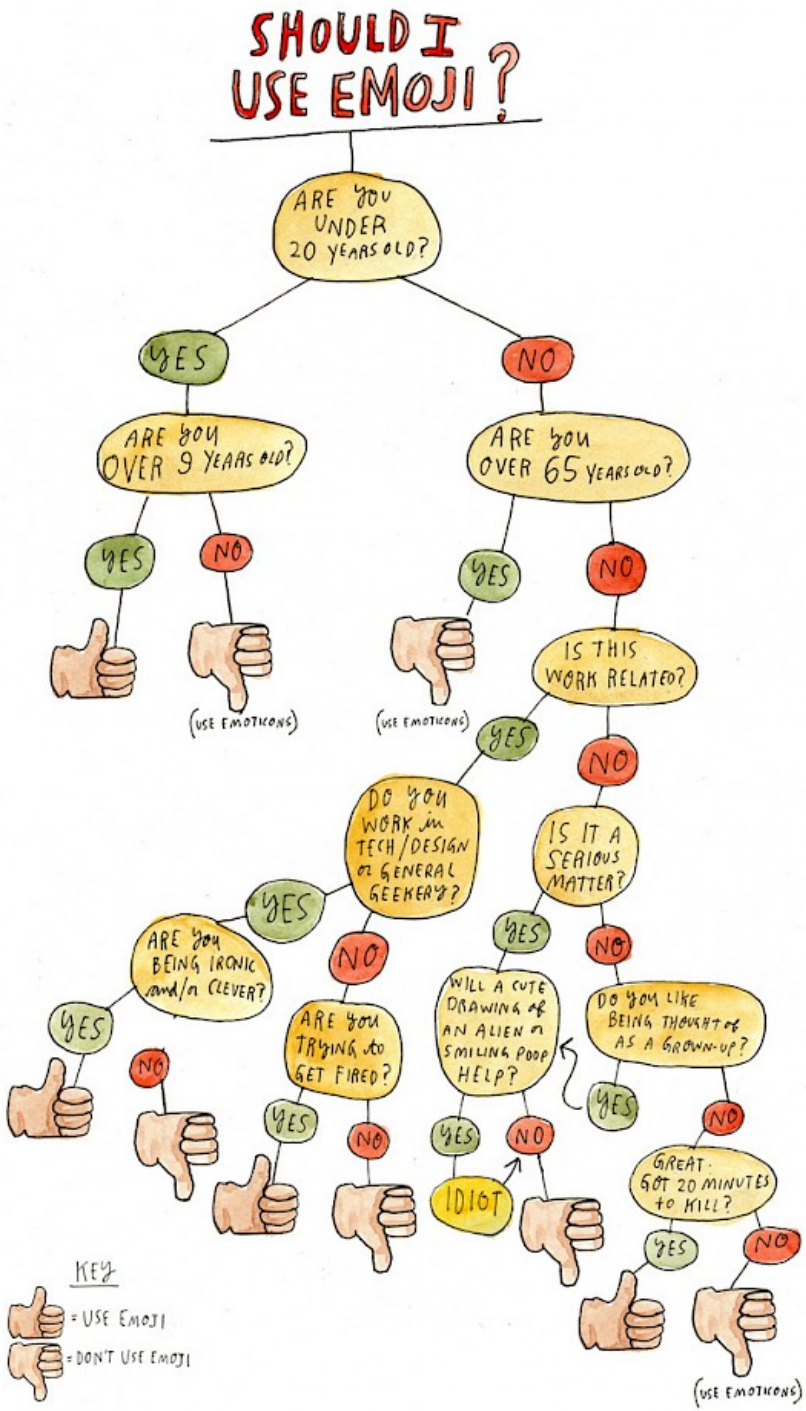
Overview


SHOULD I USE EMOJI?
ARE YOU UNDER 20 YEARS OLD?
YES
NO
ARE YOU OVER 9 YEARS OLD?
ARE YOU OVER 65 YEARS OLD?
YES
NO
YES
NO
(USE EMOTICONS)
(USE EMOTICONS)
IS THIS WORK RELATED?
YES
NO
DO YOU WORK in TECH/DESIGN or GENERAL GEEKERY?
IS IT A SERIOUS MATTER?
YES
NO
YES
NO
ARE YOU BEING IRONIC and/or CLEVER?
ARE YOU TRYING to GET FIRED?
WILL A CUTE DRAWING of AN ALIEN a SMILING POOP HELP?
DO YOU LIKE BEING THOUGHT of AS A GROWN-UP?
YES
NO
YES
NO
YES
NO
YES
NO
IDIOT
GREAT. GOT 20 MINUTES to KILL?
YES
NO
(USE EMOTICONS)
KEY
= USE EMOJI
= DON'T USE EMOJI

Lenses dataset

Presbyopic	Young	Spectacle prescription	Astigmatic	Tear production rate	Can use contact lenses
No	Yes	Myope	No	Normal	Yes
No	Yes				No
No	Yes				No
No	Yes				Yes
No	No				No
No	No				Yes
No	No				Yes
No	No	Hypermetrope	No	Reduced	No
No	No	Hypermetrope	Yes	Reduced	No
No	No	Hypermetrope	Yes	Normal	No
Yes	No	Myope	No	Normal	No
Yes	No	Hypermetrope	Yes	Normal	No
No	Yes	Myope	No	Reduced	???
No	Yes	Myope	Yes	Normal	???
No	Yes	Hypermetrope	Yes	Reduced	???
No	Yes	Hypermetrope	Yes	Normal	???
No	No	Myope	Yes	Reduced	???
No	No	Hypermetrope	No	Normal	???
Yes	No	Myope	No	Reduced	???
Yes	No	Myope	Yes	Reduced	???
Yes	No	Myope	Yes	Normal	???
Yes	No	Hypermetrope	No	Reduced	???
Yes	No	Hypermetrope	No	Normal	???
Yes	No	Hypermetrope	Yes	Reduced	???

This dataset is about 24 people who want to start using contact lenses and visit the optician (1 row=1 person)

Lenses dataset

Presbyopic	Young	Spectacle prescription	Astigmatic	Tear production rate	Can use contact lenses
No	Yes	Myope	No	Normal	Yes
No	Yes				No
No	Yes				No
No	Yes				Yes
No	No				No
No	No				Yes
No	No				Yes
No	No	Hypermetrope	No	Reduced	No
No	No	Hypermetrope	Yes	Reduced	No
No	No	Hypermetrope	Yes	Normal	No
Yes	No				No
Yes	No				No
No	Yes				???
No	Yes				???
No	Yes				???
No	Yes				???
No	No				???
No	No	Hypermetrope	No	Normal	???
Yes	No	Myope	No	Reduced	???
Yes	No	Myope	Yes	Reduced	???
Yes	No	Myope	Yes	Normal	???
Yes	No	Hypermetrope	No	Reduced	???
Yes	No	Hypermetrope	No	Normal	???
Yes	No	Hypermetrope	Yes	Reduced	???

This dataset is about 24 people who want to start using contact lenses and visit the optician (1 row=1 person)

Optician gathers information (columns 1-5) and decides if the person can use contact lenses (last column)

Lenses dataset

Presbyopic	Young	Spectacle prescription	Astigmatic	Tear production rate	Can use contact lenses
No	Yes	Myope	No	Normal	Yes
No	Yes	Myope	Yes	Reduced	No
No	Yes	Hypermetrope	No	Reduced	No
No	Yes	Hypermetrope	No	Normal	Yes
No	No	Myope	No	Reduced	No
No	No	Myope	No	Normal	Yes
No	No	Myope	Yes	Normal	Yes
No	No	Hypermetrope	No	Reduced	No
No	No	Hypermetrope	Yes	Reduced	No
No	No	Hypermetrope	Yes	Normal	No
Yes	No	Myope	No	Normal	No
Yes	No	Hypermetrope	Yes	Normal	No
No	Yes	Myope	No	Reduced	???
No	Yes	Myope	Yes	Normal	???
No	Yes	Hypermetrope	Yes	Reduced	???
No	Yes	Hypermetrope	Yes	Normal	???
No	No	Myope	Yes	Reduced	???
No	No	Hypermetrope	No	Normal	???
Yes	No	Myope	No	Reduced	???
Yes	No	Myope	Yes	Reduced	???
Yes	No	Myope	Yes	Normal	???
Yes	No	Hypermetrope	No	Reduced	???
Yes	No	Hypermetrope	No	Normal	???
Yes	No	Hypermetrope	Yes	Reduced	???

Lenses dataset

Presbyopic	Young	Spectacle prescription	Astigmatic	Tear production rate	Can use contact lenses
No	Yes	Myope	No	Normal	Yes
No	Yes	Myope	Yes	Reduced	No
No	Yes	Hypermetrope	No	Reduced	No
No	Yes	Hypermetrope	No	Normal	Yes
No	No	Myope	No	Reduced	No
No	No	Myope	No	Normal	Yes
No	No	Myope	Yes	Normal	Yes
No	No	Hypermetrope	No	Reduced	No
No	No	Hypermetrope	Yes	Reduced	No
No	No	Hypermetrope	Yes	Normal	No
Yes	No	Myope	No	Normal	No
Yes	No	Hypermetrope	Yes	Normal	No
No	Yes	Hypermetrope	Yes	Normal	???

Lenses dataset

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0
0	1	1	1	0	???

Lenses dataset

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0
0	1	1	1	0	???

What would you predict for the test instance?

Lenses dataset

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0
0	1	1	1	0	???

First let's consider only the last feature (R).

How to predict?

R	L
0	1
1	0
1	0
0	1
1	0
0	1
0	1
1	0
1	0
0	0
0	0
0	0
0	???

How to predict?

R	L
0	1
1	0
1	0
0	1
1	0
0	1
0	1
1	0
1	0
0	0
0	0
0	0
0	???

Distribution

	L=0	L=1	Majority
R=0	3	4	L=1

How to predict?

R	L
0	1
1	0
1	0
0	1
1	0
0	1
0	1
1	0
1	0
0	0
0	0
0	0
0	???

Distribution

	L=0	L=1	Majority
R=0	3	4	L=1
R=1	5	0	L=0

How to predict?

R	L
0	1
1	0
1	0
0	1
1	0
0	1
0	1
1	0
1	0
0	0
0	0
0	0
0	???

Distribution

	L=0	L=1	Majority
R=0	3	4	L=1
R=1	5	0	L=0

We agreed to predict the most frequent class in the training data!

If R=0 then L=1
If R=1 then L=0

How to predict?

R	L
0	1
1	0
1	0
0	1
1	0
0	1
0	1
1	0
1	0
0	0
0	0
0	0
0	???

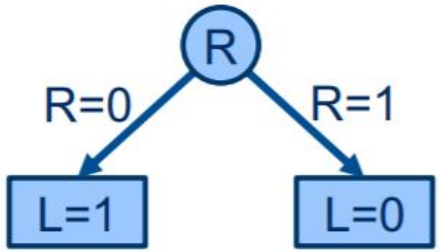
Distribution

	L=0	L=1	Majority
R=0	3	4	L=1
R=1	5	0	L=0

We agreed to predict the most frequent class in the training data!

If R=0 then L=1
If R=1 then L=0

This is called a **decision stump**, that is a decision tree with 1 decision node



How to predict in this case?

A	R	L
0	0	1
1	1	0
0	1	0
0	0	1
0	1	0
0	0	1
1	0	1
0	1	0
1	1	0
1	0	0
0	0	0
1	0	0
1	0	???

How to predict in this case?

A	R	L
0	0	1
1	1	0
0	1	0
0	0	1
0	1	0
0	0	1
1	0	1
0	1	0
1	1	0
1	0	0
0	0	0
1	0	0
1	0	???

Distribution

	L=0	L=1	Majority
A=0, R=0	1	3	L=1
A=0, R=1	3	0	L=0
A=1, R=0	2	1	L=0
A=1, R=1	2	0	L=0

How to predict in this case?

A	R	L
0	0	1
1	1	0
0	1	0
0	0	1
0	1	0
0	0	1
1	0	1
0	1	0
1	1	0
1	0	0
0	0	0
1	0	0
1	0	???

Distribution

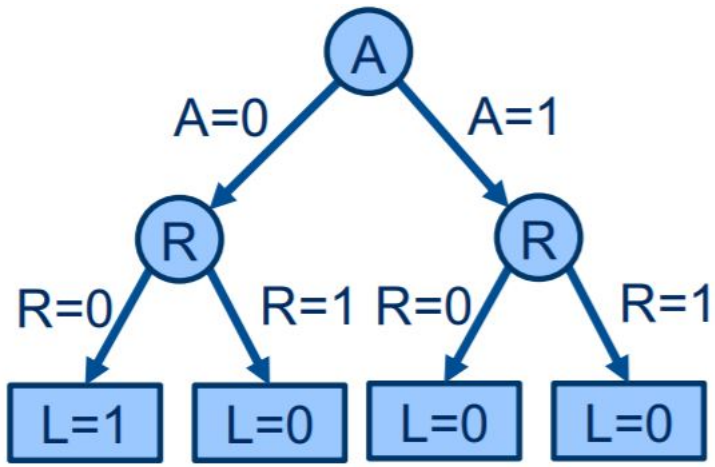
	L=0	L=1	Majority
A=0, R=0	1	3	L=1
A=0, R=1	3	0	L=0
A=1, R=0	2	1	L=0
A=1, R=1	2	0	L=0

We agreed to predict the most frequent class in the training data!

If A=0, R=0 then L=1
If A=0, R=1 then L=0
If A=1, R=0 then L=0
If A=1, R=1 then L=0

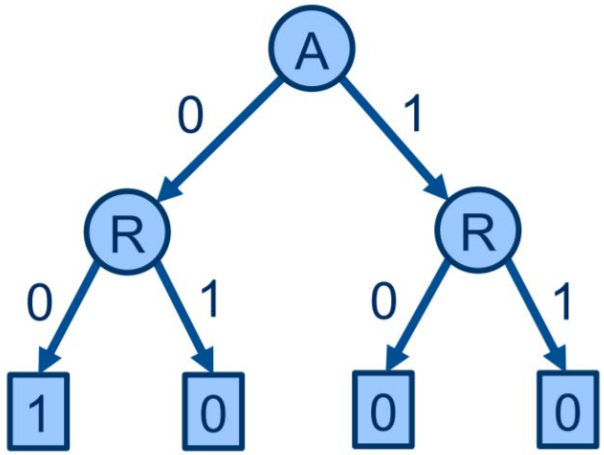
Decision Tree (features A, R)

If A=0, R=0 then L=1
If A=0, R=1 then L=0
If A=1, R=0 then L=0
If A=1, R=1 then L=0



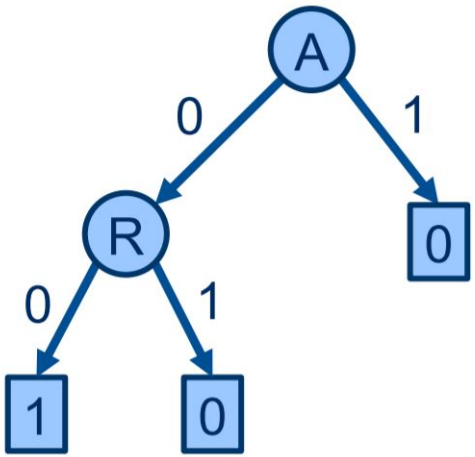
Decision Tree (features A, R)

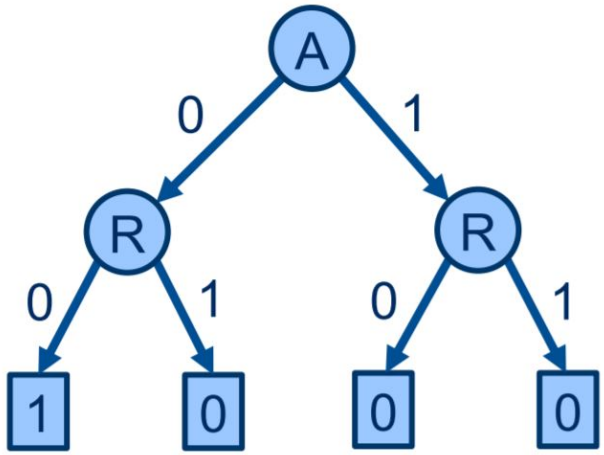
If A=0, R=0 then L=1
If A=0, R=1 then L=0
If A=1, R=0 then L=0
If A=1, R=1 then L=0



Decision Tree (features A, R)

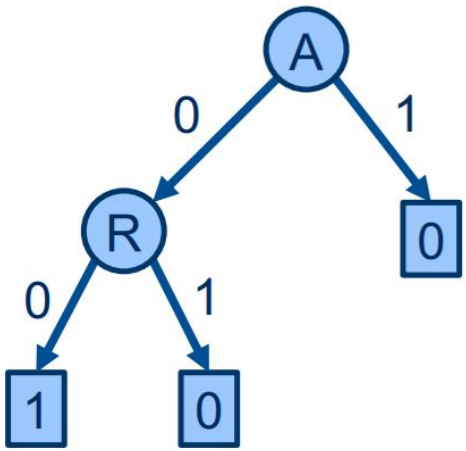
If A=0, R=0 then L=1
If A=0, R=1 then L=0
If A=1, R=0 then L=0
If A=1, R=1 then L=0

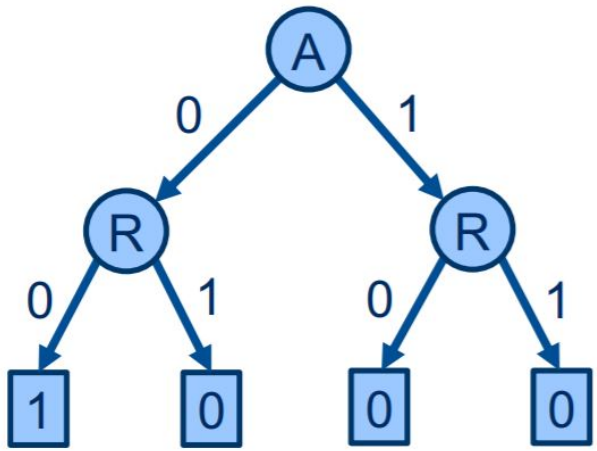


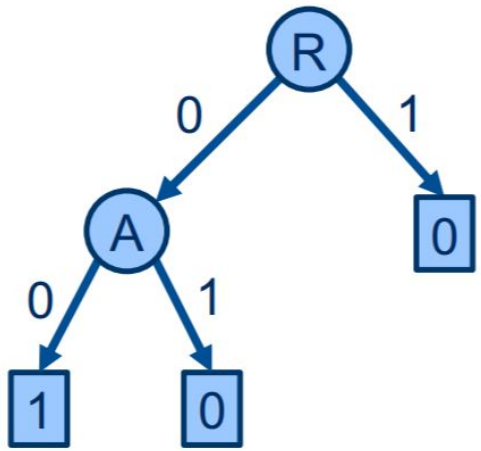


Decision Tree (features A, R)

If A=0, R=0 then L=1
If A=0, R=1 then L=0
If A=1, R=0 then L=0
If A=1, R=1 then L=0

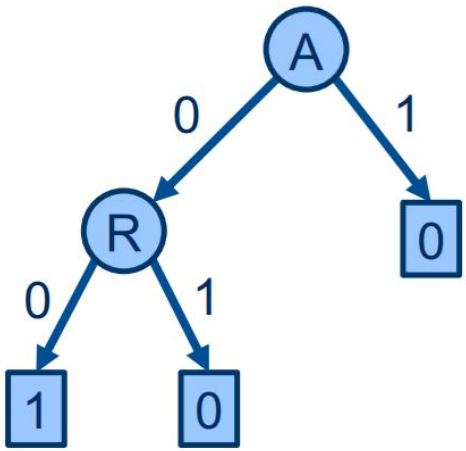


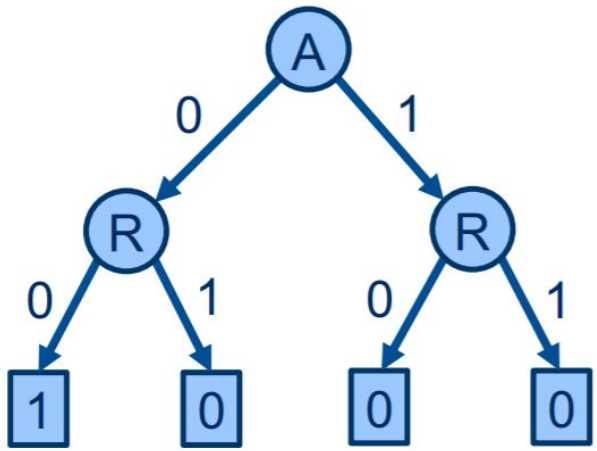


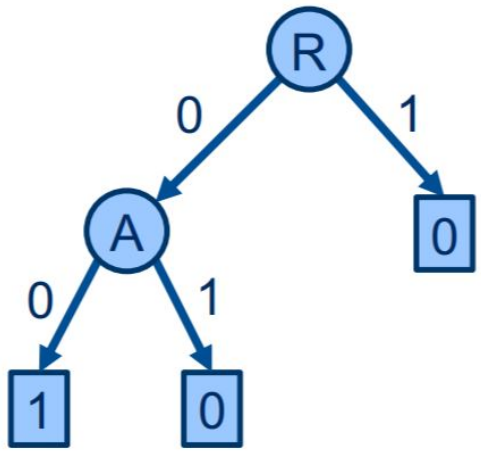


Decision Tree (features A, R)

If A=0, R=0 then L=1
If A=0, R=1 then L=0
If A=1, R=0 then L=0
If A=1, R=1 then L=0







These decision trees are equivalent!

Adding one more feature

H	A	R	L
0	0	0	1
0	1	1	0
1	0	1	0
1	0	0	1
0	0	1	0
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0
1	1	0	0
0	0	0	0
1	1	0	0
1	1	0	???

Adding one more feature

H	A	R	L
0	0	0	1
0	1	1	0
1	0	1	0
1	0	0	1
0	0	1	0
0	0	0	1
0	1	0	1
1	0	1	0
1	1	1	0
1	1	0	0
0	0	0	0
1	1	0	0
1	1	0	???

Distribution

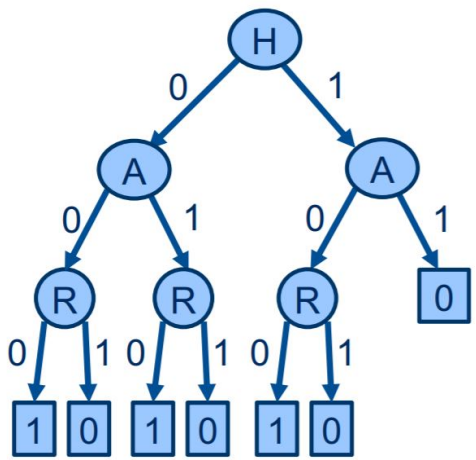
	L=0	L=1	Majority
H=0, A=0, R=0	1	2	L=1
H=0, A=0, R=1	1	0	L=0
H=0, A=1, R=0	0	1	L=1
H=0, A=1, R=1	1	0	L=0
H=1, A=0, R=0	0	1	L=1
H=1, A=0, R=1	2	0	L=0
H=1, A=1, R=0	2	0	L=0
H=1, A=1, R=1	1	0	L=0

Decision Tree (features H, A, R)

	L=0	L=1	Majority
H=0, A=0, R=0	1	2	L=1
H=0, A=0, R=1	1	0	L=0
H=0, A=1, R=0	0	1	L=1
H=0, A=1, R=1	1	0	L=0
H=1, A=0, R=0	0	1	L=1
H=1, A=0, R=1	2	0	L=0
H=1, A=1, R=0	2	0	L=0
H=1, A=1, R=1	1	0	L=0

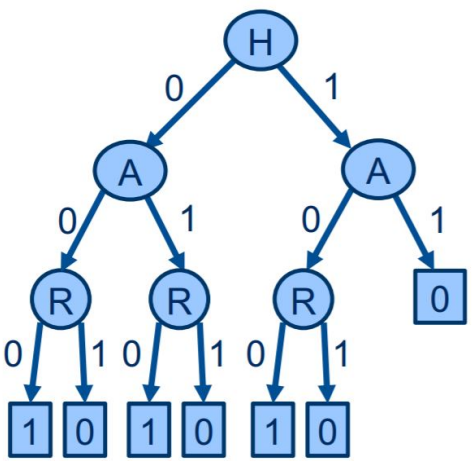
Decision Tree (features H, A, R)

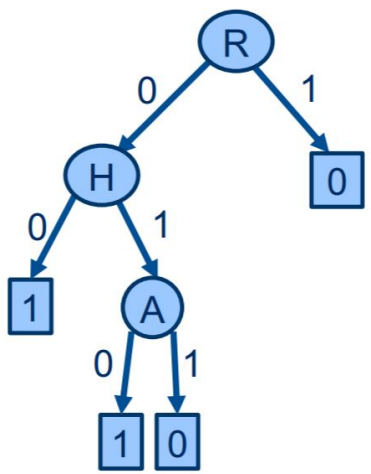
	L=0	L=1	Majority
H=0, A=0, R=0	1	2	L=1
H=0, A=0, R=1	1	0	L=0
H=0, A=1, R=0	0	1	L=1
H=0, A=1, R=1	1	0	L=0
H=1, A=0, R=0	0	1	L=1
H=1, A=0, R=1	2	0	L=0
H=1, A=1, R=0	2	0	L=0
H=1, A=1, R=1	1	0	L=0



Decision Tree (features H, A, R)

	L=0	L=1	Majority
H=0, A=0, R=0	1	2	L=1
H=0, A=0, R=1	1	0	L=0
H=0, A=1, R=0	0	1	L=1
H=0, A=1, R=1	1	0	L=0
H=1, A=0, R=0	0	1	L=1
H=1, A=0, R=1	2	0	L=0
H=1, A=1, R=0	2	0	L=0
H=1, A=1, R=1	1	0	L=0





Decision tree on all features?

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0
0	1	1	1	0	???

Decision tree on all features?

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0
0	1	1	1	0	???

Distribution

	L=0	L=1
...	...	...
P=0, Y=1, H=1, A=0, R=1	1	0
P=0, Y=1, H=1, A=1, R=0	0	0
...	...	...

Decision tree on all features?

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0
0	1	1	1	0	???

Distribution

	L=0	L=1
...	...	...
P=0, Y=1, H=1, A=0, R=1	1	0
P=0, Y=1, H=1, A=1, R=0	0	0
...	...	...

What to predict for this case?

Building a decision tree by recursively splitting the training data

- Let's fix the order of decision nodes to be P, Y, H, A, R

Building a decision tree by recursively splitting the training data

- Let's fix the order of decision nodes to be P, Y, H, A, R
- Split by first feature P:

	L=0	L=1	Pure?
P=0	6	4	No
P=1	2	0	Yes, L=0

Building a decision tree by recursively splitting the training data

- Let's fix the order of decision nodes to be P, Y, H, A, R
- Split by first feature P:

	L=0	L=1	Pure?
P=0	6	4	No
P=1	2	0	Yes, L=0

- P=1 is **pure** (all points belong to the same class), no need to split further

Building a decision tree by recursively splitting the training data

- Let's fix the order of decision nodes to be P, Y, H, A, R
- Split by first feature P:

	L=0	L=1	Pure?
P=0	6	4	No
P=1	2	0	Yes, L=0

- P=1 is **pure** (all points belong to the same class), no need to split further
- Split P=0 by feature Y:

	L=0	L=1	Pure?
P=0, Y=0	4	2	No
P=0, Y=1	2	2	No
P=1	2	0	Yes, L=0

Building a decision tree by recursively splitting the training data

- Let's fix the order of decision nodes to be P, Y, H, A, R
- Split by first feature P:

	L=0	L=1	Pure?
P=0	6	4	No
P=1	2	0	Yes, L=0

- P=1 is **pure** (all points belong to the same class), no need to split further
- Split P=0 by feature Y:

	L=0	L=1	Pure?
P=0, Y=0	4	2	No
P=0, Y=1	2	2	No
P=1	2	0	Yes, L=0

- Two new nodes are not pure, continue splitting

Building a decision tree by recursively splitting the training data

- Let's fix the order of decision nodes to be P, Y, H, A, R
- Split by first feature P:

	L=0	L=1	Pure?
P=0	6	4	No
P=1	2	0	Yes, L=0

- P=1 is **pure** (all points belong to the same class), no need to split further
- Split P=0 by feature Y:

	L=0	L=1	Pure?
P=0, Y=0	4	2	No
P=0, Y=1	2	2	No
P=1	2	0	Yes, L=0

- Two new nodes are not pure, continue splitting
- Split recursively P=0, Y=0 and P=0, Y=1 by H

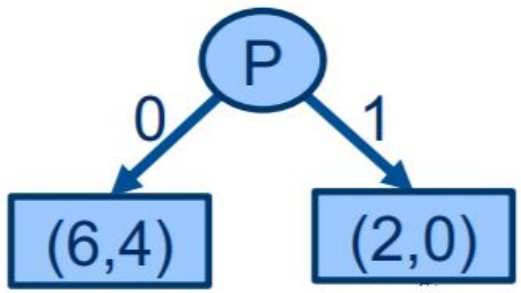
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0

(8,4)

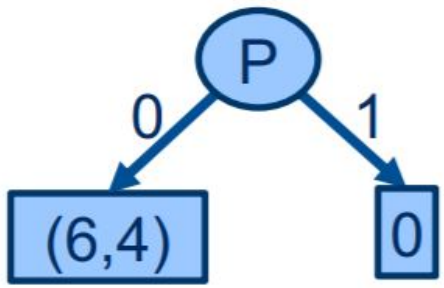
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



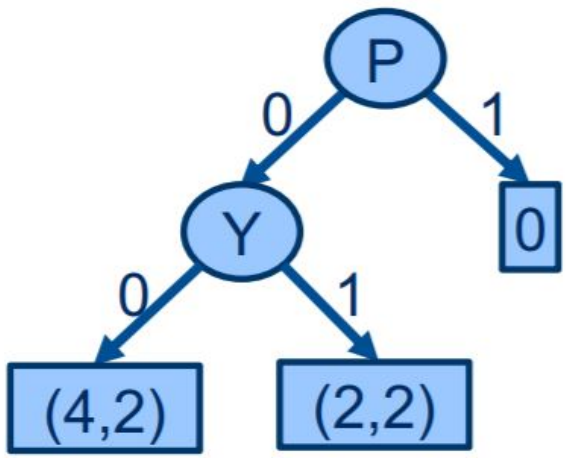
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



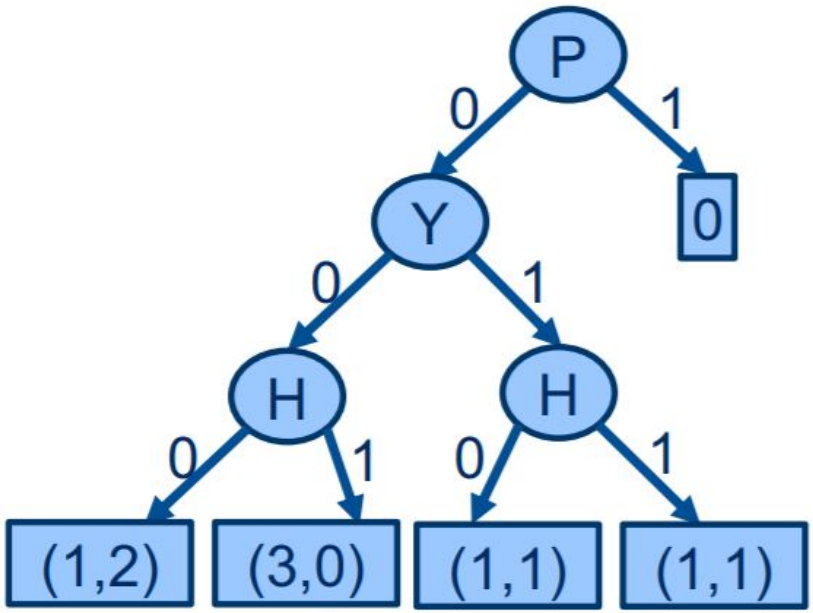
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



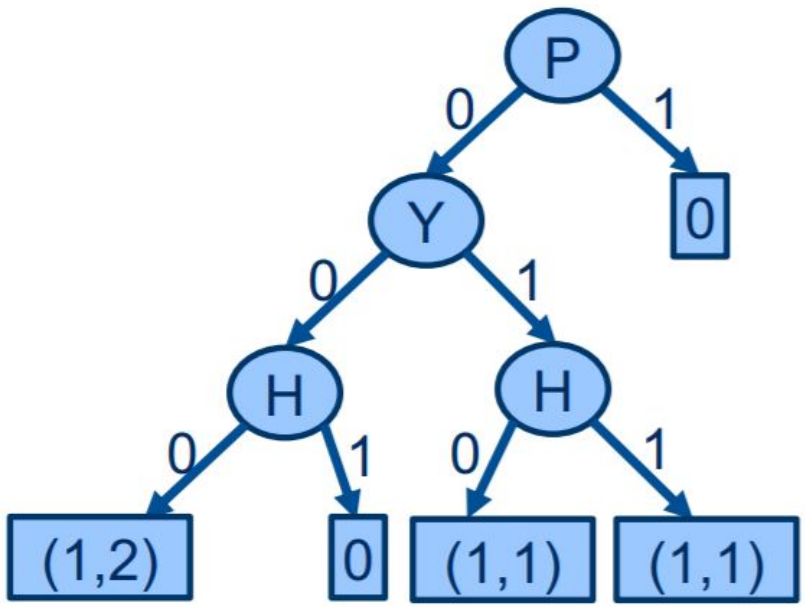
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



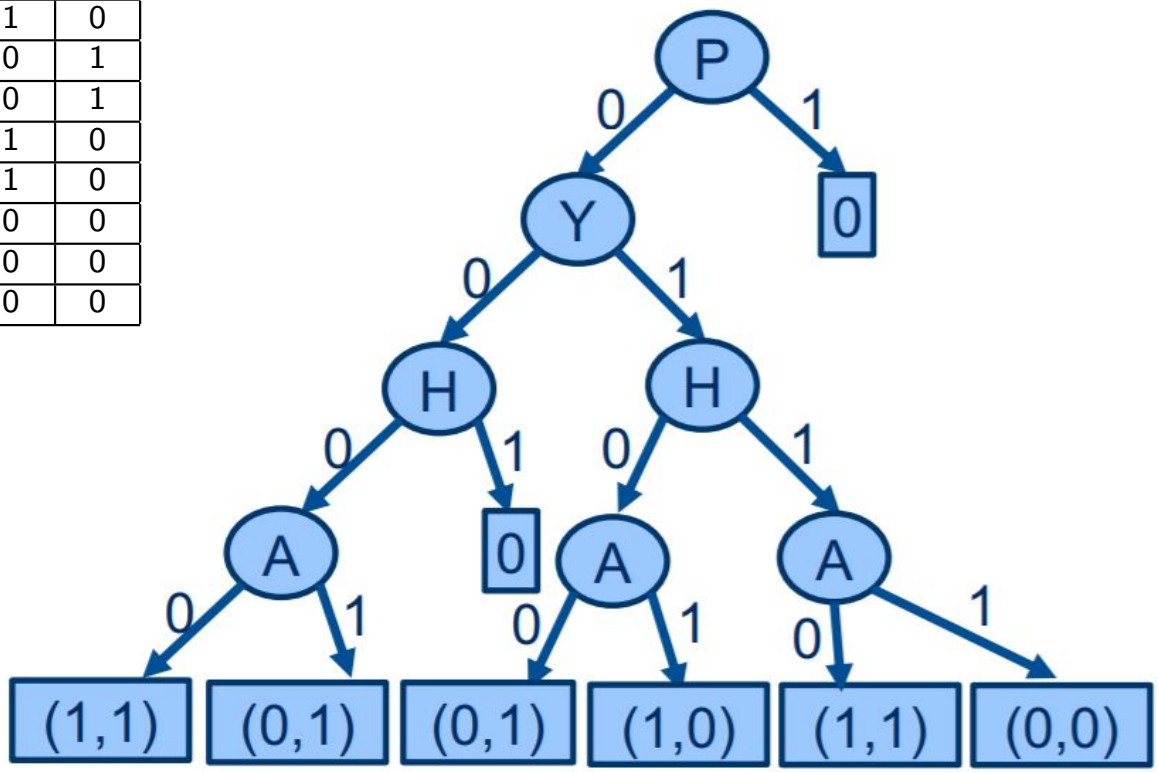
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



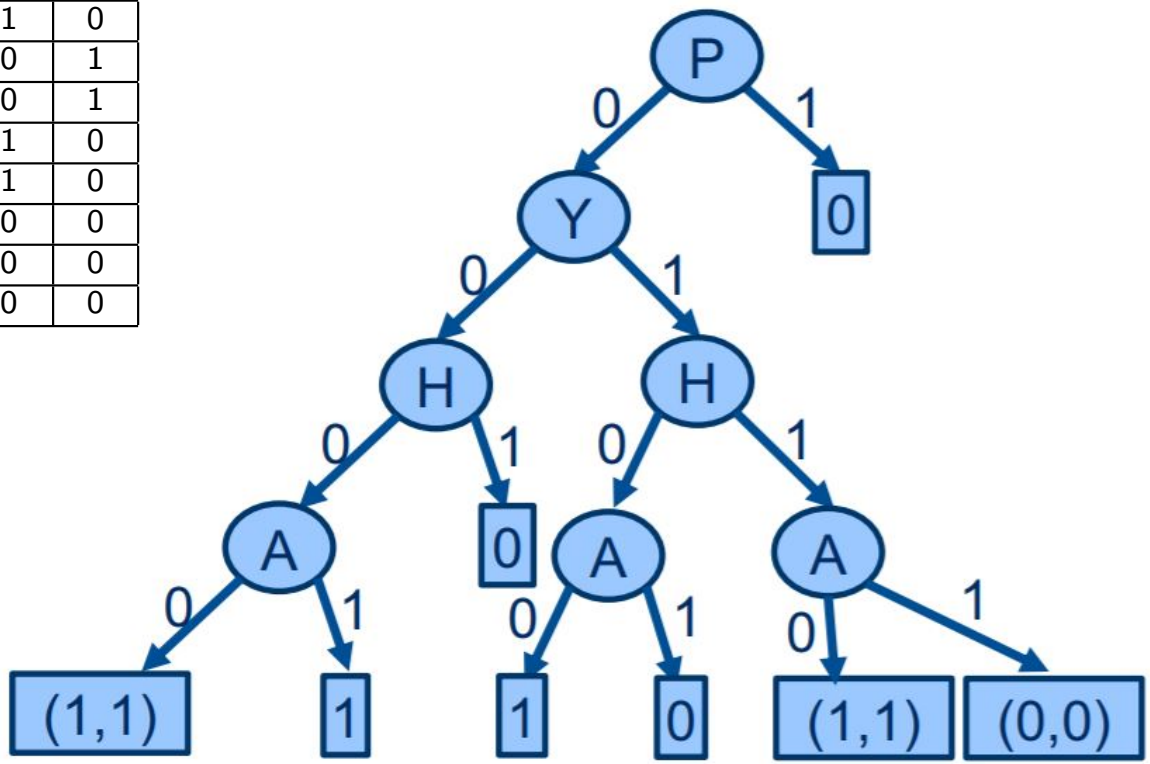
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



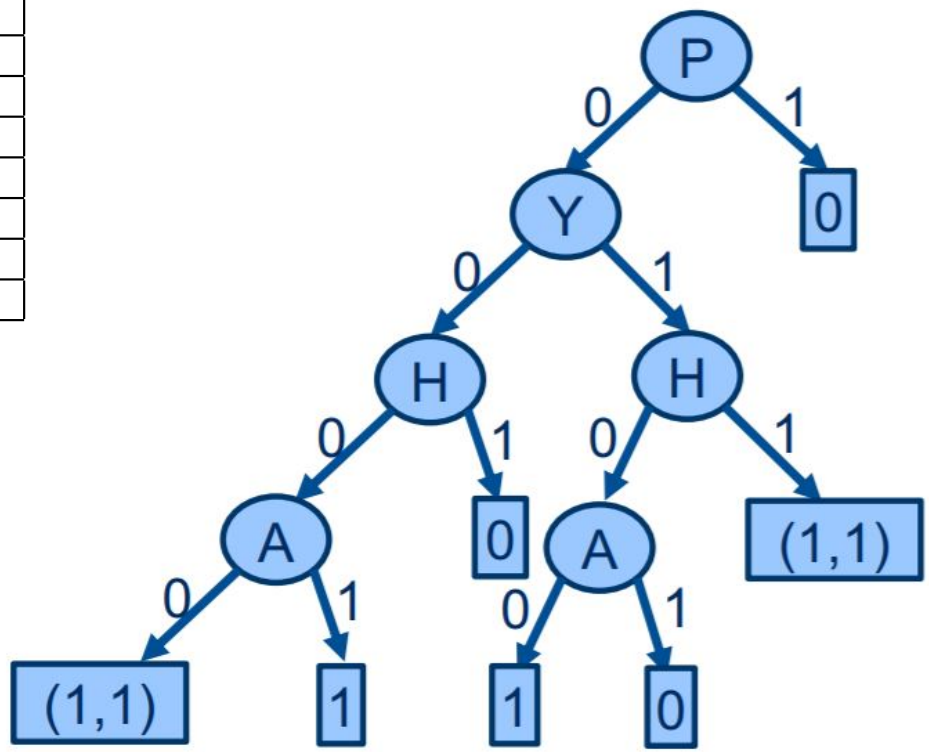
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



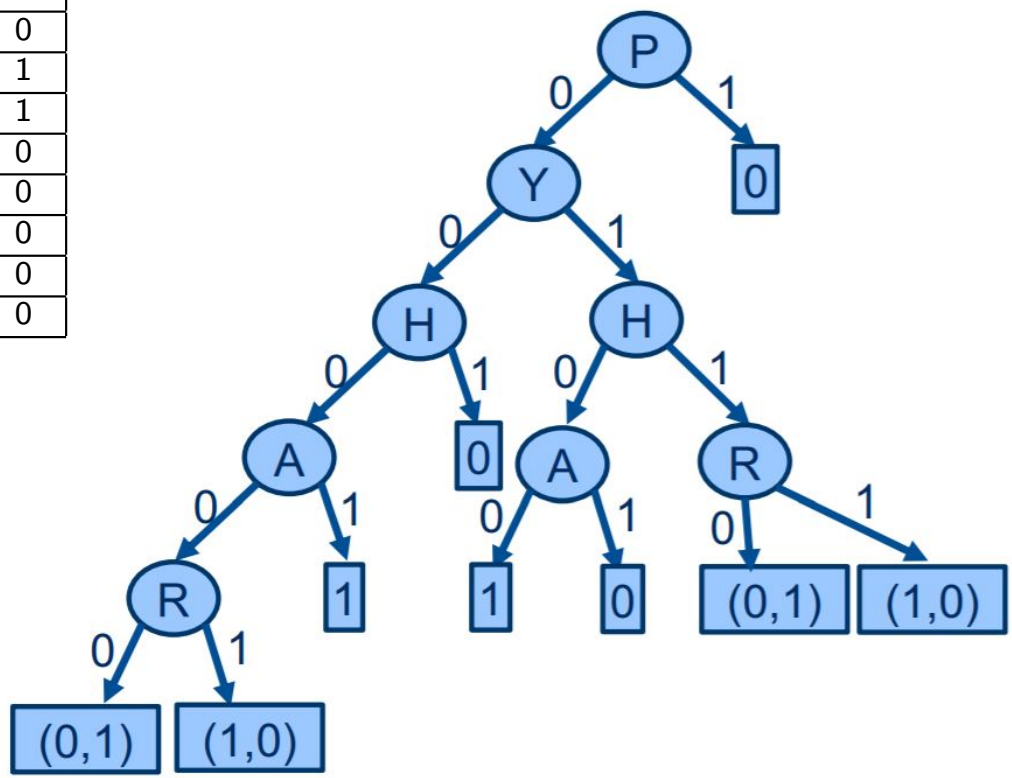
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



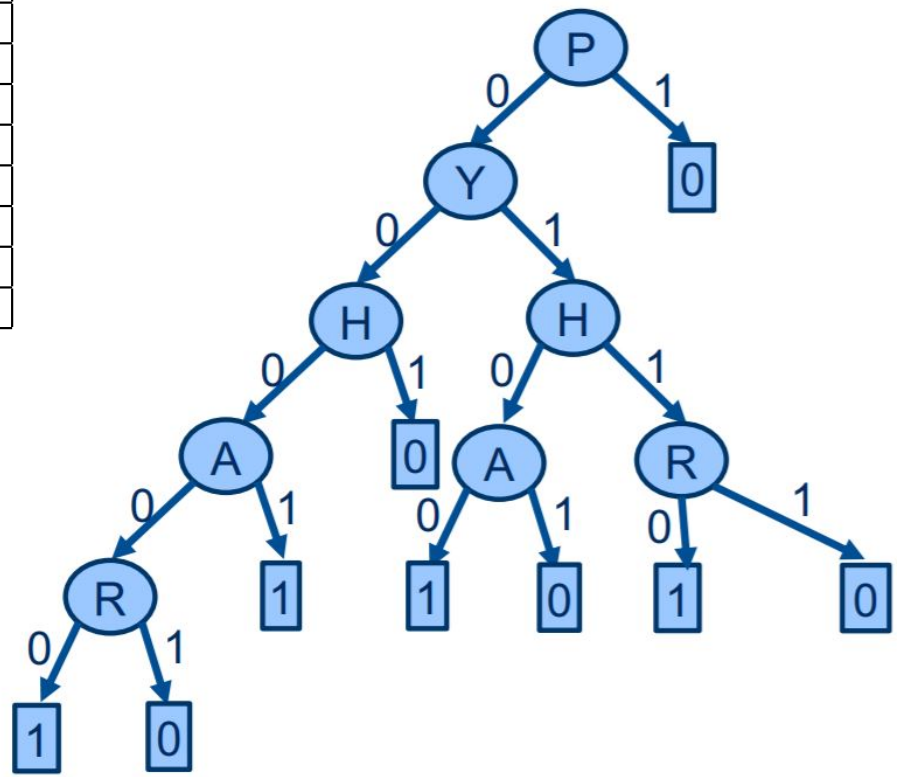
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



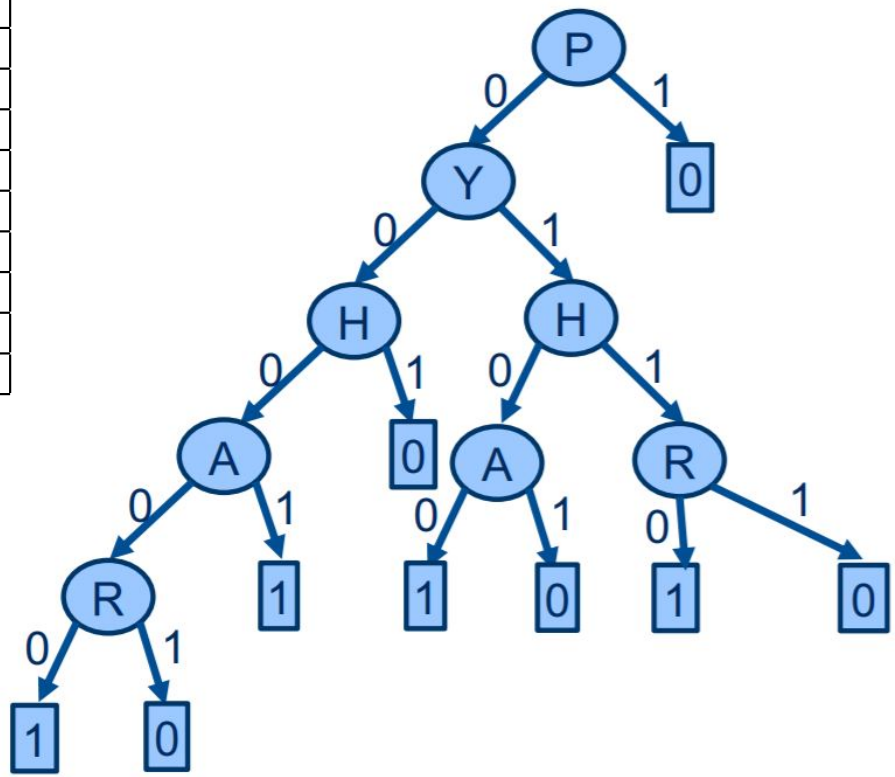
Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



Building a decision tree by recursively splitting the training data

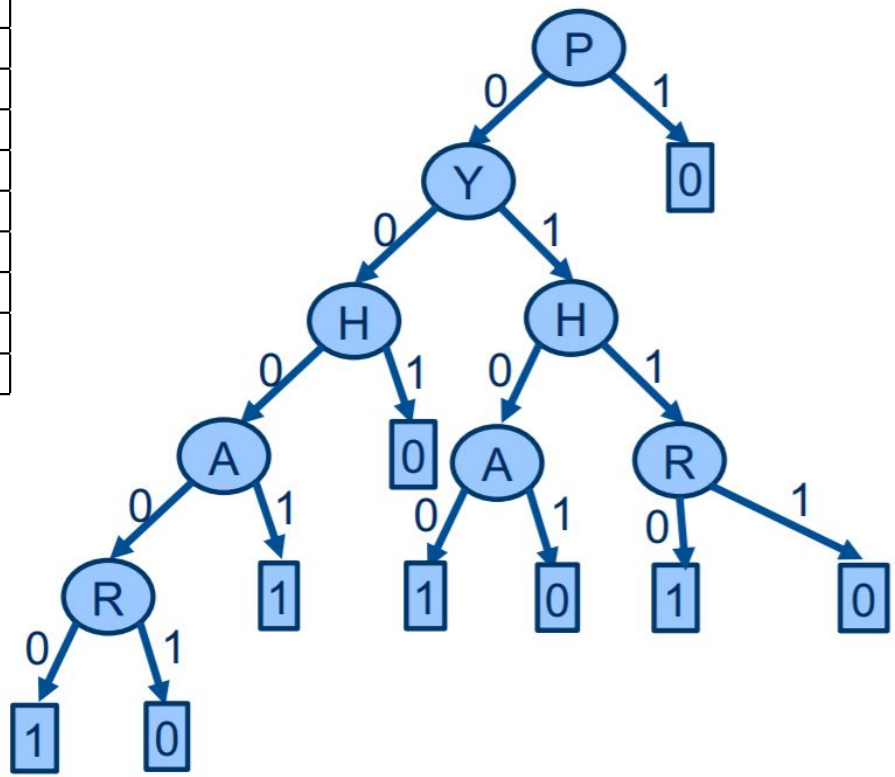
P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



Predict for test instance **P=0, Y=1, H=1, A=1, R=0**:

Building a decision tree by recursively splitting the training data

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



Predict for test instance **P=0, Y=1, H=1, A=1, R=0**: **L=1**

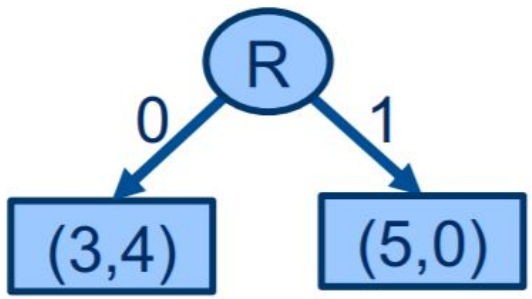
Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0

(8,4)

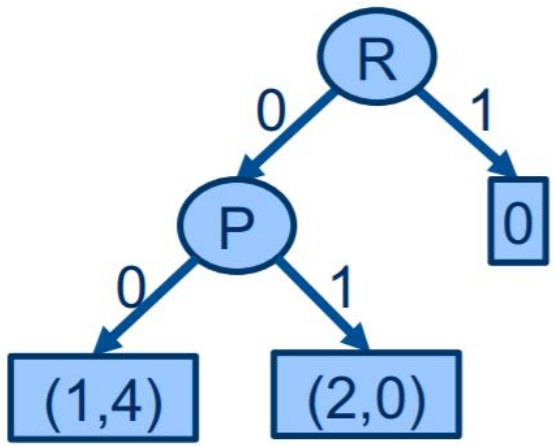
Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



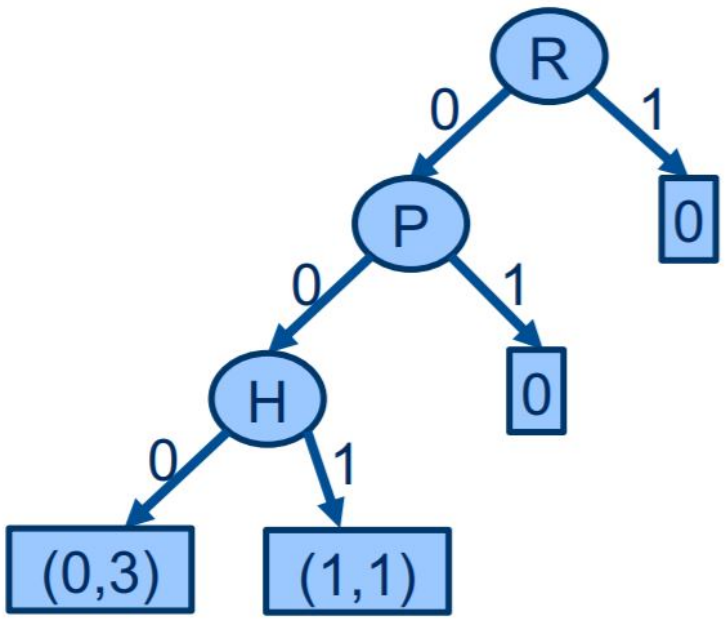
Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



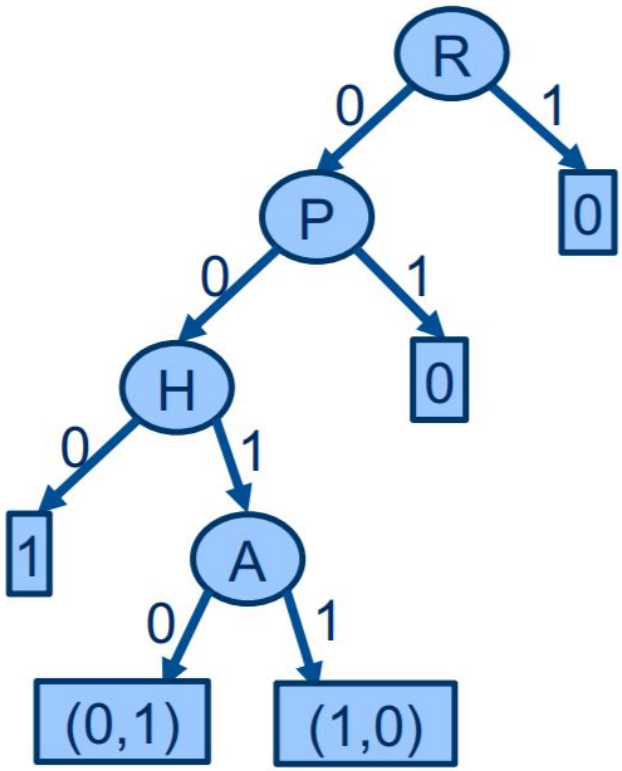
Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



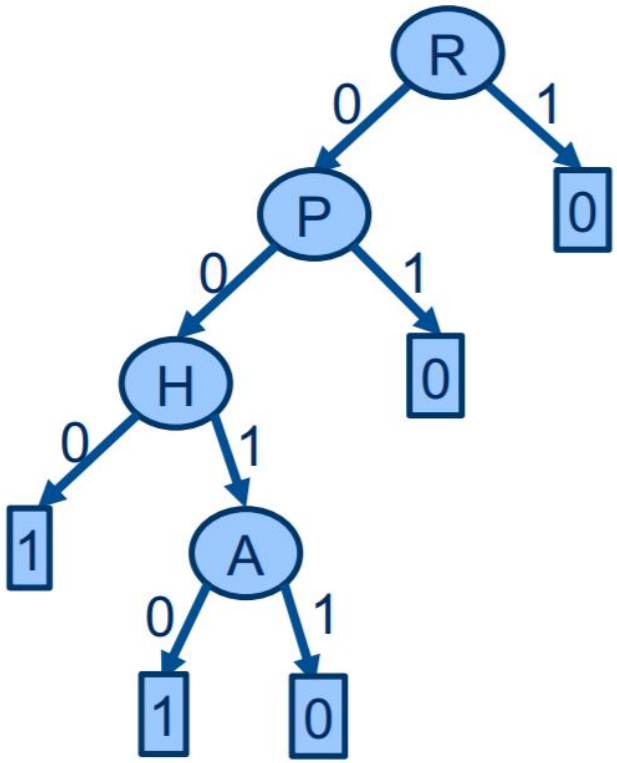
Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



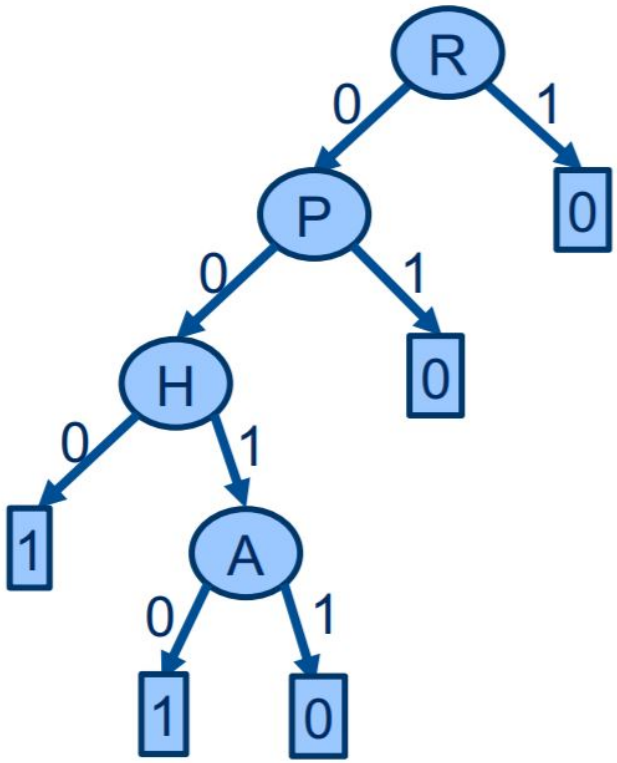
Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

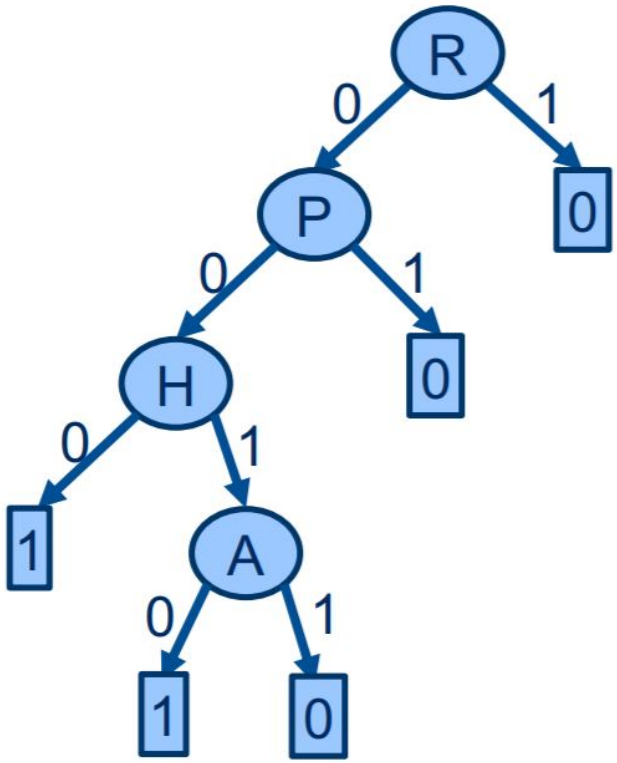
P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



Predict for test instance **P=0, Y=1, H=1, A=1, R=0**:

Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

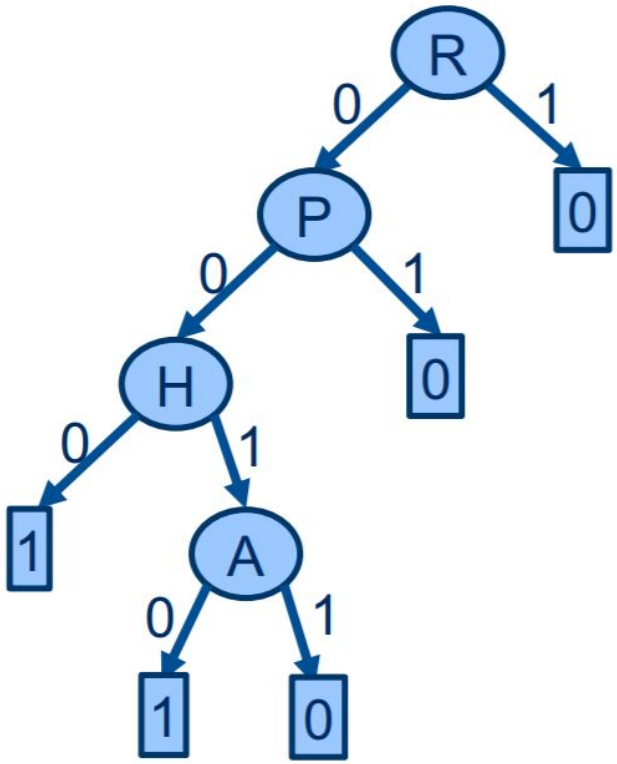
P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



Predict for test instance **P=0, Y=1, H=1, A=1, R=0**: **L=0**

Building a decision tree by recursively splitting the training data (different fixed order: R, P, H, A, Y)

P	Y	H	A	R	L
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	0	0	1
0	0	0	0	1	0
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	1	0
0	0	1	1	0	0
1	0	0	0	0	0
1	0	1	1	0	0



Predict for test instance **P=0, Y=1, H=1, A=1, R=0**: **L=0**

Different order of features resulted in a different tree and prediction

Which decision tree is better?

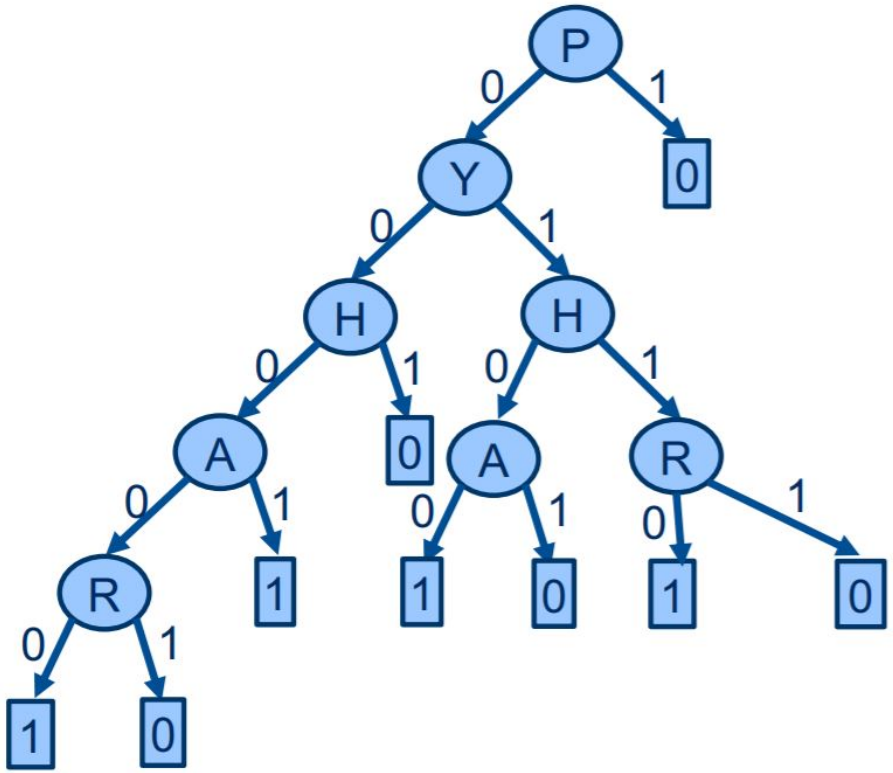
- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**

Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?

Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?
  - Decision tree from order P, Y, H, A, R:

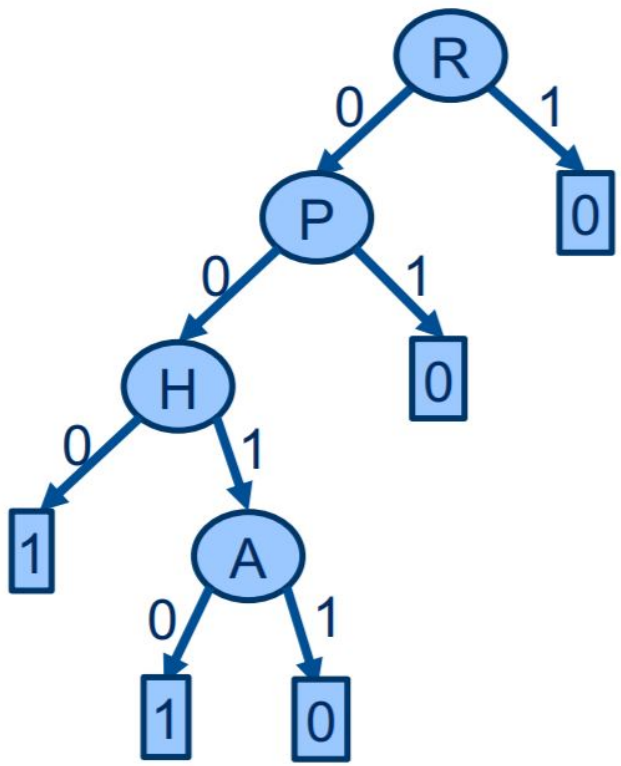


Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?
  - Decision tree from order P, Y, H, A, R: **L=1** (learned from 1 training instance)

Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?
  - Decision tree from order P, Y, H, A, R: **L=1** (learned from 1 training instance)
  - Decision tree from order R, P, H, A, Y:



Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?
  - Decision tree from order P, Y, H, A, R: **L=1** (learned from 1 training instance)
  - Decision tree from order R, P, H, A, Y: **L=0** (learned from 5 training instance)

Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?
  - Decision tree from order P, Y, H, A, R:
    "**I predict L=1 because the only training instance with P=0,Y=0,H=0,A=1 (but R=0) had L=1**"
  - Decision tree from order R, P, H, A, Y:
    "**I predict L=0 because all 5 training instances with R=1 always had L=0**"

Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?
  - Decision tree from order P, Y, H, A, R:
    "**I predict L=1 because the only training instance with P=0,Y=0,H=0,A=1 (but R=0) had L=1**"
  - Decision tree from order R, P, H, A, Y:
    "**I predict L=0 because all 5 training instances with R=1 always had L=0**"
- Which one would you believe?

Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?
  - Decision tree from order P, Y, H, A, R:
    "**I predict it is ok for you to use contact lenses because optician said ok to another person who was also prepresbyopic, myopic and astigmatic except that unlike you he/she had normal tear production rate**"
  - Decision tree from order R, P, H, A, Y:
    "**I predict you should not use contact lenses because optician suggested so to all 5 people with reduced tear production rate**"

Which decision tree is better?

- Consider another test instance: **P=0, Y=0, H=0, A=1, R=1**
- What do the 2 trees predict?
  - Decision tree from order P, Y, H, A, R:
    "**I predict it is ok for you to use contact lenses because optician said ok to another person who was also prepresbyopic, myopic and astigmatic except that unlike you he/she had normal tear production rate**"
  - Decision tree from order R, P, H, A, Y:
    "**I predict you should not use contact lenses because optician suggested so to all 5 people with reduced tear production rate**"
- Which one would you believe now?

Which decision tree is better?

- Usually, pure decision nodes covered by more training instances predict better
- Therefore, **usually** smaller trees are better, because on average, they have higher coverage
- Often it is better to stop splitting if the node becomes small (< 10 or < 20 instances), even if the node is not yet pure

How to order the features in splitting?

- What is a good order?

How to order the features in splitting?

- What is a good order?
  - Achieving purity sooner is better

How to order the features in splitting?

- What is a good order?
  - Achieving purity sooner is better
  - Because then the tree would be smaller

How to order the features in splitting?

- What is a good order?
  - Achieving purity sooner is better
  - Because then the tree would be smaller
- How can we achieve purity sooner?

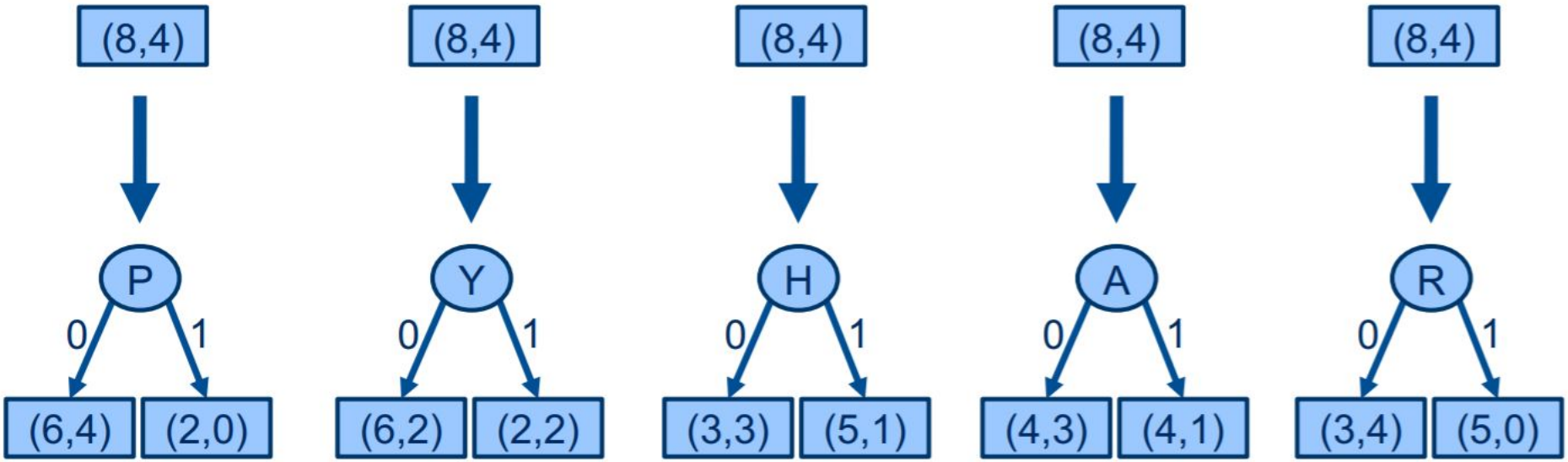
How to order the features in splitting?

- What is a good order?
  - Achieving purity sooner is better
  - Because then the tree would be smaller
- How can we achieve purity sooner?
- Choose the feature which increases purity the most

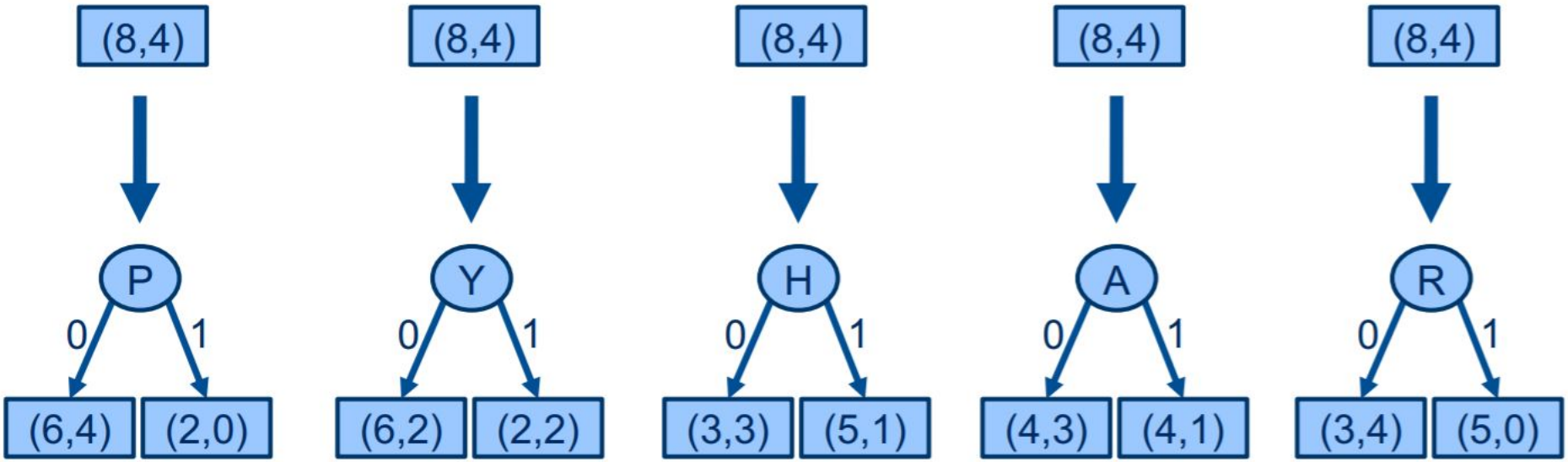
How to order the features in splitting?

- What is a good order?
  - Achieving purity sooner is better
  - Because then the tree would be smaller
- How can we achieve purity sooner?
- Choose the feature which increases purity the most
- Therefore, we need to be able to measure purity

Which split increases purity the most?

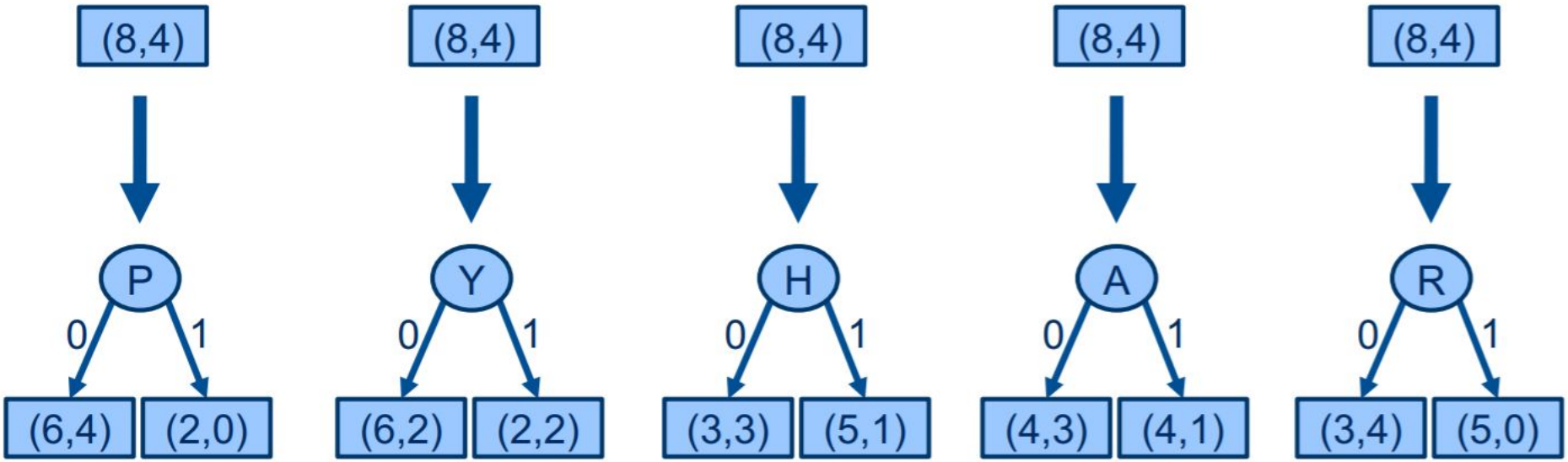


Which split increases purity the most?



Gini impurity of (N_0, N_1): $Gini = 2p(1-p)$ where $p = \dfrac{N_1}{N_0 + N_1}$

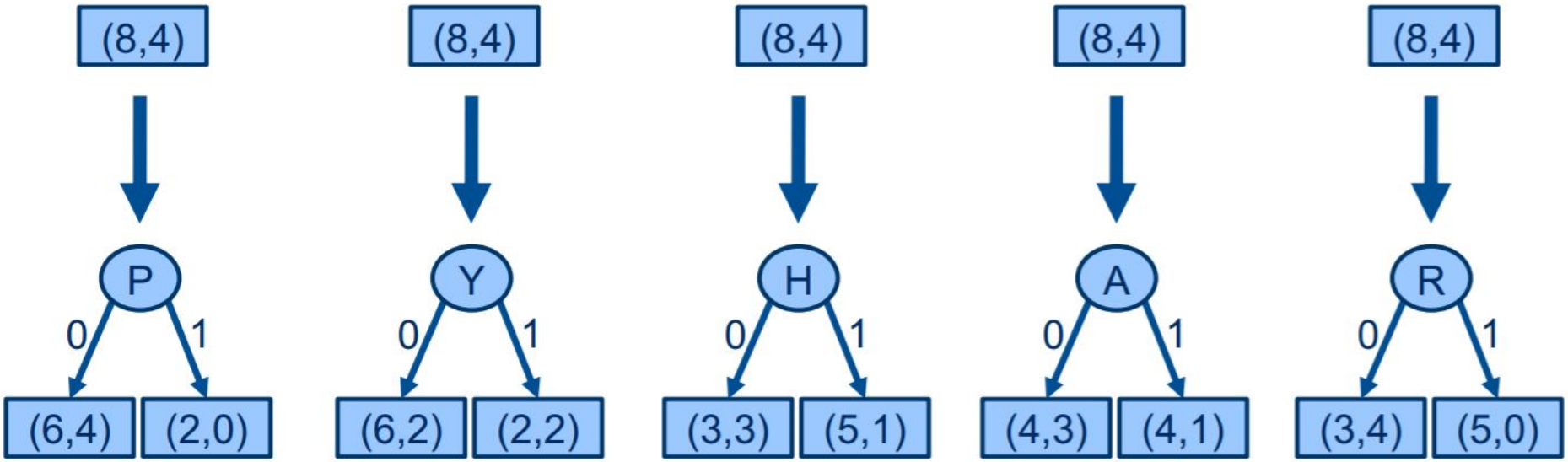
Which split increases purity the most?



p : 0.4 0 0.25 0.5 0.5 0.17 0.43 0.2 0.57 0

Gini impurity of (N_0, N_1): $Gini = 2p(1-p)$ where $p = \dfrac{N_1}{N_0 + N_1}$

Which split increases purity the most?

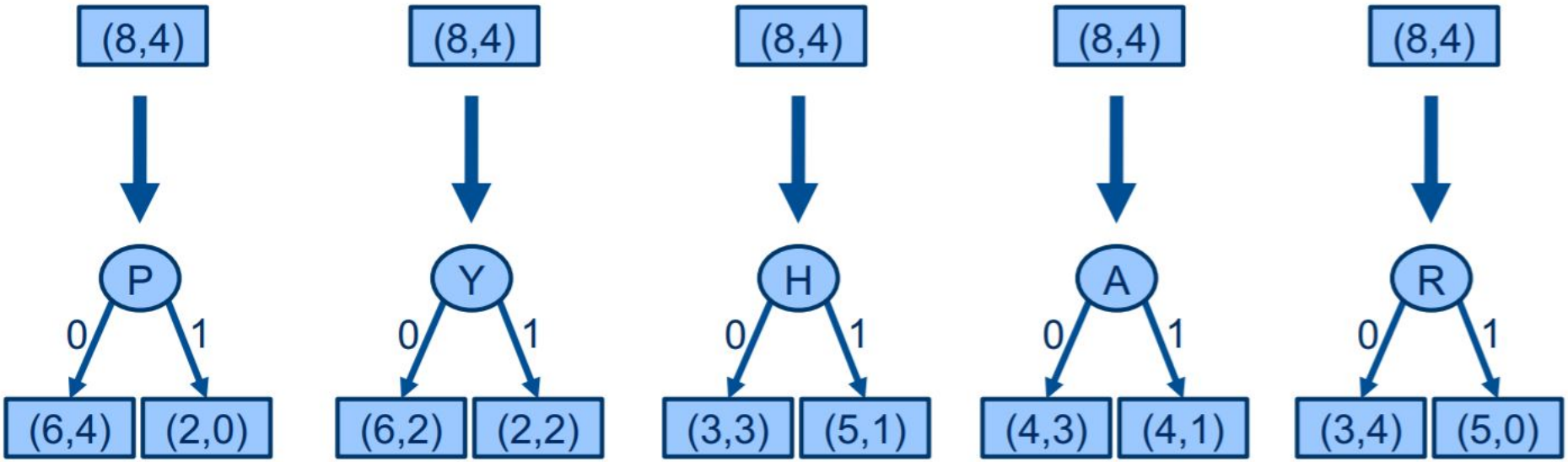


p : 0.4 0 0.25 0.5 0.5 0.17 0.43 0.2 0.57 0

$Gini$: 0.48 0 0.38 0.5 0.5 0.28 0.49 0.32 0.49 0

Gini impurity of (N_0, N_1): $Gini = 2p(1-p)$ where $p = \dfrac{N_1}{N_0 + N_1}$

Which split increases purity the most?



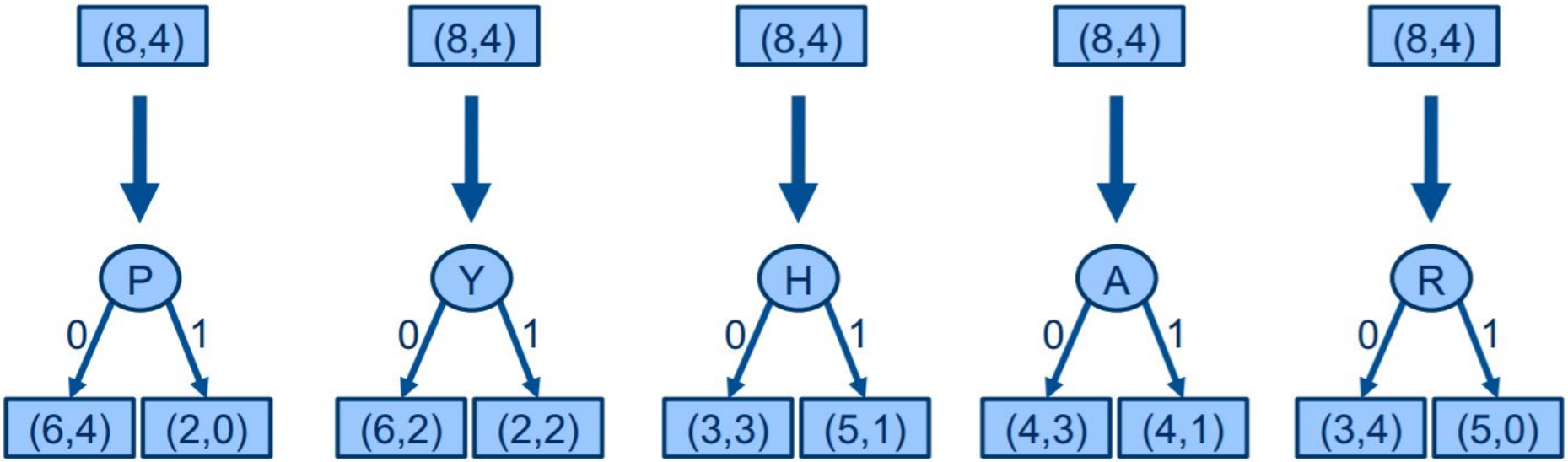
p :	0.4	0	0.25	0.5	0.5	0.17	0.43	0.2	0.57	0
$Gini$:	0.48	0	0.38	0.5	0.5	0.28	0.49	0.32	0.49	0
$AvgGini$:	0.4		0.42		0.39		0.42		0.29	

Gini impurity of (N_0, N_1): $Gini = 2p(1-p)$ where $p = \dfrac{N_1}{N_0 + N_1}$

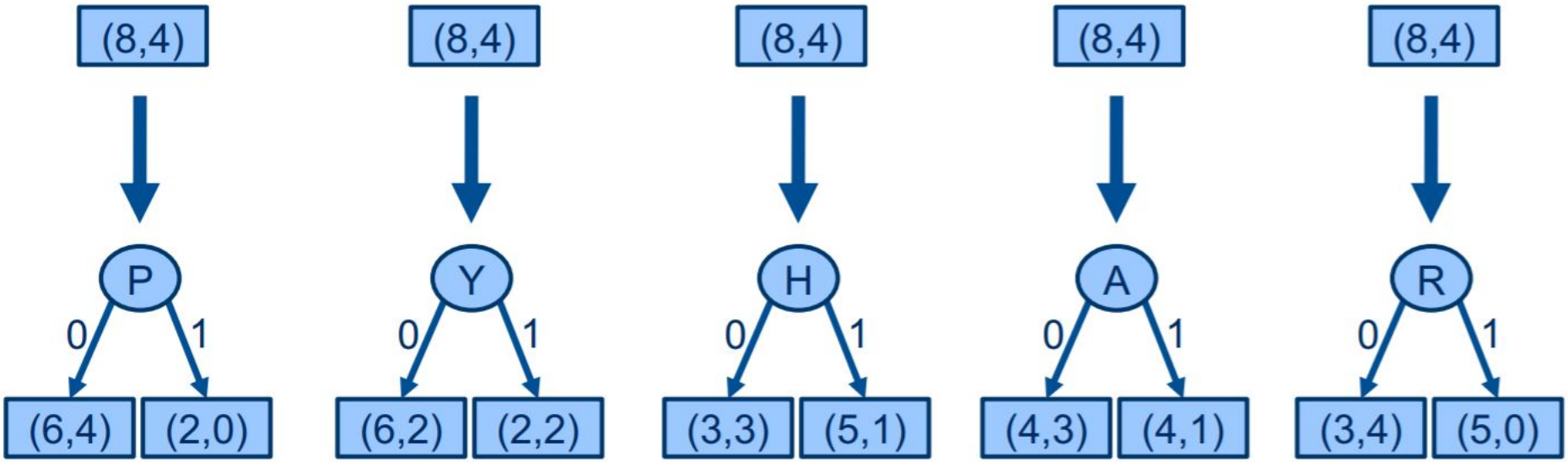
Average Gini impurity after split:

$$AvgGini = \frac{Size_{left} Gini_{left} + Size_{right} Gini_{right}}{Size_{left} + Size_{right}}$$

Which split increases purity the most?

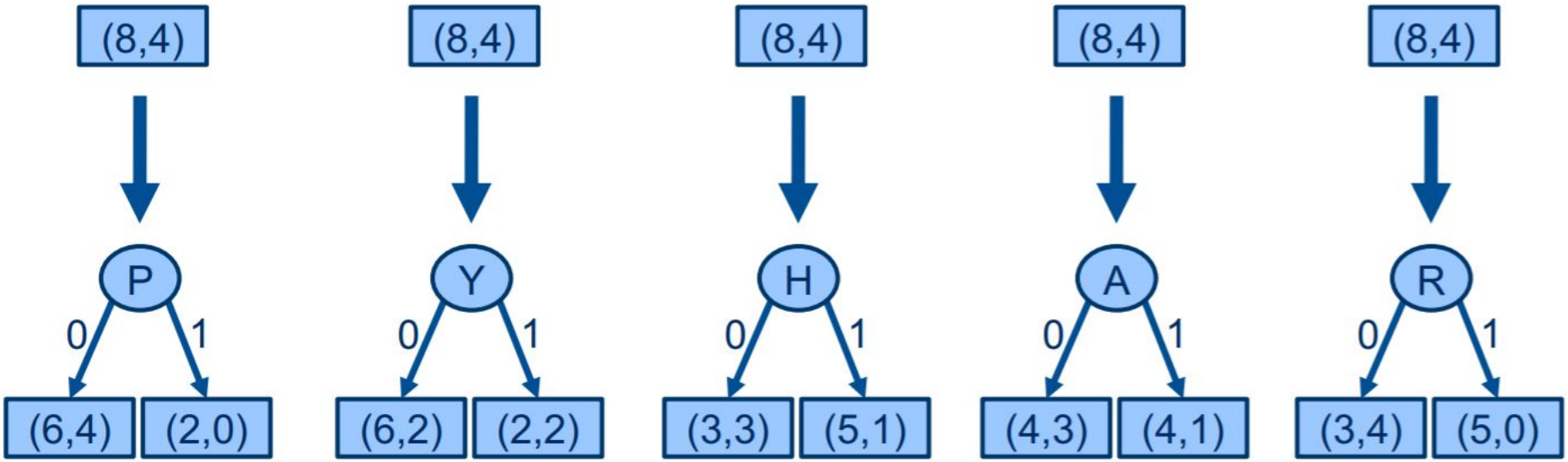


Which split increases purity the most?



Entropy of (N_0, N_1): $E = -p\log_2(p) - (1-p)\log_2(1-p)$ where $p = \dfrac{N_1}{N_0 + N_1}$

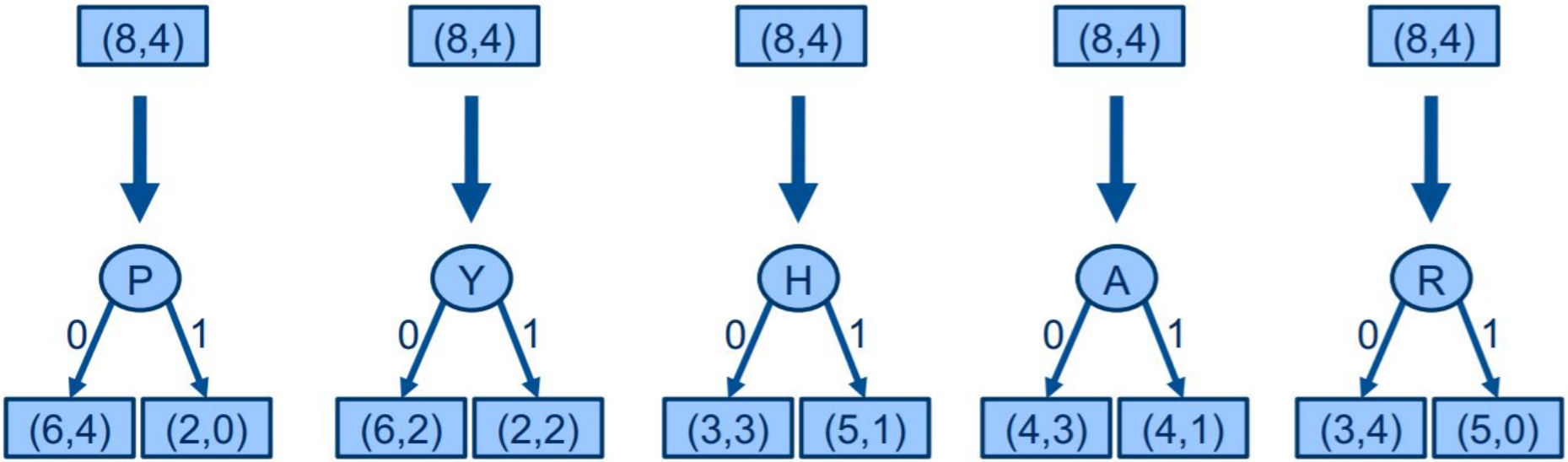
Which split increases purity the most?



p : 0.4 0 0.25 0.5 0.5 0.17 0.43 0.2 0.57 0

Entropy of (N_0, N_1): $E = -p\log_2(p) - (1-p)\log_2(1-p)$ where $p = \dfrac{N_1}{N_0 + N_1}$

Which split increases purity the most?

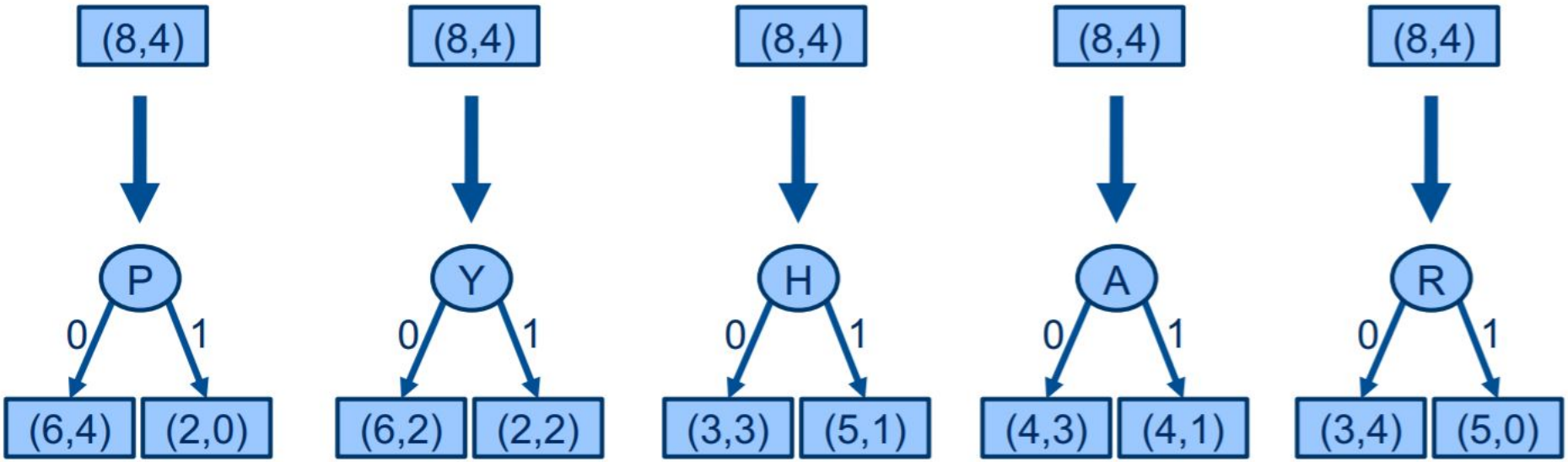


p : 0.4 0 | 0.25 0.5 | 0.5 0.17 | 0.43 0.2 | 0.57 0

E : 0.97 0 | 0.81 1 | 1 0.65 | 0.99 0.72 | 0.99 0

Entropy of (N_0, N_1): $E = -p\log_2(p) - (1-p)\log_2(1-p)$ where $p = \dfrac{N_1}{N_0 + N_1}$

Which split increases purity the most?



p :	0.4	0		0.25	0.5		0.5	0.17	
E :	0.97	0		0.81	1		1	0.65	
IG :	0.11			0.04			0.09		

p :	0.43	0.2		0.57	0				
E :	0.99	0.72		0.99	0				
IG :	0.04			0.34					

Entropy of (N_0, N_1): $E = -p\log_2(p) - (1-p)\log_2(1-p)$ where $p = \dfrac{N_1}{N_0 + N_1}$

Information gain after split: $IG = E_{parent} - \dfrac{Size_{left}E_{left} + Size_{right}E_{right}}{Size_{left} + Size_{right}}$

Choosing the best split

- Which is the best feature to split on?
  - The one which minimizes the size of the tree
  - Cannot know the size without recursively building the tree (this would be computationally very hard)
- Intuitively, the splits which result in class distributions closer to purity potentially lead to smaller trees
- DT learners estimate how much purity improves (or how much impurity is reduced) after the considered split

Gini Index

- Gini index: $2\dot{p}(1-\dot{p})$
- Measures the expected proportion of misclassified instances if the leaf was labelled randomly: positive with probability $\dot{p}$ and negative with probability $1-\dot{p}$
- The probability of a false positive is then $\dot{p}(1-\dot{p})$ and the probability of a false negative is $(1-\dot{p})\dot{p}$
- In case of multiclass classification (there are C classes) the formula is

$$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

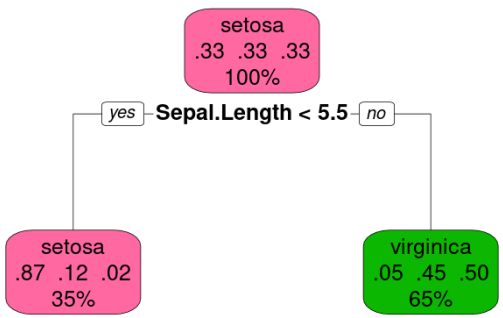
Entropy

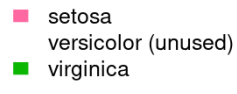
- Entropy: $-\dot{p}\log_2\dot{p} - (1-\dot{p})\log_2(1-\dot{p})$
- Measures the expected information, in bits, conveyed by somebody telling you the class of a randomly drawn example
- The purer the set of examples, the more predictable this message becomes and the smaller the expected information
- In case of multiclass classification (there are C classes) the formula is

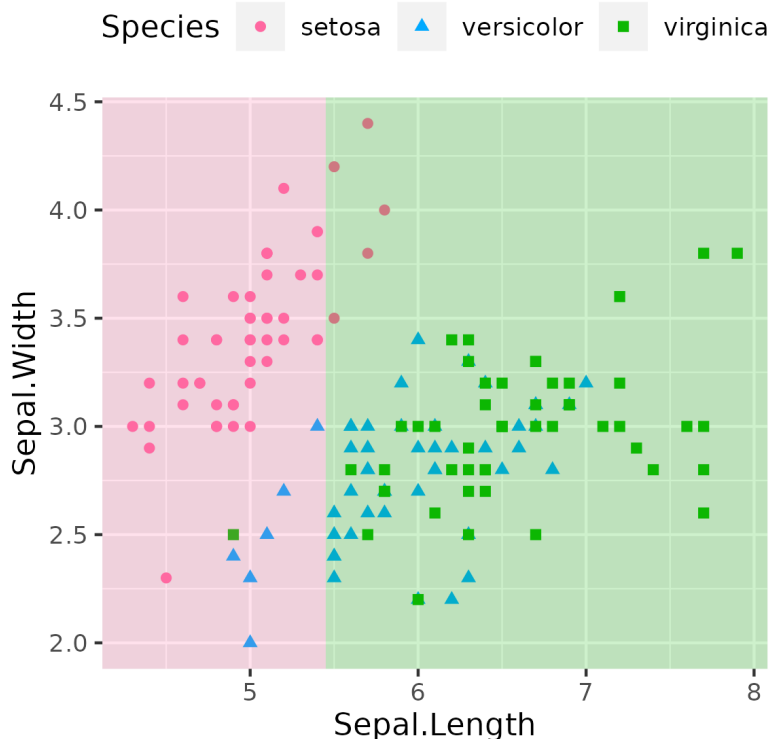
$$Entropy = -\sum_{i=1}^{C} p_i \cdot \log_2 p_i$$

TREE GROWING

1. Start with a root node of all data.
2. Search for feature and split point that minimizes the empirical risk in child nodes – makes label distribution more homogenous.

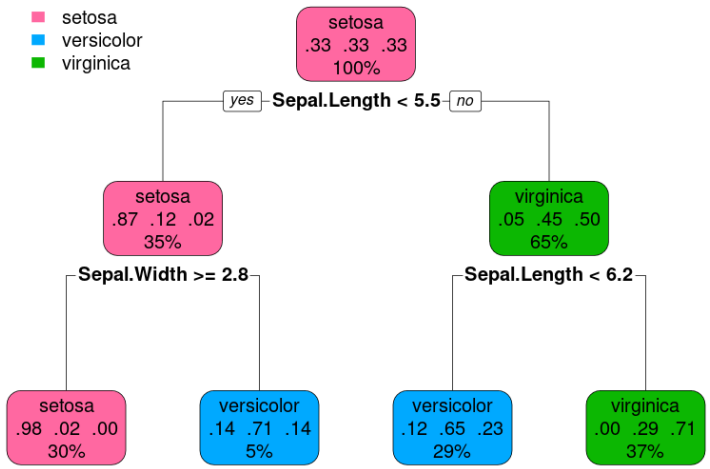


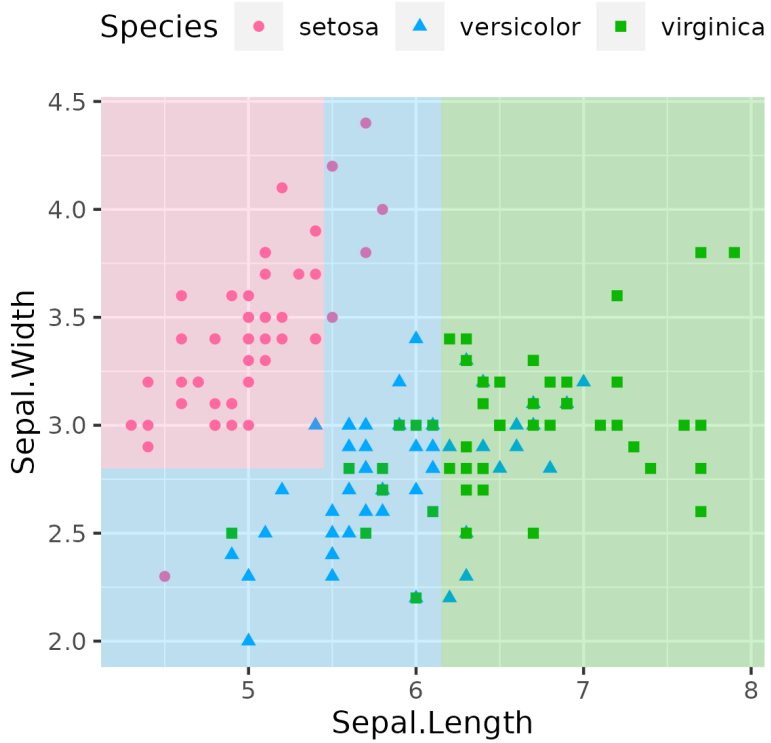




TREE GROWING

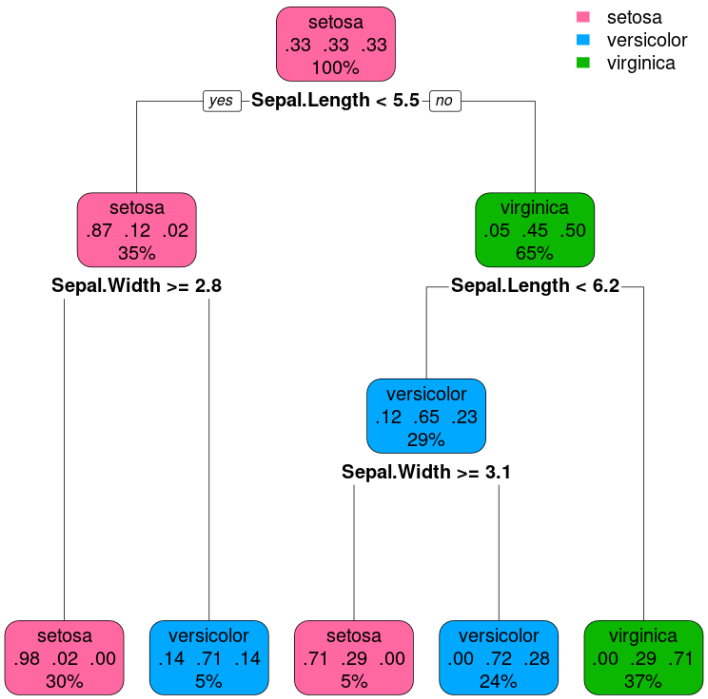
3. Proceed recursively for each child node: Select best split and divide data from parent node into left and right child nodes.

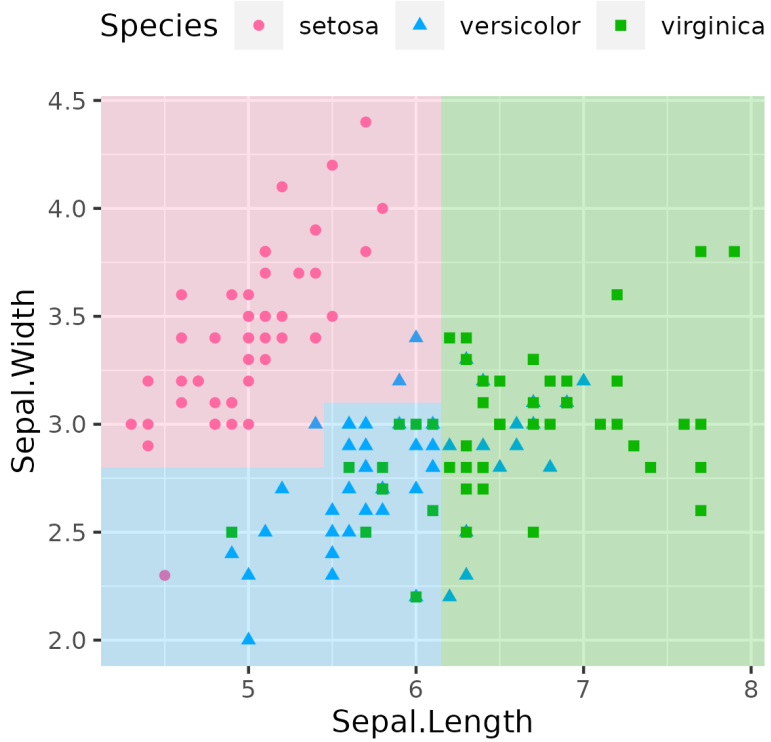




TREE GROWING

4. Repeat until we reach a stop criterion, e.g., until each leaf cannot be split further.





Decision Trees are powerful models capable of capturing complex patterns.

However, they can easily overfit the training data, leading to 100% accuracy on the training set but poor performance on unseen data.

To reduce overfitting, we can regularize the Decision Tree.
Two common techniques for regularization:

1. **Pre-pruning:** Stops tree growth early, based on criteria like maximum depth or minimum samples per leaf.
2. **Post-pruning:** Prunes back a fully grown tree by removing branches that provide little to no improvement.

Pre-pruning

- Pre-pruning means that during decision tree learning some nodes are not split further
- Decision not to split is typically made if one of the following conditions hold:
  - Node size is below a threshold
  - One of the child nodes would have size below a threshold
  - Depth of the tree is above a threshold
  - Impurity gain is below a threshold

Post-pruning

- Post-pruning means that after the tree has been learned, some of the subtrees are removed (replaced by their parent node)
- Decision to prune usually done based on tree performance on a hold-out dataset
- Reduced-error pruning removes subtrees which perform worse than majority class (within the respective parent node)

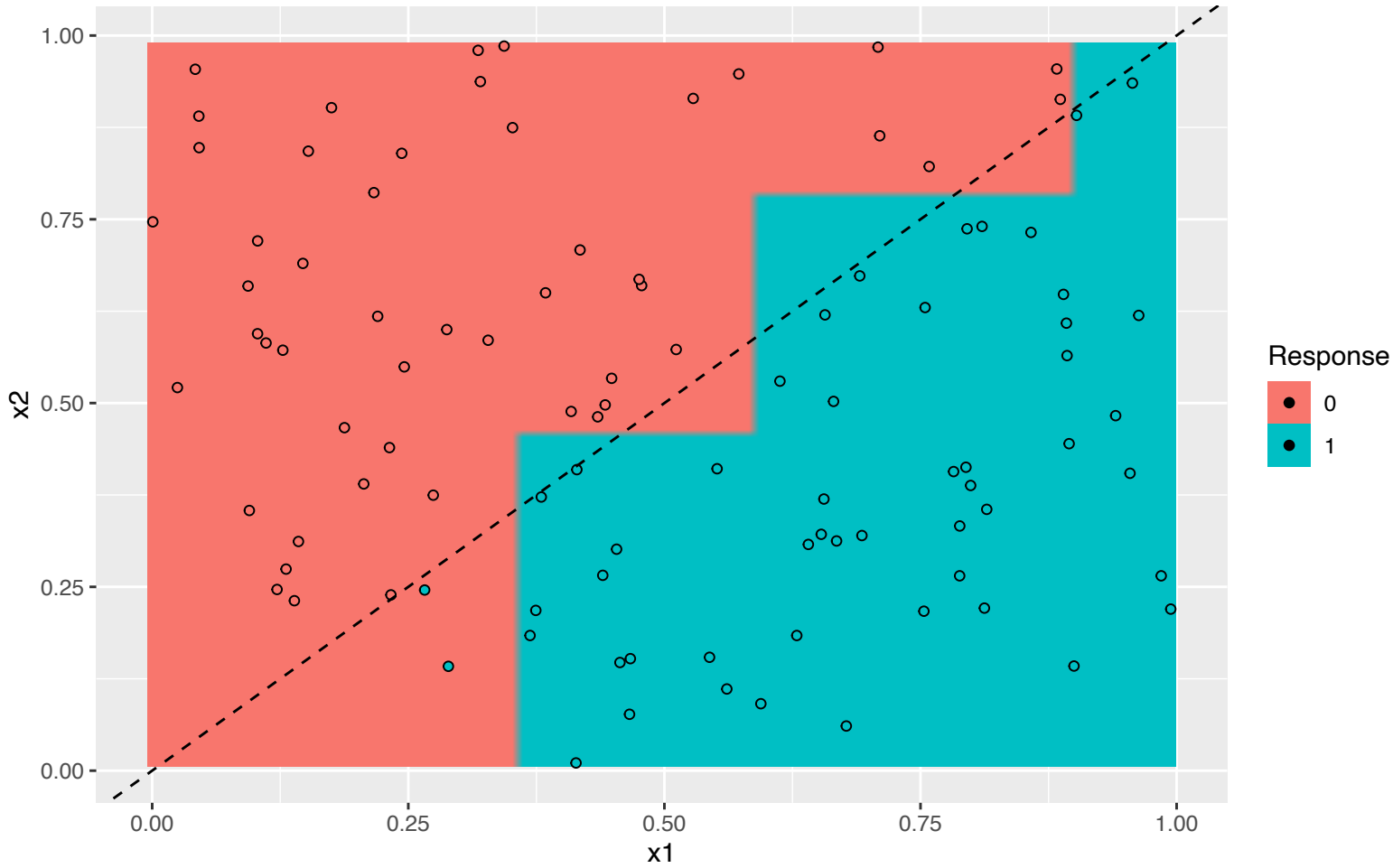
Pre- or Post-pruning?

- Generally good to do both
- Pre-pruning is faster and does not need any extra data
- Post-pruning is slower, needs extra data, but can "undo" some of the (potentially) bad greedy decisions made at the end of decision tree learning

ADVANTAGES

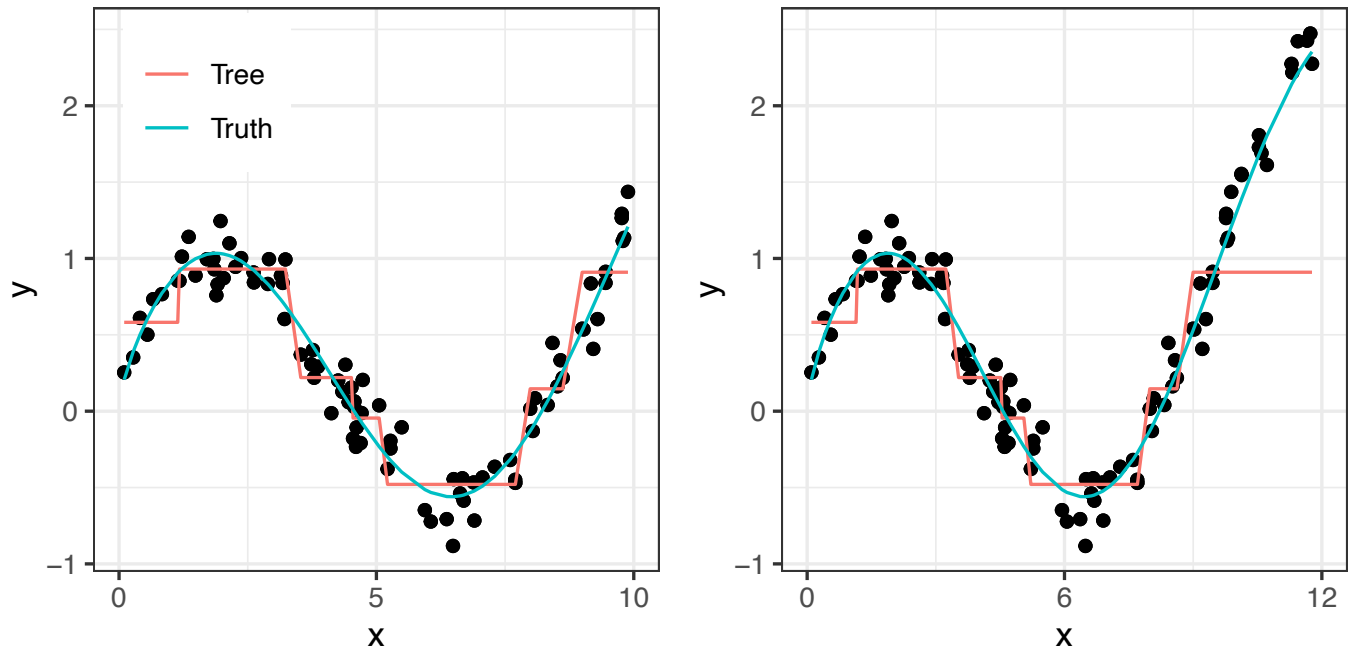
- Fairly easy to understand, interpret and visualize.
- Not much preprocessing required:
  - Automatic handling of non-numerical features
  - Automatic handling of missing values via surrogate splits
  - No problems with outliers in features
  - Monotone transformations do not affect the model fit: scaling becomes irrelevant
- Interaction effects between features are easily possible
- Can model discontinuities and non-linearities
- Performs automatic feature selection
- Relatively fast, scales well with larger data
- Flexibility through the definition of custom split criteria or leaf-node prediction rules: clustering trees, semi-supervised trees, density estimation, etc.

DISADVANTAGE: LINEAR DEPENDENCIES



Linear dependencies must be modeled over several splits. Logistic regression would model this easily with fewer parameters.

DISADVANTAGES: SMOOTH FUNCTIONS AND EXTRAPOLATION



Prediction functions of trees are never smooth as they are always step functions and do not extrapolate well beyond the training observations.

DISADVANTAGE: INSTABILITY

- High instability (variance) of the trees
- Small changes in the data may lead to very different splits/trees
- This leads to a) less trust in interpretability b) is a reason why the prediction error of trees is usually comparably high

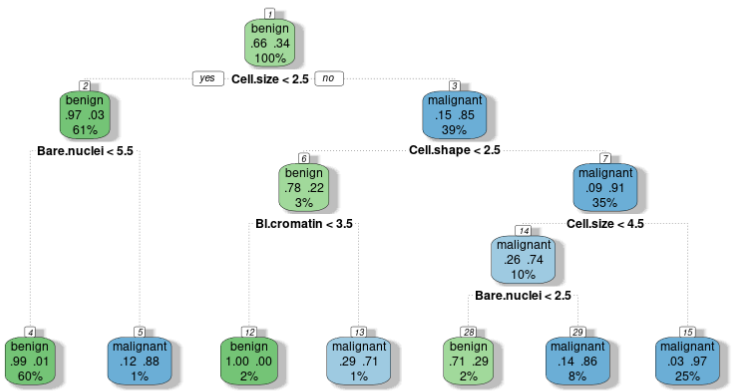
Consider the Wisconsin Breast Cancer data set with 699 observations on 9 features and binary target ("benign" vs. "malignant"). We fit two trees: (A) with the full data set and (B) where we eliminated a single observation. Results in label flip for 17 observations of the training data:

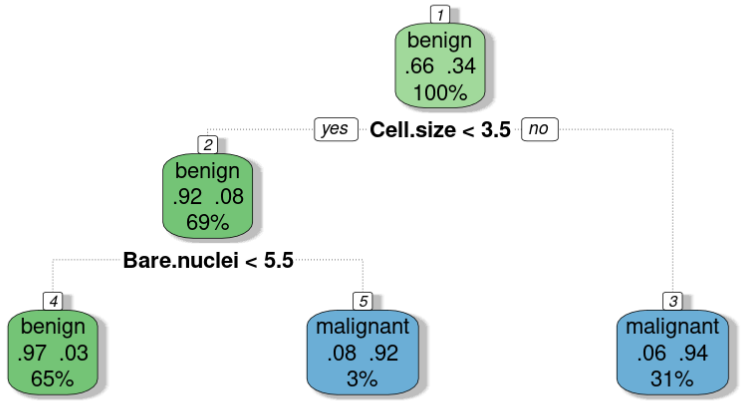
	benign	malignant
benign	445	6
malignant	11	236

Rows: Predictions of (A), columns: Predictions of (B)

DISADVANTAGE: INSTABILITY

The resulting decision trees look very different:



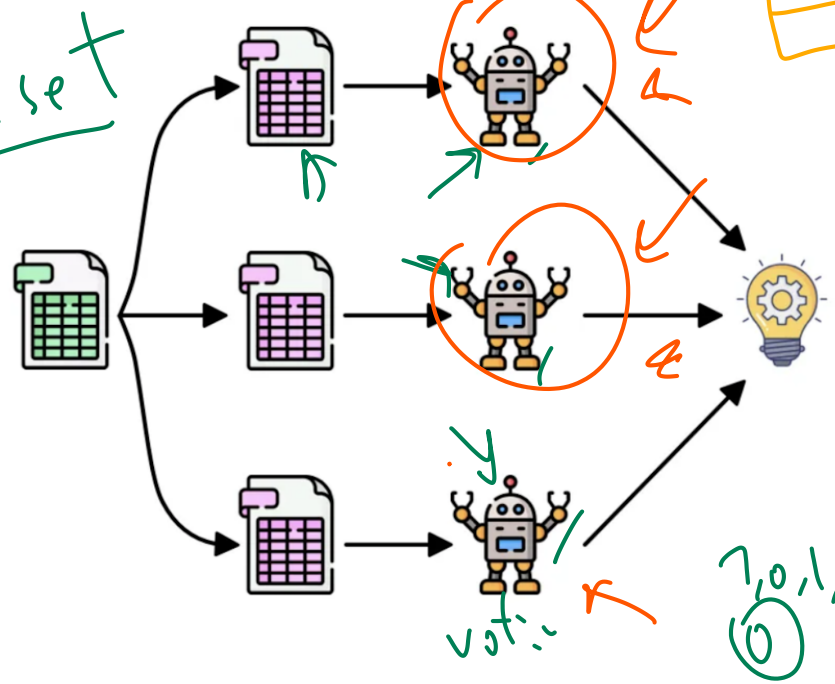


CART IN PRACTICE

- Compared to other learners CART has suboptimal predictive performance, mainly because of the problems previously shown.
- However, most disadvantages can be overcome when trained in ensembles: bagging or random forests.
- Furthermore, trees are attractive tools if an interpretable model is desired or legally required.

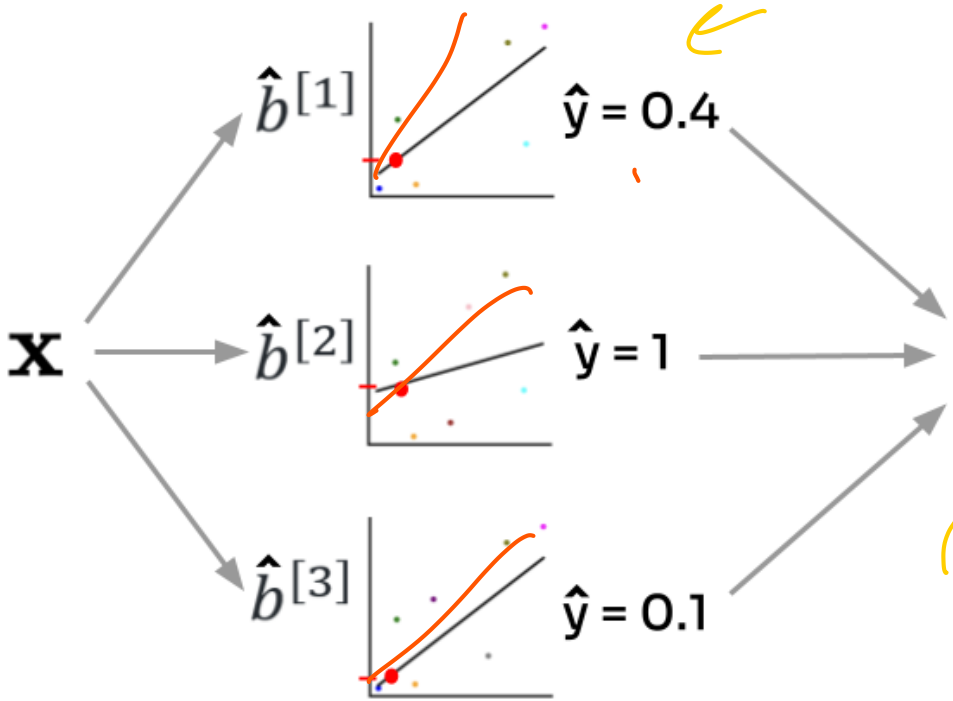
BAGGING

- Bagging is short for **B**ootstrap **Agg**regation
- **Ensemble method**, combines models into large "meta-model"; ensembles usually better than single **base learner**
- Homogeneous ensembles always use same BL class (e.g. CART), heterogeneous ensembles can use different classes
- Bagging is homogeneous



PREDICTING WITH A BAGGED ENSEMBLE

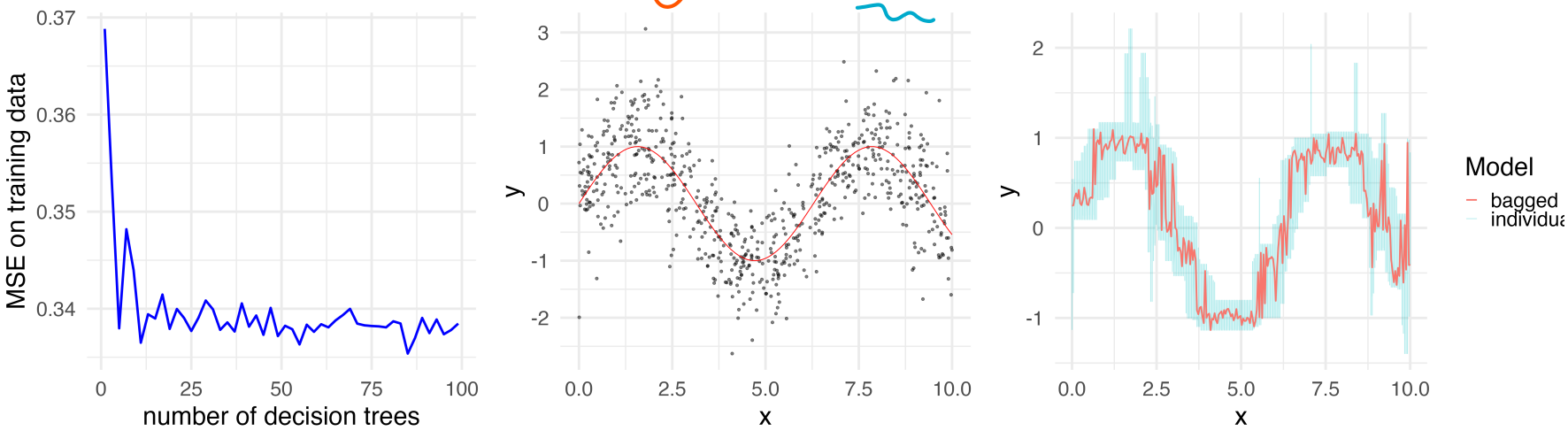
Average predictions of *M* fitted models for ensemble:
(here: regression)



$$\hat{f}(\mathbf{x}) = \frac{0.4 + 1 + 0.1}{3} = 0.5$$

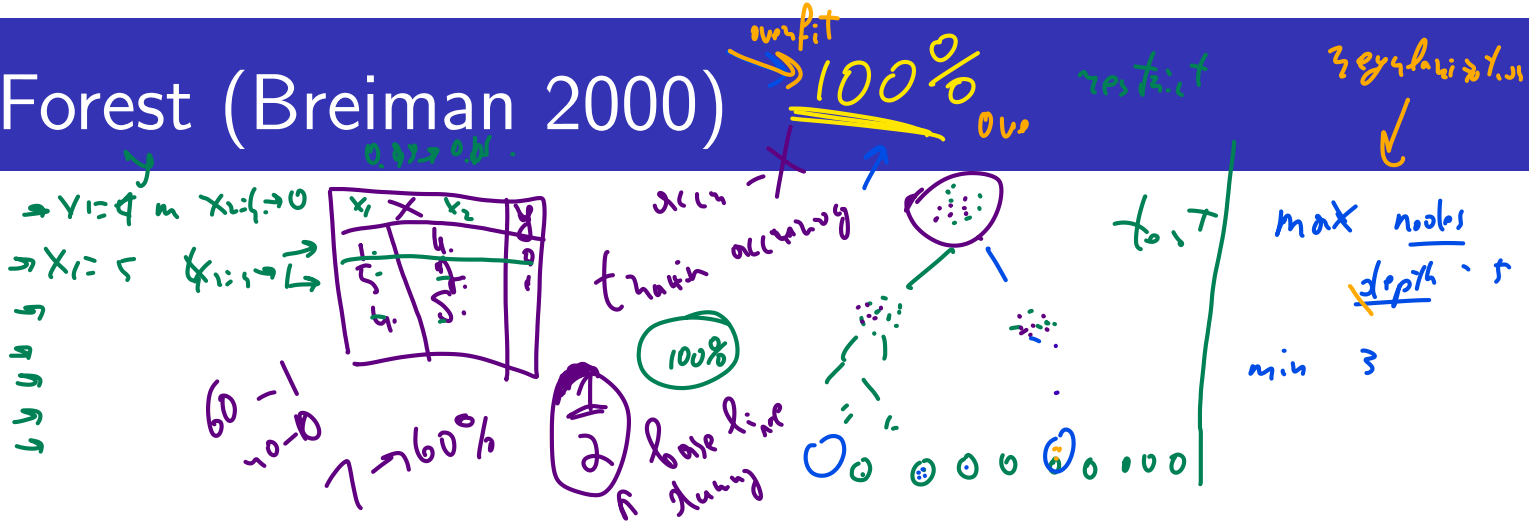
WHY/WHEN DOES BAGGING HELP?

- Bagging reduces the variability of predictions by averaging the outcomes from multiple BL models
- It is particularly effective when the errors of a BL are mainly due to (random) variability rather than systematic issues

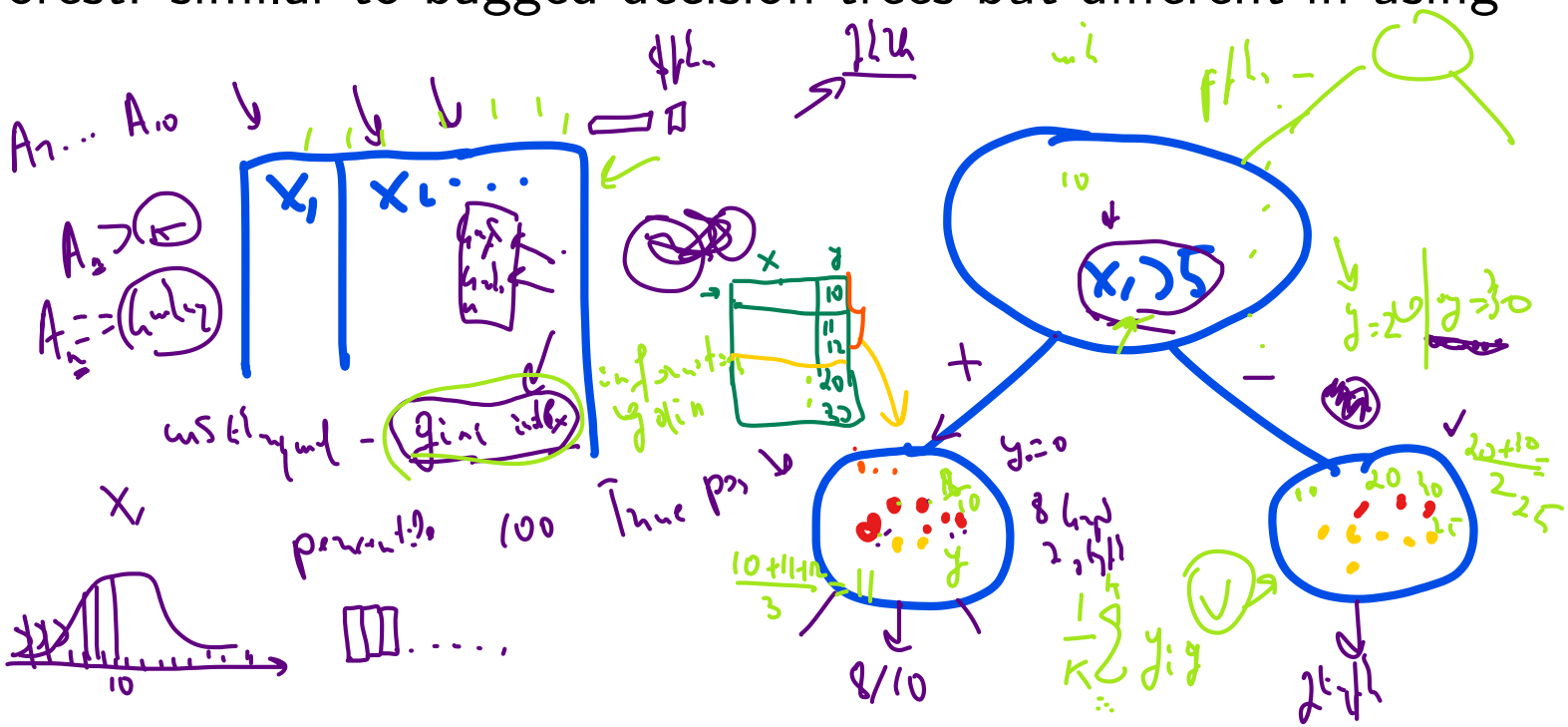


- Increasing **nr. of BLs** improves performance, up to a point, optimal ensemble size depends on inducer and data distribution

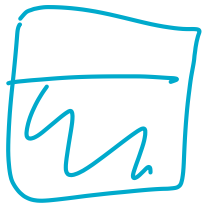
Random Forest (Breiman 2000)



- Random forest: similar to bagged decision trees but different in using features



Random Forest (Breiman 2000)

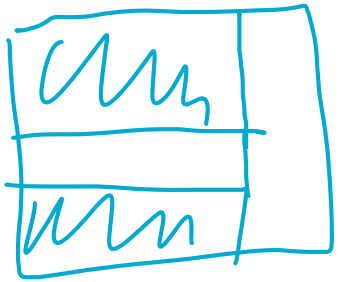


- Random forest: similar to bagged decision trees but different in using features
- **In each recursive step** of learning decision trees:

Random Forest (Breiman 2000)

- Random forest: similar to bagged decision trees but different in using features
- **In each recursive step** of learning decision trees:
  - **Randomly select F features** out of all P given features

Random Forest (Breiman 2000)



- Random forest: similar to bagged decision trees but different in using features
- **In each recursive step** of learning decision trees:
  - **Randomly select F features** out of all P given features
  - Find the best split among these features

Random Forest (Breiman 2000)

- Random forest: similar to bagged decision trees but different in using features
- **In each recursive step** of learning decision trees:
  - **Randomly select F features** out of all P given features
  - Find the best split among these features
- Parameter F is usually fixed to be $F = \sqrt{P}$ for classification

Random Forest

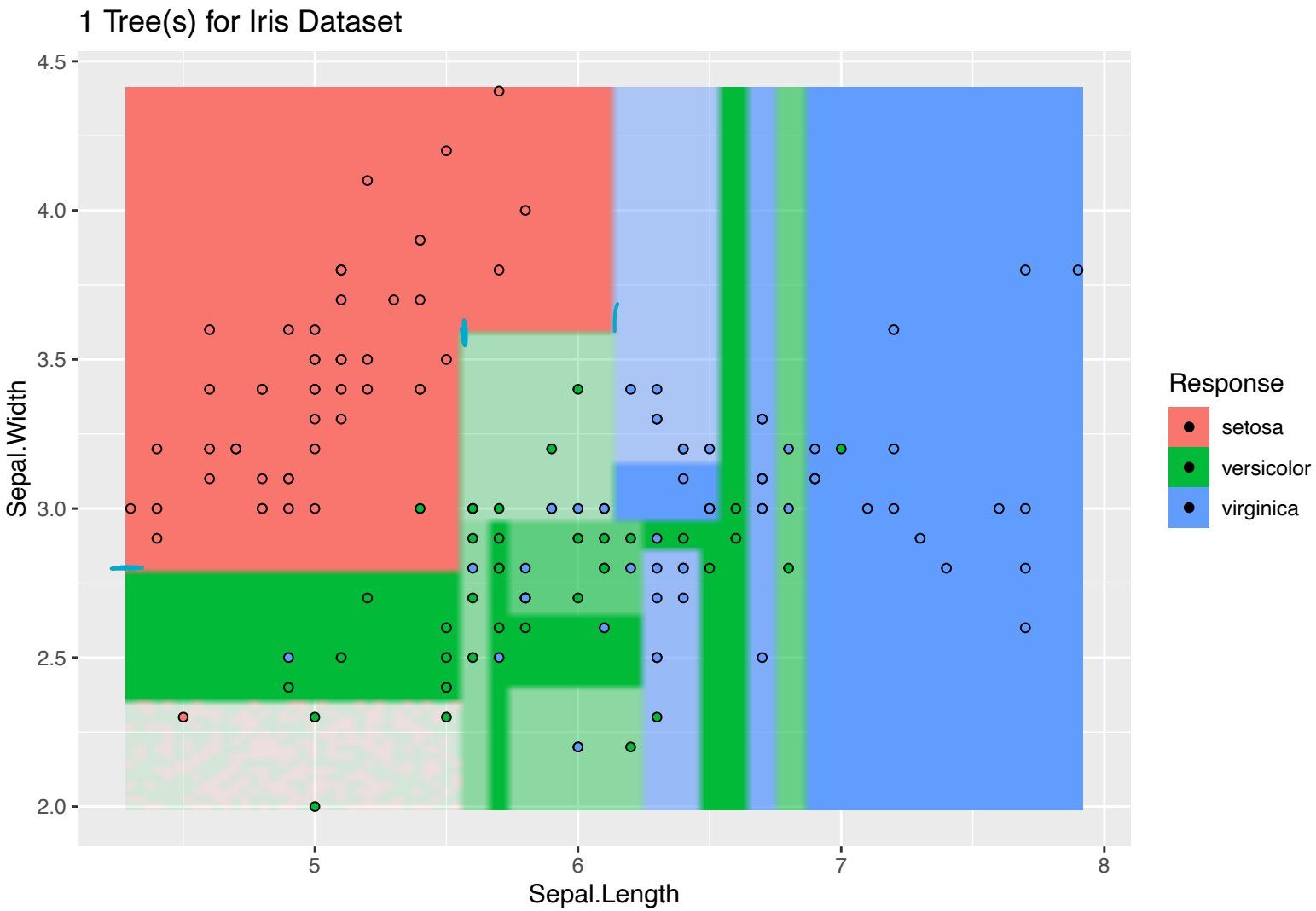
Algorithm RandomForest(D, T, d) – train an ensemble of tree models from bootstrap samples and random subspaces.

Input : data set D; ensemble size T; subspace dimension d.

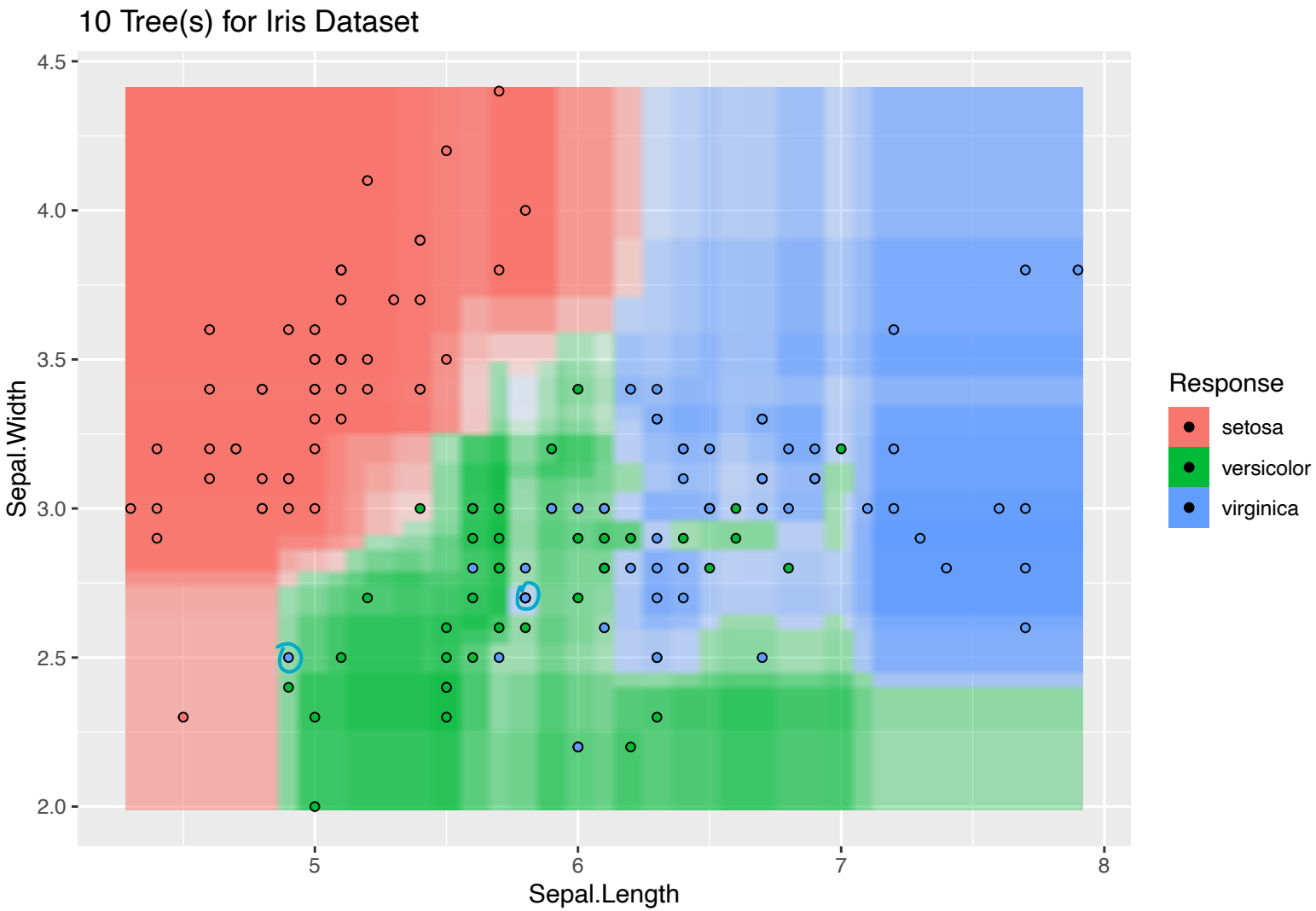
Output : ensemble of tree models whose predictions are to be combined by voting or averaging.

1 **for** $t = 1$ to T **do**
2 build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;
3 select d features at random and reduce dimensionality of D_t accordingly;
4 train a tree model M_t on D_t without pruning;
5 **end**
6 **return** $\{M_t | 1 \leq t \leq T\}$

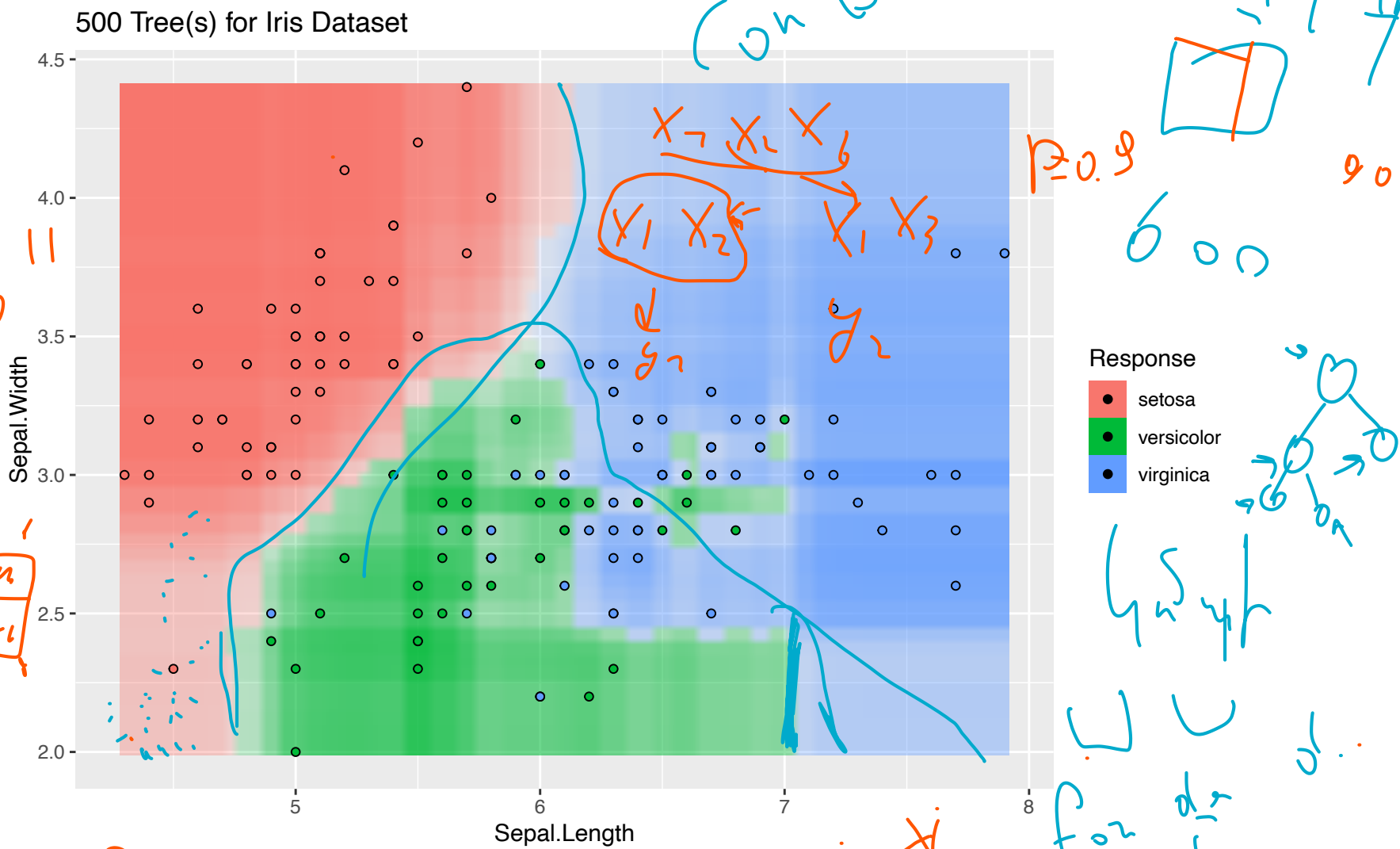
EFFECT OF ENSEMBLE SIZE



EFFECT OF ENSEMBLE SIZE



EFFECT OF ENSEMBLE SIZE



PROXIMITIES

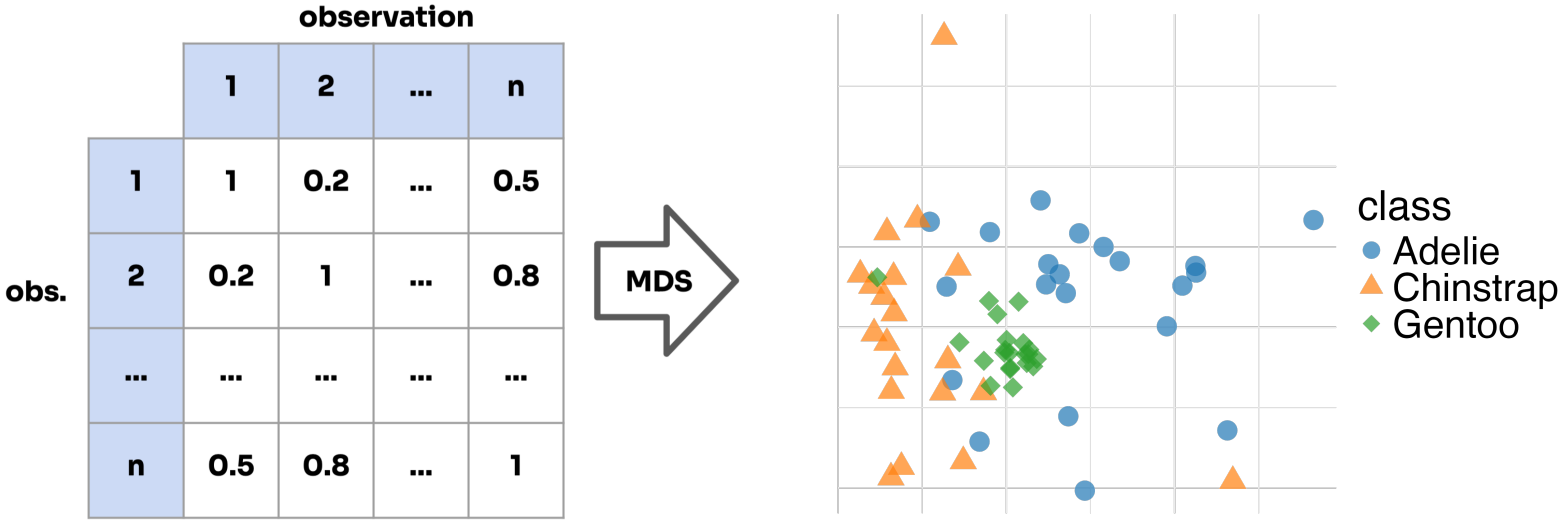
RFs have built-in similarity measure for pairs of observations:

ID	Color	Form	Length
1	yellow	oblong	14

ID	Color	Form	Length
2	brown	oblong	10

- After training, push all observations through each tree
- To calculate prox $\left(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}\right)$: Percentage of how often both points are placed in **same terminal node of a tree**
- Here: prox $\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}\right) = 2/3$
- All proximities are arranged in symmetric $n \times n$ matrix

VISUALIZING PROXIMITIES

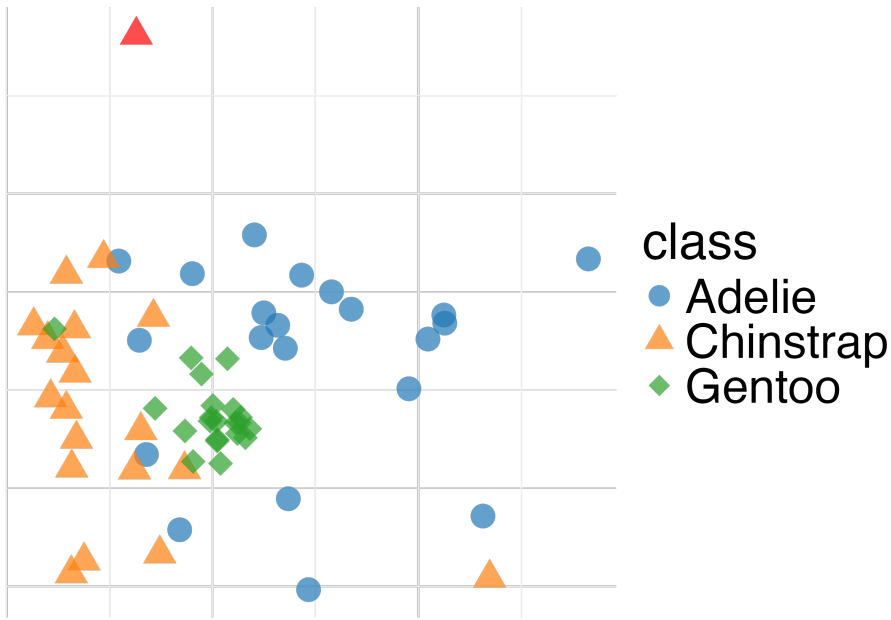


Can visualize the proximity matrix by projecting it into lower-dim. space, e.g., via multidim. scaling (might have to turn proximities into distances)

- Samples from same class usually form **identifiable clusters**
- **Offers some error-inspection**, e.g., Adelie has high within-class variance and has overlaps with other classes

OUTLIER DETECTION

- Can also be used to **locate outliers**
- Or mislabeled points, especially in manually labeled data sets



IMPUTING MISSING DATA

ID	Color	Form	Origin	Length
1	yellow	round	domestic	14
2	brown	oblong	imported	???
3	brown	oblong	imported	19
4	???	round	domestic	14

1. Replace missings per feature by median (of available values)
2. Compute proximities (NB: data has changed)
3. Replace missings in $\mathbf{x}^{(i)}$ by weighted average of non-missings; weights proportional to proximities

Steps 2 and 3 are iterated a few times.

IMPUTING MISSING DATA

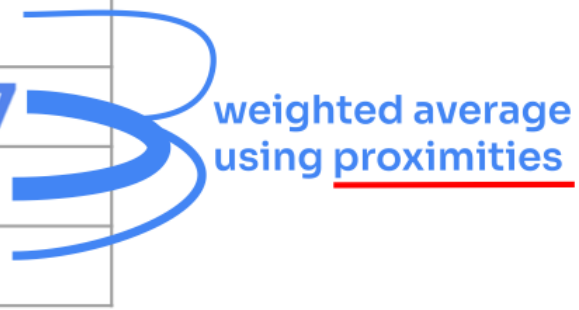
ID	Color	Form	Origin	Length
1	yellow	round	domestic	14
2	brown	oblong	imported	14
3	brown	oblong	imported	19
4	brown	round	domestic	14

1. Replace missings per feature by median (of available values)
2. Compute proximities (NB: data has changed)
3. Replace missings in $\mathbf{x}^{(i)}$ by weighted average of non-missings; weights proportional to proximities

Steps 2 and 3 are iterated a few times.

IMPUTING MISSING DATA

ID	Color	Form	Origin	Length
1	yellow	round	domestic	14
2	brown	oblong	imported	17
3	brown	oblong	imported	19
4	brown	round	domestic	14



1. Replace missings per feature by median (of available values)
2. Compute proximities (NB: data has changed)
3. Replace missings in $\mathbf{x}^{(i)}$ by weighted average of non-missings; weights proportional to proximities

Steps 2 and 3 are iterated a few times.

Further reading

- Visual introduction of Decision Trees:
  http://www.r2d3.us/visual-intro-to-machine-learning-part-1
- Detailed lecture notes by UoT:
  https://www.cs.toronto.edu/~axgao/cs486686_f21/lecture_notes/Lecture_07_on_Decision_Trees.pdf
- Decision Tree for Regression:
  https://saedsayad.com/decision_tree_reg.htm