



INTRODUCTION TO MACHINE LEARNING

Decision tree learning

Task of classification

- Automatically assign **class** to observations with **features**
- Observation: **vector of features**, with a **class**
- Automatically assign **class** to new observation with **features**, using previous observations
- **Binary classification:** two classes
- **Multiclass classification:** more than two classes

Example

- A dataset consisting of **persons**
- **Features:** age, weight and income
- **Class:**
 - **binary:** happy or not happy
 - **multiclass:** happy, satisfied or not happy

Examples of features

- Features can be numerical
 - age: 23, 25, 75, ...
 - height: 175.3, 179.5, ...
- Features can be categorical
 - travel_class: first class, business class, coach class
 - smokes?: yes, no

The decision tree

- Suppose you're classifying patients as **sick or not sick**
- Intuitive way of classifying: ask questions

Is the patient young or old?

The decision tree

- Suppose you're classifying patients as **sick or not sick**
- Intuitive way of classifying: ask questions

Is the patient young or old?

Old

The decision tree

- Suppose you're classifying patients as **sick or not sick**
- Intuitive way of classifying: ask questions

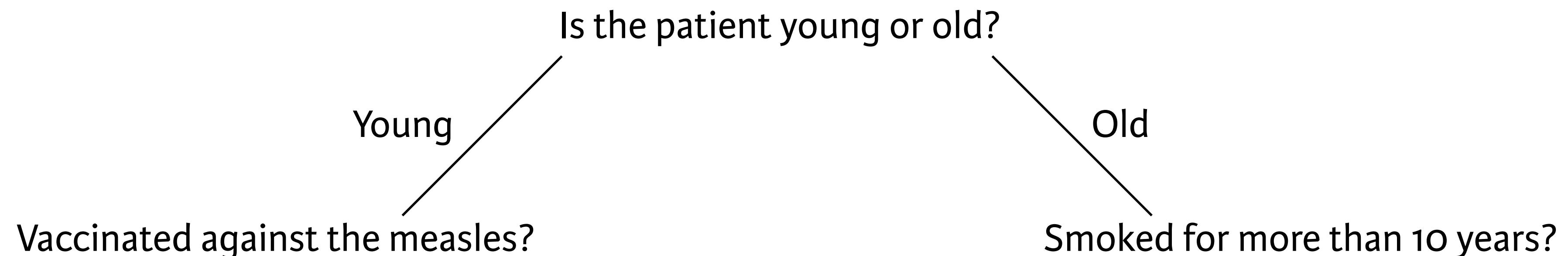
Is the patient young or old?

Old

Smoked for more than 10 years?

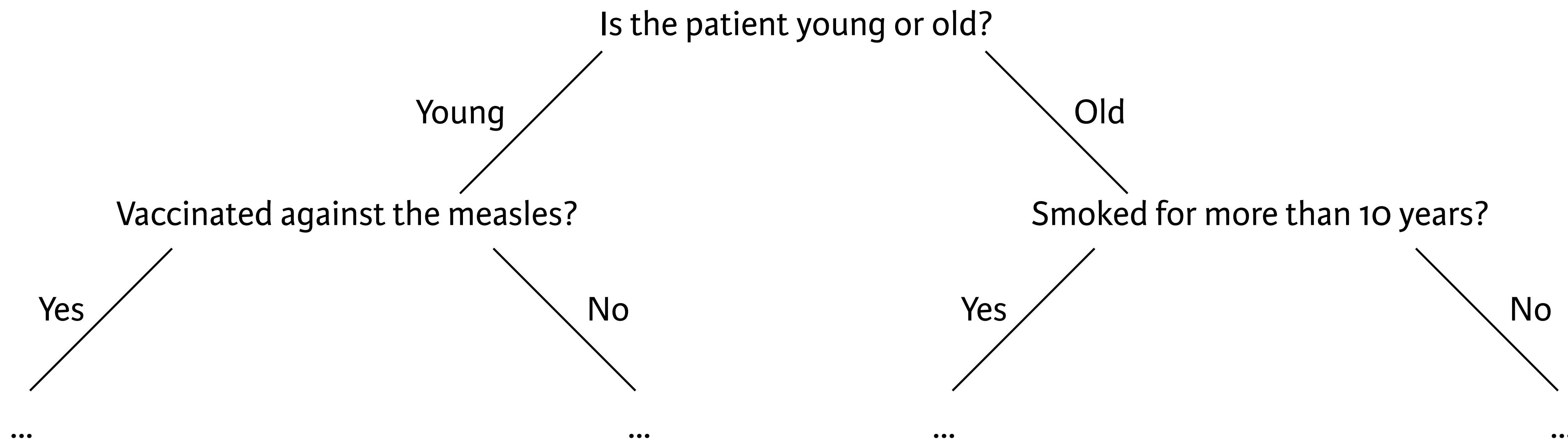
The decision tree

- Suppose you're classifying patients as **sick or not sick**
- Intuitive way of classifying: ask questions



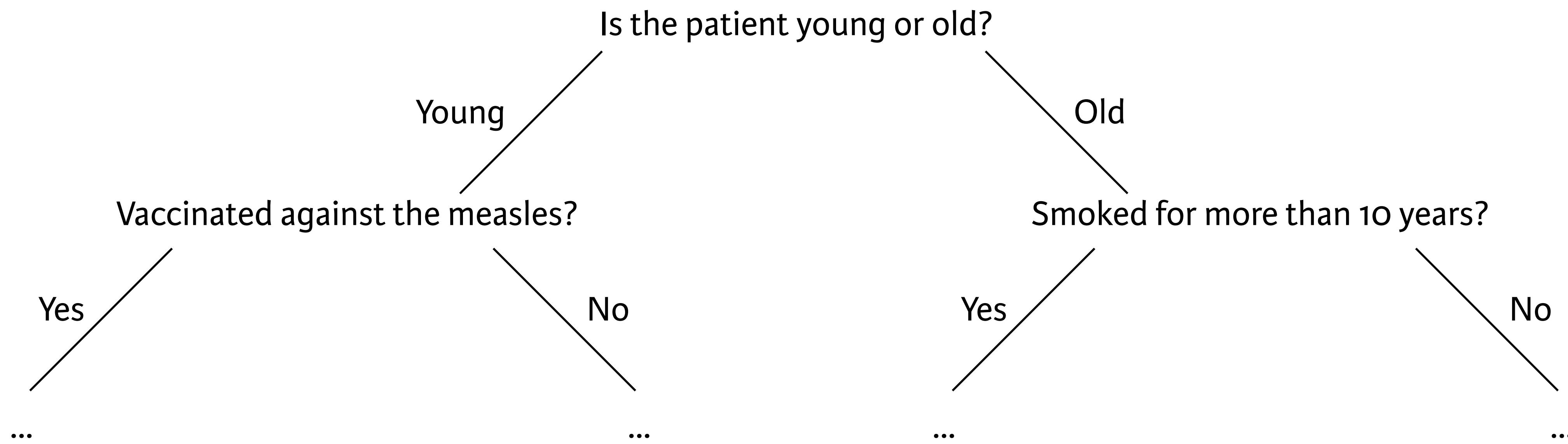
The decision tree

- Suppose you're classifying patients as **sick or not sick**
- Intuitive way of classifying: ask questions



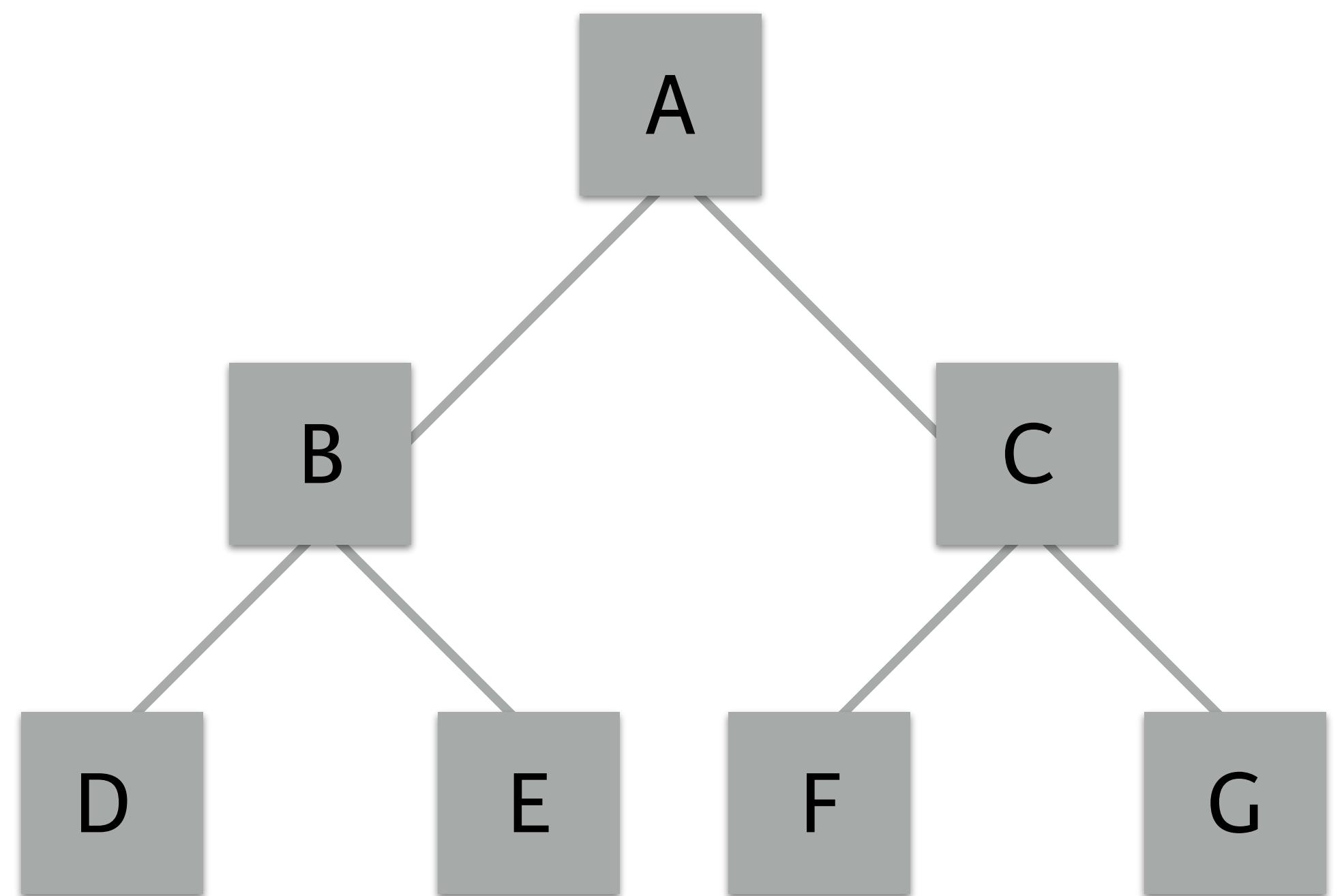
The decision tree

- Suppose you're classifying patients as **sick or not sick**
- Intuitive way of classifying: ask questions



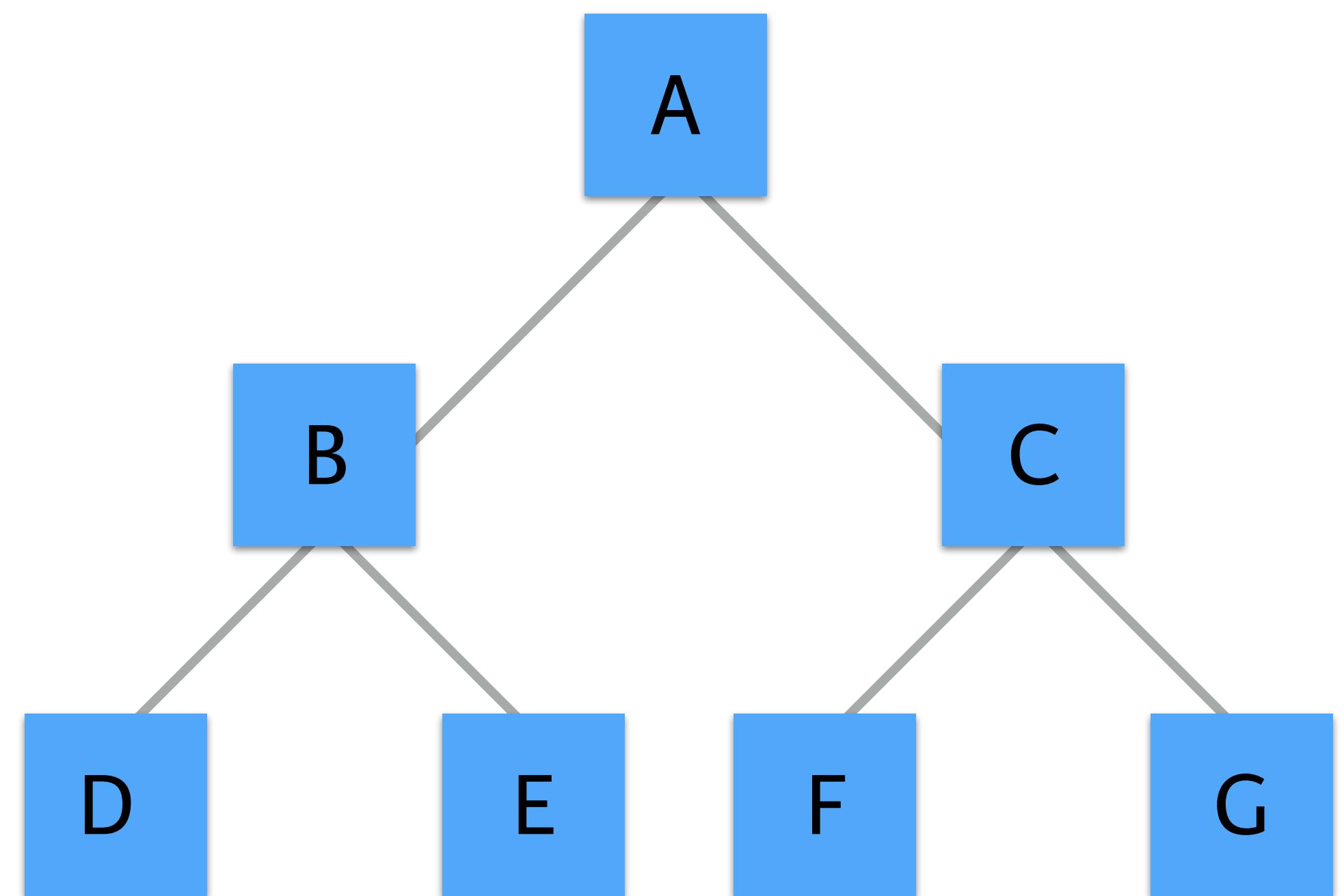
It's a decision tree!!!

Define the tree



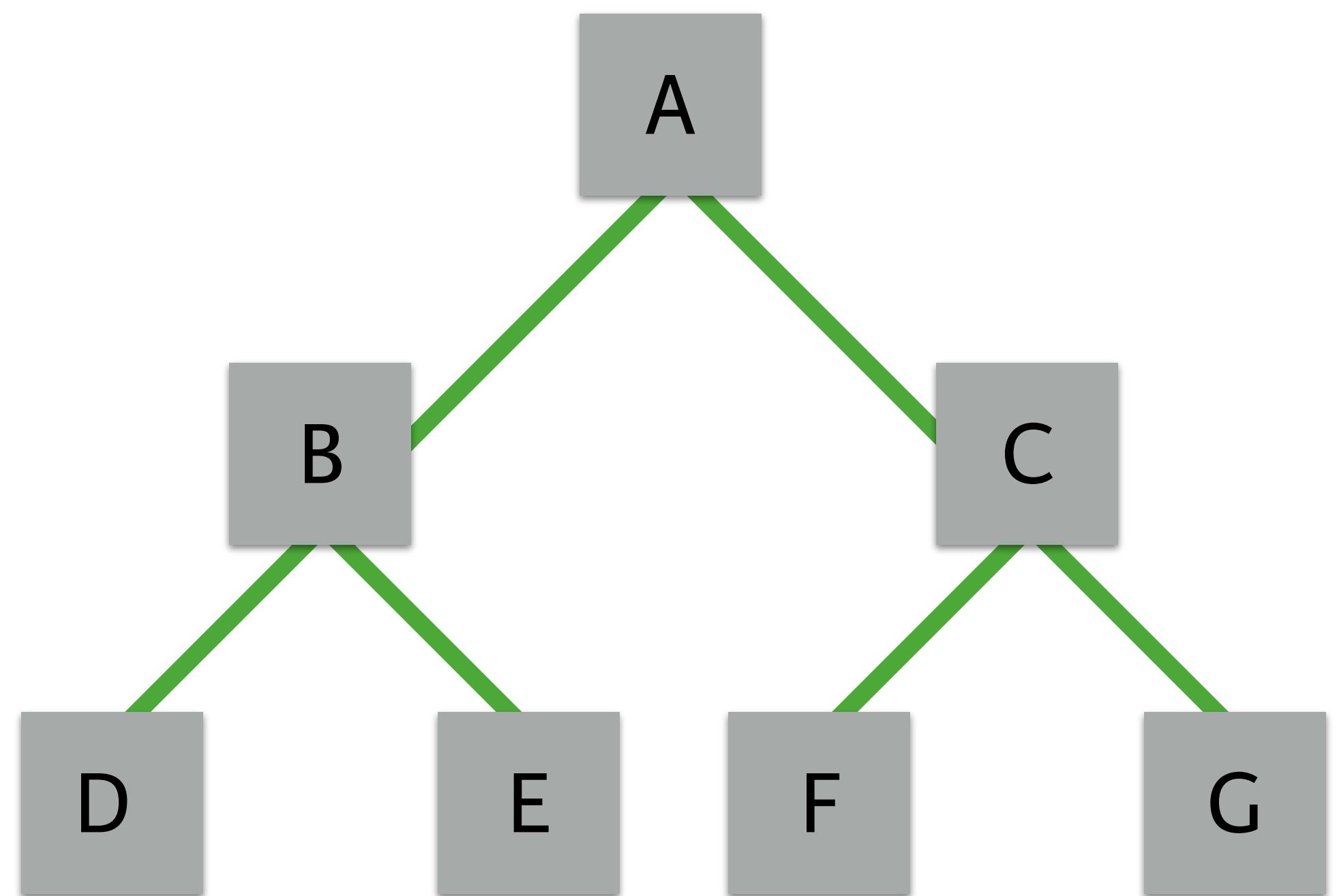
Define the tree

Nodes

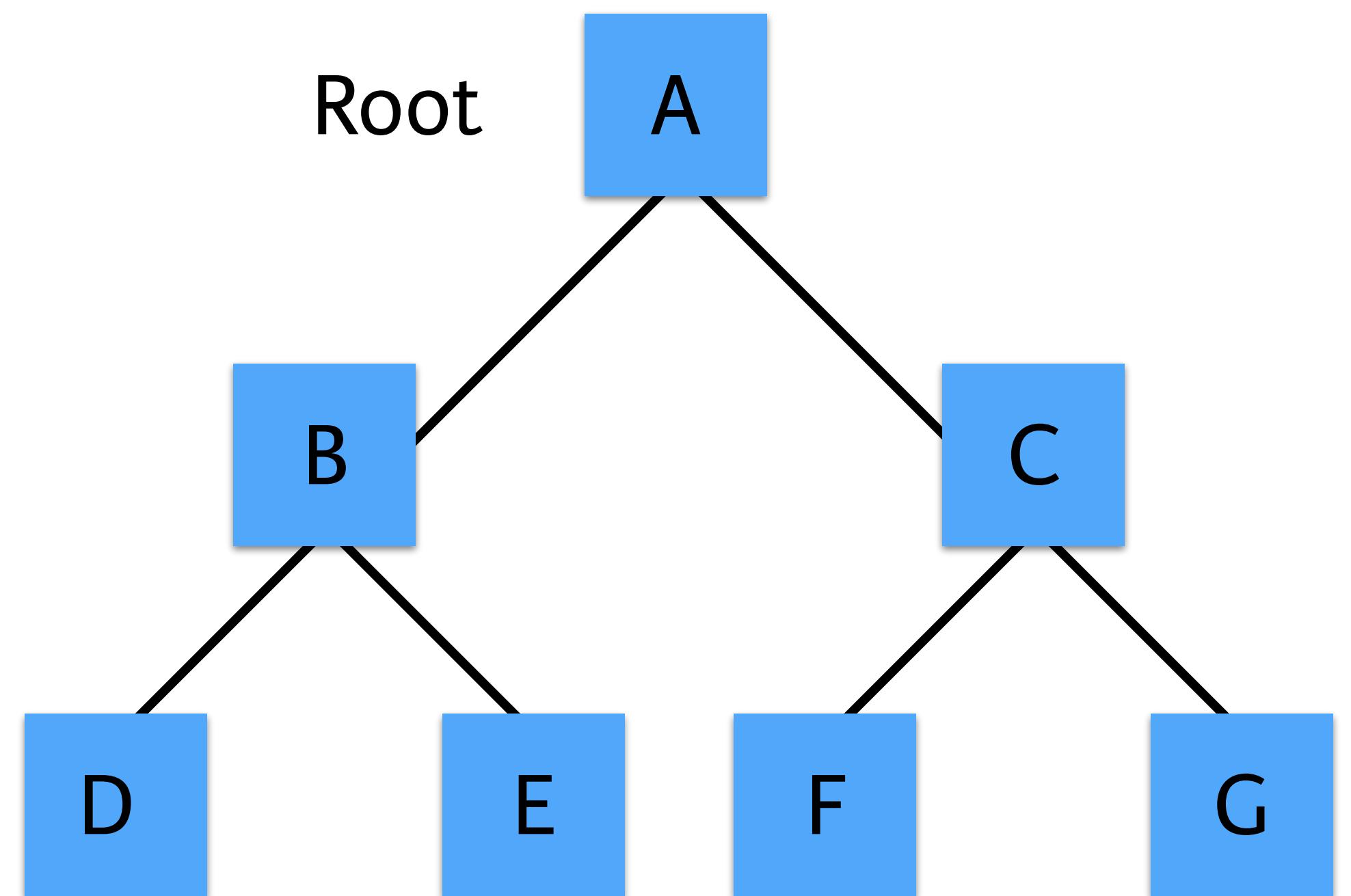


Define the tree

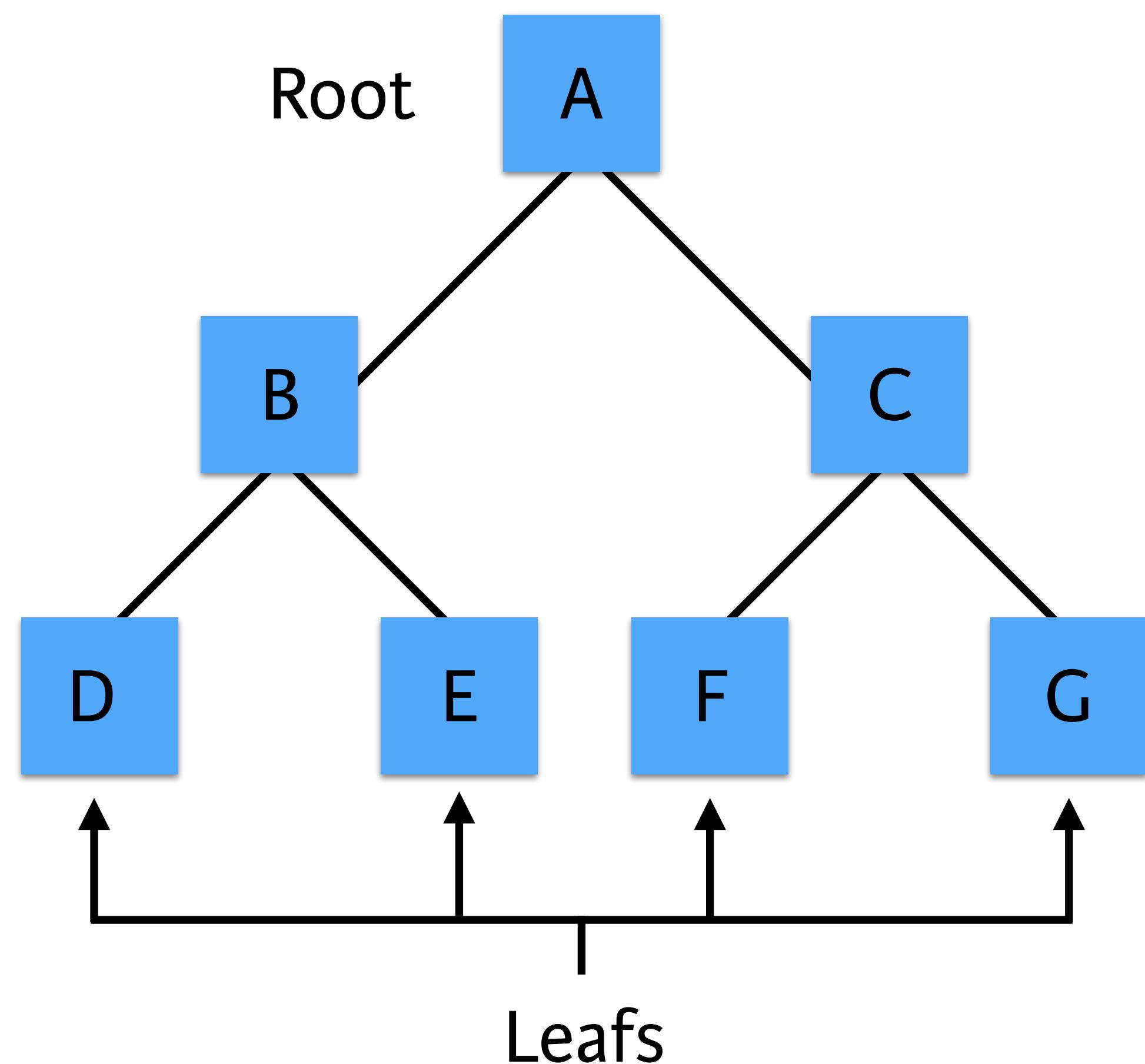
Edges



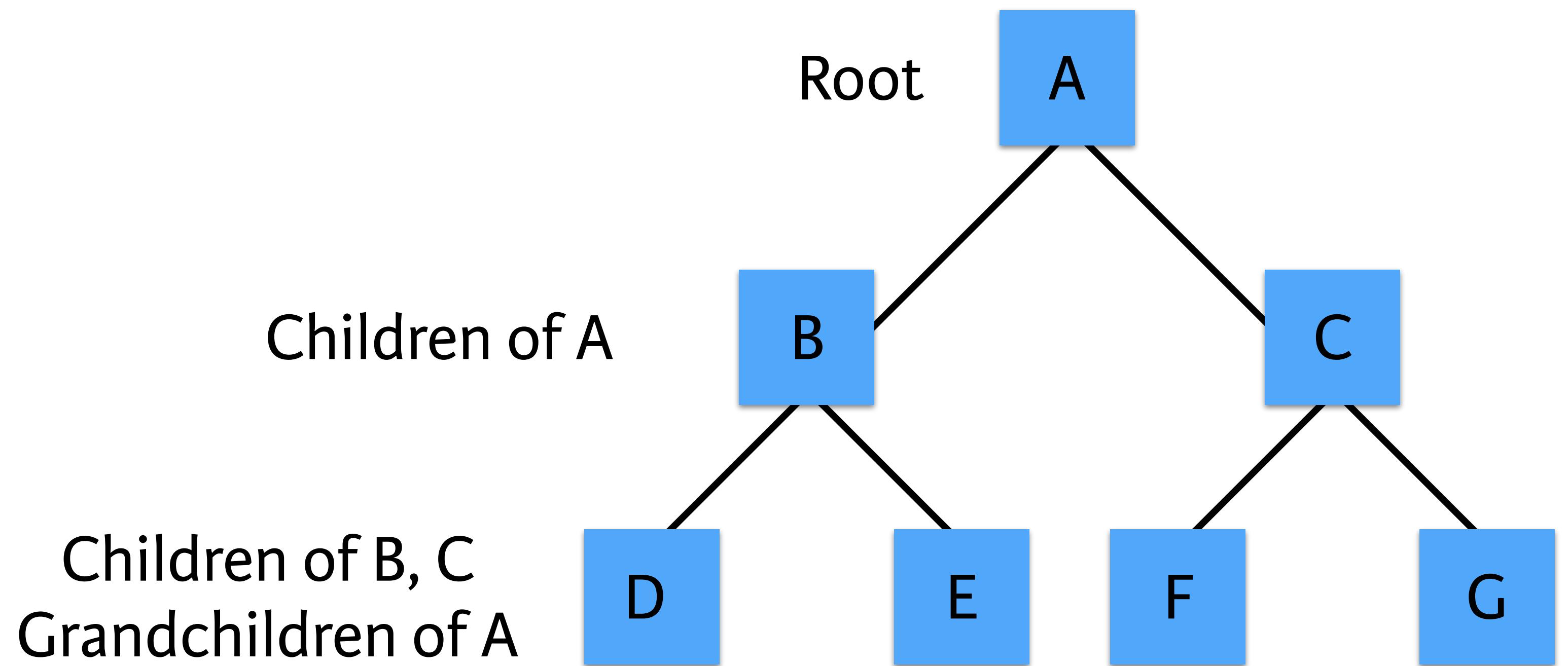
Define the tree



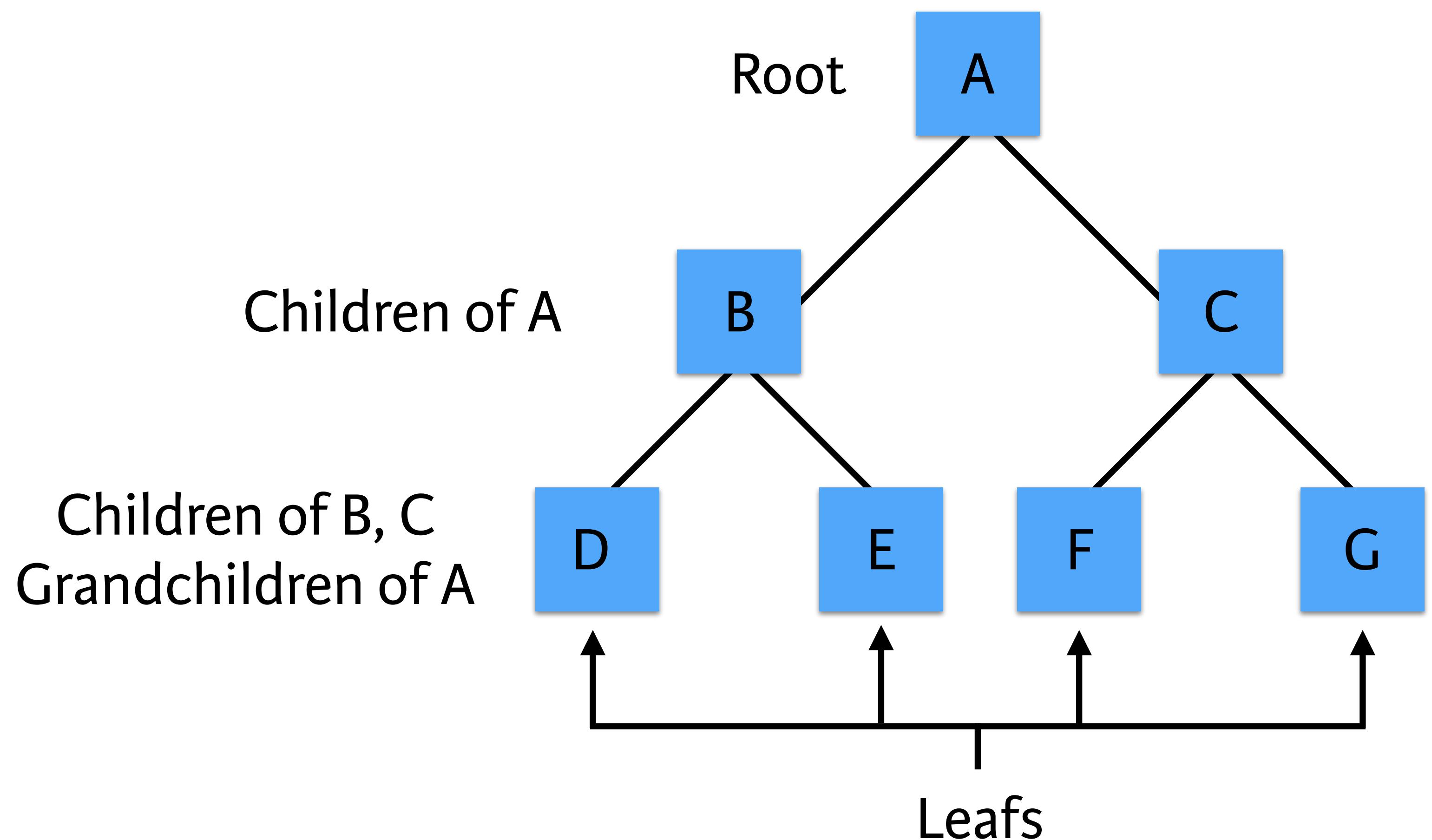
Define the tree



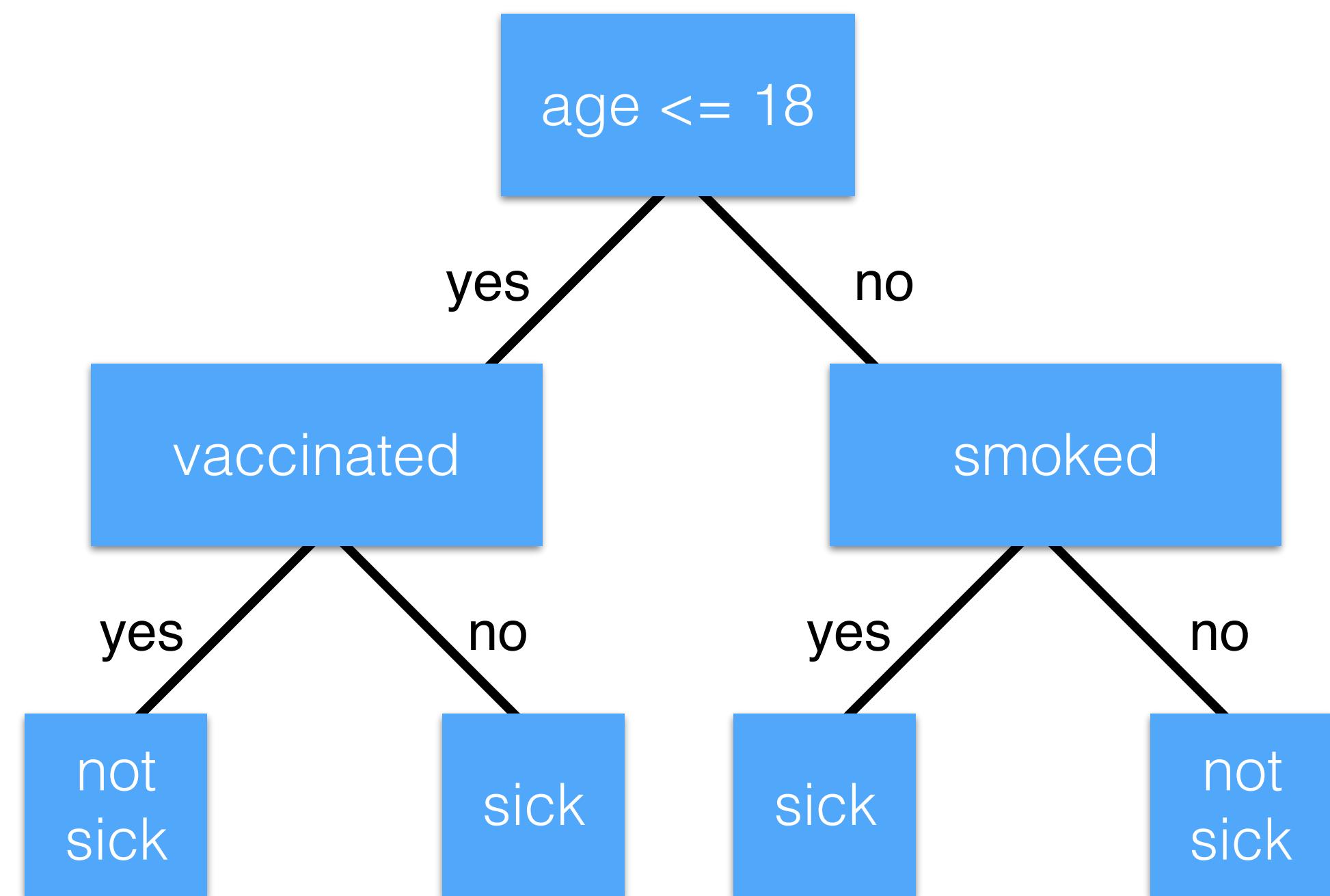
Define the tree



Define the tree

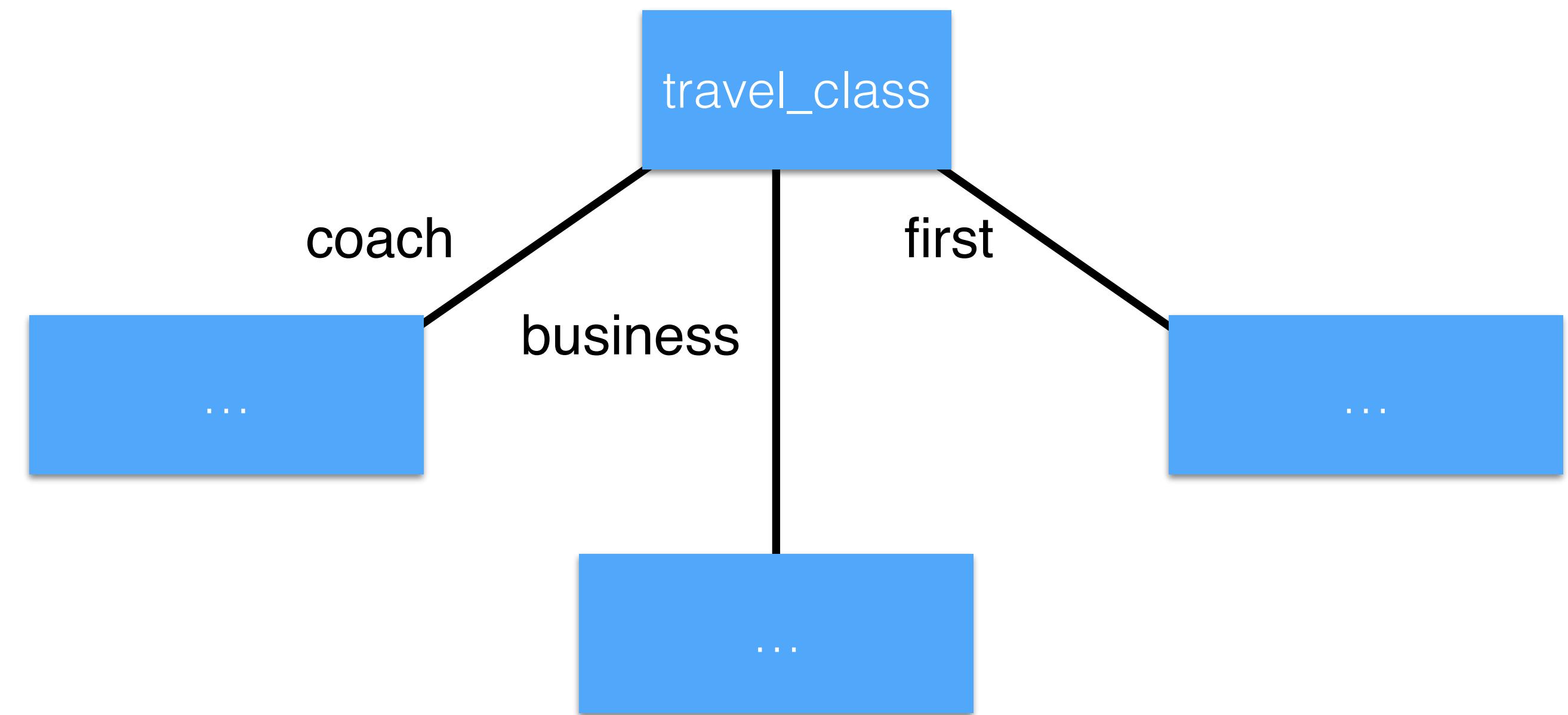


Questions to ask



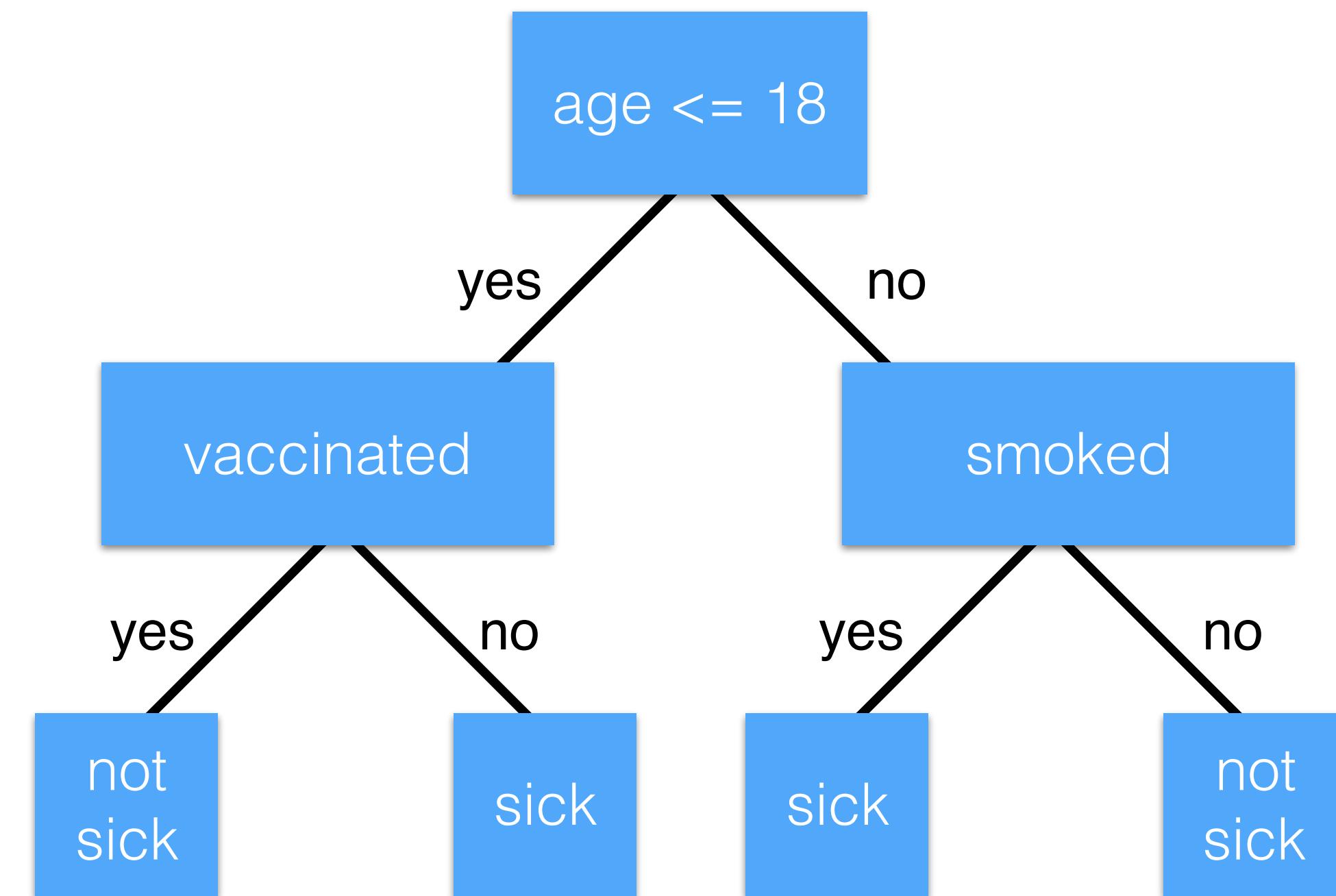
Categorical feature

- Can be a feature test on itself
- `travel_class`: coach, business or first



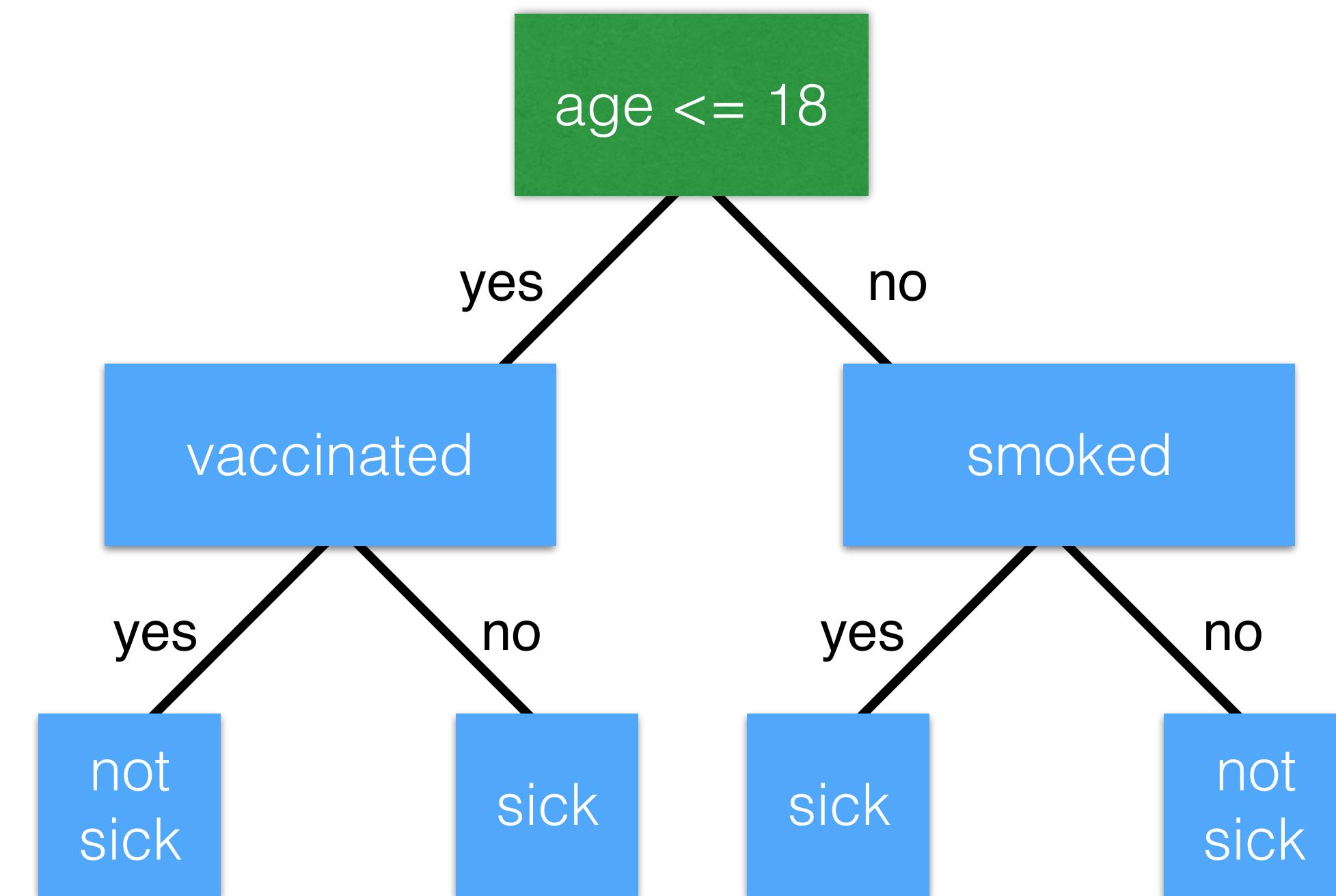
Classifying with the tree

Observation: patient of 40 years, vaccinated and didn't smoke



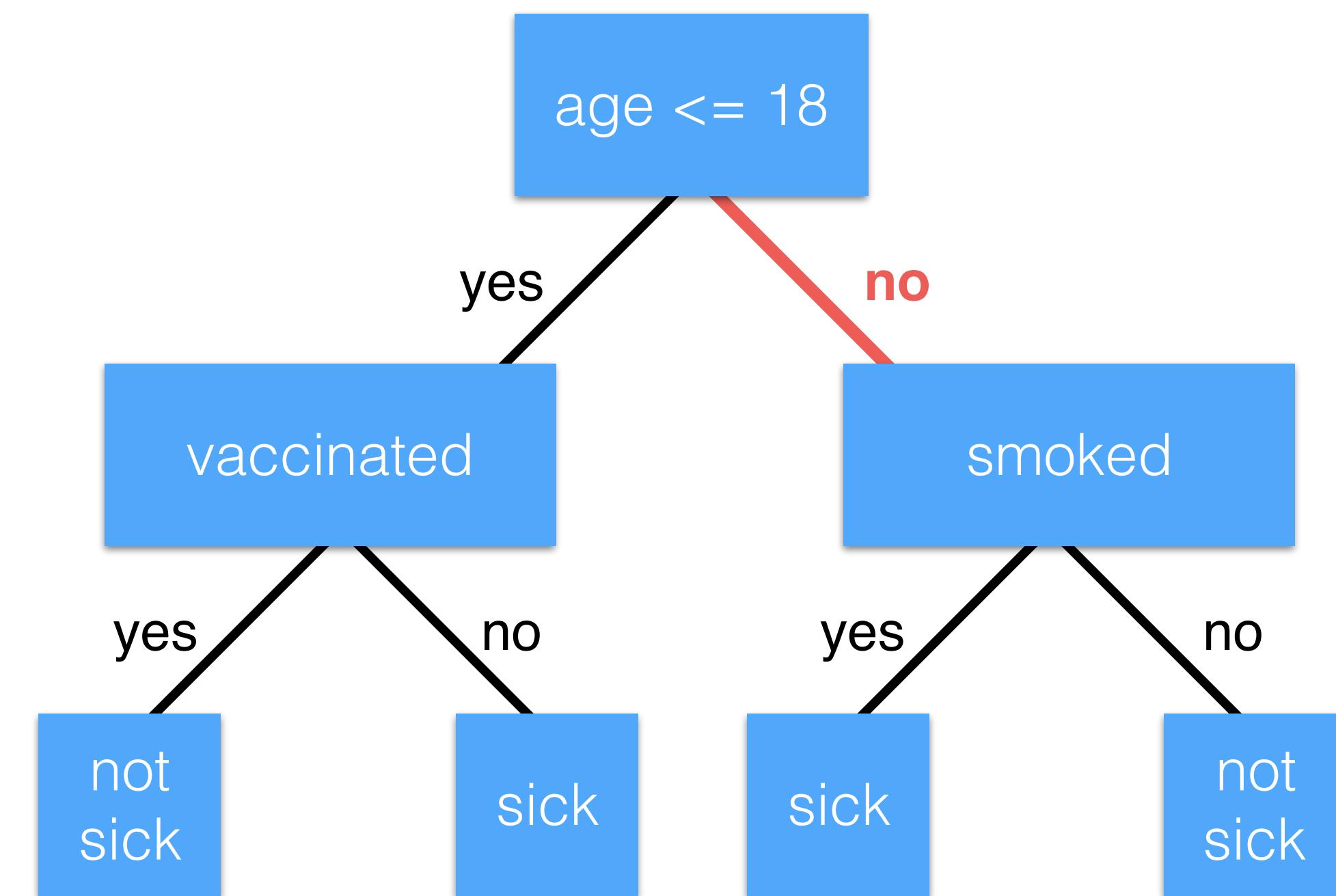
Classifying with the tree

Observation: patient of 40 years, vaccinated and didn't smoke



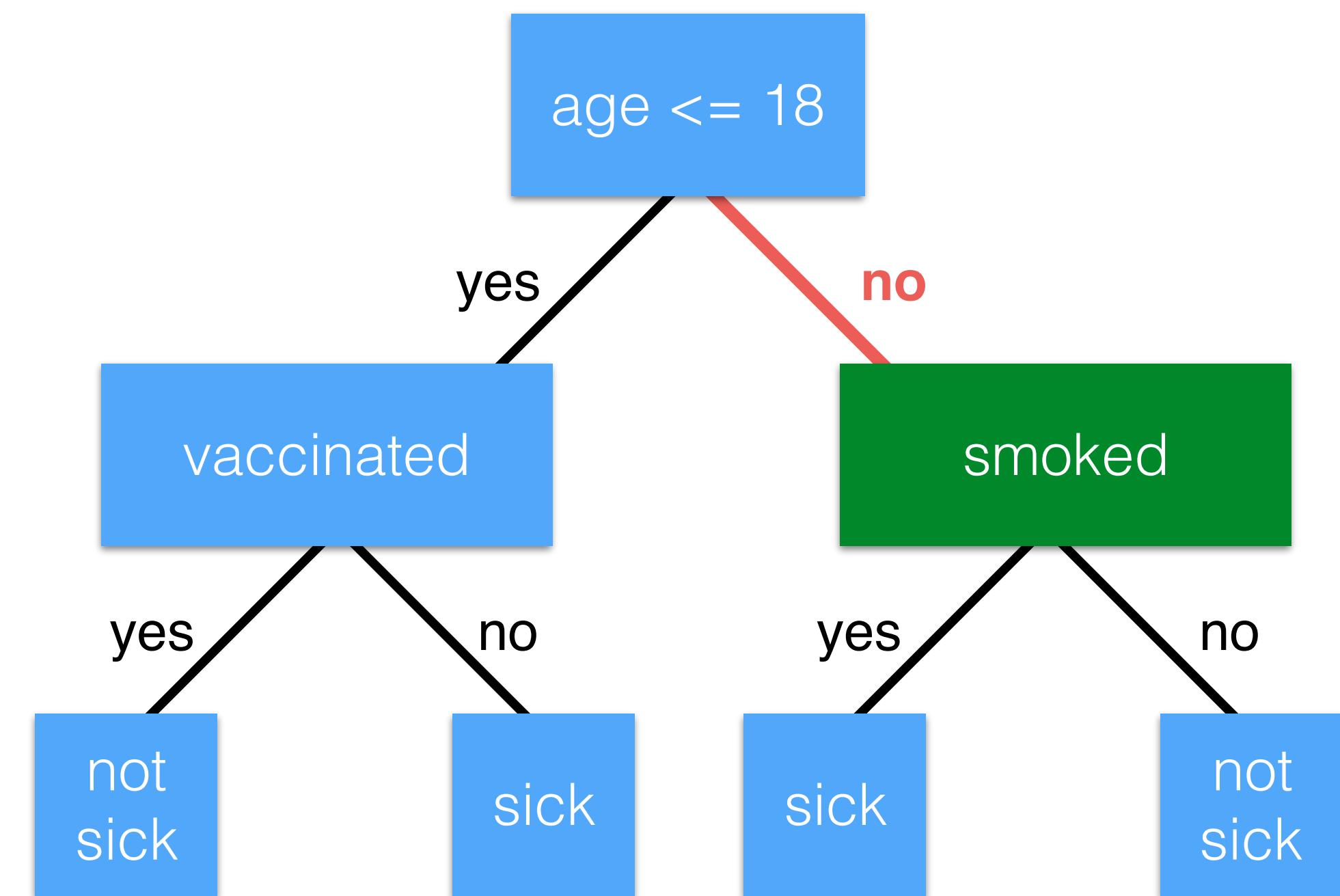
Classifying with the tree

Observation: patient of 40 years, vaccinated and didn't smoke



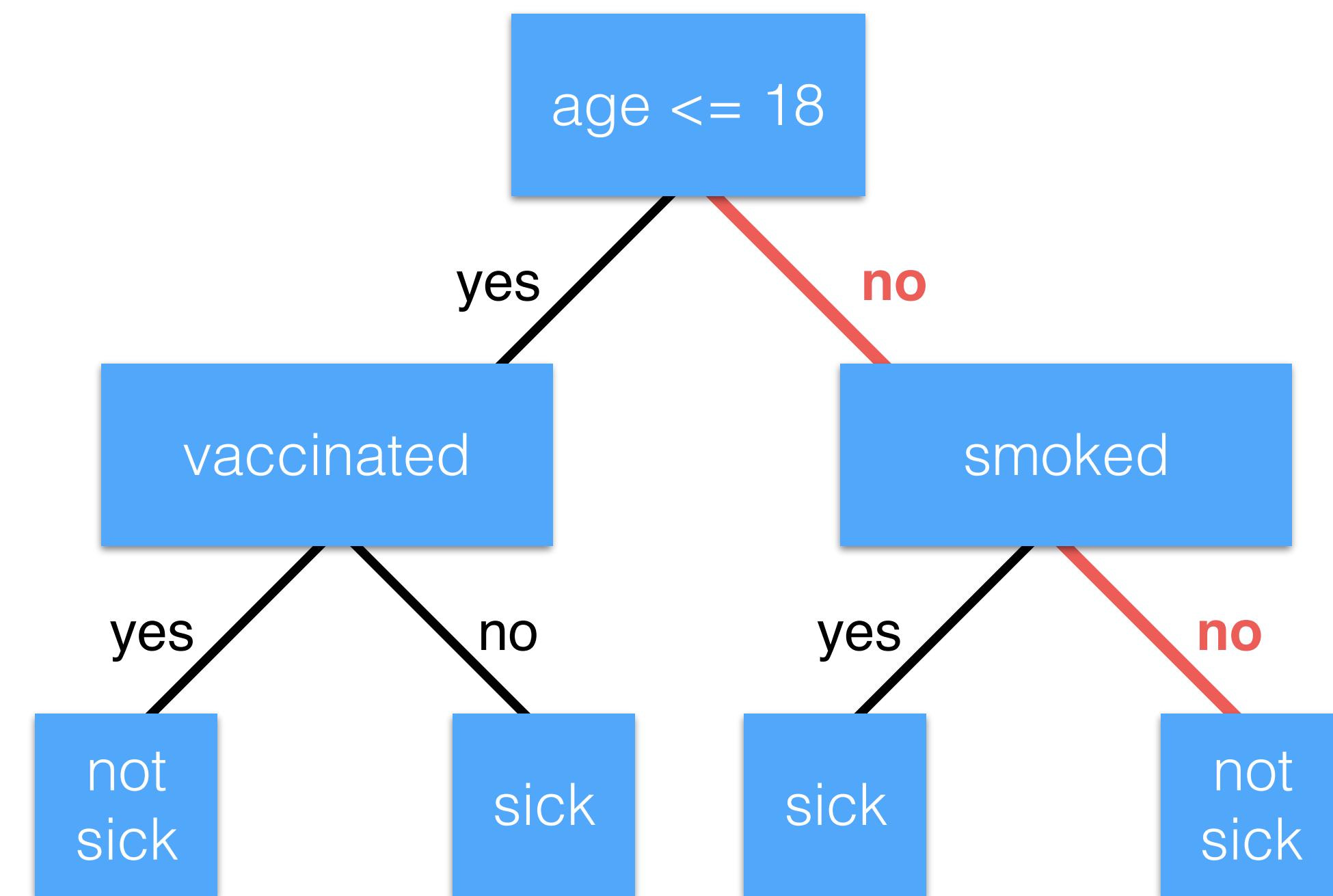
Classifying with the tree

Observation: patient of 40 years, vaccinated and didn't smoke



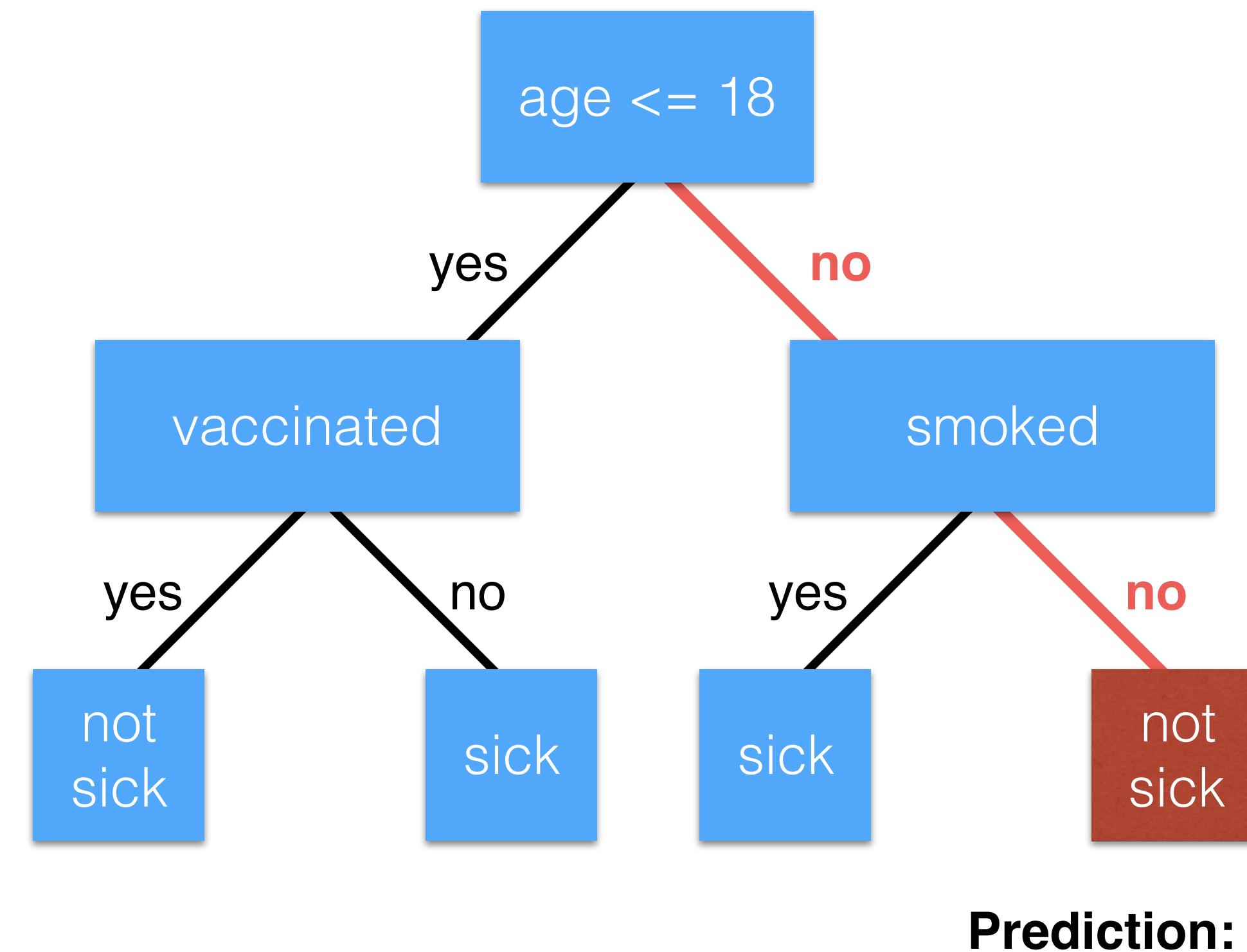
Classifying with the tree

Observation: patient of 40 years, vaccinated and didn't smoke



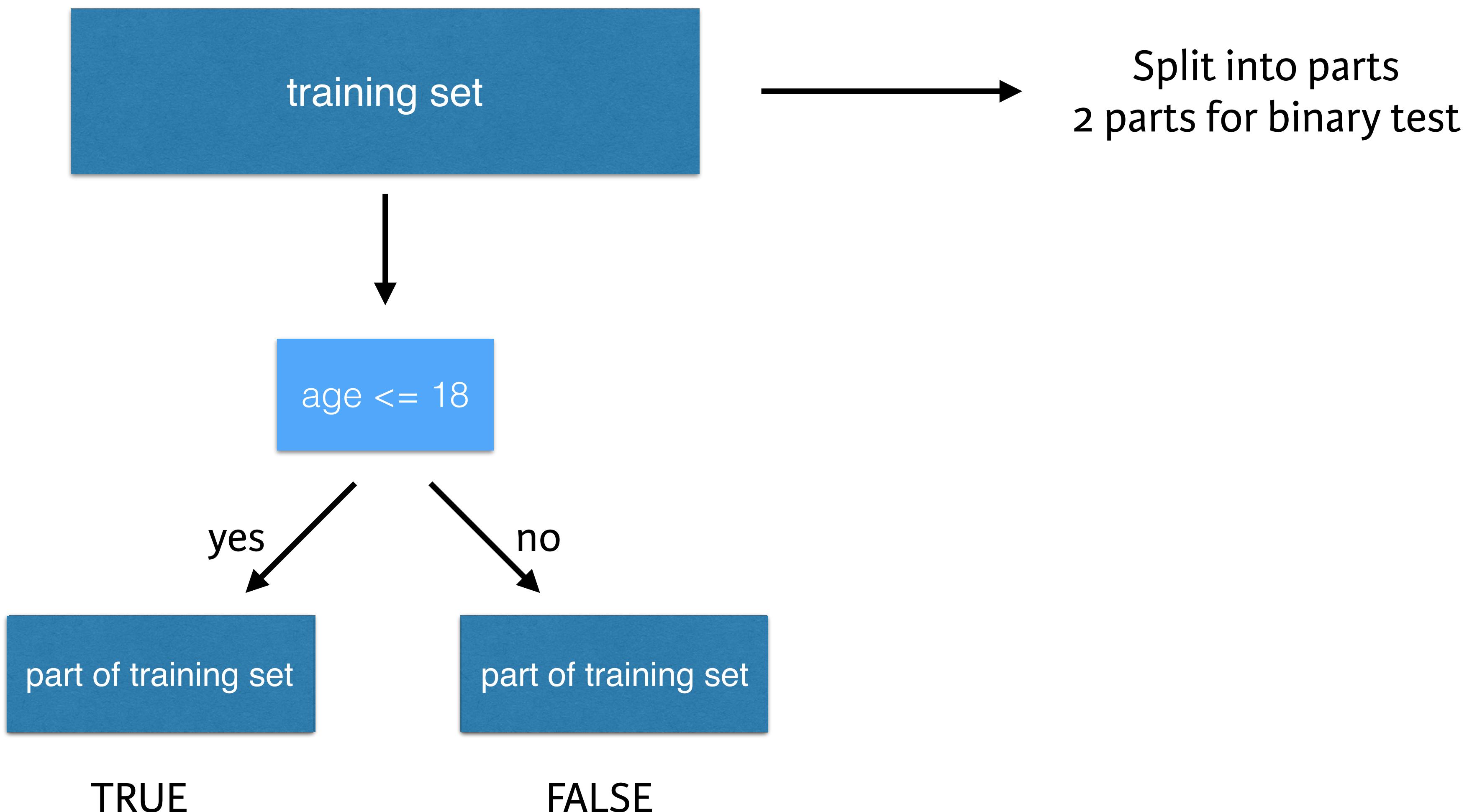
Classifying with the tree

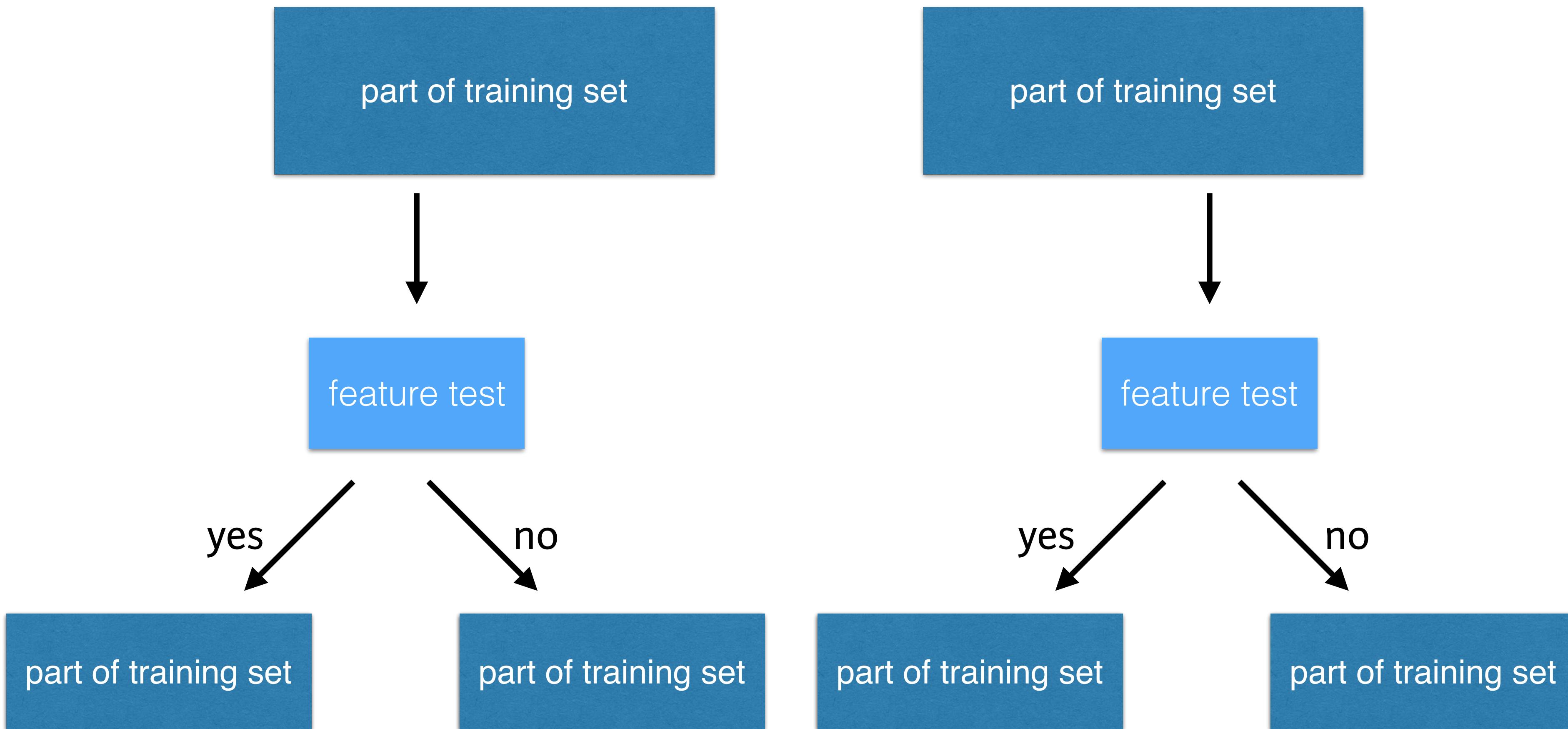
Observation: patient of 40 years, vaccinated and didn't smoke

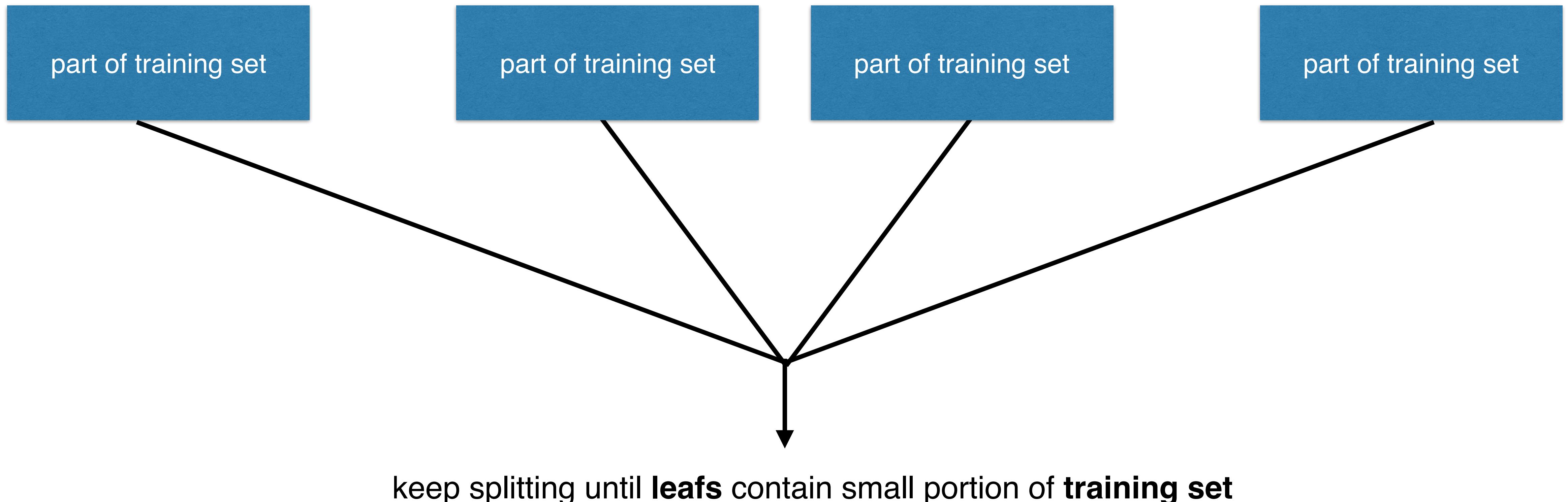


Learn a tree

- Use training set
- Come up with queries (feature tests) at each node

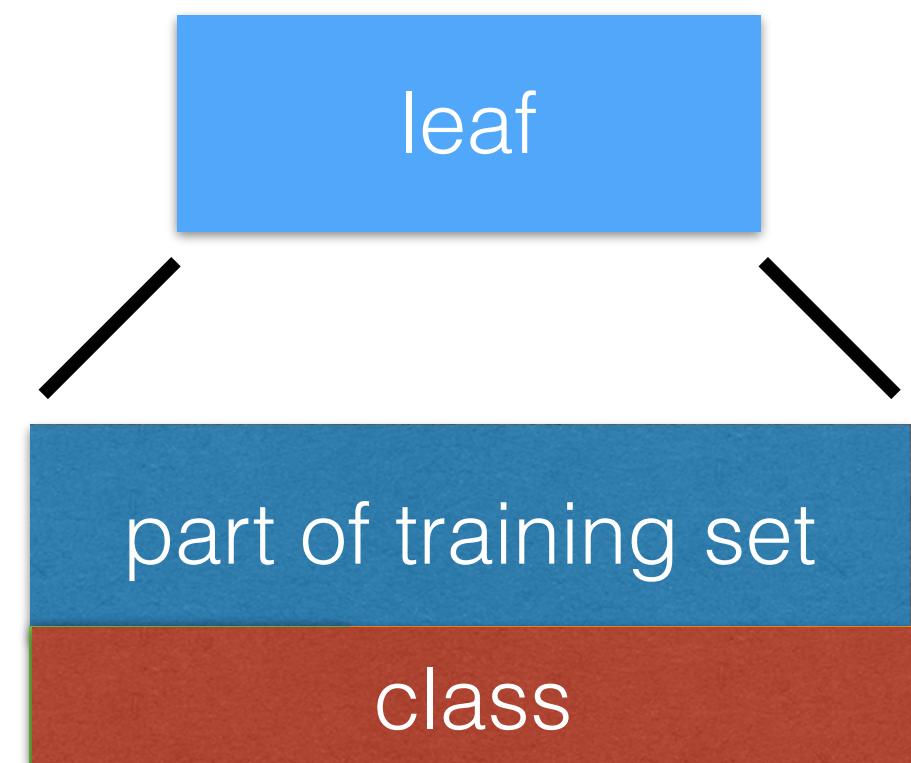






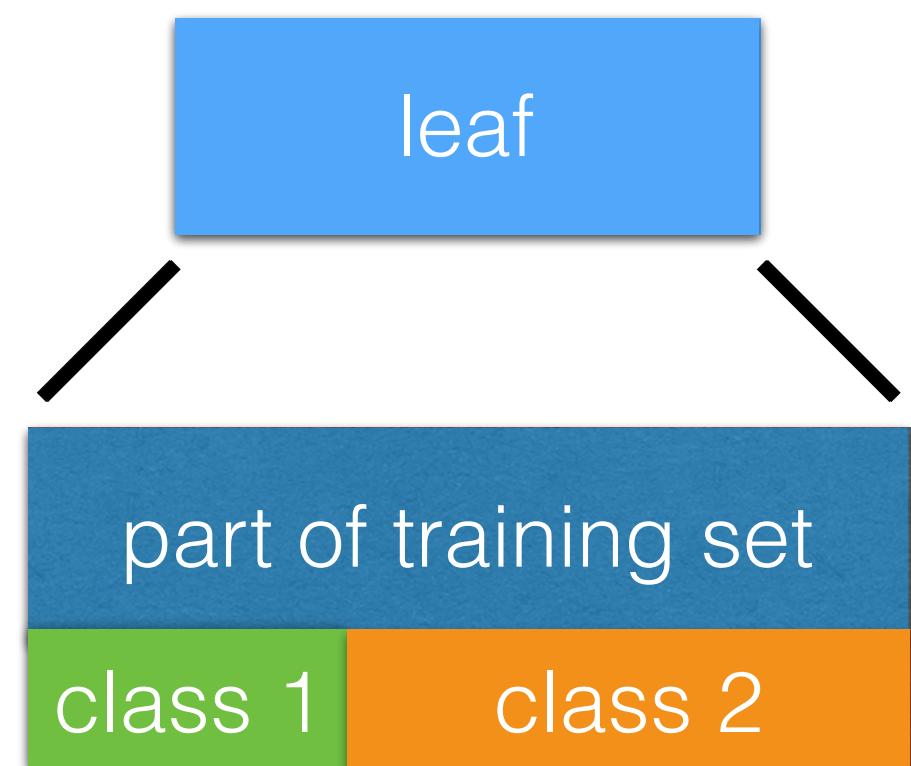
Learn the tree

- Goal: end up with **pure leafs** — leafs that contain observations of one particular class



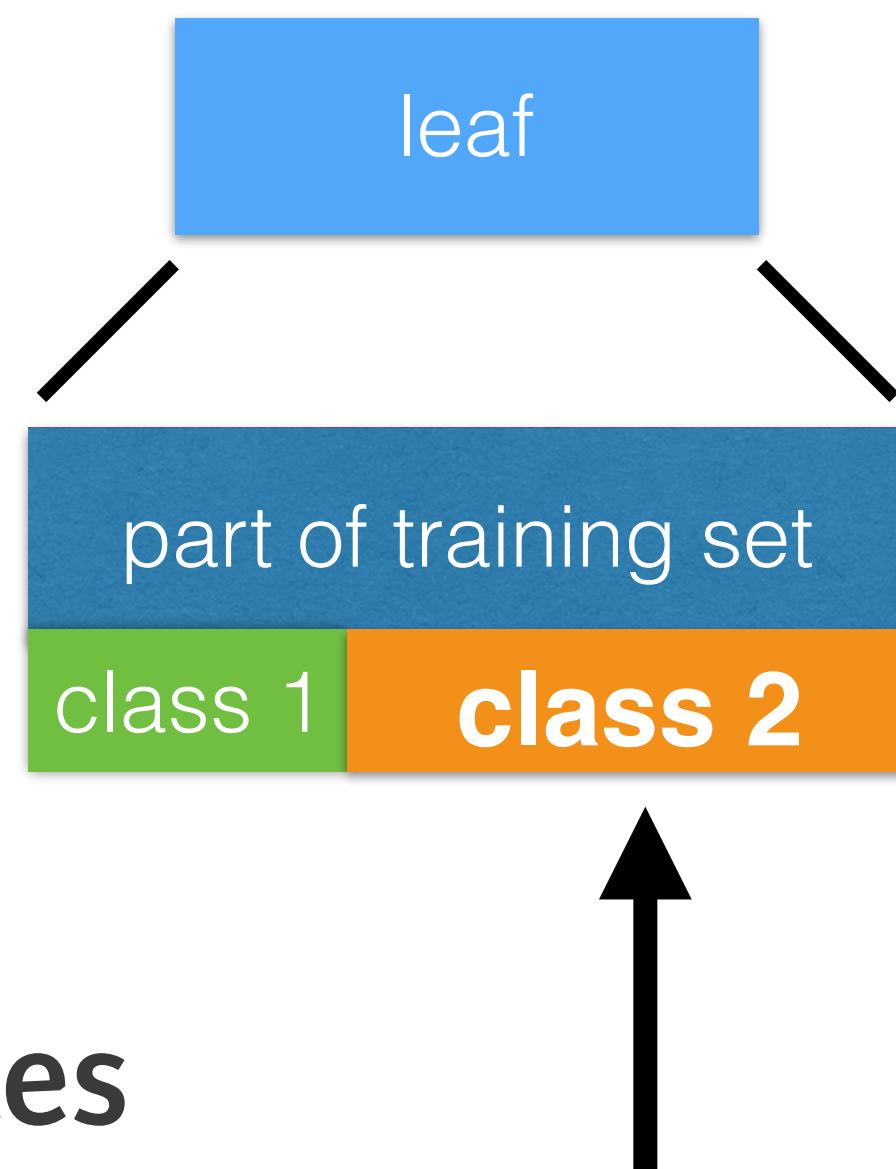
Learn the tree

- Goal: end up with **pure leafs** — leafs that contain observations of one particular class
- In practice: almost never the case — noise
- When classifying new instances
 - end up in leaf



Learn the tree

- Goal: end up with **pure leafs** — leafs that contain observations of one particular class
- In practice: almost never the case — noise
- When classifying new instances
 - end up in leaf
 - assign class of majority of training instances



Learn the tree

- At each node
 - Iterate over different feature tests
 - Choose the best one
- Comes down to two parts
 - **Make list of feature tests**
 - **Choose test with best split**

Construct list of tests

- Categorical features
 - Parents/grandparents/... didn't use the test yet
- Numerical features
 - Choose feature
 - Choose threshold

Choose best feature test

- More complex
- Use splitting criteria to decide which test to use
- Information gain ~ entropy

Information gain

- Information gained from split based on **feature test**
- **Test leads to nicely divided classes**
 - > **high information gain**
- **Test leads to scrambled classes**
 - > **low information gain**
- **Test with highest information gain will be chosen**

Pruning

- Number of nodes influences chance on overfit
- Restrict size — higher bias
 - Decrease chance on overfit
 - Pruning the tree



INTRODUCTION TO MACHINE LEARNING

Let's practice!



INTRODUCTION TO MACHINE LEARNING

k-Nearest Neighbors

Instance-based learning

- Save training set in memory
- No real model like decision tree
- Compare unseen instances to training set
- Predict using the comparison of unseen data and the training set

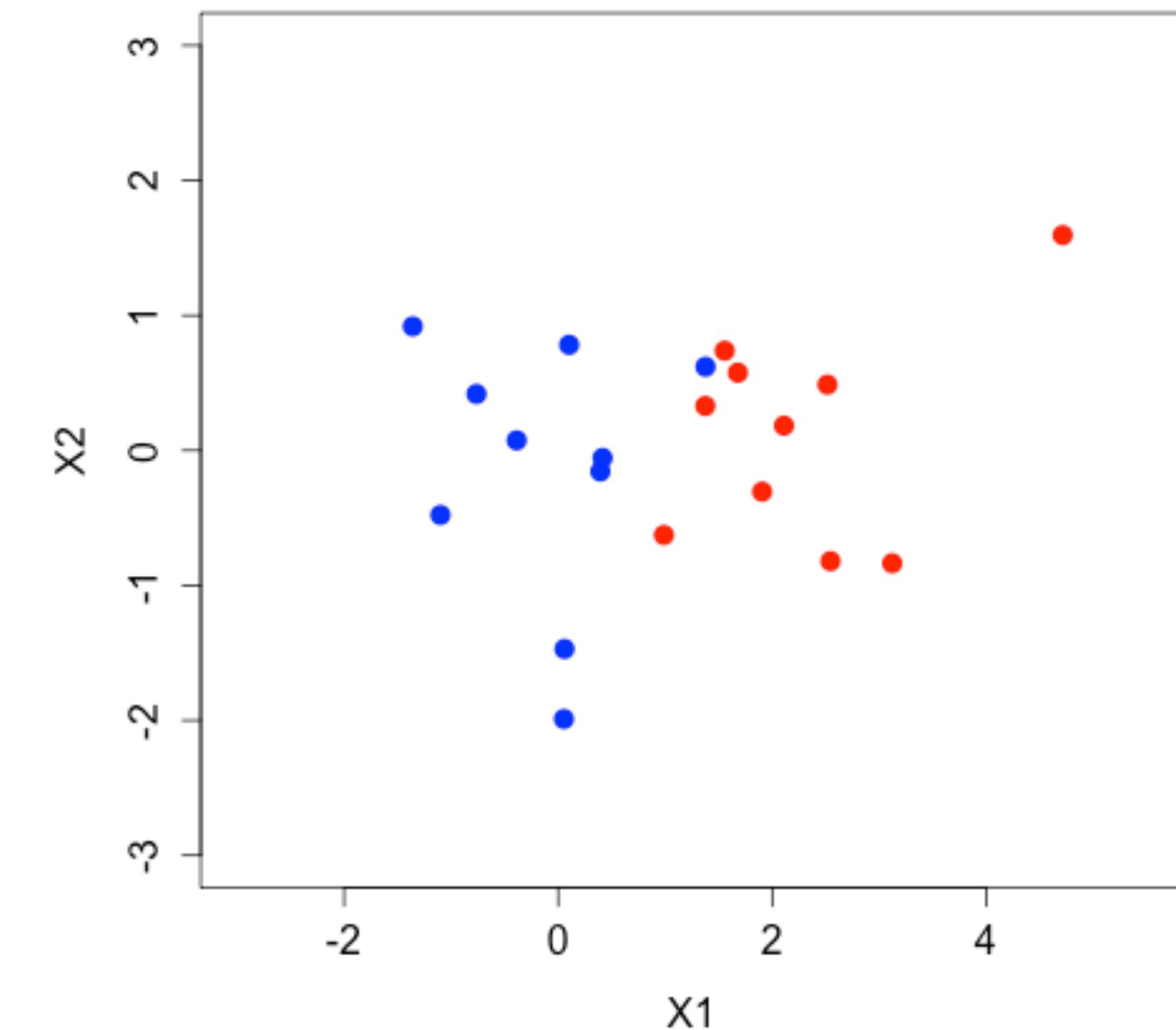
k-Nearest Neighbor

- Form of instance-based learning
- Simplest form: 1-Nearest Neighbor or Nearest Neighbor

Nearest Neighbor - example

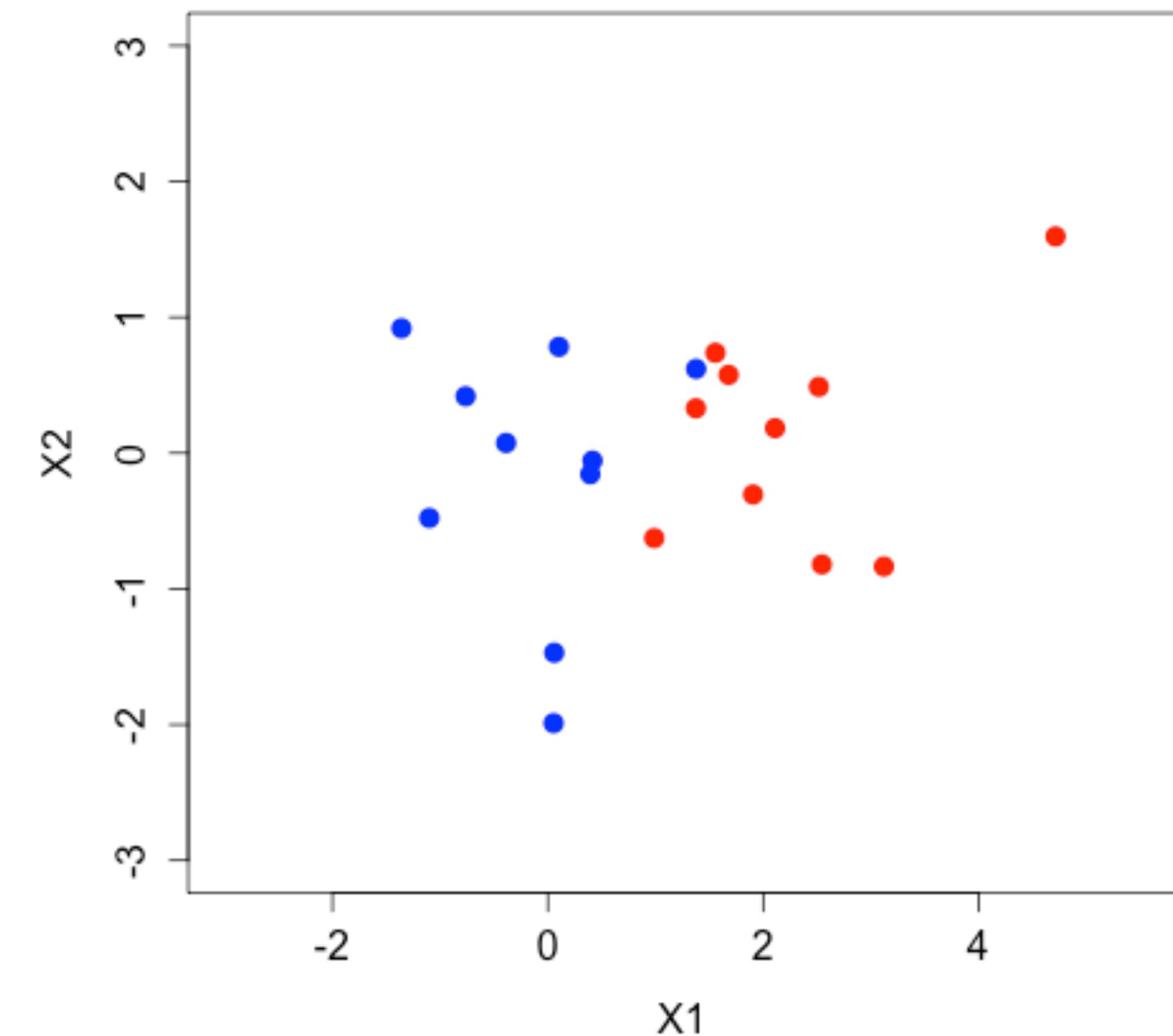
- 2 features: X_1 and X_2
- Class: red or blue
- Binary classification

Nearest Neighbor - example



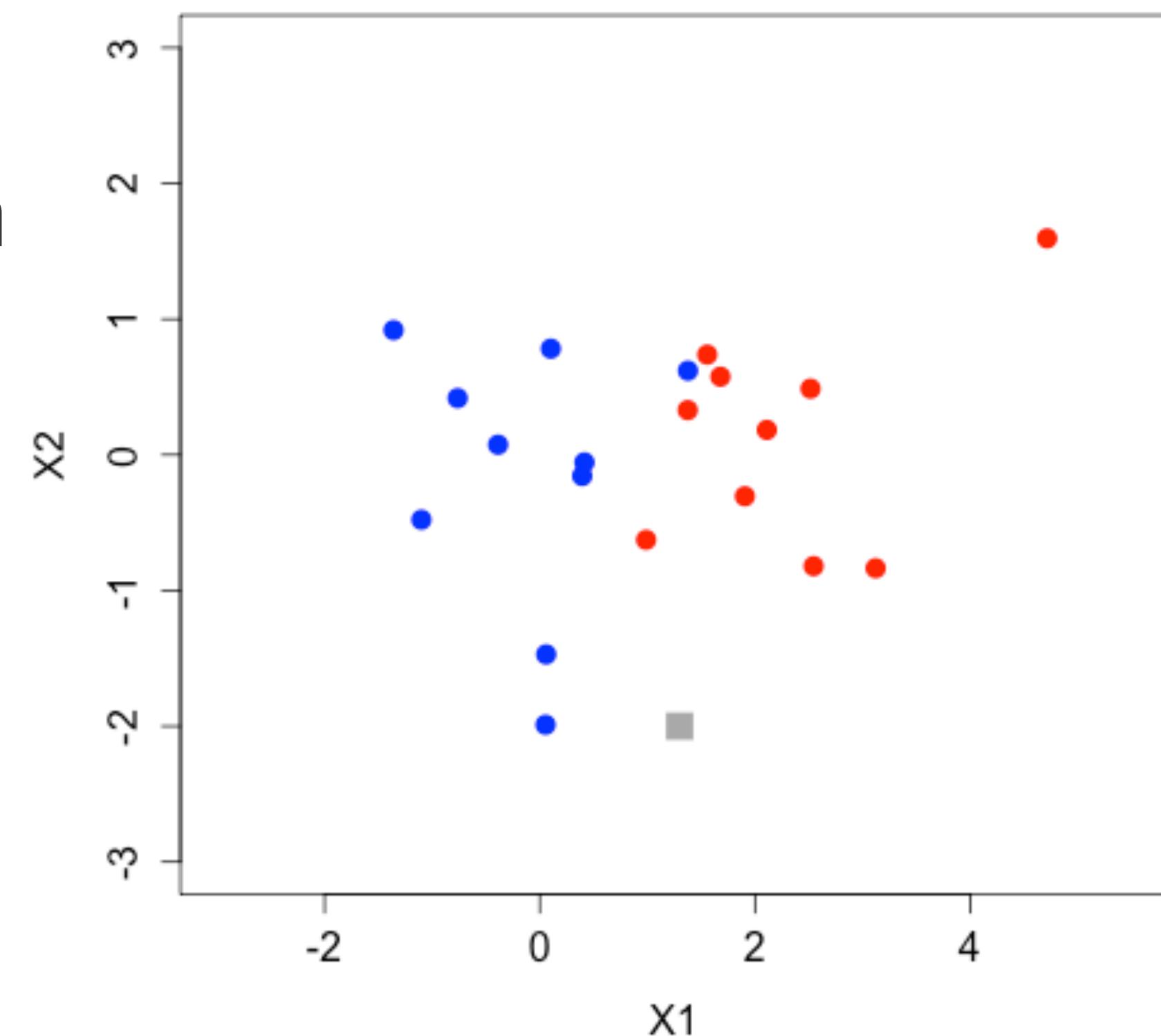
Nearest Neighbor - example

- Save complete training set



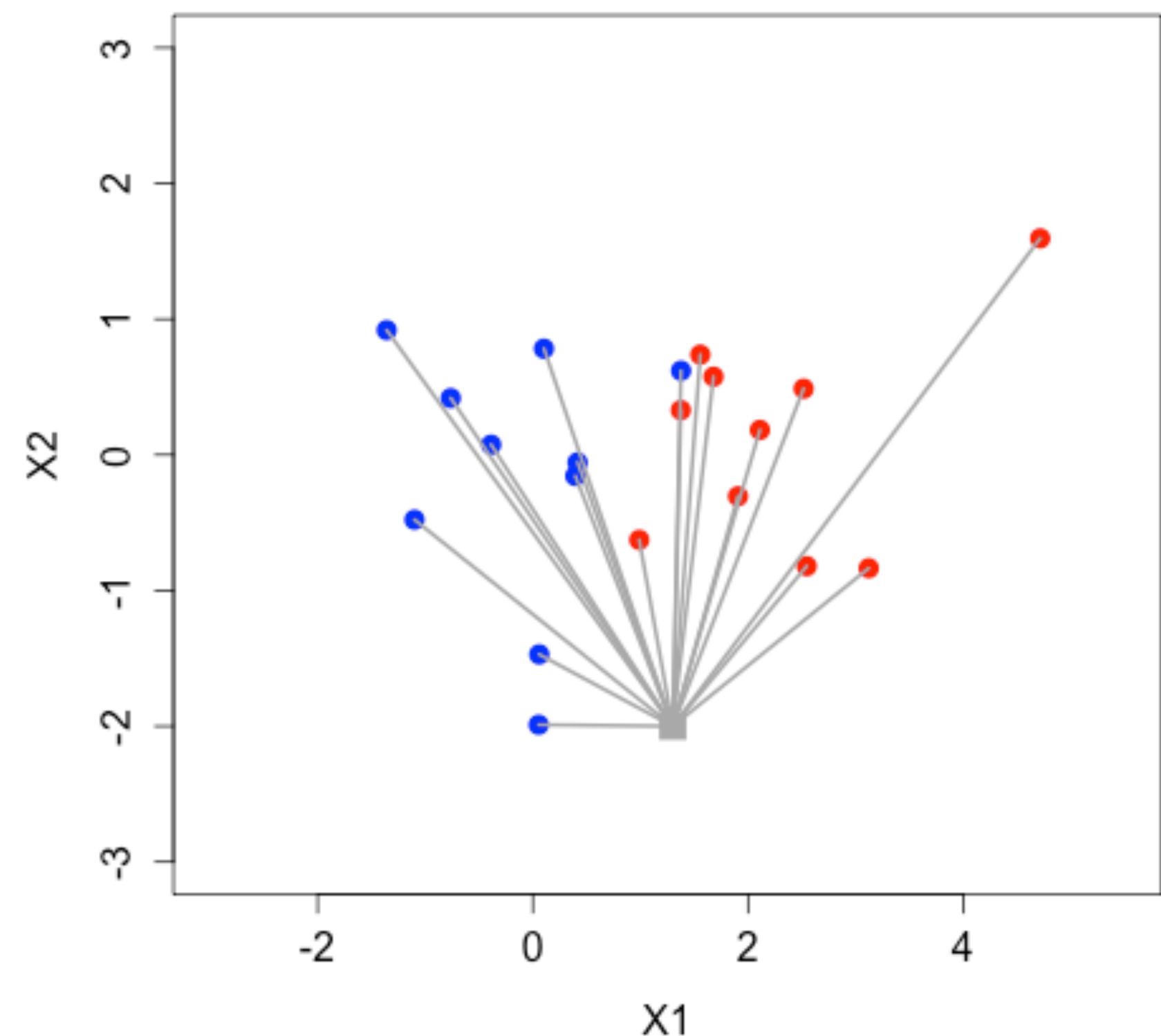
Nearest Neighbor - example

- Save complete training set
- Given: unseen observation with features $X = (1.3, -2)$



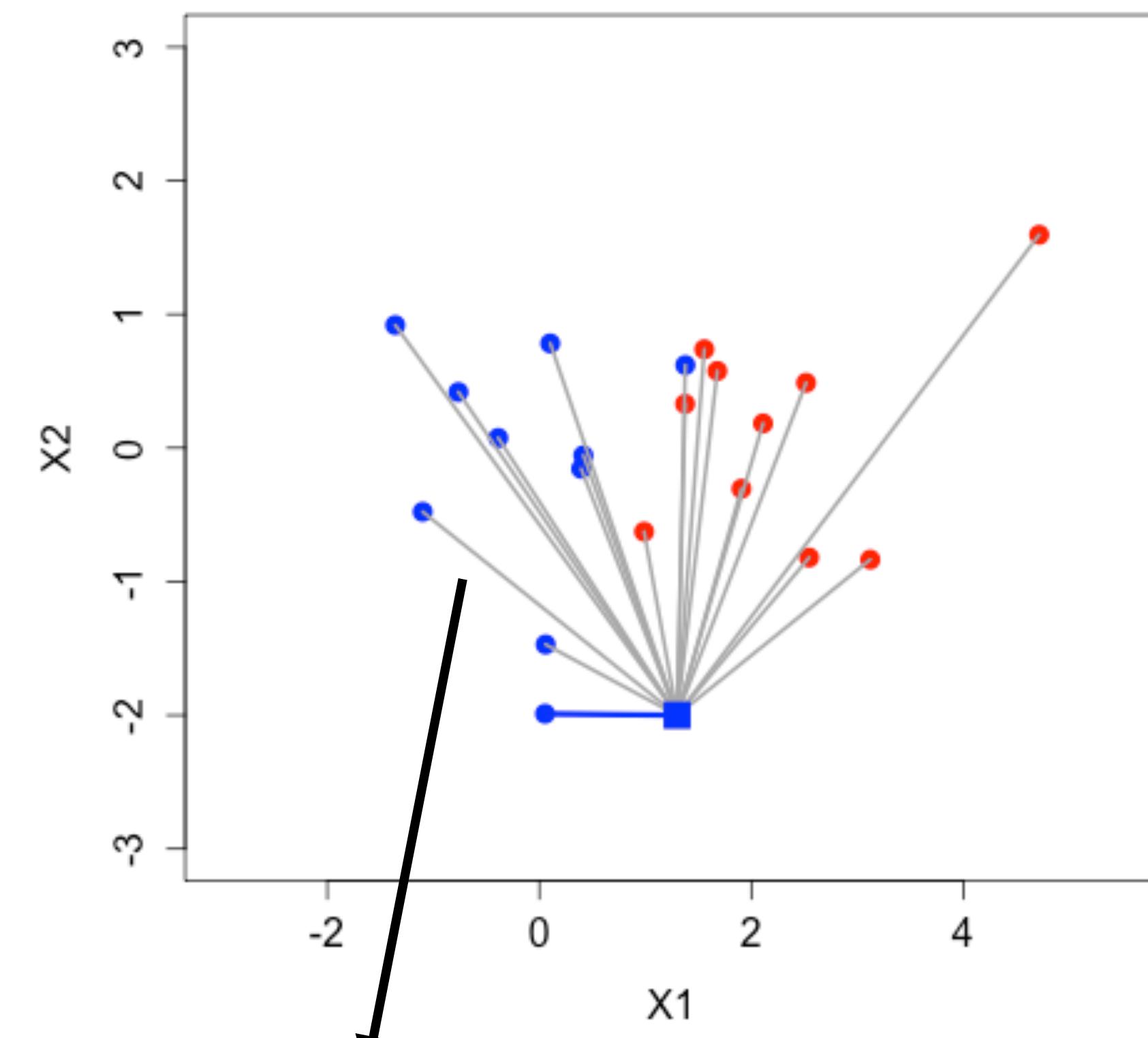
Nearest Neighbor - example

- Save complete training set
- Given: unseen observation with features $X = (1.3, -2)$
- Compare training set with new observation



Nearest Neighbor - example

- Save complete training set
- Given: unseen observation with features $X = (1.3, -2)$
- Compare training set with new observation
- Find **closest** observation — nearest neighbor — and assign same class



just Euclidean distance, nothing fancy

k-Nearest Neighbors

- k is the amount of neighbors
- If $k = 5$
 - Use 5 most similar observations (neighbors)
 - Assigned class will be the most represented class within the 5 neighbors

Distance metric

- Important aspect of k-NN

Distance metric

- Important aspect of k-NN

- Euclidian distance:

$$d_E(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^D (a_i - b_i)^2}$$

Distance metric

- Important aspect of k-NN

- Euclidian distance:

$$d_E(\mathbf{a}, \mathbf{b}) = \sqrt{\sum_{i=1}^D (a_i - b_i)^2}$$

- Manhattan distance:

$$d_M(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^D |a_i - b_i|$$



Scaling - example

- Dataset with
 - 2 features: weight and height
 - 3 observations

	height (m)	weight (kg)
1	1.83	80
2	1.83	80.5
3	1.70	80

distance: 0.5

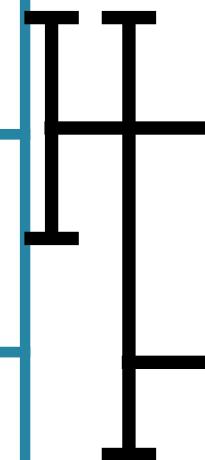
distance: 0.13

Scaling - example

- Dataset with
 - 2 features: weight and height
 - 3 observations

Scale influences distance!

	height (cm)	weight (kg)
1	183	80
2	183	80.5
3	170	80



distance: 0.5

distance: 13

Scaling

- Normalize all features
- e.g. rescale values between 0 and 1
- Gives better measure of real distance
- Don't forget to scale new observations

Categorical features

- How to use in distance metric?
- Dummy variables
- 1 categorical features with N possible outcomes to N binary features (2 outcomes)

Dummy variables — Example

mother tongue: Spanish, Italian or French

mother_tongue	spanish	italian	french
Spanish	1	0	0
Italian	0	1	0
Italian	0	1	0
Spanish	1	0	0
French	0	0	1
French	0	0	1
French	0	0	1





INTRODUCTION TO MACHINE LEARNING

Let's practice!



INTRODUCTION TO MACHINE LEARNING

Introducing: The ROC curve

Introducing

- Very powerful performance measure
- For binary classification
- Receiver Operator Characteristic Curve (ROC Curve)

Probabilities as output

- Used decision trees and k-NN to predict class
- They can also output probability that instance belongs to class

Probabilities as output - example

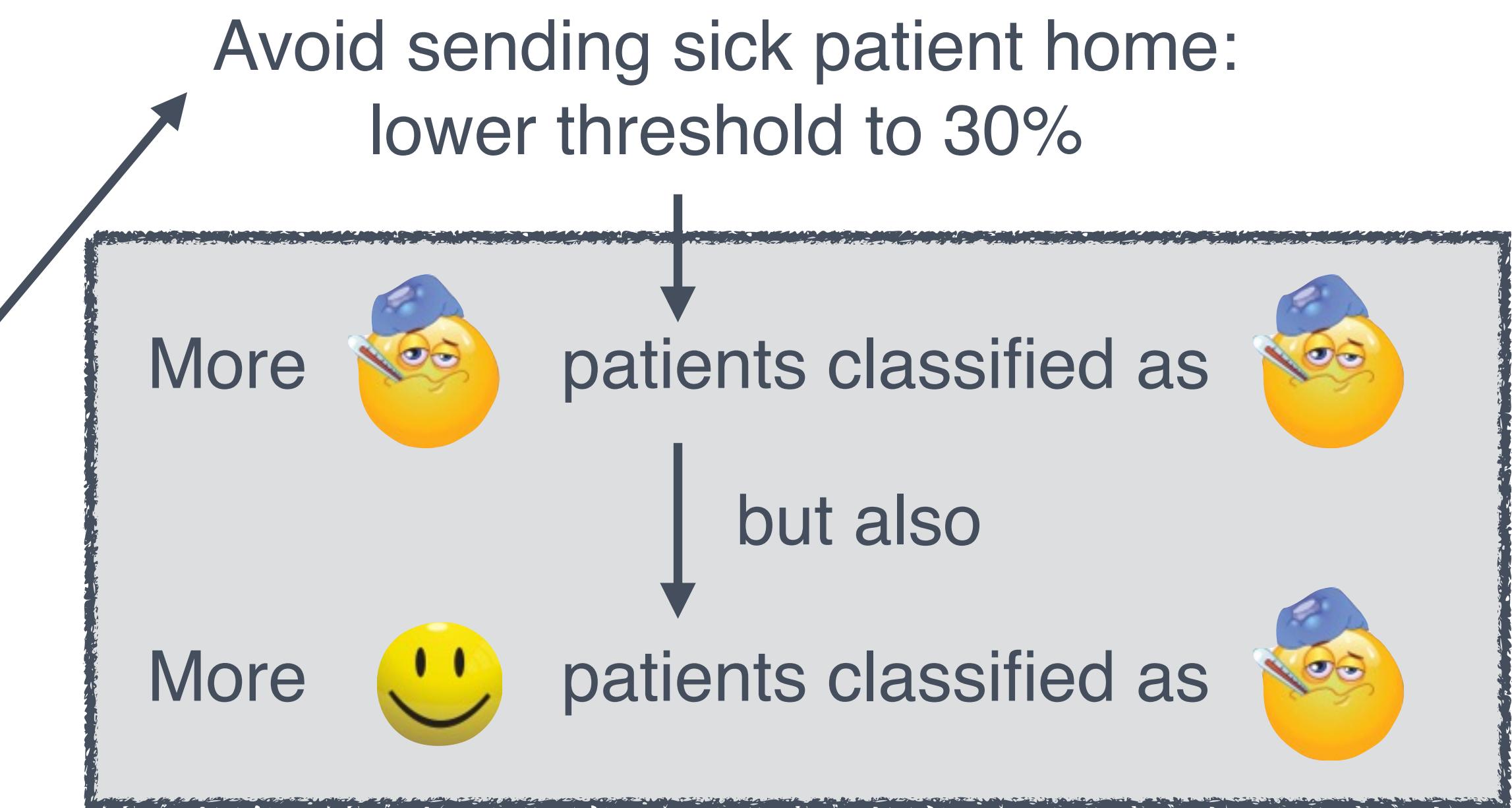
- Binary classification



- Decide whether patient is sick or not sick → **decision function!**
- Define probability threshold from which you decide patient to be sick

Decision tree:

New patient: 70%

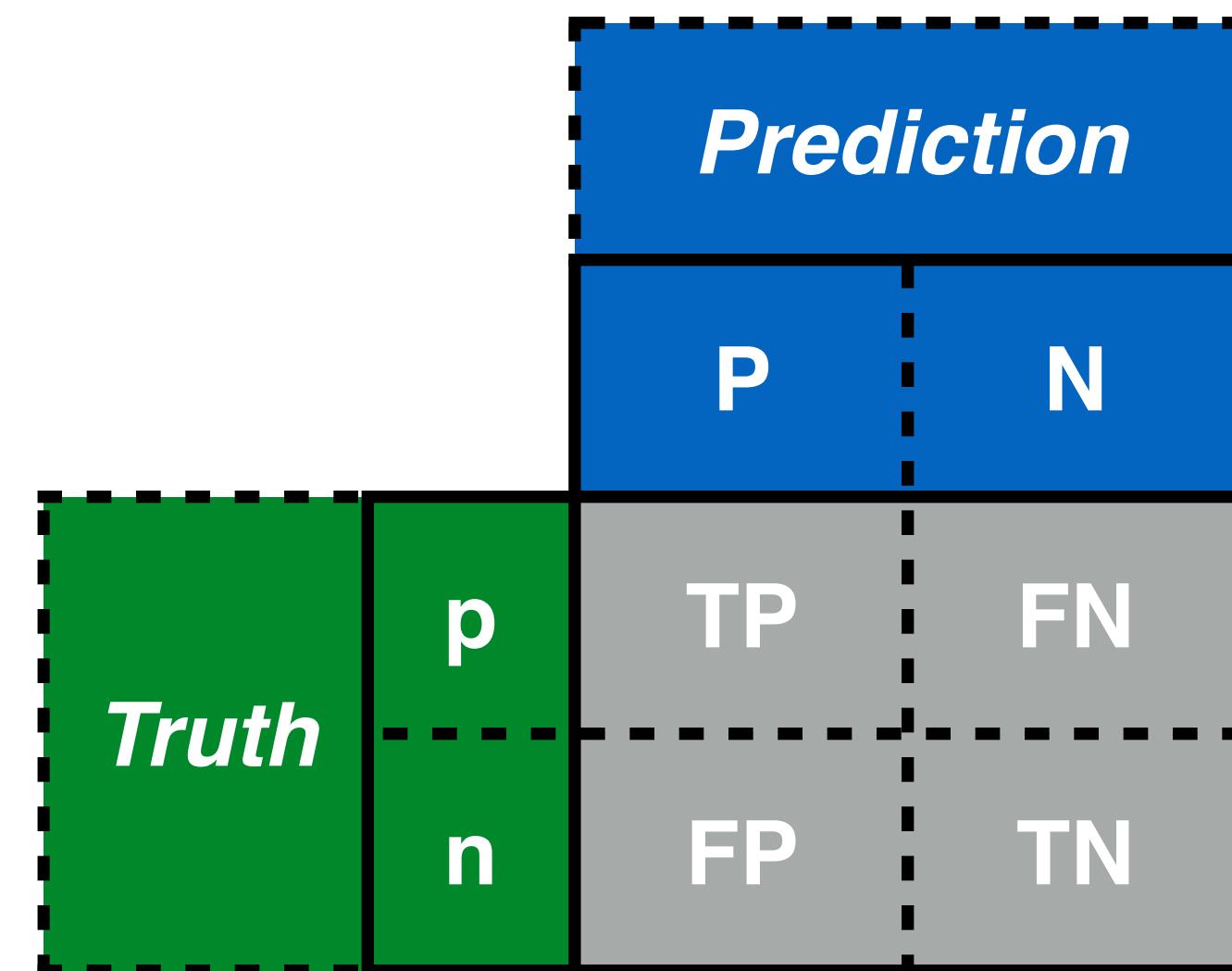


Confusion matrix

- Other performance measure for classification
- Important to construct the ROC curve

Confusion matrix

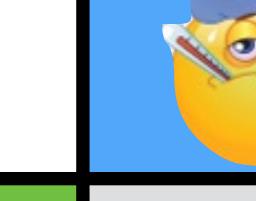
- Binary classifier: positive or negative (1 or 0)



Confusion matrix

- Binary classifier: positive or negative (1 or 0)

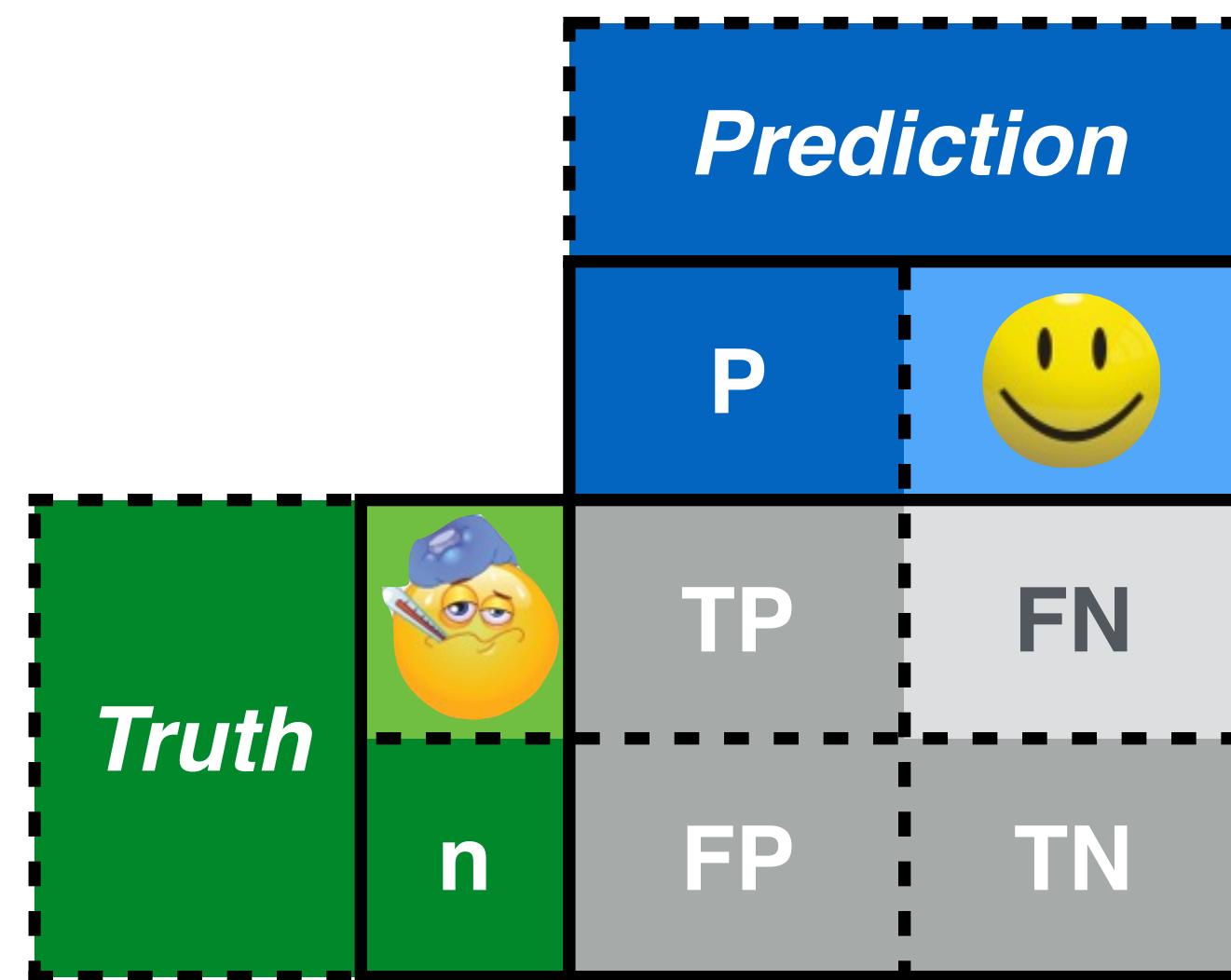
True Positives
Prediction: P
Truth: P

		<i>Prediction</i>	
		 N	N
<i>Truth</i>	P	TP	FN
	n	FP	TN

Confusion matrix

- Binary classifier: positive or negative (1 or 0)

False Negatives
Prediction: N
Truth: P

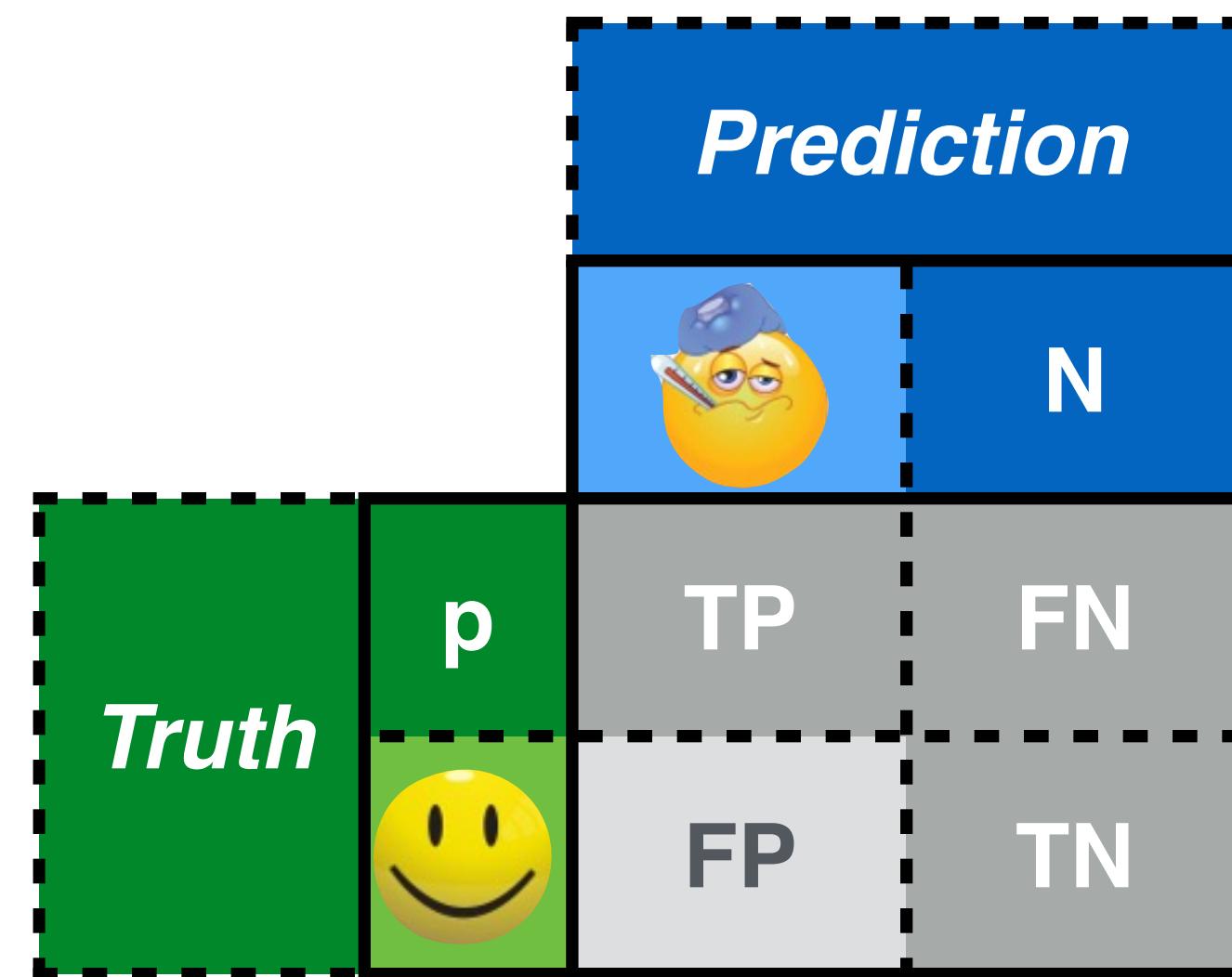


Confusion matrix

- Binary classifier: positive or negative (1 or 0)

False Positives
Prediction: P
Truth: N

		<i>Prediction</i>	
		N	
<i>Truth</i>	P	TP	FN
		FP	TN



Confusion matrix

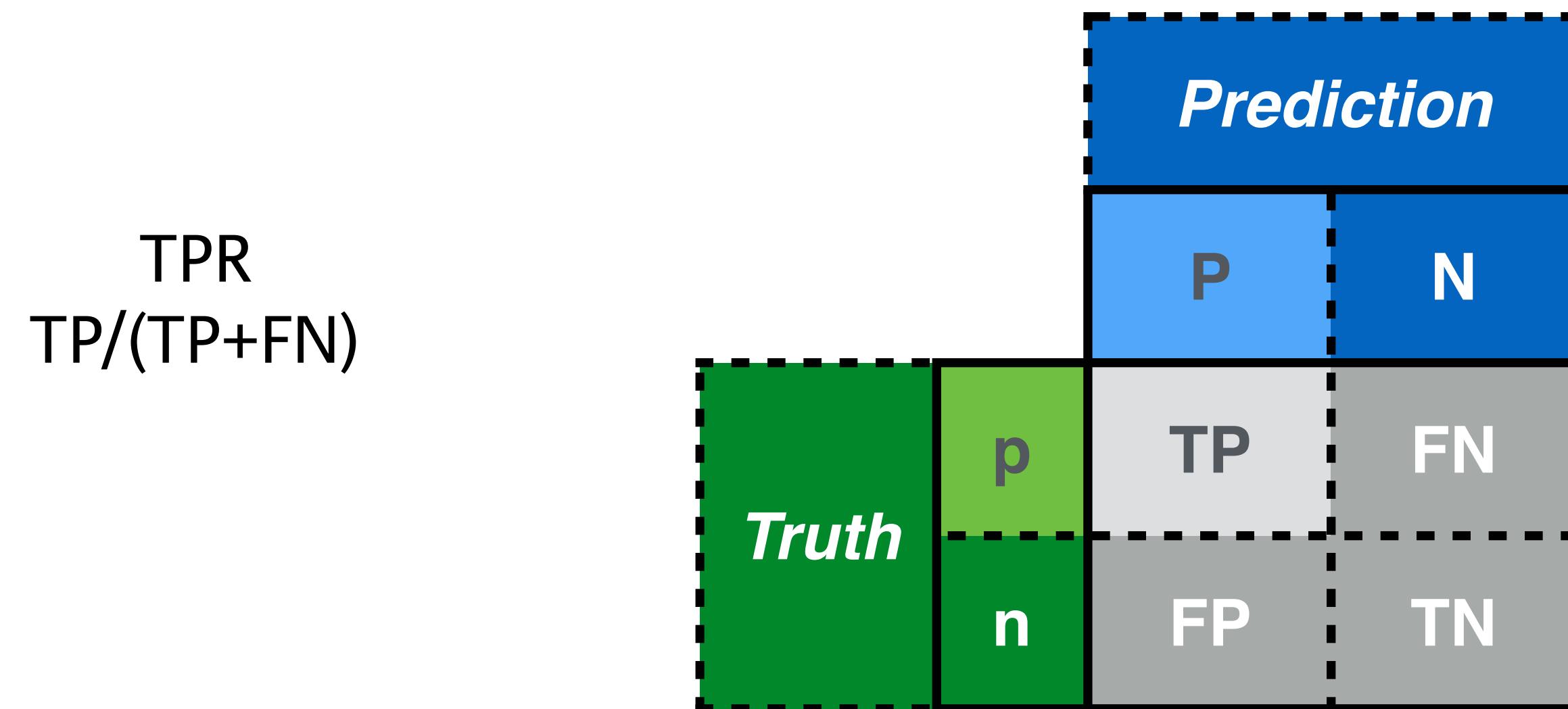
- Binary classifier: positive or negative (1 or 0)

True Negatives
Prediction: N
Truth: N

		<i>Prediction</i>	
		P	Not P
<i>Truth</i>	Not P	p	TP
	P	FP	TN

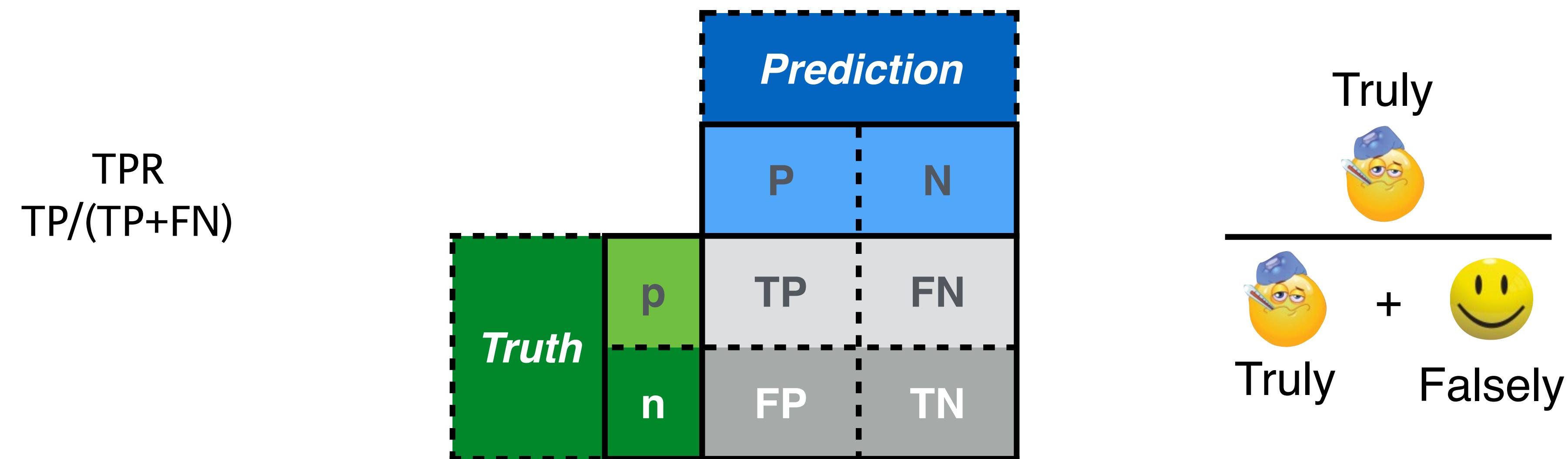
Ratios in the confusion matrix

- True positive rate (TPR) = recall
- False positive rate (FPR)



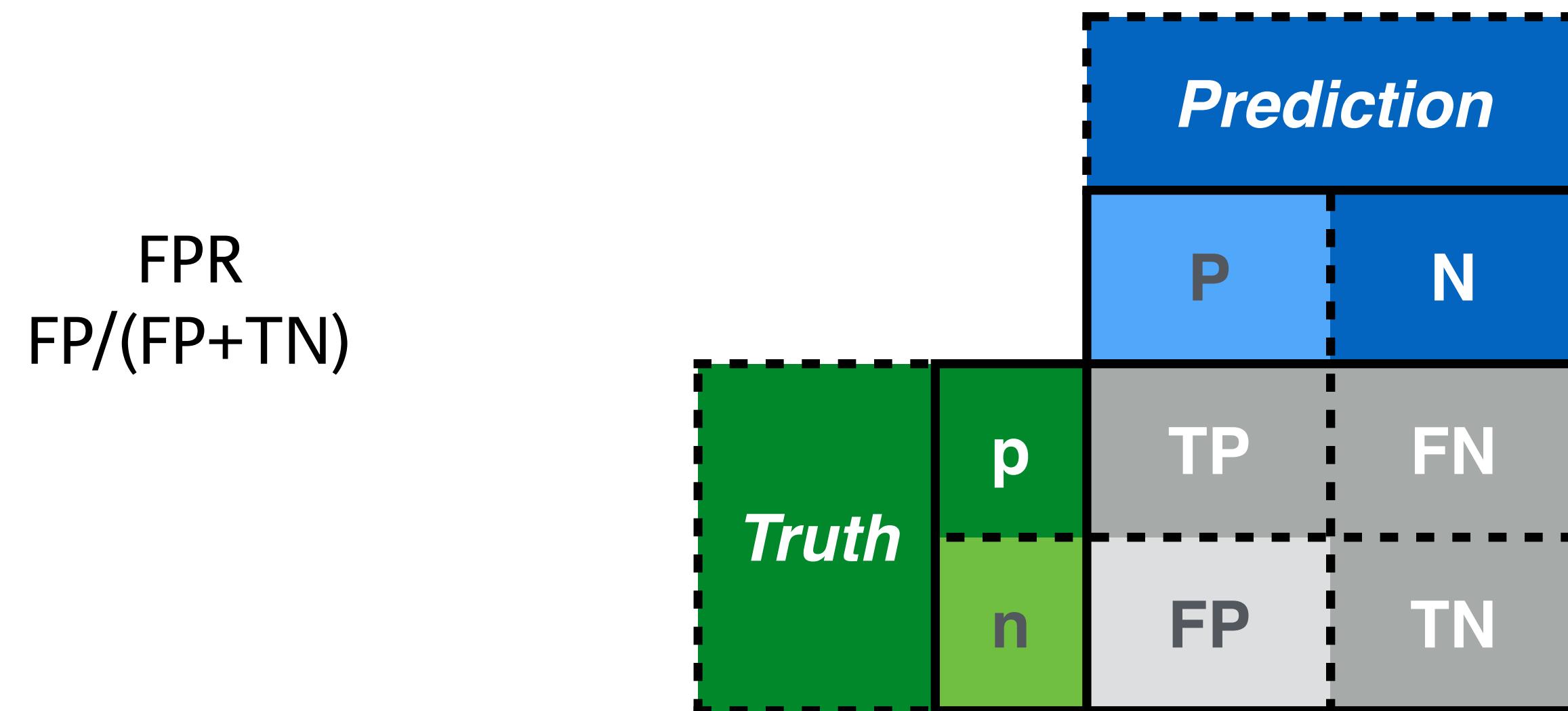
Ratios in the confusion matrix

- True positive rate (TPR) = recall
- False positive rate (FPR)



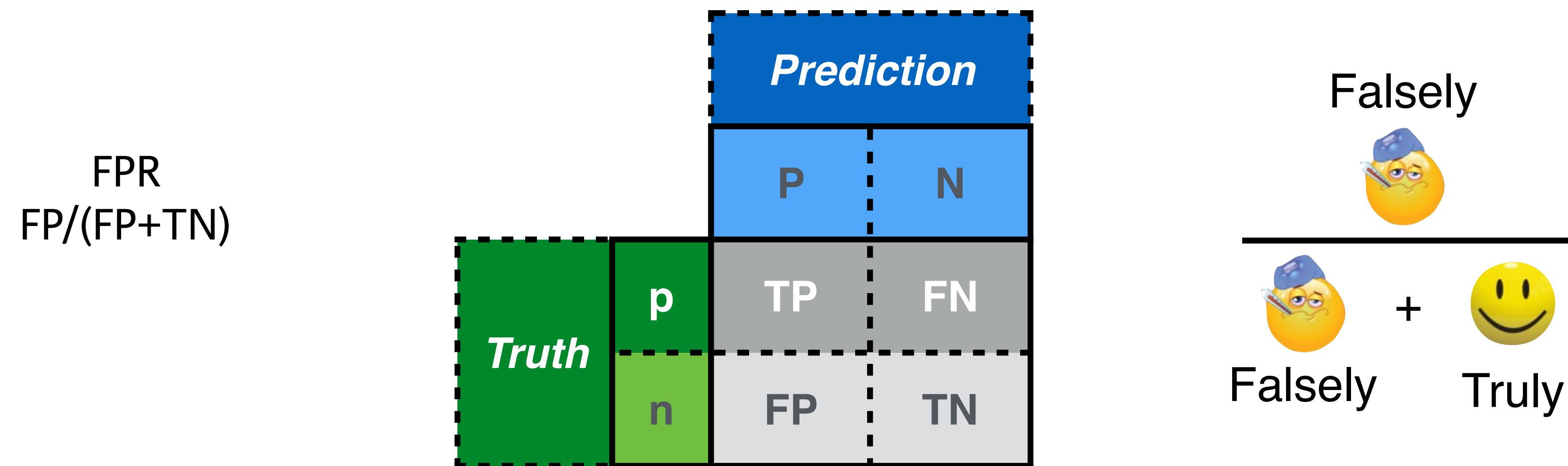
Ratios in the confusion matrix

- True positive rate (TPR) = recall
- False positive rate (FPR)



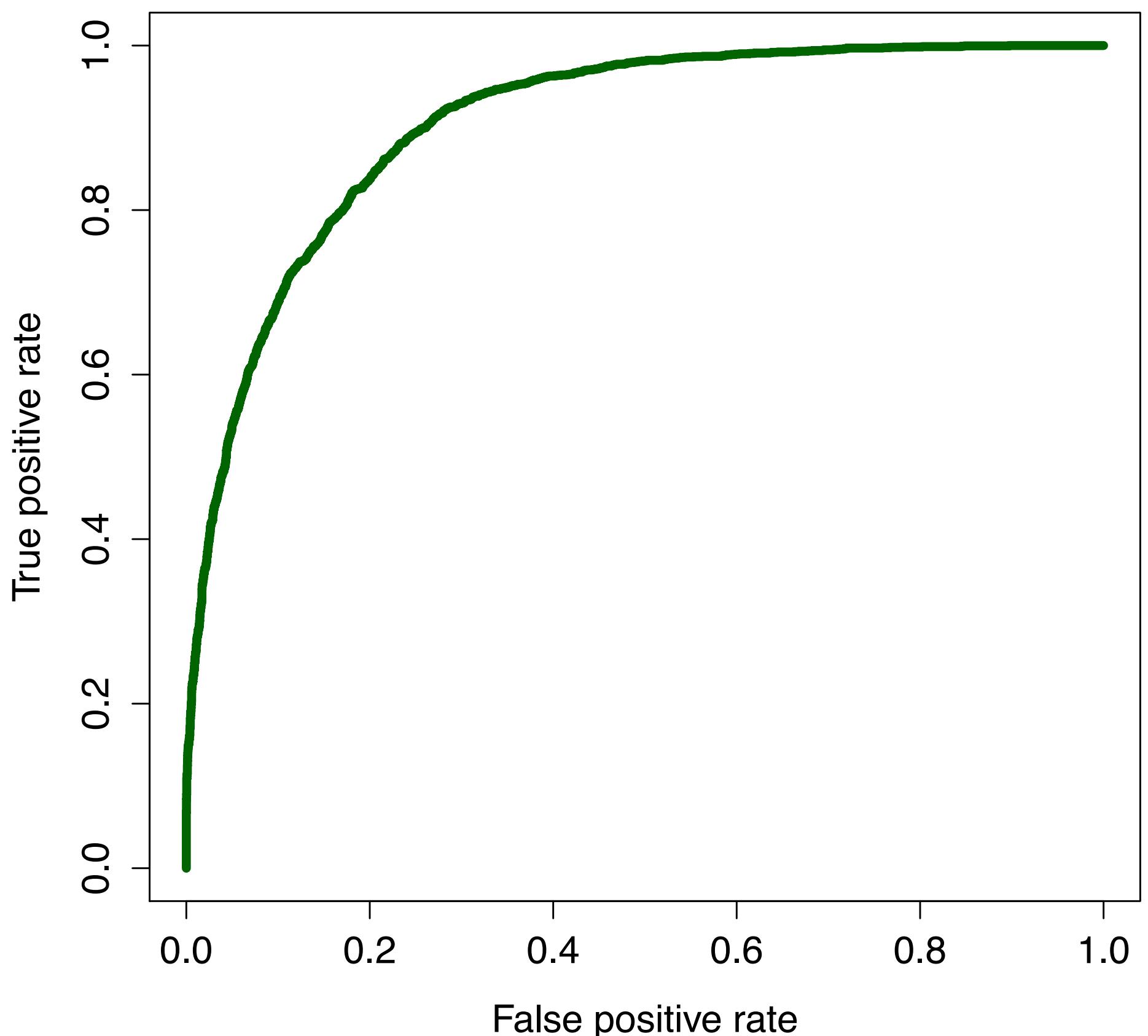
Ratios in the confusion matrix

- True positive rate (TPR) = recall
- False positive rate (FPR)



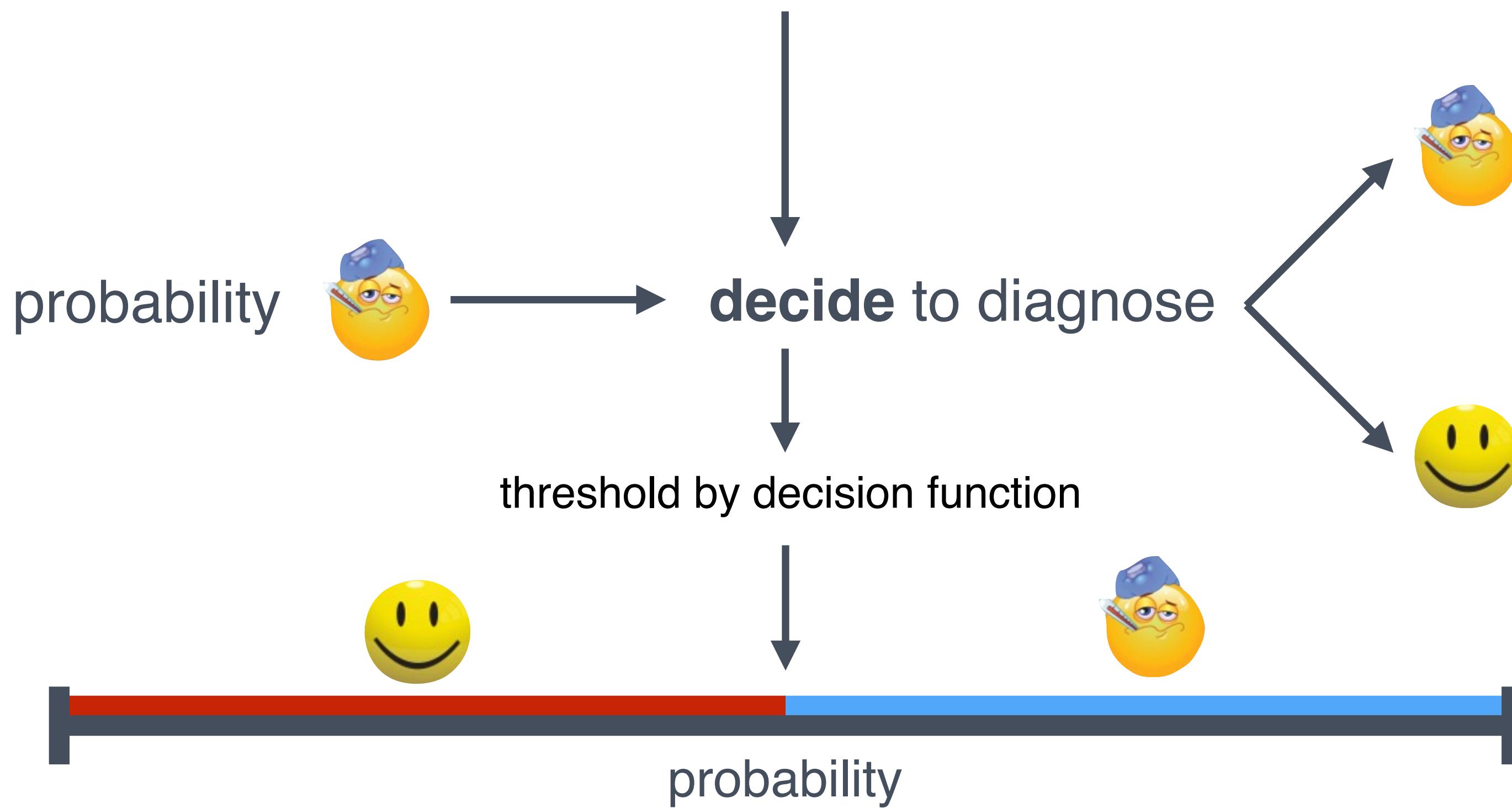
ROC curve

- Horizontal axis: FPR
- Vertical axis: TPR
- How to draw the curve?



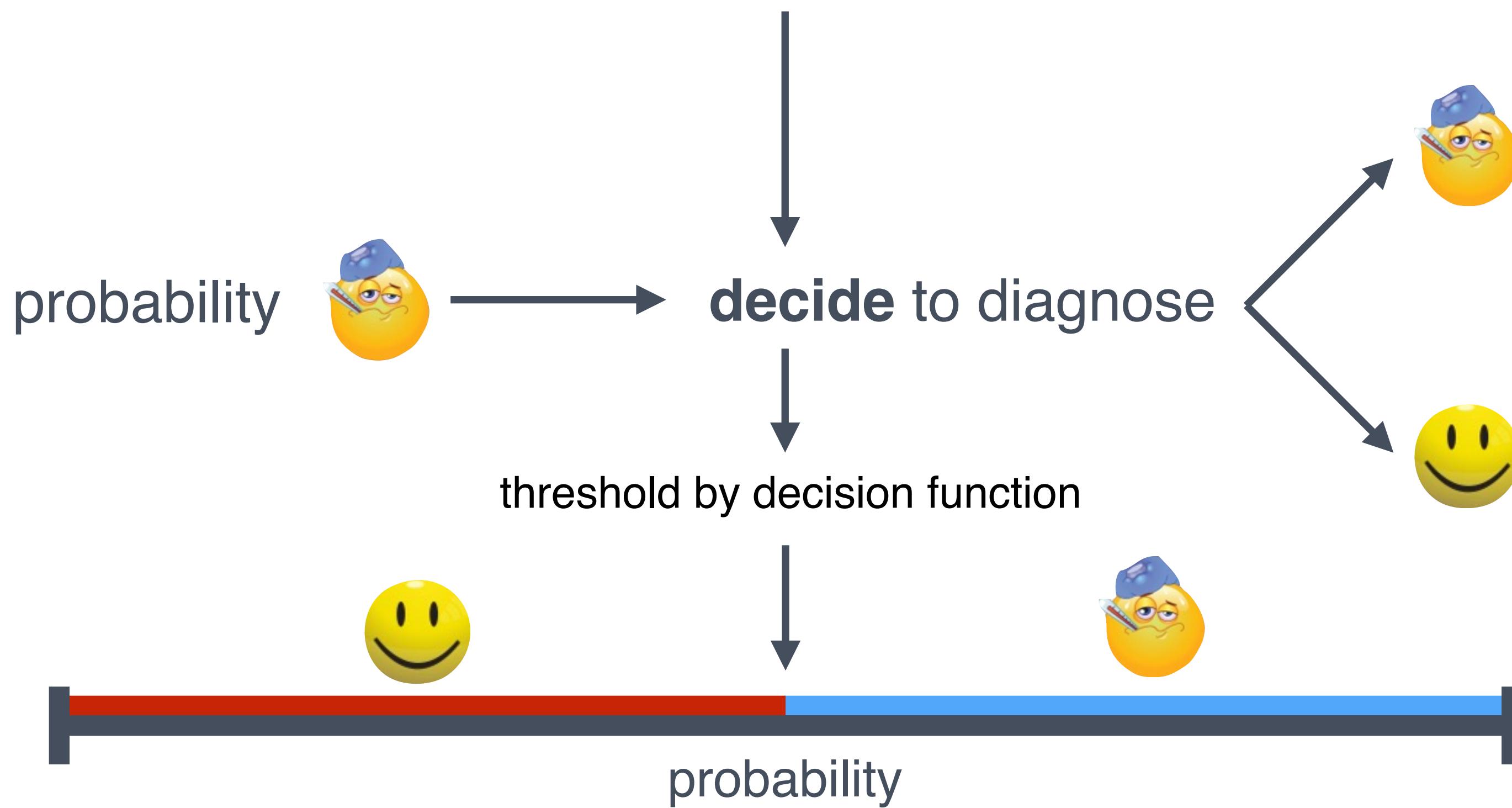
Draw the curve

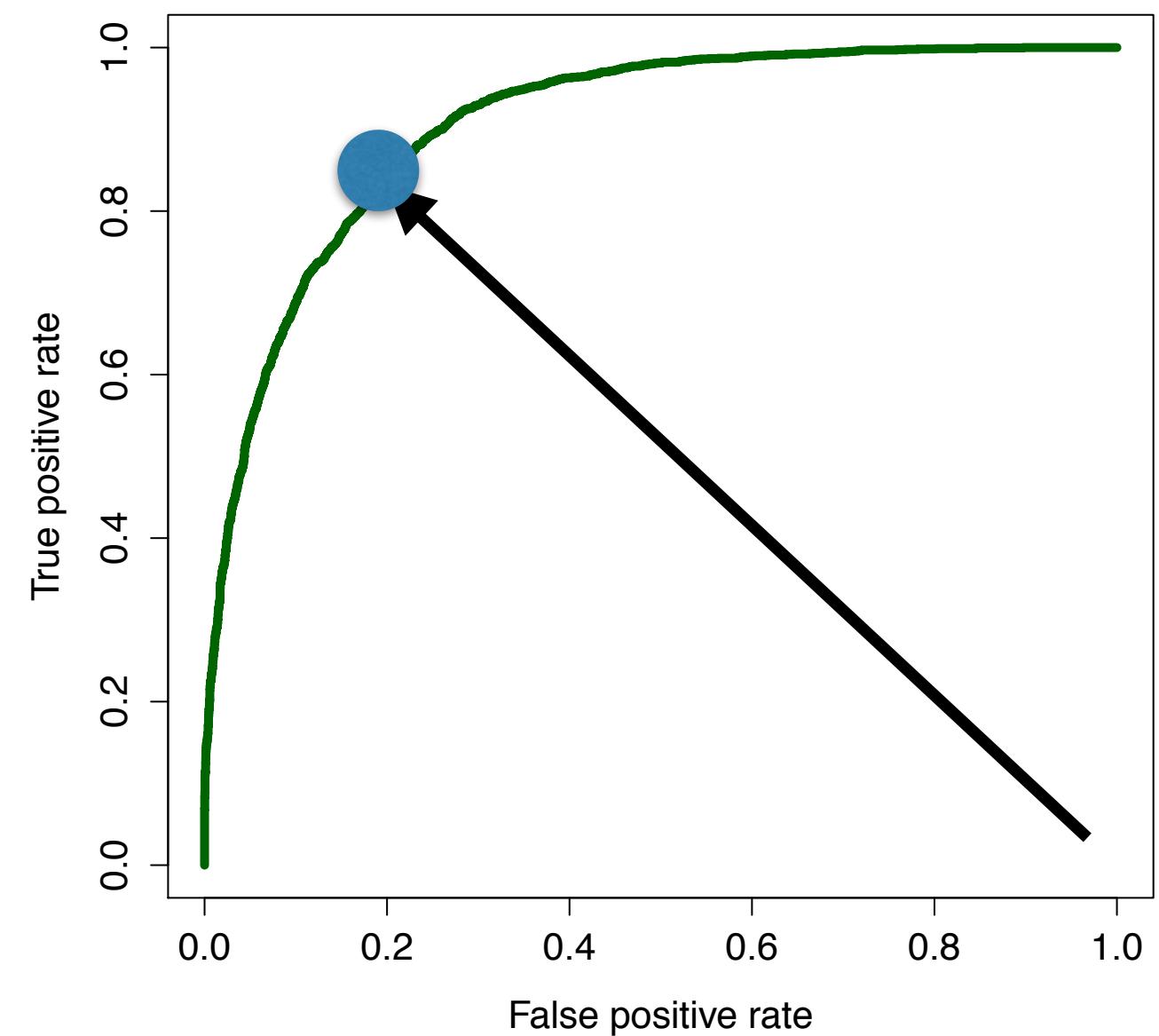
- Need classifier which **outputs probabilities**
- The decision function



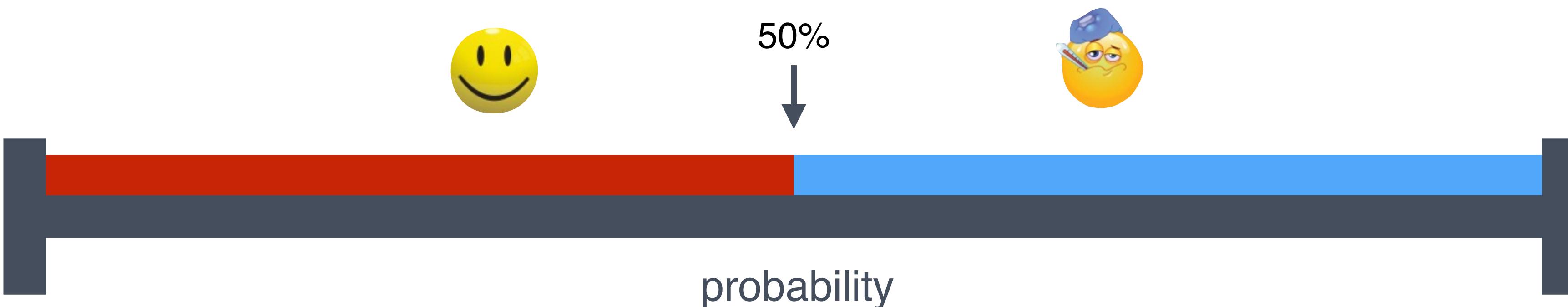
Draw the curve

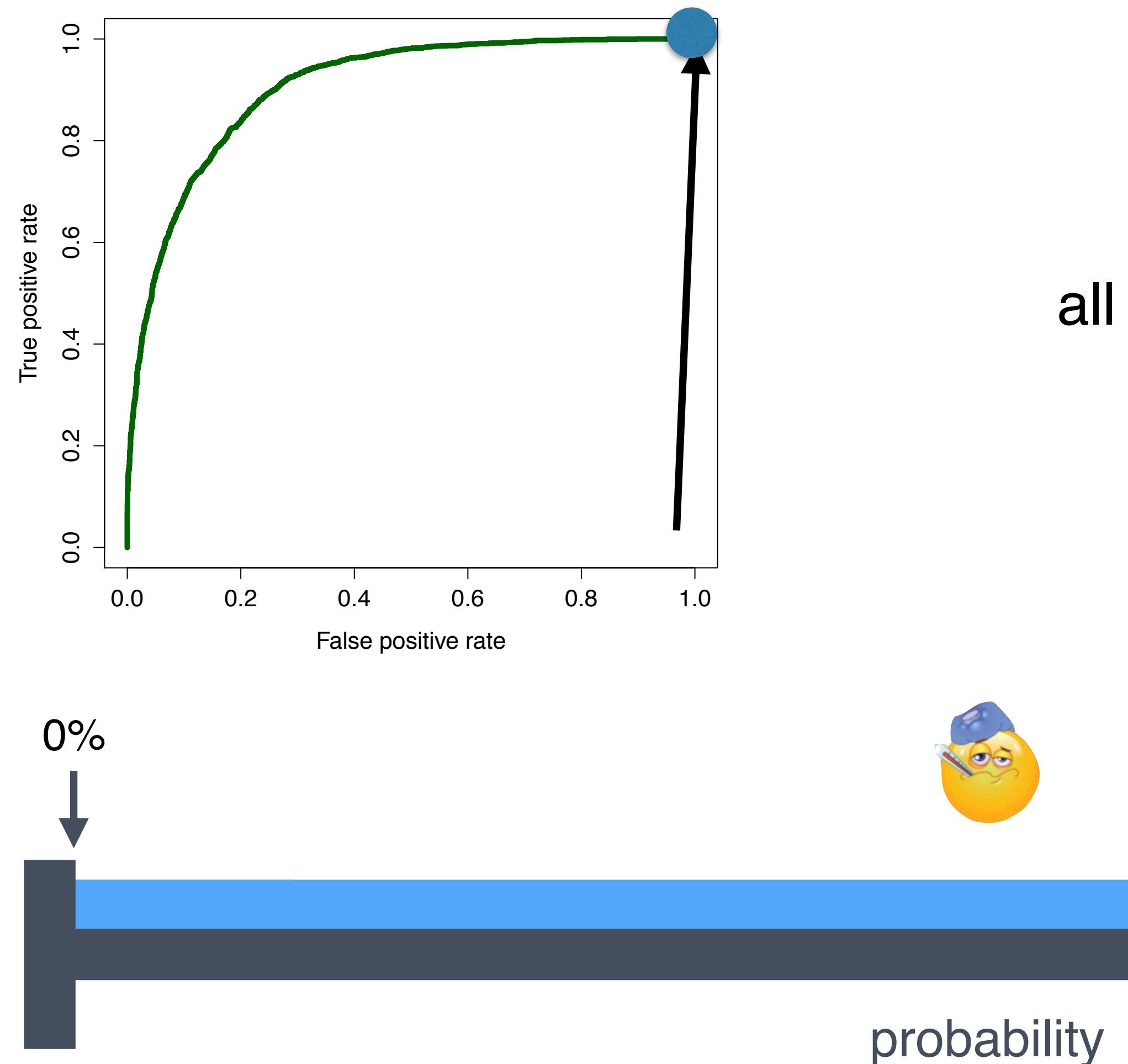
- Need classifier which outputs probabilities
- The decision function

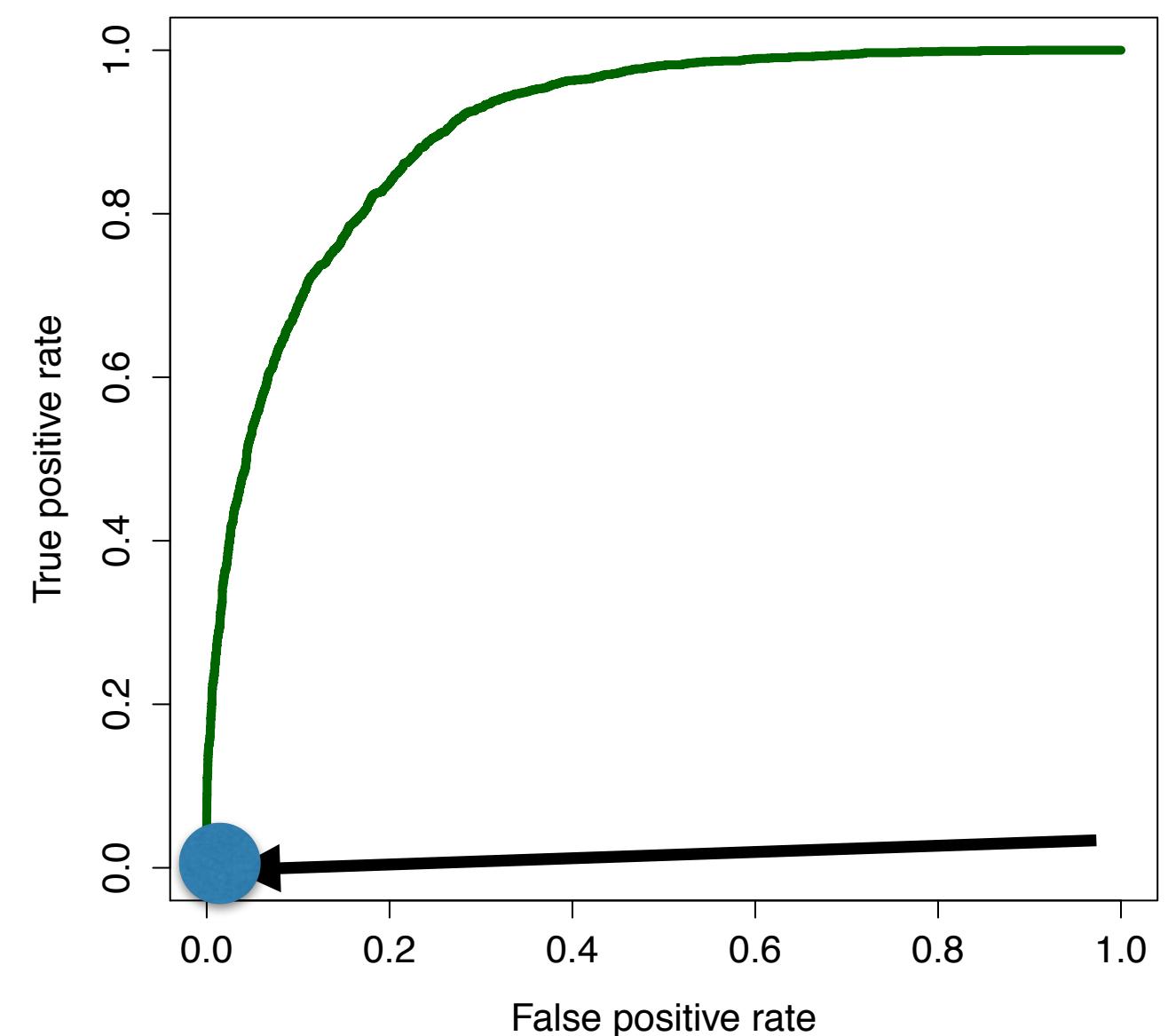




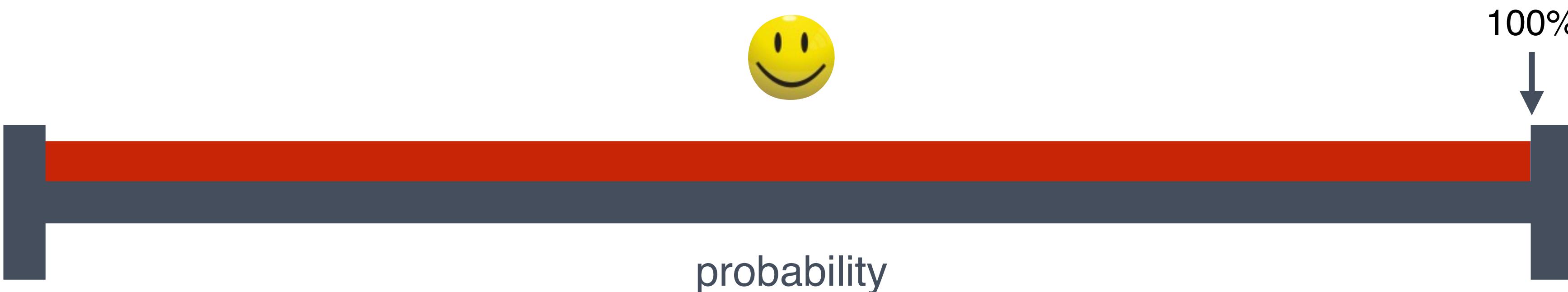
$\geq 50\%$: sick
 $< 50\%$: healthy





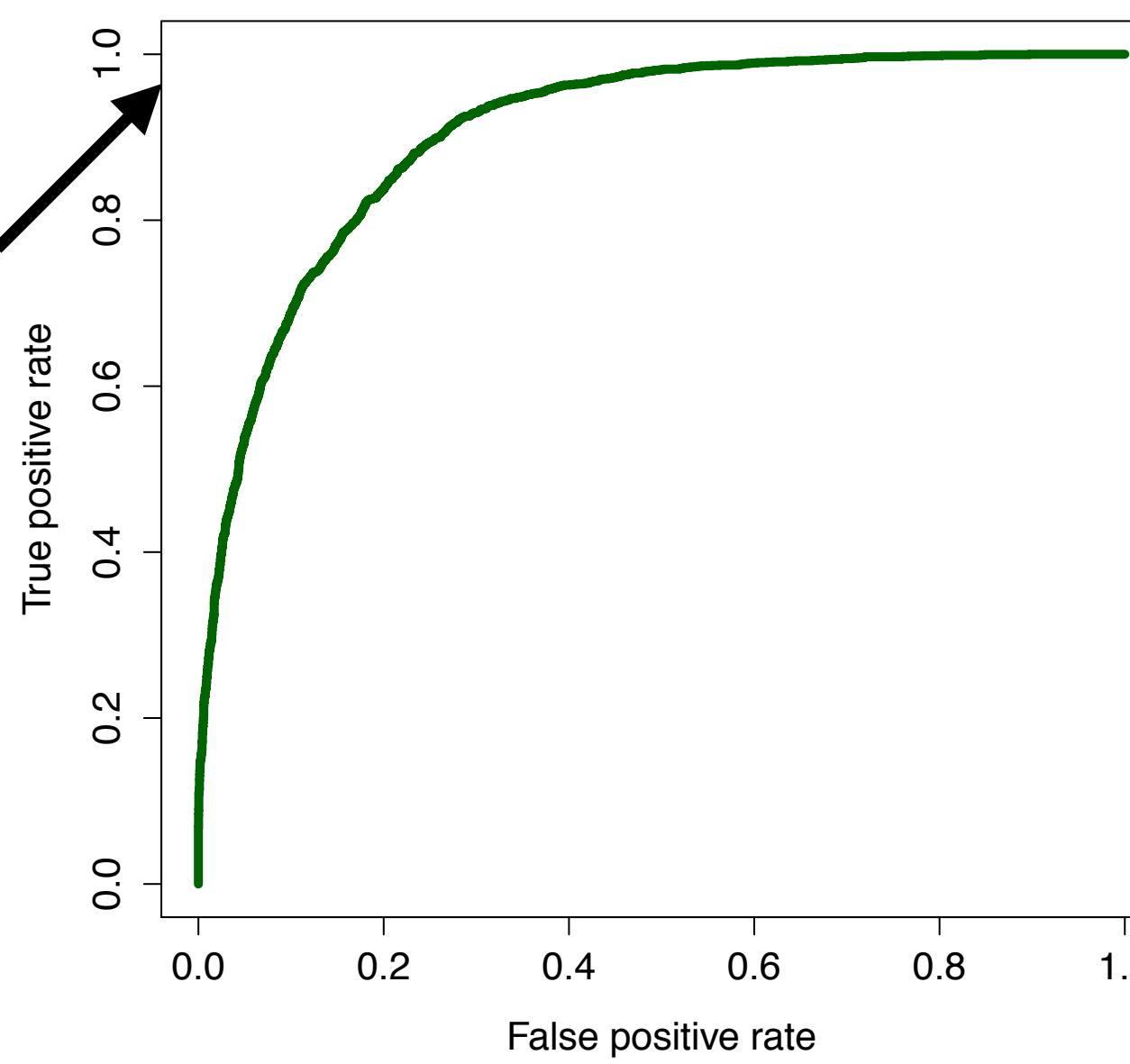


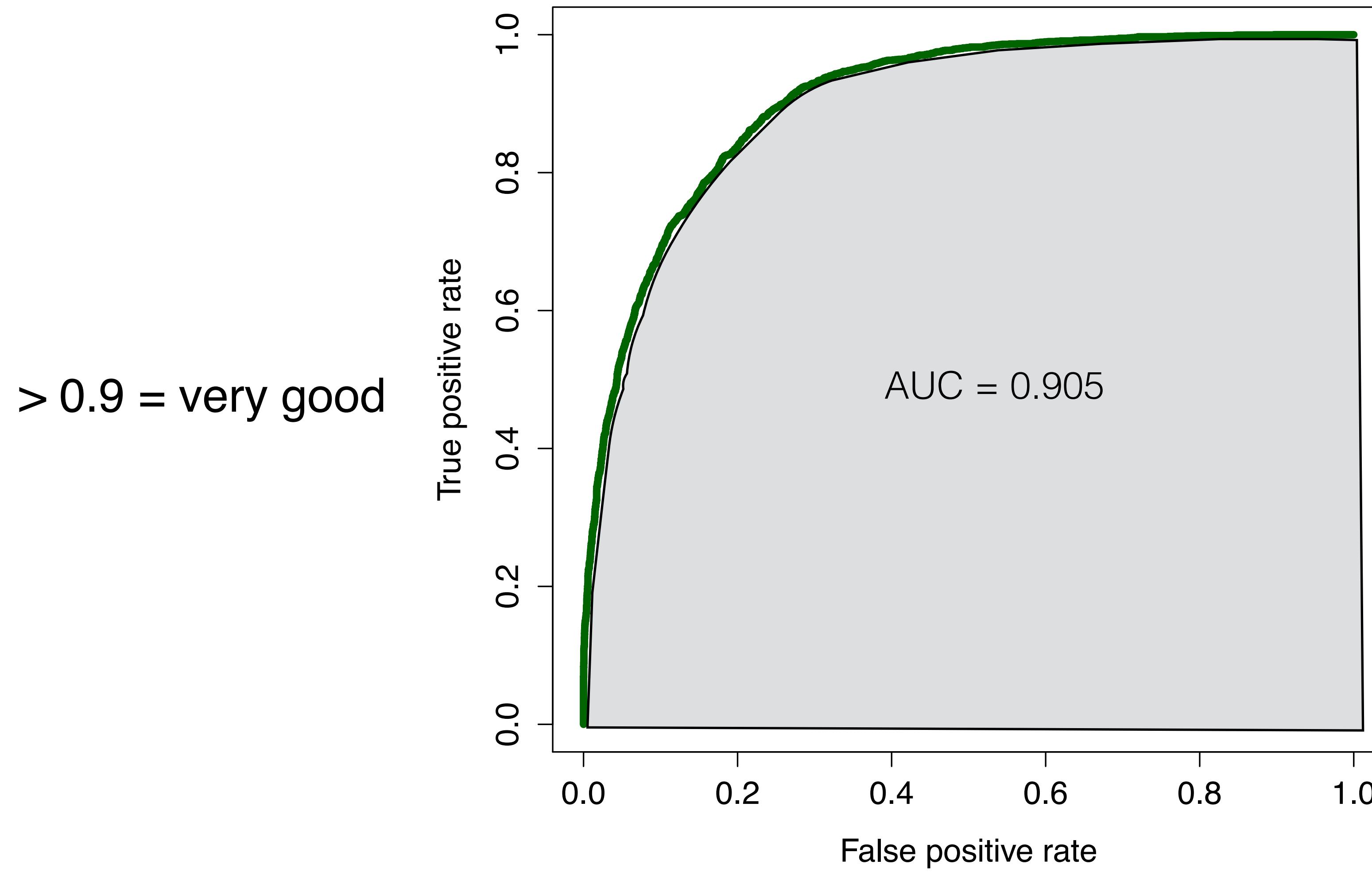
all healthy



Interpreting the curve

- Is it a good curve?
- Closer to left upper corner = better
- Good classifiers have big area under the curve





Area under the curve (AUC)



INTRODUCTION TO MACHINE LEARNING

Let's practice!