Get a Bit More Familiar with **Supervised Learning**
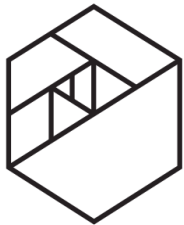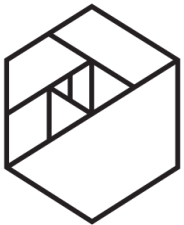
METIS

datascope

Time Series
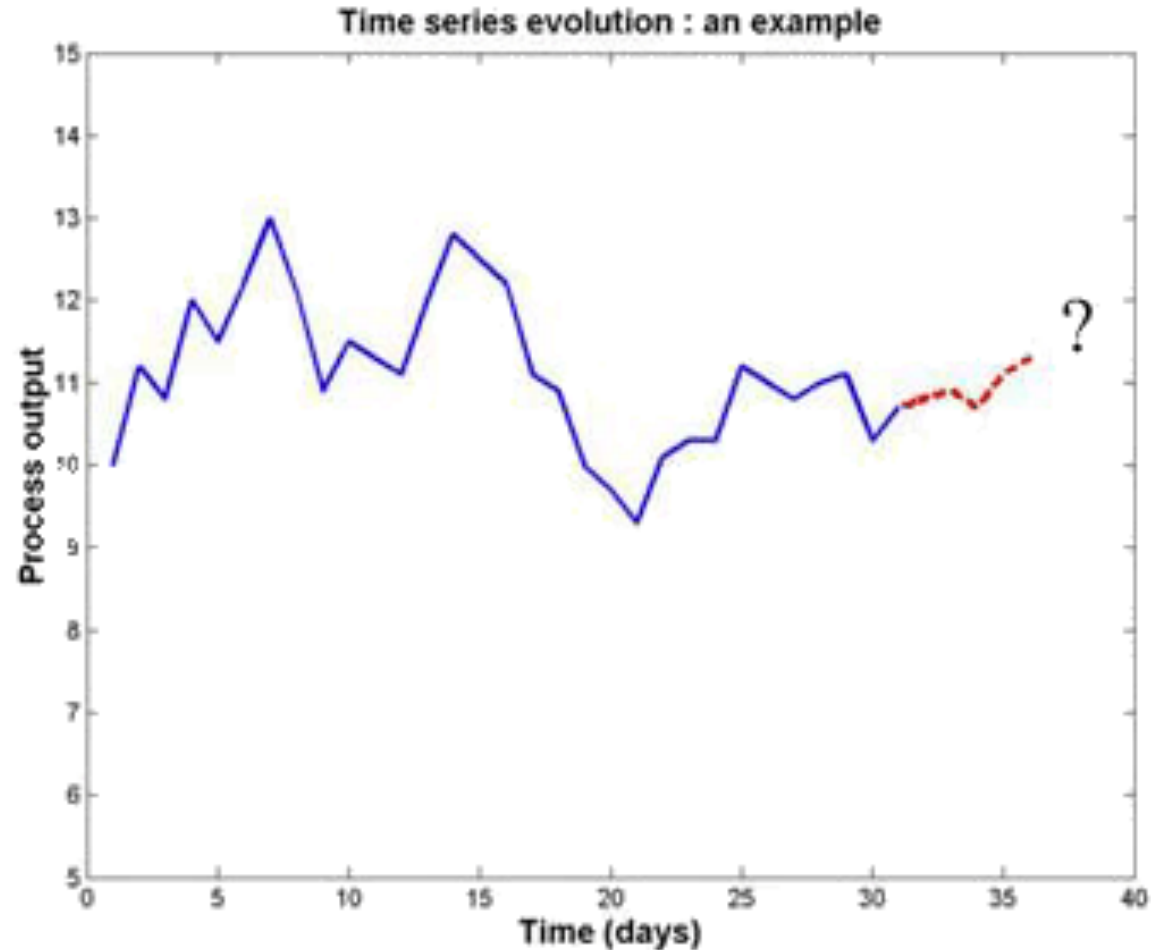
Time Series Prediction

Training on previous data, predicting numeric values: **Regression**.

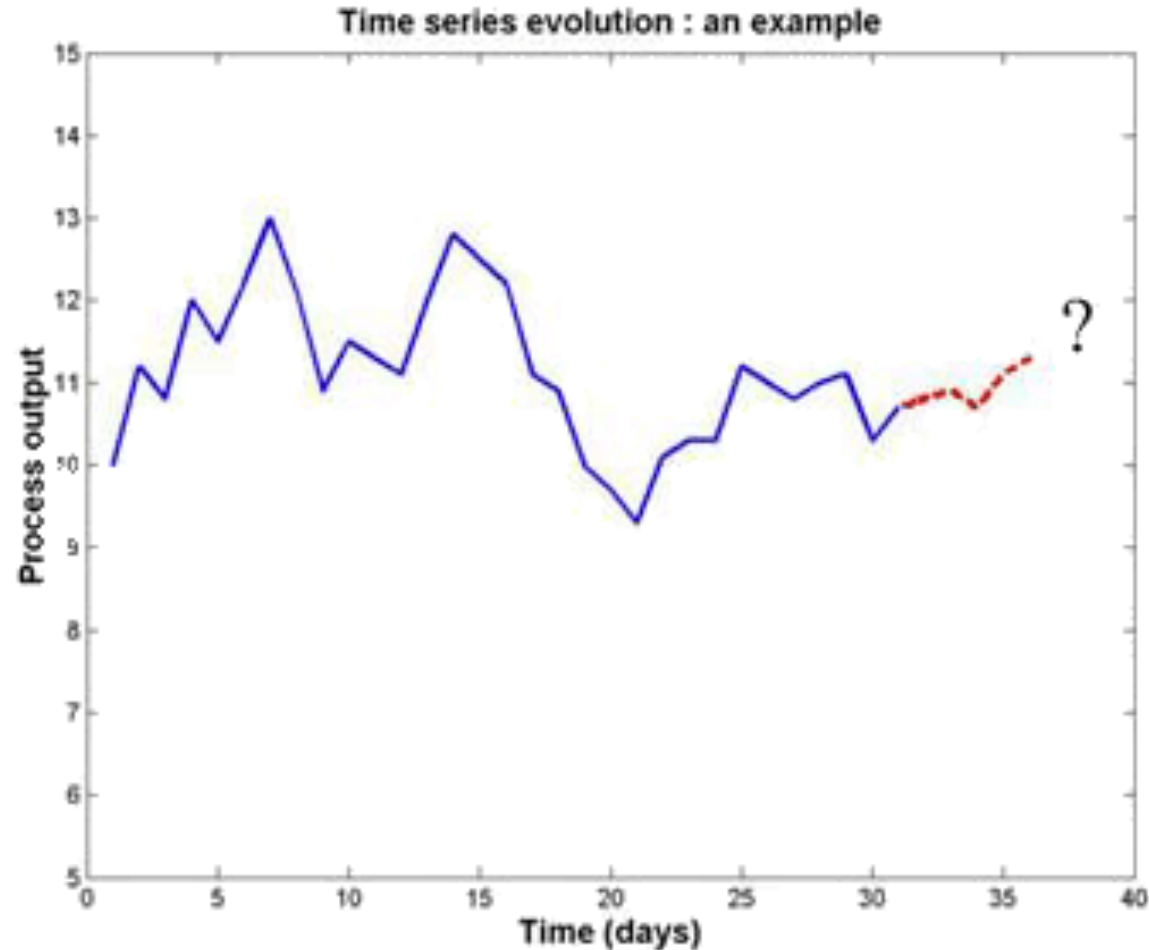Time series evolution : an example

Time Series Prediction

WARNING!
Time Series are autocorrelated.

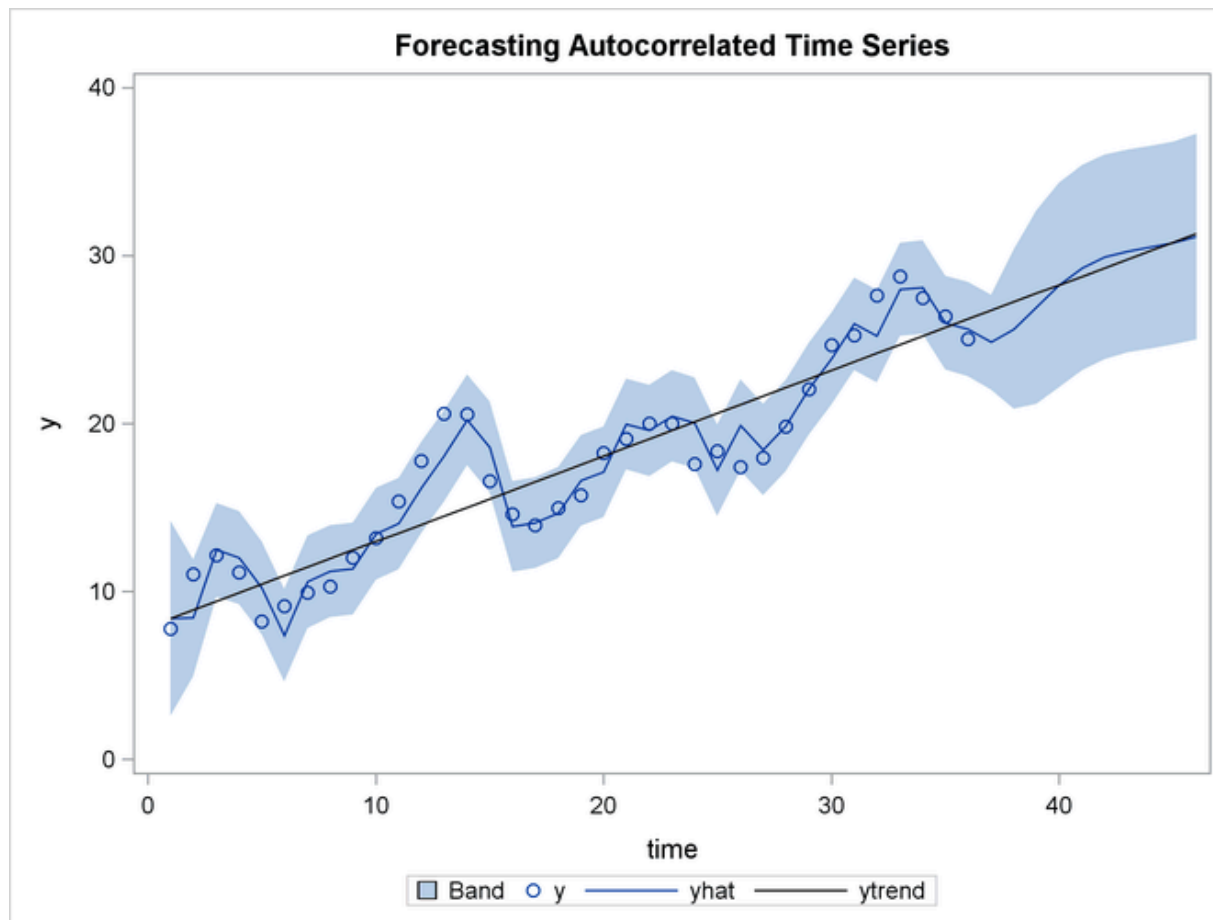Your regression is useless.

Unless…..

Time series evolution : an example

Autoregressive Models

AR1

$$y_\beta(x_t) = \beta_0 + \beta_t x_{t-1} + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon$$
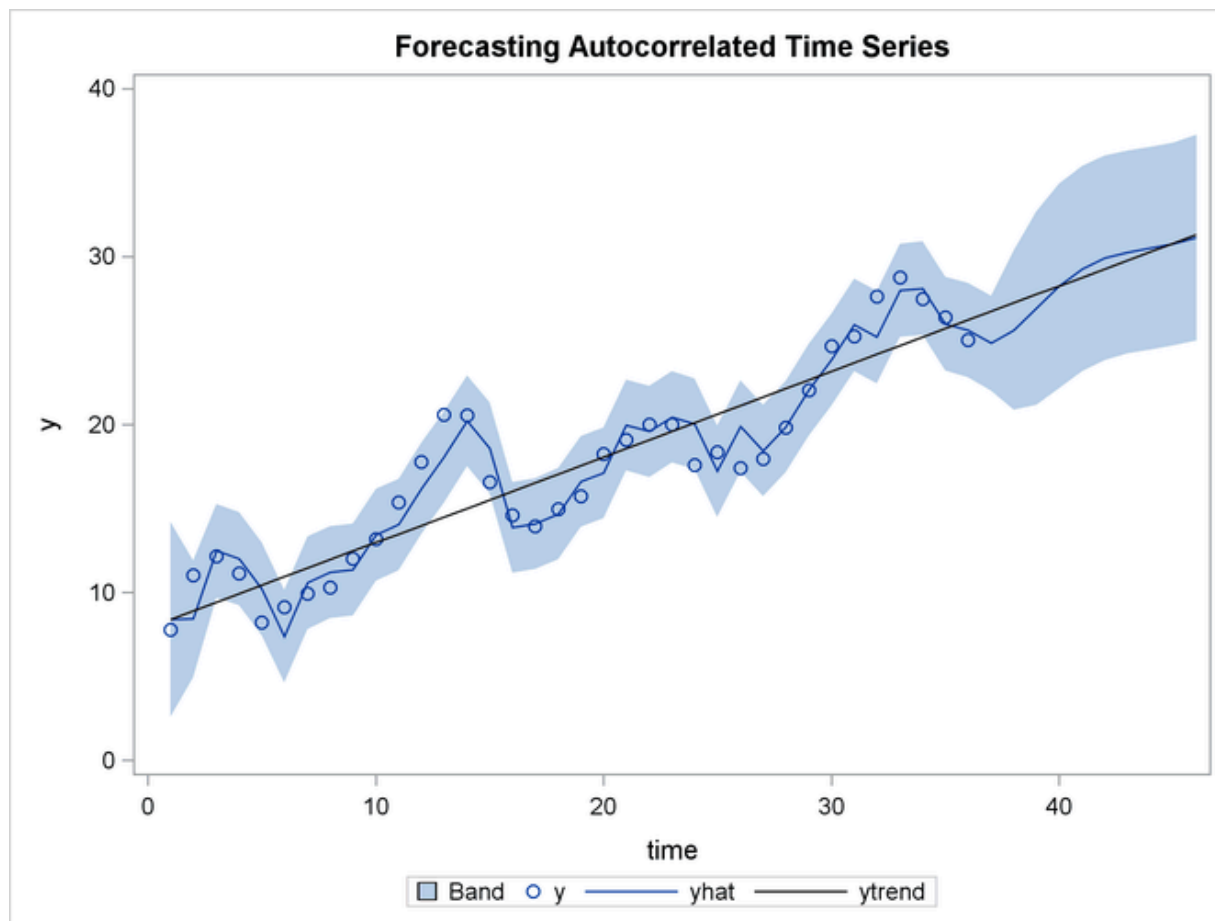

Forecasting Autocorrelated Time Series

Autoregressive Models

AR2

$$y_\beta(x_t) = \beta_0 + \beta_{t1}x_{t-1} + \beta_{t2}x_{t-2} + \beta_1 x_1 + ...$$
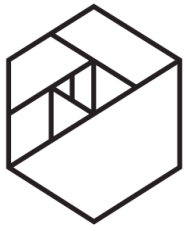

Forecasting Autocorrelated Time Series
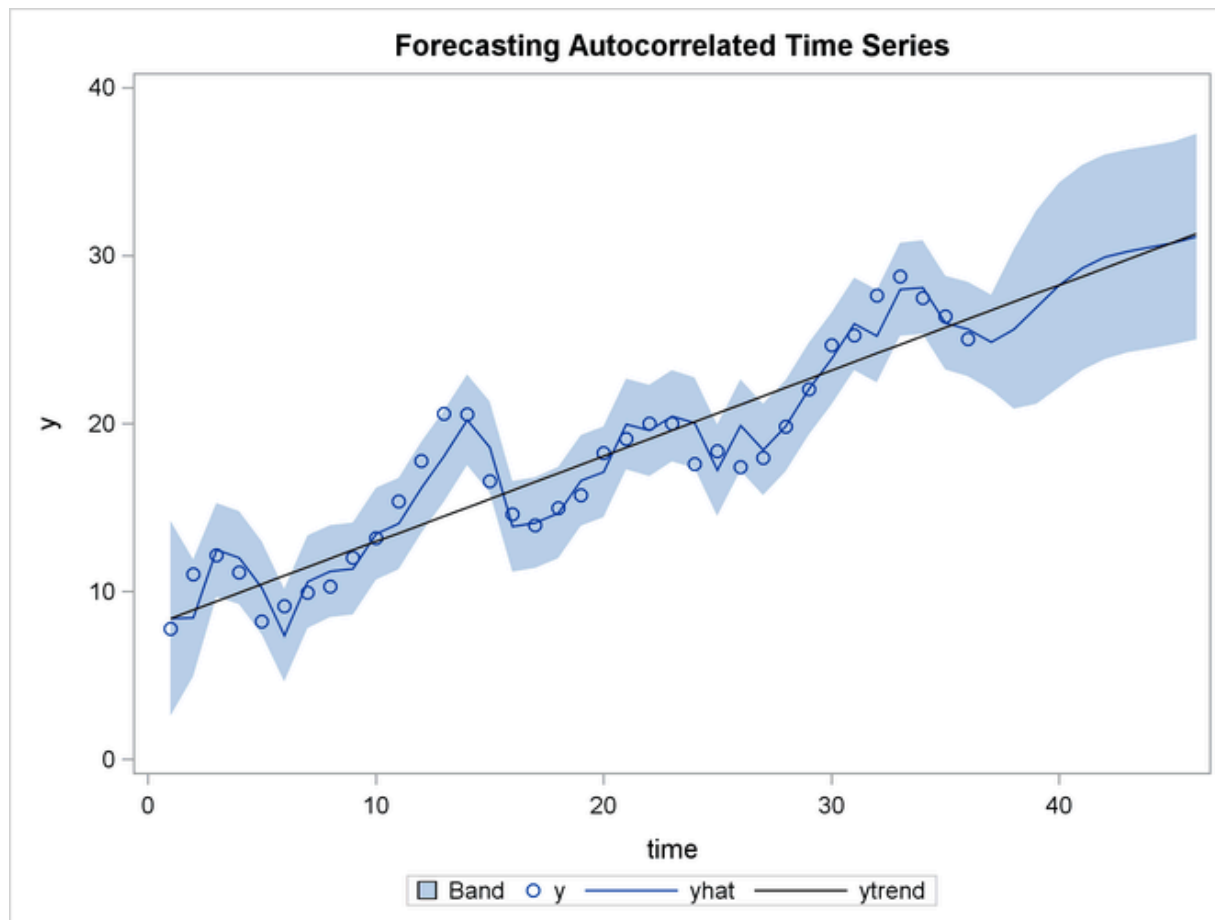
Autoregressive Models

$$y_\beta(x_t) = \beta_0 + \boxed{\beta_{t1}x_{t-1}} + \boxed{\beta_{t2}x_{t-2}} + \ldots + \text{average of last few points}$$

Forecasting Autocorrelated Time Series

Legend: □ Band ○ y — yhat — ytrend

Axis labels: y (vertical), time (horizontal)

Decision Trees

Decision Trees

Prediction:

Based on a bunch of IF ... THEN ... ELSE ... rules

Fitting:

Find a bunch of IF ... THEN ... ELSE ... rules to cover all cases as best you can.

Day	Outlook	Temp	Hum.	Wind	Play Tennis
Dl	Sunny	Hot	High	Weak	No
02	Sunny	Hot	High	Strong	No
03	Overcast	Hot	High	Weak	Yes
04	Rain	Mild	High	Weak	Yes
DS	Rain	Cool	Nml	Weak	Yes
06	Rain	Cool	Nml	Strong	No
07	Overcast	Cool	Nml	Strong	Yes
D B	Sunny	Mild	High	Weak	No
09	Sunny	Cool	Nml	Weak	Yes
DlO	Rain	Mild	Nml	Weak	Yes
Dll	Sunny	Mild	Nml	Strong	Yes
012	Overcast	Mild	High	Strong	Yes
013	Overcast	Hot	Nml	Weak	Yes
014	Rain	Mild	High	Strong	No

```
                        ┌─────────┐
                        │ Outlook │
                        └─────────┘
              Sunny       Overcast        Rain

         ┌──────────┐                  ┌──────┐
         │ Humidity │        Yes       │ Wind │
         └──────────┘                  └──────┘

    High         Normal          Strong        Weak

    No            Yes             No            Yes
```
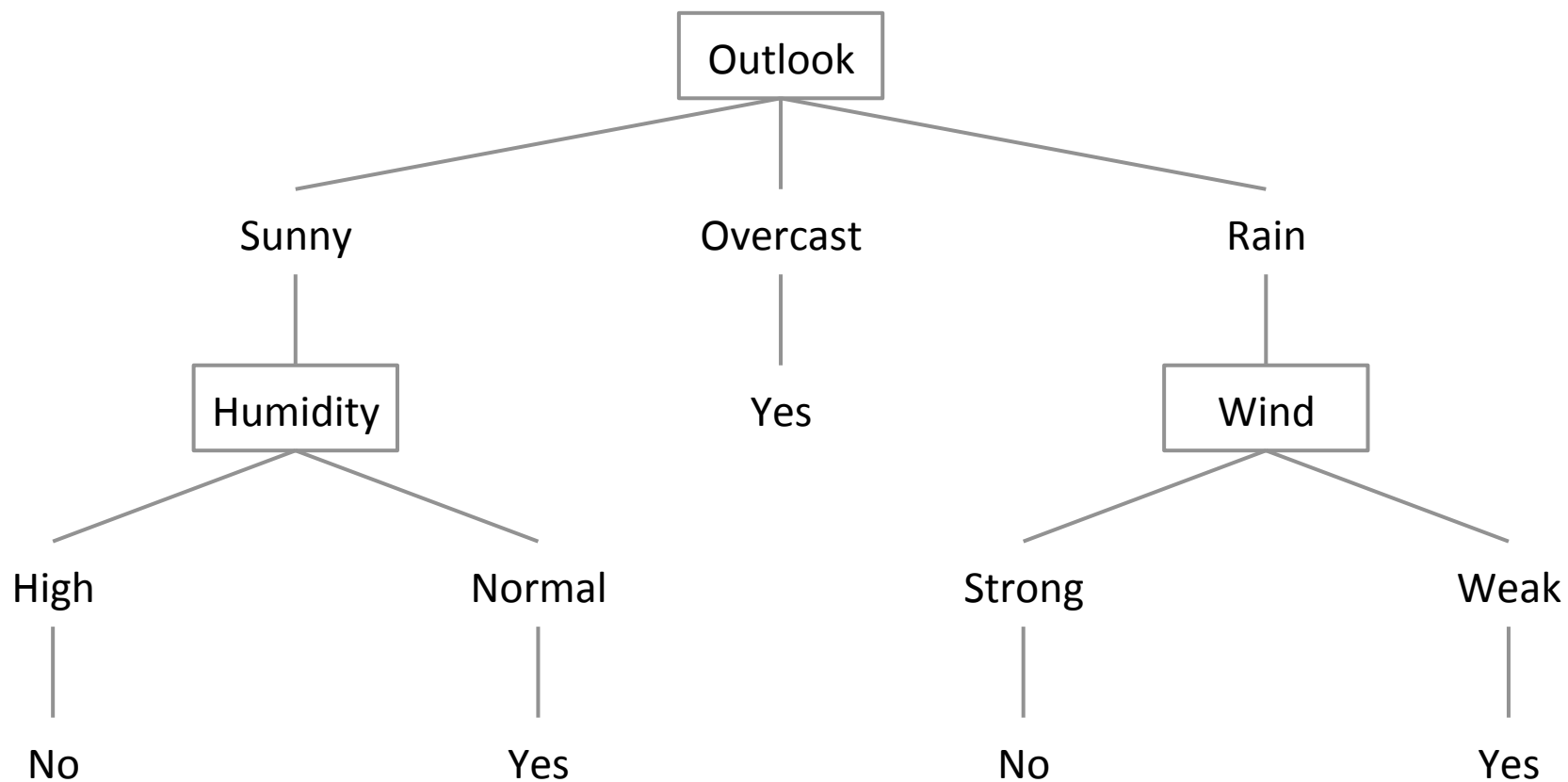
Each inner node is a decision based on a feature
Each leaf node is a class label
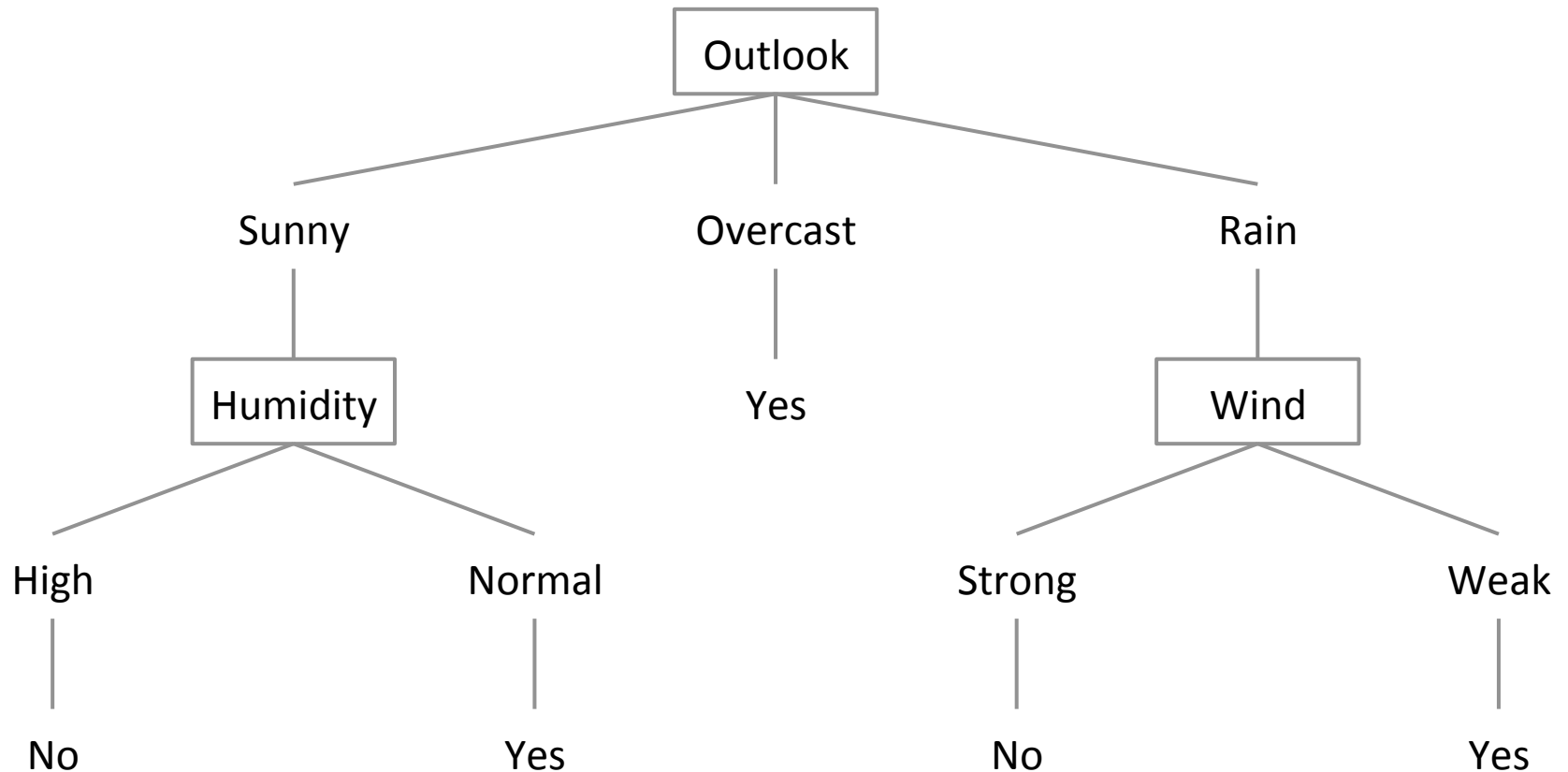
Each inner node is a decision based on a feature
Each leaf node is a **class label**

Predicting Titanic Survivors

Yes   **Is sex male?**   No

Is age > 9.5?   Survived
0.73   36%

Died
0.17   61%

Is sibsp > 2.5?

Died   Survived
0.05   2%   0.89   2%
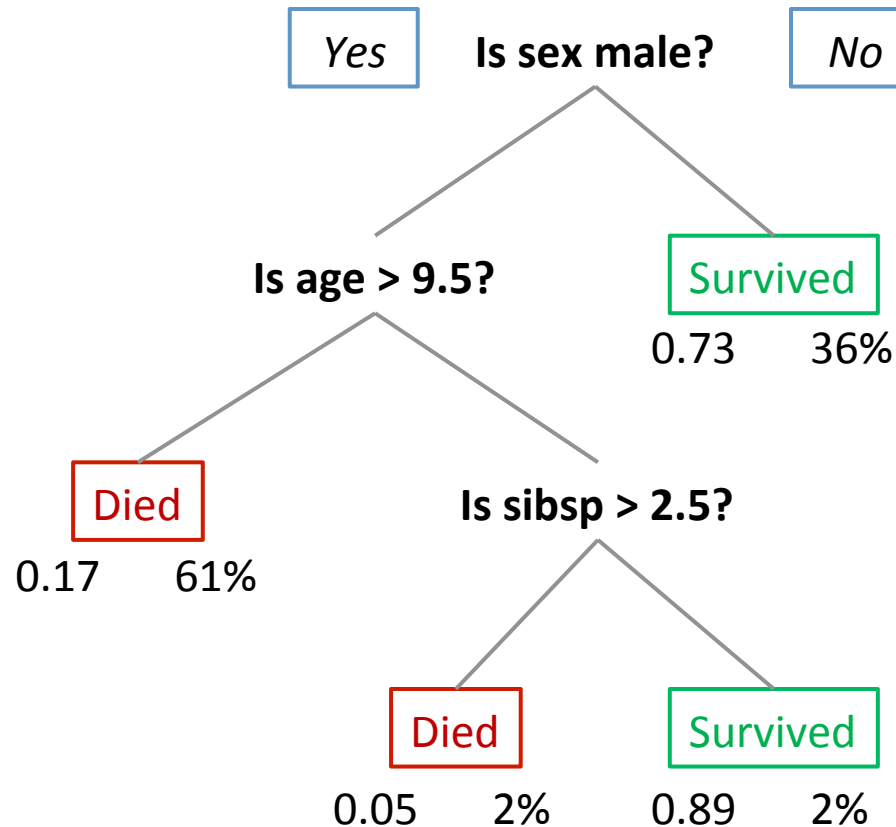
It can also be used for regression

CART
(Classification and Regression Tree)

Each inner node is a decision based on a feature

Each leaf node is a **predicted value**

Predicting precipitation in Himalayas from elevation, slope and position
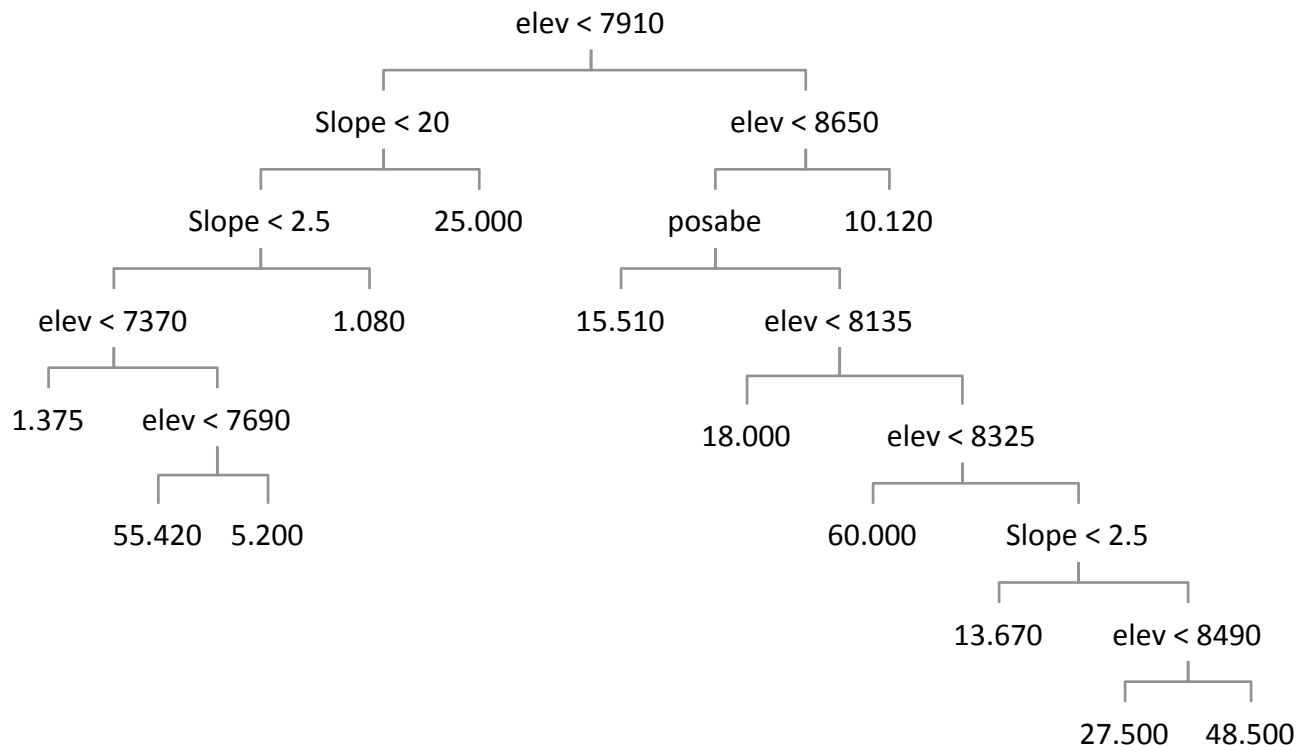
Decision Tree Regression
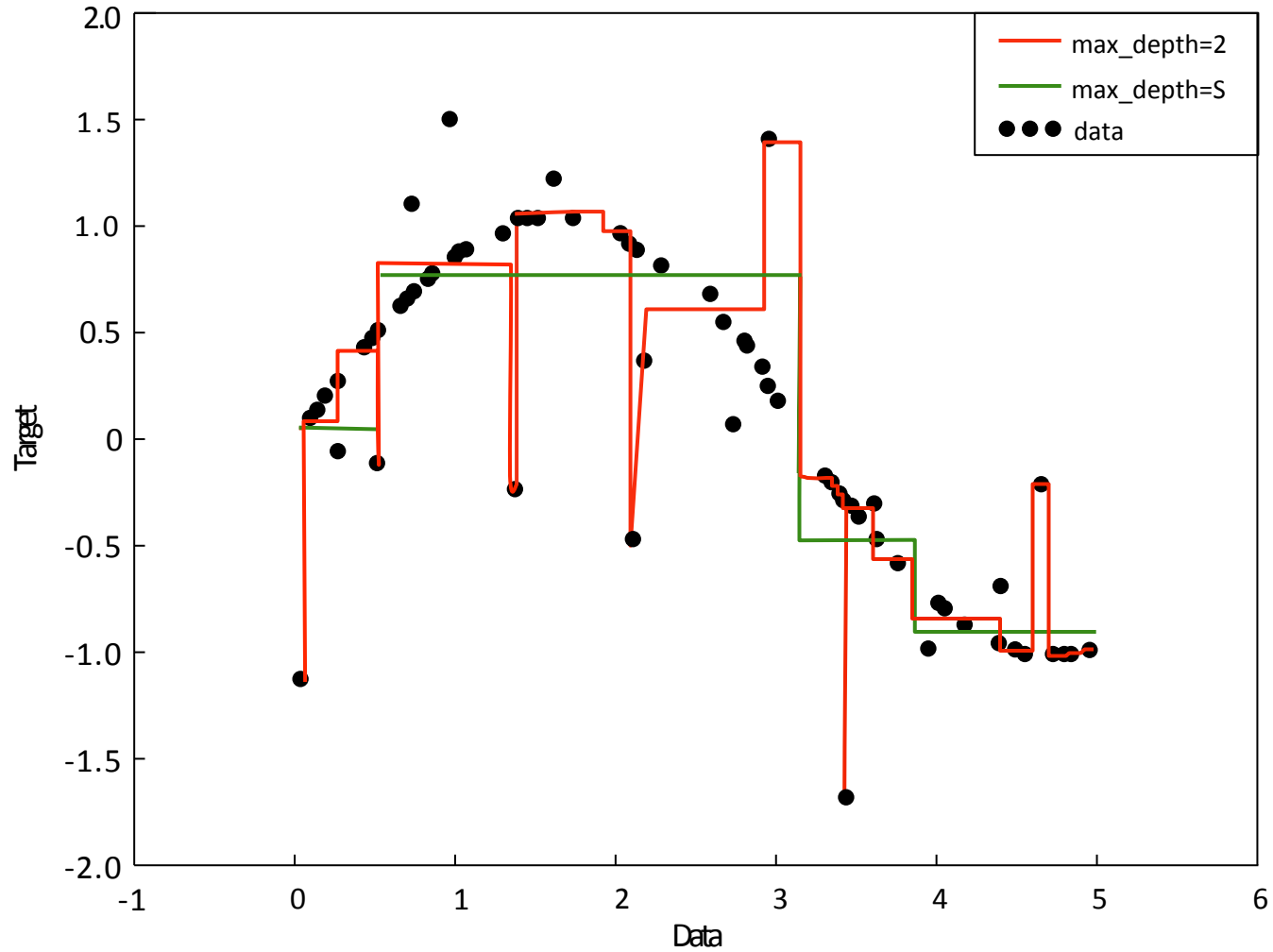
```python
from sklearn.tree import DecisionTreeClassifier

from sklearn.tree import DecisionTreeRegressor
```

Advantages:

The decisions are easy to understand and interpret.
Both numerical and categorical features can be used naturally.
Natural multiclass classifier.

Advantages:

The decisions are easy to understand and interpret.
Both numerical and categorical features can be used naturally.
Natural multiclass classifier.

Disadvantages:

Can overfit to training data with complex trees.
Small changes in input data can result in totally different trees.
Can make mistakes with unbalanced classes.
No confidence intervals (regression).

So, how do we build the best tree?

Like we fit any other model?
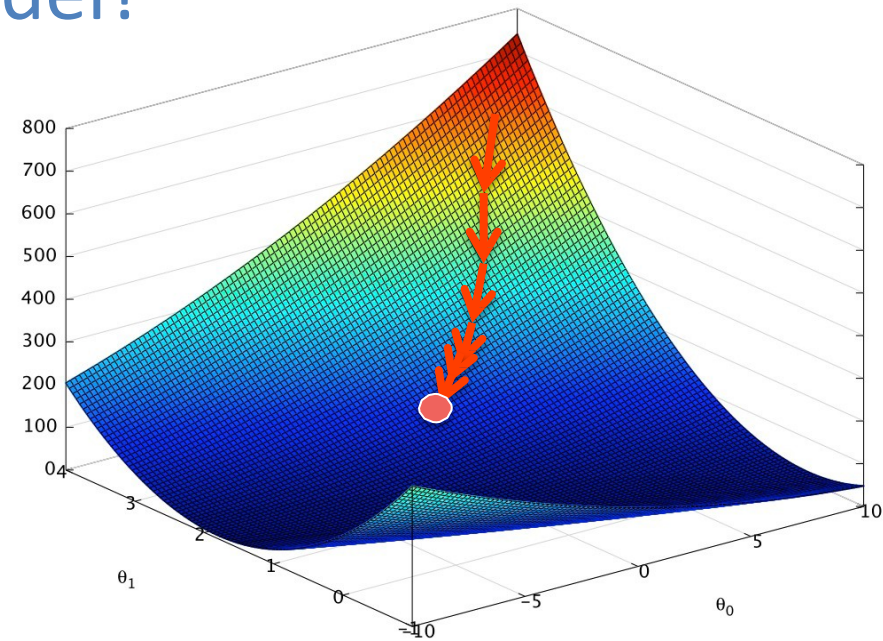
Define a cost function:
a score for a specific model (tree)

Try a bunch of models (trees), choose best

Use optimization algorithm to try in a smart way

Like we fit another model?

800
700

Define a cost
a score f

Try a bundle models (trees), best

Use optimization algorithm to try in a smart way

θ_0

Build tree split by split,
find the best split you can at each step

Build tree split by split,
find the best split you can at each step

Yes **Is sex male?** No

Survived
0.73   36%

Build tree split by split,
find the best split you can at each step

Yes	**Is sex male?**	No

Is age > 9.5?

Survived

0.73   36%

Died

0.17   61%

Build tree split by split,
find the best split you can at each step

| Yes | **Is sex male?** | No |

Is age > 9.5?

Survived
0.73   36%

Died
0.17   61%

Is sibsp > 2.5?

Died
0.05   2%

Survived
0.89   2%
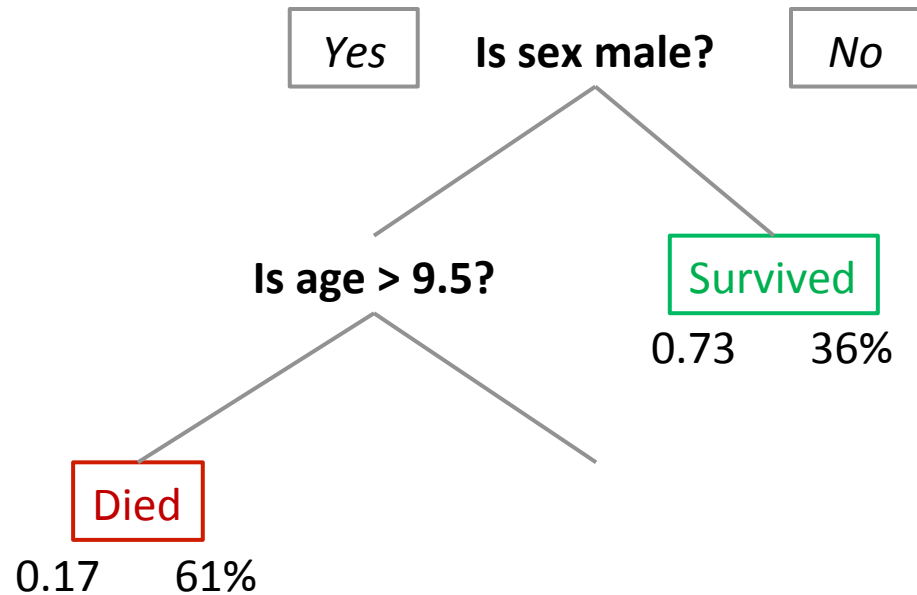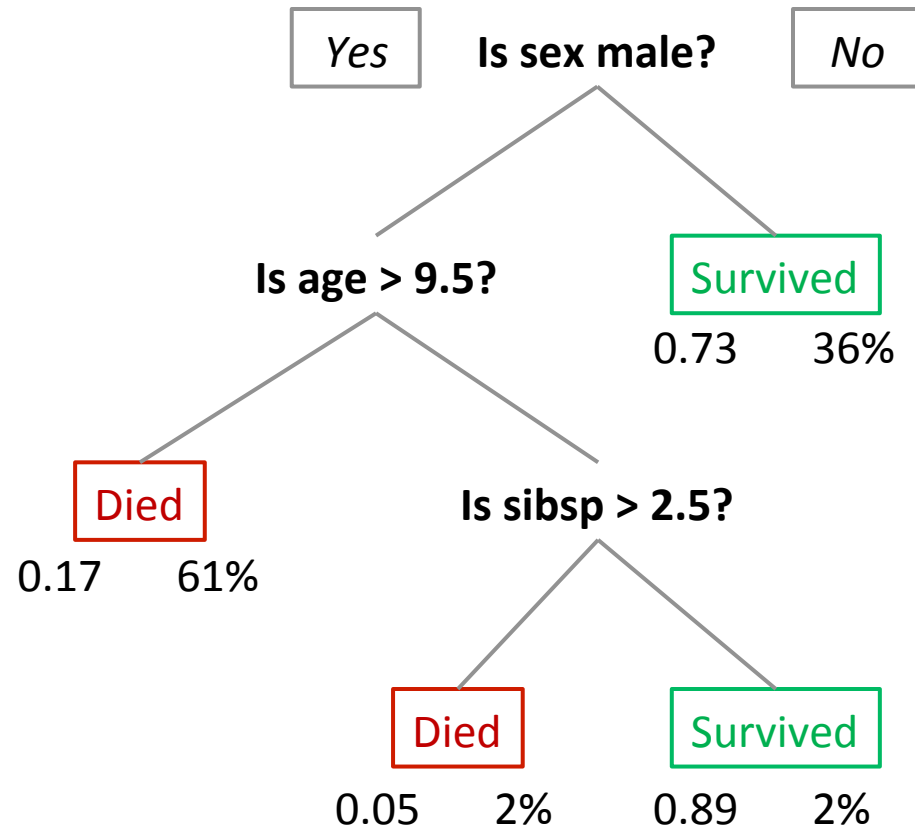
Use a heuristic: Greedy search

Build tree split by split,
find the **best split** you can at each step

?

Information Entropy
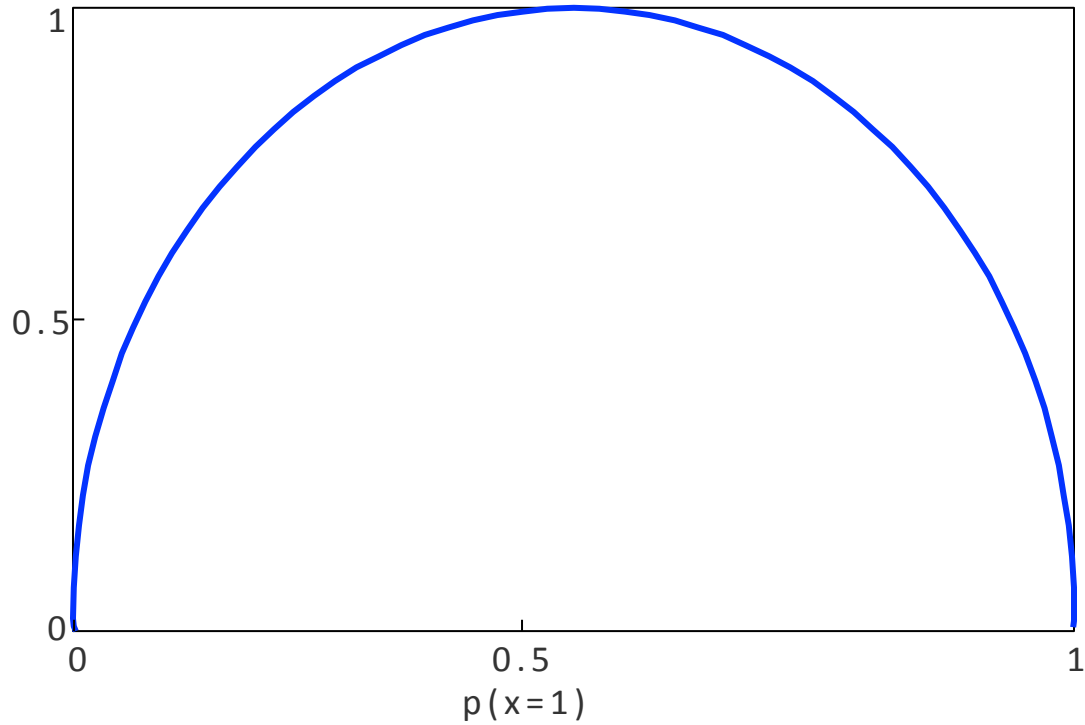
$$H = \sum_i p(x_i) \log p(x_i)$$

Split with the max entropy rule. Repeat.
Leaf node if only one class left
(or a performance metric reached,
or maximum depth reached)

Split with the max entropy rule. Repeat.
Leaf node if only one class left
(or a performance metric reached,
or maximum depth reached)

| Yes | **Is sex male?** | No |

Survived

0.73   36%

Split with the max entropy rule. Repeat.
Leaf node if only one class left
(or a performance metric reached,
or maximum depth reached)

Yes   **Is sex male?**   No

Is age > 9.5?

Survived
0.73   36%

Died
0.17   61%

Split with the max entropy rule. Repeat.
Leaf node if only one class left
(or a performance metric reached, or maximum depth reached)

| Yes | **Is sex male?** | No |

Is age > 9.5?

Survived
0.73   36%

Died
0.17   61%

Is sibsp > 2.5?

Died
0.05   2%

Survived
0.89   2%
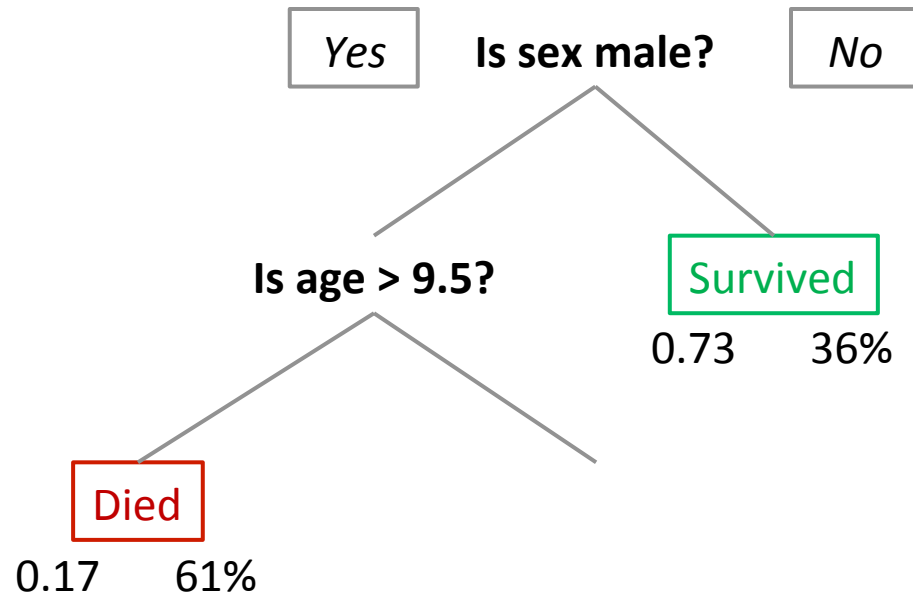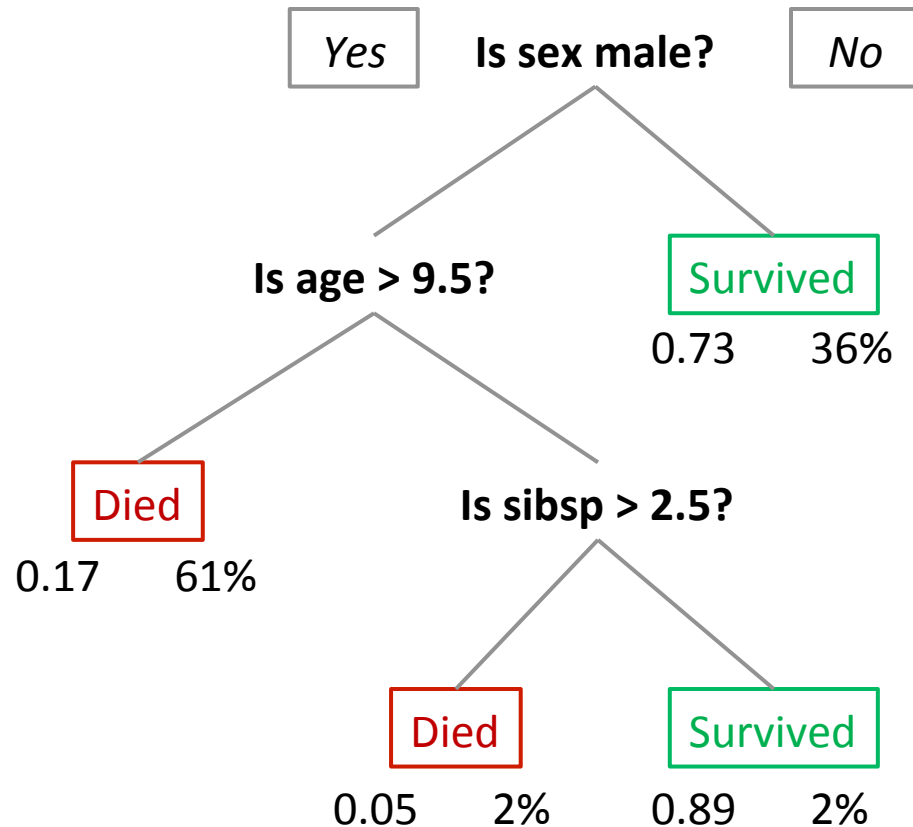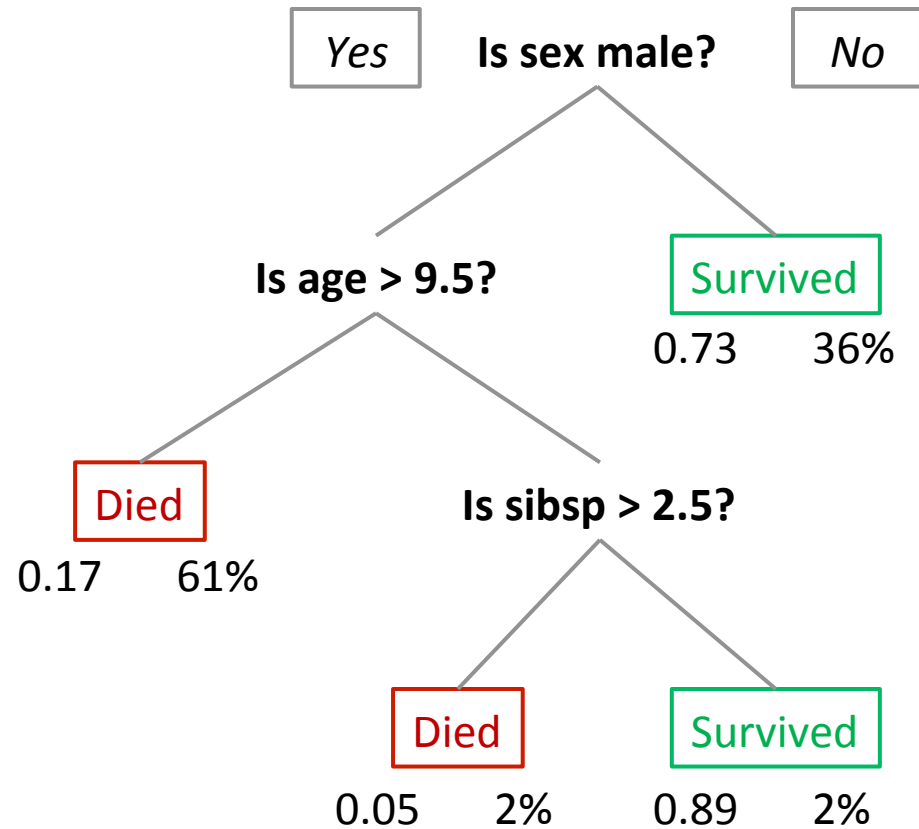
May overfit. Prune.

May overfit. Prune.

Yes   **Is sex male?**   No

Is age > 9.5?   Survived
0.73   36%

Died   Survived
0.17   61%
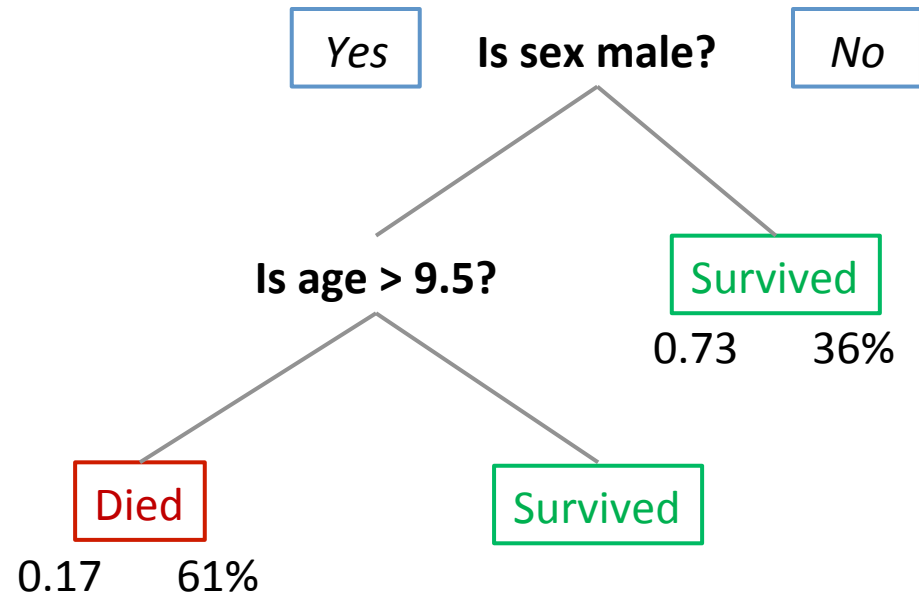
May overfit. Prune.
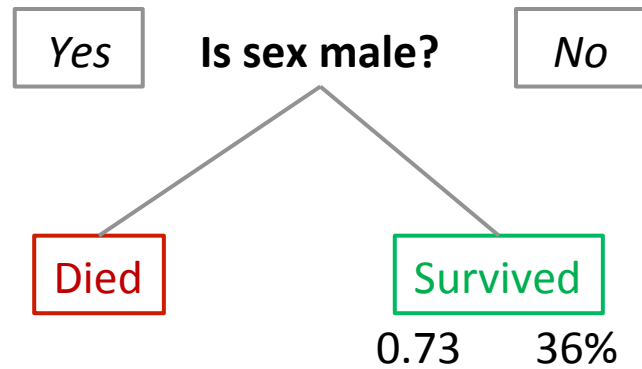
Better yet: Use ensemble methods
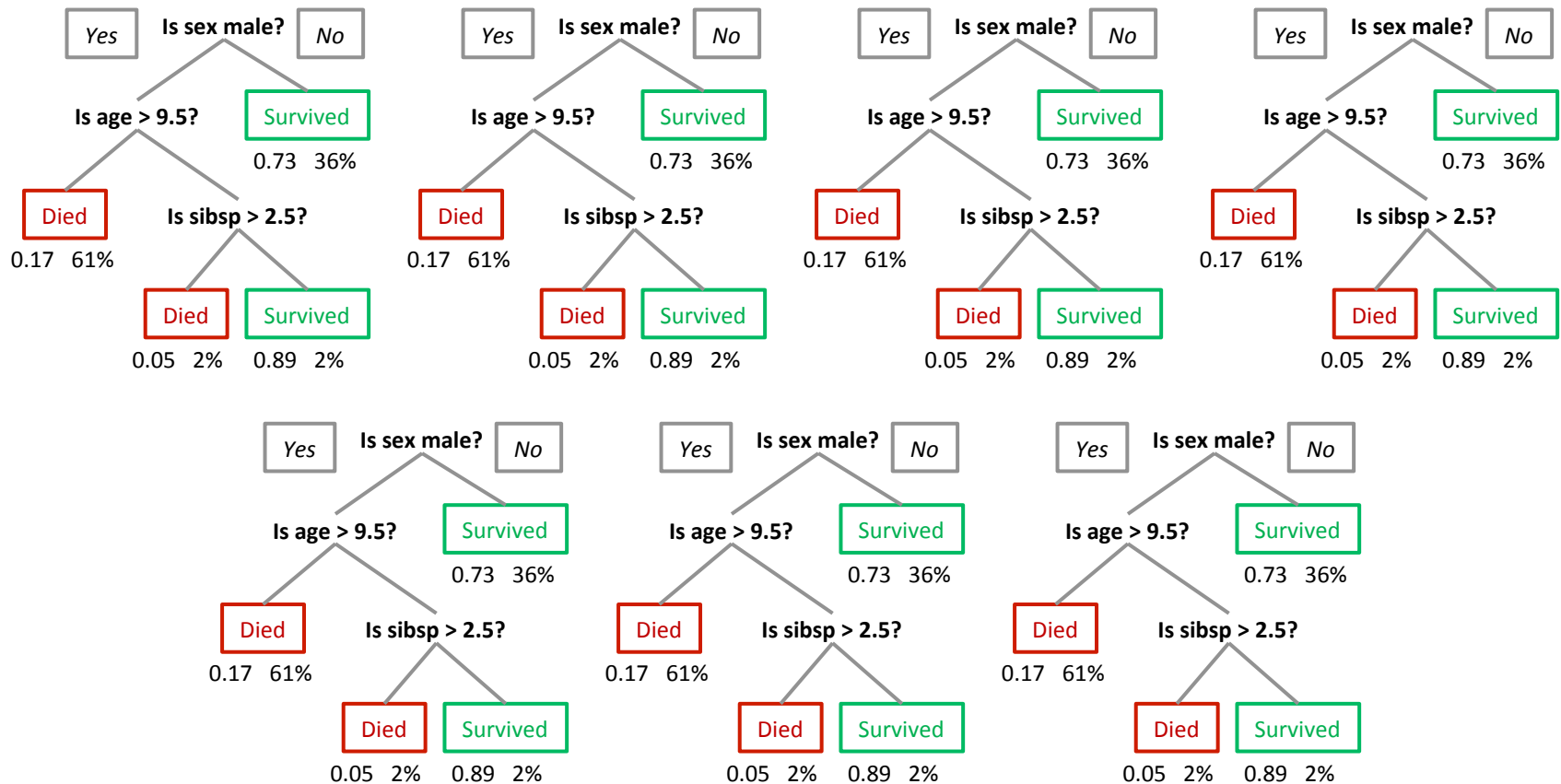
Ensemble Methods:
Bagging
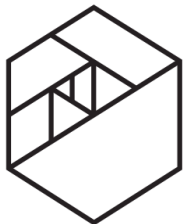
METIS

datascope

Bootstrap aggregating

Problem:

Overfitting to training set

Solution:

Bootstrap training set into multiple sets

Fit a model to each random set

Each model has one vote, choose max vote

	Date	Title	Budget	DomesticTotalGross	Director	Rating	Runtime
0	2013-11-22	The Hunger Games: Catching Fire	130000000	424668047	Francis Lawrence	PG-13	146
1	2013-05-03	Iron Man 3	200000000	409013994	Shane Black	PG-13	128
2	2013-11-22	Frozen	150000000	400738009	Chris BuckJennifer Lee	PG	108
3	2013-07-03	Despicable Me 2	760000000	368061265	Pierre CoffinChris Renaud	PG	98
4	2013-06-14	Man of Steel	225000000	291045518	Zack Snyder	PG-13	143
5	2013-10-04	Gravity	100000000	274092705	Alfonso Cuaron	PG-13	91
6	2013-06-21	Monsters University	NaN	268492764	Dan Scanion	G	107
7	2013-12-13	The Hobbit: The Desolation of Smaug	NaN	258366855	Peter Jackson	PG-13	161
8	2013-05-24	Fast & Furious 6	160000000	238679850	Justin Lin	PG-13	130
9	2013-03-08	OZ The Great and Powerful	215000000	234911825	Sam Raimi	PG	127
10	2013-05-16	Star Trek Into Darkness	190000000	228778661	J.J. Abrams	PG-13	123
11	2013-11-08	Thor: The Drak World	170000000	206362140	Alan Taylor	PG-13	120
12	2013-06-21	World War Z	190000000	202359711	Marc Forster	PG-13	116
13	2013-03-22	The Croods	135000000	187168425	Kirk De MiccoChris Sanders	PG	98
14	2013-06-28	The Heat	43000000	159582188	Paul Feig	R	117
15	2013-08-07	We're the Millers	37000000	150394119	Rawson Marshall Thurber	R	110
16	2013-12-13	American Hustle	40000000	150117807	David O. Russell	R	138
17	2013-05-10	The Great Gatsby	105000000	144840419	Baz Luhrmann	PG-13	143

Training set

Test set

Bootstrap: Sample with replacement

Bootstrap: Sample with replacement
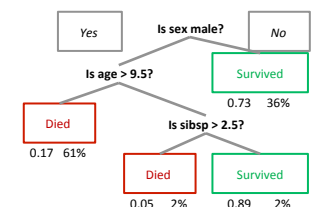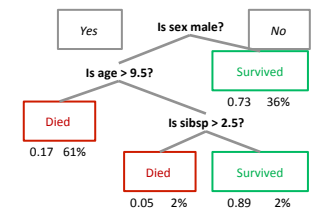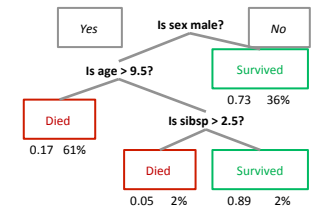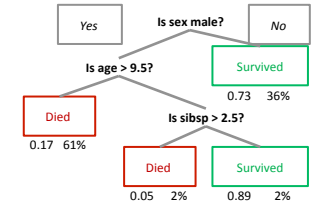
Classification: Major vote
Regression: Average outcome
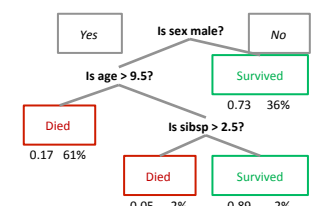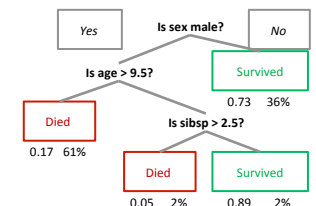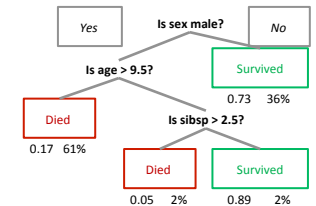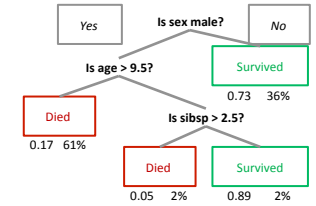
```python
from sklearn.ensemble import BaggingClassifier
```

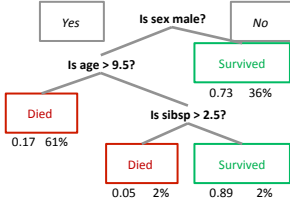# Ensemble Methods:
Random Forests
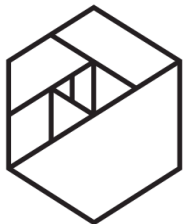
Tree Bagging with a twist
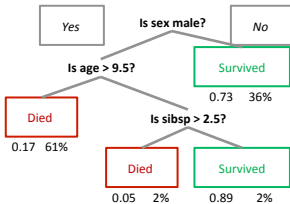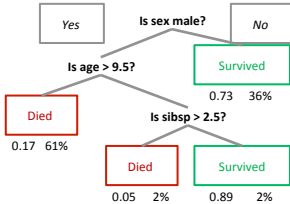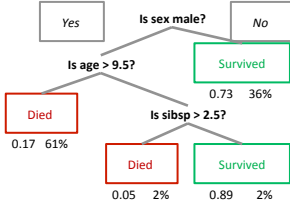
Introduce randomness when building each tree.

Introduce randomness when building each tree.
For a split, do not take the "best" feature split.

Introduce randomness when building each tree.
For a split, do not take the "best" feature split.
First, randomly choose sqrt(n_feat) features.

Yes	**Is sex male?**	No

Is age > 9.5?   Survived
                          0.73   36%
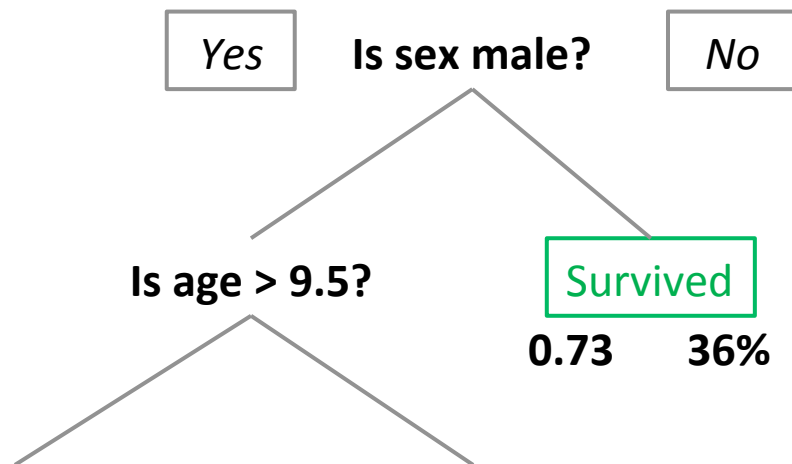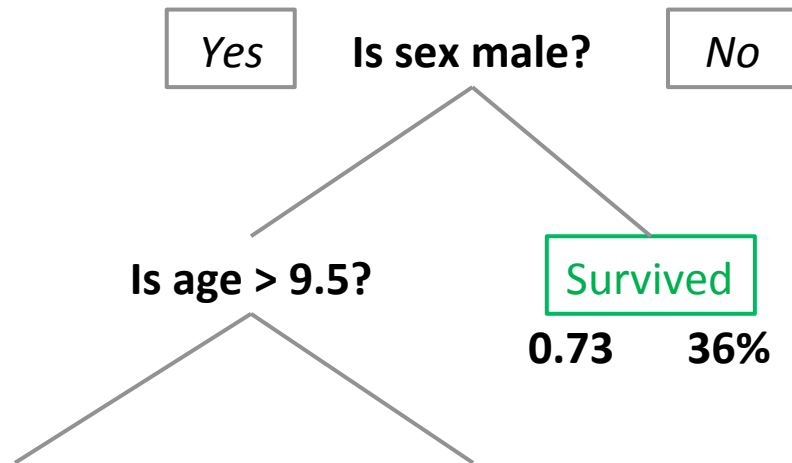
Introduce randomness when building each tree.
For a split, do not take the "best" feature split.
First, randomly choose sqrt(n_feat) features.
Only choose the best split among these.
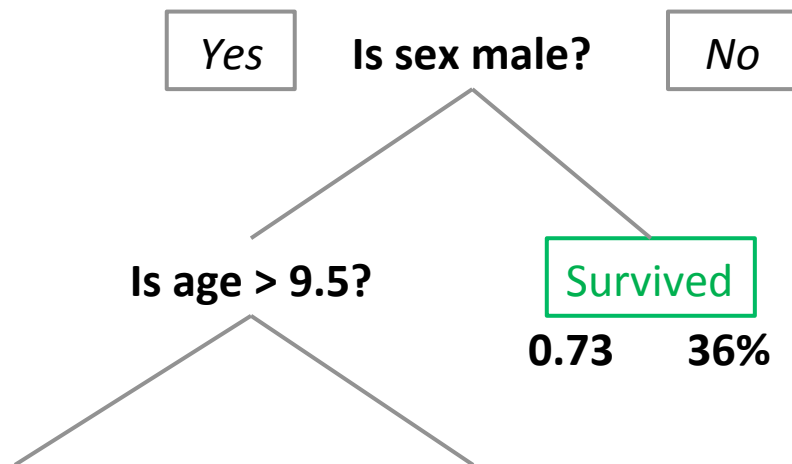
```python
from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import RandomForestRegressor
```

```python
from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import RandomForestRegressor


from sklearn.ensemble import ExtraTreesClassifier?
```

```python
from sklearn.ensemble import RandomForestClassifier

from sklearn.ensemble import RandomForestRegressor


from sklearn.ensemble import ExtraTreesClassifier
```

Even more randomized:

For each feature, split rule is random, not optimal