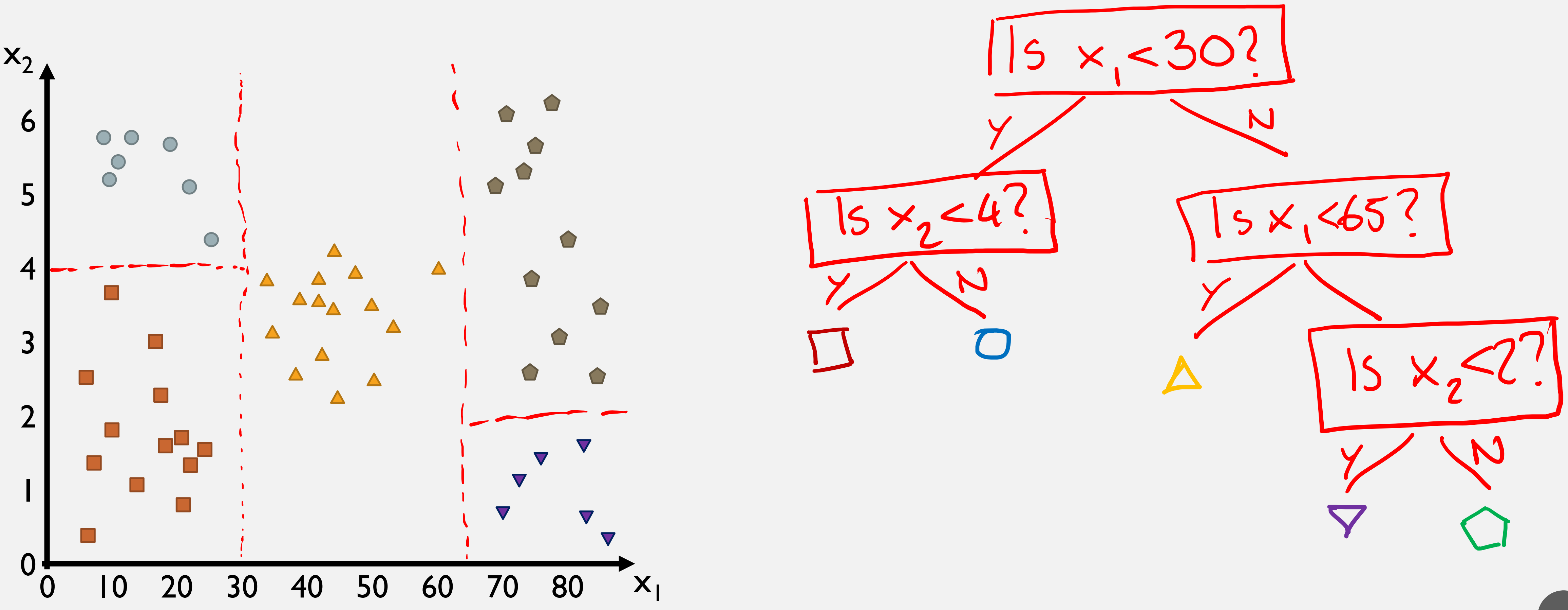
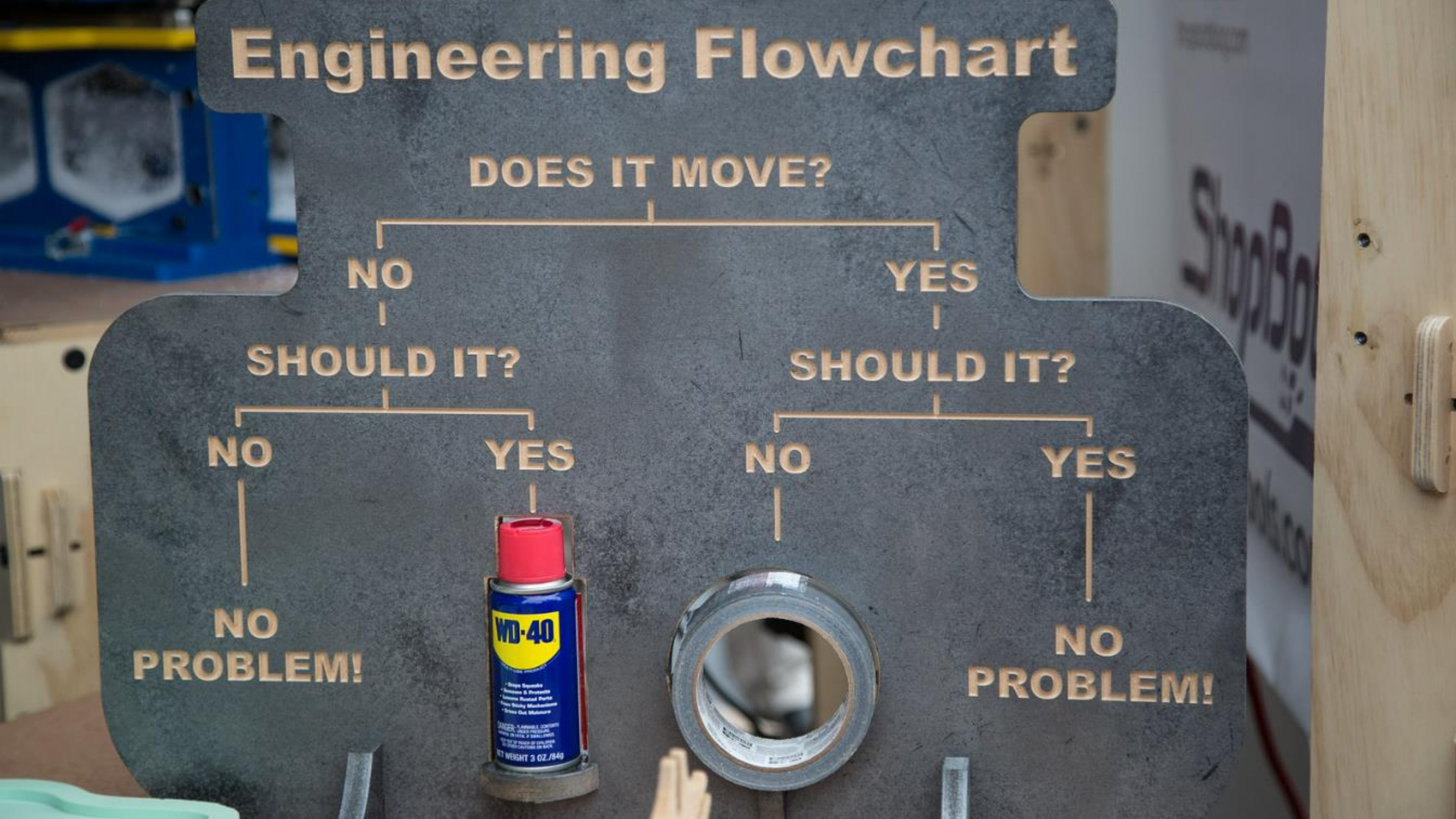
TREE-BASED MODELS

Lecture 3

MAL1, 2024

DECISION TREES




Engineering Flowchart
DOES IT MOVE?
NO
YES
SHOULD IT?
SHOULD IT?
NO
YES
NO
YES
NO PROBLEM!
WD-40
NO PROBLEM!

HOW DECISION TREES WORK

- Step 1

  Find the feature that is the best predictor of your data

- Step 2

  Partition instances of your training set according to that feature

  data points

- Step 3

  Repeat 1-2 recursively

- Stop when

  All instances in a given node belong to the same class

  or

  There are no more ways to split

A LOAN IN THE BANK

A FICTITIOUS EXAMPLE

id	salary	savings	debt	class
1	Low	High	True	Approved
2	Low	Low	False	Declined
3	High	Low	False	Approved
4	Low	Low	True	Declined
5	High	Low	True	Approved
6	High	High	False	Approved
7	High	Low	False	Approved
8	Low	Low	True	Declined
9	High	High	True	Approved
10	Low	Low	False	Declined
11	Low	High	False	Approved
12	Low	Low	True	Declined

How do we decide which feature to branch off on?

many possibilities

most common is the Gini impurity index

"How much information is gained by selecting a certain feature"

A LOAN IN THE BANK

A FICTITIOUS EXAMPLE

id	salary	savings	debt	class
1	Low 1	High	True	Approved
2	Low 2	Low	False	Declined
3	High	Low	False	Approved
4	Low 3	Low	True	Declined
5	High	Low	True	Approved
6	High	High	False	Approved
7	High	Low	False	Approved
8	Low 4	Low	True	Declined
9	High	High	True	Approved
10	Low 5	Low	False	Declined
11	Low 6	High	False	Approved
12	Low 7	Low	True	Declined

The Gini impurity index

Gini index

$$G(D) = 1 - \sum_j p_j^2 = 1 - \left(\frac{7}{12}\right)^2 - \left(\frac{5}{12}\right)^2 = 0.49$$

Dataset

sum over classes j

sum over partition

$$G_k(D) = \sum_i \frac{n_i}{n} G(D_i)$$

feature

$$G_{\text{salary}}(D) = \underbrace{\frac{7}{12}\left(1 - \left(\frac{2}{7}\right)^2 - \left(\frac{5}{7}\right)^2\right)}_{\text{low salary}} + \underbrace{\frac{5}{12}\left(1 - \left(\frac{5}{5}\right)^2 - \left(\frac{0}{5}\right)^2\right)}_{\text{high salary}}$$

$= 0.24$

A LOAN IN THE BANK

A FICTITIOUS EXAMPLE

id	salary	savings	debt	class
1	Low	High	True	Approved
2	Low	Low	False	Declined
3	High	Low	False	Approved
4	Low	Low	True	Declined
5	High	Low	True	Approved
6	High	High	False	Approved
7	High	Low	False	Approved
8	Low	Low	True	Declined
9	High	High	True	Approved
10	Low	Low	False	Declined
11	Low	High	False	Approved
12	Low	Low	True	Declined

The Gini impurity index

$$G_{salary}(D) = 0.24$$
$$G_{savings}(D) = 0.31$$
$$G_{debt}(D) = 0.47$$

The best predictor always has the lowest Gini index

A LOAN IN THE BANK

A FICTITIOUS EXAMPLE

id	salary	savings	debt	class
1	Low	High	True	Approved
2	Low	Low	False	Declined
~~3~~	~~High~~	~~Low~~	~~False~~	~~Approved~~
4	Low	Low	True	Declined
~~5~~	~~High~~	~~Low~~	~~True~~	~~Approved~~
~~6~~	~~High~~	~~High~~	~~False~~	~~Approved~~
~~7~~	~~High~~	~~Low~~	~~False~~	~~Approved~~
8	Low	Low	True	Declined
~~9~~	~~High~~	~~High~~	~~True~~	~~Approved~~
10	Low	Low	False	Declined
11	Low	High	False	Approved
12	Low	Low	True	Declined

Beginning to draw the tree

salary high

y

n

approved

~~7 total~~

2 appr, 5 decl

recalculate Gini indices for this dataset

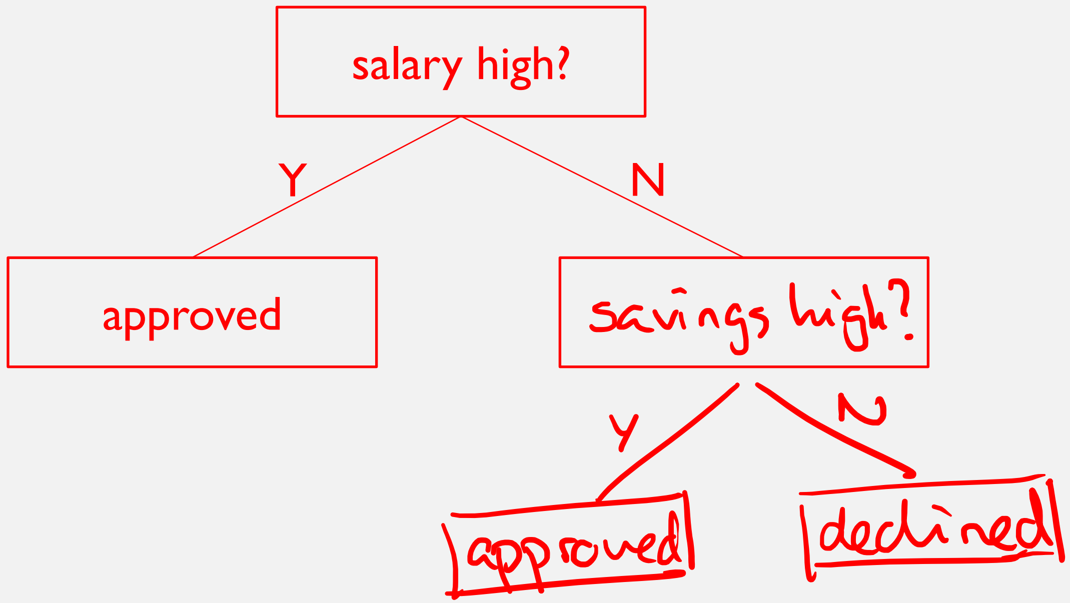
⇒ savings

A LOAN IN THE BANK

A FICTITIOUS EXAMPLE

id	salary	savings	debt	class
1	Low	High	True	Approved
2	Low	Low	False	Declined
3	High	Low	False	Approved
4	Low	Low	True	Declined
5	High	Low	True	Approved
6	High	High	False	Approved
7	High	Low	False	Approved
8	Low	Low	True	Declined
9	High	High	True	Approved
10	Low	Low	False	Declined
11	Low	High	False	Approved
12	Low	Low	True	Declined

Finishing the tree



LEARNING DECISION TREES

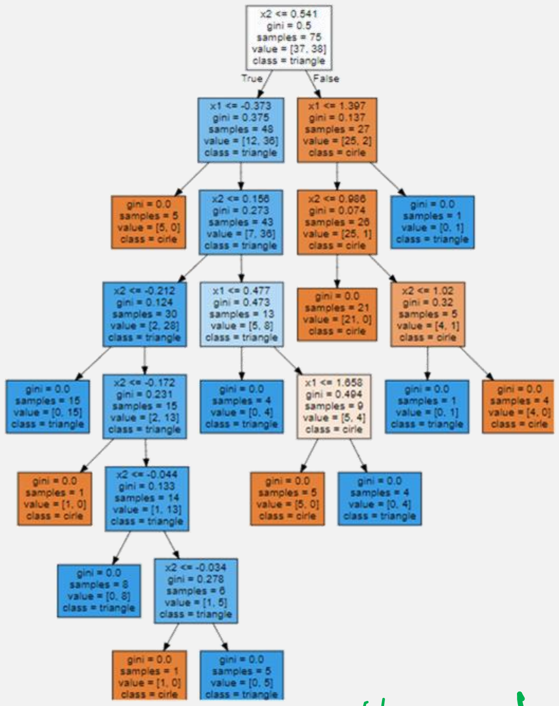
- means learning the sequence of if/else questions that gets us to the best answer most quickly
- the questions may be yes/no but usually of the form "is feature i > value a?"
- the algorithm searches over all possible tests and finds the most informative one

VISUALIZATION

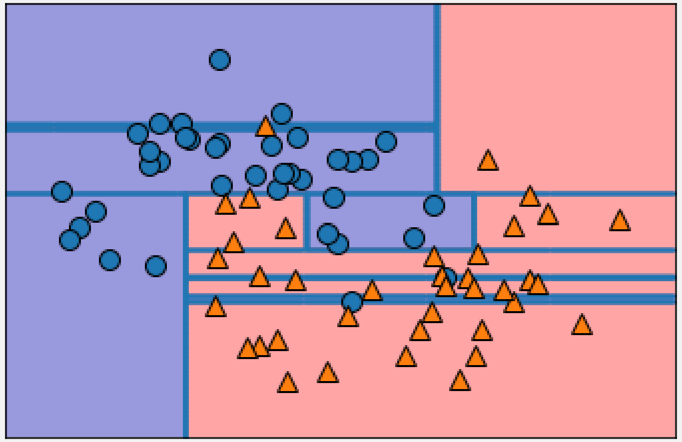
Jupyter Notebook **Decision Trees I: Visualization and hyperparameters**

VISUALIZATION

TREE



DECISION BOUNDARY



:) Informative + easily explained to a non-expert

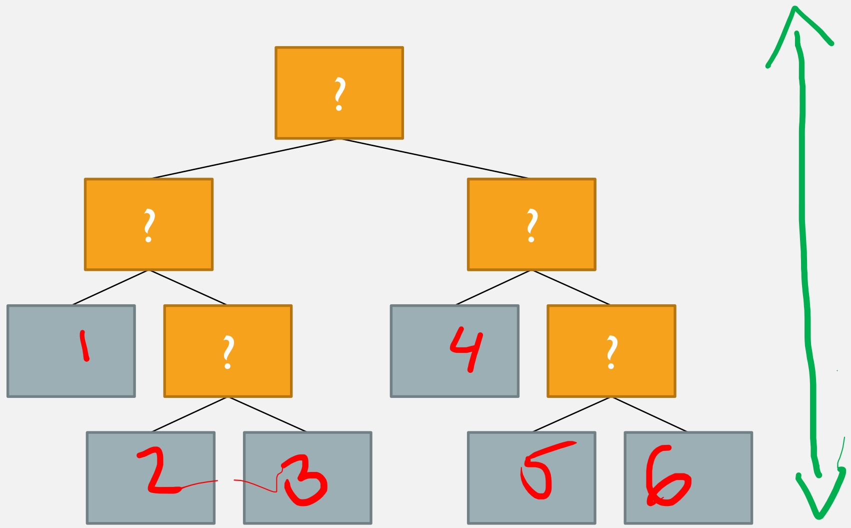
:(Quickly gets overwhelming

Easily interpreted

Only for 2D data

OVERFITTING AND HYPERPARAMETERS

```
Accuracy on training data: 1.0
Accuracy on testing data: 0.92
```



max_depth

max # of questions in a branch

max_leaf_nodes

max # of leaves

min_samples_split

min # of data points a node should have to allow splitting

(**criterion**)

how we split (Gini)

Tuning these parameters is called *pre-pruning*

PRE-PRUNING

Jupyter Notebook **Decision Trees I: Visualization and hyperparameters**

PROS AND CONS OF DECISION TREES

Pros

Easily visualized and explained to a non-expert

Fast

Completely invariant to data scaling

Cons

Tends to overfit, even with pre-pruning

ENSEMBLES OF DECISION TREES

methods that combine several trees to make more powerful models

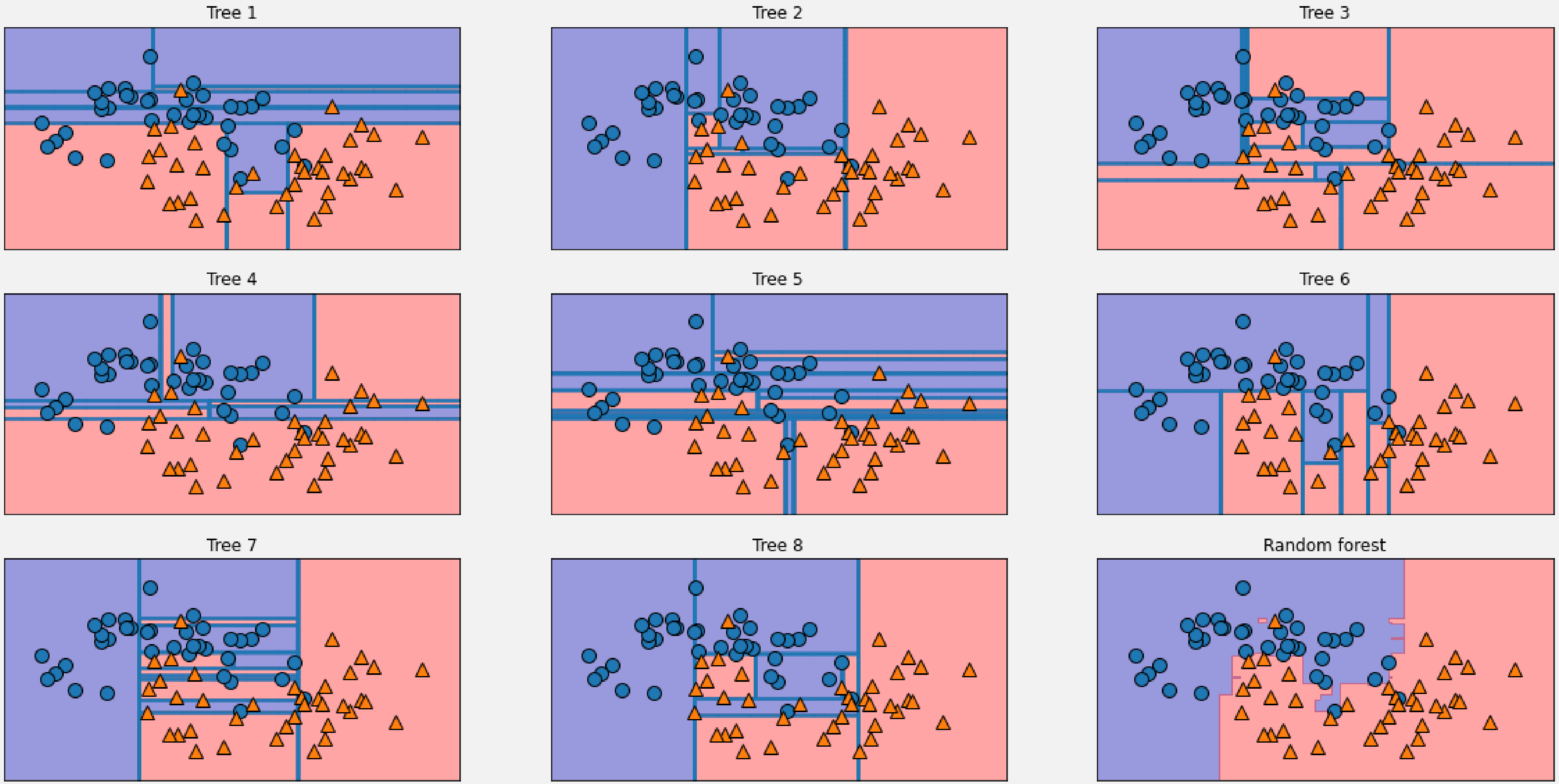
- **Random forests** *(bagging)*

  a collection of slightly different trees that overfit differently

- **Gradient boosted decision trees** *(boosting)*

  a sequence of trees where each tree tries to correct the mistakes of the previous one

RANDOM FORESTS



RANDOMIZATION I: BOOTSTRAPPING

	f_1	f_2	f_3	f_4	f_5	f_6
x_1	45	5	21	45	15	1
x_2	87	2	12	44	64	2
x_3	24	8	15	43	36	3
x_4	67	7	17	44	87	2
x_5	13	5	12	44	65	3
x_6	87	4	16	42	34	1
x_7	89	7	13	42	2	2
x_8	68	3	14	43	54	3
x_9	35	6	11	41	63	2

RNG

Numbers 9

Min 1

Max 9

Go

7
9
4
8
7
2
3
3
8

A bootstrap dataset

	f_1	f_2	f_3	f_4	f_5	f_6
x_7	89	7	13	42	2	2
x_9	35	6	11	41	63	2
x_4	67	7	17	44	87	2
x_8	68	3	14	43	54	3
x_7	89	7	13	42	2	2
x_2	87	2	12	44	64	2
x_3	24	8	15	43	36	3
x_3	24	8	15	43	36	3
x_8	68	3	14	43	54	3

RANDOMIZATION I: BOOTSTRAPPING

Dataset for tree 1

	f_1	f_2	f_3	f_4	f_5	f_6
x_7	89	7	13	42	2	2
x_9	35	6	11	41	63	2
x_4	67	7	17	44	87	2
x_8	68	3	14	43	54	3
x_7	89	7	13	42	2	2
x_2	87	2	12	44	64	2
x_3	24	8	15	43	36	3
x_3	24	8	15	43	36	3
x_8	68	3	14	43	54	3

Dataset for tree 2

	f_1	f_2	f_3	f_4	f_5	f_6
x_6	87	4	16	42	34	1
x_8	68	3	14	43	54	3
x_2	87	2	12	44	64	2
x_2	87	2	12	44	64	2
x_3	24	8	15	43	36	3
x_7	89	7	13	42	2	2
x_4	67	7	17	44	87	2
x_2	87	2	12	44	64	2
x_8	68	3	14	43	54	3

Dataset for tree 3

	f_1	f_2	f_3	f_4	f_5	f_6
x_3	24	8	15	43	36	3
x_3	24	8	15	43	36	3
x_8	68	3	14	43	54	3
x_7	89	7	13	42	2	2
x_1	45	5	21	45	15	1
x_1	45	5	21	45	15	1
x_6	87	4	16	42	34	1
x_5	13	5	12	44	65	3
x_7	89	7	13	42	2	2

RANDOMIZATION II: FEATURE SELECTION

Dataset for tree 1

	f_1	f_2	f_3	f_4	f_5	f_6
x_7	89	7	13	42	2	2
x_9	35	6	11	41	63	2
x_4	67	7	17	44	87	2
x_8	68	3	14	43	54	3
x_7	89	7	13	42	2	2
x_2	87	2	12	44	64	2
x_3	24	8	15	43	36	3
x_3	24	8	15	43	36	3
x_8	68	3	14	43	54	3

For each node, randomly select a subset of features and ask the best question involving those features

e.g. "is $f_2 > 6$?"

RANDOMIZATION II: FEATURE SELECTION

Dataset for tree 1

	f_1	f_2	f_3	f_4	f_5	f_6
x_7	89	7	13	42	2	2
x_9	35	6	11	41	63	2
x_4	67	7	17	44	87	2
x_8	68	3	14	43	54	3
x_7	89	7	13	42	2	2
x_2	87	2	12	44	64	2
x_3	24	8	15	43	36	3
x_3	24	8	15	43	36	3
x_8	68	3	14	43	54	3

`max_features` controls how large the subset is (= 3 here)

`max_features = n_features`

no randomness is injected

`max_features = 1`

force it to use a certain feature

RANDOMIZATION II: FEATURE SELECTION

Dataset for tree 1

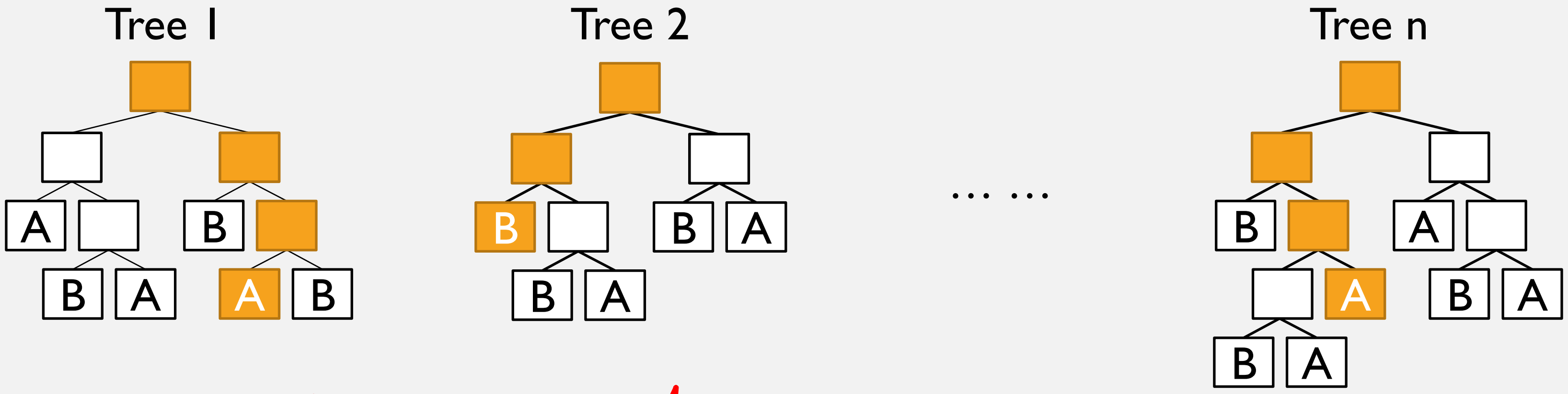
	f_1	f_2	f_3	f_4	f_5	f_6
x_7	89	7	13	42	2	2
x_9	35	6	11	41	63	2
x_4	67	7	17	44	87	2
x_8	68	3	14	43	54	3
x_7	89	7	13	42	2	2
x_2	87	2	12	44	64	2
x_3	24	8	15	43	36	3
x_3	24	8	15	43	36	3
x_8	68	3	14	43	54	3

A low value of `max_features` yields very different but deep trees

A high value of `max_features` yields very similar but shallow trees

A rule of thumb $\sqrt{n\_features}$

PREDICTIONS USING RANDOM FORESTS



- Each tree makes a prediction

"Soft voting": Average the probabilities of belonging to a certain class
→ predict highest-probability one

PROS AND CONS OF RANDOM FORESTS

Pros

Very powerful

Requires little parameter tuning

Share many benefits of trees
- make up for many deficiencies

Cons

Slow

Different random states ⇒ different results

Difficult to visualize and interpret

TREES VS. FORESTS

Jupyter Notebook **Decision Trees 2: Feature importance and ensembles of trees**

GRADIENT BOOSTED DECISION TREES

OR GRADIENT BOOSTED REGRESSION TREES *OR* GRADIENT BOOSTING MACHINES

- Build a sequence of trees where each tree tries to correct the mistakes of the previous one
- Use shallow trees ("weak learners")

HYPERPARAMETERS

n_estimators

how many trees to train

max_depth

of each tree

learning_rate

how strongly each tree depends on previous one

CODING BOOSTED TREES

Jupyter Notebook **Decision Trees 2: Feature importance and ensembles of trees**

PROS AND CONS OF GRADIENT BOOSTED DECISION TREES

Pros

One of the most powerful ML models

Cons

Requires careful parameter tuning

WHEN TO USE WHAT

Tree	Forest	Boosted tree
When VISUALIZATION	ROBUSTNESS	ACCURACY
		is important
(fast)	(slowest)	(slower)


WHERE DOES A DATA
SCIENTIST CAMP?
IN A RANDOM FOREST