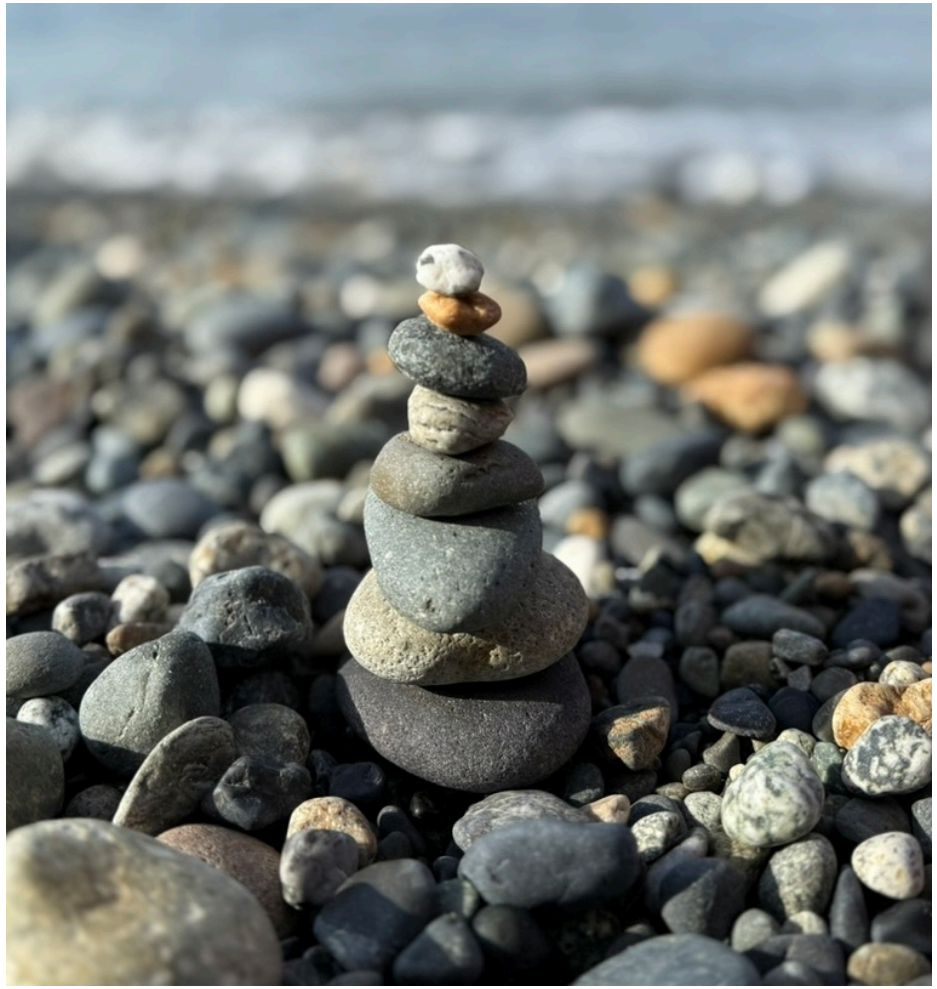
Lecture 2: From Data to a First Model

Terminology, Baselines, Decision Trees

Varada Kolhatkar

Focus on the breath!



Announcements

- **Homework 1:** due September 14, 11:59 pm
- **Homework 2:** released; due September 21, 11:59 pm
- Watch the pre-lecture videos and check pinned posts on Ed.
- Course deadlines
- iClicker Cloud link: https://join.iclicker.com/GHJB
- My OH: Tuesdays and Thursdays from 12:30 to 1 PM.

Today's big questions

- How do we formulate a prediction problem as a **supervised machine learning problem**?
- How do we **train, predict, and evaluate** models using `scikit-learn`?
- How does a **decision tree learn rules and make predictions**?

Formulate a prediction problem

Which job might make you happy?

You've graduated and have several job offers. **Which one would make you happiest?**

You decide to look for patterns in the experiences of friends who value similar things in a job.

Image created by GPT-6 Astra

What do we need to know?

Before thinking about machine learning, let's define the prediction problem.

- **Problem**: Help someone choose a job they'll be happy in.
- **What things about a job might affect how happy you are with it?**
- **What exactly should we predict? How could we represent it?**

Toy job happiness dataset

Supportive colleagues	Salary ($)	Free coffee	Boss vegan	Happy?
0	70,000	0	1	**Unhappy**
1	60,000	0	0	**Unhappy**
1	80,000	1	0	**Happy**
1	110,000	0	1	**Happy**
1	120,000	1	0	**Happy**
1	150,000	1	1	**Happy**
0	150,000	1	0	**Unhappy**

Each row describes one person in a job. **1 = yes; 0 = no.**

Recap: the goal of supervised learning

Use examples with known **features** X and **targets** y to learn a mapping function f.

$$\underbrace{X_{\text{new}}}_{\text{features}} \xrightarrow{\text{learned model } f} \underbrace{\hat{y}_{\text{new}}}_{\text{predicted targets}}$$

The goal: predict targets accurately for **new examples**.

Your turn: identify X and y

Supportive colleagues	Salary ($)	Free coffee	Boss vegan	Happy?
0	70,000	0	1	Unhappy
1	60,000	0	0	Unhappy
1	80,000	1	0	Happy

First three examples: **which columns are inputs, and which is the target?**

Features X: the four job attributes

Target y: `Happy?`

Example: one row

$n = 7$ examples and $d = 4$ features

Training vs. Prediction

- Training: learn the mapping
  - Known features X + known targets $y \rightarrow$ fitted model f
- Prediction: use the mapping
  - New features $X_{\text{new}} \rightarrow$ fitted model $f \rightarrow$ predictions $\hat{y}_{\text{new}}$

We need known targets to train the model, not to make predictions.

Same job features, different targets

Supportive colleagues	Salary ($)	Free coffee	Boss vegan	Happy?	happiness_score
0	70,000	0	1	**Unhappy**	35
1	60,000	0	0	**Unhappy**	48
1	80,000	1	0	**Happy**	85

- **Happy or unhappy** $\longrightarrow$ predict a discrete label $\longrightarrow$ **classification**
- **Happiness score from 0 to 100** $\longrightarrow$ predict a numerical quantity $\longrightarrow$ **regression**

Other regression examples: predicting temperature, house price, or travel time.

iClicker 2.1: Classification or regression?

Link: https://join.iclicker.com/GHJB

Select all regression problems.

- **A.** Predict whether a flight will be delayed.
- **B.** Predict a flight's delay in minutes
- **C.** Predict a student's percentage grade in CPSC 330.
- **D.** Predict whether you'll get a seat on the bus.
- **E.** Predict whether you'll hit snooze tomorrow morning.

Train, predict, and evaluate models using `scikit-learn`

`scikit-learn`

- In this part of the course we use a framework called `scikit-learn`.
- A popular framework for tabular data
- Easy to use.
- 67.3K stars on Github

Separate the inputs from the target

```
1 X = toy_happiness_df.drop(columns=['happy?'])
2 y = toy_happiness_df['happy?']
```

Object	Contains	Shape
`X`	Four feature values for each person	`(7, 4)`
`y`	One known happiness label per person	`(7,)`

Keep the answer out of the inputs.

Your turn: predict without the features

If you must predict **the same label for everyone**, which label should you choose? How many predictions will be correct?

A baseline gets four out of seven right

Actual	Baseline prediction	Correct?
Unhappy	Happy	No
Unhappy	Happy	No
Happy	Happy	Yes
Happy	Happy	Yes
Happy	Happy	Yes
Happy	Happy	Yes
Unhappy	Happy	No

Always predict Happy: accuracy $= 4/7 \approx 57\%$

Classification error: $3/7 \approx 43\% = 1 - \text{accuracy}$

Fit a baseline with scikit-learn

```
1 from sklearn.dummy import DummyClassifier
2
3 dummy = DummyClassifier(strategy='most_frequent')
4 dummy.fit(X, y);
```

`fit(X, y)` learns from the training data.

Here, it finds the most frequent target. **This baseline ignores the feature values.**

Predict, then check accuracy

```
1  dummy.predict(X)
```

```
array(['Happy', 'Happy', 'Happy', 'Happy', 'Happy', 'Happy', 'Happy'],
      dtype='<U5')
```

```
1  print(f'Training accuracy: {dummy.score(X, y):.1%}')
```

```
Training accuracy: 57.1%
```

`predict(X)` returns predicted labels.

`score(X, y)` compares predictions with known targets.

Method	You provide	It does
`fit(X, y)`	Features and known targets	Learns the model
`predict(X_new)`	Features	Returns predicted targets
`score(X, y)`	Features and known targets	Returns a default evaluation measure

Why doesn't `predict` require `y`?

We don't know the answer yet!

Decision trees

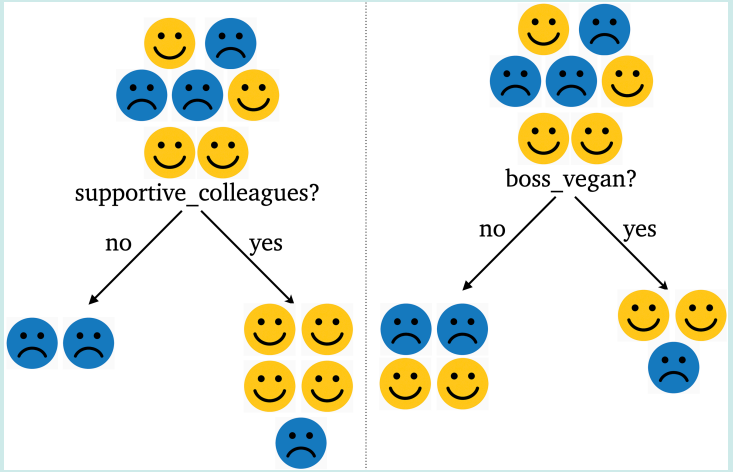
Can we do better than the baseline?

Supportive colleagues	Salary ($)	Free coffee	Boss vegan	Happy?
0	70,000	0	1	Unhappy
1	60,000	0	0	Unhappy
1	80,000	1	0	Happy
1	110,000	0	1	Happy
1	120,000	1	0	Happy
1	150,000	1	1	Happy
0	150,000	1	0	Unhappy

Propose one yes/no question that separates Happy from Unhappy.

Which question is more effective?

Visual | Data



Question	No group	Yes group	Comments
Supportive colleagues?	0 😊, 2 ☹️	4 😊, 1 ☹️	Cleaner split
Boss vegan?	2 😊, 2 ☹️	2 😊, 1 ☹️	Mixed labels

Supportive colleagues gives a cleaner split.

What are we trying to learn?

- We want to learn which questions to ask and in what order.
- How many different combinations of feature-based questions could a decision tree potentially ask?

Supportive colleagues	Salary ($)	Free coffee	Boss vegan	Happy?
0	70,000	0	1	Unhappy
1	60,000	0	0	Unhappy
1	80,000	1	0	Happy
1	110,000	0	1	Happy
1	120,000	1	0	Happy
1	150,000	1	1	Happy
0	150,000	1	0	Unhappy

Decision tree Training (high level)

- Training a decision tree is a search process: we look for the “best” tree among many possible ones.
- There are different algorithms for learning trees. Check this out.
- At each step, we evaluate candidate questions using measures such as:
  - Information gain
  - Gini index
- The goal is to split the data into groups with greater certainty (more homogeneous outcomes).

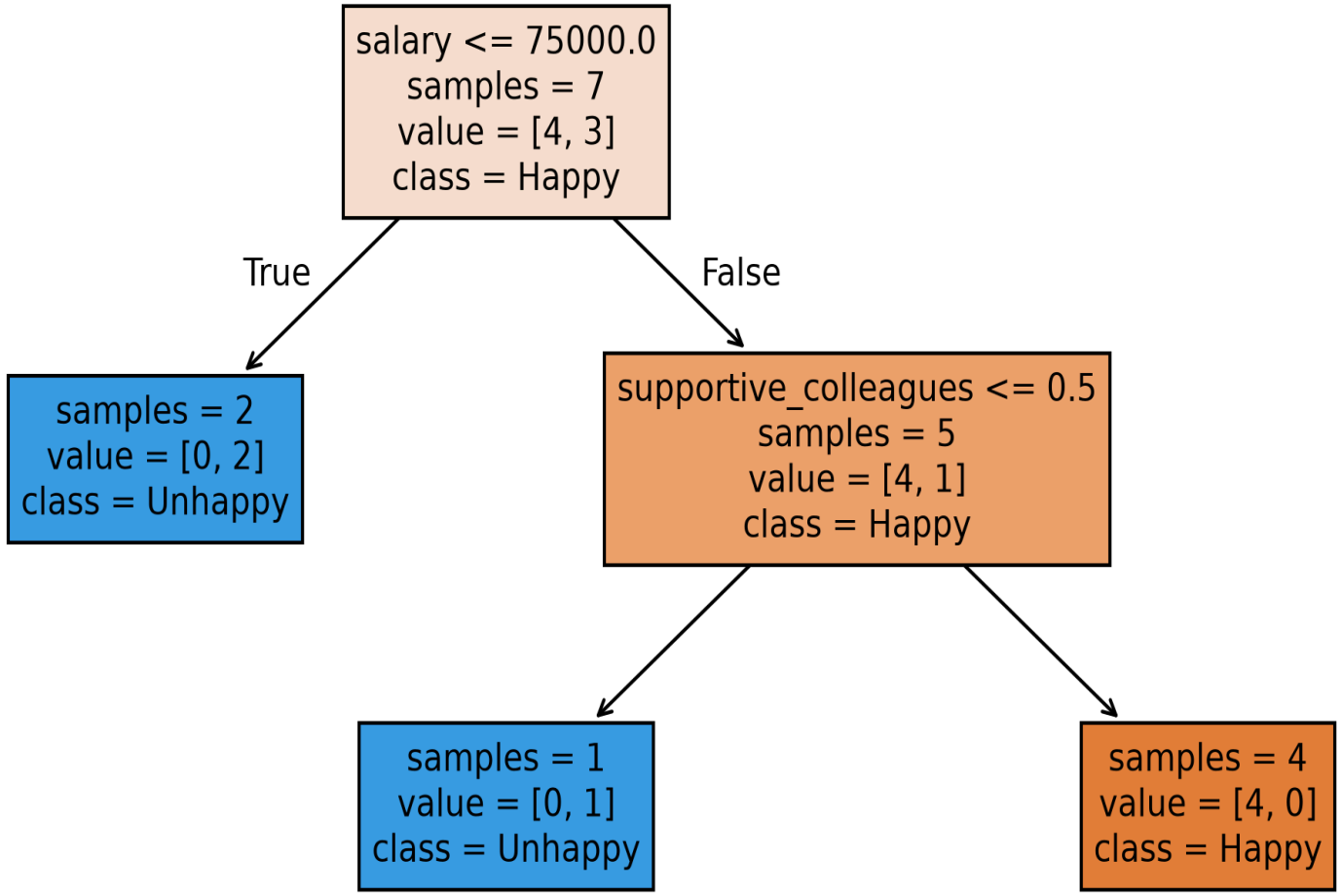
Fit a decision tree

```
1 from sklearn.tree import DecisionTreeClassifier
2
3 model = DecisionTreeClassifier(max_depth=2, random_state=1)
4 model.fit(X, y);
```

Same `fit` interface. A different learning algorithm.

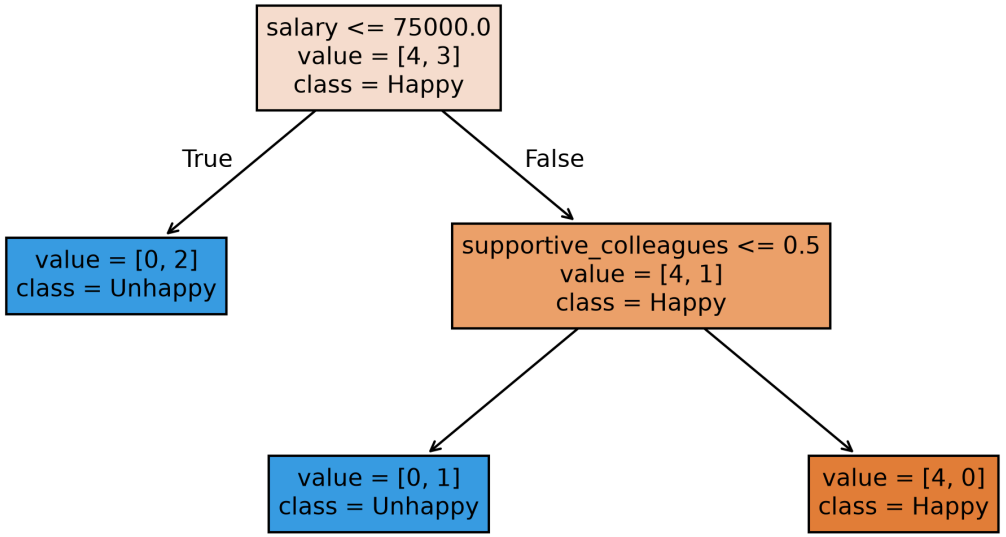
`max_depth=2` allows at most two questions along a prediction path.

Read the learned tree



Prediction

What would be the prediction for the example below using this tree ?



- supportive_colleagues = 0
- salary = 100,000
- coffee_machine = 0
- vegan_boss = 1

Prediction with `sklearn`

- supportive_colleagues = 0
- salary = 100,000
- coffee_machine = 0
- vegan_boss = 1

```
1 test_example = [[0, 100000, 0, 1]]
2 print("Model prediction: ", model.predict(test_example))
```

```
Model prediction:  ['Unhappy']
```

Does the tree beat the baseline?

```
1 model.score(X, y)
```

```
1.0
```

Model	Correct / total	Training accuracy
Always Happy	4 / 7	57%
Decision tree (max_depth=2)	7 / 7	100%

Yes, on the training data.

We still don’t know how well it predicts happiness for new people.

iClicker Check-in

Link: https://join.iclicker.com/GHJB

How are you feeling about the workflow and decision trees?

- **A.** I can explain both to a neighbour.
- **B.** I understand the ideas but need to review the code.
- **C.** I understand the code but need to review how decision trees work.
- **D.** I'd benefit from a pause and another example.

Parameters vs. Hyperparameters

Before `fit`: hyperparameters	**During `fit`: parameters**
We set `max_depth=2`.	The tree learns to split on `salary` at **75,000**.
This limits the allowed tree depth.	It learns another split on `supportive_colleagues` at **0.5**.

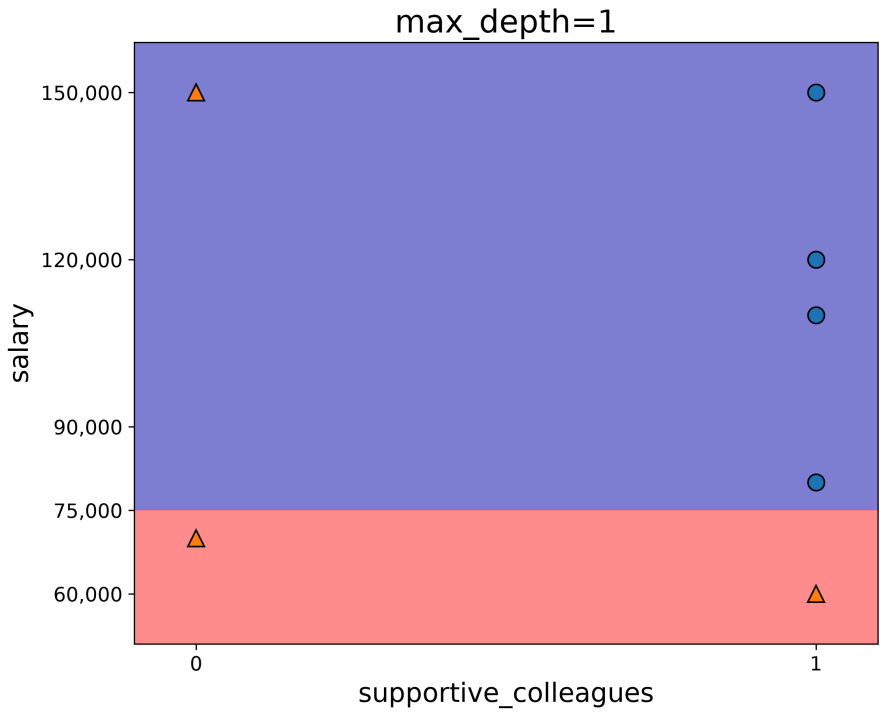
Tree depth: the number of edges on the longest root-to-leaf path.

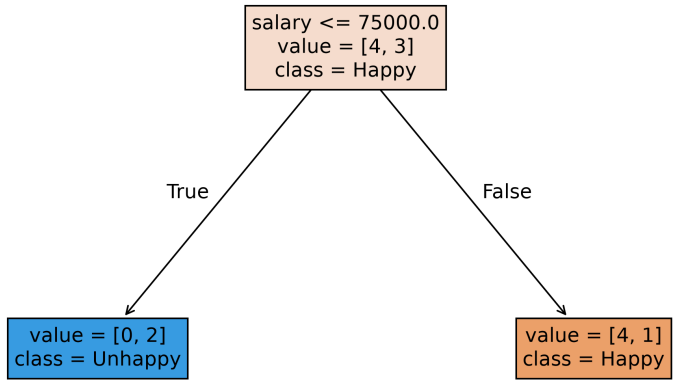
Decision boundary

- A **decision boundary** is the line, curve, or surface that separates classes.
- Points on one side $\longrightarrow$ Model predicts Class Happy
- Points on the other side $\longrightarrow$ Model predicts Class Unhappy

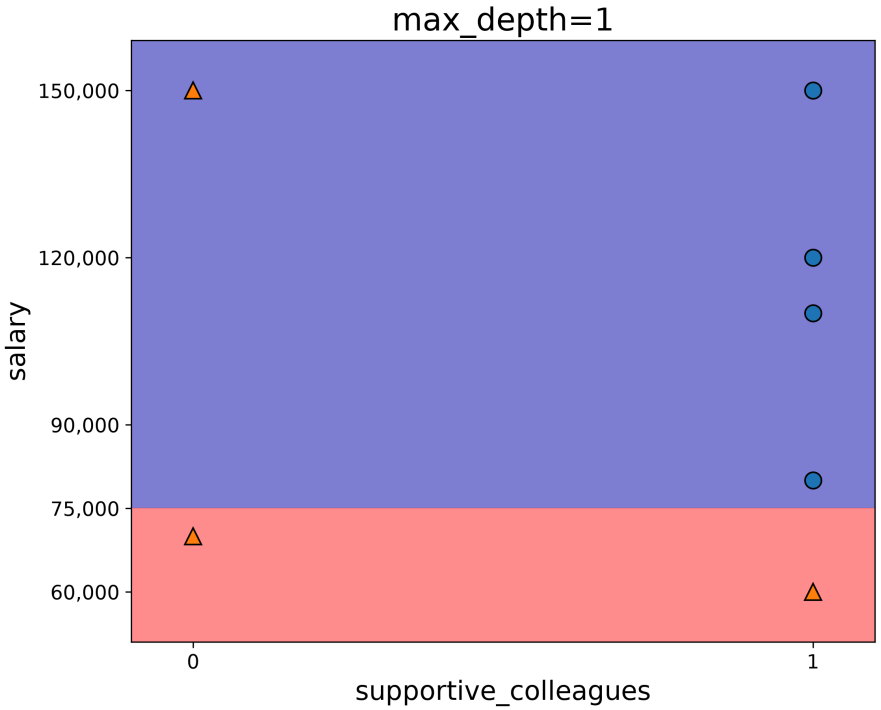
Decision boundary with `max_depth=1`

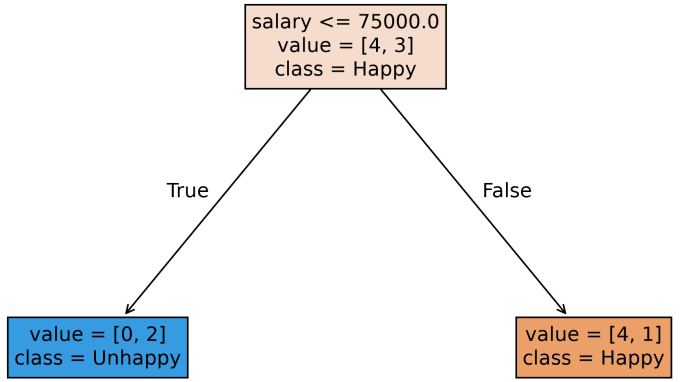
- **Points:** observed people and their actual labels
- **Shaded regions:** the model's predicted class
- **Decision boundary:** where the predicted class changes





Decision boundary with `max_depth=1`



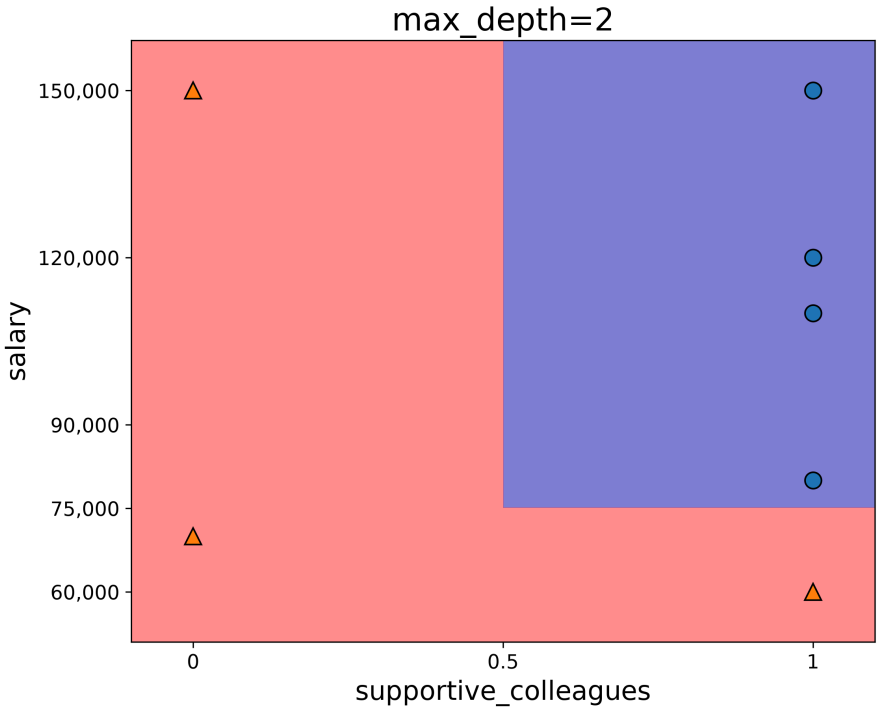


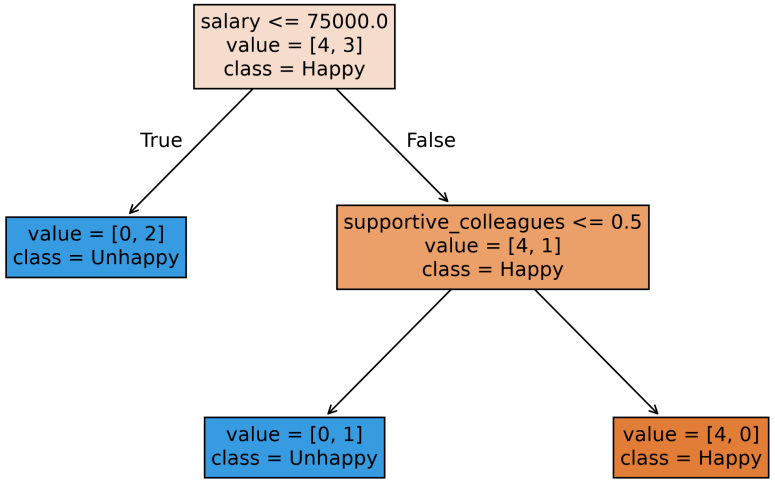
A depth-one tree is called a **decision stump**. A shallow tree can stop too soon. Which person does this tree get wrong?

The person earning **$150,000 without supportive colleagues**.

Decision boundary with `max_depth=2`

Contrast it with the depth-two tree, which can ask another question within the higher-salary group.





iClicker 2.3: Baselines and trees

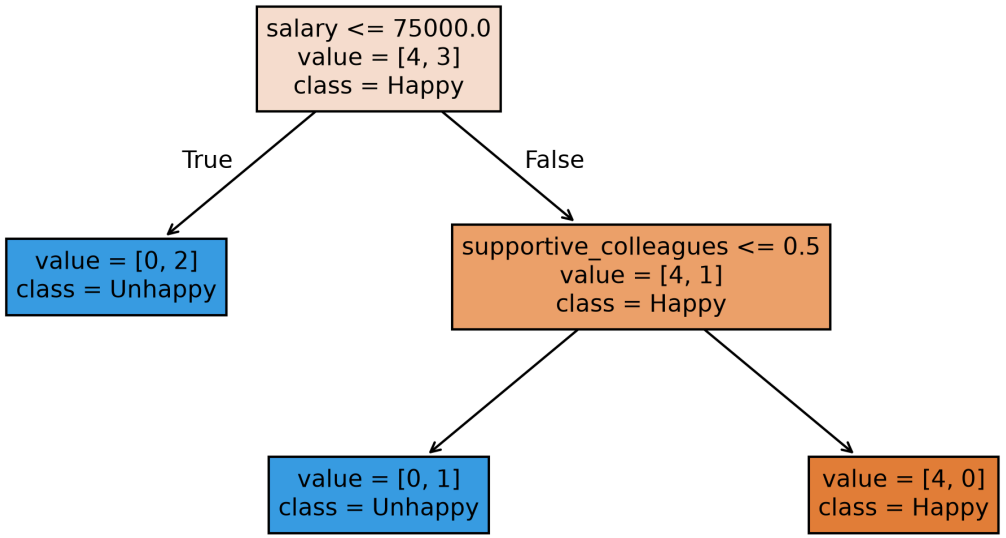
Link: https://join.iclicker.com/GHJB

Select all true statements.

- **A.** Changing salary values must change a most-frequent baseline's predictions.
- **B.** `predict` needs features; `score` also needs known targets.
- **C.** Decision-tree features must be binary.
- **D.** A tree predicts by following a path from root to leaf.

Exit ticket

A new job offers **$100,000**, **supportive colleagues = 1**, and **free coffee**.



1. What path and prediction does it get?
2. Name one learned parameter and one hyperparameter.
3. Does 100% training accuracy make this prediction reliable?

Next: will it work on new examples?

We can now represent a problem with X **and** y, compare against a **baseline**, and **fit and read a decision tree**.

A model can get every training example right.
What would convince us it has learned something useful?

Lecture 3: evaluating on unseen data, overfitting, and model complexity.

More details on this material: Chapter 2: From Data to a First Model