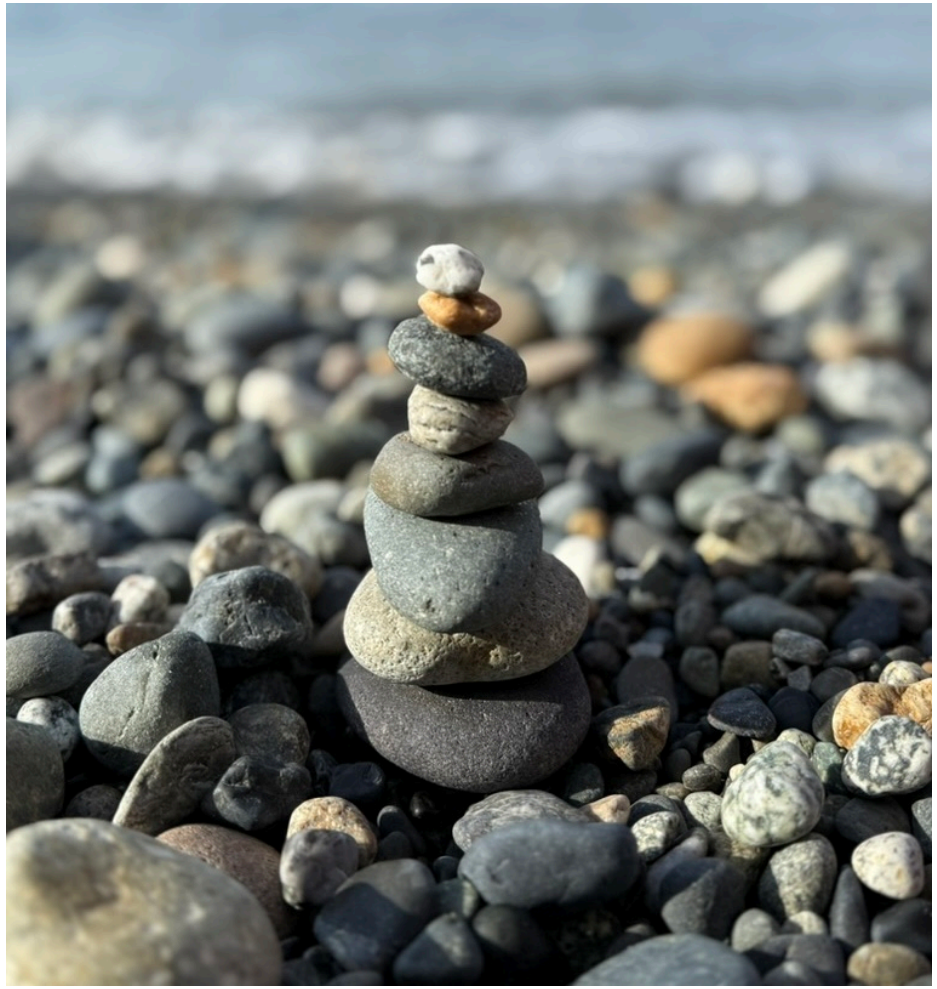
Lecture 3: Will It Work on New Data?

Generalization, Data Splitting, Model Complexity

Varada Kolhatkar

Focus on the breath!



Announcements

- **Homework 2:** released; due September 21, 11:59 pm
  - You may broadly discuss it with classmates; final answers and submissions must be your own.
  - Group submissions are not allowed for this assignment.
- Practice, and start early on homework assignments.
- iClicker Cloud link: https://join.iclicker.com/GHJB

iClicker recap: what does a tree learn?

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

We create `DecisionTreeClassifier(max_depth=2)`, then call `fit(X, y)`.

Which statement is correct?

- **A.** The model learns `max_depth=2` from the data.
- **B.** We set the depth limit; the model learns splits and leaf predictions.
- **C.** We must supply every split before calling `fit`.
- **D.** The fitted tree must have depth exactly 2.

Recap: fit, predict, score

Method	What it does
`fit(X, y)`	Learns from features and known targets
`predict(X)`	Returns predicted targets
`score(X, y)`	Evaluates predictions against known targets

Last lecture, we fitted and scored on **the same examples**.

A perfect training score?

A tree gets every training example right.

Would you trust it to predict happiness for a new person? Why?

We need evidence about **new examples**.

Generalization: how well a model performs on unseen data from the population we want to predict for.

Today's big questions

- How can we estimate performance on **unseen data**?
- How do we choose **model complexity** without overfitting?
- How do we keep our final evaluation **independent of model selection**?

We'll investigate these questions in a **live housing-price demo**.

Class demo

Why split the data?

A model has already used the training targets to learn.

Scoring those same examples does not tell us how well it predicts unseen examples.

Reserve data to evaluate predictions on examples **not used in that fit**.

Exam analogy

Imagine your midterm is a few weeks away and you have a set of **practice questions with solutions**.

For this analogy, suppose all questions are multiple choice.

If you can answer every practice question correctly, are you ready for the exam?

Training: learn from practice

Image created by GPT-6 Astra

Use practice questions **and their solutions** to learn.

- **Features X:** information in each question.
- **Targets y:** the correct answers.
- **Training:** learn patterns from these examples.

A training score measures performance on questions you have already studied.

Validation: choose how to study

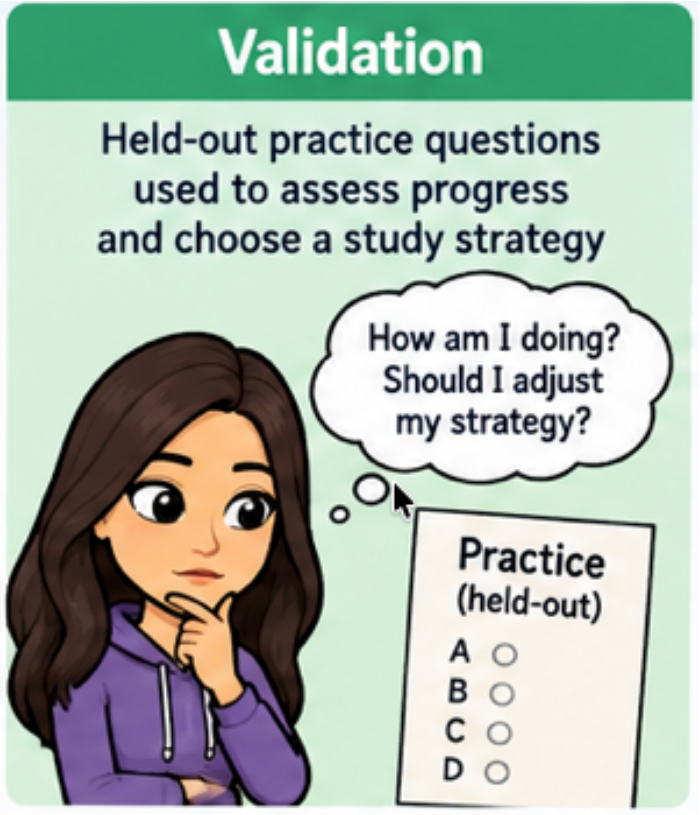


Image created by GPT-6 Astra

Hold out some practice questions from your study material.

Try them to check whether your preparation transfers to **unseen questions**.

Use the results to choose a study strategy.

In ML: compare models and choose hyperparameters such as `max_depth`.

Test: the actual exam

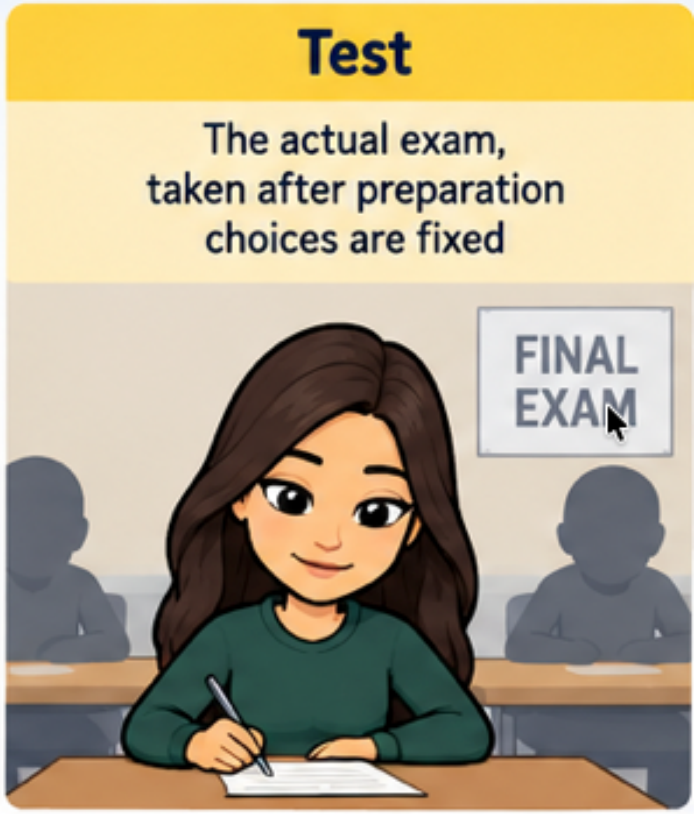


Image created by GPT-6 Astra

Take the actual exam **after your preparation choices are fixed**.

In ML: evaluate the selected model on the held-out test set.

If you use the exam answers to change how you prepare, it is no longer an independent assessment.

Deployment: new interview questions

Image created by GPT-6 Astra

Now apply what you learned in an interview.

In ML: use the fitted model to predict targets for new examples.

- Correct answers may be unavailable or arrive later.
- Interview questions may differ from the exam.

Would a high exam score guarantee success?

Training, validation, test, and deployment

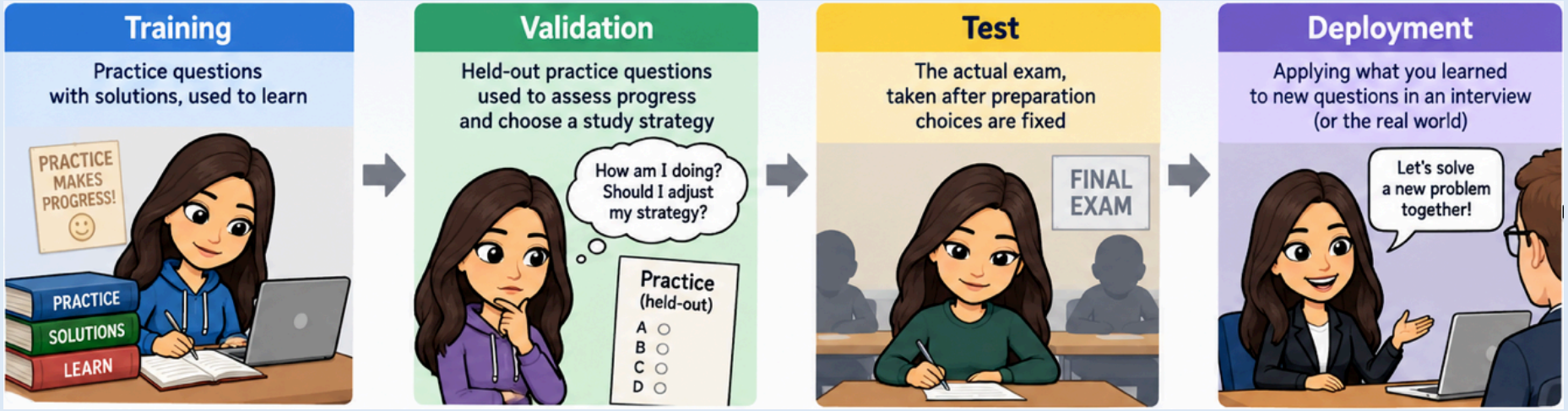


Image created by GPT-6 Astra

Train to learn. Validate to choose. Test to assess. Deploy to predict.

Validation results influence our choices. **Test results must not.**

Train, validation, test, and deployment data in `sklearn`

	`fit`	`score`	`predict`
Train	✔️	✔️	✔️
Validation		✔️	✔️
Test		Final evaluation	Final evaluation
Deployment			✔️

Keep test results out of modeling choices; the number of method calls is not the issue.

The golden rule

Test data must not influence training or model selection.

Knowing the actual exam answers before studying would undermine the assessment.

- Set it aside before exploring features or choosing models.
- Use training and validation data to develop the model.
- Evaluate on the test set after your choices are fixed.

iClicker 3.1: Data splitting

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

Select all true statements.

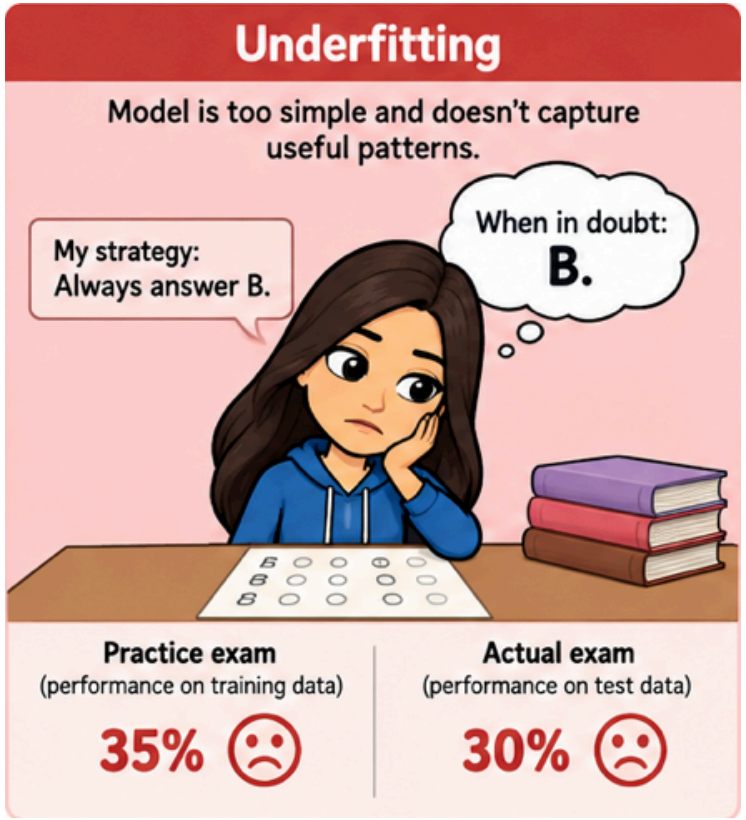
- **A.** An unrestricted decision tree is likely to score very well on its training data.
- **B.** Held-out data help us assess generalization.
- **C.** Deployment data always come with known targets for scoring.
- **D.** Validation data can be used to choose `max_depth`.
- **E.** Randomly shuffling before splitting is appropriate for every prediction task.

Back to the demo: split and compare

- Split the data and fit a fresh decision tree on the training subset.
- Compare training and held-out scores with a dummy baseline.
- Try a stump and a deeper tree. Predict the scores before running.

Which dataset is influencing our choices?

Underfitting: an overly simple strategy



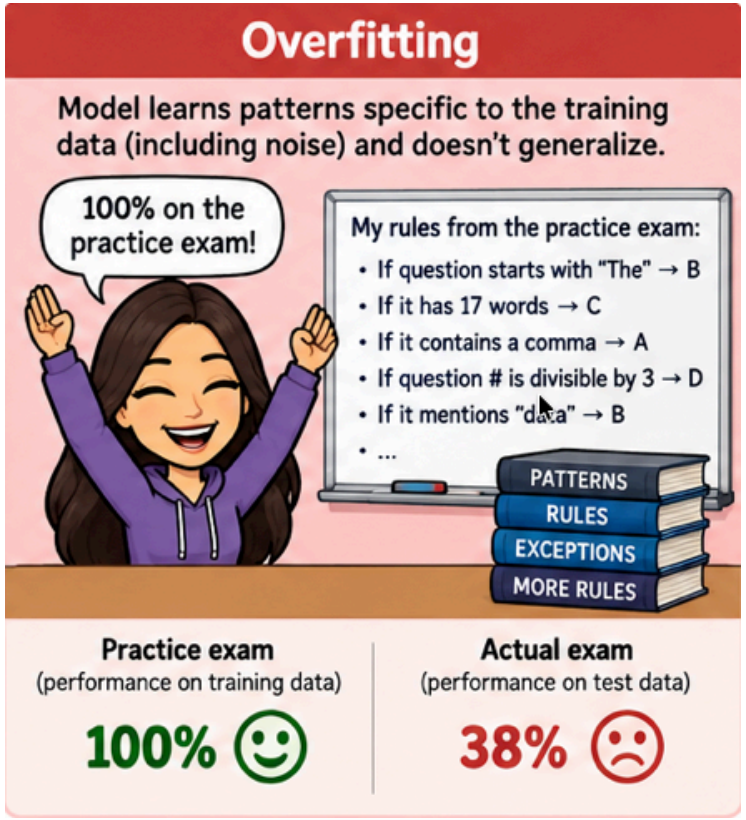
"When in doubt, always answer B."

This rule misses useful patterns **even in the practice questions**.

In ML: the model is too simple; training and validation performance are both poor.

Demo connection: could a decision stump capture the housing-price relationships?

Overfitting: memorize the practice details



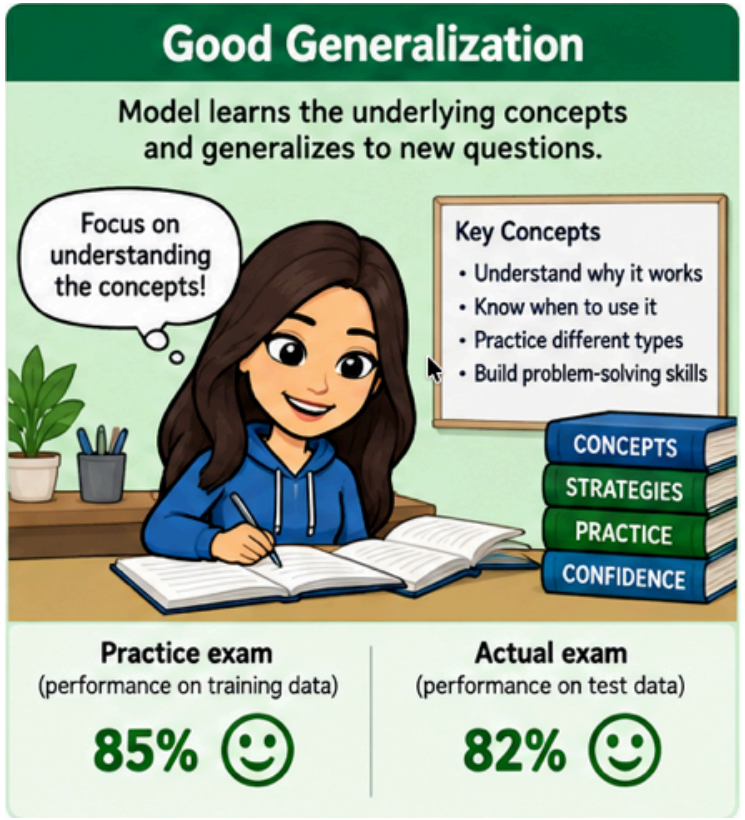
"If the question has 17 words, choose C."

The rules fit the practice questions, but do not transfer to new ones.

In ML: the model fits training-specific details, including noise.

Demo connection: a deep tree scores highly on training data but much worse on validation data.

Good generalization: transfer to new questions



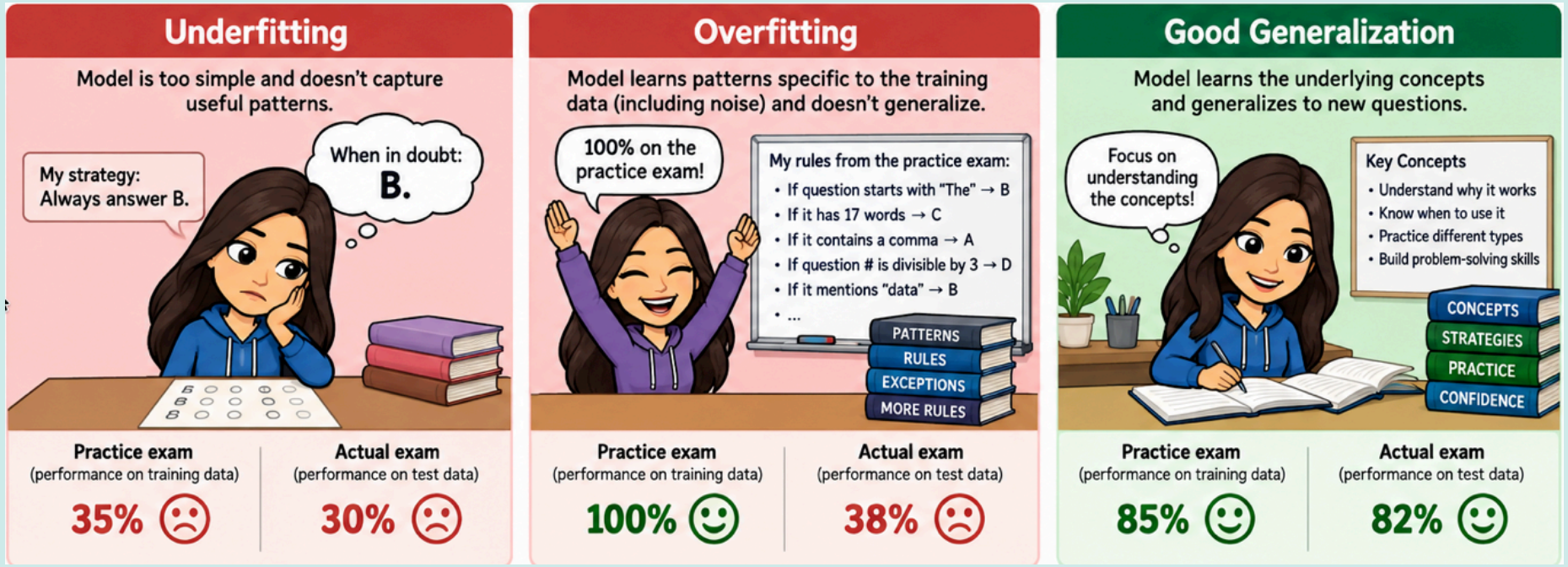
Learn patterns that help solve **unfamiliar questions**.

You do not need a perfect practice score to do well on the exam.

In ML: seek strong performance on representative unseen data.

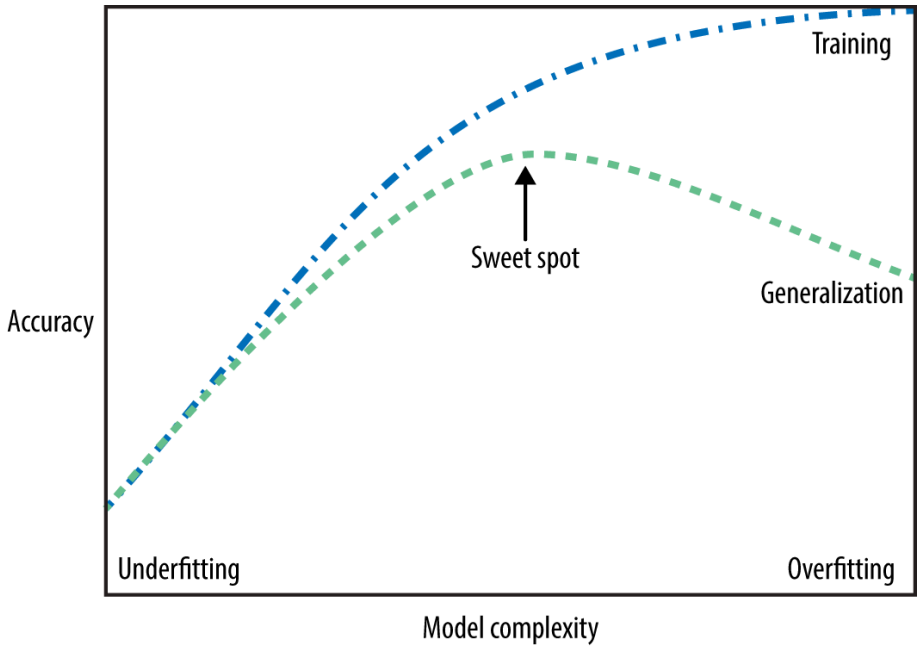
Demo connection: choose depth using validation evidence, not the highest training score.

Compare the three patterns



Which student needs a richer strategy? Which needs a strategy that transfers?

The fundamental tradeoff



- Too simple: misses useful patterns
- More complexity: both scores may improve
- Too complex: training improves while validation may worsen
- Choose complexity using **validation scores**

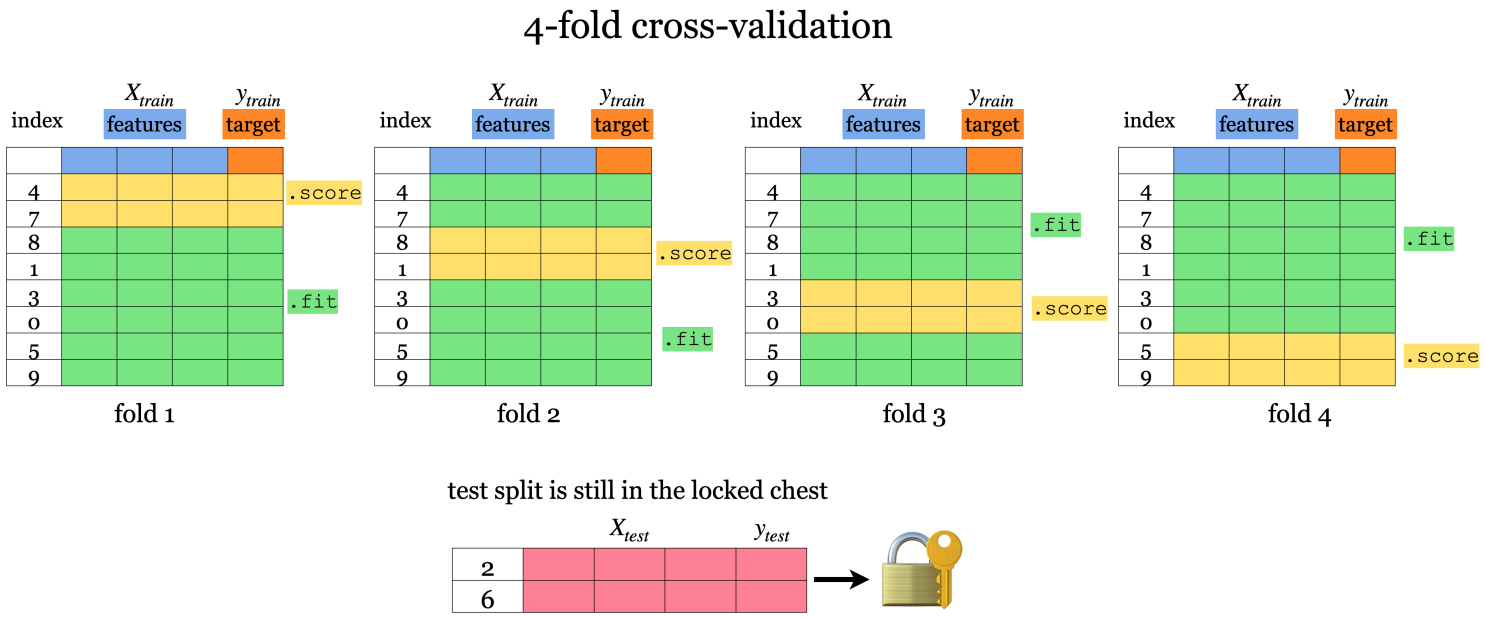
Back to the demo: choose a depth

Create a **validation subset** from the training data.

- Loop over `max_depth` values.
- Plot training and validation scores.
- Choose a depth using validation performance.

Would we choose the same depth with a different validation split?

Cross-validation



Fit a fresh model for each fold. Keep the final test set aside.

Why cross-validation?

- Depend less on one particular training–validation split.
- Use every non-test example for training and validation, **in different fits**.
- Examine variation across folds alongside the mean score.

Cost: more model fits. **Limit:** it does not replace the final test set.

Back to the demo: compare depths with CV

Use five-fold CV on **all non-test training data**.

- Compare mean validation scores and their variation across folds.
- Choose a depth and explain your reasoning.

Does `cross_validate`'s `test_score` mean our final test score?

No. Here it is the score on each **validation fold**.

iClicker 3.2: Complexity and CV

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

Select all true statements.

- **A.** k-fold CV calls `fit` k times for one hyperparameter configuration.
- **B.** CV makes evaluation less dependent on one validation split.
- **C.** A much higher mean training score than mean validation score can indicate overfitting.
- **D.** Whenever training score increases, validation score must decrease.
- **E.** A decision stump on a complex prediction problem can underfit.

Back to the demo: refit and evaluate

1. Fix the selected depth using CV results.
2. Fit a fresh model on **all non-test training data**.
3. Evaluate it on the test set.

In this demo, we used test results earlier to compare models. This score is therefore not an independent final assessment.

The workflow to take away

1. **Split:** reserve test data before exploring or fitting.
2. **Explore:** perform EDA on training data.
3. **Compare:** establish a baseline and select hyperparameters using CV on training data.
4. **Refit:** train the selected configuration on all training data.
5. **Evaluate:** assess it on the untouched test set.

Keep test results out of model selection.

Exit ticket

A classmate proposes choosing the tree with the **highest training score**.

1. Use today's demo to explain the problem with this proposal.
2. How would you choose a depth instead?
3. Which data would you use for the final fit and evaluation?

Recap: Chapter 3

- Why do we split the data? What are train/validation/test splits?
- What are the benefits of cross-validation?
- What are underfitting and overfitting?
- What is the fundamental trade-off in supervised learning?
- What is the golden rule of machine learning?

Chapter 3: ML fundamentals