Error Diagnosis

ECBS5200 — Week 4

You made a recommendation. How do you know you were right?

Where we left off

Model	Accuracy	Macro F1
Encoder LoRA (ModernBERT 149M)	56.6%	0.209
Decoder LoRA (Qwen 0.5B, 494M)	57.0%	0.240

Decoder wins by **0.031 macro F1**. ~3× slower per example.

The black box you've been calling

```
from utils.data_utils import load_course_data
train_ds, val_ds, test_ds = load_course_data()
```

Four weeks of using this. **What does it do to the raw CFPB data?**

Before you diagnose errors today, see the pipeline that made your dataset.

Raw: 153 Issue labels, many redundant

Label	Raw count
Incorrect information on credit report	7,607
Incorrect information on **your** report	7,208

CFPB **reworded the complaint form in April 2017.** Historical complaints kept old labels.

`data/label_merge_mapping.json` catches 36 pairs. **153 → 120.**

The mapping is imperfect in both directions

No-op we left in:

`"Can't repay my loan" → "Can't repay my loan"`

Duplicates we missed — 5 "Advertising" classes survive to the canonical 113:

Class	Train
Advertising	7
Advertising and marketing	142
Advertising and marketing, including promotional offers	98
Advertising, marketing or disclosures	7
Confusing or misleading advertising or marketing	18

Over-merges we made:

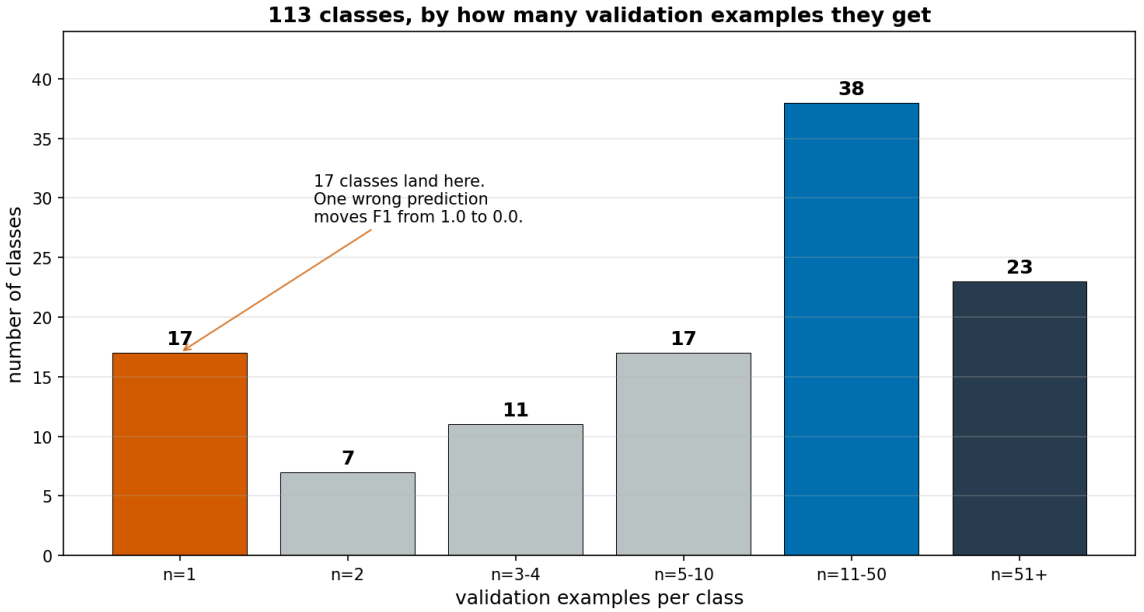
`"Struggling to pay your bill" → "Struggling to pay your loan"`

The filter and the tail

`MIN_CLASS_COUNT=5` : **7 classes dropped** (23 examples lost).

Stratified split, `seed=42` : **57,846 / 6,430 / 21,432.**

17 val classes have 1 example each. One wrong prediction moves that class's F1 from 1.0 to 0.0.



You inherited this. Now diagnose.

Upstream **schema choices, filters, and splits** shape what your model CAN learn.

You did not build this pipeline.
In industry you will not build most of them.

Knowing what was decided for you is part of the job.

The question

> "I made a recommendation. **How do I know I was right?**"

Two models. One bag of diagnostic tools. Which one is *really* better — and how would you know?

D'Amour et al. 2022 call this *underspecification*: an ML pipeline can return predictors with equivalent held-out metrics that behave very differently in deployment.

`readings/week4/damour2022_underspecification.pdf`

From problem to approach

Underspecification is a **problem statement.** What's the **approach?**

D'Amour's answer: **stress tests.** Don't aggregate more metrics — design evaluations targeted at where predictors *differ*.

Your five diagnostic tools this week are all lightweight stress tests:

- **Slice analysis** — stress on input subsets
- **Calibration** — stress on model's confidence claims
- **Confusion patterns** — stress on WHAT gets confused with WHAT
- **Noise floor** — stress against resampling variability
- **Bug hunt** — stress an allegedly-broken model

Each one isolates a failure mode the aggregate number can't see.

Three ways aggregate metrics lie

1. **Averages hide subgroups.** A model that's 80% accurate might be 95% on head classes and 40% on the tail.
2. **Accuracy doesn't measure confidence.** A model can be right 80% of the time while claiming 99%.
3. **Small gaps may not be real.** Does 0.03 macro F1 survive resampling?

Aggregate numbers answer "how well on average." They don't answer "well enough for what I'm doing."

Today's plan

Block 1 — Lecture (after the quiz)

- Aggregate metrics and their limits
- Slice analysis: the where-does-it-fail tool
- Calibration: the can-you-trust-it tool
- Confusion matrices at scale: patterns, not cells
- Noise floor: how much of a gap is real?
- A broken model you'll diagnose in the lab

Block 2 — Lab

- Load both models, six diagnostic sections
- Predict, measure, explain. End with the bug hunt.

Section 1 — Slice Analysis

The where-does-it-fail tool

The problem with averages

Your decoder's macro F1 is 0.240.

- Is it 0.240 on short complaints AND long complaints?
- On complaints with redacted XXXX markers AND clean ones?
- On complaints that start with "I" AND everything else?

An average number doesn't answer any of those.

Oakden-Rayner et al. 2020 call this *hidden stratification*: aggregate accuracy routinely hides >20% performance gaps on unidentified subgroups — with real clinical consequences in their medical-imaging setting.

`readings/week4/oakdenrayner2020_hidden_stratification.pdf`

What is a slice?

A **slice** is a subset of the val set defined by a property of the INPUT.

Examples:

- Complaints under 200 characters
- Complaints with at least one XXXX redaction marker
- Complaints that start with "I"
- Complaints with heavy all-caps usage
- Complaints mentioning specific legal language

For each slice, compute per-slice macro F1 separately. Compare models slice-by-slice, not just on the average.

The six axes you'll test

In the lab, you'll use two axes. In the homework, all six:

Axis	Split
Character length	Quartile buckets (q1 short → q4 long)
Redaction	XXXX marker present or not
Token length	1–30 / 31–80 / 81–127 / truncated at 128
Numeric content	Dollar sign or 4+ digit number
Opener	Starts with "I" or not
All-caps usage	Above or below median rate

These are one-liners in Python. The signal they produce isn't.

Per-slice F1 example (illustrative)

Slice	n	Model A	Model B	Diff
slice X	1,500	0.20	0.26	+0.06
slice Y	1,500	0.22	0.24	+0.02
slice Z	1,500	0.28	0.28	+0.00

Not all slices are the same. **Where the gap disappears is often more interesting than where it's biggest.**

Null-result axes

Sometimes an axis shows no signal — encoder and decoder behave identically across every slice on that axis.

This is **evidence**, not absence of evidence:

- The axis doesn't distinguish the models
- Whatever feature that axis captures, both models handle it the same way
- You learn what ISN'T driving the gap

Report null-result axes. Don't quietly drop them.

Important caveat: these are **exploratory diagnostics, not confirmatory hypothesis tests.** You're characterizing where models differ, not rejecting null hypotheses with multiple-comparison correction. With six axes tested, some apparent "signal" slices might be noise — cross-check with the bootstrap CI.

When you don't know what axis to try

The six axes you'll test are **hand-picked.** In practice you'd want to:

- Test dozens of axes, not six
- Discover axes you didn't think of
- Automate the coherence check

Automated slice discovery is an active area:

- **Chung et al. 2020** (TKDE) — SliceFinder: automated slice search over predicate combinations
- **Eyuboglu et al. 2022** (ICLR) — Domino: embedding-based slice discovery for hidden failure modes
- **Yu et al. 2026** (AAAI) — slice coherence without predefined labels

You'll see the hand-picked version in lab. These are reading-list papers if you want to go further.

`readings/week4/chung2020_slicefinder.pdf`

`readings/week4/yu2026_manifold_slicing.pdf`

Section 2 — Calibration

Can you trust what the model says about itself?

The calibration question

When the model says "I'm 80% confident in this prediction" —

is it right 80% of the time?

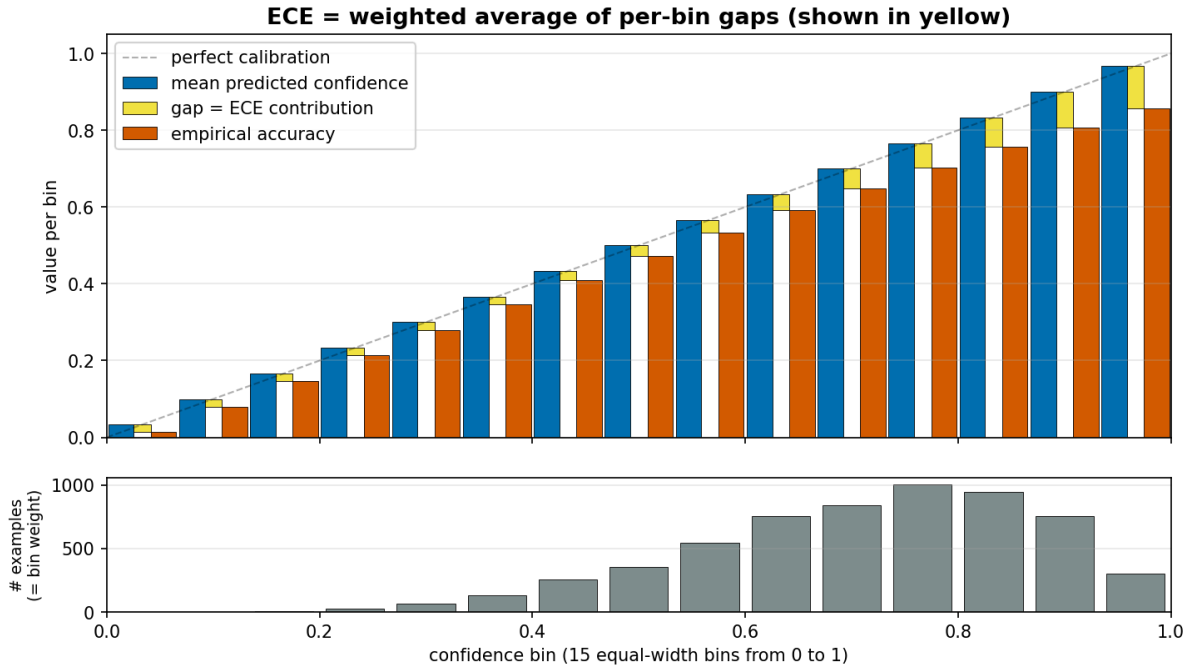
A **calibrated** model's confidence matches its accuracy.

An **overconfident** one says 95%, is right 70%. *(Common for networks fine-tuned with cross-entropy on imbalanced classification — but the effect is measured, not assumed.)*

An **underconfident** one says 60%, is right 85%. *(Rare.)*

ECE: Expected Calibration Error

Pre-work module 5 covered this. Today you measure it. **Guo et al. 2017** popularized ECE as the standard single-number calibration metric.



ECE = weighted average of per-bin gaps. Each bin's gap contributes proportional to its sample count (bottom panel).

ECE works, with caveats

ECE is the standard. It's also imperfect.

Chidambaram et al. 2024 showed ECE is **discontinuous** in predictor space — where you put the 15 bin boundaries can shift ECE without the underlying calibration changing.

They propose **Logit-Smoothed ECE** (LS-ECE) to fix this.

Empirical finding: on pretrained image classifiers, binned ECE tracks LS-ECE closely anyway. The theoretical pathology rarely bites in practice.

Takeaway: teach ECE. Measure ECE. **Don't worship ECE.**

`readings/week4/chidambaram2024_ece_flawed.pdf`

The reliability diagram

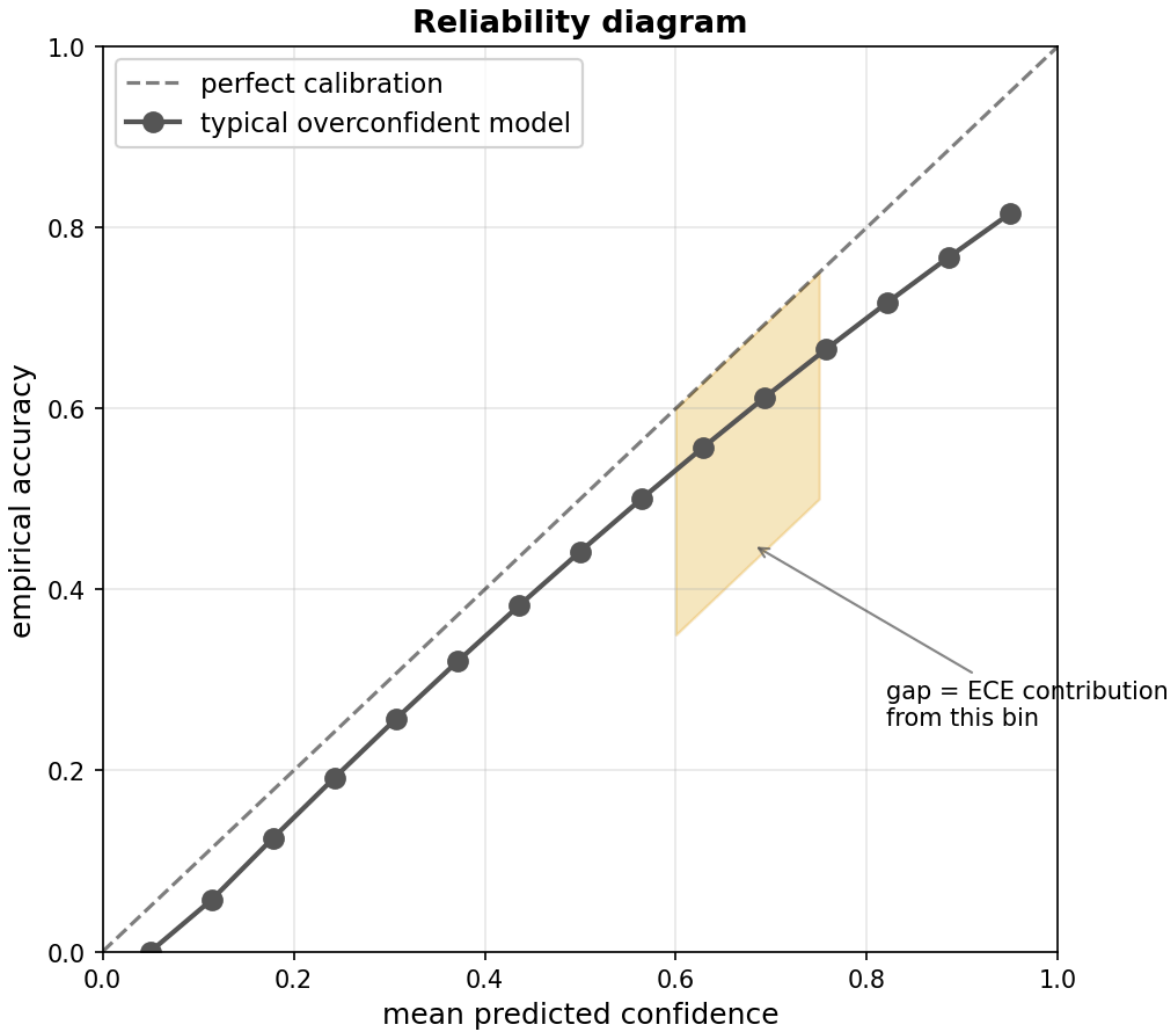
For each confidence bin:

- **X** = mean confidence the model claimed
- **Y** = empirical accuracy on those examples

Perfect calibration = the diagonal.

Below the diagonal = overconfident.
Above the diagonal = underconfident.

In the lab you'll plot this for both models.



An intuition check

You have two models:

- **Encoder** — 149M parameters, ~2T pretraining tokens (English web)
- **Decoder** — 494M parameters, ~18T pretraining tokens (multilingual)

Both fine-tuned with cross-entropy for 3 epochs on the same data.

Before measuring: do you expect their post-fine-tuning ECE to be **similar** or **meaningfully different**? If different, in which **direction**, and by **how much**?

Guo 2017 already showed larger models can be WORSE calibrated than smaller ones — don't treat "bigger = better" as the prior. Fine-tuning dynamics can shift calibration in either direction. Write down your prediction; measure it in the lab.

Temperature scaling — a one-parameter fix

Fit a single scalar T on held-out calibration data. Then at inference:

$$\text{probs}_{\text{scaled}} = \text{softmax}(\text{logits}/T)$$

- T > 1 → softer distribution (fixes overconfidence)
- T < 1 → sharper distribution (fixes underconfidence)
- T = 1 → no change

Critical property *(for post-hoc scalar T applied to fixed logits at inference)*: T is scalar → argmax unchanged → macro F1 unchanged.

This does NOT generalize to all calibration methods. Per-class scaling, Platt scaling, or retraining-based approaches can change argmax.

`readings/week4/guo2017_calibration.pdf`

Homework temperature-scaling exercise

In the homework you'll:

1. Split val 50/50 — **calibration fold** and **eval fold**
2. Fit T on calibration fold (minimize NLL on `softmax(logits / T)`)
3. Apply T to eval fold, compute ECE before and after
4. Also compute macro F1 before and after — should be **identical**
5. Report: does scaling change which model is better-calibrated?

The last question is the one that matters.

Section 3 — Confusion Matrix Patterns

At scale, patterns matter. Individual cells do not.

Pre-work 04: the 5×5 matrix

You drew a 5×5 confusion matrix in pre-work module 4. 25 cells. You could read each one.

Today you have 113×113 = 12,769 cells.

You cannot read 12,769 cells. You have to look for patterns.

Frequency tiers

Group the 113 classes by training frequency into three tiers:

Tier	Count	Training frequency range
Head	20	Top 20 (most common)
Mid	40	Next 40
Tail	53	Bottom 53 (least common)

Then ask questions at the TIER level, not the class level.

Jin et al. 2017 call this the **confusion community** view: at scale, identify groups of classes that systematically confuse each other, then analyze at the group level rather than cell-by-cell.

`readings/week4/jin2017_confusion_graph.pdf`

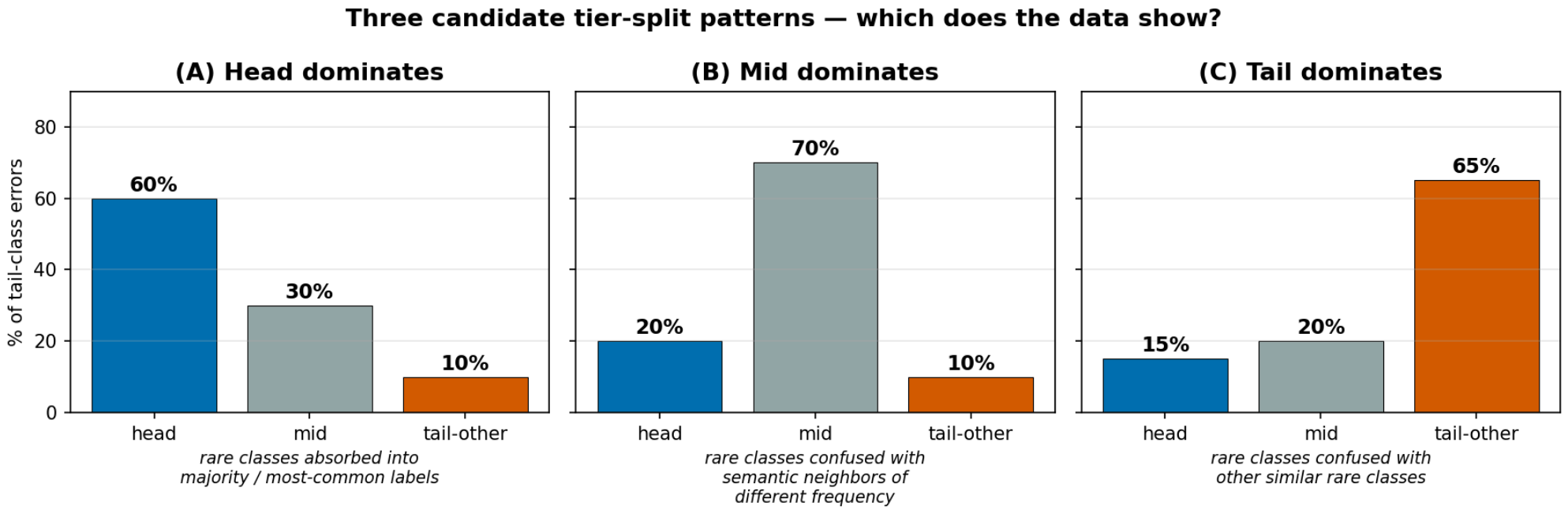
The headline question

When a **tail-class** complaint is misclassified, the model's wrong prediction lands on:

- **(A) A head class** — the model defaults to something common
- **(B) A mid-tier class** — confused with medium-frequency classes
- **(C) Another tail class** — confused with a similar rare class

In the lab, you'll predict the percentage breakdown. Then measure.

Three candidate stories



Each pattern implies a **different fix.** A → class weighting. B → better features for semantic neighbors. C → more data for those specific rare classes.

Same metric, three diagnoses. The data will tell you which.

Section 4 — Val-Set Reliability

Not every F1 number is equally trustworthy.

The problem

113 classes. 6,430 val examples.

If examples were evenly distributed, that'd be ~57 per class.

They aren't. Some classes have 100+ val examples. Some have a handful.

Some have exactly 1.

What a 1-example class does to your F1

Class has 1 val example. Model gets it right → F1 = 1.0. Wrong → F1 = 0.

Nothing in between. A single prediction flips the F1 by 1.0.

Macro F1 is the average of all 113 per-class F1s.

Every 1-example class is a coin flip contributing equally to that average.

Why this matters for model comparison

Your encoder-vs-decoder macro F1 gap is ~0.03.

If 15% of your macro F1 is noisy single-example classes, what fraction of that 0.03 gap could just be noise from those specific classes?

You can't answer that from point estimates. You need **confidence intervals.**

Bootstrap: measure the noise floor

Efron 1979 introduced the bootstrap for exactly this kind of problem — what's the sampling variability of a statistic computed from finite data?

Method: Resample the val set *with replacement* N times. Recompute macro F1 each time.

Now you have a **distribution** of macro F1 values — not a point estimate.

For two models:

- Resample encoder val → distribution of encoder macro F1
- Resample decoder val → distribution of decoder macro F1
- Compute the difference distribution

If the CI for the difference excludes 0, the gap is detectable above the data-side resampling noise you've measured. Model-side noise is a separate question (next slide).

`readings/week4/efron1979_bootstrap.pdf`

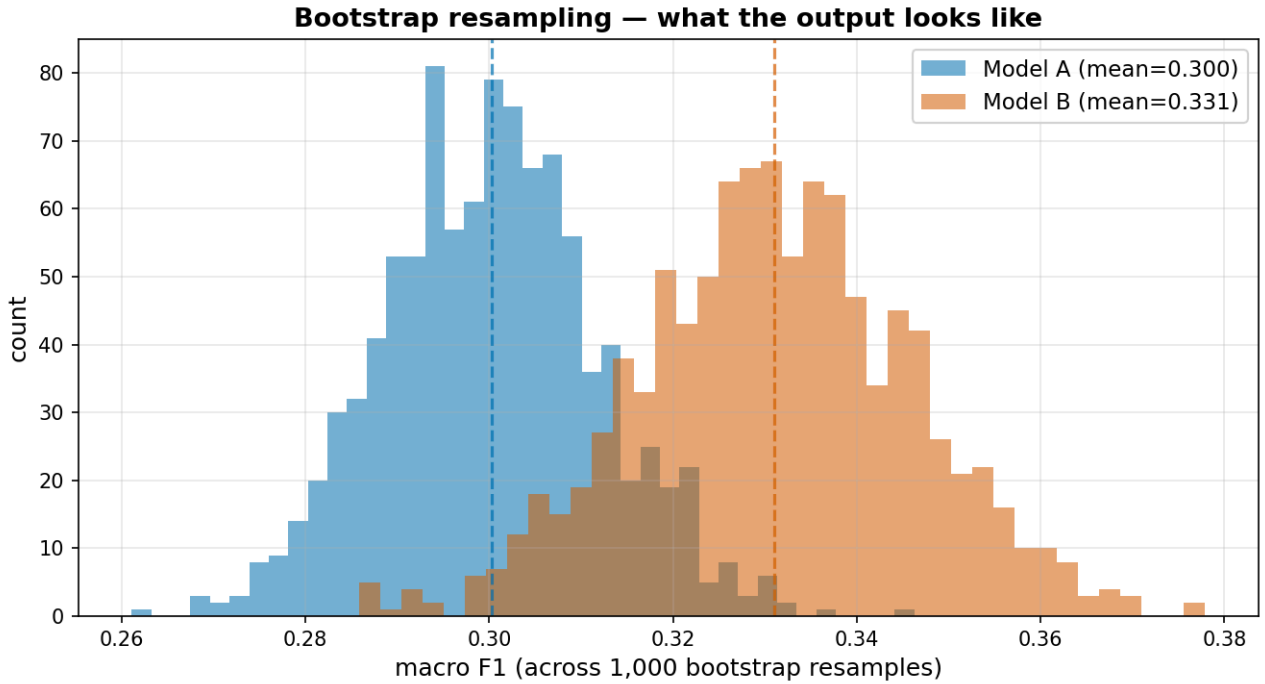
Reading a bootstrap CI

Two overlapping histograms: encoder and decoder macro F1 across 1,000 resamples.

Questions to ask:

- How much do they overlap?
- Where's the 95% CI of the difference?
- Does the difference CI contain 0?

If yes → **you can't tell the two models apart** within noise.
If no → **the gap is real** within noise.



What wide CIs look like in practice

A 95% CI of **[+0.010, +0.048]** says:

- The decoder IS better than the encoder within noise
- BUT the actual advantage is somewhere between tiny and substantial
- **Don't over-interpret the specific point estimate.** It's one draw from this distribution.

In the homework you compute this CI. Write it in your memo. Defend the interpretation.

The noise you haven't measured

Bootstrap quantifies **data-side** variability — uncertainty from which val examples you happened to sample.

There's a second source: **model-side** variability — uncertainty from which random seed / checkpoint / training run you happened to pick.

Sälevä et al. 2025 show that accounting for only one source substantially underestimates real replication variability.

For the memo: if you retrained the decoder with a different seed, how would macro F1 move? You don't know. That's noise you haven't measured.

Practical takeaway: bootstrap CIs are a **lower bound** on your true uncertainty. The real "is the gap real?" range is at least this wide, probably wider.

`readings/week4/saleva2025_uncertainty_nlp.pdf`

Section 5 — Bug Hunt Preview

Given a broken model, diagnose it from symptoms alone.

The scenario

Someone trained a decoder on this same data. Same base model. Same LoRA config. Same 3 epochs. Pushed it to HuggingFace Hub:

> `earino/ecbs5200-week4-flawed-checkpoint`

It's broken. You don't know how.

You get to load it, run inference, and diagnose from symptoms alone.

*This is a **deliberately clean instructional case** — real production failures are messier (distribution drift, partial OOV input, gradient updates gone wrong during continuous training) and rarely preserve everything else while scrambling one thing. The clean version teaches the diagnostic move. The messy version is what you face in a job.*

Failure modes that all look "random" at aggregate

When accuracy and macro F1 are near random, several very different bugs fit:

Bug	Behavior
Never trained	All predictions are argmax of random init
Collapsed to majority	Always predicts the most common class
Output space scrambled	Coherent predictions, wrong targets
Wrong features	Responds to noise instead of signal

You distinguish these from confusion-matrix structure, not from accuracy.

Diagnosing models you haven't inspected yet

In lab you diagnose ONE specific broken model. Someone told you it was broken.

In production: you don't know which models are broken. You need continuous monitoring that fires BEFORE you know to diagnose.

Nguyen et al. 2025 (NeurIPS): **D3M** — Disagreement-Driven Deterioration Monitoring.

- Train multiple models (or keep snapshots at different checkpoints)
- In deployment, track how often they disagree
- Disagreement-rate spike → alert → investigate
- **No ground-truth labels required.**

Today's lab is the *diagnostic* side. Production ML systems need *detection* — the alert that says **something** is wrong before you know what.

Week 6 preview: that disagreement signal is also the input to distillation.

`readings/week4/nguyen2025_d3m.pdf`

What you'll do in lab

1. Load the flawed checkpoint, run inference
2. See the metrics (near-random)
3. Build a diagnostic plot — "most-predicted class per true class"
4. Recognize the pattern
5. Apply a **one-line transformation** that recovers performance

If you pick the right transformation, the model's accuracy **jumps from near-random back to ~57%**.

Section 6 — Wrap + Week 5 Preview

Today's toolkit, restated

1. **Slice analysis** — averages hide subgroups
2. **Calibration** — accuracy ≠ trustworthiness
3. **Confusion-matrix patterns** — wrong how, not just wrong
4. **Val-set reliability** — small val = noisy F1
5. **Bug hunt** — given a broken model, narrow the failure mode

Five tools. They compound. Each rules out a different class of failure.

What Week 5 covers

Val reliability made you look at how classes are DISTRIBUTED.

Next week: how the data pipeline PRODUCES that distribution.

- Why 113 classes and not 153?
- Why does a cluster of classes have so few val examples?
- What decisions upstream of training created these shapes?

We'll open up the pipeline that made your dataset.

What Week 6 covers

You'll pull examples where encoder and decoder disagree.

Those disagreement examples have structure — some patterns favor the encoder, some the decoder.

Week 6: distillation. Use the decoder to generate labels for the disagreement set, train the encoder to match.

The diagnostic work you do this week is the INPUT to that.

Homework arc

Section	Work	Points
Slice analysis	All 6 axes; interpret 2	20
Calibration	Temperature scaling experiment	20
Confusion matrix	Per-class drill-down on worst 3 classes/model	25
Bootstrap CI	1000 resamples, read the CIs	20
Synthesis	Your Week 3 recommendation — has your confidence changed?	15

~4.5 hours. Memo is embedded. Due: Wednesday morning.

Today — in one sentence

You entered with two models and a recommendation.

You leave with the tools to defend or update that recommendation — with evidence.