Machine Learning

Imperial College London

Nuri Cingillioglu

Part 3: Good Machine Learning Practice

The art of alchemy...

The Alchemist's Dilemma

The "Kaggle / Alchemy" Trap

1. Load CSV file into a pandas dataframe.
2. Throw 10 different complex models at it.
3. Tweak random seeds and hyperparameters until leaderboard metric ticks up by $+0.02$.
4. Declare victory and deploy to production...
5. **System fails silently in production.**



> The Alchemist in Search of the Philosopher's Stone, by Joseph Wright, 1771, currently at Derby Museum Gallery

Machine Learning as Science & Engineering

- In **Part 1 & 2**, we treated ML mathematically:
  - Optimising convex loss surfaces ($\mathbf{w}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$)
  - Squashing probabilities ($\sigma(\mathbf{w}^T\mathbf{x})$)
  - Expanding polynomial feature bases ($\phi(\mathbf{x})$)
- In the real world, **mathematics meets messy reality**:
  - Missing records, corrupt labels, class imbalance
  - Subtly leaked future information
  - Misaligned evaluation metrics
  - Non-stationary environments in the wild
- Your LLM buddy could miss one of these too…

Step 1. Framing. Stop, Think, Plan

1. **What is the exact decision being made?**
    - Are we diagnosing a patient? Pricing a house? Blocking a financial transaction?
2. **Who is the stakeholder and what do they care about?**
    - A clinician cares about patient safety; a CFO cares about operational cost.
3. **What is the decision frequency and latency budget?**
    - Millisecond web serving vs. overnight batch processing.
4. **What are the consequences of being wrong?**
    - Are False Positives and False Negatives equally costly? *(They almost never are.)*
5. **How will this model fail silently?**
    - What happens when input distributions shift next month? The LLM hallucinates?

Example: Hospital Readmission

- **Goal:** Predict whether a discharged patient will suffer unplanned hospital readmission within 30 days $y \in \{0, 1\}$.
- **Patient Cohort:** $N = 10,000$ patient records.
- **Features:** Demographics (age, sex), vital signs (blood pressure, heart rate), laboratory biomarkers (glucose, creatinine), prior admission history.
- **Asymmetric Danger:**
  - Missing an at-risk patient (FN) $\implies$ severe health relapse.
  - Extra follow-up checkup (FP) $\implies$ nurse consultation fee.



That is not the goal...

- The goal is to reduce costs.
- The goal is to open up beds for new patients.
- The goal is to move "care to local centres".
- The goal is to free up doctor's time.
- The goal is to deny insurance claims.
- The goal is to ...

Predicting patient readmission is a **ML framing** of the problem.

Where is AI?

- We can reframe this problem as **AI** hospital and patient management instead of basic classification problem.
- Oh look, our **AI** takes X-ray images, patient records with multi-modal LLM to determine whether they'll need to come back to the hospital.
- We save the public sector billions! Raise venture capital based on our AI model...

My AI can predict patient readmissions to reduce hospital costs and free up doctor's time!

> In other words, AI is everywhere! Especially where money is these days.

Step 2: Data, daita, DATA!

Garbage In $\rightarrow$ Garbage Out

For example:

- **Measurement Noise:** Faulty pulse oximeters reporting noisy oxygen saturation.
- **Label Noise:** Two attending doctors disagreeing on whether a discharge diagnosis constitutes pneumonia vs. bronchitis.
- **Sample Selection Bias:** Training exclusively on patients admitted to a wealthy private hospital, then deploying to an underfunded urban clinic.
- **Survivor Bias:** Evaluating treatment efficacy only on patients who survived long enough to complete all 4 follow-up appointments.

> I once had pneumonia and was initially misdiagnosed with a strained rib cage.

Load the data: Tabular

patient_id	age	sex	systolic_bp	hr	glucose	creatinine	prior_adm_12m	readm_30d
PT-00104	72	M	148	88	162	1.8	2	1
PT-00219	45	F	118	72	94	0.8	0	0
PT-00382	68	F	134	78	138	1.4	1	0
PT-00411	81	M	158	96	185	2.3	4	1
PT-00507	34	M	122	64	88	0.9	0	0

> Careful now, each row is an actual human... I obvivously made this data up but you get the idea of what I'm trying to say.

Load the data: Images & Videos

An image is a matrix (or tensor) of pixel values, for ML we usually make them the same size:

$$X \in \mathbb{R}^{N \times H \times W \times C}$$

where

- N is the number of images
- $H \times W$ is the height and width of the images
- C is the number of channels (1 for greyscale, 3 for RGB)
- Values range from 0 to 255.

A video is just a collection of images over time T:$V \in \mathbb{R}^{N \times T \times H \times W \times C}$

Load the data: Text

A string is a sequence of characters but we usually *tokenise* it into parts

- words
- subwords
- characters

Each token is then assigned a unique ID, for example:

```
# tokeniser from LLM

text = "The cat sat on the mat."
tokens = tokenizer.encode(text)
# output: [42, 17, 556, 20, 321, 13, 42, 997, 14]
```

You'll learn more in advanced modules on how to deal with images, videos, text, audio and other data types.



MIT and Toyota release innovative dataset to accelerate autonomous driving research. MIT News, 2020.

Standard Benchmark Datasets

Tabular

- **Iris** (Fisher, 1936): $N = 150$, 4 features, 3 plant classes.
- **Adult Census** (1996): Income $> \$50\text{k}$ prediction; standard for fairness.
- **California Housing**: Continuous median property values ($N \approx 20\text{k}$).

Vision

- **MNIST** (1998): 28×28 handwritten digits; the "Hello World" of ML.
- **CIFAR-10 / 100** (2009): 32×32 natural colour images.
- **ImageNet** (2009): 1.4M+ images, 1k classes; sparked modern Deep Learning.

Text & Audio

- **IMDb** (2011): 50k movie reviews for sentiment classification.
- **SQuAD** (2016): Reading comprehension QA from Wikipedia.
- **LibriSpeech** (2015): 1,000 hrs of read English speech for ASR.

Structured vs Unstructured

There is often a distinction made between structured and unstructured data. It mainly boils down to whether it is tabular or not.

Examples of **structured data** are:

- patient records
- credit card transactions
- house prices

Examples of **unstructured data** are:

- images
- videos
- text
- audio

> Notice the category where **generative AI** has worked really well?

Data Processing: Scaling

$$\theta := \theta - \alpha \nabla_\theta J(\theta) = \theta - \alpha(\mathbf{X}^T\mathbf{X}\theta - \mathbf{X}^T\mathbf{y})$$

If we have features with very different scales, the gradient will be much larger for features with larger scales. The training goes BOOM.

For example: patient age (e.g. 0-100) vs creatinine level (e.g. 0.5-2.5).

Solution: Scale the features so that they are in a similar, and often standard, range.

> This applies to neural networks too since if you remember we started of with a artificial neuron (Perceptron) and ended up here.

Scaling cont.

Gradient descent and distance metrics (k-NN, Ridge, Lasso) are sensitive to feature scales:

1. Standardisation (Z-score Normalisation)

$$z_i = \frac{x_i - \mu}{\sigma} \quad \Longrightarrow \quad \mu_z = 0,\ \sigma_z = 1$$

- Good for bell-shaped, Gaussian-like distributions.

2. Min-Max Normalisation (Bounded Rescaling)

$$x_{\text{scaled}} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \in [0, 1]$$

- Bounds features to $[0, 1]$; highly vulnerable to extreme outliers.

For examples, image pixels are usually scaled to $[0, 1]$ or $[-1, 1]$.

Data Processing: Missing Values

Deletion

Have too much data? Get rid of the rows/columns with missing values.

Imputation

definition: to calculate something when you do not have exact information, by comparing it to something similar

- **Mean / Median**: replace missing values with the mean / median of the feature.
- **k-NN**: replace missing values with the mean / median of the k-nearest neighbours.
- **Constant**: replace missing values with a constant value.
- **Binary Indicator**: add an explicit binary indicator: $m_i = \mathbb{I}(x_i \text{ is missing})$.

> This business does not come up as much in Deep Learning as it does in tabular ML.

Categorical Encodings

Categorical variables (e.g., Hospital Department: *Cardiology, Oncology, Neurology*) must be mapped to numbers:

- **One-Hot Encoding:** Maps K categories to K binary indicator columns:

$$\begin{cases} \text{Cardiology} & \to [1,0,0]^T \\ \text{Oncology} & \to [0,1,0]^T \\ \text{Neurology} & \to [0,0,1]^T \end{cases}$$

The Sacred Tri-Partition

Never evaluate a model on the data used to train or tune it:

Training Set (~70%)

Purpose: Learning the model parameters.

Training error is a measure of optimisation capacity, **never** of generalisation.

Validation Set (~15%)

Purpose: Hyperparameter tuning and model selection.

Also used for debugging the question "my model is not learning anything".

Test Set (~15%)

Purpose: Unbiased estimation of true generalisation.

Touch **ONCE** at project completion.

If you tune models based on test results, the test set becomes a validation set!

k-Fold Cross-Validation

When datasets are moderate in size, a single train/validation split yields high evaluation variance $\rightarrow$ we can't trust the single result.

The k-Fold Process

1. Partition dataset $\mathcal{D}$ into k equal-sized, mutually exclusive subsets (folds) $\{F_1, F_2, \ldots, F_k\}$
2. For each iteration $i \in \{1, \ldots, k\}$:
    - **Validation Fold:** F_i
    - **Training Set:** $\bigcup_{j \neq i} F_j$
    - Fit pipeline on training folds, record metric M_i on F_i.
3. Compute the mean performance and standard error across folds:

$$\bar{M} = \frac{1}{k} \sum_{i=1}^{k} M_i$$

k-Fold Cross-Validation

Key Advantages

- Every data point serves as a validation sample exactly once.
- Mitigates lucky or unlucky validation splits.
- Standard choices: $k = 5$ or $k = 10$.
- Trade-off: k times the computational training cost.

> With large deep learning era datasets, k-fold cross validation is often too computationally expensive. In these cases, a single train/val split is often still used but the datasets are very large (millions of samples) and we assume we have enough samples for the test set to be representative of the distribution.

Stratified k-Fold

The Problem

- In our hospital readmission dataset, only **5%** of patients are readmitted ($y = 1$).
- With standard random partitioning:
  - Small validation folds may contain **zero** or very few positive cases!
  - Evaluation metrics fluctuate wildly or collapse mathematically.

The Solution

- **Stratified k-Fold** tries to preserve the class ratio $P(y = 1) : P(y = 0)$ in every single fold.
- **Rule of Thumb:** In classification, **always** use Stratified k-Fold unless explicit groups or time series require otherwise.

Stratified k-Fold

```
from sklearn.model_selection import StratifiedKFold, cross_val_score

# Stratified K-Fold automatically maintains 5% positive rate across all 5 folds
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(full_pipeline, X, y, cv=skf, scoring="...")
print(f"Stratified CV: {cv_scores.mean():.3f} +/- {cv_scores.std():.3f}")
```

Group k-Fold

What happens when individual entities appear multiple times in the dataset?

The Scenario

- Patient #42 visits the hospital **4 times** across the calendar year.
- If we perform standard random splitting:
  - 3 visits land in the Training set.
  - 1 visit lands in the Validation set.

The Catastrophic Failure

- The model memorises Patient #42's unique traits (blood type, genetics, rare chronic conditions).
- Validation score: **97% accuracy!**
- Deployed on brand new patients: **Performance collapses to 62%!**

Group k-Fold

- Splits are assigned at the **group level** (by `patient_id`).
- All records belonging to a patient are strictly assigned to either Train **or** Validation.

```
from sklearn.model_selection import GroupKFold

# Ensure Patient IDs never straddle train and test
gkf = GroupKFold(n_splits=5)
scores = cross_val_score(full_pipeline, X, y,
                         groups=patient_ids, cv=gkf)
```

Cross-Validation Splitting Protocols

Matching the validation protocol to the underlying data generation process

1. Standard k-Fold

IID Tabular Data (Random Uniform Folds)



Assumes independent & identically distributed (IID)

2. Stratified k-Fold

Imbalanced Classes (Preserves P(y=1) in every fold)



Guarantees minority class presence in all folds

3. Group k-Fold

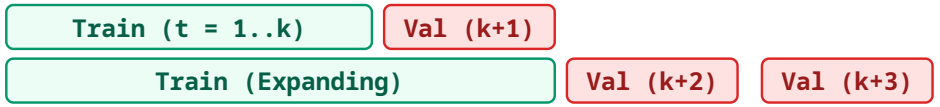
Clustered Entities (Patient IDs strictly un-split)



Zero patient overlap between Train and Validation

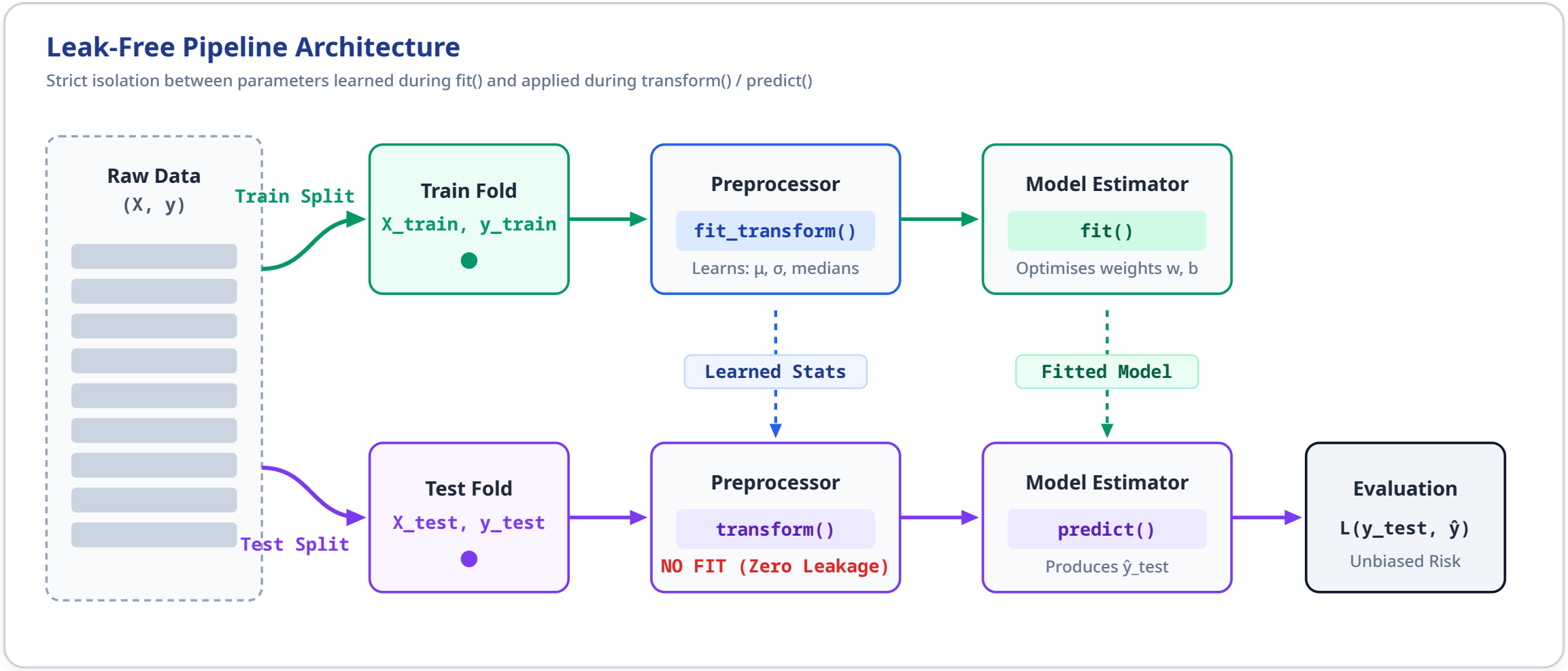
4. Time-Series

Sequential Records (Respects temporal causality)



Never shuffle; past strictly predicts future

Overview of Pipeline



Scikit-Learn Pipeline

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.linear_model import LogisticRegression

# Define numerical and categorical feature subsets
num_features = ["age", "blood_pressure", "glucose", "creatinine"]
cat_features = ["department", "admission_type"]

# Preprocessing sub-pipelines
num_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="median")),  # Robust to outliers
    ("scaler", StandardScaler())
])

cat_pipeline = Pipeline([
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(handle_unknown="ignore"))
])

# Assemble master preprocessor
preprocessor = ColumnTransformer([
    ("num", num_pipeline, num_features),
    ("cat", cat_pipeline, cat_features)
])

# Complete leak-free pipeline: preprocessor + model
full_pipeline = Pipeline([
    ("preprocessor", preprocessor),
    ("classifier", LogisticRegression(max_iter=1000, C=1.0))
])
```

The Modern Python ML Stack

Library	Primary Abstraction	Role in the Ecosystem
NumPy	N-dimensional array (`ndarray`)	Vectorised numerical linear algebra
Pandas	DataFrame / Series	Data ingestion, tabular wrangling, indexing & grouping
Scikit-Learn	`Estimator`, `Pipeline`	Classical algorithms, preprocessing, validation, etc.
SciPy	Sparse matrices, statistical tests	Hypothesis testing (`scipy.stats`), distribution fittings
Matplotlib	Plots, figures	Thankfully LLMs do this job now
PyTorch	Tensor with autograd engines	Deep learning models and training

> It was never about writing code manually. I hated, absolutely hated matplotlib.

Step 3. Model

The more exciting part of the journey.

- Depends on the task you are solving: supervised, unsupervised, reinforcement learning.
- Depends on the data you have. Is it tabular? Images? Text? Audio? Video?
- How much data do you have? Not enough = maybe fine-tune, more data = train from scratch.

The Non-ML Baseline

Never train a complex machine learning model until you have established a baseline.

- **Sanity check:** Verifies data loading and pipeline integrity.
- **Cost-benefit benchmark:** If a 3-line threshold heuristic achieves 92% of an LLM's performance, is the operational complexity worth it?
- Inexperienced practitioners jump straight to deep networks, LLMs and what not.
- Without a baseline, you cannot tell if an 88% score represents breakthrough intelligence or catastrophic underperformance.

A Hierarchy of Baselines

Level	Baseline Type	Hospital Readmission Example	Primary Purpose
0	**Random Guess**	Flip coin: predict 1 with prob 0.5	Absolute lower floor
1	**Majority Class**	Predict 0 (Never readmitted) for all patients	Exposes class imbalance
2	**Simple Domain Heuristic**	If $\text{Age} > 75$ and $\text{Prior Admissions} \geq 2 \implies 1$	Tests human expert rules
3	**Simple Linear Model**	Logistic Regression on standard features	Quantifies linear signal
4	**AI / Complex ML Model**	Deep Networks...	Justifies compute/latency

> Most tasks you would work on would already have an existing model you can compare against. Start from that given baseline.

Establishing Baselines

```python
import numpy as np
from sklearn.dummy import DummyClassifier
from sklearn.linear_model import LogisticRegression

# 1. Majority class baseline: predicts most frequent class (always y=0)
dummy_majority = DummyClassifier(strategy="most_frequent")
dummy_majority.fit(X_train, y_train)
acc_majority = dummy_majority.score(X_test, y_test)
print(f"Majority Baseline Accuracy: {acc_majority:.3f}")  # Often 95%+ !

# 2. Simple human domain rule: Age > 75 and Prior Admissions >= 2
y_pred_heuristic = ((X_test["age"] > 75) & (X_test["prior_admissions"] >= 2)).astype(int)

# 3. Simple parametric linear baseline (Logistic Regression)
simple_linear = LogisticRegression(max_iter=1000)
simple_linear.fit(X_train, y_train)
acc_linear = simple_linear.score(X_test, y_test)
print(f"Linear Baseline Accuracy:   {acc_linear:.3f}")
```

My AI Master 9000 model

The future of AI hospital management…

A mission-critical, paradigm-shifting, state-of-the-art multimodal large foundation model (LFM) featuring an agentic, zero-shot, transformer-based mixture-of-experts (MoE) architecture, seamlessly augmented via cross-embodiment, self-supervised spatial-temporal vision-language-action (VLA) robot pre-training, diffusion-policy synthesis, and closed-loop sim-to-real reinforcement learning with human-in-the-loop AI feedback (RLAIF) to enable hyper-scalable, sovereign, edge-native, general-purpose physical intelligence and holistic neuro-symbolic reasoning across dynamic enterprise ecosystems.

You may have detected by now a cynical view, you are not mistaken.

Step 4. Training

- Start small, test on a small subset to rule out errors and check if your baselines are working. *This is probably my most important advice.*
- **Research engineering** is the art of getting things to work, and ~~prompting~~ debugging is a large part of it.
- Can you estimate training times?
- How is the loss doing? Is it going down?
- Is the model **improving**? We'll look into that after training.

Empirical Risk vs. True Risk

The Generalisation Problem

Let data samples $(\mathbf{x}, y)$ be drawn from an unknown joint probability distribution $\mathcal{D}$.

- **True Generalisation Risk (Expected Loss):**

$$\mathcal{R}(\hat{f}) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[L(\hat{f}(\mathbf{x}), y)]$$

- **Empirical Risk (Observed Training Loss on Sample S of size N):**

$$\hat{\mathcal{R}}_S(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(\hat{f}(\mathbf{x}_i), y_i)$$

where $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ i.i.d. samples from $\mathcal{D}$.

The Optimisation Paradox

- We **optimise** empirical risk:

$$\hat{f}^* = \arg\min_{\hat{f} \in \mathcal{F}} \hat{\mathcal{R}}_S(\hat{f})$$

- But we **care only** about true risk: $\mathcal{R}(\hat{f}^*)$

.

The Generalisation Gap

$$\Delta_{\text{gen}} = \mathcal{R}(\hat{f}^*) - \hat{\mathcal{R}}_S(\hat{f}^*)$$

- Overfitting occurs when $\hat{\mathcal{R}}_S \to 0$ while true risk $\mathcal{R}$ explodes!

The common solution is to get **more and moar data** like we already discussed in the statistical foundations lectures.

> A great example of this is the self-driving car dream. How much driving data do you need until you think it can handle every possible condition out there?



Taxi from driverless firm veers into crime scene, BBC News, Winnie Agbonlahor, Published: 24 April 2026.

Step 5. Evaluation

- Did we do anything useful?
- How do we know if intelligence is round the corner?

Evaluation is a very important but difficult to define in full. We will look at some common metrics to help start the process.

The Accuracy Paradox

The 95% Illusion

In our hospital dataset of $N = 10,000$ patients where 5% are readmitted:

$$\text{Class 0 (Not Readmitted): } 9,500 \quad | \quad \text{Class 1 (Readmitted): } 500$$

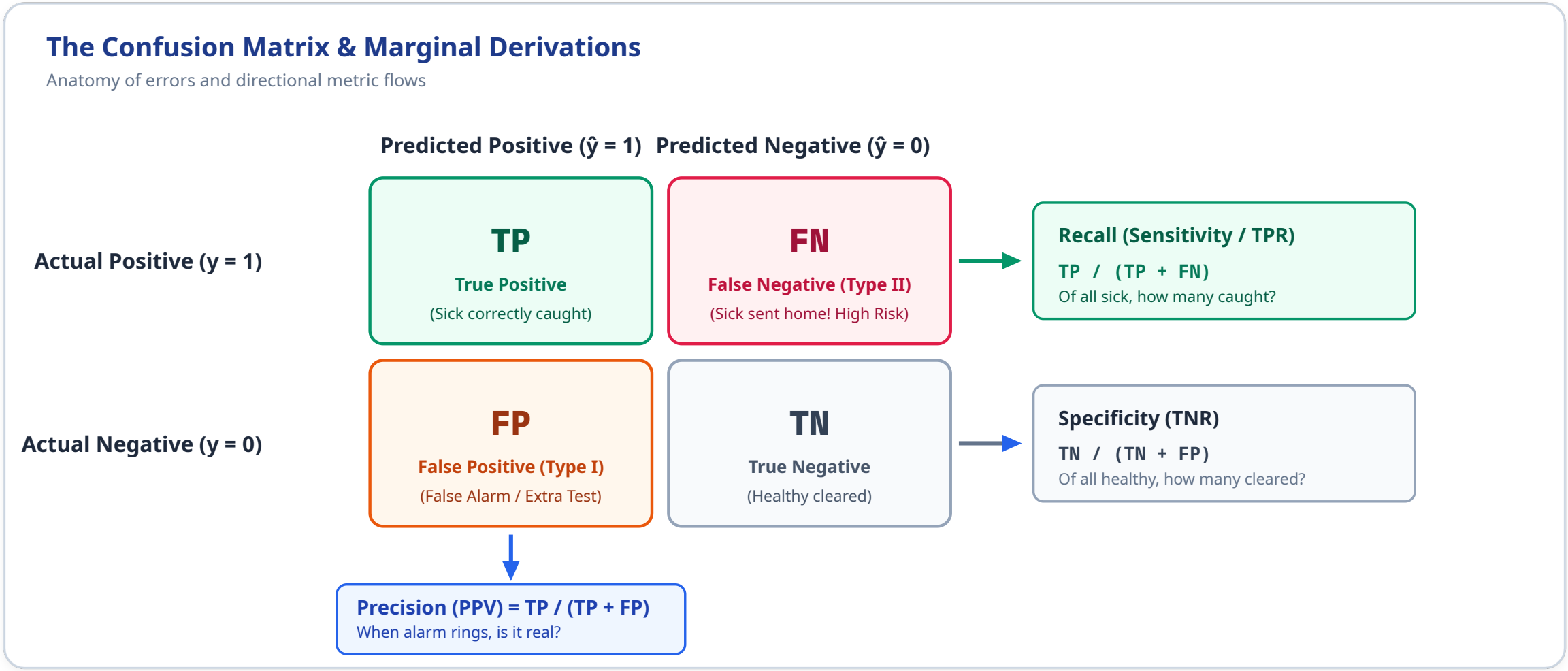
A dummy model that ignores patient features and always outputs $\hat{y} = 0$:

$$\text{Accuracy} = \frac{9,500 + 0}{10,000} = \mathbf{95.0\%}$$

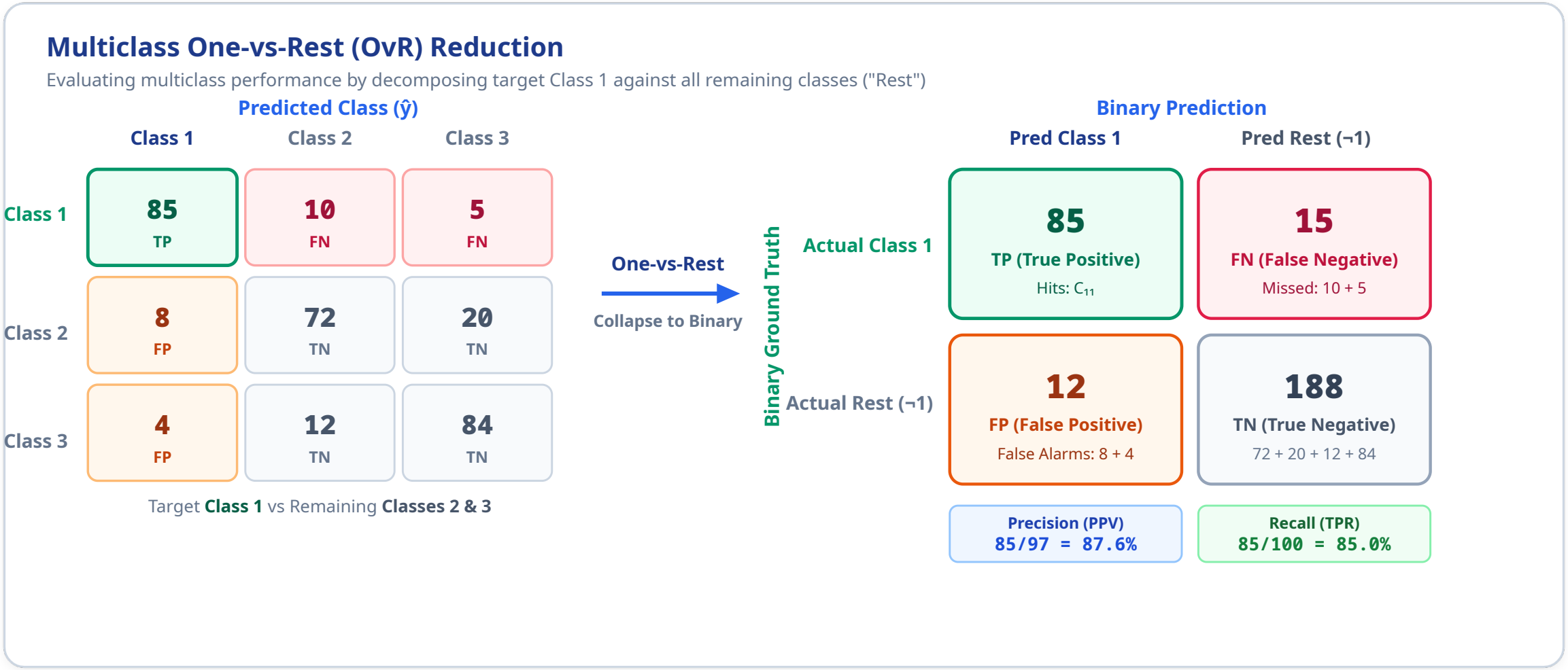
It has zero diagnostic value and catches zero at-risk patients!

Goodhart's Law: *"When a measure becomes a target, it ceases to be a good measure."*

The Confusion Matrix



Multiclass Confusion Matrix



Precision: The Alarm Sounder's Metric

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{\text{True Positives}}{\text{All Predicted Positives}}$$

Intuition

"When the model sounds the alarm and flags a patient, how often is it actually right?"

High Precision is critical when:
The intervention is expensive (e.g., performing an exploratory invasive biopsy, shutting down a power grid).

The Cost of Low Precision

- **"Alarm fatigue"**
- If 95% of model alerts are false alarms, ignore or disable the predictive system entirely.

Recall (Sensitivity): The Safety Net Metric

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{\text{True Positives}}{\text{All Actual Positives}}$$

Intuition

"Of all patients who were genuinely at risk of readmission, what fraction did the model successfully catch?"

High Recall is critical when:
The consequence of missing a positive case is catastrophic (e.g., fatal cardiac arrest, fraudulent cyberattack).

The Cost of Low Recall

At-risk patients are sent home without follow-up care, leading to severe health complications and preventable emergency readmissions.

Harmonic Mean: The F_1 Score

Why can't we simply take the arithmetic mean $\frac{P+R}{2}$ to balance Precision and Recall?

Arithmetic Mean

- Consider a useless model with:

$$\text{Precision} = 1.0, \quad \text{Recall} = 0.001$$

- Arithmetic mean: $\frac{1.0+0.001}{2} \approx \mathbf{0.5}$
- Hides the true behaviour of the model.

The F_1 Score

- Harmonic mean:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} = \mathbf{0.002}$$

- **The harmonic mean approaches zero if either metric collapses.**

Moving the Decision Threshold τ

Classification models (like Logistic Regression) output a continuous probability $\hat{p} \in [0, 1]$:

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{p} \geq \tau \\ 0 & \text{if } \hat{p} < \tau \end{cases}$$

The Myth of $\tau = 0.5$

The default software threshold $\tau = 0.5$ is completely arbitrary. You can play with it to tune your model's performance based on your needs.

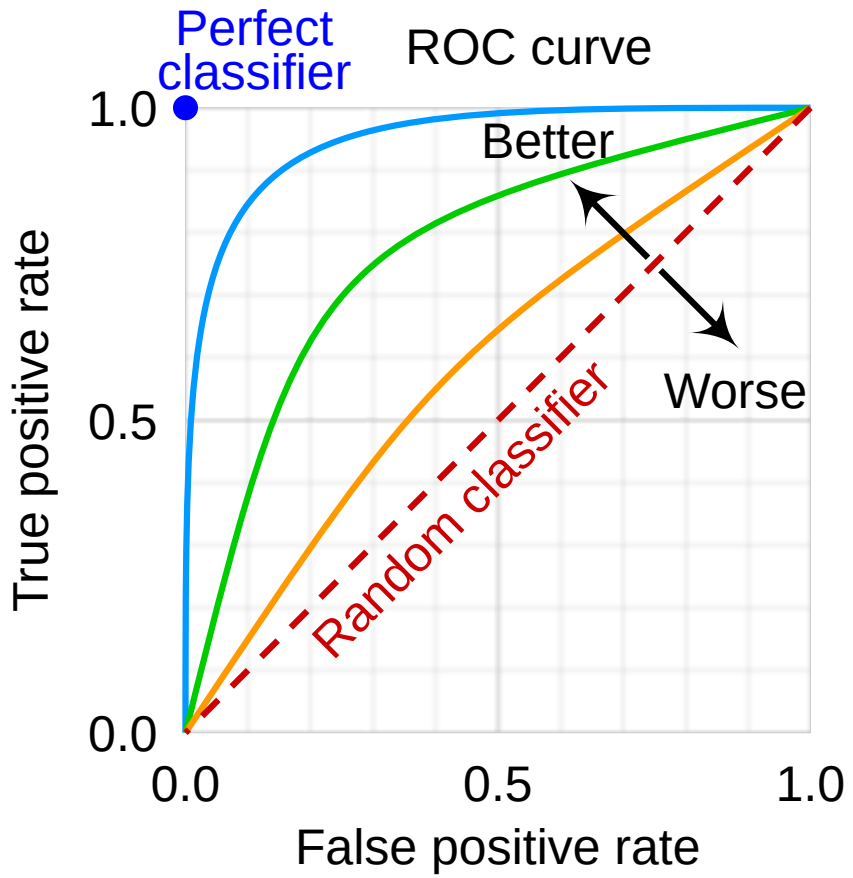
- **Lowering τ (e.g., $\tau = 0.15$):**
  - Flags more patients $\implies$ **Recall $\uparrow$, Precision $\downarrow$.**
- **Raising τ (e.g., $\tau = 0.85$):**
  - Flags only confident patients $\implies$ **Precision $\uparrow$, Recall $\downarrow$.**

The ROC Curve & AUROC

The Receiver Operating Characteristic (ROC) curve evaluates ranking quality across **all possible thresholds**:

ROC Mechanics

- Sweeps threshold τ continuously from $1.0 \to 0.0$.
- Plots:
  - **Y-axis:** True Positive Rate (Recall) $= \frac{TP}{TP+FN}$
  - **X-axis:** False Positive Rate $= \frac{FP}{FP+TN} = 1 - \text{Specificity}$
- **Diagonal:** Random guessing ($\text{AUROC} = 0.50$).



Probabilistic Interpretation of AUROC

$$\text{AUROC} = P(f(\mathbf{x}^+) > f(\mathbf{x}^-))$$

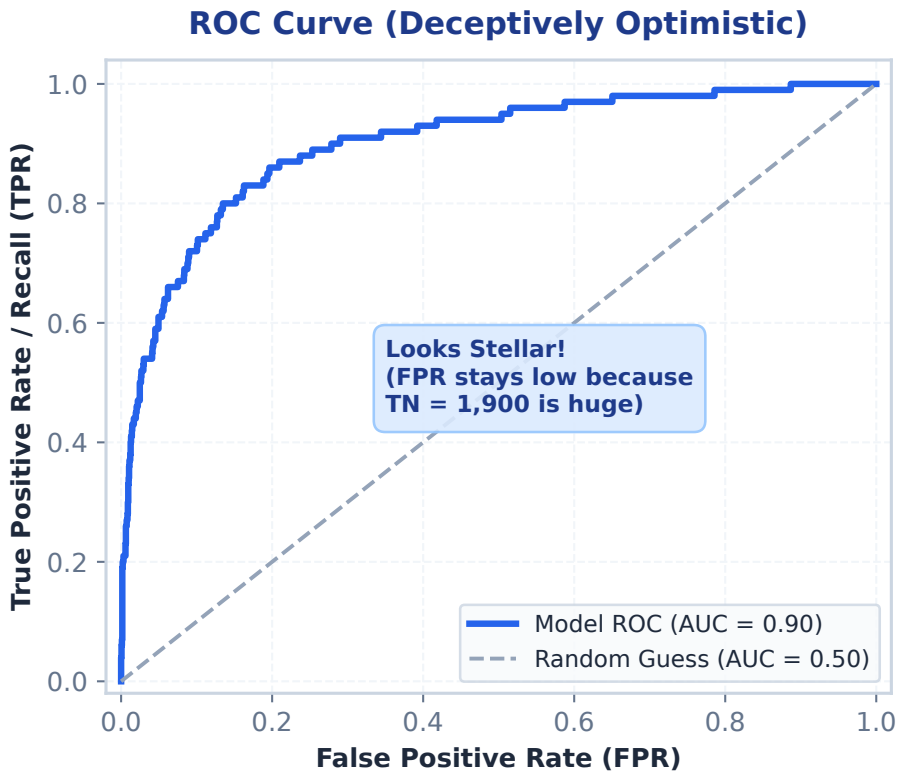
The probability that a randomly chosen positive patient receives a higher predicted risk score than a randomly chosen negative patient.

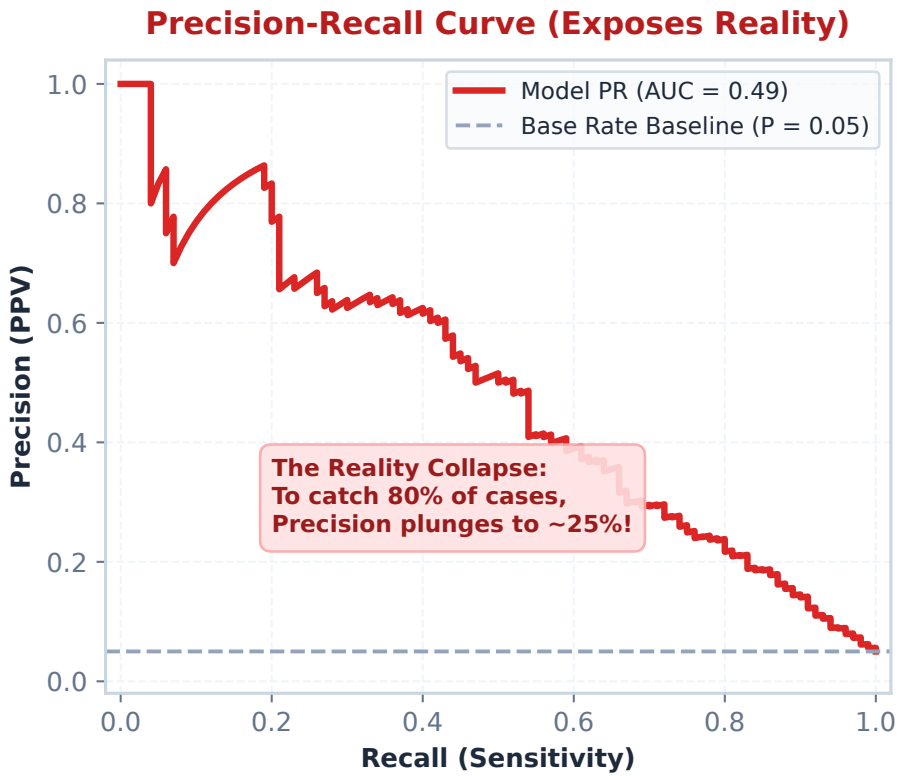
Ranking Perspective

- **Pairwise ordering:** AUROC measures how well the model ranks samples, not whether output values are calibrated probabilities.
- $\text{AUROC} = 0.5$ **(Random):** Equivalent to a fair coin toss; positive and negative distributions overlap entirely.
- $\text{AUROC} = 1.0$ **(Perfect):** Every positive is ranked strictly above every negative.

> The formal derivation is out of scope for us and ventures further into order statistics and probability theory.

ROC vs. PR Curves Under Extreme Imbalance





- When the negative class is massive ($TN = 1,900$), false alarms do not impact the FPR denominator: $\frac{FP}{FP+TN}$. The ROC curve stays high.
- The **P-R Curve** exposes the collapse: to reach 80% recall, Precision plunges to 25%.

Regression Metrics: When $y \in \mathbb{R}$

When predicting continuous targets (e.g., patient Length of Stay in hospital days):

Metric	Formula	Penalty Profile	Dimensionality
Mean Squared Error (MSE)	$\frac{1}{N}\sum(y_i - \hat{y}_i)^2$	Quadratic: heavily punishes large errors	Units^2 (non-intuitive)
Root Mean Squared Error (RMSE)	$\sqrt{\frac{1}{N}\sum(y_i - \hat{y}_i)^2}$	Quadratic penalty, but on original scale	Units (interpretable)
Mean Absolute Error (MAE)	$\frac{1}{N}\sum\lvert y_i - \hat{y}_i\rvert$	Linear: robust to extreme outliers	Units (median-like)

> **Selection Rule:** Use **RMSE** when large errors are disproportionately disastrous; use **MAE** when data contains anomalous measurement spikes.

Evaluation Studio

VISUAL LAB **Machine Learning Evaluation Metrics**

SCENARIOS: Balanced (20%) | Rare Fraud (2%) | Medical Screening (Low τ) | Spam Gate (High τ) | Weak Model (Low d')

1. DISTRIBUTIONS & CUTOFF

N = 1,000

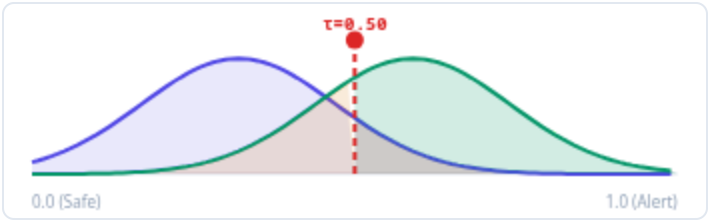
Class Separation (d') 1.80

Prevalence (% Positive) 20%

Decision Threshold (τ) 0.50

SCORE DENSITY P(S | CLASS)

Drag canvas to shift τ



Negatives (0) Positives (1) Cutoff τ

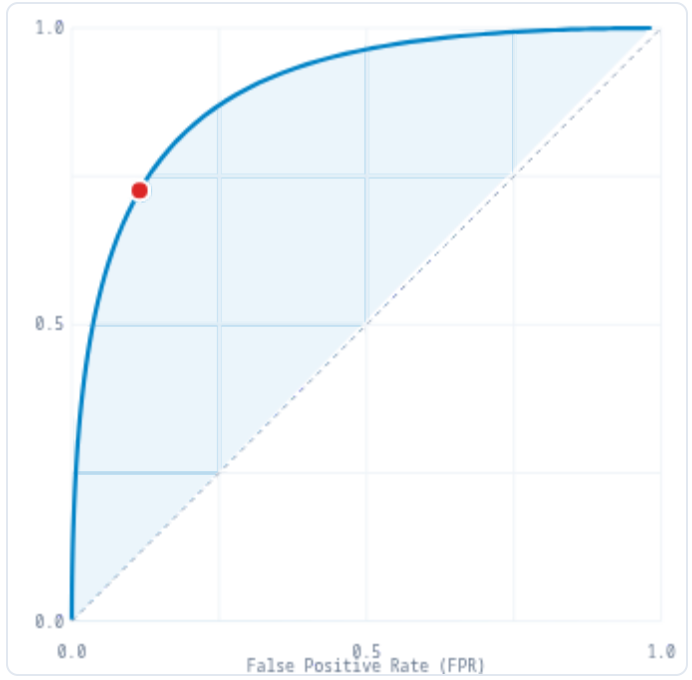
CONFUSION MATRIX

Real-Time Counts

ACTUAL \ PRED	Pred (+)	Pred (-)	Total
Actual (+)	TP 145 True Hit	FN 55 Miss (Type II)	200
Actual (-)	FP 92 False Alarm (I)	TN 708 Clean Rejection	800

2. ROC SPACE (TPR VS FPR)

AUROC = 0.898



ROC Curve Random Guess (0.50) Operating Point (τ)

ROC Properties: Evaluates threshold-invariant discrimination. Unaffected by changes in class prevalence.

`Operating: FPR = 11.5% | TPR = 72.6%`

3. PR SPACE (PRECISION VS RECALL)

AUPRC = 0.729



PR Curve Baseline (π) Operating Point

METRIC SCOREBOARD (T = 0.50)

Live Evaluation

ACCURACY	PRECISION (PPV)
85.3%	61.2%
RECALL (TPR)	**F1-SCORE**
72.5%	0.66

⚖️ **Balanced Operational Operating Point:**

Threshold τ is 0.50. Out of 200 positive cases, 145 are caught (72.5% recall). For every positive alert, 61.2% are genuine hits.

Qualitative Evaluation

Never settle for scalar summary metrics! Inspect individual failure samples. Try to understand what is the model good at, what is it bad at.

Clever Hans

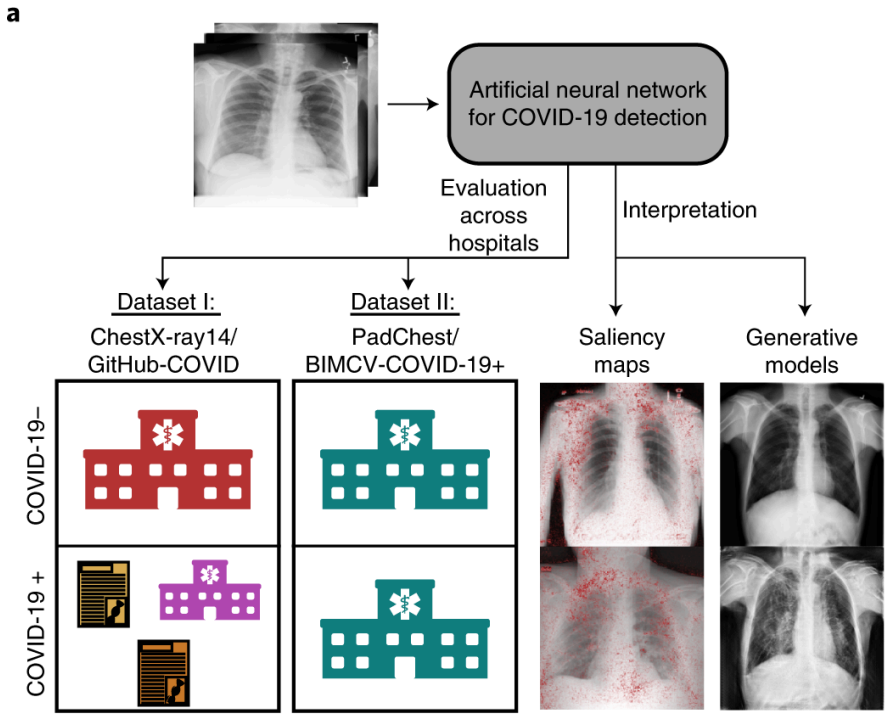
In 1904, a German horse named **Clever Hans** was famous for performing arithmetic.

Investigation revealed Hans could not do math: he was detecting subtle, involuntary head micro-tensions from the questioner.



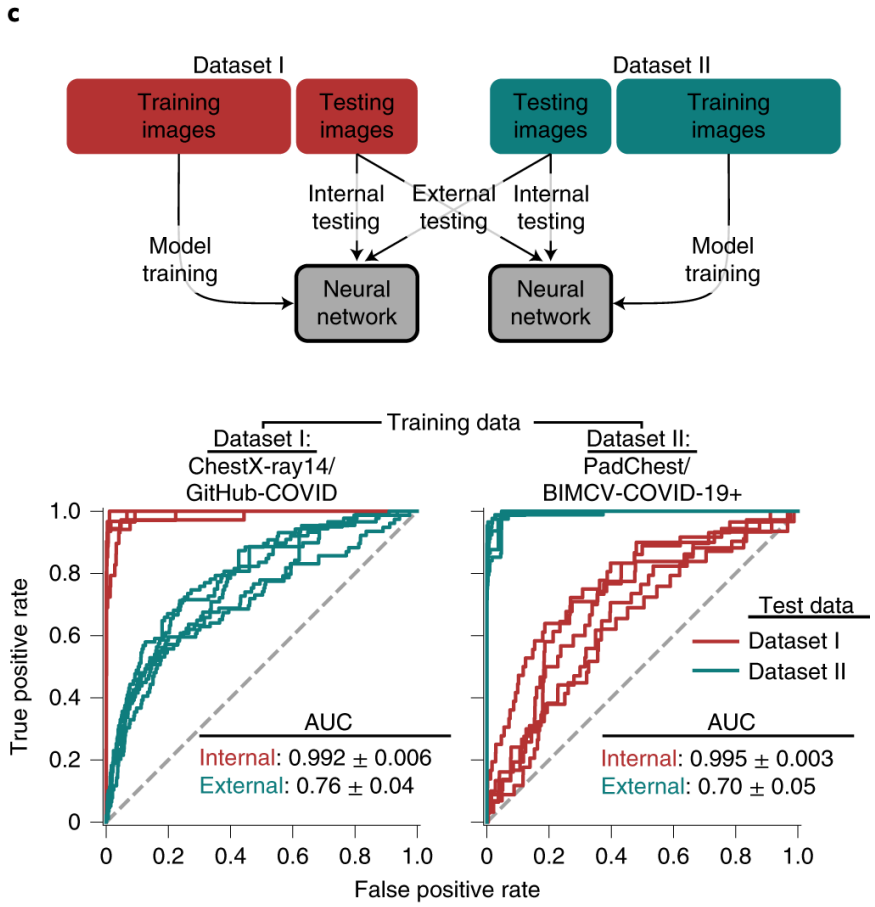
> Clever Hans (1904), CC BY-SA 4.0 WikiMedia Commons

Modern Clever Hans



b

	Dataset I			Dataset II		
	Combined	Chest-X-ray14	GitHub-COVID	Combined	PadChest	BIMCV-COVID-19+
No. radiographs	112,528	112,120	408	97,866	96,270	1,596
No. patients	31,067	30,805	262	64,954	63,939	1,105
% COVID-19+	0.2	0	76.5	1.6	0	100
% AP images	39.9	40	26	5.6	4.7	58.1



DeGrave et al. AI for radiographic COVID-19 detection selects shortcuts over signal. (2021)
https://doi.org/10.1038/s42256-021-00338-7

Step 6. Tuning Hyperparameters

Before tuning a model, distinguish between internal weights and structural controls:

Model Parameters (θ, $\mathbf{w}$, b)

- Learned automatically from the **Training Set** via numerical optimisation:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w})$$

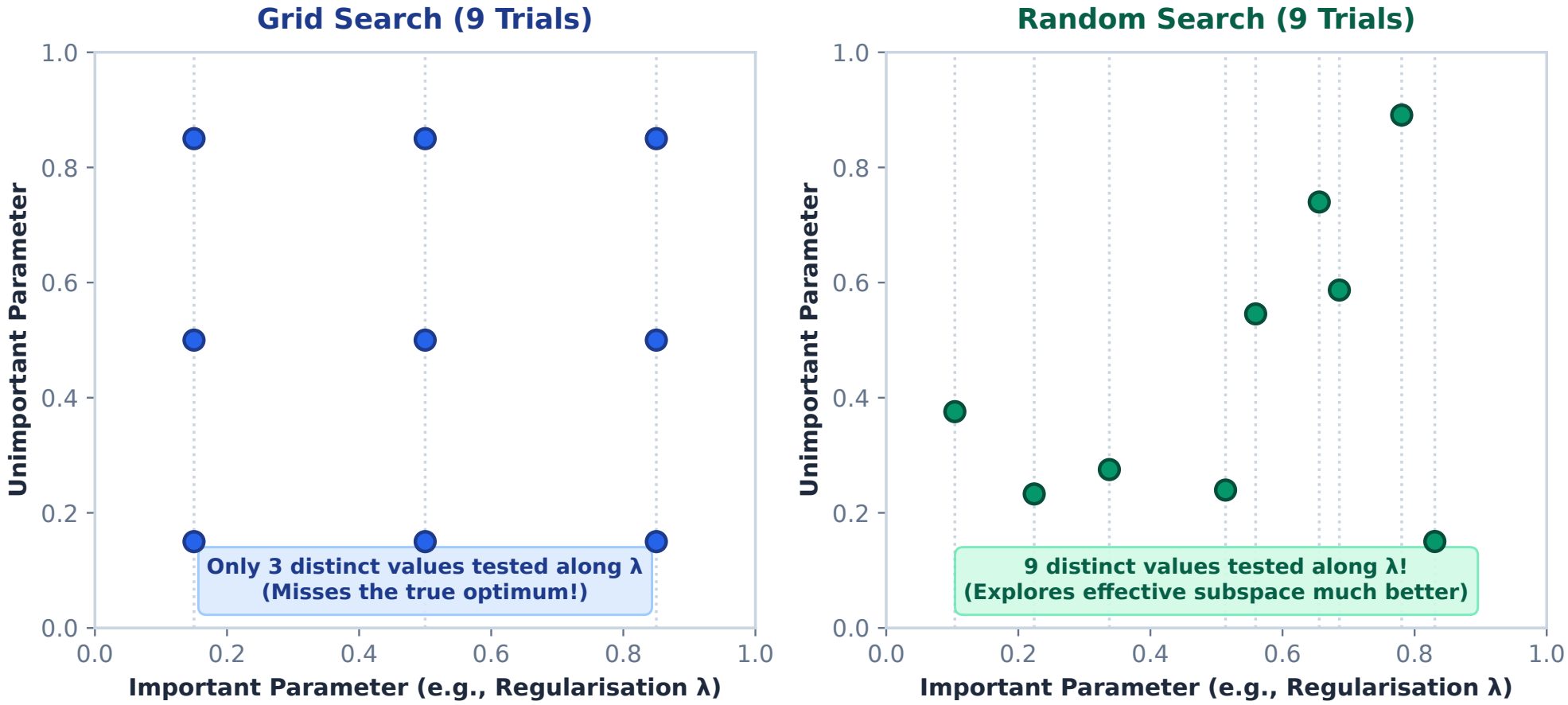
- *Examples:*
  - Weights in neural network ($\mathbf{w}$)
  - Bias terms (b)
  - Decision tree split thresholds

Hyperparameters (γ, λ, k, τ)

- Structural configuration dials that govern model capacity and regularisation.
- **Cannot be learned by minimising training loss!**
- *Examples:*
  - Regularisation weight λ
  - Number of neighbours k in k-NN
  - Maximum tree depth $d_{\max}$
  - Learning rate η in gradient descent

Grid Search vs. Random Search

How should we explore multi-dimensional hyperparameter spaces?



Nested Cross-Validation

What if you tune hyperparameters on cross-validation folds and report the best CV score as your result?

- If you have large amounts of data, tune on the validation set.
- If you have moderate sized datasets, use nested cross-validation.

The Nested CV Solution

- **Outer Loop (e.g., 5-Fold):** Evaluates overall generalisation performance of the modelling process.
- **Inner Loop (e.g., 3-Fold):** Conducts hyperparameter tuning strictly within each outer training slice.
- Guarantees that reported performance has never been observed during hyperparameter selection.

Code: Hyperparameter Tuning

```
from sklearn.model_selection import RandomizedSearchCV, StratifiedKFold

# Define hyperparameter search distributions
param_distributions = {
    "classifier__C": [1e-4, 1e-3, 1e-2, 1e-1, 1, 1e1, 1e2],
    "classifier__penalty": ["l1", "l2"],
    "classifier__solver": ["liblinear"]
}

# Wrap pipeline inside RandomizedSearchCV
search = RandomizedSearchCV(
    estimator=full_pipeline,
    param_distributions=param_distributions,
    n_iter=50,                                    # 50 random parameter samples
    scoring="roc_auc",
    cv=StratifiedKFold(n_splits=5, shuffle=True),
    random_state=42,
    n_jobs=-1
)

# Fit search: all preprocessors inside pipeline are refitted per fold!
search.fit(X_train, y_train)
print(f"Optimal Hyperparameters: {search.best_params_}")
print(f"Best Inner CV ROC-AUC:   {search.best_score_:.3f}")

# Evaluate tuned pipeline on untouched, vaulted test set
final_test_auc = search.score(X_test, y_test)
print(f"Vaulted Test Set ROC-AUC: {final_test_auc:.3f}")
```

Trust me bro benchmarks

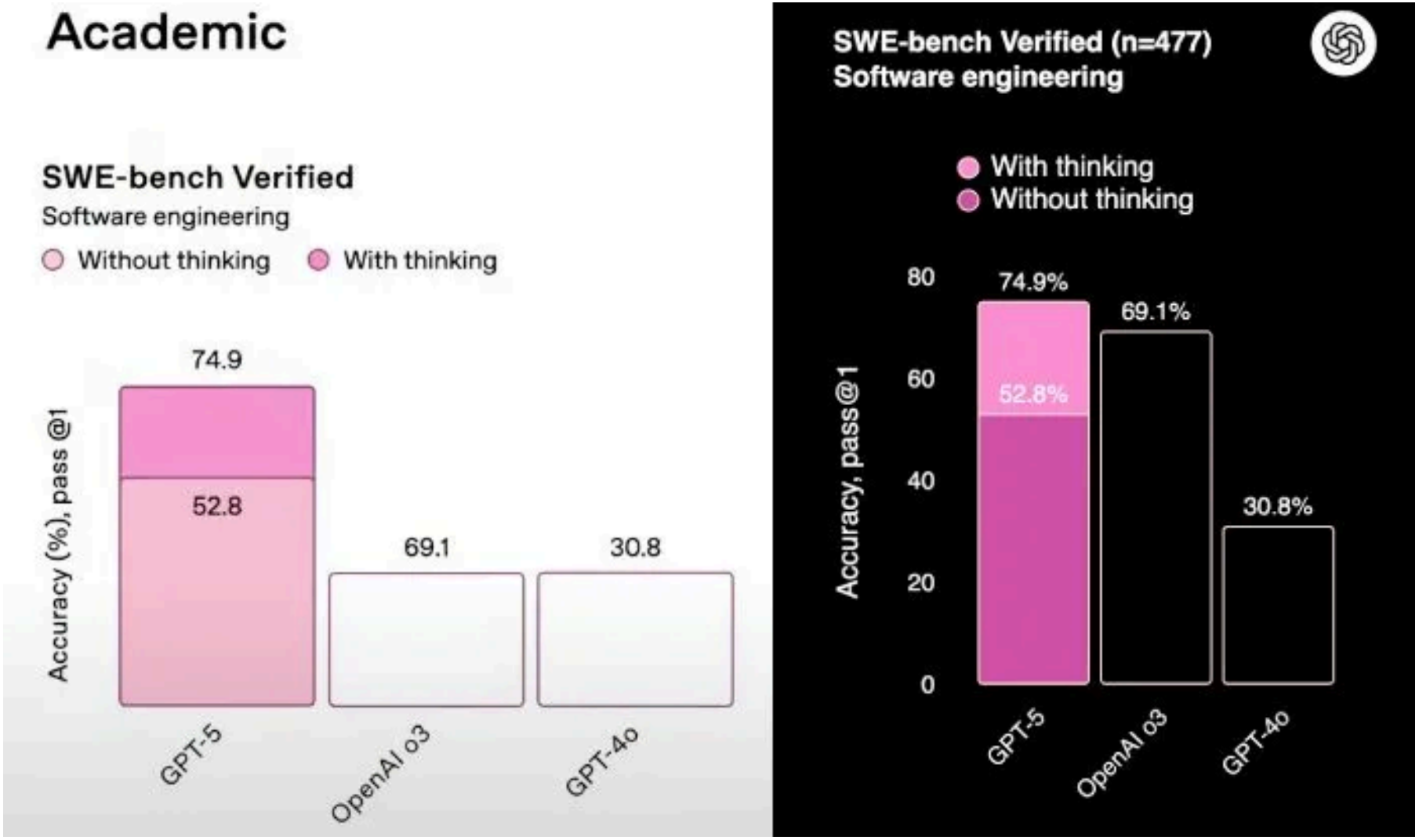
The Scenario

- **Model A (GPT-41):** Test Accuracy = 89.1%
- **Model B (GPT-42):** Test Accuracy = 89.8%

The Question

> *Did Model B uncover genuine intelligence, or did it just win the random seed lottery on this specific test partition?*

In AI and ML, we mostly nowadays highlight the best result we can find on a benchmark in **bold** and move on.



OpenAI's performance charts in the GPT-5 launch video are such a mess you have to think GPT-5 itself probably made them, and the company's attempted fixes raise even more questions.

PC Gamer, Jeremy Laird, August 12, 2025

Confidence Interval

The test set consists of N i.i.d. pairs $(x_i, y_i) \sim \mathcal{D}$. For each sample, the model prediction is $\hat{y}_i = f(x_i)$. So far so good.

Let's define:

$$X_i = \mathbb{I}(\hat{y}_i = y_i) = \begin{cases} 1 & \text{if correct} \\ 0 & \text{if incorrect} \end{cases}$$

Here, $X_i \sim \text{Bernoulli}(p)$, where $p = \mathbb{P}_{(x,y)\sim\mathcal{D}}(f(x) = y)$ is the unknown true population accuracy. We cannot access p, but we can **estimate** it using the test set.

$$X_i \sim \text{Bernoulli}(p) \implies \mathbb{E}[X_i] = p, \quad \text{Var}(X_i) = p(1-p)$$

2. Sample Mean & Central Limit Theorem

Test accuracy is the sample mean $\hat{p} = \frac{1}{N}\sum_{i=1}^{N} X_i$, so $\mathbb{E}[\hat{p}] = p$ and $\mathbf{Var}(\hat{p}) = \frac{p(1-p)}{N}$.
By the **Central Limit Theorem**, the standardised error converges to a standard normal:

$$Z = \frac{\hat{p} - p}{\sqrt{\frac{p(1-p)}{N}}} \to \mathcal{N}(0,1) \quad \text{as } N \to \infty$$

So we got a normal distribution for our test accuracy:

$$\hat{p} \dot{\sim} \mathcal{N}\left(p,\ \frac{p(1-p)}{N}\right)$$

But we don't know p...

> Sometimes it's okay not to know something, you can stop and enjoy other things or...

3. Plug-in Estimator

Plugging in $\hat{p}$ for p in the standard error (Slutsky's theorem) which guarantees that as $N \to \infty$ we still converge to $\mathcal{N}(0, 1)$. So we can write:

$$\mathrm{SE}(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$

- **What is the standard error?** It is the standard deviation of your estimate if you were to repeat the experiment many times.
- While standard deviation (σ) measures the variability of individual data points, standard error (SE) measures the variability of a summary statistic (like test accuracy $\hat{p}$).

Wald (Confidence) Interval

After plugging in $\hat{p}$ and solving for true p:

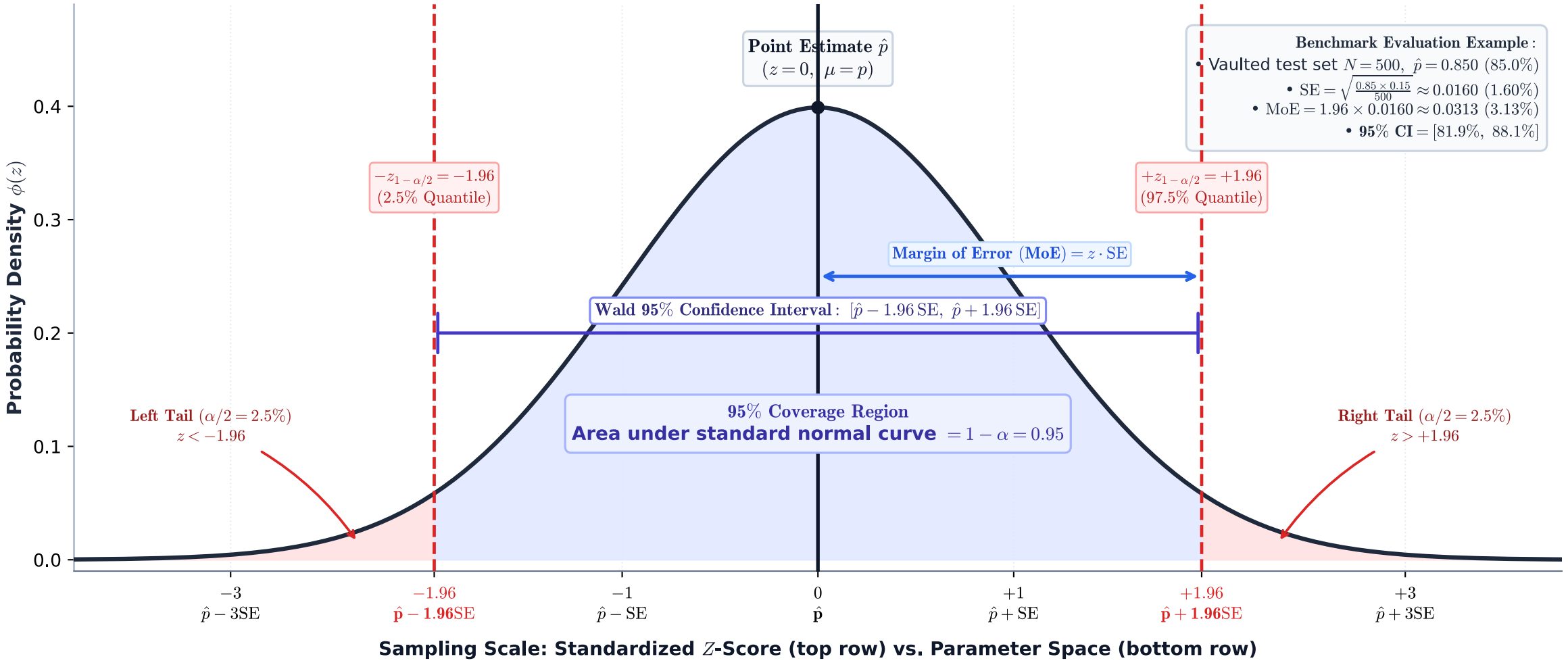
$$\mathbb{P}\left(-z_{1-\alpha/2} \leq \frac{\hat{p}-p}{\sqrt{\hat{p}(1-\hat{p})/N}} \leq z_{1-\alpha/2}\right) = 1-\alpha \implies p \in \hat{p} \pm z_{1-\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$

The **Wald Confidence Interval** (Normal approximation) at level $1-\alpha$:

$$\mathrm{CI}_{1-\alpha} = \hat{p} \pm z_{1-\alpha/2}\sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$

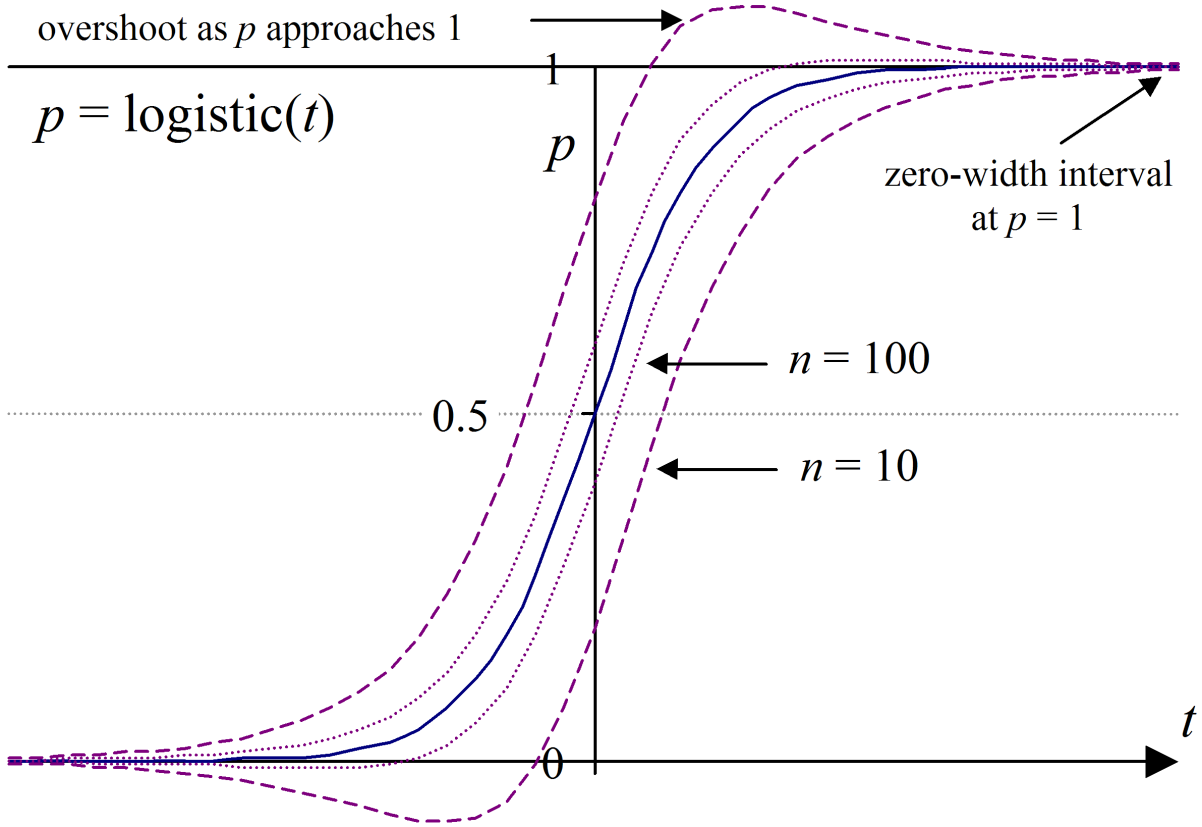
For a 95% confidence interval ($z_{0.975} \approx 1.96$):

$$\mathrm{CI}_{95\%} = \hat{p} \pm 1.96\sqrt{\frac{\hat{p}(1-\hat{p})}{N}}$$


Anatomy of the Wald Confidence Interval ($\mathcal{N}$-Approximation for Accuracy)
Point Estimate $\hat{p}$
$(z = 0,\ \mu = p)$
Benchmark Evaluation Example:
• Vaulted test set $N = 500,\ \hat{p} = 0.850\ (85.0\%)$
• $\mathrm{SE} = \sqrt{\frac{0.85 \times 0.15}{500}} \approx 0.0160\ (1.60\%)$
• $\mathrm{MoE} = 1.96 \times 0.0160 \approx 0.0313\ (3.13\%)$
• 95% CI $= [81.9\%,\ 88.1\%]$
$-z_{1-\alpha/2} = -1.96$
(2.5% Quantile)
$+z_{1-\alpha/2} = +1.96$
(97.5% Quantile)
Margin of Error (MoE) $= z \cdot \mathrm{SE}$
Wald 95% Confidence Interval: $[\hat{p} - 1.96\,\mathrm{SE},\ \hat{p} + 1.96\,\mathrm{SE}]$
95% Coverage Region
Area under standard normal curve $= 1 - \alpha = 0.95$
Left Tail $(\alpha/2 = 2.5\%)$
$z < -1.96$
Right Tail $(\alpha/2 = 2.5\%)$
$z > +1.96$
Probability Density $\phi(z)$
0.0
0.1
0.2
0.3
0.4
-3
$\hat{p} - 3\mathrm{SE}$
-1.96
$\hat{\mathbf{p}} - \mathbf{1.96}\mathrm{SE}$
-1
$\hat{p} - \mathrm{SE}$
0
$\hat{\mathbf{p}}$
$+1$
$\hat{p} + \mathrm{SE}$
$+1.96$
$\hat{\mathbf{p}} + \mathbf{1.96}\mathrm{SE}$
$+3$
$\hat{p} + 3\mathrm{SE}$
Sampling Scale: Standardized Z-Score (top row) vs. Parameter Space (bottom row)

Limitations of the Wald Interval

Can produce bounds outside $[0, 1]$ when $\hat{p} \approx 0$ or 1, or when N is small. If $\hat{p} = 1.0$, estimated variance collapses to 0. It also does not generalise to non-linear metrics like F_1 or AUROC.



Empirical Confidence Intervals

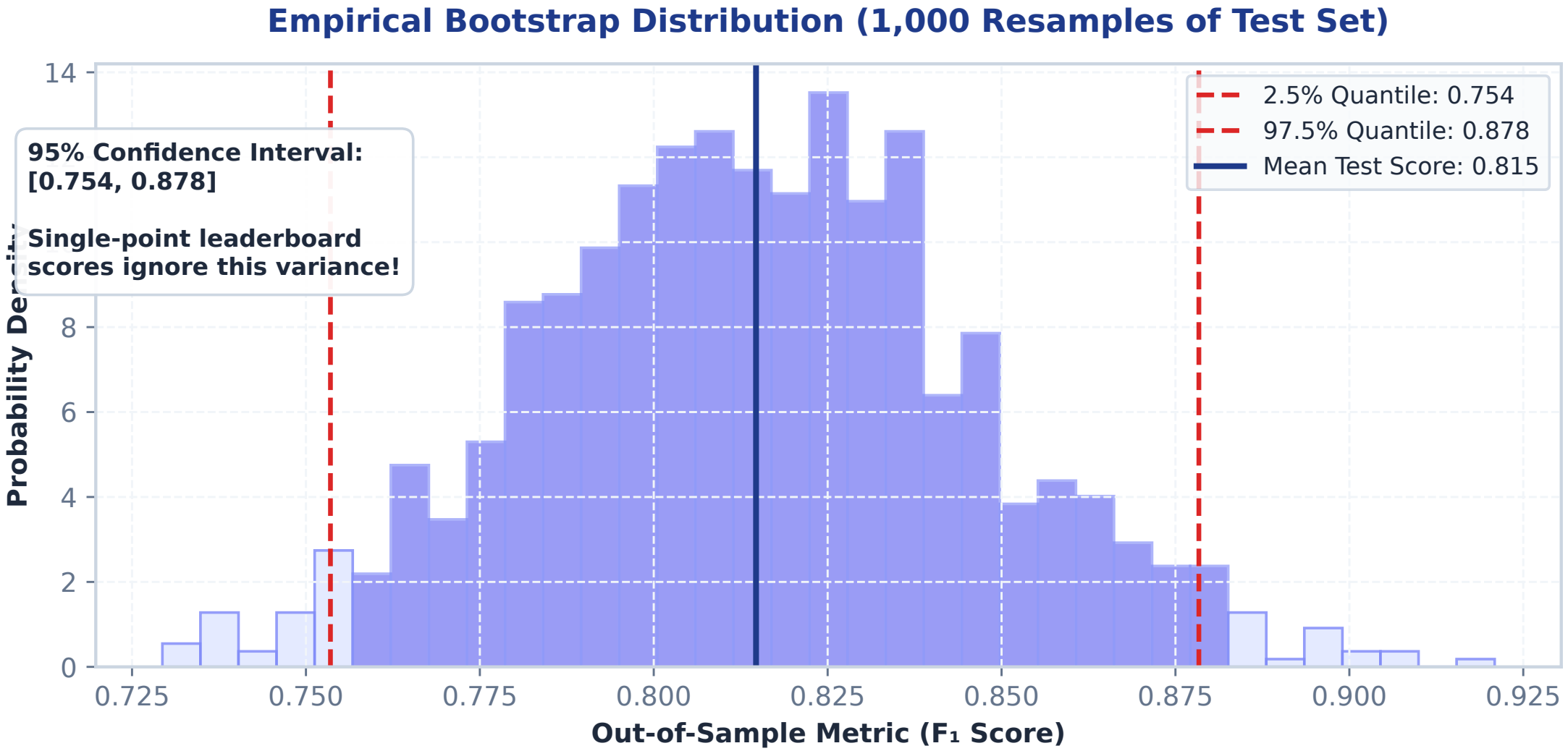
How do we quantify estimation uncertainty on our test metrics without making assumptions?

The Bootstrap Protocol

Given test set $S_{\text{test}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$:

1. For $b = 1, \ldots, B$ (e.g., $B = 1,000$ iterations):
   - Sample N records **with replacement** from S_{test}.
   - Compute metric $M^{(b)}$ (e.g., F_1 or AUROC).
2. Compute the 95% Confidence Interval directly from empirical quantiles:

$$\text{CI}_{95\%} = [q_{0.025}(M),\ q_{0.975}(M)]$$



Bootstrap intervals make zero assumptions about normality and accurately reflect test set sample variance.

Bootstrap Confidence Intervals in Python

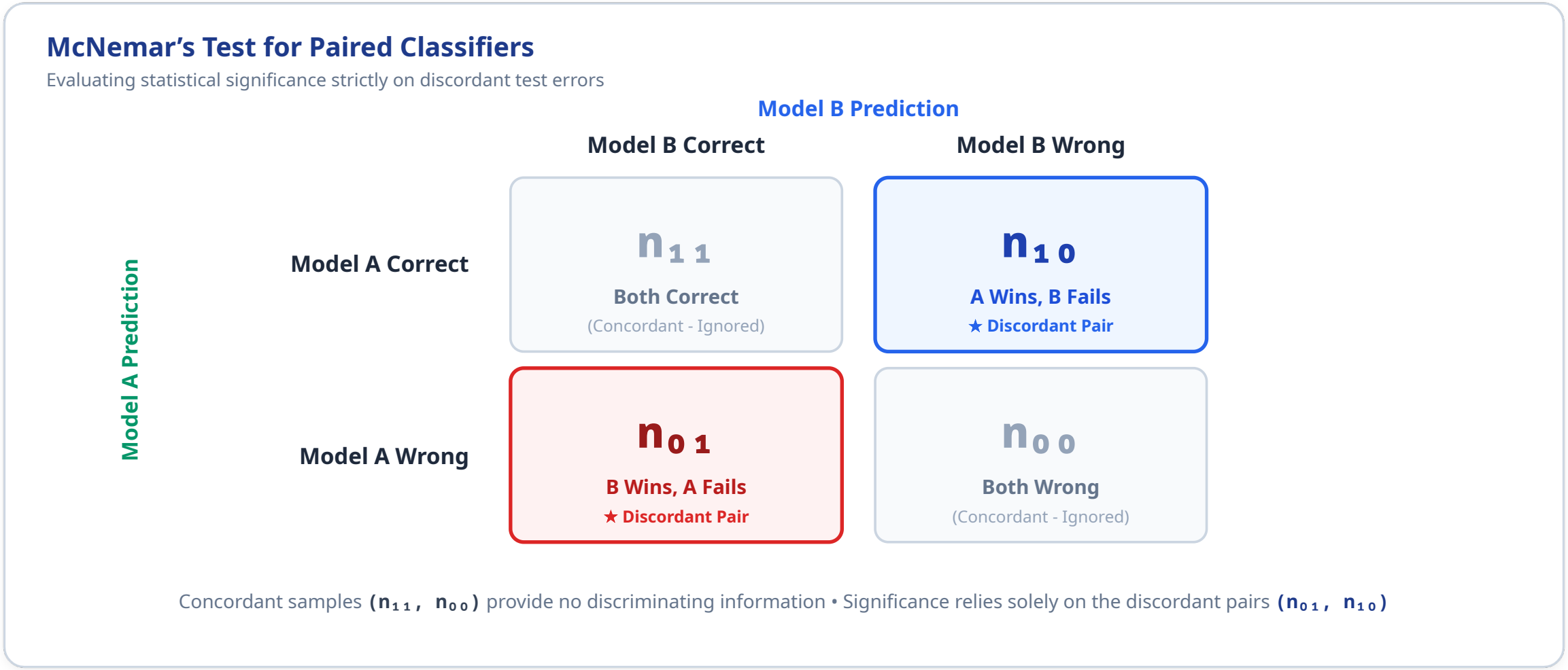
Compute non-parametric confidence intervals for paired evaluation metrics:

```
import numpy as np
from scipy.stats import bootstrap
from sklearn.metrics import f1_score

# Paired test labels and model predictions
data = (y_true, y_pred)

# Compute empirical 95% bootstrap confidence interval
res = bootstrap(
    data,
    statistic=...,
    paired=True,              # Resample paired (y_true_i, y_pred_i) together
    n_resamples=2000,
    confidence_level=0.95,
    method='percentile',
    random_state=42
)
print(f"95% CI: [{res.confidence_interval.low:.3f}, {res.confidence_interval.high:.3f}]")
```

McNemar’s Test for Paired Classifiers



McNemar's Test for Paired Classifiers

$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} \sim \chi^2(1)$$

where n_{01} (B wins) and n_{10} (A wins) are discordant error counts:

- Evaluate both models on the **identical test set** to form the paired contingency table.
- Under $H_0 : P(\text{A wins}) = P(\text{B wins})$, the statistic follows a χ^2 distribution with 1 df.
- If $\chi^2 > 3.84$ ($p < 0.05$), we reject the null hypothesis: Model B's edge is statistically significant!

> Just follow along for now.

Example

Setup: Comparing ResNet-50 (M_A) vs. Transformer (M_B) on $N = 2{,}000$ test images.

- M_A Accuracy: 83.5% (1,670 correct)
- M_B Accuracy: 85.5% (1,710 correct)

	M_B **Correct**	M_B **Incorrect**	**Marginal Totals**
M_A **Correct**	$a = 1{,}590$	$b = 80$	$1{,}670$
M_A **Incorrect**	$c = 120$	$d = 210$	330
Marginal Totals	$1{,}710$	290	$N = 2{,}000$

Discordant sample size: $b + c = 80 + 120 = 200 \geq 25$ (large-sample approximation valid).

$$\chi^2 = \frac{(|b - c| - 1)^2}{b + c} = \frac{(|80 - 120| - 1)^2}{80 + 120}$$

$$\chi^2 = \frac{(39)^2}{200} = \frac{1{,}521}{200} = \mathbf{7.605}$$

- **Critical Values:** $\chi^2_{\text{crit}}(0.05) = 3.841 \quad | \quad \chi^2_{\text{crit}}(0.01) = 6.635$
- **p-value:** $p \approx 0.0058$ the probability of observing a discordant split at least this extreme if both models were equally accurate (H_0).

$$\chi^2 = 7.605 > 6.635 \implies p < 0.01$$

Conclusion: Reject H_0. The 2.0% accuracy advantage of M_B is statistically significant. ■

```
from scipy.stats import binomtest, chi2

b, c = 80, 120
p_chi2  = chi2.sf((abs(b - c) - 1)**2 / (b + c), df=1)  # p = 0.0058
p_exact = binomtest(b, n=b + c, p=0.5).pvalue           # p = 0.0057
```

What the heck?

Okay, okay, okay. *I see the pain, I feel the pain, I am the pain.*

Step 1: FOCUS on the $n = b + c$ Disagreements

- Ignore the $1{,}800$ images where both models agree (a and d).
- Look **only** at the $n = b + c = 200$ images where one model is right.

Step 2: Null Hypothesis (H_0) = Fair Coin Flips

- If M_A and M_B are equally good, each disagreement is a **50/50 coin toss** ($p = 0.5$).
- Let b be the number of times M_A wins out of n disagreements:

$$b \sim \text{Binomial}(n, 0.5)$$

- Expected wins and variance under H_0:

$$\mu = np = \frac{n}{2} = 100 \qquad \sigma^2 = np(1-p) = \frac{n}{4} = 50$$

What the heck?

Step 3: Normal Approximation (Z-Score)

- Central Limit Theorem (CLT) $\rightarrow$ standardise b using $Z = \frac{b-\mu}{\sigma}$:

$$Z = \frac{b - \frac{n}{2}}{\sqrt{\frac{n}{4}}} = \frac{b - \frac{b+c}{2}}{\frac{\sqrt{b+c}}{2}} = \frac{b-c}{\sqrt{b+c}} \sim \mathcal{N}(0,1)$$

Step 4: Square Z

- The square of a standard normal variable is a $\chi^2(1)$ variable ($Z^2 \sim \chi^2_1$):

$$\chi^2 = Z^2 = \frac{(b-c)^2}{b+c}$$

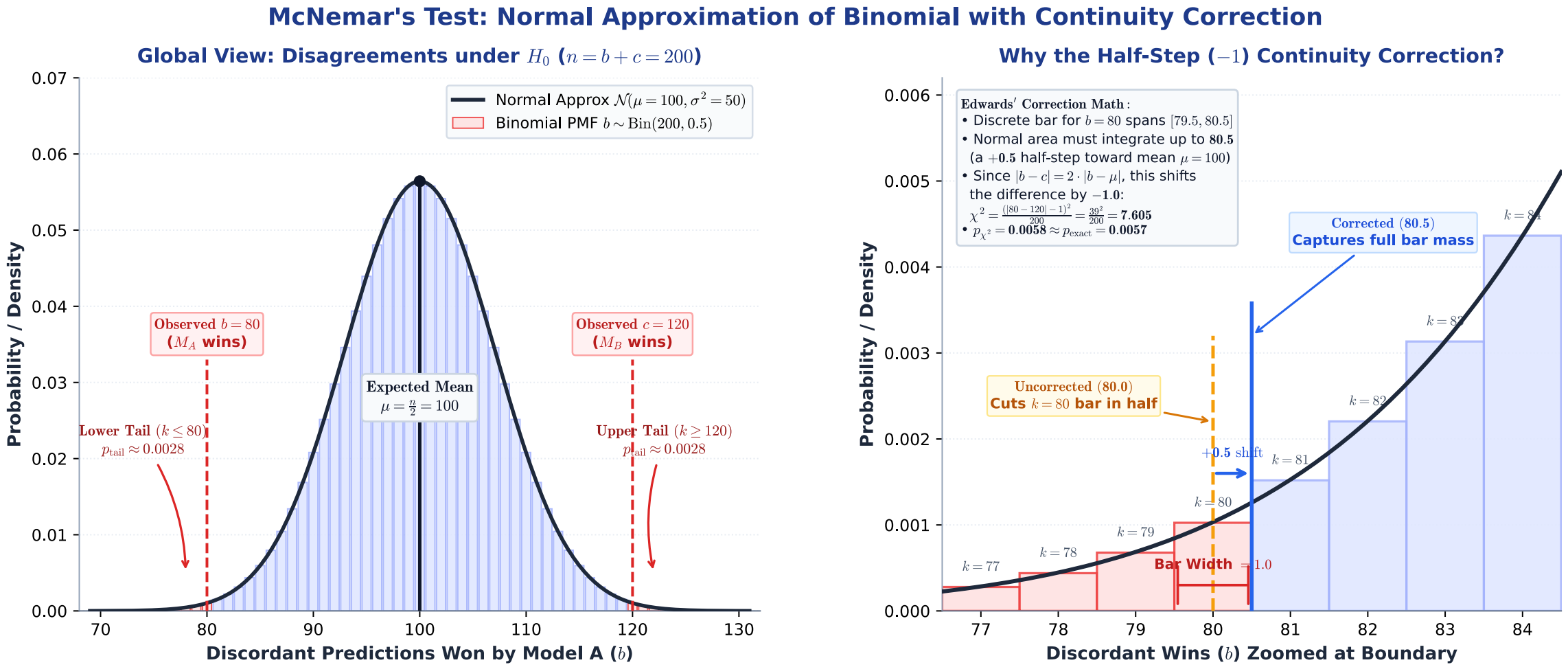
Step 5: Add Continuity Correction

- **Why the -1?** Because b and c are discrete integers, shifting 1 sample from b to c changes $|b - c|$ by 2. Subtracting 1 (Edwards' continuity correction) adjusts for the half-step on the continuous curve:

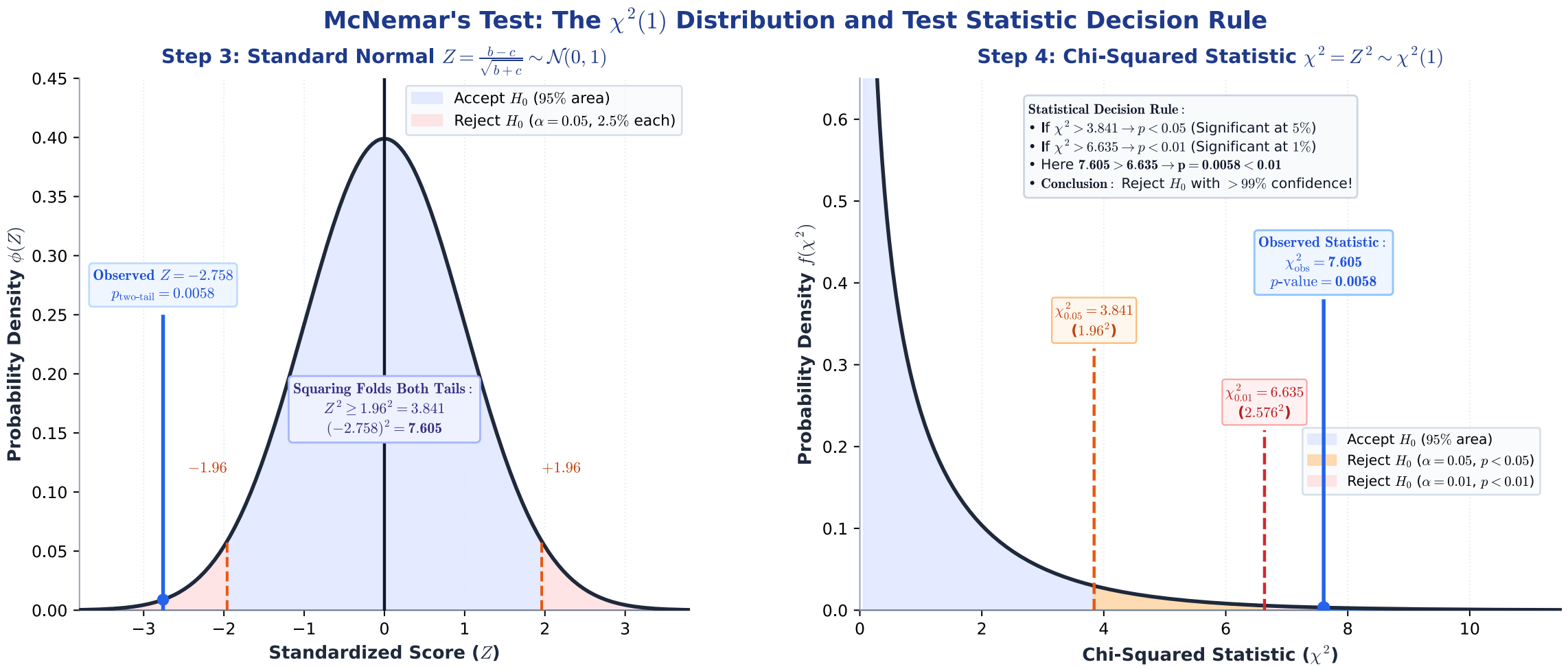
$$\chi^2 = \frac{(|n_{01} - n_{10}| - 1)^2}{n_{01} + n_{10}} = \frac{(|b - c| - 1)^2}{b + c} \sim \chi^2(1)$$

Dietterich TG. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. Neural Comput. 1998 Sep 15;10(7):1895-1923. doi: 10.1162/089976698300017197. PMID: 9744903.

McNemar’s Test: Continuity Correction



McNemar's Test: The χ^2 Distribution



Step 7. Deployment

Evaluation metrics mean nothing if the model cannot serve predictions within operational SLAs:

1. **Latency Budgets:**
   - **p50 (Median):** Typical user experience (e.g., $< 10\text{ms}$).
   - **p99 (Tail Latency):** 99th percentile (e.g., $< 45\text{ms}$).
2. **Throughput:**
   - Queries per second (QPS) per server core.
3. **Model Footprint & Edge Deployment:**
   - Memory footprint (RAM/VRAM)
   - Quantisation (FP32 $\rightarrow$ INT8/INT4)
   - Pruning non-essential weights

> Think cheap but worse LLMs being more popular.



Distribution Shift

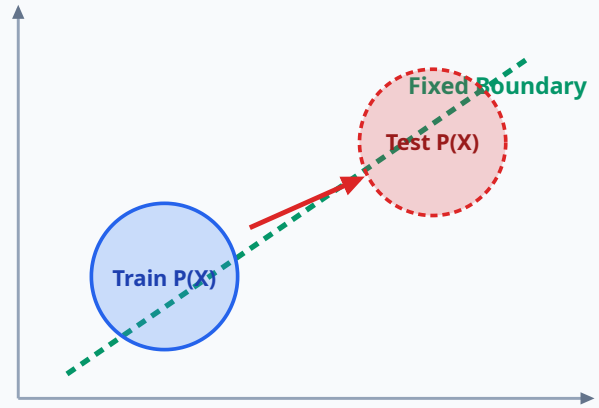
Taxonomy of Distributional Shifts

When the real world violates the stationary distribution assumption

1. Covariate Shift

`P(X) shifts | P(Y|X) fixed`

Inputs move; decision law stays same

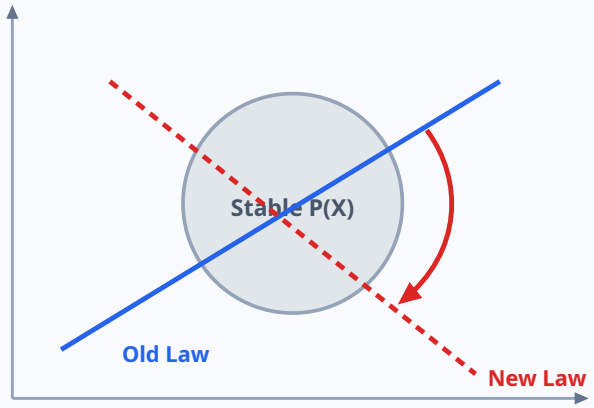


Example: Older patients visiting hospital

2. Concept Drift

`P(Y|X) shifts | P(X) fixed`

Data static; underlying relationship changes

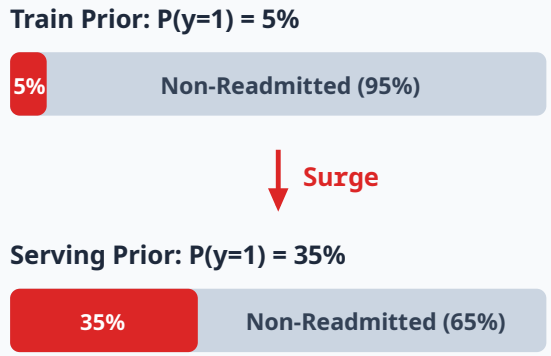


Example: New medical drug alters disease course

3. Prior Shift

`P(Y) shifts | P(X|Y) fixed`

Base rate changes; class conditional fixed



Example: Epidemic surge shifts prior base rate

Monitoring & Drift Detection

How do we detect silent model degradation before clinical or financial catastrophe strikes?

- **Data Integrity Alarms:** Missing value surge, unexpected categorical values.
- **Prediction Drift:** Tracking moving average of model outputs $\hat{y}$.
- **Performance Drift:** Computing delayed ground-truth feedback (30-day readmissions).
- **Automated Circuit Breakers:** Fallback to heuristic rules if prediction anomaly detected.

> I cannot emphasise how important distributional shift is. We train the model we have but the world moves on...

A good example you hear all the time are LLM knowledge cutoffs. That is distributional shift in action.

Where is my artificial brain?

I know, I know. We have been cooking some serious stats in here, but:

AI $\rightarrow$ ML $\rightarrow$ Numbers game

The Question That Sparks the Connectionist Renaissance

> *What if an artificial brain could **learn the representation mapping** $\phi_{\mathbf{W}}(\mathbf{x})$ automatically from raw data, layer by layer?*

Next Lecture: Part 4: Multi-Layer Perceptrons & Backpropagation!