

A Primer of Practical AI Concepts

A decision-first guide featuring 22 key mathematical and statistical Quick Cases for students preparing for AI careers.

Michael Ghilissen

Public Health Department

This collection is a quick-reference primer, not a course or a textbook. Each Quick Case is designed for rapid conceptual orientation: enough to recognize when a mathematical idea applies to a decision, to know what question to ask, and to identify the right tool to reach for next. Rigorous derivation, proof, and extended practice are left to dedicated coursework; the goal here is recognition and fluency, not mastery of the underlying machinery.

The worked examples are deliberately simple — chosen to make the reasoning visible in one pass, not to model the ambiguity of real data. That gap is intentional: a reference sheet's job is to build the pattern-recognition a practitioner brings to messier, real-world analysis, not to simulate that messiness itself.

The range of 22 concepts is a deliberate design choice, not a substitute for depth. Practicing AI developers and the professionals who work alongside them routinely encounter this full spread of situations — skewed averages, miscalibrated probabilities, unstable models, correlated features — and need to recognize which reasoning tool is relevant before they have time to specialize in any one of them.

The aim is broad situational coverage rather than deep technical expertise. This is also why the material is written for Public Health students who will increasingly design, commission, or oversee AI-supported operations — such as triage algorithms, resource allocation, and risk-scoring systems — rather than for students training to deliver clinical care. The target competency is applied AI literacy: knowing what to ask of a model or a vendor and where common failure points lie. It is not intended to prepare students to practice medicine.

1. Probability — Decision Under Uncertainty

1. Expectation

Decision hook: Which uncertain AI strategy should the company choose?

Scenario: A retailer must choose among three demand-forecasting policies. One has the highest upside but also costly downside scenarios. Executives ask for one planning number.

Decision question: What single summary should drive the decision, and when would the mean be misleading?

Mathematical engine: Expected value as a probability-weighted average; expected utility/cost intuition; mean versus median under skew.

Practical implementation: Compute expected profit for each policy from a small outcome table; then change one rare-event probability and reassess.

Numerical example: Strategy A yields €100k with probability 0.6 and €20k with probability 0.4, so $E[A] = 0.6 \times 100 + 0.4 \times 20 = €68k$. Strategy B yields €75k for sure, so $E[B] = €75k$. On expected profit alone, choose B. But if A instead yields €130k with probability 0.6 and €20k with probability 0.4, $E[A]$ becomes €86k and the decision reverses.

Common pitfall: Treating the expected value as a guaranteed outcome; ignoring tail losses or asymmetric costs.

2. Variance / Standard Deviation

Decision hook: Two models have the same average error. Which is safer to deploy?

Scenario: Two forecasting models have identical mean absolute performance over 12 weeks, but one swings sharply across regions and weeks.

Decision question: Which model should be deployed, and what operational safeguards does high variability require?

Mathematical engine: Variance and standard deviation as measures of instability around an average.

Practical implementation: Calculate mean and standard deviation of weekly errors; compare two models with equal means.

Numerical example: Model X has weekly errors [8, 8, 8, 8], while Model Y has [2, 6, 10, 14]. Both have mean error 8, but $SD(X)=0$ and $SD(Y) \approx 4.47$. If operational reliability matters, X is safer despite identical average error.

Common pitfall: Choosing only on average performance; failing to distinguish model instability from irreducible noise.

3. Conditional Probability

Decision hook: A diagnostic AI flags a rare event. How seriously should the flag be taken?

Scenario: A hospital model has high sensitivity and specificity, but the condition it detects is rare. A clinician sees a positive alert.

Decision question: What is the probability that the patient truly has the condition given the positive alert, and should the downstream process change?

Mathematical engine: $P(A|B)$, base rates, confusion-matrix reasoning.

Practical implementation: Build a 10,000-patient frequency table and calculate the positive predictive probability.

Numerical example: Disease prevalence is 1%, sensitivity 95%, specificity 95%. Among 10,000 patients: 100 are diseased, about 95 test positive; 9,900 are healthy, about 495 test positive falsely. Thus $P(\text{disease} | \text{positive}) = 95 / (95 + 495) \approx 16.1\%$, not 95%.

Common pitfall: Confusing $P(\text{positive} | \text{disease})$ with $P(\text{disease} | \text{positive})$; neglecting prevalence.

4. Bayes' Rule

Decision hook: New evidence arrives. Should the decision change?

Scenario: A fraud team begins with a low prior probability for a transaction. A device signal and unusual purchase pattern arrive sequentially.

Decision question: How should beliefs be updated, and at what posterior probability should the team intervene?

Mathematical engine: Prior $\rightarrow$ likelihood $\rightarrow$ posterior; evidence updates rather than replaces prior information.

Practical implementation: Perform a simple Bayesian update using frequencies or odds.

Numerical example: A transaction has a 2% prior fraud probability. A signal occurs in 80% of frauds but 10% of legitimate transactions. Posterior fraud probability = $(0.80 \times 0.02) / [(0.80 \times 0.02) + (0.10 \times 0.98)] \approx 14.0\%$. The signal raises risk sevenfold, but the posterior is still only 14%.

Common pitfall: Starting from an implicit 50/50 prior; double-counting correlated evidence.

5. Common Probability Distributions

Decision hook: Which probability model matches the process generating the data?

Scenario: An AI operations team must model clicks, daily incident counts, and time between failures to simulate system load.

Decision question: Which distribution family fits each process, and what operational error follows from a poor choice?

Mathematical engine: Bernoulli/binomial, categorical, Gaussian, Poisson, exponential, uniform and heavy-tailed intuition.

Practical implementation: Match short datasets/process descriptions to candidate distributions and simulate a few outcomes.

Numerical example: If each of 1,000 visitors independently clicks with probability 0.03, clicks can be modeled Binomial($n=1000, p=0.03$), with expected clicks $np=30$. If incidents arrive at an average of 4 per hour, a Poisson($\lambda=4$) model gives $P(0 \text{ incidents}) = e^{-4} \approx 1.8\%$. If mean time between failures is 10 hours under an exponential model, the rate is $\lambda=0.1/\text{hour}$.

Common pitfall: Selecting a distribution because its histogram 'looks right' without checking the generative story.

6. Sampling & Sampling Distributions

Decision hook: Can results from these users be generalized to everyone?

Scenario: A model was evaluated on 800 volunteer users from two cities. Product leadership wants to claim a population-wide improvement.

Decision question: What can legitimately be inferred, and is the sample adequate for the deployment decision?

Mathematical engine: Sample versus population; random sampling; sampling variability; standard error.

Practical implementation: Compare estimates from repeated samples or compute a simple margin of error.

Numerical example: In a random sample of $n=800$ users, 60% prefer the new model. Approximate SE = $\sqrt{[0.6 \times 0.4 / 800]} \approx 0.0173$, giving a 95% margin of error $\approx 1.96 \times 0.0173 = 3.4$ percentage points. A random sample supports roughly 56.6%–63.4%; a volunteer sample may still be biased regardless of n .

Common pitfall: Equating large sample size with representativeness; ignoring selection bias.

7. Law of Large Numbers & Central Limit Theorem

Decision hook: When does an average become trustworthy?

Scenario: A platform's transaction values are highly skewed. Managers want to use average daily value for staffing and financial planning.

Decision question: With enough observations, what stabilizes, what becomes approximately normal, and what uncertainty remains?

Mathematical engine: LLN for stabilization of sample averages; CLT for the distribution of standardized averages.

Practical implementation: Simulate repeated sample means from a skewed distribution and inspect their spread.

Numerical example: Suppose transaction value has mean €50 and SD €100. For $n=25$ transactions, the sample-mean SE is $100/\sqrt{25}=\text{€}20$; for $n=400$, it falls to $100/\sqrt{400}=\text{€}5$. LLN explains stabilization around €50, while the CLT says the distribution of many such sample means is approximately normal under suitable conditions.

Common pitfall: Believing the raw data become normal; assuming large n fixes dependence or biased sampling.

8. Bootstrap

Decision hook: How can we quantify uncertainty when the analytic formula is unavailable?

Scenario: A team has only 50 labeled examples and a complex evaluation metric. Management asks whether the new model is truly better.

Decision question: How wide is the uncertainty interval around the observed performance difference, and does it support deployment?

Mathematical engine: Resampling with replacement; bootstrap distribution; percentile confidence interval intuition.

Practical implementation: Bootstrap the metric or difference in metrics in Python/spreadsheet and report a 95% interval.

Numerical example: Observed accuracies are 82% for Model A and 76% for Model B on 50 cases, a +6-point difference. After 2,000 bootstrap resamples, suppose the 2.5th and 97.5th percentiles of the accuracy difference are -4 and +16 points. Because the interval includes 0, the sample does not provide strong evidence that A is reliably better.

Common pitfall: Treating bootstrap samples as new independent data; ignoring dependence or severe sample non-representativeness.

2. Calculus & Optimization — How Models Learn

9. Derivatives & Partial Derivatives

Decision hook: Which small parameter change matters most?

Scenario: A model's loss changes when individual parameters are nudged. The team needs to decide where local tuning effort is most useful.

Decision question: Which parameter has the strongest local effect, and what does a near-zero derivative actually tell us?

Mathematical engine: Derivative as local sensitivity; partial derivative holding other variables fixed; local linear approximation.

Practical implementation: Estimate slopes numerically by perturbing one parameter at a time.

Numerical example: For loss $L(w) = (w-3)^2$, $dL/dw = 2(w-3)$. At $w=5$, the derivative is 4: increasing w slightly raises loss, so moving w downward should help. For $L(w_1, w_2) = w_1^2 + 4w_2^2$ at $(2, 1)$, partial derivatives are 4 and 8, so locally the loss is twice as sensitive per unit change to w_2 .

Common pitfall: Interpreting a local derivative as a global causal effect or global feature importance.

10. Gradient

Decision hook: In which direction should all parameters move?

Scenario: A neural network has thousands of weights. After a training batch, engineers need one coordinated update direction.

Decision question: What vector points downhill, and what do its direction and magnitude imply?

Mathematical engine: Gradient as the vector of partial derivatives; steepest local ascent/descent intuition.

Practical implementation: Calculate a two-parameter gradient and take one update step.

Numerical example: For $L(w_1, w_2) = w_1^2 + 4w_2^2$ at $(2, 1)$, $\nabla L = (4, 8)$. The steepest local descent direction is $-(4, 8)$. With learning rate 0.1, one update gives $(w_1, w_2) = (2, 1) - 0.1(4, 8) = (1.6, 0.2)$, reducing loss from 8 to 2.72.

Common pitfall: Reading gradient magnitude as business importance; ignoring scaling across parameters.

11. Chain Rule

Decision hook: How does an output error reach an early layer?

Scenario: A deep network's output loss changes, but the team must understand why early layers barely update.

Decision question: How is sensitivity propagated through composed functions, and what does vanishing/exploding gradient behavior imply?

Mathematical engine: Chain rule; composition; backpropagation as repeated local derivatives.

Practical implementation: Manually differentiate a tiny two-layer computation graph.

Numerical example: Let $h=2x$, $y=h^2$, and loss $L=(y-10)^2$. At $x=1$: $h=2$, $y=4$. $dL/dy=2(4-10)=-12$, $dy/dh=2h=4$, $dh/dx=2$. Therefore $dL/dx=(-12) \times 4 \times 2=-96$. The output error is transmitted backward through the product of local derivatives.

Common pitfall: Memorizing backprop equations without seeing multiplication of local sensitivities.

12. Gradient Descent & Learning Rate

Decision hook: Training is stuck or unstable. What should change?

Scenario: Training loss oscillates at one learning rate and barely moves at another. The team must choose the next training configuration.

Decision question: Should the learning rate increase, decrease, or be scheduled, and when should training stop?

Mathematical engine: Iterative descent; step size; stochastic/mini-batch updates; convergence versus divergence.

Practical implementation: Run gradient descent on a simple quadratic with three learning rates and compare trajectories.

Numerical example: For $L(w)=(w-3)^2$ starting at $w=0$, gradient $=2(w-3)=-6$. With $\eta=0.1$, w becomes 0.6 and loss falls from 9 to 5.76. With $\eta=1.1$, w becomes 6.6 and loss rises to 12.96, illustrating an overly aggressive step. With $\eta=0.01$, w becomes only 0.06, so learning is stable but slow.

Common pitfall: Assuming lower training loss always means better generalization; changing optimizers before diagnosing step size.

3. Linear Algebra — Geometry of Data & Models

13. Vectors, Dot Products & Norms

Decision hook: Which item is most similar to this query or user?

Scenario: A retrieval system represents a query and candidate documents as embeddings. Two distance measures rank candidates differently.

Decision question: Which similarity/distance should drive retrieval, and when does the choice of norm change the result?

Mathematical engine: Vectors as feature representations; dot product; cosine similarity; L1/L2 norms and distance.

Practical implementation: Compute dot product, cosine similarity, L1 and L2 distances for three short vectors.

Numerical example: Query $q=(1,1)$, document $A=(2,2)$, document $B=(1,0)$. Dot products are $q \cdot A=4$ and $q \cdot B=1$, so A ranks higher. Cosine similarities are 1.00 for A and 0.707 for B. L2 distances from q are $\sqrt{2} \approx 1.41$ to A and 1 to B, so Euclidean distance instead ranks B closer: the metric changes the retrieval decision.

Common pitfall: Using raw dot products when vector magnitude is irrelevant; assuming all distance metrics encode the same notion of similarity.

14. Matrices as Linear Transformations

Decision hook: What is this layer doing to the data?

Scenario: A model applies a weight matrix to thousands of feature vectors. Small input changes sometimes create large output changes.

Decision question: How does the matrix transform the feature space, and when should numerical sensitivity worry the team?

Mathematical engine: Matrix multiplication as simultaneous linear transformation; scaling, rotation/shearing intuition; conditioning.

Practical implementation: Apply a 2×2 matrix to several points and visualize/inspect the transformed coordinates.

Numerical example: With $A = \begin{bmatrix} 2 & 0 \\ 0 & 0.5 \end{bmatrix}$, input $x = (3, 4)$ becomes $Ax = (6, 2)$: the transformation stretches the first axis by 2 and compresses the second by 0.5. Two nearby inputs $(3, 4)$ and $(3.1, 4)$ map to $(6, 2)$ and $(6.2, 2)$, making the directional scaling explicit.

Common pitfall: Seeing matrices only as tables of numbers; overlooking scale and conditioning.

15. Projections

Decision hook: What information is lost when data are forced into a smaller subspace?

Scenario: A system compresses high-dimensional observations before reconstruction. The residual is small overall but large for a minority pattern.

Decision question: Is the projection acceptable for the downstream task, and what does the residual tell us?

Mathematical engine: Orthogonal projection; subspaces; reconstruction and residual.

Practical implementation: Project a vector onto a line/subspace and calculate the residual norm.

Numerical example: Project $x = (3, 4)$ onto the x-axis. The projection is $(3, 0)$, and the residual is $(0, 4)$ with norm 4. If the downstream task only needs horizontal position, the lost component may be acceptable; if the vertical component carries risk information, the same projection is unacceptable.

Common pitfall: Assuming small average reconstruction error means no important information was lost.

16. Eigenvectors, Eigenvalues & PCA

Decision hook: Can 500 variables be reduced to 20 without destroying useful information?

Scenario: A production pipeline is slow and noisy because it carries hundreds of correlated features.

Decision question: Which directions should be retained, how many components are enough, and what information might PCA discard?

Mathematical engine: Eigenvectors/eigenvalues as important directions and associated variation; PCA; explained variance.

Practical implementation: Run PCA on a small feature matrix and compare 2, 5, and 10-component reconstruction/performance.

Numerical example: Suppose PCA eigenvalues are $[6, 3, 1]$. Total variance is 10, so PC1 explains 60%, PC1+PC2 explain 90%, and all three explain 100%. If the production requirement is at least 85% explained variance, retain two components; a 95% requirement forces all three.

Common pitfall: Equating maximum variance with maximum predictive or business relevance; fitting PCA before the train/test split.

4. Statistics & Machine Learning — From Fit to Reliable Decisions

17. Loss Functions & Residuals

Decision hook: Why does a model with fewer mistakes still create worse business outcomes?

Scenario: Two price-prediction models have similar error counts, but one makes a few catastrophic misses and shows systematic residual patterns.

Decision question: Should the team change MSE/MAE/Huber or features/model specification?

Mathematical engine: Loss as the optimization objective; squared versus absolute penalties; residuals as diagnostic evidence.

Practical implementation: Calculate MSE and MAE on the same predictions and plot/inspect residuals by subgroup or predicted value.

Numerical example: True values are [100,100,100] and predictions are [90,100,140], giving residuals [10,0,-40]. $MAE = (10+0+40)/3 = 16.7$, while $MSE = (100+0+1600)/3 = 566.7$. MSE penalizes the single large miss much more strongly, so it is preferable when catastrophic errors are disproportionately costly.

Common pitfall: Choosing a loss by convention rather than decision cost; ignoring structured residual patterns.

18. Bias–Variance Trade-off

Decision hook: The model fails on new data. What should be tried first?

Scenario: A complex model excels on training data but degrades on validation data, while a simpler model is consistently mediocre.

Decision question: Is the dominant problem high bias or high variance, and should the team add data, simplify, regularize, or improve features?

Mathematical engine: Underfitting versus overfitting; approximation error versus sensitivity to training data.

Practical implementation: Compare train/validation learning curves for models of different complexity.

Numerical example: Model A has training error 4% and validation error 18%; Model B has 14% training and 16% validation. A's 14-point gap suggests high variance/overfitting, whereas B's consistently higher errors suggest higher bias. For A, try stronger regularization or more data; for B, richer features or a more flexible model may help.

Common pitfall: Using 'bias' here as demographic bias; assuming more features always reduce bias safely.

19. Regularization

Decision hook: How much model flexibility should be sacrificed for generalization?

Scenario: A model overfits. L1, L2, dropout and early stopping each improve validation performance differently.

Decision question: Which regularizer matches the failure mode, and how strong should it be?

Mathematical engine: Penalized objectives; L1 sparsity; L2 shrinkage; dropout/early stopping intuition.

Practical implementation: Fit a small regression/classifier across regularization strengths and inspect train/validation performance.

Numerical example: A regression model has coefficients [8,5,0.8,0.2]. With L1 regularization, a tuned solution might become [6.5,3.8,0,0], creating sparsity. With L2, it might become [6.7,4.1,0.6,0.15], shrinking all weights without forcing most to zero. If validation error falls from 12% to 9% and then rises to 11% as λ increases, choose the intermediate strength.

Common pitfall: Tuning on the test set; interpreting regularization solely as a training-speed technique.

20. Uncertainty & Calibration

Decision hook: When a model says 80%, does 80% really mean 8 out of 10?

Scenario: A risk model outputs probabilities used to allocate scarce interventions. Predictions around 0.8 are correct only about 0.65 of the time.

Decision question: Are the probabilities reliable enough for threshold decisions, and what should change if the model is overconfident?

Mathematical engine: Predictive uncertainty; calibration; reliability curves; distinction between ranking and probability quality.

Practical implementation: Bin predictions into probability ranges and compare predicted versus observed frequencies.

Numerical example: Among 100 cases assigned predicted risk near 0.80, only 65 actually experience the event. The empirical frequency is 65%, so the model is overconfident by about 15 percentage points in that bin. A threshold policy based on '80% risk' should therefore not be trusted without recalibration.

Common pitfall: Confusing high discrimination/AUC with good calibration; interpreting confidence as certainty.

21. Performance Metrics & Business Goals

Decision hook: Accuracy is 99%, yet the system is failing. Which metric matters?

Scenario: A fraud detector has excellent accuracy because fraud is rare, but it misses many costly cases and creates expensive false alerts.

Decision question: Should the team optimize precision, recall, F1, AUC, expected cost, or a thresholded business objective?

Mathematical engine: Confusion matrix; precision/recall; false-positive/false-negative costs; threshold dependence.

Practical implementation: Calculate metrics at several thresholds and attach simple monetary costs to errors.

Numerical example: Out of 10,000 transactions, 100 are fraud. A detector catches 80 frauds (TP=80), misses 20 (FN=20), and falsely flags 180 legitimate transactions (FP=180).
 $\text{Accuracy} = (80 + 9,720) / 10,000 = 98\%$; $\text{precision} = 80 / 260 \approx 30.8\%$; $\text{recall} = 80 / 100 = 80\%$. If a missed fraud costs €1,000 and a false alert €20, expected error cost is $20 \times 1,000 + 180 \times 20 = €23,600$, which may be more decision-relevant than accuracy.

Common pitfall: Optimizing a metric detached from operational costs; reporting one threshold-free score as the whole deployment decision.

5. Logic & Clear Reasoning — Avoiding Subtle Failures

22. Implication, Necessary/Sufficient Conditions & Quantifiers

Decision hook: Does the evidence actually support the stakeholder's claim?

Scenario: A vendor says, 'If model score > 0.8, the customer will convert' and 'all high-risk cases are caught.' Internal data contains counterexamples.

Decision question: Which claims are logically valid, what is necessary versus sufficient, and what counterexample should change the decision rule?

Mathematical engine: Implication and converse; necessary/sufficient conditions; universal/existential quantifiers; sets/intersections; conditional reasoning.

Practical implementation: Translate three business claims into symbolic or set notation and test them against a small table of cases.

Numerical example: A rule states: 'If score > 0.8, the customer converts.' In 100 customers with score > 0.8, 72 convert and 28 do not. One non-converter is enough to falsify the universal implication. If all 60

actual high-risk cases have score >0.8 , then score >0.8 may be necessary for observed high risk in this sample, but it is not sufficient if many score >0.8 without being high risk.

Common pitfall: Reversing an implication; treating high accuracy as sufficient for deployment; ignoring a single counterexample to a universal claim.

Compact Map

Area	Practical concepts/methods	Typical Quick Case decision
Sampling & statistical reasoning	Sampling; expectation/variance; LLN/CLT; bootstrap	We trained on 800 patients. Can we trust the observed improvement?
Convergence & optimization	Derivatives; gradients; chain rule; gradient descent; learning rate	The loss barely changes—or explodes. What should we change?
Loss & model diagnosis	Loss functions; residuals; bias–variance; regularization	Two models make similar mistakes. Why does one generalize better?
Uncertainty & prediction	Conditional probability; Bayes; calibration; expected value	The model says 80%. What does 80% actually mean?
Performance decisions	Confusion matrix; precision/recall; false-positive/false-negative costs	Which model is better when missing a positive case is costly?
Probability models	Bernoulli/binomial; Gaussian; Poisson; exponential; heavy tails	What distribution reasonably describes these outcomes?
Geometry of data	Vectors; dot products; norms; matrices; projections; PCA	Which direction preserves most useful information?
Logic	Implication; necessary/sufficient; quantifiers; sets	Does high confidence imply the prediction is correct?