# Parameter Estimation, Model Evaluation and Ensemble Methods

## Unit 5: MLE, Confidence Intervals, Validation, ROC and Ensembles

Dr. Gopal Chandra Jana

Dept. of CSE, Sharda University
Machine Learning (CSE473) — Unit 5

July 23, 2026

Course Page:
https://www.gcjana.in/courses/shardauniversity/2601/machine-learning/

# Outline

# Learning Outcomes of Unit 5

## After this unit, a student will be able to...

1. **Define** point estimation and evaluate estimators by bias, variance and MSE.
2. **Derive** maximum-likelihood estimators and compute them numerically.
3. **Explain** unbiasedness and prove why $s^2$ divides by $n-1$.
4. **Construct** confidence intervals for one mean, two means and a variance.
5. **Compare** holdout, $k$-fold, LOOCV, subsampling and bootstrap validation.
6. **Build** a confusion matrix, ROC curve and compute AUC two ways.
7. **Explain** ensemble theory and apply voting, bagging, boosting and stacking.

## Mapping to the Syllabus

**A.** Point estimation, MLE, unbiasedness, confidence intervals. **B.** Validation methods and the ROC curve. **C.** Ensemble theory, size, voting/averaging, bagging, boosting, stacking.

# The Parameter Estimation Problem

## Setting

We observe a sample $X_1, \ldots, X_n$ drawn i.i.d. from a distribution $f(x;\theta)$ whose **parameter $\theta$ is unknown**. We wish to infer $\theta$.

## Two kinds of estimate

- **Point estimate**: a single value $\hat{\theta}$ — e.g. "the mean is $25.3$".
- **Interval estimate**: a range with a stated confidence — e.g. "the mean lies in $(24.0, 26.0)$ with $95\%$ confidence".

## Why this belongs in an ML course

Fitting *any* model is parameter estimation: linear-regression weights, logistic coefficients, Gaussian mixtures — all are $\hat{\theta}$ obtained from data.

## Estimator vs. estimate

- An **estimator** $\hat{\theta} = g(X_1, \ldots, X_n)$ is a *function* of the sample — hence a **random variable**.
- An **estimate** is the *number* it produces for one particular sample.

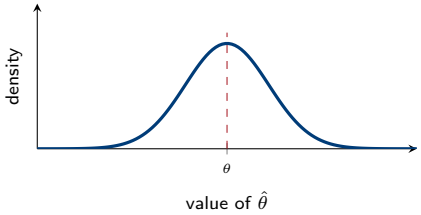



Figure: The *sampling distribution* of an estimator — different samples give different estimates.

# Properties of a Good Estimator

## Bias, variance and mean squared error

$$\text{Bias}(\hat{\theta}) = \mathbb{E}[\hat{\theta}] - \theta, \qquad \text{Var}(\hat{\theta}) = \mathbb{E}\big[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2\big]$$

$$\boxed{\text{MSE}(\hat{\theta}) = \mathbb{E}\big[(\hat{\theta} - \theta)^2\big] = \text{Var}(\hat{\theta}) + \big[\text{Bias}(\hat{\theta})\big]^2}$$

## Desirable properties

- **Unbiased**: $\mathbb{E}[\hat{\theta}] = \theta$ — correct *on average*.
- **Consistent**: $\hat{\theta} \xrightarrow{p} \theta$ as $n \to \infty$.
- **Efficient**: smallest variance among unbiased estimators (attains the Cramér–Rao bound).
- **Sufficient**: uses all the information in the sample about $\theta$.

## Cramér–Rao lower bound

For any unbiased $\hat{\theta}$,

$$\text{Var}(\hat{\theta}) \;\geq\; \frac{1}{n\, I(\theta)}, \qquad I(\theta) = -\mathbb{E}\left[\frac{\partial^2 \ln f(X;\theta)}{\partial \theta^2}\right]$$

where $I(\theta)$ is the **Fisher information**.

## The same trade-off as in Unit 2

A *slightly biased* estimator with much smaller variance can have **lower MSE** than the unbiased one — exactly the logic behind Ridge regression.

# Numerical Example — Bias, Variance and MSE

## Two competing estimators of a mean $\theta = 10$

Estimator $A$ is unbiased with $\mathrm{Var} = 4$. Estimator $B$ shrinks toward zero: $\mathbb{E}[B] = 9$ (bias $= -1$) but $\mathrm{Var} = 1$.

## Compute the MSE

$$\mathrm{MSE}(A) = \mathrm{Var} + \mathrm{Bias}^2 = 4 + 0^2 = 4.0$$

$$\mathrm{MSE}(B) = 1 + (-1)^2 = 1 + 1 = 2.0$$

## Conclusion

Although $B$ is **biased**, its MSE is **half** that of the unbiased estimator. On a squared-error criterion, $B$ is the better choice.

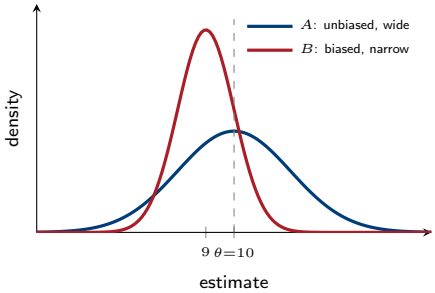



Figure: $B$ misses the target slightly but far more *consistently*.

# Maximum Likelihood Estimation — The Principle

## The likelihood function

For i.i.d. observations, $L(\theta) = \prod_i f(x_i;\theta)$ and $\ell(\theta) = \ln L(\theta) = \sum_i \ln f(x_i;\theta)$. The **MLE** makes the observed data **most probable**: $\hat{\theta}_{\text{ML}} = \arg\max_\theta \ell(\theta)$.

## The standard recipe

1. Write the likelihood $L(\theta)$.
2. Take logs to get $\ell(\theta)$ (turns products into sums).
3. Differentiate: solve $\dfrac{d\ell}{d\theta} = 0$.
4. Verify it is a maximum: $\dfrac{d^2\ell}{d\theta^2} < 0$.

## Why take logarithms?

- $\ln$ is **monotone**, so the maximiser is unchanged.
- Products $\rightarrow$ sums: much easier to differentiate.
- Avoids **numerical underflow** when multiplying thousands of small probabilities.

## Already seen in this course

Least squares = MLE under Gaussian noise; log loss = MLE under a Bernoulli model; the EM algorithm maximises a mixture likelihood.

# Numerical Example — MLE for a Bernoulli Parameter

## Problem

A coin is tossed $n = 10$ times and lands heads $k = 7$ times. Estimate $p = P(\text{head})$.

## Derivation

$$L(p) = p^k(1-p)^{n-k} = p^7(1-p)^3$$

$$\ell(p) = 7\ln p + 3\ln(1-p)$$

$$\frac{d\ell}{dp} = \frac{7}{p} - \frac{3}{1-p} = 0$$

$$7(1-p) = 3p \;\Rightarrow\; 7 = 10p \;\Rightarrow\; \boxed{\hat{p} = 0.7}$$

In general $\hat{p} = k/n$ — the sample proportion.

## Second-derivative check

$\frac{d^2\ell}{dp^2} = -\frac{7}{p^2} - \frac{3}{(1-p)^2} < 0$ everywhere $\Rightarrow$ a genuine **maximum**. ✓

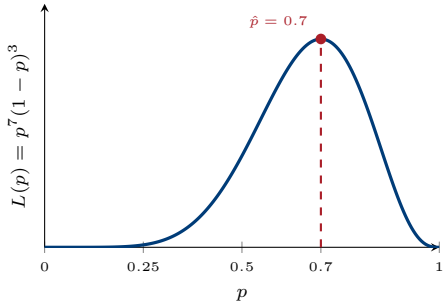



Figure: The likelihood peaks at $p = 0.7$. Check:

$L(0.6) = 0.00179 < L(0.7) = 0.00222 > L(0.8) = 0.00168$.

# Numerical Example — MLE for a Gaussian

## Derivation

$\ell(\mu, \sigma^2) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\sigma^2 - \frac{1}{2\sigma^2}\sum_i (x_i - \mu)^2$, hence

$$\frac{\partial \ell}{\partial \mu} = 0 \Rightarrow \hat{\mu} = \bar{x}; \qquad \frac{\partial \ell}{\partial \sigma^2} = 0 \Rightarrow \hat{\sigma}^2 = \frac{1}{n}\sum_i (x_i - \hat{\mu})^2$$

## Data: $2, 4, 4, 4, 5, 5, 7, 9 \quad (n = 8)$

$$\hat{\mu} = \frac{2+4+4+4+5+5+7+9}{8} = \frac{40}{8} = 5$$

Deviations: $-3, -1, -1, -1, 0, 0, 2, 4$; squares $9, 1, 1, 1, 0, 0, 4, 16$, summing to $32$.

$$\hat{\sigma}^2_{\text{ML}} = \frac{32}{8} = 4 \quad \Rightarrow \quad \hat{\sigma} = 2$$

## The unbiased alternative

$$s^2 = \frac{32}{n-1} = \frac{32}{7} = 4.5714, \qquad s = 2.138$$

## Note the discrepancy

The MLE **under-estimates** the variance:

$$\mathbb{E}[\hat{\sigma}^2_{\text{ML}}] = \frac{n-1}{n}\sigma^2 = \frac{7}{8}\sigma^2 = 0.875\,\sigma^2$$

We prove this on the next slide.

# Unbiased Estimation — Why $n-1$?

## The claim

$\mathbb{E}\left[\frac{1}{n}\sum_i (X_i - \bar{X})^2\right] = \frac{n-1}{n}\sigma^2$ — the MLE is **biased downwards**.

## Proof sketch

Expand about the true mean: $\sum_i (X_i - \bar{X})^2 = \sum_i (X_i - \mu)^2 - n(\bar{X} - \mu)^2$. Using $\mathbb{E}[(X_i - \mu)^2] = \sigma^2$ and $\mathbb{E}[(\bar{X} - \mu)^2] = \sigma^2/n$:

$$\mathbb{E}\Big[\sum_i (X_i - \bar{X})^2\Big] = n\sigma^2 - \sigma^2 = (n-1)\sigma^2$$

Dividing by $n-1$ gives an **unbiased** estimator:

$$\boxed{s^2 = \frac{1}{n-1}\sum_i (X_i - \bar{X})^2}$$

## The intuition

$\bar{X}$ is itself computed *from the data*, so the deviations $X_i - \bar{X}$ are systematically **smaller** than the deviations from the true $\mu$. We have "used up" one **degree of freedom**, so we divide by $n-1$, not $n$.

## Practical note

The difference matters only for **small** $n$: at $n = 8$ it is $12.5\%$; at $n = 1000$ it is $0.1\%$. And note $s = \sqrt{s^2}$ is *still* slightly biased — unbiasedness does not survive non-linear transformation.

# More MLE Examples and the Properties of MLE

## Poisson

$f(x;\lambda) = \dfrac{e^{-\lambda}\lambda^{x}}{x!}$, so $\ell = -n\lambda + \ln\lambda \sum x_i - \sum \ln x_i!$ and

$$\frac{d\ell}{d\lambda} = -n + \frac{\sum x_i}{\lambda} = 0 \;\Rightarrow\; \hat{\lambda} = \bar{x}$$

*Data* $3, 5, 2, 4, 6$: $\hat{\lambda} = \dfrac{20}{5} = 4$.

## Exponential

$f(x;\lambda) = \lambda e^{-\lambda x}$, so $\ell = n\ln\lambda - \lambda \sum x_i$ and

$$\hat{\lambda} = \frac{n}{\sum x_i} = \frac{1}{\bar{x}}$$

*Data* $2, 3, 5, 4, 6$: $\hat{\lambda} = \dfrac{5}{20} = 0.25$ (mean lifetime $4$).

## Properties of MLE (large $n$)

- **Consistent**: $\hat{\theta} \xrightarrow{p} \theta$.
- **Asymptotically unbiased** (though possibly biased for finite $n$, as we just saw).
- **Asymptotically efficient**: attains the Cramér–Rao bound.
- **Asymptotically normal**: $\hat{\theta} \dot{\sim} \mathcal{N}\big(\theta,\ [nI(\theta)]^{-1}\big)$.
- **Invariant**: the MLE of $g(\theta)$ is $g(\hat{\theta})$.

## Limitations

Can be badly biased for small $n$; may not exist or be unique; sensitive to model misspecification; can overfit (hence regularization = MAP estimation).

# Confidence Intervals — The Concept

## Definition

A $100(1-\alpha)\%$ confidence interval is a random interval $(L, U)$ with

$$P\big(L \le \theta \le U\big) = 1 - \alpha$$

Common choices: $\alpha = 0.05$ (95%), $\alpha = 0.01$ (99%).

## The correct interpretation

$\theta$ is **fixed**, the interval is **random**. "95% confidence" means that *if we repeated the experiment many times*, about $95\%$ of the intervals constructed would contain $\theta$.

It does **not** mean "there is a $95\%$ probability that $\theta$ lies in *this* interval".

## General form

$$\text{estimate} \pm \underbrace{(\text{critical value}) \times (\text{standard error})}_{\text{margin of error}}$$

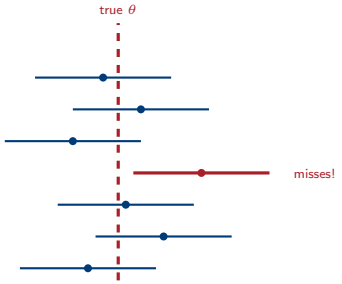



Figure: Seven repeated samples; six intervals capture $\theta$, one does not.

# CI for One Mean — $\sigma$ Known ($z$-interval)

## Derivation

If $X_i \sim \mathcal{N}(\mu, \sigma^2)$ (or $n$ is large, by the CLT) then $\bar{X} \sim \mathcal{N}(\mu, \sigma^2/n)$, so $Z = \frac{\bar{X}-\mu}{\sigma/\sqrt{n}} \sim \mathcal{N}(0,1)$. From $P(-z_{\alpha/2} \le Z \le z_{\alpha/2}) = 1-\alpha$, rearranging for $\mu$:

$$\boxed{\bar{x} \;\pm\; z_{\alpha/2}\,\sigma/\sqrt{n}}$$

## Worked example

$n = 64$, $\bar{x} = 25$, $\sigma = 4$ known. Build a 95% CI.

$$\text{SE} = \sigma/\sqrt{n} = 4/8 = 0.5$$
$$\text{MOE} = 1.96(0.5) = 0.98$$
$$\text{CI} = 25 \pm 0.98 = (24.02,\ 25.98)$$

## Critical values

| Confidence | $z_{\alpha/2}$ |
|---|---|
| 90% | 1.645 |
| 95% | **1.960** |
| 99% | 2.576 |

## Observe

Higher confidence ⇒ **wider** interval. A 99% CI here would be $25 \pm 2.576(0.5) = (23.71, 26.29)$. **Precision and confidence trade off.**

# CI for One Mean — $\sigma$ Unknown ($t$-interval)

## The $t$-distribution

When $\sigma$ is estimated by $s$, the standardised quantity follows Student's $t$ with $n-1$ degrees of freedom:

$$T = \frac{\bar{X} - \mu}{s/\sqrt{n}} \sim t_{n-1} \qquad \Longrightarrow \qquad \boxed{\bar{x} \;\pm\; t_{\alpha/2,\, n-1}\,\frac{s}{\sqrt{n}}}$$

The $t$ density has **heavier tails** than the normal, widening the interval to account for the extra uncertainty in $s$; as $n \to \infty$, $t \to z$.

## Worked example: $12, 15, 14, 10, 13, 16, 11, 13$

$$\bar{x} = \frac{104}{8} = 13, \qquad \sum (x_i - \bar{x})^2 = 28$$

$$s^2 = \frac{28}{7} = 4 \;\Rightarrow\; s = 2$$

$$\text{SE} = \frac{2}{\sqrt{8}} = 0.7071, \qquad t_{0.025,7} = 2.365$$

$$\text{CI} = 13 \pm 2.365(0.7071) = 13 \pm 1.672$$

$$= (11.33,\ 14.67)$$

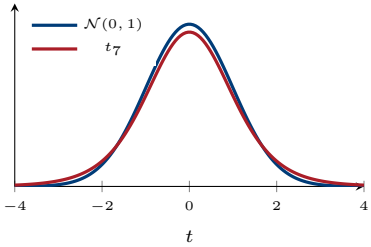



Figure: $t_7$ (red) has fatter tails, so $t_{0.025,7} = 2.365 > 1.96$.

# CI for the Difference of Two Means

## Pooled-variance $t$-interval (equal variances assumed)

$$s_p^2 = \frac{(n_1-1)s_1^2 + (n_2-1)s_2^2}{n_1+n_2-2}, \qquad \text{SE} = s_p\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}$$

$$\boxed{(\bar{x}_1 - \bar{x}_2) \;\pm\; t_{\alpha/2,\, n_1+n_2-2}\ \text{SE}}$$

## Worked example — two teaching methods

Group 1: $n_1 = 10,\ \bar{x}_1 = 85,\ s_1 = 6$. Group 2: $n_2 = 12,\ \bar{x}_2 = 80,\ s_2 = 5$.

$$s_p^2 = \frac{9(36) + 11(25)}{20} = \frac{324 + 275}{20} = 29.95$$

$$s_p = 5.473, \qquad \text{SE} = 5.473\sqrt{\tfrac{1}{10}+\tfrac{1}{12}} = 2.343$$

With $t_{0.025,20} = 2.086$ and $\bar{x}_1 - \bar{x}_2 = 5$:

$$\text{CI} = 5 \pm 2.086(2.343) = 5 \pm 4.888 = (0.11,\ 9.89)$$

## Interpretation — the decision rule

The interval **excludes zero**, so at the $5\%$ level there *is* a statistically significant difference: method 1 scores higher.

Had the CI contained $0$, we could not have ruled out "no difference".

## If variances differ

Use **Welch's** interval, with $\text{SE} = \sqrt{s_1^2/n_1 + s_2^2/n_2}$ and the Welch–Satterthwaite degrees of freedom. For **paired** data, form the differences and use a one-sample interval.

# CI for a Variance — the $\chi^2$ Interval

## Derivation

For a normal sample $\frac{(n-1)s^2}{\sigma^2} \sim \chi^2_{n-1}$, so

$$\left( \frac{(n-1)s^2}{\chi^2_{\alpha/2,\, n-1}}, \quad \frac{(n-1)s^2}{\chi^2_{1-\alpha/2,\, n-1}} \right)$$

## Worked example

$n = 20$, $s^2 = 25$, df $= 19$; $\chi^2_{0.025,19} = 32.852$, $\chi^2_{0.975,19} = 8.907$.

$$\text{lower} = \frac{19(25)}{32.852} = \frac{475}{32.852} = 14.46$$

$$\text{upper} = \frac{19(25)}{8.907} = \frac{475}{8.907} = 53.33$$

CI for $\sigma^2 = (14.46,\ 53.33)$; for $\sigma$: $(3.80,\ 7.30)$.

## Note the asymmetry

Unlike the mean intervals this one is **not symmetric** about $s^2 = 25$ — because $\chi^2$ is **right-skewed**.

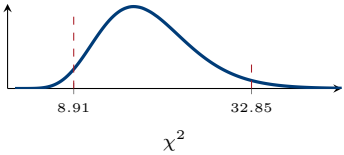



Figure: $\chi^2_{19}$ is right-skewed, so the interval is asymmetric.

# Quick Check — Round 1 (Parameter Estimation)

## [Q1]

An estimator has bias $2$ and variance $5$. Compute its MSE.

## [Q2]

In $20$ Bernoulli trials there are $12$ successes. Give the MLE of $p$ and state the general formula.

## [Q3]

For data with $\sum(x_i - \bar{x})^2 = 45$ and $n = 10$, compute both the MLE and the unbiased estimate of the variance.

## [Q4]

A sample has $n = 100$, $\bar{x} = 50$, $\sigma = 10$ (known). Construct the $95\%$ confidence interval for $\mu$.

Answers on the next slide.

# Answers — Round 1

## [A1]

MSE $= \mathrm{Var} + \mathrm{Bias}^2 = 5 + 2^2 = 5 + 4 =$ **9**.

## [A2]

$\hat{p} = \dfrac{k}{n} = \dfrac{12}{20} =$ **0.6**; in general the MLE of a Bernoulli parameter is the **sample proportion** $k/n$.

## [A3]

MLE: $\hat{\sigma}^2 = \dfrac{45}{10} =$ **4.5** (biased low).

Unbiased: $s^2 = \dfrac{45}{9} =$ **5.0**.

## [A4]

SE $= \dfrac{10}{\sqrt{100}} = 1$; MOE $= 1.96(1) = 1.96$;

$$\text{CI} = 50 \pm 1.96 = \textbf{(48.04, 51.96)}$$

# Why Model Evaluation Matters

## The central question

*How will this model perform on data it has never seen?*

Training error is an **optimistically biased** estimate — the model has already adapted to that data.

## What can go wrong

- **Overfitting** missed because we scored on training data.
- **Data leakage** between train and test.
- Tuning on the **test set** (it becomes validation data).
- Wrong metric for the problem (accuracy on imbalanced data).
- **Distribution shift** after deployment.

## Two families of validation

- **Automated / statistical**: holdout, $k$-fold, LOOCV, subsampling, bootstrap.
- **Human-in-the-loop**: expert review, simulation, override mechanisms.

## The golden rule (restated)

**Train** on the training set, **tune** on the validation set, and touch the **test set exactly once**, at the very end.

# ML Model Validation by Humans

## What it means

Domain experts inspect the model's predictions, explanations and failure cases — checking not just *whether* it is right but *why*.

## Typical activities

- Reviewing a random and a **worst-case** sample of predictions.
- Checking **feature importances** for plausibility.
- Auditing for **bias** across demographic groups.
- **Sanity tests**: does the model behave correctly on cases whose answer we already know?
- **A/B tests** with real users.

## Why metrics are not enough

A model can score $95\%$ accuracy while relying on a **spurious** feature — the classic case of a classifier that detected snow rather than wolves. Only a human looking at explanations catches that.

## Where it is mandatory

**High-stakes** domains — medicine, credit, hiring, criminal justice — where regulation and ethics require a human to be accountable for the decision.

Human validation *complements* statistical validation; it does not replace it.

# Holdout Set Validation

## Method

Split the data *once* into disjoint sets:

$$\text{train } (\sim 70\%) \mid \text{validation } (15\%) \mid \text{test } (15\%)$$

Train on the first, tune on the second, report on the third.

## Numeric

With $m = 1000$ examples: $700$ train, $150$ validation, $150$ test. A test accuracy of $88\%$ has standard error

$$\sqrt{\frac{0.88(0.12)}{150}} = 0.0265$$

so the $95\%$ CI is $0.88 \pm 1.96(0.0265) = (0.828,\ 0.932)$ — a **wide** band.


## Assessment

+ Fast — only **one** model is trained.
+ Simple; the standard choice for **large** datasets.

− **High variance**: the estimate depends on which points happened to land in the test set.
− **Wasteful**: the model never learns from $30\%$ of the data.
− Risky when $m$ is small.

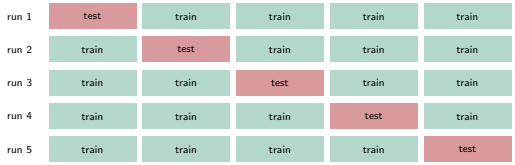
Use **stratified** splitting to preserve class proportions.

# $k$-Fold Cross-Validation

## Method

Split the data into $k$ equal folds. For $i = 1..k$: train on the other $k-1$ folds and test on fold $i$. Then average:

$$\text{CV}_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{Err}_i$$




## Data usage, $m = 1000$

| $k$ | test size | train size | models |
|---|---|---|---|
| 5 | 200 | 800 | 5 |
| 10 | 100 | 900 | 10 |

## Properties

- **Every point** is used for testing exactly once and for training $k-1$ times.
- Much **lower variance** than a single holdout.
- Cost: $k$ **models** must be trained.
- Standard values: $k = 5$ or $k = 10$.

# Numerical Example — 5-Fold Cross-Validation

## Fold accuracies

$$0.82, \quad 0.85, \quad 0.79, \quad 0.88, \quad 0.86$$

## Compute mean and spread

$$\bar{a} = \frac{0.82 + 0.85 + 0.79 + 0.88 + 0.86}{5} = \frac{4.20}{5} = 0.840$$

Deviations: $-0.02,\ 0.01,\ -0.05,\ 0.04,\ 0.02$; squares sum to $0.005$.

$$s = \sqrt{\frac{0.005}{4}} = \sqrt{0.00125} = 0.0354$$

$$\text{SE} = \frac{0.0354}{\sqrt{5}} = 0.0158$$

## Report it properly

$$\text{Accuracy} = 0.840 \pm 0.035$$

With $t_{0.025,4} = 2.776$, the 95% CI for the true accuracy is

$$0.840 \pm 2.776(0.0158) = (0.796,\ 0.884)$$

## The lesson

Never report a bare single number. The **spread across folds** tells you whether a rival model scoring $0.85$ is genuinely better — here it clearly is *not*.

# Leave-One-Out Cross-Validation (LOOCV)

## Method

The extreme case $k = m$: each single observation is held out in turn.

$$\mathrm{CV}_{(m)} = \frac{1}{m} \sum_{i=1}^{m} L\big(y_i, \hat{f}^{-i}(\mathbf{x}_i)\big)$$

where $\hat{f}^{-i}$ is trained without example $i$.

## A remarkable shortcut for linear models

For least-squares regression, LOOCV needs **only one fit**:

$$\mathrm{CV}_{(m)} = \frac{1}{m} \sum_{i=1}^{m} \left( \frac{y_i - \hat{y}_i}{1 - h_{ii}} \right)^2$$

where $h_{ii}$ is the $i$-th diagonal of the hat matrix $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$.

## Trade-offs

+ Almost **unbiased** — each model sees $m - 1$ points.
+ **Deterministic** — no random split, so it is reproducible.

– **Expensive**: $m$ model fits.
– **High variance**: the $m$ training sets are nearly identical, so the errors are highly correlated.

## Numeric

If $y_i - \hat{y}_i = 0.6$ and $h_{ii} = 0.25$, that point contributes

$$\left( \frac{0.6}{0.75} \right)^2 = 0.8^2 = 0.64$$

— larger than the raw squared residual $0.36$, because high-leverage points are penalised.

# Random Subsampling and the Teach-and-Test Method

## Random subsampling (Monte-Carlo CV)

Repeat $R$ times: draw a **random** train/test split (say $80/20$), train, and record the error. Average over the $R$ runs.

- **Free choice** of $R$ and of the test proportion — unlike $k$-fold, where they are linked.
- **Some points may never be tested**, others tested many times.
- Splits **overlap**, so the runs are not independent.

## Numeric

$R = 10$ runs at $80/20$ on $m = 500$: each run trains on $400$ and tests on $100$; a point is tested on average $10(0.2) = 2$ times.

## The teach-and-test method

The classical two-phase protocol: **teach** (train) the model on one dataset, then **test** it on a separate, previously unseen dataset — ideally collected independently, perhaps at a later time or another site.

## Why it is the strongest evidence

It tests **external validity** (generalisation to a genuinely new population), not merely internal validity. A model that survives an external test on a different hospital's patients is far more trustworthy.

## Related idea

For time series, use a **temporal split**: train on the past, test on the future. Random shuffling would leak future information.

# Bootstrapping as a Validation Method

## Method (Efron)

Draw $B$ bootstrap samples of size $m$ **with replacement**; train on each; test on the examples **not** drawn (the out-of-bag set).

## How much is out of bag?

$$P(\text{point } i \text{ never drawn}) = \left(1 - \frac{1}{m}\right)^m \xrightarrow{m \to \infty} \frac{1}{e} = 0.368$$

| $m$ | $(1 - 1/m)^m$ | OOB |
|---|---|---|
| 10 | 0.3487 | 34.9% |
| 100 | 0.3660 | 36.6% |
| 1000 | 0.3677 | 36.8% |

So each model trains on about **63.2%** of the distinct examples.

## The .632 estimator

The OOB error is pessimistic (smaller effective training set), so blend it with the optimistic training error:

$$\text{Err}_{.632} = 0.368\,\text{Err}_{\text{train}} + 0.632\,\text{Err}_{\text{OOB}}$$

## Numeric

With $\text{Err}_{\text{train}} = 0.02$ and $\text{Err}_{\text{OOB}} = 0.30$:

$$0.368(0.02) + 0.632(0.30) = 0.0074 + 0.1896 = 0.197$$

## Also gives uncertainty

The spread across the $B$ replicates yields **confidence intervals** for any statistic — a major advantage over a single split.

# Simulation and Overriding-Mechanism Methods

## Running AI model simulations

Evaluate the model inside a **simulated environment** rather than on a static test set.

- **Backtesting**: replay historical data as if live (trading, demand).
- **Shadow deployment**: run alongside the current system without acting on its outputs.
- **Stress testing**: synthetic extreme scenarios.
- **Digital twins**: a full simulated plant or vehicle.

## Why simulate

It captures **feedback effects** that a static test set cannot: the model's own actions change the future data it sees.

## The overriding mechanism method

Deploy the model with a **human override** in the loop, and treat the **override rate** as a live validation metric.

- Every override is a **labelled failure case**.
- A rising override rate signals **drift** or degradation.
- Overrides feed back as new training data (*human-in-the-loop* learning).

## Design principle

In safety-critical systems the model should **never be the last word**: provide a clear override path, log every use of it, and monitor the trend.

Both methods validate the model *in situ* — where it actually operates.

# Comparison of Validation Methods

| Method | Models | Bias | Variance | Best used when |
|---|---|---|---|---|
| Holdout | 1 | high | high | data is large; speed matters |
| $k$-fold ($k$=5, 10) | $k$ | low | moderate | the **default** choice |
| LOOCV | $m$ | very low | high | $m$ is small; linear models (shortcut) |
| Random subsampling | $R$ | low | moderate | you need control of $R$ and split size |
| Teach and test | 1 | lowest | — | an independent dataset exists |
| Bootstrap / .632 | $B$ | low | low | small samples; you want CIs |
| Simulation | — | — | — | feedback effects matter |
| Override monitoring | — | — | — | live, safety-critical systems |

## Practical guidance

Use **stratified $k$-fold** as the default. Add a **nested** CV loop when you are *also* tuning hyperparameters — otherwise the reported score is optimistic. Reserve a true **holdout** for the final, one-time report.

# The Confusion Matrix and Its Metrics

## Definitions

| | Pred. + | Pred. − |
|---|---|---|
| **Actual +** | TP | FN |
| **Actual −** | FP | TN |

$$\text{Accuracy} = \frac{TP+TN}{n}, \quad \text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall (TPR)} = \frac{TP}{TP+FN}, \quad \text{FPR} = \frac{FP}{FP+TN}$$

$$F_1 = \frac{2PR}{P+R}$$

## Numeric: $TP$=80, $FN$=20, $FP$=30, $TN$=870 ($n$=1000)

$$\text{Accuracy} = \frac{80+870}{1000} = 0.950$$

$$\text{Precision} = \frac{80}{110} = 0.727$$

$$\text{Recall} = \frac{80}{100} = 0.800$$

$$\text{Specificity} = \frac{870}{900} = 0.967, \qquad \text{FPR} = \frac{30}{900} = 0.033$$

$$F_1 = \frac{2(0.727)(0.800)}{1.527} = 0.762$$

## The baseline trap

Predicting "always negative" would score $\frac{900}{1000} = \mathbf{90\%}$ accuracy — close to our $95\%$, yet useless. This is why **precision, recall and $F_1$** matter on imbalanced data.

# The ROC Curve — Construction

## Definition

A classifier that outputs **scores** becomes a family of classifiers, one per threshold $\tau$. The **ROC curve** plots

$$\text{TPR}(\tau) = \frac{TP}{P} \quad \text{against} \quad \text{FPR}(\tau) = \frac{FP}{N}$$

as $\tau$ sweeps from $+\infty$ to $-\infty$.

## Reading the curve

- $(0, 0)$: classify everything negative.
- $(1, 1)$: classify everything positive.
- $(0, 1)$: the **perfect** classifier.
- The **diagonal** is random guessing.
- Curves nearer the **top-left** are better.

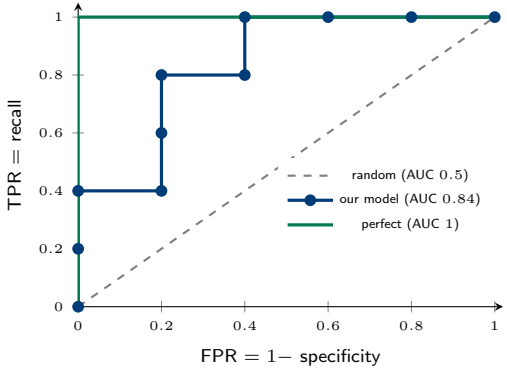



Figure: The step-wise ROC curve built on the next slide.

# Numerical Example — Building the ROC Curve

## Ten test examples sorted by predicted score ($5$ positives, $5$ negatives)

| Score | .95 | .90 | .80 | .75 | .60 | .50 | .45 | .30 | .20 | .10 |
|---|---|---|---|---|---|---|---|---|---|---|
| True | + | + | − | + | + | − | + | − | − | − |
| TP | 1 | 2 | 2 | 3 | 4 | 4 | 5 | 5 | 5 | 5 |
| FP | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 3 | 4 | 5 |
| TPR | .2 | .4 | .4 | .6 | .8 | .8 | 1.0 | 1.0 | 1.0 | 1.0 |
| FPR | 0 | 0 | .2 | .2 | .2 | .4 | .4 | .6 | .8 | 1.0 |

## How the table is built

Lower the threshold one example at a time. A **positive** example moves the curve **up** by $1/P = 0.2$; a **negative** moves it **right** by $1/N = 0.2$.

## Choosing an operating point

At threshold $0.60$ we get $\text{TPR} = 0.8$ with only $\text{FPR} = 0.2$ — often the best compromise. **Youden's index** $J = \text{TPR} - \text{FPR}$ is maximised here: $0.8 - 0.2 = 0.6$.

# AUC — Two Ways to Compute It

## Method 1 — trapezoidal area

Sum the areas of the vertical strips of the step curve:

$$[0, 0.2]: \ 0.2 \times 0.4 = 0.08$$
$$[0.2, 0.4]: \ 0.2 \times 0.8 = 0.16$$
$$[0.4, 0.6]: \ 0.2 \times 1.0 = 0.20$$
$$[0.6, 0.8]: \ 0.2 \times 1.0 = 0.20$$
$$[0.8, 1.0]: \ 0.2 \times 1.0 = 0.20$$
$$\text{Total} = 0.84$$

## Method 2 — Mann–Whitney interpretation

AUC is the **probability that a randomly chosen positive scores higher than a randomly chosen negative**:

$$\text{AUC} = \frac{\#\{(p, q) : s_p > s_q\}}{P \times N}$$

Counting over the $5 \times 5 = 25$ pairs gives 21 correctly ordered pairs:

$$\text{AUC} = \frac{21}{25} = 0.84 \quad \checkmark$$

## Rules of thumb

$0.5$ random; $0.7$–$0.8$ acceptable; $0.8$–$0.9$ good; $> 0.9$ excellent.
AUC is **threshold-free** and **insensitive to class balance** — but for very rare positives, prefer the **precision–recall** curve.

# Quick Check — Round 2 (Model Evaluation)

## [Q1]

A confusion matrix has $TP = 40$, $FP = 10$, $FN = 20$, $TN = 130$. Compute accuracy, precision, recall and $F_1$.

## [Q2]

5-fold CV gives $0.90, 0.86, 0.88, 0.92, 0.84$. Report the mean and the sample standard deviation.

## [Q3]

In bootstrap validation, what fraction of examples is out-of-bag, and what is the .632 estimator for $\text{Err}_{\text{train}} = 0.05$ and $\text{Err}_{\text{OOB}} = 0.25$?

## [Q4]

A model has AUC $= 0.5$. What does this mean? And why can $95\%$ accuracy be a *bad* result?

# Answers — Round 2

## [A1]

$n = 200$. Accuracy $= \frac{40+130}{200} =$ **0.85**; Precision $= \frac{40}{50} =$ **0.80**; Recall $= \frac{40}{60} =$ **0.667**; $F_1 = \frac{2(0.8)(0.667)}{1.467} =$ **0.727**.

## [A2]

Mean $= \frac{4.40}{5} =$ **0.88**. Deviations $0.02, -0.02, 0, 0.04, -0.04$; squares sum to $0.004$; $s = \sqrt{0.004/4} =$ **0.0316**.
Report $0.880 \pm 0.032$.

## [A3]

OOB fraction $\to 1/e \approx$ **36.8%**.
$\text{Err}_{.632} = 0.368(0.05) + 0.632(0.25) = 0.0184 + 0.158 =$ **0.176**.

## [A4]

AUC $= 0.5$ means the model ranks positives above negatives no better than **random guessing**. And $95\%$ accuracy is poor if $95\%$ of the data is already negative — the **trivial always-negative** classifier matches it.

# Ensemble Theory — Why Combine Models?

## The idea

Train $T$ models (**base learners**) and combine their predictions. The ensemble is often **more accurate than any individual member**.

## Condorcet's jury theorem (majority vote)

If $T$ classifiers are **independent** and each is wrong with probability $p < 0.5$, the majority vote is wrong with probability

$$P_{\text{ens}} = \sum_{k=\lceil T/2 \rceil}^{T} \binom{T}{k} p^k (1-p)^{T-k}$$

and $P_{\text{ens}} \to 0$ as $T \to \infty$.

## The two conditions

Members must be (i) **better than random** ($p < 0.5$) and (ii) **diverse** — making *different* errors. Identical models gain nothing.

## Numeric: $p = 0.3$

For $T = 3$, the ensemble errs when $\geq 2$ members err:

$$\binom{3}{2}(0.3)^2(0.7) = 3(0.09)(0.7) = 0.189$$
$$+ (0.3)^3 = 0.027$$
$$= 0.216 \quad \text{(vs } 0.300 \text{ alone)}$$

| $T$ | $p = 0.3$ | $p = 0.4$ |
|---|---|---|
| 1 | 0.300 | 0.400 |
| 3 | 0.216 | 0.352 |
| 5 | 0.163 | 0.317 |
| 11 | 0.078 | 0.247 |
| 21 | 0.026 | 0.174 |

Note how much faster the error falls when the individuals are stronger.

# Why Ensembles Work — Bias, Variance and Diversity

## Variance of an average

For $T$ models each of variance $\sigma^2$ with pairwise correlation $\rho$: $\mathrm{Var}\left(\frac{1}{T}\sum_t \hat{f}_t\right) = \rho\sigma^2 + \frac{1-\rho}{T}\sigma^2$.

## Numeric ($\sigma^2 = 1$)

| $\rho$ | $T = 1$ | $T = 5$ | $T = 25$ |
|---|---|---|---|
| 0.0 | 1.000 | 0.200 | 0.040 |
| 0.3 | 1.000 | 0.440 | 0.328 |

With $\rho = 0.3$, even $T = \infty$ leaves variance $0.3$ — **diversity is the binding constraint**.

## Three complementary explanations (Dietterich)

- **Statistical**: with limited data many hypotheses fit equally well; averaging reduces the risk of picking a bad one.
- **Computational**: local search gets stuck; different starts explore more.
- **Representational**: the true function may lie *outside* $\mathcal{H}$, but inside the span of combinations.

## Which error do they reduce?

**Bagging** mainly reduces **variance**; **boosting** mainly reduces **bias**; **stacking** can reduce both.

# Ensemble Size

## How many members?

- **Bagging / random forests**: more is never worse — the curve **plateaus** (typically $100$–$500$ trees). Choose $T$ by when the OOB error stops improving.
- **Boosting**: more *can* be worse — too many rounds **overfits**. Use early stopping on a validation set.
- **Voting/stacking**: usually a handful of *diverse*, strong models.

## Diminishing returns

From the variance formula, going from $T = 25$ to $T = 100$ at $\rho = 0.3$ moves the variance from $0.328$ to $0.307$ — a $6\%$ gain for $4\times$ the compute.

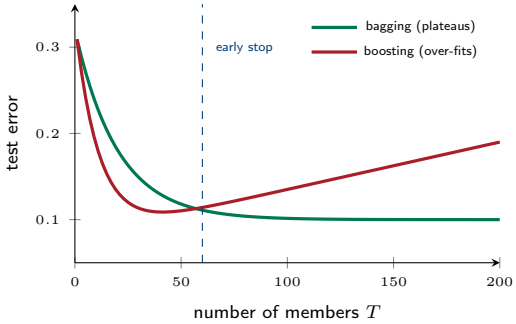



Figure: Bagging saturates safely; boosting has an optimum $T$.

# Voting and Averaging Based Methods

## Classification — voting

$$\text{Hard: } \hat{y} = \text{mode}\{h_1(\mathbf{x}), \ldots, h_T(\mathbf{x})\}$$

$$\text{Soft: } \hat{y} = \arg\max_c \frac{1}{T} \sum_{t=1}^{T} p_t(c \mid \mathbf{x})$$

**Soft voting** usually wins — it uses the models' confidence, not just their labels.

## Regression — averaging

$$\hat{y} = \frac{1}{T} \sum_t \hat{f}_t(\mathbf{x}) \qquad \text{or} \qquad \hat{y} = \sum_t w_t \hat{f}_t(\mathbf{x}), \quad \sum_t w_t = 1$$

## Numeric — hard vs. soft voting

Three classifiers give $P(\text{class } 1)$:

| Model | $P(1)$ | Hard vote |
|---|---|---|
| $h_1$ | 0.45 | class 0 |
| $h_2$ | 0.45 | class 0 |
| $h_3$ | 0.90 | class 1 |

**Hard vote**: 2–1 for class $0$.
**Soft vote**: mean $= \frac{0.45+0.45+0.90}{3} = 0.60 > 0.5 \Rightarrow$ class $1$.
The two disagree — $h_3$'s *confidence* outweighs two hesitant votes.

## Weighted average

Predictions $12, 15, 14$ with weights $0.5, 0.2, 0.3$:

$$0.5(12) + 0.2(15) + 0.3(14) = 6 + 3 + 4.2 = 13.2$$

(simple average would give $13.67$).

# Bagging — Bootstrap Aggregating

## Algorithm (Breiman, 1996)

1. For $t = 1, \ldots, T$: draw a bootstrap sample $S_t$ of size $m$ **with replacement**.
2. Train base learner $h_t$ on $S_t$.
3. Combine: **majority vote** (classification) or **average** (regression).

$$\hat{f}_{\text{bag}}(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} h_t(\mathbf{x})$$

## Key facts

- Each sample omits $\approx$ **36.8%** of points $\Rightarrow$ free **OOB** error estimate.
- Reduces **variance**, leaves bias roughly unchanged.
- Base learners should be **unstable** (deep trees) — bagging a linear model achieves little.
- **Embarrassingly parallel**.

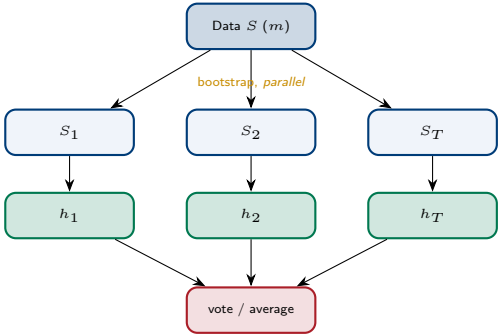



Figure: Bagging trains all members *independently and in parallel*.

## Extension

Adding random **feature subsampling** at each split turns bagging into a **random forest** (Unit 3).

# Boosting — The AdaBoost Algorithm

## Algorithm (Freund & Schapire, 1997)

Initialise weights $w_i^{(1)} = 1/m$. For $t = 1, \dots, T$:

$$\text{(1) train } h_t \text{ on the weighted data;} \qquad \epsilon_t = \sum_{i:\, h_t(\mathbf{x}_i) \neq y_i} w_i^{(t)}$$

$$\text{(2)} \quad \alpha_t = \frac{1}{2} \ln\!\left( \frac{1 - \epsilon_t}{\epsilon_t} \right); \qquad \text{(3)} \quad w_i^{(t+1)} = \frac{w_i^{(t)} \exp\big( - \alpha_t y_i h_t(\mathbf{x}_i) \big)}{Z_t}$$

Final classifier: $H(\mathbf{x}) = \operatorname{sign}\left( \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x}) \right)$.

## The mechanism

Misclassified points get their weight **increased**, so the next learner is forced to concentrate on them. Learners with lower error $\epsilon_t$ receive a larger vote $\alpha_t$.

## Behaviour of $\alpha_t$

| $\epsilon_t$ | $\alpha_t$ | meaning |
|---|---|---|
| 0.01 | 2.30 | near-perfect, huge vote |
| 0.20 | 0.69 | good |
| 0.40 | 0.20 | weak |
| 0.50 | 0.00 | useless — no vote |

# Numerical Example — One AdaBoost Round

## Setup: $m = 5$, uniform weights $w_i = 0.2$; $h_1$ misclassifies points $2$ and $4$

$$\epsilon_1 = 0.2 + 0.2 = 0.4, \qquad \alpha_1 = \tfrac{1}{2}\ln\!\left(\tfrac{1-0.4}{0.4}\right) = \tfrac{1}{2}\ln 1.5 = 0.2027$$

## Weight update

Correct points: $w \times e^{-0.2027} = 0.2(0.8165) = 0.1633$.
Wrong points: $w \times e^{+0.2027} = 0.2(1.2247) = 0.2449$.

$$Z_1 = 3(0.1633) + 2(0.2449) = 0.9798$$

| Point | old $w$ | unnorm. | new $w$ |
|---|---|---|---|
| 1 (ok) | 0.20 | 0.1633 | 0.1667 |
| 2 (*wrong*) | 0.20 | 0.2449 | 0.2500 |
| 3 (ok) | 0.20 | 0.1633 | 0.1667 |
| 4 (*wrong*) | 0.20 | 0.2449 | 0.2500 |
| 5 (ok) | 0.20 | 0.1633 | 0.1667 |

## Check and interpret

Weights sum to $1.000$ ✓

The two hard points rose from $0.20$ to **0.25**; the three easy ones fell to **1/6**.

## A theorem worth noticing

The misclassified points now carry total weight

$$2(0.25) = 0.50$$

**exactly one half**. AdaBoost always reweights so that the *previous* hypothesis would score exactly $50\%$ — i.e. becomes useless — on the new distribution, forcing $h_2$ to learn something genuinely new.

# Bagging vs. Boosting

| Aspect | Bagging | Boosting |
|---|---|---|
| Training | **parallel**, independent | **sequential**, each depends on the last |
| Sampling | bootstrap, uniform weights | full data, **adaptive** weights |
| Base learner | strong, low bias (deep trees) | **weak**, high bias (stumps) |
| Reduces | **variance** | **bias** (and some variance) |
| Combination | equal vote / average | **weighted** by $\alpha_t$ |
| Overfitting | resistant; more $T$ is safe | possible; needs early stopping |
| Noise/outliers | robust | **sensitive** (keeps up-weighting them) |
| Examples | random forest, extra trees | AdaBoost, gradient boosting, XGBoost |

## Choosing between them

High-variance, overfitting base model $\Rightarrow$ **bagging**. Underfitting, high-bias base model $\Rightarrow$ **boosting**. Noisy labels $\Rightarrow$ prefer bagging. On clean tabular data, **gradient boosting** is usually the strongest single choice.

# Stacking (Stacked Generalization)

## Idea (Wolpert, 1992)

Instead of a *fixed* rule for combining, **learn** the combiner.

1. Train $T$ diverse **level-0** models.
2. Generate **out-of-fold** predictions for every training point.
3. Train a **level-1 meta-learner** on those predictions as features.

## The critical detail

The meta-features **must** come from out-of-fold predictions. Using in-sample predictions lets an overfitting base model look perfect, and the meta-learner will trust it blindly.

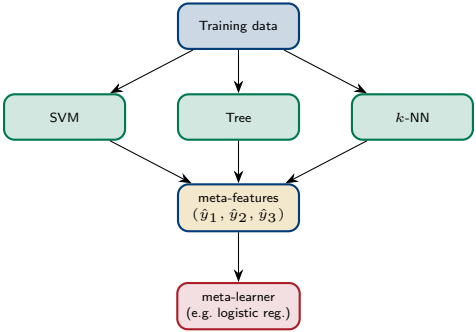



Figure: Two levels — diverse base models feeding a learned combiner.

## Practice

Use **heterogeneous** base models (different inductive biases) and a **simple** meta-learner (regularized linear) to avoid overfitting level 1. **Blending** is the simpler variant using one holdout set instead of folds.

# Quick Check — Round 3 (Ensembles)

## [Q1]

Three independent classifiers each have error $p = 0.2$. Compute the majority-vote error of the ensemble.

## [Q2]

In AdaBoost a weak learner has $\epsilon_t = 0.25$. Compute $\alpha_t$. $[\ln 3 = 1.0986]$

## [Q3]

Four models predict $10,\ 12,\ 11,\ 15$ with weights $0.4,\ 0.3,\ 0.2,\ 0.1$. Compute the weighted average and compare with the simple average.

## [Q4]

State two differences between bagging and boosting, and say which you would choose for noisy labels.

# Answers — Round 3

## [A1]

The ensemble errs if $\geq 2$ of 3 err: $\binom{3}{2}(0.2)^2(0.8) + (0.2)^3 = 0.096 + 0.008 = \mathbf{0.104}$ — a drop from $0.20$ to about **0.10**.

## [A2]

$\alpha_t = \frac{1}{2}\ln\frac{1-0.25}{0.25} = \frac{1}{2}\ln 3 = \frac{1.0986}{2} = \mathbf{0.549}$.

## [A3]

Weighted $= 0.4(10) + 0.3(12) + 0.2(11) + 0.1(15) = 4 + 3.6 + 2.2 + 1.5 = \mathbf{11.3}$.
Simple $= \frac{48}{4} = \mathbf{12.0}$. The weighting pulls the prediction toward the most-trusted model.

## [A4]

(i) Bagging trains in **parallel** on bootstrap samples; boosting is **sequential** with adaptive weights. (ii) Bagging reduces **variance**; boosting reduces **bias**. For **noisy labels choose bagging** — boosting keeps up-weighting the mislabelled points.

# Summary of Unit 5

## What we covered

1. **Point estimation** — estimator vs. estimate; bias, variance, MSE $=$ Var $+$ Bias$^2$; consistency, efficiency, Cramér–Rao.
2. **MLE** — the log-likelihood recipe; derivations for Bernoulli ($k/n$), Gaussian ($\bar{x}$, $\frac{1}{n}\sum(x-\bar{x})^2$), Poisson and exponential.
3. **Unbiasedness** — proof that $\mathbb{E}[\hat{\sigma}^2_{\text{ML}}] = \frac{n-1}{n}\sigma^2$, hence the $n-1$ divisor.
4. **Confidence intervals** — $z$-interval, $t$-interval, two-mean pooled interval, $\chi^2$ variance interval; all worked numerically.
5. **Validation** — human review, holdout, $k$-fold, LOOCV (with the hat-matrix shortcut), random subsampling, teach-and-test, bootstrap $.632$, simulation and override monitoring.
6. **Metrics** — confusion matrix, precision/recall/$F_1$, the imbalance trap.
7. **ROC and AUC** — construction from ranked scores; AUC computed by area *and* by the Mann–Whitney pair count (both $0.84$).
8. **Ensembles** — Condorcet's theorem, variance of an average, ensemble size.
9. **Methods** — hard/soft voting, weighted averaging, bagging, AdaBoost (with a full reweighting round), bagging vs. boosting, stacking.

**Estimate carefully, validate honestly, and combine wisely.**

# Practice Questions (Exam Oriented)

## Short answer (2–5 marks)

1. Define bias, variance and MSE of an estimator and state the relation between them.
2. Derive the MLE of the Bernoulli parameter $p$.
3. Explain why the sample variance divides by $n - 1$.
4. State the $z$- and $t$-interval formulas and say when each applies.
5. Differentiate $k$-fold cross-validation from LOOCV on three points.
6. Define TPR, FPR, precision and recall from a confusion matrix.
7. State Condorcet's jury theorem and its two conditions.

## Long answer (8–10 marks) — include numerics

8. Derive the MLE of $\mu$ and $\sigma^2$ for a Gaussian and compute both for a given sample.
9. Construct $95\%$ confidence intervals for one mean, a difference of two means and a variance from given data, and interpret each.
10. Compare all the validation methods on bias, variance, cost and use case.
11. Given ten scored test examples, build the ROC table, sketch the curve and compute the AUC by both methods.
12. Explain ensemble theory; compute the majority-vote error for $T = 3, 5$ with a given $p$.
13. Perform one complete AdaBoost round: compute $\epsilon_t$, $\alpha_t$, $Z_t$ and the updated weights.

# Think & Explore (Take-Home)

## Numeric drill

- Data $4, 7, 6, 9, 8, 6$: compute $\bar{x}$, the MLE and unbiased variances, and a $95\%$ $t$-interval for $\mu$.
- Build the ROC table and AUC for eight scored examples of your own.
- Run two AdaBoost rounds by hand on six points and write down $H(\mathbf{x})$.

## Coding (self-learning)

With `scikit-learn`: compare `cross_val_score` at $k = 5, 10$ and LOOCV on one dataset and plot the spread. Then plot ROC curves for three classifiers on one axis, and compare `VotingClassifier`, `BaggingClassifier`, `AdaBoostClassifier` and `StackingClassifier`.

## Reading

- Hastie, Tibshirani & Friedman, *ESL* — Ch. 7 (assessment), Ch. 8, 10, 15–16 (ensembles).
- Bishop, *PRML* — Ch. 1–2 (estimation), Ch. 14 (combining models).
- Zhou, *Ensemble Methods: Foundations and Algorithms*.
- Efron & Tibshirani, *An Introduction to the Bootstrap*.

## Closing thought

You now have the full pipeline: frame the problem (U1), fit linear models (U2), go non-linear (U3), find structure without labels (U4), and **estimate, validate and combine** (U5). **The hardest part is honest evaluation.**

# References

## Textbooks


1. T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning*, 2nd ed., Springer. **[Ch. 7, 8, 15, 16]**
2. C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer.
3. R. V. Hogg, J. McKean, A. T. Craig, *Introduction to Mathematical Statistics*, Pearson.
4. Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*, CRC Press.
5. B. Efron, R. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall.


## Classic papers


6. Y. Freund, R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *JCSS*, 55(1):119–139, 1997.
7. L. Breiman, "Bagging predictors," *Machine Learning*, 24:123–140, 1996.
8. D. H. Wolpert, "Stacked generalization," *Neural Networks*, 5(2):241–259, 1992.
9. T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, 27(8):861–874, 2006.

# Thank You

Any Questions?

*"It is better to be roughly right than precisely wrong."*
— attributed to John Maynard Keynes

End of the Machine Learning (CSE473) lecture series — Units 1 to 5.

Post your doubts on the course page →
https://www.gcjana.in/courses/shardauniversity/2601/machine-learning/#Post-doubts