TMA4268 V2026 Mock Exam 6 — Solution Proposal

Compiled for Anders Bekkevard

Companion to `mock-exam-6.tex` (same directory).

Mock for: May 18, 2026

This document is a worked-solution proposal in the style of the official Stefanie/Sara solutions to the 2024 and 2025 finals. Point values are quoted in the heading of each solved sub-part. Partial-credit hints are inline. Refer back to `mock-exam-6.tex` for the problem statements.

Problem 1 (10 %) — Fill-in-the-blank concepts

Solution *(1 P per blank.)*

(1) **parametric**

(2) **prediction**

(3) **inference**

(4) **irreducible**

(5) **ridge**

(6) **nested cross-validation**

(7) **mini-batch stochastic gradient descent**

(8) **dropout**

(9) **early stopping**

(10) **label smoothing**

Grading: 1 P per correct blank. No partial credit for "close" alternatives. For (5), "lasso" is the textbook trap — lasso does *zero coefficients (corner of L^1 ball), ridge does not. For (6), "stratified cross-validation" is about preserving class balance, not about honest hyperparameter assessment.*

Problem 2 (28 %) — Multiple choice, T/F, short numeric

a) Bias–variance and benign overfitting *(3 %)*

Solution *(0.75 P per statement.)*

(i) **True**. The decomposition is an algebraic identity. The first cross-term $2(f - \hat{f}) \cdot \varepsilon$ vanishes because $\mathbb{E}[\varepsilon] = 0$ and ε is independent of $\hat{f}$; the second cross-term $2(f - \mathbb{E}[\hat{f}]) \cdot (\mathbb{E}[\hat{f}] - \hat{f})$ vanishes after taking expectation over $\hat{f}$ because the second factor has mean zero by construction. Both rely on independence/zero-mean structure, not on any property of $\hat{f}$.

(ii) **True**. In the classical view ridge introduces bias to lower variance, so one usually expects a trade. But *moving from OLS to ridge changes the estimator*, which can shift both curves: if OLS suffers from collinearity its variance explodes, while a modestly regularized ridge fit can have both lower variance *and* negligible bias (because the data don't really distinguish between the collinear coefficients anyway). The prof's "trade-off" skepticism leans on exactly this. Either example argument earns credit.

(iii) **False**. This is benign overfitting / double descent. With $p \gg n$ and an appropriate optimizer (e.g. the min-norm interpolator picked by gradient descent), an interpolating model can still generalize.

(iv) **True**. Larger irreducible noise σ^2 makes complex models spend variance chasing noise; the optimum on the bias/variance trade shifts toward simpler (less flexible) models.

b) Cross-validation, with and without nesting *(4 %)*

Solution *(0.5 %)* **(i) True**. Each LOO/large-k fold uses $n-1$ ($\approx n$) training points, so the CV-fitted models are closer to the model that would be trained on all n data points $\Rightarrow$ less bias. Bias of CV decreases as k grows.

Solution *(0.5 %)* **(ii) True**. The n leave-one-out training sets overlap almost completely, so their per-fold errors are highly correlated; averaging strongly correlated numbers does not reduce variance much, and the LOOCV estimate has higher variance than the 5-fold one. (Prof's L11 line: *"leave-one-out has better bias but K-fold will have a much better variance, and typically you're winning by having less variance."*)

Solution *(1 %)* **(iii) False**. This is the canonical "wrong-way CV" pipeline: the predictor screening uses y, so the held-out folds have already informed which predictors are retained. The CV estimate is biased *downward* (it can be near zero even when the truth is pure noise). Fix: redo the correlation screen *inside* each training fold.

Solution *(1 %)* **(iv) 1-SE rule**: Let $\hat{\theta} = \arg\min_\theta \mathrm{CV}(\theta)$ achieving CV-MSE $m^\star$ with estimated standard error $\mathrm{SE}^\star$ across folds. Pick the *simplest* (largest-λ, smallest model size) θ whose CV-MSE is within $m^\star + \mathrm{SE}^\star$.

Solution *(1 %)* **(v)** Sum = 22.0, so

$$\widehat{\text{CV-MSE}} = \frac{22.0}{10} = \boxed{2.20}.$$

Sample variance of the 10 fold MSEs (using $n-1=9$ in the denominator): squared deviations from 2.20 are $(0.10)^2, (0.20)^2, (0.20)^2, (0.40)^2, (0)^2, (0.30)^2, (0.10)^2, (0.30)^2, (0.20)^2, (0.20)^2$ $= 0.01 + 0.04 + 0.04 + 0.16 + 0 + 0.09 + 0.01 + 0.09 + 0.04 + 0.04 = 0.52$. Sample SD $s = \sqrt{0.52/9} \approx 0.240$.

$$\widehat{\mathrm{SE}}(\widehat{\text{CV-MSE}}) = \frac{s}{\sqrt{10}} \approx \frac{0.240}{3.162} \approx \boxed{0.08}.$$

Accept anywhere in $[0.07, 0.09]$.

c) Mini-batch SGD and learning rate *(3 %)*

Solution *(0.75 P per statement.)*

(i) **True**. Powers of two map cleanly onto GPU memory layouts; the choice is engineering rather than statistical. (Prof L23.)

(ii) **False**. Larger batch sizes *decrease* noise in the gradient estimate, but decreased noise means *less* implicit L^2 regularization, not more. Small-batch SGD is the one that injects the noise that biases toward minimum-norm solutions. Direction reversed.

(iii) **True**. Prof's L24 anecdote: $\eta = 2$ "horrible, bouncing everywhere," $\eta \approx 0.1$ fine.

(iv) **True**. Backprop *is* the chain rule applied so each intermediate quantity is used (not recomputed). SGD is the update rule that consumes backprop's gradients.

d) Odds, log-odds, interactions *(3 %)*

Solution *(1 %)* **(i)** $p = \dfrac{\text{odds}}{1+\text{odds}} = \dfrac{0.25}{1.25} = \boxed{0.20}$.

Solution *(1 %)* **(ii)** Odds multiplier $= \exp(\hat{\beta} \cdot \Delta) = \exp(0.18 \cdot 5) = \exp(0.90) \approx \boxed{2.46}$.

Solution *(1 %)* **(iii)** For a treated patient, a unit increase in `age` changes the log-odds by $\hat{\beta}_{\texttt{age}} + \hat{\beta}_{\texttt{age:treatment}} = 0.04 - 0.03 = 0.01$, so the odds multiply by $\exp(0.01) \approx \boxed{1.010}$.
Grading: full credit for ≈ 1.010. *Half credit for* $\exp(0.04) \approx 1.041$ *(ignoring the interaction — the standard trap). Zero for treating the interaction in isolation.*

e) Collinearity and identifiability *(3 %)*

Solution *(0.75 P per statement.)*

(i) **True**. Exact collinearity makes $X^\top X$ singular, so $(X^\top X)^{-1}$ does not exist and $\hat{\beta} = (X^\top X)^{-1} X^\top y$ is not uniquely defined. Any solution along the affine subspace $\beta_2 = (\text{const}) - 2\beta_1$ achieves the same fitted values.

(ii) **True**. Under near-collinearity, the diagonal of $\sigma^2 (X^\top X)^{-1}$ inflates (the individual SEs blow up), while the variance of an appropriate linear combination of the coefficients (essentially their sum) can stay controlled because the off-diagonal covariances are large and negative and "cancel." This is the "a million minus a million" intuition.

(iii) **True**. With high correlation the individual t-tests can both fail to reject while the partial F-test on the pair rejects strongly. Concluding from the individual t-tests alone that neither predictor matters is exactly the trap; the variables matter *jointly.*

(iv) **False**. K dummies together with an intercept produces a rank-deficient design matrix ($\sum$ of dummies $\equiv 1$, the intercept column). Use $K-1$ dummies and absorb the reference level into the intercept (or drop the intercept and use K dummies).

f) Bootstrap standard error *(3 %)*

Solution *(1 %)* **(i)** With $\bar{\theta}^* = \frac{1}{B}\sum_{b=1}^{B} \hat{\theta}_b^*$,

$$\widehat{\mathrm{SE}}_{\mathrm{boot}}(\hat{\theta}) \;=\; \sqrt{\frac{1}{B-1}\sum_{b=1}^{B} \big(\hat{\theta}_b^* - \bar{\theta}^*\big)^2}.$$

Solution *(1 %)* **(ii)** Mean $\bar{\theta}^* = \frac{1}{4}(2.1 + 2.5 + 1.8 + 2.4) = \frac{8.8}{4} = 2.20$. Squared deviations from 2.20: $(2.1-2.2)^2 = 0.01$, $(2.5-2.2)^2 = 0.09$, $(1.8-2.2)^2 = 0.16$, $(2.4-2.2)^2 = 0.04$. Sum $= 0.30$. Dividing by $B-1 = 3$ gives $0.30/3 = 0.10$, and

$$\widehat{\mathrm{SE}}_{\mathrm{boot}}(\hat{\theta}) = \sqrt{0.10} \approx \boxed{0.32}.$$

Solution *(1 %)* **(iii) (A) False** — bootstrap is *with* replacement. **(B) True** — the bootstrap quantifies the sampling *variability* of $\hat{\theta}$ around itself, not its distance from the true θ; biased estimators stay biased. **(C) True** — $B \in \{1000, 10000\}$ is the standard range.
Grading: 0.5 each for any one wrong, 1 P all correct.

g) Boosting: AdaBoost, gradient boosting, XGBoost *(3 %)*

Solution *(0.75 P per statement.)*

(i) **True**. With $\alpha_m = \frac{1}{2}\log\frac{1-\mathrm{err}_m}{\mathrm{err}_m} > 0$ when $\mathrm{err}_m < 0.5$, the weight-update $w_i \leftarrow w_i \exp(-\alpha_m\, y_i G_m(x_i))$ multiplies misclassified weights by $e^{\alpha_m} > 1$ and correctly classified weights by $e^{-\alpha_m} < 1$; after renormalization the misclassified points end up with strictly higher *relative* weight at the next iteration. (Some textbook conventions instead multiply only the misclassified weights and then renormalize; the relative-weight statement in the question is true under both.)

(ii) **True**. The Hastie–Tibshirani–Friedman (HTF) gradient-boosting algorithm fits the new tree to the negative gradient $r_{im} = -[\partial L/\partial f]_{f=f_{m-1}}$ of the loss; for $L = \frac{1}{2}(y-f)^2$ this gradient is exactly $-(y-f)$, so the residual is the negative gradient up to sign.

(iii) **False**. The directions are swapped. *Random forest:* B is not a tuning parameter (more trees can only help, by variance reduction); just pick "enough." *Boosting:* M *is* a real tuning parameter because boosting *can* overfit — once residuals are noise, additional trees fit that noise.

(iv) **True**. All three are core XGBoost ingredients; row/column subsampling, L^2 leaf-weight penalty (and pruning parameter γ), and second-order Newton-style information are exactly what XGBoost adds on top of plain gradient boosting.

h) Neural-network regularization *(3 %)*

Solution *(0.75 P per statement.)*

(i) **False**. Dropout is a *training-time-only* mechanism; at inference time the full network is used (with the surviving activations rescaled in the standard "inverted dropout" formulation).

(ii) **True**. Prof L24: "20% is very common, never use 50%."

(iii) **True**. This is exactly what early stopping does — the validation-loss curve U-shapes and we return the weights at the minimum.

(iv) **False**. Prof L26: *"I can't think of an example where you'd ever want to train a neural network without a regularization."* Even a small architecture relative to n should use *some* regularizer (mini-batch SGD itself counts).

i) Principal component analysis *(3 %)*

Solution *(1 %)* **(i)** Cumulative PVE: 0.60, 0.85, 0.95, 1.00. The first time the cumulative crosses 0.90 is at $\boxed{3}$ components.

Solution *(1 %)* **(ii)** $z_1^* = \phi_1^\top x^* = 0.60 \cdot 1.0 + (-0.20)\cdot(-0.5) + 0.70 \cdot 2.0 + 0.30 \cdot 1.0 = 0.60 + 0.10 + 1.40 + 0.30 = \boxed{2.40}$.

Solution *(1 %)* **(iii) (A) True** — without standardization the largest-scale predictor dominates the variance computation and tends to define PC1. **(B) True** — loading vectors are unit-norm by construction (otherwise PVE wouldn't be a proportion). **(C) False** — the first principal component is, by definition, the unit-norm linear combination of the centered variables with the *largest* sample variance; later components have progressively smaller variances under the orthogonality constraint.
Grading for (iii): 0.33 per correct mark; full credit for the (T, T, F) pattern. Half credit if (C) is marked True — a very common misread.

Problem 3 (16 %) — Theory, math, and pseudocode

a) The mathy one — bias–variance decomposition and shrinkage *(8 %)*

Solution *(4 %)* **(i) Derivation.** Write $y_0 = f(x_0) + \varepsilon_0$ with $\mathbb{E}[\varepsilon_0] = 0$ and $\mathrm{Var}(\varepsilon_0) = \sigma^2$. Add and subtract $\mathbb{E}[\hat{f}(x_0)]$ inside the squared term:

$$(y_0 - \hat{f}(x_0)) = \underbrace{\big(f(x_0) - \mathbb{E}[\hat{f}(x_0)]\big)}_{=:A} + \underbrace{\big(\mathbb{E}[\hat{f}(x_0)] - \hat{f}(x_0)\big)}_{=:B} + \underbrace{\varepsilon_0}_{=:C}.$$

Square and take expectation. A is non-random (depends only on the true f and on the distribution of $\hat{f}$), so $\mathbb{E}[A^2] = A^2$. Expand:

$$\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = A^2 + \mathbb{E}[B^2] + \mathbb{E}[C^2] + 2A\,\mathbb{E}[B] + 2A\,\mathbb{E}[C] + 2\,\mathbb{E}[BC].$$

Cross-terms vanish.

- $\mathbb{E}[B] = \mathbb{E}[\hat{f}(x_0)] - \mathbb{E}[\hat{f}(x_0)] = 0$, so $2A\,\mathbb{E}[B] = 0$.
- $\mathbb{E}[C] = \mathbb{E}[\varepsilon_0] = 0$, so $2A\,\mathbb{E}[C] = 0$.
- For $\mathbb{E}[BC]$: B is a function of $\hat{f}$ (so of the training set $\mathcal{T}$), $C = \varepsilon_0$ is independent of $\mathcal{T}$ by assumption, hence $B \perp C$ and $\mathbb{E}[BC] = \mathbb{E}[B]\mathbb{E}[C] = 0$.

The three surviving terms:

$$\begin{aligned} A^2 &= \big(f(x_0) - \mathbb{E}[\hat{f}(x_0)]\big)^2 =: \mathrm{Bias}^2[\hat{f}(x_0)], \\ \mathbb{E}[B^2] &= \mathbb{E}\Big[(\hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)])^2\Big] = \mathrm{Var}(\hat{f}(x_0)), \\ \mathbb{E}[C^2] &= \mathrm{Var}(\varepsilon_0) = \sigma^2. \end{aligned}$$

Hence

$$\boxed{\mathbb{E}\Big[(y_0 - \hat{f}(x_0))^2\Big] = \mathrm{Bias}^2[\hat{f}(x_0)] + \mathrm{Var}(\hat{f}(x_0)) + \sigma^2.}$$

Sources of randomness. The outer expectation in $\mathbb{E}[(y_0 - \hat{f}(x_0))^2]$ is taken *jointly* over (a) the random training sample $\mathcal{T}$ used to fit $\hat{f}$, and (b) the test-point noise ε_0 in $y_0 = f(x_0) + \varepsilon_0$. The inner $\mathbb{E}[\hat{f}(x_0)]$ and $\mathrm{Var}(\hat{f}(x_0))$ are over the training-set distribution only. f and x_0 are treated as fixed.

Naming the terms. Squared bias and variance are both *reducible* (one can lower them by choosing a different estimator or different model class). The noise variance σ^2 is *irreducible* — no estimator built from X alone can predict ε_0.

Grading: 2 P for both cross-terms vanishing with justification; 1 P for the "random over what" explanation; 1 P for the names and the reducibility comment. Deduct 0.5 if σ^2 is dropped, 0.5 if the bias is reported without the square. Watch for sign errors in the expansion.

Solution *(3 %)* **(ii) Application to $\hat{\mu}_c = c\bar{y}$.**

We have $\bar{y} = \mu + \bar{\varepsilon}$ with $\bar{\varepsilon} \sim N(0, \sigma^2/n)$, so $\mathbb{E}[\bar{y}] = \mu$ and $\mathrm{Var}(\bar{y}) = \sigma^2/n$. Hence:

$$\begin{aligned} \mathrm{Bias}(\hat{\mu}_c) &= \mathbb{E}[c\bar{y}] - \mu = c\mu - \mu = -(1-c)\mu, \\ \mathrm{Bias}^2(\hat{\mu}_c) &= (1-c)^2\,\mu^2, \\ \mathrm{Var}(\hat{\mu}_c) &= c^2\,\mathrm{Var}(\bar{y}) = \frac{c^2\sigma^2}{n}. \end{aligned}$$

Reducible MSE $= (1-c)^2\mu^2 + c^2\sigma^2/n$. Differentiate w.r.t. c and set to zero:

$$-2(1-c)\mu^2 + 2c\,\sigma^2/n = 0 \implies c\big(\mu^2 + \sigma^2/n\big) = \mu^2 \implies \boxed{c^\star = \frac{\mu^2}{\mu^2 + \sigma^2/n}} \in (0,1).$$

Sanity: when $\sigma^2/n \to 0$ (lots of data, low noise), $c^\star \to 1$ (no shrinkage, just use the sample mean). When $\sigma^2/n \gg \mu^2$ (noisy, small sample), $c^\star \to 0$ — the optimal estimator collapses toward zero, sacrificing all bias to kill variance. *Either comment earns the closing point.*
Grading: 1 P for $\text{Bias}^2 = (1-c)^2\mu^2$*; 1 P for* $\text{Var} = c^2\sigma^2/n$*; 1 P for the optimum* $c^\star$*. Accept stating* $c^\star$ *without the limit comment if the algebra is right.*

Solution *(1 %)* **(iii) "Trade-off" skepticism.** The decomposition is exact, but it is not always true that lowering one term *requires* raising the other. Two cases the prof flagged in the lectures:

(a) *Regularization can reduce both.* Adding a ridge penalty $\lambda\|\beta\|^2$ to a collinear OLS problem shrinks coefficients (small bias hit) but cuts the variance massively because $(X^\top X + \lambda I)^{-1}$ tames the inverse — both terms can drop relative to plain OLS.

(b) *Benign overfitting / double descent.* In the over-parameterized regime ($p \gg n$), once the model interpolates the training data, additional flexibility only changes which zero-training-error solution is picked (minimum-norm by the implicit bias of SGD); variance often *decreases* again as p grows.

Either of the two earns the point.

b) Pseudocode for nested cross-validation *(5 %)*

Solution *(3 %)* **(i) Pseudocode.**

Input: data $D = \{(x_i, y_i)\}_{i=1}^n$; hyperparameter grid $\Lambda = \{\lambda_1, \ldots, \lambda_T\}$; outer folds K, inner folds K'.
Output: an estimate of generalization error of the procedure "ridge regression with λ chosen by inner CV."

1. Randomly partition D into K disjoint *outer* folds $F_1, \ldots, F_K$.
2. **For** $k = 1, \ldots, K$:
   (a) Let $D_k^{\text{out}} = F_k$ (held-out test fold) and $D_k^{\text{tr}} = D \setminus F_k$ (outer training).
   (b) Randomly partition D_k^{tr} into K' *inner* folds $G_1, \ldots, G_{K'}$.
   (c) **For** each $\lambda \in \Lambda$: **For** $j = 1, \ldots, K'$: fit ridge with penalty λ on $D_k^{\text{tr}} \setminus G_j$; record MSE on G_j as $\text{MSE}_{k,j}(\lambda)$.
   Compute inner-CV score $\text{IC}_k(\lambda) = \frac{1}{K'} \sum_{j=1}^{K'} \text{MSE}_{k,j}(\lambda)$.
   (d) Pick $\lambda_k^\star = \arg\min_{\lambda \in \Lambda} \text{IC}_k(\lambda)$. *(The choice can differ across outer folds.)*
   (e) Refit ridge with penalty $\lambda_k^\star$ on all of D_k^{tr}, predict on $D_k^{\text{out}} = F_k$, and record outer-fold MSE $e_k = \text{MSE}_{D_k^{\text{out}}}$.
3. Return $\widehat{\text{Err}} = \frac{1}{K} \sum_{k=1}^K e_k$ as the nested-CV estimate of generalization error of the whole pipeline.

Grading: 1 P for explicit outer/inner loop structure; 1 P for "$\lambda_k^\star$ *chosen using only* D_k^{tr}*, and the outer fold* F_k *never participates in selection"; 1 P for the final aggregation across outer folds. Deduct 0.5 if the analyst's chosen* λ *is presented as a single global* $\hat{\lambda}$ *instead of* $\lambda_k^\star$ *varying with* k*.*

Solution *(1 %)* **(ii)** If you pick λ by single K-fold CV and *report the minimum CV-MSE* as the test-error estimate, you are taking the minimum of a noisy set of estimates — i.e. you are reporting an "argmin" statistic without correcting for the selection bias. The reported CV-MSE will be *optimistically biased downward*: the lower the noise at the picked λ relative to others, the more λ "benefits" from the selection. Honest assessment requires an outer loop whose held-out folds never participated in selecting λ.

Solution *(1 %)* **(iii)** If you screen predictors using their correlation with y *outside* the CV loop, the held-out CV folds have already informed the variable-selection step, so the supposed

"held-out" data is no longer held out — it has helped shape the model. The CV estimate is therefore biased *downward*, and can be near zero even when the truth is pure noise (the canonical ISLP exercise: $p = 5000$ random predictors, two equiprobable labels; naive pipeline reports 0% error). **Structural fix:** redo the correlation screen *inside each training fold*, using only that fold's training portion.

c) Backpropagation in a tiny network *(3 %)*

Solution *(2 %)* **(i) Derivation.** Forward pass: $z = w_1 x + b_1$, $h = \sigma(z)$, $\hat{y} = w_2 h + b_2$, $L = \frac{1}{2}(\hat{y} - y)^2$.

Output-layer gradient. Chain rule:

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2} = (\hat{y} - y) \cdot h.$$

Hidden-layer gradient.

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial h} \cdot \frac{\partial h}{\partial z} \cdot \frac{\partial z}{\partial w_1} = (\hat{y} - y) \cdot w_2 \cdot h(1 - h) \cdot x.$$

Grading: 1 P for $\partial L/\partial w_2 = (\hat{y} - y) \cdot h$; 1 P for the full chain through to $\partial L/\partial w_1 = (\hat{y} - y) \cdot w_2 \cdot h(1 - h) \cdot x$. Accept either $\sigma'(z)$ or $h(1 - h)$.

Solution *(1 %)* **(ii) Numerical.** $z = 0.5 \cdot 2 + 0 = 1.0$. $h = \sigma(1) = 1/(1 + e^{-1}) = 1/(1 + 0.3679) = 1/1.3679 \approx 0.7311$. $\hat{y} = 1 \cdot 0.7311 + 0 \approx \boxed{0.731}$. $\hat{y} - y \approx -0.269$. $h(1 - h) \approx 0.7311 \cdot 0.2689 \approx 0.1966$. $\frac{\partial L}{\partial w_1} = (-0.269) \cdot 1 \cdot 0.1966 \cdot 2 \approx \boxed{-0.106}$.

Accept any answer in $[-0.110, -0.100]$. Sign matters: a negative gradient means SGD will increase w_1 to reduce loss, which is correct (current $\hat{y}$ undershoots $y = 1$, and increasing w_1 pushes z up, hence h and $\hat{y}$ up).

Problem 4 (18 %) — Concrete compressive strength

a) Linear regression with collinearity, interactions, encoding *(8 %)*

Solution *(1 %)* **(i)** Count rows in the coefficient table: intercept, `cement`, `slag`, `water`, `superplast`, `coarse_agg`, `fine_agg`, log(`age`), two `mix_type` dummies, two `cement:mix_type` interactions $= \boxed{12}$ parameters. Sanity: residual d.f. $= 700 - 12 = 688$. ✓

Solution *(2 %)* **(ii) Collinearity diagnosis.**

- **(a)** The variance-covariance matrix of OLS is $\mathrm{Var}(\hat{\beta}) = \sigma^2 (X^\top X)^{-1}$. When two columns of X are near-collinear, $X^\top X$ has a near-zero eigenvalue in their joint direction; the corresponding diagonal entries of $(X^\top X)^{-1}$ blow up, so the individual SEs of $\hat{\beta}_{\texttt{water}}$ and $\hat{\beta}_{\texttt{superplast}}$ inflate, killing their individual t-statistics. The *sum* $\beta_{\texttt{water}} + \beta_{\texttt{superplast}}$ (or any other well-conditioned combination orthogonal to the collinearity direction) remains well-estimated, which is why the joint F-test rejects: the model can be sure that *some* combination of `water` and `superplast` matters, just not which of the two carries the effect.
- **(b) No.** Dropping both predictors on the basis of their individual t-tests would be wrong: the joint F-test on the pair rejects with $p < 10^{-9}$, which means at least one of them belongs in the model. One sensible response is to drop one and keep the other (and live with the fact that you can't disentangle them), or to use a regularized fit (ridge / lasso / elastic net) that handles correlated predictors gracefully.

Grading: 1 P for the $\mathrm{Var}(\hat{\beta}) = \sigma^2(X^\top X)^{-1}$ *argument plus the SE-inflation conclusion; 1 P for explicitly rejecting the "drop both" inference and giving a reasonable remedy.*

Solution *(2 %)* **(iii) Cement effect by mix type.** A unit increase in `cement` changes the predicted `strength` (in MPa) by:

$$\begin{aligned}
\text{Standard mix:} &\quad \hat{\beta}_{\texttt{cement}} = 0.110 \text{ MPa per kg/m}^3. \\
\text{High-strength:} &\quad \hat{\beta}_{\texttt{cement}} + \hat{\beta}_{\texttt{cement:mix_high}} = 0.110 + 0.030 = \boxed{0.140} \text{ MPa per kg/m}^3. \\
\text{Lightweight:} &\quad \hat{\beta}_{\texttt{cement}} + \hat{\beta}_{\texttt{cement:mix_light}} = 0.110 + (-0.020) = \boxed{0.090} \text{ MPa per kg/m}^3.
\end{aligned}$$

The question asks for high-strength and lightweight; both shown.
Grading: 1 P each. Half credit if the student reports 0.030 *or* −0.020 *alone (the interaction term in isolation — the standard trap). Zero if both terms are reported as* 0.110 *(interaction entirely ignored).*

Solution *(1 %)* **(iv) Encoding flip.** Switching the reference level from *standard* to *lightweight*:

- **(a) Residual standard error: unchanged** (the column space of X is identical; only the basis changes).
- **(b) Coefficient on `cement`: changes** (it is now the slope *for the lightweight reference mix*, which is $0.110 - 0.020 = 0.090$ instead of 0.110).
- **(c) Coefficient on `mix_high`: changes** (it is now the contrast "high-strength vs. lightweight" instead of "high-strength vs. standard").
- **(d) Predicted strength of any given mix: unchanged** — fitted values are invariant under reparameterization.

Grading: 0.25 P per part. All-or-nothing on each. The (a) and (d) "unchanged" items are the most commonly missed; flag if the student claims that "fits change" under re-encoding.

Solution *(1 %)* **(v) CI vs. PI.**

- **(a)** A 95% *confidence interval* for $\mathbb{E}[\texttt{strength} \mid x_0]$ contains the (population-level) mean strength of the sub-population of concrete mixes with predictor values x_0; a 95% *prediction interval* contains the future `strength` of *one new individual* mix with predictor values x_0.
- **(b)** The prediction interval is wider. Its variance picks up an extra additive σ^2 term for the irreducible noise on the individual observation: $\hat{y}_0 \pm t\,\hat{\sigma}\sqrt{1 + x_0^\top (X^\top X)^{-1} x_0}$ vs. $\hat{y}_0 \pm t\,\hat{\sigma}\sqrt{x_0^\top (X^\top X)^{-1} x_0}$. As $n \to \infty$ the CI shrinks to a point but the PI stays bounded below by $\pm t\,\hat{\sigma}$.

Solution *(1 %)* **(vi) Fanning residuals.** The constant-variance (homoscedasticity) assumption is violated. Common remedies: transform the response (e.g. $\log \texttt{strength}$ or $\sqrt{\texttt{strength}}$), or use weighted least squares with weights inversely proportional to the local residual variance.

b) Ridge regression with 10-fold CV *(5 %)*

Solution *(1 %)* **(i)** The ridge penalty $\lambda \sum_j \beta_j^2$ is scale-dependent; without standardization a predictor in larger units (e.g. `cement` in kg/m^3 vs. a binary dummy) is penalized far less because its coefficient is naturally small. Standardizing to mean 0 and variance 1 puts every predictor on the same footing so shrinkage reflects signal, not units.

Solution *(1 %)* **(ii)** Ridge minimizes

$$\hat{\beta}^R_\lambda \;=\; \arg\min_\beta \; \sum_{i=1}^{n}(y_i - \beta_0 - x_i^\top \beta)^2 \;+\; \lambda \sum_{j=1}^{p} \beta_j^2.$$

As $\lambda \to 0$, the penalty disappears and $\hat{\beta}^R_\lambda \to \hat{\beta}^{\text{OLS}}$. As $\lambda \to \infty$, every coefficient is shrunk to 0 and the fit collapses to the unpenalized intercept ($\bar{y}$).

Solution *(1 %)* **(iii) 1-SE rule.** Among all λ whose CV-MSE is within one standard error (across folds) of the minimum CV-MSE, choose the *largest* (most regularized, simplest model). Formally,

$$\hat{\lambda}_{\text{1SE}} = \max\Big\{\lambda :\; \text{CV}(\lambda) \le \text{CV}(\hat{\lambda}_{\min}) + \widehat{\text{SE}}(\text{CV}(\hat{\lambda}_{\min}))\Big\}.$$

Solution *(2 %)* **(iv) Bias–variance interpretation.**

- **(a)** OLS (test MSE 41.5) is beaten by ridge at $\hat{\lambda}_{\min}$ (test MSE 39.8). This suggests OLS is *over-fitting* on this dataset — variance is the binding term, and a moderate amount of bias buys a bigger reduction in variance, lowering test MSE. The concrete-strength data are a plausible case for this: with the strong collinearity between `water` and `superplast`, $X^\top X$ is poorly conditioned, OLS coefficients are unstable, and ridge specifically tames that instability.
- **(b)** The 1-SE choice sits in between because it is *more* regularized than $\hat{\lambda}_{\min}$. It picks up extra bias (every coefficient is shrunk further), giving up a little of the variance saving; the net effect is worse than $\hat{\lambda}_{\min}$ on test MSE here. We accept the 1-SE penalty in exchange for a simpler, more stable model.

c) Gradient boosting *(5 %)*

Solution *(2 %)* **(i) One iteration.**

Initialize $\hat{f}(x) \leftarrow \bar{y}$. Let $r_i = y_i - \hat{f}(x_i)$ be the current residuals. At iteration $b+1$:

1. Fit a regression tree $\hat{g}^{(b+1)}(x)$ of (small) interaction depth d on the data (x_i, r_i) — i.e. *the new tree is fit to the current residuals.*
2. Update the ensemble: $\hat{f}(x) \leftarrow \hat{f}(x) + \nu \cdot \hat{g}^{(b+1)}(x)$.
3. Update residuals: $r_i \leftarrow r_i - \nu \cdot \hat{g}^{(b+1)}(x_i)$, or equivalently recompute $r_i = y_i - \hat{f}(x_i)$ with the updated $\hat{f}$.

After M iterations the final predictor is $\hat{f}(x) = \bar{y} + \sum_{b=1}^{M} \nu\, \hat{g}^{(b)}(x)$.

Where ν enters: in step 2 (and equivalently step 3), shrinking each new tree's contribution to the ensemble. With squared-error loss the residual is the negative gradient of the loss, so this is a special case of the general gradient-boosting recipe.
Grading: 1 P for "fit the new tree to current residuals"; 1 P for the update with the ν shrinkage shown explicitly. Accept either residuals-first or gradient-first framing.

Solution *(2 %)* **(ii) Hyperparameters.**

- *ν (shrinkage / learning rate):* fix small ($\nu \le 0.1$ typical) — not really tuned by CV, just chosen small.
- *d (interaction depth):* tune by CV. Small d (e.g. 1–4): each tree captures only low-order interactions, lowest variance per tree; larger d admits higher-order interactions, more variance per tree.

- *M (number of trees):* tune by CV (or early stopping on a validation curve). Boosting *can* overfit if M is too large.
- **Coupling.** If ν is halved, each tree contributes half as much, so roughly *twice as many* trees are needed to reach the same overall fit. Practically: $M \cdot \nu \approx$ constant for a given total "capacity."

Solution *(1 %)* **(iii) Ridge vs. boosting gap.** Ridge merely shrinks linear coefficients; boosting fits a sum of (small) trees and can therefore capture *non-linear shapes* and *interactions among predictors* that no linear model can. The large drop from 39.8 (ridge) to 26.3 (boosting) suggests the predictors enter the truth non-linearly and/or with non-trivial interactions — which is consistent with the prior knowledge that, say, `cement`, `water`, and `age` affect strength through complex chemical-curing dynamics that no linear-in-the-coefficients model would capture.

Problem 5 (28 %) — Telecom customer churn

a) Logistic regression with interaction *(8 %)*

Solution *(2 %)* **(i)** For a one-unit increase in `monthly` the log-odds change by the slope at that contract level.

$$\text{Month-to-month (reference)} : \Delta\hat{\eta} = 0.030;$$
$$\text{Two-year} : \Delta\hat{\eta} = \hat{\beta}_{\texttt{monthly}} + \hat{\beta}_{\texttt{monthly:contract\_2yr}} = 0.030 + (-0.040) = -0.010.$$

For an increase of $\Delta = 10$ EUR:

$$\text{Month-to-month: odds} \times \exp(10 \cdot 0.030) = \exp(0.30) \approx \boxed{1.350}.$$
$$\text{Two-year: odds} \times \exp(10 \cdot (-0.010)) = \exp(-0.10) \approx \boxed{0.905}.$$

Interpretation: on a month-to-month contract, extra EUR is associated with *more* churn; on a two-year contract the sign flips and the same EUR increase is associated with slightly *lower* churn (the customer is locked in and modest price changes don't drive them out).
Grading: 1 P each. Half credit if the student reports $\exp(0.030)$ *and* $\exp(-0.010)$ *alone (per-unit factors, ignoring the requested* $\Delta = 10$*). Zero if the two-year case drops the interaction term entirely.*

Solution *(1 %)* **(ii)** The main-effect coefficient 0.030 is the per-EUR effect of `monthly` on log-odds *only on the reference (month-to-month) contract*, because the interactions `monthly:contract_1yr` and `monthly:contract_2yr` shift the slope on the other levels. To describe the effect of `monthly` on churn "on average" across contracts you must combine $\hat{\beta}_{\texttt{monthly}}$ with the relevant interaction coefficient on a per-contract basis:

$$\text{slope for contract level } c \;=\; \hat{\beta}_{\texttt{monthly}} + \hat{\beta}_{\texttt{monthly:}c} \quad \text{(zero for the reference level).}$$

Solution *(3 %)* **(iii) Predicted probability.** For the given customer, the linear predictor is

built term by term:

$$\begin{aligned}
\hat{\eta} &= -1.90 && \text{(intercept)}\\
&+ (-0.040)\cdot 12 = -0.480 && (\texttt{tenure})\\
&+ 0.030 \cdot 70 = 2.100 && (\texttt{monthly})\\
&+ (-0.010)\cdot 45 = -0.450 && (\texttt{age})\\
&+ 1.50 && (\texttt{contract_1yr})\\
&+ (-0.020)\cdot 70 = -1.400 && (\texttt{monthly:contract_1yr})\\
&+ 0 && (\texttt{senior} = 0,\ \texttt{contract_2yr} = 0)\\
&= \boxed{-0.630}.
\end{aligned}$$

Sigmoid:

$$\hat{p} = \frac{1}{1+e^{-\hat{\eta}}} = \frac{1}{1+e^{0.630}} = \frac{1}{1+1.878} = \frac{1}{2.878} \approx \boxed{0.348}.$$

Accept any answer in $[0.34, 0.36]$.
Grading: 1 P for assembling all relevant terms (incl. the one-year contract main effect and its monthly-interaction; deduct if either is dropped); 1 P for the value of $\hat{\eta} \approx -0.63$; 1 P for the correctly applied sigmoid. Deduct 0.5 for using `contract_2yr` *instead of* `contract_1yr`*.*

Solution *(1 %)* **(iv)** $\hat{p} \approx 0.35 < 0.5$, so the customer is predicted **not** to churn at the default threshold. But the base rate of churn is $\approx 26\%$, so $\hat{p} = 0.35$ is already well above the base rate; a more sensible classification threshold might be lower (e.g. around 0.26, the base rate), in which case the same customer would be flagged as a likely churner — 0.5 is a poor default here.

Solution *(1 %)* **(v) Re-encoding.** If the colleague swaps the reference level of `contract` from *month-to-month* to *two-year*:

- **(a) No change.** The predicted probability of churn for any given customer is invariant under reparameterization of the design matrix.
- **(b) Yes, changes.** The two dummies are now "month-to-month vs. two-year" and "one-year vs. two-year," which is a different basis; the estimates and SEs are different numbers (though they encode the same fitted model).
- **(c) No change.** The overall fit (residual deviance, $-2\log L$, AIC) is invariant under reparameterization of the same model.

b) AdaBoost with pseudocode *(6 %)*

Solution *(3 %)* **(i) Pseudocode.**

Input: training data $\{(x_i, y_i)\}_{i=1}^n$ with $y_i \in \{-1, +1\}$; weak-learner class $\mathcal{G}$; number of rounds M.
Output: ensemble classifier $G(x)$.

1. Initialize weights $w_i^{(1)} = \frac{1}{n}$ for $i = 1, \ldots, n$.
2. **For** $m = 1, \ldots, M$:
   (a) Fit weak classifier $G_m \in \mathcal{G}$ to the training data using weights $w_i^{(m)}$.
   (b) Compute the weighted misclassification error
   $$\mathrm{err}_m \;=\; \frac{\sum_{i=1}^n w_i^{(m)} \mathbb{1}\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^n w_i^{(m)}}.$$
   (c) Compute the classifier weight
   $$\alpha_m \;=\; \tfrac{1}{2} \log \frac{1 - \mathrm{err}_m}{\mathrm{err}_m}.$$
   (d) Update observation weights
   $$w_i^{(m+1)} \;=\; w_i^{(m)} \cdot \exp\big(-\alpha_m \, y_i \, G_m(x_i)\big), \quad i = 1, \ldots, n,$$
   and renormalize so $\sum_i w_i^{(m+1)} = 1$.
3. Return the final classifier $G(x) = \mathrm{sign}\Big(\sum_{m=1}^M \alpha_m \, G_m(x)\Big)$.

Grading: 1 P for the weight initialization, the err_m *definition, and the classifier weight* α_m *(the trio is standard); 1 P for the multiplicative weight update with* $y_i G_m(x_i) \in \{\pm 1\}$*; 1 P for the final weighted-vote classifier with* sign*. Accept the equivalent "factor-of-2" convention* $\alpha_m = \log \frac{1-\mathrm{err}_m}{\mathrm{err}_m}$ *(without the* $\frac{1}{2}$*); the prof has used both. Deduct 0.5 if labels are coded as* $\{0, 1\}$ *instead of* $\{-1, +1\}$ *— the weight update only works with the signed convention.*

Solution *(1 %)* **(ii)** With $\mathrm{err}_m = 0.30$:

$$\alpha_m \;=\; \tfrac{1}{2} \log \frac{1 - 0.30}{0.30} = \tfrac{1}{2} \log \frac{0.70}{0.30} = \tfrac{1}{2} \log(2.333) = \tfrac{1}{2} \cdot 0.847 \approx \boxed{0.42}.$$

Accept any answer in $[0.40, 0.44]$. *If the student uses the unscaled convention* $\alpha_m = \log \frac{1-\mathrm{err}_m}{\mathrm{err}_m} \approx 0.85$*, accept that too.*

Solution *(1 %)* **(iii) Weight updates.** Under the $\{-1, +1\}$ convention with $\alpha_m \approx 0.42$:

- **Misclassified** ($y_i G_m(x_i) = -1$): $\exp(+\alpha_m) = \exp(0.42) \approx \boxed{1.53}$.
- **Correctly classified** ($y_i G_m(x_i) = +1$): $\exp(-\alpha_m) = \exp(-0.42) \approx \boxed{0.65}$.

(With the unscaled $\alpha_m \approx 0.85$: misclassified factor ≈ 2.33, correct factor ≈ 0.43. Either convention is accepted, provided consistent with the student's choice in (ii).)

Solution *(1 %)* **(iv) Why weak learners (bias–variance).** Boosting builds a high-bias low-variance *ensemble* out of *many* simple learners, each contributing a small, shrunken correction. If the base learner were itself deep (low-bias, high-variance), each tree would already fit much of the training signal — including noise — and only a few iterations would suffice; the ensemble would inherit the per-tree variance and overfit aggressively. Shallow "stumps" have high bias individually but low variance; the sequential corrections drive bias down without re-introducing the variance that deep trees would bring back in.

c) Neural-network classifier with regularization *(8 %)*

Solution *(2 %)* **(i)** Each layer contributes $(\#\text{inputs}+1)\cdot\#\text{outputs}$ parameters (the +1 counts the bias).

$$\begin{aligned} \text{input} \to \text{hidden}: &\quad (7+1)\cdot 16 = 128, \\ \text{hidden} \to \text{output}: &\quad (16+1)\cdot 1 = 17, \\ \text{Total}: &\quad \boxed{128+17=145} \text{ parameters.} \end{aligned}$$

Grading: 1 P for the per-layer breakdown, 1 P for the total. Deduct 0.5 if any biases are dropped (e.g. a student forgetting all biases reports $7\cdot 16+16\cdot 1=128$*, or forgetting only the output bias reports* $128+16=144$*). The dropout, label smoothing, and early stopping add* no *parameters; the sigmoid on the output adds none.*

Solution *(2 %)* **(ii) Forward pass.**

$$z = b + \sum_{j=1}^{7} w_j x_j = -0.2 + 0.5\cdot 1 + (-0.3)\cdot 0 + 0.1\cdot 2 + 1.0\cdot 1 + (-0.5)\cdot 0 + 0.2\cdot(-1) + 0.4\cdot 1.$$

Term by term: $0.5+0+0.2+1.0+0-0.2+0.4=1.9$; with the bias, $z=-0.2+1.9=\boxed{1.7}$. ReLU: $h=\max(0,z)=\max(0,1.7)=\boxed{1.7}$.

Solution *(1 %)* **(iii)** A network with only identity (linear) activations collapses across layers: $\hat{y} = W_L\, W_{L-1}\cdots W_1\, x + (\text{biases}) = W_{\text{eff}}\, x + b_{\text{eff}}$, where W_{eff} is a single matrix. The composite model is mathematically equivalent to ordinary (multivariate) linear regression — depth adds no representational capacity, because the composition of linear maps is itself linear.

Solution *(1 %)* **(iv) The three regularizers.**

- **Dropout:** During *training only*, randomly zero out a fraction (here 20%) of the hidden-layer activations on each forward pass; the surviving activations are rescaled. At *inference* the full network is used (no dropout, no rescaling beyond what was applied in training).
- **Early stopping:** During *training*, monitor the loss on a held-out validation set; stop training when the validation loss has stopped improving, and return the weights from the validation-loss minimum (not the final iteration).
- **Label smoothing:** During *training only*, replace the hard target $y \in \{0,1\}$ by a softened target $\big((1-\varepsilon)\,y+\varepsilon/2\big)$ for each class (here $\varepsilon=0.05$); discourages the network from driving its logits to $\pm\infty$ to approach $\{0,1\}$.

Solution *(1 %)* **(v) Dropout rate of** 50%**.** Not a good idea — the prof's convention is "20% is common, 50% is too aggressive and almost never used." Dropping half of the hidden units in every forward pass strips the network's representational capacity to the point where it underfits.

Solution *(1 %)* **(vi) Training without regularization.** The classic over-fitting signature: training loss continues to decrease monotonically (eventually toward zero) while the validation loss drops initially, reaches a minimum, and then *rises* as the model memorizes training-set noise. The gap between train and validation loss widens with epochs.

d) LDA vs. QDA, plus random-forest variable importance *(4 %)*

Solution *(2 %)* **(i)** LDA's two main modelling assumptions:

1. **Gaussian class-conditional densities.** Within each class, the predictors are multivariate normal: $\mathbf{X} \mid Y=k \sim \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$.

2. **Common (pooled) covariance.** All classes share the *same* $\boldsymbol{\Sigma}$.

Why prefer LDA when one class is small. QDA estimates a separate covariance matrix $\boldsymbol{\Sigma}_k$ per class — here $K = 2$ with the minority class "churners" having roughly 900 training points. QDA must estimate $p(p+1)/2$ covariance parameters from those 900 points; with $p = 5$ that is 15 parameters from 900 obs, not catastrophic, but the variance of $\widehat{\boldsymbol{\Sigma}}_{\text{churn}}$ is much higher than the pooled $\widehat{\boldsymbol{\Sigma}}$ used by LDA. The bias QDA pays for assuming distinct covariances is small if they really are similar; the variance saving of LDA's pooled estimate is real either way. In a small-class regime LDA typically wins on test error — as is consistent here (0.196 for LDA, 0.221 for QDA).

Solution *(1 %)* **(ii)** For classification the standard default is $\texttt{mtry} = \sqrt{p} = \sqrt{7} \approx 2.65$, conventionally rounded up to $\texttt{mtry} = 3$. We want $\texttt{mtry} < p$ so the individual trees are forced to use different predictors at the top splits and become *decorrelated*; averaging decorrelated trees yields better variance reduction than averaging the highly-correlated trees that bagging ($\texttt{mtry} = p$) would produce.

Solution *(1 %)* **(iii) What variable importance *cannot* tell you.** It does not tell you the *direction* (sign) or *shape* of any predictor's effect on $\hat{p}$(churn). For example, the plot tells you that `tenure` matters a lot, but it does not tell you whether long tenure raises or lowers the churn probability, nor what the functional form of that relationship looks like (linear? saturating?). To recover those, you would compute partial dependence plots (or ICE curves) or compare with a parametric model. *Also acceptable: it says nothing about causality — a predictor that proxies for a confounder will have high importance without having any causal effect.*

e) Class imbalance *(2 %)*

Solution *(1 %)* **(i)** A constant "no churn" classifier mis-classifies exactly the 26% of customers who churn and correctly classifies the rest, so its test error rate is $\approx \boxed{0.26}$.

Solution *(1 %)* **(ii)** The naive baseline (0.26) is *worse* than LDA (0.196), QDA (0.221), and the random forest (0.179), so plain accuracy does usefully discriminate here — but only modestly, and at the price of saying nothing about the model's ability to catch the (commercially valuable) churners. In this kind of imbalanced setting one should privilege the *pair* (**sensitivity, specificity**), or a threshold-independent summary such as **AUC**, or a cost-weighted metric reflecting that one retained customer is worth far more than the cost of one unnecessary retention offer. Reporting only the overall test error rate hides the fact that, at threshold 0.5, all three classifiers are still missing a substantial fraction of churners (sensitivity well below 1).

End of solution proposal. Total awarded: $10 + 28 + 16 + 18 + 28 = 100$ points.