TMA4268 V2026 Mock Exam 9 — Solutions

Compiled for Anders Bekkevard

Companion to `mock-exam-9.tex` (same directory).

Mock for: May 18, 2026

Worked-solution proposal in the style of the official Stefanie/Sara solutions to the 2024 and 2025 finals. Point values appear in each subsection heading. Partial-credit hints are inline. Refer back to `mock-exam-9.tex` for the problem statements.

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

Solution *(1 P per blank.)*

(1) **regression**

(2) **inference**

(3) **linear**

(4) **QDA**

(5) L_2 **norm**

(6) **permutation-based**

(7) 1/3

(8) **simplest**

(9) **exponential loss**

(10) **second-order**

Grading: 1 P per correct blank, no partial credit for "close" alternatives. Common traps: (3) "quadratic" confuses LDA with QDA. (5) "training loss" is wrong — every interpolating solution has zero training loss, the implicit-regularization claim is about which one of those zero-loss solutions SGD lands on. (6) "Gini-based" is the impurity importance the prof explicitly disrecommended. (7) $1/e \approx 0.368$ *are out-of-bag; this rounds to* $1/3$*, not* $2/3$ *(*$2/3$ *would be the in-bag fraction). (10) "first-order only" is plain GBM, not XGBoost.*

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

a) Bias–variance, numerically *(3 %)*

Solution

(i) *(1 %)* $\text{Bias}^2[\hat{f}(x_0)] = (\mathbb{E}[\hat{f}(x_0)] - f(x_0))^2 = (4.6 - 5.0)^2 = (-0.4)^2 = \boxed{0.16}$.

(ii) *(1 %)* $\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = \text{Bias}^2 + \text{Var}[\hat{f}(x_0)] + \sigma^2 = 0.16 + 0.30 + 0.40 = \boxed{0.86}$.

(iii) *(1 %)* For $\hat{f}'$: $\text{Bias}^2 = (5.0 - 5.0)^2 = 0$, $\text{Var} = 0.85$, $\sigma^2 = 0.40$ (unchanged), so $\mathbb{E}[(y_0 - \hat{f}'(x_0))^2] = 0 + 0.85 + 0.40 = 1.25$. Since $0.86 < 1.25$, the original $\hat{f}$ is **better**: $\hat{f}'$ trades all of its squared bias for too much variance. Classic "small bias buys lots of variance" regime — flexibility hurt here.

See ISLP §2.2.2 for the decomposition.

b) CV: bias, variance, and the one-SE rule *(4 %)*

Solution

(i) *(1 %)* **True**. LOOCV trains on $n - 1$ points, 5-fold on $4n/5$. The closer the training-set size to n, the smaller the systematic upward bias of the CV estimate of test error at sample size n. (See ISLP §5.1.4.)

(ii) *(1 %)* **False**. The standard reversal: LOOCV has higher variance because the n training sets overlap almost completely, so the n fold errors are strongly correlated and their average has larger variance than the average of 5 or 10 much-less-correlated fold errors. The "average over more terms = lower variance" intuition fails when the terms are correlated.

(iii) *(2 %)* Minimum CV-MSE is at $\lambda = 2$, $\text{CV}^\star = 0.458$ with $\widehat{\text{SE}}^\star = 0.022$.

$$\text{CV}^\star + \widehat{\text{SE}}^\star = 0.458 + 0.022 = \boxed{0.480}.$$

Candidate λ values whose CV-MSE is ≤ 0.480:

$$\{\lambda = 0.50\ (0.471),\ \lambda = 1.00\ (0.464),\ \lambda = 2.00\ (0.458),\ \lambda = 5.00\ (0.466)\}.$$

($\lambda = 0.10$ gives $0.485 > 0.480$ and $\lambda = 10$ gives $0.487 > 0.480$, both excluded.) The one-SE rule picks the *simplest* (here largest) such λ, so $\hat{\lambda}_{\text{1SE}} = \boxed{5.00}$.

c) Backpropagation: what it is and what it is not *(3 %)*

Solution

(i) *(1 %)* **False**. Backprop is the *gradient-computation* algorithm; the parameter-update rule is (mini-batch) SGD (or Adam, etc.). Without backprop you could still in principle compute gradients (e.g. by finite differences); it would just be hopelessly slow at NN scale.

(ii) *(1 %)* **True**. Backpropagation = the chain rule applied to the computation graph of the network, organised to reuse the pre-activations and activations stored on the forward pass.

(iii) *(1 %)* **False**. Plain backprop assumes a *directed acyclic* graph; cyclic / recurrent connections require *backpropagation through time*, which unrolls the recurrence into a feedforward DAG over time steps before applying the chain rule.

d) Mini-batch SGD *(3 %)*

Solution

(i) *(1 %)* **True**. With $\mathcal{B}$ a uniform random subset of size m, $\mathbb{E}[\widehat{\nabla L}] = \frac{1}{N}\sum_{i=1}^{N} \nabla L_i$ = full-batch gradient, and $\text{Var}(\widehat{\nabla L})$ scales like $1/m$, strictly positive — noisier than the full-batch gradient.

(ii) *(1 %)* 200,000/256 = 781.25, so $\boxed{781}$ full mini-batches per epoch, plus a final partial batch of 64 examples (whether to count this as an extra update is implementation-dependent; $\lfloor N/m \rfloor = 781$ is the conventional answer, $\lceil N/m \rceil = 782$ is also acceptable).

(iii) *(1 %)* **False**. Powers of two are a hardware convention (GPU memory access and SIMD lane sizes prefer 2^k); there is no statistical-theoretic optimum that selects $32, 64, \ldots, 512$.

e) Dropout, label smoothing, and early stopping *(3 %)*

Solution *(1 P per statement.)*

(i) **False**. The course recommendation was $p \approx 0.2$ as a default. $p = 0.5$ exists but is far from a universal recommendation; smaller rates are used in serious applications, not just toy examples.

(ii) **True**. Label smoothing replaces $(0, \ldots, 1, \ldots, 0)$ with $(\varepsilon/(C-1), \ldots, 1-\varepsilon, \ldots)$; one stated motivation is exactly that not every training label is trustworthy, and softening the target discourages over-confident logits.

(iii) **True**. Early stopping monitors validation error each epoch and returns the parameters at the epoch where validation error first stopped improving (the validation minimum), *not* the final epoch's parameters (where training error is lowest but validation error has typically started rising).

f) XGBoost vs. vanilla gradient boosting *(3 %)*

Solution

(i) **True**. XGBoost uses the gradient *and* the Hessian of the loss with respect to the current ensemble predictions when choosing splits and computing leaf weights (a second-order Newton step in function space); vanilla gradient boosting uses only the gradient.

(ii) **True**. The XGBoost regularizer is $\Omega(T) = \gamma|T| + \frac{1}{2}\lambda\|w\|_2^2 + \alpha\|w\|_1$, where w are the leaf-output values — both an L_1 and an L_2 penalty on the leaves plus a per-leaf complexity penalty $\gamma|T|$.

(iii) **False**. XGBoost still has a learning-rate / shrinkage hyperparameter η (here ν); the Newton step is just the per-tree weight before shrinkage. Each tree's contribution is still $\eta \cdot \hat{T}_m$, exactly as in vanilla GBM.

g) Collinearity in regression output *(3 %)*

Solution

(i) *(1 %)* **False**. Standardisation rescales columns but does not change the *angle* between them; the rank of $\mathbf{X}^\top\mathbf{X}$ is unchanged. If two columns are nearly proportional before standardisation they remain nearly proportional after, and the matrix remains near-singular. (Standardisation matters for ridge/lasso and for unit-free interpretation, not for the rank.)

(ii) *(1 %)* **True**. Textbook collinearity fingerprint: $\hat{\beta}_4, \hat{\beta}_5$ have inflated standard errors (the diagonal of $\sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$ is large in the near-degenerate direction), so individual t-tests fail to reject, but the joint partial-F test on the pair (or the overall regression) remains highly significant because the pair jointly explains y — the data just cannot allocate the effect between the two near-proportional columns.

(iii) *(1 %)* **True**. Adding a predictor mechanically increases R^2 (or leaves it unchanged in the degenerate case), but the adjusted $R^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$ penalises p and can drop when the gain in R^2 is too small to overcome the p adjustment.

h) Logistic regression with an interaction *(3 %)*

Solution The linear predictor is $\hat{\eta} = -10.0 + 2.5\,\texttt{bal} - 0.6\,\texttt{stu} + 0.3\,\texttt{bal}\cdot\texttt{stu}$. A unit increase in `balance` (i.e. +\$1,000) changes $\hat{\eta}$ by $\hat{\beta}_{\texttt{bal}} + \hat{\beta}_{\texttt{bal:stu}}\cdot\texttt{stu}$, so the odds multiply by $\exp(\hat{\beta}_{\texttt{bal}} + \hat{\beta}_{\texttt{bal:stu}}\cdot\texttt{stu})$.

(i) *(1 %)* Non-student ($\texttt{stu} = 0$): odds factor $= \exp(2.5) \approx \boxed{12.18}$.

(ii) *(1 %)* Student ($\texttt{stu} = 1$): odds factor $= \exp(2.5 + 0.3) = \exp(2.8) \approx \boxed{16.44}$.

(iii) *(1 %)* **False**. The conclusion assumes no interaction. With the interaction, the multiplicative effect of `student` at a given balance is $\exp(-0.6 + 0.3\cdot\texttt{bal})$. At $\texttt{bal} = 0$ this is $\exp(-0.6) \approx 0.55$, but at $\texttt{bal} = 5$ it is $\exp(-0.6 + 1.5) = \exp(0.9) \approx 2.46 > 1$. So at high balance, being a student *increases* odds of default. The sign of the student effect depends on balance. (ISLP §4.3.5 makes exactly this point on the same data.)

i) Random forests and OOB *(3 %)*

Solution

(i) *(1 %)* **True**. With $V(\bar{T}) = \rho\sigma^2 + (1-\rho)\sigma^2/B$, smaller `mtry` forces splits to consider fewer features and decorrelates the trees (smaller ρ), so averaging reduces variance more.

(ii) *(1 %)* **False**. B for a random forest is *not* a CV-tuned hyperparameter — averaging only reduces variance, so test error is monotonically non-increasing in B. Pick "enough" trees (typically 500–1000). The contrast with boosting is the key point: in boosting, M *is* a real tuning parameter because boosting can overfit.

(iii) *(1 %)* Each observation is OOB for a given tree with probability $(1-1/n)^n \to 1/e \approx 0.368$. So $1000 \times 0.368 \approx \boxed{368}$ observations are OOB for any given tree.

Problem 3 (16 %) — Theory, hand calculations, pseudocode

a) The mathy one — LDA decision boundary derivation *(8 %)*

Solution *(2 %)* **(i) Deriving $\delta_k(x)$.** By Bayes' rule $\Pr(K = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_\ell \pi_\ell f_\ell(x)}$. The denominator does not depend on k, so $\arg\max_k \Pr(K = k \mid x) = \arg\max_k \pi_k f_k(x) = \arg\max_k \log(\pi_k f_k(x))$.

Substituting the multivariate-normal density,

$$\begin{aligned}\log(\pi_k f_k(x)) &= \log\pi_k - \tfrac{p}{2}\log(2\pi) - \tfrac{1}{2}\log|\boldsymbol{\Sigma}| - \tfrac{1}{2}(x-\boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}_k)\\ &= \log\pi_k - \tfrac{p}{2}\log(2\pi) - \tfrac{1}{2}\log|\boldsymbol{\Sigma}| - \tfrac{1}{2}x^\top\boldsymbol{\Sigma}^{-1}x + x^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \tfrac{1}{2}\boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k.\end{aligned}$$

Terms discarded. The constants $-\frac{p}{2}\log(2\pi)$ and $-\frac{1}{2}\log|\boldsymbol{\Sigma}|$ do not depend on k. *Crucially*, the quadratic-in-x term $-\frac{1}{2}x^\top\boldsymbol{\Sigma}^{-1}x$ also does not depend on k because $\boldsymbol{\Sigma}$ is shared. So when we compare $\delta_A(x)$ to $\delta_B(x)$, those three terms cancel and what remains is

$$\delta_k(x) = x^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k - \tfrac{1}{2}\boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k + \log\pi_k. \qquad \square$$

Solution *(3 %)* **(ii) Explicit boundary, equal priors.** With $\boldsymbol{\Sigma} = \mathbf{I}_2$, $\boldsymbol{\Sigma}^{-1} = \mathbf{I}_2$, so $x^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k = x^\top\boldsymbol{\mu}_k$ and $\boldsymbol{\mu}_k^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_k = \|\boldsymbol{\mu}_k\|^2$.

For class A: $\boldsymbol{\mu}_A = (0,1)^\top$, $\|\boldsymbol{\mu}_A\|^2 = 1$,

$$\delta_A(x) = 0\cdot x_1 + 1\cdot x_2 - \tfrac{1}{2}\cdot 1 + \log(0.5) = x_2 - 0.5 + \log(0.5).$$

For class B: $\boldsymbol{\mu}_B = (2,3)^\top$, $\|\boldsymbol{\mu}_B\|^2 = 4+9 = 13$,

$$\delta_B(x) = 2x_1 + 3x_2 - \tfrac{1}{2}\cdot 13 + \log(0.5) = 2x_1 + 3x_2 - 6.5 + \log(0.5).$$

Set $\delta_A = \delta_B$ (the $\log(0.5)$ cancels):

$$\begin{aligned} x_2 - 0.5 &= 2x_1 + 3x_2 - 6.5 \\ 0 &= 2x_1 + 2x_2 - 6 \\ \boxed{x_1 + x_2 &= 3.} \end{aligned}$$

Sanity check: the boundary is the perpendicular bisector of the segment from $\boldsymbol{\mu}_A$ to $\boldsymbol{\mu}_B$, which passes through the midpoint $(1,2)$ and is orthogonal to $\boldsymbol{\mu}_B - \boldsymbol{\mu}_A = (2,2)$. The point $(1,2)$ satisfies $1+2=3$. ✓

Solution *(1 %)* **(iii) Where did the quadratic go?** The quadratic-in-x term in $\log(\pi_k f_k(x))$ is $-\frac{1}{2}x^\top \boldsymbol{\Sigma}_k^{-1} x$. Under LDA $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}$ is shared across classes, so this term is class-independent and cancels in $\delta_A - \delta_B$. Relaxing to $\boldsymbol{\Sigma}_A \neq \boldsymbol{\Sigma}_B$ (QDA), the term becomes $-\frac{1}{2}x^\top(\boldsymbol{\Sigma}_A^{-1} - \boldsymbol{\Sigma}_B^{-1})x$ in the difference, which is generically nonzero quadratic in x — hence the QDA boundary is quadratic.

Solution *(2 %)* **(iv) Unequal priors.** Now $\log \pi_A = \log(0.8)$, $\log \pi_B = \log(0.2)$. Setting $\delta_A = \delta_B$:

$$\begin{aligned} x_2 - 0.5 + \log(0.8) &= 2x_1 + 3x_2 - 6.5 + \log(0.2) \\ 0 &= 2x_1 + 2x_2 - 6 + \log(0.2) - \log(0.8) \\ 2x_1 + 2x_2 &= 6 + \log(0.8) - \log(0.2) = 6 + \log(0.8/0.2) = 6 + \log 4 \\ &\approx 6 + 1.386 = 7.386. \end{aligned}$$

Hence

$$\boxed{x_1 + x_2 \approx 3.693.}$$

Plain English. The boundary shifted by ≈ 0.693 units (in the $(1,1)/\sqrt{2}$ direction) *away from $\boldsymbol{\mu}_A$ and toward $\boldsymbol{\mu}_B$*. Because class A is now four times more prevalent, the Bayes-optimal classifier assigns more of $\mathbb{R}^2$ to class A: it requires x to be closer to $\boldsymbol{\mu}_B$ before predicting B. Class A now occupies the larger region.
Grading: 2 P for the derivation of δ_k (must explicitly call out which terms are dropped and why $-\frac{1}{2}x^\top\Sigma^{-1}x$ is one of them). 3 P for the explicit boundary in (ii) — 1 P for plugging $\Sigma = I$, 1 P for setting $\delta_A = \delta_B$, 1 P for simplifying to $x_1 + x_2 = 3$. 1 P for the QDA contrast in (iii). 2 P for (iv): 1 P for the numeric c', 1 P for naming the shift direction and the dominant region.

b) Pseudocode — mini-batch SGD training of a feedforward NN *(5 %)*

Solution *(4 %)* **(i)** Network: p inputs $\to$ M hidden ReLU $\to$ scalar output. Parameters $\boldsymbol{\theta} = (W^{(1)}, b^{(1)}, w^{(2)}, b^{(2)})$ with shapes $W^{(1)} \in \mathbb{R}^{M\times p}$, $b^{(1)} \in \mathbb{R}^M$, $w^{(2)} \in \mathbb{R}^M$, $b^{(2)} \in \mathbb{R}$.

Input: training set $\{(x_i, y_i)\}_{i=1}^N$, mini-batch size m, learning rate η, epochs E.

1. **Initialise $\boldsymbol{\theta}$** randomly (e.g. Glorot / He: each weight in $W^{(1)}, w^{(2)}$ drawn iid from a scaled Gaussian; biases set to 0).
2. **For** $e = 1, \ldots, E$:
   (a) Randomly permute $\{1, \ldots, N\}$ and partition into $\lfloor N/m \rfloor$ mini-batches $\mathcal{B}_1, \ldots, \mathcal{B}_{\lfloor N/m \rfloor}$ of size m (plus optional partial batch).
   (b) **For** each mini-batch $\mathcal{B}$:

i. **Forward pass** (per example $i \in \mathcal{B}$): compute and *store*

$$\begin{aligned} z_i^{(1)} &= W^{(1)} x_i + b^{(1)}, \\ a_i^{(1)} &= \mathrm{ReLU}(z_i^{(1)}), \\ \hat{y}_i &= w^{(2)\top} a_i^{(1)} + b^{(2)}, \\ \ell_i &= \tfrac{1}{2}(y_i - \hat{y}_i)^2. \end{aligned}$$

ii. **Backward pass**: using the stored $(x_i, z_i^{(1)}, a_i^{(1)}, \hat{y}_i)$, compute $\nabla_{\boldsymbol{\theta}} \ell_i$ via back-prop (chain rule).

iii. **Aggregate** mini-batch gradient: $\widehat{\nabla_{\boldsymbol{\theta}} L} = \frac{1}{m} \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell_i$.

iv. **Update**: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \cdot \widehat{\nabla_{\boldsymbol{\theta}} L}$.

3. **Return** $\boldsymbol{\theta}$.

Unbiasedness. If $\mathcal{B}$ is a uniform random subset of size m from $\{1, \ldots, N\}$, then $\mathbb{E}[\widehat{\nabla_{\boldsymbol{\theta}} L}] = \frac{1}{N} \sum_{i=1}^{N} \nabla_{\boldsymbol{\theta}} \ell_i$, which is exactly the full-batch gradient. So mini-batch SGD takes *noisy but unbiased* steps along the population gradient direction.

Solution *(1 %)* **(ii) Headline NN fact.** In the over-parameterized regime ($p \gg n$, many global minima of the training loss), mini-batch SGD has an *implicit regularization* effect: among all parameter vectors that interpolate the training data, it tends to pick (approximately) the one with smallest L_2 norm, which is empirically what generalizes well. (The prof's "super weird" fact.)

Grading: 4 P for (i): 0.5 P initialisation, 0.5 P epoch loop, 0.5 P random mini-batching, 1 P forward pass with explicit storage of intermediates, 0.5 P backward pass step, 0.5 P mini-batch averaging, 0.5 P SGD update. Deduct 0.5 if the unbiasedness remark is omitted.

c) Bootstrap by hand — the OOB probability *(3 %)*

Solution

(i) *(1 %)* A single draw from $\{1, \ldots, n\}$ misses observation i with probability $1 - 1/n$, and the n draws are independent (with-replacement), so

$$\Pr(i \notin \text{boot sample}) = (1 - 1/n)^n.$$

At $n = 5$: $(4/5)^5 = 4^5/5^5 = 1024/3125 = \boxed{0.328}$ (to three decimals).

(ii) *(1 %)* As $n \to \infty$:

$$(1 - 1/n)^n = \exp\big(n \log(1 - 1/n)\big) \to \exp\big(n \cdot (-1/n)\big) = e^{-1} \approx 0.368,$$

using $\log(1 - 1/n) = -1/n + O(1/n^2)$. Hence $\Pr(i \in \text{boot sample}) \to 1 - 1/e \approx \boxed{0.632}$.

(iii) *(1 %)* Each observation is OOB for roughly $1/e \approx 1/3$ of the B trees in the random forest, so for that observation we have $\approx B/3$ trees that never saw it — we can predict on it using only those trees and treat it as a held-out test point. Averaging over all n observations gives an OOB error estimate that is asymptotically equivalent to a CV error estimate, but with *zero* extra computational cost: it's a free by-product of fitting the forest.

Problem 4 (22 %) — Data analysis: fuel-economy regression

a) OLS with B-spline + categorical *(8 %)*

Solution *(1 %)* **(i) Parameter count.** Intercept (1) + B-spline basis on `weight` (4) + `displacement` (1) + `horsepower` (1) + `year` (1) + two `origin` dummies (2) = $\boxed{10}$. Residual d.f. $= 292 - 10 = 282$, matches the printout. ✓

Solution *(2 %)* **(ii) Reference level and the intercept.** (a) **Reference**: the omitted level, `origin_American`. (b) The intercept $\hat{\beta}_0 = 29.40$ estimates the mean predicted `mpg` for an American car (`origin_European` = `origin_Japanese` = 0) with all standardized continuous predictors at 0 and all B-spline basis values at 0. (c) With three dummies plus an intercept, the $1+3 = 4$ columns would sum to the all-ones column for the `origin` encoding, so they are linearly dependent — $\mathbf{X}^\top\mathbf{X}$ becomes singular and the model is unidentifiable. ("Dummy-variable trap.")

Solution *(2 %)* **(iii) Japanese-car prediction at all-zero standardized predictors.** With `origin_Japanese` = 1, all other dummies = 0, all standardized continuous predictors = 0 and all B-spline basis values = 0:

$$\begin{aligned}\widehat{\texttt{mpg}} &= 29.40 + 0 \cdot (\text{B-spline}) + 0 \cdot \texttt{disp} + 0 \cdot \texttt{hp} + 0 \cdot \texttt{year} + 1.95 \cdot 0 + 2.30 \cdot 1 \\ &= 29.40 + 2.30 = \boxed{31.70} \text{ mpg.}\end{aligned}$$

Solution *(2 %)* **(iv) What the four B-spline coefficients mean.** They are the loadings on the four basis functions $b_1, \ldots, b_4$ that together specify the smooth piecewise-cubic curve $\hat{f}(\texttt{weight}) = \sum_{j=1}^{4} \hat{\beta}_j b_j(\texttt{weight})$ — the shape of that curve, not any car's predicted `mpg`. The claim "`weight` has *no effect* on `mpg`" is $\hat{\beta}_1 = \hat{\beta}_2 = \hat{\beta}_3 = \hat{\beta}_4 = 0$ simultaneously, which is most naturally tested by a *joint* (F or partial-F) test on the four B-spline coefficients rather than four separate t-tests.

Solution *(1 %)* **(v) The displacement p-value.** "$p = 0.732$, drop it" is sloppy because of collinearity: $\text{cor}(\texttt{weight}, \texttt{displacement}) \approx 0.93$, so the displacement column carries almost no information *beyond* the B-spline of `weight` already in the model — the large p-value reflects redundancy, not absence of effect. Marginally (without `weight`), `displacement` would be highly significant.

b) Diagnosing collinearity, and a fix *(4 %)*

Solution *(2 %)* **(i) Two symptoms together.** (1) The `displacement` *row*: small estimate (-0.130) with a relatively large standard error (0.380), giving $t = -0.34$ and $p = 0.732$. (2) The header datum: $\text{cor}(\texttt{weight}, \texttt{displacement}) = 0.93$. Mechanism: $\text{Var}(\hat{\beta}_j) = \sigma^2[(\mathbf{X}^\top\mathbf{X})^{-1}]_{jj}$, and the near-degeneracy in the `weight` / `displacement` subspace produces a tiny eigenvalue of $\mathbf{X}^\top\mathbf{X}$ in that direction, so the corresponding $[(\mathbf{X}^\top\mathbf{X})^{-1}]_{jj}$ is huge — the SE inflates exactly as observed. Either symptom alone could have other causes; together they are diagnostic.

Solution *(1 %)* **(ii)** Training MSE essentially unchanged ($8.13 \to 8.14$), test MSE essentially unchanged ($8.95 \to 8.92$): the collinearity was hurting *interpretation* (which of `weight` and `displacement` carries the effect), not *prediction* ($\hat{y}$ is well-determined in the subspace the data span).

Solution *(1 %)* **(iii)** Prefer PCR over ridge when the goal is *compression* — you genuinely believe the predictors share a low-rank latent structure (here, "car size" carried by PC1) and want to model in that lower-dimensional space directly. Prefer ridge when the goal is *shrinkage* on the original predictors — you want to keep their unit interpretation but trade some bias for variance reduction along the collinear directions.

c) PCR with cross-validation *(6 %)*

Solution *(2 %)* **(i) One-SE rule for PCR.** CV-min is at $M = 4$, $\text{CV}^\star = 8.51$, $\widehat{\text{SE}}^\star = 0.80$. Bound:

$$8.51 + 0.80 = \boxed{9.31}.$$

Candidates with $\text{CV}(M) \leq 9.31$: $M = 2$ (9.40? — $9.40 > 9.31$ by 0.09, so *excluded*; $M = 3$ (8.55) and $M = 4$ (8.51), both included. (At $M = 2$ the CV-MSE is 9.40, which is just *above* the bound 9.31 by 0.09; this is within rounding of the bound, and many graders would treat $M = 2$ as borderline. The conservative reading is that only $M \in \{3, 4\}$ qualifies.) Among the qualifying M values the simplest (smallest) is $\boxed{M = 3}$.
Grading: 1 P for the bound, 1 P for $M = 3$. Accept $M = 2$ if the student notes that 9.40 is within rounding of 9.31 and treats it as borderline.

Solution *(2 %)* **(ii) Interpretation of ϕ_1.** The first PC loads roughly equally and positively on `weight, displacement, horsepower` (all $\approx +0.55$), and weakly negatively on `year` (-0.20). Plain English: PC1 is essentially a "**car-size / engine-power** axis" — big cars are heavy, large-engined, and powerful (the three positively correlated variables, $\text{cor}(\texttt{weight}, \texttt{disp}) = 0.93$, $\text{cor}(\texttt{weight}, \texttt{hp}) = 0.86$), and they tend to be slightly older models (the small negative loading on `year`). The near-equal $+0.55$ on the three size variables is exactly what one expects when three predictors are nearly collinear — they all move together along the same dominant direction of variation.

Solution *(2 %)* **(iii) Training-MSE-selected PCR.** (a) Adding components only adds flexibility (each z_M is a new regressor), so training MSE is monotonically non-increasing in M and is minimised at the maximum $M = 4$. (b) Picking $M = 4$ here would discard the whole point of PCR: $M = 4$ regresses on a full-rank rotation of the predictors and is essentially equivalent to OLS on the original four predictors — with the same collinearity problem that motivated PCR in the first place. CV would have flagged this; training MSE never will.

d) GAM with smoothing splines *(4 %)*

Solution

(i) *(1 %)* (a) **Smaller edf ⇔ larger λ**: more penalty smooths f_j harder, shrinking it toward a linear fit (edf $\to 1$) or further toward a constant. (b) $\widehat{\text{edf}}(f_3) = 1.0$ implies the LOOCV-optimal smoothing on `year` reduces f_3 to a *linear* function (a straight-line trend in `year`); the smoothing spline collapses to its single linear basis function.

(ii) *(1 %)* OLS 8.95 > ridge 8.60 > GAM 7.95 > boosted trees 6.10. The clear ordering says there is meaningful *nonlinearity* in the predictor–response relationship (GAM beats OLS), and meaningful *interaction structure* beyond pure additivity (boosted trees beat GAM by a substantial $\sim 23\%$).

(iii) *(1 %)* "Always use boosting" ignores interpretability. The GAM gives a fitted univariate effect f_j per predictor that you can plot and explain to stakeholders; the boosted ensemble of ~ 2000 trees is a black box (you need partial-dependence / permutation importance to claw back any insight). For prediction-only deployments the boosting wins; for inference / communication the GAM is preferable.

(iv) *(1 %)* Boosting is *sequential* — each tree fits the residuals of the current ensemble — so with far too many trees the ensemble starts fitting noise and test MSE *rises*. The $M^\star = 2000$ was chosen by CV at the bias–variance optimum; going to $M = 20{,}000$ overshoots into overfit. In a random forest, by contrast, averaging only reduces variance so test MSE is monotonically non-increasing in B.

Problem 5 (24 %) — Data analysis: bank loan default classification

a) Logistic regression, odds and a categorical contrast *(7 %)*

Solution *(2 %)* **(i) Odds factor for +1 standardized FICO.** $\exp(\hat{\beta}_{\texttt{fico}}) = \exp(-1.20) \approx \boxed{0.301}$. The sign matches the economic prior: higher FICO score (better credit history) is associated with *lower* odds of default — each one-SD increase in FICO multiplies the odds by ≈ 0.30, i.e. cuts them by roughly 70%.

Solution *(2 %)* **(ii) Linear predictor and predicted probability.** Renter (reference) so both home-ownership dummies are 0:

$$\begin{aligned}\hat{\eta} &= -1.80 + (-1.20)\cdot(-1) + 0.85\cdot(+1) + 0.40\cdot 0 + (-0.20)\cdot 0 + 0 + 0 \\ &= -1.80 + 1.20 + 0.85 = \boxed{0.25}.\end{aligned}$$

Sigmoid step:

$$\hat{p} = \sigma(0.25) = \frac{1}{1+e^{-0.25}} = \frac{1}{1+0.7788} = \frac{1}{1.7788} \approx \boxed{0.562}.$$

Solution *(2 %)* **(iii) Owner vs. renter.** On the *odds* scale the contrast is $\exp(\hat{\beta}_{\texttt{own}} - 0) = \exp(-0.90) \approx \boxed{0.41}$: the owner's odds are $0.41\times$ the renter's. On the *probability* scale this does *not* reduce to "$\hat{p}_{\text{own}} = 0.41\hat{p}_{\text{rent}}$" in general. If $\hat{p}$ is the renter's predicted probability, then

$$\hat{p}_{\text{own}} = \frac{0.41\cdot\hat{p}/(1-\hat{p})}{1+0.41\cdot\hat{p}/(1-\hat{p})} = \frac{0.41\,\hat{p}}{(1-\hat{p})+0.41\,\hat{p}} = \frac{0.41\,\hat{p}}{1-0.59\,\hat{p}},$$

which only equals $0.41\,\hat{p}$ when $\hat{p}$ is very small (so $1-0.59\hat{p}\approx 1$). At $\hat{p}=0.562$ (part (ii)), the owner's probability is $0.41\cdot 0.562/(1-0.59\cdot 0.562) \approx 0.230/0.668 \approx 0.345$, not $0.41\cdot 0.562 \approx 0.230$. *Grading: 1 P for the odds factor* 0.41*, 1 P for noting that the probability does not scale by* 0.41 *(the standard "odds-multiplier vs. probability-multiplier" trap).*

Solution *(1 %)* **(iv) Causal claim, rebutted.** A logistic regression on observational loan-application data is a model of *conditional association*, not causation — the prof's recurring "fancy correlations, not causal" point. People who own their home outright are systematically different from renters in many unobserved ways (wealth, income stability, age, geography), so any of those confounders could explain the lower default odds. Inferring *causation* would require a randomised intervention on home-ownership or much richer confounder adjustment.

b) LDA vs. QDA — explicit discriminant computation *(6 %)*

Solution *(2 %)* **(i) LDA classification.** With $\hat{\boldsymbol{\Sigma}} = \mathbf{I}_2$, $\hat{\boldsymbol{\Sigma}}^{-1} = \mathbf{I}_2$, so $x^\top\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_k = x^\top\hat{\boldsymbol{\mu}}_k$ and $\hat{\boldsymbol{\mu}}_k^\top\hat{\boldsymbol{\Sigma}}^{-1}\hat{\boldsymbol{\mu}}_k = \|\hat{\boldsymbol{\mu}}_k\|^2$. Test point $x_0 = (-0.5,\, 0.8)^\top$.

Class 0: $\hat{\boldsymbol{\mu}}_0 = (0.20,\, -0.30)$, $\|\hat{\boldsymbol{\mu}}_0\|^2 = 0.04 + 0.09 = 0.13$, $\log(0.80) \approx -0.223$.

$$\begin{aligned}\delta_0(x_0) &= (-0.5)(0.20) + (0.8)(-0.30) - \tfrac{1}{2}(0.13) + (-0.223) \\ &= -0.100 - 0.240 - 0.065 - 0.223 \\ &= \boxed{-0.628}.\end{aligned}$$

Class 1: $\hat{\boldsymbol{\mu}}_1 = (-0.80,\, 1.20)$, $\|\hat{\boldsymbol{\mu}}_1\|^2 = 0.64 + 1.44 = 2.08$, $\log(0.20) \approx -1.609$.

$$\begin{aligned}\delta_1(x_0) &= (-0.5)(-0.80) + (0.8)(1.20) - \tfrac{1}{2}(2.08) + (-1.609) \\ &= 0.400 + 0.960 - 1.040 - 1.609 \\ &= \boxed{-1.289}.\end{aligned}$$

$\delta_0 = -0.628 > \delta_1 = -1.289$, so LDA predicts $\boxed{\text{class 0 (no default)}}$.

Solution *(2 %)* **(ii) QDA classification.** $\hat{\boldsymbol{\Sigma}}_0 = \mathbf{I}_2$ so $|\hat{\boldsymbol{\Sigma}}_0| = 1$, $\log|\hat{\boldsymbol{\Sigma}}_0| = 0$, $\hat{\boldsymbol{\Sigma}}_0^{-1} = \mathbf{I}_2$. $\hat{\boldsymbol{\Sigma}}_1 = \mathrm{diag}(1.5, 0.5)$ so $|\hat{\boldsymbol{\Sigma}}_1| = 0.75$, $\log|\hat{\boldsymbol{\Sigma}}_1| = \log(1.5) + \log(0.5) \approx 0.405 - 0.693 = -0.288$, $\hat{\boldsymbol{\Sigma}}_1^{-1} = \mathrm{diag}(1/1.5, 1/0.5) = \mathrm{diag}(2/3,\ 2)$.

Class 0. $x_0 - \hat{\boldsymbol{\mu}}_0 = (-0.5 - 0.20,\ 0.8 - (-0.30)) = (-0.70,\ 1.10)$:

$$\begin{aligned}\delta_0^{\mathrm{QDA}}(x_0) &= -\tfrac{1}{2}(0) - \tfrac{1}{2}\big((-0.70)^2 + (1.10)^2\big) + \log(0.80)\\ &= 0 - \tfrac{1}{2}(0.49 + 1.21) - 0.223\\ &= -\tfrac{1}{2}(1.70) - 0.223 = -0.850 - 0.223 = \boxed{-1.073}.\end{aligned}$$

Class 1. $x_0 - \hat{\boldsymbol{\mu}}_1 = (-0.5 - (-0.80),\ 0.8 - 1.20) = (0.30,\ -0.40)$:

$$\begin{aligned}\delta_1^{\mathrm{QDA}}(x_0) &= -\tfrac{1}{2}(-0.288) - \tfrac{1}{2}\left[\frac{(0.30)^2}{1.5} + \frac{(-0.40)^2}{0.5}\right] + \log(0.20)\\ &= 0.144 - \tfrac{1}{2}\left[\frac{0.09}{1.5} + \frac{0.16}{0.5}\right] - 1.609\\ &= 0.144 - \tfrac{1}{2}(0.06 + 0.32) - 1.609\\ &= 0.144 - 0.190 - 1.609 = \boxed{-1.655}.\end{aligned}$$

$\delta_0^{\mathrm{QDA}} = -1.073 > \delta_1^{\mathrm{QDA}} = -1.655$, so QDA also predicts $\boxed{\text{class 0 (no default)}}$.

Solution *(1 %)* **(iii) Linear vs. conic.** Under LDA the shared $\hat{\boldsymbol{\Sigma}}$ makes the term $-\frac{1}{2}x^\top\hat{\boldsymbol{\Sigma}}^{-1}x$ class-independent, so it cancels in $\delta_0 - \delta_1$ and the boundary is linear in x. Under QDA each class has its own $\hat{\boldsymbol{\Sigma}}_k$, so the $-\frac{1}{2}x^\top\hat{\boldsymbol{\Sigma}}_k^{-1}x$ term differs between classes and $\delta_0 - \delta_1$ retains a quadratic form $x^\top(\hat{\boldsymbol{\Sigma}}_1^{-1} - \hat{\boldsymbol{\Sigma}}_0^{-1})x/2$ — a conic section.

Solution *(1 %)* **(iv) Effect of changing $\hat{\pi}_0$.** The priors enter δ_k only through $\log\hat{\pi}_k$. Going from $\hat{\pi}_0 = 0.80$ to $\hat{\pi}_0 = 0.50$ changes $\log(\hat{\pi}_0/\hat{\pi}_1)$ from $\log 4 \approx 1.386$ to $\log 1 = 0$, removing the ≈ 1.386 "boost" that class 0 previously enjoyed. The boundary shifts toward $\hat{\boldsymbol{\mu}}_0$ (class 0's region shrinks, class 1's region grows) by ≈ 1.386 in the relevant scalar projection.

c) XGBoost: the regularization knobs *(8 %)*

Solution *(2 %)* **(i) Which two derivatives.** XGBoost uses both the *first* derivative (gradient $g_i = \partial L/\partial\hat{y}_i$) and the *second* derivative (Hessian $h_i = \partial^2 L/\partial\hat{y}_i^2$) of the loss with respect to the current ensemble's predictions; vanilla GBM uses only the gradient. Geometrically, the gradient gives a *linear* (first-order) approximation to the loss — equivalent to a gradient-descent step in function space — whereas adding the Hessian gives a *quadratic* (second-order, Newton) approximation. The Newton step automatically scales the per-tree update by local curvature, so it takes more accurate steps near the optimum and is less reliant on a tiny learning rate.

Solution *(3 %)* **(ii) Three XGBoost regularizers.**

- $\boldsymbol{\lambda}$ (L_2 on leaf weights): exactly analogous to ridge regression on linear regression coefficients — shrinks every leaf output smoothly toward zero, reducing variance of the per-tree fit.
- $\boldsymbol{\alpha}$ (L_1 on leaf weights): analogous to lasso — can drive small leaf outputs *exactly* to zero, neutralising leaves whose contribution doesn't survive the L_1-corner geometry.
- $\boldsymbol{\gamma|T|}$ (per-leaf complexity penalty): analogous to cost-complexity pruning — adding a leaf must reduce the loss by more than γ to be accepted, so γ controls how aggressively splits are kept or pruned.

Solution *(2 %)* **(iii) Doubling M and η.**

- (a) The "preserve bias–variance balance" claim is wrong. In boosting M and η are *coupled*: smaller η at fixed M adds less per tree, smaller total cumulative effect; the folklore is that the cumulative ensemble strength is approximately $\eta \cdot M$. Doubling *both* more than doubles the effective ensemble's cumulative correction, dramatically increasing variance and risk of overfit, not "preserving the balance."
- (b) The diagnostic is a **CV-error curve as a function of** M (with η fixed at 0.05): plot CV-MSE / CV-AUC versus M, identify the validation minimum (or the elbow); pick M there. The early-stopping signal — "test error has started to rise" — is the standard way to choose M in boosting.

Solution *(1 %)* **(iv) True**. The L_1 corner geometry of $\alpha\|w\|_1$ on the leaf-weight vector w can drive individual leaf weights to exactly zero (a leaf whose optimal w_j would have small magnitude in the absence of α gets clipped to 0), which is on top of the $\gamma|T|$ leaf-count penalty (the latter penalises the *number* of leaves; α shrinks the *values* on those leaves and can sparsify them exactly).

d) Sensitivity, specificity, ROC, and class imbalance *(3 %)*

Solution *(1 %)* **(i) Trivial baseline.** "Always predict no-default" is correct on all 1200 non-defaulters and wrong on all 300 defaulters:

$$\text{accuracy}_{\text{trivial}} = \frac{1200}{1500} = \boxed{0.80}.$$

The three classifier rows show accuracies of 0.84, 0.85, 0.85 — only 0.04–0.05 above the trivial baseline. Accuracy compresses sensitivity and specificity into a single scalar that is dominated by the majority class, so it cannot distinguish a model that catches more defaulters (XGBoost, sens 0.55) from one that catches almost none but is slightly more careful with the non-defaulters (LDA, sens 0.36). Use sensitivity / specificity (or ROC-AUC) instead.

Solution *(2 %)* **(ii) Deployment recommendation.** Among the three classifiers at the default threshold $\hat{p} = 0.5$, only candidates with specificity ≥ 0.90 are admissible: all three pass. Sensitivities are 0.40 (logistic) $<$ 0.55 (XGBoost), with LDA's 0.36 lowest. The clear winner under the stated objective is **XGBoost**: highest sensitivity (0.55) while keeping specificity $0.93 \geq 0.90$, plus highest AUC (0.88 vs. $0.83, 0.81$) for further threshold flexibility.

Threshold adjustment. Lowering the XGBoost threshold below $\hat{p} = 0.5$ classifies more applicants as positive (defaulter), which raises the true-positive rate (*sensitivity goes up*) at the cost of more false positives (*specificity goes down*). This is exactly the trade-off encoded by the ROC curve. The team would move the threshold downward until specificity hits its 0.90 floor, harvesting the maximum sensitivity along the way.

End of solution proposal. Total awarded: $10 + 28 + 16 + 22 + 24 = 100$ points.