TMA4268 V2026 Mock Exam 8 — Solution Proposal

Compiled for Anders Bekkevard

Companion to `mock-exam-8.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-8.tex` for the problem statements.

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

Solution *(1 P per blank.)*

(1) **generative**

(2) **linear**

(3) **pooled**

(4) **collinearity**

(5) **ridge regression**

(6) **random forests**

(7) **out-of-bag error**

(8) **backpropagation**

(9) **label smoothing**

(10) **bootstrap**

Grading: 1 P per correct blank, no partial credit. The standard traps are: (2) "quadratic" (that's QDA, the very thing LDA is contrasted with); (3) "diagonal" (that's naive Bayes, not standard LDA); (5) "the lasso" (lasso uses L_1, not L_2, and does *zero coefficients); (6) "gradient boosting" (random forests is the one defined by predictor sub-sampling at each split); (8) "the EM algorithm" (EM is for latent-variable models, not for chain-rule gradient computation); (9) "dropout" (dropout zeros activations, not targets). Note: distractor order has been rotated across items so that the correct answer's position varies across (1)–(10); read each list of choices carefully.*

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

a) Bias–variance, applied to a shrinkage estimator *(3 %)*

Solution

(i) *(1 %)* Since $\mathbb{E}[\hat{\mu}_c] = c\mathbb{E}[\bar{X}] = c\mu$ and $\mathrm{Var}(\hat{\mu}_c) = c^2\mathrm{Var}(\bar{X}) = c^2\sigma^2/n$:

$$\boxed{\mathrm{Bias}^2(\hat{\mu}_c) = (c\mu - \mu)^2 = (1-c)^2\mu^2, \qquad \mathrm{Var}(\hat{\mu}_c) = \frac{c^2\sigma^2}{n}.}$$

(ii) *(1 %)* **False**. Whether $c < 1$ helps depends on the signal-to-noise ratio: if μ^2 is huge relative to σ^2/n, the squared-bias cost $(1-c)^2\mu^2$ swamps the variance saving $(1-c^2)\sigma^2/n$, and $c = 1$ (OLS / unbiased) wins. The MSE-optimal $c^\star$ derived in P3(b) is strictly < 1, but only when noise is non-negligible.

(iii) *(1 %)* **False**. The decomposition $\mathbb{E}[(y_0 - \hat{f}(x_0))^2] = \mathrm{Bias}^2 + \mathrm{Var} + \sigma^2$ is an *algebraic identity* (P3-style derivation in last year's solution). It holds in every regime, including over-parameterized / double-descent. Double descent is the observation that $\mathrm{Var}(p)$ has a non-monotone shape, not that the formula fails.

Grading: 1 P each. (i) full credit requires both formulas; deduct 0.5 if bias is written un-squared. (ii) the common trap is to say "True — shrinkage helps," which conflates the qualitative bias–variance message with the claim that shrinkage dominates the unbiased estimator regardless of μ and σ^2/n. (iii) the trap is to confuse the shape *of variance-vs-flexibility with the validity of the identity.*

b) Cross-validation: LOOCV vs k-fold and the OLS shortcut *(4 %)*

Solution

(i) *(1 %)* **True**. The bias of LOOCV as an estimator of test error trained on n is smallest (each training set has $n-1$ points), but the n leave-one-out training sets differ by exactly one observation, the per-fold errors are highly positively correlated, and the variance of their average inflates. 5- or 10-fold CV strikes the usual bias–variance balance.

(ii) *(1 %)* The closed-form LOOCV formula for OLS:

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

Each term divides the full-data residual by $1 - h_{ii}$, the "leverage adjustment" that converts a fitted residual into the leave-one-out residual without refitting n times.

(iii) *(1 %)* **False**. Random partitioning destroys temporal structure: the training set will contain points immediately adjacent to the test set's hold-out, leaking next-day-style information and biasing CV-MSE *downward*. The correct approach for time series is forward-chaining CV (train on $[1, \ldots, t]$, test on $[t+1, \ldots, t+h]$) so that no test fold is preceded by a training point that is its own future.

(iv) *(1 %)* The k per-fold MSEs are *not* independent — the k training sets share most of their observations — so the sample standard deviation of the k values is not a calibrated standard error of the mean. The slide's wording ("strictly speaking, not quite valid") is the prof's gentle warning; the quantity is used anyway because nothing better is cheap, and the one-SE rule is robust to the approximation.

Grading: 1 P each. For (ii) full credit requires the $(1-h_{ii})^2$ in the denominator; the "shortcut without the leverage" yields the fitted residual MSE, not the LOOCV MSE. For (iv) accept any phrasing that names the dependence between folds.

c) Bootstrap: percentile CI, basic CI, bias correction *(3 %)*

Solution

(i) *(1 %)* The percentile 95% bootstrap CI is

$$\boxed{\left[q^*_{0.025},\ q^*_{0.975}\right]},$$

i.e. the empirical 2.5% and 97.5% quantiles of $\{\hat{\theta}^*_1, \ldots, \hat{\theta}^*_B\}$.

(ii) *(1 %)* (A) **False**. Bootstrap resamples are drawn *with* replacement; the same row appears multiple times in a typical resample (and about 37% of rows do not appear at all, the OOB rate). (B) **True**. The bootstrap distribution is centred on $\hat{\theta}$, not on θ, so it quantifies variability but cannot itself correct bias (one needs a bias-corrected bootstrap or another mechanism). (C) **True**. Course recommendation: $B \approx 1{,}000$ for SE, $B \in [1000, 10000]$ for stable percentile-CI quantiles.

(iii) *(1 %)* The two MSEs $\text{MSE}^{\text{test}}_A$ and $\text{MSE}^{\text{test}}_B$ are computed on the *same* test pairs (x_t, y_t), so their per-observation squared errors are highly positively correlated (an unusually hard pair inflates both losses). The paired bootstrap preserves that correlation, dramatically shrinking $\text{Var}(\widehat{\Delta}) = \text{Var}(A) + \text{Var}(B) - 2\text{Cov}(A, B)$; resampling independently destroys $\text{Cov}(A, B)$ and inflates the SE of the difference.

Grading: 1 P each. (ii) all three sub-statements must be right for the point.

d) Mini-batch SGD, activations, vanishing gradients *(3 %)*

Solution *(1 P per statement.)*

(i) **True**. The mini-batch gradient $\widehat{\nabla L} = \frac{1}{m}\sum_{i\in\mathcal{B}} \nabla L_i$ has expectation $\frac{1}{N}\sum_i \nabla L_i$ (the full-batch gradient) when $\mathcal{B}$ is a uniform random subset of size m. Variance scales like $1/m$, so smaller batches are noisier — a trade-off the prof flagged as the source of SGD's implicit-regularization effect.

(ii) **False**. The course-stated practical range is $\eta \in [10^{-3}, 10^{-1}]$ with 0.01–0.1 typical, *not* $\eta \approx 2$. A learning rate of 2 would overshoot every basin and oscillate without converging, while $\eta = 0.1$ sits at the high end of the usable range and is a perfectly reasonable practical value.

(iii) **True**. For large positive z, $\text{ReLU}'(z) = 1$ exactly, while $\sigma'(z) = \sigma(z)(1-\sigma(z)) \to 0$ as $\sigma(z) \to 1$. The standard "vanishing gradient" story: in deep networks the chain rule multiplies many activation derivatives; if each is $\ll 1$, the product collapses, and the lower layers stop receiving useful gradient signal. ReLU keeps the derivative at 1 on its active half-line and is the textbook fix.

Grading: 1 P each. The (ii) trap is to think "larger = faster" and accept $\eta \approx 2$ as typical — the prof explicitly cited divergent-loss screenshots at large η.

e) Boosting: AdaBoost, gradient boosting, XGBoost *(4 %)*

Solution

(i) *(1 %)* With $\text{err}_{m^*} = 0.70$:

$$\alpha_{m^*} = \log\left(\frac{1-0.70}{0.70}\right) = \log\left(\frac{0.30}{0.70}\right) = \log(0.4286) \approx \boxed{-0.85}.$$

The negative sign *flips* the classifier's vote in the final sign-sum: a worse-than-random G_{m^*} is used by AdaBoost *inverted*, which is constructive (if its error is consistently above 0.5 it carries information about the right answer with the wrong label).

(ii) *(1 %)* **True**. With $L(y, F) = \frac{1}{2}(y-F)^2$, the negative gradient w.r.t. F is $-\partial L/\partial F = y - F$, exactly the current residual. So squared-error gradient boosting and "forward stagewise regression on residuals" are the same algorithm.

(iii) *(1 %)* **False**. Each tree's effective contribution to the ensemble is $\nu \cdot \hat{T}_m$, so halving ν roughly *doubles* (not halves) the trees needed to reach the same cumulative fit. The statement gets the direction backwards. This is the well-known (M, ν) joint trade-off; the rule of thumb is "smaller ν is better if you can afford the trees."

(iv) *(1 %)* **True**. XGBoost's two structural additions over vanilla gradient boosting are (a) explicit L_1 and L_2 regularization on the per-leaf weights and (b) a second-order Taylor expansion of the loss, so split scores depend on both the gradient g_i and the Hessian h_i of L at the current prediction — the $\sum_i g_i$ and $\sum_i h_i$ partial sums that XGBoost accumulates.

Grading: 1 P each. (i) full credit requires both the numeric and the sign-flip interpretation. (iii) is exactly the reverse misreading "halving ν halves M" — it gets the qualitative direction backward and is therefore False.

f) Neural network regularization *(3 %)*

Solution

(i) *(1 %)* (A) **True**. Dropout is a training-time stochastic mask; at test time all units are kept and outputs rescaled (or inverted dropout rescales during training so no test-time rescaling is needed). (B) **False**. Early stopping monitors *validation* loss, not training loss — training loss typically decreases monotonically, so it would never stop. (C) **True**. Course default is ≈ 0.20; 0.5 is generally regarded as too aggressive (under-fits).

(ii) *(1 %)* With $\varepsilon = 0.05$ the hard target $(1, 0)$ becomes the soft target $(1-\varepsilon,\, \varepsilon) = \boxed{(0.95,\ 0.05)}$.

(iii) *(1 %)* **False**. Weight decay *constrains* the optimizer to weights with smaller L_2 norm; it therefore *raises* the achievable training loss (a smaller feasible region cannot have a lower minimum). The point of weight decay is to reduce *test* loss via a bias–variance trade.

Grading: 1 P each. For (i) all three sub-statements required. For (ii) accept the multi-class soft target $(1-\varepsilon, \varepsilon)$ or its equivalent $(0.95, 0.05)$.

g) Logistic regression: odds, log-odds, interaction *(3 %)*

Solution

(i) *(1 %)* The linear predictor at $x_0 = 5$ is

$$\hat{\eta} = -2.0 + 0.40 \cdot 5 = -2.0 + 2.0 = 0,$$

so

$$\hat{p} = \sigma(0) = \frac{1}{1+e^0} = \boxed{0.500}.$$

(ii) *(1 %)* For the *untreated* (`treatment` = 0) patient, the interaction contributes 0, so a one-year increase in `age` changes log-odds by $\hat{\beta}_{\texttt{age}} = 0.05$; odds multiplier $\exp(0.05) \approx \boxed{1.051}$. For the *treated* patient (`treatment` = 1), the interaction kicks in: $\Delta \log \text{odds} = 0.05 + (-0.02) = 0.03$; odds multiplier $\exp(0.03) \approx \boxed{1.030}$.

(iii) *(1 %)* **True**. Relabelling the reference of a categorical predictor is a re-parameterization of the same fitted model: the intercept and the coefficient on `treatment` both change to compensate, but $\hat{p}(x)$ for any observation is unchanged. This is the standard "parameter sign flip $\neq$ prediction change" lesson.

Grading: 1 P each. (ii) full credit requires both odds factors and that the treated case uses both $\hat{\beta}_{age}$ and the interaction — forgetting the interaction is the standard trap.

h) Discriminant analysis and naive Bayes *(2 %)*

Solution

(i) *(1 %)* (A) **True**. Equal Gaussian covariances kill the quadratic term in $\log f_k(x)$ that depends on k, leaving a function linear in $\mathbf{x}$. (B) **True**. QDA's class-specific Σ_k means the term $\mathbf{x}^\top \Sigma_k^{-1} \mathbf{x}$ depends on k and survives the difference of discriminants; LDA's shared Σ makes it cancel. (C) **False**. Increasing $\hat{\pi}_k$ raises $\log \hat{\pi}_k$ in δ_k and moves the boundary *away from* class k, classifying *more* area as k. The statement has the direction backwards.

(ii) *(1 %)* **True**. The naive-Bayes assumption "features are conditionally independent given the class" is exactly the assumption that the within-class covariance matrix is diagonal. When the class-conditional densities are Gaussian, diagonal-covariance LDA = Gaussian naive Bayes.

Grading: 1 P each. (i)C is the most common trap — careful with the geometric direction.

i) Principal component analysis *(3 %)*

Solution

(i) *(1 %)* For standardised variables $\sum_j \lambda_j = p = 5$ (check: $2.1 + 1.4 + 0.8 + 0.5 + 0.2 = 5.0$). Cumulative PVE:

$$\begin{aligned} \text{PC1}: &\quad 2.1/5 = 0.420, \\ \text{PC1+PC2}: &\quad 0.420 + 1.4/5 = 0.420 + 0.280 = 0.700, \\ \text{PC1+PC2+PC3}: &\quad 0.700 + 0.8/5 = 0.700 + 0.160 = 0.860. \end{aligned}$$

We first cross 0.80 at PC3, so $\boxed{3}$ components are needed.

(ii) *(1 %)* $z_1^* = \phi_1^\top x^*$:

$$\begin{aligned} z_1^* &= 0.55 \cdot 2 + 0.45 \cdot 1 + (-0.35) \cdot 0 + 0.40 \cdot (-1) + (-0.45) \cdot 1 \\ &= 1.10 + 0.45 + 0 - 0.40 - 0.45 \\ &= \boxed{0.70}. \end{aligned}$$

(iii) *(1 %)* **False**. PCA *is* scale-dependent: without standardisation, a predictor measured in large units (large variance on its native scale) will dominate the leading component simply because of unit choice. The classic prof example: a dataset with height in cm and weight in kg yields a wildly different PC1 than the same data with height in m and weight in g. Standardisation is what makes PCA scale-invariant.

Grading: 1 P each. Accept ± 0.005 *rounding in (i)–(ii).*

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

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

Solution *(3 %)* **(i) Derivation of $\delta_k(\mathbf{x})$.** Starting from the Gaussian class-conditional density:

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\Big(-\tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big).$$

Take logs of $\pi_k f_k(\mathbf{x})$:

$$\log\big(\pi_k f_k(\mathbf{x})\big) = \log \pi_k - \tfrac{p}{2}\log(2\pi) - \tfrac{1}{2}\log|\Sigma| - \tfrac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_k).$$

Expand the quadratic form:

$$(\mathbf{x}-\boldsymbol{\mu}_k)^\top \Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) = \mathbf{x}^\top\Sigma^{-1}\mathbf{x} - 2\mathbf{x}^\top\Sigma^{-1}\boldsymbol{\mu}_k + \boldsymbol{\mu}_k^\top\Sigma^{-1}\boldsymbol{\mu}_k,$$

using Σ^{-1} symmetric (so the two cross-terms collapse to one -2).

Discardable terms. The classifier's decision compares $\log\big(\pi_k f_k(\mathbf{x})\big)$ across k. Any term that does *not* depend on k contributes the same constant to every class and therefore drops out of the arg max:

- $-\frac{p}{2}\log(2\pi)$ has no k-dependence: *drop.*
- $-\frac{1}{2}\log|\Sigma|$ has no k-dependence (*because* Σ *is shared*): *drop.*
- $-\frac{1}{2}\mathbf{x}^\top\Sigma^{-1}\mathbf{x}$: the term depends on $\mathbf{x}$ but *not on* k (again because Σ does not depend on k): *drop.*

What is left, after multiplying through by the sign and absorbing the $-\frac{1}{2}\cdot(-2) = 1$:

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

Which term would have survived under QDA. If we allowed Σ_k to depend on k, the $-\frac{1}{2}\log|\Sigma_k|$ term and the $-\frac{1}{2}\mathbf{x}^\top\Sigma_k^{-1}\mathbf{x}$ term would *both* depend on k, so neither would cancel. The discriminant would then be *quadratic* in $\mathbf{x}$ (the $\mathbf{x}^\top\Sigma_k^{-1}\mathbf{x}$ piece), giving the QDA boundary — a conic section rather than a hyperplane.

Solution *(3 %)* **(ii)** Plug in $\Sigma = 2I$, $\Sigma^{-1} = \frac{1}{2}I$, $\boldsymbol{\mu}_0 = (2,1)^\top$, $\boldsymbol{\mu}_1 = (0,3)^\top$, $\pi_0 = \pi_1 = 0.5$. The priors will cancel ($\log\pi_0 = \log\pi_1$), so the boundary is $\delta_0(\mathbf{x}) = \delta_1(\mathbf{x})$:

$$\mathbf{x}^\top\Sigma^{-1}\boldsymbol{\mu}_0 - \tfrac{1}{2}\boldsymbol{\mu}_0^\top\Sigma^{-1}\boldsymbol{\mu}_0 = \mathbf{x}^\top\Sigma^{-1}\boldsymbol{\mu}_1 - \tfrac{1}{2}\boldsymbol{\mu}_1^\top\Sigma^{-1}\boldsymbol{\mu}_1.$$

Linear-in-x part.

$$\Sigma^{-1}\boldsymbol{\mu}_0 = \tfrac{1}{2}\begin{pmatrix}2\\1\end{pmatrix} = \begin{pmatrix}1\\0.5\end{pmatrix}, \qquad \Sigma^{-1}\boldsymbol{\mu}_1 = \tfrac{1}{2}\begin{pmatrix}0\\3\end{pmatrix} = \begin{pmatrix}0\\1.5\end{pmatrix}.$$

$$\mathbf{x}^\top\Sigma^{-1}\boldsymbol{\mu}_0 = x_1\cdot 1 + x_2\cdot 0.5 = x_1 + 0.5x_2.$$
$$\mathbf{x}^\top\Sigma^{-1}\boldsymbol{\mu}_1 = x_1\cdot 0 + x_2\cdot 1.5 = 1.5x_2.$$

LHS minus RHS, on the linear side:

$$\big(x_1 + 0.5x_2\big) - \big(1.5x_2\big) = x_1 - x_2.$$

Constant pieces.

$$\boldsymbol{\mu}_0^\top \Sigma^{-1} \boldsymbol{\mu}_0 = (2,1) \cdot (1, 0.5)^\top = 2 \cdot 1 + 1 \cdot 0.5 = 2.5.$$

$$\boldsymbol{\mu}_1^\top \Sigma^{-1} \boldsymbol{\mu}_1 = (0,3) \cdot (0, 1.5)^\top = 0 + 4.5 = 4.5.$$

LHS minus RHS, on the constant side (note the $-\frac{1}{2}$):

$$-\tfrac{1}{2} \cdot 2.5 - \left(-\tfrac{1}{2} \cdot 4.5\right) = -1.25 + 2.25 = +1.$$

Put it together. $\delta_0 - \delta_1 = 0$ gives:

$$x_1 - x_2 + 1 = 0,$$

or equivalently

$$\boxed{x_2 = x_1 + 1.}$$

Solution *(1 %)* **(iii)** At $\mathbf{x}^* = (1,1)^\top$:

$$\text{LHS of the boundary equation: } x_1 - x_2 + 1 = 1 - 1 + 1 = +1.$$

A positive value means $\delta_0 > \delta_1$ (the boundary is the level set "$= 0$"; the side with δ_0 dominating is the positive side, since the algebra was LHS$= \delta_0$-side minus RHS$= \delta_1$-side). So $\mathbf{x}^*$ is assigned to **class 0**.

Direct verification.

$$\begin{aligned}\delta_0(\mathbf{x}^*) &= \mathbf{x}^{*\top} \Sigma^{-1} \boldsymbol{\mu}_0 - \tfrac{1}{2} \boldsymbol{\mu}_0^\top \Sigma^{-1} \boldsymbol{\mu}_0 + \log(0.5) \\ &= (1 \cdot 1 + 1 \cdot 0.5) - \tfrac{1}{2}(2.5) + \log(0.5) \\ &= 1.5 - 1.25 + \log(0.5) = 0.25 + \log(0.5).\end{aligned}$$

$$\begin{aligned}\delta_1(\mathbf{x}^*) &= \mathbf{x}^{*\top} \Sigma^{-1} \boldsymbol{\mu}_1 - \tfrac{1}{2} \boldsymbol{\mu}_1^\top \Sigma^{-1} \boldsymbol{\mu}_1 + \log(0.5) \\ &= (1 \cdot 0 + 1 \cdot 1.5) - \tfrac{1}{2}(4.5) + \log(0.5) \\ &= 1.5 - 2.25 + \log(0.5) = -0.75 + \log(0.5).\end{aligned}$$

$\delta_0 - \delta_1 = 0.25 - (-0.75) = +1.0 > 0$, so **class 0**. ✓

Solution *(1 %)* **(iv)** Raising π_0 from 0.5 to 0.8 raises $\log \pi_0$ in δ_0 and lowers $\log \pi_1$ in δ_1, so $\delta_0(\mathbf{x}) - \delta_1(\mathbf{x})$ gains $\log(0.8/0.2) = \log 4 \approx 1.39$ at every $\mathbf{x}$. The decision boundary, the level set $\delta_0 = \delta_1$, therefore *moves toward class 1's centroid* (away from class 0): *more* of the predictor space is now classified as class 0 (the now-more-prevalent class).

Grading: (i) 1 P for the expansion / discarded-terms identification, 1 P for the Σ-shared cancellation of $\mathbf{x}^\top \Sigma^{-1} \mathbf{x}$, 1 P for the QDA one-liner. (ii) 1 P each for the linear-in-$\mathbf{x}$ side, the constant-side computation, and the final collected equation. (iii) full credit for either the boundary check or the direct δ_k comparison. (iv) standard direction-of-effect trap — it is easy to invert. The prof has flagged this kind of "algebra in public" as a place to write every line.

b) Mini-batch SGD pseudocode and implicit regularization *(4 %)*

Solution *(2 %)* **(i) Pseudocode for one epoch of mini-batch SGD:**

> **Input:** training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, current parameters $\boldsymbol{\theta}$, batch size m (assume $m \mid N$, so the number of batches per epoch is $B = N/m$), learning rate η.
>
> 1. **Shuffle.** Draw a random permutation π of $\{1, \ldots, N\}$. Define the B disjoint mini-batches $\mathcal{B}_b = \{\pi((b-1)m + 1), \ldots, \pi(bm)\}$ for $b = 1, \ldots, B$.

2. **For each mini-batch** $b = 1, \ldots, B$**:**
   (a) **Mini-batch gradient (via the gradient oracle, i.e. backprop):**
   $$\widehat{\nabla L}(\boldsymbol{\theta}) \;=\; \frac{1}{m} \sum_{i \in \mathcal{B}_b} \nabla_{\boldsymbol{\theta}} \ell_i(\boldsymbol{\theta}).$$
   (b) **Parameter update:**
   $$\boldsymbol{\theta} \;\leftarrow\; \boldsymbol{\theta} \;-\; \eta \cdot \widehat{\nabla L}(\boldsymbol{\theta}).$$
3. **End of epoch.** Every observation has been seen exactly once (in exactly one mini-batch). Return the updated $\boldsymbol{\theta}$ (and proceed to the next epoch, or stop if the early-stopping / fixed-E criterion is met).

Important detail. The shuffle in step 1 must be redone at the *start* of each epoch; without it, the same batches in the same order constitute an undesirable deterministic schedule. The gradient oracle is the backpropagation algorithm internally; you may treat $\nabla_{\boldsymbol{\theta}} \ell_i$ as a known primitive.

Solution *(1 %)* **(ii) (A) True.** Since each batch is a uniform random sample (without replacement at the within-epoch level, but the same expectation argument as with replacement applies), $\mathbb{E}_{\text{batch}}[\widehat{\nabla L}] = (1/N) \sum_{i=1}^{N} \nabla \ell_i = \nabla L(\boldsymbol{\theta})$. **(B) False.** For m i.i.d. summands, $\text{Var}(\widehat{\nabla L}) \propto 1/m$, so doubling m *halves* (not doubles) the variance, all else equal. The statement inverts the relationship. **(C) True.** The course's claim (Apr 21): the per-step gradient *noise* from small batches biases the optimizer toward *flat* minima (which generalize better) and away from *sharp* minima. This is the prof's "implicit L^2 regularization" story.

Solution *(1 %)* **(iii)** The implicit-regularization story is a bias-for-variance trade applied to the *optimizer's trajectory*, not to the explicit loss: the SGD-noise bumps the iterates away from sharp minima where small parameter perturbations cause large loss changes (high effective variance under data resampling), at the cost of not finding the absolute global minimum of the training loss (a small upward bias in training error). Explicit regularizers (ridge, lasso, dropout, early stopping) act on either the loss surface or the parameters / activations directly; mini-batch SGD's gradient noise achieves a related effect "for free" through the optimization dynamics.
Grading: (i) 2 P — needs the shuffle at the start of the epoch, the gradient on the mini-batch, the update rule $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\widehat{\nabla L}$*, and the "every observation seen exactly once per epoch" detail; 0.5 P off for missing the shuffle, 0.5 P off for forgetting that the update happens* within *the inner loop (not once at the end of the epoch). (ii) 1 P for the full T/F/T pattern. (iii) 1 P for any defensible bias-for-variance framing tying SGD noise to flat-vs-sharp minima or to the same effect as explicit regularizers. Common trap on (ii)(C): saying "False — smaller batches mean more variance, not more regularization" confuses the variance of the gradient estimator with the regularization effect on the learned model. The two run in the same direction here, by design.*

c) k-fold CV pseudocode and the one-standard-error rule *(4 %)*

Solution *(2 %)* **(i) Pseudocode for** $(\text{CV}_k(\theta), \widehat{\text{SE}}(\text{CV}_k(\theta)))$**:**

Input: training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, model class $\mathcal{M}(\theta)$, grid Θ, number of folds k.

1. Randomly partition $\{1, \ldots, n\}$ into k disjoint folds $F_1, \ldots, F_k$ of roughly equal size.
2. For each $\theta \in \Theta$:
   (a) For each fold $j = 1, \ldots, k$:
   - Train $\hat{f}_\theta^{(-j)} = \mathcal{M}(\theta)$ on $\mathcal{D} \setminus F_j$.
   - Compute the per-fold MSE $E_j(\theta) = \frac{1}{|F_j|} \sum_{i \in F_j} (y_i - \hat{f}_\theta^{(-j)}(x_i))^2$.

(b) Aggregate:

$$\mathrm{CV}_k(\theta) = \frac{1}{k}\sum_{j=1}^{k} E_j(\theta), \qquad \widehat{\mathrm{SE}}(\mathrm{CV}_k(\theta)) = \sqrt{\frac{1}{k(k-1)}\sum_{j=1}^{k}\big(E_j(\theta) - \mathrm{CV}_k(\theta)\big)^2}.$$

(Equivalently: sample SD of the k per-fold MSEs divided by $\sqrt{k}$.)

3. Return $\{(\mathrm{CV}_k(\theta), \widehat{\mathrm{SE}}(\mathrm{CV}_k(\theta)))\}_{\theta\in\Theta}$.

Important detail. The fold partition in step 1 is created *once* and reused across all $\theta \in \Theta$, so that the same observation i is held out in the same fold for every hyperparameter — the CV curves are then more comparable than if independent partitions were drawn per θ. Recall (P2(b)(iv)) that $\widehat{\mathrm{SE}}$ is "not quite valid" as a true SE because the folds share observations.

Solution *(1 %)* **(ii) One-standard-error rule.** Let $\theta_{\min}$ be the minimizer of $\mathrm{CV}_k(\theta)$ with CV value $m^\star$ and standard error $\mathrm{SE}^\star$. Among all $\theta \in \Theta$ whose CV is within one SE of $m^\star$, pick the *simplest*:

$$\hat\theta_{1\mathrm{SE}} = \arg \max_{\substack{\theta\in\Theta \\ \mathrm{CV}_k(\theta)\le m^\star + \mathrm{SE}^\star}} [\mathrm{simplicity}(\theta)].$$

"Simpler" direction:

- Ridge λ: simpler = *larger* λ (more shrinkage).
- Polynomial degree: simpler = *smaller* degree (fewer basis functions).
- k in KNN: simpler = *larger* k (smoother decision boundary).

Solution *(1 %)* **(iii) Read off.** $\hat\lambda_{\min} = \boxed{0.1}$ with $m^\star = 0.418$ and $\mathrm{SE}^\star = 0.022$. **One-SE band:**

$$[\,0.418,\ 0.418 + 0.022\,] = [\,0.418,\ 0.440\,].$$

Which λ in $\{0.001, 0.01, 0.1, 1, 10\}$ satisfy $\mathrm{CV}_{10}(\lambda) \le 0.440$?

- $\lambda = 0.001$: CV $= 0.452 > 0.440$. No.
- $\lambda = 0.01$: CV $= 0.430 \le 0.440$. Yes.
- $\lambda = 0.1$: CV $= 0.418 \le 0.440$. Yes (the minimum itself).
- $\lambda = 1$: CV $= 0.435 \le 0.440$. Yes.
- $\lambda = 10$: CV $= 0.510 > 0.440$. No.

The largest λ in the band is

$$\boxed{\hat\lambda_{1\mathrm{SE}} = 1.}$$

The recommendation walks one grid-step toward the simpler / more-regularized side, picking up a slightly worse CV (0.435 vs. 0.418) but well within one SE and with substantially more shrinkage.

Grading: (i) 1 P for the partition / fit / aggregate skeleton, 1 P for the explicit $\widehat{\mathrm{SE}}$ *formula. (ii) 1 P, must include both the rule and all three "simpler"-direction labels. (iii) 1 P, must show the* $[0.418, 0.440]$ *band explicitly. The common trap is to pick* $\lambda = 0.01$ *(the band's other tail) instead of* $\lambda = 1$ *— the one-SE rule is asymmetric, it picks the* simplest *(largest-*λ*) value, not the opposite.*

Problem 4 (22 %) — Data analysis: regression on bike-sharing demand

a) OLS with a polynomial, an interaction, and collinearity *(10 %)*

Solution *(1 %)* **(i)** Count rows of the coefficient table: intercept, `temp`, $I(\texttt{temp}^2)$, `feel_temp`, `humidity`, `windspeed`, `workingday`, 3 season dummies, 2 weather dummies, `temp:workingday` $= 1{+}1{+}1{+}1{+}1{+}1{+}1{+}3{+}2{+}1 = \boxed{13}$ parameters. Residual d.f. $= n_{\text{train}} - p = 530 - 13 = 517$, matching the printed value. ✓

Solution *(2 %)* **(ii) Two numerical symptoms.** (1) The standard errors of `temp` (0.85) and `feel_temp` (0.87) are 5–10× larger than those of the other continuous predictors (`humidity` 0.09, `windspeed` 0.08). (2) The two coefficient estimates have nearly opposite signs (+2.10 and −0.80), even though both predictors must be *positively* associated with bike count individually (warmer feels-like temperature should also encourage cycling): the model has shuffled the joint effect across the two columns arbitrarily. **(a)** The phenomenon is **collinearity** (between `temp` and `feel_temp`, $r = 0.99$). **(b)** Under near-proportional columns, $\mathbf{X}^\top\mathbf{X}$ has a tiny eigenvalue in the corresponding direction, so $(\mathbf{X}^\top\mathbf{X})^{-1}$ has huge diagonal entries on the two collinear coefficients — $\text{Var}(\hat{\beta}_j) = \sigma^2[(\mathbf{X}^\top\mathbf{X})^{-1}]_{jj}$ blows up. The joint contribution of the two predictors to $\hat{y}$ is still well-determined (the *sum* $\beta_{\text{temp}} \cdot \texttt{temp} + \beta_{\text{feel\_temp}} \cdot \texttt{feel\_temp}$ is identifiable from the data, even when the individual coefficients are not), which is why the joint significance test remains overwhelming. **(c)** The diagnostic note tells us that test MSE is essentially unchanged when `feel_temp` is dropped, and that $\hat{\beta}_{\texttt{temp}}$ collapses from +2.10 (SE 0.85) to +1.32 (SE 0.09). So collinearity here is hurting *interpretation* (we cannot tell "which of the two thermometers matters"), not *prediction* (held-out MSE is stable). This is the classic "prediction stable, interpretation unstable" fingerprint.
Grading: 1 P for naming collinearity and quoting the $(\mathbf{X}^\top\mathbf{X})^{-1}$ mechanism; 1 P for the "prediction stable / interpretation unstable" distinction in (c) backed by the diagnostic note. Half credit if (c) says "yes, dropping will degrade prediction" — a standard inversion of the right answer.

Solution *(2 %)* **(iii)** On a weekend (`workingday`$= 0$) the temperature-dependent part of the prediction is

$$g(\texttt{temp}) = 2.10 \cdot \texttt{temp} + (-0.55) \cdot \texttt{temp}^2 + \text{const.}$$

Differentiate and set to zero:

$$g'(\texttt{temp}) = 2.10 - 1.10 \cdot \texttt{temp} = 0 \quad \Longrightarrow \quad \boxed{\texttt{temp}^\star = \frac{2.10}{1.10} \approx 1.9.}$$

Second derivative $g''(\texttt{temp}) = -1.10 < 0$, so this is a maximum (so predicted `count` is maximized here). *Sanity:* the quadratic in `temp` opens downward (negative leading coefficient), so a single interior maximum exists.

Why a polynomial in temperature is reasonable for bike-share count. Demand is low when it's very cold (cyclists stay home), rises with comfortable warmth, and falls again at extreme heat (uncomfortable, dehydrating) — the classic inverted-U pattern that a quadratic captures with one extra parameter.
Grading: 1 P for the derivative-set-to-zero step and the numeric, 1 P for the interpretation. Deduct 0.5 if the second-derivative / opens-downward check is missing for the "maximum vs. minimum" question.

Solution *(2 %)* **(iv)** The partial-effect-of-`temp` function is now

$$g_{\text{wd}}(\texttt{temp}) = 2.10 \cdot \texttt{temp} + (-0.55) \cdot \texttt{temp}^2 + 0.40 \cdot \texttt{temp} \cdot \texttt{workingday}.$$

Workday (workingday = 1). $\Delta\widehat{\texttt{count}}$ when temp goes $0 \to +1$:

$$\begin{aligned}\Delta\widehat{\texttt{count}}_{\text{wd}} &= [2.10 \cdot 1 - 0.55 \cdot 1^2 + 0.40 \cdot 1 \cdot 1] - [0] \\ &= 2.10 - 0.55 + 0.40 \\ &= \boxed{+1.95 \text{ thousand rentals.}}\end{aligned}$$

Weekend (workingday = 0). $\Delta\widehat{\texttt{count}}$ when temp goes $0 \to +1$:

$$\begin{aligned}\Delta\widehat{\texttt{count}}_{\text{we}} &= [2.10 \cdot 1 - 0.55 \cdot 1^2 + 0.40 \cdot 1 \cdot 0] - [0] \\ &= 2.10 - 0.55 + 0 \\ &= \boxed{+1.55 \text{ thousand rentals.}}\end{aligned}$$

Comparison. Both deltas are positive (warmer days drive more rentals), but the workday delta exceeds the weekend delta by 0.40. **Sign of the interaction:** $\hat{\beta}_{\texttt{temp:wd}} = +0.40 > 0$ means temperature *amplifies* demand more on workdays than on weekends — plausibly because workday demand is dominated by commuter cyclists who switch to the bike specifically when the weather cooperates, whereas weekend demand is anchored to recreational use that is less weather-sensitive.
Grading: 1 P for the workday delta (must include all three terms), 1 P for the weekend delta and a one-sentence interpretation. Deduct 0.5 if the interaction is omitted from the workday case.

Solution *(1 %)* **(v)** Wrong because of the **hierarchy principle**: when the model contains an interaction `temp:workingday`, the main effect `workingday` must remain in the model regardless of its individual p-value, otherwise the interaction is interpreted relative to an implicit zero main effect (and the model is no longer invariant under simple recoding of `workingday`). The $p = 0.074$ on `workingday` is the test of the "workday effect at `temp` $= 0$ specifically" (i.e. at the mean of standardized temp), not the test of "workingday matters anywhere."

Solution *(1 %)* **(vi)** The violated assumption is **homoscedasticity** (constant-variance residuals); "fanning out" is heteroscedasticity. Common remedy: log-transform the response (or use a Box–Cox / variance-stabilising transform), or fit a weighted least squares with weights inversely proportional to the estimated variance.

Solution *(1 %)* **(vii)** The **prediction interval** is wider. The confidence interval for $\mathbb{E}[\,\texttt{count} \mid x_0\,]$ quantifies only sampling uncertainty in $\hat{\beta}$, contributing $\hat{\sigma}^2 x_0^\top (\mathbf{X}^\top \mathbf{X})^{-1} x_0$. The prediction interval additionally accounts for the irreducible noise on the new observation, adding $\hat{\sigma}^2$ on top — so its half-width has the extra $\hat{\sigma}^2$ term and is always wider.

b) Ridge vs. lasso under collinearity *(5 %)*

Solution *(1 %)* **(i)** The ridge objective is

$$\boxed{\hat{\beta}^R_\lambda = \underset{\beta_0,\beta}{\operatorname{argmin}} \sum_{i=1}^{n} \big(y_i - \beta_0 - x_i^\top \beta\big)^2 + \lambda \sum_{j=1}^{p} \beta_j^2, \qquad \lambda \geq 0.}$$

The intercept β_0 is **not** penalised — the penalty sum runs over $j = 1, \ldots, p$ only.

Why standardise first. The L_2 penalty is scale-dependent: a predictor measured in large units has small fitted coefficients and is barely shrunk, while a predictor in small units has large coefficients and is heavily shrunk. Standardising every continuous predictor to mean 0 and unit variance ensures the penalty acts on every predictor on the same footing, so shrinkage is determined by signal strength rather than units.

Solution *(2 %)* **(ii)** The constraint regions differ in their *corners*:

- Ridge's $\sum_j \beta_j^2 \le t$ is a smooth ball, with no preferred axes. Along the collinear direction (`temp`, `feel_temp`), the optimum projects onto the ball at a point with both coordinates non-zero — the joint effect is *split* between them, each at roughly half the OLS magnitude (this minimises $\beta^2_{\texttt{temp}} + \beta^2_{\texttt{feel\_temp}}$ subject to a fixed sum, by symmetry).
- Lasso's $\sum_j |\beta_j| \le t$ is a diamond with corners on the coordinate axes. The optimum tends to lie at one of those corners, zeroing out exactly one of the collinear partners and concentrating the joint effect on the remaining one (the corner of the diamond is generically the point that minimises a smooth objective subject to the L_1 constraint).

This is why ridge "splits" and lasso "picks one" under collinearity — a structural geometric difference between the two penalties.

Solution *(1 %)* **(iii) One-SE rule.** Let $\hat{\lambda}_{\min}$ minimise the CV curve with value $m^\star$ and SE SE*. Choose the *largest* (most-regularized) λ such that CV$(\lambda) \le m^\star +$SE*. **Why the practitioner's preference is reasonable.** The 6-variable lasso loses only 0.023 of test MSE relative to the minimum-CV ridge (0.825 vs. 0.802), well within typical test-MSE fluctuations, in exchange for a model that fits on a single dashboard slide and is far easier to defend. The one-SE rule formalises this: "simplest model not statistically worse than the best."

Solution *(1 %)* **(iv)** The procedure is **principal-component regression (PCR)**. **Structural cost compared to lasso:** PCR's regressors (the leading PC scores) are *linear combinations of all original predictors*, so PCR does not produce a sparse model in the original variables — every original predictor enters every score. Lasso, by contrast, sets some original $\hat{\beta}_j$ exactly to zero, giving genuine variable selection.
Grading: (i) 1 P, must mention intercept exclusion. (ii) 2 P split as above; deduct 0.5 if only one of the two geometries is named. (iii) 1 P, must include both the rule statement and the interpretability justification. (iv) 1 P, must name PCR and *the no-sparsity-in-the-original-predictors cost.*

c) Gradient boosting and learning-rate / depth tradeoffs *(4 %)*

Solution *(2 %)* **(i) One round of gradient boosting (squared-error loss):**

> **Given:** current ensemble $\hat{f}^{(m-1)}(x)$ and shrinkage $\nu \in (0, 1]$.
>
> 1. Compute pseudo-residuals $r_i^{(m)} = y_i - \hat{f}^{(m-1)}(x_i)$ for $i = 1, \dots, n$ (this is the negative gradient of squared error w.r.t. $\hat{f}$, evaluated at the current ensemble).
> 2. Fit a regression tree T_m of depth d to the pairs $\{(x_i, r_i^{(m)})\}_{i=1}^n$.
> 3. Update the ensemble: $\hat{f}^{(m)}(x) = \hat{f}^{(m-1)}(x) + \nu \cdot T_m(x)$.
>
> After M rounds, the final ensemble is $\hat{f}^{(M)}(x) = \hat{f}^{(0)}(x) + \nu \sum_{m=1}^M T_m(x)$, where $\hat{f}^{(0)}$ is the initial prediction (typically the global mean $\bar{y}$).

Key points: (a) the new tree T_m is fit to the *current residuals*, i.e. what the ensemble has not yet explained; (b) the shrinkage ν enters as a multiplicative damping factor on the new tree's contribution; (c) the final ensemble is a ν-weighted sum.

Solution *(1 %)* **(ii) Smallest CV-MSE in the table:** $(d, \nu, M) = (5, 0.01, 5000)$ at $\boxed{0.54}$. The first row's $0.56 \to 0.61$ as M moves $1500 \to 5000$ shows **overfitting**: with $\nu = 0.10$ and shallow trees, 1500 trees already exhausts the structure in the data and additional trees fit residual noise, so test (here CV) error *rises*. A random forest does *not* display this trend because its trees are averaged (not summed) on bootstrap replicates, and averaging only reduces variance — "too many trees" is harmless for a random forest.

Solution *(1 %)* **(iii) Why** $(d, \nu, M) = (5, 0.01, 200)$ **is a poor configuration.** A small ν means each tree contributes a tiny correction; reaching a good fit requires *many* trees. At $M = 200$

with $\nu = 0.01$, only ≈ 2 units of effective "learning budget" have been spent, and the CV-MSE is the worst in the entire table (0.92, underfitting). The pair (M, ν) is coupled — halving ν requires roughly doubling M for comparable fit. Picking $M = 200$ wastes the slow-learning regime's main payoff.

d) GAM with B-splines *(3 %)*

Solution *(1 %)* **(i)** A cubic regression spline (bs) with $K = 3$ interior knots and degree $p = 3$, with the global intercept excluded, consumes

$$K + p = 3 + 3 = \boxed{6} \text{ degrees of freedom.}$$

(With the global intercept included it would be $K + p + 1 = 7$; the problem says "intercept removed," hence 6.)

Solution *(1 %)* **(ii) Total GAM degrees of freedom:**

- Global intercept β_0: 1.
- $s(\texttt{temp}, \text{df} = 5)$: 5.
- $s(\texttt{humidity}, \text{df} = 4)$: 4.
- $\text{bs}(\texttt{windspeed}, \text{knots} = \{q_{0.25}, q_{0.5}, q_{0.75}\})$: 6 (from (i)).
- $\beta_{\text{wd}} \cdot \texttt{workingday}$: 1.
- $g(\texttt{season})$ (4 levels, 1 reference, so 3 dummies): 3.
- $h(\texttt{weather})$ (3 levels, 1 reference, so 2 dummies): 2.

Total $= 1 + 5 + 4 + 6 + 1 + 3 + 2 = \boxed{22}$.

Solution *(1 %)* **(iii) Structural limitation:** a GAM is *additive* by construction — $\hat{y} = \beta_0 + \sum_j f_j(x_j)$ — so it cannot represent *interactions* between predictors (the predicted partial effect of one predictor does not depend on the value of another). A gradient-boosted ensemble of depth-d trees with $d \geq 2$ captures d-way interactions automatically by tree structure, which is exactly the gap a $\texttt{temp} \times \texttt{workingday}$ interaction (from part (a)) would have to be patched into the GAM manually as a tensor-product term to recover.
Grading: (i) 1 P, full credit for 6 *(*7 *with intercept earns half credit if the intercept choice is stated explicitly). (ii) 1 P, must match the sum. (iii) 1 P, must name the additivity constraint.*

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

a) Logistic regression with a continuous–continuous interaction *(10 %)*

Solution *(2 %)* **(i) Encoding assumption.** `debt_to_income` is in percentage points and `credit_score` is on the 300–850 scale; one "unit increase in `debt_to_income`" means one extra percentage point, not one extra standardised unit.

A one-percentage-point increase in `debt_to_income` changes the log-odds by

$$\Delta\hat{\eta} = \hat{\beta}_{\texttt{dti}} + \hat{\beta}_{\texttt{dti:score}} \cdot \texttt{credit\_score} = 0.250 + (-0.00030) \cdot \texttt{credit\_score},$$

so the odds multiply by $\exp(\Delta\hat{\eta})$.

At credit_score = 600:

$$\Delta\hat{\eta} = 0.250 + (-0.00030) \cdot 600 = 0.250 - 0.180 = 0.070,$$
$$\text{odds factor} = e^{0.070} \approx \boxed{1.073}.$$

At credit_score = 800:

$$\Delta\hat{\eta} = 0.250 + (-0.00030) \cdot 800 = 0.250 - 0.240 = 0.010,$$
$$\text{odds factor} = e^{0.010} \approx \boxed{1.010}.$$

Interpretation. The negative interaction means high-credit-score applicants are *less* sensitive to an extra percentage point of DTI than low-credit-score applicants — plausibly because a high-credit-score borrower has more headroom before DTI starts dominating their risk profile. *Grading: 1 P each for the two odds factors, all-or-nothing. Half credit if the interaction is dropped (gives* $\exp(0.250) \approx 1.284$ *for both, the standard trap).*

Solution *(1 %)* **(ii)** The main-effect $\hat{\beta}_2 = 0.250$ is the partial effect of `debt_to_income` *at* `credit_score` $= 0$, not the "average effect." Within the model's interaction, the effect varies with `credit_score`. The credit-score value at which $\hat{\beta}_2$ equals the partial DTI effect is exactly

$$\texttt{credit\_score} = 0,$$

which is outside the data's 300–850 range. A more honest reading would centre `credit_score` (subtract a mean like 650) before the interaction, after which the main-effect coefficient becomes the effect at the average credit score.

Solution *(3 %)* **(iii) Applicant.** `annual_income` $= 50$, `debt_to_income` $= 25$, `credit_score` $= 650$, `n_inquiries` $= 3$, `employment` = *self-employed* (so `employment_self` $= 1$, `employment_temp` $= 0$), `home_ownership` $= 0$.

$$\begin{aligned}
\hat{\eta} &= 4.50 \\
&\quad + (-0.012) \cdot 50 && = -0.600 \\
&\quad + (0.250) \cdot 25 && = +6.250 \\
&\quad + (-0.012) \cdot 650 && = -7.800 \\
&\quad + (0.150) \cdot 3 && = +0.450 \\
&\quad + (0.450) \cdot 1 && = +0.450 \quad \text{(employment_self)} \\
&\quad + (0.700) \cdot 0 && = 0 \quad \text{(employment_temporary)} \\
&\quad + (-0.400) \cdot 0 && = 0 \quad \text{(home_ownership)} \\
&\quad + (-0.00030) \cdot 25 \cdot 650 && = -4.875 \\
&= 4.50 - 0.600 + 6.250 - 7.800 + 0.450 + 0.450 + 0 + 0 - 4.875 \\
&= \boxed{-1.625}.
\end{aligned}$$

Sigmoid step:

$$\hat{p} = \sigma(-1.625) = \frac{1}{1 + e^{1.625}} = \frac{1}{1 + 5.078} = \frac{1}{6.078} \approx \boxed{0.164}.$$

Grading: 1 P each for (a) intercept + main effects + interaction laid out per term, (b) value $\hat{\eta} \approx -1.625$*, (c) sigmoid step / final* $\hat{p} \approx 0.164$*. Accept any answer in* $[0.16, 0.17]$*. The standard trap is to forget the interaction term (*$\hat{\eta}$ *drifts to* $+3.25$*,* $\hat{p} \to 0.96$*, an unrealistic flip).*

Solution *(1 %)* **(iv)** At threshold $\hat{p} = 0.5$ the applicant ($\hat{p} \approx 0.164$) is predicted to **not default**. **Threshold sensibility:** $\hat{p} = 0.5$ is generally *not* a sensible threshold for this loan-screening task, because (a) the base default rate is $\approx 18\%$, so almost no applicant will exceed 0.5 and the

classifier will hardly ever flag anyone, and (b) the cost asymmetry (a defaulted loan costs far more than a denied healthy applicant) calls for a *lower* threshold tuned by cost–benefit, not the symmetric 0.5 default.

Solution *(1 %)* **(v)** The fitted coefficient $\hat{\beta}_{\texttt{credit\_score}} = -0.012$ reflects a *conditional association* on observational data — "fancy correlation, not causal" (prof's wording). Credit score is itself an outcome of past financial behaviour and is correlated with omitted confounders (income stability, life events, financial literacy) that also drive default risk. The decrement is not a causal experiment; raising a customer's credit score by intervention (e.g. erasing a black mark on their report) would not necessarily lower their default probability by the amount the model predicts.

Solution *(2 %)* **(vi)** TP = 108, FN = 108, FP = 79, TN = 905; total = 1200 (216 actual defaults, 984 actual non-defaults).

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP}+\text{FN}} = \frac{108}{108+108} = \frac{108}{216} = \boxed{0.50},$$
$$\text{Specificity} = \frac{\text{TN}}{\text{TN}+\text{FP}} = \frac{905}{905+79} = \frac{905}{984} \approx \boxed{0.92},$$
$$\text{Error rate} = \frac{\text{FN}+\text{FP}}{\text{total}} = \frac{108+79}{1200} = \frac{187}{1200} \approx \boxed{0.16}.$$

Lowering the threshold $0.5 \to 0.3$**.** Sensitivity *rises* (more borderline cases become predicted defaults, catching more true defaulters) and specificity *falls* (more non-defaulters cross the lower threshold and become false alarms). This is the standard ROC trade-off, moving up-and-to-the-right on the ROC curve.
Grading: 1 P for all three metrics (must be correct to two decimals), 1 P for the direction of sens/spec under threshold lowering. Accept 0.91–0.92 *for specificity rounding.*

b) AdaBoost from exponential loss *(6 %)*

Solution *(2 %)* **(i) Inner-min over G reduces to weighted misclassification.** Split the sum at stage m by whether G classifies x_i correctly:

$$\sum_{i=1}^{N} w_i^{(m)} \exp\big(-y_i \alpha G(x_i)\big) = \sum_{i:\, y_i = G(x_i)} w_i^{(m)} e^{-\alpha} + \sum_{i:\, y_i \neq G(x_i)} w_i^{(m)} e^{+\alpha}.$$

Here we used $y_i G(x_i) = +1$ when correct and -1 when misclassified, so the per-observation exponent is $-\alpha$ or $+\alpha$.

Let $W = \sum_i w_i^{(m)}$ (total weight, fixed once $f^{(m-1)}$ is) and $\mathcal{E} = \sum_{i:\, y_i \neq G(x_i)} w_i^{(m)}$ (the weighted misclassification mass). Then

$$\begin{aligned}\sum_i w_i^{(m)} \exp(-y_i \alpha G(x_i)) &= e^{-\alpha}(W - \mathcal{E}) + e^{+\alpha}\mathcal{E} \\ &= e^{-\alpha} W + (e^{\alpha} - e^{-\alpha})\mathcal{E}.\end{aligned}$$

For fixed $\alpha > 0$, the multiplier $e^{\alpha} - e^{-\alpha} > 0$, and W does not depend on G. Minimising the objective over G is therefore equivalent to minimising $\mathcal{E}$, i.e.

$$\boxed{G_m = \operatorname*{argmin}_G \sum_{i=1}^{N} w_i^{(m)} \mathbb{1}[y_i \neq G(x_i)].}$$

The optimal G_m is the classifier that minimises the weighted misclassification rate under the stage-m weights. □

Solution *(2 %)* **(ii) Closed-form α_m.** Insert G_m from (i) into the objective and write it in terms of the weighted error rate $\mathrm{err}_m = \mathcal{E}/W$:

$$\begin{aligned} J(\alpha) &= e^{-\alpha}W + (e^{\alpha} - e^{-\alpha})\mathcal{E} \\ &= W\big[e^{-\alpha}(1-\mathrm{err}_m) + e^{+\alpha}\mathrm{err}_m\big]. \end{aligned}$$

$W > 0$ is irrelevant to the arg min; minimise the bracket. Differentiate w.r.t. α:

$$\frac{dJ/W}{d\alpha} = -e^{-\alpha}(1-\mathrm{err}_m) + e^{+\alpha}\mathrm{err}_m.$$

Set to zero:

$$e^{+\alpha}\mathrm{err}_m = e^{-\alpha}(1-\mathrm{err}_m) \quad \Longrightarrow \quad e^{2\alpha} = \frac{1-\mathrm{err}_m}{\mathrm{err}_m}.$$

Take logs and divide by 2:

$$\boxed{\alpha_m = \tfrac{1}{2}\log\frac{1-\mathrm{err}_m}{\mathrm{err}_m}.}$$

Second-derivative check: $d^2J/(W\,d\alpha^2) = e^{-\alpha}(1-\mathrm{err}_m) + e^{\alpha}\mathrm{err}_m > 0$, so this is a minimum. The textbook AdaBoost.M1 slide drops the $\frac{1}{2}$, absorbing it into the final sign-vote (which is invariant to a global positive rescaling of all α_m). □

Solution *(1 %)* **(iii)** From $f^{(m)} = f^{(m-1)} + \alpha_m G_m$:

$$w_i^{(m+1)} = \exp\big(-y_i f^{(m)}(x_i)\big) = \exp\big(-y_i f^{(m-1)}(x_i)\big)\cdot\exp\big(-y_i\alpha_m G_m(x_i)\big) = w_i^{(m)}\cdot\exp\big(-y_i\alpha_m G_m(x_i)\big).$$

Why misclassified observations get more weight. For an observation correctly classified by G_m, $y_i G_m(x_i) = +1$, so the multiplier is $e^{-\alpha_m} < 1$ (weight shrinks). For a misclassified observation, $y_i G_m(x_i) = -1$, so the multiplier is $e^{+\alpha_m} > 1$ (weight grows). Whatever the previous ensemble misclassified is exactly what the next learner is steered toward.

Solution *(1 %)* **(iv) AdaBoost vs. logistic comparison on the test set:**

	sens	spec	error
Logistic ($\hat{p} = 0.5$)	0.50	0.92	0.16
AdaBoost ($M = 300$)	0.56	0.92	0.16

AdaBoost has the same specificity and error rate but *higher sensitivity* (0.56 vs. 0.50), catching more true defaulters at the same false-alarm rate.

Recommendation. If a *defaulting loan is the costlier error*: deploy **AdaBoost** — its sensitivity advantage means it flags ≈ 13 more true defaulters per 216 at no cost in specificity. If a *denied healthy applicant is the costlier error* (or interpretability / fair-lending defensibility matters more): keep **logistic regression** — equal specificity and error, plus inspectable coefficients and odds ratios that meet regulatory transparency expectations.
Grading: (i) 1 P for the split-by-correct-vs-wrong, 1 P for arguing W is G-free. (ii) 1 P each for the derivative-set-to-zero step and the explicit closed form. (iii) 1 P for both the algebra and the multiplier interpretation. (iv) 1 P for the comparison table and the cost-asymmetry recommendation, must address both *directions of cost asymmetry.*

c) A neural-network classifier and a backprop derivation *(8 %)*

Solution *(1 %)* **(i) Layer-by-layer parameter count.**

- Input → hidden: weights $7 \times 12 = 84$, biases $12 \Rightarrow 96$ parameters.
- Hidden → output: weights $12 \times 1 = 12$, bias $1 \Rightarrow 13$ parameters.

Total $= 96 + 13 = \boxed{109}$ parameters.

Solution *(1 %)* **(ii)** Pre-activation:

$$\begin{aligned} z &= 0.30 \cdot 1.0 + (-0.20) \cdot (-0.5) + 0.10 \cdot 0 + 0.50 \cdot (-1.0) \\ &\quad + (-0.40) \cdot 1.0 + 0.20 \cdot 0 + 0.60 \cdot 0.5 + (-0.10) \\ &= 0.30 + 0.10 + 0 - 0.50 - 0.40 + 0 + 0.30 - 0.10 \\ &= \boxed{-0.30}. \end{aligned}$$

Post-activation: $h = \mathrm{ReLU}(-0.30) = \max(0, -0.30) = \boxed{0.00}$. The neuron is "dead" on this input — a common ReLU phenomenon.

Solution *(3 %)* **(iii) Backprop chain rule.**

(a) Compute $\partial L/\partial \eta$ for $L = -[y \log \hat{p} + (1-y)\log(1-\hat{p})]$ and $\hat{p} = \sigma(\eta)$.

$$\frac{\partial L}{\partial \hat{p}} = -\frac{y}{\hat{p}} + \frac{1-y}{1-\hat{p}} = \frac{-y(1-\hat{p}) + (1-y)\hat{p}}{\hat{p}(1-\hat{p})} = \frac{\hat{p} - y}{\hat{p}(1-\hat{p})}.$$

And $\partial \hat{p}/\partial \eta = \sigma'(\eta) = \hat{p}(1-\hat{p})$ (the standard sigmoid identity). Chain together:

$$\frac{\partial L}{\partial \eta} = \frac{\partial L}{\partial \hat{p}} \cdot \frac{\partial \hat{p}}{\partial \eta} = \frac{\hat{p} - y}{\hat{p}(1-\hat{p})} \cdot \hat{p}(1-\hat{p}) = \boxed{\hat{p} - y.} \quad \square$$

(The $\hat{p}(1-\hat{p})$ factors cancel cleanly — this is exactly why sigmoid + cross-entropy is the canonical output pair.)

(b) For the output-layer weight β_k $(k \geq 1)$, $\eta = \beta_0 + \sum_l \beta_l h_l$, so $\partial \eta/\partial \beta_k = h_k$. Chain:

$$\frac{\partial L}{\partial \beta_k} = \frac{\partial L}{\partial \eta} \cdot \frac{\partial \eta}{\partial \beta_k} = (\hat{p} - y) \cdot h_k = \boxed{(\hat{p} - y)\, h_k.}$$

(c) For the hidden-layer weight α_{kj}, follow the chain $L \to \eta \to h_k \to z_k \to \alpha_{kj}$:

$$\frac{\partial L}{\partial \alpha_{kj}} = \underbrace{\frac{\partial L}{\partial \eta}}_{\hat{p}-y} \cdot \underbrace{\frac{\partial \eta}{\partial h_k}}_{\beta_k} \cdot \underbrace{\frac{\partial h_k}{\partial z_k}}_{g'(z_k)} \cdot \underbrace{\frac{\partial z_k}{\partial \alpha_{kj}}}_{x_j} = \boxed{(\hat{p} - y)\, \beta_k\, g'(z_k)\, x_j.}$$

For ReLU activations, $g'(z_k) = \mathbb{1}[z_k > 0]$, so the gradient is exactly zero through any hidden unit with $z_k \leq 0$ (e.g. the neuron in part (ii)) — the source of ReLU's "dying ReLU" pathology when too many units sit on the inactive side.

Solution *(2 %)* **(iv) Three regularizers.**

- **Dropout.** Active *during training only*: each forward pass independently zeros each hidden unit with probability $p_{\text{drop}} = 0.2$ (here), and the surviving units are rescaled by $1/(1-p_{\text{drop}})$ (inverted-dropout convention). At test time *all* units are kept and outputs are not rescaled. Concretely, dropout randomises the architecture seen by each mini-batch, training an implicit ensemble that prevents co-adaptation between units.

- **Early stopping.** Active across both training and inference — training is halted at the epoch where validation loss first stops improving, and the parameters from that epoch are returned. At inference the (frozen) early-stopped parameters are used. Concretely, early stopping limits the effective optimisation horizon, acting as an implicit regularizer.

- **Label smoothing** $(\varepsilon = 0.05)$. Active during training only — the targets fed to the loss are softened. With $\varepsilon = 0.05$, the binary raw label $y = 1$ becomes the soft target $(t_1, t_0) = \boxed{(0.95,\ 0.05)}$ (the network is told "aim for 0.95, not 1"). At inference, predictions are unchanged — the smoothing only enters the loss.

Solution *(1 %)* **(v)** A dropout rate of 0.5 is *not* this course's standard practice; the prof's recommendation is ≈ 0.2, with anything above 0.4 generally regarded as too aggressive. **Bias–variance reason.** A high drop rate randomises the architecture so severely that the network under-fits: the effective capacity becomes too small, Bias^2 grows faster than the variance saving, and total test error *rises*. The bias–variance trade-off has a sweet spot at small drop rate, not at the extreme.
Grading: (i) 1 P; deduct 0.5 if biases are forgotten ($84 + 12 = 96$ *alone is the common slip). (ii) 1 P, full credit for* $z = -0.30$ *and* $h = 0$*. (iii) 1 P each for (a)/(b)/(c), must show the chain explicitly. (iv) 1 P each for the dropout / early-stop / label-smoothing summary with the correct training/inference flag and the* 0.95 *soft target. (v) 1 P, must invoke the bias–variance reason, not merely "unusual."*

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