TMA4268 V2026 Mock Exam 5 — Solution Proposal

Compiled for Anders Bekkevard

Companion to `mock-exam-5.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-5.tex` for the problem statements.

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

Solution *(1 P per blank.)*

(1) **collinearity**

(2) **shrinkage**

(3) **principal components**

(4) **dropout**

(5) **early stopping**

(6) **label smoothing**

(7) **mini-batches**

(8) **nested cross-validation**

(9) **AdaBoost**

(10) **gradient boosting**

Grading: 1 P per correct blank. For (1) "heteroscedasticity" is a hard zero (different diagnostic). For (2) "bagging" is wrong (bagging averages models, it does not shrink coefficients). For (9) "random forests" is wrong — random forests are parallel/independent bagged trees, not the re-weighting sequential scheme. The pairing AdaBoost (re-weighting) + gradient boosting (negative gradient) is the canonical contrast the prof drew in Module 9.

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

a) Bias–variance and double descent *(3 %)*

Solution *(1 P per statement.)*

(i) **False**. $\sigma^2 = \mathrm{Var}(\varepsilon)$ is a property of the data-generating noise; no amount of training data shrinks it. That is exactly why it is called the *irreducible* error.

(ii) **False**. The decomposition is an algebraic identity (the cross term vanishes by independence of ε and the training set — see Problem 3a) and holds for *any* estimator $\hat{f}$, including over-parameterized ones. Double descent reflects that the *variance term itself* can be non-monotone past the interpolation peak, not that the identity breaks.

(iii) **True**. That is the operational definition of "well-tuned": the regularizer is chosen (e.g. by CV) precisely so that the variance reduction exceeds the bias it introduces. If it did not, you would not use that λ.

b) Cross-validation and its pitfalls *(3 %)*

Solution *(1 P per statement.)*

(i) **True**. Each LOOCV training set has $n-1$ observations versus $\sim 0.9n$ for 10-fold; the estimator on $n-1$ training observations is a closer proxy for the estimator fit on n, so LOOCV has lower bias. (It pays for that with higher variance, since the n fold errors are highly correlated.)

(ii) **True**. This is the textbook definition of nested CV: inner folds tune, outer folds evaluate. The inner-fold winner is refit on the outer training data and scored on the outer held-out fold; the average across outer folds is the honest test-error estimate of the full *procedure* (tuning included).

(iii) **False**. Random folds shuffle the temporal order and put future and past observations in the same fold, leaking information across the time axis. The honest protocol is forward-chaining / rolling-window CV, where each training fold contains only observations earlier than its test fold; $k = 10$ does not fix the structural leak.

c) Bootstrap, standard errors and CIs *(3 %)*

Solution *(1 %)* **(i)** For $n = 10$, the probability that a particular original observation is *not* drawn in any of the n resamples is $(1 - 1/n)^n = (9/10)^{10}$. So

$$P(\text{included}) = 1 - \left(\tfrac{9}{10}\right)^{10} = 1 - 0.3487 \approx \boxed{0.651}.$$

Solution *(1 %)* **(ii)** The percentile 95% CI is the interval between the **2.5**% and **97.5**% quantiles of the bootstrap distribution:

$$\left[\,\hat{\theta}^*_{(0.025)},\ \hat{\theta}^*_{(0.975)}\,\right] \;=\; \left[\,\hat{\theta}^*_{(25)},\ \hat{\theta}^*_{(975)}\,\right] \quad \text{when } B = 1000,$$

i.e. the 25th and 975th order statistics of the sorted $\{\hat{\theta}^*_b\}_{b=1}^{1000}$.

Solution *(1 %)* **(iii)** **(A) True** — this is the bootstrap standard error by definition: $\widehat{\mathrm{SE}}_{\mathrm{boot}}(\hat{\theta}) = \sqrt{\frac{1}{B-1}\sum_b (\hat{\theta}^*_b - \bar{\hat{\theta}}^*)^2}$. **(B) False** — the bootstrap quantifies the *variability* of $\hat{\theta}$ (its sampling distribution / standard error / confidence interval); it does *not* correct for bias. The bootstrap distribution is centered around the original $\hat{\theta}$, not around the unknown true θ, so a biased estimator stays biased. **(C) False** — bootstrap is *with replacement* by definition. Drawing n observations without replacement from a size-n sample is just a permutation and yields no resampling variability.
Grading: 1 P all-or-nothing for the correct $(\mathrm{T}, \mathrm{F}, \mathrm{F})$ *pattern; deduct* 0.5 *if (B) is marked True (the canonical confusion: bootstrap quantifies variance, not bias).*

d) Neural network regularization *(3 %)*

Solution *(1 P per statement.)*

(i) **False**. Dropout is active *only at training time*; at test time the full network is used and the weights are scaled by the keep-probability (or, equivalently, the "inverted" dropout convention divides by the keep-probability during training so that no rescaling is needed at inference). Always-on dropout would introduce stochasticity into predictions.

(ii) **True**. Label smoothing softens the targets so the network is penalized for assigning probability $\to 1$ on the training labels; the cross-entropy gradient never vanishes and the network is less prone to over-confident, badly calibrated predictions on unseen data.

(iii) **False**. L2 weight decay penalizes large weights, which forces the optimizer toward a less expressive solution and generally *increases* training error (an unconstrained OLS-style optimum can only do at least as well on training as a constrained one). Its purpose is to reduce *test* error (variance reduction at the cost of some bias).

e) Mini-batch SGD and backpropagation *(3 %)*

Solution *(1 P per statement.)*

(i) **False**. Backpropagation only *computes* the gradient $\partial L/\partial\theta$ for every parameter θ, via a chain-rule pass that reuses intermediate activations from the forward pass. The actual parameter update is performed by the *optimizer* (SGD, Adam, etc.) using the gradients that backprop delivered. Conflating "computes the gradient" with "performs the update" is the canonical confusion.

(ii) **True**. The mini-batch gradient $\frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}}\nabla L_i$ has expectation $\frac{1}{n}\sum_{i=1}^{n}\nabla L_i$ (the full-data gradient), provided the mini-batch is drawn uniformly at random; hence it is unbiased.

(iii) **True**. Powers of two align well with GPU memory architecture / warp sizes; the choice is a hardware/throughput convention, not a statistical one (within reason the statistical properties of SGD are largely insensitive to whether the batch is 128 or 130).

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

Solution *(1 %)* **(i)** With $\text{err}_m = 0.20$:

$$\alpha_m = \log\frac{1-0.20}{0.20} = \log\frac{0.80}{0.20} = \log 4 \approx \boxed{1.39}.$$

A correctly-classified weak learner ($\text{err} < 0.5$) gets a positive vote-weight; a coin-flip ($\text{err} = 0.5$) gets weight 0; an anti-correlated one ($\text{err} > 0.5$) would get a negative weight (its votes are flipped).

Solution *(1 %)* **(ii)** With $L(y,f) = \frac{1}{2}(y-f)^2$,

$$-\left.\frac{\partial L}{\partial f}\right|_{f=\hat{f}^{(m-1)}(x_i)} = (y-f)\big|_{\hat{f}^{(m-1)}(x_i)} = y_i - \hat{f}^{(m-1)}(x_i) \;=\; r_i^{(m)}.$$

The negative gradient *is* the current **residual**. This is why gradient boosting with squared-error loss reduces to "fit a tree to the residuals at each round."

Solution *(2 %)* **(iii)** *(0.5 P per statement.)*

- **(A) True**. Unlike random forests, boosting is sequential and each tree fits residual structure; with too many trees it starts fitting noise. M is a real tuning parameter chosen by CV (or a validation set).

- **(B) True**. RF averages independent bootstrap trees; averaging only reduces variance and never inflates bias, so test error is monotonically non-increasing in B. One simply picks "enough" (typically 500–1000).
- **(C) False**. Smaller learning rate ν means each tree contributes less, so you need *more* trees (M) to reach the same level of fit. The pair (M, ν) is tuned jointly — halving ν approximately doubles the required M.
- **(D) True**. XGBoost (Chen & Guestrin, 2016) adds Newton-style second-order information (uses both gradient and Hessian of the loss), L1/L2 penalties on the leaf weights, plus engineering tricks (column subsampling, sparsity-aware splits, parallelism).

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

Solution *(1 %)* **(i)** A change of $+2$ mmol/L in `ldl` multiplies the odds by

$$\exp(0.30 \cdot 2) = e^{0.60} \approx \boxed{1.82}.$$

Equivalently, an 82% *increase in the odds of CHD per* 2 *mmol/L of LDL.*

Solution *(1 %)* **(ii)** For a *smoker*, a one-year increase in `age` changes the log-odds by $\hat{\beta}_{\text{age}} + \hat{\beta}_{\text{age:smoker}} = 0.04 + 0.03 = 0.07$, so the odds multiply by

$$e^{0.07} \approx \boxed{1.07}.$$

For a non-smoker the factor would be $e^{0.04} \approx 1.04$. *The standard interaction trap is to forget the main effect and quote* $e^{0.03} \approx 1.03$ *for the smoker increment, which is wrong — the smoker's per-year effect is the* sum *of the main and interaction coefficients.*

Solution *(1 %)* **(iii) False**. The relationship between $\hat{\beta}_j$ and $\Pr(Y = 1)$ is multiplicative-on-the-odds, not additive-on-the-probability. The local probability derivative is $\partial p / \partial x_j = \hat{\beta}_j \, p(1 - p)$, which depends on the current p; near $p = 0.5$ it is $\approx 0.25\hat{\beta}_j$ (smaller than $\hat{\beta}_j$), and near $p = 0$ or $p = 1$ it is essentially zero (the sigmoid saturates).

h) PCA: explained variance and loadings *(3 %)*

Solution *(1 %)* **(i)** For *standardized* variables each has variance 1, so the total variance equals the number of variables: $\sum_j \lambda_j = p = \boxed{6}$. *Sanity:* $2.7 + 1.5 + 0.9 + 0.5 + 0.3 + 0.1 = 6.0$. ✓

$$\text{PVE}_{1+2} = \frac{2.7 + 1.5}{6} = \frac{4.2}{6} = \boxed{0.70}\ (70\%).$$

Solution *(1 %)* **(ii)** Cumulative sums of $\lambda_k/6$:

$$\begin{aligned} \text{PC1}&: 2.7/6 = 0.450, \\ \text{PC1+PC2}&: 4.2/6 = 0.700, \\ \text{through PC3}&: 5.1/6 = 0.850, \\ \text{through PC4}&: 5.6/6 \approx 0.933, \\ \text{through PC5}&: 5.9/6 \approx 0.983. \end{aligned}$$

We first cross 0.90 at PC4, so $\boxed{4}$ components are needed.

Solution *(1 %)* **(iii) True**. PCA is unsupervised: the loadings are the eigenvectors of $\frac{1}{n}\mathbf{X}^\top\mathbf{X}$ (or the SVD of $\mathbf{X}$), computed without reference to y. (This is precisely why PCA *can* discard directions correlated with the response if those directions happen to be low-variance — a known limitation; PLS is the supervised alternative that does look at y.)

i) Discriminant analysis and splines *(3 %)*

Solution *(1 %)* **(i) True**. With a shared $\mathbf{\Sigma}$ the quadratic term $\mathbf{x}^\top \mathbf{\Sigma}^{-1}\mathbf{x}$ cancels across classes, leaving a discriminant $\delta_k(\mathbf{x}) = \mathbf{a}_k^\top \mathbf{x} + c_k$ that is linear in $\mathbf{x}$; hence boundaries $\delta_k = \delta_\ell$ are hyperplanes.

Solution *(1 %)* **(ii) True**. Class-specific $\mathbf{\Sigma}_k$ means $\mathbf{x}^\top \mathbf{\Sigma}_k^{-1}\mathbf{x}$ depends on k and no longer cancels; the discriminant picks up a quadratic-in-$\mathbf{x}$ term, so the boundary $\delta_k(\mathbf{x}) = \delta_\ell(\mathbf{x})$ is a quadratic surface (conic in 2D).

Solution *(1 %)* **(iii)** A cubic regression spline with K interior knots has $K+3$ spline basis terms (the truncated-power basis $1, x, x^2, x^3$ contributes 3 non-constant columns, and each interior knot adds one $(x-\xi_j)_+^3$ term). Including the global intercept this gives $K+3+1 = \boxed{K+4}$ parameters.

Equivalent route: the cubic polynomial uses 4 basis functions $\{1, x, x^2, x^3\}$, then each of the K interior knots adds one more, giving $K+4$ in total. The course convention used in Mock 2 follows this count.

Problem 3 (16 %) — Theory and pseudocode

a) The mathy one — bias–variance decomposition *(7 %)*

Solution *(2 %)* **(i)** Substitute $y_0 = f(x_0) + \varepsilon$:

$$\begin{aligned}\mathbb{E}\big[(y_0 - \hat f(x_0))^2\big] &= \mathbb{E}\big[(f(x_0) + \varepsilon - \hat f(x_0))^2\big] \\ &= \mathbb{E}\big[(f(x_0) - \hat f(x_0))^2\big] + 2\,\mathbb{E}\big[\varepsilon\,(f(x_0) - \hat f(x_0))\big] + \mathbb{E}[\varepsilon^2].\end{aligned}$$

The third term is $\mathbb{E}[\varepsilon^2] = \mathrm{Var}(\varepsilon) + (\mathbb{E}[\varepsilon])^2 = \sigma^2 + 0 = \sigma^2$. For the cross term, note that $f(x_0)$ is deterministic and $\hat f(x_0)$ is a function of the training set $\mathcal{D}$, which is *independent* of the test-point noise ε. By independence and $\mathbb{E}[\varepsilon] = 0$:

$$\mathbb{E}\big[\varepsilon\,(f(x_0) - \hat f(x_0))\big] = \mathbb{E}[\varepsilon] \cdot \mathbb{E}\big[f(x_0) - \hat f(x_0)\big] = 0 \cdot (\cdots) = 0. \quad \square$$

Hence

$$\mathbb{E}\big[(y_0 - \hat f(x_0))^2\big] = \mathbb{E}\big[(f(x_0) - \hat f(x_0))^2\big] + \sigma^2.$$

Grading: 1 P for expanding the square; 1 P for the explicit independence / zero-mean argument that kills the cross term. Half credit if the student writes σ^2 appearing "by the variance of ε" without the cross-term step.

Solution *(3 %)* **(ii)** Add and subtract $\mathbb{E}[\hat f(x_0)]$ inside the squared difference:

$$\begin{aligned}\mathbb{E}\big[(f(x_0) - \hat f(x_0))^2\big] &= \mathbb{E}\big[\big((f(x_0) - \mathbb{E}[\hat f(x_0)]) + (\mathbb{E}[\hat f(x_0)] - \hat f(x_0))\big)^2\big] \\ &= \underbrace{(f(x_0) - \mathbb{E}[\hat f(x_0)])^2}_{=\mathrm{Bias}^2[\hat f(x_0)]} + 2\,\underbrace{\mathbb{E}\big[(f(x_0) - \mathbb{E}[\hat f(x_0)]) \cdot (\mathbb{E}[\hat f(x_0)] - \hat f(x_0))\big]}_{(\star)} + \underbrace{\mathbb{E}\big[(\hat f(x_0) - \mathbb{E}[\hat f(x_0)]}_{=\mathrm{Var}(\hat f(x_0))}\end{aligned}$$

The first term has no randomness ($f(x_0)$ is fixed and $\mathbb{E}[\hat f(x_0)]$ is a number) and is by definition $\mathrm{Bias}^2[\hat f(x_0)]$. The third is, by definition, $\mathrm{Var}(\hat f(x_0))$.

The cross term $(\star)$ vanishes: the factor $(f(x_0) - \mathbb{E}[\hat f(x_0)])$ is deterministic and pulls out, leaving

$$(\star) = (f(x_0) - \mathbb{E}[\hat f(x_0)]) \cdot \mathbb{E}\big[\mathbb{E}[\hat f(x_0)] - \hat f(x_0)\big] = (f(x_0) - \mathbb{E}[\hat f(x_0)]) \cdot 0 \;=\; 0,$$

since $\mathbb{E}[\hat{f}(x_0) - \mathbb{E}[\hat{f}(x_0)]] = 0$. Adding back the σ^2 from part (i):

$$\boxed{\mathbb{E}\big[(y_0 - \hat{f}(x_0))^2\big] = \underbrace{(f(x_0) - \mathbb{E}[\hat{f}(x_0)])^2}_{\text{Bias}^2} + \underbrace{\text{Var}(\hat{f}(x_0))}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Irreducible}}.}$$

What each term captures:

- **Bias**^{2}: systematic error from the model class. Captures the gap between the truth $f(x_0)$ and the *average* prediction at x_0 across training sets. Large for too-rigid models (e.g. linear fit to a non-linear f).
- **Variance**: how much $\hat{f}(x_0)$ jitters across re-draws of the training set $\mathcal{D}$. Large for too-flexible models (they chase noise).
- **Irreducible** σ^2: the variance of the test-point noise ε, a property of the data, untouched by any estimator.

Grading: 1 P for the correct algebraic add-and-subtract setup; 1 P for explicitly showing the cross term has expectation zero (using $\mathbb{E}[\hat{f} - \mathbb{E}[\hat{f}]] = 0$); 1 P for naming and explaining the three terms.

Solution *(1 %)* **(iii) (a) Bias higher under lasso:** the L1 penalty $\lambda\|\beta\|_1$ shrinks every coefficient toward zero and sets many exactly to zero, so the fitted function is on average further from the truth than OLS (which is unbiased for the true β). **(b) Variance lower under lasso:** shrunk / zeroed coefficients have less room to react to training-set noise, so $\hat{\beta}$ (and hence $\hat{f}(x_0)$) is less variable across re-draws of $\mathcal{D}$. By picking λ via CV we choose the sweet spot where the variance saving exceeds the bias introduced.

Solution *(1 %)* **(iv)** Regularization can lower variance much more than it raises bias (part (iii)), and in the over-parameterized / double-descent regime the implicit min-norm regularization shrinks variance *back down* past the interpolation peak — so the bias and variance can both fall as flexibility grows there. "Trade-off" describes the classical U-shape but is not a universal law.

b) Cross-validation: pseudocode and nested CV *(4 %)*

Solution *(2 %)* **(i)** k-fold CV at a fixed hyperparameter θ:

Input: training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, model $\mathcal{M}$ with hyperparam θ, integer k.
Output: CV estimate $\widehat{\text{CV}}_k(\theta)$ of the test MSE.

1. Randomly partition the index set $\{1, \ldots, n\}$ into k roughly equal-sized folds $\mathcal{F}_1, \ldots, \mathcal{F}_k$.
2. **for** $j = 1, \ldots, k$ **do**
   2.1 Let $\mathcal{D}_{-j} = \{(x_i, y_i) : i \notin \mathcal{F}_j\}$ (training set) and $\mathcal{D}_j = \{(x_i, y_i) : i \in \mathcal{F}_j\}$ (held-out fold).
   2.2 Fit $\hat{f}^{(-j)} \leftarrow \mathcal{M}(\mathcal{D}_{-j}; \theta)$ *on $\mathcal{D}_{-j}$ only.*
   2.3 Compute the per-fold MSE $\text{MSE}_j = \frac{1}{|\mathcal{F}_j|} \sum_{i \in \mathcal{F}_j} (y_i - \hat{f}^{(-j)}(x_i))^2$.
3. **return** $\widehat{\text{CV}}_k(\theta) = \frac{1}{k} \sum_{j=1}^k \text{MSE}_j$ (the mean of the k per-fold errors).

Notes: (a) partitioning is random and disjoint (every observation appears in exactly one held-out fold); (b) at iteration j the model is fit on $\mathcal{D}_{-j}$ only and scored on $\mathcal{D}_j$ — no leakage; (c) aggregation is a simple average of the k per-fold MSEs.
Grading: 1 P for partition + the fit-on-train, score-on-held-out logic; 1 P for correct aggregation. Deduct 0.5 for any leakage (e.g. standardizing using $\mathcal{D}$ rather than $\mathcal{D}_{-j}$ inside the fold).

Solution *(1 %)* **(ii)** Hyperparameter selection from a grid:

1. For each $\theta_g \in \{\theta_1, \ldots, \theta_G\}$, compute $\widehat{\text{CV}}_k(\theta_g)$ using the primitive in (i).
2. Set $\hat{\theta} = \arg\min_g \widehat{\text{CV}}_k(\theta_g)$ (or use the 1-SE rule, see Problem 4c).

3. Refit $\mathcal{M}$ on the *full* training set $\mathcal{D}$ at $\theta = \hat{\theta}$; this is the deployed model.

For an honest test-error estimate of *the entire procedure* (including the tuning), wrap the above inside an outer CV loop — nested CV (Problem 2b(ii)).

Solution *(1 %)* **(iii)** The proposed pipeline is biased *downward* because the top-25 predictor selection in Step 1 was done on the *full* training data using y, so the held-out folds in Step 2 are not actually held-out — their y values were used to choose the predictors that the inner CV will see. This is the canonical CV leak.

Corrected procedure: put the predictor-screening step *inside* every CV fold. Within each of the 10 folds: compute marginal correlations using *only* the training portion of that fold, choose the top 25 predictors, fit logistic regression on those, predict on the held-out fold. The fold errors are then honest.

c) Bootstrap by hand *(3 %)*

Solution *(1 %)* **(i)** A bootstrap sample of size n draws n observations independently and uniformly with replacement. The probability that a particular observation i is *not* drawn at a single draw is $1 - 1/n$; over n independent draws,

$$P(i \text{ not in sample}) = \left(1 - \tfrac{1}{n}\right)^n \xrightarrow[n\to\infty]{} e^{-1} \approx 0.368,$$

by the standard limit $(1 - 1/n)^n \to e^{-1}$.

Solution *(1 %)* **(ii)** Bootstrap estimate of $\widehat{\mathrm{SE}}(\hat{\theta})$ for $\hat{\theta}$ = sample median:

1. **for** $b = 1, \ldots, B$ **do**
   1.1 Draw a bootstrap sample $\mathcal{D}_b^*$ of size n **with replacement** from the original sample $\{X_1, \ldots, X_n\}$.
   1.2 Compute $\hat{\theta}_b^* = \mathrm{median}(\mathcal{D}_b^*)$.
2. **return** $\widehat{\mathrm{SE}}_{\mathrm{boot}}(\hat{\theta}) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}(\hat{\theta}_b^* - \bar{\hat{\theta}}^*)^2}$, where $\bar{\hat{\theta}}^* = \frac{1}{B}\sum_b \hat{\theta}_b^*$.

Typical B is 1000–10,000. The procedure works because the bootstrap distribution of $\hat{\theta}_b^*$ around $\hat{\theta}$ mimics the sampling distribution of $\hat{\theta}$ around θ.

Solution *(1 %)* **(iii)** The closed-form SE comes from the inverse Fisher information at the MLE, which assumes (a) the logistic model is correctly specified (linear log-odds, independent observations) and (b) we are in the large-sample regime where the normal approximation is good. The bootstrap is non-parametric: it samples actual rows of the data, so it captures real departures from those assumptions — e.g. *model mis-specification* (the true log-odds are non-linear in `balance`) or *dependence/heterogeneity* not encoded in the likelihood. The bootstrap SE being 2.5× larger suggests the Fisher SE is over-optimistic; the bootstrap is the more trustworthy of the two here.

d) Hierarchical clustering by hand *(2 %)*

Solution *(1 %)* **(i)** The starting off-diagonal entries:

$$d_{12} = 2,\ d_{13} = 8,\ d_{14} = 7,\ d_{23} = 9,\ d_{24} = 6,\ d_{34} = 3.$$

Merge 1 (height 2). The smallest dissimilarity is $d_{12} = 2$, so we merge $\{1\}$ and $\{2\}$ into $\{1, 2\}$ at height $\mathbf{h_1 = 2}$. Under *complete linkage* the new row/column entries are the *maxima* of the merged rows:

$$d(\{1,2\},\, 3) = \max(d_{13}, d_{23}) = \max(8, 9) = 9,$$
$$d(\{1,2\},\, 4) = \max(d_{14}, d_{24}) = \max(7, 6) = 7.$$

The recomputed (3×3) dissimilarity matrix is

$$D^{(1)} = \begin{pmatrix} 0 & 9 & 7 \\ 9 & 0 & 3 \\ 7 & 3 & 0 \end{pmatrix} \quad \text{(rows/cols: } \{1,2\}, \{3\}, \{4\}\text{)}.$$

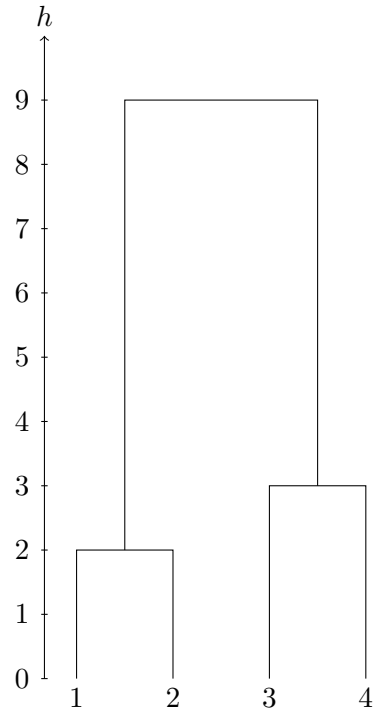
Merge 2 (height 3). The smallest off-diagonal in $D^{(1)}$ is $d(\{3\},\{4\}) = 3$; merge to get $\{3,4\}$ at height $\mathbf{h_2 = 3}$.

Solution *(1 %)* **(ii)** Only two clusters remain after Merge 2: $\{1,2\}$ and $\{3,4\}$. Under complete linkage the inter-cluster distance is the *maximum* pairwise distance:

$$d(\{1,2\},\{3,4\}) = \max(d_{13}, d_{14}, d_{23}, d_{24}) = \max(8,7,9,6) = \boxed{9}.$$

The final fusion therefore occurs at height $\mathbf{h_3 = 9}$.

Dendrogram (for orientation).



*Grading: 1 P for the two correct early fusions and heights; 1 P for the correct final fusion height (*9*, the max over the four cross-cluster pairs). Deduct 0.5 if the linkage update is taken as a min (single linkage) by mistake.*

Problem 4 (22 %) — Energy use of homes

a) Linear regression with a polynomial and an interaction *(7 %)*

Solution *(1 %)* **(i)** Count the rows of the coefficient table: intercept, $\widetilde{\texttt{area}}$, $\widetilde{\texttt{age}}$, $\widetilde{\texttt{age}}^2$, `insulation_standard`, `insulation_premium`, `heatpump`, $\widetilde{\texttt{area}}$:`heatpump`, $\widetilde{\texttt{rooms}}$ = $\boxed{9}$ parameters. Sanity: residual d.f. $= n_{\text{train}} - p = 600 - 9 = 591$, which matches the printout. ✓

Solution *(2 %)* **(ii)** `cost` is in 1000 EUR. For an increase of +1 SD in $\widetilde{\texttt{area}}$, holding everything else fixed:

No heat pump (`heatpump` = 0): $\Delta\widehat{\texttt{cost}} = \hat{\beta}_{\widetilde{\text{area}}} = 0.80\ (1000\,\text{EUR}) = \boxed{+800\ \text{EUR}}$,

Heat pump (`heatpump` = 1): $\Delta\widehat{\texttt{cost}} = \hat{\beta}_{\widetilde{\text{area}}} + \hat{\beta}_{\widetilde{\text{area}}:\text{heatpump}} = 0.80 + (-0.50) = 0.30\ (1000\,\text{EUR}) = \boxed{+300}$

Sign of the difference. The marginal cost per extra unit of area is *smaller* for heat-pump houses: heat pumps mitigate the area-driven growth in heating cost, exactly what you would expect of an efficient heat source whose efficiency does not degrade much in larger volumes.
Grading: 1 P each. Half credit if the EUR/1000 EUR conversion is forgotten. Deduct 1 P if the heat-pump effect is reported as −0.50 alone (forgetting the main $\widetilde{\texttt{area}}$ effect) — the standard interaction trap.

Solution *(2 %)* **(iii)** The two houses both have $\widetilde{\texttt{area}} = 0$, $\widetilde{\texttt{rooms}} = 0$, `insulation`=poor (reference, so both insulation dummies = 0), `heatpump` = 0. The fitted equation reduces to

$$\widehat{\texttt{cost}} = 2.80 + (-0.05)\,\widetilde{\texttt{age}} + 0.30\,\widetilde{\texttt{age}}^2.$$

House A ($\widetilde{\texttt{age}} = -1$):

$$\widehat{\texttt{cost}}_A = 2.80 + (-0.05)(-1) + 0.30(1) = 2.80 + 0.05 + 0.30 = 3.15 \quad \Rightarrow \boxed{\text{EUR}\,3{,}150/\text{yr}}.$$

House B ($\widetilde{\texttt{age}} = +2$):

$$\widehat{\texttt{cost}}_B = 2.80 + (-0.05)(2) + 0.30(4) = 2.80 - 0.10 + 1.20 = 3.90 \quad \Rightarrow \boxed{\text{EUR}\,3{,}900/\text{yr}}.$$

What the $\widetilde{\texttt{age}}^2$ term implies. The marginal effect of age is $\partial\widehat{\texttt{cost}}/\partial\widetilde{\texttt{age}} = -0.05 + 0.60\,\widetilde{\texttt{age}}$; at the average house ($\widetilde{\texttt{age}} = 0$) age has essentially no effect (-0.05), but at older houses the marginal cost of an extra year grows linearly with age (since the derivative itself grows with $\widetilde{\texttt{age}}$). Cost *accelerates* at older ages — a U-shape in age, minimised near $\widetilde{\texttt{age}} \approx 0.083$.

Solution *(1 %)* **(iv)** The high p-value of the *main* $\widetilde{\texttt{age}}$ effect is not evidence that `age` is irrelevant: the model also contains $\widetilde{\texttt{age}}^2$ with a highly significant $\hat{\beta} = 0.30$ ($p < 0.001$). The total marginal effect of age is $-0.05 + 0.60\,\widetilde{\texttt{age}}$, which is non-zero everywhere except $\widetilde{\texttt{age}} \approx 0.083$. Under the *hierarchical principle* the course follows, if a higher-order term is kept then its lower-order siblings must be kept too; dropping $\widetilde{\texttt{age}}$ alone would force the parabola to pass through the origin in its derivative, which is a strong unjustified constraint.

Solution *(1 %)* **(v)** The classmate's interpretation ignores the $\widetilde{\texttt{area}}$:`heatpump` interaction: $\hat{\beta}_{\texttt{heatpump}} = -0.60$ is the heat-pump effect *at* $\widetilde{\texttt{area}} = 0$, not on average over all houses. The actual change in predicted cost from installing a heat pump at standardized area $\widetilde{\texttt{area}} = +1$ is

$$\Delta\widehat{\texttt{cost}} = \hat{\beta}_{\text{heatpump}} + \hat{\beta}_{\widetilde{\text{area}}:\text{heatpump}} \cdot 1 = -0.60 + (-0.50) = -1.10\ (1000\,\text{EUR}) = \boxed{-\text{EUR}\,1{,}100/\text{yr}},$$

so a heat pump in a one-SD-larger-than-average house saves about EUR 1,100/year, not EUR 600. The saving *grows with house size* — which is exactly why heat pumps are even more cost-effective in larger homes.

b) Collinearity diagnosis *(4 %)*

Solution *(1 %)* **(i)** The two numerical symptoms in the second row:

- The point estimate of $\hat{\beta}_{\widetilde{\text{area}}}$ collapsed dramatically (from 0.90 alone to 0.55 in the joint model) — and a new coefficient $\hat{\beta}_{\widetilde{\text{rooms}}} = 0.42$ "took" the rest. The two estimates are not identified separately; the data only constrains their sum.

- The standard errors *ballooned*: $\mathrm{SE}(\hat{\beta}_{\widetilde{\texttt{area}}})$ went from 0.06 to 0.15 (a 2.5× increase); similarly for $\widetilde{\texttt{rooms}}$.
- *Bonus tell:* adjusted R^2 barely budged (0.43 → 0.44) despite adding a "new" predictor — the two predictors carry essentially the same information.

Solution *(1 %)* **(ii)** With $r = 0.92$ the columns of $\mathbf{X}$ are nearly linearly dependent, so $\mathbf{X}^\top\mathbf{X}$ has a near-zero eigenvalue and $(\mathbf{X}^\top\mathbf{X})^{-1}$ has a near-infinite eigenvalue. Since $\mathrm{Var}(\hat{\beta}) = \sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$, the coefficient variances inflate; equivalently, the variance-inflation factor (VIF) for each of the two collinear predictors is approximately $1/(1-r^2) = 1/(1-0.846) \approx 6.5$.

Solution *(1 %)* **(iii) No, predictions are not unstable.** While the two *individual* coefficients are poorly identified, the *sum* $\hat{\beta}_{\widetilde{\texttt{area}}}\widetilde{\texttt{area}} + \hat{\beta}_{\widetilde{\texttt{rooms}}}\widetilde{\texttt{rooms}}$ is well-determined — that is exactly what the data identifies. So $\hat{y}$ is fine; only the *interpretation* of which predictor "causes" the change is muddled. Collinearity is an interpretation problem, not a prediction problem (assuming the test-time predictors stay in the same collinear region).

Solution *(1 %)* **(iv)** Two methods (1 SD between trade-offs each):

- **Ridge regression.** Adds a penalty $\lambda\sum_j \beta_j^2$ that stabilizes $(\mathbf{X}^\top\mathbf{X} + \lambda I)^{-1}$, distributing the shared signal evenly across collinear predictors. *Trade-off:* introduces bias (shrinkage toward zero) but reduces coefficient variance dramatically — and is the canonical remedy for collinearity in this course.
- **Principal components regression (PCR)**. Replace the correlated predictors with their orthogonal principal components, then regress on the leading ones. *Trade-off:* the PCs are perfectly uncorrelated by construction (so OLS on them is well-conditioned), but interpretation in terms of the original predictors is lost — each PC is a linear combination.

Other defensible answers: the lasso (gives sparsity, can arbitrarily "pick" one of two near-identical predictors — not always desirable); partial least squares (supervised PCR); combining the two predictors into a derived variable like "area per room".

c) Lasso with 10-fold CV *(5 %)*

Solution *(1 %)* **(i)** The L1 penalty applies separately to each dummy column, but the columns of an indicator encoding depend on which level is the reference. Changing the reference level reparameterizes the dummies (e.g. what used to encode "standard vs poor" now encodes "standard vs premium"), with different coefficient magnitudes and hence different penalties — so the lasso path is *not* invariant.

Practical consequence. If the lasso shrinks one dummy to exactly zero, the interpretation is "no effect of *that* level relative to the chosen reference," not "no effect of that level in absolute terms." One can have `insulation_premium` dropped while `insulation_standard` stays in, and that just says "premium and the reference are indistinguishable on this λ"; changing the reference can give a quite different sparsity pattern. (The cleaner alternative is a *group lasso*, which keeps all dummies of a factor in or out together.)

Solution *(1 %)* **(ii) One-standard-error rule.** Among all values of λ whose CV-MSE is within one standard error (across the k folds) of the minimum CV-MSE, choose the *largest* (= most-regularized, simplest) one. Formally, with $\hat{\lambda}_{\min} = \arg\min_\lambda \mathrm{CV}(\lambda)$ achieving CV-MSE $m^\star$ and standard error $\mathrm{SE}^\star$,

$$\hat{\lambda}_{1\mathrm{SE}} = \max\{\lambda : \mathrm{CV}(\lambda) \le m^\star + \mathrm{SE}^\star\}.$$

Why prefer it: the CV minimum is itself a noisy estimate of test error, so picking the exact argmin tends to slightly overfit the validation folds. The 1-SE choice yields a sparser, more

interpretable, more stable model whose test performance is, within noise, indistinguishable from the best.

Solution *(1 %)* **(iii)** The small improvement (test MSE 0.205 → 0.196, ≈ 4.4% reduction) together with two coefficients dropped suggests that the two dropped predictors carry *little signal* given the others — their OLS estimates were noisy and shrinking them to zero removed more variance than the bias it introduced, hence the modest gain. But the gain is small, which means the two predictors were not zero either; their contribution to test MSE is real but minor. Likely candidates given the part (a) table: $\widetilde{\texttt{age}}$ main effect (insignificant in OLS, $p = 0.617$) and $\widetilde{\texttt{rooms}}$ ($p = 0.268$).

Solution *(2 %)* **(iv) Recommend** $\hat{\lambda}_{\text{1SE}}$ for interpretability. The 1-SE choice gives a 4-variable model versus the 6-variable model at $\hat{\lambda}_{\min}$ — two fewer effects to explain to stakeholders.

Concrete trade-off in test MSE: test MSE rises from 0.196 at $\hat{\lambda}_{\min}$ to 0.203 at $\hat{\lambda}_{\text{1SE}}$, a difference of 0.007 (about 3.5%). For the practitioner, that is a small predictive cost for a halved model size and a model whose CV performance is statistically indistinguishable from the optimum (that is exactly what the 1-SE rule guarantees). The $\hat{\lambda}_{\text{1SE}}$ model is also still below the OLS test MSE (0.205), so simplification did not push us past where we started.
Grading: 1 P for the recommendation with rationale; 1 P for the concrete numeric trade-off ($0.196 \to 0.203$*, +0.007 test MSE, −2 predictors).*

d) Gradient boosting interpretation and pseudocode *(6 %)*

Solution *(2 %)* **(i)** Squared-error gradient boosting (the "fit-residuals" view):

> **Input:** training data $\{(x_i, y_i)\}_{i=1}^n$, number of trees B, learning rate $\nu \in (0, 1]$, tree depth d.
> **Output:** ensemble $\hat{f}(x)$.
>
> 1. **Initialize** $\hat{f}^{(0)}(x) = \bar{y}$ (mean of y_i), and set residuals $r_i^{(1)} = y_i - \bar{y}$.
> 2. **for** $b = 1, \ldots, B$ **do**
>   2.1 Fit a regression tree T_b of depth d to the current residuals $\{(x_i, r_i^{(b)})\}_{i=1}^n$.
>   2.2 Update $\hat{f}^{(b)}(x) = \hat{f}^{(b-1)}(x) + \nu\, T_b(x)$.
>   2.3 Update residuals $r_i^{(b+1)} = y_i - \hat{f}^{(b)}(x_i) = r_i^{(b)} - \nu\, T_b(x_i)$.
> 3. **return** $\hat{f}(x) = \hat{f}^{(B)}(x) = \bar{y} + \nu \sum_{b=1}^B T_b(x)$.

Notes: (a) initialization to $\bar{y}$ is the constant that minimises the squared loss; (b) each round fits the *residuals* (gradient of squared loss, up to sign), so each new tree attacks what the ensemble so far is getting wrong; (c) shrinkage ν takes a small step in the direction of the new tree, which slows learning and improves generalization.

Solution *(1 %)* **(ii)** With squared-error loss $L(y, f) = \frac{1}{2}(y - f)^2$,

$$-\frac{\partial L}{\partial f}\bigg|_{f=\hat{f}^{(m-1)}(x_i)} = (y_i - f)\big|_{\hat{f}^{(m-1)}(x_i)} = y_i - \hat{f}^{(m-1)}(x_i) = r_i^{(m)}.$$

The current residual *is* the negative gradient; fitting a tree to the residuals is literally fitting to $-\partial L/\partial f$. For other (non-squared) losses the trick generalises: at each round fit a tree to the negative gradient and step in that direction.

Solution *(1 %)* **(iii)** Tuning (B, d, ν):

- $\boldsymbol{B}$ (number of trees): tune by CV (or a validation set). Too small ⇒ underfitting (high bias); too large ⇒ overfits (variance grows because each late tree fits noise).
- $\boldsymbol{d}$ (tree depth / interaction order): typically $d \in \{1, 2, 4, 6\}$. Small d (stumps) is purely additive (low variance per tree, may bias if real interactions exist); large d enables higher-order interactions but inflates variance.

- $\boldsymbol{\nu}$ (learning rate): typically $\nu \in [0.001, 0.1]$. Smaller ν takes shorter steps per round, reducing variance, but requires more trees to fit (so B must grow roughly proportionally to $1/\nu$). (B, ν) are tuned jointly.

Solution *(1 %)* **(iv) Not without early stopping.** Unlike random forests (where test error is monotone in B because trees are independent), boosting is sequential and can *overfit* for large B — late trees fit residual noise rather than signal. With $B = 10{,}000$ and $\nu = 0.1$, the chance of overfitting is real. The right move is to pick B by CV *or* use $B = 10{,}000$ together with early stopping (monitor validation loss and stop when it stops decreasing). With a sufficiently small ν and early stopping, "B too large" is benign — but the colleague's plan does not include either.

Solution *(1 %)* **(v)** The gradient-boosting test MSE (0.151) substantially below OLS/lasso (≈ 0.20) and below the additive linear-plus-polynomial model in part (a) suggests that the true relationship contains *interactions* that the additive linear / lasso models cannot capture — e.g. depth-d trees readily encode three-way interactions like `heatpump`×$\widetilde{\texttt{area}}$×`insulation`, while OLS would need every such product term entered by hand. To recover interpretability from the black-box ensemble, compute either (i) a **variable-importance plot** (gain or permutation importance) to rank the predictors by total contribution, or (ii) **partial-dependence plots** to see the average direction and shape of each top predictor's effect on `cost`.

Problem 5 (24 %) — Hospital-readmission classification

a) Logistic regression with an interaction *(7 %)*

Solution *(1 %)* **(i) Encoding assumption:** `diabetes`= 1 means the patient has diabetes (else 0); `insulin`= 1 means receiving insulin therapy (else 0); `sex`= 1 for male (female = 0, the stated reference).

Odds factor per additional prior admission. A unit increase in `prev_adm` multiplies the odds by

$$\exp(0.40) \approx \boxed{1.49}.$$

That is, each additional prior admission in the past year raises the odds of 90-day readmission by about 49%, holding everything else fixed.

Solution *(2 %)* **(ii)** Joint contribution of the three diabetes/insulin terms to the linear predictor, $\eta_{\text{DM,Ins}}(d, s) = 0.10\,d + 0.05\,s + 0.90\,ds$:

(`diabetes`, `insulin`)	Contribution to $\hat{\eta}$	Comment
$(0, 0)$	0	reference
$(1, 0)$	0.10	diabetes alone, modest
$(0, 1)$	0.05	insulin without diabetes, negligible
$(1, 1)$	$0.10 + 0.05 + 0.90 = 1.05$	dominated by the interaction

Odds factor (diabetic + insulin vs reference):

$$e^{1.05} \approx \boxed{2.86}.$$

A diabetic patient receiving insulin has roughly 2.86× the odds of 90-day readmission of a non-diabetic, non-insulin patient (holding other predictors fixed). Note that the main effects of `diabetes` and `insulin` are individually small, but together they "light up" through the interaction — the typical pattern for "diabetic-on-insulin = severe disease."
Grading: 1 P for the four correct contributions to $\hat{\eta}$; 1 P for the odds factor $e^{1.05} \approx 2.86$. Half credit if the interaction term is forgotten and only $e^{0.10+0.05} = e^{0.15} \approx 1.16$ is reported.

Solution *(2 %)* **(iii)** For a 70-year-old male patient with `los`= 6, `prev_adm`= 2, `diabetes`= 1, `insulin`= 1, `sex`= 1:

$$\begin{aligned}\hat{\eta} &= -4.20 + 0.030 \cdot 70 + 0.080 \cdot 6 + 0.40 \cdot 2 + \underbrace{0.10 + 0.05 + 0.90}_{\text{diabetes/insulin block (from (ii))}} + 0.15 \cdot 1 \\ &= -4.20 + 2.10 + 0.48 + 0.80 + 1.05 + 0.15 \\ &= \boxed{0.38}.\end{aligned}$$

Sigmoid step:

$$\hat{p} = \sigma(0.38) = \frac{1}{1 + e^{-0.38}} = \frac{1}{1 + 0.684} = \frac{1}{1.684} \approx \boxed{0.594}.$$

Grading: 1 P for the per-term assembly with the right diabetes/insulin block; 1 P for the sigmoid step. Accept any answer in $[0.59, 0.60]$*. Deduct 0.5 if the interaction is forgotten (*$\hat{\eta} = -0.52, \hat{p} \approx 0.37$*).*

Solution *(1 %)* **(iv)** The main-effect p-values of `diabetes` (0.578) and `insulin` (0.804) are testing whether each variable has an effect *at the value* 0 *of the other*. Because the interaction $\hat{\beta}_{\text{diab:ins}} = 0.90$ is highly significant ($p = 0.003$), the variables clearly matter — but mostly through their joint effect, not individually. By the hierarchical principle, if the interaction is in the model both main effects must stay; dropping them would force the joint effect to a peculiar parametric shape and break the canonical interpretation of main effects as conditional contrasts at zero.

Solution *(1 %)* **(v) Sensitivity** in this setting is the proportion of patients who *actually were* readmitted within 90 days that the classifier correctly flags as high-risk: "of all true readmissions, what fraction did we catch?" (Equivalently $\Pr(\hat{Y} = 1 \mid Y = 1)$.)

b) Confusion matrix and threshold tuning *(3 %)*

Solution *(1 %)* **(i)** Confusion-matrix totals: $54 + 126 = 180$ true readmissions, $72 + 648 = 720$ true non-readmissions, $n = 900$.

$$\begin{aligned}\text{Sensitivity} &= \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{54}{180} = \boxed{0.30}, \\ \text{Specificity} &= \frac{\text{TN}}{\text{TN} + \text{FP}} = \frac{648}{720} = \boxed{0.90}, \\ \text{Error rate} &= \frac{\text{FP} + \text{FN}}{n} = \frac{72 + 126}{900} = \frac{198}{900} = \boxed{0.22}.\end{aligned}$$

Solution *(1 %)* **(ii)** Lowering the threshold from 0.5 to 0.3 classifies *more* patients as readmissions. Therefore **sensitivity goes up** (more true readmissions now cross the lower bar and are caught) and **specificity goes down** (more true non-readmissions are now incorrectly flagged).

Solution *(1 %)* **(iii)** I would prioritize **sensitivity**: a false negative (a high-risk patient discharged without targeted follow-up) can have severe downstream consequences (another acute event, possibly fatal), while a false positive (extra follow-up call / care coordination) is relatively cheap. The clinical asymmetry of costs argues for catching as many true readmissions as possible, even at the cost of more false alarms — i.e. lower the threshold.

c) AdaBoost: pseudocode and a small numerical iteration *(6 %)*

Solution *(2 %)* **(i)** AdaBoost with M rounds (binary classification, labels $y_i \in \{-1, +1\}$):

Input: $\{(x_i, y_i)\}_{i=1}^N$, weak learner class $\mathcal{G}$, number of rounds M.
Output: ensemble classifier $G(x)$.

1. **Initialize** weights $w_i = 1/N$ for $i = 1, \ldots, N$.
2. **for** $m = 1, \ldots, M$ **do**
   2.1 Fit weak classifier $G_m \in \mathcal{G}$ to the training data, weighted by $\{w_i\}$.
   2.2 Compute the weighted misclassification rate: $\text{err}_m = \dfrac{\sum_{i=1}^N w_i \mathbb{1}\{y_i \neq G_m(x_i)\}}{\sum_{i=1}^N w_i}$.
   2.3 Compute the classifier weight: $\alpha_m = \log\dfrac{1 - \text{err}_m}{\text{err}_m}$.
   2.4 Update each weight: $w_i \leftarrow w_i \cdot \exp\big(\alpha_m \cdot \mathbb{1}\{y_i \neq G_m(x_i)\}\big)$.
   2.5 Renormalize: $w_i \leftarrow w_i / \sum_j w_j$ so that $\sum_i w_i = 1$.
3. **return** $G(x) = \text{sign}\Big(\sum_{m=1}^M \alpha_m G_m(x)\Big)$.

Notes: the weight update doubles down on the observations the current learner gets *wrong* (weighted by the learner's accuracy via α_m); the next learner therefore focuses on the hard cases. Renormalization keeps the weights interpretable as a probability distribution.
Grading: 0.5 P each for (a) initialization, (b) err_m *formula, (c)* α_m *formula, (d) weight update with* $\mathbb{1}\{y_i \neq G_m(x_i)\}$ *exponent; bonus 0 P for the final sign-of-weighted-sum ensemble (expected).*

Solution *(2 %)* **(ii)** Initial weights $w_i = 1/10 = 0.10$; two of ten are misclassified.

$$\text{err}_1 = \frac{2 \cdot 0.10}{10 \cdot 0.10} = \frac{0.20}{1.00} = \boxed{0.20},$$
$$\alpha_1 = \log\frac{1 - 0.20}{0.20} = \log\frac{0.80}{0.20} = \log 4 \approx \boxed{1.39}.$$

Un-normalized weight updates:

- Correctly-classified observation ($\mathbb{1}\{\cdot\} = 0$): $w_i \leftarrow 0.10 \cdot e^0 = \boxed{0.10}$ (unchanged).
- Misclassified observation ($\mathbb{1}\{\cdot\} = 1$): $w_i \leftarrow 0.10 \cdot e^{1.39} = 0.10 \cdot 4 = \boxed{0.40}$ (quadrupled).

Sanity: total weight after update is $8 \cdot 0.10 + 2 \cdot 0.40 = 0.80 + 0.80 = 1.60$, so post-normalization each correctly-classified weight is $0.10/1.60 = 0.0625$ and each misclassified one is $0.40/1.60 = 0.25$ — the misclassified observations now carry 25% of the weight each (up from 10%).
Grading: 0.5 P each for err_1*,* α_1*, correct-obs update, and misclassified-obs update. Accept* $\alpha_1 = \ln 4 \approx 1.386$ *to two decimals; the closed form* $\ln 4$ *is also full credit.*

Solution *(1 %)* **(iii)** Comparing the two test confusion-matrix summaries:

Model	Sensitivity	Specificity	Error rate
Logistic regression (5b(i))	0.30	0.90	0.22
AdaBoost ($M = 200$)	0.55	0.86	0.16

AdaBoost dominates on every clinically relevant criterion — substantially higher sensitivity (0.55 vs 0.30, almost twice as many true readmissions caught), only a small drop in specificity ($0.90 \to 0.86$, a 4 pp increase in false alarms), and a lower overall error rate (0.16 vs 0.22). Given that missing a high-risk patient is far costlier than an unnecessary follow-up call (part (b)(iii)), I would recommend **AdaBoost**, primarily on the *sensitivity* criterion.

Solution *(1 %)* **(iv) True**. AdaBoost can be derived as forward stagewise additive modelling under the exponential loss $L(y, f) = e^{-yf}$ with $y \in \{-1, +1\}$ (Friedman, Hastie & Tibshirani 2000). The resulting update rule for the additive ensemble matches AdaBoost's classifier weights α_m and weight updates exactly. This is also the reason boosting was eventually re-cast in the more general *gradient-boosting* framework: AdaBoost is one specific loss; squared error, log loss (logitboost), etc., all fit the same template.

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

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

$$\begin{aligned}\text{input} \to \text{hidden}: &\quad (7+1)\cdot 10 = 80,\\ \text{hidden} \to \text{output}: &\quad (10+1)\cdot 1 = 11,\\ \text{Total}: &\quad 80+11 = \boxed{91} \text{ parameters.}\end{aligned}$$

Layer-by-layer: $7\cdot 10 = 70$ input-to-hidden weights, 10 hidden-layer biases, $10\cdot 1 = 10$ hidden-to-output weights, 1 output-layer bias; $70+10+10+1 = 91$. ✓

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

$$\begin{aligned}v &= b + \sum_{j=1}^{7} w_j x_j\\ &= -0.20 + 0.20\cdot 0.5 + (-0.50)\cdot(-0.5) + 1.00\cdot 2 + 0.40\cdot 1 + (-0.60)\cdot 0 + 0.30\cdot 1 + 0.10\cdot(-1)\\ &= -0.20 + 0.10 + 0.25 + 2.00 + 0.40 + 0 + 0.30 - 0.10\\ &= 2.75.\end{aligned}$$

ReLU activation: $\text{ReLU}(2.75) = \max(0, 2.75) = \boxed{2.75}$.
Grading: 0.5 P for the correct pre-activation; 0.5 P for the ReLU step. Accept 2.74–2.76.

Solution *(3 %)* **(iii)** Backprop mini-derivation for $L_i = \frac{1}{2}(y_i - \hat{f}(x_i))^2$.

(a) *(1 %)* Differentiate L_i with respect to $\hat{f}(x_i)$, treating y_i as a constant:

$$\boxed{\frac{\partial L_i}{\partial \hat{f}(x_i)} = -\big(y_i - \hat{f}(x_i)\big) = \hat{f}(x_i) - y_i.}$$

(b) *(1 %)* Apply the chain rule with $\hat{f}(x_i) = \beta_0 + \sum_k \beta_k z_k$, so $\partial \hat{f}(x_i)/\partial \beta_k = z_k$ for $k \geq 1$:

$$\boxed{\frac{\partial L_i}{\partial \beta_k} = \frac{\partial L_i}{\partial \hat{f}(x_i)} \cdot \frac{\partial \hat{f}(x_i)}{\partial \beta_k} = -\big(y_i - \hat{f}(x_i)\big)\cdot z_k.}$$

(For β_0 the second factor is 1, giving $\partial L_i/\partial \beta_0 = -(y_i - \hat{f}(x_i))$.)

(c) *(1 %)* For the input-to-hidden weights, chain through $\hat{f} \to z_k \to v_{ik} \to \alpha_{kj}$. With $z_k = g(v_{ik})$ and $v_{ik} = \alpha_{k0} + \sum_j \alpha_{kj} x_{ij}$:

$$\begin{aligned}\frac{\partial L_i}{\partial \alpha_{kj}} &= \frac{\partial L_i}{\partial \hat{f}(x_i)} \cdot \frac{\partial \hat{f}(x_i)}{\partial z_k} \cdot \frac{\partial z_k}{\partial v_{ik}} \cdot \frac{\partial v_{ik}}{\partial \alpha_{kj}}\\ &= \big(-(y_i - \hat{f}(x_i))\big)\cdot \beta_k \cdot g'(v_{ik}) \cdot x_{ij}.\end{aligned}$$

$$\boxed{\frac{\partial L_i}{\partial \alpha_{kj}} = -\big(y_i - \hat{f}(x_i)\big)\,\beta_k\, g'(v_{ik})\, x_{ij}.}$$

Sanity: the gradient signal at the output, $(\hat{f} - y)$, is propagated backward through β_k, then through $g'(v_{ik})$ (vanishes when ReLU saturates at $v_{ik} \leq 0$, hence the classic "dead-ReLU" problem), then scaled by x_{ij} (the input that drives this weight). This is exactly the chain-rule structure that backprop exploits in O(forward-pass) time.
Grading: 1 P each for (a) the loss-output gradient, (b) the β_k gradient via chain rule, (c) the α_{kj} gradient via the full three-step chain. Deduct 0.5 in (c) for omitting the $g'(v_{ik})$ factor (the most common slip).

Solution *(2 %)* **(iv)** Two regularization techniques (must include one besides L2):

- **Dropout**, $p_{\text{drop}} = 0.2$ on the hidden layer (each neuron has a 20% chance of being zeroed at each training step). *Role:* prevents co-adaptation of hidden units — the network cannot rely on any one neuron being present, so it learns more robust, redundant features. Active only at training time; at test time the full network is used.
- **Early stopping** (no associated λ; the "hyperparameter" is the patience, e.g. 10 epochs without validation-loss improvement). *Role:* train on the training set but monitor validation loss; halt as soon as it stops improving. This caps the effective complexity of the fit before the network has time to memorise training noise. (Equivalent to a data-driven L2 in some idealised regression settings.)

Other defensible choices: L2 weight decay with $\lambda = 10^{-4}$ (the one the question *excluded*); L1 weight decay; label smoothing $\varepsilon = 0.05$ (Problem 2d(ii)); data augmentation if applicable.

Solution *(1 %)* **(v) Mini-batch SGD provides implicit regularization for free.** The gradient noise from drawing different mini-batches at each step acts like injecting noise into the optimization trajectory, biasing SGD toward *flat minima* of the loss landscape (regions where the loss is insensitive to small parameter perturbations); flat minima have been observed empirically to generalize better than sharp ones. This implicit regularization is one of the prof's hobbyhorses — it explains why an over-parameterized network trained with SGD can still generalize without explicit weight decay, and is closely related to the double-descent / benign-overfitting story.

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