TMA4268 V2026 Mock Exam 7 — Solution Proposal

Compiled for Anders Bekkevard

Companion to `mock-exam-7.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-7.tex` for the problem statements.

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

Solution *(1 P per blank.)*

(1) **regularization**

(2) **generalization error**

(3) **collinearity**

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

(5) **backpropagation**

(6) **label smoothing**

(7) **bagging and random forests**

(8) **boosting**

(9) **bootstrap**

(10) **nested cross-validation**

Grading: 1 P per correct blank. No partial credit for "close" alternatives. For (2), "training error" is a hard zero — the whole point of regularization is that it can leave training error untouched (or even raise it) while lowering test error. For (5), "boosting" is the most common trap; backpropagation is the gradient-computation algorithm, boosting is an ensemble method. For (10), plain "k-fold CV" is wrong because it conflates selection and assessment on the same folds, which biases the reported error downward.

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

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

Solution *(0.75 P per statement.)*

(i) **False**. σ^2 is the variance of the noise ε in $y_0 = f(x_0) + \varepsilon$; it is a property of the data-generating process, not of the estimator. The name "irreducible" is literal — no choice of $\hat{f}$ can lower it.

(ii) **True**. With large σ^2 a flexible model spends extra variance chasing noise that is irreducible by definition, so test error rises. Less-flexible models win when noise dominates — the prof's standard "high-noise tilts the trade-off toward simpler models" rule.

(iii) **True**. The double-descent / benign-overfitting regime: when $p \gg n$, an interpolating model can still generalize because the optimization implicitly picks the minimum-norm interpolator from the infinite family of zero-training-error solutions. (Prof's hobbyhorse.)

(iv) **True**. The lecturer explicitly renamed the trade-off "avoid bad overfitting" on the grounds that in the over-parameterized regime increasing p past n can lower both bias and variance simultaneously, so the classical strict trade-off framing is misleading.

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

Solution

(i) *(1 %)* **False**. The validation-set approach is a *single* train/test split. 2-fold CV averages over *both* splits (each half plays the role of training and test in turn), so it uses all of the data twice and is a better estimator. The two procedures give numerically different answers.

(ii) *(1 %)* **True**. Each LOOCV training set has $n - 1$ points (low bias as an estimator of test error trained on n), but the n training sets overlap almost completely, so the n fold errors are highly correlated and their average has higher variance than the 5- or 10-fold average.

(iii) *(1 %)* **False**. This is the canonical "right way / wrong way" CV trap. The Bayes error is 50% by construction, so any CV estimate well below that is artefactual. The variable-selection step (top-25 by marginal correlation with y) was done *before* CV using all of y, including the held-out folds, which leaks information and biases the CV error downward. The selection step must be *inside* every CV iteration.

(iv) *(1 %)* **True**. That is the definition of nested CV: inner folds choose θ, outer folds estimate the generalization error of the entire selection-plus-fit pipeline. The two roles must be on disjoint partitions or the outer estimate is biased downward.

c) Mini-batch SGD, dropout, label smoothing, early stopping *(4 %)*

Solution

(i) *(1 %)* **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, with variance scaling like $1/m$ but *positive*; reducing m trades a higher-variance estimate for a cheaper one (and supplies the noise that gives the implicit-L2 regularization the prof flagged as the headline NN fact).

(ii) *(1 %)* **True**. The course-stated range was 0.2–0.5, with the prof's recommended default 0.2; dropout is active only during *training*, and at test time all units are kept (with outgoing weights rescaled by the keep-probability, or the "inverted dropout" trick is applied at training time).

(iii) *(1 %)* **True**. Label smoothing replaces $(0, \dots, 0, 1, 0, \dots, 0)$ by $(\varepsilon/(C-1), \dots, 1-\varepsilon, \dots)$, which discourages the network from producing extreme logits and is partly motivated by the observation that hard labels over-confidently assume every training example is perfectly labelled.

(iv) *(1 %)* **False**. Early stopping uses the *validation* error, not the training error: training is halted at the epoch where validation error first stops improving (or starts rising), and the weights from that epoch are returned. Stopping at the training-error minimum would never stop — training error typically decreases monotonically — and would entirely miss the moment when generalization begins to deteriorate, which is the whole point of early stopping.

d) Boosting — AdaBoost and gradient boosting *(4 %)*

Solution

(i) *(1 %)* **True**. With squared-error loss $L(y, \hat{y}) = \frac{1}{2}(y-\hat{y})^2$ the negative gradient with respect to $\hat{y}$ is exactly $y - \hat{y}$, the residual. Fitting the next tree to the residual is fitting to the negative gradient, which is why squared-error gradient boosting is also called "forward stagewise regression on residuals."

(ii) *(1 %)* **False**. Boosting reduces *bias* relative to a single shallow tree, by fitting trees sequentially to the residuals (or negative gradient) of the current ensemble; the trees are *not* fit on independent bootstrap replicates, and they are *not* simply averaged — they are added in a stagewise sum with shrinkage. Variance reduction by bootstrap averaging is the mechanism of *bagging* and random forests, not boosting.

(iii) *(1 %)* **True**. Each tree's contribution to the ensemble is $\nu \cdot \hat{T}_m(x)$, so halving ν requires roughly twice as many trees for the cumulative contribution to reach the same level. The pair (M, ν) is tuned jointly — the conventional folklore is "smaller ν is better if you can afford the trees."

(iv) *(1 %)* **True**. $\alpha_m = \log((1-\text{err}_m)/\text{err}_m)$ is negative whenever $\text{err}_m > 0.5$, i.e. when the base classifier is worse than random; the sign flip lets AdaBoost *invert* a worse-than-chance vote and use it constructively (though if $\text{err}_m > 0.5$ regularly the weak learner is misspecified).

e) Collinearity *(4 %)*

Solution

(i) **True**. The textbook fingerprint of collinearity: large standard errors on the two affected coefficients, individual t-tests insignificant, but the partial F-test on the pair (or the overall regression) remains highly significant because the collinear pair jointly explains variation in y — the data just cannot tell which of the two does the explaining.

(ii) **False**. The reverse trap: predictions $\hat{y}$ are *stable* under collinearity (the fitted hyperplane is well-determined in the direction the data span), but the individual coefficients are unstable (any reallocation of the joint effect between the two nearly proportional columns leaves $\hat{y}$ unchanged). Interpretation suffers, prediction does not.

(iii) **False**. Standardisation rescales each column to unit variance but does not change the angle between columns or the rank of $\mathbf{X}^\top\mathbf{X}$. If two columns are nearly proportional before standardisation, they remain nearly proportional after, and $\mathbf{X}^\top\mathbf{X}$ remains nearly singular. Standardisation matters for ridge / lasso and for unit-free interpretation, not for collinearity.

(iv) **True**. Adding $\lambda\mathbf{I}$ shifts every eigenvalue of $\mathbf{X}^\top\mathbf{X}$ up by $\lambda > 0$, so $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$ is positive definite regardless of how degenerate $\mathbf{X}^\top\mathbf{X}$ was. Under near-perfect collinearity the ridge solution distributes the joint effect roughly equally between the two correlated coefficients (its smooth, energy-minimising allocation), which is the standard reason ridge is recommended for collinear designs.

f) Bootstrap for the standard error of a derived quantity *(3 %)*

Solution *(2 %)* **(i)** Pseudocode:

> **Input:** training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, query point x_0, $B = 1000$.
> **For** $b = 1, \ldots, B$:
>
> 1. Draw $\mathcal{D}^{*(b)} = n$ rows sampled *with replacement* from $\mathcal{D}$.
> 2. Fit logistic regression on $\mathcal{D}^{*(b)}$ to get $\hat{\beta}^{*(b)}$.
> 3. Compute and store $\hat{p}^{*(b)} = \sigma\big(x_0^\top \hat{\beta}^{*(b)}\big)$.
>
> **Return:**
>
> - $\widehat{\mathrm{SE}}_{\mathrm{boot}}(\hat{p}(x_0)) = \sqrt{\frac{1}{B-1}\sum_{b=1}^{B}\big(\hat{p}^{*(b)} - \overline{\hat{p}^*}\big)^2}$, where $\overline{\hat{p}^*}$ is the sample mean of the B bootstrap replicates;
> - the 95% percentile CI: the empirical 2.5% and 97.5% quantiles of $\{\hat{p}^{*(b)}\}_{b=1}^B$.

Choice of B: $B = 1000$ is the standard course recommendation — large enough that the Monte Carlo error in the SE estimate is small (the SE of an SE estimate scales like $1/\sqrt{B}$), and the percentile-CI quantiles at 2.5% and 97.5% are stable.
Grading: 1 P for the resampling loop and the sigmoid step inside the loop, 1 P for both outputs (SE and *percentile CI) with a defensible B. Half credit if $\hat{\beta}$ is re-resampled instead of the data (a common slip — you must refit on each resampled dataset).*

Solution *(1 %)* **(ii) False**. The bootstrap quantifies the *variability* of $\hat{p}(x_0)$ (its sampling distribution, standard error, and confidence interval); it does *not* correct for bias. The bootstrap distribution is centered around the original $\hat{p}(x_0)$ on $\mathcal{D}$, not around the unknown true probability, so a biased estimator stays biased.

g) Logistic regression coefficients *(3 %)*

Solution *(1 %)* **(i)** A Δx change in a continuous predictor multiplies the odds by $\exp(\hat{\beta} \cdot \Delta x)$:

$$\exp(0.18 \cdot 5) = \exp(0.9) \approx \boxed{2.46}.$$

Grading: full credit for ≈ 2.46. *Half credit for* $\exp(0.18) \approx 1.20$ *(forgetting the* $\Delta x = 5$ *multiplier).*

Solution *(1 %)* **(ii)** $\hat{p} = \sigma(1.2) = \dfrac{1}{1 + e^{-1.2}} = \dfrac{1}{1 + 0.3012} = \dfrac{1}{1.3012} \approx \boxed{0.769}$.

Solution *(1 %)* **(iii) True**. The L1 penalty's geometry (the diamond) puts mass at axis-aligned vertices, so the constrained MLE can sit exactly on a coordinate axis, zeroing the corresponding $\hat{\beta}_j$. With λ chosen by CV, the lasso for logistic regression performs simultaneous coefficient estimation and variable selection (one of the prof's standard "why lasso, not ridge" points).

h) Random forests *(1 %)*

Solution *(0.5 P per statement.)*

(i) **True**. Smaller `mtry` forces each split to consider fewer features, which decorrelates the trees (different trees fixate on different subsets of strong predictors). The variance of the average of B trees is $\rho\sigma^2 + (1-\rho)\sigma^2/B$, so reducing the pairwise correlation ρ is exactly how random forests improve over bagging.

(ii) **False**. The test error of a random forest is *monotonically non-increasing* in B (averaging only reduces variance, never causes overfitting), so the curve is not U-shaped and B is *not* tuned by CV. One simply picks "enough" trees — typically 500–1000. The standard contrast: in *boosting*, B *is* a real tuning parameter because boosting can overfit.

i) Principal component analysis *(2 %)*

Solution *(1 %)* **(i)** For standardized variables the total variance equals the number of variables, $\sum_j \lambda_j = p = 4$ (sanity: $1.80 + 1.10 + 0.70 + 0.40 = 4.00$). Cumulative PVE:

$$\begin{aligned} \text{PC1}: &\quad 1.80/4 = 0.450, \\ \text{PC1+PC2}: &\quad 0.450 + 1.10/4 = 0.450 + 0.275 = 0.725, \\ \text{PC1+PC2+PC3}: &\quad 0.725 + 0.70/4 = 0.725 + 0.175 = 0.900. \end{aligned}$$

We first cross 0.85 at PC3, so $\boxed{3}$ components are needed.

Solution *(1 %)* **(ii)** The score is the inner product $z_1^* = \phi_1^\top x^*$:

$$\begin{aligned} z_1^* &= 0.60 \cdot 1 + 0.50 \cdot (-1) + 0.40 \cdot 0.5 + 0.50 \cdot 0 \\ &= 0.60 - 0.50 + 0.20 + 0 \\ &= \boxed{0.30}. \end{aligned}$$

Grading: 1 P each. Accept 0.85 *vs* 0.90 *rounding for (i). For (ii) accept any answer in* $[0.29, 0.31]$. *The loadings as stated have* $\|\phi_1\|^2 = 0.36 + 0.25 + 0.16 + 0.25 = 1.02$ *(slightly off unit-norm rounding) — do not penalise students who notice.*

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

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

Solution *(2 %)* **(i) Assumptions.**

1. ε is **independent of the training set $\mathcal{D}$.** The noise on the test point is generated separately from the data used to fit $\hat{f}$, so ε and $\hat{f}(x_0)$ are independent random variables.
2. $\mathbb{E}[\varepsilon] = 0$. The noise is zero-mean.

What the outer expectation is over. The outer $\mathbb{E}[\,\cdot\,]$ in $\mathbb{E}[(y_0 - \hat{f}(x_0))^2]$ is taken jointly over (a) the random training sample $\mathcal{D}$ used to fit $\hat{f}$, and (b) the noise ε in the test response $y_0 = f(x_0) + \varepsilon$. The query point x_0 and the true function f are fixed (deterministic).

Solution *(4 %)* **(ii) Step 1 — write $y_0 - \hat{f}(x_0)$ as the sum of a noise piece and a $\mathcal{D}$-piece.**

$$y_0 - \hat{f}(x_0) = f(x_0) + \varepsilon - \hat{f}(x_0) = \varepsilon + \big(f(x_0) - \hat{f}(x_0)\big).$$

Step 2 — expand the square.

$$\big(y_0 - \hat{f}(x_0)\big)^2 = \varepsilon^2 + 2\varepsilon\big(f(x_0) - \hat{f}(x_0)\big) + \big(f(x_0) - \hat{f}(x_0)\big)^2.$$

Step 3 — take expectations, eliminate the cross-term. By independence, $\mathbb{E}[\varepsilon(f(x_0) - \hat{f}(x_0))] = \mathbb{E}[\varepsilon] \cdot \mathbb{E}[f(x_0) - \hat{f}(x_0)] = 0 \cdot (\cdots) = 0$. And $\mathbb{E}[\varepsilon^2] = \mathrm{Var}(\varepsilon) + (\mathbb{E}[\varepsilon])^2 = \sigma^2 + 0 = \sigma^2$. So

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

Step 4 — add and subtract $\mathbb{E}[\hat{f}(x_0)]$.

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

Square it:

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

Step 5 — take expectations. The first term $(f(x_0) - \mathbb{E}[\hat{f}(x_0)])^2$ is deterministic, hence equals itself. The cross-term has expectation zero because the deterministic factor $f(x_0) - \mathbb{E}[\hat{f}(x_0)]$ pulls out and the remaining $\mathbb{E}\big[\mathbb{E}[\hat{f}(x_0)] - \hat{f}(x_0)\big] = 0$. The last term is $\mathrm{Var}(\hat{f}(x_0))$ by definition.

Step 6 — identify and collect.

$$\begin{aligned}\mathbb{E}\big[(y_0 - \hat{f}(x_0))^2\big] &= \sigma^2 + \underbrace{(f(x_0) - \mathbb{E}[\hat{f}(x_0)])^2}_{\mathrm{Bias}^2[\hat{f}(x_0)]} + \underbrace{\mathrm{Var}(\hat{f}(x_0))}_{\text{variance}} \\ &= \boxed{\mathrm{Bias}^2[\hat{f}(x_0)] + \mathrm{Var}[\hat{f}(x_0)] + \sigma^2.} \ \square\end{aligned}$$

The three pieces are: **squared bias** (systematic gap between truth and the average prediction, error from wrong model class); **variance** (how much $\hat{f}(x_0)$ jitters across re-draws of the training set, error from chasing noise); **irreducible** σ^2 (test-point noise, independent of estimator choice). *Grading: 1 P for setting up the noise/learner split and writing out the cross-term; 1 P for vanishing the cross-term via $\mathbb{E}[\varepsilon] = 0$ + independence; 1 P for the add-and-subtract trick on $\mathbb{E}[\hat{f}(x_0)]$; 1 P for naming each of the three resulting terms. Deduct 0.5 if bias is written without the square.*

Solution *(1 %)* **(iii)** The decomposition is an *algebraic identity* that holds for every estimator $\hat{f}$; nothing in the derivation assumed any particular shape for how bias and variance depend on flexibility. The classical U-shape (variance up, bias down with flexibility, combining to a single minimum) and the double-descent shape (variance descends a second time past $p \approx n$ thanks to implicit minimum-norm regularization) are both compatible with the identity — they correspond to different shapes for $\mathrm{Bias}^2(p)$ and $\mathrm{Var}(p)$, not to violations of the formula.

Solution *(1 %)* **(iv)** If the variance saving from a small bias-inducing modification (ridge / lasso / dropout / boosting with appropriate M) is larger than the squared bias it introduces, total $\mathrm{MSE} = \mathrm{Bias}^2 + \mathrm{Var} + \sigma^2$ goes down even though OLS is unbiased — "trade some bias for more variance."

b) Pseudocode — k-fold and nested cross-validation *(4 %)*

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

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

1. Randomly partition $\{1, \ldots, n\}$ into k disjoint folds $F_1, \ldots, F_k$ of roughly equal size.

2. For each $j = 1, \ldots, k$:
   (a) Train $\hat{f}^{(-j)}$ on $\mathcal{D} \setminus F_j$ using model class $\mathcal{M}(\theta)$.
   (b) Compute the fold error $E_j = \frac{1}{|F_j|} \sum_{i \in F_j} L(y_i, \hat{f}^{(-j)}(x_i))$, where L is the chosen loss (squared error for regression, 0/1 for classification).
3. Return $\mathrm{CV}_k(\theta) = \frac{1}{k} \sum_{j=1}^{k} E_j$.

Selecting θ from a grid: run the procedure above at every $\theta \in \{\theta_1, \ldots, \theta_T\}$, pick $\hat{\theta} = \arg\min_t \mathrm{CV}_k(\theta_t)$, and refit the final model $\hat{f}$ on the *entire* training set $\mathcal{D}$ at $\theta = \hat{\theta}$. The CV score itself is a *biased-down* estimate of the test error of this final model and should not be reported as such (see (ii)).

Solution *(2 %)* **(ii) Nested CV.**

Outer loop: partition $\mathcal{D}$ into k_{out} folds $F_1^{\text{out}}, \ldots, F_{k_{\text{out}}}^{\text{out}}$.
For each $j = 1, \ldots, k_{\text{out}}$:

1. Let $\mathcal{D}_j^{\text{tr}} = \mathcal{D} \setminus F_j^{\text{out}}$ (outer training set), F_j^{out} = outer test fold.
2. **Inner loop** (on $\mathcal{D}_j^{\text{tr}}$ only): run ordinary k_{in}-fold CV across the hyperparameter grid $\{\theta_1, \ldots, \theta_T\}$ to choose $\hat{\theta}_j^* = \arg\min_t \mathrm{CV}_{k_{\text{in}}}(\theta_t; \mathcal{D}_j^{\text{tr}})$.
3. Refit $\mathcal{M}(\hat{\theta}_j^*)$ on all of $\mathcal{D}_j^{\text{tr}}$; predict on the held-out outer fold F_j^{out} and compute $E_j^{\text{out}} = \frac{1}{|F_j^{\text{out}}|} \sum_{i \in F_j^{\text{out}}} L(y_i, \hat{f}_j(x_i))$.

Return $\mathrm{CV}_{\text{nested}} = \frac{1}{k_{\text{out}}} \sum_j E_j^{\text{out}}$.

Why naive $\min_\theta \mathrm{CV}_k(\theta)$ is biased down: taking the minimum across a grid of T noisy CV estimates favours whichever θ_t happens to look best on *these* particular folds, which makes $\min_t \mathrm{CV}_k(\theta_t)$ underestimate the true test error of the corresponding model. Nested CV avoids this because the model selection (inner) and the performance assessment (outer) are run on disjoint partitions.
*Grading: 1 P each for clean k-fold and nested pseudocode (random partition, fit / evaluate / aggregate visible in each); deduct 0.5 if the refit-on-all-of-$\mathcal{D}$ step is missing from (i); deduct 0.5 if the "*min*-over-noisy-grid" bias explanation is not stated.*

c) AdaBoost by hand — one round *(4 %)*

Solution *(1 %)* **(i)** Misclassified set is $\{i = 2, i = 5\}$. With all weights equal to 0.2 and $\sum_i w_i^{(1)} = 1$:

$$\mathrm{err}_1 = \frac{0.20 + 0.20}{1.0} = \boxed{0.40}.$$

Solution *(1 %)* **(ii)**

$$\alpha_1 = \log\left(\frac{1 - 0.40}{0.40}\right) = \log\left(\frac{0.60}{0.40}\right) = \log(1.5) \approx \boxed{0.405}.$$

Solution *(2 %)* **(iii)** Pre-normalisation update factors: 1 for correctly-classified observations (multiplier $e^0 = 1$) and $e^{\alpha_1} = 1.5$ for the two misclassified observations.

i	$w_i^{(1)}$	multiplier	$\tilde{w}_i^{(2)} = w_i^{(1)} \cdot \text{mult}$	$w_i^{(2)} = \tilde{w}_i^{(2)} / Z$	
1	0.20	1	0.20	$0.20/1.20 \approx 0.167$	(correct)
2	0.20	1.5	0.30	$0.30/1.20 = 0.250$	(wrong)
3	0.20	1	0.20	$0.20/1.20 \approx 0.167$	(correct)
4	0.20	1	0.20	$0.20/1.20 \approx 0.167$	(correct)
5	0.20	1.5	0.30	$0.30/1.20 = 0.250$	(wrong)
			$Z = 1.20$	$\sum_i w_i^{(2)} = 1$	✓

So

$$\boxed{w_1^{(2)} \approx 0.167,\ \ w_2^{(2)} = 0.250,\ \ w_3^{(2)} \approx 0.167,\ \ w_4^{(2)} \approx 0.167,\ \ w_5^{(2)} = 0.250.}$$

Interpretation. Observations 2 and 5 — the ones G_1 got wrong — have their weights raised from 0.20 to 0.25, while the three correct ones drop from 0.20 to ≈ 0.167. This is the entire AdaBoost idea: the next weak learner G_2 is trained on a re-weighted distribution that emphasises exactly the points the previous ensemble struggled with.
Grading: 1 P for the un-normalised update with the right multipliers, 1 P for the normalisation step (must use $Z = 1.20$, not 1.0). Deduct 0.5 if any weight rounds wrong by more than 0.005.

Problem 4 (20 %) — Wine quality (regression)

a) OLS with a polynomial term and an interaction *(8 %)*

Solution *(1 %)* **(i)** Count the rows of the coefficient table: intercept, `alcohol`, $I($`alcohol`$^2)$, `volatile_acidity`, `sulphates`, `pH`, `density`, `residual_sugar`, `free_so2`, `total_so2`, `alcohol:volatile_aci` $= \boxed{11}$ parameters. Residual d.f. $= n_{\text{train}} - p = 800 - 11 = 789$, which matches the printout. ✓

Solution *(2 %)* **(ii)** The fingerprint is *strong collinearity* between `free_so2` and `total_so2`. The two pieces of information that together diagnose it: (1) the *pairwise correlation* 0.99 printed in the problem header, and (2) the *inflated standard errors* on the two affected coefficients (0.180 and 0.190, about 5× those of the other predictors). Either piece alone could have other explanations; together they are conclusive — $(\mathbf{X}^\top\mathbf{X})^{-1}$ has tiny eigenvalues in the near-degenerate `free_so2` / `total_so2` direction, blowing up $\text{Var}(\hat\beta_j)$ in that subspace and producing the large SE / individually insignificant pattern.
Grading: 1 P for identifying collinearity, 1 P for naming both diagnostic pieces (correlation and *inflated SE). Half credit for naming only one.*

Solution *(2 %)* **(iii) Baseline (everything at the mean = all standardized values 0):**

$$\widehat{\texttt{quality}} = \hat\beta_0 = \boxed{5.62}.$$

Alcohol moves $0 \to +1$, volatile_acidity stays at 0:

$$\begin{aligned}\widehat{\texttt{quality}} &= 5.62 + 0.420 \cdot 1 + (-0.110) \cdot 1^2 + (-0.140) \cdot 1 \cdot 0 \\ &= 5.62 + 0.420 - 0.110 + 0 \\ &= \boxed{5.93}.\end{aligned}$$

Change: +0.310.
Grading: 1 P for the baseline at the intercept, 1 P for the alcohol jump including the `alcohol`^{2} *term. Deduct 0.5 if the* `alcohol`^{2} *is forgotten (the typical slip that yields +0.420 instead of +0.310).*

Solution *(2 %)* **(iv)** Same calculation but with standardized `volatile_acidity` $= +1$. **Starting point (alcohol $= 0$, vol_acid $= +1$):**

$$\widehat{\texttt{quality}}_{\text{start}} = 5.62 + 0 + 0 + (-0.305) \cdot 1 + 0 = 5.315.$$

End point (alcohol $= +1$, vol_acid $= +1$):

$$\begin{aligned}\widehat{\texttt{quality}}_{\text{end}} &= 5.62 + 0.420 \cdot 1 + (-0.110) \cdot 1^2 + (-0.305) \cdot 1 + (-0.140) \cdot 1 \cdot 1 \\ &= 5.62 + 0.420 - 0.110 - 0.305 - 0.140 \\ &= 5.485.\end{aligned}$$

Change in predicted quality: $5.485 - 5.315 = \boxed{+0.170}$, compared to +0.310 in part (iii).

Interpretation. The negative interaction $\hat{\beta}_{\texttt{alcohol:volatile\_acidity}} = -0.140$ says that the positive effect of `alcohol` on `quality` is *dampened* when `volatile_acidity` is high: in a wine that is already volatile-acidic, raising alcohol gives less of a quality boost (here, only +0.17 versus +0.31 at average acidity). Plausibly, volatile acidity overpowers other flavour cues.
Grading: 1 P for the change +0.170, 1 P for the interpretation. Deduct 0.5 if the interaction sign is read as "alcohol effect at high vol_acid" rather than "effect on the slope."

Solution *(1 %)* **(v)** $p = 0.334$ means we lack evidence that the `density` coefficient differs from zero *conditional on the other predictors in the model*; that is not the same as "density has no effect." Density is collinear with other chemistry variables (alcohol, residual sugar), so the marginal effect of density may be absorbed by them. "Not significant" is not "zero"; deciding to drop it should rest on a CV or AIC comparison, not the p-value alone.

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

Solution *(2 %)* **(i)** With `free_so2` and `total_so2` both in the model, the two columns of $\mathbf{X}$ are nearly proportional (cor = 0.99), so $\mathbf{X}^\top\mathbf{X}$ has a tiny eigenvalue in the corresponding direction and $(\mathbf{X}^\top\mathbf{X})^{-1}$ has a huge entry. This inflates both $\mathrm{Var}(\hat{\beta}_{\texttt{free\_so2}})$ and the magnitude of $\hat{\beta}_{\texttt{free\_so2}}$ itself (the estimate compensates for whatever the partner column does), so the coefficient can swing wildly across resamples and across choices of which collinear partner to include. Dropping `total_so2` removes the near-degeneracy, the corresponding eigenvalue recovers, the SE shrinks by roughly 4×, and $\hat{\beta}_{\texttt{free\_so2}}$ snaps to a value of much smaller magnitude.

Solution *(1 %)* **(ii)** Training and test MSE essentially unchanged between the full and reduced models tells us the collinearity was *not hurting prediction*: as discussed in the collinearity item of P2(e), the fitted hyperplane is well-determined in the direction the data span, even when the individual coefficient allocation is unstable. The cost of collinearity here was entirely in *interpretation* (which of the two SO_2 channels carries the effect), not in $\hat{y}$.

Solution *(1 %)* **(iii)** Combining is preferable to dropping when the two collinear predictors *both carry partial signal* and the analyst has a substantive reason to expect their combination to be a meaningful quantity (e.g. summing `free_so2` and the bound-SO_2 implied by `total_so2` − `free_so2` produces an interpretable "effective antimicrobial reservoir"). Pure dropping throws away whatever extra information the dropped column carried beyond the kept one.

c) Ridge regression and the one-SE rule *(5 %)*

Solution *(1 %)* **(i)**

$$\hat{\beta}^R_\lambda = \arg\min_{\beta_0,\beta} \sum_{i=1}^{n}(y_i - \beta_0 - x_i^\top\beta)^2 \;+\; \lambda\sum_{j=1}^{p}\beta_j^2, \qquad \lambda \geq 0.$$

The intercept β_0 is *not* penalised — the penalty sum runs over $j = 1, \ldots, p$ only. **Why standardise:** the ridge penalty is scale-dependent — a predictor measured in large units has small coefficients and is barely penalised, while a predictor in small units has large coefficients and is heavily penalised. Standardising all predictors to mean 0 and variance 1 ensures the penalty acts on every predictor on the same footing, so shrinkage is determined by signal strength rather than units.

Solution *(2 %)* **(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 $\mathrm{SE}^\star$,

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

Bias–variance reason ridge beats OLS here. OLS is unbiased but high-variance in directions of collinearity (the `free_so2` / `total_so2` subspace). Ridge introduces a small bias (shrinking those coefficients toward zero / toward each other) in exchange for a substantial variance reduction in that same subspace. Because the variance saving exceeds the squared bias introduced, total test MSE drops.
Grading: 1 P for stating the rule precisely (mention both the within-one-SE and the "simplest/largest λ" parts); 1 P for a defensible bias–variance argument tied to collinearity. Half credit if the rule is stated correctly but the bias–variance argument is just "ridge has less variance" without saying why on this *data set.*

Solution *(1 %)* **(iii) Lasso at $\hat{\lambda}^{\text{lasso}}_{\text{1SE}}$.** Its 6 nonzero coefficients (vs. 11 for ridge / OLS) yield a sparser, more interpretable model at a test-MSE cost of only $0.439 - 0.430 = 0.009$. The interpretability gain — five fewer predictors to defend — justifies the small predictive sacrifice.

Solution *(1 %)* **(iv)** The lasso's L1 geometry has corners at the coordinate axes; shrinkage along the near-degenerate `free_so2` / `total_so2` direction therefore typically lands on one of those axes, zeroing out one collinear partner exactly. Ridge's L2 geometry is a smooth sphere with no corners — it shrinks the two coefficients toward each other (distributing the joint effect roughly equally) but never zeroes either exactly.

d) Gradient boosting for comparison *(3 %)*

Solution *(1 %)* **(i)** Boosting 0.382 < ridge 0.430 < OLS 0.453. The gap between boosting and the linear methods (a $\sim 12\%$ MSE reduction) suggests substantive *nonlinearity and / or higher-order interactions* among the predictors that a linear-in-features model cannot capture — the part (a) significant $\texttt{alcohol}^2$ term and the `alcohol:volatile_acidity` interaction are the linear model's partial attempts to absorb the same structure that boosting captures organically.

Solution *(1 %)* **(ii)** Boosting is *sequential*: each tree is fit to the residuals of the current ensemble. With far too many trees the residuals become pure noise, and additional trees fit it — the ensemble overfits and test MSE *rises*. A random forest, by contrast, averages B independently bootstrap-trained trees, and averaging only reduces variance, so test MSE is monotonically non-increasing in B — "too many trees" is harmless.

Solution *(1 %)* **(iii) Plot:** *partial-dependence plot* for predictor X_j — shows how the model's predicted $\hat{y}$ varies as X_j sweeps its range, marginalising the others over their empirical distribution. *Answers:* the typical shape (rough monotonicity, nonlinearity, threshold effects) of X_j's effect. *Does not answer:* causality ("raising X_j would cause higher quality" is an unwarranted leap from a regression curve), nor how X_j interacts with other predictors. **Summary statistic:** *permutation variable-importance* for X_j — the increase in test loss when X_j's values are randomly shuffled. *Answers:* how much the model's predictive power depends on X_j. *Does not answer:* the *direction* of the effect (only magnitude).

Problem 5 (26 %) — Customer churn (classification)

a) Logistic regression with an interaction *(10 %)*

Solution *(2 %)* **(i) Encoding assumption.** The categorical `contract` is dummy-coded against *month-to-month* (reference), so on a month-to-month contract both `contract_1year` and `contract_2year` are 0 and the interaction terms `tenure:contract_1year` and `tenure:contract_2year` also vanish. On a 1-year contract `contract_1year` = 1 and `tenure:contract_1year` = `tenure`; on a 2-year contract analogously.

A one-month increase in `tenure`, holding everything else fixed, changes the linear predictor by

$$\Delta\hat{\eta} = \hat{\beta}_{\texttt{tenure}} + \hat{\beta}_{\texttt{tenure:contract\_1year}} \cdot \mathbb{1}[\text{1-year}] + \hat{\beta}_{\texttt{tenure:contract\_2year}} \cdot \mathbb{1}[\text{2-year}],$$

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

(a) Month-to-month: $\exp(-0.060) \approx \boxed{0.942}$,

(b) 1-year: $\exp(-0.060 + 0.025) = \exp(-0.035) \approx \boxed{0.966}$,

(c) 2-year: $\exp(-0.060 + 0.040) = \exp(-0.020) \approx \boxed{0.980}$.

Grading: $\frac{2}{3}$ *P each, all-or-nothing for each contract level. Half credit for (b) and (c) if the interaction term is forgotten (the standard interaction trap from P2(d) and the wine problem). Accept rounding* ± 0.002.

Solution *(1 %)* **(ii)** On a long-term contract the customer has already committed; an additional month of tenure shortens the remaining lock-in proportionally less than it would on a month-to-month plan. Practically: long-contract customers' churn risk is dominated by the lock-in itself, not by their accumulated tenure, so tenure's marginal effect is dampened (positive interaction). The base effect of one more tenure month is still negative on all three contract types, just smaller in magnitude on 2-year (-0.020) than on month-to-month (-0.060).

Solution *(3 %)* **(iii)** Customer profile: `tenure` $= 12$, `monthly_charges` $= 85$, `senior` $= 0$, month-to-month (so all contract dummies and their interactions $= 0$), `tech_support` $= 0$, `online_security` $= 0$.

$$\begin{aligned}\hat{\eta} &= -0.40 + (-0.060) \cdot 12 + 0.018 \cdot 85 + 0.40 \cdot 0 + 0 + 0 + 0 + 0 + 0 \\ &= -0.40 - 0.72 + 1.53 \\ &= \boxed{0.41}.\end{aligned}$$

Sigmoid step:

$$\hat{p} = \sigma(0.41) = \frac{1}{1 + e^{-0.41}} = \frac{1}{1 + 0.6637} = \frac{1}{1.6637} \approx \boxed{0.601}.$$

Grading: 1 P each for the per-term contributions / correctly assembled $\hat{\eta}$*, the value* $\hat{\eta} \approx 0.41$*, and the sigmoid step / final* $\hat{p} \approx 0.60$*. Accept any answer in* $[0.595, 0.605]$.

Solution *(2 %)* **(iv)** Same customer but now on a 2-year contract: `contract_2year` $= 1$, `tenure:contract_2year` $= 12$.

The two extra contributions to $\hat{\eta}$ are:

$$\hat{\beta}_{\texttt{contract\_2year}} \cdot 1 + \hat{\beta}_{\texttt{tenure:contract\_2year}} \cdot 12 = -2.00 + 0.040 \cdot 12 = -2.00 + 0.48 = -1.52.$$

So

$$\hat{\eta}_{\text{2-yr}} = 0.41 + (-1.52) = \boxed{-1.11}, \qquad \hat{p}_{\text{2-yr}} = \sigma(-1.11) = \frac{1}{1 + e^{1.11}} \approx \frac{1}{4.034} \approx \boxed{0.248}.$$

Marketing takeaway. Moving the same customer from month-to-month to a 2-year contract drops the predicted churn probability from 0.601 to 0.248 — more than halving it. That strongly motivates lock-in incentives (e.g. discounts contingent on signing a 2-year contract).
Grading: 1 P for the corrected $\hat{\eta} = -1.11$ *(must include both* `contract_2year` *and the interaction), 1 P for* $\hat{p} \approx 0.25$ *and a one-sentence business interpretation. Deduct 0.5 if the interaction is forgotten (yields* $\hat{\eta} = -1.59$*,* $\hat{p} \approx 0.17$ *— the standard trap).*

Solution *(2 %)* **(v)** Logistic regression on observational data is a model of *conditional association*, not causation — the prof's recurrent "fancy correlations, not causal" point. The positive

$\hat{\beta}_{\texttt{senior}} = 0.40$ tells us that seniors have higher churn odds *at the same* (tenure, monthly_charges, contract, etc.) profile observed in the data, but seniors might churn more for confounding reasons that are correlated with `senior` and not adjusted for here (e.g. income shocks, life events, preference for landline competitors). Inferring causation would require a randomised intervention on `senior` (impossible) or rich enough confounder adjustment to plausibly close every backdoor path — neither is offered by the table.

b) LDA vs. logistic regression on the same data *(5 %)*

Solution *(2 %)* **(i) LDA.** TP = 180, FN = 240, FP = 130, TN = 950; 420 true churns, 1080 true non-churns, $n = 1500$.

$$\text{Sensitivity}_{\text{LDA}} = \frac{180}{420} \approx \boxed{0.43},$$
$$\text{Specificity}_{\text{LDA}} = \frac{950}{1080} \approx \boxed{0.88},$$
$$\text{Error rate}_{\text{LDA}} = \frac{130 + 240}{1500} = \frac{370}{1500} \approx \boxed{0.25}.$$

Logistic (threshold 0.5). TP = 210, FN = 210, FP = 135, TN = 945.

$$\text{Sensitivity}_{\text{logit}} = \frac{210}{420} = \boxed{0.50},$$
$$\text{Specificity}_{\text{logit}} = \frac{945}{1080} \approx \boxed{0.88},$$
$$\text{Error rate}_{\text{logit}} = \frac{135 + 210}{1500} = \frac{345}{1500} = \boxed{0.23}.$$

Grading: 1 P per method for the (sens, spec, error) triple — all three correct to two decimals.

Solution *(1 %)* **(ii) Modelling difference.** LDA assumes the class-conditional predictor distributions are *multivariate Gaussian* with a *shared covariance matrix*; logistic regression only assumes the log-odds are *linear* in X, with no distributional constraint on X. With a categorical variable like `contract` (dummy-coded 0/1) in the input vector, LDA's Gaussian assumption is the more obviously violated — a Bernoulli component of X cannot be normally distributed. Logistic regression can fold 0/1 dummies in without any distributional contradiction.

Solution *(2 %)* **(iii) When QDA wins.** If the two classes really do have substantially different covariance structures (e.g. churners have wider spread in `monthly_charges` or correlations between predictors that non-churners do not), the LDA shared-covariance assumption is violated and QDA's class-specific covariances buy bias reduction. This is most likely when the boundary is genuinely curved in the predictor space.

When QDA loses. When per-class sample sizes are small (here the churn class has only ≈ 980 training observations) and / or predictors are mostly binary, QDA's $K \cdot p(p+1)/2$ extra covariance parameters are estimated noisily; the variance penalty exceeds the bias saving and QDA test error rises above LDA's. This is the regime the present problem is in — six predictors, four of which are binary, ~ 980 churners — so I'd expect QDA to do *worse* than LDA here.

c) A boosting classifier *(8 %)*

Solution *(2 %)* **(i) Hyperparameter justifications.**

- $\boldsymbol{M = 800}$ (number of trees) was chosen by 10-fold CV. Unlike random forests, boosting overfits if M is too large (the ensemble eventually starts fitting noise in the residuals); CV finds the bias–variance sweet spot.

- $\boldsymbol{d = 3}$ (interaction depth) lets each tree capture interactions up to order 3, which is plausible for a churn problem with known interactions like `tenure`×`contract`. Larger d raises per-tree variance; smaller d forces additive structure. $d \in \{2, 3, 4\}$ is the typical default.
- $\boldsymbol{\nu = 0.05}$ (shrinkage / learning rate) is small enough that each tree corrects only a slice of the current residuals, so the ensemble's bias falls slowly but smoothly. Smaller ν generalises better at the cost of requiring more trees — the well-known (ν, M) joint trade-off: halving ν roughly doubles required M.

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

- **AdaBoost.** Each new weak classifier G_m is fit on data re-weighted to emphasise the points the current ensemble misclassifies; its ± 1 vote is weighted by $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$ in the final sign-vote ensemble.
- **Gradient boosting.** Each new tree $\hat{T}_m$ is fit to the *negative gradient* of the loss L with respect to the current ensemble's predictions; under squared loss this is the residual, under log-loss it is the pseudo-residual; the tree contribution is then added with shrinkage ν to the ensemble.

AdaBoost as a special case. AdaBoost is gradient boosting on the *exponential loss* $L(y, F) = \exp(-yF)$ with classification trees as base learners — the re-weighting $w_i^{(m+1)} \propto w_i^{(m)} e^{\alpha_m \mathbb{1}[\cdot]}$ is exactly the discrete update that falls out of forward stagewise minimisation of exponential loss.

Solution *(2 %)* **(iii)** TP = 260, FN = 160, FP = 145, TN = 935.

$$\text{Sensitivity}_{\text{GB}} = \frac{260}{420} \approx \boxed{0.62},$$
$$\text{Specificity}_{\text{GB}} = \frac{935}{1080} \approx \boxed{0.87},$$
$$\text{Error rate}_{\text{GB}} = \frac{145 + 160}{1500} = \frac{305}{1500} \approx \boxed{0.20}.$$

Recommendation. Boosting wins on every metric — in particular, its sensitivity 0.62 vs. 0.50 for logistic regression at the same threshold means it catches 50 more true churners per 1,500 test customers, at the cost of only 10 extra false alarms. If a missed churner costs much more than a false alarm, deploy the boosting model.
Grading: 1 P for the (sens, spec, error) triple, 1 P for the recommendation with a defensible cost-asymmetry argument. Accept rounding ± 0.005.

Solution *(2 %)* **(iv) (a) What permutation variable importance is.** For each predictor X_j, randomly permute its values among the OOB (or held-out test) observations, push them through the trained ensemble, and report the increase in loss relative to the baseline. Larger increase $\Rightarrow$ the model's predictive accuracy depended more on the genuine X_j signal.

(b) What it cannot answer that a logistic-regression coefficient can. It does *not* tell you the *direction* of the effect — whether longer `tenure` *lowers* or *raises* churn probability. A logistic-regression coefficient $\hat{\beta}_{\texttt{tenure}} = -0.060$ tells you both (a) tenure matters and (b) more tenure means lower churn, with a quantified odds multiplier per month. Permutation importance only ranks the former; for the latter you need partial-dependence or ICE plots on top.

d) Class imbalance and the metric debate *(3 %)*

Solution *(1 %)* **(i)** A trivial "always no churn" classifier misclassifies exactly the 420 true churners out of 1500:

$$\text{error}_{\text{trivial}} = \frac{420}{1500} = \boxed{0.28}.$$

This matches the population churn rate (the data are class-imbalanced at roughly 28% churners / 72% non-churners, which is also reflected in the test sample). Notice that the trivial classifier's 0.28 is *worse* than every classifier in (b)–(c) here — but only marginally so for LDA at 0.25. With a much rarer event (say 5% churn) the trivial classifier can easily beat well-tuned models on accuracy, which is exactly the trap accuracy hides.

Solution *(2 %)* **(ii) Accuracy is the wrong criterion.** With imbalanced classes and asymmetric costs (a missed churner is much more expensive than a false alarm), accuracy collapses the cost matrix into a single scalar and hides the trade-off the business actually cares about.

(a) Better metrics. The pair (**sensitivity, specificity**) — separately measuring how well the model catches churners and how well it leaves loyal customers alone — or a single threshold-independent summary like **AUC** (or precision-recall AUC, which is more sensitive under imbalance).

(b) Which decision it makes easier. Comparing two classifiers with nearly identical accuracy but very different false-negative rates: under accuracy they look tied; under (sensitivity, specificity) the one with higher sensitivity is the clear pick when missing a churner is the costlier error — which directly encodes the marketing team's cost asymmetry.

End of solution proposal. Total awarded: $10 + 28 + 16 + 20 + 26 = 100$ points.