# TMA4268 Statistical Learning V2026

Mock Exam 4 (solutions)

Compiled for Anders Bekkevard

Companion to `mock-exam-4.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-4.tex` for the problem statements.*

---

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

**Solution** *(1 P per blank.)*

(1) **collinearity**

(2) **ridge regression**

(3) **$k$-fold cross-validation**

(4) **nested cross-validation**

(5) **mini-batch SGD**

(6) **backpropagation**

(7) **dropout**

(8) **early stopping**

(9) **label smoothing**

(10) **gradient boosting**

*Grading: 1 P per correct blank. No partial credit for "close" alternatives (e.g. "the lasso" for (2), where the passage explicitly says "a quadratic penalty").*

---

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

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

**Solution** *(1 P per statement.)*

(i) **True**. The bias–variance decomposition is an algebraic identity that follows from adding/subtracting $\mathbb{E}[\hat{f}(x_0)]$ inside the squared expectation; it does not require $\hat{f}$ to be the squared-error minimiser.

(ii) **False**. The decomposition continues to hold in the over-parameterised regime; what "double descent" shows is that the *variance* term can decrease again past the interpolation threshold, not that the identity itself breaks.

(iii) **False**. This is the canonical reason the prof calls the result a *decomposition* rather than a *trade-off*. Regularisation, more data, or structural changes can reduce variance *without* a matching increase in bias.

### b) Cross-validation and the wrong-way CV trap *(4 %)*

**Solution** *(1 P per statement.)*

(i) **True**. With $k = 10$ the model is fit exactly 10 times (once per held-out fold). LOOCV is the $k = n$ special case.

(ii) **False**. This is the classic *wrong-way CV* trap (ISLR §5.3 / lecture 6). Pre-selecting predictors on the full data leaks $y$-information into every fold, so CV underestimates the test error. The selection step must move *inside* each fold.

(iii) **True**. This is exactly the structural definition of nested CV: inner loop selects, outer loop assesses the procedure that includes the selection step.

(iv) **True**. PRESS identity: LOOCV $= \frac{1}{n}\sum_i \left(\frac{y_i - \hat{y}_i}{1 - h_{ii}}\right)^2$, so OLS-LOOCV needs only one full fit plus the diagonal of the hat matrix.

### c) Neural network parameters and forward pass *(3 %)*

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

$$\begin{aligned} \text{input}(4) \to \text{hidden}(5): &\quad 4 \cdot 5 + 5 = 25, \\ \text{hidden}(5) \to \text{hidden}(3): &\quad 5 \cdot 3 + 3 = 18, \\ \text{hidden}(3) \to \text{output}(1): &\quad 3 \cdot 1 + 1 = 4, \\ \text{Total:} &\quad \boxed{25 + 18 + 4 = 47} \text{ parameters.} \end{aligned}$$

**Solution** *(2 %)* **(ii)** Pre-activation:

$$\begin{aligned} z \;=\; b + \sum_{j=1}^{4} w_j x_j \;&=\; -1.5 + 1.0 \cdot 1 + (-0.5) \cdot 2 + 2.0 \cdot (-1) + 0 \cdot 4 \\ &=\; -1.5 + 1 - 1 - 2 + 0 \;=\; -3.5. \end{aligned}$$

ReLU: $\max(0, -3.5) = \boxed{0}$.
*Grading: 1 P for $z = -3.5$, 1 P for the ReLU step. Common slip: forgetting the sign on the bias (gives $z = -0.5$, still ReLU $= 0$); accept full credit only if the bias term is correctly placed.*

### d) Backprop and mini-batch SGD *(3 %)*

**Solution** *(0.75 P per statement.)*

(i) **False**. Backpropagation is the *algorithm that computes the gradient* (via the chain rule). The optimiser that uses that gradient to update weights is (mini-batch) SGD.

(ii) **True**. A uniformly drawn mini-batch gives an unbiased estimator of the full-data gradient: $\mathbb{E}_{\text{batch}}[\nabla L_{\text{batch}}] = \nabla L_{\text{full}}$.

(iii) **True**. Larger batches have lower-variance gradient estimates; the gradient noise from small batches is itself a (mild) regulariser, so reducing it reduces that regularisation.

(iv) **False**. With $L = \frac{1}{2}(y_i - \hat{f}(x_i))^2$, $\partial L/\partial \hat{f} = -(y_i - \hat{f}(x_i))$, so the backward-pass seed is the *negative* residual, $\delta_i^{\text{out}} = -(y_i - \hat{f}(x_i))$. The sign matters: backprop pushes the gradient of the loss, not the residual itself.

### e) Neural-network regularization *(3 %)*

**Solution** *(0.75 P per statement.)*

(i) **False**. Dropout is active *only* at training time. At test time all units are used, with outputs scaled appropriately (or weights scaled, in "inverted dropout").

(ii) **True**. The prof's typical default is $\sim 20\%$ dropout for fully-connected hidden layers on tabular data; she explicitly flagged 50% as too aggressive on signal-poor tabular problems (it tends to under-fit). The statement just records both facts together.

(iii) **False**. Early stopping monitors the *validation* loss and returns the parameters from the epoch that minimised it. Stopping when training loss plateaus would defeat the purpose.

(iv) **True**. Label smoothing replaces one-hot targets with softened ones; the prof explicitly motivated it by label noise in the training data.

### f) Boosting flavors *(4 %)*

**Solution** *(1 P per statement.)*

(i) **True**. AdaBoost up-weights misclassified observations each round via $w_i \leftarrow w_i e^{\alpha_m \mathbb{1}[y_i \neq G_m(x_i)]}$.

(ii) **True**. For squared-error loss $L = \frac{1}{2}(y - f)^2$, the negative gradient w.r.t. $f$ is $-(\partial L/\partial f) = y - f =$ residual. Fitting the next tree to residuals is therefore gradient descent in function space.

(iii) **False**. The relationship is the opposite: smaller $\nu$ means each tree contributes *less*, so *more* trees are needed to reach the same fit. (Roughly: halving $\nu$ doubles the required $M$.)

(iv) **True**. XGBoost's distinguishing features are (a) second-order Taylor expansion of the loss (uses both gradient and Hessian), and (b) explicit $L_1/L_2$ regularisation on the leaf weights, in addition to engineering optimisations.

### g) Collinearity *(3 %)*

**Solution** *(1 P per statement.)*

(i) **True**. With near-singular $\mathbf{X}^\top\mathbf{X}$, the diagonal entries of $\sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$ blow up, inflating standard errors. The joint $F$-test on the collinear bloc can still be highly significant.

(ii) **False**. Collinearity inflates the variance of individual *coefficient estimates*, but the *prediction* $\hat{y} = \mathbf{X}\hat{\boldsymbol{\beta}}$ can still be accurate (the unstable directions are exactly the ones that don't affect the fitted values much).

(iii) **False**. *Perfect* collinearity (`age_decades` = `age`/10) makes $\mathbf{X}^\top\mathbf{X}$ exactly singular: the OLS estimates are *not defined* at all (no unique solution), so it is not the case that the estimates exist but with infinite SEs. Most software silently drops one of the two columns.

### h) Principal component analysis *(3 %)*

**Solution** *(1 %)* **(i)** For standardised variables each has variance 1, so the total variance equals the number of variables: $\sum_j \lambda_j = p = 7$ ($2.5+1.5+1.1+0.8+0.5+0.4+0.2 = 7$, ✓). Cumulative proportion of explained variance:

$$\begin{aligned} \text{PC1: } & 2.5/7 = 0.357, \\ \text{PC1–2: } & 4.0/7 = 0.571, \\ \text{PC1–3: } & 5.1/7 = 0.729, \\ \text{PC1–4: } & 5.9/7 = 0.843 \ \geq \ 0.80. \ \checkmark \end{aligned}$$

So $\boxed{4}$ principal components are needed to reach at least 80%.

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

$$\begin{aligned} z_1^* &= 0.50 \cdot 2 + (-0.40) \cdot 0 + 0.45 \cdot 1 + 0.30 \cdot (-1) + (-0.35) \cdot 0 + 0.40 \cdot 1 + 0.15 \cdot 2 \\ &= 1.00 + 0 + 0.45 - 0.30 + 0 + 0.40 + 0.30 \ = \ \boxed{1.85}. \end{aligned}$$

**Solution** *(1 %)* **(iii) False**. PCA is performed on the (centered) *covariance matrix* of the predictors, which is sensitive to scale. Without standardisation, variables measured on larger scales dominate the first PC; standardising removes this scale dependence and is the default whenever the variables are on different scales.

### i) Hierarchical clustering and $K$-means *(2 %)*

**Solution** *(1 P per statement.)*

(i) **True**. At the first merge each observation is its own cluster, so all three linkages (single / complete / average) reduce to "find the smallest off-diagonal entry of $D$." The linkage rule only kicks in from the second merge onward (when at least one cluster contains more than one point).

(ii) **True**. The standard $K$-means algorithm (Lloyd's algorithm) is only guaranteed to converge to a *local* minimum of the within-cluster sum of squares; different random initialisations can yield different solutions. The recommended practical fix is exactly what the statement says: run the algorithm from several random starts and keep the one with the smallest within-cluster SS.

---

## Problem 3 (16 %) — Theory and hand calculations

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

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

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

The cross term *vanishes*: $\varepsilon$ is independent of the training set $\mathcal{D}$ (hence of $\hat{f}$) and of the fixed $f(x_0)$, and $\mathbb{E}[\varepsilon] = 0$, so

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

And $\mathbb{E}[\varepsilon^2] = \mathrm{Var}(\varepsilon) = \sigma^2$ since $\mathbb{E}[\varepsilon] = 0$. Therefore

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

*Grading: 1 P for the expansion; 1 P for explicitly killing the cross term using independence + zero mean. Deduct 0.5 if the cross-term is dropped without justification.*

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

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

The first term is constant (pulled out of the expectation since $f(x_0)$ is fixed and $\mathbb{E}[\hat{f}(x_0)]$ is a constant). The middle (cross) term vanishes:

$$\mathbb{E}\Big[\mathbb{E}[\hat{f}(x_0)] - \hat{f}(x_0)\Big] = \mathbb{E}[\hat{f}(x_0)] - \mathbb{E}[\hat{f}(x_0)] = 0.$$

The third term is exactly $\mathrm{Var}(\hat{f}(x_0))$ by definition. Hence

$$\boxed{\mathbb{E}\Big[(f(x_0) - \hat{f}(x_0))^2\Big] = \underbrace{\big(f(x_0) - \mathbb{E}[\hat{f}(x_0)]\big)^2}_{\mathrm{Bias}^2[\hat{f}(x_0)]} + \underbrace{\mathrm{Var}(\hat{f}(x_0))}_{\text{variance}}.}$$

*Grading: 1 P for the add-subtract trick; 1 P for vanishing of the cross term; 1 P for identifying the two surviving terms as* Bias$^2$ *and* Var*.*

**Solution** *(1 %)* **(iii)** Combining (i) and (ii),

$$\mathbb{E}\Big[(y_0 - \hat{f}(x_0))^2\Big] = \underbrace{\big(f(x_0) - \mathbb{E}[\hat{f}(x_0)]\big)^2}_{\mathrm{Bias}^2} + \underbrace{\mathrm{Var}(\hat{f}(x_0))}_{\text{variance}} + \underbrace{\sigma^2}_{\text{irreducible}}.$$

$\sigma^2$ is the *irreducible* error (the variance of the noise $\varepsilon$). The inner $\mathbb{E}[\hat{f}(x_0)]$ and $\mathrm{Var}[\hat{f}(x_0)]$ are taken over the random training set $\mathcal{D}$; the outer $\mathbb{E}[\cdot]$ above is jointly over $\mathcal{D}$ and over the test-point noise $\varepsilon$.

**Solution** *(1 %)* **(iv)** "Decomposition" is preferred because the identity is an exact algebraic equality, not a constraint that forces bias to grow when variance shrinks. Regularisers such as ridge / lasso / dropout can reduce variance *without* a matching increase in bias, and in the over-parameterised / double-descent regime variance can even decrease further past the interpolation threshold.

**Solution** *(1 %)* **(v)** At a small positive $\lambda$, lasso introduces a little *bias* (shrinkage toward zero) but reduces the *variance* of the fit; the irreducible $\sigma^2$ is unchanged. The net is that variance falls more than bias$^2$ rises, so test MSE drops — exactly the bias–variance lever the penalty is designed to pull.

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

**Solution** *(3 %)* **(i)**

```
INPUT: dataset (X, y) of size n;
       hyperparameter grid Lambda = {lambda_1, ..., lambda_L};
       K_outer = 5, K_inner = 5.

partition (X, y) into K_outer outer folds F_1, ..., F_{K_outer}

FOR k = 1 to K_outer:                                    # OUTER (assessment) loop
    test_k       = F_k                                   # held out for HONEST assessment
    trainval_k   = union of all other outer folds

    partition trainval_k into K_inner inner folds G_1, ..., G_{K_inner}

    FOR each lambda in Lambda:                           # HYPERPARAMETER selection
        FOR j = 1 to K_inner:                            # INNER (selection) loop
            val_j   = G_j
            train_j = trainval_k minus G_j
            fit model_{lambda, j} on train_j             # data used to FIT
            err_{lambda, j} = error(model_{lambda, j}, val_j)
        end
        innerCV(lambda) = mean over j of err_{lambda, j}
    end

    lambda_k* = argmin_lambda innerCV(lambda)            # selection step
    refit model_k on ALL of trainval_k at lambda_k*       # refit on full outer-train fold
    score_k    = error(model_k, test_k)                  # OUTER assessment on test_k

end

test_error_estimate = mean over k of score_k             # aggregate outer scores
RETURN test_error_estimate
```

The structure exposes the three data uses cleanly: *fit* = `train_j`; *select* = `val_j` (through the inner-CV average); *assess* = `test_k` (never touched until selection is complete on `trainval_k`). *Grading: 1 P for the correct nesting (outer loop wraps the inner CV); 1 P for labelling the three data roles (fit / select / assess); 1 P for aggregating outer-fold scores into the final estimate. Deduct 1 if $\lambda$ is selected once globally rather than once per outer fold.*

**Solution** *(1 %)* **(ii)** The shortcut reuses the same data both for *choosing* $\lambda$ and for *reporting* the test error. The argmin of CV error is itself a noisy, optimistic statistic, so the reported error is biased *downward* (a "winner's curse" / selection bias) — the very phenomenon nested CV is designed to correct.

## c) Bootstrap standard error *(4 %)*

**Solution** *(2 %)* **(i)**

```
INPUT: i.i.d. sample x = (x_1, ..., x_n); statistic theta_hat = median(x);
       number of bootstrap replicates B (e.g. B = 1000).

FOR b = 1 to B:
    draw x*_b = (x*_{b,1}, ..., x*_{b,n}) by sampling n times WITH REPLACEMENT
                                                  from (x_1, ..., x_n)
    compute   theta*_b = median(x*_b)
end

theta_bar = (1/B) * sum_{b=1}^B theta*_b

SE_boot   = sqrt( (1/(B-1)) * sum_{b=1}^B (theta*_b - theta_bar)^2 )
```

```
RETURN SE_boot
```

The three required ingredients are explicit: (a) each bootstrap sample is drawn *with replacement* from the original sample at size $n$; (b) the recomputed statistic on each bootstrap sample is the sample median; (c) the final SE is the sample standard deviation of the $B$ bootstrap medians (divisor $B-1$).

**Solution** *(1 %)* **(ii)** For one bootstrap sample of size $n$, the probability that observation $X_1$ is *not* chosen on any given draw is $(1-1/n)^n$, so

$$P(X_1 \in \text{bootstrap sample}) = 1 - \left(1 - \tfrac{1}{n}\right)^n .$$

At $n = 6$:

$$P = 1 - (5/6)^6 = 1 - 0.33490 = \boxed{0.665} \text{ (three decimals).}$$

As $n \to \infty$, $(1-1/n)^n \to 1/e$, so

$$P(X_1 \in \text{boot}) \;\longrightarrow\; 1 - 1/e \;\approx\; \boxed{0.632}.$$

**Solution** *(1 %)* **(iii) Percentile bootstrap CI.** Compute the $B$ bootstrap replicates $\hat\theta^*_1, \ldots, \hat\theta^*_B$ as in (i); the 95% percentile CI is the interval $[\hat\theta^*_{(2.5\%)},\ \hat\theta^*_{(97.5\%)}]$, i.e. the 2.5th and 97.5th empirical quantiles of the bootstrap distribution.

---

# Problem 4 (22 %) — Apartment-rent regression

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

**Solution** *(1 %)* **(i)** Count the rows of the coefficient table: intercept, `area`, `rooms`, `area_per_room`, `age`, $I(\texttt{age}^2)$, `floor`, `distance`, `has_balcony`, `type_studio`, `type_loft`, `distance:has_balcony` $= \boxed{12}$ parameters. Consistency check: residual d.f. $= n_{\text{train}} - p = 500 - 12 = 488$, matching the printout. ✓

**Solution** *(2 %)* **(ii)** The triple (`area`, `rooms`, `area_per_room`) satisfies `area` = `rooms`·`area_per_room`, and even after standardisation the three columns are highly correlated. This is **(near-)collinearity**: $\mathbf{X}^\top\mathbf{X}$ is nearly singular, so the diagonal of $\sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$ blows up and the individual standard errors of `rooms` (1.95) and `area_per_room` (1.40) are large enough to wipe out their $t$-statistics, even though the joint information they carry about size is real (`area` survives with $t = 2.76$). **Effect of dropping** `area_per_room`: the remaining two predictors are now less collinear, so their SEs would *shrink* and their $t$-values would (typically) become significant.

**Solution** *(1 %)* **(iii)** The high $p$-values on `rooms` and `area_per_room` are an artefact of inflated standard errors caused by their collinearity with `area`, not evidence that the two predictors carry no information. A joint $F$-test on the three size-related predictors would (very likely) reject; their information *is* captured by the model, just smeared across three highly correlated columns.

**Solution** *(2 %)* **(iv)** The partial effect of `age` on `rent` (holding all other predictors fixed) is

$$\frac{\partial\,\widehat{\texttt{rent}}}{\partial\,\texttt{age}} \;=\; \hat\beta_{\texttt{age}} + 2\,\hat\beta_{\texttt{age}^2}\cdot\texttt{age} \;=\; -0.080 + 2\cdot 0.00045\cdot\texttt{age} \;=\; -0.080 + 0.0009\cdot\texttt{age}.$$

Setting the derivative to zero (and noting $\hat\beta_{\texttt{age}^2} > 0$ makes this a minimum):

$$\texttt{age}^* \;=\; \frac{0.080}{0.0009} \;\approx\; \boxed{88.9 \text{ years}}.$$

*Interpretation:* the model's predicted rent decreases with age up to $\sim$ 89 years, then turns upward (presumably picking up classic / heritage stock).

**Solution** *(2 %)* **(v) Convention.** `distance` is standardised (mean 0, SD 1) as stated in the preamble, so "a 1-SD increase" means a change of $\Delta x = 1$ in the standardised `distance`.

$$\text{No balcony } (\texttt{has\_balcony} = 0): \quad \Delta\widehat{\texttt{rent}} = \hat{\beta}_{\texttt{dist}} \cdot 1 = \boxed{-1.95} \text{ (1000 NOK).}$$

$$\text{With balcony } (\texttt{has\_balcony} = 1): \quad \Delta\widehat{\texttt{rent}} = (\hat{\beta}_{\texttt{dist}} + \hat{\beta}_{\text{dist:bal}}) \cdot 1 = -1.95 + 0.55 = \boxed{-1.40} \text{ (1000 NOK).}$$

Both effects are negative (apartments further from the centre rent for less), but the balcony *attenuates* the penalty by $+0.55$ — plausibly because balconies are more valuable in the quieter, leafier zones away from the centre, so the rent loss with distance is dampened for balcony-equipped flats.

**Solution** *(1 %)* **(vi) Training $R^2$:** can only *increase* (or stay equal) — adding any new column to an OLS fit cannot raise the RSS, since the smaller model is nested inside the larger one. **Adjusted $R^2$:** depends on the size of the increase relative to the $d.f.$ cost; if the cubic term explains real curvature it rises, otherwise it may fall.

**Solution** *(1 %)* **(vii)** Use an $F$**-test** (partial / nested) comparing the full model to the reduced model that drops both `type` dummies. The null hypothesis is

$$H_0 : \beta_{\texttt{type\_studio}} = \beta_{\texttt{type\_loft}} = 0,$$

i.e. apartment *type* carries no joint information about rent after controlling for the other predictors.

### b) Lasso with 10-fold CV *(5 %)*

**Solution** *(1 %)* **(i)** The lasso objective on the standardised design $\mathbf{X} \in \mathbb{R}^{n \times p}$ is

$$\hat{\boldsymbol{\beta}}^{\text{lasso}} \;=\; \arg\min_{\beta_0, \boldsymbol{\beta}} \; \frac{1}{2n} \sum_{i=1}^{n} \big(y_i - \beta_0 - \mathbf{x}_i^\top \boldsymbol{\beta}\big)^2 \;+\; \lambda \sum_{j=1}^{p} |\beta_j|.$$

The intercept $\beta_0$ is *not* penalised (otherwise the solution depends on the units of $y$); the penalty sums absolute values of the slope coefficients only.

**Solution** *(1 %)* **(ii)** The $\ell_1$ ball $\{\boldsymbol{\beta} : \|\boldsymbol{\beta}\|_1 \le t\}$ has *corners on the coordinate axes*; the RSS contour first hits this ball at a corner with positive probability, producing exact zeros. The ridge $\ell_2$ ball is smooth and round, so the contour generically hits it at an interior point and no coefficient is set exactly to zero.

**Solution** *(2 %)* **(iii)** Lasso-$\hat{\lambda}_{\min}$ improves on OLS by 3.85 $\to$ 3.40 (test MSE) while zeroing 3 coefficients. In bias–variance terms, OLS is suffering from *high variance* on the collinear (`area`, `rooms`, `area_per_room`) bloc identified in (a)(ii); the lasso buys back variance by introducing some bias (shrinkage and selection), and the variance saving dominates. This is exactly the bias–variance lever the lasso is designed to pull, and the gain confirms that part of OLS's error was variance from the collinear directions.

**Solution** *(1 %)* **(iv) One-standard-error (1-SE) rule.** Among all $\lambda$ whose CV-MSE is within one standard error of the minimum CV-MSE, choose the *largest* (simplest / most-regularised model). The practitioner's preference is reasonable because the CV minimum is itself a noisy estimate, $\hat{\lambda}_{\text{1SE}}$ yields a sparser, more interpretable, more stable model whose test performance (3.50) is statistically indistinguishable from $\hat{\lambda}_{\min}$ (3.40).

## c) Gradient boosting *(4 %)*

**Solution** *(2 %)* **(i)** One round of gradient boosting for squared-error regression, starting from a running ensemble $\hat{f}^{(m-1)}$:

```
INPUT: training data (x_i, y_i), i = 1..n;
       current ensemble F_{m-1}(x);
       shrinkage nu in (0, 1]; tree size d.

# (i) compute the (negative) gradient = residual under squared loss
FOR i = 1..n:
    r_i = y_i - F_{m-1}(x_i)        # = -dL/df  with L = (1/2)(y - f)^2
end

# (ii) fit a regression tree of size d to the residuals
fit tree T_m to (x_i, r_i), i = 1..n,  with at most d splits / depth d

# (iii) update the ensemble with shrinkage nu
F_m(x)  <-  F_{m-1}(x) + nu * T_m(x)

RETURN F_m
```

The new tree $T_m$ is fit to the *current residuals*, $\nu$ scales each tree's contribution (smaller $\nu$ = slower learning, needs larger $M$), and the running ensemble accumulates the shrunken trees additively.

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

$$\frac{\partial L}{\partial f} = -(y - f), \qquad -\frac{\partial L}{\partial f} = y - f = \text{residual}.$$

So fitting the next tree to the residuals *is* fitting it to the negative gradient of the squared-error loss — gradient descent in function space. ✓

**Solution** *(1 %)* **(iii)** Boosting's strong improvement (MSE 2.20 vs. lasso 3.40 vs. OLS 3.85) suggests the rent surface is *nonlinear and/or has interactions* that the linear models cannot capture (e.g. nonlinear age effects, distance×balcony curvature, type-specific slopes). For interpretability, produce a **variable-importance plot** (mean reduction in node-split loss across all $M = 800$ trees), and optionally a **partial dependence plot** for the top variables to recover direction-of-effect.

## d) GAM degrees of freedom *(3 %)*

**Solution** *(1 %)* **(i)** A cubic regression spline with $K$ interior knots uses $K + 4$ basis functions including the intercept, or $K + 3$ once the global intercept is removed. Here $K = 3$ (knots at $2, 4, 7$), so

$$\text{bs}(\texttt{distance}) \text{ consumes } \boxed{K + 3 = 3 + 3 = 6} \text{ d.f.}$$

**Solution** *(1 %)* **(ii)** Total d.f. including the global intercept:

$$\begin{aligned} \text{intercept} \; &+ s(\texttt{area}, 4) + s(\texttt{age}, 5) + \text{bs}(\texttt{distance}) \\ &+ \texttt{floor} + \texttt{has\_balcony} + g(\texttt{type}) \\ &= 1 + 4 + 5 + 6 + 1 + 1 + 2 = \boxed{20} \text{ d.f.} \end{aligned}$$

($g(\texttt{type})$ contributes 2 because the 3-level categorical is coded by 2 dummy contrasts against the *flat* reference.)

**Solution** *(1 %)* **(iii)** The GAM is *additive* by construction: each predictor enters through its own one-dimensional smooth, so the model cannot represent *interactions* between predictors unless they are added by hand (e.g. `distance:has_balcony`, or tensor-product smooths). Gradient-boosted trees of depth $d > 1$ pick up such interactions automatically.

---

# Problem 5 (24 %) — Skin-lesion classification

## a) Logistic regression with an interaction *(9 %)*

**Solution** *(2 %)* **(i)** With the interaction `size_mm : age`, the partial log-odds change per extra mm of `size_mm`, at fixed `age`, is

$$\frac{\partial \hat{\eta}}{\partial\, \texttt{size\_mm}} \;=\; \hat{\beta}_{\texttt{size\_mm}} + \hat{\beta}_{\texttt{size:age}} \cdot \texttt{age} \;=\; 0.20 + 0.0020 \cdot \texttt{age}.$$

The corresponding odds factor is $\exp(\cdot)$:

$$\begin{aligned} \texttt{age} = 30: &\quad \exp(0.20 + 0.0020 \cdot 30) = \exp(0.26) \approx \boxed{1.297}, \\ \texttt{age} = 70: &\quad \exp(0.20 + 0.0020 \cdot 70) = \exp(0.34) \approx \boxed{1.405}. \end{aligned}$$

*Grading: 1 P per age. Deduct 1 P if both factors are* $\exp(0.20) \approx 1.22$*, which ignores the interaction — the standard trap.*

**Solution** *(1 %)* **(ii)** The two factors differ because the interaction term makes the size effect itself depend on `age`: each extra year of `age` adds 0.0020 to the size slope. The main-effect coefficient $\hat{\beta}_{\texttt{size\_mm}} = 0.20$ is *only* the size effect at $\texttt{age} = 0$ (an extrapolation outside the data), so it cannot summarise the size effect on its own.

**Solution** *(3 %)* **(iii)** Compute the linear predictor term by term:

$$\begin{aligned} \hat{\eta} &= -6.20 + 0.20 \cdot 8 + 2.50 \cdot 0.6 + (-0.012) \cdot 120 + 0.030 \cdot 60 + 0.18 \cdot 1 + 0.55 \cdot 1 + 0.0020 \cdot (8 \cdot 60) \\ &= -6.20 + 1.60 + 1.50 - 1.44 + 1.80 + 0.18 + 0.55 + 0.96 \\ &= -6.20 + 5.15 \;=\; \boxed{-1.05}. \end{aligned}$$

Sigmoid step:

$$\hat{p} \;=\; \frac{1}{1 + e^{-\hat{\eta}}} \;=\; \frac{1}{1 + e^{1.05}} \;=\; \frac{1}{1 + 2.858} \;\approx\; \boxed{0.259}.$$

*Grading: 1 P for the per-term sum, 1 P for* $\hat{\eta} = -1.05$*, 1 P for the sigmoid yielding* $\hat{p} \approx 0.26$*.*

**Solution** *(1 %)* **(iv)** $\hat{p} \approx 0.26 < 0.5$, so at the default threshold the patient is **not flagged as malignant** (predicted class = benign).

**Solution** *(2 %)* **(v)**

- **Sensitivity** $= P(\hat{Y} = \text{malignant} \mid Y = \text{malignant})$: the fraction of *truly malignant lesions* that the model correctly flags as malignant. ("How many real cancers do we catch?")
- **Specificity** $= P(\hat{Y} = \text{benign} \mid Y = \text{benign})$: the fraction of *truly benign lesions* the model correctly leaves alone. ("How many healthy moles do we avoid alarming?")

**Which to weigh more.** For a screening tool in dermatology, *sensitivity* is more important: a missed malignant melanoma is potentially fatal, whereas a false positive only generates a follow-up biopsy. The threshold should therefore be tuned down from 0.5 to push sensitivity up at some cost in specificity.

## b) Neural network with regularization *(7 %)*

**Solution** *(1 %)* **(i)** Layer-by-layer:

$$\begin{aligned} \text{input}(6) \to \text{hidden}(32): &\quad 6 \cdot 32 + 32 = 224, \\ \text{hidden}(32) \to \text{hidden}(16): &\quad 32 \cdot 16 + 16 = 528, \\ \text{hidden}(16) \to \text{output}(1): &\quad 16 \cdot 1 + 1 = 17, \\ \text{Total:} &\quad 224 + 528 + 17 = \boxed{769} \text{ parameters.} \end{aligned}$$

**Solution** *(2 %)* **(ii) (a)** With $\sim 769$ parameters and only 4,500 training images, the network is in the over-parameterised regime: 100% training accuracy combined with 0.78 test accuracy is textbook overfitting — the model memorises the training set, so test accuracy degrades. Without explicit regularisation a network with this much capacity is essentially guaranteed to overfit on a tabular signal this small. **(b)** Mini-batch SGD with small batches injects *gradient noise* into every step; this stochastic noise acts as an implicit regulariser, biasing the optimiser toward flatter, more generalisable minima even before dropout / early stopping / label smoothing are added.

**Solution** *(2 %)* **(iii)**

- **Dropout ($p = 0.2$).** At each training step, each hidden unit's activation is set to 0 independently with probability 0.2 (and the remaining units are typically scaled by $1/(1-p)$ in "inverted dropout"). At *test time* dropout is switched off and the full network is used — this is the train/test asymmetry.
- **Early stopping.** Monitor the validation loss across epochs; halt training when the validation loss stops improving for a configurable patience window, and return the parameters from the best epoch.
- **Label smoothing ($\varepsilon = 0.05$).** Replace the one-hot target with a softened version (e.g. $1 - \varepsilon = 0.95$ on the true class, $\varepsilon = 0.05$ on the other), so the cross-entropy loss penalises over-confident predictions and is robust to mislabeled examples.

**Solution** *(1 %)* **(iv)** The prof's typical default is $\sim 20\%$ dropout for tabular, relatively small networks; jumping to 50% tends to *under-fit* on signal-poor tabular data (too much information is destroyed at every step) and is mostly justified on much larger CNNs with ample training data, which is not this setting.

**Solution** *(1 %)* **(v)** The three regularisers traded a small increase in *bias* for a substantial decrease in *variance*, lowering test error from 0.22 to 0.14 (accuracy $0.78 \to 0.86$). The irreducible $\sigma^2$ (label noise / intrinsic difficulty) is unchanged; the gain is pure variance reduction.

## c) AdaBoost *(5 %)*

**Solution** *(2 %)* **(i)** The exponential weight update $w_i \leftarrow w_i \exp(\alpha_m \mathbb{1}[y_i \neq G_m(x_i)])$ inflates the weights of *repeatedly misclassified* observations: each successive weak learner is forced to focus on the hardest residual examples. This is fundamentally different from bagging / random forests, which *resample rows i.i.d.* with uniform probability: bagging treats every observation symmetrically and averages independent fits, while AdaBoost *adapts* sequentially to the hard cases by re-weighting (no row resampling).

**Solution** *(1 %)* **(ii)** If $\text{err}_{m^*} = 0.5$ then

$$\alpha_{m^*} = \log\left(\frac{1 - 0.5}{0.5}\right) = \log(1) = \boxed{0}.$$

**Effect:** the weak learner contributes *nothing* to the final ensemble's prediction ($\alpha_{m^*} G_{m^*}(x) = 0$); an error rate of 0.5 is the threshold at which a binary classifier is no better than a coin flip and AdaBoost ignores it.

**Solution** *(1 %)* **(iii) Deploy AdaBoost.** It dominates logistic regression on *sensitivity* (0.84 vs. 0.55) while only giving up a small amount on specificity (0.91 vs. 0.95). For a screening tool, catching melanomas matters more than the few extra false positives; AdaBoost flags 84% of real cancers vs. logistic's 55%, a clinically decisive gap.

**Solution** *(1 %)* **(iv) Exponential loss:** $\boxed{L(y, f) = \exp(-y \cdot f)}$ with $y \in \{-1, +1\}$. AdaBoost is gradient boosting under this loss in function space.

### d) Class imbalance and threshold tuning *(3 %)*

**Solution** *(1 %)* **(i)** If the classifier always predicts "benign," it is correct on every truly benign image and wrong on every malignant one. With $\sim$ 80% benign in the population, accuracy $\approx \boxed{0.80}$.

**Solution** *(1 %)* **(ii)** The naive classifier has sensitivity 0 (it never flags malignant) and specificity 1 (it never raises a false alarm) — useless. The logistic regression at threshold 0.5 achieves modest sensitivity ($\sim$ 0.55 in part (c)) and high specificity ($\sim$ 0.95), so even though its overall accuracy is only marginally better, it is genuinely informative: it actually detects a majority of the cancers.

**Solution** *(1 %)* **(iii)** Lowering the threshold from 0.5 to 0.3 flags more observations as malignant: **sensitivity increases**, **specificity decreases**, and on the ROC curve the operating point moves **up and to the right** (higher TPR, higher FPR).

---

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