

Name: _____

Entry number: _____

There are 2 questions for a total of 10 points.

1. Answer the following MCQ-type questions:

- (a) ($\frac{1}{2}$ point) In multiple linear regression with predictors $x_1, x_2, \dots, x_p$, the usual model is

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \varepsilon.$$

What does β_0 represent?

- (A) The slope of the first predictor
 - (B) **The intercept of the model**
 - (C) The residual error
 - (D) The number of predictors
- (b) ($\frac{1}{2}$ point) In the ordinary least squares method, the coefficients are chosen to minimize
- (A) the sum of residuals
 - (B) the sum of absolute predictor values
 - (C) **the sum of squared residuals**
 - (D) the number of nonzero coefficients
- (c) ($\frac{1}{2}$ point) The root mean squared error, RMSE, is useful because
- (A) it is always equal to R^2
 - (B) **it is measured in the same units as the response variable**
 - (C) it is independent of prediction errors
 - (D) it is always between 0 and 1
- (d) ($\frac{1}{2}$ point) The coefficient of determination R^2 measures
- (A) the number of predictors in the model
 - (B) **the proportion of variation in the response explained by the model**
 - (C) the average value of the response variable
 - (D) the probability that all coefficients are zero
- (e) ($\frac{1}{2}$ point) If $R^2 = 0.82$, which interpretation is most appropriate?
- (A) The model makes correct predictions 82% of the time.
 - (B) **About 82% of the variability in the response is explained by the model.**
 - (C) Every coefficient has value 0.82.
 - (D) The test error is exactly 0.18.

- (f) ($\frac{1}{2}$ point) Suppose the coefficient of a predictor x_1 is estimated to be 5, while all other variables are held fixed. What does this mean?
- (A) **Increasing x_1 by one unit is associated with an increase of 5 units in the predicted response.**
 - (B) The response must always equal 5.
 - (C) The predictor x_1 has no effect on the prediction.
 - (D) The residual error is always 5.
- (g) ($\frac{1}{2}$ point) Which statement about adding more predictors to a linear regression model is true?
- (A) **Training error can never increase when a predictor is added.**
 - (B) Test error must always decrease when a predictor is added.
 - (C) R^2 must always decrease when a predictor is added.
 - (D) Adding predictors always removes multicollinearity.
- (h) ($\frac{1}{2}$ point) A model with very low training error but high test error is likely suffering from
- (A) underfitting
 - (B) **overfitting**
 - (C) perfect generalization
 - (D) absence of noise
- (i) ($\frac{1}{2}$ point) In regression, a model with *high bias* usually means that the model
- (A) fits the training data too closely
 - (B) **is too simple to capture the true relationship**
 - (C) has very low training error
 - (D) changes drastically when the training set changes
- (j) ($\frac{1}{2}$ point) In regression, a model with *high variance* usually means that the model
- (A) is too simple
 - (B) ignores the predictor variables
 - (C) **is very sensitive to the training data**
 - (D) always has high training error
- (k) ($\frac{1}{2}$ point) Suppose a multiple linear regression model has both high training error and high test error. This is most likely an example of
- (A) overfitting
 - (B) **underfitting**
 - (C) perfect fitting
 - (D) data leakage
- (l) ($\frac{1}{2}$ point) Suppose a polynomial regression model has very low training error but much larger test error. This is most likely an example of
- (A) underfitting
 - (B) **overfitting**
 - (C) high bias
 - (D) no variance

2. (4 points) Give the output of the following code that executes two steps of the gradient descent method on a given dataset.

```
import numpy as np

y = np.array([1, 2, 3, 4])
X = np.array([[1, 1, 0, 0],
              [1, 0, 1, 1],
              [1, 0, 0, 1],
              [1, 1, 0, 1]])

X

array([[1, 1, 0, 0],
       [1, 0, 1, 1],
       [1, 0, 0, 1],
       [1, 1, 0, 1]])

theta = np.zeros(4)
n = 4
lr = 0.1

for i in range(2):
    gradient = (2 / n) * X.T @ (X @ theta - y)
    print('Gradient in step', i, ' is ', gradient)
    theta = theta - lr * gradient
    print('theta in step', i, ' is ', theta)
    mse_loss = np.mean((y - X @ theta)**2)
    print('mse_loss in step', i, ' is ', mse_loss)
```

Gradient in step 0 is [-5. -2.5 -1. -4.5]
 theta in step 0 is [0.5 0.25 0.1 0.45]
 mse_loss in step 0 is 3.2518749999999996
 Gradient in step 1 is [-3.025 -1.525 -0.475 -2.9]
 theta in step 1 is [0.8025 0.4025 0.1475 0.74]
 mse_loss in step 1 is 1.6213640624999996