

DATA/STAT/BIOSTAT 558
SPRING QUARTER 2025

Homework # 3

Online Submission to Canvas: due Wednesday May 14th, 5pm PST

Instructions: You may discuss the homework problems in small groups, but you must write up the final solutions and code yourself. Please turn in your code used for the problems that involve coding. However, code without written answers will receive no credit. To receive credit, you must explain your answers and show your work. All plots should be appropriately labeled and legible, with axis labels, legends, etc., as needed. **Please submit your work in PDF form.**

Problem 1 In this problem, we will examine how to use cross-validation correctly and incorrectly in order to select and then evaluate the test error of a model. In Lecture 4, we discussed “Procedure 1” and “Procedure 2”: two possible ways to perform the validation set approach for a modeling strategy where you identify the q features most correlated with the response, and then fit an OLS model to predict the response using just those q features. If you missed that lecture, then please familiarize yourself with the lecture notes (Canvas) before you continue.

Consider $n = 100$ independent observations generated independently and identically as follows, with $p = 10,000$:

$$\begin{aligned} X_1, \dots, X_n &\sim N_p(0, I), \\ Y_1, \dots, Y_n &\sim N(0, 2). \end{aligned}$$

- (a) Why can OLS not be meaningfully applied here? Provide three approaches mentioned in class to address this issue.

Solution : Here, the number of features($p=10000$) is much greater than observations ($n=100$). With more predictors than observations, OLS can perfectly fit the training data but there can be a lot of variability in the least squares fit, resulting in overfitting and consequently poor predictions on unseen data. Hence OLS cannot be meaningfully applied in the setting. Three approaches to address this issue in terms of cross validation include:

- 1 Leave-One-Out Cross-Validation (LOOCV): Cross-validation approach where each data point is left out one at a time.

- 2 Validation Set Approach (Sample Splitting): The dataset is split into a training set and a validation set. Feature selection and model fitting are done on the training set and error is estimated on the validation set.
- 3 K-Fold Cross-Validation: The data is split into K folds. The model is trained on K - 1 folds and validated on the remaining fold, repeated K times.

The three approaches to address issue when number of predictors are greater than n are:

- 1 Subset Selection: Choosing a subset of predictors
- 2 Shrinkage/Regularization: Ridge or Lasso to penalize model complexity
- 3 Dimensionality Reduction

- (b) What is the irreducible error of this problem?

Solution : Irreducible Error = $\text{Var}(Y) = 2$

- (c) Simulate a dataset and calculate the correlation between each feature and the response. Make a histogram of these correlations.

Solution :

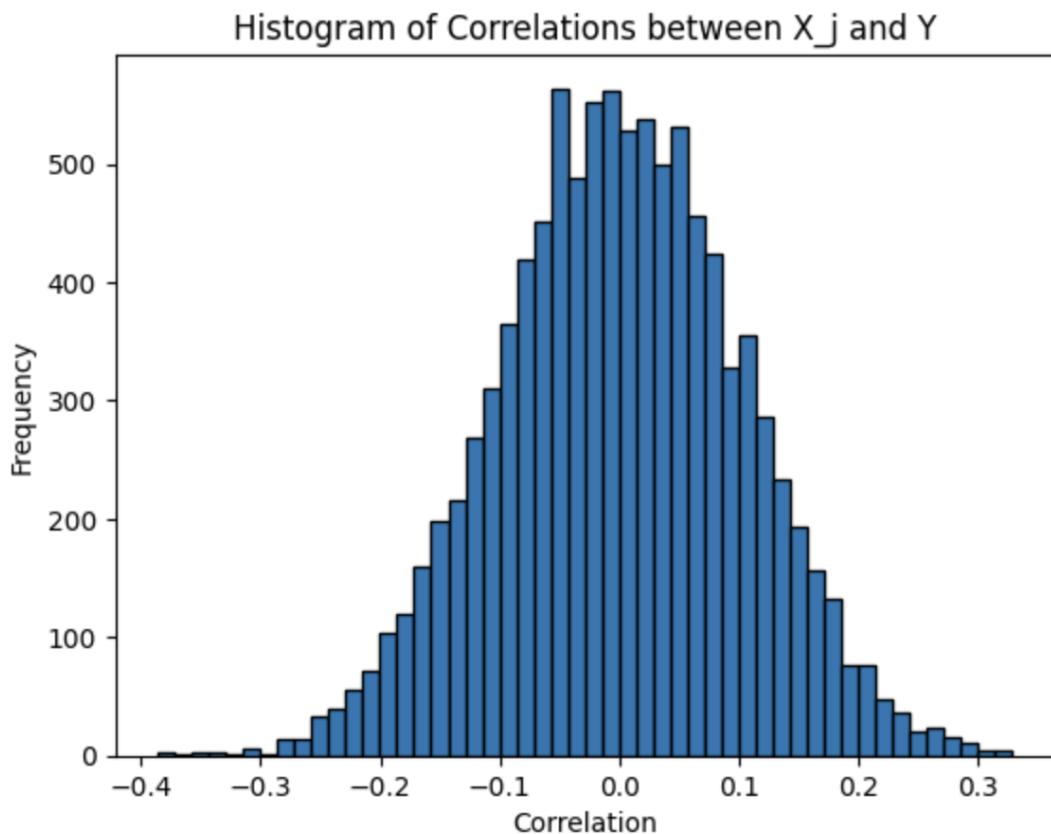


Figure 1: (1c) Histogram of correlations between X_i and Y

- (d) Simulate a test data set of 1000 points. Now try out Procedure 1 with $q = 5$ to obtain an estimate of the expected prediction error. Repeat this process 1,000 times and compute the average of the estimates of the expected prediction error.

Solution : Avg expected prediction error : 1.6836

- (e) Repeat (d), but this time with Procedure 2.

Solution : Avg expected prediction error : 3.3010

- (f) Comment on your results in (d) and (e) in relation to the irreducible error. How does this relate to the discussion of the two procedures from lecture? Explain why Procedure 1 gave you a useless (i.e. misleading, inaccurate, wrong) estimate of the expected prediction error.

Solution : The irreducible error in this problem is 2. In Procedure 1, the average estimated prediction error was 1.6836, which is lower than the irreducible error. This is not possible in a realistic modeling scenario, since no model no matter how good can achieve a prediction error below the irreducible noise in the data. This indicates that Procedure 1 is providing a misleading estimate of the expected prediction error.

In contrast, Procedure 2 produced an average estimated prediction error of 3.3010, which is higher than the irreducible error. Procedure 2 estimate is valid because it separates feature selection and model evaluation: both steps are performed only using the training data and the test data is truly unseen. Procedure 1 fails because it uses the entire dataset to select the top q features most correlated with the response. This causes information leakage, the model is indirectly optimized on the test data which leads to overfitting and an overly optimistic error estimate. This violates the principles of proper cross-validation.

- (g) Suppose you wanted to choose the value of q using the data. Describe a procedure (no need to code) which would enable you to choose a value of q using the data, and would provide a valid estimate of the expected prediction error.

Solution : A valid procedure to choose the value of q and obtain an unbiased estimate of the expected prediction error. The steps are as follows:

- 1 Split the data into a training set and a test set (80% training, 20% test)
- 2 Within the training set, either:
 - further split it into a smaller training and validation set, or
 - use k -fold cross-validation.
- 3 For each candidate value of q (e.g., $q = 1, 2, \dots, 20$):
 - Select the top q features most correlated with the response using only the training portion.
 - Fit a linear model on those q features.
 - Evaluate the model on the validation set and record the validation error.

- 4 Choose the value of q that yields the lowest validation error.
- 5 Refit the model using the selected q and all of the training data.
- 6 Evaluate the final model on the test set to obtain a valid estimate of the expected prediction error.

Problem 2 Leave-one-out cross validation (LOOCV) can be computationally costly. In this exercise, we will verify a shortcut applicable to ordinary least-squares (OLS) regression in the context of mean-squared-error. Recall that the LOOCV error is defined as

$$\text{CV}_n^{\text{LOOCV}} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{f}_{[-i]}(y_i))^2$$

where $\hat{f}_{[-i]}$ is model fitted to all but the i th training sample. In this model, we will fit all models using least squares.

In the case of models fit using least squares, equation (5.2) in the textbook says that

$$\text{CV}_n^{\text{shortcut}} := \frac{1}{n} \sum_{i=1}^n \frac{y_i - \hat{y}_i}{1 - h_i}^2 \quad (1)$$

where h_i is the *leverage*, defined as the i th element along the diagonal of the *hat matrix*

$$H := X(X^\top X)^{-1}X^\top,$$

i.e., $h_i = H_{ii}$, where $X \in \mathbb{R}^{n \times (p+1)}$ since you have n observations and p covariates plus an intercept term (the first column of X , the intercept, is all ones!).

In simulation, show that, for a least squares model that predicts Y using X , it is the case that

$$\text{CV}_n^{\text{LOOCV}} = \text{CV}_n^{\text{shortcut}}. \quad (2)$$

To conduct the simulation, sample $X_1, \dots, X_n \sim N(0, 1)$ for $n = 20$. Sample $Y_i \sim \exp(X_i) + \varepsilon_i$ where $\varepsilon_i \sim N(0, 0.25)$. Report the leave-one-out cross-validation error estimate obtained by actually performing LOOCV, and the estimate obtained using the shortcut in the textbook on a sample dataset. (They should be the same!)

Solution : Errors (the two are equal)

LOOCV error (actual): **1.43087**

LOOCV error (shortcut): **1.43087**

Problem 3 In this problem, we will study how different forms of cross-validation compare in estimating the expected prediction error across different estimators.

Consider training datasets of $n = 20$ independent observations generated independently and identically as follows:

$$\begin{aligned} X_i &\sim N(0, 1), \\ \varepsilon_i &\sim N(0, 0.25), \\ Y_i &= \exp(X_i) + \varepsilon \end{aligned}$$

- (a) Simulate and plot a training dataset.

Solution :

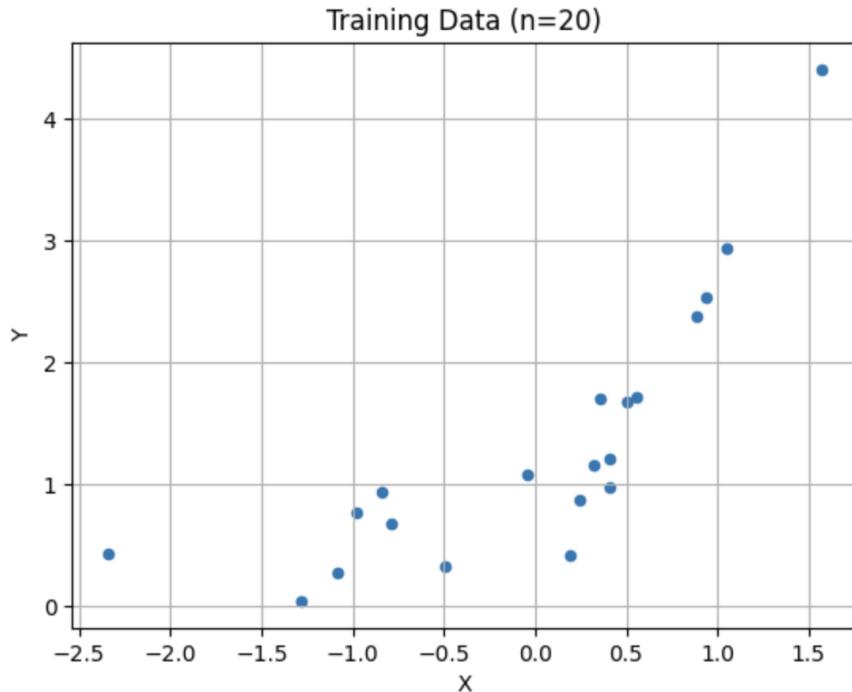


Figure 2: (3a) Training Data ($n = 20$)

- (b) Fit an OLS model using just X_i and an intercept to all 20 training data points and compute its test error on 10000 simulated test data points. Repeat this process 100 times on new training datasets, and report the mean and standard deviation of the test error across the 100 training datasets.

Solution : OLS model using intercept and X_i

Mean test error = 2.614

Standard deviation of test error = 1.125

- (c) Using your efficient result from Problem 2, perform LOOCV on one of the training sets to estimate the expected prediction error. Repeat this process 100 times on 100 training sets, and report the mean and standard deviation of the LOOCV error across the 100 training datasets.

Solution : LOOCV error

Mean error = 2.278

Standard deviation of error = 4.422

- (d) Repeat step (c) using 2-fold CV and 5-fold CV.

Solution : K-fold error

Mean error ($k=2$) = 2.792

Standard deviation of error ($k=2$) = 5.053

Mean error ($k=5$) = 2.614

Standard deviation of error ($k=5$) = 4.874

- (e) Repeat steps (b, c, d) using instead an OLS model on X_i, X_i^2, X_i^3, X_i^4 , and an intercept. Note, the efficient LOOCV shortcut is still applicable here, but with a new matrix of covariates \tilde{X} with 5 columns for the 5 above features.

Solution : Polynomial model

Mean test error (OLS) = 1.651

Standard deviation of test error (OLS) = 8.002

Mean error (LOOCV) = 5.188

Standard deviation of error (LOOCV) = 29.369

Mean error (2-Fold CV) = 27.270

Standard deviation of error (2-Fold CV) = 83.236

Mean error (5-Fold CV) = 4.624

Standard deviation of error (5-Fold CV) = 24.576

- (f) Provide all results in a 2x4 table: 2 rows for the two OLS models, and 4 columns for the OLS expected prediction error in (b), and three estimates of expected prediction error obtained in (c) and (d). Each element of the table should report both the mean of the estimated expected prediction error, and (in parentheses) the standard error of the mean. How do the four estimates of prediction error compare for a given model? How do the estimated prediction errors compare across models? Explain your answers.

Solution :

Model	Test	LOOCV	2-Fold CV	5-Fold CV
Linear (X)	2.614 (0.113)	2.278 (0.442)	2.792 (0.5053)	2.614 (0.4874)
Poly (X, X^2, X^3, X^4)	1.651 (0.8002)	5.188 (2.9369)	27.270 (8.3236)	4.624 (2.4576)

Table 1: Estimates of expected prediction error for linear and polynomial OLS models. Values shown are mean of the estimated expected prediction error (standard error of the mean)

For the **linear model** (using intercept and X), the four estimates of prediction error are consistent with each other and close to the test error of 2.614. All cross-validation methods (LOOCV, 2-fold, and 5-fold) yield

similar estimates with relatively small standard errors. This indicates that the linear model is stable and generalizes well and that any of the CV methods provides a reliable estimate of its prediction performance. In contrast, for the **polynomial model** (using intercept, X, X^2, X^3, X^4), there is significant variability across the different estimates. Although the test error is lower (1.651), the cross-validation estimates are much higher especially for 2-fold CV which gives a highly inflated error of 27.270 with a large standard error. This suggests that the polynomial model, while more flexible, is also much more sensitive to small training sets and is prone to overfitting. The inflated CV errors are a result of high variance due to limited training data per fold and model complexity. This comparison shows the bias-variance tradeoff and the importance of using appropriate cross-validation strategies when evaluating more complex models.

Problem 4 In this problem, we will use cross-validation to perform and evaluate a model selection procedure. You will analyze a (real, not simulated) dataset of your choice with a quantitative response Y , and $p \geq 10$ predictors.

- (a) What is your dataset, how many features are there, and what is the response Y ?

Solution : Data used: Boston Data, a data set containing housing values in 506 suburbs of Boston.

Source: UCI Machine Learning Repository

Number of Features: There are 13 predictor variables (features) in the dataset.

Response Variable (Y): The response is "medv" which represents the median value of owner-occupied homes in \$1000s.

- (b) Fit a least squares linear model to the data, and provide an estimate of the test error. (Explain how you got this estimate.)

Solution : To estimate the test error, the data was split into training (80%) and test(20%). Linear Regression model was fit on the training set and the reported test error was calculated on the test set.

Test MSE: 33.92

- (c) Fit a ridge regression model to the data, with a range of values of the tuning parameter λ . Make a plot like the left-hand panel of Figure 6.4 in the textbook.

Solution :

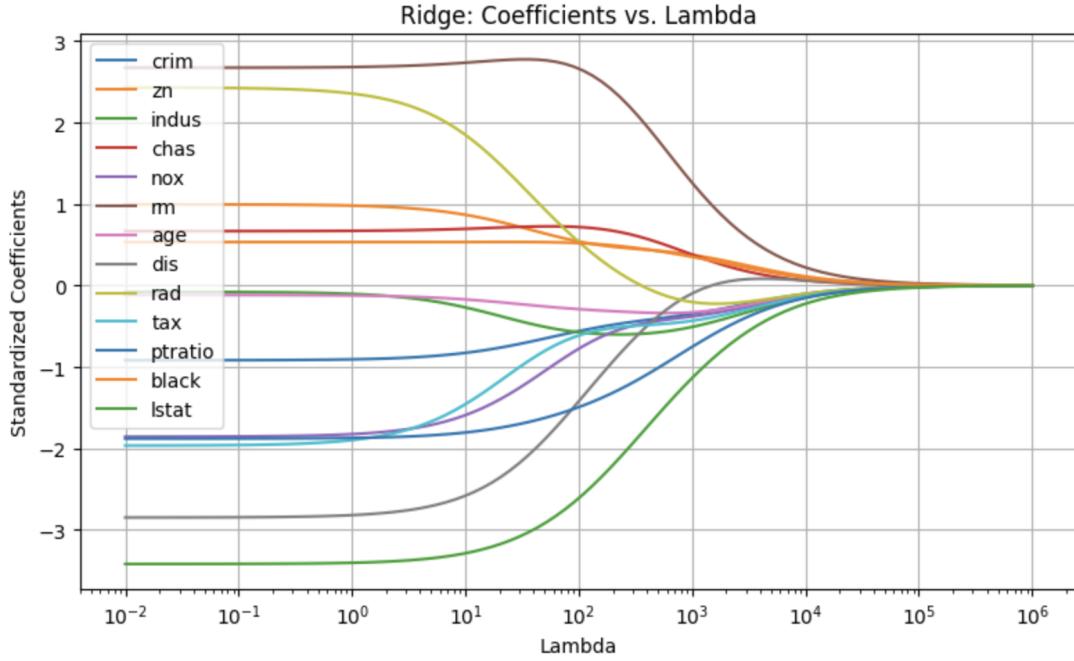


Figure 3: (4c) Ridge coefficients with *lambda*

- (d) What value of λ in the ridge regression model provides the smallest estimated test error? Report this estimate of test error. (Also, explain how you estimated test error.)

Solution : To estimate the test error, the data was split into training (80%) and test(20%). The data was standardized before fitting, since ridge regression is sensitive to the scale of predictors. RidgeCV with cross-validation was used on the training data to find the best λ value that minimizes prediction error. The range of λ varied from 10^{-2} to 10^6 . The model was fit using this optimal tuning parameter on the training data. The estimated test error was then calculated on the test set.

λ value: 4.6416

Estimated test error: 34.09

- (e) Repeat (c), but for a lasso model.

Solution :

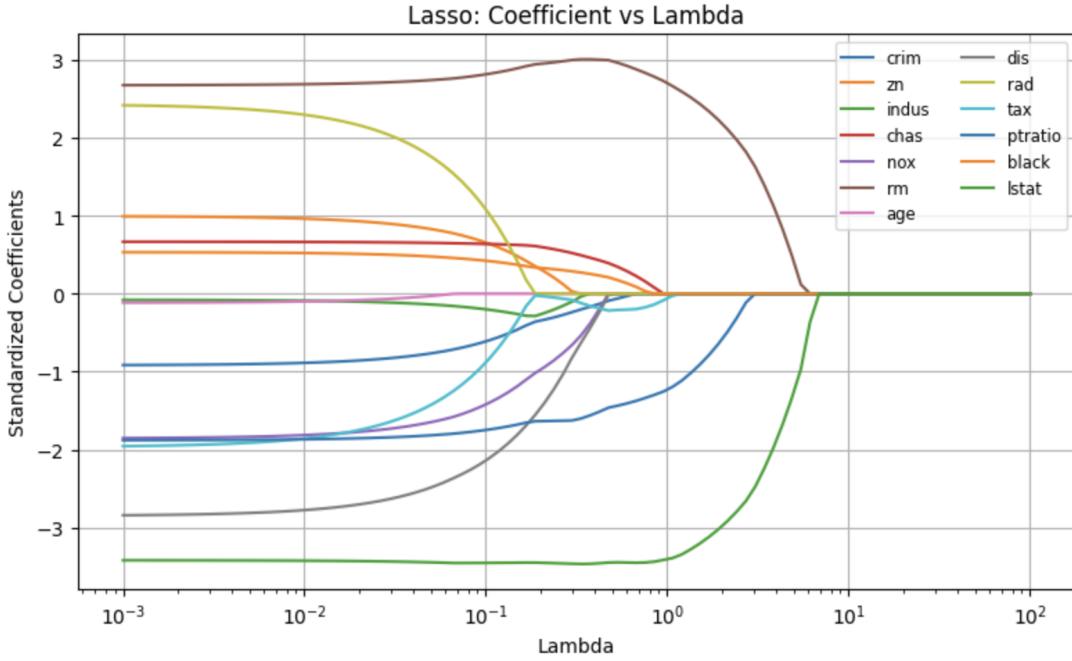


Figure 4: (4e) Lasso coefficients with *lambda*

(f) Repeat (d), but for a lasso model.

Solution : To estimate the test error, the data was split into training (80%) and test(20%). The data was standardized before fitting, since lasso regression is sensitive to the scale of predictors. LassoCV with cross-validation was used on the training data to find the best λ value that minimizes prediction error. The range of λ varied from 10^{-3} to 10^2 . The model was fit using this optimal tuning parameter on the training data. The estimated test error was then calculated on the test set.

λ value: 0.0140

Estimated test error: 34.01

(g) How could you select (in a valid manner) which model (least squares linear model, ridge regression, or lasso) is the best and how could you provide an estimate of the test error?

Solution : The best model should be chosen based on cross-validation performance on training data. The test set should be used only once at the end to estimate the true test error of the selected model. This ensures that model selection is unbiased and generalizes well to new data.

Our results:

- OLS Test MSE: 24.29
- Ridge Test MSE: 34.09
- Lasso Test MSE: 34.01

Although OLS achieved the lowest test MSE in your particular split, both ridge and lasso used cross-validation to tune complexity and may gener-

alize better on different splits or larger datasets. However, since OLS was evaluated fairly (on a held-out test set), its lower MSE suggests it may be the better model on this data and split.

Problem 5 In this problem, we will study properties of the ridge regression estimator. Let's consider doing OLS and ridge regression under a very simple setting, in which $p = 1$, and $\sum_{i=1}^n y_i = \sum_{i=1}^n x_i = 0$ (i.e. have been centered). We consider regression without an intercept. (It's usually a bad idea to do regression without an intercept, but if our feature and response each have mean zero, then it is okay to do this!)

- (a) The least squares solution is the value of $\beta \in \mathbb{R}$ that minimizes

$$\sum_{i=1}^n (y_i - \beta x_i)^2.$$

Write out an analytical (closed-form) expression for this OLS solution. Your answer should be a function of x_1, \dots, x_n and y_1, \dots, y_n .

Hint: Think derivatives!!

Solution : Least square solution of $\sum_{i=1}^n (y_i - \beta x_i)^2$.

The derivative with respect to β should be 0 at the OLS solution ($\hat{\beta}_{\text{OLS}}$)

$$\frac{d}{d\beta} \sum_{i=1}^n (y_i - \beta x_i)^2 = -2 \sum_{i=1}^n x_i (y_i - \beta x_i) = 0$$

$$\begin{aligned} \sum_{i=1}^n x_i y_i &= \beta \sum_{i=1}^n x_i^2 \\ \hat{\beta}_{\text{OLS}} &= \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2} \end{aligned}$$

This is the (closed-form) expression for this OLS solution.

- (b) For a given value of λ , the ridge regression solution minimizes

$$\sum_{i=1}^n (y_i - \beta x_i)^2 + \lambda \beta^2.$$

Write out an analytical (closed-form) expression for the ridge regression solution, in terms of x_1, \dots, x_n and y_1, \dots, y_n and λ .

Solution : Least square solution of $\sum_{i=1}^n (y_i - \beta x_i)^2$.

The derivative with respect to β should be 0 at the Ridge solution ($\hat{\beta}_{\text{Ridge}}$)

$$\frac{d}{d\beta} \left(\sum_{i=1}^n (y_i - \beta x_i)^2 + \lambda \beta^2 \right) = 0$$

$$-2 \sum_{i=1}^n x_i(y_i - \beta x_i) + 2\lambda\beta = 0$$

$$\beta \sum_{i=1}^n x_i^2 + \lambda\beta = \sum_{i=1}^n x_i y_i$$

$$\hat{\beta}_{\text{Ridge}} = \frac{\sum_{i=1}^n x_i y_i}{\lambda + \sum_{i=1}^n x_i^2}$$

- (c) Throughout the following problems, suppose that the true data-generating model is

$$Y = -3X + \epsilon,$$

where ϵ has mean zero, and X is fixed (non-random). What is the expectation of the OLS estimator from (a)? Is it biased or unbiased?

Solution : From (a)

$$\hat{\beta}_{\text{OLS}} = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}$$

$$\hat{\beta}_{\text{OLS}} = \frac{\sum_{i=1}^n x_i(-3x_i + \epsilon_i)}{\sum_{i=1}^n x_i^2}$$

$$\hat{\beta}_{\text{OLS}} = \frac{-3 \sum_{i=1}^n x_i^2 + \sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2}$$

$$\hat{\beta}_{\text{OLS}} = -3 + \frac{\sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2}$$

$$E(\hat{\beta}_{\text{OLS}}) = E\left(-3 + \frac{\sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2}\right)$$

$$E(\hat{\beta}_{\text{OLS}}) = E(-3) + E\left[\frac{\sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2}\right]$$

Since x_i is fixed and $E[\epsilon_i] = 0$:

$$E(\hat{\beta}_{\text{OLS}}) = -3 + \frac{\sum_{i=1}^n x_i E(\epsilon_i)}{\sum_{i=1}^n x_i^2} = -3$$

$E(\hat{\beta}_{\text{OLS}}) = -3$. The expected value of the OLS estimated of β matches the true value in the true mode, hence the OLS estimator is unbiased.

- (d) What is the expectation of the ridge regression estimator from (b)? Is it biased or unbiased? Explain how the bias changes as a function of λ .

Solution : From (b)

$$\hat{\beta}_{\text{Ridge}} = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2 + \lambda}$$

$$\hat{\beta}_{\text{Ridge}} = \frac{\sum_{i=1}^n x_i(-3x_i + \epsilon_i)}{\sum_{i=1}^n x_i^2 + \lambda}$$

$$\hat{\beta}_{\text{Ridge}} = \frac{-3 \sum_{i=1}^n x_i^2 + \sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2 + \lambda}$$

$$E(\hat{\beta}_{\text{Ridge}}) = E\left[\frac{-3 \sum_{i=1}^n x_i^2 + \sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2 + \lambda}\right]$$

$$E(\hat{\beta}_{\text{Ridge}}) = E\left[\frac{-3 \sum_{i=1}^n x_i^2}{\sum_{i=1}^n x_i^2 + \lambda}\right] + E\left[\frac{\sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2 + \lambda}\right]$$

Since x_i, λ is fixed and $E[\epsilon_i] = 0$:

$$E(\hat{\beta}_{\text{Ridge}}) = \frac{-3 \sum_{i=1}^n x_i^2}{\lambda + \sum_{i=1}^n x_i^2}$$

$$Bias = E(\hat{\beta}_{\text{Ridge}}) - (-3)$$

Since $E(\hat{\beta}_{\text{Ridge}}) \neq -3$, this is biased. Since $E(\hat{\beta}_{\text{Ridge}})$ increases with increasing λ , the bias increases with increasing λ .

As $\lambda \rightarrow 0$, the $E(\hat{\beta}_{\text{Ridge}})$ approaches -3, so bias equals 0 and ridge behaves like OLS.

As $\lambda \rightarrow \infty$, the $E(\hat{\beta}_{\text{Ridge}})$ tends to zero (high bias).

- (e) Now suppose additionally that the variance of ε is σ^2 and $\text{Cov}(\epsilon_i, \epsilon_{i'}) = 0$ for all $i \neq i'$. What is the variance of the OLS estimator from (a)?

Solution : From (a)

$$\hat{\beta}_{\text{OLS}} = \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2}$$

$$\hat{\beta}_{\text{OLS}} = \frac{\sum_{i=1}^n x_i(-3x_i + \epsilon_i)}{\sum_{i=1}^n x_i^2}$$

$$\hat{\beta}_{\text{OLS}} = \frac{-3 \sum_{i=1}^n x_i^2 + \sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2}$$

$$\hat{\beta}_{\text{OLS}} = -3 + \frac{\sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2}$$

$$Var(\hat{\beta}_{\text{OLS}}) = Var\left(-3 + \frac{\sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2}\right)$$

$$Var(\hat{\beta}_{\text{OLS}}) = Var\left(\frac{\sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2}\right)$$

Since x_i is fixed and $Var(\epsilon_i) = \sigma^2$:

$$Var(\hat{\beta}_{\text{OLS}}) = \frac{\sum_{i=1}^n x_i^2 Var(\epsilon_i)}{(\sum_{i=1}^n x_i^2)^2}$$

$$Var(\hat{\beta}_{\text{OLS}}) = \frac{\sigma^2}{\sum_{i=1}^n x_i^2}$$

- (f) What is the variance of the ridge estimator from (b)? How does the variance change as a function of λ ?

Solution : From (b)

$$\begin{aligned}\hat{\beta}_{\text{Ridge}} &= \frac{\sum_{i=1}^n x_i y_i}{\sum_{i=1}^n x_i^2 + \lambda} \\ \hat{\beta}_{\text{Ridge}} &= \frac{\sum_{i=1}^n x_i (-3x_i + \epsilon_i)}{\sum_{i=1}^n x_i^2 + \lambda} \\ \hat{\beta}_{\text{Ridge}} &= \frac{-3 \sum_{i=1}^n x_i^2 + \sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2 + \lambda} \\ \text{Var}(\hat{\beta}_{\text{Ridge}}) &= \text{Var}\left(\frac{-3 \sum_{i=1}^n x_i^2 + \sum_{i=1}^n x_i \epsilon_i}{\sum_{i=1}^n x_i^2 + \lambda}\right)\end{aligned}$$

Since x_i is fixed and $\text{Var}(\epsilon_i) = \sigma^2$:

$$\begin{aligned}\text{Var}(\hat{\beta}_{\text{Ridge}}) &= \frac{\text{Var}(\sum_{i=1}^n x_i \epsilon_i)}{(\sum_{i=1}^n x_i^2 + \lambda)^2} \\ \text{Var}(\hat{\beta}_{\text{Ridge}}) &= \frac{\sum_{i=1}^n x_i^2 \text{Var}(\epsilon_i)}{(\sum_{i=1}^n x_i^2 + \lambda)^2} \\ \text{Var}(\hat{\beta}_{\text{Ridge}}) &= \frac{\sigma^2 \sum_{i=1}^n x_i^2}{(\lambda + \sum_{i=1}^n x_i^2)^2}\end{aligned}$$

Variance $\text{Var}(\hat{\beta}_{\text{Ridge}})$ decreases with increasing λ .

As $\lambda \rightarrow 0$, the $\text{Var}(\hat{\beta}_{\text{Ridge}})$ approaches the OLS variance.

As $\lambda \rightarrow \infty$, the $\text{Var}(\hat{\beta}_{\text{Ridge}})$ tends to zero.

- (g) In light of your answers to parts (d) and (f), argue that λ in ridge regression allows us to control model complexity by trading off bias for variance.

Solution : The tuning parameter λ in ridge regression controls model complexity. Based on results from parts (d) and (f), a small λ results in a more flexible model with low bias but potentially high variance. A large λ produces a less flexible model with higher bias but lower variance. When λ is close to zero, ridge regression behaves like ordinary least squares (OLS), which has low bias but potentially high variance. Therefore the tuning parameter offers a way to trade off bias and variance: increasing λ increases bias but decreases variance while decreasing λ does the opposite. The optimal tuning parameter balances the two to minimize total error.

- (h) How does the value of λ in the lasso trade off bias and variance? (Please provide a qualitative explanation, not a rigorous mathematical argument.)

Solution : Lasso also introduces bias like ridge, but due to its l_1 penalty, it shrinks some coefficients exactly to zero. This performs variable selection which can reduce model complexity even more aggressively. Bias-variance tradeoff still applies: when λ is small, the lasso model behaves more like

OLS with low bias and higher variance. As λ increases, the model becomes sparser and simpler with high bias and low variance. This makes lasso particularly useful when we believe that only a small subset of predictors are truly relevant to the response.

Hint: For this problem, you might want to brush up on some basic properties of means and variances! For instance, if $Cov(Z, W) = 0$, then $Var(Z + W) = Var(Z) + Var(W)$. And if a is a constant, then $Var(aW) = a^2Var(W)$, and $Var(a + W) = Var(W)$.

✓ APPENDIX

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression, RidgeCV, Ridge, LassoCV, Las
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.preprocessing import PolynomialFeatures
from sklearn.preprocessing import StandardScaler
```

✓ Question1

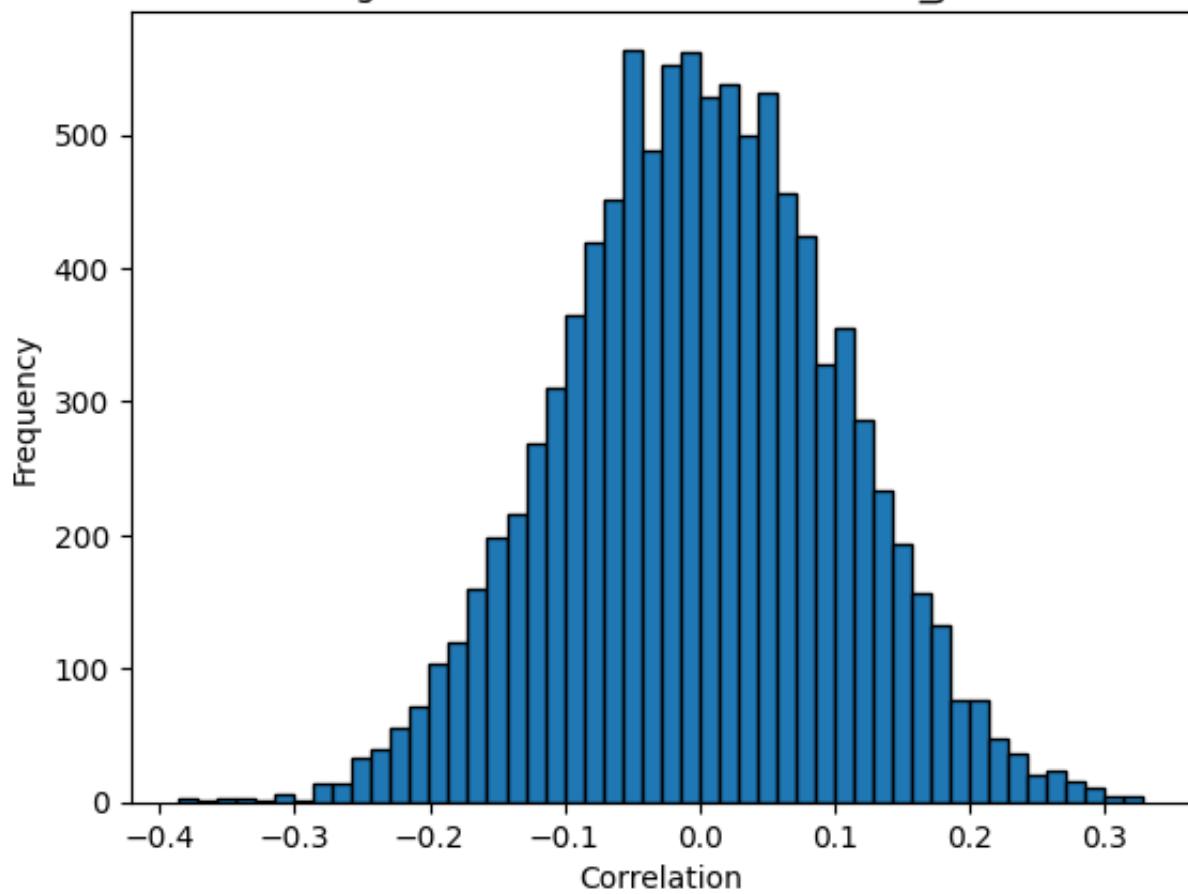
Question 1c

```
np.random.seed(10)
n, p = 100, 10000
X = np.random.normal(0, 1, (n, p))
Y = np.random.normal(0, np.sqrt(2), n)
correlations = np.corrcoef(X.T, Y)[-1, :-1]

plt.hist(correlations, bins=50, edgecolor='k')
plt.title("Histogram of Correlations between X_j and Y")
plt.xlabel("Correlation")
plt.ylabel("Frequency")
plt.show()
```



Histogram of Correlations between X_j and Y



Question 1d

```
top_q = np.argsort(np.abs(correlations))[-5:]
def procedure_1():
    X_train, X_test, Y_train, Y_test = train_test_split(X[:, top_q], Y,
                                                       test_size=0.5)
    model = LinearRegression().fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    return mean_squared_error(Y_test, Y_pred)

errors_proc1 = [procedure_1() for _ in range(1000)]
print("Avg prediction error (Procedure 1):", np.mean(errors_proc1))

→ Avg prediction error (Procedure 1): 1.6836239935711252

def procedure_2():
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.5)
    X_centered = X_train - X_train.mean(axis=0)
    Y_centered = Y_train - Y_train.mean()
    correlations = (X_centered.T @ Y_centered) / (
        np.sqrt(np.sum(X_centered ** 2, axis=0))*np.sqrt(np.sum(Y_centered**2)))
    top_q_indices = np.argsort(np.abs(correlations))[-5:]
    model = LinearRegression().fit(X_train[:, top_q_indices], Y_train)
    Y_pred = model.predict(X_test[:, top_q_indices])
    return mean_squared_error(Y_test, Y_pred)

errors_proc2 = [procedure_2() for _ in range(1000)]
print("Avg prediction error (Procedure 2):", np.mean(errors_proc2))

→ Avg prediction error (Procedure 2): 3.3088195167758085
```

▼ Question 2

Question 2 : LOOVC Actual

```
np.random.seed(11)
n = 20
X = np.random.normal(0, 1, n)
epsilon = np.random.normal(0, 0.25, n)
y = np.exp(X) + epsilon
mse_actual = []
for i in range(n):
    X_train = np.delete(X, i, axis=0).reshape(-1, 1)
    Y_train = np.delete(y, i)
    X_test = X[i].reshape(1, -1)
    Y_test = y[i]

    model = LinearRegression().fit(X_train, Y_train)
    Y_pred = model.predict(X_test)
    mse_actual.append((Y_pred[0] - Y_test)**2)

print("LOOCV actual error:", np.mean(mse_actual))
```

→ LOOCV actual error: 1.4308704668794814

Question 2 : LOOVC Shortcut

```
X_s = np.column_stack((np.ones(n), X.reshape(-1, 1)))
model = LinearRegression().fit(X_s, y)
y_hat = model.predict(X_s)

H = X_s @ np.linalg.inv(X_s.T @ X_s) @ X_s.T
h = np.diag(H)
error = np.mean(((y - y_hat) / (1 - h)) ** 2)
print("LOOCV shortcut error:", error)
```

→ LOOCV shortcut error: 1.4308704668794825

▼ Question 3

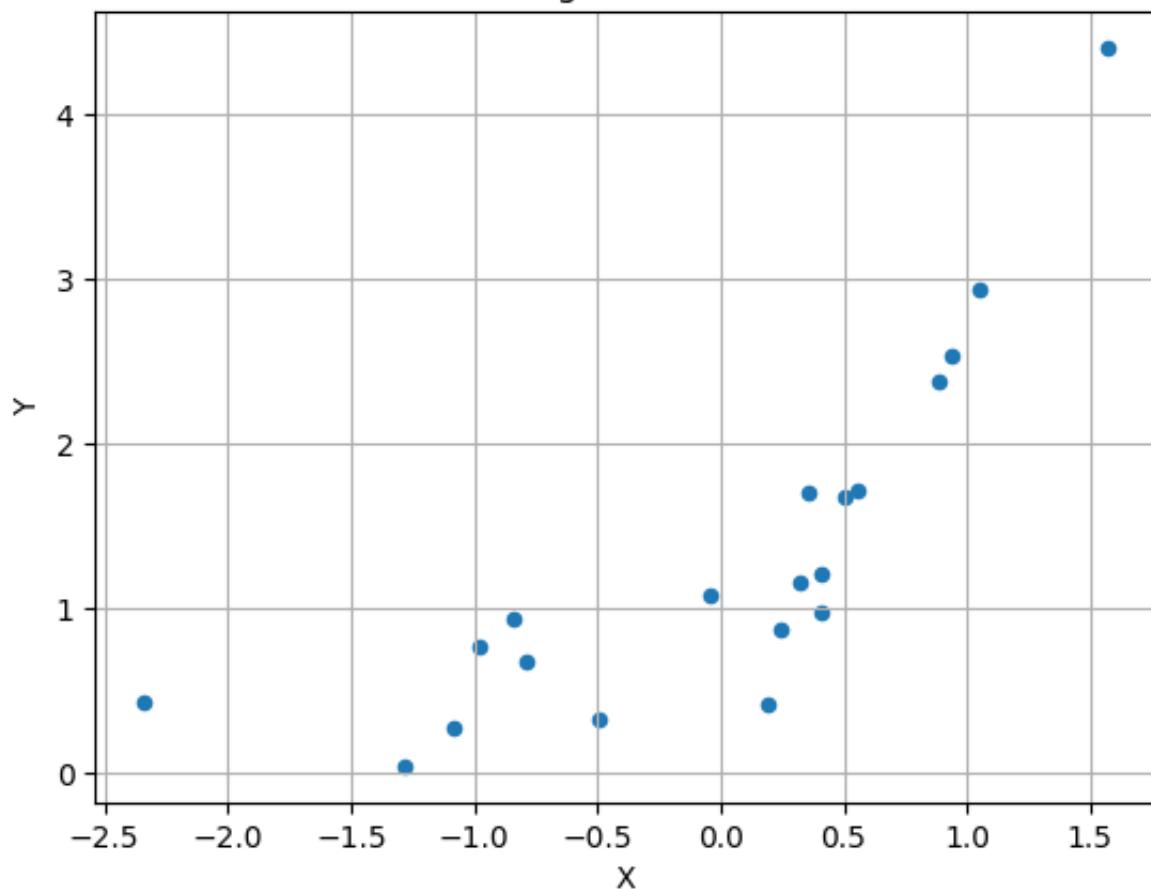
Question 3a

```
np.random.seed(20)
def generate_data(n):
    X = np.random.normal(0, 1, n)
    epsilon = np.random.normal(0, 0.25, n)
    Y = np.exp(X) + epsilon
    return X, Y
```

```
X_plot, Y_plot = generate_data(20)
plt.scatter(X_plot, Y_plot, s=20)
plt.title("Training Data (n=20)")
plt.xlabel("X")
plt.ylabel("Y")
plt.grid(True)
plt.show()
```



Training Data (n=20)



Question 3b

```
X_test, Y_test = generate_data(10000)
num_repeats = 100
results = {
    "linear": {"test": [], "loocv": [], "k2": [], "k5": []},
```

```
"poly": {"test": [], "loocv": [], "k2": [], "k5": []}  
}  
  
def loocv_error(X, Y):  
    model = LinearRegression().fit(X, Y)  
    Y_hat = model.predict(X)  
    H = X @ np.linalg.pinv(X.T @ X) @ X.T  
    leverages = np.diag(H)  
    errors = ((Y - Y_hat) / (1 - leverages))**2  
    return np.mean(errors)  
  
def k_fold_cv_error(X, Y, k):  
    kf = KFold(n_splits=k, shuffle=True, random_state=22)  
    errors = []  
    for train_idx, val_idx in kf.split(X):  
        model = LinearRegression().fit(X[train_idx], Y[train_idx])  
        Y_pred = model.predict(X[val_idx])  
        errors.append(mean_squared_error(Y[val_idx], Y_pred))  
    return np.mean(errors)  
  
poly = PolynomialFeatures(degree=4, include_bias=True)  
  
for _ in range(num_repeats):  
    X_train, Y_train = generate_data(20)  
    X_train = X_train.reshape(-1, 1)  
  
    # Linear  
    model = LinearRegression().fit(X_train, Y_train)  
    Y_pred = model.predict(X_test.reshape(-1, 1))  
    results['linear']['test'].append(mean_squared_error(Y_test, Y_pred))  
    results['linear']['loocv'].append(loocv_error(X_train, Y_train))  
    results['linear']['k2'].append(k_fold_cv_error(X_train, Y_train, 2))  
    results['linear']['k5'].append(k_fold_cv_error(X_train, Y_train, 5))  
  
    # Poly  
    X_train_poly = poly.fit_transform(X_train)  
    X_test_poly = poly.transform(X_test.reshape(-1, 1))  
    model_poly = LinearRegression().fit(X_train_poly, Y_train)  
    Y_pred_poly = model_poly.predict(X_test_poly)  
    results["poly"]["test"].append(mean_squared_error(Y_test, Y_pred_poly))  
    results["poly"]["loocv"].append(loocv_error(X_train_poly, Y_train))  
    results["poly"]["k2"].append(k_fold_cv_error(X_train_poly, Y_train, 2))  
    results["poly"]["k5"].append(k_fold_cv_error(X_train_poly, Y_train, 5))
```

```
summary_df = pd.DataFrame({
    "Test Error": [
        f"{np.mean(results['linear']['test']):.3f} ({np.std(results['linear']['test']):.3f})",
        f"{np.mean(results['poly']['test']):.3f} ({np.std(results['poly']['test']):.3f})"
    ],
    "LOOCV": [
        f"{np.mean(results['linear']['loocv']):.3f} ({np.std(results['linear']['loocv']):.3f})",
        f"{np.mean(results['poly']['loocv']):.3f} ({np.std(results['poly']['loocv']):.3f})"
    ],
    "2-Fold CV": [
        f"{np.mean(results['linear']['k2']):.3f} ({np.std(results['linear']['k2']):.3f})",
        f"{np.mean(results['poly']['k2']):.3f} ({np.std(results['poly']['k2']):.3f})"
    ],
    "5-Fold CV": [
        f"{np.mean(results['linear']['k5']):.3f} ({np.std(results['linear']['k5']):.3f})",
        f"{np.mean(results['poly']['k5']):.3f} ({np.std(results['poly']['k5']):.3f})"
    ],
}, index=["OLS on X", "OLS on poly"])

print("\nCross-Validation Summary Table Mean and STD:\n")
print(summary_df)
```



Cross-Validation Summary Table Mean and STD:

	Test Error	LOOCV	2-Fold CV	5-Fold CV
OLS on X	2.614(1.125)	2.278 (4.422)	2.792 (5.053)	2.614 (4.874)
OLS on poly	1.651(8.002)	5.188 (29.369)	27.270 (83.236)	4.624 (24.576)

```
summary_df = pd.DataFrame({  
    "Test Error": [  
        f"{np.mean(results['linear']['test']):.3f} ({np.std(results['linear']['test'])/np.sqrt(n)}),  
        f"{np.mean(results['poly']['test']):.3f} ({np.std(results['poly']['test'])/np.sqrt(n)})  
    ],  
    "LOOCV": [  
        f"{np.mean(results['linear']['loocv']):.3f} ({np.std(results['linear']['loocv'])/np.sqrt(n)}),  
        f"{np.mean(results['poly']['loocv']):.3f} ({np.std(results['poly']['loocv'])/np.sqrt(n)})  
    ],  
    "2-Fold CV": [  
        f"{np.mean(results['linear']['k2']):.3f} ({np.std(results['linear']['k2'])/np.sqrt(n)}),  
        f"{np.mean(results['poly']['k2']):.3f} ({np.std(results['poly']['k2'])/np.sqrt(n)})  
    ],  
    "5-Fold CV": [  
        f"{np.mean(results['linear']['k5']):.3f} ({np.std(results['linear']['k5'])/np.sqrt(n)}),  
        f"{np.mean(results['poly']['k5']):.3f} ({np.std(results['poly']['k5'])/np.sqrt(n)})  
    ],  
}, index=["OLS on X", "OLS on poly"])  
  
print("\nCross-Validation Summary Table Mean and standard Error:\n")  
print(summary_df)
```



Cross-Validation Summary Table Mean and standard Error:

	Test Error	LOOCV	2-Fold CV	5-Fold CV
OLS on X	2.614(0.113)	2.278 (0.442)	2.792 (0.505)	2.614 (0.487)
OLS on poly	1.651(0.800)	5.188 (2.937)	27.270 (8.324)	4.624 (2.458)

▼ Question 4

```
boston = pd.read_csv('Boston.csv')
boston.head()
```

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90

Next steps:

Generate code with boston

View recommended plots

New interactive sheet

```
len(boston)
```

506

```
X = boston[boston.columns[:-1]]
y = boston['medv']
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=40)
model = LinearRegression().fit(X_train, y_train)
y_pred = model.predict(X_test)
mse_lin = mean_squared_error(y_test, y_pred)
print(f"Linear Regression Test MSE: {mse_lin:.2f}")
```

Linear Regression Test MSE: 33.92

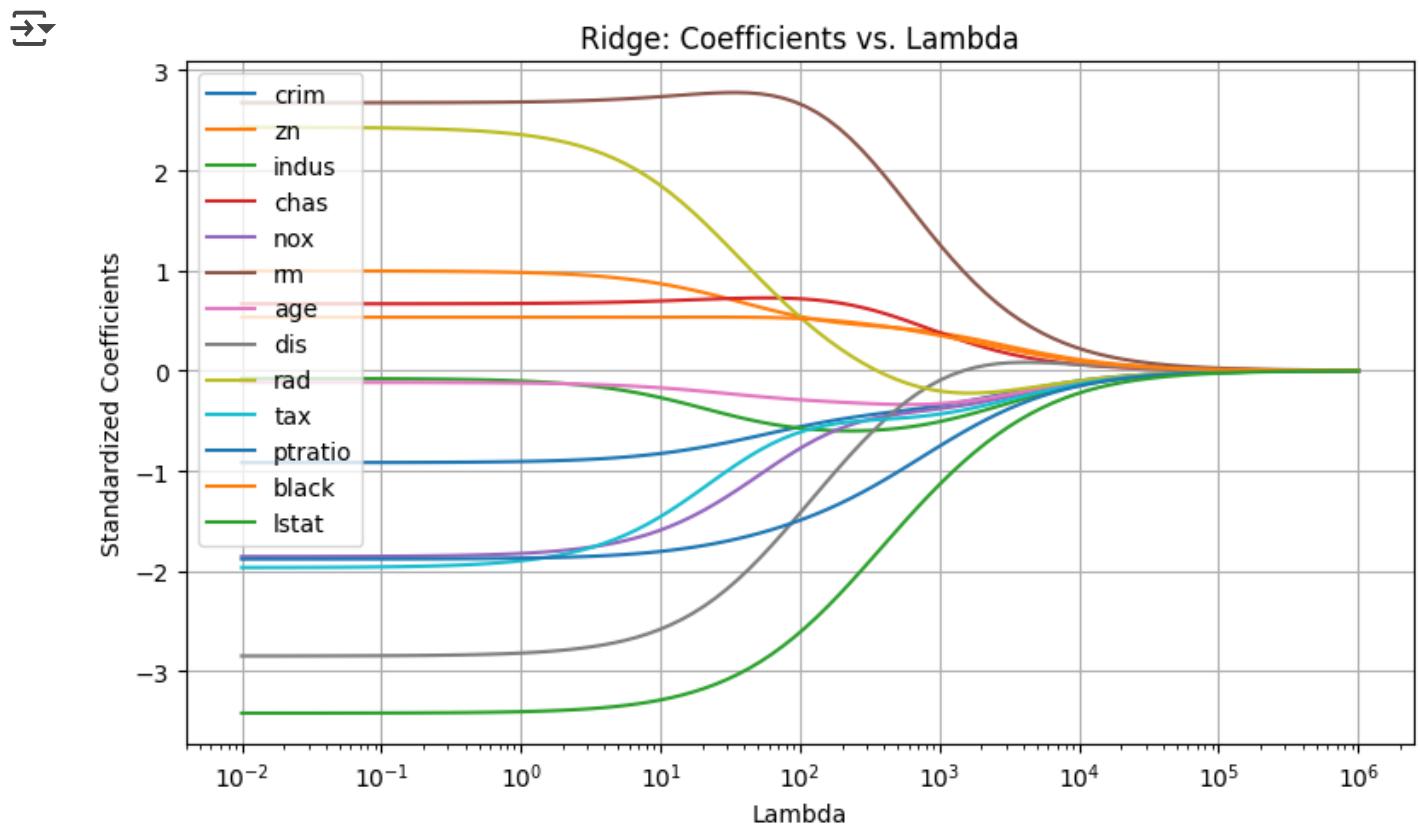
```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
lamb = np.logspace(-2, 6, 100)
coefs = []
for l in lamb:
    ridge = Ridge(alpha=l, fit_intercept=True)
    ridge.fit(X_train_scaled, y_train)
    coefs.append(ridge.coef_)
```

```
coefs = np.array(coefs)
```

```
plt.figure(figsize=(8, 5))
for i in range(coefs.shape[1]):
    plt.plot(lamb, coefs[:, i], label=X.columns[i])

plt.xscale('log')
plt.xlabel('Lambda')
plt.ylabel('Standardized Coefficients')
plt.title('Ridge: Coefficients vs. Lambda')
plt.grid(True)
plt.tight_layout()
plt.legend()
plt.show()
```



```
ridge_cv = RidgeCV(alphas=lamb, store_cv_values=True).fit(X_train_scaled,
                                                        y_train)
best_lambda_ridge = ridge_cv.alpha_
y_pred_ridge = ridge_cv.predict(X_test_scaled)
mse_ridge = mean_squared_error(y_test, y_pred_ridge)
print(f"Ridge Regression Best Lambda: {best_lambda_ridge:.4f}")
print(f"Ridge Regression Test MSE: {mse_ridge:.2f}")

→ Ridge Regression Best Lambda: 4.6416
Ridge Regression Test MSE: 34.09
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_ridge.py:2385
    warnings.warn(
```

```
# (e) Lasso regression with CV
lasso_cv = LassoCV(cv=5, random_state=42, max_iter=10000)
lasso_cv.fit(X_train_scaled, y_train)
best_lambda_lasso = lasso_cv.alpha_
y_pred_lasso = lasso_cv.predict(X_test_scaled)
mse_lasso = mean_squared_error(y_test, y_pred_lasso)
print(f"Lasso Regression Best Lambda: {best_lambda_lasso:.4f}")
print(f"Lasso Regression Test MSE: {mse_lasso:.2f}")
```

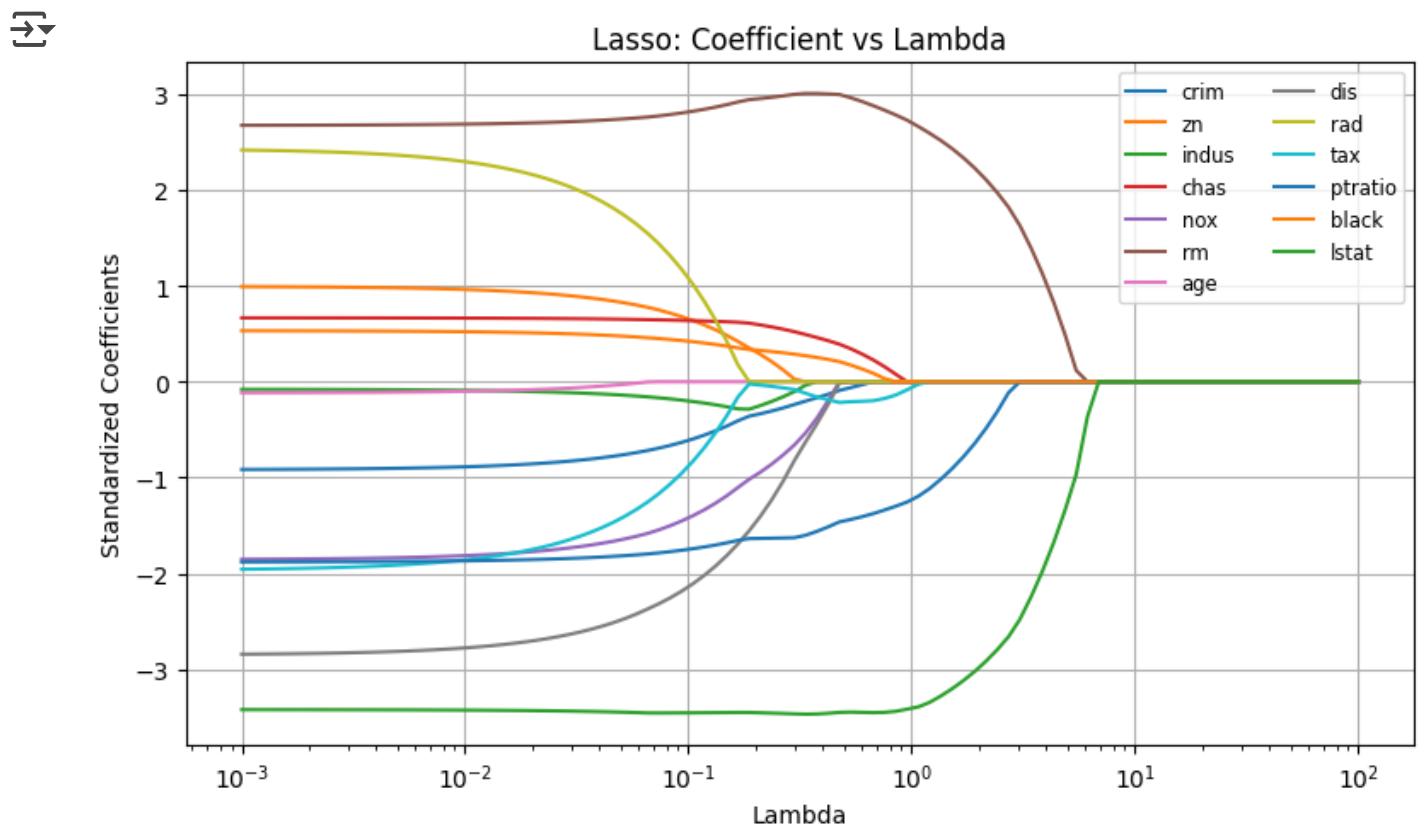
```
→ Lasso Regression Best Lambda: 0.0140
Lasso Regression Test MSE: 34.01
```

```
lasso_coefs = []
lamb = np.logspace(-3, 2, 100)
for l in lamb:
    lasso = Lasso(alpha=l, max_iter=10000)
    lasso.fit(X_train_scaled, y_train)
    lasso_coefs.append(lasso.coef_)

lasso_coefs = np.array(lasso_coefs)

plt.figure(figsize=(8,5))
for i in range(lasso_coefs.shape[1]):
    plt.plot(lamb, lasso_coefs[:, i], label=X.columns[i])

plt.xscale('log')
plt.xlabel('Lambda')
plt.ylabel('Standardized Coefficients')
plt.title('Lasso: Coefficient vs Lambda')
plt.grid(True)
plt.legend(loc='best', fontsize='small', ncol=2)
plt.tight_layout()
plt.show()
```



Start coding or generate with AI.

