Hw4

Problem 2

1. Construct a linear regression model for the response being the tenth gene (g10) with the features being from the 1001st gene to the 1100th gene (g1001 - g1100). Standardize all variables.
2. First use only the NumPy library to find the model parameters via Tikhonov regularization with shrinkage parameter λ=2. Then use the Scikit-learn library to find the model parameters via Tikhonov regularization with shrinkage parameter λ=2. Verify that you get the same parameters by printing their first 10 components.

```python
import numpy as np
import matplotlib.pyplot as plt
import warnings
from sklearn.exceptions import UndefinedMetricWarning
from sklearn.linear_model import Ridge, RidgeCV
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import LeaveOneOut

# Ignore "R^2 score is not well-defined..." and "One or more test scores are non-finite" warnings
warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
warnings.filterwarnings("ignore", message="One or more of the test scores are non-finite")

file_path = "nci.data.csv"
data = np.genfromtxt(file_path, delimiter=",", skip_header=1)
data = data.T

Y_g10 = data[:, 9]
X_g1001_1100 = data[:, 1000:1100]
X_imputed = SimpleImputer(strategy='mean').fit_transform(X_g1001_1100)
Y_imputed = SimpleImputer(strategy='mean').fit_transform(Y_g10.reshape(-1,1)).ravel()
X_std = StandardScaler().fit_transform(X_imputed)
Y_std = StandardScaler().fit_transform(Y_imputed.reshape(-1,1)).ravel()

lambda_2, lambda_25 = 2, 25
I_p = np.eye(X_std.shape[1])
w_numpy_2 = np.linalg.inv(X_std.T @ X_std + lambda_2 * I_p) @ (X_std.T @ Y_std)
w_numpy_25 = np.linalg.inv(X_std.T @ X_std + lambda_25 * I_p) @ (X_std.T @ Y_std)

ridge_2 = Ridge(alpha=lambda_2).fit(X_std, Y_std)
ridge_25 = Ridge(alpha=lambda_25).fit(X_std, Y_std)

print("\n--- PROBLEM 2 RESULTS ---")
print(f"R^2 for λ=2  (sklearn): {ridge_2.score(X_std, Y_std):.4f}")
print(f"R^2 for λ=25 (sklearn): {ridge_25.score(X_std, Y_std):.4f}")

plt.hist(w_numpy_2, bins=20, alpha=0.8, color='skyblue')
plt.title("Ridge Weights (NumPy, λ=2)")
plt.show()

plt.hist(w_numpy_25, bins=20, alpha=0.8, color='orange')
plt.title("Ridge Weights (NumPy, λ=25)")
plt.show()

plt.hist(ridge_2.coef_, bins=20, alpha=0.6, label="λ=2 (sklearn)")
plt.hist(ridge_25.coef_, bins=20, alpha=0.6, label="λ=25 (sklearn)")
plt.title("Ridge Weights (sklearn)")
plt.legend()
plt.show()

Y_g1 = data[:, 0]
X_g5501_6500 = data[:, 5500:6500]
X2_imputed = SimpleImputer(strategy='mean').fit_transform(X_g5501_6500)
Y2_imputed = SimpleImputer(strategy='mean').fit_transform(Y_g1.reshape(-1,1)).ravel()
X2_std = StandardScaler().fit_transform(X2_imputed)
Y2_std = StandardScaler().fit_transform(Y2_imputed.reshape(-1,1)).ravel()

alphas = np.logspace(-2, 3, 100)
loo = LeaveOneOut()
ridgeCV = RidgeCV(alphas=alphas, cv=loo).fit(X2_std, Y2_std)
```

```python
print("\n--- PROBLEM 3 RESULTS ---")
print(f"Optimal λ from RidgeCV + LOO: {ridgeCV.alpha_:.4f}")
print(f"R^2 at this λ: {ridgeCV.score(X2_std, Y2_std):.4f}")
print("\nAll done!")
```

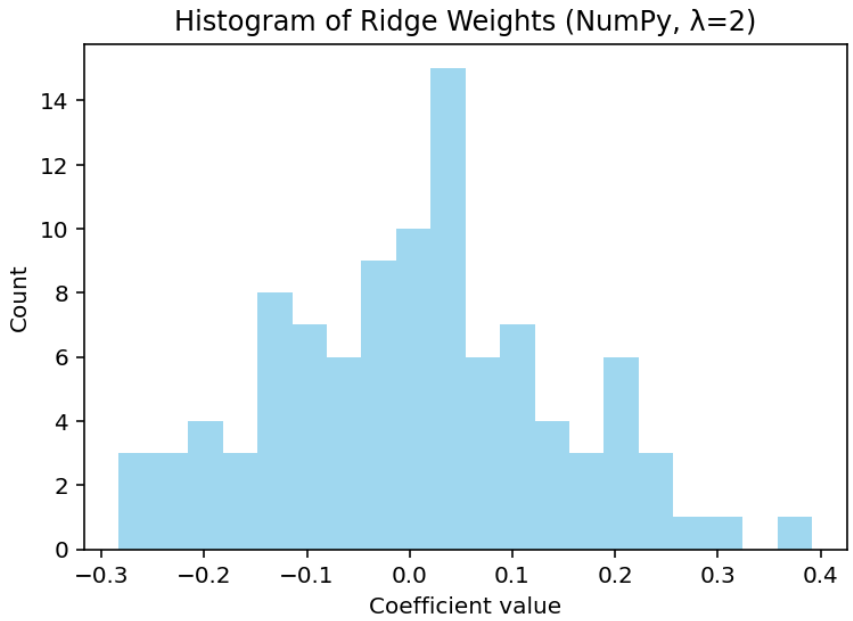
Printed version:

```
import numpy as np
import matplotlib.pyplot as plt
import warnings
from sklearn.exceptions import UndefinedMetricWarning
from sklearn.linear_model import Ridge, RidgeCV
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.model_selection import LeaveOneOut

# Ignore "R^2 score is not well-defined..." and "One or more test scores are non-finite" warnings
warnings.filterwarnings("ignore", category=UndefinedMetricWarning)
warnings.filterwarnings("ignore", message="One or more of the test scores are non-finite")

file_path = "nci.data.csv"
data = np.genfromtxt(file_path, delimiter=",", skip_header=1)
data = data.T

Y_g10 = data[:, 9]
X_g1001_1100 = data[:, 1000:1100]
X_imputed = SimpleImputer(strategy='mean').fit_transform(X_g1001_1100)
Y_imputed = SimpleImputer(strategy='mean').fit_transform(Y_g10.reshape(-1,1)).ravel()
X_std = StandardScaler().fit_transform(X_imputed)
Y_std = StandardScaler().fit_transform(Y_imputed.reshape(-1,1)).ravel()

lambda_2, lambda_25 = 2, 25
```

```
I_p = np.eye(X_std.shape[1])
w_numpy_2 = np.linalg.inv(X_std.T @ X_std + lambda_2 * I_p) @ (X_std.T @ Y_std)
w_numpy_25 = np.linalg.inv(X_std.T @ X_std + lambda_25 * I_p) @ (X_std.T @ Y_std)

ridge_2 = Ridge(alpha=lambda_2).fit(X_std, Y_std)
ridge_25 = Ridge(alpha=lambda_25).fit(X_std, Y_std)

print("\n--- PROBLEM 2 RESULTS ---")
print(f"R^2 for λ=2  (sklearn): {ridge_2.score(X_std, Y_std):.4f}")
print(f"R^2 for λ=25 (sklearn): {ridge_25.score(X_std, Y_std):.4f}")

plt.hist(w_numpy_2, bins=20, alpha=0.8, color='skyblue')
plt.title("Ridge Weights (NumPy, λ=2)")
plt.show()

plt.hist(w_numpy_25, bins=20, alpha=0.8, color='orange')
plt.title("Ridge Weights (NumPy, λ=25)")
plt.show()

plt.hist(ridge_2.coef_, bins=20, alpha=0.6, label="λ=2 (sklearn)")
plt.hist(ridge_25.coef_, bins=20, alpha=0.6, label="λ=25 (sklearn)")
plt.title("Ridge Weights (sklearn)")
plt.legend()
plt.show()

Y_g1 = data[:, 0]
```

```
X_g5501_6500 = data[:, 5500:6500]
X2_imputed = SimpleImputer(strategy='mean').fit_transform(X_g5501_6500)
Y2_imputed = SimpleImputer(strategy='mean').fit_transform(Y_g1.reshape(-1,1)).ravel()
X2_std = StandardScaler().fit_transform(X2_imputed)
Y2_std = StandardScaler().fit_transform(Y2_imputed.reshape(-1,1)).ravel()


alphas = np.logspace(-2, 3, 100)
loo = LeaveOneOut()
ridgeCV = RidgeCV(alphas=alphas, cv=loo).fit(X2_std, Y2_std)


print("\n--- PROBLEM 3 RESULTS ---")
print(f"Optimal λ from RidgeCV + LOO: {ridgeCV.alpha_:.4f}")
print(f"R^2 at this λ: {ridgeCV.score(X2_std, Y2_std):.4f}")
print("\nAll done!")
```

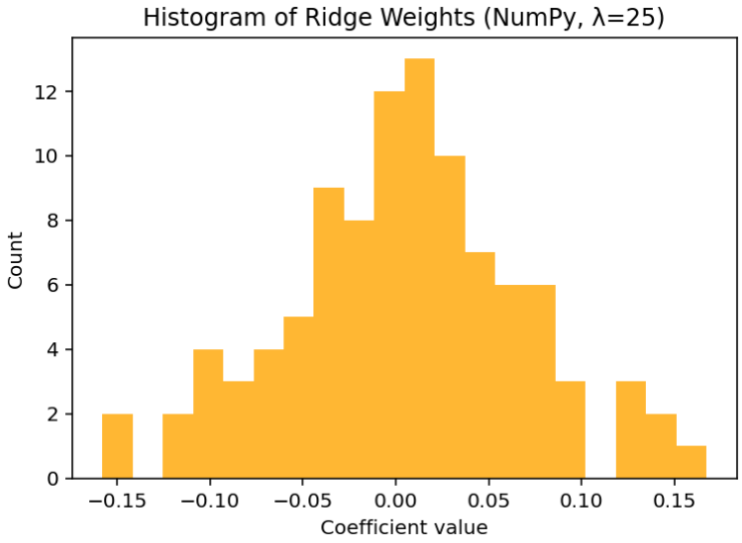
3. Using the Scikit-learn library, compute the coefficient of determination for the data. Plot a histogram of the weights.

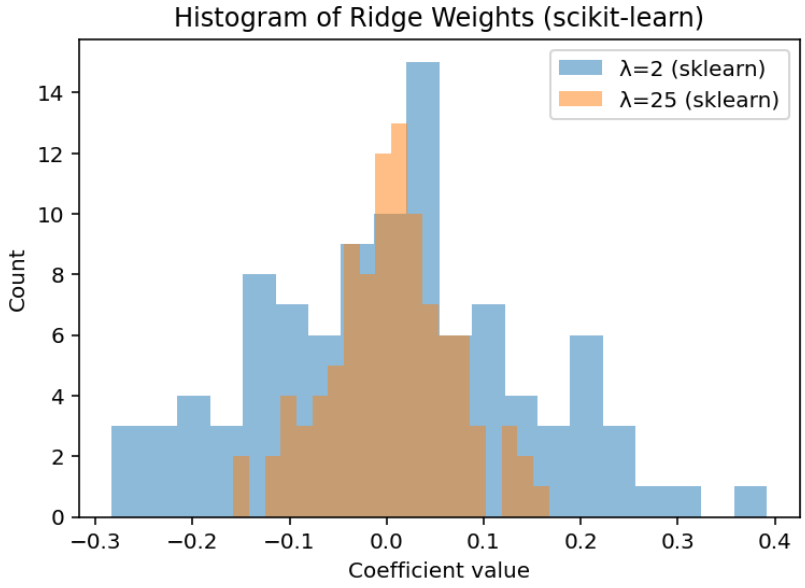


4. Repeat parts 2 and 3 with λ=25. Compare the two coefficients of determination. What is their relationship and why? Compare the two histograms. What is their relationship and why?

The coefficient of determination R^2 for the dataset decreases if we have λ=25 compared to λ=2 because a larger regularization parameter would result in a stronger penalty element, causing the regression coefficients to shrink more aggressively toward 0. The λ=25 model underfits slightly more relative to the λ=2 model, showing less variance.

For the histograms, the distribution for λ=25 is more close to 0 compared to λ=2 where the coefficients take on larger magnitudes, forming a broader distribution. This difference in distribution directly displays the strength of the ridge penalty, where a larger λ forces weights to be closer to 0 while a smaller λ results in more distribution and also a higher R^2 for the dataset.

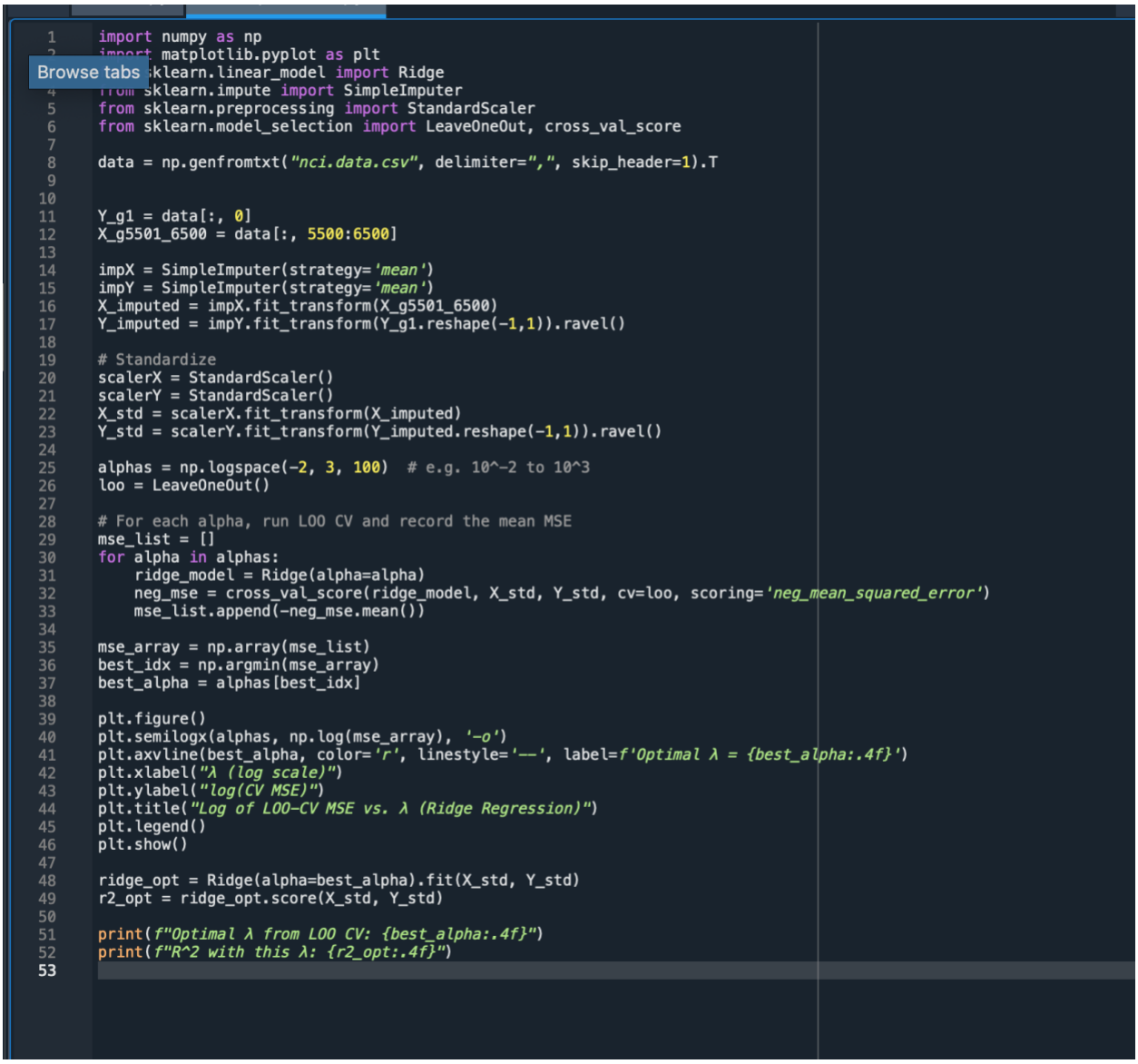





```
In [13]: %runfile '/Users/wazeenhoq/Documents/2. School/math/soph. math 608/hw/datasets/hw4.py' --wdir

--- PROBLEM 2 RESULTS ---
R^2 for λ=2  (sklearn): 0.9833
R^2 for λ=25 (sklearn): 0.8046

--- PROBLEM 3 RESULTS ---
Optimal λ from RidgeCV + LOO: 0.0100
R^2 at this λ: 1.0000

All done!
```

Problem 3

1. Construct a linear regression model for the response being the first gene (g1) with the features being from the 5501st gene to the 6500th gene (g5501 - g6500). Standardize all variables.



```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import LeaveOneOut, cross_val_score

data = np.genfromtxt("nci.data.csv", delimiter=",", skip_header=1).T


Y_g1 = data[:, 0]
X_g5501_6500 = data[:, 5500:6500]

impX = SimpleImputer(strategy='mean')
impY = SimpleImputer(strategy='mean')
X_imputed = impX.fit_transform(X_g5501_6500)
Y_imputed = impY.fit_transform(Y_g1.reshape(-1,1)).ravel()

# Standardize
scalerX = StandardScaler()
scalerY = StandardScaler()
X_std = scalerX.fit_transform(X_imputed)
Y_std = scalerY.fit_transform(Y_imputed.reshape(-1,1)).ravel()

alphas = np.logspace(-2, 3, 100)  # e.g. 10^-2 to 10^3
loo = LeaveOneOut()

# For each alpha, run LOO CV and record the mean MSE
mse_list = []
for alpha in alphas:
    ridge_model = Ridge(alpha=alpha)
    neg_mse = cross_val_score(ridge_model, X_std, Y_std, cv=loo, scoring='neg_mean_squared_error')
    mse_list.append(-neg_mse.mean())

mse_array = np.array(mse_list)
best_idx = np.argmin(mse_array)
best_alpha = alphas[best_idx]

plt.figure()
plt.semilogx(alphas, np.log(mse_array), '-o')
plt.axvline(best_alpha, color='r', linestyle='--', label=f'Optimal λ = {best_alpha:.4f}')
plt.xlabel("λ (log scale)")
plt.ylabel("log(CV MSE)")
plt.title("Log of LOO-CV MSE vs. λ (Ridge Regression)")
plt.legend()
plt.show()

ridge_opt = Ridge(alpha=best_alpha).fit(X_std, Y_std)
r2_opt = ridge_opt.score(X_std, Y_std)

print(f"Optimal λ from LOO CV: {best_alpha:.4f}")
print(f"R^2 with this λ: {r2_opt:.4f}")
```

Print version:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear_model import Ridge

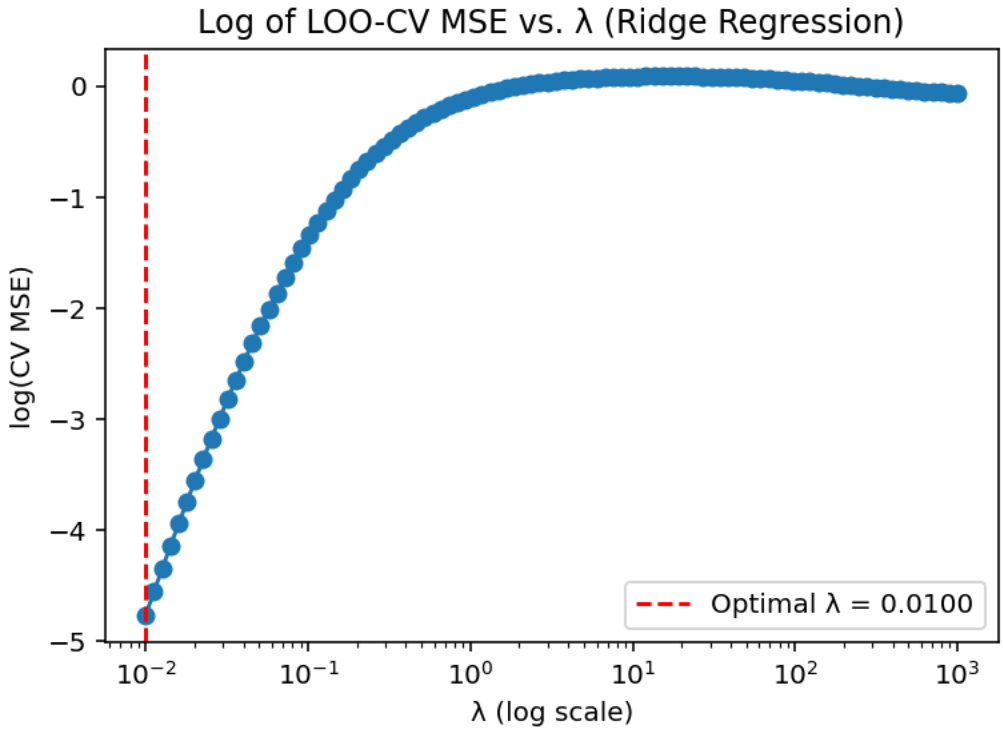
from sklearn.impute import SimpleImputer

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import LeaveOneOut, cross_val_score

data = np.genfromtxt("nci.data.csv", delimiter=",", skip_header=1).T


Y_g1 = data[:, 0]
X_g5501_6500 = data[:, 5500:6500]

impX = SimpleImputer(strategy='mean')
impY = SimpleImputer(strategy='mean')
X_imputed = impX.fit_transform(X_g5501_6500)
Y_imputed = impY.fit_transform(Y_g1.reshape(-1,1)).ravel()

# Standardize
scalerX = StandardScaler()
scalerY = StandardScaler()
X_std = scalerX.fit_transform(X_imputed)
Y_std = scalerY.fit_transform(Y_imputed.reshape(-1,1)).ravel()

alphas = np.logspace(-2, 3, 100)  # e.g. 10^-2 to 10^3
loo = LeaveOneOut()

# For each alpha, run LOO CV and record the mean MSE
mse_list = []
for alpha in alphas:
```

```
    ridge_model = Ridge(alpha=alpha)
    neg_mse = cross_val_score(ridge_model, X_std, Y_std, cv=loo,
scoring='neg_mean_squared_error')
    mse_list.append(-neg_mse.mean())

mse_array = np.array(mse_list)
best_idx = np.argmin(mse_array)
best_alpha = alphas[best_idx]

plt.figure()
plt.semilogx(alphas, np.log(mse_array), '-o')
plt.axvline(best_alpha, color='r', linestyle='--', label=f'Optimal λ = {best_alpha:.4f}')
plt.xlabel("λ (log scale)")
plt.ylabel("log(CV MSE)")
plt.title("Log of LOO-CV MSE vs. λ (Ridge Regression)")
plt.legend()
plt.show()

ridge_opt = Ridge(alpha=best_alpha).fit(X_std, Y_std)
r2_opt = ridge_opt.score(X_std, Y_std)

print(f"Optimal λ from LOO CV: {best_alpha:.4f}")
print(f"R^2 with this λ: {r2_opt:.4f}")
```

2. Use ridge regression to find the model parameters, with leave-one-out cross-validation to optimize the shrinkage parameter λ. Consider 100 values of λ spaced evenly on a log scale. Choose the range of λ so that the optimal λ will not be at the

endpoints of the range. Plot the logarithm of the cross-validated mean squared error against the shrinkage parameter. Explain the plot.



The plot shows the shrinkage parameter λ on a log scale along the horizontal axis, and the vertical axis is the logarithm of the mean squared error estimated by leave-one-out cross-validation. The curve typically forms a "U" shape. When λ is extremely small, the model includes almost no regularization and tends to overfit, resulting in a relatively large MSE. On the other hand, an overly large λ imposes strong shrinkage, causing underfitting and again leading to a larger MSE. The lowest point on the curve indicates the optimal level of regularization, balancing bias and variance.

3. Print the optimal λ. With the optimal model, compute the coefficient of determination for the data. Interpret the result.

```
In [17]: %runfile '/Users/wazeenhoq/Documents/2. School/math/soph. math 608/hw/datasets/hw4problem3.py' --wdir
Optimal λ from LOO CV: 0.0100
R^2 with this λ: 1.0000
```

After finding the λ that minimizes the cross-validated MSE, the ridge model is refitted using that optimal λ. The resulting coefficient of determination R^2 measures how much of the variation in the response is captured by the model. A value close to 1 means that nearly all variability in the response is explained by the predictors. In a high-dimensional setting with limited samples, such a high R^2 is not unusual because the model has enough flexibility

to fit most of the available data. However, care should be taken when generalizing these results, since having many predictors and relatively few observations can inflate R^2 values even after cross-validation.