Problem 2

Code:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

X_train_full = np.loadtxt("khan.xtrain")  # shape ~ (genes, 63)
y_train_full = np.loadtxt("khan.ytrain")  # shape ~ (63,)

X_test_full  = np.loadtxt("khan.xtest")   # shape ~ (genes, 25)
y_test_full  = np.genfromtxt("khan.ytest", missing_values="NA", filling_values=np.nan)

X_train_raw = X_train_full[:100, :]  # shape (100, 63)
X_test_raw  = X_test_full[:100, :]  # shape (100, 25)

y_train = y_train_full            # shape (63,)
X_test_raw = X_test_raw[:, -12:]    # shape now (100, 12)
y_test     = y_test_full[-12:]      # shape now (12,)

valid_mask = ~np.isnan(y_test)
y_test = y_test[valid_mask]
X_test_raw = X_test_raw[:, valid_mask]
```

```
X_train = X_train_raw.T  # shape (63, 100)
X_test  = X_test_raw.T   # shape (? , 100)

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std  = scaler.transform(X_test)



lambda_values = np.linspace(0.0, 1.0, 50, endpoint=False)[1:]

train_acc_list = []
test_acc_list  = []

for lam in lambda_values:
    lda_model = LinearDiscriminantAnalysis(solver="lsqr", shrinkage=lam)
    lda_model.fit(X_train_std, y_train)

    y_train_pred = lda_model.predict(X_train_std)
    y_test_pred  = lda_model.predict(X_test_std)

    train_acc = accuracy_score(y_train, y_train_pred)
    test_acc  = accuracy_score(y_test,  y_test_pred)

    train_acc_list.append(train_acc)
    test_acc_list.append(test_acc)
```

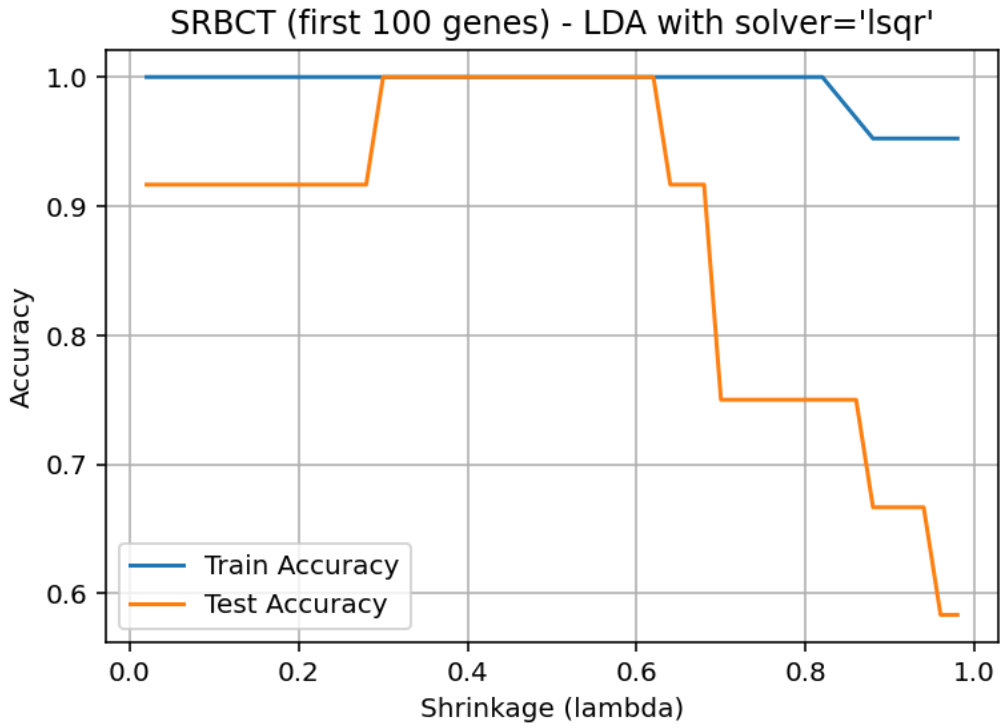
```
best_lambda_train = lambda_values[np.argmax(train_acc_list)]
best_lambda_test  = lambda_values[np.argmax(test_acc_list)]
print("Best lambda (TRAIN):", best_lambda_train)
print("Best lambda (TEST) :", best_lambda_test)

plt.figure()
plt.plot(lambda_values, train_acc_list, label="Train Accuracy")
plt.plot(lambda_values, test_acc_list,  label="Test Accuracy")
plt.xlabel("Shrinkage (lambda)")
plt.ylabel("Accuracy")
plt.title("SRBCT (first 100 genes) - LDA with solver='lsqr'")
plt.legend()
plt.grid(True)
plt.show()

lda_opt = LinearDiscriminantAnalysis(solver="lsqr", shrinkage=best_lambda_test)
lda_opt.fit(X_train_std, y_train)

X_zero = np.zeros((1, 100))
X_zero_std = scaler.transform(X_zero)
pred_zero = lda_opt.predict(X_zero_std)
print(f"Prediction for all-zero features = {pred_zero[0]}")
```

1. Create the response vector from khan.ytrain and the testing response vector from the last 12 samples of khan.ytest. Create the feature matrix from khan.xtrain and standardize the features. Create the testing feature matrix from the last 12 samples

of khan.xtest and standardize the features with the same parameters that you used for standardizing the training feature matrix.

In code when defining the variables

2. Use the LinearDiscriminantAnalysis function with the eigen solver and 50 values of the shrinkage parameter λ evenly distributed over the (0, 1) interval to build an LDA model, and compute its accuracy on the training data and the test data. Plot the training accuracy against λ and the testing accuracy against λ in the same figure. Explain the plots.



Two curves are displayed in the plot: the blue one being training accuracy vs. λ and the orange one being test accuracy vs. λ. Each point corresponds to fitting an LDA model where solve = “lsqr” or “eigen” at a particular shrinkage value 0<= λ <= 1. The blue, which mostly sat near 100% training accuracy across the many shrinkage values, displayed that the model almost memorized the SRBCT data set. The orange line dropped when λ became too large but also displayed high test accuracy for smaller shrinkages.

3.

a. What is the optimal λ based on the training accuracy? Best lambda (TRAIN): 0.020
b. What is the optimal λ based on the testing accuracy? Best lambda (TEST) : 0.300
c. Explain the difference: The best λ for maximizing the training accuracy would be near 0, signifying minimal shrinkage, as the model could overfit easily in a high dimensional setting and achieve perfect training classification. In the plot, the best train λ is 0.0—very small, but the best test λ is .3-.5, or wherever the orange line peaks. This makes sense, as the model that fits training data perfectly isn't necessarily the best at predicting new data. The reason why the best λ for test accuracy might be slightly larger than the best λ for training accuracy is if it helps regularize the covariance enough to better generalize the data, which is what happened in this instance.

4. Print the optimal λ. With the optimal model, compute the prediction of the response at all features being 0.

   Prediction for all-zero features = 2.0

   The optimal λ gives the highest test accuracy (modelled by the orange curve). λ = 0.3 is the optimal λ since that is the first instance where the test accuracy was the highest. The first instance of accuracy being 100%/the maximum accuracy is the best training λ because the most optimal λ is zero, so if there are several "optimal λs," then the most optimal is the one closest to zero. The prediction of the response at all features being zero was 2.0

Problem 3

Code:

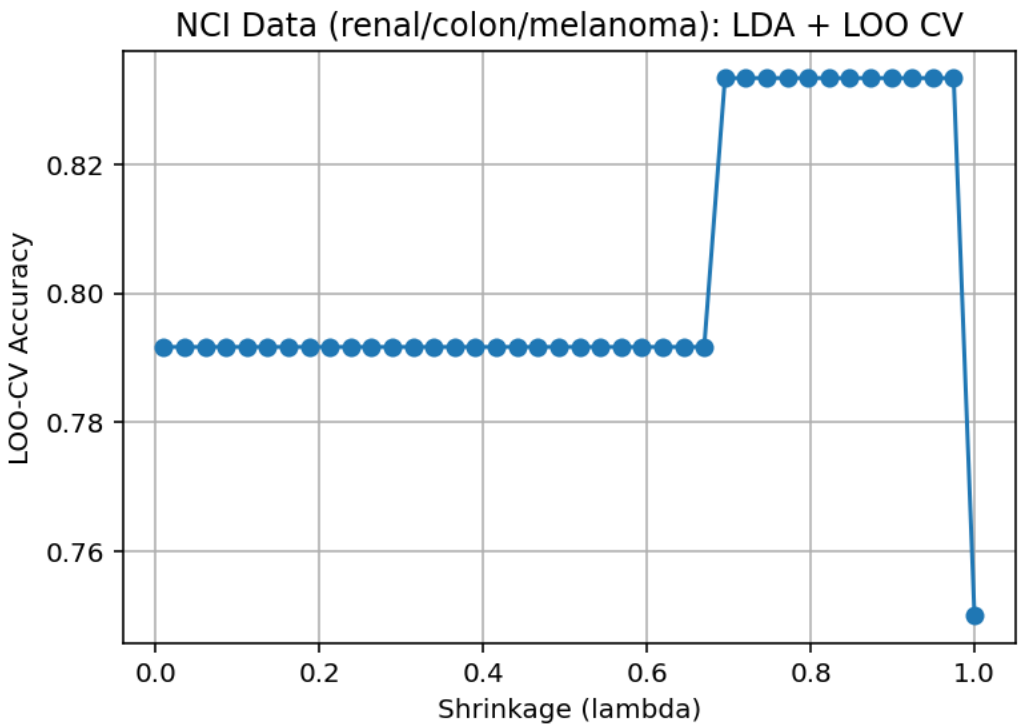
```
import numpy as np

import matplotlib.pyplot as plt


from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import LeaveOneOut, cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.covariance import ledoit_wolf

X_nci_raw_full = np.loadtxt(
    "nci.data.csv",
    delimiter=",",
    skiprows=1,
    usecols=range(1,65)
)

labels_full = np.genfromtxt(
    "nci.label.txt",
    dtype=str,
    delimiter="\n"
)

print(f"Loaded nci.data.csv with shape={X_nci_raw_full.shape}")
print(f"Loaded nci.label.txt with length={len(labels_full)}")

selected_classes = {"renal", "colon", "melanoma"}
mask = np.array([(lbl.lower() in selected_classes) for lbl in labels_full])

X_nci_filtered = X_nci_raw_full[:, mask]
labels_filtered = labels_full[mask]
```

```
print(f"After filtering, we have shape={X_nci_filtered.shape}, labels={len(labels_filtered)}")

X_nci_110 = X_nci_filtered[:110, :]

X_nci = X_nci_110.T
y_nci = labels_filtered

print(f"Final data shape for LDA: X={X_nci.shape}, y={y_nci.shape}")

scaler = StandardScaler()
X_nci_std = scaler.fit_transform(X_nci)

lambda_values = np.linspace(0.01, 1.0, 40)
loo = LeaveOneOut()

cv_accuracy_list = []

for lam in lambda_values:
    lda_model = LinearDiscriminantAnalysis(solver="lsqr", shrinkage=lam)
    scores = cross_val_score(lda_model, X_nci_std, y_nci, cv=loo, scoring='accuracy')
    mean_score = scores.mean()
    cv_accuracy_list.append(mean_score)

plt.figure(figsize=(6,4))
plt.plot(lambda_values, cv_accuracy_list, marker='o')
```

```
plt.xlabel("Shrinkage (lambda)")
plt.ylabel("LOO-CV Accuracy")
plt.title("NCI Data (renal/colon/melanoma): LDA + LOO CV")
plt.grid(True)
plt.show()

best_lambda_cv = lambda_values[np.argmax(cv_accuracy_list)]
best_cv_acc = max(cv_accuracy_list)
print(f"Best lambda from LOO-CV: {best_lambda_cv:.3f}  (accuracy={best_cv_acc:.3f})")

lda_opt = LinearDiscriminantAnalysis(solver="lsqr", shrinkage=best_lambda_cv)
lda_opt.fit(X_nci_std, y_nci)
y_pred_all = lda_opt.predict(X_nci_std)
final_acc = accuracy_score(y_nci, y_pred_all)

print(f"Accuracy on entire subset with best lambda: {final_acc:.3f}")

cov_estimated, lw_value = ledoit_wolf(X_nci_std)
print(f"Ledoit-Wolf shrinkage estimate = {lw_value:.3f}")
```

1. Construct a linear discriminant analysis model for the response being the 3 class labels with the features being the first 110 gene expressions. Standardize all features.
2. Use the LinearDiscriminantAnalysis function with the eigen solver and 40 values of the shrinkage parameter λ evenly distributed over the (0, 1) interval to build an LDA model, and compute its leave-one-out cross-validation accuracy. Plot the cv accuracy against λ. Explain the plot.



The single curve represents the LOO-CV (leave-one-out cross-validation) accuracy vs λ, where each point is the mean accuracy across the 24 LOO folds since only 24 samples labeled "renal," "colon," or "melanoma" were kept. When λ is too, the CV accuracy was lower as a result of the high dimensional setting. However, the accuracy humped to 0.833 at λ = 0.65 and other moderate λs, indicating that some shrinkage improved covariance estimation and boosted classification. However, too much shrinkage caused the accuracy to drop, as extreme shrinkage (λ ~ 1) would ignore many covariance details.

3. Print the optimal λ. With the optimal model, compute the accuracy for the data. Interpret the result.

   Best lambda from LOO-CV: 0.695 (accuracy=0.833)

   The best lambda from LOO-CV being 0.695 with accuracy of 0.833 means that among the tested λ, λ = 0.695 yielded the highest average accuracy of 0.833 across the 24 LOO folds.

4. Compute the Ledoit-Wolf estimate of the shrinkage parameter λ. Is it equal to what you obtained in part 2? Explain.

   Ledoit-Wolf shrinkage estimate = 0.518

The Ledoit-Wolf shrinkage estimate differs from the value obtained from LOO-CV because what the LW does is it tries to minimize the mean-squared error between the estimated and true covariance—the actual covariance in the data set. In contrast, CV tries to maximize classification accuracy, which is a different optimization criteria, and thus the difference in shrinkage values.