Codes For Final Assignment

Bamdad Booyeh 25/05/2024

IRIS DATASET

Listing 1: Matlab Code.

```
% Load the data
load('IrisDataAnnotated.mat');

% Set random number generator for reproducibility
rng('default');
s = rng;

% Define data matrix X (assuming X is loaded from the .mat file)
X = ...; % Load or assign the data matrix X here

% Dimension and size of the data matrix
dim = size(X, 1);
[n, m] = size(X);
p = m;

% Distance matrix D using L1 norm
D = zeros(p, p);
for i = 1:p
    for j = i:p
        if i == j
            D(i, j) = 0;
        else
            D(i, j) = norm(X(:, i) - X(:, j), 1); % L1 norm
        end
    end
end
D = D + D'



I_m = [33, 99, 101];


figure();
plot(X(1, :), X(2, :), 'bo', 'MarkerSize', 8);
```

```
hold on;
plot(X(1, I_m), X(2, I_m), 'mp', 'MarkerSize', 12, 'LineWidth', 2);
legend('Data', 'Random Centroids');

% Initialize parameters for k-medoids
Err = 1;
itmax = 100;
tol = 1.0e-10;
iter = 0;

% Initial distance to medoids
D_m = D(:, I_m);
[q, I_assign] = min(D_m, [], 2);
Q = sum(q);
k = length(I_m);
oldI_m = I_m;

% Main loop for k-medoids algorithm
while Err > tol && iter < itmax
    iter = iter + 1;
    clear I_m;

    for ell = 1:k
        I_ell = find(I_assign == ell);
        D_ell = D(I_ell, I_ell);
        [~, j] = min(sum(D_ell, 1));
        I_m(ell) = I_ell(j);
    end

    newI_m = I_m;

    % Recompute assignments
    D_m = D(:, I_m);
    [q, I_assign] = min(D_m, [], 2);
    Q_new = sum(q);

    % Check for convergence
    Err = abs(Q_new - Q);
    Q = Q_new;
    oldI_m = newI_m;
end

% Assign data to clusters
X_l = cell(1, k);
for j = 1:k
    X_l{j} = X(:, I_assign == j);
end
```

```
83
84 % Plot final clusters and centroids
85 figure();
86 plot(X(1, :), X(2, :), 'bo', 'MarkerSize', 8);
87 hold on;
88 plot(X(1, I_m), X(2, I_m), 'mp', 'MarkerSize', 12, 'LineWidth', 2);
89 legend('Data', 'Final Centroids');
90
91 % 3D scatter plot of clusters
92 figure();
93 colors = {'cyan', 'red', 'green'};
94 for j = 1:k
95     scatter3(X_l{j}(1, :), X_l{j}(2, :), X_l{j}(3, :), [], colors{j}, '↩
          filled');
96     hold on;
97 end
98 scatter3(X(1, I_m), X(2, I_m), X(3, I_m), 100, 'rx', 'LineWidth', 5);
99 title('Final clustering K Medoids');
100 legend('Cluster 1 (Cyan)', 'Cluster 2 (Red)', 'Cluster 3 (Green)', '↩
       Medoids');
101
102 % Plot each cluster
103 figure();
104 for j = 1:k
105     scatter3(X_l{j}(1, :), X_l{j}(2, :), X_l{j}(3, :), [], colors{j}, '↩
          filled');
106     hold on;
107 end
108 scatter3(X(1, I_m), X(2, I_m), X(3, I_m), 100, 'rx', 'LineWidth', 5);
109 title('Final clustering K Medoids');
110 legend('Cluster 1 (Cyan)', 'Cluster 2 (Blue)', 'Cluster 3 (Green)', '↩
       Medoids');
111
112 % Confusion matrix plot (assuming I is the true label vector)
113 figure();
114 cm = confusionchart(I, I_assign);
115
116 % Load the data
117 load('IrisDataAnnotated.mat');
118
119 % Transpose the data
120 X = X';
121
122 % Perform k-means clustering
123 rng('default'); % For reproducibility
124 [idx, C] = kmeans(X, 3);
125
```

```
% Visualize the clusters and centroids in 2D
figure;
plot(X(idx == 1, 1), X(idx == 1, 2), 'r.', 'MarkerSize', 9);
hold on;
plot(X(idx == 2, 1), X(idx == 2, 2), 'b.', 'MarkerSize', 9);
plot(X(idx == 3, 1), X(idx == 3, 2), 'g.', 'MarkerSize', 9);
plot(C(:, 1), C(:, 2), 'mp', 'MarkerSize', 10, 'LineWidth', 1.5);
legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Centroids', 'Location', 'NW↩
    ');
title('Cluster Assignments and Centroids');
hold off;

% Visualize the clusters and centroids in 3D
figure;
scatter3(X(idx == 1, 1), X(idx == 1, 2), X(idx == 1, 3), 'r.');
hold on;
scatter3(X(idx == 2, 1), X(idx == 2, 2), X(idx == 2, 3), 'b.');
scatter3(X(idx == 3, 1), X(idx == 3, 2), X(idx == 3, 3), 'g.');
scatter3(C(:, 1), C(:, 2), C(:, 3), 200, 'mp', 'filled');
legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster Mean', 'Location', ↩
    'NW');
title('K-means Clusters and Centroids');
xlabel('Feature 1');
ylabel('Feature 2');
zlabel('Feature 3');
hold off;

% Plot the confusion matrix
figure;
cm = confusionchart(I, idx);
cm.Title = 'Confusion Matrix for K-means Clustering';
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';


load('IrisDataAnnotated.mat'); % X = data matrix, I = annotation vector

% Center the data within each class
X1_centered = X(:, I == 1) - mean(X(:, I == 1), 2);
X2_centered = X(:, I == 2) - mean(X(:, I == 2), 2);
X3_centered = X(:, I == 3) - mean(X(:, I == 3), 2);

% Calculate the within-class scatter matrix
Sw = X1_centered * X1_centered' + X2_centered * X2_centered' + X3_centered↩
    * X3_centered';

% Compute the between-class scatter matrix
```

```
mu = mean(X, 2);
Sb = length(find(I == 1)) * (mean(X(:, I == 1), 2) - mu) * (mean(X(:, I == 1), 2) - mu)' + ...
     length(find(I == 2)) * (mean(X(:, I == 2), 2) - mu) * (mean(X(:, I == 2), 2) - mu)' + ...
     length(find(I == 3)) * (mean(X(:, I == 3), 2) - mu) * (mean(X(:, I == 3), 2) - mu)';

% Perform eigenvalue decomposition to find the LDA directions
[V, D] = eig(pinv(Sw) * Sb);
[~, indices] = sort(diag(D), 'descend');
eigenvectors_lda = V(:, indices);

% Project the data onto the first two LDA directions
LDA_projection = eigenvectors_lda(:, 1:2)' * X;

% Perform PCA on the data matrix
[coeff, score, ~, ~, explained] = pca(X');

% Project the data onto the first two PCA directions
PCA_projection = coeff(:, 1:2)' * X;

% Plot the data projected on the first two LDA directions
figure;
subplot(1, 2, 1);
scatter(LDA_projection(1, I == 1), LDA_projection(2, I == 1), 100, 'r.', 'DisplayName', 'Iris setosa');
hold on;
scatter(LDA_projection(1, I == 2), LDA_projection(2, I == 2), 100, 'g.', 'DisplayName', 'Iris versicolor');
scatter(LDA_projection(1, I == 3), LDA_projection(2, I == 3), 100, 'b.', 'DisplayName', 'Iris virginica');
title('LDA Projection');
xlabel('LDA Component 1');
ylabel('LDA Component 2');
legend('Location', 'best');
grid on;
set(gca, 'FontSize', 12);

% Plot the data projected on the first two PCA directions
subplot(1, 2, 2);
scatter(PCA_projection(1, I == 1), PCA_projection(2, I == 1), 100, 'r.', 'DisplayName', 'Iris setosa');
hold on;
scatter(PCA_projection(1, I == 2), PCA_projection(2, I == 2), 100, 'g.', 'DisplayName', 'Iris versicolor');
scatter(PCA_projection(1, I == 3), PCA_projection(2, I == 3), 100, 'b.', '
```

```
    DisplayName', 'Iris virginica');
title('PCA Projection');
xlabel('PCA Component 1');
ylabel('PCA Component 2');
legend('Location', 'best');
grid on;
set(gca, 'FontSize', 12);

% Classification using LDA projection
class_means_lda = [mean(LDA_projection(:, I == 1), 2), mean(LDA_projection(:, I == 2), 2), mean(LDA_projection(:, I == 3), 2)];
predicted_labels_lda = zeros(1, size(X, 2));

for i = 1:size(X, 2)
    distances = sum((LDA_projection(:, i) - class_means_lda).^2, 1);
    [~, predicted_labels_lda(i)] = min(distances);
end

% Classification using PCA projection
class_means_pca = [mean(PCA_projection(:, I == 1), 2), mean(PCA_projection(:, I == 2), 2), mean(PCA_projection(:, I == 3), 2)];
predicted_labels_pca = zeros(1, size(X, 2));

for i = 1:size(X, 2)
    distances = sum((PCA_projection(:, i) - class_means_pca).^2, 1);
    [~, predicted_labels_pca(i)] = min(distances);
end

% Add confusion charts
figure;
subplot(1, 2, 1);
confusionchart(I, predicted_labels_lda);
title('Confusion Matrix for LDA Classification');

subplot(1, 2, 2);
confusionchart(I, predicted_labels_pca);
title('Confusion Matrix for PCA Classification');
```

BIOPSY DATASET

Listing 2: Matlab Code.

```

% Load the data
```

```
load('BiopsyData.mat');

% Clean the data by removing columns with any NaN values
X_clean = X(:, all(~isnan(X)));

% Set random number generator for reproducibility
rng(1);

% Get the size of the cleaned data
[n, m] = size(X_clean);

% Initialize distance matrix
D = zeros(m, m);

% Define the norm type (Euclidean norm)
norm_type = 2;

% Compute the pairwise distance matrix using the specified norm
for i = 1:m
    for j = i:m
        if i == j
            D(i, j) = 0;
        else
            D(i, j) = norm(X_clean(:, i) - X_clean(:, j), norm_type);
        end
    end
end
D = D + D'; % Make the distance matrix symmetric

% Initial random medoid indices
I_m = [180, 356];

% Plot initial data and random centroids in 2D
figure;
plot(X_clean(1, :), X_clean(2, :), 'bo', 'MarkerSize', 8);
hold on;
plot(X_clean(1, I_m), X_clean(2, I_m), 'mp', 'MarkerSize', 12, 'LineWidth', 2);
legend('Data', 'Random Centroids');
title('Initial Random Centroids');
hold off;

% Parameters for k-medoids
Err = 1;
itmax = 100;
tol = 1.0e-10;
iter = 0;
```

```
49
50 % Compute initial distance matrix for medoids
51 D_m = D(:, I_m);
52 [q, I_assign] = min(D_m, [], 2);
53 Q = sum(q);
54
55 k = length(I_m);
56 oldI_m = I_m;
57 clear I_m;
58
59 % Main loop for k-medoids algorithm
60 while Err > tol && iter < itmax
61     iter = iter + 1;
62
63     for ell = 1:k
64         I_ell = find(I_assign == ell);
65         D_ell = D(I_ell, I_ell);
66         s = sum(D_ell, 2);
67         [qq(ell), j] = min(s);
68         I_m(ell) = I_ell(j);
69     end
70
71     D_m = D(:, I_m);
72     [q, I_assign] = min(D_m, [], 2);
73     Q_new = sum(q);
74
75     Err = abs(Q - Q_new);
76     Q = Q_new;
77 end
78
79 % Assign data to clusters
80 X_l = cell(1, k);
81 for j = 1:k
82     X_l{j} = X_clean(:, I_assign == j);
83 end
84
85 % Plot final medoids in 2D
86 figure;
87 plot(X_clean(1, :), X_clean(2, :), 'bo', 'MarkerSize', 8);
88 hold on;
89 plot(X_clean(1, I_m), X_clean(2, I_m), 'mp', 'MarkerSize', 12, 'LineWidth'↩
       , 2);
90 legend('Data', 'Final Medoids (Euclidean Norm)');
91 title('Final Medoids');
92 hold off;
93
94 % Plot final clusters in 2D
```

```
figure;
hold on;
scatter(X_l{1}(1, :), X_l{1}(2, :), 'blue', 'filled');
scatter(X_l{2}(1, :), X_l{2}(2, :), 'red', 'filled');
plot(X_clean(1, I_m), X_clean(2, I_m), 'mp', 'MarkerSize', 12, 'LineWidth', 2);
title('Final Clustering K-Medoids (Euclidean Norm)');
legend('Cluster 1', 'Cluster 2', 'Medoids');
hold off;

% Plot each cluster in 3D
figure;
hold on;
colors = {'blue', 'red', 'green'}; % Update the colors if more clusters
for j = 1:k
    scatter3(X_l{j}(1, :), X_l{j}(2, :), X_l{j}(3, :), 36, colors{j}, 'filled');
end
% Plot medoids in 3D
scatter3(X_clean(1, I_m), X_clean(2, I_m), X_clean(3, I_m), 100, 'kx', 'LineWidth', 5);
title('Final Clustering K-Medoids (3D Visualization)');
legend('Cluster 1', 'Cluster 2', 'Medoids'); % Update legend for more clusters
grid on;
view(3);
hold off;

% Assuming I is the true label vector
figure;
cm = confusionchart(I, I_assign);
cm.Title = 'Confusion Matrix for K-medoids Clustering';
cm.RowSummary = 'row-normalized';
cm.ColumnSummary = 'column-normalized';
```

CONGRESSVOTE DATASET

Listing 3: Matlab Code.

```
% Load the data
load('CongressionalVote.mat');

% Remove rows with all zero values
X(sum(X ~= 0, 2) == 0, :) = [];
```

```
I(sum(X ~= 0, 2) == 0) = [];

% Handle missing data by replacing zeros with NaN
X(X == 0) = NaN;

% Replace NaN values with the mean of the respective feature
for i = 1:size(X, 2)
    nan_idx = isnan(X(:, i));
    X(nan_idx, i) = mean(X(~nan_idx, i));
end

% Perform Linear Discriminant Analysis (LDA)
lda = fitcdiscr(X, I);

% Predict using LDA
lda_pred = predict(lda, X);

% Plot the confusion matrix for LDA
figure();
cm_lda = confusionchart(I, lda_pred);
title('Confusion Chart (LDA)');

% Transform the data using LDA coefficients for visualization
X_lda = X * lda.Coeffs(1, 2).Linear;

% Since LDA reduces to a single dimension, plot in 2D for clarity
X_lda_2D = [X_lda X_lda];

% Plot the LDA transformed data
figure;
gscatter(X_lda_2D(:,1), X_lda_2D(:,2), I);
title('LDA of Congressional Vote Data');
xlabel('LDA Dimension 1');
ylabel('LDA Dimension 2');
grid on;

% Customize the plot for clarity
legend('Democrat', 'Republican');

% Calculate and display sensitivity and specificity
sensitivity = sum((I == 1) & (lda_pred == 1)) / sum(I == 1);
specificity = sum((I == 0) & (lda_pred == 0)) / sum(I == 0);

disp(['Sensitivity (Recall): ', num2str(sensitivity)]);
disp(['Specificity (True Negative Rate): ', num2str(specificity)]);
```

WINE DATASET

Listing 4: Matlab Code.

```
% Load the data
load("WineData.mat")

% Initialize random seed for reproducibility
rng(1);

% Get dimensions of the data matrix
[n, m] = size(X);
p = m;

% Compute the distance matrix using norm 1 (Manhattan distance)
D = zeros(p);
for i = 1:p
    for j = i:p
        if i == j
            D(i, j) = 0;
        else
            D(i, j) = norm(X(:, i) - X(:, j), 2);
        end
    end
end
D = D + D';

% Initial medoids
I_m = [1, 100, 150];

% Plot initial data and medoids
figure()
plot(X(1,:), X(2,:), 'o', 'MarkerSize', 8, 'MarkerFaceColor', 'c', '↩
    MarkerEdgeColor', 'k');
hold on;
plot(X(1, I_m), X(2, I_m), 'p', 'MarkerSize', 12, 'LineWidth', 2, '↩
    MarkerEdgeColor', 'm', 'MarkerFaceColor', 'm');
legend('Data', 'Random Centroids', 'Location', 'best')
title('Initial Data and Random Centroids');
hold off;

% K-medoids algorithm
Err = 1;
itmax = 100;
tol = 1.0e-10;
iter = 0;
```

```
91
92  D_m = D(1:end, I_m);
93  [q, I_assign] = min(D_m');
94  Q = sum(q);
95  k = length(I_m);
96  oldI_m = I_m;
97  clear I_m;
98
99  % Iteratively refine medoids
100 while Err > tol && iter < itmax
101     iter = iter + 1;
102     for ell = 1:k
103         I_ell = find(I_assign == ell);
104         D_ell = D(I_ell, I_ell);
105         s = sum(D_ell);
106         [~, j] = min(s);
107         I_m(ell) = I_ell(j);
108     end
109     D_m = D(1:end, I_m);
110     [q, I_assign] = min(D_m');
111     Qnew = sum(q);
112     Err = abs(Q - Qnew);
113     Q = Qnew;
114 end
115
116 % Plot final data and medoids
117 figure()
118 plot(X(1,:), X(2,:), 'o', 'MarkerSize', 8, 'MarkerFaceColor', 'y', '↩
       MarkerEdgeColor', 'k');
119 hold on;
120 plot(X(1, I_m), X(2, I_m), 'p', 'MarkerSize', 12, 'LineWidth', 2, '↩
       MarkerEdgeColor', 'r', 'MarkerFaceColor', 'r');
121 legend('Data', 'Final Centroids', 'Location', 'best')
122 title('Final Data and Medoids');
123 hold off;
124
125 % Separate data points into clusters
126 for j = 1:k
127     X_l{j} = X(:, find(I_assign == j));
128 end
129
130 % 3D scatter plot of clusters with different colors
131 figure()
132 hold on;
133 scatter3(X_l{1}(1,:), X_l{1}(2,:), X_l{1}(3,:), 36, 'g', 'filled');
134 scatter3(X_l{2}(1,:), X_l{2}(2,:), X_l{2}(3,:), 36, 'b', 'filled');
135 scatter3(X_l{3}(1,:), X_l{3}(2,:), X_l{3}(3,:), 36, 'm', 'filled');
```

```
scatter3(X(1, I_m), X(2, I_m), X(3, I_m), 100, 'kx', 'LineWidth', 5);
title('Final clustering K Medoids');
legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Medoids', 'Location', 'best');
hold off;

% Confusion matrix chart
figure()
cm = confusionchart(I, I_assign);
title('Confusion Matrix for K-Medoids');
% Load the data
load("WineData.mat")

% Transpose the data
X = X';

% Perform k-means clustering
rng('default'); % For reproducibility
[idx, C] = kmeans(X, 3);

% Visualize the clusters and centroids with larger data point size
figure;
scatter(X(idx==1, 1), X(idx==1, 2), 100, 'r.', 'MarkerFaceColor', 'r');
hold on;
scatter(X(idx==2, 1), X(idx==2, 2), 100, 'b.', 'MarkerFaceColor', 'b');
scatter(X(idx==3, 1), X(idx==3, 2), 100, 'g.', 'MarkerFaceColor', 'g');
plot(C(:, 1), C(:, 2), 'mp', 'MarkerSize', 12, 'LineWidth', 2);
legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Centroids', 'Location', 'NW');
title('Cluster Assignments and Centroids');
hold off;

% 3D scatter plot of clusters and centroids with larger data point size
figure;
scatter3(X(idx==1, 1), X(idx==1, 2), X(idx==1, 3), 100, 'r.');
hold on;
scatter3(X(idx==2, 1), X(idx==2, 2), X(idx==2, 3), 100, 'b.');
scatter3(X(idx==3, 1), X(idx==3, 2), X(idx==3, 3), 100, 'g.');
scatter3(C(:, 1), C(:, 2), C(:, 3), 200, 'd', 'filled');
legend('Cluster 1', 'Cluster 2', 'Cluster 3', 'Centroids', 'Location', 'NW');
title('K-means Clusters and Centroids');
xlabel('Feature 1');
ylabel('Feature 2');
zlabel('Feature 3');
hold off;

```

```
% Confusion matrix chart
figure;
cm = confusionchart(I, idx);
title('Confusion Matrix for K-means');

% Load the data
load("WineData.mat");

% Separate data into subsets based on cluster annotation
I1 = find(I == 1);
I2 = find(I == 2);
I3 = find(I == 3);

% Compute class means
mu1 = mean(X(:, I1), 2);
mu2 = mean(X(:, I2), 2);
mu3 = mean(X(:, I3), 2);

% Compute within-class scatter matrix
Sw = cov(X(:, I1)', 1) + cov(X(:, I2)', 1) + cov(X(:, I3)', 1);

% Compute between-class scatter matrix
Sb = length(I1) * (mu1 - mean(X, 2)) * (mu1 - mean(X, 2))' + ...
     length(I2) * (mu2 - mean(X, 2)) * (mu2 - mean(X, 2))' + ...
     length(I3) * (mu3 - mean(X, 2)) * (mu3 - mean(X, 2))';

% Solve the generalized eigenvalue problem
[V, D] = eig(Sb, Sw);

% Sort eigenvectors and eigenvalues in descending order
[D_sorted, idx] = sort(diag(D), 'descend');
V_sorted = V(:, idx);

% Project the data onto the LDA directions
LDA_projection = V_sorted(:, 1:3)' * X;

% Plot two-dimensional projections for each LDA direction
for i = 1:size(LDA_projection, 1)
    figure;
    scatter(LDA_projection(i, I1), LDA_projection(i, I1), 100, 'r.');
    hold on;
    scatter(LDA_projection(i, I2), LDA_projection(i, I2), 100, 'b.');
    scatter(LDA_projection(i, I3), LDA_projection(i, I3), 100, 'g.');
    title(['LDA Projection ' num2str(i)]);
    xlabel(['LDA Component ' num2str(i)]);
    ylabel(['LDA Component ' num2str(i)]);
    legend('Cluster 1', 'Cluster 2', 'Cluster 3');
```

```
end

% Histogram of LDA Components
figure;
for i = 1:size(LDA_projection, 1)
    subplot(size(LDA_projection, 1), 1, i);
    histogram(LDA_projection(i, I1), 'FaceColor', 'r', 'EdgeColor', 'none'↩
        );
    hold on;
    histogram(LDA_projection(i, I2), 'FaceColor', 'b', 'EdgeColor', 'none'↩
        );
    histogram(LDA_projection(i, I3), 'FaceColor', 'g', 'EdgeColor', 'none'↩
        );
    title(['Histogram of LDA Component ' num2str(i)]);
    legend('Cluster 1', 'Cluster 2', 'Cluster 3');
    ylabel('Frequency');
    xlabel(['LDA Component ' num2str(i)]);
    hold off;
end

% Perform PCA
[coeff, score, ~, ~, explained] = pca(X');

% Project the data onto the first three PCA directions
PCA_projection = coeff(:, 1:3)' * X;

% Plot two-dimensional projections for each PCA direction
for i = 1:size(PCA_projection, 1)
    figure;
    scatter(PCA_projection(i, I1), PCA_projection(i, I1), 100, 'r.');
    hold on;
    scatter(PCA_projection(i, I2), PCA_projection(i, I2), 100, 'b.');
    scatter(PCA_projection(i, I3), PCA_projection(i, I3), 100, 'g.');
    title(['PCA Projection ' num2str(i)]);
    xlabel(['PCA Component ' num2str(i)]);
    ylabel(['PCA Component ' num2str(i)]);
    legend('Cluster 1', 'Cluster 2', 'Cluster 3');
end

% Histogram of PCA Components
figure;
for i = 1:size(PCA_projection, 1)
    subplot(size(PCA_projection, 1), 1, i);
    histogram(PCA_projection(i, I1), 'FaceColor', 'r', 'EdgeColor', 'none'↩
        );
    hold on;
    histogram(PCA_projection(i, I2), 'FaceColor', 'b', 'EdgeColor', 'none'↩
```

```
        );
270     histogram(PCA_projection(i, I3), 'FaceColor', 'g', 'EdgeColor', 'none'↩
        );
271     title(['Histogram of PCA Component ' num2str(i)]);
272     legend('Cluster 1', 'Cluster 2', 'Cluster 3');
273     ylabel('Frequency');
274     xlabel(['PCA Component ' num2str(i)]);
275     hold off;
276 end
277
278 % Confusion Matrix for LDA
279 LDA_labels = classify(LDA_projection(1:2, :)', LDA_projection(1:2, :)', I'↩
    ); % Use first two LDA components
280 confMat_LDA = confusionmat(I, LDA_labels);
281
282 % Plot the confusion matrix for LDA
283 figure;
284 confusionchart(confMat_LDA, {'Cluster 1', 'Cluster 2', 'Cluster 3'});
285 title('Confusion Matrix for LDA Classification of Wine Data');
286
287 % Confusion Matrix for PCA
288 PCA_labels = classify(PCA_projection(1:2, :)', PCA_projection(1:2, :)', I'↩
    ); % Use first two PCA components
289 confMat_PCA = confusionmat(I, PCA_labels);
290
291 % Plot the confusion matrix for PCA
292 figure;
293 confusionchart(confMat_PCA, {'Cluster 1', 'Cluster 2', 'Cluster 3'});
294 title('Confusion Matrix for PCA Classification of Wine Data');
```

CARDIAC DATASET

Listing 5: Matlab Code.

```
298 % Load the data
299 clear;
300 close all;
301 load CardiacSPECT.mat
302
303 % Get the size of the data matrix
304 [n, p] = size(X);
305
306 % Initialize the distance matrix
307 D = zeros(p);
```

```
308
309 % Compute the distance matrix by comparing columns of X
310 for i = 1:(p-1)
311     for j = (i+1):p
312         D(i, j) = sum(X(:, i) ~= X(:, j)) / n;
313     end
314 end
315
316 % Make the distance matrix symmetric
317 D = D + D';
318
319 % Set the random number generator seed for reproducibility
320 rng('default');
321
322 % Number of clusters
323 k = 2;
324
325 % Number of trials for k-medoids
326 n_init = 20;
327
328 % Initialize variables for storing results
329 I_m_ni = cell(1, n_init);
330 I_assign_ni = cell(1, n_init);
331 Q_ni = zeros(1, n_init);
332
333 % Run k-medoids algorithm with multiple initializations
334 for ni = 1:n_init
335     % Random initial medoids
336     I_m_ni{ni} = sort(randperm(length(D), k));
337     D_m = D(:, I_m_ni{ni});
338     [~, I_assign_ni{ni}] = min(D_m');
339     Q_ni(ni) = sum(min(D_m'));
340 end
341
342 % Find the best clustering with minimum tightness (coherence)
343 [~, ni_min_tightness] = min(Q_ni);
344 I_m = I_m_ni{ni_min_tightness};
345 I_assign = I_assign_ni{ni_min_tightness};
346
347 % Refine the clustering iteratively
348 Err = 1;
349 itmax = 100;
350 iter = 0;
351 tol = 1.0e-10;
352 Q = sum(min(D(:, I_m)));
353
354 while (Err >= tol && iter < itmax)
```

```
    [~, I_assign] = min(D(:, I_m)');
    Qnew = Q;
    Q = sum(min(D(:, I_m)));

    for ell = 1:k
        I_ell = find(I_assign == ell);
        D_ell = D(I_ell, I_ell);
        s = sum(D_ell);
        [~, j] = min(s);
        I_m(ell) = I_ell(j);
    end

    Err = abs(Q - Qnew);
    iter = iter + 1;
end

% Final medoids
I_bar = I_m;
flag = Err < tol;

disp('flag');
disp(flag);
disp('final medoids');
disp(I_m);

% Change I_assign to become 0 and 1 as the I vector
I_assign = I_assign - 1;

% Perform k-means clustering
[idx_kmeans, ~] = kmeans(X', k, 'Replicates', 20);

% Create confusion matrix chart for k-medoids
figure();
cm_medoids = confusionchart(I, I_assign);
title('Confusion Matrix for K-medoids');

% Create confusion matrix chart for k-means
figure();
cm_kmeans = confusionchart(I, idx_kmeans - 1);
title('Confusion Matrix for K-means');

% Perform PCA for 2D visualization
[coeff, score] = pca(X');

% Scatter plot of the first two principal components for k-medoids
figure();
gscatter(score(:, 1), score(:, 2), I_assign, 'rb', 'ox');
```

```
xlabel('Principal Component 1');
ylabel('Principal Component 2');
title('PCA of K-medoids Clustering Results');
legend('Cluster 0', 'Cluster 1');

% Scatter plot of the first two principal components for k-means
figure();
gscatter(score(:, 1), score(:, 2), idx_kmeans - 1, 'gb', 'ox');
xlabel('Principal Component 1');
ylabel('Principal Component 2');
title('PCA of K-means Clustering Results');
legend('Cluster 0', 'Cluster 1');

% Create matrix C for k-medoids
c11_medoids = length(find(I_assign == 1 & I == 1));
c12_medoids = length(find(I_assign == 0 & I == 1));
c21_medoids = length(find(I_assign == 1 & I == 0));
c22_medoids = length(find(I_assign == 0 & I == 0));
C_medoids = [c11_medoids c12_medoids; c21_medoids c22_medoids];

% Create matrix C for k-means
c11_kmeans = length(find(idx_kmeans - 1 == 1 & I == 1));
c12_kmeans = length(find(idx_kmeans - 1 == 0 & I == 1));
c21_kmeans = length(find(idx_kmeans - 1 == 1 & I == 0));
c22_kmeans = length(find(idx_kmeans - 1 == 0 & I == 0));
C_kmeans = [c11_kmeans c12_kmeans; c21_kmeans c22_kmeans];

% Display matrices for k-medoids
disp('Matrix C for K-medoids:');
disp(C_medoids);

% Display matrices for k-means
disp('Matrix C for K-means:');
disp(C_kmeans);

% Additional visualization: Clustering in the original feature space for k-medoids
figure;
hold on;
for i = 1:p
    cluster = I_assign(i) + 1; % 1 or 2
    scatter(1:n, X(:, i), 36);
end
xlabel('Feature Index');
ylabel('Binary Value');
title('K-medoids Clustering in Original Feature Space');
legend('Cluster 0', 'Cluster 1');
```

```
448 hold off;
449
450 % Additional visualization: Clustering in the original feature space for k↩
        -means
451 figure;
452 hold on;
453 for i = 1:p
454     cluster = idx_kmeans(i); % 1 or 2
455     scatter(1:n, X(:, i), 36, colors(cluster), markers(cluster));
456 end
457 xlabel('Feature Index');
458 ylabel('Binary Value');
459 title('K-means Clustering in Original Feature Space');
460 legend('Cluster 0', 'Cluster 1');
461 hold off;
462 % Load the dataset
463 load CardiacSPECT.mat;
464
465 % Perform LDA
466 classLabels = unique(I);
467 numClasses = numel(classLabels);
468 numFeatures = size(X, 1);
469 meanVecs = zeros(numFeatures, numClasses);
470 Sw = zeros(numFeatures, numFeatures);
471 Sb = zeros(numFeatures, numFeatures);
472
473 % Calculate class means
474 for i = 1:numClasses
475     meanVecs(:, i) = mean(X(:, I == classLabels(i)), 2);
476 end
477
478 % Calculate within-class scatter matrix
479 for i = 1:numClasses
480     Xi = X(:, I == classLabels(i));
481     Sw = Sw + (Xi - meanVecs(:, i)) * (Xi - meanVecs(:, i))';
482 end
483
484 % Calculate between-class scatter matrix
485 meanOverall = mean(X, 2);
486 for i = 1:numClasses
487     Ni = sum(I == classLabels(i));
488     Sb = Sb + Ni * (meanVecs(:, i) - meanOverall) * (meanVecs(:, i) - ↩
          meanOverall)';
489 end
490
491 % Solve the generalized eigenvalue problem
492 [V, D] = eig(Sb, Sw);
```

```
493
494 % Sort eigenvectors by eigenvalues in descending order
495 [~, indices] = sort(diag(D), 'descend');
496 W = V(:, indices);
497
498 % Project the data onto the first two LDA directions
499 LDA_projection = W(:, 1:2)' * X;
500
501 % Plot the data projected on the first two LDA directions
502 figure;
503 scatter(LDA_projection(1, I == 0), LDA_projection(2, I == 0), 100, 'b.', '↩
       DisplayName', 'Normal');
504 hold on;
505 scatter(LDA_projection(1, I == 1), LDA_projection(2, I == 1), 100, 'r.', '↩
       DisplayName', 'Abnormal');
506 title('LDA Projection for Cardiac SPECT Dataset');
507 xlabel('LDA Component 1');
508 ylabel('LDA Component 2');
509 legend('Location', 'best');
510 grid on;
511 set(gca, 'FontSize', 12);
512
513 % Classification using LDA projection
514 class_means_lda = [mean(LDA_projection(:, I == 0), 2), mean(LDA_projection↩
       (:, I == 1), 2)];
515 predictedLabels_lda = zeros(1, size(X, 2));
516
517 for i = 1:size(X, 2)
518     distances = sum((LDA_projection(:, i) - class_means_lda).^2, 1);
519     [~, predictedLabels_lda(i)] = min(distances);
520 end
521
522 % Convert 1-based index to original labels
523 predictedLabels_lda = predictedLabels_lda - 1;
524
525 % Create a confusion matrix for LDA
526 confMat_lda = confusionmat(I, predictedLabels_lda);
527
528 % Plot the confusion matrix for LDA
529 figure;
530 confusionchart(confMat_lda, {'Normal', 'Abnormal'});
531 title('Confusion Matrix for LDA Classification of Cardiac SPECT Dataset');
532
533 % Perform PCA
534 [coeff, score, ~, ~, explained] = pca(X');
535
536 % Project the data onto the first two PCA directions
```

```
PCA_projection = coeff(:, 1:2)' * X;

% Plot the data projected on the first two PCA directions
figure;
scatter(PCA_projection(1, I == 0), PCA_projection(2, I == 0), 100, 'b.', '↩
    DisplayName', 'Normal');
hold on;
scatter(PCA_projection(1, I == 1), PCA_projection(2, I == 1), 100, 'r.', '↩
    DisplayName', 'Abnormal');
title('PCA Projection for Cardiac SPECT Dataset');
xlabel('PCA Component 1');
ylabel('PCA Component 2');
legend('Location', 'best');
grid on;
set(gca, 'FontSize', 12);

% Classification using PCA projection
class_means_pca = [mean(PCA_projection(:, I == 0), 2), mean(PCA_projection↩
    (:, I == 1), 2)];
predictedLabels_pca = zeros(1, size(X, 2));

for i = 1:size(X, 2)
    distances = sum((PCA_projection(:, i) - class_means_pca).^2, 1);
    [~, predictedLabels_pca(i)] = min(distances);
end

% Convert 1-based index to original labels
predictedLabels_pca = predictedLabels_pca - 1;

% Create a confusion matrix for PCA
confMat_pca = confusionmat(I, predictedLabels_pca);

% Plot the confusion matrix for PCA
figure;
confusionchart(confMat_pca, {'Normal', 'Abnormal'});
title('Confusion Matrix for PCA Classification of Cardiac SPECT Dataset');
```

HANDWRITTENDIGITS DATASET

Listing 6: Matlab Code.

```
% Load the dataset
load HandwrittenDigits.mat % X = data matrix, I = annotation vector

```

```
% Create a data matrix X04 containing images of digits '0' and '4'
I0 = find(I == 0); % Indices of digit '0'
I4 = find(I == 4); % Indices of digit '4'
X04 = [X(:, I0), X(:, I4)]; % Combine images of digits '0' and '4'

% Calculate centered matrices for within-class scatter
X0_centered = X(:, I0) - mean(X(:, I0), 2);
X4_centered = X(:, I4) - mean(X(:, I4), 2);

% Calculate within-class scatter matrix
Sw = X0_centered * X0_centered' + X4_centered * X4_centered';

% Compute between-class scatter matrix
Sb = length(I0) * (mean(X(:, I0), 2) - mean(X04, 2)) * (mean(X(:, I0), 2) - mean(X04, 2))' + ...
    length(I4) * (mean(X(:, I4), 2) - mean(X04, 2)) * (mean(X(:, I4), 2) - mean(X04, 2))';

% Perform eigenvalue decomposition for LDA
[V, D] = eig(pinv(Sw) * Sb);
[~, indices] = sort(diag(D), 'descend');
eigenvectors = V(:, indices);

% Project data onto first two LDA directions
LDA_projection = eigenvectors(:, 1:2)' * X04;

% Plot LDA projection
figure;
scatter(LDA_projection(1, 1:length(I0)), LDA_projection(2, 1:length(I0)), 100, 'b.', 'DisplayName', 'Digit ''0''');
hold on;
scatter(LDA_projection(1, length(I0)+1:end), LDA_projection(2, length(I0)+1:end), 100, 'r.', 'DisplayName', 'Digit ''4''');
title('Projection of Digits ''0'' and ''4'' on LDA Directions');
xlabel('LDA Component 1');
ylabel('LDA Component 2');
legend('Location', 'best');
grid on;
set(gca, 'FontSize', 12);

% NMF algorithm
k_values = [5, 10, 20];

for k = k_values
    disp(['Running NMF for k = ', num2str(k)]);
    rng(1);
    W = rand(size(X04, 1), k);
```

```
    H = rand(k, size(X04, 2));
    nmax = 100;
    tau = 0.01;
    ch = NaN(1, nmax);

    for t = 1:nmax
        H = max(0, H .* (W' * X04) ./ (W' * W * H + eps));
        W = max(0, W .* (X04 * H') ./ (W * H * H' + eps));
        ch(t) = norm(X04 - W * H, 'fro') / norm(X04, 'fro');
        if ch(t) < tau
            disp(['Converged at iteration ', num2str(t)]);
            break;
        end
    end

    % Plot convergence diagnostics
    figure;
    plot(1:t, ch(1:t), 'b-', 'LineWidth', 2);
    title(['Convergence Diagnostics (k = ', num2str(k), ')']);
    xlabel('Iteration');
    ylabel('Relative Change');
    grid on;
    set(gca, 'FontSize', 12);

    % Plot feature vectors as images
    figure;
    for j = 1:k
        subplot(ceil(sqrt(k)), ceil(sqrt(k)), j);
        imagesc(reshape(W(:, j), 16, 16)');
        colormap(gray);
        axis('square');
        axis('off');
        title(['Feature ', num2str(j)]);
    end
end

% Identify specific feature vector
[~, idx] = max(H(:));
[idx_row, idx_col] = ind2sub(size(H), idx);
feature_vector = H(:, idx_col);
digit_classification = I0(idx_row);

if digit_classification == 0
    disp('The feature vector represents a ''0''.');
elseif digit_classification == 4
    disp('The feature vector represents a ''4''.');
else
```

```
664     disp('The digit classification is different from ''0'' and ''4''.');
665 end
666
667 % Perform PCA for comparison
668 [coeff, score, ~, ~, explained] = pca(X04');
669 PCA_projection = coeff(:, 1:2)' * X04;
670
671 % Plot PCA and LDA projections
672 figure;
673 subplot(1, 2, 1);
674 scatter(LDA_projection(1, 1:length(I0)), LDA_projection(2, 1:length(I0)), ↩
       100, 'b.', 'DisplayName', 'Digit ''0''');
675 hold on;
676 scatter(LDA_projection(1, length(I0)+1:end), LDA_projection(2, length(I0)↩
       +1:end), 100, 'r.', 'DisplayName', 'Digit ''4''');
677 title('LDA Projection');
678 xlabel('LDA Component 1');
679 ylabel('LDA Component 2');
680 legend('Location', 'best');
681 grid on;
682 set(gca, 'FontSize', 12);
683
684 subplot(1, 2, 2);
685 scatter(PCA_projection(1, 1:length(I0)), PCA_projection(2, 1:length(I0)), ↩
       100, 'b.', 'DisplayName', 'Digit ''0''');
686 hold on;
687 scatter(PCA_projection(1, length(I0)+1:end), PCA_projection(2, length(I0)↩
       +1:end), 100, 'r.', 'DisplayName', 'Digit ''4''');
688 title('PCA Projection');
689 xlabel('PCA Component 1');
690 ylabel('PCA Component 2');
691 legend('Location', 'best');
692 grid on;
693 set(gca, 'FontSize', 12);
```