- **Question 1**

1. Part A.1

First, with the use of MATLAB (R2023b) function mvnrnd (multivariate normal distribution), and the given prior, mean, and covariance matrices, the 4-D vector **X** and the corresponding label were generated and saved for the following questions.

2. Part A.2

We evaluated the posteriors, p(x|L=1) and p(x|L=0) with a home-build function g = evalGaussian(x,mu,Sigma), and we define their Quotient as *discriminator*. Next, we defined a group of thresholds γ with the range of [0, +∞), when implementing in the code, we took the min(*discriminator*), max(*discriminator*), and the midpoint of each two *discriminator*s as our γ, which gave us a range which is closed to [0, +∞). Our decision is based on the ratio of the class posteriors, decision is 1 when the ratio is larger than γ, and the decision is 0 when the ratio is smaller than γ. For each of the γ, a decision will be generated. To draw the ROC curve, we calculated the True Positive Rate (TPR) as TP/(TP+FN), and False Positive Rate (FPR) as FP/(FP+TN), TP, FN, FP, and TN are represented for true positive, false negative, false negative, and true negative, respectively. And the probability of error (P_{error}) is calculated as (FP+FN)/N, where N is the total number of the iid samples. The x-axis of the ROC curve is FPR, while the y-axis of the ROC curve is TPR. **Figure 1** presents the ROC curve for different threshold γ.

3. Part A.3

The minimum P_{error} = 0.0365 was achieved when γ = 0.5471, as shown by the superimposed green star in Figure 1. Theoretically optimal threshold is, when $\gamma = \frac{\lambda_{10}-\lambda_{00}}{\lambda_{01}-\lambda_{11}} \times \frac{\mathrm{P(L=0)}}{\mathrm{P(L=1)}} = 0.5385$, P_{error} = 0.0369. The P_{error} regarding the empirically selected γ is closed to the theoretical optimal threshold (with 1.08% absolute difference).

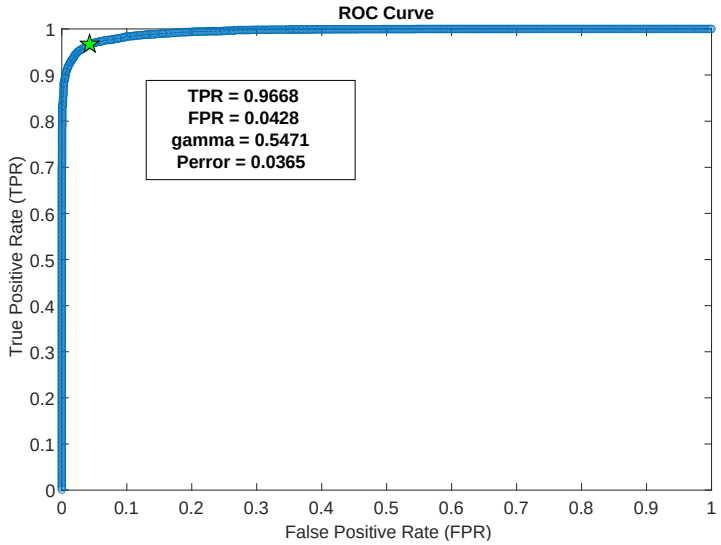

Figure 1. The ROC curve with different threshold γ, the star shows the point with the minimum P_{error}, with a threshold $\gamma = 0.5471$.

4. Part B

By multiplying a diagonal matrix to the covariance matrices, we achieved new covariance matrices with off-diagonal entries equal to zeros. Next, we repeated Part A. **Figure 2** shows the corresponding ROC Curve. The model mismatch results in an increase in the minimum achievable probability of error, the P_{error} increased to 0.043 from 0.0365 (17.8% higher), while the selected γ = 0.3947. The AUC decreased compared to the AUC from Figure 1, which means that model mismatch has a negative impact on the ROC curve and P_{error}.

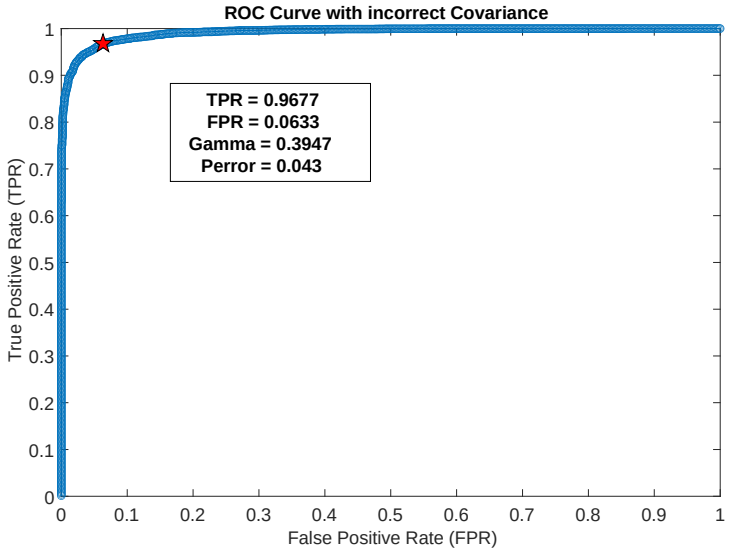

Figure 2. The ROC curve with incorrect covariance matrices. the star shows the point with the minimum P_{error}, with a threshold $\gamma = 0.3947$.

5. Part C

To perform the Linear Discriminant Analysis (LDA) algorithm, the estimated means of the x (with L=1, and L=0) were first estimated, and the between class scatter matrix $s_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$, within-class scatter matrix $S_W = \Sigma_1 + \Sigma_2$, where Σ_1 and Σ_2 are the estimated covariances of x with different labels. To separate 2 categories, the objective function of Fisher LDA is

$$arg\ max_\omega = \frac{\omega^T S_B \omega}{\omega^T S_W \omega}$$

After calculating the gradient and set the gradient to be 0, the following equation can be derived:

$$S_w^{-1} S_B \omega = (\frac{\omega^T S_B \omega}{\omega^T S_W \omega})\omega$$

The optimal ω_{LDA} is the eigenvector of $S_w^{-1} S_B$, that corresponds to the largest eigenvalue of this matrix. After deriving ω_{LDA}, we calculated the output of Fisher LDA which equals to $y = \omega_{LDA} x$, we defined y as the *discriminator* and drew the ROC curve (as shown in **Figure 3**). The threshold $\tau = -0.31$ which resulted in the minimum probability of error P_{error} = 0.043. The performance of the Fisher LDA classifier is very close to the result of Part B, in the current case the P_{error} are the same as both methods assume features are independent given each class label.

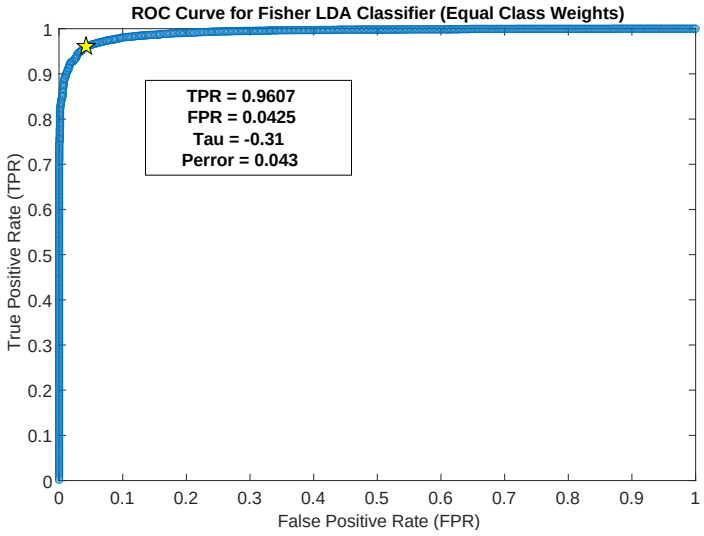

Figure 3. The ROC curve with Fisher LDA, the point marked with star indicates the threshold $\tau = -0.31$ which resulted in the minimum probability of error P_{error} = 0.043.

The code of Question 1 is in Appendix A.

- **Question 2**

1. Part A.1

First, we manually defined the means and covariances of the four Gaussians and formed X and the corresponding labels using these means, covariances and the given priors. **Figure 4** displays the random vector **X**.

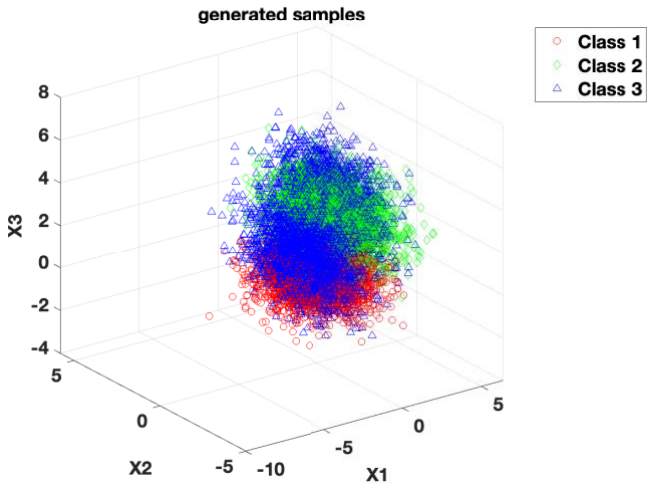

Figure 4. A 3-D random vector X which takes values from a mixture of four Gaussians.

2. Part A.2

The loss matrix is defined as [0,1,1;1,0,1;1,1,0], which is, Loss=0 if the classified label=true label. The decision rule is based on the minimum probability of error (or, maximum a posterior, MAP decision rule):

$$\mathrm{R}(L_P|X) = \Sigma_{L_i \neq L_p} \mathrm{P}(L_i|X)$$

Using Bayles's rule, the class posteriors can be calculated as

$$\mathrm{P}(L_i|\, X) \;= \frac{\mathrm{P}(X|\, L_i) \times \mathrm{P}(L_i)}{\mathrm{P}(X)}$$

where i, $\mathrm{P}(L_i)$, $\mathrm{P}(X)$, and $\mathrm{P}(X|\, L_i)$ represents the labels (0 or 1), priors, evidence and the likelihood, respectively. When implementing in the code, the expected risk is calculated as loss*class posteriors, and the classified label is $\mathrm{L_i}$ when the expected risk is minimum. **Figure 5** presents the confusion matrix, leveraging the MATLAB function confusionmat, the diagonal elements of the confusion matrix indices the labels which have been classified correctly, we noted here the entry (3,3)=0.6398, which means 63.98% of the sample with label 3 have been classified correctly.

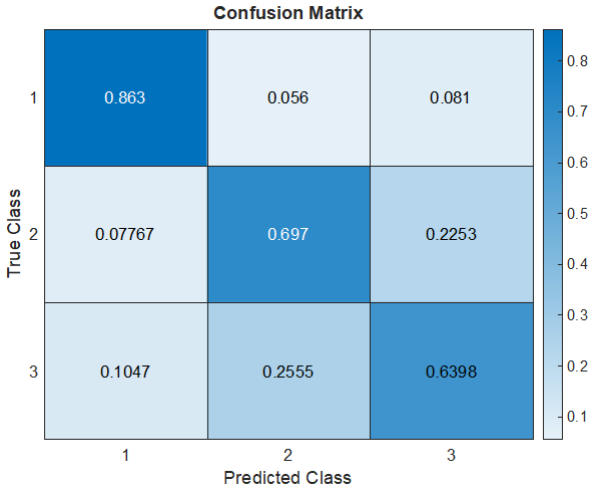

Figure 5. The confusion matrix from the minimum probability of error (with 0-1 loss) classifier, the true class label is along the y-axis, and the predicted class label is along the x-axis.

3. Part A.3

Figure 6 displays the data and the classification results, where green indicates that the current sample has been classified correctly, while red means it has been incorrectly classified, the shapes of circle, diamond, and triangle represents class 1, 2, and 3, respectively.

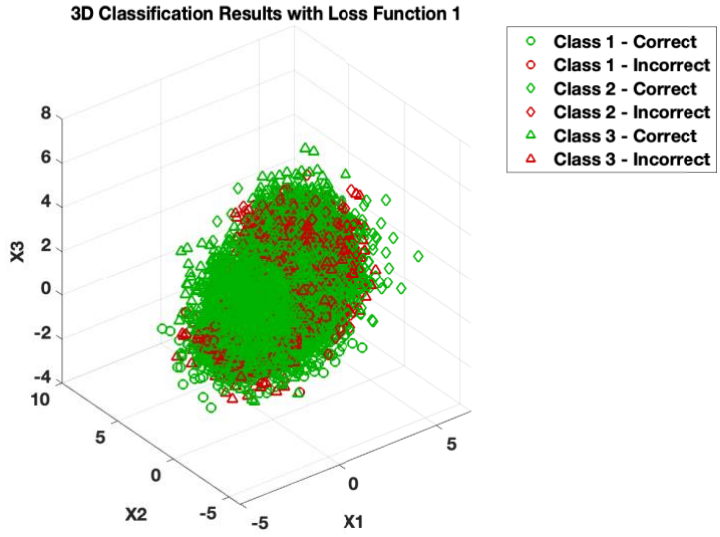

Figure 6. The classification results of the minimum probability of error model with 0-1 loss. Green means the current label has been classified correctly and red means incorrect classification. Classes 1, 2, and 3 are resented as circle, diamond, and triangle, respectively.

4. Part B

Figure 7 and **Figure 9** show the classification results with loss matrices Λ_{10} and Λ_{100}, while **Figure 8** and **Figure 10** show the corresponding confusion matrices. The entry (3,3) of the confusion matrix with Λ_{10}= 0.9938 and equals to 1 with Λ_{100}. That indicates when increasing the loss for a specific class, the model intends not to make incorrect classification on that label, as shown in the confusion matrices and the classification results. Although increasing a loss to a certain label will increase the TPR, but we can also observe that the FPR increases as well, as can be seen from the entry (1,3) and (2,3) from **Figure 8** and **Figure 10**. That indicates that when choosing loss matrix, we might need to consider and balance both the TPR and FPR.

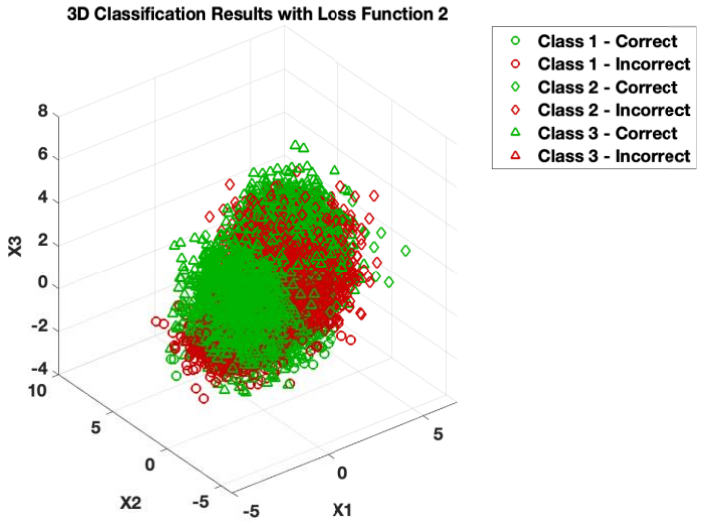

Figure 7. The classification results with loss matrix Λ_{10}.

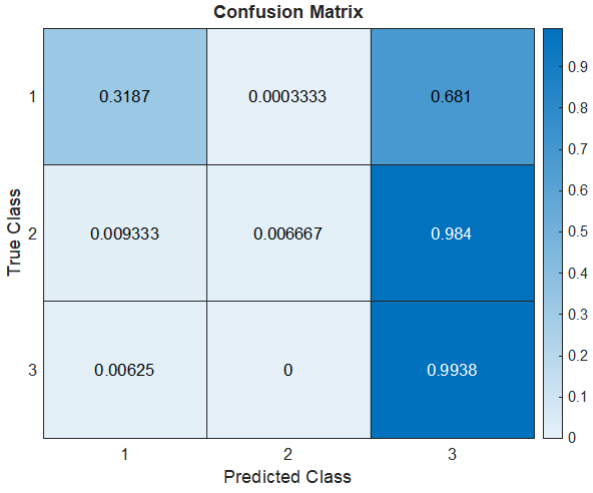

Figure 8. The confusion matrix of loss matrix Λ_{10}.

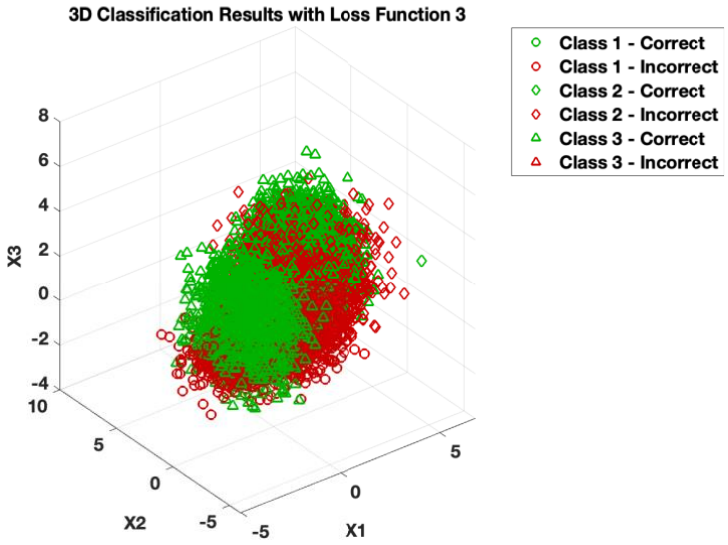

Figure 9. The classification results with loss matrix Λ_{100}.

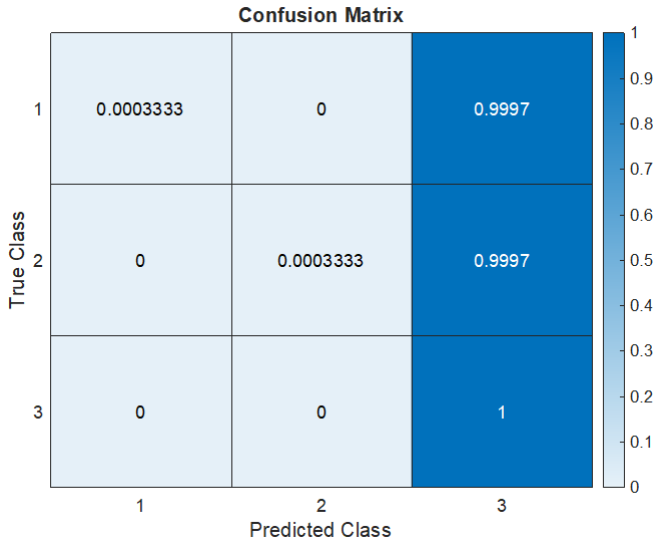

Figure 10. The confusion matrix of loss matrix Λ_{100}.

The code of Question 2 is in Appendix B.

- **Question 3**

1. **Part A**

The red wine dataset contains N=1599 samples, and d=11 features, and the classes are [3:8], the white wine dataset contains N=4898 samples, d=11 features, and the classes are [3:9]. The Human activity recognition dataset contains N=10299 samples (here the test and train sets have been combined), and d=561 features, and the classes are [1:6].

We used sample counts for each class as the priors. Under the assumption that the class conditional PDF for each class is a Gaussian, we can estimate the mean and covariance of each class. To avoid ill-condition of the covariance and then evaluate the Gaussian pdf (the inverse of the covariance will be calculated), when estimating the covariance of samples of each class, a regularization term has been added under the condition the *det* of the covariance matrix is less than 1e-6. In this code, the parameter of the regularization term λI, λ was fixed as 0.01 during the experiment. Similar to the previous question, the posterior was calculated, and the expected risk was calculated as loss×class posteriors. The classification label was decided with the minimum expected risk.

After performing the minimum-probability-of-error classifier to red wine and white wine datasets, the probability of error and the confusion matrix were calculated. **Table 1** shows the probability of error on the three datasets (red wine, white wine, and human activity recognition). The confusion matrices of the three datasets are shown in **Figure 11**.

Table 1. The probabilities of error on the red wine, white wine, and the human activity recognition datasets.

Dataset	Red Wine	White Wine	Human Activity Recognition
Probability of Error	0.3984	0.4624	0.0163

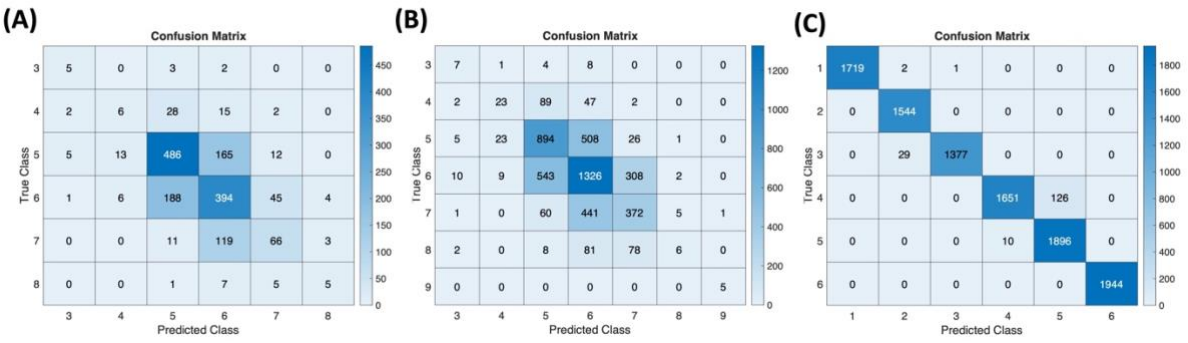

Figure 11. (A-C) The confusion matrices from the red wine, white wine, and human activity recognition datasets.

2. Part B

Using Principal Component Analysis (PCA), the datasets can be visualized as 2D or 3D projections, as shown in **Figure 12**, where **Figure 12 (A)** and **(B)** displays the 2D projection and 3D projection of the red wine dataset, **(C)** and **(D)** shows the projections of the white dataset, and **(E)** and **(F)** are for human activity recognition dataset, respectively.

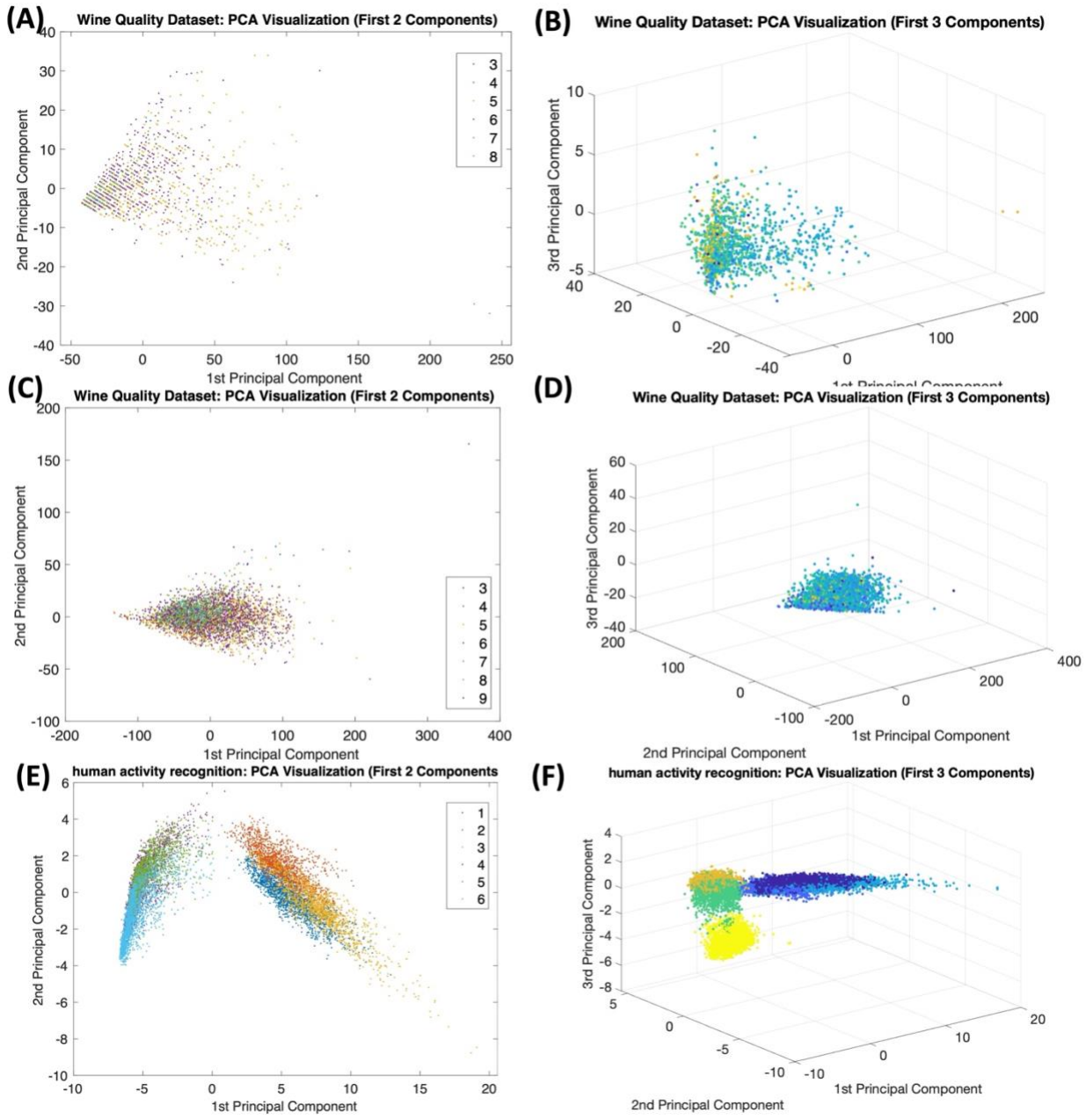

Figure 12. The first two and three principal components using PCA on the three datasets. (A-B), red wine dataset, (C-D), white wine dataset, and (E-F), human activity dataset.

Based the probability of errors, the confusion matrices, and the reducing dimension visualization results, the following conclusions can be drawn: Gaussian class conditional model is not appropriate on the wine quality dataset, while it is well aligned with the human activity recognition dataset. With the reduced dimension visualization plot (Figure 12), the samples of different classes are not separatable on the wine quality dataset, in this case the Bayes error is the lower bound of all possible error probabilities, but inherently the error is considerable.

The code of Question 3 is in Appendix C.

Appendix A

1. Matlab Code for Question 1

```
% Espected Risk Minimization
clear; close all; clc;
%--------------------------------------------------------------------------
%% Preparation
%--------------------------------------------------------------------------
if (0)
n = 4; % number of feature dimensions
N = 10000; % number of iid samples
m(:,1) = [-1;-1;-1;-1];
m(:,2) = [1;1;1;1];
C(:,:,1) = [2 -0.5 0.3 0; -0.5 1 -0.5 0; 0.3 -0.5 1 0; 0 0 0 2];
C(:,:,2) = [1 0.3 -0.2 0; 0.3 2 0.3 0; -0.2 0.3 1 0; 0 0 0 3];
p = [0.35,0.65]; % class priors for labels 0 and 1 respectively
label = rand(1,N) >= p(1); % True label
Ncount = [length(find(label==0)), length(find(label==1))]; % number of samples from each class
x = zeros(n,N);
x(:,label==0) = mvnrnd(m(:,1),C(:,:,1),Ncount(1))';
x(:,label==1) = mvnrnd(m(:,2),C(:,:,2),Ncount(2))';
save('erm_test_data.mat') % save the work space for the following questions
end

%--------------------------------------------------------------------------
%% Part A 1
%--------------------------------------------------------------------------

load('/Users/ln915/Documents/Lvx/2024_Fall/Intro2ML/Assignments/1/data/erm_test_data.mat')

lambda=[0 1; 1 0];% 0-1 loss, theoretical
gamma0 = (lambda(2,1)-lambda(1,1))/(lambda(1,2)-lambda(2,2))*p(1)/p(2);
discriminator=evalGaussian(x,m(:,2),C(:,:,2))./ evalGaussian(x,m(:,1),C(:,:,1));
[sortedScores,ind] = sort(discriminator,'ascend');
thresholdList = [min(sortedScores)-eps,(sortedScores(1:end-1)+sortedScores(2:end))/2, max(sortedScores)+eps];
gamma1=thresholdList;
n_gamma = length(gamma1);

%--------------------------------------------------------------------------
%% Part A 2
%--------------------------------------------------------------------------

TPR = zeros(1, n_gamma); % True Positive Rate, y-axis of the ROC curve
FPR = zeros(1, n_gamma); % False Positive Rate, x-axis of the ROC curve
P_error=zeros(1, n_gamma);

for i = 1: n_gamma
decision = (discriminator >= gamma1(i));
TP = sum(decision == 1 & label == 1); % True Positives
FP = sum(decision == 1 & label == 0); % False Positives
FN = sum(decision == 0 & label == 1); % False Negatives
TN = sum(decision == 0 & label == 0); % True Negatives
TPR(i) = TP / (TP + FN); % record the current value
FPR(i) = FP / (FP + TN);
P_error(i)=(FP+FN)/N;
end

figure(1);% Plot ROC Curve
plot(FPR, TPR, '-o');
xlabel('False Positive Rate (FPR)');
ylabel('True Positive Rate (TPR)');
title('ROC Curve');
grid on;
hold on;

%--------------------------------------------------------------------------
```

```
%% Part A 3
%------------------------------------------------------------------------

best_gamma=find(P_error==min(P_error));
gamma1_best = gamma1(best_gamma);

for i=1:size(best_gamma)
plot( FPR(best_gamma(i)),  TPR(best_gamma(i)), 'kp', 'MarkerSize', 15, 'MarkerFaceColor', 'g');
hold on;
end

min_Perror=min(P_error);
%------------------------------------------------------------------------
%% Part B
%------------------------------------------------------------------------
clearvars -except gamma1_best FPR(best_gamma)  TPR(best_gamma)  min_Perror

load('/Users/ln915/Documents/Lvx/2024_Fall/Intro2ML/Assignments/1/data/erm_test_data.mat')

C_incorrect=C.*diag(ones(1, n));
discriminator=evalGaussian(x,m(:,2),C_incorrect(:,:,2))./ evalGaussian(x,m(:,1),C_incorrect(:,:,1));

[sortedScores,ind] = sort(discriminator,'ascend');
thresholdList = [min(sortedScores)-eps,(sortedScores(1:end-1)+sortedScores(2:end))/2, max(sortedScores)+eps];
gamma1=thresholdList;

n_gamma = length(gamma1);

TPR_new = zeros(1, n_gamma); % True Positive Rate, y-axis of the ROC curve
FPR_new = zeros(1, n_gamma); % False Positive Rate, x-axis of the ROC curve
P_error=zeros(1, n_gamma);

for i = 1: n_gamma
decision = (discriminator >= gamma1(i));
TP = sum(decision == 1 & label == 1); % True Positives
FP = sum(decision == 1 & label == 0); % False Positives
FN = sum(decision == 0 & label == 1); % False Negatives
TN = sum(decision == 0 & label == 0); % True Negatives
TPR_new(i) = TP / (TP + FN); % record the current value
FPR_new(i) = FP / (FP + TN);
P_error(i)=(FP+FN)/N;
end

figure(2);% Plot ROC Curve
plot(FPR_new, TPR_new, '-o');
xlabel('False Positive Rate (FPR)');
ylabel('True Positive Rate (TPR)');
title('ROC Curve with incorrect Covariance ');
grid on; hold on;

best_gamma=find(P_error==min(P_error));
gamma2_best = gamma1(best_gamma);

for i=1:size(best_gamma)
plot( FPR_new(best_gamma(i)),  TPR_new(best_gamma(i)), 'kp', 'MarkerSize', 15, 'MarkerFaceColor', 'r');
hold on;
end

min_Perror_new = min(P_error);

%------------------------------------------------------------------------
%% Part C Linear Discriminant Analysis LDA
%------------------------------------------------------------------------
load('/Users/ln915/Documents/Lvx/2024_Fall/Intro2ML/Assignments/1/data/erm_test_data.mat')

mu1hat = mean(x(:,label==0),2);
mu2hat = mean(x(:,label==1),2);
Sb=(mu1hat-mu2hat)*(mu1hat-mu2hat)';
Sw=cov(x(:,label==0)')+cov(x(:,label==1)'); % equal weights

[V, D] = eig(Sw \ Sb);
[~, max_idx] = max(diag(D));  % Find the eigenvector corresponding to the largest eigenvalue
```

```
w_LDA = V(:, max_idx);       % Fisher LDA projection vector

if (mean(w_LDA' * x(:,label==0))>mean(w_LDA' * x(:,label==1)))
w_LDA=-w_LDA;
end

y = (w_LDA)' * x;

discriminator=y;

[sortedScores,ind] = sort(discriminator,'ascend');
thresholdList = [min(sortedScores)-eps,(sortedScores(1:end-1)+sortedScores(2:end))/2, max(sortedScores)+eps];
n_gamma = length(thresholdList);

TPR = zeros(1, n_gamma); % True Positive Rate, y-axis of the ROC curve
FPR = zeros(1, n_gamma); % False Positive Rate, x-axis of the ROC curve
P_error=zeros(1, n_gamma);
true_positives = [];
false_positives = [];

for i = 1: n_gamma
decision = (discriminator>=thresholdList(i));
TP = sum(decision == 1 & label == 1); % True Positives
FP = sum(decision == 1 & label == 0); % False Positives
FN = sum(decision == 0 & label == 1); % False Negatives
TN = sum(decision == 0 & label == 0); % True Negatives
TPR(i) = TP / (TP + FN); % record the current value
FPR(i) = FP / (FP + TN);
P_error(i)=(FP+FN)/N;
end

figure(3);% Plot ROC Curve
plot(FPR, TPR, 'o');
xlabel('False Positive Rate (FPR)');
ylabel('True Positive Rate (TPR)');
title('ROC Curve for Fisher LDA Classifier (Equal Class Weights)');
grid on; hold on;

best_gamma=find(P_error==min(P_error));
gamma3_best = thresholdList(best_gamma);

for i=1:size(best_gamma)
plot( FPR(best_gamma(i)),  TPR(best_gamma(i)), 'kp', 'MarkerSize', 15, 'MarkerFaceColor', 'y');
hold on;
end
min_Perror_3 = min(P_error);

function g = evalGaussian(x,mu,Sigma)
% Evaluates the Gaussian pdf N(mu,Sigma) at each coumn of X
[n,N] = size(x);
C = ((2*pi)^n * det(Sigma))^(-1/2);
E = -0.5*sum((x-repmat(mu,1,N)).*(inv(Sigma)*(x-repmat(mu,1,N))),1);
g = C*exp(E);
end
```

2. Matlab Code for Question 2

```
% Minimum probability of error classification/ MAP classifier
clear; close all; clc;
%--------------------------------------------------------------------------
%% Part A 1
%--------------------------------------------------------------------------
C = 3; % classes
N = 10000; % sample numbers
p = [0.3 0.3 0.4]; % priors
n = p.*N; % samples per class

mu_1 = [0 0 0]; % let the distances to be 2-3 for each two
mu_2 = [2 1 2];
mu_3_1 = [-1.5 -0.5 1.5];
mu_3_2 = [2.5 1.5 2.5];

sigma1 = diag([2, 1, 0.5]);
sigma2 = diag([1.5, 2, 1]);
sigma3 = diag([0.7, 1.8, 1.2]);

Xclass1 = mvnrnd(mu_1, sigma1, n(1));
Xclass2 = mvnrnd(mu_2, sigma2, n(2));
Xclass3_1 = mvnrnd(mu_3_1, sigma3, n(3)/2);
Xclass3_2 = mvnrnd(mu_3_2, sigma3, n(3)/2);
Xclass3 = [Xclass3_1; Xclass3_2];

X=[Xclass1; Xclass2; Xclass3];
label = [ones(n(1),1); 2*ones(n(2),1); 3*ones(n(3),1)];

%--------------------------------------------------------------------------
%% Part A 2
%--------------------------------------------------------------------------

p_x_given_L1 = mvnpdf(X, mu_1, sigma1);
p_x_given_L2 = mvnpdf(X, mu_2, sigma2);
p_x_given_L3_1 = mvnpdf(X, mu_3_1, sigma3);
p_x_given_L3_2 = mvnpdf(X, mu_3_2, sigma3);
p_x_given_L3 = 0.5*(p_x_given_L3_1+p_x_given_L3_2);

p_x_given_L = [p_x_given_L1, p_x_given_L2 p_x_given_L3]';
p_x = p*p_x_given_L;
classPosteriors = (p_x_given_L .* p') ./ p_x; % P(L=l|x)

loss_matrices = {
   ones(C, C) - eye(C), ... % 0-1 loss
   [0 10 10; 1 0 10; 1 1 0], ...% Define loss matrices for the different cases
   [0 100 100; 1 0 100; 1 1 0]
   }; %

% Loop over different loss matrices
for i = 1:length(loss_matrices)
   loss = loss_matrices{i};
   expectedRisks = loss * classPosteriors;
   [~, predicted_label] = min(expectedRisks, [], 1);

   % Confusion matrix
   conf_mat = confusionmat(label, predicted_label');
   % Normalize the confusion matrix
   conf_mat_norm = conf_mat ./ sum(conf_mat, 2);
   disp(['Confusion Matrix for Loss Function ', num2str(i)]);
   disp(conf_mat_norm);

   figure(i+3);
   heatmap([1 2 3], [1 2 3], conf_mat_norm, 'Title', 'Confusion Matrix', ...
```

```
    'XLabel', 'Predicted Class', 'YLabel', 'True Class');

    figure(i); % Create a new figure for each loss function
    hold on;
    markers = {'o', 'd', '^'};
    correct_class = find(predicted_label' == label);
    incorrect_class = find(predicted_label' ~= label);

    for class_idx = 1:3
        class_samples = find(label == class_idx);
        correct_class_samples = intersect(correct_class, class_samples);
        incorrect_class_samples = intersect(incorrect_class, class_samples);
        % Correct classifications: unfilled green markers
        scatter3(X(correct_class_samples, 1), X(correct_class_samples, 2), X(correct_class_samples, 3), ...
            50, 'MarkerEdgeColor', [0 0.7 0], 'Marker', markers{class_idx}, 'LineWidth', 1.5);
        % Incorrect classifications: unfilled red markers
        scatter3(X(incorrect_class_samples, 1), X(incorrect_class_samples, 2), X(incorrect_class_samples, 3), ...
            50, 'MarkerEdgeColor', [0.8 0 0], 'Marker', markers{class_idx}, 'LineWidth', 1.5);
    end

    view(3); % 3D view
    grid on;
    xlabel('X1'); ylabel('X2'); zlabel('X3');
    title(['3D Classification Results with Loss Function ', num2str(i)]);
    legend('Class 1 - Correct', 'Class 1 - Incorrect', ...
        'Class 2 - Correct', 'Class 2 - Incorrect', ...
        'Class 3 - Correct', 'Class 3 - Incorrect');
    hold off;
end

figure(7); % show the distribution of X
scatter3(X(label==1,1), X(label==1,2), X(label==1,3), 'ro'); % Class 1 with stars
hold on;
scatter3(X(label==2,1), X(label==2,2), X(label==2,3), 'gd'); % Class 2 with diamonds
scatter3(X(label==3,1), X(label==3,2), X(label==3,3), 'b^'); % Class 3 with triangles
xlabel('X1'); ylabel('X2'); zlabel('X3');
title('generated samples');
legend('Class 1', 'Class 2', 'Class 3');
grid on;
hold off;
```

Appendix C

3. Matlab Code for Question 3 Wine

```
% Minimum probability of error classifiers
clear; close all; clc;
%-------------------------------------------------------------------------
%% Wine Quality dataset
%-------------------------------------------------------------------------
data_path = '/Users/ln915/Documents/Lvx/2024_Fall/Intro2ML/Assignments/1/data/wine+quality/';
red_wine = readmatrix([data_path 'winequality-red.csv']);
white_wine = readmatrix([data_path 'winequality-white.csv']);

% Separate features and labels
x = white_wine(:,1:end-1);
label =white_wine(:,end);

classes = unique(label); % although labels should be 0-10, but only 3 4 5 6 7 8
C = length(classes);  % Number of classes
[N, d] = size(x);     % N = number of samples, d = number of features

mu = zeros(C, d);
cov_matrices = cell(C, 1);
priors = zeros(C, 1);

lambda = 1e-2;% Regularization parameter (lambda)

for i = 1:C % Estimate mean vectors, covariance matrices, and class priors
    class_samples = x(label == classes(i), :);
    mu(i, :) = mean(class_samples);
    cov_sample = cov(class_samples);
    if abs(det(cov_sample)) < 1e-6     % Regularize covariance matrix if needed
        cov_matrices{i} = cov_sample + lambda * eye(d);
    end
    priors(i) = size(class_samples, 1) / N;
end

%-------------------------------------------------------------------------
%% Classify the samples using minimum-P(error) classification rule
%-------------------------------------------------------------------------

for i=1:C
    p_x_given_c(:, i)=mvnpdf(x, mu(i,:), cov_matrices{i});
end

p_x=p_x_given_c*priors;
classPosteriors = (p_x_given_c .* priors') ./ p_x; % P(C=c|x)

predicted_labels = zeros(N, 1);
loss = ones(C, C) - eye(C);
expectedRisks = loss * classPosteriors';

[~, predicted_label] = min(expectedRisks, [], 1);
predicted_labels = classes(predicted_label);

% Calculate confusion matrix
conf_mat = confusionmat(label, predicted_labels);
disp('Confusion Matrix:');
disp(conf_mat);

% Calculate classification error
error_rate = sum(predicted_labels ~= label) / N;
fprintf('Classification Error Rate: %.2f%%\n', error_rate * 100);

figure;
heatmap(classes, classes, conf_mat, 'Title', 'Confusion Matrix', ...
    'XLabel', 'Predicted Class', 'YLabel', 'True Class');
```

```
%--------------------------------------------------------------------------
%% Visulization: use PCA to reduce the dimensions
%--------------------------------------------------------------------------

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

figure;
gscatter(score(:,1), score(:,2), label);
xlabel('1st Principal Component');
ylabel('2nd Principal Component');
title('Wine Quality Dataset: PCA Visualization (First 2 Components)');
legend('show');

figure;
scatter3(score(:,1), score(:,2), score(:,3), 10, label, 'filled');
xlabel('1st Principal Component');
ylabel('2nd Principal Component');
zlabel('3rd Principal Component');
title('Wine Quality Dataset: PCA Visualization (First 3 Components)');
grid on;
```

4. Matlab Code for Question 3 Human activity

```
% Minimum probability of error classifiers
clear; close all; clc;
%--------------------------------------------------------------------------
%% human activity recognition using smartphones
%--------------------------------------------------------------------------
data_path_train =
'/Users/ln915/Documents/Lvx/2024_Fall/Intro2ML/Assignments/1/data/human+activity+recognition+using+smartphones/UCI_HAR_Dataset/train/';
data_path_test =
'/Users/ln915/Documents/Lvx/2024_Fall/Intro2ML/Assignments/1/data/human+activity+recognition+using+smartphones/UCI_HAR_Dataset/test/';

% Load the training data
X_train = readmatrix([data_path_train 'X_train.txt']);
y_train = readmatrix([data_path_train 'y_train.txt']);

% Load the testing data
X_test = readmatrix([data_path_test 'X_test.txt']);
y_test = readmatrix([data_path_test 'y_test.txt']);

% combine train and test
x = [X_train; X_test];
y=[y_train; y_test];

classes = unique(y); % although labels should be 0-10, but only 3 4 5 6 7 8
C = length(classes);  % Number of classes
[N, d] = size(x);     % N = number of samples, d = number of features

mu = zeros(C, d);
cov_matrices = cell(C, 1);
priors = zeros(C, 1);

lambda = 1e-2;% Regularization parameter (lambda)
% alpha=5;
for i = 1:C % Estimate mean vectors, covariance matrices, and class priors
    class_samples = x(y == classes(i), :);
    mu(i, :) = mean(class_samples);
    cov_sample = cov(class_samples);
    if abs(det(cov_sample)) < 1e-6     % Regularize covariance matrix if needed
        % lambda=alpha*trace(cov_sample)/rank(cov_sample);
        cov_matrices{i} = cov_sample + lambda * eye(d);
    end
    priors(i) = size(class_samples, 1) / N;
end
```

```
%-------------------------------------------------------------------------
%% Classify the samples using minimum-P(error) classification rule
%-------------------------------------------------------------------------

for i=1:C
    p_x_given_c(:, i)=mvnpdf(x, mu(i,:), cov_matrices{i});
end

p_x=p_x_given_c*priors;
classPosteriors = (p_x_given_c .* priors') ./ p_x; % P(C=c|x)

loss = ones(C, C) - eye(C);
expectedRisks = loss * classPosteriors';

[~, predicted_label] = min(expectedRisks, [], 1);
predicted_labels = classes(predicted_label);

% Calculate confusion matrix
conf_mat = confusionmat(y, predicted_labels);
disp('Confusion Matrix:');
disp(conf_mat);

% Calculate classification error
error_rate = sum(predicted_labels ~= y) / N;
fprintf('Classification Error Rate: %.2f%%\n', error_rate * 100);

figure;
heatmap(classes, classes, conf_mat, 'Title', 'Confusion Matrix', ...
    'XLabel', 'Predicted Class', 'YLabel', 'True Class');

%-------------------------------------------------------------------------
%% Visulization: use PCA to reduce the dimensions
%-------------------------------------------------------------------------

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

figure;
gscatter(score(:,1), score(:,2), y);
xlabel('1st Principal Component');
ylabel('2nd Principal Component');
title('human activity recognition: PCA Visualization (First 2 Components)');
legend('show');

figure;
scatter3(score(:,1), score(:,2), score(:,3), 10, y, 'filled');
xlabel('1st Principal Component');
ylabel('2nd Principal Component');
zlabel('3rd Principal Component');
title('human activity recognition: PCA Visualization (First 3 Components)');
grid on;
```