ECE 271A: Statistical Learning I

Homework 2

Name	:	Sanchit Gupta
PID	:	A59010276
Date of Submission	:	24th October 2022

Solution for Question 6

a) The maximum likelihood estimate for the prior probabilities is calculated by taking a ratio of the length of foreground and background training data with the total length of the training data respectively. Foreground corresponds to cheetah and background corresponds to grass. The prior probabilities obtained are as follows:

- P_Y (cheetah) = 0.1919
- P_Y (grass) = 0.8081

The histogram for the prior probabilities is shown in Figure 1. These estimates are actually the same as obtained in Homework 1.

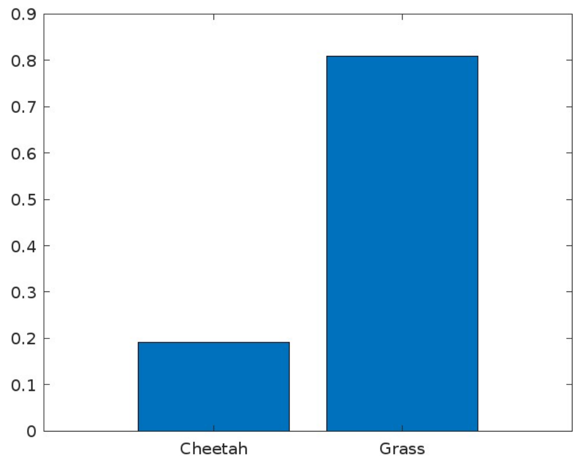

Figure 1: Histogram of prior probabilities

The code snippet written to calculate the prior probabilities is shown below.

```
% Part a: Calculation of Prior Probabilities
length_TrainSampleFG     = length(TrainsampleDCT_FG);
length_TrainSampleBG     = length(TrainsampleDCT_BG);

P_cheetah                = length_TrainSampleFG / (length_TrainSampleFG +
length_TrainSampleBG);
P_grass                  = length_TrainSampleBG / (length_TrainSampleFG +
length_TrainSampleBG);

Y                        = [P_cheetah, P_grass];
X                        = categorical({'Cheetah','Grass'});
bar(X,Y)
saveas(gcf, 'Prior_histogram.jpg')
```

b) The plots for marginal densities for two classes

$$P_{(X|Y)}\ (x \mid cheetah)\ and\ P_{(X|Y)}\ (x \mid grass), k = 1, \ldots\ldots, 64$$

are shown in Figures 2 and 3. Blue line in the plots represents $P_{(X|Y)}\ (x \mid cheetah)$ and red line represents $P_{(X|Y)}\ (x \mid grass)$.

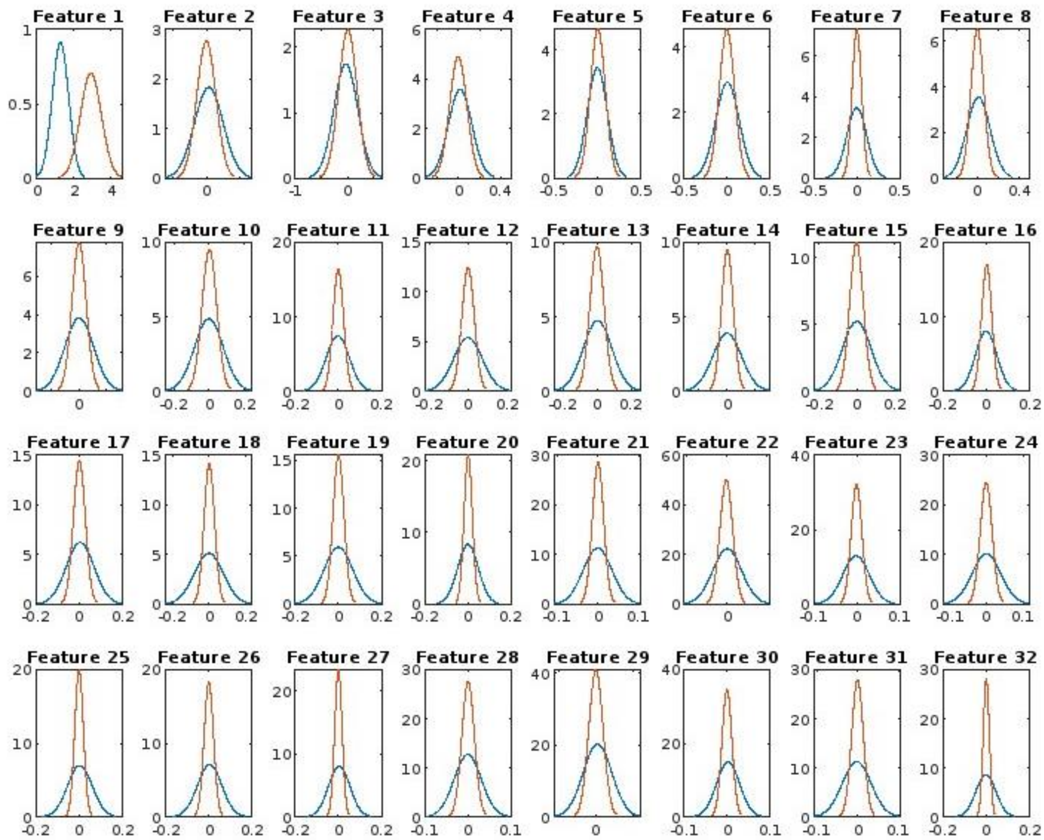

Figure 2: Marginal densities plots for features 1 to 32

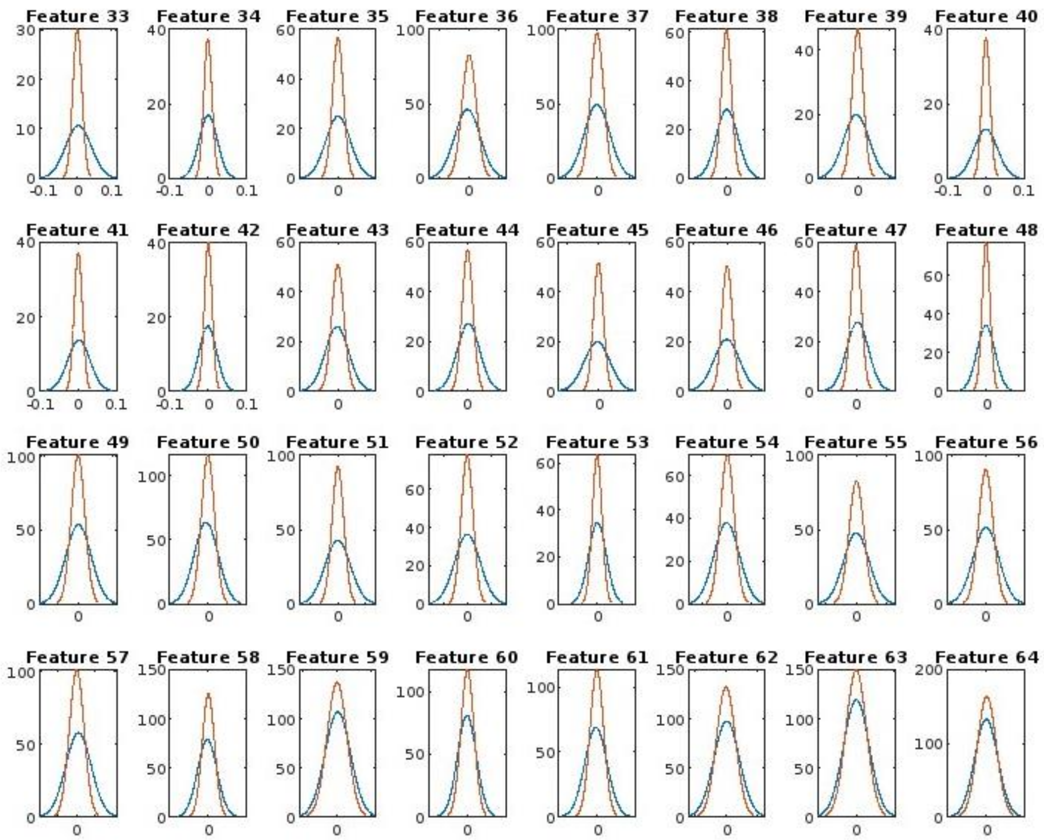

Figure 3: Marginal densities plots for features 33 to 64

By visual inspection, the best and worst 8 features are selected. They are listed as below.

Best 8 features: [1, 12, 19, 21, 27, 32, 40, 47]

Worst 8 features: [3, 4, 5, 6, 59, 62, 63, 64]

The marginal densities plots for the 8 best features are shown in Figure 4 and for the worst 8 features is shown in Figure 5. Blue line in the plots represents $P_{(X|Y)}\,(x \mid cheetah)$ and red line represents $P_{(X|Y)}\,(x \mid grass)$.

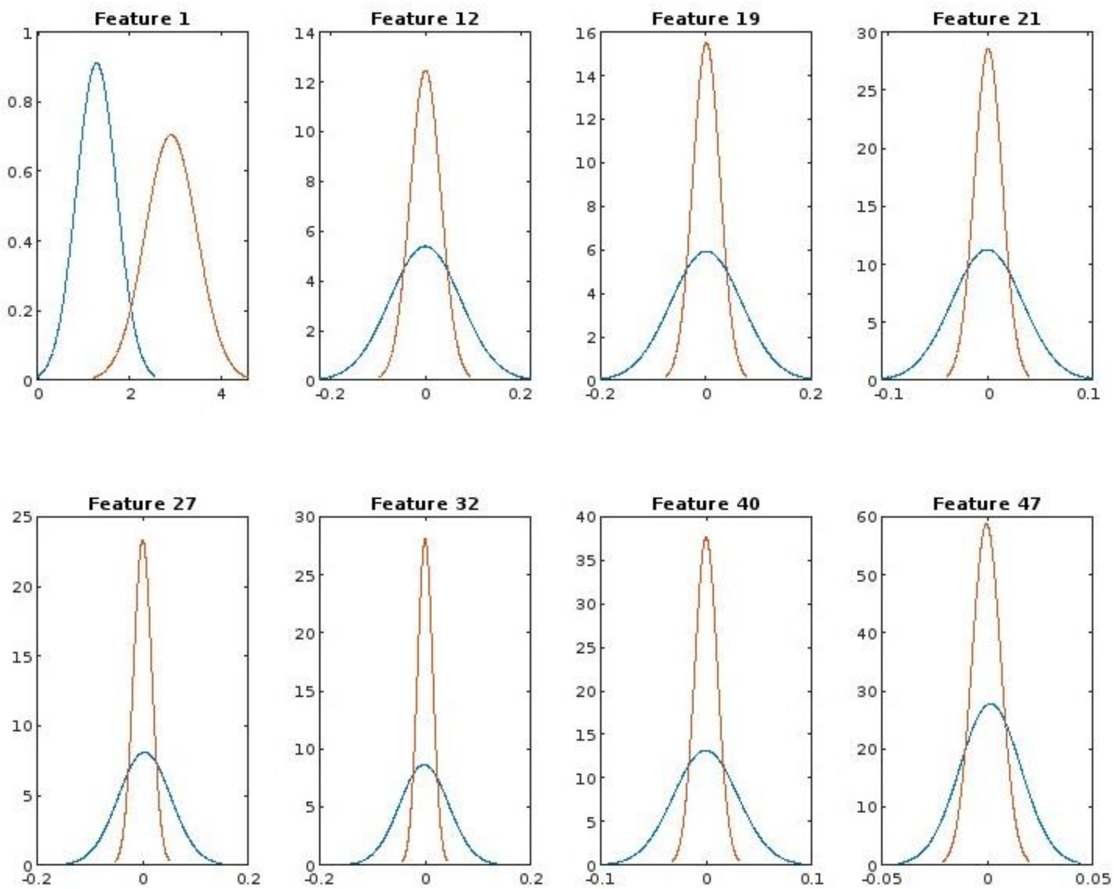

Figure 4: Marginal densities plots for 8 best features

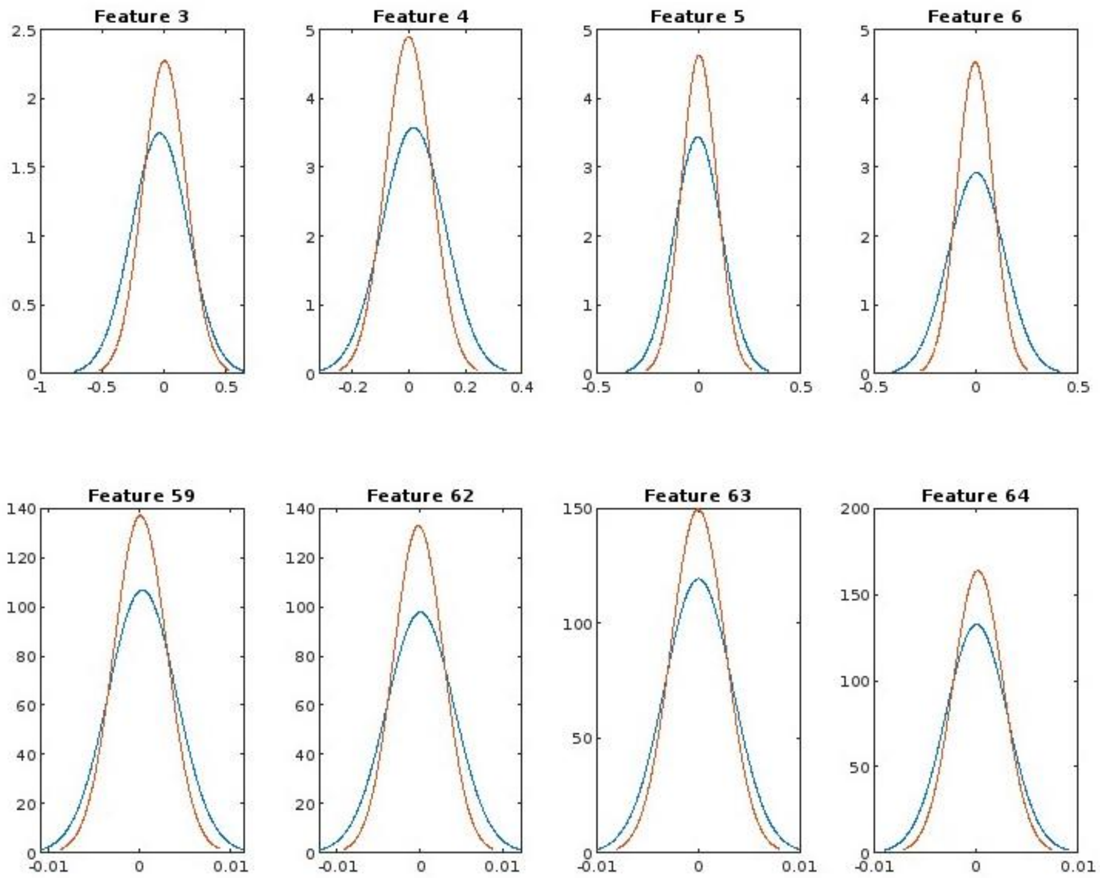

Figure 5: Marginal densities plots for 8 worst features

c) The Bayesian decision rule to classify between cheetah and grass classes is implemented using the expression given below:

$$P_{(X|Y)}(x \mid i) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu_i)^T\,\Sigma^{-1}(x-\mu_i)\right)$$

After the computation of DCT coefficients for different blocks in the input image ‘cheetah.bmp’, the class conditional probabilities $P_{(X|Y)}\,(x \mid cheetah)$ and $P_{(X|Y)}\,(x \mid grass)$ are calculated for each DCT vector. For the mask creation, the pixel is classified as 1 if the class conditional probability $P_{(X|Y)}\,(x \mid cheetah)$ is greater than $P_{(X|Y)}\,(x \mid grass)$ and 0 otherwise. Padding is not implemented in the creation of mask.

The mask obtained using 64 dimensional Gaussians is shown in Figure 6 and the one obtained using 8 dimensional Gaussians of best features is shown in Figure 7.

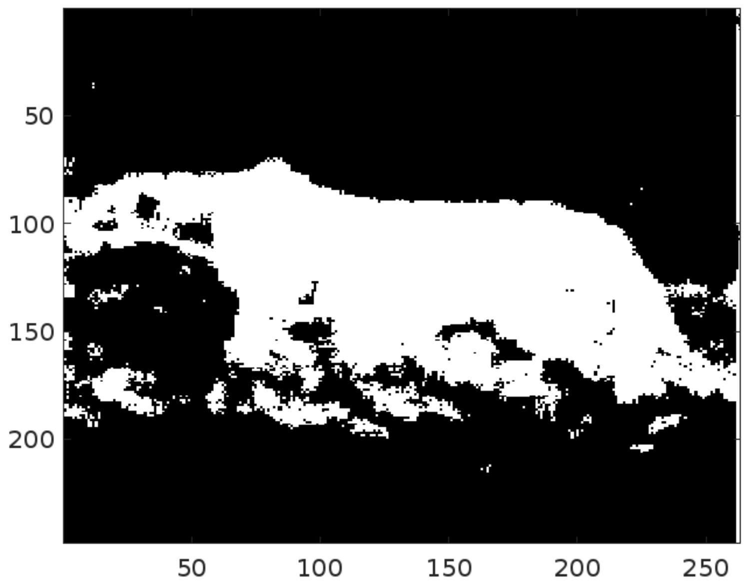

Figure 6: Mask obtained using 64 dimensional Gaussians for the given input image 'cheetah.bmp'

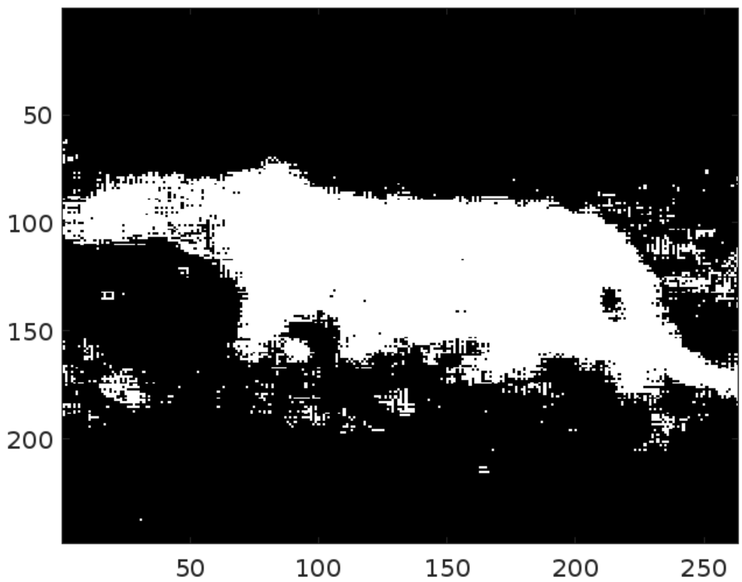

Figure 7: Mask obtained using 8 dimensional Gaussians of best features for the given input image 'cheetah.bmp'

d) The probability of error for the mask obtained using 64 dimensional Gaussians, as shown in Figure 6, is **0.0922** or **9.22%**.

The probability of error for the mask obtained using 8 dimensional Gaussians of best features, as shown in Figure 7, is **0.0575** or **5.75%**.

It can be concluded that the mask obtained using best features has less probability of error in comparison to that of the mask obtained using all 64 features. Since many worst features are also included in the 64 features, they make the classification using Bayesian Decision Rule worse. It can be seen in the marginal densities plots of 8 best features that there is significant difference in the probabilities of foreground and background, which help the Bayesian Decision Rule to perform better.

Full code written to solve the quiz is shown below.

```
clc; clear; close all;
load("TrainingSamplesDCT_8_new.mat")

% Part a: Calculation of Prior Probabilities
length_TrainSampleFG    = length(TrainsampleDCT_FG);
length_TrainSampleBG    = length(TrainsampleDCT_BG);

P_cheetah               = length_TrainSampleFG / (length_TrainSampleFG +
length_TrainSampleBG);
P_grass                 = length_TrainSampleBG / (length_TrainSampleFG +
length_TrainSampleBG);

Y                       = [P_cheetah, P_grass];
X                       = categorical({'Cheetah','Grass'});
bar(X,Y)
saveas(gcf, 'Prior_histogram.jpg')

% Part b: MLE computation and Marginal density plots for both the classes

num_Features    = 64;

% Calculation of mean for both the datasets
mean_FG         = mean(TrainsampleDCT_FG);
mean_BG         = mean(TrainsampleDCT_BG);

% Calculation of standard deviation for both the datasets
std_FG          = std(TrainsampleDCT_FG);
std_BG          = std(TrainsampleDCT_BG);

% Calculation of Gaussian PDF for features and their plots
% 1 corresponds to FG and 2 corresponds to BG

figure;

for i = 1:num_Features
    x1  = (mean_FG(i) - 3*std_FG(i)) : std_FG(i)/100 : (mean_FG(i) + 3*std_FG(i));
    x2  = (mean_BG(i) - 3*std_BG(i)) : std_BG(i)/100 : (mean_BG(i) + 3*std_BG(i));
    y1  = zeros(length(x1),1);
    y2  = zeros(length(x2),1);

    for j = 1:length(x1)
        y1(j) = exp( -( x1(j) - mean_FG(i) )^2 / (2*(std_FG(i)^2)) ) /
(sqrt(2*pi)*std_FG(i));
    end

    for k = 1:length(x2)
        y2(k) = exp( -( x2(k) - mean_BG(i) )^2 / (2*(std_BG(i)^2)) ) /
(sqrt(2*pi)*std_BG(i));
    end
```

```
    if i < 33
        subplot(4,8,i)
        plot(x1,y1)
        hold
        plot(x2,y2)
        title(['Feature ', num2str(i)], 'FontSize', 5)
        ax = gca;
        ax.FontSize = 5;
    elseif i == 33
        saveas(ax, 'Gaussianplots_1.jpg')
        figure;
    end

    if i > 32
        subplot(4,8,i-32)
        plot(x1,y1)
        hold
        plot(x2,y2)
        title(['Feature ', num2str(i)], 'FontSize', 5)
        ax = gca;
        ax.FontSize = 5;
    end

end
saveas(gcf, 'Gaussianplots_2.jpg')

best_Features   = [1, 12, 19, 21, 27, 32, 40, 47];
worst_Features  = [3, 4, 5, 6, 59, 62, 63, 64];

% Plot for 8 best features
figure;
for iter = 1:8
    i   = best_Features(iter);
    x1  = (mean_FG(i) - 3*std_FG(i)) : std_FG(i)/100 : (mean_FG(i) + 3*std_FG(i));
    x2  = (mean_BG(i) - 3*std_BG(i)) : std_BG(i)/100 : (mean_BG(i) + 3*std_BG(i));
    y1  = zeros(length(x1),1);
    y2  = zeros(length(x2),1);

    for j = 1:length(x1)
        y1(j) = exp( -( x1(j) - mean_FG(i) )^2 / (2*(std_FG(i)^2)) ) /
(sqrt(2*pi)*std_FG(i));
    end

    for k = 1:length(x2)
        y2(k) = exp( -( x2(k) - mean_BG(i) )^2 / (2*(std_BG(i)^2)) ) /
(sqrt(2*pi)*std_BG(i));
    end

    subplot(2,4,iter)
    plot(x1,y1)
```

```
    hold
    plot(x2,y2)
    title(['Feature ', num2str(i)], 'FontSize', 8)
    ax = gca;
    ax.FontSize = 5;

end
saveas(gcf, 'Bestfeatures.jpg')

% Plot for 8 worst features
figure;
for iter = 1:8
    i   = worst_Features(iter);
    x1  = (mean_FG(i) - 3*std_FG(i)) : std_FG(i)/100 : (mean_FG(i) + 3*std_FG(i));
    x2  = (mean_BG(i) - 3*std_BG(i)) : std_BG(i)/100 : (mean_BG(i) + 3*std_BG(i));
    y1  = zeros(length(x1),1);
    y2  = zeros(length(x2),1);

    for j = 1:length(x1)
        y1(j) = exp( -( x1(j) - mean_FG(i) )^2 / (2*(std_FG(i)^2)) ) / 
(sqrt(2*pi)*std_FG(i));
    end

    for k = 1:length(x2)
        y2(k) = exp( -( x2(k) - mean_BG(i) )^2 / (2*(std_BG(i)^2)) ) / 
(sqrt(2*pi)*std_BG(i));
    end

    subplot(2,4,iter)
    plot(x1,y1)
    hold
    plot(x2,y2)
    title(['Feature ', num2str(i)], 'FontSize', 8)
    ax = gca;
    ax.FontSize = 5;

end
saveas(gcf, 'Worstfeatures.jpg')

% Part C: Classification of cheetah image to form mask
inputImg     = imread("cheetah.bmp");
inputImg     = im2double(inputImg);
img_Size     = size(inputImg);
img_Width    = img_Size(1);
img_Height   = img_Size(2);
winSize      = 8;
img_DCT      = zeros(img_Width - winSize + 1 * img_Height - winSize + 1, 
num_Features);
state_Y_BG   = zeros(img_Width - winSize + 1, img_Height - winSize + 1);
state_Y_FG   = zeros(img_Width - winSize + 1, img_Height - winSize + 1);
A            = zeros(img_Width - winSize + 1, img_Height - winSize + 1);
```

```
A_best      = zeros(img_Width - winSize + 1, img_Height - winSize + 1);

fileID      = fopen('Zig-Zag Pattern.txt','r');
global zigzag
zigzag      = fscanf(fileID, '%d');

% Calculation of covariance of whole training sample
cov_FG      = cov(TrainsampleDCT_FG);
cov_BG      = cov(TrainsampleDCT_BG);
det_cov_FG  = det(cov_FG);
det_cov_BG  = det(cov_BG);

% Mask formation using all 64 features
for j = 1:img_Height - winSize + 1
    for i = 1:img_Width - winSize + 1
        block                   = inputImg(i:i+winSize-1, j:j+winSize-1);
        block_DCT               = dct2(block);
        dct_Vector              = matrix_to_zigzag_vector(block_DCT);
        P_x_FG                  = exp( -0.5*( (dct_Vector - mean_FG) * inv(cov_FG)
* (dct_Vector - mean_FG)' ) ) / (sqrt( ((2*pi)^num_Features)*det_cov_FG ) );
        P_x_BG                  = exp( -0.5*( (dct_Vector - mean_BG) * inv(cov_BG)
* (dct_Vector - mean_BG)' ) ) / (sqrt( ((2*pi)^num_Features)*det_cov_BG ) );

        if P_x_FG > P_x_BG
            A(i,j)  = 1;
        else
            A(i,j)  = 0;
        end
    end
end

figure;
imagesc(A)
colormap(gray(255))
saveas(gcf,'Mask_64_features.png')

% Calculation of mean and covariance of best features
trainData_BF_FG = zeros(length_TrainSampleFG,8);
trainData_BF_BG = zeros(length_TrainSampleBG,8);

for i = 1:8
    trainData_BF_FG(:,i) = TrainsampleDCT_FG(:, best_Features(i));
    trainData_BF_BG(:,i) = TrainsampleDCT_BG(:, best_Features(i));
end

mean_best_FG        = mean(trainData_BF_FG);
mean_best_BG        = mean(trainData_BF_BG);
cov_best_FG         = cov(trainData_BF_FG);
cov_best_BG         = cov(trainData_BF_BG);
det_cov_best_FG     = det(cov_best_FG);
det_cov_best_BG     = det(cov_best_BG);
```

```
% Mask formation using best features
for j = 1:img_Height - winSize + 1
    for i = 1:img_Width - winSize + 1
        block                   = inputImg(i:i+winSize-1, j:j+winSize-1);
        block_DCT               = dct2(block);
        dct_Vector              = matrix_to_zigzag_vector(block_DCT);
        P_x_FG                  = exp( -0.5*( (dct_Vector(best_Features) -
mean_best_FG) * inv(cov_best_FG) * (dct_Vector(best_Features) - mean_best_FG)' ) )
/ (sqrt( ((2*pi)^num_Features)*det_cov_best_FG ) );
        P_x_BG                  = exp( -0.5*( (dct_Vector(best_Features) -
mean_best_BG) * inv(cov_best_BG) * (dct_Vector(best_Features) - mean_best_BG)' ) )
/ (sqrt( ((2*pi)^num_Features)*det_cov_best_BG ) );

        if P_x_FG > P_x_BG
            A_best(i,j)  = 1;
        else
            A_best(i,j)  = 0;
        end
    end
end

figure;
imagesc(A_best)
colormap(gray(255))
saveas(gcf,'Mask_best_features.png')

% Calculation of probability of error
ground_Truth_Mask   = imread("cheetah_mask.bmp");
ground_Truth_Mask   = im2double(ground_Truth_Mask);

error_64_features   = sum( abs(A - ground_Truth_Mask(1:img_Width - winSize + 1,
1:img_Height - winSize + 1)), "all" );
error_64_features   = error_64_features / (img_Width * img_Height);

error_best_features = sum( abs(A_best - ground_Truth_Mask(1:img_Width - winSize +
1, 1:img_Height - winSize + 1)), "all" );
error_best_features = error_best_features / (img_Width * img_Height);


function dct_vector = matrix_to_zigzag_vector(img_dct_block)
    dct_vector  = zeros(1,64);
    global zigzag
    for i = 1:8
        for j = 1:8
            index = zigzag( (i-1)*8 + j ) + 1;
            dct_vector(index) = img_dct_block(i,j);
        end
    end
end
```