```
clear;
tic;

% Load training set images
source_dir = 1:2;
% Initialize train and test counts
train_count = 1:1807;
test_count = 1808:2107;

% Display the train and test counts
disp('Train count:');
```

```
Train count:
```

```
disp(train_count);
```

```
         1          2          3          4          5          6          7          8
```

```
disp('Test count:');
```

```
Test count:
```

```
disp(test_count);
```

```
      1808       1809       1810       1811       1812       1813       1814       1815
```

```

% Parameters for resizing
resize_width = 120;
resize_height = 120;
vector_length = resize_width * resize_height;

% Preallocate the matrix with a smaller size for training data
train_data = zeros(vector_length, length(source_dir) * length(train_count));
test_data = zeros(vector_length, length(source_dir) * length(test_count));

% Load training images
for dir_idx = source_dir
    for img_idx = train_count
        idx = length(train_count) * (dir_idx - 1) + img_idx;
        directory_name = sprintf('%d', dir_idx);
        fname = sprintf('%d.JPG', img_idx);
        full_filename = fullfile("D:\uvic meng\Summer 2024\Courses\ECE 596A Selected Topic- AI\AI- Project\Data", directory_name, fname);
        A1 = imread(full_filename);
        A = rgb2gray(A1);
        A_resized = imresize(A, [resize_height, resize_width]);
        train_data(:, idx) = A_resized(:);
    end
end

% Load test images
for dir_idx = source_dir
    for img_idx = test_count
        idx = length(test_count) * (dir_idx - 1) + (img_idx - min(test_count) + 1);
        directory_name = sprintf('%d', dir_idx);
        fname = sprintf('%d.JPG', img_idx);
        full_filename = fullfile("D:\uvic meng\Summer 2024\Courses\ECE 596A Selected Topic- AI\AI- Project\Data", directory_name, fname);
        A1 = imread(full_filename);
        A = rgb2gray(A1);
        A_resized = imresize(A, [resize_height, resize_width]);
        test_data(:, idx) = A_resized(:);
    end
end

% Assign class labels for training and test sets
class_train = ones(length(train_count) * length(source_dir), 1);
class_train(length(train_count) + 1 : length(train_count) * length(source_dir)) = 2;
class_test = ones(length(test_count) * length(source_dir), 1);
class_test(length(test_count) + 1 : length(test_count) * length(source_dir)) = 2;
disp('Data input has been finished successfully');
```

```
Data input has been finished successfully
```

% PCA

```matlab
dim_keep = 100;
% a) Find mean of the training data
train_data = train_data';
test_data = test_data';
mean_train = mean(train_data);
mean_test = mean(test_data);

% b) Find cov of the training data
cov_train = cov(train_data, 1);

% c) Find the SVD (Singular Value Decomposition) of the cov matrix of training data
[U, D, V] = svd(cov_train);

% Get the projection Matrix
W = U(:, 1:dim_keep);

% Perform PCA normalization on both training and evaluation sets
train_data_pca = (train_data - mean_train) * W;
test_data_pca  = (test_data - mean_test) * W;
disp('PCA has been finished successfully');
```

```
PCA has been finished successfully
```

% KNN

```
CF = zeros(2, 2);
k = 5;
train_data = train_data_pca';

% Use fitcknn to train the model
Mdl = fitcknn(train_data_pca, class_train, "NumNeighbors", k);

% Predict classes and get scores
[matlab_found, scores] = predict(Mdl, test_data_pca);

for nn = 1:size(test_data, 1)
    found_class = matlab_found(nn);
    true_class  = class_test(nn);
    disp('Test example number:');
    disp(nn);
    CF(true_class, found_class) = CF(true_class, found_class) + 1;
end
```

```
Test example number:
     1
Test example number:
     2
Test example number:
     3
Test example number:
     4
Test example number:
     5
Test example number:
     6
Test example number:
     7
Test example number:
     8
Test example number:
     9
Test example number:
    10
Test example number:
    11
Test example number:
    12
Test example number:
    13
Test example number:
    14
Test example number:
    15
Test example number:
    16
Test example number:
    17
Test example number:
    18
Test example number:
    19
Test example number:
    20
Test example number:
    21
Test example number:
    22
Test example number:
    23
Test example number:
    24
Test example number:
    25
Test example number:
    26
Test example number:
    27
Test example number:
    28
Test example number:
    29
Test example number:
```

```
Test example number:
   566
Test example number:
   567
Test example number:
   568
Test example number:
   569
Test example number:
   570
Test example number:
   571
Test example number:
   572
Test example number:
   573
Test example number:
   574
Test example number:
   575
Test example number:
   576
Test example number:
   577
Test example number:
   578
Test example number:
   579
Test example number:
   580
Test example number:
   581
Test example number:
   582
Test example number:
   583
Test example number:
   584
Test example number:
   585
Test example number:
   586
Test example number:
   587
Test example number:
   588
Test example number:
   589
Test example number:
   590
Test example number:
   591
Test example number:
   592
Test example number:
   593
Test example number:
   594
Test example number:
   595
Test example number:
   596
Test example number:
   597
Test example number:
   598
Test example number:
   599
Test example number:
   600
```

```
CF
```

```
CF = 2×2
   261    39
    80   220
```

```
elapsedTime = toc; % End timing and store elapsed time
toc;
```

```
Elapsed time is 2238.273031 seconds.
```

```
fprintf('Elapsed time: %.2f seconds\n', elapsedTime);
```

```
Elapsed time: 2238.27 seconds
```

```
Accuracy = (CF(1,1) + CF(2,2)) / ((CF(1,1) + CF(2,2) + CF(1,2) + CF(2,1)));
disp('KNN has been finished successfully');
```

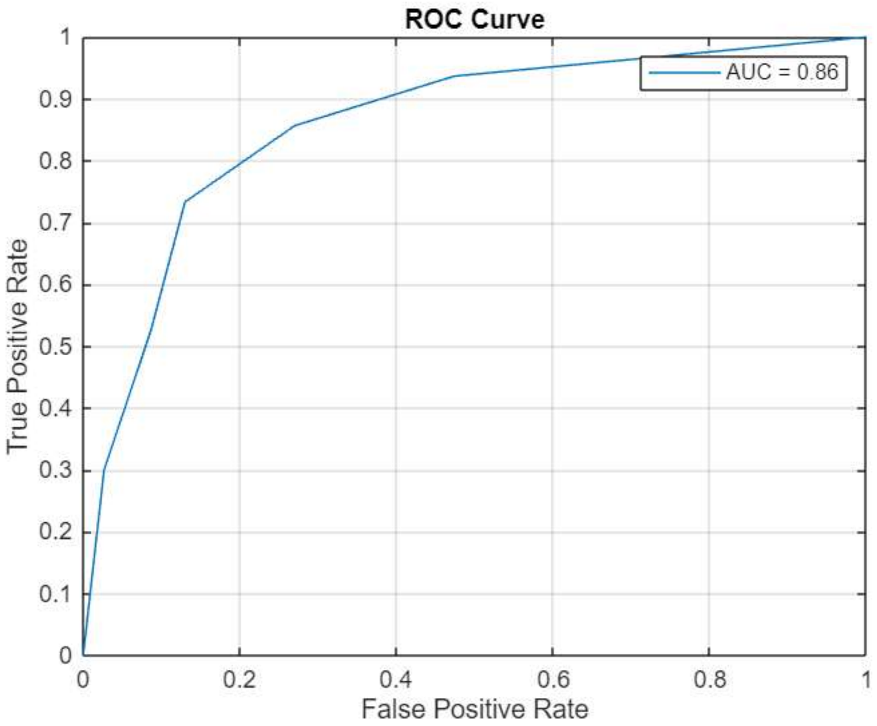
```
KNN has been finished successfully
```

% Final Result

```
TPR = CF(1,1) / (CF(1,1) + CF(1,2));
FNR = CF(1,2) / (CF(1,1) + CF(1,2));
TNR = CF(2,2) / (CF(1,2) + CF(2,2));
FPR = CF(1,2) / (CF(1,2) + CF(2,2));
Precision = CF(1,1) / (CF(1,1) + CF(2,1));
Recall = CF(1,1) / (CF(1,1) + CF(1,2));
F1 = (2 * Precision * Recall) / (Precision + Recall);
```

% Plot ROC curve

```
[X, Y, T, AUC] = perfcurve(class_test, scores(:, 2), 2);
figure;
plot(X, Y);
xlabel('False Positive Rate');
ylabel('True Positive Rate');
title('ROC Curve');
legend(sprintf('AUC = %.2f', AUC));
grid on;
```



```
disp('ROC plot has been finished successfully');
```

```
ROC plot has been finished successfully
```

```
fprintf('TPR = %d\n', TPR);
```

```
TPR = 8.700000e-01
```

```
fprintf('FNR = %d\n', FNR);
```

```
FNR = 1.300000e-01
```

```
fprintf('TNR = %d\n', TNR);
```

```
TNR = 8.494208e-01
```

```
fprintf('FPR = %d\n', FPR);
```

```
FPR = 1.505792e-01
```

```
fprintf('Precision = %d\n', Precision);
```

```
Precision = 7.653959e-01
```

```
fprintf('Recall = %d\n', Recall);
```

```
Recall = 8.700000e-01
```

```
fprintf('F1 Score = %d\n', F1);
```

```
F1 Score = 8.143526e-01
```