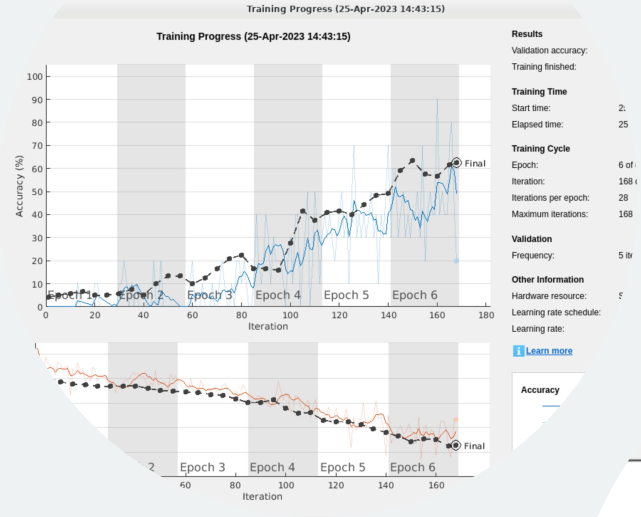
Project 4

Emmitt Hasty

Deliverables



Training Progress (25-Apr-2023 14:43:15)

Training Progress (25-Apr-2023 14:43:15)

Results

Validation accuracy:

Training finished:

Training Time

Start time: 2:

Elapsed time: 25

Training Cycle

Epoch: 6 of

Iteration: 168

Iterations per epoch: 28

Maximum iterations: 168

Validation

Frequency: 5 it

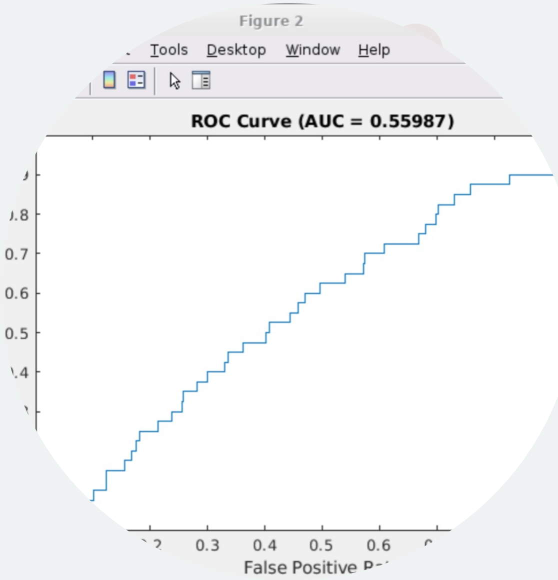
Other Information

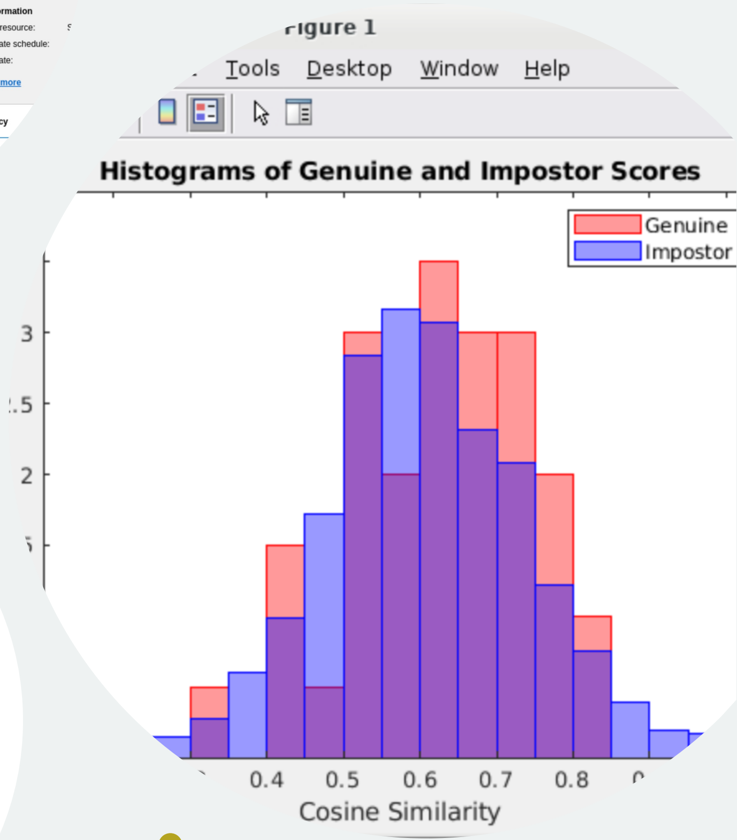
Hardware resource:

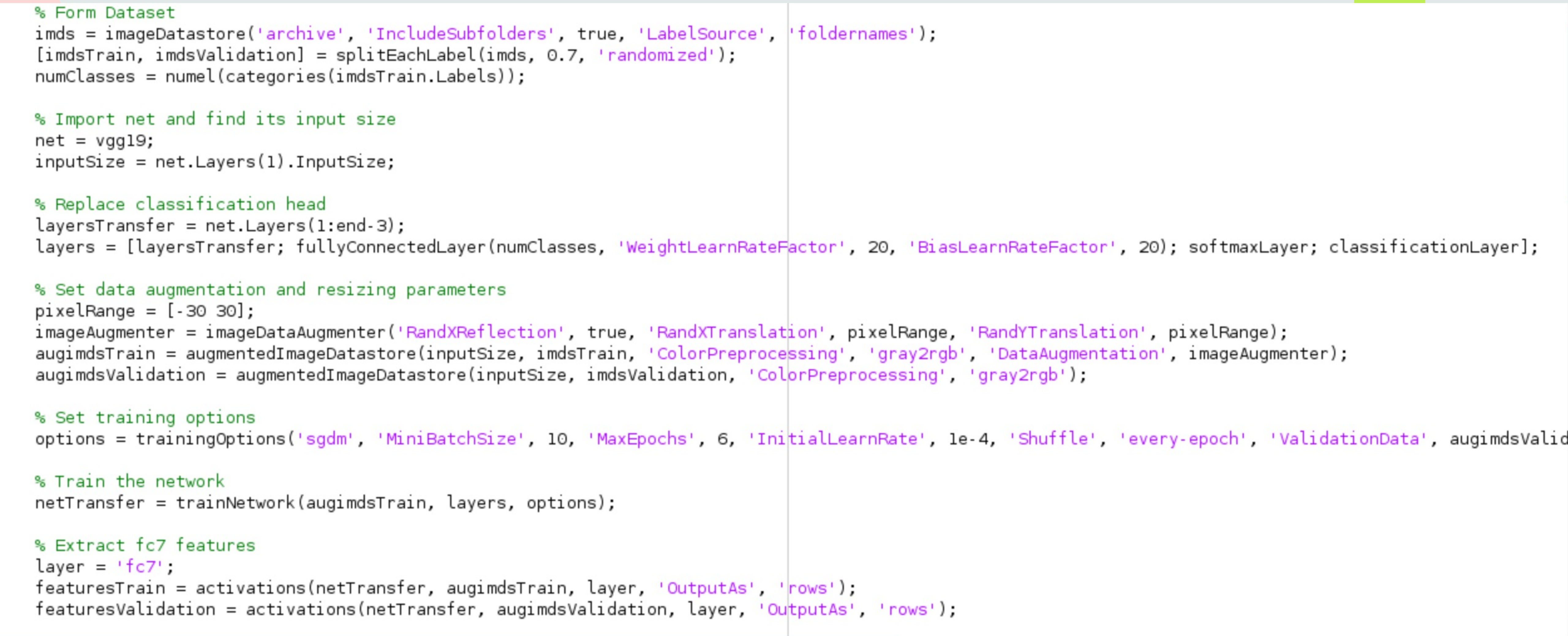
Learning rate schedule:

Learning rate:

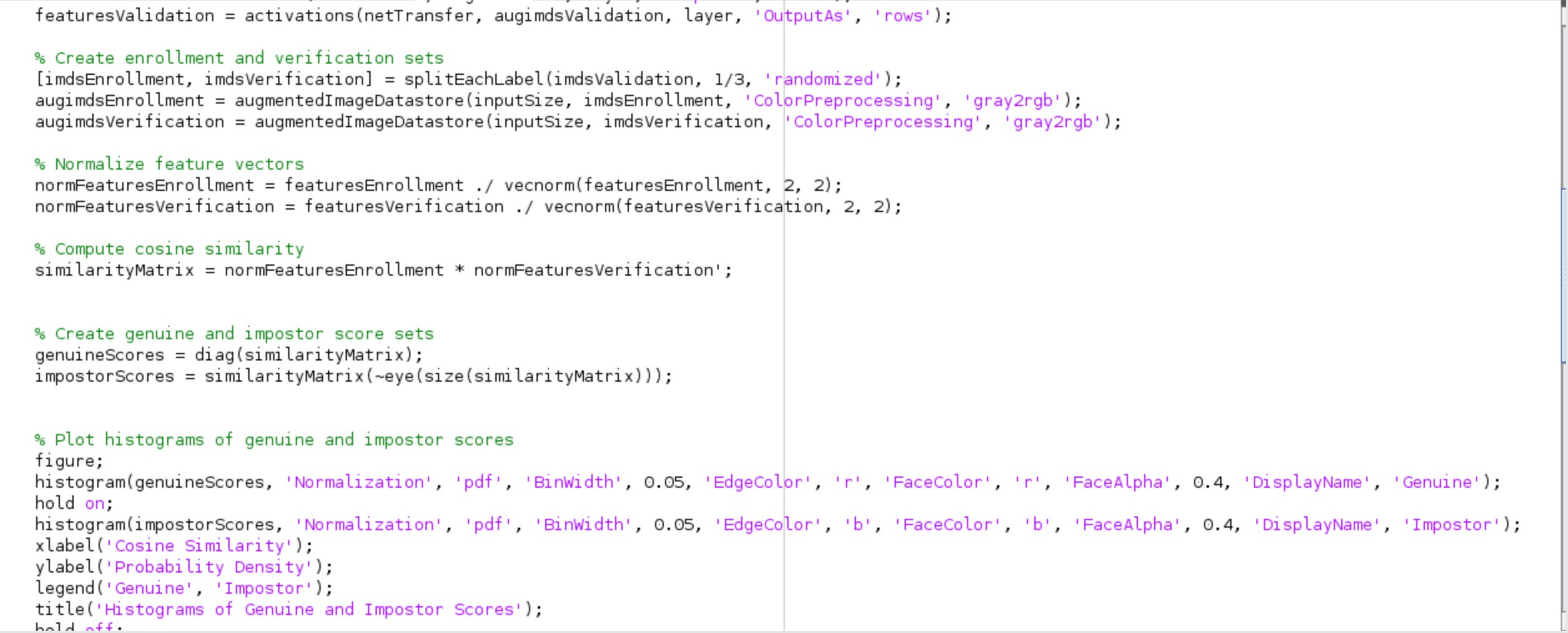
Learn more



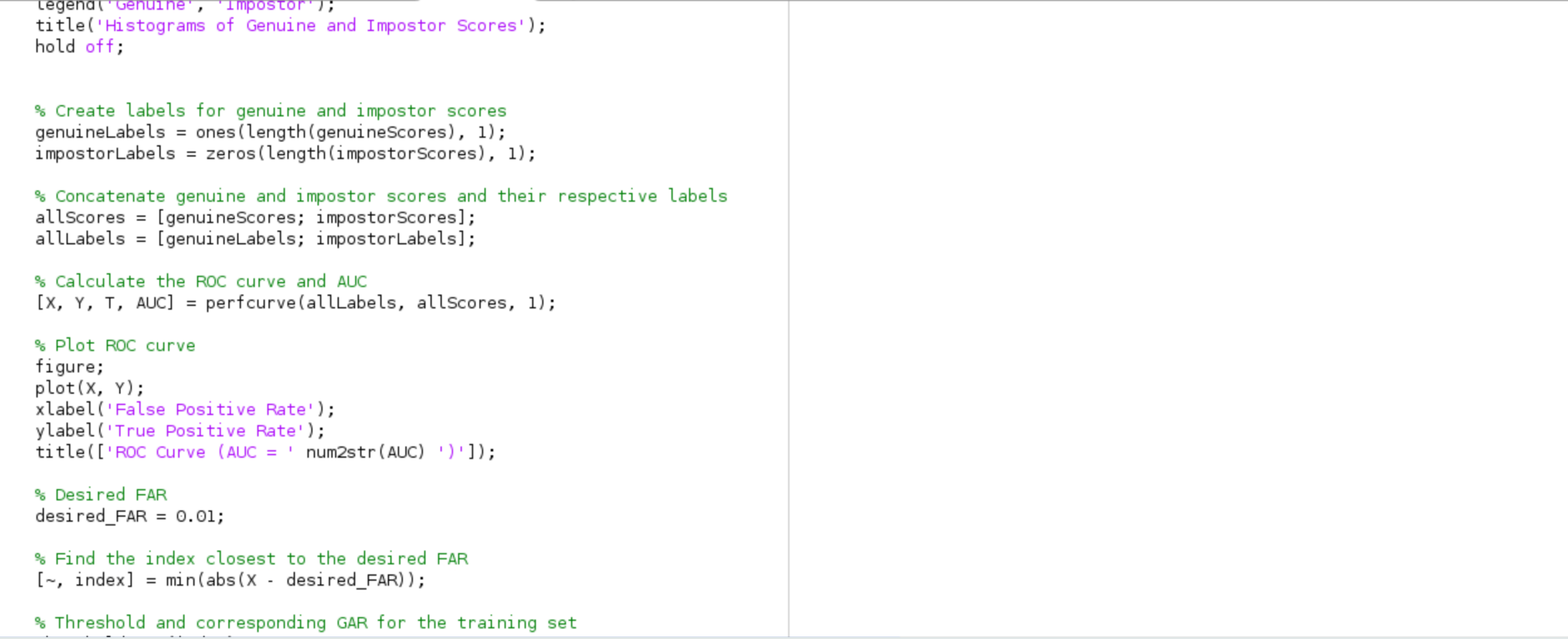




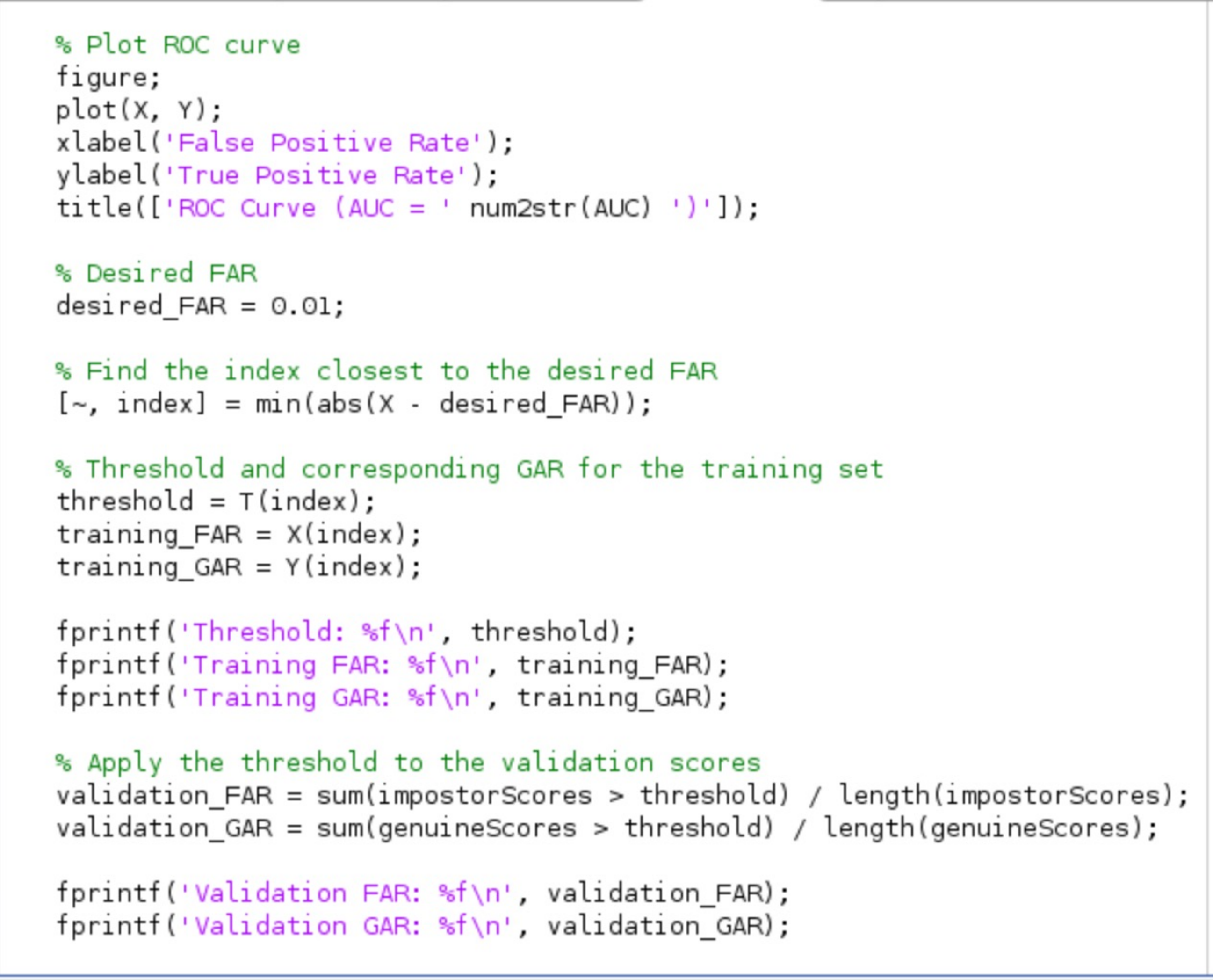
```
% Form Dataset
imds = imageDatastore('archive', 'IncludeSubfolders', true, 'LabelSource', 'foldernames');
[imdsTrain, imdsValidation] = splitEachLabel(imds, 0.7, 'randomized');
numClasses = numel(categories(imdsTrain.Labels));

% Import net and find its input size
net = vgg19;
inputSize = net.Layers(1).InputSize;

% Replace classification head
layersTransfer = net.Layers(1:end-3);
layers = [layersTransfer; fullyConnectedLayer(numClasses, 'WeightLearnRateFactor', 20, 'BiasLearnRateFactor', 20); softmaxLayer; classificationLayer];

% Set data augmentation and resizing parameters
pixelRange = [-30 30];
imageAugmenter = imageDataAugmenter('RandXReflection', true, 'RandXTranslation', pixelRange, 'RandYTranslation', pixelRange);
augimdsTrain = augmentedImageDatastore(inputSize, imdsTrain, 'ColorPreprocessing', 'gray2rgb', 'DataAugmentation', imageAugmenter);
augimdsValidation = augmentedImageDatastore(inputSize, imdsValidation, 'ColorPreprocessing', 'gray2rgb');

% Set training options
options = trainingOptions('sgdm', 'MiniBatchSize', 10, 'MaxEpochs', 6, 'InitialLearnRate', 1e-4, 'Shuffle', 'every-epoch', 'ValidationData', augimdsValid

% Train the network
netTransfer = trainNetwork(augimdsTrain, layers, options);

% Extract fc7 features
layer = 'fc7';
featuresTrain = activations(netTransfer, augimdsTrain, layer, 'OutputAs', 'rows');
featuresValidation = activations(netTransfer, augimdsValidation, layer, 'OutputAs', 'rows');
```



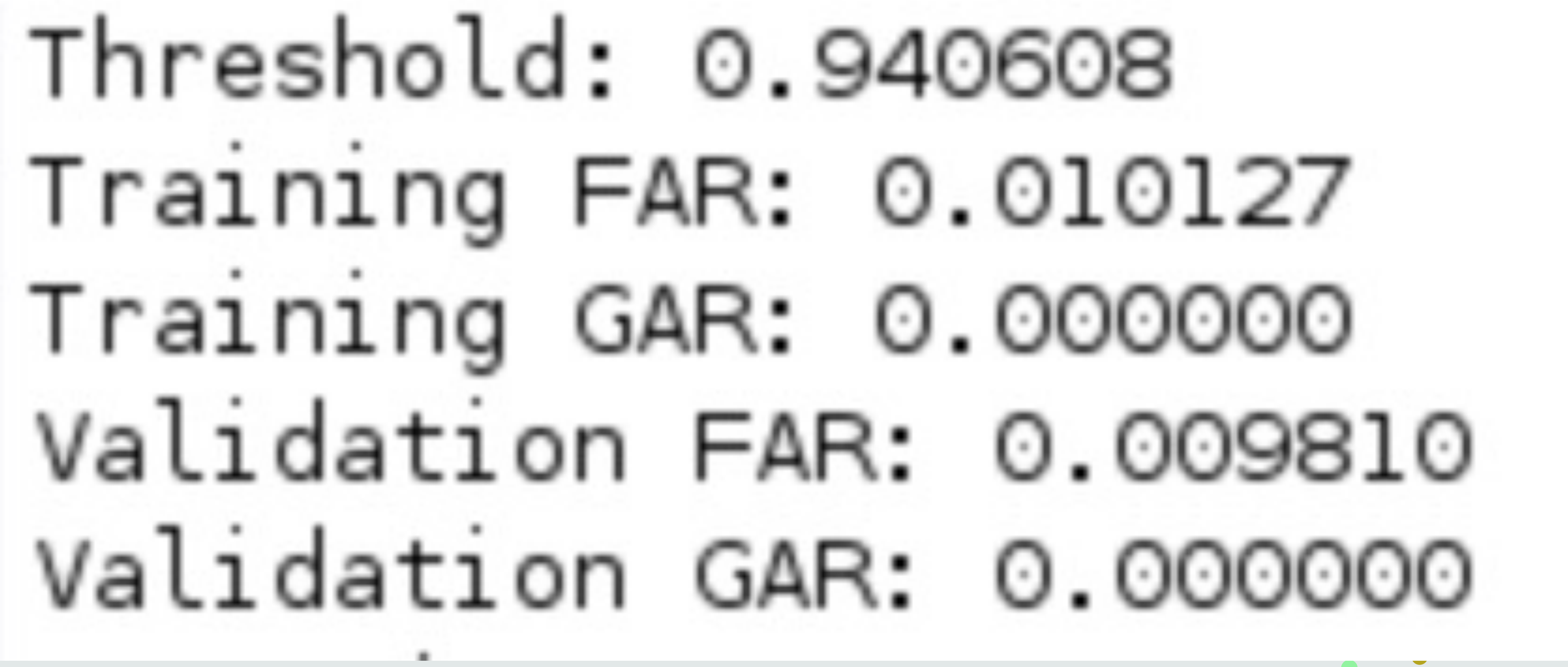
```
featuresValidation = activations(netTransfer, augimdsValidation, layer, 'OutputAs', 'rows');

% Create enrollment and verification sets
[imdsEnrollment, imdsVerification] = splitEachLabel(imdsValidation, 1/3, 'randomized');
augimdsEnrollment = augmentedImageDatastore(inputSize, imdsEnrollment, 'ColorPreprocessing', 'gray2rgb');
augimdsVerification = augmentedImageDatastore(inputSize, imdsVerification, 'ColorPreprocessing', 'gray2rgb');

% Normalize feature vectors
normFeaturesEnrollment = featuresEnrollment ./ vecnorm(featuresEnrollment, 2, 2);
normFeaturesVerification = featuresVerification ./ vecnorm(featuresVerification, 2, 2);

% Compute cosine similarity
similarityMatrix = normFeaturesEnrollment * normFeaturesVerification';


% Create genuine and impostor score sets
genuineScores = diag(similarityMatrix);
impostorScores = similarityMatrix(~eye(size(similarityMatrix)));


% Plot histograms of genuine and impostor scores
figure;
histogram(genuineScores, 'Normalization', 'pdf', 'BinWidth', 0.05, 'EdgeColor', 'r', 'FaceColor', 'r', 'FaceAlpha', 0.4, 'DisplayName', 'Genuine');
hold on;
histogram(impostorScores, 'Normalization', 'pdf', 'BinWidth', 0.05, 'EdgeColor', 'b', 'FaceColor', 'b', 'FaceAlpha', 0.4, 'DisplayName', 'Impostor');
xlabel('Cosine Similarity');
ylabel('Probability Density');
legend('Genuine', 'Impostor');
title('Histograms of Genuine and Impostor Scores');
hold off;
```



```
legend('Genuine', 'Impostor');
title('Histograms of Genuine and Impostor Scores');
hold off;


% Create labels for genuine and impostor scores
genuineLabels = ones(length(genuineScores), 1);
impostorLabels = zeros(length(impostorScores), 1);

% Concatenate genuine and impostor scores and their respective labels
allScores = [genuineScores; impostorScores];
allLabels = [genuineLabels; impostorLabels];

% Calculate the ROC curve and AUC
[X, Y, T, AUC] = perfcurve(allLabels, allScores, 1);

% Plot ROC curve
figure;
plot(X, Y);
xlabel('False Positive Rate');
ylabel('True Positive Rate');
title(['ROC Curve (AUC = ' num2str(AUC) ')']);

% Desired FAR
desired_FAR = 0.01;

% Find the index closest to the desired FAR
[~, index] = min(abs(X - desired_FAR));

% Threshold and corresponding GAR for the training set
```



```
% Plot ROC curve
figure;
plot(X, Y);
xlabel('False Positive Rate');
ylabel('True Positive Rate');
title(['ROC Curve (AUC = ' num2str(AUC) ')']);

% Desired FAR
desired_FAR = 0.01;

% Find the index closest to the desired FAR
[~, index] = min(abs(X - desired_FAR));

% Threshold and corresponding GAR for the training set
threshold = T(index);
training_FAR = X(index);
training_GAR = Y(index);

fprintf('Threshold: %f\n', threshold);
fprintf('Training FAR: %f\n', training_FAR);
fprintf('Training GAR: %f\n', training_GAR);

% Apply the threshold to the validation scores
validation_FAR = sum(impostorScores > threshold) / length(impostorScores);
validation_GAR = sum(genuineScores > threshold) / length(genuineScores);

fprintf('Validation FAR: %f\n', validation_FAR);
fprintf('Validation GAR: %f\n', validation_GAR);
```



```
Threshold: 0.940608
Training FAR: 0.010127
Training GAR: 0.000000
Validation FAR: 0.009810
Validation GAR: 0.000000
```