```matlab
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The Following raw matlab code is used for analyzing actigraphy data captured from sleep study subjects
% undergoing simultaneous sleep study. They have been identified to be having either mild or severe PLMS,
% characterized by PLM Index by the registered technologist.
%
% This code analyzes the actigraphy data by conducting extensive data conditioning, feature extraction and
% and machine learning, to develop a tool for estimating the severity of PLMS during sleep.
%
% This code has been developed through a joint collaboration between the Signal Analysis Research (SAR) lab at Ryerson University
% and Sunnybrook Hospital - and is intended to be used only for experimental and non-profit purposes.
%
% Authors: Yashodhan Athavale, Sridhar Krishnan, Nardin Kirolos, Afsaneh Raissi, Brian J. Murray, Mark I. Boulos
%
% Organizations: Ryerson University and Sunnybrook Hospital, Toronto, Canada
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

plm_sheet = xlsread('FirstData - Ryerson - June 1, 2017.xlsx','Sheet3'); % Read ground truth information

% Left Leg Actigraphy Signals
dsl = dir('C:\Users\Yashodhan\Google Drive\PhD Studies\Research Work\Clinical Journal - TriAxial SimpleFeatExtract\Consolidated Data\Left\*.csv'); % Data directory
dsln = natsortfiles({dsl.name}); % Sort by filename eg. A1, A4, A2, A9, A5 will be sorted as A1,A2,A4,A5,A9
dsln = string(dsln); % Convert to string array

% Right Leg Actigraphy Signals
dsr = dir('C:\Users\Yashodhan\Google Drive\PhD Studies\Research Work\Clinical Journal - TriAxial SimpleFeatExtract\Consolidated Data\Right\*.csv'); % Data directory
dsrn = natsortfiles({dsr.name}); % Sort by filename eg. A1, A4, A2, A9, A5 will be sorted as A1,A2,A4,A5,A9
dsrn = string(dsrn); % Convert to string array

% Initialize Feature-set variables
featsetl = zeros(length(dsln),15); % Left Leg features
featsetr = zeros(length(dsrn),15); % Right Leg features


for i = 1:length(dsln)

    % Check if file exists, otherwise exit to the end of the FOR loop
    if exist(dsln(i)) == 0
            continue;
    end

    x = csvread(dsln(i),1,0); % Read csv data file ignoring first header row
    if mod(x(1,:),1) == zeros(1,3)
        x = x/2048; % Perform data conversion from "g" units as given by actigraph spec sheet
    end

    % Signal Pre-processing & Initial variable setup
    x(isnan(x)) = 0; % remove NaN values
    y = []; % define empty matrix for filtered signal
    for j = 1:3
        x(:,j) = x(:,j) - mean(x(:,j)); % detrend by subtracting mean value from each sample, to remove dc drift
        y(:,j) = filter(acclpfv3,x(:,j)); % apply low pass filter to remove high frequency components
    end
```

```

        v = sqrt(y(:,1).^2 + y(:,2).^2 + y(:,3).^2); % compute vector magnitude signal


        fs = 25; % given sampling frequency

        % Extract Time and Frequency Domain Features from the signal
        % Statistical Features
        [meanf, sd, var, rms, maxi, p2p, p2rms] = basicfeaturesyash(v);
        % peak-to-avg and peak-to-avg power ratio
        [par, papr] = parANDpapr(v, rms);
        % frequency domain features
        [fmed, fmean, sndr, bandp] = frequency_features(v);

        % Compute Periodicity index
        [sig_peaks,sig_peaks_locs] = findpeaks(v,fs,'MinPeakHeight',mean(v),'MinPeakWidth',0.04); % Find peaks in the signal
        interval_count = length(sig_peaks) - 1; % Calculate # of intervals between peak events in NREM sleep
        plmidxt = plm_sheet(i,2);
        periodicity_index = plmidxt/interval_count; % Compute Periodicity index

        % Create feature set for corresponding test subject or patient
        featsetl(i,2:15) = [meanf, sd, var, rms, maxi, p2p, p2rms, par, papr, fmed, fmean, sndr, bandp, periodicity_index];
        if plmidxt <= 15
            featsetl(i,1) = 0;
        elseif plmidxt > 15 && plmidxt <= 30
            featsetl(i,1) = 1;
        elseif plmidxt > 30 && plmidxt <= 50
            featsetl(i,1) = 2;
        elseif plmidxt > 50
            featsetl(i,1) = 3;
        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%% ANALYSE RIGHT LEG ACTIGRAPHY SIGNALS %%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    for i = 1:length(dsrn)

        % Check if file exists, otherwise exit to the end of the FOR loop
        if exist(dsrn(i)) == 0
                continue;
        end

        x = csvread(dsrn(i),1,0); % Read csv data file ignoring first header row
        if mod(x(1,:),1) == zeros(1,3)
            x = x/2048; % Perform data conversion from "g" units as given by actigraph spec sheet
        end

        % Signal Pre-processing & Initial variable setup
        x(isnan(x)) = 0; % remove NaN values
        y = []; % define empty matrix for filtered signal
        for j = 1:3
            x(:,j) = x(:,j) - mean(x(:,j)); % detrend by subtracting mean value from each sample, to remove dc drift
            y(:,j) = filter(acclpfv3,x(:,j)); % apply low pass filter to remove high frequency components
        end

        v = sqrt(y(:,1).^2 + y(:,2).^2 + y(:,3).^2); % compute vector magnitude signal
        fs = 25; % given sampling frequency

        % Extract Time and Frequency Domain Features from the signal
        % Statistical Features
        [meanf, sd, var, rms, maxi, p2p, p2rms] = basicfeaturesyash(v);
        % peak-to-avg and peak-to-avg power ratio
        [par, papr] = parANDpapr(v, rms);
        % frequency domain features
        [fmed, fmean, sndr, bandp] = frequency_features(v);

        % Compute Periodicity index
        [sig_peaks,sig_peaks_locs] = findpeaks(v,fs,'MinPeakHeight',mean(v),'MinPeakWidth',0.04); % Find peaks in the signal
        interval_count = length(sig_peaks) - 1; % Calculate # of intervals between peak events in NREM sleep
        plmidxt = plm_sheet(i,3);
        periodicity_index = plmidxt/interval_count; % Compute Periodicity index

        % Create feature set for corresponding test subject or patient
        featsetr(i,2:15) = [meanf, sd, var, rms, maxi, p2p, p2rms, par, papr, fmed, fmean, sndr, bandp, periodicity_index];
        if plmidxt <= 15
            featsetr(i,1) = 0;
        elseif plmidxt > 15
            featsetr(i,1) = 1;
        elseif plmidxt > 30 && plmidxt <= 50
            featsetr(i,1) = 2;
        elseif plmidxt > 50
            featsetr(i,1) = 3;
        end
    end

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%  PATTERN CLASSIFICATION %%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % Sort feature sets based on labels in first column
    featsetl = sortrows(featsetl,1);
    featsetr = sortrows(featsetr,1);
    % Combine feature sets
    featset = [featsetl;featsetr];
    featset = sortrows(featset,1); % sort rows based on labels in first column

    % Extract Normal and Abnormal datasets
    noridx = find(featset(:,1) == 0);
    norfeat = featset(noridx,:);
    lnor = length(norfeat);
    abnoridx = find(featset(:,1) == 1);
    abnorfeat = featset(abnoridx,:);
    labnor = length(abnorfeat);

    % Create Training set using 70% of feature data
    feattrain = [norfeat(1:round(0.7*lnor),2:15);abnorfeat(1:round(0.7*labnor),2:15)];
    trainlabel = [norfeat(1:round(0.7*lnor),1);abnorfeat(1:round(0.7*labnor),1)];

    % Create Testing set using 30% of feature data
    feattest = [norfeat((round(0.7*lnor))+1:end,2:15);abnorfeat((round(0.7*labnor))+1:end,2:15)];
    testlabel = [norfeat((round(0.7*lnor))+1:end,1);abnorfeat((round(0.7*labnor))+1:end,1)];

    % Apply to Naive-Bayes Classifier
    nbtrain = fitcnb(feattest,testlabel);
    [nbpredict, nbscores] = predict(nbtrain,feattrain);
    nbclassperf = classperf(nbpredict,trainlabel);
    nbmat = confusionmat(nbpredict,testlabel);
    cvnb = crossval(nbtrain,'Leaveout','on'); % Perform cross-validation
    [fpr,tpr,thr,auc,optpt]=perfcurve(trainlabel,nbscores(:,2),1);

    % Apply to LDA Classifier
    ldtrain = fitcdiscr(feattest,testlabel);
    [ldpred,ldscores] = predict(ldtrain,feattrain);
    ldcp = classperf(ldpred,trainlabel);
    ldmat = confusionmat(ldpred,trainlabel);
```

```
[fpld,tpld,thrld,aucld,optptld] = perfcurve(trainlabel,ldscores(:,2),0);

% Apply to SVM
svtrain = svmtrain(feattest,testlabel,'kernel_function','quadratic');
svpred = svmclassify(svtrain,feattrain);
svpred(isnan(svpred))=0;
svmat = confusionmat(svpred,trainlabel);
svcp = classperf(svpred,trainlabel);

% Generate ROC curves
figure;plot(fpr,tpr);title('ROC Curve NB & LDA Classifiers');
xlabel('False Positive Rate');ylabel('True Positive Rate');
hold on; plot(optpt(1),optpt(2),'ro'); plot(fpld,tpld);plot(optptld(1),optptld(2),'ro');
legend('ROC Curve for Naive-Bayes','Optimal Operating Point','ROC Curve for LDA');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%% END OF ALGORITHM %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```