APPENDIX C

SAMPLES CODE OF IMPLEMENTATION AND EVALUATION OF THE PROPOSED METHOD: COMPLEX ACTIVITY RECOGNIZER THROUGH WRIST VELOCITY (CARWV)

This appendix contains samples code for implementation of the proposed method and steps of evaluating it using HAR cycle. It consists of following sections:

Data labelling.

Data segmentation.

Data filtering, features extraction and selection.

Pre-processing.

Summarizing Data.

Visualizing Data.

Modelling Data.

Classification and results evaluation.

%%%%%%%%%%%%% **Data labelling** %%%%%%%%%%%%%%%%%%

It contains steps to prepare labelling file from log files of users.

%%

1. Create Labels (Ms Excel) file contains: sub actions, actions, atomic, and complex.
2. In the raw Text log file Replace duplicate digits in Timestamp column into spaces.
3. Create CSV file: add labels.

//////////////////////

1. Select range: Select first cell, write last cell address in box, then Shift+Enter
2. F2, Fill cell.
3. CTRL+Enter

////////////////////

1. Add labels to log files for activities, experiements, and subjects
2. Delete Zero lables.
3. Devide log to complete Acc and Gyro files.
4. Delete Zero from Acc and Gyro files using filter.

%%%%%%%%%%%%%%% **Data segmentation** %%%%%%%%%%%%%%

It read data from file, extract g=9.80665 force then segments data of accelerometer and gyroscope, for two datasets using MATLAB function y=buffer(x,n,p,'nodelay').

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```
% Randomly partition data between training, test and validation sets
% [trainInd,valInd,testInd] = dividerand(size(X,2),0.7,0.15,0.15);
% featTest = feat(testInd,:);
% actidTest = actid(testInd,:);
%% My Raw data
Acc_tr =dataset('File','Acc_Exp01_User16.csv','Delimiter',',');
Acc_tr_Trans=double(Acc_tr (:,[3:8,10:11]));
Gyro_tr =dataset('File','Gyro_Exp01_User16.csv','Delimiter',',');
Gyro_tr_Trans=double(Gyro_tr (:,[3:5]));
diff=size(Acc_tr_Trans,1)-size(Gyro_tr_Trans,1);
Gyro_tr_Trans=[Gyro_tr_Trans; zeros(diff,size(Gyro_tr_Trans,2))];
Total_tr_Trans=[Gyro_tr_Trans ,Acc_tr_Trans];
% MyDatasetRaw_Train=Total_tr_Trans;
%
MyDatasetRaw_TrainWithoutG_AllSub=MyDatasetRaw_TrainWithoutG_SelSub;
MyDatasetRaw_TrainWithoutG_AllSub=[MyDatasetRaw_TrainWithoutG_AllSub;
Total_tr_Trans];
%end
save('MyDatasetRaw_TrainWithoutG_AllSub.mat','MyDatasetRaw_TrainWithoutG
_AllSub');
```

```
Acc_te =dataset('File','Acc_Exp02_User19.csv','Delimiter',',');

Acc_te_Trans=double(Acc_te (:,[3:8,10:11]));

Gyro_te =dataset('File','Gyro_Exp02_User19.csv','Delimiter',',');

Gyro_te_Trans=double(Gyro_te (:,[3:5]));

diff=size(Acc_te_Trans,1)-size(Gyro_te_Trans,1);

Gyro_te_Trans=[Gyro_te_Trans; zeros(diff,size(Gyro_te_Trans,2))];

Total_te_Trans=[Gyro_te_Trans ,Acc_te_Trans];

% MyDatasetRaw_Test=Total_te_Trans;

% MyDatasetRaw_TestWithoutG_AllSub=MyDatasetRaw_TestWithoutG_SelSub;

MyDatasetRaw_TestWithoutG_AllSub=[MyDatasetRaw_TestWithoutG_AllSub;
Total_te_Trans];

%end

save('MyDatasetRaw_TestWithoutG_AllSub.mat','MyDatasetRaw_TestWithoutG_A
llSub');


%Acc&Gyro_tr extracting g=9.80665 force

%File:D:\Academic\2014-2015\Complex Activities\Implementation\Sample
data\Acc&Gyro&Sound\Processed data\Train

% Acc_tr

% Acc_tr =dataset('File','Acc_Exp01_User01.csv','Delimiter',',');

Acc_tr =dataset('File','Acc_Exp01_User06.csv','Delimiter',',');

Acc_tr_Trans=double(Acc_tr (:,[3:8,10:11]))';%

%Gyro_tr =dataset('File','Gyro_Exp01_User02.csv','Delimiter',',');

%Gyro_tr_Trans=double(Gyro_tr (:,[3:8,10:11])); %

%Total_tr_Trans=[Acc_tr_Trans(:,3:5),Gyro_tr_Trans];

%MyDatasetRaw_Train=[MyDatasetRaw_Train;Total_tr_Trans];

%save('MyDatasetRaw_Train.mat','MyDatasetRaw_Train');

%sub=MyDatasetRaw_TrainWithoutG_SelSub(:,11)==2;

%Acc_tr_Trans=MyDatasetRaw_TrainWithoutG_SelSub(sub,[1:3,7:11])';

g=9.80665;

%Acc_tr_X=Acc_tr_Trans(1,:)/g;

Acc_tr_X=Acc_tr_Trans(1,:)/g;
```

```
Acc_tr_Y=Acc_tr_Trans(2,:)/g;
Acc_tr_Z=Acc_tr_Trans(3,:)/g;
Acc_tr_ActId=Acc_tr_Trans(4,:);
Acc_tr_ActIdSec=Acc_tr_Trans(5,:);
Acc_tr_ActIdTh=Acc_tr_Trans(6,:);
%Acc_tr_Subject=Acc_tr_Trans(8,:);

%% My Buffer
%sub=MyDatasetRaw_TrainWithoutG_SelSub(:,11)==19;
%Acc_tr_Trans=MyDatasetRaw_TrainWithoutG_SelSub(sub,[1:3,7:11])';
%sub=MyDatasetRaw_TrainWithoutG_AllSub(:,11)==20;
%Acc_tr_Trans=MyDatasetRaw_TrainWithoutG_AllSub(sub,[1:3,7:11])';
sub=MyDatasetRaw_TestWithoutG_AllSub(:,11)==19;
Acc_tr_Trans=MyDatasetRaw_TestWithoutG_AllSub(sub,[1:3,7:11])';

Acc_tr_X=Acc_tr_Trans(1,:);
Acc_tr_Y=Acc_tr_Trans(2,:);
Acc_tr_Z=Acc_tr_Trans(3,:);
Acc_tr_ActId=Acc_tr_Trans(4,:);
Acc_tr_ActIdSec=Acc_tr_Trans(5,:);
Acc_tr_ActIdTh=Acc_tr_Trans(6,:);
Acc_tr_Subject=Acc_tr_Trans(8,:);
%n=50;p=25;
n=400;p=200;

x=Acc_tr_X;
y=buffer(x,n,p,'nodelay');
Acc_tr_XResh=y';

x=Acc_tr_Y;
```

```
y=buffer(x,n,p,'nodelay');
Acc_tr_YResh=y';

x=Acc_tr_Z;
y=buffer(x,n,p,'nodelay');
Acc_tr_ZResh=y';

x=Acc_tr_ActId;
y=buffer(x,n,p,'nodelay');
Acc_tr_ActIdResh=y';
Acc_tr_ActIdReshFirst=Acc_tr_ActIdResh(:,1);

x=Acc_tr_ActIdSec;
y=buffer(x,n,p,'nodelay');
Acc_tr_ActIdResh=y';
Acc_tr_ActIdReshSec=Acc_tr_ActIdResh(:,1);

x=Acc_tr_ActIdTh;
y=buffer(x,n,p,'nodelay');
Acc_tr_ActIdResh=y';
Acc_tr_ActIdReshTh=Acc_tr_ActIdResh(:,1);

x=Acc_tr_Subject;
y=buffer(x,n,p,'nodelay');
Acc_tr_SubResh=y';
Acc_tr_SubResh=Acc_tr_SubResh(:,1);


%Gyro_tr_Trans=MyDatasetRaw_TrainWithoutG_SelSub(sub,[4:6])';
%Gyro_tr_Trans=MyDatasetRaw_TrainWithoutG_AllSub(sub,[4:6])';
```

```
Gyro_tr_Trans=MyDatasetRaw_TestWithoutG_AllSub(sub,[4:6])';


Gyro_tr_X=Gyro_tr_Trans(1,:);
Gyro_tr_Y=Gyro_tr_Trans(2,:);
Gyro_tr_Z=Gyro_tr_Trans(3,:);
x=Gyro_tr_X;
y=buffer(x,n,p,'nodelay');
Gyro_tr_XResh=y';


 x=Gyro_tr_Y;
y=buffer(x,n,p,'nodelay');
Gyro_tr_YResh=y';


 x=Gyro_tr_Z;
y=buffer(x,n,p,'nodelay');
Gyro_tr_ZResh=y';
rawSensorDataTrain = [...
    Acc_tr_XResh, Acc_tr_YResh, Acc_tr_ZResh, ...
    Gyro_tr_XResh, Gyro_tr_YResh, Gyro_tr_ZResh, ...
Acc_tr_ActIdReshFirst,Acc_tr_ActIdReshSec,Acc_tr_ActIdReshTh,Acc_tr_SubRes
h];



%MyDatasetRaw_TrainWithoutG_SelSub8WinBuffer=rawSensorDataTrain;
%MyDatasetRaw_TrainWithoutG_SelSub8WinBuffer=[MyDatasetRaw_TrainWitho
utG_SelSub8WinBuffer;rawSensorDataTrain];
%MyDatasetRaw_TrainWithoutG_AllSub8WinBuffer=[MyDatasetRaw_TrainWitho
utG_AllSub8WinBuffer; rawSensorDataTrain];
MyDatasetRaw_TestWithoutG_AllSub8WinBuffer=[MyDatasetRaw_TestWithoutG
_AllSub8WinBuffer; rawSensorDataTrain];
%save('MyDatasetRaw_TestWithoutG_AllSub3WinBuffer.mat','MyDatasetRaw_Te
stWithoutG_AllSub3WinBuffer');
```

```
%save('MyDatasetRaw_TrainWithoutG_AllSub8WinBuffer.mat','MyDatasetRaw_TrainWithoutG_AllSub8WinBuffer');

%save('MyDatasetRaw_TrainWithoutG_SelSub8WinBuffer.mat','MyDatasetRaw_TrainWithoutG_SelSub8WinBuffer');

%% Opportunity Dataset Raw


% Subject 1

% training

sadl1 = load('S1-ADL1.dat');

sadl2 = load('S1-ADL2.dat');

sadl3 = load('S1-ADL3.dat');

% test

sadl4 = load('S1-ADL4.dat');

sadl5 = load('S1-ADL5.dat');



TrainSub=[sadl1;sadl2;sadl3];

TestSub=[sadl4;sadl5];

TrainLabels=ones(size(TrainSub,1),1)*1;

TestLabels=ones(size(TestSub,1),1)*1;


OpportunityDatasetTrain=[TrainSub,TrainLabels];

OpportunityDatasetTest=[TestSub,TestLabels];


% Subject 2

% training

sadl1 = load('S2-ADL1.dat');

%sadl2 = load('S2-ADL2.dat');

sadl3 = load('S2-ADL3.dat');

% test

sadl4 = load('S2-ADL4.dat');
```

```
sadl5 = load('S2-ADL5.dat');

TrainSub=[sadl1;sadl2;sadl3];
TestSub=[sadl4;sadl5];
TrainLabels=ones(size(TrainSub,1),1)*2;
TestLabels=ones(size(TestSub,1),1)*2;

OneSubTr=[TrainSub,TrainLabels];
OneSubTe=[TestSub,TestLabels];

OpportunityDatasetTrain=[OpportunityDatasetTrain; OneSubTr];
OpportunityDatasetTest=[OpportunityDatasetTest; OneSubTe];

% Subject 3
% training
sadl1 = load('S3-ADL1.dat');
sadl2 = load('S3-ADL2.dat');
sadl3 = load('S3-ADL3.dat');
% test
sadl4 = load('S3-ADL4.dat');
sadl5 = load('S3-ADL5.dat');

TrainSub=[sadl1;sadl2;sadl3];
TestSub=[sadl4;sadl5];
TrainLabels=ones(size(TrainSub,1),1)*3;
TestLabels=ones(size(TestSub,1),1)*3;

OneSubTr=[TrainSub,TrainLabels];
OneSubTe=[TestSub,TestLabels];
```

```
OpportunityDatasetTrain=[OpportunityDatasetTrain; OneSubTr];
OpportunityDatasetTest=[OpportunityDatasetTest; OneSubTe];

% Subject 4
% training
sadl1 = load('S4-ADL1.dat');
sadl2 = load('S4-ADL2.dat');
sadl3 = load('S4-ADL3.dat');
% test
sadl4 = load('S4-ADL4.dat');
sadl5 = load('S4-ADL5.dat');

TrainSub=[sadl1;sadl2;sadl3];
TestSub=[sadl4;sadl5];
TrainLabels=ones(size(TrainSub,1),1)*4;
TestLabels=ones(size(TestSub,1),1)*4;

OneSubTr=[TrainSub,TrainLabels];
OneSubTe=[TestSub,TestLabels];

OpportunityDatasetTrain=[OpportunityDatasetTrain; OneSubTr];
OpportunityDatasetTest=[OpportunityDatasetTest; OneSubTe];

%selectedCol = [51:56 244:245 248 250:251]; % Motion Jacket only
selectedCol = [51:56 245 251]; % Motion Jacket only
OpportunityDatasetTrain = OpportunityDatasetTrain(:,selectedCol);
OpportunityDatasetTest = OpportunityDatasetTest(:,selectedCol);

NullValuesTrain=isnan(OpportunityDatasetTrain(:,1));
NullValuesTest=isnan(OpportunityDatasetTest(:,1));
```

```
OpportunityDatasetTrain(NullValuesTrain,:)=0;OpportunityDatasetTest(NullValues
Test,:)=0;


ZeroLabelTrain=OpportunityDatasetTrain(:,7)==0;ZeroLabelTest=OpportunityDatas
etTest(:,7)==0;

OpportunityDatasetTrain(ZeroLabelTrain,:)=[];
OpportunityDatasetTest(ZeroLabelTest,:)=[];


save('OpportunityDataset.mat','OpportunityDatasetTrain','OpportunityDatasetTest');

%save('OpportunityDatasetAllLevels.mat','OpportunityDatasetTrainAllLevels','Oppo
rtunityDatasetTestAllLevels');


%load('OpportunityDataset');


%% Opportunity Dataset Buffer

%sub=OpportunityDatasetTrain(:,8)==2;

%Acc_tr_Trans=OpportunityDatasetTrain(sub,[1:3,7:8])';

%sub=OpportunityDatasetTrain(:,11)==1;

%Acc_tr_Trans=OpportunityDatasetTrain(sub,[1:3,7:11])';

% Acc_tr_X=Acc_tr_Trans(1,:);

% Acc_tr_Y=Acc_tr_Trans(2,:);

% Acc_tr_Z=Acc_tr_Trans(3,:);

% Acc_tr_ActIdLocM=Acc_tr_Trans(4,:);

% Acc_tr_ActIdTh=Acc_tr_Trans(5,:);

% Acc_tr_ActIdLL=Acc_tr_Trans(6,:);

% Acc_tr_ActIdML=Acc_tr_Trans(7,:);

% Acc_tr_Subject=Acc_tr_Trans(8,:);

%csvwrite('OpportunityDatasetTrain.csv',OpportunityDatasetRaw_TrainWithoutG_
AllSub1WinBuffer)


sub=OpportunityDatasetTest(:,8)==4;

Acc_tr_Trans=OpportunityDatasetTest(sub,[1:3,7:8])';
```

```
Acc_tr_X=Acc_tr_Trans(1,:);
Acc_tr_Y=Acc_tr_Trans(2,:);
Acc_tr_Z=Acc_tr_Trans(3,:);
Acc_tr_ActIdTh=Acc_tr_Trans(4,:);
Acc_tr_Subject=Acc_tr_Trans(5,:);
%fs=30Hz;
%n=30;p=15;
n=240;p=120;

x=Acc_tr_X;
y=buffer(x,n,p,'nodelay');
Acc_tr_XResh=y';

x=Acc_tr_Y;
y=buffer(x,n,p,'nodelay');
Acc_tr_YResh=y';

x=Acc_tr_Z;
y=buffer(x,n,p,'nodelay');
Acc_tr_ZResh=y';

x=Acc_tr_ActIdTh;
y=buffer(x,n,p,'nodelay');
Acc_tr_ActIdResh=y';
Acc_tr_ActIdReshTh=Acc_tr_ActIdResh(:,1);

x=Acc_tr_Subject;
y=buffer(x,n,p,'nodelay');
Acc_tr_SubResh=y';
Acc_tr_SubResh=Acc_tr_SubResh(:,1);
```

```
% x=Acc_tr_ActIdLocM;
% y=buffer(x,n,p,'nodelay');
% Acc_tr_ActIdResh=y';
% Acc_tr_ActIdReshLocM=Acc_tr_ActIdResh(:,1);
%
% x=Acc_tr_ActIdLL;
% y=buffer(x,n,p,'nodelay');
% Acc_tr_ActIdResh=y';
% Acc_tr_ActIdReshLL=Acc_tr_ActIdResh(:,1);
%
% x=Acc_tr_ActIdML;
% y=buffer(x,n,p,'nodelay');
% Acc_tr_ActIdResh=y';
% Acc_tr_ActIdReshML=Acc_tr_ActIdResh(:,1);

%Gyro_tr_Trans=OpportunityDatasetTrain(sub,[4:6])';
Gyro_tr_Trans=OpportunityDatasetTest(sub,[4:6])';
Gyro_tr_X=Gyro_tr_Trans(1,:);
Gyro_tr_Y=Gyro_tr_Trans(2,:);
Gyro_tr_Z=Gyro_tr_Trans(3,:);
x=Gyro_tr_X;
y=buffer(x,n,p,'nodelay');
Gyro_tr_XResh=y';

 x=Gyro_tr_Y;
y=buffer(x,n,p,'nodelay');
Gyro_tr_YResh=y';

 x=Gyro_tr_Z;
```

```
y=buffer(x,n,p,'nodelay');
Gyro_tr_ZResh=y';
rawSensorDataTrain = [...
   Acc_tr_XResh, Acc_tr_YResh, Acc_tr_ZResh, ...
   Gyro_tr_XResh, Gyro_tr_YResh, Gyro_tr_ZResh, ...
   Acc_tr_ActIdReshTh,Acc_tr_SubResh];

% rawSensorDataTrain = [...
%    Acc_tr_XResh, Acc_tr_YResh, Acc_tr_ZResh, ...
%    Gyro_tr_XResh, Gyro_tr_YResh, Gyro_tr_ZResh, ...
%    Acc_tr_ActIdReshLocM, Acc_tr_ActIdReshLL, Acc_tr_ActIdReshML, ...
%    Acc_tr_ActIdReshTh,Acc_tr_SubResh];

%MyDatasetRaw_TrainWithoutG_AllSubBuffer=MyDatasetRaw_TrainWithoutG_S
elSubBuffer;
%OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer=rawSensorDataTrain
;
%OpportunityDatasetRaw_TrainWithoutG_AllSub8WinBuffer=[OpportunityDataset
Raw_TrainWithoutG_AllSub8WinBuffer; rawSensorDataTrain];
OpportunityDatasetRaw_TestWithoutG_AllSub8WinBuffer=[OpportunityDatasetRa
w_TestWithoutG_AllSub8WinBuffer; rawSensorDataTrain];
%save('OpportunityDatasetRaw_TestWithoutG_AllSub8WinBuffer.mat','Opportunit
yDatasetRaw_TestWithoutG_AllSub8WinBuffer');
%save('OpportunityDatasetRaw_TrainWithoutG_AllSub8WinBuffer.mat','Opportuni
tyDatasetRaw_TrainWithoutG_AllSub8WinBuffer');

%Act=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,182)~=0;
%LowLevelAct=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(Act,:)
;
```

%%%%%%%%%% **Data filtering, features extraction and selection** %%%%%%

It tests Complex Activity dataset and public dataset by extracting shallow hand craft features of accelerometer and gyroscope using my function feat=My_Feature_Extraction(acc_x, acc_y, acc_z, gy_x, gy_y, gy_z,fs).

%%

```
%% My Full Features Extraction function
function feat=My_Feature_Extraction(acc_x, acc_y, acc_z, gy_x, gy_y, gy_z,fs)
RawTrainData=[acc_x, acc_y, acc_z, gy_x, gy_y, gy_z];
% Initialize feature vector
feat = zeros(1,132);

% Average value in signal buffer for all three acceleration components (1 each)
feat(1:6) = mean(RawTrainData,1);

% "Decouple" acceleration due to body dynamics from gravity
AccTrData=[acc_x, acc_y, acc_z];

Fstop = 0.4;    % Stopband Frequency
Fpass = 0.8;       % Passband Frequency
Astop = 60;      % Stopband Attenuation (dB)
Apass = 1;       % Passband Ripple (dB)
match = 'passband';  % Band to match exactly

% Construct an FDESIGN object and call its ELLIP method.
h  = fdesign.highpass(Fstop, Fpass, Astop, Apass, fs);
fhp = design(h, 'cheby2', 'MatchExactly', match);
AccTrBody = filter(fhp,AccTrData); % Original AccTrData
RawTrainData=[AccTrBody,  gy_x, gy_y, gy_z];

%AccTeData=[Acc_te_XResh, Acc_te_YResh, Acc_te_ZResh];
```

```
%AccTeBody = filter(fhp,AccTeData);
%RawTestData=[AccTeData, Gyro_te_XResh, Gyro_te_YResh, Gyro_te_ZResh];

% RMS value in signal buffer for all three acceleration components (1 each)
feat(7:12) = rms(RawTrainData,1);

% Autocorrelation as a feature
featsraw = zeros(1,3);
minprom = 0.0005;
mindist_xunits = 0.3;
minpkdist = floor(mindist_xunits/(1/fs));
for k = 1: size(RawTrainData,2)
[c, lags] = xcorr(RawTrainData (:,k)); %Error: array exceeds maximum array size preference
[pks,locs] = findpeaks(c,...
      'minpeakprominence',minprom,...
      'minpeakdistance',minpkdist);

   tc = (1/fs)*lags;
   tcl = tc(locs);
   % Feature 1 - peak height at 0
   if(~isempty(tcl))   % else f1 already 0
      featsraw(3*(k-1)+1) = pks((end+1)/2);
   end
   % Features 2 and 3 - position and height of first peak
   if(length(tcl) >= 3)   % else f2,f3 already 0
      featsraw(3*(k-1)+2) = tcl((end+1)/2+1);
      featsraw(3*(k-1)+3) = pks((end+1)/2+1);
   end
end
feat(13:30)=featsraw(:);
```

```
%Finding Maxima or Peaks of the power spectral density (PSD)
featsraw = zeros(1,6*12);
fmindist = 0.25;                % Minimum distance in Hz
%N = 2*(length(f)-1);            % Number of FFT points
%minpkdist = floor(fmindist/(fs/N)); % Minimum number of frequency bins
nfinalpeaks = 6;
for k = 1: size(RawTrainData,2)
[p,f] = pwelch(RawTrainData(:,k),[],[],[],fs);
% Find max 6 peaks, at least 0.25Hz apart from each other and with a given
prominence value
[pks,locs] = findpeaks(p,'MinPeakDistance',6);
%[pks,locs] = findpeaks(p,'npeaks',8,'minpeakdistance',minpkdist,
'minpeakprominence', 0.15);


opks = zeros(nfinalpeaks,1);
   if(~isempty(pks))
      mx = min(6,length(pks));
      [spks, idx] = sort(pks,'descend');
      slocs = locs(idx);


      pkssel = spks(1:mx);
      locssel = slocs(1:mx);


      [olocs, idx] = sort(locssel,'ascend');
      opks = pkssel(idx);
   end
  ofpk = f(olocs);
 % Features 1-12 positions of highest 6 peaks
   featsraw(12*(k-1)+(1:length(opks))) = ofpk;
   % Features 12-24 power levels of highest 6 peaks
   featsraw(12*(k-1)+(7:7+length(opks)-1)) = opks;
```

```
end
feat(31:102)=featsraw(:);

%Spectral Power
edges = [0.5, 1.5, 5, 10, 15, 20];
[xpsd, f] =periodogram(RawTrainData,[],4096,fs);
featstmp = zeros(5,6);
for kband = 1:length(edges)-1
   featstmp(kband,:) = sum(xpsd( (f>=edges(kband)) & (f<edges(kband+1)), :),1);
end
feat(103:132) = featstmp(:);
end

% Complex Activity dataset features
% Shallow hand craft features
%Acc_tr_XResh=MyDatasetRaw_TrainWithoutG_SelSub8WinBuffer(:,1:50);
%50,100,200,400

Acc_tr_XResh=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,1:50);
Acc_tr_YResh=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,51:100);
Acc_tr_ZResh=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,101:150);
Gyro_tr_XResh=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,151:200);
Gyro_tr_YResh=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,201:250);
Gyro_tr_ZResh=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,251:300);
Acc_tr_ActIdReshTh=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,303);
rawfeat = zeros(size(Acc_tr_XResh,1),132);

Acc_te_XResh=MyDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,1:50);
Acc_te_YResh=MyDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,51:100);
Acc_te_ZResh=MyDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,101:150);
```

```
Gyro_te_XResh=MyDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,151:200);
Gyro_te_YResh=MyDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,201:250);
Gyro_te_ZResh=MyDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,251:300);
Acc_te_ActIdReshTh=MyDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,303);
rawfeat_test = zeros(size(Acc_te_XResh,1),132);

%Gyro=zeros(size(Acc_tr_XResh,1),size(Acc_tr_XResh,2));

fs = 50; % Sampling Frequency
profile -memory on
for k = 1:size(Acc_tr_XResh,1)
   % Extract features for current data buffers
   rawfeat(k,:) = My_Feature_Extraction(Acc_tr_XResh (k,:)', Acc_tr_YResh(k,:)',
Acc_tr_ZResh(k,:)', ...
   Gyro_tr_XResh(k,:)', Gyro_tr_YResh(k,:)', Gyro_tr_ZResh(k,:)', fs);
   fprintf('Buffer #%g\n',k)
end
profile report
profile off

for k = 1:size(Acc_te_XResh,1)
   % Extract features for current data buffers
   rawfeat_test(k,:) = My_Feature_Extraction(Acc_te_XResh (k,:)',
Acc_te_YResh(k,:)', Acc_te_ZResh(k,:)', ...
   Gyro_te_XResh(k,:)', Gyro_te_YResh(k,:)', Gyro_te_ZResh(k,:)', fs);
   fprintf('Buffer #%g\n',k)
end

% Normalise features
%You'd need to normalize each feature by itself before adding to other features.
fmean = mean(rawfeat,1);
```

```
fstd = std(rawfeat,[],1);
feat = bsxfun(@minus,rawfeat,fmean);
feat = bsxfun(@rdivide,feat,fstd);
featFirst=[feat Acc_tr_ActIdReshFirst];
featFirst(isnan(feat))=0;
featSec=[feat Acc_tr_ActIdReshSec];
featSec(isnan(feat))=0;
featTh=[feat MyDataset_tr_ActIdReshTh];
featTh(isnan(feat))=0;
%featTh_selectedFeatures=featTh(:,[1:12,103:133]);


%classificationLearner

featTh=[rawfeat Acc_tr_ActIdReshTh];
featTh(isnan(rawfeat))=0;
featTh_selectedFeatures=featTh(:,[1:12,103:133]);

featTh_test=[rawfeat_test Acc_te_ActIdReshTh];
featTh_test(isnan(rawfeat_test))=0;
featTh_test_selectedFeatures=featTh_test(:,[1:12,103:133]);

% Complex Activity dataset classification
%Xtrain=featTh_selectedFeatures(1:2705,1:42);
Xtrain=featTh_selectedFeatures(:,1:42);
Ytrain=featTh_selectedFeatures(:,43);
Xtest=featTh_test_selectedFeatures(:,1:42);
Ytest=featTh_test_selectedFeatures(:,43);
%2705,1420,710,354
%Xtrain=featTh(1:2705,1:132);
```

```
%Ytrain=featTh(1:2705,133);
%Xtest=featTh(2706:end,1:132);
%Ytest=featTh(2706:end,133);
% Xtrain=fftFeatures/dctFeatures_train;
% Xtest=fftFeatures/dctFeatures_test;
% Xtest=featuresTest; Ytest=Testing_Labels_Bin_G;

% Complex Activity dataset features
% Shallow hand craft features
Acc_tr_XResh=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,1:30)
;
Acc_tr_YResh=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,31:60
);
Acc_tr_ZResh=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,61:90
);
Gyro_tr_XResh=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,91:1
20);
Gyro_tr_YResh=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,121:
150);
Gyro_tr_ZResh=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,151:
180);
Acc_tr_ActIdReshTh=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(
:,181);
rawfeat = zeros(size(Acc_tr_XResh,1),132);

Acc_te_XResh=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,1:30);
Acc_te_YResh=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,31:60)
;
Acc_te_ZResh=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,61:90)
;
Gyro_te_XResh=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,91:1
20);
Gyro_te_YResh=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,121:
150);
```

```
Gyro_te_ZResh=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,151:1
80);

Acc_te_ActIdReshTh=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(:
,181);

rawfeat_test = zeros(size(Acc_te_XResh,1),132);


fs = 30; % Sampling Frequency

for k = 1:size(Acc_tr_XResh,1)

   % Extract features for current data buffers

   rawfeat(k,:) = My_Feature_Extraction(Acc_tr_XResh (k,:)', Acc_tr_YResh(k,:)',
Acc_tr_ZResh(k,:)', ...

   Gyro_tr_XResh(k,:)', Gyro_tr_YResh(k,:)', Gyro_tr_ZResh(k,:)', fs);

   fprintf('Buffer #%g\n',k)

end


for k = 1:size(Acc_te_XResh,1)

   % Extract features for current data buffers

   rawfeat_test(k,:) = My_Feature_Extraction(Acc_te_XResh (k,:)',
Acc_te_YResh(k,:)', Acc_te_ZResh(k,:)', ...

   Gyro_te_XResh(k,:)', Gyro_te_YResh(k,:)', Gyro_te_ZResh(k,:)', fs);

   fprintf('Buffer #%g\n',k)

end


save('OpportunityDatasetFeatures_TrainWithoutG_AllSub1WinBuffer.mat','rawfeat',
'rawfeat_test');

%classificationLearner


featTh=[rawfeat Acc_tr_ActIdReshTh];

featTh(isnan(rawfeat))=0;

featTh_selectedFeatures=featTh(:,[1:12,103:133]);


featTh_test=[rawfeat_test Acc_te_ActIdReshTh];
```

```
featTh_test(isnan(rawfeat_test))=0;

featTh_test_selectedFeatures=featTh_test(:,[1:12,103:133]);

save('OpportunityDatasetFeatures_TrainWithoutG_AllSub1WinBuffer.mat','featTh','
featTh_test');


% Complex Activity dataset classification

Xtrain=featTh(:,1:132);

Ytrain=featTh(:,133);

Xtest=featTh_test(:,1:132);

Ytest=featTh_test(:,133);


Xtrain=featTh_selectedFeatures(:,1:42);

Ytrain=featTh_selectedFeatures(:,43);

Xtest=featTh_test_selectedFeatures(:,1:42);

Ytest=featTh_test_selectedFeatures(:,43);

% Xtrain=fftFeatures/dctFeatures_train;

% Xtest=fftFeatures/dctFeatures_test;

% Xtest=featuresTest; Ytest=Testing_Labels_Bin_G;


%%%%%%%%% %%%Pre-processing %%%%%%%%%%%%%%%%%%%%

Load data from text file, and preparing it to further processing later such as dealing
with missing, outliers, and smoothing data

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clc;clear

%PublicDatasetRaw_Acc =dataset('File','Acc_Exp01.csv','Delimiter',',');

%PublicDatasetRaw_Acc =double(PublicDatasetRaw_Acc);

%PublicDatasetRaw_Gyro =dataset('File','Gyro_Exp01.csv','Delimiter',',');

%PublicDatasetRaw_Gyro =double(PublicDatasetRaw_Gyro);

%PublicDatasetRaw=[PublicDatasetRaw_Acc PublicDatasetRaw_Gyro];

%save('PublicDatasetRaw.mat','PublicDatasetRaw')
```

```
load('MyDatasetRaw_Train.mat', 'MyDatasetRaw_Train')
%g=9.80665;
%Load and Plot Data from Text File
count=MyDatasetRaw_Train(:,1:6);
%count=PublicDatasetRaw_Acc;
%count=MyDatasetRaw_Train;
%sub=count(:,11)==1;
%count=count(sub,:);
%exp=count(:,10)==1;
%count=count(exp,4:6);
[n,p] = size(count)
t = 1:n;
plot(t,count(:,1:3));
legend('Acc_X','Acc_Y','Acc_Z')
xlabel('Time'), ylabel('Coordinate Value')
title('Accelerometer Axis')

plot(t,count(:,4:6));
legend('Gyro_X','Gyro_Y','Gyro_Z')
xlabel('Time'), ylabel('Coordinate Value')
title('Gyroscope Axis')

%Preprocessing
c3=MyDatasetRaw_Train(:,4);

%Missing Data
c3NaNCount = sum(isnan(MyDatasetRaw_Train))
%If you remove NaNs frequently, consider creating a small function that you
%can call
function X = exciseRows(X)
```

```
X(any(isnan(X),2),:) = [];

C = corrcoef(excise(X));

%Outliers (Inconsistent Data)
c3=MyDatasetRaw_TrainWithoutG(:,1);
bin_counts = hist(c3); % Histogram bin counts
N = max(bin_counts); % Maximum bin count
mu3 = mean(c3); % Data mean
sigma3 = std(c3); % Data standard deviation
 hist(c3) % Plot histogram
hold on
plot([mu3 mu3],[0 N],'r','LineWidth',2) % Mean
X = repmat(mu3+(1:2)*sigma3,2,1);
Y = repmat([0;N],1,2);
plot(X,Y,'g','LineWidth',2) % Standard deviations
legend('Data','Mean','Stds')
hold off
%The plot shows that some of the data are more than two standard deviations
%above the mean. If you identify these data as errors (not features), replace
%them with NaN values as follows:
%outliers = (c3 - mu3) > 2*sigma3;
%c3m = c3; % Copy c3 to c3m
%c3m(outliers) = NaN; % Add NaN values

%countWithTh=MyDatasetRaw_TrainWithoutG(:,[1:6,9]);
%count=countWithTh(:,1:6);
mu = mean(count)
sigma = std(count)
[n,p] = size(count);
```

```
% Create a matrix of mean values by
% replicating the mu vector for n rows
MeanMat = repmat(mu,n,1);
% Create a matrix of standard deviation values by
% replicating the sigma vector for n rows
SigmaMat = repmat(sigma,n,1);
% Create a matrix of zeros and ones, where ones indicate
% the location of outliers
outliers = abs(count - MeanMat) > 3*SigmaMat;
% Calculate the number of outliers in each column
nout = sum(outliers)
%NoOfRows=size(count,1);
%nout_per=nout/NoOfRows*100
%countWithTh(any(outliers,2),:) = [];
%count(any(outliers,2),:) = [];

%Smoothing and Filtering
%plot(c3m,'o-')
%hold on

%span = 3; % Size of the averaging window
%window = ones(span,1)/span;
%smoothed_c3m = convn(c3m,window,'same');
%smoothed_c3m = convn(c3,window,'same');
%h = plot(smoothed_c3m,'ro-');
%legend('Data','Smoothed Data')



%The filter function is also used for smoothing data:
%smoothed2_c3m = filter(window,1,c3m);
```

```
%smoothed2_c3m = filter(window,1,c3);
%delete(h)
%plot(smoothed2_c3m,'ro-');

a = 1;
b = [1/4 1/4 1/4 1/4];
x = count(:,4);
y = filter(b,a,x);
figure
t = 1:length(x);
plot(t,x,'--',t,y,'-'),grid on
legend('Original Data','Smoothed Data',2)
title('Plot of Original and Smoothed Data')

a = [1 0.2];
b = [2 3];
y = filter(b,a,x);
t = 1:length(x);
plot(t,x,'-.',t,y,'-'), grid on
legend('Original Data','Shaped Data',2)
title('Plot of Original and Shaped Data')

%Smoothing estimates the center of the distribution of response values at
%each value of the predictor. you can use smoothed data to identify a model, but
%avoid using smoothed data to fit a model.

%Median Filter:
  fs = 100;                        % Sampling rate
  t = 0:1/fs:1;                    % Time vector
  x = sin(2*pi*t*3)+.25*sin(2*pi*t*40);   % Noise Signal - Input
```

```
  y = medfilt1(x,10);                  % Median filtering - Output
 %yMedFilt = medfilt1(x,5,'truncate');
  plot(t,x,'k',t,y,'r');
 %plot(t,x,t,yMedFilt)
  grid;           % Plot
  legend('Original Signal','Filtered Signal')

%Outlier Removal via Hampel Filter
  x = sin(2*pi*(0:99)/100);
  x(6) = 2;
  x(20) = -2;
  hampel(x)
 %y=hampel(y,13)
 %legend('location','best')

%Remove Linear Trends from Data
t = 0:225180;
sdata=count(:,4);
mean(sdata)
figure
plot(t,sdata);
legend('Original Data','Location','northwest');
xlabel('Time');
ylabel('Acc_X');

detrend_sdata=detrend(sdata);
trend = sdata - detrend_sdata;
mean(detrend_sdata)
hold on
plot(t,trend,':r')
```

```
plot(t,detrend_sdata,'m')
plot(t,zeros(size(t)),':k')
legend('Original Data','Trend','Detrended Data',...
'Mean of Detrended Data','Location','northwest')
xlabel('Time');
ylabel('Acc_X');
```

%%%%%%%%%% %%% **Summarizing Data**%%%%%%%%%%%%%%%%%%

It measures the location, scale, and shape of a distribution of dataset.

%%

```
%Measures of Location
count=MyDatasetRaw_Train(:,1:6);
x1 = mean(count)
x2 = median(count)
x3 = mode(count)
%Measures of Scale
dx1 = max(count)-min(count)
dx2 = std(count)
dx3 = var(count)
%Shape of a Distribution
figure
hist(count)
legend('Gyro_X','Gyro_Y','Gyro_Z',...
'Acc_X','Acc_Y','Acc_Z')
%legend('Acc_X','Acc_Y','Acc_Z',...
% 'Gyro_X','Gyro_Y','Gyro_Z')
```

```
%Parametric models give analytic summaries of distribution shapes
c1 = count(:,4); % Data at intersection 1
[bin_counts,bin_locations] = hist(c1);
bin_width = bin_locations(2) - bin_locations(1);
hist_area = (bin_width)*(sum(bin_counts));
figure
hist(c1)
hold on
mu1 = mean(c1);
exp_pdf = @(t)(1/mu1)*exp(-t/mu1); % Integrates
% to 1
t = 0:150;
y = exp_pdf(t);
plot(t,(hist_area)*y,'r','LineWidth',2)
legend('Distribution','Exponential Fit')
```

%%%%%%%%% %%% **Visualizing Data** %%%%%%%%%%%%%%%%%%%

Exploring dataset using graphs functions such as scatter, scatter3, plot, plot3, plotmatrix, boxplot, and hist. Also using graphs tools such as data cursor tool.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Visualizing Data
%2-D Scatter Plots
%ds=dataset('File','DrawAcc_Exp01_User01.csv','Delimiter',',');
%ds.x1_075029373;ds.x1;ds.x9_691419014
%matrix=double(Dataset(:,[]));
c1 = count(:,3); % Data at intersection 1
c2 = count(:,4); % Data at intersection 2
%c2 = MyDatasetRaw_Train(:,9); % Data at intersection 2
figure
scatter(c1,c2,'filled')
```

```
xlabel('Acc_X')
ylabel('Classes')


%M=R;
L={'X','Y','Z'};
n = length(L);
imagesc(M); % plot the matrix
set(gca, 'XTick', 1:n); % center x-axis ticks on bins
set(gca, 'YTick', 1:n); % center y-axis ticks on bins
set(gca, 'XTickLabel', L); % set x-axis labels
set(gca, 'YTickLabel', L); % set y-axis labels
title('Your Title Here', 'FontSize', 14); % set title
colormap('jet'); % set the colorscheme
colorbar ; % enable colorbar

%3-D Scatter Plots
figure
c3 = count(:,5); % Data at intersection 3
scatter3(c1,c2,c3,'filled')
xlabel('Acc_X')
ylabel('Acc_Y')
zlabel('Acc_Z')

%count=countWithTh(:,1);
%ThridLables=countWithTh(:,7);
ThridLables=MyDatasetRaw_TrainWithoutG(:,9);
[~,~,id] = unique(ThridLables);
uniqueGroups=unique(ThridLables);
colors =hsv(12);
```

```
for idx = 1 : length(uniqueGroups)
    data = count (id == idx,:);
    plot3(data(:,1), data(:,2), data(:,3),'.','color',colors(idx,:),'markersize',20);
    hold on;
end
hold off
grid;
legend('Prepairing', 'Tea', 'Sandwish', 'Cleaning')

%
for idx = 1 : length(uniqueGroups)
    data = ds (id == idx,:);
    plot(data(:,2) ,data(:,9) ,'.','color',colors(idx,:),'markersize',20);
    %xlim([7169374 7392987])
    hold on;
legend('Prepairing', 'Tea', 'Sandwish', 'Cleaning')
end
hold off
grid on

%
(https://www.mathworks.com/help/deeplearning/examples/sequence-to-sequence-classification-using-deep-learning.html)
load HumanActivityTrain
XTrain
X = XTrain{1}(1,:);
classes = categories(YTrain{1});

figure
for j = 1:numel(classes)
    label = classes(j);
```

```
    idx = find(YTrain{1} == label);
    hold on
    plot(idx,X(idx))
end
hold off

xlabel("Time Step")
ylabel("Acceleration")
title("Training Sequence 1, Feature 1")
legend(classes,'Location','northwest')

%Measure the strength of the linear relationship among the variables in the
%three-dimensional scatter
vars = eig(cov([c1 c2 c3]))
explained = max(vars)/sum(vars)

%Scatter Plot Arrays
%count=MyDatasetRaw_Train(:,1:6);
figure
plotmatrix(count)

%Exploring Data in Graphs
%Data Cursor
scatter(count(:,1),count(:,4))
%Select the Data Cursor Tool. Click the rightmost data point
%Data brushing also shows one standard deviation of the mean
% with other information from Tool->Data Statistics
%Also Tools > Brushing > Create new variable. Then select points
%Linked plots, or data linking
%Data Linking tool on a figure's toolbar.
```

```
figure
scatter(count(:,3),count(:,4))
xlabel ('count(:,1)')
ylabel ('count(:,2)')
figure
scatter(count(:,5),count(:,4))
xlabel ('count(:,3)')
ylabel ('count(:,2)')

openvar count

Setup
DatasetMine=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer;
DatasetOpport=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer;

Activity = nominal(DatasetMine(:,303));

predictor's value summary statistics
%Measures of Location
count=MyDatasetRaw_Train(:,1:6);
x1 = mean(count)
x2 = median(count)
x3 = mode(count)
%Measures of Scale
dx1 = max(count)-min(count)
dx2 = std(count)
dx3 = var(count)

p = 0:.25:1;
breaks = quantile(DatasetMine(:,1),p)
```

```
breaks = quantile(DatasetOpport(:,1),p)

load('MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer.mat')
Xtrain=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(1:6233,1:300);
Ytrain=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(1:6233,303);
Xtest=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(6234:end,1:300);
Ytest=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(6234:end,303);
count=[Xtrain Ytrain];
mu = mean(count);
sigma = std(count);
[n,p] = size(count);
% Create a matrix of mean values by
% replicating the mu vector for n rows
MeanMat = repmat(mu,n,1);
% Create a matrix of standard deviation values by
% replicating the sigma vector for n rows
SigmaMat = repmat(sigma,n,1);
% Create a matrix of zeros and ones, where ones indicate
% the location of outliers
outliers = abs(count - MeanMat) > 2*SigmaMat;
XtrainWithY= count(~any(outliers,2),:);
Xtrain=XtrainWithY(:,1:300);
Ytrain=XtrainWithY(:,301);

Predictor's categories
Activity = nominal(DatasetMine(:,303));
%any(Activity=='Canada')
any(ismember(Activity,'Sweden'))
Activity = Activity(Activity~='Sweden');
Activity = droplevels(Activity,'Sweden');
```

```
ix = find(Activity=='France')
Activity(ix)
find(isundefined(Activity))

Activity = nominal(DatasetMine(:,303));
getlevels(Activity)
tabulate(Activity)

[Min,Mean,Max] = grpstats(DatasetMine(:,[1,51,101]),Activity,{'mean','min','max'})

figure
boxplot(DatasetMine(:,1),Activity)
title('X axes, Grouped by Activities')
```

Scatterplot Matrices

Viewing slices through lower dimensional subspaces is one way to partially work around

the limitation of two or three dimensions.

For example, we can use the gplotmatrix function to display

an array of all the bivariate scatterplots between

our five variables, along with a univariate histogram for each variable.

```
x = [DatasetMine(:,1),DatasetMine(:,51),DatasetMine(:,101)];
gplotmatrix(x,[],DatasetMine(:,303),[],'+xo.')

% Acc raw data
count=MyDatasetRaw_TrainWithoutG_SelSub(:,[1:6,11]);
[n,p] = size(count)
t = 1:n;
plot(t,count(:,1:3));
```

```
legend('Acc_X','Acc_Y','Acc_Z')
xlabel('Time'), ylabel('Coordinate Value')
title('Accelerometer Axis')

%% 3-D Scatter Plots for Acc raw data
ThridLables=count(:,7);
[~,~,id] = unique(ThridLables);
uniqueGroups=unique(ThridLables);
colors =hsv(12);
for idx = 1 : length(uniqueGroups)
   data = count (id == idx,:);
   plot3(data(:,1), data(:,2), data(:,3),'.','color',colors(idx,:),'markersize',20);
   hold on;
end
hold off
grid;
legend('Prepairing', 'Tea', 'Sandwish', 'Cleaning')



figure()
gscatter(DatasetMine(:,1),DatasetMine(:,51),DatasetMine(:,303),'bgrk','x.o+')
title('X vs. Y, Grouped by Activity')

x=DatasetMine(:,1);
y=DatasetMine(:,51);
z=DatasetMine(:,101);
g=DatasetMine(:,303);
h = gscatter(x, y, g);
% for each unique group in 'g', set the ZData property appropriately
gu = unique(g);
```

```
for k = 1:numel(gu)
    set(h(k), 'ZData', z( g == gu(k) ));
end
view(3)

figure;
scatterhist(x,y)
xlabel('x')
ylabel('y')

scatterhist(x,y,'Group',g,'Kernel','on')

h = scatterhist(x,y,'Group',g);
hold on;
clr = get(h(1),'colororder');
boxplot(h(2),x,g,'orientation','horizontal',...
'label',{","","","},'color',clr);
boxplot(h(3),y,g,'orientation','horizontal',...
'label', {","","","},'color',clr);
set(h(2:3),'XTickLabel',");
view(h(3),[270,90]); % Rotate the Y plot
axis(h(1),'auto'); % Sync axes
hold off;


% Dataset Analysis
ds=dataset('File','DrawAcc_Exp01_User01.csv','Delimiter',',');
[~,~,id] = unique(ds.x1);
uniqueGroups=unique(ds.x1);%Labels
colors =hsv(12);
```

```
%colors = brewermap(length(uniqueGroups),'Set1');
%for idx = 1 : length(id)
for idx = 1 : length(uniqueGroups)
   data = ds(id == idx,:);
   %Timestamp,gravity
   plot(data(:,1) ,data(:,3),'.','color',colors(idx,:),'markersize',20);
   %Min and Max value in timestamp
   xlim([7169374 7392987])
   hold on;
   legend('Prepairing', 'Tea', 'Sandwish', 'Cleaning')
end
hold off
grid on
%legend('Prepairing', 'Tea', 'Sandwish', 'Cleaning')
%legend('group 1','group 2','group 3','group 4')
%
plot(ds.x9_423865318);legend('Magnitude')
% This
[~,~,id] = unique(ds.x1_2);
uniqueGroups=unique(ds.x1_2);
colors =hsv(12);
for idx = 1 : length(uniqueGroups)
   data = ds (id == idx,:);
   plot3(data(:,3), data(:,4), data(:,5),'.','color',colors(idx,:),'markersize',20);
   hold on;
end
hold off
grid;
legend('Prepairing', 'Tea', 'Sandwish', 'Cleaning')
%
```

```
for idx = 1 : length(uniqueGroups)
    data = ds (id == idx,:);
    plot(data(:,2) ,data(:,9) ,'.','color',colors(idx,:),'markersize',20);
    xlim([7169374 7392987])
    hold on;
legend('Prepairing', 'Tea', 'Sandwish', 'Cleaning')
end
hold off
grid on
```

%%%%%%%%% %%% **Modelling Data** %%%%%%%%%%%%%%%%%%%

It checks each predictor's distribution, predictor's correlation, pre-test assumption (distribution) for predictors and response model, and finally it tests model quality and modify (corrleation, Outlier, Parameters, Baysian Optimization).

%%

```
figure();
histogram(DatasetMine(Activity=='1'))
title('Histogram of First Activity')
```

Each predictor distribution

Predictor's correlation

```
ActOne= DatasetMine(Activity=='1');
```

```
ActTwo = DatasetMine(Activity=='2');
[h,p] = ttest2(ActOne,ActTwo)
%p =
anovan(DatasetMine(:,303),{DatasetMine(:,1),DatasetMine(:,51),DatasetMine(:,101)
})
aoctool(DatasetMine(:,1),DatasetMine(:,51),DatasetMine(:,101),DatasetMine(:,303))
;
```

Pertest assumption (distribution)

Predictors and response model

Model quality and modify (corrleation, Outlier, Parameters, Baysian Optimization)

Generate test dataset

Predict new data

Share fitted model

```
%Polynomial Regression
count=MyDatasetRaw_Train(:,1:6);
c3 = count(:,4); % Data at intersection 3
%load count.dat
%c3 = count(:,3); % Data at intersection 3
tdata = (1:225181)';
p_coeffs = polyfit(tdata,c3,6);
figure
plot(c3,'o-')
hold on
%tfit = (1:0.01:225181)';
tfit = (1:1:225181)';
```

```
yfit = polyval(p_coeffs,tfit);
plot(tfit,yfit,'r-','LineWidth',2)
legend('Data','Polynomial Fit','Location','NW')

%General Linear Regression
c3 = count(:,4); % Data at intersection 3
tdata = (1:225181)';
X = [ones(size(tdata)) cos((2*pi/12)*(tdata-7))];
s_coeffs = X\c3;
figure
plot(c3,'o-')
hold on
tfit = (1:1:225181)';
yfit = [ones(size(tfit)) cos((2*pi/12)*(tfit-7))]*s_coeffs;
plot(tfit,yfit,'r-','LineWidth',2)
legend('Data','Sinusoidal Fit','Location','NW')

[s_coeffs,stdx,mse] = lscov(X,c3)

Fs = 1; % Sample frequency (per hour)
n = length(c3); % Window length
Y = fft(c3); % DFT of data
f = (0:n-1)*(Fs/n); % Frequency range
P = Y.*conj(Y)/n; % Power of the DFT
figure
plot(f,P)
xlabel('Frequency')
ylabel('Power')
predicted_f = 1/12
```

%%%%%%%%% **Classification and results evaluation** %%%%%%%%%%

It evaluates the performance and efficiency of our proposed method comparing to three states of arts those are: SACAAR (Saguna et al., 2013), SC2 (Liu et al., 2016), and PEMAR (Vaka et al., 2015).

%%

%My system

profile -memory on

% load(**'Evaluation_MyDatasetTrain_SelAct.mat'**)

CommonAtributes=OnlyData;

rhohat = corr(CommonAtributes(:,1:50),CommonAtributes(:,51:100));

range=max(CommonAtributes(:,101:150))-min(CommonAtributes(:,101:150));

rx=CommonAtributes(:,1:50);

ry=CommonAtributes(:,51:100);

rz=CommonAtributes(:,101:150);

gx=CommonAtributes(:,151:200);

gy=CommonAtributes(:,201:250);

gz=CommonAtributes(:,251:300);

r= sqrt(rx.^2+ry.^2+rz.^2);

rollx= 180*atan2(mean(ry),sqrt(rx.^2+rz.^2))/pi;

pitchy= 180*atan2(mean(rx),sqrt(ry.^2+rz.^2))/pi;

yawz= 180*atan2(mean(rz),sqrt(rx.^2+rz.^2))/pi;

a=180*acos(mean(ry)/r)/pi;

b=180*acos(mean(rx)/r)/pi;

c=180*acos(mean(rz)/r)/pi;

```
d=180*acos(((ry+gy)/(2*r))/100)/pi;
e=180*acos(((rx+gx)/(2*r))/100)/pi;
f=180*acos(((rz+gz)/(2*r))/100)/pi;

Xtrain=OnlyFeaturesWithLabels(:,[5:9,14:16]);
Ytrain=OnlyFeaturesWithLabels(:,3);
% Ytrain=OnlyFeaturesWithLabels(:,2);

%Decision Tree
Mdl = fitctree(Xtrain,Ytrain);
CVMdl = crossval(Mdl,'KFold',5);
kloss = kfoldLoss(CVMdl, 'LossFun', 'ClassifError')
ValidationAccuracy = (1 - kloss)*100

[YPredic,escore] = kfoldPredict(CVMdl);
C_knn = confusionmat(Ytrain,YPredic);
C_knn = bsxfun(@rdivide,C_knn,sum(C_knn,2)) * 100
[c_matrix,Result,RefereceResult]= confusion.getMatrix(Ytrain,YPredic);

profile report
profile off

%Theory (SACAAR):
profile -memory on
% Xtrain=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(1:6233,1:300);
% Ytrain=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(1:6233,303);
% Xtest=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(6234:end,1:300);
% Ytest=MyDatasetRaw_TrainWithoutG_AllSub1WinBuffer(6234:end,303);
Xtrain=OnlyData;
```

```
Ytrain=OnlyFeaturesWithLabels(:,3);
% Ytrain=OnlyFeaturesWithLabels(:,2);

%Decision Tree
Mdl = fitctree(Xtrain,Ytrain);
CVMdl = crossval(Mdl,'KFold',5);
kloss = kfoldLoss(CVMdl, 'LossFun', 'ClassifError')
ValidationAccuracy = (1 - kloss)*100

[YPredic,escore] = kfoldPredict(CVMdl);
C_knn = confusionmat(Ytrain,YPredic);
C_knn = bsxfun(@rdivide,C_knn,sum(C_knn,2)) * 100
[c_matrix,Result,RefereceResult]= confusion.getMatrix(Ytrain,YPredic);

profile report
profile off
CA=['Cooking omelette for breakfast in kitchen'];
CAW=[0.59];
A={'A3', 'A2', 'A5', 'A18', 'A21', 'A6', 'A17', 'A11', 'A23', 'A10', 'A9'};
AW=[0.10 0.10 0.05 0.10 0.10 0.07 0.10 0.10 0.08 0.10 0.10];
C={'C2', 'C7', 'C14', '-C14', '-C7', '-C2'};
CW=[0.19 0.12 0.19 0.19 0.12 0.19];
SA={'C2', 'A18', 'A5' ,'A21', 'A23'};
EA={'-C7', 'A9', '-C14'};
TS=[07:06];
TE=[07:22];
TL=[16];
TLR=[10-20];
S={'in kitchen'};
```

```
ComplexActivity{1}{1}= CA;
ComplexActivity{1}{2}= CAW;
ComplexActivity{1}{3}= A;
ComplexActivity{1}{4}= AW;
ComplexActivity{1}{5}= C;
ComplexActivity{1}{6}= CW;
ComplexActivity{1}{7}= SA;
ComplexActivity{1}{8}= EA;
ComplexActivity{1}{9}= TS;
ComplexActivity{1}{10}= TE;
ComplexActivity{1}{11}= TL;
ComplexActivity{1}{12}= TLR;
ComplexActivity{1}{13} = S;

CA=['Preparing coffee in office kitchen'];
CAW=[0.65];
A={'A3', 'A2', 'A32', 'A33'};
AW=[0.25 0.25 0.25 0.25];
C={'C2', 'C7', '-C2', '-C7'};
CW=[0.03 0.03 0.13 0.26];
SA={'A32', 'C2'};
EA={'-A33', '-C2'};
TS=[18:16];
TE=[18:22];
TL=[6];
TLR=[5-10];
S={'in office kitchen'};

ComplexActivity{2}{1}= CA;
ComplexActivity{2}{2}= CAW;
```

```
ComplexActivity{2}{3}= A;
ComplexActivity{2}{4}= AW;
ComplexActivity{2}{5}= C;
ComplexActivity{2}{6}= CW;
ComplexActivity{2}{7}= SA;
ComplexActivity{2}{8}= EA;
ComplexActivity{2}{9}= TS;
ComplexActivity{2}{10}= TE;
ComplexActivity{2}{11}= TL;
ComplexActivity{2}{12}= TLR;
ComplexActivity{2}{13} = S;

CA=['Working on a doc in office room'];
CAW=[0.90];
A={'A1', 'A47', 'A36', 'A48'};
AW=[0.02 0.02 0.02 0.02];
C={'C13'};
CW=[1.00];
SA={'A48'};
EA={'-C48'};
TS=[10:10];
TE=[11:30];
TL=[80];
TLR=[10-60];
S={'in office room'};

ComplexActivity{3}{1}= CA;
ComplexActivity{3}{2}= CAW;
ComplexActivity{3}{3}= A;
ComplexActivity{3}{4}= AW;
```

```
ComplexActivity{3}{5}= C;
ComplexActivity{3}{6}= CW;
ComplexActivity{3}{7}= SA;
ComplexActivity{3}{8}= EA;
ComplexActivity{3}{9}= TS;
ComplexActivity{3}{10}= TE;
ComplexActivity{3}{11}= TL;
ComplexActivity{3}{12}= TLR;
ComplexActivity{3}{13} = S;
///
This:
x = inputdlg('Enter Atoimc Activities list');
obs{1} = strsplit(x{:});
x = inputdlg('Enter Context List');
obs{2} = strsplit(x{:});
x = inputdlg('Enter Situation');
obs{3} = cellstr(x{:});
celldisp(obs)
obs{3}{1}
//
A=A3 A2 A18 A6 A17 A11 A10 A9
C=C2 C7 C14 -C14 -C7 -C2
S=in kitchen
//
A=A3 A2 A5 A18 A21 A6 A17 A11 A23 A10
C=C2 C7 C14 -C7 -C2
S=in kitchen
///
This:
for i=1:size(ComplexActivity,2)
```

```
%Check S
index = ismember(ComplexActivity{i}{13}, obs{3});
if (index==1)
%Check SA
itemSA = intersect(ComplexActivity{i}{7}, obs{1});
itemEAC = intersect(ComplexActivity{i}{8}, [obs{1} obs{2}]);
if (length(itemSA)~=0 & length(itemEAC)~=0) %Check
%Check A.
%while(time counter< TLmaxk). SATS(Input)+CTL(ComplexActivity{1}{11})
[itemsA InF InS] = intersect(ComplexActivity{i}{3}, obs{1});
if (length(itemsA)~=0)
ACW=sum(ComplexActivity{i}{4}(InF));
end

%Check C.
[itemsC InF InS] = intersect(ComplexActivity{i}{5}, obs{2});
if (length(itemsC)~=0)
ACW=ACW+sum(ComplexActivity{i}{6}(InF));
%disp(ACW)
end
%ComplexActivity{1}{8}= EA; ComplexActivity{1}{2}= CAW.

if (ACW>=ComplexActivity{i}{2}) %Check
disp(itemsA)
disp(itemsC)
disp(ComplexActivity{i}{1})
end

end
end
```

```
end
////
This with occur of only start and last NaN columns:
% Input dialog box
clear all
n=inputdlg('Enter number of variances:');
for i=1:str2double(n)
 x = inputdlg('Enter space-separated numbers:',...
          'Sample', [1 50]);
obs{i} = str2num(x{:});
end
%celldisp(obs)
% Transition Matrix
transitions = cellfun(@(x)([x(1:length(x)-1); x(2:length(x))]), obs, 'UniformOutput', false);
alltransitions = cell2mat(transitions)';
[uniqueTransitions, ~, i]=unique(alltransitions,'rows','stable');
v=arrayfun(@(x) sum(i==x),1:size(uniqueTransitions,1))';
p = v/sum(v);
transitionMatrix = sparse(uniqueTransitions(:,1), uniqueTransitions(:,2), p, max([obs{:}]),max([obs{:}]));

% Convert Cell array to Matrix and full elements
%obs={[1 2 3 4];[4 5 6];[3 4 5]}
maxSize = max(cellfun(@numel,obs));    %# Get the maximum vector size
fcn = @(x) [x nan(1,maxSize-numel(x))];  %# Create an anonymous function
rmat = cellfun(fcn,obs,'UniformOutput',false);  %# Pad each cell with NaNs
obs = vertcat(rmat{:});
%obs = reshape(obs, 4, 3).';
%obs(obs==0) = nan;
%obs=(obs~=0);
```

```
%Probability of each element in matrix
a = unique(obs);
a(isnan(a)) = [];
ocur = [a,histc(obs(:),a)/numel(obs)];

%Individual Probability of element in column
[r, c] = size(obs);
y = zeros(r,c);
p = zeros(r,c);
for i = 1:c                         %Looping through the column.
    for j = 1:r                     %Looping through the row in that that column.
        y(j,i) = sum(obs(:,[i]) == obs(j,i)); %Compare each column of the matrix with the entries in that column.
        p(j,i) = y(j,i)/r;          %Probability of each entry of a column within that column.
    end
end
%disp(obs)
%disp('Individual Probability of element in column: ');disp(p);


%Probability of only start and last NaN columns in matrix
LastCol=obs(sub2ind(size(obs),1:size(obs,1),max(bsxfun(@times,~isnan(obs),1:size(obs,2)),[],2).')).';
FirstCol=obs(:,1);
StartEndCols=[FirstCol LastCol];
UnFirst = unique(FirstCol);
OcurFirst = [UnFirst,histc(FirstCol(:),UnFirst)/numel(FirstCol)];
UnLast = unique(LastCol);
OcurLast = [UnLast,histc(LastCol(:),UnLast)/numel(LastCol)];
%StartEndOcur=[OcurFirst,OcurLast];
```

```
%Path Probability
obs=obs.';
[un, ~, id] = unique(obs.', 'rows'); %'// Assigning each event a unique ID
un = un.'; %'// Transpose to ensure compatibility
N = size(un,2); %// Get total number of unique scenarios
M = size(obs,2); %// Get total number of scenarios
path = histc(id, 1 : N) / M; %// Finding probabilities
%disp(un)

%Display result
disp('Obs: ');disp(obs.')
disp('Prob of each element: ');disp(sortrows(ocur,-2))
disp('Individual Probability of element in column: ');disp(p)
disp('Prob of first and last columns: ');disp(OcurFirst);disp(OcurLast);
disp('TransitionMatrix: ');disp(sortrows(transitionMatrix,-2))
%disp('Path Probability: ');disp(path)
disp('Path Probability: ');disp([un',path])
///

%Transition Matrix (PEMAR):
profile -memory on
Xtrain=OnlyData;
Ytrain=OnlyFeaturesWithLabels(:,3);

%rng(123);
rng('default');
K = 3;
%K = 4;
```

```
% K=length(unique(Ytrain));

opts = statset('MaxIter', 500, 'Display', 'iter');
[clustIDX, clusters, interClustSum, Dist] = kmeans(Xtrain, K, 'options',opts, ...
   'distance','sqEuclidean', 'EmptyAction','singleton', 'replicates',3);
% %numObservarations=2765;
% numObservarations=size(Xtest,1);
% D = zeros(numObservarations, K);     % init distances
% for k=1:K
%     %d = sum((x-y).^2).^0.5
%     D(:,k) = sum( ((Xtest - repmat(clusters(k,:),numObservarations,1)).^2), 2);
% end
%  [minDists, clusterIndices] = min(D, [], 2);

 NewData=[Xtrain clustIDX Ytrain];
%  NewDataTest=[Xtest clusterIndices Ytest];

 %HMM:Real dataset (at bnt-master)

humanActivityData = Xtrain; %rawfeat
humanActivityArray =Xtrain;
Acc_tr_ActIdReshFirst=clustIDX;
[uV,~,seqs] = unique(humanActivityArray);
seqs=reshape(seqs,size(humanActivityData,1),size(humanActivityData,2));
seqs2 = rem(seqs,9)+1;  %redefine as only 10 states
seq_len = length(seqs2);
%Note: the discrete alphabet is assumed to be {1, 2, ..., O}, where O = size(obsmat,
2).
%Hence data cannot contain any 0s.

% Q = 4;
```

```
Q = 3;    %# number of states (sun,rain,fog). 4
O = 9;    %# number of discrete observations (umbrella, no umbrella)
[TRANS_Train, EMIS_EST] = hmmestimate(seqs2, Acc_tr_ActIdReshFirst);

% humanActivityData = Xtest; %rawfeat
% humanActivityArray =Xtest;
% Acc_tr_ActIdReshFirst=clusterIndices;
% [uV,~,seqs] = unique(humanActivityArray);
% seqs=reshape(seqs,size(humanActivityData,1),size(humanActivityData,2));
% seqs2 = rem(seqs,9)+1;  %redefine as only 10 states
% seq_len = length(seqs2);
% %Note: the discrete alphabet is assumed to be {1, 2, ..., O}, where O =
size(obsmat, 2).
% %Hence data cannot contain any 0s.
%
% Q = 3;    %# number of states (sun,rain,fog). 4
% O = 9;    %# number of discrete observations (umbrella, no umbrella)
% [TRANS_Test, EMIS_EST] = hmmestimate(seqs2, Acc_tr_ActIdReshFirst);
%Xtrain=TRANS_Train;Ytrain=[1;2;3;4];
Xtrain=TRANS_Train;Ytrain=[1;2;3];
% Xtest=TRANS_Test;Ytest=[1;2;3];
%Decision Tree
Mdl = fitctree(Xtrain,Ytrain);
CVMdl = crossval(Mdl,'KFold',5);
kloss = kfoldLoss(CVMdl, 'LossFun', 'ClassifError')
ValidationAccuracy = (1 - kloss)*100

[YPredic,escore] = kfoldPredict(CVMdl);
C_knn = confusionmat(Ytrain,YPredic);
C_knn = bsxfun(@rdivide,C_knn,sum(C_knn,2)) * 100
[c_matrix,Result,RefereceResult]= confusion.getMatrix(Ytrain,YPredic);
```

```
profile report
profile off

%Time series Shapelet (SC2):
profile -memory on

fi=1;
names = {'Adiac','Beef','ChlorineConcentration','Coffee','DiatomSizeReduction',...
   'DP_Little','DP_Middle','DP_Thumb','ECGFiveDays','FaceFour', ...
   'Gun_Point','ItalyPowerDemand','Lighting7','MedicalImages','MoteStrain',...
   'MP_Little','MP_Middle','Herrings', 'PP_Little','PP_Middle',...

'PP_Thumb','SonyAIBORobotSurface','Symbols','synthetic_control','Trace','TwoLea
dECG' };

% fp = fopen('results.txt','w');
% fprintf(fp,'%25s %25s %25s','data set', 'test accuracy', 'discovery time(s)');
% fprintf(fp,'\r\n');
% fclose(fp);
% \alpha_1 and \alpha_2 from cross-validation
alpha1 = [ 0.10, 0.10, 0.08, 0.10, 0.02,...
   0.04, 0.10, 0.04, 0.04, 0.02,...
   0.02, 0.08, 0.10, 0.04, 0.04,...
   0.04, 0.06, 0.08, 0.02, 0.10,...
   0.02, 0.04, 0.08, 0.08, 0.02, 0.10];
alpha2 = [ 0.03, 0.03, 0.01, 0.01, 0.02,...
   0.04, 0.03, 0.01, 0.03, 0.04,...
   0.03, 0.05, 0.05, 0.01, 0.04,...
   0.01, 0.02, 0.01, 0.01, 0.01,...
   0.01, 0.04, 0.04, 0.02, 0.02, 0.02];
```

```
acc = zeros(length(names),1);
distime = zeros(length(names),1);
 % initialization
   name = names{fi};
%     TRAIN = importdata(['data/',name,'_TRAIN']);
%     TEST = importdata(['data/',name,'_TEST']);
   TRAIN=OnlyData;
   TRAIN_class_labels=OnlyFeaturesWithLabels(:,3);


   TEST=OnlyData;
   TEST_class_labels=OnlyFeaturesWithLabels(:,3);
%%Out of memory
%
LowAct=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(:,182)~=0;
%     Train=OpportunityDatasetRaw_TrainWithoutG_AllSub1WinBuffer(LowAct,:);
%
%
LowAct=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(:,182)~=0;
%     Test=OpportunityDatasetRaw_TestWithoutG_AllSub1WinBuffer(LowAct,:);
%
%     TRAIN=Train(:,1:180);
%     TRAIN_class_labels=Train(:,182);
%     TEST=Test(:,1:180);
%     TEST_class_labels=Test(:,182);
%
%     TRAIN_class_labels(TRAIN_class_labels==401) = 1;
%     TRAIN_class_labels(TRAIN_class_labels==402) = 2;
%     TRAIN_class_labels(TRAIN_class_labels==403) = 3;
%     TRAIN_class_labels(TRAIN_class_labels==404) = 4;
%     TRAIN_class_labels(TRAIN_class_labels==405) = 5;
```

```
%     TRAIN_class_labels(TRAIN_class_labels==406) = 6;
%     TRAIN_class_labels(TRAIN_class_labels==407) = 7;
%     TRAIN_class_labels(TRAIN_class_labels==408) = 8;
%     TRAIN_class_labels(TRAIN_class_labels==409) = 9;
%     TRAIN_class_labels(TRAIN_class_labels==410) = 10;
%     TRAIN_class_labels(TRAIN_class_labels==411) = 11;
%     TRAIN_class_labels(TRAIN_class_labels==412) = 12;
%     TRAIN_class_labels(TRAIN_class_labels==413) = 13;
%
%     TEST_class_labels(TEST_class_labels==401) = 1;
%     TEST_class_labels(TEST_class_labels==402) = 2;
%     TEST_class_labels(TEST_class_labels==403) = 3;
%     TEST_class_labels(TEST_class_labels==404) = 4;
%     TEST_class_labels(TEST_class_labels==405) = 5;
%     TEST_class_labels(TEST_class_labels==406) = 6;
%     TEST_class_labels(TEST_class_labels==407) = 7;
%     TEST_class_labels(TEST_class_labels==408) = 8;
%     TEST_class_labels(TEST_class_labels==409) = 9;
%     TEST_class_labels(TEST_class_labels==410) = 10;
%     TEST_class_labels(TEST_class_labels==411) = 11;
%     TEST_class_labels(TEST_class_labels==412) = 12;
%     TEST_class_labels(TEST_class_labels==413) = 13;

    initialization;

    % shapelets learning
    shapelets =[];
    index = [];
    randn('seed',1);
    t = tic;
```

```
    if fi==11 || fi==15
        v = admm(C, 1,2,alpha1(fi), alpha2(fi));
        block = extracts (v);
        [shapelets,index] =
AutoShapeletGeneration(block,1,TRAIN,TRAIN_class_labels);

    else
        for classiter = 1:K
            v = admmmul(C, classiter,alpha1(fi), alpha2(fi));
            block = extracts (v);
            [shapeletstmp,indextmp] =
AutoShapeletGeneration(block,classiter,TRAIN,TRAIN_class_labels);
            shapelets = [shapelets;shapeletstmp];
            index = [index;indextmp];
        end
    end

    distime(fi) = toc(t);
    %use z-normalized euclidean distance to transform the data
    D_tr = transnew(TRAIN',shapelets,index);
    D_ts =transnew(TEST',shapelets,index);

Xtrain=D_tr; Ytrain=TRAIN_class_labels;
% Xtest=D_ts; Ytest=TEST_class_labels;
Mdl = fitctree(Xtrain,Ytrain);
CVMdl = crossval(Mdl,'KFold',5);
kloss = kfoldLoss(CVMdl, 'LossFun', 'ClassifError')
ValidationAccuracy = (1 - kloss)*100

[YPredic,escore] = kfoldPredict(CVMdl);
C_knn = confusionmat(Ytrain,YPredic);
```

```
C_knn = bsxfun(@rdivide,C_knn,sum(C_knn,2)) * 100
[c_matrix,Result,RefereceResult]= confusion.getMatrix(Ytrain,YPredic);
profile report
profile off
```