classdef EmgAnalyser\_App < matlab.apps.AppBase

% Properties that correspond to app components

properties (Access = public)

figure1 matlab.ui.Figure

axes1 matlab.ui.control.UIAxes

text1 matlab.ui.control.Label

pushbutton3 matlab.ui.control.Button

pushbutton4 matlab.ui.control.Button

text4 matlab.ui.control.Label

pushbutton1 matlab.ui.control.Button

pushbutton2 matlab.ui.control.Button

text5 matlab.ui.control.Label

text2 matlab.ui.control.Label

end

% Callbacks that handle component events

methods (Access = private)

% Code that executes after component creation

function EmgAnalyser\_OpeningFcn(app, varargin)

% Create GUIDE-style callback args - Added by Migration Tool

[hObject, eventdata, handles] = convertToGUIDECallbackArguments(app); %#ok<ASGLU>

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to EmgAnalyser (see VARARGIN)

% Choose default command line output for EmgAnalyser

handles.output = hObject;

% Setting background of GUI

ah = axes('unit','normalized','position',[0 0 1 1]);

bg = imread('bg11.png'); imagesc(bg);

% set = (ah,'handlevisibility','off','visible','off');

%uistack(ah,'bottom')

% Setting bg of button

button\_bg = imread('button3.png');

set(handles.pushbutton3, 'CData', button\_bg);

button\_bg = imread('button3.png');

set(handles.pushbutton4, 'CData', button\_bg);

button\_bg = imread('button3.png');

set(handles.pushbutton1, 'CData', button\_bg);

button\_bg = imread('button3.png');

set(handles.pushbutton2, 'CData', button\_bg);

% Update handles structure

guidata(hObject, handles);

end

% Button pushed function: pushbutton1

function pushbutton1\_Callback(app, event)

% Create GUIDE-style callback args - Added by Migration Tool

[hObject, eventdata, handles] = convertToGUIDECallbackArguments(app, event); %#ok<ASGLU>

% hObject handle to pushbutton1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[filename,pathname] = uigetfile({'\*.mat'},'Pick a raw EMG signal');

data = fullfile(pathname,filename);

xy = load(data);

signal = xy.x;

fs = 512; % Sampling rate

dt = 1/fs; % Time-period

npnts = fs \* 2; % Number of points

time = [0:length(signal)-1].\*dt; % Time-vector for plotting

n = length(time);

N = length(signal);

signal = abs(signal); % Rectification

threshold = mean(signal)+4\*std(signal);

outliers = signal > threshold;

F = griddedInterpolant(time(~outliers),signal(~outliers));

signalR = signal;

signalR(outliers) = F(time(outliers));

mov\_avg = movmean(signalR,35,1);

kurt\_val = kurtosis(mov\_avg);

mean\_freq = meanfreq(mov\_avg);

med\_freq = medfreq(mov\_avg);

[c,l] = wavedec(mov\_avg,10,'db2');

approximation = appcoef(c,l,'db2');

[d5,d10] = detcoef(c,l,[5 10]);

d\_med = median(abs(d10));

se = approximateEntropy(mov\_avg);

xfilt = abs(hilbert(filterFGx(mov\_avg',fs,5,5)));

x4dfa = cumsum(xfilt-mean(xfilt));

nScales = 20;

ranges = round(N\*[0.01 0.2]); % range from 1% of signal to 20%

scales = ceil(logspace(log10(ranges(1)),log10(ranges(2)),nScales));

rmses = zeros(2,nScales); %creating a zero matrix to store the future rms values

for scalei = 1:nScales

% No. of epochs for this scale

n = floor(N/scales(scalei));

% RMS for the s/g

epochs = reshape(x4dfa(1:n\*scales(scalei)),scales(scalei),n);

depochs = detrend(epochs);

rmses(2,scalei) = mean(sqrt(mean(depochs.^2,1)));

end

A = [ ones(nScales,1) log10(scales)' ]; % linear model

dfa1 = (A'\*A) \ (A'\*log10(rmses(2,:))'); % fit to s/g

dfa = dfa1(2);

% TABLE

varNames = ["KURTOSIS","MEAN\_FREQ","MED\_FREQ","D\_MED","ENTROPY","DFA"];

sz = [1 6];

varTypes = ["double","double","double","double","double","double"];

result\_dataset = table('Size',sz,'VariableTypes',varTypes,'VariableNames',varNames);

result\_dataset(1,:) = {kurt\_val,mean\_freq,med\_freq,d\_med,se,dfa};

% PREDICTION

load RUSBoost.mat

yfit = bagTrees.predictFcn(result\_dataset);

if yfit==0

a = "The exercise is done incorrectly";

else

a = "The exercise is done correctly";

end

set(handles.text1,'String',a);

end

% Button pushed function: pushbutton2

function pushbutton2\_Callback(app, event)

% Create GUIDE-style callback args - Added by Migration Tool

[hObject, eventdata, handles] = convertToGUIDECallbackArguments(app, event); %#ok<ASGLU>

% hObject handle to pushbutton2 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton5.

[filename,pathname] = uigetfile({'\*.mat'},'Pick a raw EMG signal');

data = fullfile(pathname,filename);

signal = load(data);

x = signal.x;

N = length(x);

start\_sig = x(1:round(N/2));

end\_sig = x(round(N/2)+1:round(N));

meanfreq\_start = meanfreq(start\_sig);

meanfreq\_end = meanfreq(end\_sig);

diff\_freq = meanfreq\_end-meanfreq\_start;

if diff\_freq<0

b = "The muscle is becoming fatigued towards the end";

else

b = "The muscle is not becoming fatigued towards the end";

end

set(handles.text2,'String',b);

end

% Button pushed function: pushbutton3

function pushbutton3\_Callback(app, event)

% Create GUIDE-style callback args - Added by Migration Tool

[hObject, eventdata, handles] = convertToGUIDECallbackArguments(app, event); %#ok<ASGLU>

% hObject handle to pushbutton3 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[filename,pathname] = uigetfile({'\*.mat'},'Pick a raw EMG signal');

data = fullfile(pathname,filename);

signal = load(data);

x = signal.x;

N = length(x);

start\_sig = x(1:round(N/2));

end\_sig = x(round(N/2)+1:round(N));

meanfreq\_start = meanfreq(start\_sig);

meanfreq\_end = meanfreq(end\_sig);

clc;

plot(handles.axes1,[meanfreq\_start meanfreq\_end],'o-','MarkerFaceColor','red','MarkerEdgeColor','yellow','MarkerSize', 10)

ylabel(handles.axes1,'Mean Frequency (Hz)','FontSize',20,'FontWeight','bold','Color','k')

xlabel(handles.axes1,'Signal Lifetime','FontSize',20,'FontWeight','bold','Color','k')

title(handles.axes1,'MUSCLE FATIGUE ANALYSIS','FontSize',20,'FontWeight','bold','Color','k')

end

% Button pushed function: pushbutton4

function pushbutton4\_Callback(app, event)

% Create GUIDE-style callback args - Added by Migration Tool

[hObject, eventdata, handles] = convertToGUIDECallbackArguments(app, event); %#ok<ASGLU>

% hObject handle to pushbutton4 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

[filename,pathname] = uigetfile({'\*.mat'},'Pick a raw EMG signal');

data = fullfile(pathname,filename);

xy = load(data);

signal = xy.x;

fs = 512; % Sampling rate

dt = 1/fs; % Time-period

npnts = fs \* 2; % Number of points

time = [0:length(signal)-1].\*dt; % Time-vector for plotting

signal = abs(signal); % Rectification

threshold = mean(signal)+4\*std(signal);

outliers = signal > threshold;

F = griddedInterpolant(time(~outliers),signal(~outliers));

signalR = signal;

signalR(outliers) = F(time(outliers));

% Rectification and moving average

mov\_avg = movmean(signalR,35,1);

% Plotting preprocessed data

clc;

plot(handles.axes1,time,signal,'b');

hold(handles.axes1,'on');

plot(handles.axes1,time,mov\_avg,'r','linew',4);

hold(handles.axes1,'off')

xlabel(handles.axes1,'Time(s)','FontSize',20,'FontWeight','bold','Color','k')

ylabel(handles.axes1,'Amplitude (mV)','FontSize',20,'FontWeight','bold','Color','k')

title(handles.axes1,'SIGNAL AFTER PROCESSING','FontSize',20,'FontWeight','bold','Color','k')

legend(handles.axes1,'Raw signal','Preprocessed signal')

end

end

% Component initialization

methods (Access = private)

% Create UIFigure and components

function createComponents(app)

% Create figure1 and hide until all components are created

app.figure1 = uifigure('Visible', 'off');

app.figure1.Position = [1087 1043 896 646];

app.figure1.Name = 'EmgAnalyser';

app.figure1.Resize = 'off';

app.figure1.HandleVisibility = 'callback';

app.figure1.Tag = 'figure1';

% Create axes1

app.axes1 = uiaxes(app.figure1);

app.axes1.AmbientLightColor = [0.8824 0.9216 0.9137];

app.axes1.PlotBoxAspectRatio = [1.9802371541502 1 1];

app.axes1.FontSize = 10;

app.axes1.XColor = [0 0 0];

app.axes1.YColor = [0 0 0];

app.axes1.Color = [0.8784 0.9216 0.9098];

app.axes1.NextPlot = 'replace';

app.axes1.Tag = 'axes1';

app.axes1.Position = [360 329 526 275];

% Create text1

app.text1 = uilabel(app.figure1);

app.text1.Tag = 'text1';

app.text1.BackgroundColor = [0.87843137254902 0.92156862745098 0.909803921568627];

app.text1.HorizontalAlignment = 'center';

app.text1.FontName = 'Times New Roman';

app.text1.FontSize = 20;

app.text1.FontWeight = 'bold';

app.text1.Position = [360 133 492 51];

app.text1.Text = '';

% Create pushbutton3

app.pushbutton3 = uibutton(app.figure1, 'push');

app.pushbutton3.ButtonPushedFcn = createCallbackFcn(app, @pushbutton3\_Callback, true);

app.pushbutton3.Tag = 'pushbutton3';

app.pushbutton3.BackgroundColor = [1 1 1];

app.pushbutton3.FontName = 'Times New Roman';

app.pushbutton3.FontSize = 20;

app.pushbutton3.FontWeight = 'bold';

app.pushbutton3.Position = [9 40 224 40];

app.pushbutton3.Text = 'FATIGUE PLOT';

% Create pushbutton4

app.pushbutton4 = uibutton(app.figure1, 'push');

app.pushbutton4.ButtonPushedFcn = createCallbackFcn(app, @pushbutton4\_Callback, true);

app.pushbutton4.Tag = 'pushbutton4';

app.pushbutton4.BackgroundColor = [1 1 1];

app.pushbutton4.FontName = 'Times New Roman';

app.pushbutton4.FontSize = 20;

app.pushbutton4.FontWeight = 'bold';

app.pushbutton4.Position = [11 305 221 38];

app.pushbutton4.Text = 'EMG PLOT';

% Create text4

app.text4 = uilabel(app.figure1);

app.text4.Tag = 'text4';

app.text4.BackgroundColor = [0.913725490196078 0.968627450980392 0.968627450980392];

app.text4.HorizontalAlignment = 'center';

app.text4.FontName = 'Times New Roman';

app.text4.FontSize = 30;

app.text4.FontWeight = 'bold';

app.text4.Position = [9 559 259 78];

app.text4.Text = 'EMG ANALYZER';

% Create pushbutton1

app.pushbutton1 = uibutton(app.figure1, 'push');

app.pushbutton1.ButtonPushedFcn = createCallbackFcn(app, @pushbutton1\_Callback, true);

app.pushbutton1.Tag = 'pushbutton1';

app.pushbutton1.FontName = 'Times New Roman';

app.pushbutton1.FontSize = 25;

app.pushbutton1.FontWeight = 'bold';

app.pushbutton1.FontColor = [0.0706 0.0706 0.0627];

app.pushbutton1.Position = [12 350 221 72];

app.pushbutton1.Text = 'Exercise Analyzer';

% Create pushbutton2

app.pushbutton2 = uibutton(app.figure1, 'push');

app.pushbutton2.ButtonPushedFcn = createCallbackFcn(app, @pushbutton2\_Callback, true);

app.pushbutton2.Tag = 'pushbutton2';

app.pushbutton2.FontName = 'Times New Roman';

app.pushbutton2.FontSize = 25;

app.pushbutton2.FontWeight = 'bold';

app.pushbutton2.Position = [10 88 223 77];

app.pushbutton2.Text = 'Fatigue Analyzer';

% Create text5

app.text5 = uilabel(app.figure1);

app.text5.Tag = 'text5';

app.text5.BackgroundColor = [0.87843137254902 0.92156862745098 0.909803921568627];

app.text5.HorizontalAlignment = 'center';

app.text5.FontName = 'Times New Roman';

app.text5.FontSize = 25;

app.text5.FontWeight = 'bold';

app.text5.Position = [722 195 130 51];

app.text5.Text = 'RESULTS';

% Create text2

app.text2 = uilabel(app.figure1);

app.text2.Tag = 'text2';

app.text2.BackgroundColor = [0.87843137254902 0.92156862745098 0.909803921568627];

app.text2.HorizontalAlignment = 'center';

app.text2.FontName = 'Times New Roman';

app.text2.FontSize = 20;

app.text2.FontWeight = 'bold';

app.text2.Position = [360 70 492 51];

app.text2.Text = '';

% Show the figure after all components are created

app.figure1.Visible = 'on';

end

end

% App creation and deletion

methods (Access = public)

% Construct app

function app = EmgAnalyser\_App(varargin)

% Create UIFigure and components

createComponents(app)

% Register the app with App Designer

registerApp(app, app.figure1)

% Execute the startup function

runStartupFcn(app, @(app)EmgAnalyser\_OpeningFcn(app, varargin{:}))

if nargout == 0

clear app

end

end

% Code that executes before app deletion

function delete(app)

% Delete UIFigure when app is deleted

delete(app.figure1)

end

end

end