

به نام خدا



Blind Source Separation (BSS)

پروژه پایانی

محمد رضا آرانی

810100511

دانشگاه تهران

1402/03/17

جدول محتویات

3	بخش اول:
6	مسئله‌ی Spatial Filter Selection:
7	مسئله‌ی Frequency Selection:
9	مسئله‌ی Channel Selection:
12	مسئله‌ی همگام‌سازی داده‌های زمانی برای هر Trial:
12	مسئله‌ی رسم Confusion Matrix:
15	ساختار کلی
26	جدول نهایی مقایسه خروجی‌ها

بخش اول:

در این پروژه می خواهیم یک مساله ی machine learning مرتبط با درس را پیاده سازی کنیم.

داده های دومین مسابقه ی ملی با موضوع BCI در فولدر dataset در اختیار شما قرار گرفته است. داده های ۱۵ فرد (subject) مختلف در ماتریس های subj_n.mat (برای $n=1,2,...,15$) قرار داده شده است. شرح کامل داده ها در فایل Recording.pdf آورده شده است. ابتدا این فایل را مطالعه کنید و همزمان داده ی یک subject خاص را در متلب لود کنید و نکات مطرح شده در فایل Recording.pdf را کاملاً درک کنید.

از روش هایی که در این درس می آموزید و یا آموخته اید استفاده کنید و ساختاری را برای طبقه بندی این داده ها پیشنهاد کنید.

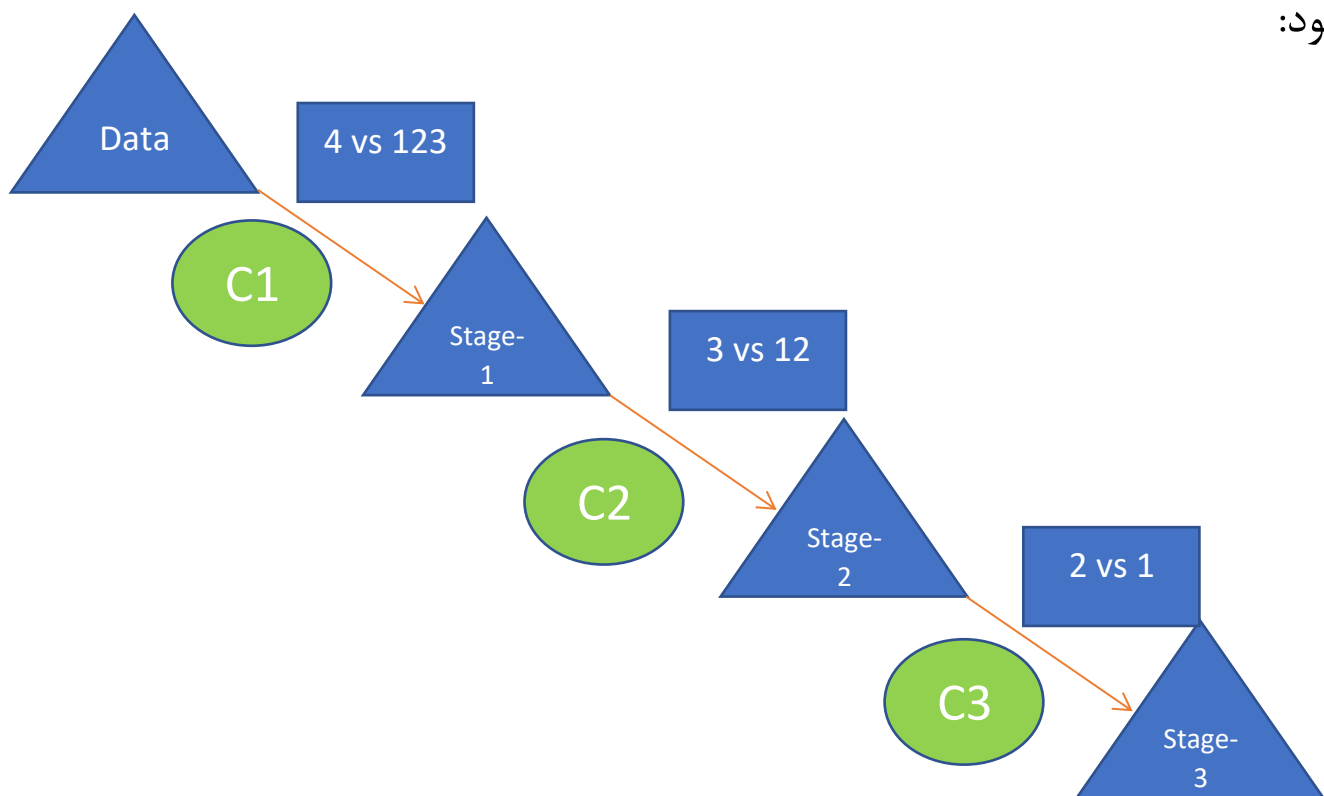
در این پروژه، با توجه به یک مسئله ی بسیار مهم و واقعی، نیاز است تا با دقت بسیار بالا به کلاس بندی داده ها بپردازیم و در نهایت مدل خود را بر اساس متوذهای قابل قبول آزموده و نتیجه ی نهایی را گزارش کنیم.

با توجه به ماهیت پروژه، ما با داده های زمانی با یک لیبیل مشخص روبه رو هستیم که به ازای 15 فرد به صورت cell های 4 در 1 داده شده اند. هر cell دارای یک ماتریس با ابعاد $M \times T \times P$ می باشد که در آن M کانال های نمونه برداری فیزیکی داده هستند،

- قرار بر این است که دو ماتریس *Confusion* یکی برای داده های تست و دیگری برای داده های *Train* در خروجی ارائه شود.

با توجه به ماهیت مسئله و محتوای درسی این *Course*، گزینه‌ی مطلوب برای کلاس‌بندی داده‌های هر شخص، استفاده از روش *Common Spatial Patterns* برای بردن به فضای جدیدی که ویژگی‌های منحصر به فردی دارد که در آن فضا جدایی پذیری *Feature*‌های ما بالاست، توصیه می‌شود.

طبق متودولوژی ارائه شده در کلاس درس، به دلیل ماهیت عملکردی روش *CSP* که برای دو کلاس است، برای کلاس‌بندی این 4 کلاس، از ساختار درختی زیر استفاده می‌شود:



شکل 1

طبق ساختار فوق، ابتدا در کلاس‌بند اول، کلاس 4 را از جمیع کلاس‌های دیگر یعنی مجموع کلاس‌های 1 و 2 و 3 تفکیک می‌کنیم. دلیل اینکار جدایی پذیری بالای این دو کلاس از یکدیگر است. در واقع طبق دسته‌بندی داده‌شده کلاس 4 مربوط به حالت بدون حرکت بوده در صورتی که کلاس 1 و 2 و 3 همگی شامل حرکت اند.

این جدایی و تفکیک توسط یک فیلتر CSP انجام می‌شود که در یک طرف واریانس داده‌های کلاس 4 و در طرف دیگر واریانس داده‌های کلاس کلی 123 را گرفته و سعی در پیدا کردن تبدیلی می‌کند که در فضای پس از تبدیل این دو داده بیشترین نسبت واریانس را داشته باشند.

پس از این مرحله باید طراحی یک جداساز برای تعیین Label کلاس داده‌ها انجام شود! برای اینکار نیز از روش Linear Discriminator Analysis استفاده می‌کنیم.

ترکیب این دو روش در کاربری‌های مربوط به (Brain-Computer Interfaces (BCI بر اساس سیگنال‌های electroencephalography (EEG بسیار معروف بوده و رایج است.

بر این اساس، مطابق ساختار فیلتر کردن و جدایی‌سازی پیاده‌سازی شده در تکلیف شماره 3 سوم، کلاس‌بندی‌ها انجام می‌شود.

پس از جدایی کلاس 4 و 123، نوبت به کلاس بعدی برای تفکیک می‌رسد! این کلاس را کلاس شماره 3 معرفی می‌کنیم چرا که تفکیک‌پذیری آن باید مطابق فلسفه‌ی به

هم پیوستگی داده‌ها راحت تر از جدایی‌پذیری کلاس 2 از 13 باشد چرا که کلاس 3 برای سیگنال‌های متناسب با حرکات پا بوده و کلاس 1 و 2 مربوط به حرکت انگشت شست دست راست و بازو می‌باشند.

با این تفاسیر، کلاس‌بند آخر نیز وظیفه‌ی جداسازی کلاس 1 و 2 از یکدیگر را خواهد داشت.

مسئله‌ی *Spatial Filter Selection*

در این قسمت، در واقع با توجه به انتخاب m تا فیلتر از تمامی فیلترهای یافت شده در روش CSP، که متناسب با آن یک ماتریس مربعی برای W_{CSP} داشتیم، باید تعیین کرد که این مقدار m چه عددی باشد تا تفکیک‌پذیری به درستی انجام شود!

این مسئله بر اساس فهم فیزیکی ما از سیگنال‌های دریافتی قابل انجام نیست چرا که پس از فیلتر کردن، دیگر داده‌ها مفهوم فیزیکی و مکانی خود را ندارند و در فضای جدید نوع داده‌ها برای ما مبهم است! پس به این ترتیب اقدام به انتخاب مقادیر m برای هر فیلتر به صورت جدا و با روش آزمون و خطا خواهیم کرد. البته که ساختار کلی کلاس‌بندها حفظ شده و صرفاً از بهترین فیلترها برای تفکیک کردن خروجی استفاده می‌شود. در واقع این روش، روشی مرسوم برای پیدا کردن Hyper-Parameter های یک مدل Machine Learning است.

```
% Filter Selection:
m1 = 7; m2 = 8; m3 = 7;
Filters_WCSP = [m1,m2,m3];
```

مسئله‌ی Frequency Selection

یکی از مهم‌ترین قسمت‌های این پروژه که تغییر جزئی آن منجر به تغییر بزرگی در خروجی می‌شود، انتخاب درست فیلترهای فرکانس برای هر کلاس‌بند است. با توجه به فرکانس کاری کلی سیگنال‌های مغز، بازه‌ی محدودی از فرکانس را نیاز داریم که برای این کار از فیلترهای میانگذر استفاده می‌کنیم. استفاده از فیلتر Butterworth توصیه نمی‌شود چرا که باید order بالایی برای این بازه‌های کوچک فرکانسی انتخاب شود تا که فیلتر به درستی عمل کند.

برای این کار ما از تبدیل فوریه استفاده کرده‌ایم:

```
function filtered_matrix = Data_Freq_Filter(Data,filter_Class,Freqs_Filts)
    Input_Matrix = Data;
    L_BW1 = Freqs_Filts{1,1}; H_BW1 = Freqs_Filts{1,2};
    L_BW2 = Freqs_Filts{1,3}; H_BW2 = Freqs_Filts{1,4};
    L_BW3 = Freqs_Filts{1,5}; H_BW3 = Freqs_Filts{1,6};
    if (filter_Class==1)
        %L_BW1 = 15; H_BW1 = 40;
        filtered_matrix = Freq_filter(Input_Matrix,L_BW1,H_BW1);
    elseif (filter_Class==2)
        %L_BW2 = 2; H_BW2 = 20;
        filtered_matrix = Freq_filter(Input_Matrix,L_BW2,H_BW2);
    elseif (filter_Class==3)
        %L_BW3 = 0.01; H_BW3 = 7;
        filtered_matrix = Freq_filter(Input_Matrix,L_BW3,H_BW3);
    end
end
```

در تابع فوق، متناسب با نوع کلاس‌بند، از شماره 1 تا 3، فیلتر متناسب اعمال می‌شود.

فیلتر کردن به صورت زیر است:

```
function filtered_matrix = Freq_filter(Input_Matrix,L_BW,H_BW)
% Get dimensions of input matrix
% [M, T, N] = size(Input_Matrix);
% Initialize filtered matrix
% filtered_matrix = zeros(M, T, N);
% Apply filter to each channel and trial
%   for m = 1:M
%       for n = 1:N
%           % Filter the time signal using the Butterworth filter
%           filtered_matrix(m, :, n) = filtfilt(b, a, Input_Matrix(m, :, n));
%       end
%   end
Data1_f = fftshift(fft(Input_Matrix, [], 2), 2);
%   figure
%   plot(f, abs(Data1_f(1,:,1)))
%   grid on
%   xlabel('freq (Hz)')
fs = 2400;
Nfft = 7200;
f = -fs/2 : 1/Nfft*fs : fs/2 - 1/Nfft;
Data1_f(f > H_BW | f < -H_BW) = 0;
Data1_f(f < L_BW & f > -L_BW) = 0;
filtered_matrix = ifft(ifftshift(Data1_f, 2), [], 2);
end
```

با استفاده از این روش، داده‌های فرکانسی فیلتر شده و سپس به دامنه‌ی زمان بازگردانده می‌شوند.

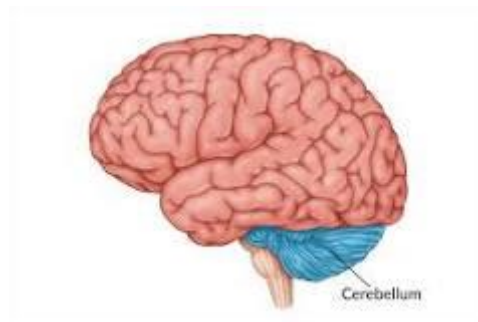
با استفاده از روش قبلی که مبتنی بر آزمون و خطا بود و دانستن بازه‌ی تقریبی فعالیت‌های مغزی، پارامترهای این قسمت نیز تعیین گردید.

```
% Frequency Filters:
Freqs_Filts = {0.01,50 , 2,25, 0.01,20};
```


مسئله‌ی Channel Selection

در این قسمت، باید تعیین کرد که کدام کانال‌های فیزیکی مشخص شده در قسمت Recording برای این تفکیک کردن و کلاس‌بندی مفید اند!

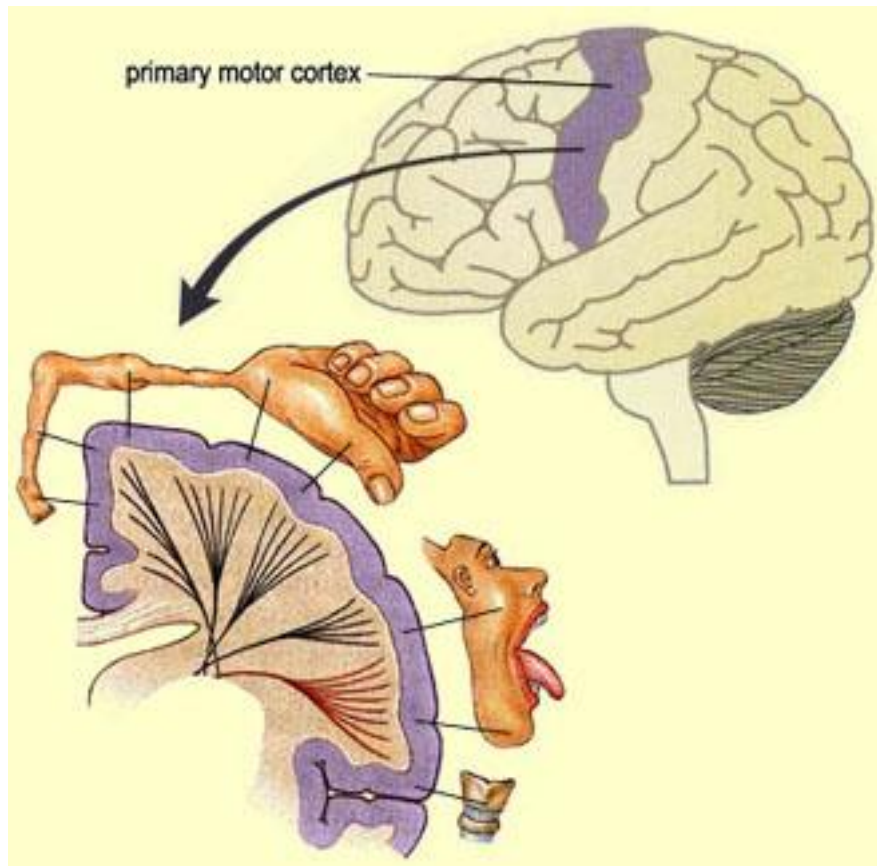
برای این قسمت نیز، متناسب با قسمت‌های قبل، از روش آزمون و خطا مبتنی بر دانش نسبی از قسمت‌های مختلف مغز استفاده شده است.



شکل 2

برای مثال در شکل فوق، قسمت Cerebellum در مغز را مشاهده می‌کنیم که اصول کارهای حرکتی بر اساس تحریک این قسمت از مغز انجام می‌شود. [1]

[1] <https://my.clevelandclinic.org/health/body/23418-cerebellum>



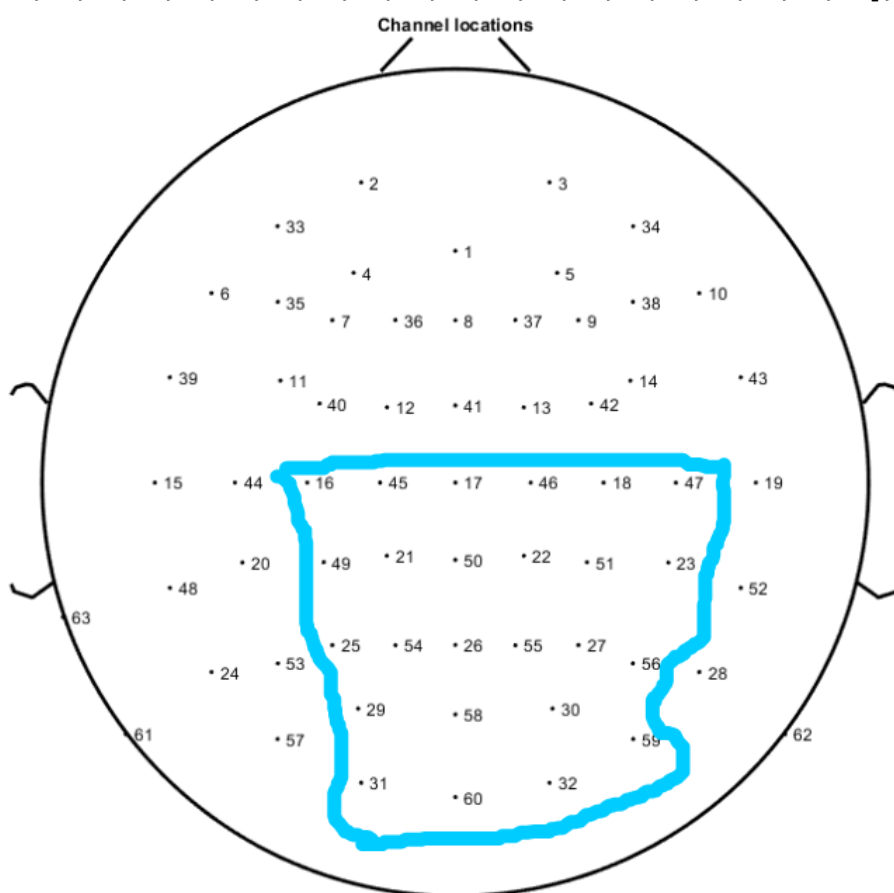
شکل 3

یا برای مثال، قسمت خاص دیگری در مغز که وظیفه‌ی تحریک موتورهای حرکتی را دارد، قسمت مشخص شده در تصویر است. [2]

[2] https://thebrain.mcgill.ca/flash/d/d_06/d_06_cr/d_06_cr_mou/d_06_cr_mou.html

با این دانش نسبی، فیلترهای زیر از روی شکل انتخاب شدند:

```
channels1 = 1:45; channels2 = 20:45; channels3 =  
[49,18,16,17,45,54,21,29,31,57,22,23,25,26,27,30,32,50,51,55,56,58,59,60,62];
```



کانال‌های انتخاب شده برای تحرک دست راست 4 شکل

مسئله‌ی همگام‌سازی داده‌های زمانی برای هر Trial:

در این قسمت، مهم است که بدانیم که داده‌های زمانی داده شده به صورت زمانی با یکدیگر sync نشده اند و زمان شروع هر حرکت برای هر شخص متفاوت است. این اندیس شروع در کانال 64ام آورده شده است و از آن برای هماهنگ‌سازی زمان شروع حرکت برای هر شخص استفاده می‌شود. نکته‌ی مهم دادن شیفت زمانی متناسب با این مقدار زمانی مشخص شده برای هر داده است.

مسئله‌ی رسم Confusion Matrix:

در این قسمت، برای رسم ماتریس آشفتگی یا همان Confusion Matrix که درواقع منبع استخراج معیارهای عملکردی مدل است، پیشنهاد ارائه می‌کنیم:

برای به دست آوردن ماتریس آشفتگی داده‌های آموزش، از روش زیر استفاده می‌شود:

```
Train_Pred = cell(1,size(Data_Train,2)); Num_of_Trials_Tot = zeros(1,size(Data_Train,2));
for j=1:size(Data_Train,2)
    Input_Train_Data = Data_Train{1,j};
    Predicted_Train = zeros(size(Data_Train{1,j},3),1) ;
    for p=1:size(Data_Train{1,j},3)
        Predicted_Train(p,1) = Tester(W_LDA, W_CSP , C,Means ,
squeeze(Input_Train_Data(:, :, p)) , Channels_Cell, Filters_WCSP);
        Train_Conf( j , Predicted_Train(p,1)) = Train_Conf( j , Predicted_Train(p,1)) +1;
    end
    Train_Pred{1,j} = Predicted_Train;
    Num_of_Trials_Tot(1,j) = p;
```

```
end
```

```
Conf_Train_Matrixes{1,i} = Train_Conf;
```

در این روش در واقع فرض شده است که سطرها اندیس درست و True بودند و ستونها اندیس Predicted که بر همین اساس ماتریس Confusion به دست می‌آید.

پس از به دست آمدن مقادیر این ماتریس برای رسم آن از قسمت زیر استفاده می‌شود:

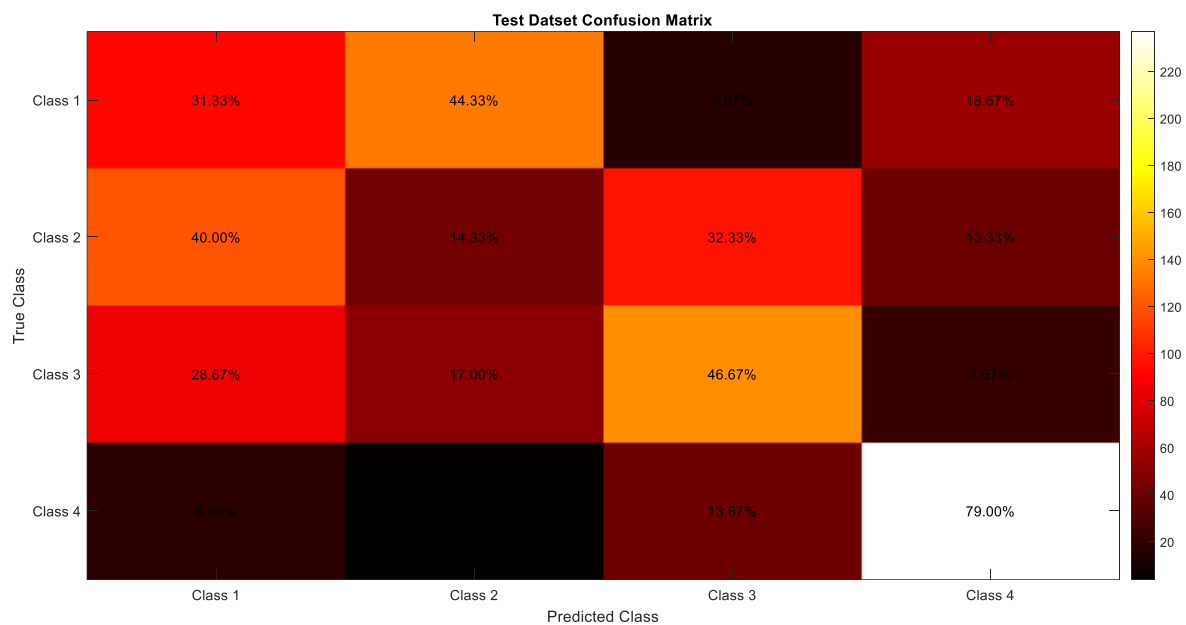
```
function draw_Conf(Final_Conf,Num_of_Iters,string_Mine)
    figure()
    imagesc(Final_Conf)
    xlabel("Predicted Class")
    ylabel("True Class")

    % Set axis labels and tick marks
    classLabels = {'Class 1', 'Class 2', 'Class 3' , 'Class 4'};
    numClasses = numel(classLabels);
    set(gca, 'XTick', 1:numClasses, 'XTickLabel', classLabels);
    set(gca, 'YTick', 1:numClasses, 'YTickLabel', classLabels);
    % Adjust the color map to emphasize differences
    colormap('hot');
    colorbar
    title(string_Mine);

    % Add the values to the cells of the confusion matrix
    for i = 1:numClasses
        for j = 1:numClasses
            text(j, i, sprintf('%.2f%%', Final_Conf(i, j)/Num_of_Iters*100), ...
                'HorizontalAlignment', 'Center');
        end
    end
end
```

```
end
```

خروجی این تابع به صورت زیر است:



شکل 5

به تصویر کشیدن ماتریس آشفتگی در متلب

ساختار کلی

ساختار کلی ارائه شده برای حل این مسئله، به صورت زیر است:

- قسمت اول که برای بارگذاری داده‌های شخص مشخص شده است

```
clear; clc; close all;
% Load sample Data:
Subject_Num = 7;
Data_Subject = load("dataset\subj_"+Subject_Num+".mat");
Data_Subject_i = Data_Subject.data;
```

- قسمت دوم که برای شروع الگوریتم و مقداردهی پارامترهای آن است:

```
%% Run The Algorithm:
```

```
Num_of_Iters = 300;
Conf_Matrixes = cell(1,Num_of_Iters);
Conf_Train_Matrixes = Conf_Matrixes;
Final_Conf = zeros(size(Data_Subject_i,2));
Train_Conf = Final_Conf;

% Channel Selection:
channels1 = 1:63; channels2 = 20:45; channels3 =
[49,18,16,17,45,54,21,29,31,57,22,23,25,26,27,30,32,50,51,55,56,58,59,60,62]; %
22,23,25,26,27,30,32,50,51,55,56,58,59,60,62,
Channels_Cell = {channels1 ,channels2 ,channels3 };
% Filter Selection:
m1 = 7; m2 = 8; m3 = 7;
Filters_WCSP = [m1,m2,m3];
% Frequency Filters:
Freqs_Filts = {0.01,50 , 2,25, 0.01,20};

for i=1:Num_of_Iters
    [Data_Train , Data_Test] = Test_Train_Dist(Data_Subject_i);
    [W_CSP , Z ] = Trainer_W_CSP(Data_Train,Channels_Cell,Filters_WCSP,Freqs_Filts);
    [W_LDA , C , Means] = Trainer_LDA(Z);
    Predicted = zeros(1,size(Data_Test,2)) ;

    % Train COnfusion Matrix
    Train_Pred = cell(1,size(Data_Train,2)); Num_of_Trials_Tot = zeros(1,size(Data_Train,2));
    for j=1:size(Data_Train,2)
        Input_Train_Data = Data_Train{1,j};
        Predicted_Train = zeros(size(Data_Train{1,j},3),1) ;
        for p=1:size(Data_Train{1,j},3)
```

```

        Predicted_Train(p,1) = Tester(W_LDA, W_CSP , C,Means ,
squeeze(Input_Train_Data(:, :, p)) , Channels_Cell, Filters_WCSP);
        Train_Conf( j , Predicted_Train(p,1)) = Train_Conf( j , Predicted_Train(p,1)) +1;
    end
    Train_Pred{1,j} = Predicted_Train;
    Num_of_Trials_Tot(1,j) = p;

end
Conf_Train_Matrxes{1,i} = Train_Conf;

% Train_Conf = Train_Conf + confusionmat([1,2,3,4], Predicted_Train) ;
% Test Confusion Matrix
for j=1:size(Data_Test,2)

        Predicted(1,j) = Tester(W_LDA, W_CSP , C,Means ,
Data_Test{1,j}, Channels_Cell, Filters_WCSP);
    end
    Conf_Matrxes{1,i} = confusionmat([1,2,3,4], Predicted) ;
    Final_Conf = Final_Conf + Conf_Matrxes{1,i};
    clc;
    disp("Progress: %" + i/Num_of_Itrs*100 );
    fprintf('\b\b\b\b\b\b\b\b3d%% [%s]', round(i / Num_of_Itrs * 100), repmat('=', 1,
round(i/Num_of_Itrs*100)));

end

disp("Final Confusion Matrix equals to: ")
disp(Final_Conf/Num_of_Itrs);
Accuracy_Test_Class_1 = Final_Conf(1,1)/Num_of_Itrs*100;
Accuracy_Test_Class_2 = Final_Conf(2,2)/Num_of_Itrs*100;
Accuracy_Test_Class_3 = Final_Conf(3,3)/Num_of_Itrs*100;
Accuracy_Test_Class_4 = Final_Conf(4,4)/Num_of_Itrs*100;
disp("Accuracies are: 1: " + Accuracy_Test_Class_1 + "% 2: " + Accuracy_Test_Class_2 + "% 3: " + ...
Accuracy_Test_Class_3 + "% 4: " + Accuracy_Test_Class_4 + "%");

draw_Conf(Final_Conf, Num_of_Itrs, "Test Dataset Confusion Matrix");

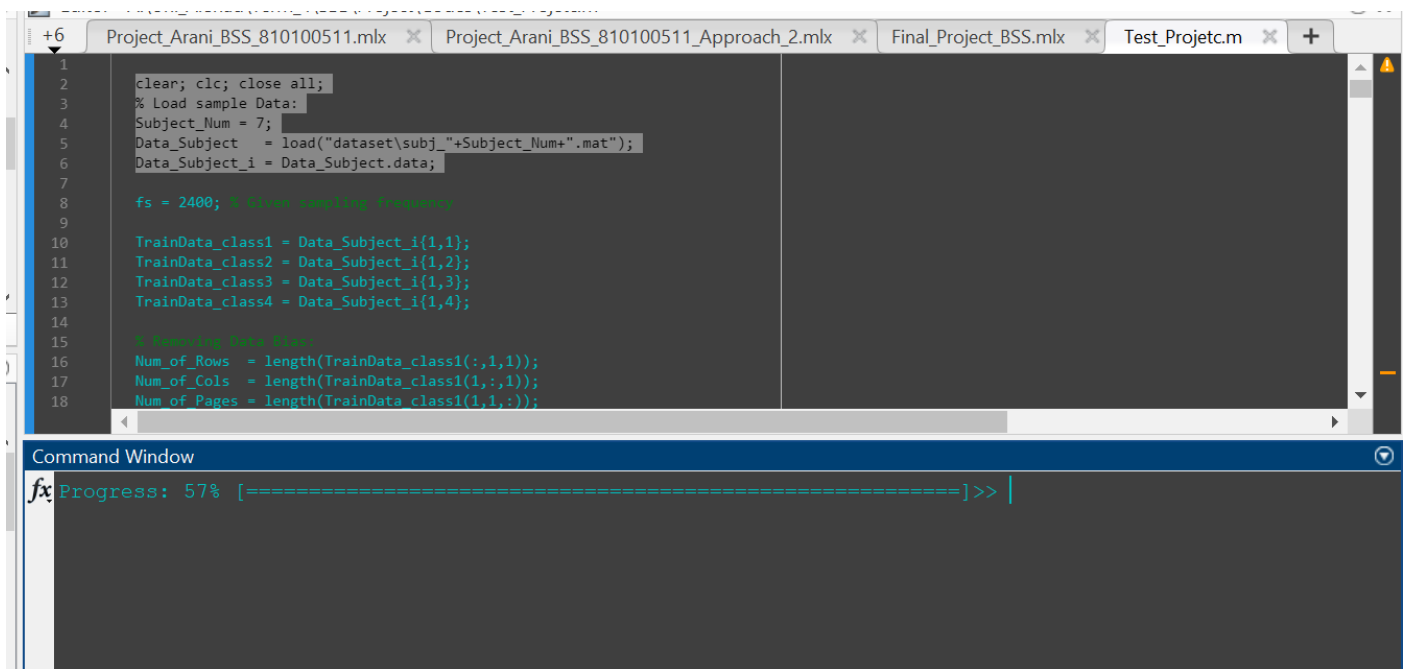
draw_Conf(Train_Conf, 1, "Train Dataset Confusion Matrix");
beep;
% Save Data:
% save("Results_Project_Test_Itrs_" + Num_of_Itrs + "_Accuracies_1_" + Accuracy_Test_Class_1
+ "_2_" + Accuracy_Test_Class_2 + "_3_" + ...
Accuracy_Test_Class_3 + "_4_" + Accuracy_Test_Class_4);

save("Results_Project_Test_Itrs_" + Num_of_Itrs + "_Accuracies_1_" + round(Accuracy_Test_Class_1)
+ "_2_" + round(Accuracy_Test_Class_2) + "_3_" + ...
round(Accuracy_Test_Class_3)
+ "_4_" + round(Accuracy_Test_Class_4));

```


"Final_Conf", "Num_of_Iters", "Filters_WCSP", "Means", "W_CSP", "W_LDA", "Subject_Num", "Channels_Cell", "Freqs_Filts", "Train_Conf", "Conf_Train_Matrixes", "Conf_Matrixes")

در این ساختار، پس از ران کردن کد، یک ProgressBar به فرمت زیر درصد پیشرفت کد را بر اساس تعداد تکرار مشخص شده نمایش می‌دهد:



The screenshot shows a MATLAB script in the Editor and the Command Window. The script defines sampling frequency, loads data, and processes it. The Command Window shows the progress bar at 57%.

```

1 clear; clc; close all;
2 % Load sample Data:
3 Subject_Num = 7;
4 Data_Subject = load("dataset\subj_"+Subject_Num+".mat");
5 Data_Subject_i = Data_Subject.data;
6
7 fs = 2400; % Given sampling frequency
8
9
10 TrainData_class1 = Data_Subject_i{1,1};
11 TrainData_class2 = Data_Subject_i{1,2};
12 TrainData_class3 = Data_Subject_i{1,3};
13 TrainData_class4 = Data_Subject_i{1,4};
14
15 % Removing Data files
16 Num_of_Rows = length(TrainData_class1(:,1,1));
17 Num_of_Cols = length(TrainData_class1(1,:,1));
18 Num_of_Pages = length(TrainData_class1(1,1,:));

```

Command Window

```

fx Progress: 57% [=====]>> |

```

شکل 6

با رسیدن به انتهای این ProgressBar، ماتریس آشفته‌گی برای Train و Test به صورت جدا نمایش داده می‌شود.

- همچنین صدای beep به صورت خودکار پس از اتمام پردازش پخش می‌شود که به معنای اتمام پردازش و آمادگی برای ران کردن بار دیگر است.

به صورت جزئی، ساختار ما شامل ماژول‌های مختلف که اولین آنها شامل تقسیم داده‌ها به دو دسته‌ی Train و Test می‌شود که کار Data Tyding نیز در این قسمت انجام می‌شود!

```
[Data_Train , Data_Test] = Test_Train_Dist(Data_Subject_i);
```

این تابع به صورت زیر پیاده‌سازی شده است:

```
function [Data_Train , Data_Test] = Test_Train_Dist(Data_Subject_i)
```

```
TrainData_class1 = Data_Subject_i{1,1};
TrainData_class2 = Data_Subject_i{1,2};
TrainData_class3 = Data_Subject_i{1,3};
TrainData_class4 = Data_Subject_i{1,4};
```

```
% Removing Data Bias:
% Num_of_Rows = length(TrainData_class1(:,1,1));
Num_of_Cols = length(TrainData_class1(1,:,1));
% Num_of_Pages = length(TrainData_class1(1,1,:));
```

```
TrainData_class1 = TrainData_class1 - repmat( mean(TrainData_class1, 2) ,1,Num_of_Cols) ;
TrainData_class2 = TrainData_class2 - repmat( mean(TrainData_class2, 2) ,1,Num_of_Cols) ;
TrainData_class3 = TrainData_class3 - repmat( mean(TrainData_class3, 2) ,1,Num_of_Cols) ;
TrainData_class4 = TrainData_class4 - repmat( mean(TrainData_class4, 2) ,1,Num_of_Cols) ;
```

```
% Distinguish Test & Train:
```

```
Test_IDX = [randi(size(TrainData_class1,3)) , randi(size(TrainData_class2,3)) , ...
            randi(size(TrainData_class3,3)), randi(size(TrainData_class4,3)) ];
```

```
Test_class1 = TrainData_class1(:, :, Test_IDX(1));
Test_class2 = TrainData_class2(:, :, Test_IDX(2));
Test_class3 = TrainData_class3(:, :, Test_IDX(3));
Test_class4 = TrainData_class4(:, :, Test_IDX(4));
```

```
% Removing Test Data from Train Dataset:
```

```
TrainData_class1(:, :, Test_IDX(1)) = [];
TrainData_class2(:, :, Test_IDX(2)) = [];
TrainData_class3(:, :, Test_IDX(3)) = [];
TrainData_class4(:, :, Test_IDX(4)) = [];
```

```
% Changing Mobility factor of class 4 into 0:
```

```
TrainData_class4(end, :, :) = 0;
```

```
% Remove Channel 64 which is the origin of recording:
```

```
TrainData_class1(end, :, :) = [];
TrainData_class2(end, :, :) = [];
```

```
TrainData_class3(end,:,:) = [];
TrainData_class4(end,:,:) = [];
```

```
% Remove Final Row from Test Data (row 64):
```

```
Test_class1(end,:,:) = [];
Test_class2(end,:,:) = [];
Test_class3(end,:,:) = [];
Test_class4(end,:,:) = [];
```

```
Data_Train = {TrainData_class1, TrainData_class2, TrainData_class3 , TrainData_class4};
Data_Test  = {Test_class1 , Test_class2 , Test_class3 , Test_class4};
```

```
end
```

همانطور که مشاهده می‌شود داده‌های ما بدون بایاس به قسمت بعدی خواهند رسید چرا که از آنها میانگین داده‌های زمانی حذف شده است!

• قسمت بعدی یکی از مهم‌ترین قسمت‌های پردازش یعنی طراحی فیلتر مکانی و انتقال داده‌ها به فضای تصویر است:

```
[W_CSP , Z ] = Trainer_W_CSP(Data_Train,Channels_Cell,Filters_WCSP,Freqs_Filts);
```

که این تابع به صورت زیر پیاده‌سازی شده است:

```
function [W_CSP , Z ] = Trainer_W_CSP(Data,Channels_Cell,Filters_WCSP,Freqs_Filts)
```

```
TrainData_class1 = Data{1,1};
TrainData_class2 = Data{1,2};
TrainData_class3 = Data{1,3};
TrainData_class4 = Data{1,4};
```

```
% Data_Filtered = Data_Freq_Filter(Data,filter_CClass)
```

```
% Channel Selection
```

```
channels1 = Channels_Cell{1,1} ; channels2 = Channels_Cell{1,2} ; channels3 = Channels_Cell{1,3} ;
```

```
% Filter Selection:
```

```
m1 = Filters_WCSP(1,1); m2 = Filters_WCSP(1,2); m3 = Filters_WCSP(1,3);
```

```
% Covariance Calc:
```

```
% R_class_1_Sum = R_SUM_Calc(TrainData_class1);
% R_class_2_Sum = R_SUM_Calc(TrainData_class2);
% R_class_3_Sum = R_SUM_Calc(TrainData_class3);
% R_class_4_Sum = R_SUM_Calc(TrainData_class4);
```

```

TrainData_class123 = cat(3,TrainData_class1, TrainData_class2 , TrainData_class3);
% R_class_123_Sum = R_SUM_Calc(TrainData_class123);

TrainData_class12 = cat(3,TrainData_class1, TrainData_class2 );
% R_class_12_Sum = R_SUM_Calc(TrainData_class12);

% Now Let's Map Data properly based on The Decision Tree Model:
% Classifier_1_for4 = CSP_Filter_W(R_class_4_Sum,R_class_123_Sum);
% Classifier_2_for3 = CSP_Filter_W(R_class_3_Sum,R_class_12_Sum);
% Classifier_3_Final = CSP_Filter_W(R_class_1_Sum,R_class_2_Sum);

% Frequency Filtering:
TrainData_class4_Filtered = Data_Freq_Filter(TrainData_class4,1,Freqs_Filts);
TrainData_class123_Filtered = Data_Freq_Filter(TrainData_class123,1,Freqs_Filts);

TrainData_class3_Filtered = Data_Freq_Filter(TrainData_class3,2,Freqs_Filts);
TrainData_class12_Filtered = Data_Freq_Filter(TrainData_class12,2,Freqs_Filts);

TrainData_class2_Filtered = Data_Freq_Filter(TrainData_class2,3,Freqs_Filts);
TrainData_class1_Filtered = Data_Freq_Filter(TrainData_class1,3,Freqs_Filts);

[Classifier_1_for4 , Z_Class4_vs_123 , Z_Class123_vs_4 ] =
Filter_CSP(TrainData_class4_Filtered(channels1,:,:) ,
TrainData_class123_Filtered(channels1,:,:));
[Classifier_2_for3 , Z_Class3_vs_12 , Z_Class12_vs_3 ] =
Filter_CSP(TrainData_class3_Filtered(channels2,:,:) ,
TrainData_class12_Filtered(channels2,:,:));
[Classifier_3_Final , Z_Class2_vs_1 , Z_Class1_vs_2 ] =
Filter_CSP(TrainData_class2_Filtered(channels3,:,:) ,
TrainData_class1_Filtered(channels3,:,:));
% Map Data:
% Z_Class4_vs_123 = tensorprod(Classifier_1_for4',TrainData_class4,2,1);
% Z_Class123_vs_4 = tensorprod(Classifier_1_for4',TrainData_class123,2,1);

% Z_Class3_vs_12 = tensorprod(Classifier_2_for3',TrainData_class3,2,1);
% Z_Class12_vs_3 = tensorprod(Classifier_2_for3',TrainData_class12,2,1);
%
% Z_Class1_vs_2 = tensorprod(Classifier_3_Final',TrainData_class1,2,1);
% Z_Class2_vs_1 = tensorprod(Classifier_3_Final',TrainData_class2,2,1);

W_CSP = {Classifier_1_for4(:, [1:m1, end-m1+1:end] ) , ...
Classifier_2_for3(:, [1:m2, end-m2+1:end] ) , ...
Classifier_3_Final(:, [1:m3, end-m3+1:end] ) };

Z = { Z_Class4_vs_123([1:m1, end-m1+1:end],:,:),Z_Class123_vs_4([1:m1, end-m1+1:end],:,:)
...

```

```
, Z_Class3_vs_12([1:m2, end-m2+1:end],:,:),Z_Class12_vs_3([1:m2, end-m2+1:end],:,:)...
, Z_Class1_vs_2([1:m3, end-m3+1:end],:,:), Z_Class2_vs_1([1:m3, end-m3+1:end],:,:)}};
```

```
end
```

در نهایت پس از این قسمت، 3 کلاس بند به همراه داده‌های تصویر شده به محیط جدید، به خروجی و قسمت بعدی خواهند رسید.

اساس کد قسمت بالا، تابع زیر است که مسئله‌ی ابتدایی جداسازی دو کلاس را حل می‌کند:

```
function W_CSP = CSP_Filter_W(R1,R2)

% Use GEVD (Generalized Eigen-Value Decomposition) over R_class_1_Mean on
% top and R_class_2_Mean in denominator:

[W_CSP,Lambda] = eig(R1,R2) ;
[~, ind] = sort(diag(Lambda),'descend'); % Sort based on eigen values
W_CSP = W_CSP(:, ind);
% Maximizing the Rayleigh Ratio requires the biggest Lambda and its
% corresponding Eigen Vector to be chosen!

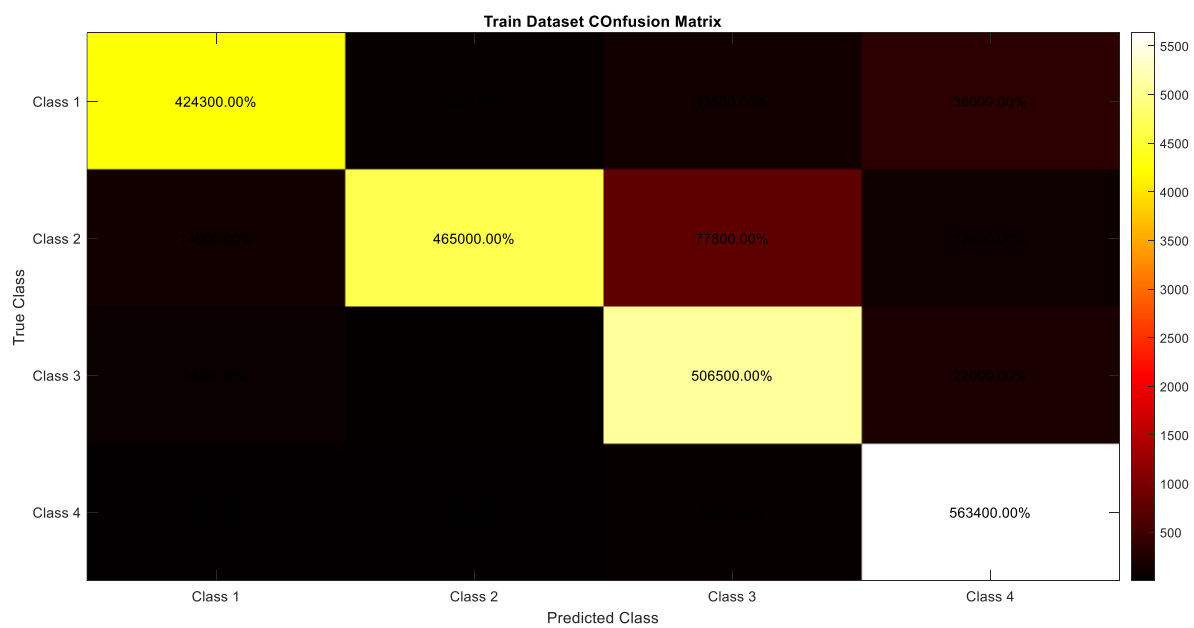
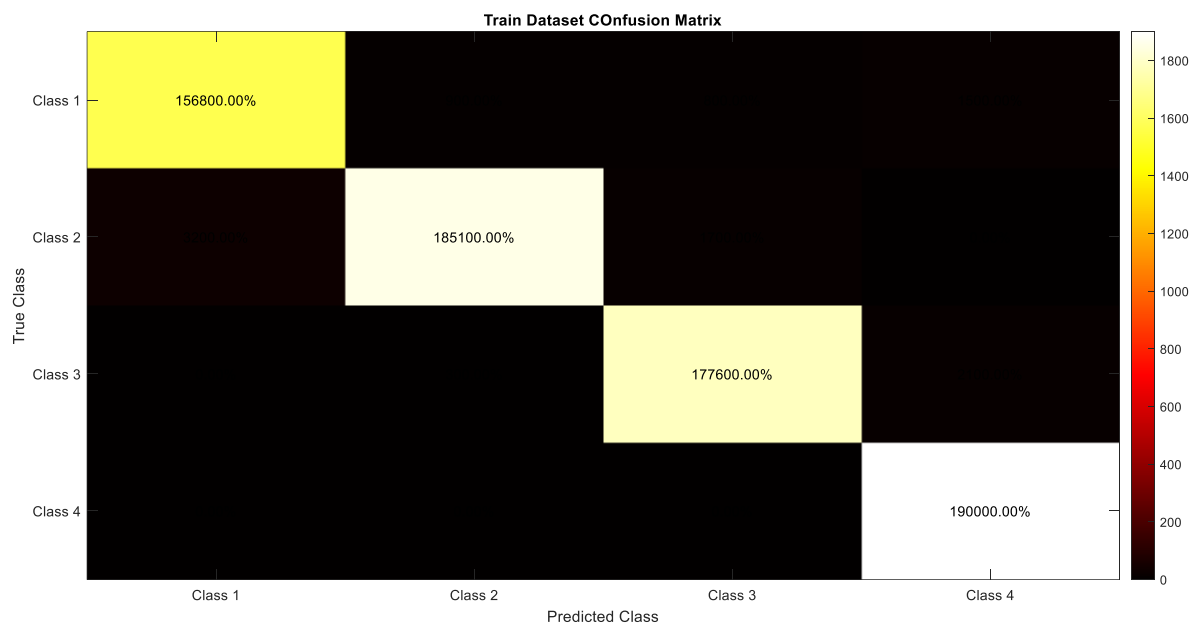
% Normalizing W Columns:
Norm_Matrix_of_Cols_W = W_CSP'*W_CSP;
for j=1:length(R1)
    W_CSP(:,j) = W_CSP(:,j)/(Norm_Matrix_of_Cols_W(j,j));
end
end
```

قسمت نهایی نیز، طراحی LDA می‌باشد:

```
[W_LDA , C , Means] = Trainer_LDA(Z);
```

که بر اساس داده‌های تصویر شده و Feature استخراج شده از این داده‌ها که همان واریانس داده‌ها می‌باشد، عمل می‌کند.

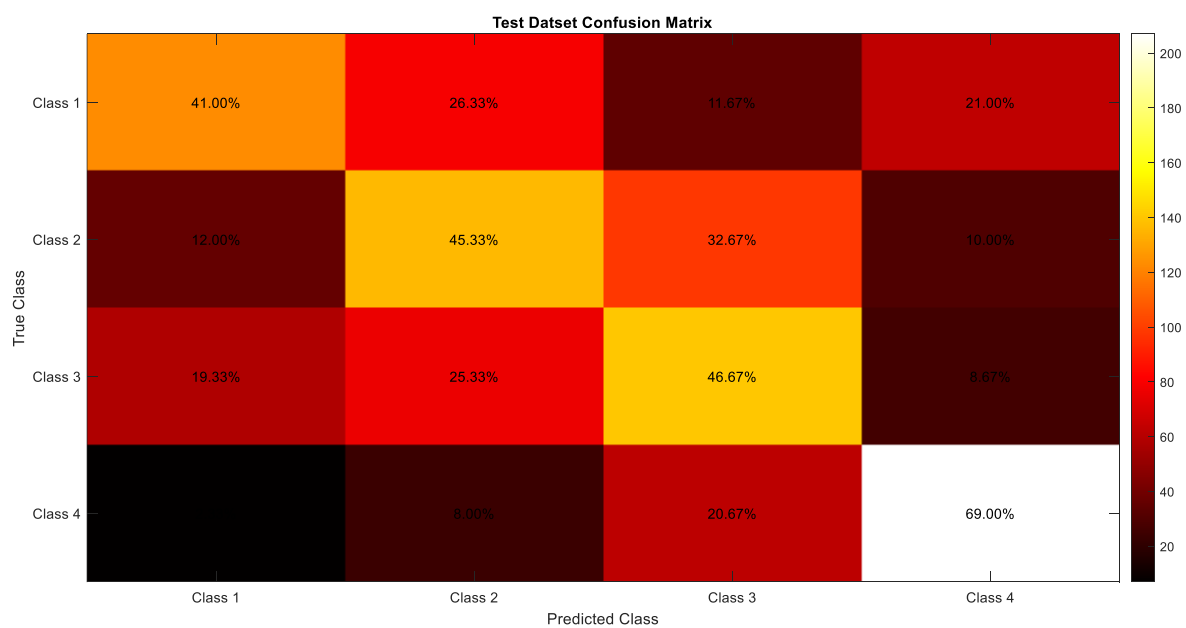
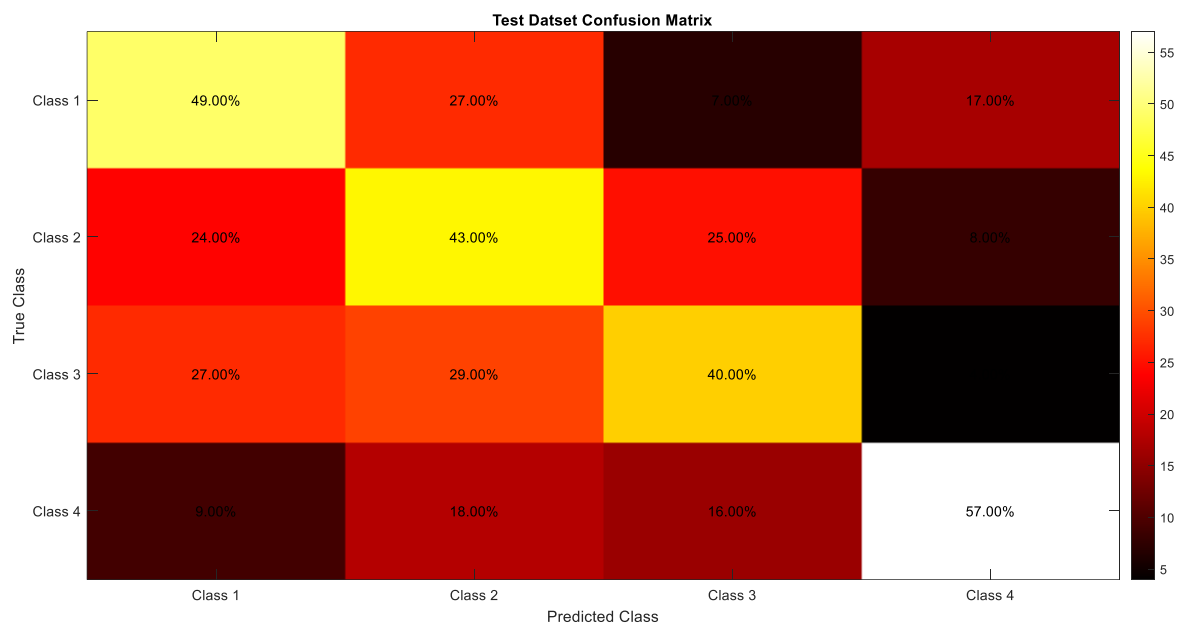
BSS-Project-2023



7 شکل

پس از کلاس‌بندی به صورت مناسب ماتریس آشفتگی برای داده‌های آموزش به صورت
فوق خواهد بود

BSS-Project-2023



شکل ۸

پس از کلاس‌بندی به صورت مناسب ماتریس آشفتگی برای داده‌های تست به صورت فوق خواهد بود

در حال حاضر بهترین عملکرد را در بین تمامی تست‌های انجام شده، کلاس 4 با عملکرد موفق 100 درصدی داشته است.

• لازم به ذکر است که تمامی داده‌های به دست آمده در فولدر پروژه ذخیره شده اند!

شبه کد زیر ساختار کلی عملکردی را نمایش می‌دهد:

1. Clean Loaded Data, Remove Final Channel, Split to Test & Train, remove Data Bias
2. Obtain optimum Filters for each classifier and MAP data based on
 - a. Frequency Filtered Data
 - b. Channel Selected Data
 - c. Selected Filters
 - d. Synchronized Data
3. Test both Train and Test data and form a confusion Matrix for each of them!
4. Play beep sound when algorithm is Done and save Important Parameters.

در نهایت با توجه به این که نتایج بر حسب ران‌های مختلف دارای نوسان جزئی اند، بهترین‌ها در جدول زیر به تناسب آورده شده اند که داده‌ی هر Run متفاوت به همراه فایل ارسالی خواهد بود:

جدول نهایی مقایسه خروجی‌ها

جدول 1 مقایسه نتایج مختلف بر اساس پارامترهای مدل

Class1	Class2	Class3	Class4	Freq_Filters	Channels	CSP_Filters	Iters
27.67	12.67	49	83.67	0.01:50; 2:25; 0.01:20	1:45 20:45 4 th Orthant	[7,8,3]	300
31.33	14.33	46	79	0.01:50; 2:25; 0.01:20	1:45 20:45 4 th Orthant	[7,8,7]	300
20	0	50	100	0.01:50; 2:25; 0.01:20	1:45 20:45 4 th Orthant	[7,8,3]	10
34	47	45	78	0.01:50; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[5,8,7]	100

BSS-Project-2023

49	43	40	57	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[13,11,11]	
39	27	27	81	?	1:63 20:60 4 th Orthant	[7,8,3]	100
38	12	45	84	0.01:50; 2:25; 0.01:20	1:45 20:45 4 th Orthant	[7,8,3]	100
34	56	56	88	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[5,8,7]	100
36	54	55	84	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[5,8,7]	100
30	55	53	82	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[5,8,7]	100
36	46	44	86	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[5,8,7]	100
35	47	47	80	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[5,11,7]	100

BSS-Project-2023

36	62	39	86	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[3,8,7]	100
30	32	40	87	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[3,15,7]	100
41	61	53	89	0.01:55; 2:25; 0.01:20	1:63 10:55 4 th Orthant	[5,8,7]	100
43	45	46	90	0.01,15 2,30 0.01,20	1:63 10:55 4 th Orthant	[3,8,9]	100
70	52	51	100 Subject6	0.01,15 2,30 0.01,20	1:63 10:55 4 th Orthant	[3,8,9]	100

پایان