# Mini Projeto IAPS

Professora Susana Brás| Gonçalo Costa 118838| Pedro Vieira 120584

# Análise de EEG para estudo das diferentes fases do sono

## Resumo

Este miniprojecto tem como principal objetivo analisar sinais EEG e compreender as suas características, com foco na identificação e interpretação das bandas de frequência associadas aos diferentes estágios. De modo a tornar a aprendizagem mais iterativa, desenvolvemos um jogo didático que mostra o comportamento das bandas *Delta*, *Teta*, *Alfa*, *Beta* e *Gama*, com o objetivo de o utilizador observar e tentar descobrir qual é o estágio com base na visualização das potências relativas e absolutas por estágio previamente mostradas.

Complementarmente, realizamos uma experiência (experiência esta que pode ser visualizada no apêndice) com recurso a inteligência artificial, onde utilizamos um classificador de *machine learning*, com o objetivo de explorar o seu desempenho na análise de sinais EEG semelhantes, aprofundando assim a compreensão do seu funcionamento.

## 1. Introdução

O sono é uma das mais importantes funções biológicas do ser humano, possuindo um papel crítico na saúde física e mental. É durante o sono que o nosso corpo regenera, reparando as células, regulando diversas funções metabólicas e psicológicas e é também essencial para manter as nossas funções cognitivas saudáveis, tais como a consolidação da memória ou a capacidade de foco.[1] Este está dividido em 5 estágios diferentes: W, N1, N2, N3, N4, REM, sendo que cada um pode ocorrer uma ou mais vezes durante todo o sono. No entanto, em 2007, a Academia Americana de Medicina do Sono (AASM) decidiu fundir os estágios N3 e N4 num único estágio de sono profundo, N3.[2][3]

Tendo como objetivo analisar as diferentes fases do sono, é essencial perceber quais as diferentes frequências presentes, assim como a amplitude e fase de cada uma delas. Para tal, temos de transformar os sinais originais que estão no domínio do tempo, para as suas representações no domínio da frequência, utilizando a função "fft". É também importante mencionar que os dados de EEG analisados são dados estocásticos e, portanto, as suas flutuações ao longo do tempo podem ser parcialmente aleatórias, o que faz com que dados futuros não possam ser totalmente determinados baseando-se nos dados anteriores. Deste modo procedemos a uma análise cuidadosa, visando obter o máximo determinismo possível.[4]

Ao longo do código recorremos a algumas funções que não foram mencionadas nas aulas, mas que eram imprescindíveis para obtermos o resultado esperado. Primeiramente, utilizamos a função "edfread" que serve para ler ficheiros no formato EDF (European Data Format), uma vez que era neste formato que se encontravam os ficheiros da base de dados utilizada. Do mesmo modo, a função "edfinfo" serve para providenciar mais informações do ficheiro EDF. O comando "containers.Map" serve como dicionário para associar cada parâmetro a um determinado valor. Para o classificador necessitámos de utilizar as funções "normalize" e "pca" e ainda escolhemos recorrer ao "TreeBagger". Tal como o nome indica, a função "normalize" padroniza os dados ao subtrair a média aos mesmos e divide pelo desvio padrão, resultando numa média nula e variância unitária para cada característica. De seguida, utilizamos o "pca" ou Principal Component Analysis, uma técnica que reduz a dimensionalidade de um conjunto de dados, extraindo as componentes principais, mas preservando o máximo de variabilidade do mesmo. É possível extrair desta função diversos outputs, tais como, os coeficientes dos componentes principais, o novo conjunto de dados transformado em função destes componentes ou ainda a percentagem de variância total que cada um explica ("coeff", "scores" e "explained" respetivamente). Finalmente, o modelo "TreeBagger" é uma implementação do algoritmo "Random Forest", que consiste num método de aprendizagem por conjuntos, cujo principal objetivo é reduzir o *overfitting* e melhorar a precisão preditiva. Simplificando, funciona através da criação de múltiplas árvores de decisão e depois combina as suas diferentes previsões de modo a chegar a uma decisão final. Por fim, utilizamos a função "confusionchart" que nos permite visualizar a matriz confusão associada ao nosso modelo de classificação, dado que esta matriz nada mais é que uma tabela avaliadora do modelo em questão.

## 2. Métodos e Discussão

```
clc;
clear all;
close all;

%% 1. Carregar Dados

PSG='SC4021E0-PSG.edf';
HYP='SC4021EH-Hypnogram.edf';

% Ler o sinal EEG
psg_dados = edfread(PSG);
eeg_sinal = psg_dados.EEGFpz_Cz;  % Pega o sinal EEG da coluna correta
fs = 100;  % Frequência de amostragem (Hz)

% Ler o hipnograma
hyp_dados = edfread(HYP);
info = edfinfo(HYP);
anotacoes = info.Annotations;
tempo = seconds(anotacoes.Onset);
estagios = anotacoes(:,"Annotations");

```

Figura 1- Extração do Sinal

Esta Figura 1 mostra a parte do nosso código que carrega os ficheiros de um paciente (EEG e hipnograma). Primeiro utilizamos edfread para ler o ficheiro do EEG no seu formato EDF. Como o sinal EEG está numa das colunas da matriz do ficheiro PSG utilizamos o EEGFpz_Cz para extrair esse sinal e consideramos a frequência de amostragem 100 Hz de acordo com os dados da base.

Para a leitura do hipnograma procedemos de maneira semelhante. Utilizamos novamente o edfread para importar os dados e o edfinfo para obter mais informações do ficheiro. Depois com o Annotations, retiramos os tempos de início de cada estágio (Onset) e os rótulos de cada estágio de sono.

```
%% 2. Pré-processamento dos Estágios (mapear para janelas de 30s)

% Dicionário dos estágios
dic_estagios = containers.Map(...
    {'Sleep stage W', 'Sleep stage 1', 'Sleep stage 2', 'Sleep stage 3', ...
    'Sleep stage 4', 'Sleep stage R'}, ...
    [1, 2, 3, 4, 5, 6]);

% Duração de cada janela
janela_seg = 30;
duracao_total = seconds(anotacoes.Onset(end)) + seconds(anotacoes.Duration(end));
num_janelas = floor(duracao_total / janela_seg);
```

Figura 2- Pré-processamento dos estágios

Nesta figura 2 prosseguimos para o pré-processamento dos estágios de sono. Para isso criamos um dicionário que associava os estágios de sono a números de 1 a 6. De seguida, definimos a duração de cada janela em 30 segundos pois era o tempo mencionado na base de dados.

Para calcular o número total de janelas, determinamos a duração total do registo em segundos. Para tal, somamos o início da última anotação com a sua duração. O valor obtido é então convertido em segundos para que seja calculado o número de janelas de 30 segundos usufruindo do floor para arredondar para baixo o número de janelas obtido.

```
% Vetor de estágios
estagios_cod = zeros(num_janelas, 1);

% Preencher o vetor de estágios janela a janela
for i = 1:height(anotacoes)
    nome_estagio = anotacoes.Annotations(i);

    if isKey(dic_estagios, nome_estagio)
        estagio_cod = dic_estagios(nome_estagio);
    else
        continue;
    end

    inicio = seconds(anotacoes.Onset(i));
    duracao = seconds(anotacoes.Duration(i));
    fim = inicio + duracao;

    % Determinar janelas que caem nesse intervalo
    idx_inicio = floor(inicio / janela_seg) + 1;
    idx_fim = floor(fim / janela_seg);

    % Proteger contra valores fora dos limites
    idx_inicio = max(1, idx_inicio);
    idx_fim = min(num_janelas, idx_fim);

    estagios_cod(idx_inicio:idx_fim) = estagio_cod;
end
```

Figura 3- Continuação do pré-processamento

Depois de realizado o cálculo do número de janelas, prosseguimos para a atribuição dos estágios de sono a cada uma das janelas como mostra a Figura 3, através de um ciclo for que percorre todas as anotações do hipnograma. Para isso, começamos por criar o vetor estagios_cod, inicialmente composto por zeros, que servirá para armazenar os valores codificados de cada estágio ao longo do tempo.

O ciclo for i = 1: height (anotacoes) é utilizado para iterar linha a linha sobre a tabela de anotações — ou seja, para cada anotação existente. Em cada iteração, o nome do estágio é extraído e verificado no dicionário criado anteriormente. Se existir correspondência, o nome é convertido para o número correspondente ao estágio; caso contrário, a anotação é ignorada e o ciclo avança.

De seguida é determinado o início e a duração de cada estágio, que convertidos em segundos dão o instante final de cada estágio.

No passo seguinte, determinamos quais as janelas que se encontram dentro do intervalo de tempo de cada anotação. Para isso, calculam-se o índice das primeira e última janelas (idx_inicio e idx_fim) que pertencem a esse intervalo. Para obter esses índices dividimos o início e o fim de cada estágio pela duração da janela adicionando 1 ao índice inicial de modo que os valores estejam de acordo com a indexação do Matlab. Para que o valor de idx_inicio não seja inferior a 1 e idx_fim não seja superior ao número total de janelas são aplicadas as funções max e min reduzindo a probabilidade de erro.

Por fim, todas as janelas pertencentes ao intervalo recebem o valor numérico correspondente ao estágio de sono atual, preenchendo o vetor estagios_cod com os valores de cada estágio ao longo do tempo.

```
%% 3. Filtro Passa-Banda (0.5 Hz - 45 Hz)

% Passar o formato para double
if iscell(eeg_sinal)
    eeg_sinal = cell2mat(eeg_sinal);
end
eeg_sinal = double(eeg_sinal);

% Parâmetros do filtro
ordem = 100;  % Reduz a ordem para verificar estabilidade
frequencias_norm = [0 0.5 45 50]/(fs/2);  % Normalização pela frequência de Nyquist
desejado = [0 0 1 0];  % Resposta desejada do filtro (passa-banda)

% Criar o filtro com firpm
filtro = firpm(ordem, frequencias_norm, desejado);
```

Figura 4- Filtragem do Sinal

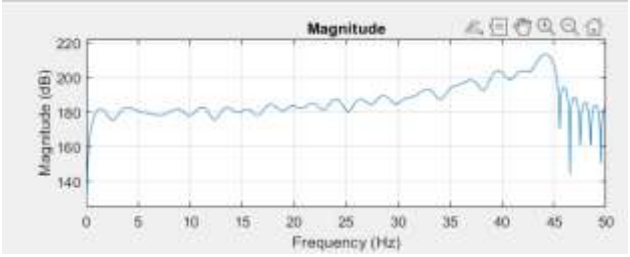


Figura 5- Resposta em frequência do filtro

Neste terceiro ponto mostrado na Figura 4 criamos um filtro “Band-Pass”.
Primeiro passamos o formato do EEG para double de modo a não ocorrer um erro na utilização. Após isso utilizamos um filtro firpm de ordem 100, de modo a controlar de forma precisa a resposta em frequência. O filtro foi criado de modo a reter as frequências entre 0.5 Hz e 45Hz, que correspondem às bandas relevantes do EEG e atenuar o ruido fora desse intervalo.[5]

Por fim utilizamos a frequência de Nyquist, que é metade de frequência de amostragem, e para garantir que os limites do filtro estejam corretamente definidos fizemos uma matriz que deixa passar as frequências que nós queremos observar como mostra a figura 5.[6]


Figura 6- Plotagem do Sinal EEG bruto vs EEG filtrado

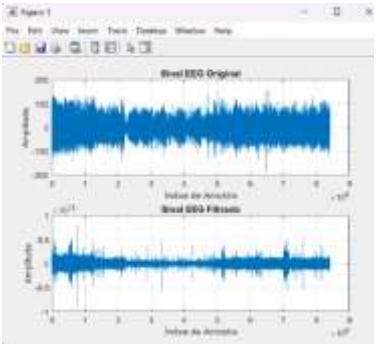


Figura 7- Sinal EEG bruto vs EEG filtrado

No ponto 4 mostrado na figura 6 prosseguimos para a filtragem do sinal através de uma convolução entre o filtro e o sinal EEG original resultando assim num sinal mais limpo e adequado para uma análise de outros parâmetros que serão mostrados mais a frente.

Esta Figura 7 mostra a atuação do filtro escolhido no sinal EEG original ruidoso enfatizando a necessidade que se tinha de filtrar o sinal para análise futura.

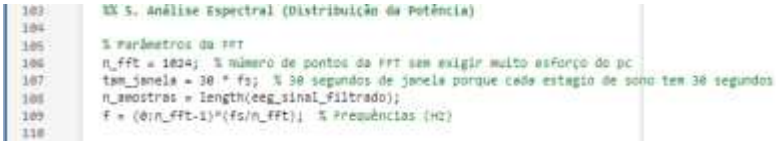


Figura 8- Parâmetros da análise espectral

Neste ponto mostrado na Figura 8 prosseguimos para o cálculo da análise espectral da potência, com o objetivo de compreender a distribuição de energia do sinal EEG ao logo das diferentes frequências. Esta análise é revelante pois permite a identificação de padrões característicos de cada estágio de sono, uma vez que cada estágio tende a concentrar a sua energia em bandas diferentes.

Inicialmente definimos os parâmetros da FFT, que permite transformar um sinal de domínio temporal para o domínio de frequência. O parâmetro n_fft define o número de pontos usados na FFT (o valor foi escolhido de modo a obtermos uma boa resolução com o menor esforço computacional).

De seguida definimos o número de amostras por janela multiplicando o tempo de cada janela pela frequência de amostragem (este valor indica quantas amostras correspondem a uma janela de 30 segundos).

Depois, definimos o número total de amostras no sinal EEG e criamos o vetor de frequências correspondente aos pontos da FFT. A expressão (0: n_fft-1) cria um vetor com os índices dos pontos da FFT. Já (fs/n_fft) calcula a resolução em frequência (quantos Hz estão representados por cada ponto da FFT). Ao multiplicar os índices do vetor pela resolução em frequência, obtemos o vetor f, que contém os valores de frequência associados a cada ponto da FFT.

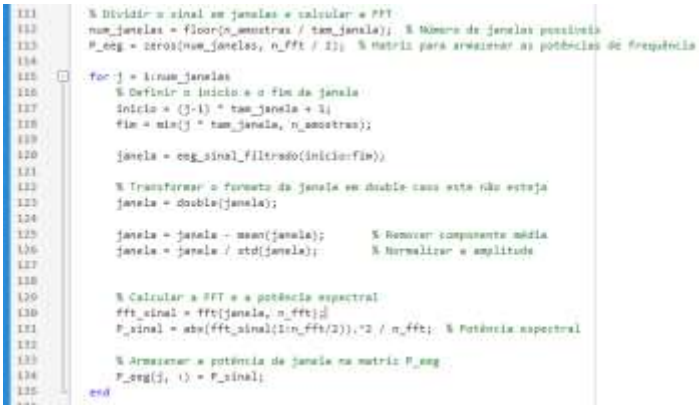


Figura 9- Cálculo da potência e da FFT

Nesta parte do código mostrada na figura 9 dividimos o sinal EEG em janelas de 30 segundos para fazer a análise espectral por janela. Esta abordagem é importante pois os sinais EEG são sinais estocásticos e não estacionários, ou seja, as suas características podem mudar ao longo do tempo e cada amostra é imprevisível.

Para isso, determinamos o número total de janelas, dividindo o número de amostras pela duração das janelas. Depois, criamos uma matriz de zeros com o nome P_eeg para armazenar a potência espectral de cada uma das janelas.

Através de um ciclo for, percorremos todas as janelas para cada iteração. Começamos por determinar o início e o fim de cada janela. Para isso utilizamos uma abordagem semelhante a usada no cálculo dos estágios adicionando 1 na janela inicial pois os índices dos vetores começam em 1 e utilizando o min para a última janela não exceder o tamanho do sinal EEG. Depois extraímos

o sinal correspondente, e após isso convertemos o formato da janela para double de modo a evitar que o programa dê erro no cálculo da FFT. Por fim antes do cálculo da FFT e da Potência Espectral removemos a média e normalizamos a amplitude.

A remoção da média foi feita com o objetivo de retirar as componentes de baixa frequência que estão normalmente associadas a derivações lentas do sinal, centrando assim, o sinal em torno de 0 de forma a melhorar a análise espectral. Já a normalização da amplitude, foi feita de modo a reduzir as variações e a facilitar comparação entre as janelas.[7]

De seguida efetuamos o cálculo da FFT ao sinal da janela e posteriormente o cálculo da potencia espectral elevando o valor absoluto da FFT ao quadrado. Para o cálculo da potência apenas os primeiros pontos foram considerados pois a FFT é simétrica.[8]

Por fim armazenamos os valores da potência na matriz P_eeg onde cada coluna contém a potência espectral numa janela de 30 segundos.

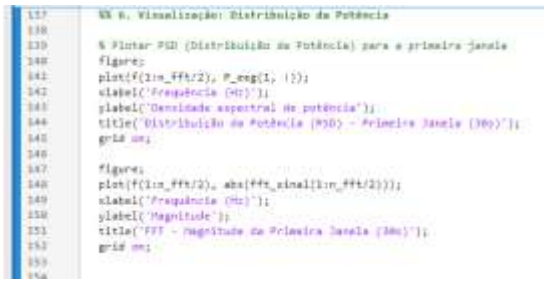

Figura 10- Plot da FFT vs PSD

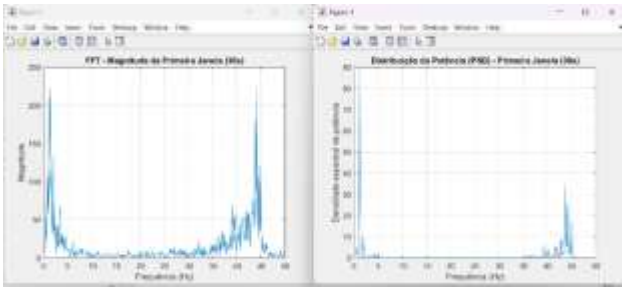

Figura 11- Grafico da FFT vs PSD

No ponto 6 mostrado na figura 10 prosseguimos para a plotagem da nossa PSD simplificada e da FFT com o intuito de observar as diferenças entre as duas e perceber o motivo da utilização da PSD em vez da FFT. Com os plots mostrados na figura 11 percebemos claramente que a FFT apresenta valores de magnitude elevados, podendo distorcer a interpretação da energia distribuída nas frequências. Já na PSD, a energia do sinal está mais bem distribuída e mais suavizada permitindo uma melhor identificação das bandas de frequência.

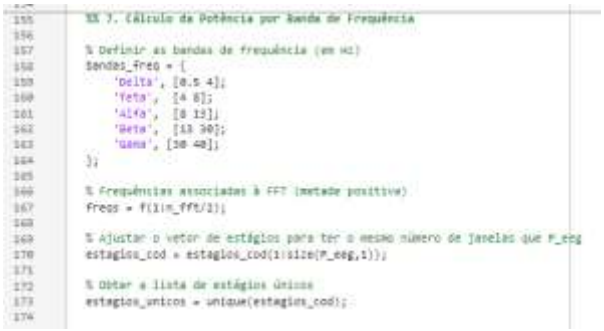

Figura 12- Parâmetros para o cálculo da Potência por banda

Neste ponto 7 mostrado na figura 12 calculámos os parâmetros necessários para calcular a potência por banda. Para isso, começamos por definir as 5 principais bandas do EEG utilizando uma estrutura cell (tabela) onde cada linha representa uma banda com o seu nome e respetivo intervalo de frequências, facilitando a iteração com elas. De seguida extraímos a parte positiva do vetor frequências (o método utilizado foi o mesmo que foi aplicado na potência).[5]

Depois ajustamos o vetor “estagios_cod” para este ter o mesmo tamanho que o vetor P_eeg e criamos um vetor com os estágios todos de 1 a 6. Sempre que esta ação sobre o vetor estágios_cod for executada independentemente do sinal considerado o tamanho do vetor irá diminuir devido à função floor que garante que apenas as janelas completas são consideradas.

Figura 13- Cálculo da Potência por banda e normalização

Após definir as bandas e ajustar o vetor de estágios prosseguimos para o cálculo da potência média por banda para cada estágio como mostrado na figura 13 de modo a observar a distribuição de cada banda pelos diferentes estágios. Assim, criamos uma matriz nula chamada “pot_por_banda” onde cada linha está associada a um estágio e cada coluna com a frequência.

Depois iniciamos dois ciclos “for” sendo o primeiro direcionado para os estágios e o segundo para as frequências.

No primeiro ciclo, identificamos todas as janelas pertencentes ao estágio de sono e depois é calculada a média da potência para cada frequência através do comando mean resultando num vetor. No segundo ciclo selecionamos os índices que pertencem a cada faixa (exemplo: Delta= [0.5;4]), e com o auxílio do comando find calculamos a potência daquela banda com o somatório dos valores da media_psd naquela faixa para o estágio que está a ser analisado. No final esse valor é armazenado na matriz pot_por_banda.

Por fim, normalizamos a matriz em percentagem para que a soma de cada banda em cada estágio some 100% (ex: delta em W+ N1 + N2 + N3 + REM = 100%). visualizando em que estágio é que a banda é mais intensa.

```
%% 8. Visualização da Potência Relativa por Banda e Estágio

% Gráfico de barras
figure;
bar(pot_por_banda, 'grouped');
xlabel('Estágio do Sono');
ylabel('Potência Relativa (%)');
title('Distribuição da Potência por Banda em Cada Estágio de Sono');
xticklabels({'W', 'N1', 'N2', 'N3', 'N4', 'REM'});
legend({'Delta', 'Teta', 'Alfa', 'Beta', 'Gama'});
grid on;
```

Figura 14- Plot do histograma da potência por banda

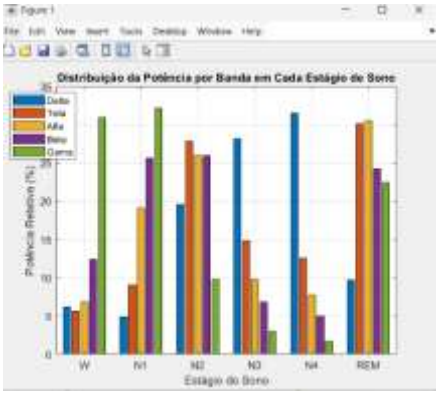


Figura 15- Histograma da potência por banda

No ponto 8 da figura 14 fizemos a plotagem de um histograma da potência média por banda em cada estágio de sono e prosseguimos para a análise.

Analisando cada estágio da figura 15:

No estágio W, verificamos uma maior atividade das bandas Beta e Gama e menor atividade das bandas Delta e Teta, o que é esperado pois as bandas de maior atividade estão associadas a estado de alerta e atenção com os olhos fechados.[9]

No estágio N1, verificamos um aumento da banda Alfa e diminuição da banda Delta, o que não era esperado. A banda alfa está mais associada a um estado de alerta e tende a diminuir com a progressão do sono. Assim, esse aumento pode indicar que N1, neste caso, representa um momento de transição.[9]

No estágio N2, a banda Teta aumentou o que é de esperar pois é nessa banda que Teta costuma ser predominante. Por outro lado, a banda Alfa aumentou e a banda beta manteve-se podendo indicar microdespertares.[9]

No estágio N3, as bandas de frequências elevadas (Gama e Beta) reduzem de forma drástica e Delta aumentou tal como esperado pois neste estágio o paciente encontra-se em sono profundo.[9]

No estágio N4, Delta atinge o pico e Alfa, Beta e Gama estão quase ausentes como é esperado no sono profundo.[9]

Por fim, no estágio REM observamos um aumento de Teta, Beta e Gama o que é esperado pois neste estágio a atividade cerebral é elevado pois esta está associada aos sonhos.[9]

```
%% 9. Potencia Média por Banda e Estágio

% Nomes das bandas
nomes_bandas= {'Delta', 'Teta', 'Alfa', 'Beta', 'Gama'};
num_bandas = length(nomes_bandas);

% Vetores para características e rótulos
caracteristicas = [];
rotulos = [];

for j = 1:num_janelas
    potencias_banda = zeros(1, num_bandas);

    for b = 1:num_bandas
        faixa = bandas_freq(b, :);
        indices_freq = freqs >= faixa(1) & freqs <= faixa(2);
        potencias_banda(b) = sum(P_eeg(:, indices_freq));
    end

    caracteristicas = [caracteristicas; potencias_banda];

    % Associar o estágio à janela j
    if j <= length(estagios_cod)
        rotulos = [rotulos; estagios_cod(j)];
    else
        rotulos = [rotulos; 0];  % caso não tenha rótulo
    end
end
```

Figura 16- Cálculo dos vetores rotulos e caracteristicas

No ponto 9 da figura 16 iniciamos o cálculo da Potência Média por banda e por estágio de modo a perceber as diferenças entre os valores das potencias nos diversos estágios de sono. Inicialmente, criámos uma lista com os nomes das bandas que analisámos, e com isso, obtivemos o número total de bandas (num_bandas).
Depois, criámos dois vetores vazios: um para armazenar as características do sinal (a potência por banda) e outro para os rótulos dos estágios correspondentes a cada janela.

De seguida, iniciámos um ciclo for que percorre todas as janelas. Em cada iteração, criamos um vetor com cinco posições uma para cada banda. A seguir, usamos um segundo for interno que percorre cada banda e vai buscar os limites de frequência dessa faixa, tal como foi feito no ponto 7. Depois, somamos os valores da PSD nos índices que pertencem a essa banda e guardamos esse valor no vetor potencias_banda.

Esse vetor é depois adicionado à matriz caracteristicas, ficando cada linha com os valores das cinco bandas para uma janela.

Por fim, criámos o vetor rotulos, onde associamos o estágio de sono correspondente a cada janela. Usámos um if para garantir que só adicionamos rótulos se o índice da janela não ultrapassar o tamanho do vetor estagios_cod. Isto é importante porque, devido a

arredondamentos, o número de janelas pode ser maior que o número de rótulos. Caso isso aconteça, as janelas a mais recebem o rótulo 0, indicando que não têm rótulo.

```
% Estágios ordenados
num_estagios = length(estagios_unicos);

% Inicializar matriz de médias
media_por_estagio = zeros(num_estagios, num_bandas);

% Calcular a média das potências por banda para cada estágio
for i = 1:num_estagios
    idx = rotulos == estagios_unicos(i);
    media_por_estagio(i, :) = mean(caracteristicas(idx, 1:num_bandas), 1);
end
```

Figura 17- Cálculo da potência média por banda

Depois, para finalizar o ponto 9 como mostra a figura 17 determinamos o número de estágios presentes e criamos uma matriz com zeros para guardar as potências médias por banda em cada estágio.

Por fim utilizamos um ciclo for que percorre cada estágio de sono. Em cada iteração identificamos as janelas que pertencem a esse estágio através de uma comparação rótulos==estágios_unicos(i) e guardamos no vetor idx de modo a filtrar as linhas da matriz características pertencentes a esse estágio. A seguir, fizemos o cálculo da média das potências por banda nessa janela.

```
%% 10 Gráfico da Potencia média

% Gráfico de barras com escala logarítmica
figure;
bar(nomes_bandas, media_por_estagio');
xlabel('Bandas de Frequência');
ylabel('Potência Média');
title('Potência Média por Banda para Cada Estágio do Sono');
set(gca, 'YScale', 'log');  % Escala logarítmica para melhor visualização
legend({'Sleep stage W', 'Sleep stage 1', 'Sleep stage 2', ...
        'Sleep stage 3', 'Sleep stage 4', 'Sleep stage R'}, 'Location', 'northwest');
grid on;
```

Figura 18- Plot do gráfico da potência média por estágio

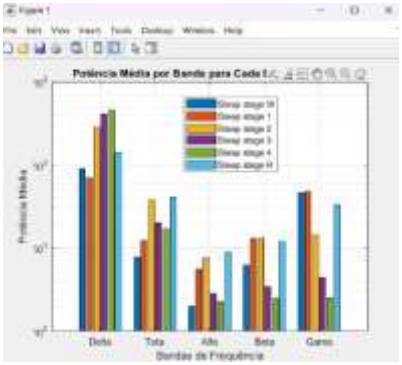


Figura 19- Histograma da potência média por estágio

No ponto 10 mostrada na figura 18 prosseguimos para o plot do histograma da potência média de cada banda para cada estágio de sono. Para uma melhor visualização utilizamos o comando set para transformar a nossa escala Y numa escala logarítmica (gca- get current axis, Yscale- escala do eixo y, log- escala logarítmica) de modo a visualizar melhor as bandas de menor frequência obtendo o seguinte histograma.

Este histograma da figura 19 além de corroborar o primeiro histograma mostrado na figura 15 ajuda-nos a perceber quais as bandas que possuem os valores mais elevados de potência média.

Com este histograma verificamos que a banda Delta tem os valores absolutos mais elevados especialmente nos estágios N3 e N4 como já tínhamos verificado anteriormente.

Já as bandas Teta, Alfa e Beta apresentam um comportamento semelhante apresentando um valor absoluto elevado nos primeiros estágios (W, N1, N2), uma redução aquando da chegada dos estágios de sono mais profundos (N3, N4) e uma nova subida em REM.

Finalmente Gama tem os seus valores mais elevados no estágio W e N1, nota se uma redução em N2 até N4 e em REM volta a ter um aumento devido a maior atividade cognitiva neste estágio.

Estes valores absolutos são importantes para a parte seguinte do nosso trabalho que é identificar o estágio de sono através de um jogo sabendo as potencias médias e comparando-as.

```
%% 11. Extração de Características para Classificação do Jogo

% Estágio aleatório entre os disponíveis
estagio_escolhido = estagios_unicos(randi(length(estagios_unicos)));

% Janelas com esse estágio
idx_estagio = find(rotulos == estagio_escolhido);

% Janela aleatória entre as que têm esse estágio
idx_central = idx_estagio(randi(length(idx_estagio)));

% Procurar para trás
inicio = idx_central;
while inicio > 1 && rotulos(inicio - 1) == estagio_escolhido
    inicio = inicio - 1;
end

% Procurar para frente
fim = idx_central;
while fim < length(rotulos) && rotulos(fim + 1) == estagio_escolhido
    fim = fim + 1;
end
```

Figura 20- Escolha do estágio aleatório

No ponto 11 mostrado na figura 20 começamos a criação do nosso jogo. Para isso começamos por escolher um estágio aleatório dentro dos 6 disponíveis com o auxílio do comando randi. Depois são extraídas todas as janelas que contêm esse estágio e uma delas é escolhida aleatoriamente. A seguir, é procurado o início da janela e o fim da mesma através de dois ciclos while. Isto é feito para conseguimos mostrar ao utilizador um estágio completo para ele analisar sem o conhecimento prévio desse estágio.

Figura 21- Extração dos índices das janelas e cálculo da potência.

Após isso, a figura 21 mostra a extração das janelas desse índice e indicando-as no display assim como o número de janelas.

Depois calculamos a potência média por banda nesse bloco da mesma forma que a calculamos no ponto 9. Por fim, é mostrado no display a potência média em cada banda do bloco escolhido e posteriormente é feito um gráfico de barras para obter uma melhor comparação.

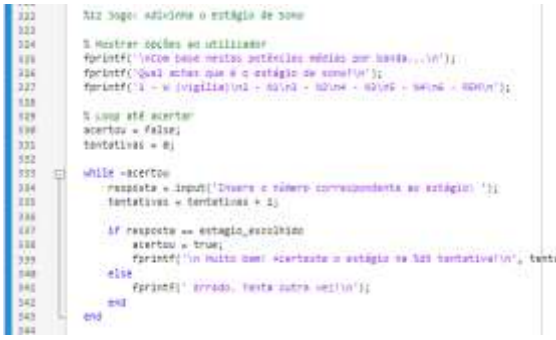

Figura 22- Loop do jogo e contador

Neste ponto 12 mostrado na figura 22 prosseguimos para o jogo propriamente dito. Após a visualização das potencias de forma absoluta de duas maneiras distintas são dadas as opções de escolha e os números relativos a resposta. De seguida através de um ciclo while calculamos o número de tentativas e verificamos se o utilizador acertou ou errou a resposta que introduziu no input. Caso a resposta esteja certa aparece no display “Muito bem!...” e o referente número de tentativas. Caso esteja errado o loop continua até o utilizador acertar.

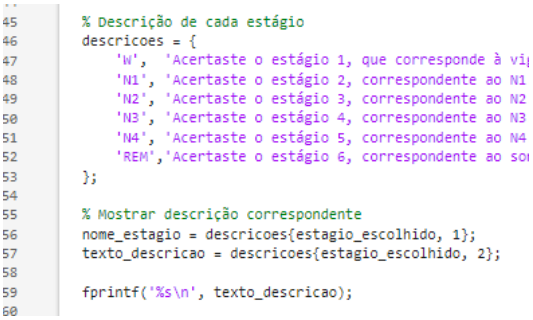

Figura 23- Descrição dos estágios

No fim, adicionamos uma pequena descrição como mostra a figura 23 relativa a cada estágio para o utilizador entender o provável porquê de ser aquele estágio:

W- Acertaste o estágio 1, que corresponde à vigília (W). Neste estado, o EEG mostra atividade de alta frequência e baixa amplitude, principalmente nas bandas alfa e beta. O cérebro está alerta e os olhos podem estar abertos ou a piscar.

N1- Acertaste o estágio 2, correspondente ao N1. Este é o primeiro estágio do sono leve. Caracteriza-se por uma redução da atividade alfa e aumento na banda teta. O EEG mostra transições suaves e o corpo começa a relaxar.

N2- Acertaste o estágio 3, correspondente ao N2. Este estágio é marcado por atividade predominante na banda teta e pela presença de fusos do sono, sinais típicos neste estágio intermediário do sono leve.

N3- Acertaste o estágio 4, correspondente ao N3. É um estágio de sono profundo, com predominância de ondas lentas (delta). O EEG mostra alta amplitude e baixa frequência. Essencial para a recuperação física e consolidação da memória.';

N4- Acertaste o estágio 5, correspondente ao N4. É um estágio mais profundo que N3, com uma predominância ainda maior da banda delta. Muitas vezes é agrupado com o N3 nas classificações atuais.

REM- Acertaste o estágio 6, correspondente ao sono REM. Apesar de ser um estágio de sono profundo, o EEG apresenta padrões semelhantes à vigília, com atividade mista nas bandas beta e teta. É o estágio em que ocorrem os sonhos mais intensos.

Como os valores utilizados são dados reais nem sempre a identificação dos estágios é linear, mas é sempre possível identificar os estágios mais prováveis.

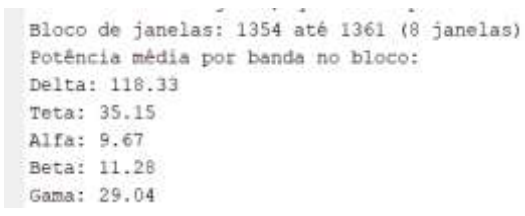


Figura 24- Potência média por banda

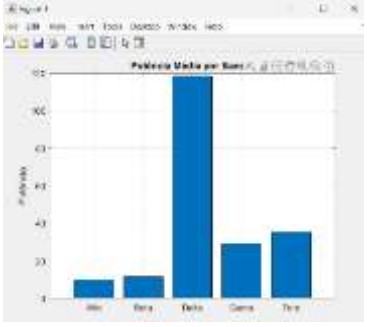

Figura 25- Gráfico de barras da potência média por banda

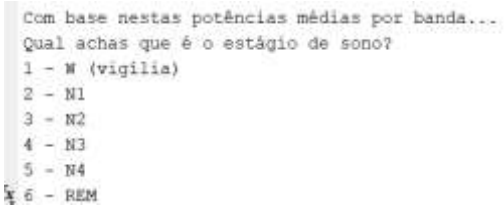


Figura 26- Opções de escolha de estágio

Ao analisar os dados mostrados nas Figuras 24 e 25, retiramos que o estágio apresentado é um estágio que possui uma atividade cerebral elevada. Assim eliminamos as opções N2, N3, e N4 sobrando apenas W, N1 e REM. Para demonstrar a eficiência do nosso programa vou escolher o estágio menos provável dos 3 (N1) pois como é uma janela avançada e todas as bandas apresentam alguma atividade a probabilidade de o estágio ser N1 é reduzida. O estágio mais provável consoante a nossa análise é o estágio REM que corresponde ao número 6 segundo a figura 26 pois apresenta uma elevada atividade cerebral, logo seria expectável que este fosse o estágio certo.

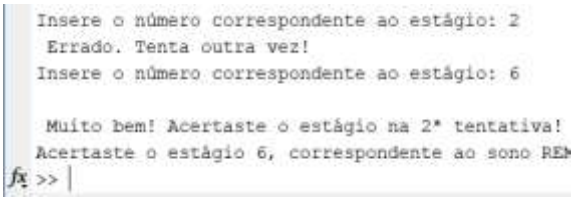


Figura 27- Print das respostas erradas e certas mostradas no display depois do teste do utilizador

Como esperado o estágio certo era o estágio REM (comprovação do estágio com a figura 27). Apesar de a nossa análise ter sido correta nem sempre isso acontece, por esse motivo é que é importante fazer a análise teórica e depois corroborar na prática, pois nem todas as pessoas apresentam os mesmos estágios de sono com as mesmas bandas de frequência.

## 3. Conclusão

Em suma, acreditamos que o projeto realizado satisfaz o que era esperado do tema escolhido, e também as espectativas que colocámos no mesmo. Conseguimos analisar bem os sinais EEG, começando por criar e aplicar um filtro sobre o sinal bruto e obtendo um sinal muito mais adequado para posteriormente obtermos a distribuição da potência ao longo do sinal e, consequentemente, conseguimos observar claramente as diferenças e características únicas em cada estágio de sono, assim como estudar relações entre os mesmos. Também consideramos que tivemos uma abordagem inovadora e criativa, tendo criado um pequeno jogo didático e interativo para com o utilizador que demostrou resultar com imensa fidelidade.

As principais dificuldades que tivemos foram na leitura e tratamento inicial dos ficheiros do EEG retirados da base de dados disponível e também no cálculo das potências médias por banda e estágio.

## 4. Referências

[1] "How Sleep Works - Why Is Sleep Important? | NHLBI, NIH." Accessed: May 30, 2025. [Online]. Available: https://www.nhlbi.nih.gov/health/sleep/why-sleep-important

[2] A. K. Patel, V. Reddy, K. R. Shumway, and J. F. Araujo, "Physiology, Sleep Stages," *StatPearls*, Jan. 2024, Accessed: May 30, 2025. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK526132/

[3] L. Duan *et al.*, "A Novel Sleep Staging Network Based on Data Adaptation and Multimodal Fusion," *Front Hum Neurosci*, vol. 15, p. 727139, Oct. 2021, doi: 10.3389/FNHUM.2021.727139/BIBTEX.

[4] S. V Vaseghi, "Stochastic Processes," in *Advanced Signal Processing and Digital Noise Reduction*, S. V Vaseghi, Ed., Wiesbaden: Vieweg+Teubner Verlag, 1996, pp. 23–64. doi: 10.1007/978-3-322-92773-6_2.

[5] C. S. Nayak and A. C. Anilkumar, "EEG Normal Waveforms," *StatPearls*, pp. 1–6, Jul. 2023, Accessed: May 29, 2025. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK539805/

[6] M. C. Wiemers, H. Laufs, and F. von Wegner, "Frequency Analysis of EEG Microstate Sequences in Wakefulness and NREM Sleep," *Brain Topogr*, vol. 37, no. 2, pp. 312–328, Mar. 2024, doi: 10.1007/s10548-023-00971-y.

[7] S. Hu, J. Ruan, P. A. Valdes-Sosa, and Z. Lv, "How do the resting EEG preprocessing states affect the outcomes of postprocessing?," *Neuroimage*, vol. 310, p. 121122, Apr. 2025, doi: 10.1016/j.neuroimage.2025.121122.

[8] MathWorks, "Power Spectral Density Estimates Using FFT." Accessed: May 05, 2025. [Online]. Available: https://www.mathworks.com/help/signal/ug/power-spectral-density-estimates-using-fft.html

[9] C. A. Espie, N. M. Broomfield, K. M. A. MacMahon, L. M. Macphee, and L. M. Taylor, "The attention–intention–effort pathway in the development of psychophysiologic insomnia: A theoretical review," *Sleep Med Rev*, vol. 10, no. 4, pp. 215–245, Aug. 2006, doi: 10.1016/J.SMRV.2006.03.002.

[10] H. Jalali, M. Pouladian, A. M. Nasrabadi, and A. Movahed, "Sleep stages classification based on feature extraction from music of brain," *Heliyon*, vol. 11, no. 1, p. e41147, Jan. 2025, doi: 10.1016/J.HELIYON.2024.E41147.

## 5. Apêndice

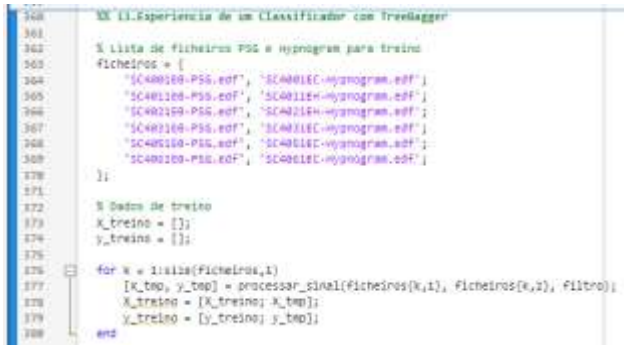

Figura 28- Extração e processamento dos ficheiros de teste

Nesta parte final do nosso miniprojecto, recorremos a uma inteligência artificial para a criação de um classificador capaz de prever o estágio de sono de EEG's reais.

Para isso recorremos a utilização do algoritmo TreeBagger, devido à sua robustez e eficácia na classificação. Inicialmente experimentamos o KNN mas estava a dar uma precisão baixa então com alguma pesquisa e ajuda de IA decidimos mudar para o TreeBagger.

Inicialmente definimos os ficheiros mostrados na figura 28 com sinais EEG semelhantes para treinarmos o classificador. Em seguida, através de um ciclo for aplicamos a função processar sinal a cada par de ficheiros extraindo as características espectrais que nós achamos mais relevantes considerar por janela e o respetivo estágio.

Figura 29- Inicio da função processar sinal upload e tratamento dos dados

Agora vamos mostrar detalhadamente a função processar estágios da figura 29-33. Esta função tem como objetivo preparar os dados de treino do classificador. Inicialmente observa se na figura 29 o mesmo processo já mostrado no código principal definindo a frequência de amostragem, a duração da janela e o seu respetivo tamanho. Depois fazemos a leitura dos ficheiros e escolhemos o canal referente ao EEG. A seguir garantimos que o EEG tem formato double de forma a minimizar os erros.

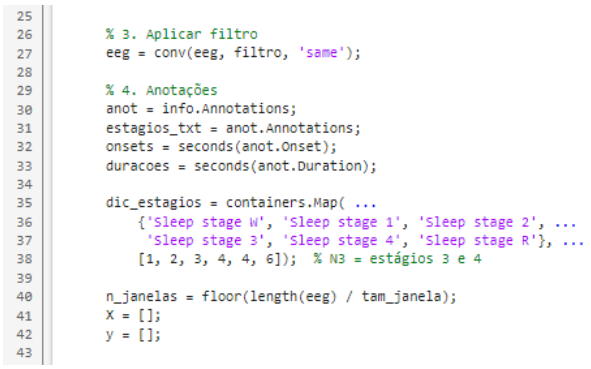

Figura 30- Aplicação do filtro e extração dos estágios

A seguir aplicamos o filtro como mostrado na figura 30 antes de fazer qualquer iteração de modo a remover o ruido presente no sinal EEG bruto e usamos o comando same para manter o tamanho original.

No ponto 4 mostrado na figura 30 extraímos as mesmas anotações que extraímos no código principal para saber os estágios de sono (os estágios 3 e 4 foram juntos pois o classificador tinha bastante dificuldade na identificação dos mesmos), os instantes de início (Onset) e as suas durações (Duration) e criamos um dicionário também como foi feito no código principal. Depois calculamos o número total de janelas e criamos o vetor x (vai armazenar os vetores características) e o vetor y (vai armazenar os rótulos dos estágios correspondentes).

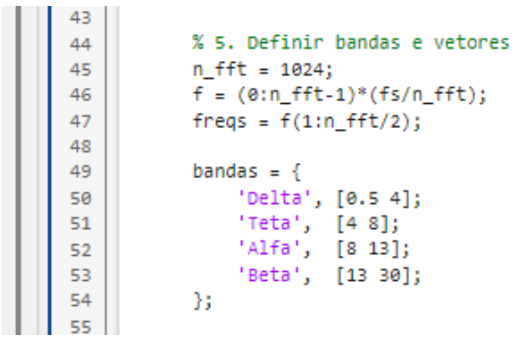

Figura 31- Definição das bandas e vetores

No ponto 5 mostrado na figura 31, utilizamos os mesmos parâmetros que já tínhamos definido no código principal, nomeadamente o n_fft, o vetor de frequências f e a versão reduzida freqs, que considera apenas a metade positiva. De seguida, definimos as principais bandas de frequência do EEG, associando cada uma ao seu respetivo intervalo em hertz.


Figura 32- Cálculo da FFT e PSD

No ponto 6 da figura 32 percorremos todas as janelas do EEG e extraímos as características mais importantes.

Para cada janela j extraímos o início e o fim da janela correspondente e guardamos no vetor janela.

Depois, calculamos o tempo da janela atual e verificamos se existe um estágio de sono correspondente e utilizamos o dic_estagios para converter o nome em um número correspondente.

Antes da análise, foi feita uma normalização (esta normalização foi realizada por uma IA) para ter média zero e desvio padrão unitário. Isto ajuda a reduzir a variabilidade entre janelas e melhorar a robustez do classificador. Depois calculamos a FFT e a PSD da mesma maneira que fizemos no código original.

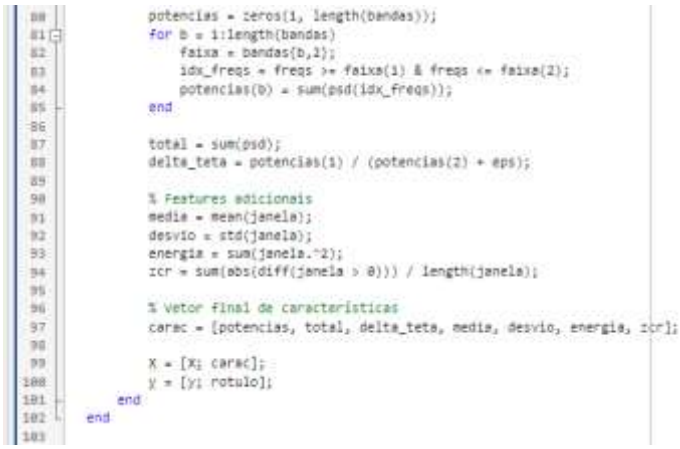


Figura 33- Cálculo e armazenamento dos features

De seguida na figura 33, calcula-se a potência para cada banda no intervalo de frequências correspondente.

Além das potencias por banda e da potencia total do sinal, foi calculado também a relação delta/teta, a média, o desvio padrão, a energia do sinal, e a frequência com que um sinal cruza o zero (o cálculo deste último feature foi explorado por nós e executado com ajuda de IA) e armazenamos esse valor no vetor características. Por fim guardamos as características no vetor x e os estágios das janelas correspondentes no vetor y.


Figura 34- Treino do classificador TreeBagger

Agora que a função processar sinal foi explicada prosseguimos para a continuação e análise do ponto 13 na figura 34. Para os valores terem a mesma escala é aplicada uma normalização, e de seguida é feito um PCA com o auxílio de IA, para reduzir a dimensionalidade do conjunto de dados reduzindo o ruido, diminuindo o tempo de treino e evitando overfitting. Por fim, selecionamos as 5 componentes principais, e com auxílio de IA foi treinado um classificador TreeBagger de modo a tentar prever o estágio de sono com base em algumas características do EEG.

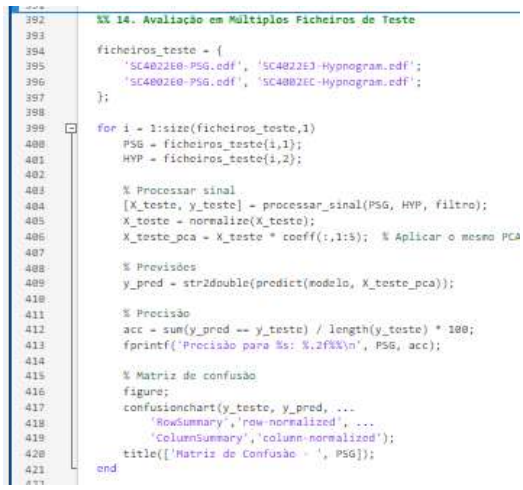


Figura 35- Escolha e processamento dos ficheiros de teste

No ponto 14 mostrado na figura 35 escolhemos dois ficheiros para teste.

Estes ficheiros foram processados da mesma maneira que os ficheiros de treino. Em seguida comparamos as previsões y_pred com os rótulos já sabidos de y_teste e calculamos a precisão de acertos.

Para uma melhor visualização do desempenho do classificador, foi criada uma matriz confusão, com a normalização por linhas e colunas, onde observamos onde o classificador mais errou e acertou.

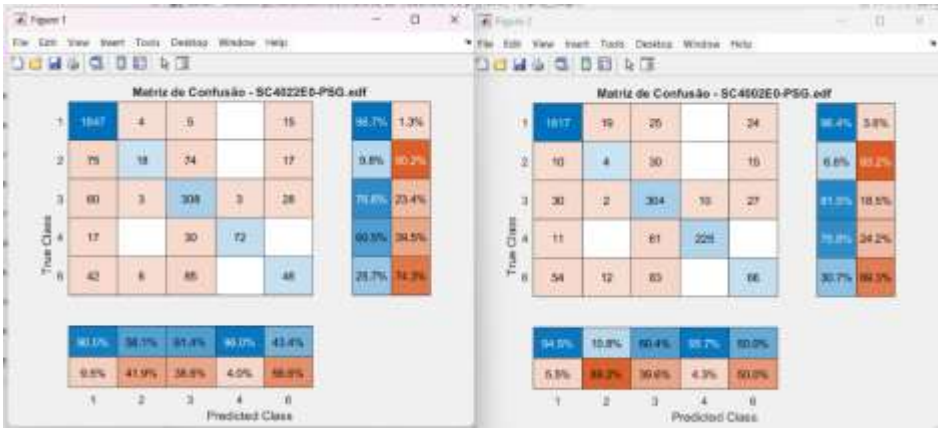


Figura 36- Matrizes confusão

```
Precisão para SC4022E0-PSG.edf: 83.16%
Precisão para SC4002E0-PSG.edf: 85.40%
```

Figura 37- Precisão obtida

Com base nestas duas figuras podemos concluir que a nossa experiência de classificação de estágios de sono foi bem-sucedida e apresentou resultados satisfatórios para os ficheiros de teste em questão. As matrizes confusão apresentaram dificuldade em identificar o estágio N1 e REM o que faz sentido de acordo com a literatura, em contrapartida os estágios W, N2 e N3 em alguns casos apresentam uma precisão de até 90%.[10]