



**Departamento de Física**

Aplicações Avançadas de Instrumentação Biomédica | Docentes: Hugo  
Gamboa e Pedro Vieira

# **Relatório – Projeto Final**

*Postural Change Detector*

56759 | João Fragoso | P3 | MIEB

52941 | Mariana Pinto | P3 | MIEB

23 de dezembro, 2022

## Índice

1. Introdução .....	1
2. Aquisição de dados.....	1
3. Preparação dos dados e extração de <i>features</i> .....	2
4. Classificadores .....	3
5. Servidor <i>Web</i> [Anexo 1] .....	4
6. Conclusão .....	4
Anexo 1.....	6

## 1. Introdução

O presente projeto foi realizado no âmbito da unidade curricular de Aplicações Avançadas de Instrumentação Biomédica, e teve como objetivo o desenvolvimento de um classificador, baseado na extração de características de sinais escolhidos pelo grupo, de forma a ser possível prever o resultado da classificação de dados que não foram anteriormente apresentados ao classificador, ou seja, que não se encontravam no conjunto de treino.

A monitorização à distância é um tema com especial interesse, principalmente nos últimos anos, no campo da Engenharia Biomédica, dado permitir uma monitorização contínua, sem existir necessidade da presença física constante de um profissional de saúde. Neste sentido, e tendo em mente a possibilidade de monitorização de movimentos realizados por diversos pacientes/indivíduos, o grupo optou pelo desenvolvimento de um classificador de movimentos, com o objetivo de identificar alterações de posição na posição sentada. As classes são, então, as seguintes: movimento lateral direito e esquerdo do tronco, extensão e flexão do tronco, e estado de repouso, com as costas direitas - que constitui a classe de rejeição.

Para tal, foi, primeiramente, realizada a aquisição de dados, de forma a criar uma base de dados com os vários movimentos em estudo; depois, fez-se a extração de *features*; estudou-se qual o melhor classificador a utilizar neste caso, e selecionou-se e programou-se o mesmo; treinou-se o classificador; e, por fim, realizou-se a classificação de uma aquisição em tempo real, nunca apresentada ao classificador, funcionando como dados de teste. Todas estas etapas são explanadas ao longo do presente relatório.

Para além disso, foi construído um servidor *Web*, adaptado a qualquer ambiente, incluindo o ambiente móvel, que permite a apresentação dos gráficos relativos aos dados adquiridos em tempo real, assim como a apresentação do resultado da classificação.

## 2. Aquisição de dados

A aquisição de dados foi feita com os sensores inerciais do telemóvel através da aplicação *SensorStreamer*, disponível para dispositivos *Android*, utilizando a configuração visível na figura 1. Este método de fixação do dispositivo foi pensado, de forma a uniformizar as recolhas e eliminar a maioria do ruído do sinal e possíveis variações. Foram escolhidos os sensores acelerómetro e giroscópio, com período de aquisição de 60 ms, fazendo cada aquisição durante 5 s. A comunicação com o computador foi feita através de *websockets*.



*Figura 1: Colocação do telemóvel, em posição horizontal e com o ecrã voltado para a frente, na zona torácica, no interior de uma bolsa, para aquisição de dados.*

Foi programado em *Python* um algoritmo de recolha, que permite a comunicação com o dispositivo móvel, recolhendo os dados dos sensores, gravando-os. Este inicia-se, solicitando ao utilizador que indique o tipo de movimento que irá efetuar, o seu número de identificação (1 = João; 2 = Mariana) e o número da aquisição. Estas informações são guardadas no nome do ficheiro CSV a ser criado. De seguida, pede para inicializar a recolha. Assim que a confirmação é dada, é criado o *socket* entre o computador e o telemóvel e os dados são recolhidos e decodificados. Num ciclo que percorre todas as *samples*, é criada uma *string* com a informação do tempo e dos dados do acelerómetro e giroscópio nos três eixos cartesianos, e acrescentada, uma por linha, ao ficheiro CSV. Completado o ciclo, a ligação pelo *socket* termina.

No total, foram recolhidas 101 aquisições com distribuição uniforme entre as 5 classes, de forma a permitir algumas variações devido exclusivamente a variantes intrínsecas ao movimento, e entre os indivíduos 1 e 2, com o objetivo de o classificador não se tornar dependente de parâmetros dependentes do executor dos movimentos, encontrando apenas semelhanças entre os movimentos em si.

Os sinais recolhidos em tempo real para posterior classificação são obtidos utilizando a mesma metodologia, de forma a permitir uma classificação precisa.

### 3. Preparação dos dados e extração de *features*

A extração de características, ou *features*, é um processo bastante importante no decorrer do treino dos vários algoritmos de classificação, uma vez que é a partir destas que os classificadores distinguem as várias classes. Deste modo, é importante que as características escolhidas sejam representativas de um dado movimento, de forma a haver a máxima separação na identificação de cada classe, para que o classificador possua o máximo de precisão na avaliação do movimento realizado.

Para uma melhor precisão na extração, foi aplicado um filtro *smoothing* aos dados adquiridos, através do método *savgol\_filter* da biblioteca *Spicy*, com o parâmetro do comprimento da janela do filtro de 9, e ordem do polinómio utilizado para ajustar as amostras igual a 3. É possível visualizar um exemplo da aplicação deste filtro na figura 2.

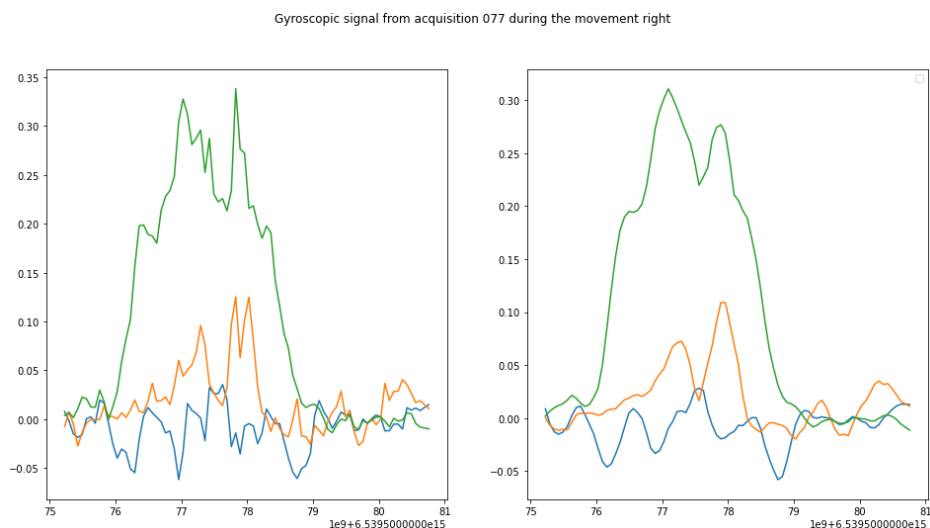


Figura 2 - (a) Gráfico referente aos dados do giroscópio de uma aquisição do movimento lateral direito. (b) O mesmo gráfico, após aplicado o filtro *smoothing*.

Para efetuar o treino e classificação de séries temporais é, então, necessário fazer extração de *features* dos sinais. Para tal, recorremos à biblioteca *TSFEL* (*time series feature*

*extraction*), que faz a extração de 106 *features* de *time series*. Considerando que foram obtidos dados em 3 planos para os 2 sensores, no total, obtivemos 1057 *features*, contando também com a *feature* que identifica o indivíduo que faz o movimento.

A *target variable* extraída estava em formato *string*, para garantir bons resultados na classificação. Considerando que alguns classificadores não funcionam com *strings*, foi aplicado o método *Label Encoder* da biblioteca *sklearn*, que ordena as classes por ordem alfabética das classes originais e as redefine de 0 a n.

## 4. Classificadores

Com o objetivo de classificar os movimentos, foram utilizados vários algoritmos de *Machine Learning* de aprendizagem supervisionada, uma vez que a gravação dos dados de treino possui o movimento correspondente.

Para selecionar as melhores *features*, ou seja, as características mais eficazes na classificação, e fazer uma escolha preliminar dos melhores classificadores, recorremos à plataforma *Orange*. Com o método *rank*, selecionamos as dez melhores *features*, que são, nomeadamente: *valor médio*, *histograma*, *mediana*, *desvio padrão* e *variância* para o sinal obtido pelo acelerómetro no eixo do y; *declive* para o sinal obtido pelo acelerómetro no eixo do z; *valor médio* e *histograma* para o sinal obtido pelo giroscópio no eixo do z; e *percentil* e *distância entre quartis* para o sinal obtido pelo giroscópio no eixo do y.

Com o método *Test and Score*, testou-se sete classificadores diferentes, com recurso ao método de *cross-validation leave one out*. Foram analisadas as matrizes de confusão obtidas e as curvas *ROC* e identificou-se o *Support Vector Machine*, *Random Forest* e *Gaussian Naive Bayes* como os melhores classificadores.

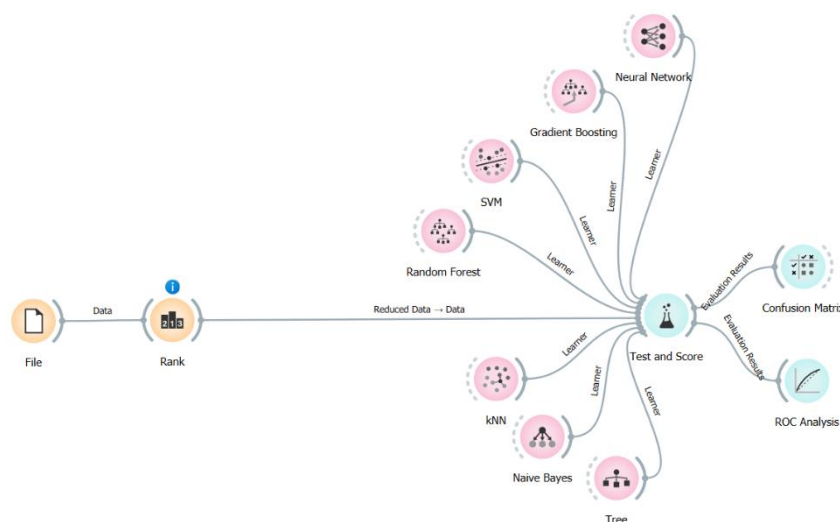


Figura 3 - Esquema do programa em Orange.

Testamos o desempenho destes classificadores na plataforma *Jupyter*, com recurso à biblioteca *sklearn*. Os dados foram separados em conjunto de treino e de teste com distribuição 70-30 através do método *train\_test\_split*. Para garantir a correta representação das classes, uma vez que apenas se tem cerca de 20 amostras por classe, definimos o parâmetro *stratify*, para que a percentagem de representação da *label* fosse igual no conjunto de teste e no conjunto de treino.

Foram, então, treinados os 3 modelos com *default parameters*. Ao efetuar a avaliação, foram obtidos ótimos resultados para o classificador *Random Forest* e para a *Gaussian Naive*

*Bayes*, com *accuracy* acima dos 93 % no conjunto de teste. Com estes resultados, não consideramos necessário efetuar otimização de parâmetros com métodos como *GridSearch*. Como o *Gaussian Naive Bayes* e *Random Forest* são ambos métodos menos complexos que *Support Vector Machine*, são preferíveis, garantido um melhor desempenho - daí não tentarmos otimização. Entre *Gaussian Naive Bayes* e *Random Forest*, aplicamos a mesma lógica: sendo o método gaussiano computacionalmente mais rápido, escolhemo-lo como o nosso classificador.

Por fim, treinamos o modelo com todos os dados e, através da biblioteca *joblib*, exportamos o modelo criado.

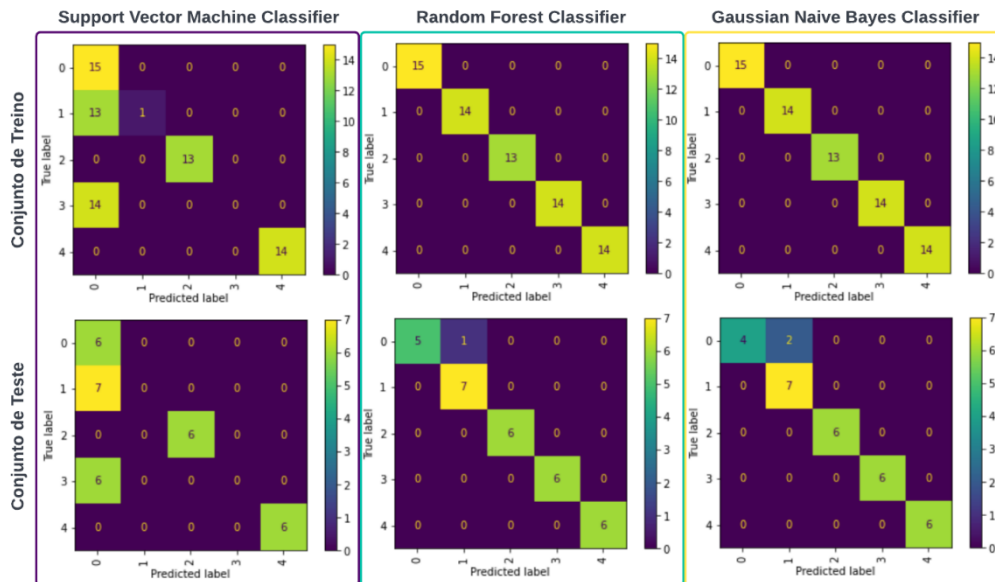


Figura 4 - Matrizes de confusão do conjunto de treino e do conjunto de teste, utilizando os classificadores Support Vector Machine, Random Forest e Gaussian Naive Bayes.

## 5. Servidor Web [Anexo 1]

O servidor *web* foi desenvolvido em *html* e *javascript*, com recurso a *templates css* disponíveis *online* para melhorar a estética do *website*. As funcionalidades desenvolvidas incluem gráficos em tempo real e a classificação do movimento, utilizando o modelo gerado. A recolha e classificação é feita pelo programa em *python* e a comunicação é estabelecida por *websockets*.

O utilizador começa por se conectar com o botão “Conectar!”; é criado o *socket* e o valor do botão muda, então, para “Começar!”. Ao clicar novamente, o programa inicia a aquisição e envia continuamente os dados das séries temporais, que são representadas graficamente, em tempo real, na plataforma, com recurso à biblioteca *plotly*.

Os dados são, então, utilizados para classificação: o programa calcula as dez *features* individualmente, importa o modelo treinado e efetua a classificação, que é, posteriormente, indicada no *website*.

## 6. Conclusão

Como forma de conclusão, pode-se referir que o grupo considerou este projeto bastante desafiante e interessante, permitindo um desenvolvimento de competências, não só de programação e *machine learning*, como de aquisição e processamento de dados - proficiências

essenciais na área da Engenharia Biomédica. Foi possível adquirir novos conhecimentos no que toca à comunicação de dados através de *WebSockets*, e a sua exposição de forma clara numa página *web*. Dada a possível aplicação real num contexto clínico ou hospital, reitera-se a importância deste trabalho na nossa formação, enquanto futuros engenheiros biomédicos.

Concluimos ainda que existe uma relativa facilidade, ao utilizar a linguagem *Python*, em extrair e utilizar os vários modelos dos classificadores, e ainda uma enorme variedade de bibliotecas com algoritmos já implementados relevantes no processamento de sinal (*Numpy*, *Scipy*), as quais se demonstraram de elevada relevância durante a execução do projeto.

Em termos de recomendações para trabalhos futuros, sugere-se a utilização de mais classes referentes a mais tipos de movimentos, dado que, numa posição sentada, ainda pode ser considerado relevante a deteção de movimentos de rotação ou de mudança de posição sentada para em pé, por exemplo. Esta pode ser, portanto, considerada uma limitação do nosso sistema: serem utilizadas apenas 5 classes.

## Anexo 1

# Postural Change Detector

João Fragoso 56759

Mariana Pinto 52941

P3 | MIEB

Docentes:

Hugo Gamboa

Pedro Vieira

AAIB 2022/23

## Contexto

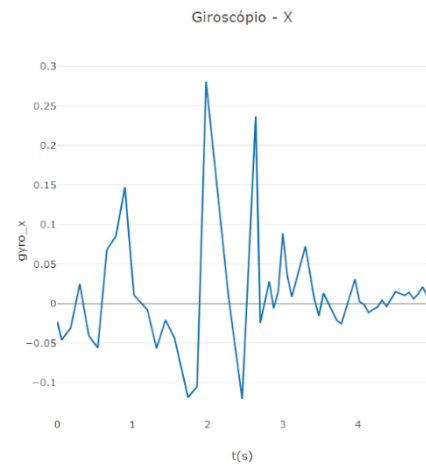
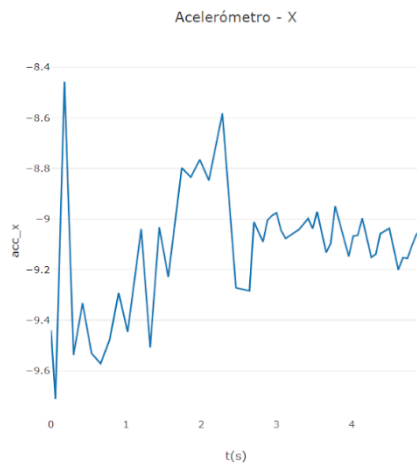
O presente projeto foi realizado no âmbito da unidade curricular de Aplicações Avançadas de Instrumentação Biomédica, e teve como objetivo o desenvolvimento de um classificador, baseado na extração de características de sinais escolhidos pelo grupo.

A monitorização à distância é um tema com especial interesse, principalmente nos últimos anos, no campo da Engenharia Biomédica. Neste sentido, e tendo em mente a possibilidade de monitorização de movimentos realizados por diversos indivíduos,


o grupo optou pelo desenvolvimento de um classificador de movimentos, com o objetivo de identificar a mudança de posições, na posição sentada. As classes são as seguintes: movimento lateral direito e esquerdo, extensão e flexão, e parado.


CONECTAR!







## Resultado

1  
Right-side bending  


2  
Left-side bending  


3  
Extension movement  


4  
Compression movement  


5  
Standing still  
