Universidade da Beira Interior
Departamento de Informática

Nº 2 - 2022: *[Machine Learning]*

Elaborado por:

45842 Diogo Manuel Paiva dos Santos

Orientador:

Professor Doutor Hugo Pedro Proença

9 de janeiro de 2023

Agradecimentos

A conclusão deste trabalho, não seria possível sem a ajuda do Professor Doutor Hugo Pedro Proença pois não teríamos o conhecimento adquirido nas aulas para a resolução do mesmo.

Conteúdo

Lista de Figuras

Lista de Excertos de Código

Acrónimos

AUC	*Area Under the Curve*
IA	Inteligência Artificial
ML	*Machine Learning*
MLP	*Multi Layer Perceptron*
ROC	*Receiver Operating Characteristic*

Capítulo

1

Introdução

1.1 Enquadramento

Este documento funciona como relatório do segundo trabalho prático da unidade curricular de Inteligência Artificial (IA).

Este trabalho, cotado para 3 valores, tenciona solidificar os conhecimentos adquiridos tanto nas aulas teóricas como práticas.

1.2 Motivação UBI

A IA tem sido um campo de pesquisa crescente nos últimos anos, com aplicações em vários setores, incluindo saúde, finanças e tecnologia. O *Machine Learning* (ML) é uma das áreas mais importantes da IA, que permite que as máquinas aprendam e melhorem automaticamente a partir de dados, sem serem explicitamente programadas. Neste relatório, apresentamos os resultados da minha pesquisa em ML, incluindo a descrição do problema, os métodos utilizados e os resultados obtidos.

1.3 Objetivos

O objetivo deste trabalho é avaliar o desempenho de um modelo de aprendizado de máquina para prever as atividades humanas a partir de séries temporais de dados de sensores. Mais especificamente, os objetivos deste trabalho são os seguintes:

- Pré-processar os dados de séries temporais para criar instâncias de treinamento para o modelo de aprendizado de máquina.

- Treinar um modelo de rede neural *Multi Layer Perceptron* (MLP) usando os dados de treinamento.
- Avaliar o desempenho do modelo usando conjuntos de teste e calcular a matriz de confusão.
- Analisar os resultados e discutir as conclusões tiradas deste trabalho.

1.4 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.
2. O segundo capítulo – **Implementação** – descreve como foi implementado este trabalho, descrevendo assim o código.
3. O terceiro capítulo – **Resultados e discussão** – fornecer uma visão geral dos resultados do trabalho e uma discussão detalhada desses resultados.
4. O quarto capítulo – **Conclusões e Trabalho Futuro** – descreve os objetivos atingidos e os não atingidos, reunidos em tópicos auto explicativos.

Capítulo

2

Implementação

2.1 Introdução

O objetivo deste capítulo é descrever como o trabalho foi implementado na prática. Inicialmente, explicaremos o processo de pré-processamento dos dados usados no treinamento do modelo de aprendizado da máquina. Em seguida, descreveremos em detalhes os métodos de treinamento e avaliação do modelo, incluindo as ferramentas e bibliotecas usadas.

2.2 Pré-Processamento dos dados

2.2.1 Função de criar as instâncias

`create_instances(t)`: lê os dados de séries temporais do arquivo CSV fornecido pelo o professor e cria instâncias de treinamento para o modelo de aprendizado de máquina. As instâncias são criadas tomando uma linha de dados como ponto de partida e adicionando os últimos três valores de cada uma das linhas de dados t*20 que se seguem, pois são 20hz de leitura. As instâncias são então armazenadas em um arquivo CSV.

2.2.2 Função de criar k *folds*

Esta função lê as instâncias e as divide em k conjuntos de *fold*. Além disso, essa função também faz a separação dos conjuntos de *fold* por id, garantindo que nenhum id aparece em mais de um conjunto de *fold*. Isso é importante porque é importante que o conjunto de treinamento e o conjunto de teste se-

jam representativos da população de dados e não contenham ids duplicados, o que poderia distorcer os resultados.

```
ids = set(ids)
        ids = len(ids)
        # vai existir ids/k fold sets (sempre arredondado para baixo)
        id_fold_set = ids//k
        id_fold_set_rest = ids % k
        # percorrer o ficheiro e meter id_fold_set em cada fold set
        aux = -1
        ids_aux = []
        for i in range(ids):
            ids_aux.append([])
        for i in range(k):
            k_fold_sets.append([])
        ids_ = []
        for i in range(len(lines)):
            linha = lines[i].replace("\n", "")
            linha = linha.split(",")
            id_i = linha[-1]
            # se o id nao estiver no fold set e ainda nao tiver todos os
                 ids volta do inicio
            if id_i not in ids_:
                aux = aux + 1
                ids_.append(id_i)
            ids_aux[aux].append(lines[i])
```

Excerto de Código 2.1: Excerto de código divisão por id

2.2.3 Normalização

Esta função de normalização lê um conjunto de dados de treinamento e calcula os valores máximo e mínimo de cada coluna. Esses valores são armazenados em um arquivo CSV separado para uso futuro. Em seguida, a função lê o conjunto de dados de treinamento e aplica a fórmula de normalização para cada valor, produzindo um novo conjunto de dados normalizado, como se pode ver no excerto de código em 2.2.3.

Esta também lê o conjunto de dados de teste e validação e aplica a mesma fórmula de normalização, utilizando os valores máximo e mínimo armazenados anteriormente. Isso garante que os valores do conjunto de teste e validação estejam na mesma escala que os valores do conjunto de treinamento.

```
    with open("csv/train/train_set_"+str(n)+".csv", "r") as f:
            lines = f.readlines()
            for i in range(len(lines)):
                linha = lines[i].replace("\n", "")
                linha = linha.split(",")
                linha_str = ""
                for j in range(tamanho):
                    linha[j] = (float(linha[j]) - max_min[j][0]) / (
                        max_min[j][1] - max_min[j][0])
                    if j == tamanho-1:
                        linha_str += str(linha[j]) + "," + linha[j+1] +
                            "," + linha[j+2]
                    else:
                        linha_str += str(linha[j]) + ","
                linhas.append(linha_str)
                # escrever para o ficheiro
        with open("csv/train/train_set_"+str(n)+".csv", "w") as f:
            for i in linhas:
                f.write(i + "\n")
```

Excerto de Código 2.2: Excerto da normalização

2.3 Treinamento e Obtenção de Resultados

Neste secção realizei da seguinte maneira:

1. Criar um classificador de rede neural de múltiplas camadas (MLP) e o treina com os dados de treinamento, como se vê no excerto de código 2.3.

2. Lê os dados de teste e usa o classificador treinado para prever os rótulos desses dados.

3. Calcula a curva *Receiver Operating Characteristic* (ROC) e a área sob a curva *Area Under the Curve* (AUC) para cada classe e para todas as classes.

4. Plota gráficos da curva ROC para cada classe e para todas as classes.

5. Calcula a matriz de confusão para cada classe e para todas as classes, somando cada classe ficando assim por *fold*.

6. Armazena os valores da AUC e das matrizes de confusão em listas para posterior análise.

7. Depois que todos os *folds* tiverem sido processados, o programa mostra os resultados finais, incluindo a média das AUCs e as matrizes de confusão médias.

```
# Multi Layer Perceptron (MLP)
classificador = MLPClassifier(hidden_layer_sizes=(60, 60),
    random_state=1, solver='lbfgs', max_iter=1500)

# Ler ficheiro de treino
with open("csv/train/train_set_"+str(n)+".csv", "r") as f:
    x, y = ler_ficheiro(f)

# Treinar o modelo
classificador.fit(x, y)
# Ler ficheiro de teste
with open("csv/test/test_validation_set_"+str(n)+".csv", "r") as
     f:
    xt, yt = ler_ficheiro(f)
# Prever os valores
y_pred = classificador.predict(xt)
```

Excerto de Código 2.3: Criação da rede neuronal e seu treino

Capítulo

3

Resultados e discussão

3.1 Introdução

O objetivo deste capítulo é apresentar os resultados obtidos durante o trabalho e discuti-los em detalhes. Inicialmente, apresentaremos os resultados de forma clara e concisa, utilizando gráficos e texto para ilustrar os dados de forma eficiente. Em seguida, discutiremos os resultados, explorando o que os resultados dizem sobre o problema em questão. Além disso, faremos uma análise crítica dos resultados, discutindo os pontos fortes e fracos do nosso trabalho e sugerindo formas de melhorá-lo no futuro.

3.2 Apresentação dos resultados

Nesta secção será apresentado todos os dados obtidos átraves da máquina. Estes dados serão:

- Gráfico da Curva ROC de cada *fold*;
- Matriz Confusão de cada *fold*;
- AUC de cada *fold*
- e por fim vai ser apresentado AUC média e o seu desvio padrão.

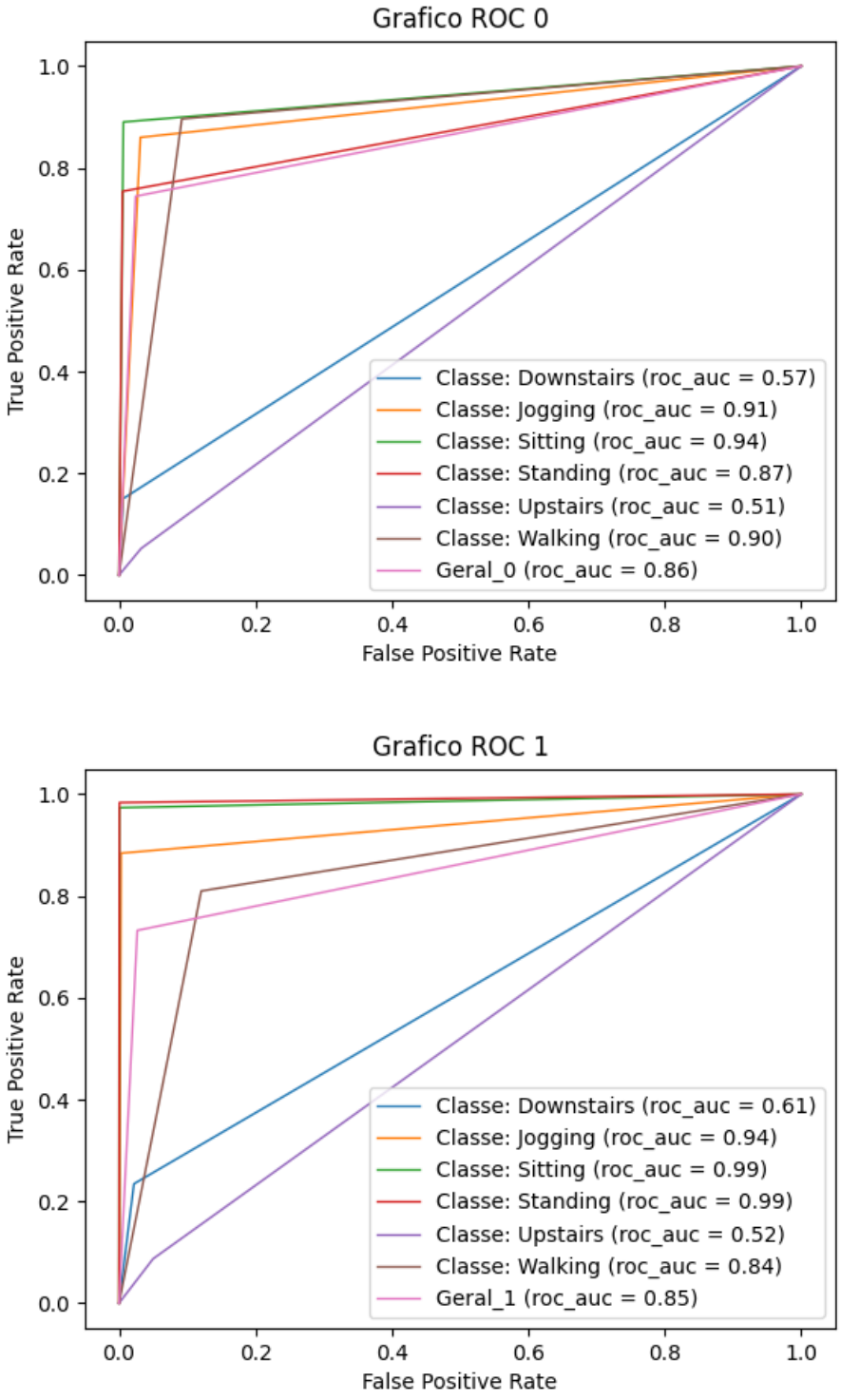

Figura 3.1: Curva ROC 0 e 1

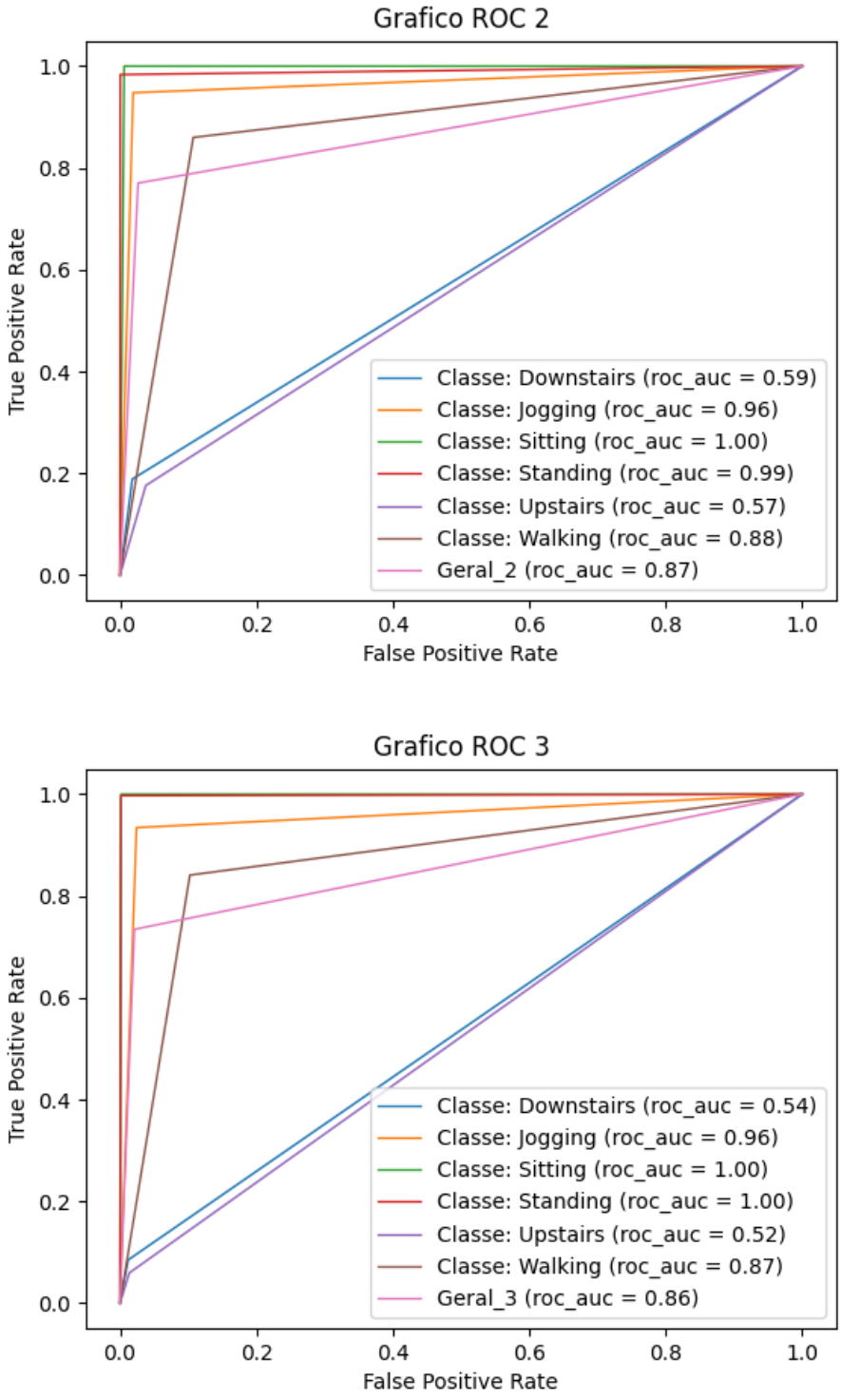

Figura 3.2: Curva ROC 2 e 3

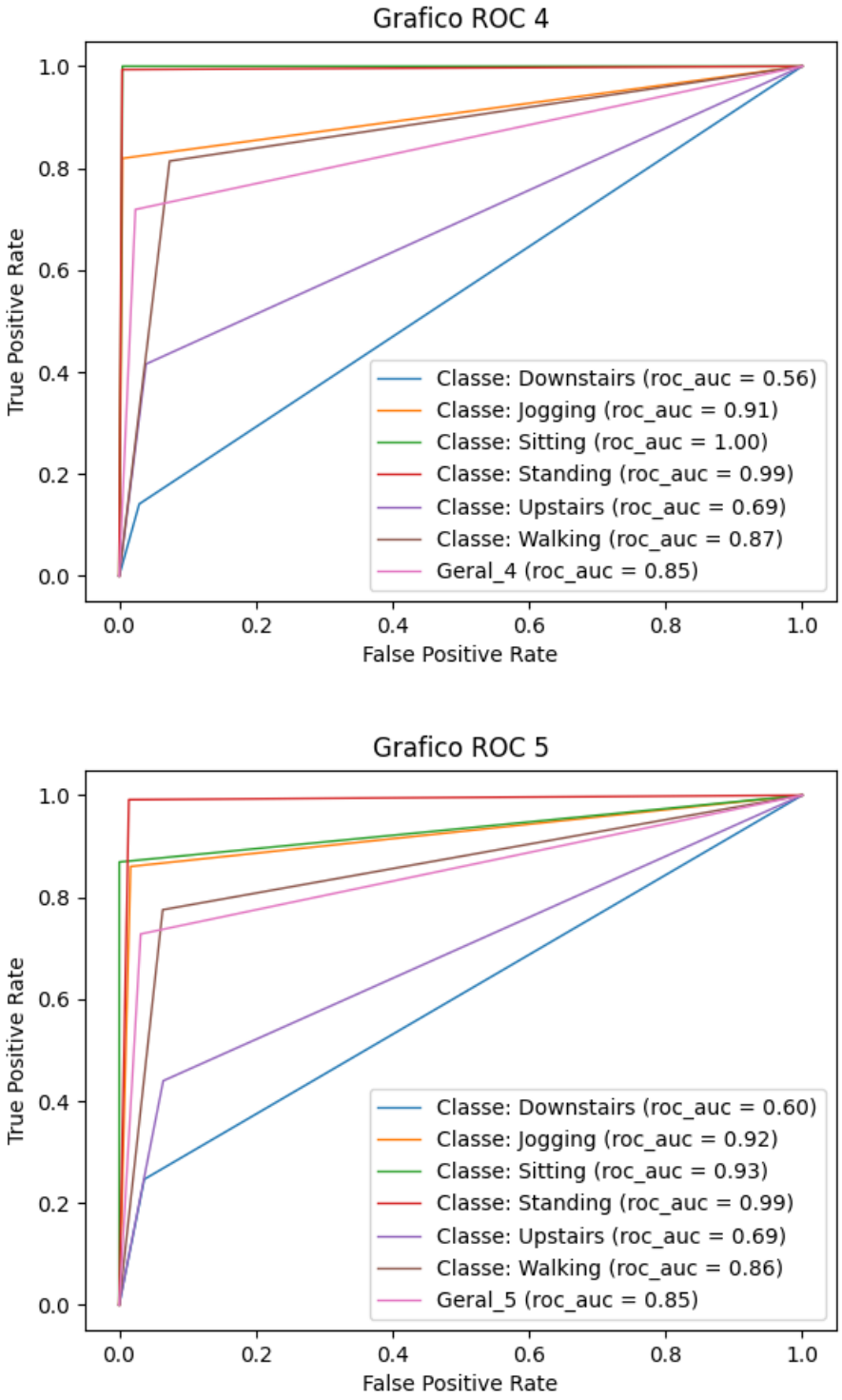

Figura 3.3: Curva ROC 4 e 5

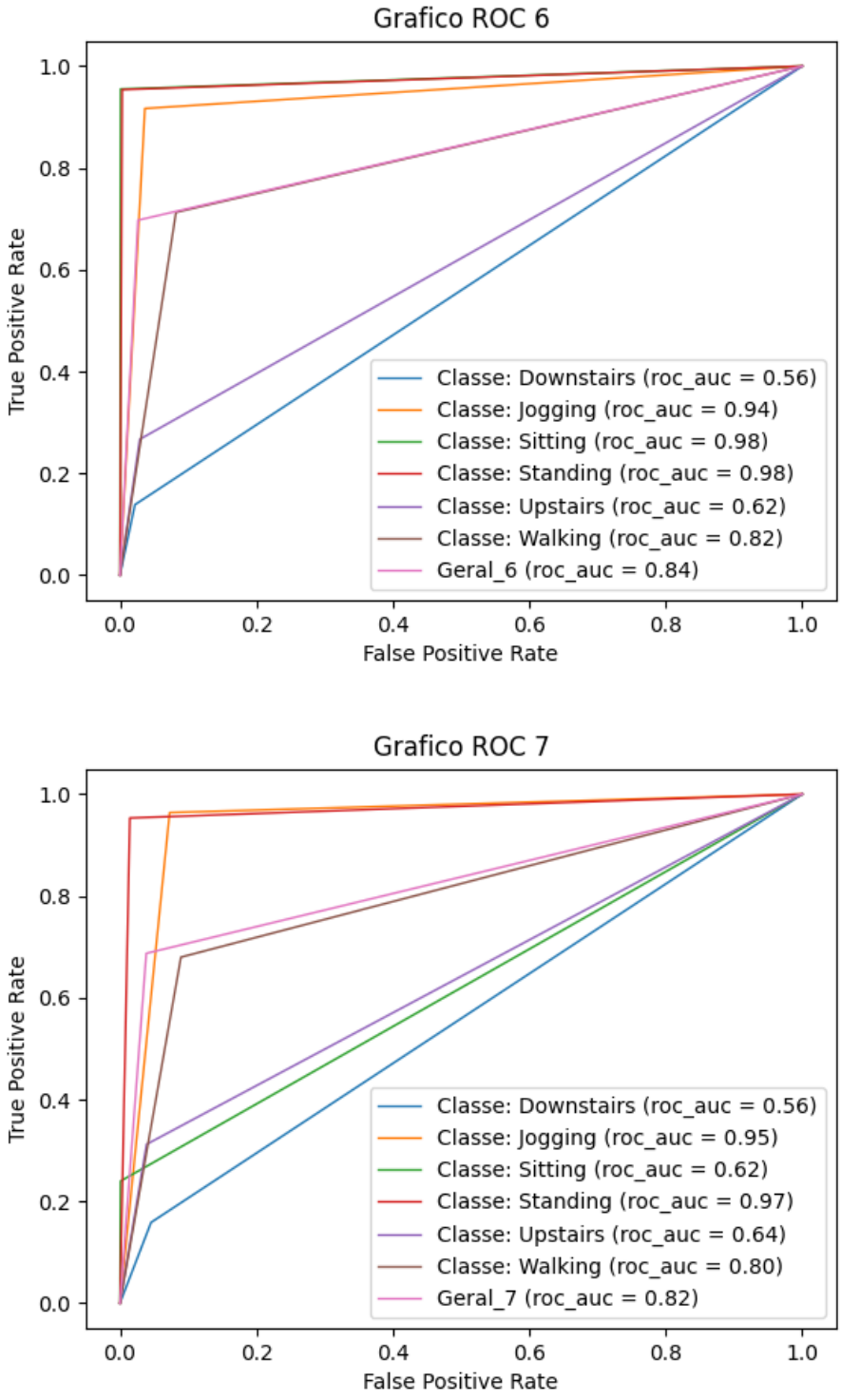

Figura 3.4: Curva ROC 6 e 7

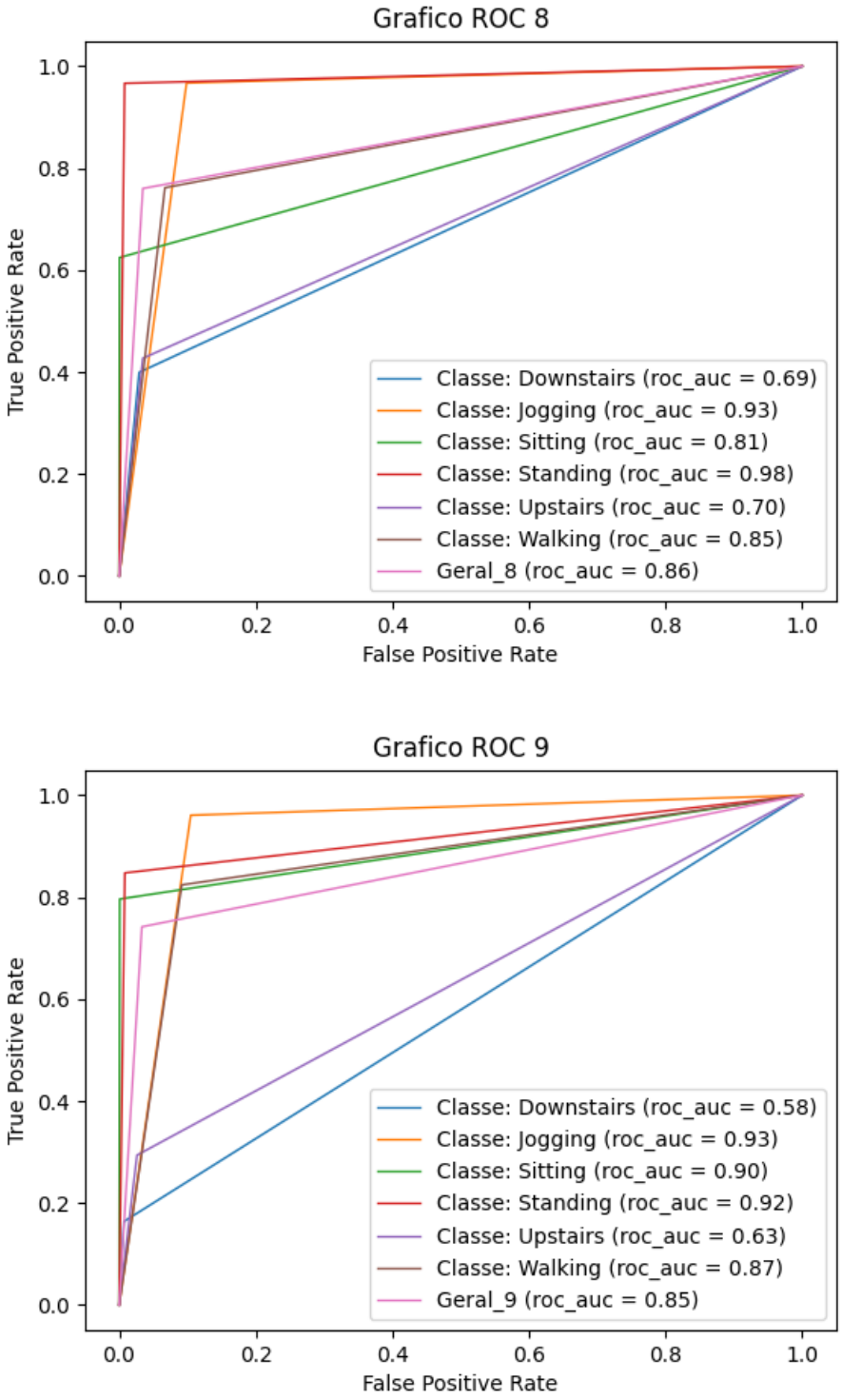

Figura 3.5: Curva ROC 8 e 9

```
Matriz de confusao 0:
[[1137210   28195]
 [  59601  173480]]
Matriz de confusao 1:
[[1044366   28674]
 [  57453  157155]]
Matriz de confusao 2:
[[1206095   32760]
 [  56946  190825]]
Matriz de confusao 3:
[[1252352   27008]
 [  67947  187925]]
Matriz de confusao 4:
[[1109564   27001]
 [  63903  163410]]
Matriz de confusao 5:
[[1104920   35885]
 [  62118  166043]]
Matriz de confusao 6:
[[952757  25823]
 [ 59243 136473]]
Matriz de confusao 7:
[[731091  29304]
 [ 47581 104498]]
Matriz de confusao 8:
[[1418817   50808]
 [  70517  223408]]
Matriz de confusao 9:
[[1039248   35667]
 [  55506  159477]]
Auc valor 0:
0.8600486526143273
Auc valor 1:
0.8527832140460747
Auc valor 2:
0.8718615172881411
Auc valor 3:
0.8566693503001501
Auc valor 4:
0.8475599723728956
Auc valor 5:
0.8481445119893408
Auc valor 6:
0.8354564777534795
Auc valor 7:
0.8242959251441685
Auc valor 8:
0.8627564854980012
```

```
Auc valor 9:
0.8543154575012908
FINAL: AUC media +- AUC desvio padrao:
0.8513891564507869 +- 0.012941373515674402
```

Excerto de Código 3.1: Matriz Confusão e AUC dos 10 k-folds

3.3 Discussão dos resultados

Ao analizar os gráficos conseguimos observar das seis atividades, duas delas nunca tem um grande aumento, sendo estas *Downstairs, Upstrais* pois estas tem um movimento parecido sendo mais difícil distinguir através do acelerômetro, precisando o modelo talvez de mais interações ou camadas ocultas.

As matrizes de confusão são usadas para resumir os resultados da classificação do modelo. Cada matriz de confusão corresponde a uma *fold* e mostra o número de instâncias da classe que foram classificadas corretamente (valores da diagonal principal) e incorretamente (valores fora da diagonal principal). Por exemplo, matriz de confusão do *fold* 0, o modelo classificou corretamente 1286690 instâncias (Verdadeiro Positivo, Verdadeiro Negativo) e classificou incorretamente 87796 instâncias (Falso Negativo, Falso Positivo).

A AUC é uma métrica comum para avaliar a qualidade de um modelo de classificação. Valores próximos de 1 indicam um bom desempenho e valores próximos de 0 indicam um mau desempenho. Portanto podemos saber que este modelo indicou um bom desempenho, mas podendo ser melhorado se aumentasse as camadas ocultas fazendo assim mais analise dos dados.

Capítulo

4

Conclusões e Trabalho Futuro

4.1 Introdução

Esta secção contém uma breve análise relativa aos objetivos cumpridos, ao que poderia ser feito para um trabalho futuro.

4.2 Conclusões e Trabalho Futuro

Neste trabalho foi possível consolidar os conhecimentos adquiridos em contexto de aula, nomeadamente:

- Pré-processamento dos dados;
- Treino e validação da máquina;
- Análise dos dados obtidos;
- Aprimoramento da linguagem python.

Para melhorar este *bot*, podemos aumentar o número de camadas ocultas, otimizar o código de pré-processamento de dados e obter mais dados para treinamento, o que pode levar a resultados ainda melhores.

Bibliografia

[1] Scikit-learn
https://scikit-learn.org/stable/index.html

[2] W3schools
https://www.w3schools.com/python/python_ml_getting_started.asp