EA/ECAC 2025
TP1 – Part B
Data de Entrega Final – 07.12.2025

ATENÇÃO: PODEM FAZER ALTERAÇÕES À PRIMEIRA META (PARTE A) E SUBMETER TODO O PROJETO NA ENTREGA FINAL PARA AVALIAÇÃO

Classificação de Atividades Humanas

Module B – Application and structured evaluation of a machine learning model to the problem. We will consider and evaluate the different approaches for feature selection of module A, perform data augmentation, and compare the performance of our feature set with embeddings learned from a deep learning model. For this module, **we will consider only activities 1 to 7**, so you can discard all segments with activities higher than 7.

1. **Data Augmentation:** The objective is to build a function capable of generating synthetic examples to add variety to the training set (features extracted for the segments).

   1.1. Analyse the balance between the number of examples available for the set of activities you are considering. Is the dataset balanced?

   1.2. Create a function that receives the dataset and implements the SMOTE method to generate K new samples for a given activity A.

   1.3. Use the function before to generate and visualize 3 new samples of activity 4 of participant 3. Use only the samples of that participant in the process. For the visualization, use a 2D scatter plot considering only the first two features, make the color of the points different by activity, and highlight the synthetic ones.

2. **Embedding features**: An alternative to explicit feature extraction is to use models that learn features directly from the raw data. We will perform a type of transfer learning, i.e. we will use a model trained with a different dataset and use it in our problem. For that, seee the functions provided in the file "embeddings_extractor.py" to extract a feature vector of embeddings for each segment of the acceleration data. The provided code uses the *harnet5* model of *ssl-wearables* project (https://github.com/OxWearables/ssl-wearables), removing the classification layers and obtaining the feature representations (embeddings) after a forward pass through the pretrained model. You can find further information about the project in the paper: https://www.nature.com/articles/s41746-024-01062-3.

   2.1. Use only the x,y,z values of the accelerometer sensor. For each 5-second segment, resample it to 30Hz and then extract the embedding features (see provided code). You should end up with a dataset of shape [n_segments, n_embeddings]. This will be referred to as "EMBEDDINGS DATASET", while the one resulting from module A will be referred to as "FEATURES DATASET".

3. **Data splitting strategy:** We will use the same splitting strategies at two levels – within subject and between subjects. Split both the FEATURES and the EMBEDDINGS datasets into training, validation, and test sets.

   3.1. Use a TVT split of 60-20-20% split, within each subject (data of a single subject is present in all sets)

   3.2. Use the same split but at the subject level: put 9 subjects for training, 3 for validation, and 3 for testing.

3.3. Discuss the differences between the two strategies. Which one gives you a better estimate of the performance of the model when applied to a new participant? Justify your answer.

3.4. Prepare your pipeline in a way that, after the split, for each dataset, you can compute and transform each set into three different scenarios:

a) All features/embeddings dataset

b) PCA-reduced features/embeddings dataset (keeping just the components that explain 90% of the variance)

c) Feature-selected features/embeddings (keeping just the top 15 features selected by the ReliefF method).

NOTE: Beware that you cannot use any test segment in the computations of the PCA or ReliefF in this question, as well as for any normalization procedure you follow.

4. **Model learning:** We will now train a model to automatically classify the activity of each segment. We will use k-Nearest Neighbors.

4.1. Implement a k-Nearest Neighbors classifier (we expect you to implement your own version, but then you can use the sci-kit learn version if you need better performance. See information in: sklearn_kNN).

4.2. Create a function that receives two lists of values: the true labels and predicted labels, and returns a set of classification metrics: confusion matrix, accuracy, f1 score, precision, recall, etc.

5. **Evaluation**: For each splitting strategy (2) applied for the 3 versions (all feats, PCA feats, ReliefF feats) of the 2 datasets (features, embeddings), train the classifier and evaluate its results:

5.1. **Hyperparameter tuning:** Select the best value of k using only the train and validation data. Then, with the best k, retrain with the dataset (train + validation) and evaluate on the test set.

5.2. **Report and analyse your results.** Check the confusion matrices and observe which activities are harder to classify. Which dataset worked best (features vs embeddings)? Did feature selection improve the performance? Etc.

5.3. Perform **hypothesis testing** to compare the models. Is the best model statistically significantly better than the others? Justify the choice of the statistical tests you use. (NOTE: repeat the train-validation-test split several times to obtain the performance distributions of each model).

6. **Deployment:** Pick the best model from question 5 and make a function that receives a numpy array of shape 256 lines and 9 columns (acc x y z, gyr x y z, mag x y z) and performs all the necessary steps to return a classification from the model (normalization, feature extraction / embeddings generation, [PCA, Feature Selection], classification).

7. **Go further:** Discuss what can be done more to improve your classification system.

7.1. (BONUS) Implement and evaluate some of those ideas!