Statistical Learning for Healthcare Data - Challenge 1

Sara Maria Pagotto, Marco Poggi, Giulio Venturini

1 Introduction

Time series classification is a prevalent task that has gained significant popularity in recent years. It is now widely utilized in various domains, such as Internet of Things (IoT), signal processing, and human activity recognition. Particularly, human activity recognition has found extensive applications in the medical field, especially when it comes to fall detection.

Several studies [1][2] have proposed different solutions for automatic fall detection, aiming to identify when a person falls and send alarms to enable quick assistance. This becomes crucial, especially for elderly individuals who are more vulnerable and prone to falls. These falls can have adverse physical and psychological consequences, leading to a loss of independence or the fear of falling again. Consequently, there is an increasing need for automated algorithms capable of promptly recognizing such events. In this regard, this study focuses on movement classification using supervised learning.

The objective of this project is to compare various machine learning algorithms for the classification of time series data. The data used in this study was acquired through wearable systems, where individuals carry a device attached to their body, and sensor data is collected. Although the literature also explores the use of other sensors like gyroscopes, magnetometers, and barometers [2], for the purposes of this study, we concentrate solely on wearable fall detection systems that utilize acceleration sensors.

The original database employed in this study comprised data from 16 young and healthy subjects. These participants were recruited to perform 12 types of activities of daily living (ADL) and three types of simulated falls while wearing an inertial sensor unit (Shimmer sensing, Ireland) attached sideways to their waist at belt height. The mean and standard deviation of the age, height, and weight of the participants were 21.9 $\pm$ 2.2 years, 178.1 $\pm$ 7.8 cm, and 70.0 $\pm$ 12.1 kg, respectively. However, for this project, only one subject out of the 16 was selected for analysis, and the age, height, and weight of this particular patient are unknown.

The goal of the project is to classify the movements of this patient into three categories: Moving, Falling and Others. The reason for approaching this problem as a multiclass classification task, rather than a simple binary one (fall or no fall), lies in the availability of information that can be extracted from different types of movements. Indeed, the direction of the fall and the pre-fall position are helpful for pre-fall and post-fall analysis [1].

2 Material and Methods

The dataset used in this analysis consists of 468 CSV files originally divided into 10 classes:

- **Class 1 (MW)** - Walking: Includes walking and fast walking activities.
- **Class 2 (MR)** - Running: Includes running and fast running activities.
- **Class 3 (MJ)** - Jumping: Includes jumping and high jumping activities.
- **Class 4 (WD)** - Walking down the stairs.
- **Class 5 (WU)** - Walking up the stairs.
- **Class 6 (FF)** - Forward fall.
- **Class 7 (FS)** - Sideways fall.
- **Class 8 (FB)** - Backward fall.
- **Class 9 (LD)** - Lying down.
- **Class 10 (OT)** - Other classes: Includes sitting, standing up, and getting up from a lying position.

For the purpose of simplification, we have reduced the number of classes to 3: Moving, Falling, and Others. The conversion was performed as follows:

- Class 1, 2, 3, 4, 5 were replaced with **Class 1 (Moving)** (number of elements: 230).

- Class 6, 7, 8 were replaced with **Class 2 (Falling)** (number of elements: 136).
- Class 9, 10 were replaced with **Class 3 (Others)** (number of elements: 102).

Each of the files we used was composed of four columns: x, y, z, and t, representing the acceleration along the x, y, and z axes at different time points. Here in Fig.1 we present an example where the x, y, and z columns were plotted as a function of time t.

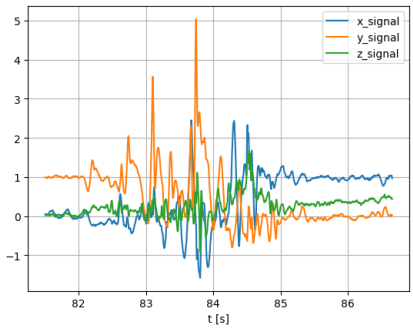

Figure 1: Example plot of acceleration along x, y, and z axes as a function of time t.

The first step will be to perform feature extraction from the files. We aim to capture meaningful characteristics of the acceleration time series that can be used for classification. Various statistical features will be computed, including mean, variance, root mean square (RMS), interquartile range (IQR) and others.

For the classification task, we will explore several machine learning algorithms, including Support Vector Machines (SVM), Random Forest, Logistic Regression, and K-Nearest Neighbors (KNN). To optimize the performance of each algorithm, we will use 10-fold cross-validation to search for the best set of hyperparameters that maximize the classification accuracy.

3 Results

3.1 Preprocessing and Selected Features

In this section, we present the results of our analysis. The dataset is randomly split into a training set (70%) and a test set (30%), while maintaining the proportions between classes.

Focusing on the training set, we extract characteristic attributes related to signal analysis. An exploratory analysis is conducted to identify the most informative features for classification. To assess the suitability of features, we represent them using boxplots, and retain those that exhibit significant differences among the members of the three classes. Here in Fig.2, we present an example of a retained feature and a non-retained feature.

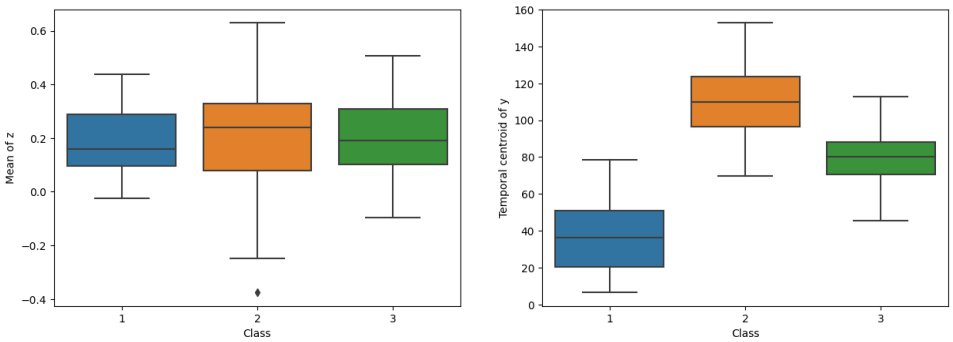

Figure 2: Example of a non-retained feature (left) and a retained feature (right).

All the extracted features are available in the attached Jupyter notebook. However, for brevity, we only mention the selected ones:

- `mean_x`, `mean_y`: Mean value of acceleration along the X and Y axes.
- `var_x`, `var_y`, `var_z`: Variance of acceleration along the X, Y and Z axes.
- `rms_x`, `rms_y`, `rms_z`: Root Mean Square of acceleration along the X, Y and Z axes.
- `iq_range_x`, `iq_range_y`, `iq_range_z`: Interquartile range of acceleration along the X, Y and Z axes.

- `max-min_x`, `max-min_y`, `max-min_z`: Difference between the maximum and minimum values along the X, Y and Z axes.
- `%_peaks_x`, `%_peaks_y`, `%_peaks_z`: Percentage of peaks above a certain threshold in the acceleration signal along the X, Y and Z axes.
- `%_positive_y`: Percentage of positive values in the acceleration signal along the Y axis.
- `power_x`, `power_y`, `power_z`: Power of the acceleration signal along the X, Y and Z axes.
- `temporal_centroid_x`, `temporal_centroid_y`, `temporal_centroid_z`: Temporal centroid of the acceleration signal along the X, Y and Z axes.

Most of these variables are self-explanatory, but we would like to clarify a couple of things.

The time series are influenced by noise and this could result in unreliable values for maximum and minimum. To focus on the trend rather than the noise, we compute the `max-min` variables based on a third-order polynomial interpolation of the time series instead of the raw time series itself. An example of an approximation using such a method is depicted in Fig.3.

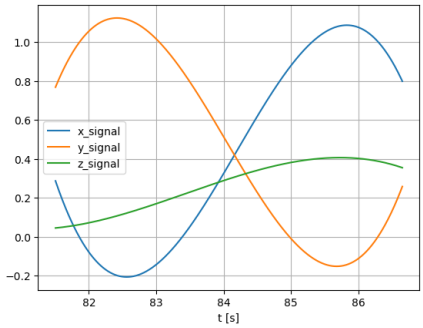

Figure 3: Third-order polynomial interpolation of the signals shown Fig. 1

.

With the exception of the `max-min` variables, all other features have been computed using the raw signals rather than the polynomial approximations.

Additionally, for peak detection, we apply different empirical thresholds: 0.5 for the X-axis, 1 for the Y-axis, and 0.5 for the Z-axis.

Moving on, we analyze the correlation between the selected variables. To do this, we generate a correlation map as shown in Fig.4.

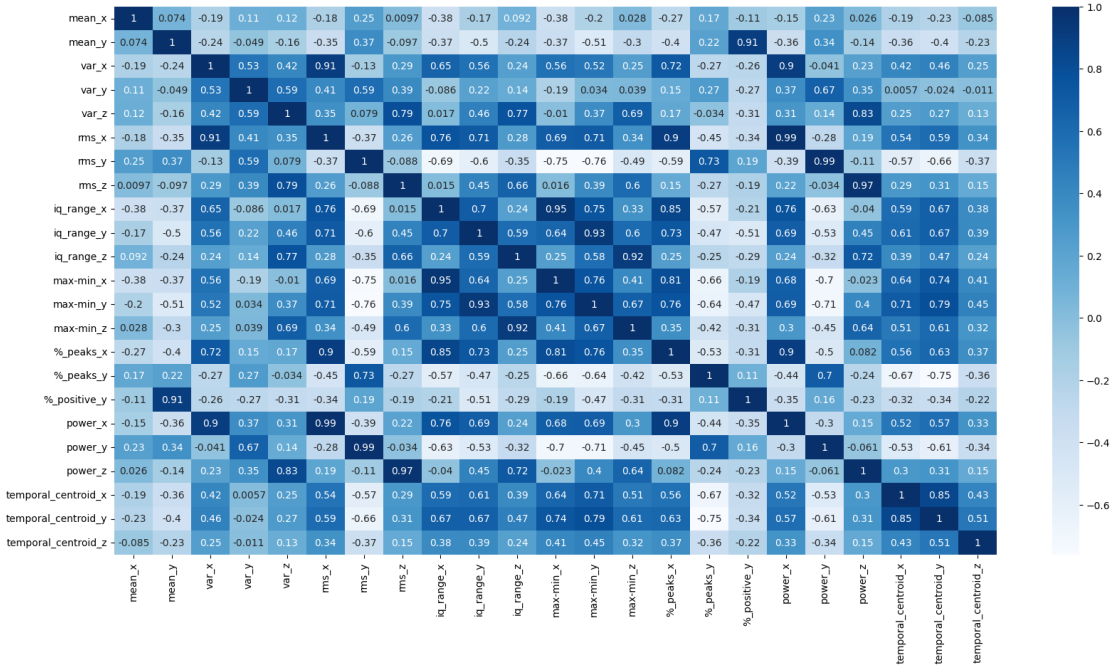

Figure 4: Correlation map of the selected variables.

Whenever two variables exhibit a correlation above 0.8, we remove one of them to eliminate redundancy. The eliminated variables are: `power_z`, `rms_x`, `power_y`, `temporal_centroid_y`, `power_x`, `max-min_y`, `%_positive_y`, `%_peaks_x`, `max-min_x`, and `max-min_z`.

It's interesting to notice that the `max-min` variables are extremely correlated with the `iq_range` ones. In this case, we specifically choose to keep the ranges, so that computation of the interpolating polynomials isn't necessary anymore.

One final preprocessing step consists of scaling the features to ensure that their magnitudes are consistent across different attributes. This is important for many machine learning algorithms, as features with different scales can lead to biased models.

3.2 Classification Models

After this long preprocessing phase, we proceed to evaluate different classification models. We tune the hyperparameters using 10-fold cross-validation. In the provided Jupyter notebook, all the tried values for the hyperparameters are presented. Here, we only report the best ones in terms of accuracy:

- **SVM**: {`C`: 10, `gamma`: 0, `kernel`: linear}
- **Random Forest**: {`max_depth`: 5, `n_estimators`: 200}
- **Logistic Regression**: No hyperparameters specified
- **KNN**: {`n_neighbors`: 5, `p`: 1}

We then calculate accuracy, precision, recall, and F1-score for each of these four models. We use the macro average version of precision, recall, and F1-score to give equal weight to all classes, even if their sizes differ. The results are summarized in Table 1 below:

Model	Accuracy	Precision	Recall	F1-Score
SVM	0.981534	0.981011	0.978710	0.978929
Random Forest	0.972443	0.971654	0.968340	0.969074
Logistic Regression	0.969413	0.967110	0.963578	0.964127
KNN	0.963447	0.962279	0.956911	0.958104

Table 1: Performance metrics for each model.

3.3 Final Model and Features Importance

All models demonstrate excellent performance, with SVM slightly outperforming the others. Therefore, we select SVM with {`C`: 10, `gamma`: 0, `kernel`: linear} as our final model. The confusion matrix for the SVM model is shown in Fig. 5 below:

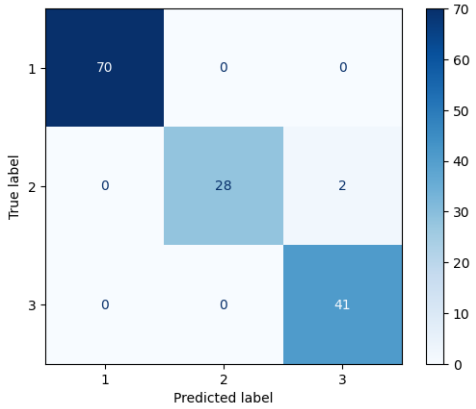

Figure 5: Confusion matrix for the best SVM model on the test set.

The SVM model with a linear kernel was selected not only for its exceptional classification performance but also for its high interpretability. The utilization of a linear kernel allows for the extraction of feature importance, as illustrated in Figure 6.

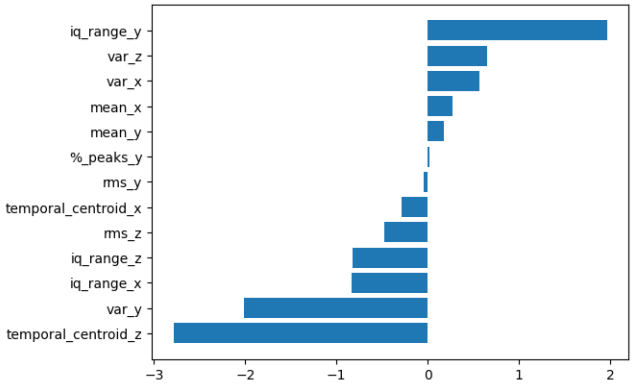

Figure 6: Feature importance of the selected model

Figure 6 presents the representation of SVM coefficients, providing a means to identify the primary features employed in the classification process. Specifically, the magnitude of the coefficients indicates the importance of the corresponding features in the decision-making process. Larger absolute coefficients signify greater importance. Positive coefficients indicate that an increase in the value of the associated feature results in an increased probability of belonging to a specific class. Conversely, negative coefficients indicate that an increase in the value of the corresponding feature decreases its likelihood of belonging to a specific class.

In this case, the most significant features were determined to be `iq_range_y` , `temporal_centroid_z`, and `var_y`.

By analyzing Figure 6, one can identify features with minimal contribution and potentially remove them from the model. However, considering that our model was already compact and demonstrated excellent performance, the decision was made to retain all features. In the future, it may be worth considering feature reduction in machine learning to speed up training, avoid overfitting, and ultimately improve classification outcomes by reducing noise in the data.

4 Discussion and Conclusion

Support Vector Machine (SVM), Random Forest, Logistic Regression, and K-Nearest Neighbors (KNN) all demonstrate strong performance in accurately classifying the activities into the categories of Moving, Falling, and Others. These machine learning algorithms showcase their effectiveness in accurately categorizing activities based on wearable sensor data.

The findings hold potential implications in clinical practice, offering opportunities for real-time activity monitoring, fall detection, and personalized care plans. Accurate activity classification can enable early detection of falls, timely interventions, and remote monitoring of an individual's functional status. These advancements have the potential to improve patient outcomes, enhance quality of life, and optimize healthcare resource allocation.

In conclusion, our study highlights the efficacy of machine learning algorithms in classifying activities based on wearable sensor data. However, further research is needed to validate and refine the models on larger datasets and in real-world scenarios. These efforts will contribute to establishing the robustness and generalizability of the proposed methods, paving the way for their integration into clinical practice and enhancing their utility in healthcare settings.

5 Addendum - 10 Classes

Since the results on the reduced to three classes problem were so good, we also tried using our final SVM model on the original problem, with all the 10 different classes.

As we can see from the following confusion matrix, the results are still very good, even if not almost perfect as they were with 3 classes.

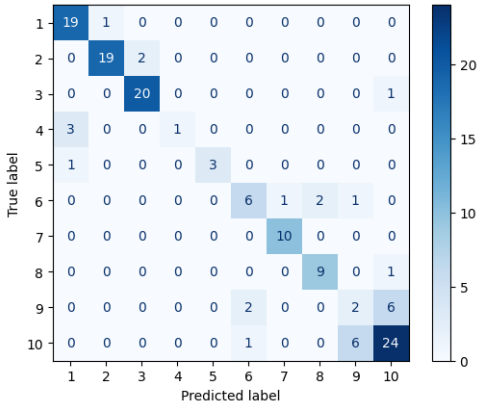

Figure 7: Confusion matrix of the SVM model for 10 classes

The accuracy of this model is 0.801418, that is already satisfying, being the model optimized for the reduced problem.

In future developments, all the model selection and hyperparamters fixing can be done on the 10 classes problem directly, and even better results can be reached.

References

[1] Y Choi, A S Ralhan, and S Ko. "A study on machine learning algorithms for fall detection and movement classification". In: *2011 International Conference on Information Science and Applications*. Jeju Island: IEEE, Apr. 2011.

[2] D Razum et al. "Optimal threshold selection for threshold-based fall detection algorithms with multiple features". In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. Opatija: IEEE, May 2018.