Impact of Feature Normalization on Machine Learning-Based Human Fall Detection

Moustafa Fayad[1(✉)], Mohamed-Yacine Hachani[1], Ahmed Mostefaoui[2], Mohammed Amine Merzoug[1], Isabelle Lajoie[1], and Réda Yahiaoui[1]

[1] Nanomedicine, Imagery, and Therapeutics Laboratory, Univ. of Franche-Comté, 25030 Besançon, France
{moustafa.fayad,mohamed-yacine.hachani,mohammed.merzoug,isabelle.lajoie, reda.yahiaoui}@univ-fcomte.fr, moustafa.fayad@hotmail.fr

[2] DISC Department, FEMTO-ST Institute, Univ. of Franche-Comté, 90000 Belfort, France
ahmed.mostefaoui@univ-fcomte.fr

Abstract. This paper investigates the impact of normalizing data acquired from different multimedia sensor devices on the performance of machine-learning-based human fall detection. Specifically, we consider two fall detection datasets (URFD and UP-Fall) and study the impact of eight normalization techniques (min-max, z-score, decimal, sigmoid, tanh, softmax, maximum absolute, and statistical column) on the accuracy and training time of four machine learning classifiers optimized using Grid-Search (namely, support vector machine with radial basis function, k-nearest neighbors, Gaussian Naive Bayes, and decision tree). The conducted experiments confirm that data normalization leads to a significant speed-up in the training of machine learning models and demonstrate which data normalization techniques are the most efficient in terms of accuracy in the context of elderly fall detection.

Keywords: Human fall detection · elderly fall detection · machine learning · data normalization impact

1 Introduction

According to the World Health Organization [31], the number of older vulnerable people is increasing sharply compared to other age groups, increasing thus the risk of falls, chronic diseases, and household accidents with severe consequences. Recent reports and statistics [28,31] also show that falling is the second most significant cause of unintentional injury mortality, especially among the elderly: 684,000 people die every year from falls, and those over 60 years old experience the highest number of fatal falls [31].

To address this public health problem, the scientific community has prioritized developing fall detection solutions that are both fast and reliable (with zero

R. Chbeir et al. (Eds.): MEDES 2023, CCIS 2022, pp. 147–161, 2024.
https://doi.org/10.1007/978-3-031-51643-6_11

false alarms) to reduce risks and consequences. Because in a critical situation, a delay or an error in the elderly monitoring system could cost a person's life. For instance, a long period of lying on the ground after a fall influences both physical and psychological health and increases mortality [6,12].

Particular attention has been given to the development of machine learning-based fall detection models [13,29], which have shown promising results in terms of minimizing decision-making errors without human intervention and are more favorable over classical processing techniques based on thresholds [24].

The efficiency of AI models depends heavily on the investment in the data preprocessing stage. In this regard, numerous works in the literature have examined different techniques that can be deployed at the different stages of fall detection models. However, to the extent of our knowledge, no studies have investigated the impact of feature normalization (scaling) on classification performance in the case of elderly fall detection. The existing research works focus mainly on feature preprocessing such as segmentation and representation domain.

In this paper, we investigate this path by studying the impact of eight normalization techniques (min-max, z-score, decimal, sigmoid, tanh, softmax, maximum absolute, and statistical column) on the performance (accuracy and training time) of four classifiers: SVM-RBG (support vector machine with radial basis function), KNN (k-nearest neighbors), GNB (Gaussian Naive Bayes), and DT (decision tree). Indeed, these 2 performance metrics are essential considerations when developing and deploying a model. The conducted experimentation has been structured as follows: *i*) dataset selection, *ii*) data preprocessing, *iii*) hyper-parameters tuning, and *iv*) evaluation of each optimized model with both normalized and unnormalized data. To have a fair analysis and comparison, we *i*) selected two public fall datasets of different sizes but with the same imbalance ratio: URFD (University of Rzeszow Fall Detection) [15] and UP-Fall [17], and *ii*) tuned the hyperparameters of each considered model using Grid-Search.

The obtained results presented in the paper confirm that data normalization has a significant impact on speeding up the training of machine learning models and also demonstrate which normalization techniques are the best and which are the worst in terms of accuracy in the context of elderly fall detection.

The remaining of this paper is organized into four sections. Section 2 presents previous studies related to our research work in the field of human fall detection. Section 3 provides the background of this study. Section 4 details the conducted experimentation and then presents and analyzes the obtained results. Finally, Sect. 5 concludes the paper with future work.

2 Related Work

The topic of fall detection has attracted much attention from researchers. Several works in the literature examine the techniques that can be deployed at different stages of the fall detection approach's life cycle. Nevertheless, there are no studies on the impact of feature Normalization. Indeed, the research community has fundamentally focused on feature preprocessing (segmentation, representation

domain, etc.). In this part, we will not compare the performance of existing works but rather present an exhaustive picture to position our new work.

(1) Data segmentation aims to carry out processing according to criteria making it possible to define the relevant information. It also aims to estimate the decision of the model correctly. Aziz et al. [1] evaluate the impact of window size and lead time on the sensitivity and specificity of a pre-impact fall detector. Kinematic data is acquired using an Opal model containing a triaxial accelerometer and a gyroscope at 128 Hz for 15 s for each trial. 18 features related to triaxial acceleration, velocity, and angular velocity are evaluated by deploying the SVM-RBF model and 10 cross-validations. In the experiment, window sizes and lead time vary between 0.125–1.125 s and 0.0625–1.125 s with an increment step of 0.0625 s. Therefore, system performance is high (>95% sensitivity and >90% specificity) for combinations of window size 0.0625–0.1875 s and lead time of 0.125–1 s.
(2) The data representation allows us to evaluate the impact of the domains (temporal, frequencies, differential, etc.) and the relationships between the variables on the performance. Wagner et al. [30] investigated the impact of 5 regularized numerical differentiation methods on the performance of a Naïve Bayes-based fall detector. The methods used are central difference, regularized central difference, Kalman filter, Tikhonov regularization, and smoothing approximation followed by analytical differentiation. Three features related to the person's center of mass (maximum downward vertical velocity, maximum horizontal velocity, and maximum total velocity) were processed using the leave one out cross-validation procedure. For the UR Fall Detection Dataset, using a regularized version of the central difference method provided superior performance (increased the true positive rate and reduced the false positive rate) compared to the other methods.

The authors in [26] explored the performance of 4 classifiers (Naïve Bayes, KNN, J48, and random Forest) separately and in an ensemble based on the stacking of classifiers. They calculated 8 features from each segment of accelerometer data provided by central one-second values around the peak. These characteristics include (resultant, variance, standard deviation, Euclidean norm, Root mean square, Kurtosis, Skewness, and the geometric mean). They demonstrated that the stacking-based ensemble had superior performance, SE 89%, and SP 95%, compared to the other classifiers.

The above studies in the literature show that different techniques strongly impact the performance of a fall detection approach. However, no research has analyzed the influence of feature scaling techniques on classification performance. Therefore, we deployed 8 normalization methods and 4 types of classifiers in our study to conduct a fair comparison against non-normalized features.

3 Background

This section presents the considered data normalization techniques and introduces the Grid-Search method utilized to optimize the considered fall detection classifiers.

3.1 Data Preprocessing and Normalization

The main critical step in fall detection approaches is measuring the relevant parameters during the target's movement. The different smart devices that allow the acquisition of such essential data can be classified into *i*) body sensors (accelerometers, gyroscopes, magnetometers, RFID) and *ii*) external sensors (piezoelectric, passive infrared, microphones, cameras) [18,21].

These devices generate data that machine-learning models cannot directly use. The raw format causes difficulties in extracting relevant information and has a significant impact on performance. Several factors could cause the quality decrease, for instance: aberrant measurements, error in the distribution of variables, absence of variables, and presence of duplicate or redundant data [8,19].

To remedy this problem and have an optimal data interpretation, data preprocessing is the first mandatory step for machine learning-based approaches. This transformation step resolves the dominance of large-scale features over the decision of which model to deploy. Specifically, it generates new values in new scales while preserving the distribution and ratios of original data.

In the following, we introduce the considered normalization techniques.

- Min-max [9] is one of the most popular data normalization techniques. As the name implies, the new measurements are calculated according to the minimum and maximum data values (Eq. 1), and the distribution of original data is kept in the new scale of [0, 1].

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

- Z-Score [9] expresses directly the distance of each measurement from the mean value in standard deviation (σ). The sign of a normalized value gives an idea about the observation position to the mean value (μ).

$$x_{scaled} = \frac{x - \mu}{\sigma} \tag{2}$$

- In decimal normalization [9], values are transformed by moving the decimal point. In other words, this scaler divides each value by the maximum absolute value of data.

$$x_{scaled} = \frac{x}{10^j} \tag{3}$$

where j is the smallest integer such that $max(|x_{scaled}|) < 1$.

– The idea of sigmoid normalization [3,32] is to bring all feature values into a range between 0 and 1. More specifically, input variables that are greater than 1 are transformed into 1 and those that are less than 0 are transformed into 0. The form of normalized values corresponds to an S-form from 0 to 1 passing through 0.5.

$$x_{scaled} = \frac{1}{1 + e^{-x}} \tag{4}$$

– Tanh [32] maps feature values into probabilistic values between 0 and 1. The tanh function has the following formula:

$$x_{scaled} = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{5}$$

– Softmax [32] converts the input vector of n real numbers into a probability distribution of n possible outcomes. The new observations are calculated according to the following equation.

$$x_{scaled} = \frac{e^{x}}{\sum_{j=1}^{n} e^{x_j}} \tag{6}$$

– Maximum absolute scaling technique [7] re-scales each feature between -1 and 1 by dividing each value by the absolute maximum of each feature.

$$x_{scaled} = \frac{x}{max(|x|)} \tag{7}$$

– Statistical column [14] calculates the new observations according to Eq. 8. The sum of transformed values of each characteristic is equal to 1.

$$x_{scaled} = 0.1 \frac{x - \frac{\sum_1^n x}{10+n}}{\frac{\sum_1^n x}{10+n}} \tag{8}$$

3.2 Grid-Search

The proper functioning of machine learning models depends strongly on the adjustment of their hyper-parameters which control the learning process. One way to achieve this is that hyper-parameters can be manually adjusted by experimenting and then choosing the optimal values with minimum error and high performance. However, such manual configuration is time-consuming and does not guarantee the selection of optimal values (at least during the first attempts).

In the literature, several methods (such as Grid-Search, Random-Search, and Genetic Algorithm) facilitate this crucial task of model building [16]. In the present study, we deployed Grid-Search, which is one of the most used techniques for the optimization of hyper-parameters [2]. As the name implies, Grid-Search treats hyper-parameter tuning as a search in a grid (Fig. 1). It checks all the combinations of hyper-parameters and compares performance to deduce the best values. In other terms, for each combination of values, the algorithm will be

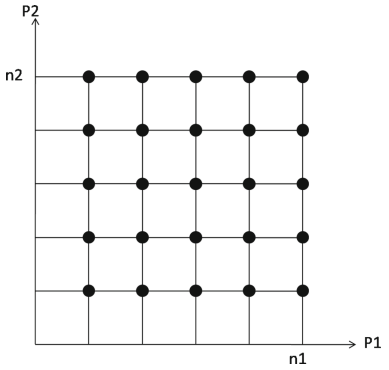

Fig. 1. A search grid presenting each possible combination of two hyper-parameters P1 and P2.

Table 1. Utilized URFD and UP-Fall datasets.

Dataset	Year	Activities	Sensors	Actors	IMU position	Camera position	IR
URFD	2014	70 (30 falls and 40 ADLs)	- 2 Kinect cameras - 1 PS Move - 1 x-IMU	5	Near the pelvis (waist)	Parallel to the floor (1 m) and ceiling configuration (2.5 m)	$\frac{9741}{1803} \approx 5$
UP-Fall	2019	11 (5 falls and 6 ADLs)	- 2 Microsoft LifeCam - 5 IMU - 1 ECG NeuroSky MindWave - 6 infrared	17	Ankle, pocket, waist, neck and wrist	Frontal view (1.82 m) Lateral view (1.82 m)	$\frac{248727}{45951} \approx 5$

trained and a score calculated. Then, the combination with the highest score will be selected. For instance, if we use Grid-Search with a model that has two parameters ($1 \leq P1 \leq n1, 1 \leq P2 \leq n2$) and k-fold cross-validation, we have $n1 \times n2 \times k$ attempts. Therefore, one should not overdo this method when defining hyper-parameter scales to avoid reaching limits regarding computational capacity and model training time.

4 Experimentations

To study the impact of data normalization on the accuracy and training time of fall detection models, we proceeded according to the following four steps (Fig. 2): *i*) dataset selection, *ii*) data preprocessing, *iii*) hyper-parameters tuning, and *iv*) evaluation of optimized models with both normalized and unnormalized data.

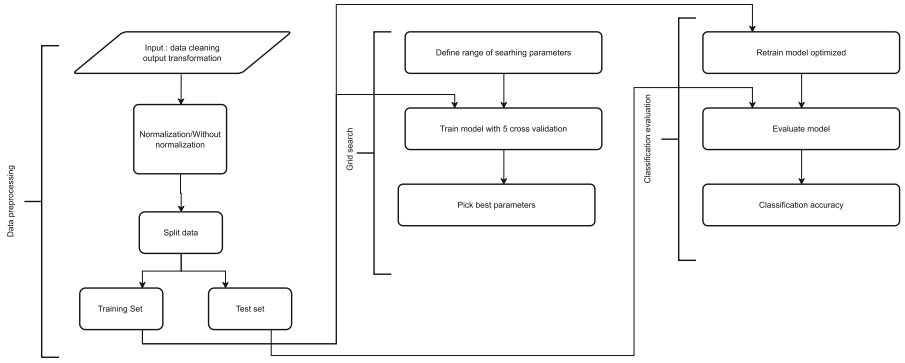

Fig. 2. Experimental approach.

4.1 Dataset Selection

We considered the publicly available URFD [15] and UP-Fall [17] datasets because *i*) they are rich in vision and movement information, and *ii*) they share the same imbalance ratio of 5. We recall that the imbalance ratio (denoted IR) is defined as follows [11,25]:

$$IR = \frac{N_-}{N_+} \tag{9}$$

where N_- is the number of majority (or negative) observations in the dataset and N_+ is the number of minority (or positive) observations [11]. Specifically, in our case, N_- is the number of both ADLs (activities of daily living) and fall-like, and N_+ is the number of fall instances. Table 1 summarizes the details of the considered datasets.

The URFD dataset was constructed using 2 Microsoft Kinect cameras and 2 accelerometers (PS Move (60 Hz) and x-IMU (256 Hz) devices). All RGB and depth images were synchronized with the corresponding motion data based on timestamp values. Camera 0 was positioned parallel to the floor at an approximate height of 1 m, while camera 1 was configured for ceiling placement 2.5 m above the floor. In addition, accelerometers were strategically mounted on the waist or near the pectoral muscles. URFD contains data from 70 activities (30 falls and 40 daily and fall-like activities) performed by 5 volunteers (actors). The activities (falling while standing and sitting on a chair, walking, sitting, lying on the floor, picking, lifting or placing objects on the floor, tying shoelaces, bending to the left, squatting) took place in conventional indoor spaces such as offices and classrooms). Our study uses features extracted from depth data captured by the cam-0. In total, 11544 samples have been used in our experiments.

The UP-Fall dataset includes 11 activities where measurements were acquired using different sensors (namely, 2 Microsoft LifeCam Cinema cameras, 5 IMUs (inertial measurement units), 1 NeuroSky electroencephalograph (EEG) headset MindWave, and 6 infrared). UP-Fall was collected with the collaboration of

17 volunteers (young healthy men and women) aged between 18 and 24, with an average height of 1.66 m and an average weight of 66.8kg. These volunteers simulated 5 falls and 6 ADLs, with 3 attempts each. UP-Fall combines over 850 GB of body, ambient, and visual sensor data. It encompasses various activities (falling forward with hands, falling forward with knees, falling backward, falling to the side, falling while sitting on a chair, walking, standing, sitting, picking up an object, jumping, and lying down). We used the 294678 available motion samples acquired from the sensors (3-axis accelerometer and 3-axis gyroscope) positioned at the waist. This choice is motivated by the fact that the waist is one of the most appropriate optimal places on the human body to detect falls [22,23].

4.2 Data Preprocessing

We preprocessed data (features and output) as follows (to make it better suited for the used fall detection classifiers): data cleaning, output transformation, and normalization.

- Data cleaning: the correct structuring of data has been validated by ensuring the absence of duplicate, outlying, and missing values.
- Output transformation: fall detection is a binary classification problem (1: fall, 0: ADL/fall-like) [20]. However, in the URFD dataset, the position/status of subjects is labeled into three groups for each measurement: (−1) the person is not lying on the ground, (0) when the person begins to fall, and (1) for post-fall (i.e., the person is lying on the ground). In our study, we replaced the output (−1) with (0) and (0, 1) with (1).
  For the UP-Fall dataset, the output is labeled from 1 to 10. The first five correspond to the different types of simulated falls by the volunteers and the last five correspond to ADLs. We replaced the (1, 2, 3, 4, 5) output with 1 and (6, 7, 8, 9, 10) with 0, where 1 represents falls and 0 represents ADLs and fall-like observations.
- Normalization: we scaled the numerical values of attributes by applying the eight considered data normalization techniques presented in Sect. 3.1. This step aims to improve the quality of features by avoiding the domination of those at a large scale on the model's performance. We have also considered an additional experimental process without data normalization to compare it to the above techniques and hence better study the impact of normalization.

After data preprocessing, we divided each dataset into two subsets, 70% for training and 30% for evaluation (Fig. 2). Thus, we maintained approximately the proportion of output values in the two subsets.

4.3 Hyper-parameters Tuning

Rather than using default (random) values, we adjusted the parameters of each model to achieve high performance. To do so, we defined the range of each hyper-parameter and used an exhaustive search on the grid proposed by scikit-learn [4].

We have also chosen a 5-fold cross-validation in the training data to make the model robust and more general. Table 2 summarizes the configuration of hyper-parameters (range of values, step, number, and type of scale). At the end of this tuning phase, we obtained the optimal parameters for each classifier.

Table 2. Hyper-parameters configuration.

Model	Tuning	Interval	Step	#elements	Scale
SVM-RBF	C gamma	$[2^{-5} : 2^{15}]$ $[2^{-13} : 2^{3}]$	2^2	11 10	Logarithmic
KNN	N_neighbors	[1:40]	1	40	Linear
GNB	Var_smoothing	$[10^{-9} : 10^{2}]$	$10^{0.11}$	100	Logarithmic
DT	Max_depth Min_samples_split Min_samples_leaf	[2:19] [2:19] [1:9]	1	18 18 9	Linear

4.4 Performance Evaluation

To assess the performance of each model, we considered two metrics: *training time* and *accuracy*. The latter is the primary evaluation criterion used in classification problems [5]. It expresses the number of correctly detected examples. For instance, in the fall detection context, it shows the success rate by calculating the fraction of positive (fall) and negative (ADL/fall-like) observations that have been correctly predicted compared to the total number of observations.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

Each classifier has been evaluated using the optimal parameters from GridSearch with the test set. This evaluation has been performed after re-training the optimized model with the training set without cross-validation. Because in practice the parameters resulting from cross-validation are already appropriate for the training dataset [10].

Before presenting the obtained results, we point out that the SVM-RBF model has been excluded from the experiments with the UP-Fall dataset because it requires a massive training time when dealing with large datasets [27].

Accuracy. The obtained results presented in Table 3 and Figs. 3 and 4 show the accuracy of the (SVM-RBF, KNN, GNB, and DT) classifiers (with and without normalization) using the URFD and UP-Fall datasets. When analyzing these results, we notice that the considered data normalization techniques impact the accuracy of classifiers as follows when compared to the non-normalization approach:

Table 3. Accuracy of (SVM-RBF, KNN, GNB, and DT) classifiers (with and without normalization) using the URFD and UP-Fall datasets.

Algorithm	SVM-RBF						KNN						GNB						DT					
Dataset	URFD			UP-Fall			URFD			UP-Fall			URFD			UP-Fall			URFD			UP-Fall		
Optimized parameters| Accuracy|Delta Technique																								
Without normalization	2^{15} 2^{-15}	93.8	-	-	-	-	7	92.1	-	10	88.7	-	0.46	86.4	-	4.64e-5	86.8	-	14 2 2	94.1	-	19 6 2	94.9	-
Min-max	2^{15} 8	96.8	3	-	-	-	1	96.8	4.7	4	96.7	8	2.15	87.3	0.9	0.02	86.9	0.1	14 2 2	94.1	0	19 6 2	94.9	0
Z-score	2^{5} 2	97.3	3.5	-	-	-	1	96.5	4.4	4	96.4	7.7	3.59	87.3	0.9	0.17	86.8	0	14 2 2	94.1	0	19 6 2	94.9	0
Decimal	2^{15} 2^{3}	87.1	-6.7	-	-	-	21	90.4	−1.7	3	96.7	8	0.36	87.3	0.9	0.003	86.9	0.1	15 2 11	91.9	−2.2	16 13 2	94.3	−0.6
Sigmoid	2^{15} 2^{3}	93.4	−0.4	-	-	-	4	95.4	3.3	6	91.5	2.8	0.6	87.1	0.7	1e-9	84.5	−2.3	14 1 7	92.6	−1.5	19 4 2	94.9	0
Tanh	2^{13} 2^{3}	94.5	0.7	-	-	-	4	95.4	3.3	4	92.6	3.9	2.15	86.7	0.3	1e-9	84.4	−2.4	14 1 7	92.6	−1.5	19 2 2	94.8	−0.1
Softmax	2^{-5} 2^{-15}	84.4	-9.4	-	-	-	5	94.6	2.5	14	97	8.3	0.1	84.4	-2	7.74	84.4	−2.4	19 2 5	92.7	−1.4	8 4 1	84.5	−10.4
Maximum absolute	2^{15} 2^{3}	96.7	2.9	-	-	-	1	96.8	4.7	4	96.7	8	1.67	87.2	0.8	0.03	86.9	0.1	14 2 2	94.1	0	19 6 2	94.9	0
Statistical column	2^{15} 2^{3}	94.5	0.7	-	-	-	1	97	4.9	8	93.1	4.4	1	87.2	0.8	4.64e-5	86.7	0.1	14 2 2	94.1	0	19 2 2	94.9	0

- SVM-RBF: accuracy difference from −9.4% to 3.5% in URFD.
- KNN: accuracy difference from −1.7% to 4.9% in URFD and from 2.8% to 8.3% in UP-Fall.
- GNB: accuracy difference from −2% to 0.9% in URFD and from −2.4% to 0.1% in UP-Fall.
- DT: accuracy difference from −2.2% to 0% in URFD and −10.4% to 0% in UP-Fall.

For instance, for the SVM-RBF model, the above percentages mean that data normalization improves its accuracy by 3.5% when compared to that without normalization, which is a significant increase in the healthcare sector. Specifically, the accuracy difference (increase or decrease) when considering each normalization technique for the SVM-RBF model is as follows: (softmax, −9.4%), (decimal, −6.70%), (sigmoid, −0.4%), (statistical column, 0.7%), (tanh, 0.7%), (maximum absolute, 2.9%), (min-max, 3%), and (z-score, 3.5%). The detailed results (optimized parameters, accuracy, and accuracy difference) of each considered normalization technique for the (SVM-RBF, KNN, GNB, and DT) classifiers with both URFD and UP-Fall datasets are depicted in Table 3.

As shown in Table 3 and Figs. 3 and 4, data normalization allows achieving a considerable accuracy improvement for SVM-RBF (3.5% in URFD) and KNN (4.9% in URFD and 8.3% in UP-Fall). In contrast, for GNB, we notice a slight improvement (0.9% in URFD and 0.1% in UP-Fall) and no improvement for DT.

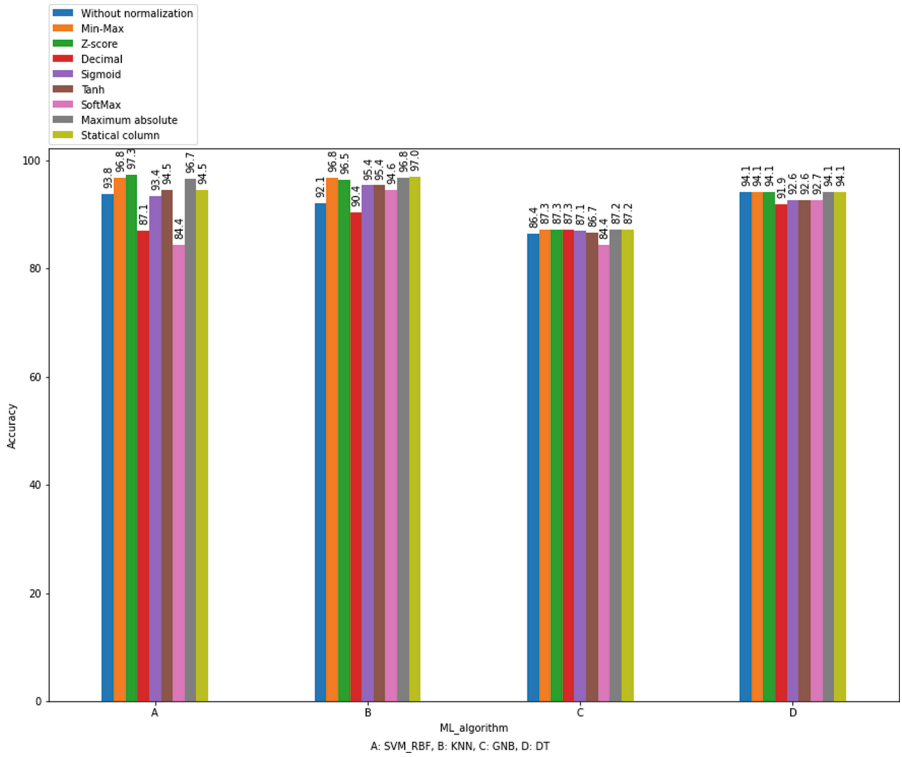

Fig. 3. Accuracy of (SVM-RBF, KNN, GNB, and DT) classifiers (with and without normalization) using the URFD dataset.

Also, note that the accuracy of models with unnormalized data is not always the lowest when compared to that with normalized data. For example, for SVM-RBF in URFD, the accuracy without normalization is 93.8%, and with softmax normalization is 84.4% (i.e., an accuracy decrease of 9.4%). As a second example, for DT in UP-Fall, the accuracy without normalization is 94.9% versus 84.5% for softmax (i.e., a decrease of 10.4%).

Overall, the conducted experiments show that min-max and z-score provide the best accuracy results and that softmax has in most cases the worst performance. This finding can be explained as follows. Data normalization techniques that keep the relationships of original data improve the performance, whereas processes that cause complications in the features (for instance, losing some properties by making values very low) decrease the performance and accuracy of classifiers.

Training Time. To analyze the impact of data normalization on training time, we applied the hard voting algorithm of ensemble learning. We stacked the optimized models with the hyper-parameters shown in Table 3 for each dataset.

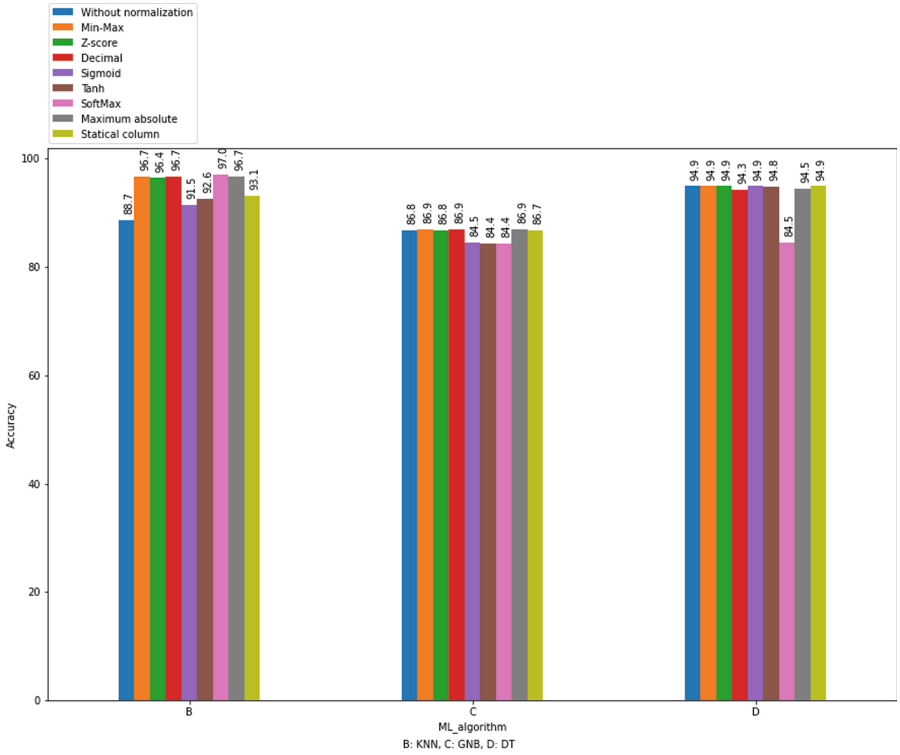

Fig. 4. Accuracy of (KNN, GNB, and DT) classifiers (with and without normalization) using the UP-Fall dataset.

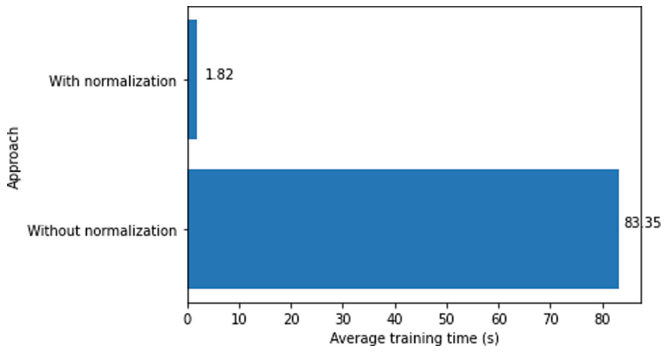

Fig. 5. Average training time of considered models.

Figure 5 shows the average training time on the two considered datasets. Indeed, changing the scale of a model's characteristics improves its training time. The obtained results demonstrate that we need 1.82 s with normalization against 83.35 s without it.

5 Conclusions and Future Work

A model with high accuracy and reasonable training time is more likely to be reliable and efficient in real-world applications. In this paper, we studied the impact of eight data normalization techniques (min-max, z-score, decimal, sigmoid, tanh, softmax, maximum absolute, and statistical column) on the performance (accuracy and training time) of four classifiers (SVM-RBG, KNN, GNB, and DT) of elderly fall detection while considering two datasets of different sizes (URFD and UP-Fall). To have a fair comparison and analysis, we tuned the hyper-parameters of each model using Grid-Search. The obtained experimental results show that data normalization improves the performance of machine learning classifiers that depend on the relationships between data (i.e., SVM-RBF and KNN) compared to those based on probability (i.e., GNB and DT) by increasing the accuracy and decreasing the training time. In particular, the conducted experiments show that min-max and z-score outperform the other normalization techniques in the context of fall detection and that softmax has the worst performance.

In future work, we aim to consider both min-max and z-score normalization to study the impact of different hyper-parameter tuning techniques on the performance of activity classifiers.

Acknowledgements. This work was supported by: The FEDER "European regional development fund" project "Reper@ge" (https://www.europe-bfc.eu/beneficiaire/reperge/).

References

1. Aziz, O., Russell, C.M., Park, E.J., Robinovitch, S.N.: The effect of window size and lead time on pre-impact fall detection accuracy using support vector machine analysis of waist mounted inertial sensor data. In: 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 30–33. IEEE (2014)
2. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(2) (2012)
3. Brownlee, J.: Develop deep learning models on theano and tensorflow using keras. Deep Learning with Python. Jason Brownlee, Melbourne (2018)
4. Buitinck, L., et al.: API design for machine learning software: experiences from the scikit-learn project. arXiv preprint arXiv:1309.0238 (2013)
5. Delahoz, Y.S., Labrador, M.A.: Survey on fall detection and fall prevention using wearable and external sensors. Sensors **14**(10), 19806–19842 (2014)
6. Fayad, M., Mostefaoui, A., Chouali, S., Benbernou, S.: Toward a design model-oriented methodology to ensure QoS of a cyber-physical healthcare system. Computing 1–27 (2022)
7. Galli, S.: Python Feature Engineering Cookbook: Over 70 Recipes for Creating, Engineering, and Transforming Features to Build Machine Learning Models. Packt Publishing Ltd, Birmingham (2020)

8. Gudivada, V., Apon, A., Ding, J.: Data quality considerations for big data and machine learning: going beyond data cleaning and transformations. Int. J. Adv. Softw. **10**(1), 1–20 (2017)
9. Han, J., Kamber, M., Pei, J.: Data Mining: Concepts and Techniques, 3rd edn. The Morgan Kaufmann Series in Data Management Systems, vol. 5, no. 4, pp. 83–124 (2011)
10. Hsu, C.W., Chang, C.C., Lin, C.J., et al.: A practical guide to support vector classification (2003)
11. Huang, L., Zhao, J., Zhu, B., Chen, H., Broucke, S.V.: An experimental investigation of calibration techniques for imbalanced data. IEEE Access **8**, 127343–127352 (2020)
12. Igual, R., Medrano, C., Plaza, I.: Challenges, issues and trends in fall detection systems. Biomed. Eng. Online **12**(1), 1–24 (2013)
13. Islam, M.M., et al.: Deep learning based systems developed for fall detection: a review. IEEE Access **8**, 166117–166137 (2020)
14. Jayalakshmi, T., Santhakumaran, A.: Statistical normalization and back propagation for classification. Int. J. Comput. Theory Eng. **3**(1), 1793–8201 (2011)
15. Kwolek, B., Kepski, M.: Human fall detection on embedded platform using depth maps and wireless accelerometer. Comput. Methods Programs Biomed. **117**(3), 489–501 (2014)
16. Liashchynskyi, P., Liashchynskyi, P.: Grid search, random search, genetic algorithm: a big comparison for NAS. arXiv preprint arXiv:1912.06059 (2019)
17. Martínez-Villaseñor, L., Ponce, H., Brieva, J., Moya-Albor, E., Núñez-Martínez, J., Peñafort-Asturiano, C.: UP-fall detection dataset: a multimodal approach. Sensors **19**(9), 1988 (2019)
18. Merzoug, M.A., Mostefaoui, A., Kechout, M.H., Tamraoui, S.: Deep learning for resource-limited devices. In: Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks, pp. 81–87 (2020)
19. Munappy, A., Bosch, J., Olsson, H.H., Arpteg, A., Brinne, B.: Data management challenges for deep learning. In: 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp. 140–147. IEEE (2019)
20. Nahar, N., Hossain, M.S., Andersson, K.: A machine learning based fall detection for elderly people with neurodegenerative disorders. In: Mahmud, M., Vassanelli, S., Kaiser, M.S., Zhong, N. (eds.) BI 2020. LNCS (LNAI), vol. 12241, pp. 194–203. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-59277-6_18
21. Nooruddin, S., Islam, M., Sharna, F.A., Alhetari, H., Kabir, M.N., et al.: Sensor-based fall detection systems: a review. J. Ambient Intell. Humaniz. Comput. 1–17 (2021)
22. Ntanasis, P., Pippa, E., Özdemir, A.T., Barshan, B., Megalooikonomou, V.: Investigation of sensor placement for accurate fall detection. In: Perego, P., Andreoni, G., Rizzo, G. (eds.) MobiHealth 2016. LNICST, vol. 192, pp. 225–232. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58877-3_30
23. Özdemir, A.T.: An analysis on sensor locations of the human body for wearable fall detection devices: principles and practice. Sensors **16**(8), 1161 (2016)
24. Rastogi, S., Singh, J.: A systematic review on machine learning for fall detection system. Comput. Intell. **37**(2), 951–974 (2021)
25. Rout, N., Mishra, D., Mallick, M.K.: Handling imbalanced data: a survey. In: Reddy, M.S., Viswanath, K., K.M., S.P. (eds.) International Proceedings on Advances in Soft Computing, Intelligent Systems and Applications. AISC, vol. 628, pp. 431–443. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-5272-9_39

26. Shrivastava, R., Pandey, M.: Ensemble of multiple classifiers for accelerometer based human fall detection. In: Smys, S., Palanisamy, R., Rocha, Á., Beligiannis, G.N. (eds.) Computer Networks and Inventive Communication Technologies. LNDECT, vol. 58, pp. 865–874. Springer, Singapore (2021). https://doi.org/10.1007/978-981-15-9647-6_67
27. Syarif, I., Prugel-Bennett, A., Wills, G.: SVM parameter optimization using grid search and genetic algorithm to improve classification performance. TELKOMNIKA (Telecommun. Comput. Electron. Control) **14**(4), 1502–1509 (2016)
28. Turner, S., Kisser, R., Rogmans, W.: Falls among older adults in the EU-28: key facts from the available statistics. EuroSafe, Amsterdam (2015)
29. Usmani, S., Saboor, A., Haris, M., Khan, M.A., Park, H.: Latest research trends in fall detection and prevention using machine learning: a systematic review. Sensors **21**(15), 5134 (2021)
30. Wagner, J., Mazurek, P., Morawski, R.Z.: Regularized numerical differentiation of depth-sensor data in a fall detection system. In: 2017 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA), pp. 234–236. IEEE (2017)
31. WHO: Falls (2021). https://www.who.int/news-room/fact-sheets/detail/falls. Accessed 28 Feb 2023
32. Zheng, A., Casari, A.: Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. O'Reilly Media Inc., Sebastopol (2018)