UNIVERSITÀ DI PISA

DATA MINING II - REPORT

Human Activity Recognition Using Smartphones Data Set

ALESSIO CASCIONE

Contents

Introduction

The analysis carried out in this report focuses on a Human Activity Recognition database counting a total of 10299 instances (divided in 7352 train observations and 2947 test samples) coming from 30 subjects performing six different daily activities while carrying a smartphone with inertial sensors. Sensor signals were pre-processed using a sliding windows of 2.56 seconds with 50% overlap, leading to time-series with 128 time-stamps each. We will take into account two different data-sets coming from the same source: a data-set of 561 features, considering triaxial acceleration, angular velocity from the gyroscope and effects of gravity in order to perform anomaly discovery and a multinomial classification task having the aim of identifying the daily activity a subject is performing; a time-series data-set over which we mainly perform a classification task and a clustering task. The data-set is already offered with a division in terms of training and test set: we will be using for most of the analyis 7352 observations (7279 after the outlier removal procedure) as training data and the remaining 2947 as test. The same division is followed for the time-series analysis, too. For most of the analysis, especially for classification purposes, all the continuous features offered in the data-set will be used[1].

1 Anomaly Detection

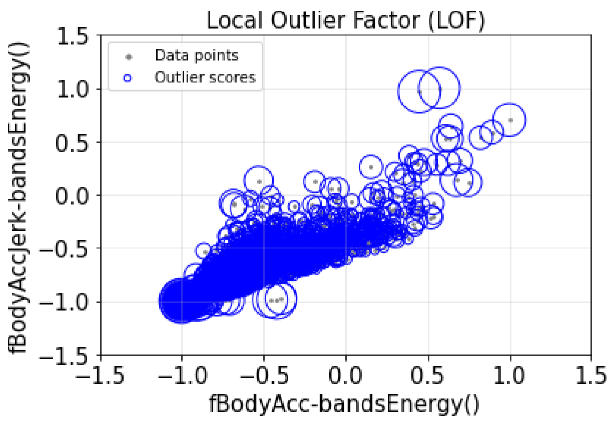
In order to identify the top 1% of outliers in the original data-set, we focused on different anomaly detection techniques, mostly using score metrics in order to evaluate the "degree of anomaly" of a single point. For each methodology used, we report the number of outliers identified and the distribution of the target class identified for the multi classification problem.

1.1 Local Outlier Factor and Grubbs' method

We exploit Local Outlier Factor (LOF) as a density-based approach for anomaly detection in the data-set, computed as the average ratio of local reachability distances of a point's neighbors with respect to the point itself. We underline here that increasing the number of neighbors over 40 to check in order to determine the LOF of the points does not relevantly affect the results, given a chosen contamination threshold of 0.1. Hence we decided to present here results considering a total number of 40 neighbors: in particular we observed a total number of 736 outliers, with a maximum outlier factor of -1.1287135909324377 and a minimum of -1.8439601601538222, identifying the "most inlying" and the "most outlying" points among the anomalies. To emphasize the result, we visualize the level of outlierness of the points in a scatter-plot[2]:

[1]Deeper explanation about features extracted from the experiments and time-series information are available in *https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones* - last check on 31/08/22.

[2]A recursive feature elimination methodology was exploited here to define a smaller set of features among which choosing the best pair to plot the points for the visualization. Further details regarding this methodology will be presented in the next sections related to the solution of a simple linear regression task and dimensionality reduction techniques.



We specify here that most of the outliers found by LOF belong to more static labels (85%) and the remaining ones to more dynamic ones, such as "Walking upstairs" and "Walking downstairs". Considering more specifically the two variables chosen for the visualization, we underline that we also employed a Grubbs' statistical method for the identification of outliers for those two specific variables. The results are not so reasonable in this case, with only 16 outliers isolated in "fBodyAccJerk-bandsEnergy()" and 2 outliers in "fBodyAcc-bandsEnergy()", given the fact that both features, as many others in the data-set, are not normally distributed. Hence we will consider less statistically oriented test for further anomaly analysis.

1.2 Angle-Based Outlier Degree and Isolation Forest

We employ a fast version of the Angle-Based Outlier Degree (ABOD) approach using k-Nearest Neighbors to reduce the sample of points directly taken into account, in this case using the same amount of neighbors exploited for LOF. We also set a default contamination threshold of 0.1, the same employed for LOF. The fast procedure leads to a total of 726 outliers, with a maximum score of $6.26e-7$ and a minimum of $5.02e-4$, specifying that lower scores express a higher degree of outlierness. In this context we have a majority of points belonging to more dynamic labels (67%) rather then static ones (33%), quite differently from LOF. On the other hand, in order to rank how anomalous a point is, Isolation Forest exploits the number of splitting required to isolate a certain observation with respect to the tree built through the random selection of vectors of a sample of the data-set: in this case we used a random selection of half of the points and a total of 200 estimators in order to identify the anomalies, keeping the same contamination threshold of the previous methods. Isolation Forest identifies 736 anomalies, 22% belonging to static labels and the rest to more dynamic ones. We can compare the results with the ones of an Extended Isolation Forest algorithm in order to see if any difference arises: this latter approach leads to a total of 624 outliers shared with the standard isolation forest, with a total of 12% of static labels against 88% of dynamic ones.

1.3 *k*-Nearest Neighbors

We exploit *k*-Nearest Neighbors as an anomaly discovery method, considering for each point its distance to its *k*th-nearest neighbor as an outlying score. Setting a reference number of neighbors of 40 and keeping the contamination threshold at 0.1, we obtain a total of 722 outliers (with a minimum outlying score of 5.32 and a maximum of 14.75), having in total 65% of points with a static label and 35% of dynamic ones. In the next sections a comparison between results of various methods will be further analyzed.

1.4 Outliers discovery comparison

Having applied different techniques in order to identify anomalies in the original data-set, we perform here a comparison among results with the aim of finding the top 1% of outliers. The following table specifies the cardinality of respective pair-wise intersections between the sets of outliers isolated by all the different approaches listed so far:

	LOF	ABOD-fast	kNN	IF	EIF
LOF	-	306	294	223	175
ABOD-fast	306	-	515	468	441
kNN	294	515	-	569	538
IF	223	468	569	-	631
EIF	175	441	538	631	-

It is evident that the LOF methodology is the most different in terms of results when performing a comparison of the amount of common anomalies recognized, especially with respect to Extended Isolation Forest. Other methods seem to show higher consistency in their pair-wise intersection, allowing an easier identification of the most outlying points. We are now in the position to remove the top 1% of anomalies from the original data-set employed, given the balanced nature of the data-set with respect to the activities performed by subjects: in order to do this we exploit the score assigned to each outlier for every single method, collecting for each anomaly result the top 3% values considering the score of the associated methodology, obtaining a total to 73 points to remove from the original data-set used for anomaly detection, thus obtaining 7279 observations to keep working with in next tasks: the final anomalies present 53% of "Walking downstairs" instances and 23% of "Laying" elements, these two being the labels associated with observations that tend to be more distant from the rest of the values.

2 Multinomial Classification Analysis

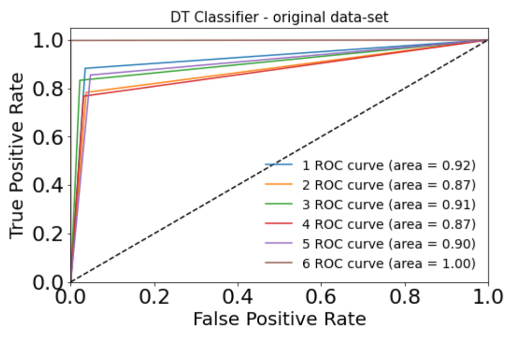
The following sections show results for different classification methods applied on both the original and modified versions of the data-set: in some cases, dimensionality reduction techniques or imbalanced learning approaches are employed for a deeper understanding of the model in question. The classification problem concerns the identification of an activity performed by a person given a set of features capturing variations in subjects' movements[3].

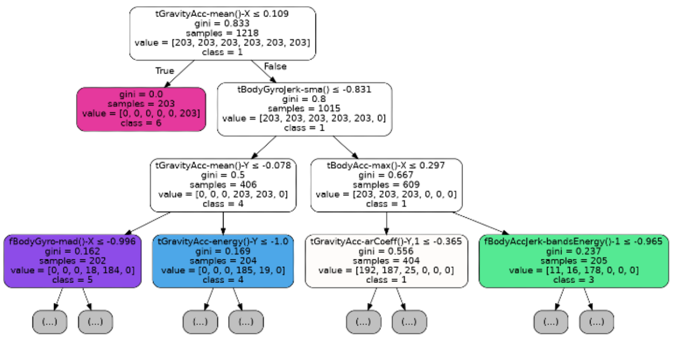
2.1 Decision Tree and *k*-Nearest Neighbors

We first consider two simple algorithms for the solution of the classification task: Decision Tree (DT) and *k*-Nearest Neighbors (*k*-NN). In order to isolate the best set of parameters for the DT classifier, we perform a grid search considering different possible depth-level limits (from none to 4) and the minimum number of samples to split a node and to consider a node a leaf (2,5,10 and 20 were the parameters considered for the estimation). The parameters allowing us to reach to best results on the original data-set were the following ones: 1 as minimum samples for a leaf, 2 as minimum samples for a split and no max depth. We follow the same approach for *k*-NN, performing a grid search considering neighborhoods' cardinalities going from 1 to 30 and two possible way of evaluating the relevance of the points in the neighborhood: considering a uniform assignment of weights or weighting points in the neighborhood according to the inverse of their distance. This last weight assignment method, with a number of neighbors of 20, allowed us to reach the the best results for the original data-set. We report here results both for both classifiers[4]. We also offer a visualization for the best DT reported inside the table, which emphasizes the role of gravity related attributes in order to perform significant splits.

DT Classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.82	0.83	0.86	0.84	0.80	1.00
Recall	0.92	0.76	0.82	0.77	0.86	1.00
F1-score	0.96	0.79	0.84	0.80	0.83	1.00

k-NN Classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.85	0.90	0.96	0.93	0.84	1.00
Recall	0.98	0.91	0.78	0.80	0.94	1.00
F1-score	0.91	0.90	0.86	0.86	0.89	1.00




tGravityAcc-mean()-X ≤ 0.109
gini = 0.833
samples = 1218
value = [203, 203, 203, 203, 203, 203]
class = 1
True
False
gini = 0.0
samples = 203
value = [0, 0, 0, 0, 0, 203]
class = 6
tBodyGyroJerk-sma() ≤ -0.831
gini = 0.8
samples = 1015
value = [203, 203, 203, 203, 203, 0]
class = 1
tGravityAcc-mean()-Y ≤ -0.078
gini = 0.5
samples = 406
value = [0, 0, 0, 203, 203, 0]
class = 4
tBodyAcc-max()-X ≤ 0.297
gini = 0.667
samples = 609
value = [203, 203, 203, 0, 0, 0]
class = 1
fBodyGyro-mad()-X ≤ -0.996
gini = 0.162
samples = 202
value = [0, 0, 0, 18, 184, 0]
class = 5
tGravityAcc-energy()-Y ≤ -1.0
gini = 0.169
samples = 204
value = [0, 0, 0, 185, 19, 0]
class = 4
tGravityAcc-arCoeff()-Y,1 ≤ -0.365
gini = 0.556
samples = 404
value = [192, 187, 25, 0, 0, 0]
class = 1
fBodyAccJerk-bandsEnergy()-1 ≤ -0.965
gini = 0.237
samples = 205
value = [11, 16, 178, 0, 0, 0]
class = 3
(...)

[3] In some plots, for visualization purposes, we will swap classes' names with numeric identifiers in the following way: 1 - "Walking", 2 - "Walking upstairs", 3 - "Walking downstairs", 4 - "Sitting", 5 - "Standing", 6 - "Laying".
[4] DT Classifier reaches an accuracy of 0.86, *k*-NN of 0.90.

We try to investigate the two algorithms' performances over an imbalanced version of the same data-set obtained by randomly removing points belonging to more dynamic labels, the ones related to walking activities, obtaining a final data-set with 15% of points belonging to walking labels and 85% of points to the other three activities. We then present results obtained by the two classifiers on different versions of the unbalanced data-set corrected with two techniques: random undersampling and SMOTE. We underline that an additional unbalanced technique used has been Condensed Nearest Neighbour, but we omit results for this one due to very poor overall performances with both unbalanced techniques.

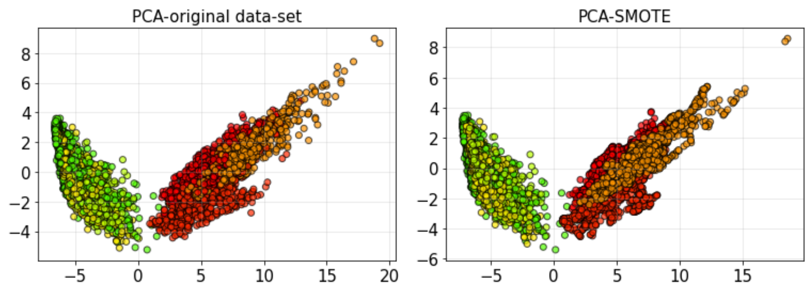
DT - RU	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.82	0.76	0.82	0.82	0.82	1.00
Recall	0.91	0.79	0.68	0.80	0.84	1.00
F1-score	0.86	0.78	0.74	0.81	0.83	1.00

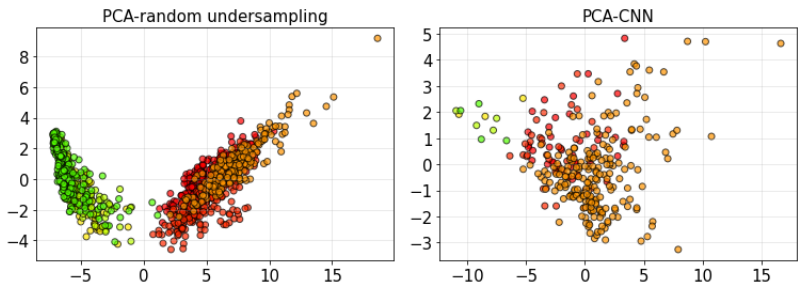
k-NN RU	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.82	0.89	0.96	0.87	0.79	1.00
Recall	0.98	0.90	0.76	0.74	0.91	0.98
F1-score	0.90	0.89	0.85	0.80	0.84	0.99

DT - SMOTE	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.84	0.85	0.83	0.84	0.80	1.00
Recall	0.92	0.79	0.80	0.77	0.87	1.00
F1-score	0.88	0.82	0.81	0.80	0.83	1.00

k-NN SMOTE	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.87	0.88	0.89	0.90	0.86	1.00
Recall	0.96	0.87	0.79	0.83	0.91	1.00
F1-score	0.91	0.88	0.84	0.86	0.89	0.99

For visualization purposes, we complete the previous table with plots of the respective data-sets used for the deeper analysis after having applied a Principal Component Analysis reducing the features to 2 main components.





2.2 Support Vector Machine

Support Vector Machines (SVM) allow us to divide the observations exploiting an hyperplane according to the kernel considered for the classification. As a pre-processing step we first performed a normalization on the data-set using a standard scaling approach, removing the mean and scaling to unit variance. We then performed the classification with different regularization parameters[5] and linear, poly and rbf kernels[6] for different hyperplane configurations, considering the best overall results. Hence we report here the metrics for the best linear and the best rbf and poly classifier identified: the first table reports the result for the best linear classifier with a regularization C parameter of 1; the second the best poly classifier with a gamma of 0.01 and a C of 1; the third the best rbf classifier with a gamma of 0.0001 and a C of 1000;[7].

SVM classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.86	0.94	0.76	0.50	0.72	0.88
Recall	0.92	0.77	0.98	0.75	0.19	1.00
F1-score	0.89	0.84	0.85	0.60	0.30	0.94
Precision	0.59	0.65	0.78	0.70	0.70	0.77
Recall	0.91	0.82	0.85	0.22	0.43	0.92
F1-score	0.72	0.73	0.81	0.34	0.53	0.84
Precision	0.93	1.00	0.17	0.95	0.79	0.28
Recall	0.16	0.04	0.08	0.32	0.83	1.00
F1-score	0.27	0.07	0.11	0.48	0.81	0.43

Results for the "Laying" label are optimal in every cases, being a class that relevantly differers from the rest. The overall perfomances on the original test set of the different classifiers is extremely similar and rather unsatisfactory. For further analysis, we also report here results of a linear SVM classifier over the imbalanced set corrected with a random undersampling approach, discussed in previous sections.

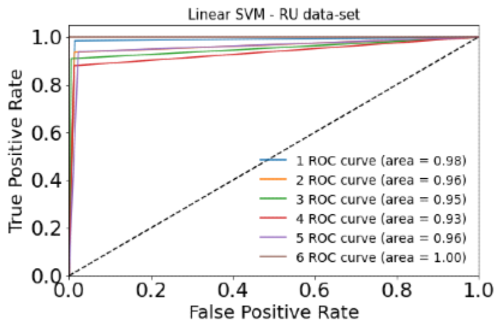
[5]We considered the following parameter regularization: 0.001, 1, 25, 70, 100, 1000.

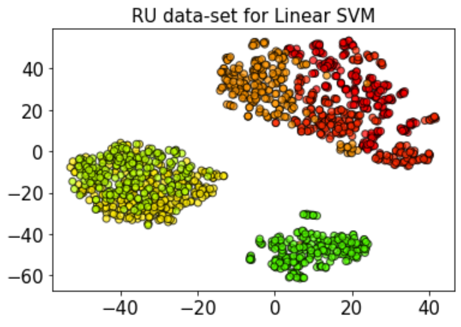
[6]The following are the different kernel coefficients considered for the last two approaches: 0.01, 0.001, 0.0001. We also specify here that for the multi-class classification problem a "one-vs-rest" strategy was adopted.

[7]For the sake of completeness, we report here the accuracy scores for linear, poly and rbf kernels: 0.75, 0.68, 0.43.

SVM classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.89	0.89	0.75	0.92	0.86	0.93
Recall	0.95	0.71	0.97	0.74	0.86	1.00
F1-score	0.92	0.79	0.85	0.82	0.86	0.96

The performances of the linear classifier over the unbalanced data-set corrected with random under-sampling are also sufficient but not satisfactory and more or less similar to the ones obtained considering the original data-set.





2.3 Naïve Bayes Classifier

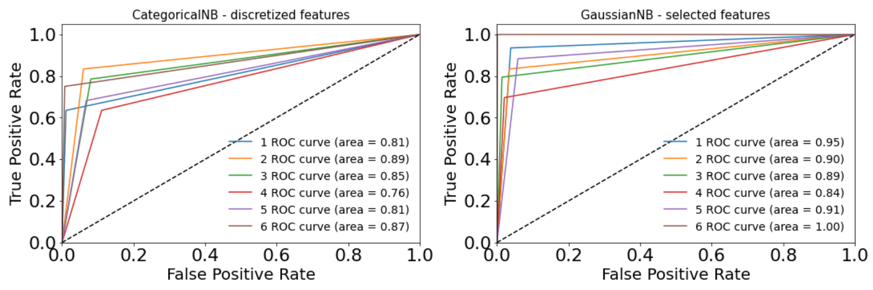
Naïve Bayes Classifiers (NB) are approaches for solving the classification task exploiting Bayes' theorem with the assumption of a strong independence between the features used for the label assignment procedure. We present here performances coming from three different kinds of NB Classifiers. In the following table, the first results come from a Gaussian classifier, one that considers the likelihood of a feature to be Gaussian[8]; the second results come from a Categorical NB Classifier applied on the original features discretized in 4 bins each; the last ones are results obtained by the first Gaussian classifier using a sub-set of 20 original features, chosen after having performed a recursive feature elimination procedure using a Random Forest as estimator[9]. We underline here that the algorithm assumes a complete independence among the variables and this requirement is not entirely respected.

[8]This implies that the probability an instance x_i has of belonging to class y is computed as follows: $P(x_i \mid y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$.

[9]The respective values of accuracy scores for each of the three classifier are the following: 0.77, 0.71, 0.86.

NB Classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.84	0.75	0.82	0.58	0.82	0.99
Recall	0.83	0.96	0.66	0.89	0.36	0.98
F1-score	0.84	0.84	0.73	0.70	0.50	0.99
Precision	0.92	0.73	0.62	0.53	0.69	0.96
Recall	0.64	0.83	0.79	0.64	0.68	0.75
F1-score	0.75	0.78	0.69	0.58	0.69	0.84
Precision	0.83	0.82	0.91	0.87	0.77	1.00
Recall	0.94	0.83	0.80	0.70	0.88	1.00
F1-score	0.88	0.83	0.85	0.77	0.82	1.00

Generally good performances seem to be reached considering the Gaussian method applied on a sub-set of the original features, allowing to reach perfect precision and recall for the "Laying" label and never going under 0.70 for these two score values considering the other classes. For visualization purposes, we report here two ROC-curve graphs for the Categorical classifier and the Gaussian applied on a sub-set of the original features.



2.4 Neural Network

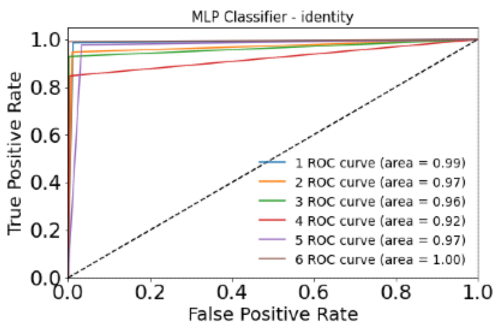
2.4.1 Multilayer Perceptron

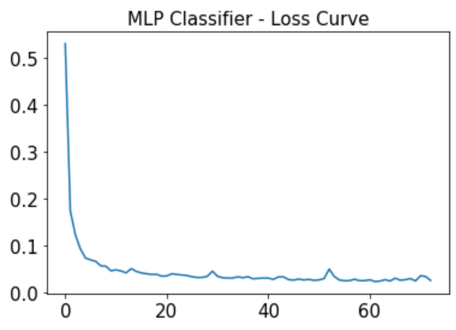
We first employ a Multilayer Perceptron as a neural network classifier (MLP) testing different possible combinations of main parameters: first of all we try different sizes of two hidden layers beside the input and output one, testing the following pairs of layers cardinalities: 23, 43; 64, 128; 32, 128; 128, 64[10]. Then we have considered different possible activation functions, determining how the weighted sum of the input is transformed into an output from a node or nodes: we considered identify function, logistic and sigmoid function, hyperbolic tangent function and rectified linear unit function. We also tested as solvers both a stochastic gradient descent and adam, as a stochastic gradient-based optimizer. We also considered three different possible initializations for the learning rate (0.001,0.01,0.1), keeping it constant or adaptive for different tests, also changing the momentum for gradient descent update with 0.1,0.5 and 0.9.

[10]We limit ourselves to the analysis of 2 hidden layers because increasing them does not result in significat gains in terms of models' performances.

As in the previous sections, we report here the three best results reached, specifying the combination of parameters leading to best performances. The first model uses identity as activation function, adam as solver and considers a constant learning rate with 32 as the size of the first hidden layer and 128 for the second; the second still adopts adam as solver, but with an adaptive learning rate and with rectified linear unit function as activation function with 64 as the size of the first hidden layer and 128 for the second; the last one applies a logistic sigmoid function with a stochastic gradient descent solver with a constant learning rate, adopting the same hidden layers size of the first model[11].

MLP Classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.94	0.94	0.98	0.97	0.87	1.00
Recall	0.99	0.95	0.93	0.85	0.98	0.99
F1-score	0.96	0.95	0.96	0.91	0.92	1.00
Precision	0.95	0.96	0.98	0.95	0.87	1.00
Recall	0.99	0.95	0.94	0.88	0.96	0.96
F1-score	0.97	0.95	0.96	0.91	0.91	0.98
Precision	0.91	0.92	0.92	0.84	0.70	1.00
Recall	0.97	0.92	0.85	0.56	0.95	0.96
F1-score	0.93	0.92	0.88	0.67	0.80	0.98



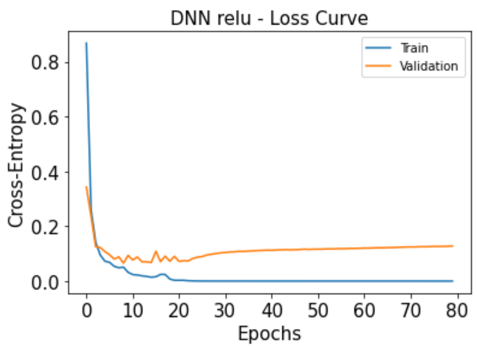


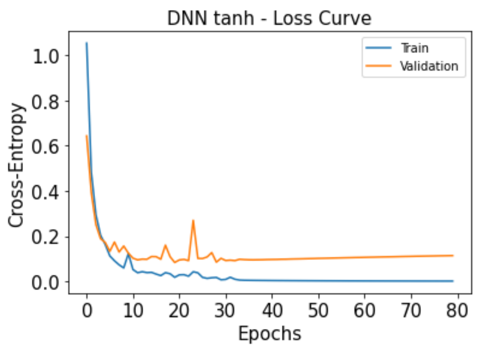
2.4.2 Deep Neural Network

In this context we consider how applying a Deep Neural Network classification approach (DNN) affects results for the multinomial task[12], focusing on models having 5 hidden layers (256,128,64,32,16 in size respectively) and applying a batch size of 80. Parameters were chosen in order to minimize tendencies of overfitting, comparing loss graphs for validation and test sets.

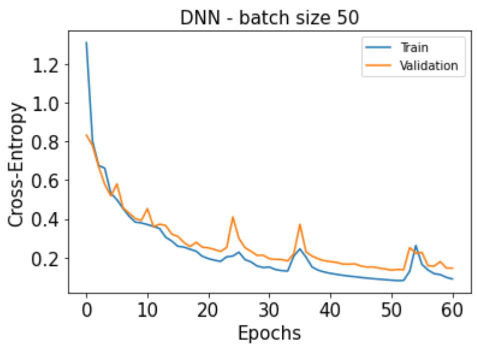
[11]We report here that the first two models share the same level of accuracy of 0.94. The last model performances are slightly worse, with 0.87 of accuracy.

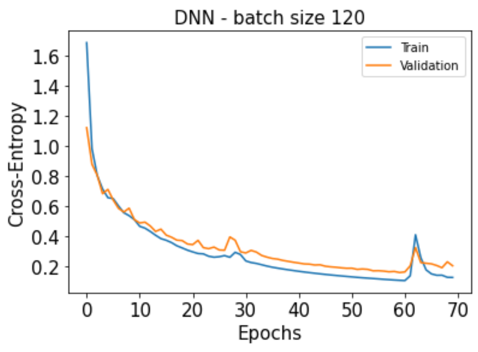
[12]We use the 20% of the training set as a validation size for the training procedure.





We then consider performances of analogous models using an L2 regularization approach and introducing an early stopping criterion to evaluate how many epochs are needed for best performances. Trying different values for L2 parameter and different batch sizes, the best results are given by models that employ relu activation function and present a regularization value of 0.001, keeping the same configuration of hidden layers used in the previous analysis. For these models, both starting with 130 epochs, the stopping criterion determined an early stop on the 60th epoch for the first classifier and on the 70th for the second. The first DNN with L2 equals to 0.001 employed a batch size of 50, the second a batch size of 120[13]. We also specify precision and recall scores focusing on the first one.





DNN - batch size 50	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.95	0.93	0.98	0.95	0.91	1.00
Recall	0.97	0.97	0.91	0.89	0.95	1.00
F1-score	0.96	0.95	0.95	0.92	0.93	1.00

DNN - batch size 120	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.96	0.93	0.93	0.89	0.92	1.00
Recall	0.93	0.95	0.95	0.93	0.89	0.97
F1-score	0.95	0.94	0.94	0.91	0.90	0.99

[13]The two models share very similar accuracy scores: 0.95 and 0.93 respectively.

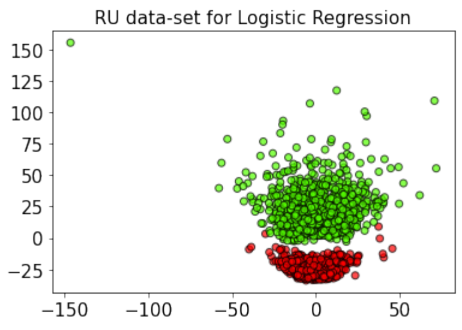
2.5 Logistic Regression

2.5.1 Classification through Logistic Regression

We adopt a Logistic Regression methodology in order to solve our multi-nomial classification task[14]. We apply a liblinear solver on a standardized version of the data-set by removing the mean and scaling to unit variance and consider two different models, one assigning to each class a weight of one and another model that follows a "balanced" approach, adjusting weights in an inversely proportional way with respect to class frequencies. Both approaches reach very similar results, with an accuracy of 0.78, an unsatisfactory result if compared to other kind of classifiers' performances. Very similar results are obtained with the "balanced" methodology, hence the following table reports only the outcome coming from this last model.

Logistic Regression	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	1.00	0.78	0.70	0.65	0.73	0.97
Recall	0.34	0.84	0.97	0.62	0.96	0.95
F1-score	0.51	0.81	0.81	0.63	0.83	0.96

We could even consider a different classification problem by exploiting the unbalanced version of the data-set, correcting it with Random Undersampling and transposing the multi-class problem into a binary one, consider more static labels vs more dynamic ones. We also offer a visualization of the modified undersampled data-set using Multi-Dimensional Scaling with two components.



The results demonstrate in general poor performances even for this simpler task, with an accuracy of 0.49[15], making Logistic Regression the weakest classification model for our analysis.

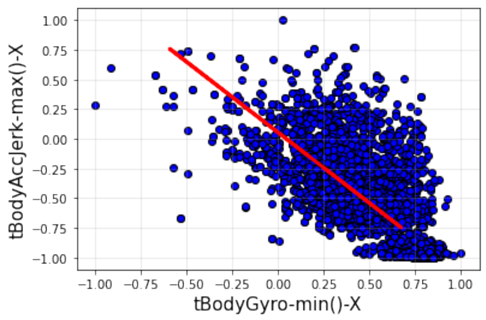
[14]Here we follow a "one-vs-rest" strategy, analogous to the one already employed for SVM classification, in which multiple models are trained and after that we collectively use those classifiers to perform the prediction.

[15]We underline that for most of the other classifiers, results for the binary classification task tend to be much more satisfactory, reaching almost perfect accuracy scores for many models. This emphasizes the weakness of the Logistic Regression approach for the data-set under analysis.

Logistic Regression - unbalanced	Static	Dynamic
Precision	0.99	0.48
Recall	0.05	1.00
F1-score	0.10	0.65

2.5.2 Linear Regression, Multiple linear regression and Gradient Boosting

We can focus our analysis on a sub-set of the original features in order to choose a satisfactory pair of variables in order to solve a simple linear regression task and plot the relative results. To do so, we use recursive feature elimination employing Decision Tree Regressors, splitting according to mean squared error and imposing a max depth of 4 for the trees. Hence , we report the evaluation metrics and the plot for the most interesting couple found, using 132 as independent variable and 89 as dependent variable: we visualize the outcome of the regression below.



We also try to solve a multiple linear regression problem applying a linear model trained with L1 prior as regularizer (Lasso) and a model that considers least squares with L2 regularization (Ridge), considering every continuous feature and then the best 30 according to their ANOVA F-value. In the Lasso case this does not really affects results given the fact that just one feature is considered relevant for the regression; for the Ridge one, we get different outcomes. For the same task, we apply a Gradient Boosting approach with 50 and 100 estimators and a learning rate of 1.0 with the aim of comparing results with the Logistic Regressors' ones in the table below.

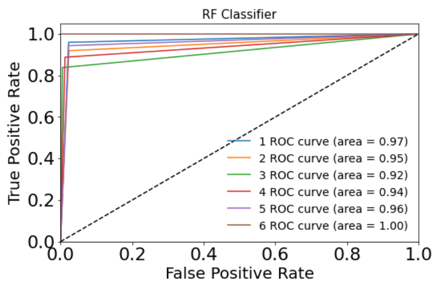
Logistic Regression	Lasso	Ridge	Ridge - feature selection	Gradient Boosting - 100 est.	Gradient Boosting - 50 est.
R2	0.131	0.948	0.901	0.939	0.942
MSE	2.631	0.157	0.301	0.185	0.176
MAE	1.395	0.286	0.440	0.275	0.262

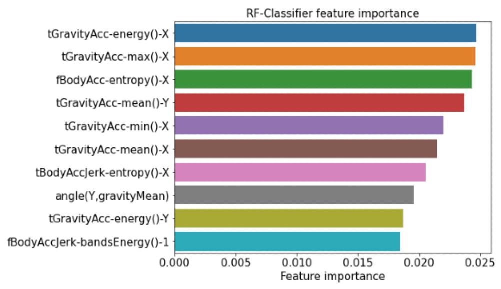
2.6 Ensemble method

Ensemble classification methods are approaches that exploit multiple classifiers (namely, an ensemble of them) in order to combine their predictions for the classification of unseen instances using some form of voting. We start with a Random Forest (RF) classification approach, which trains different Decision Trees considering a bootstrap sample of the original data-set with respect to an independent set of random vectors and then outputs the label identified as the mode of class's output by single trees. For this task, we consider different combinations of parameters for the single decision trees defined: we consider two possible criterion for splitting, Gini index and entropy, different number of estimators (30,50,100), different minimum number of samples in a node to be considered a leaf (1, 5, 10, 20) and different minimum number of samples to perform a split (2, 5, 10, 20). We report here the best result obtained with respect to the entropy criterion and the best considering the Gini index, both taking into account the original data-set. For the RF Classifier using entropy, best results were achieved considering 100 estimators, 1 as minimum value of samples for a leaf and 20 as minimum samples to perform a split; for the RF Classifier using gini, we considered 100 estimators, 10 as minimum value of samples for a leaf and 2 as minimum samples to perform a split[16].

RF Classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.90	0.88	0.96	0.94	0.90	1.00
Recall	0.96	0.92	0.84	0.89	0.94	1.00
F1-score	0.93	0.90	0.90	0.91	0.92	1.00
Precision	0.88	0.90	0.96	0.93	0.89	1.00
Recall	0.97	0.91	0.83	0.87	0.94	1.00
F1-score	0.92	0.90	0.89	0.90	0.91	1.00

We also specify here the ROC-curve and the feature importances for the RF using entropy as splitting criterion, emphasizing the role of gravity related features in order to make a split, similarly to what we had previously observed with the simple Decision Tree.





Then we consider results obtained through the employment of a Bagging classification methodology, which fits classification models on different random sub-sets of the data-set and then aggregate their results through a voting procedure in order to form a final prediction. For this task, we consider the Bagging results on the original data-set of a simple Linear SVM with a parameter regularization C=1, a Bagging of *k*-Nearest Neighbors classifiers with *k*=20 weighting points by the inverse of their distance and results coming from a Bagging of Gaussian NB Classifiers applied on the previously identified

[16]The entropy classifier leads to a 0.92 accuracy score, the gini one 0.91.

20 features isolated through recursive feature elimination. Our objective is to evaluate if the Bagging procedure is able to enhance previous classifiers' performances and quantify the potential gain in terms of accuracy.

SVM Bagging	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.96	0.98	1.00	0.98	0.90	1.00
Recall	1.00	0.96	0.98	0.88	0.98	1.00
F1-score	0.98	0.97	0.99	0.92	0.94	1.00

In the table above we report the results for the procedure that seems to affect the final outcome the most. It is interesting to notice that a Bagging approach for the linear SVM leads to far better outcome then the simple one using just one model, and gives us one of the best classification results for this data-set so far, having an accuracy of 0.96 and presenting perfect precision for "Walking downstairs" and perfect recall for "Walking"; on the other hand, a Bagging approach for both the Gaussian NB Classifier and the *k*-NN Classifier does not relevantly improve the overall accuracy and the single precision and recall scores for the various classes, hence results for these last approaches were omitted.

In the end we also report results relative to the employment of an AdaBoost approach, fitting a classifier on the original dataset and then fitting additional copies of the classifier on the same data-set where the weights of incorrectly classified instances are adjusted in order to let the successive classifier focus on instances "harder" to classify. In this case we test an "ensemble of ensemble" method, using as base model for AdaBoost a Random Forest trained according to the best parameters for the entropy criterion defined above. Results are shown in the table below.

AdaBoost RF	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.91	0.90	0.96	0.95	0.91	1.00
Recall	0.97	0.92	0.85	0.90	0.96	1.00
F1-score	0.94	0.91	0.90	0.92	0.93	1.00

The AdaBoost approach leads to a slightly higher accuracy level (0.94) and gives better result for the "Walking" and "Walking downstairs" labels.

2.6.1 Gradient Boosting

Gradient Boosting is a further ensemble methodology based on different instantiations of Decision Trees in order to update the prediction of a sample's class with the aid of the optimization of a differentiable loss function: we report here the best performances of a gradient boosting classifier for our multi-class problem, considering 50, 100 and 200 estimators with three possible different learning rates (0.3,0.9,1.0) and three different possible tree max depths (3,6,12). We present the best results obtained first considering the original data-set with every features taken into account and then the best ones obtained by isolating the first 80 attributes selected according to their ANOVA F-value[17]. We also include performances of a Gradient Boosting classifier trained on an imbalanced version of the data-set corrected with a Condensed Nearest Neighbour approach, considering 5 neighbors as parameters and with 341 resulting samples, with a learning rate of 0.3 and a tree depth of 6 : the best classifier considering the entire set of features is given by the smallest learning rate considered and a depth of 6; same parameters are considered for the performances on the smaller data-set[18]. The table reports results respectively coming from the standard GB classifier, the one applied on the data-set with feature selection and the one applied on the imbalanced data-set.

GBoost Classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.95	0.92	0.98	0.92	0.88	1.00
Recall	0.97	0.94	0.93	0.87	0.94	1.00
F1-score	0.96	0.93	0.95	0.89	0.91	1.00
Precision	0.90	0.88	0.94	0.82	0.80	1.00
Recall	0.92	0.91	0.88	0.77	0.85	1.00
F1-score	0.91	0.90	0.91	0.79	0.82	1.00
Precision	0.82	0.70	0.72	0.56	0.54	0.98
Recall	0.68	0.73	0.93	0.18	0.84	0.95
F1-score	0.74	0.71	0.81	0.27	0.66	0.96

It is interesting to notice how the "Laying" label is perfectly recognized in both cases, while the worst results are linked with the "Sitting" label, an evident phenomenon also shown with the previous used classifiers.

After having considered the performances of a standard Gradient Boosting approach, we take into account two types of optimization techniques for the same methodology: we evaluate the performances of an eXtreme Gradient Boosting approach (XGB) for the same multi-class classification task, building XGBoost trees that consider similarity scores inside the leaves to perform further splits. We take into account different values for the learning rate parameter (0.3,0.6,1.0), using softmax as objective for the multi-class resolution and different depths of the trees (6,16,32). We compare this approach with a Light Gradient Boosting approach (LGBM), which chooses the leaf with maximum delta loss to grow in order to split the tree. In both cases, the best results are achieved keeping a learning rate of 0.3 and the lowest possible depth of the tree used: the LGBM Classifier reaches slightly better results then the XGBoost one, the former having 0.94 as accuracy score and the latter 0.93. In both cases, precision and recall scores are quite high, all of them being always above 0.80. We also specify that increasing the depth of the trees taken into account strongly worsens the results for the LGBM approach, even leading to accuracy scores lower then 0.40 with the 32-depth case.

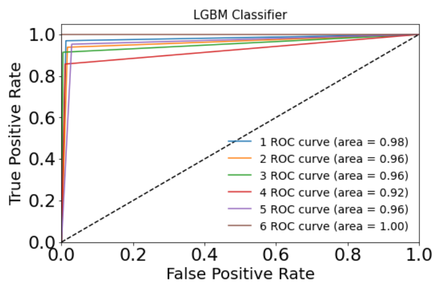
[17]This being the minimum amount of features found in order to reach an accuracy higher then 0.85 for our GB Classifier.

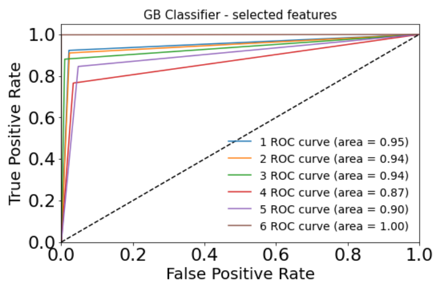
[18]The first kind of classifier reaches 0.94 accuracy, the second, considering less features, 0.89, the third, applied on the imbalanced data-set with CNN, 0.71.

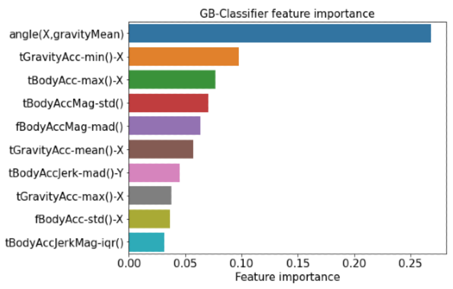
XGBoost Classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.93	0.93	0.97	0.93	0.88	1.00
Recall	0.98	0.93	0.92	0.85	0.94	1.00
F1-score	0.95	0.93	0.94	0.89	0.91	1.00

LGBM Classifier	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.94	0.93	0.97	0.94	0.88	1.00
Recall	0.97	0.93	0.93	0.85	0.95	1.00
F1-score	0.96	0.93	0.95	0.90	0.91	1.00

The two ROC-curves below show results for the LGBM Classifier and the standard GB Classifier applied on the sub-set of features chosen. For the last classifier, we also include the 10 most important features identified by the method, emphasizing the importance of variables related to measures of body and gravity acceleration for the final evaluation.

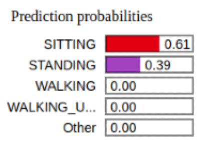


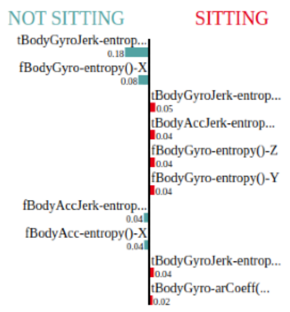


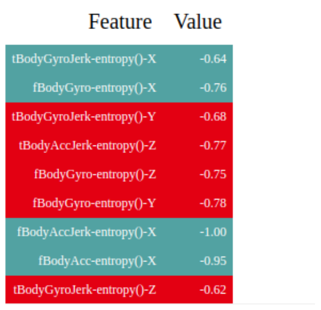


3 Explaination methods

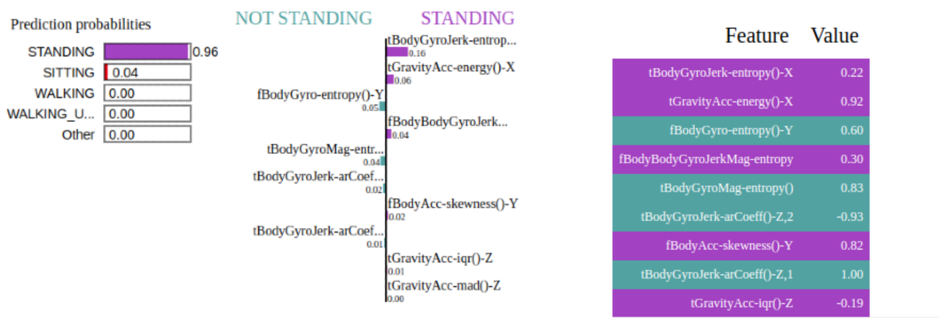
We use the Local Interpretable Model-Agnostic Explanations (LIME) approach in order to evaluate features' relevance for the classification of specific observations. As a reference model to explain, we choose the Bagging of SVM having the best reported performances. Two records classified according to the labels having the lowest F1-score in terms of classification performance ("Sitting" and "Standing") have been arbitrary chosen in order to apply LIME. We offer tables and visualizations for both records:





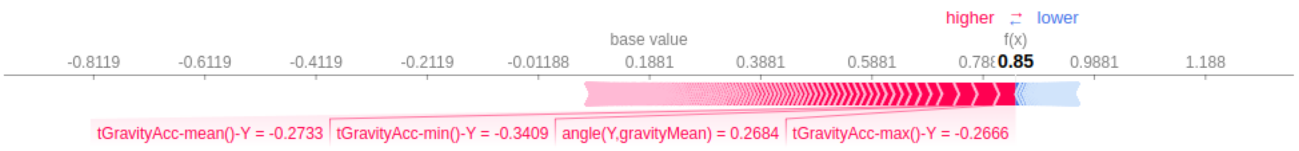


Feature	Value
tBodyGyroJerk-entropy()-X	-0.64
fBodyGyro-entropy()-X	-0.76
tBodyGyroJerk-entropy()-Y	-0.68
tBodyAccJerk-entropy()-Z	-0.77
fBodyGyro-entropy()-Z	-0.75
fBodyGyro-entropy()-Y	-0.78
fBodyAccJerk-entropy()-X	-1.00
fBodyAcc-entropy()-X	-0.95
tBodyGyroJerk-entropy()-Z	-0.62



Feature	Value
tBodyGyroJerk-entropy()-X	0.22
tGravityAcc-energy()-X	0.92
fBodyGyro-entropy()-Y	0.60
fBodyBodyGyroJerkMag-entropy	0.30
tBodyGyroMag-entropy()	0.83
tBodyGyroJerk-arCoeff()-Z,2	-0.93
fBodyAcc-skewness()-Y	0.82
tBodyGyroJerk-arCoeff()-Z,1	1.00
tGravityAcc-iqr()-Z	-0.19

The results emphasize the importance of features extracted from gyroscope sensors, especially the entropy values, in order to establish if an individual is sitting or not and the relevance of those same features and gravity acceleration in order to establish if a person is standing or not. As a second explanation task, we proceed with the application of a Shapley Additive Explanations approach (SHAP), this time considering as a reference model the best Random Forest identified for the data-set, using the entropy splitting criterion and taking into account the whole training set for integrating out features: this allows us to identify for each feature an importance value for a certain prediction representing the effect on the model prediction of including that feature. Hence, we report below the relevant force plot for a record predicted as "Sitting", this being the class for which most of the previous models seem to face more difficulties in the prediction process. We can underline the relevance of attributes related to gravity for the classification, an aspect already partly specified in previous feature importance graphs.



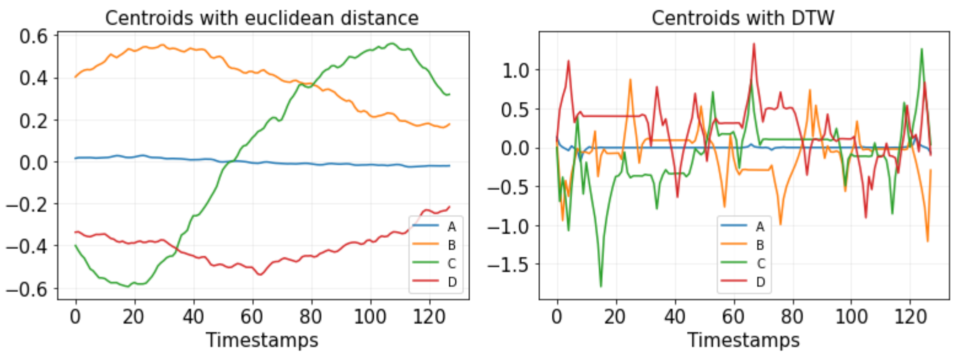
4 Time-series analysis

This section is dedicated to the understanding of the variation in time of subjects' activities. We exploit one of the time-series data-sets built for the human activity recognition task, specifically the *body_gyro_x* which takes into account signals on the x-axis coming from sensors' gyroscopes. We will be focusing on a clustering task considering different univariate time-series, a classification task and a sequential pattern mining extraction task.

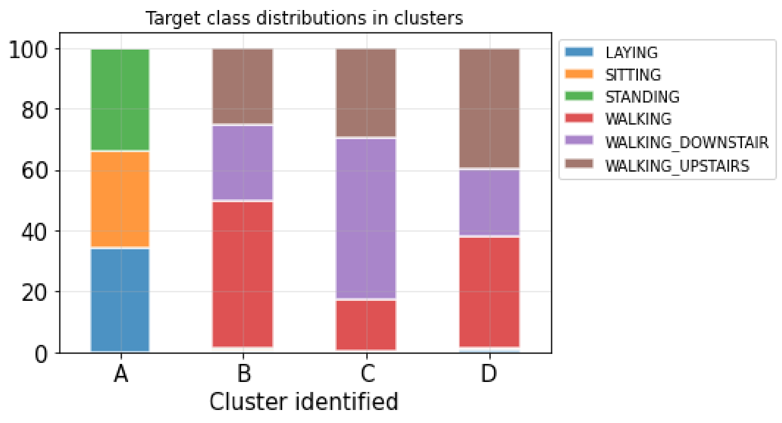
4.1 Time-series Clustering

4.1.1 *k*-Means on original time-series

At first we advance a simple clustering analysis exploiting *k*-Means algorithm on the time-series data-set taken into account. By using the *Knee method* heuristics for setting the number of centroids, we evaluate the performance of *k*-Means with two main meterics, considering a maximum number of iterations equal to 100: euclidean distance and Dynamic Time Warping (DTW). For both metrics an acceptable trade-off between the number of clusters found and SSE could be identified using 4 clusters[19]: we report here that the SSE score computed using euclidean distance is about 16.20 and the one computed using DTW is 4.00; we also offer a visualization of the respective centroids isolated using the two distance functions.



Both metrics perform quite well observing their SSE result but using DTW leads to a slightly lower error score, hence we focus further clustering analyisis on the results given by such metric. We evaluate the distribution of target class values within the clusters found through *k*-Means with DTW as metric.

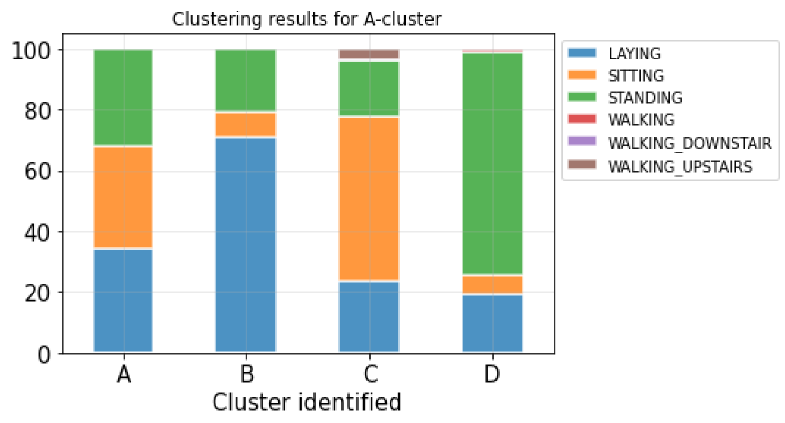


As a first observation, we notice that almost every point belonging to one of the first three classes, "Laying", "Sitting", "Standing" end up being grouped inside the A-cluster with a rather homogeneous distribution. This ends up being the most populated cluster with respect to the original data-set, with a cardinality of 4038. On the other hand, "Walking", "Walking upstairs" and "Walking downstairs" are distributed among the other three clusters: B-cluster (1221 points) hosts a consistent percentage of "Walking"-labeled points, while C-cluster (701 points) is mainly populated by "Walking downstairs". At last, D-cluster (1392 points) presents almost the same percentage of "Walking" and "Walking upstairs" points, with a small percentage of elements labeled "Walking downstairs". In this context, it seems

[19]We here specify that for clustering tasks we kept the original time-series normalization, with values ranging from -1 to +1. This is also due to the fact that applying a time-series scaler mapping original values onto the [0,1] interval led to much poorer clustering results.

that clusterization provided by *k*-Means allows to distinguish well between passive activities, such as "Laying", "Sitting" or "Standing" and more dynamic kind of movements such as walking in various forms, while failing in capturing a difference between the first three classes taken into account.

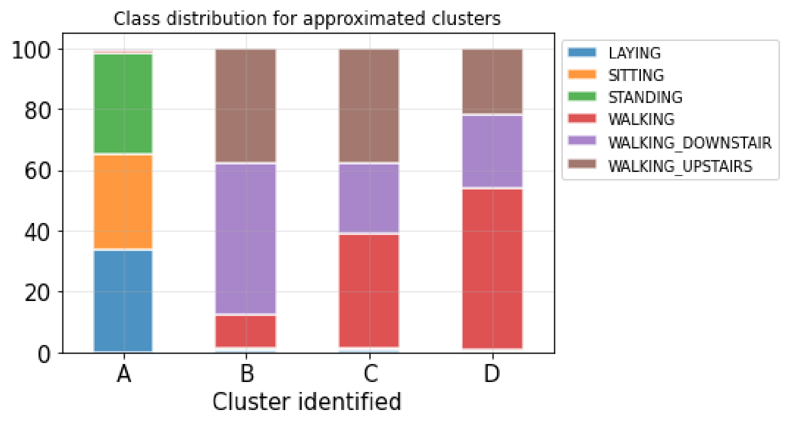
We can then check if *k*-Means with DTW metrics is able to further discriminate between more static values once the more dynamic ones are isolated from them: so we perform a clusterization only taking into account the A-cluster. We keep considering a *k*=4 in order to check clusters purity:



The result is not so satisfactory, especially for a lack of balance in terms of clusters' dimensions: in fact, we can see that B-cluster (179 points), C-cluster (122 points) and D-cluster (280 points) respectively show a majority of a single class value inside them, but we still end up having a single big homogenous cluster hosting a relevant percentage of the original sub-set considered: it is the one labeled as A-cluster, including 3457 points.

4.1.2 *k*-Means on approximated time-series

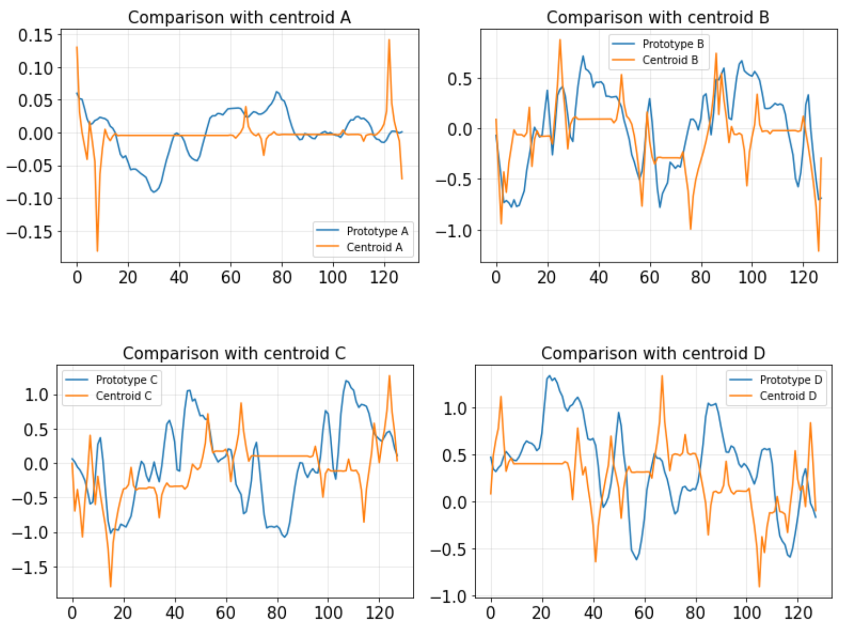
It is now interesting to consider how *k*-Means with DTW as distance function performs when applied on an approximated data-set of the original time-series: we perfomed a *Piecewise Aggregate Approximation* (PAA) considering a number of 32 segments. The relevant bar-chart is shown below:



Applying k-Means on the approximated data-set allows us to reach the best SSE value considered so far: 1.45. Still, looking at the the distribution of class values inside the identified clusters, we do not have much different results from the standard application of k-Means on the non-approximated time-series: clusters size is more or less the same, and this can be said about the distribution of class values, too. The main difference seems to arise in terms of distribution of dynamic subject activities: we have two clusters that respecitvely host as a majority of points "Walking downstairs" elements and "Walking" elements, while "Walking upstairs" has more or less the same distribution in all the three smaller clusters.

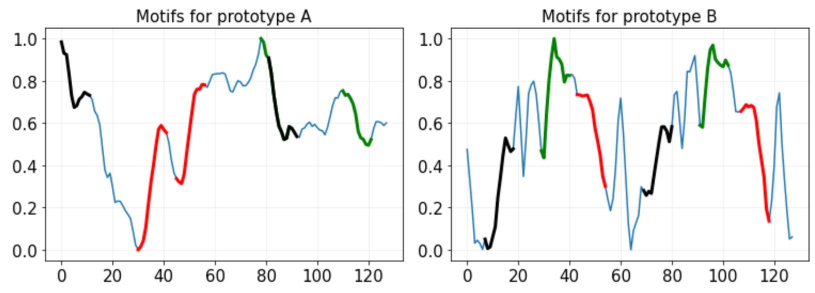
4.2 Time-series Motifs/Discords

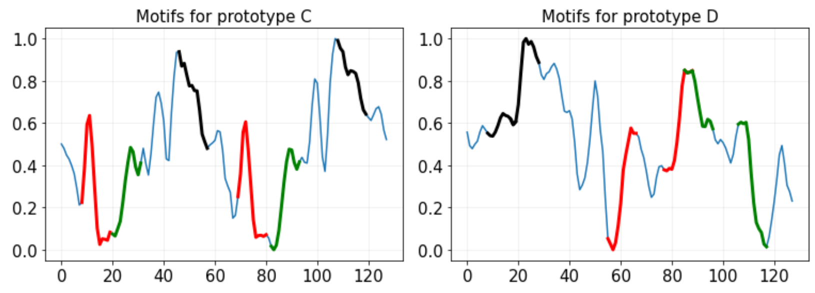
In order to define relevant times-series from the data-set for motifs and discords analysis, we exploit the previous clusterization task using k-Means with DTW as metric, considering for each cluster a prototypical time-series, namely the one having the lowest distance in terms of DTW with the centroid of that particular cluster. We visually compare the centroids with their respective most similar time-series and report distance values between the pairs.



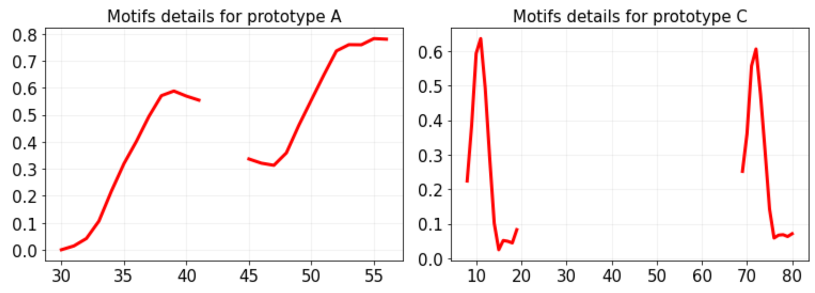
A-cluster	B-cluster	C-cluster	D-cluster
0.21	1.47	1.86	1.55

We can then proceed with a more accurate motif analysis: for this task we produce a matrix profile representation of the prototypical time-series after having perfomed a normalization scaling the values with respect to the interval $[0, 1]$, establishing a window span of 12 and using STOMP algorithm to compute it. Then we exploit the matrix profile to compute motifs from the time-series, isolating for each one a total of three motifs and visually representing them as follows:

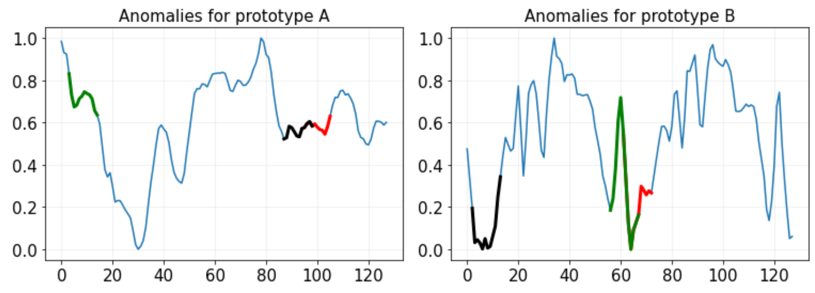


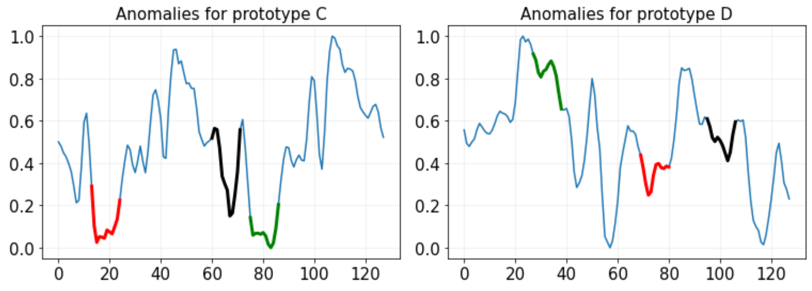


We notice here that the patterns colored as black for the prototypical time-series of cluster D end up overlapping for the window size chosen, having their starting indices respectively in time-stamp 8 and 17. On the other hand, motifs for prototype A and B seem to show a more linear tendecy when compared to motifs of prototype C, which present spikier curves. We also report here in details the portion of motifs for prototypes A and C emphasizing such difference:



In an analogous way, we represent the three main anomalies discovered for each time-series taken into account, considering the same matrix profile built for the previous task, noticing that anomalies for prototype A seem to follow "flatter" shapes while spikier discords are present for prototypes B and C.



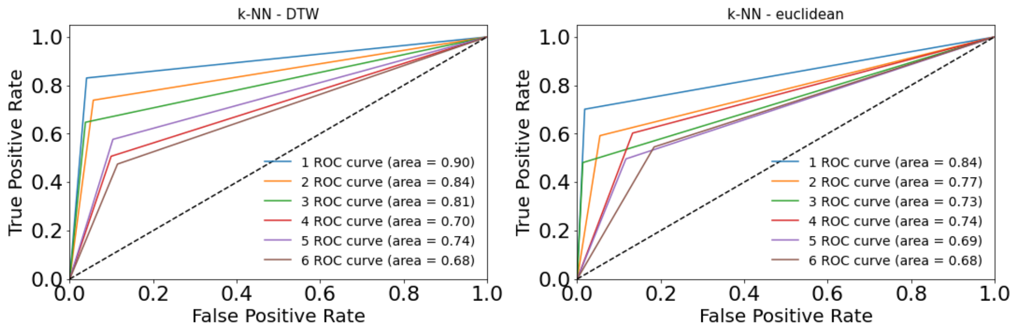


4.3 Time-series Classification

We keep considering the time-series data-set previously used for clustering and try to train and test a simple *k*-NN time-series classifier keeping a *k*=1 and considering two different distance functions for time-series comparison: euclidean distance and DTW. This simple classifier will be taken as a baseline for further comparison with more elaborated classification approaches.

k-NN euclidean	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.89	0.67	0.86	0.48	0.48	0.40
Recall	0.70	0.59	0.48	0.60	0.50	0.55
F1-score	0.78	0.63	0.62	0.53	0.49	0.46

k-NN DTW	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.81	0.71	0.74	0.51	0.55	0.48
Recall	0.83	0.74	0.65	0.51	0.58	0.48
F1-score	0.82	0.73	0.69	0.51	0.56	0.48



Reasonably, DTW leads to better results, also in terms of accuracy (0.63) against the outcomes reached with the euclidean metric, this one having a quite lower accuracy score (0.57). We also underline the overall better performances, in terms of precision and recall, for more dynamic labels and rather bad results for the "Laying" activity, suggesting how time data acquired according to gyroscope on the x-axis makes the identification of static movements harder. Having now established a baseline model, we can proceed with further analysis. We use the grabocka parameters approach to identify a suggested size to look for the shapelet, in this case 12 shapelets is identified as the best size for the classification. Then we evalutate results of a DT classifier, a *k*-NN classifier and a Logistic regression approach with gradient descent all of them using the shapelets identified considering the distances of the former ones with respect to the original time-series as a reference to perform the classification. Logistic regression appears to be the weakest approach: best results were obtained with 400 epochs, the maximum value tried in the testing phase, and considering a weight regularizer of 0.01; for DT classifier, we used a grid search checking different combinations of max depth (None to 20), minum samples for a split (2, 5, 10, 20, 30, 50, 100) and minimum for a node to be a leaf (1, 5, 10, 20, 30, 50, 100), reaching the best results with a tree wit max depth 9, 100 as minimum samples for a split and 20 as minimum samples for a leaf; at last we considered a *k*-NN Classifier testing it with a *k* parameter ranging from 1 to 50, reaching the conclusion that after *k*=30 no great gains in terms of accuracy are reached, both using a uniform weight distribution or weighting points according to distance. Therefore, we report in the table below results coming from the three models[20]. We specify here that the time-series used as a source for the extraction of shapelets were all different from the prototypical ones used for motif discovery, suggesting that the prototypes identified in the previous task are not considerable as good sources of class discrimination for the shapelet methodology.

Shapelet - SGD	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.42	0.61	0.85	0.52	0.60	0.37
Recall	0.88	0.27	0.11	0.10	0.17	0.97
F1-score	0.57	0.38	0.20	0.17	0.26	0.54

Shapelet - DT	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.65	0.62	0.64	0.52	0.49	0.44
Recall	0.70	0.61	0.57	0.47	0.62	0.38
F1-score	0.67	0.61	0.60	0.49	0.55	0.41

Shapelet - *k*-NN	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.66	0.60	0.67	0.60	0.49	0.53
Recall	0.72	0.64	0.55	0.55	0.68	0.37
F1-score	0.69	0.62	0.60	0.57	0.57	0.44

In all the three cases, the results were rather unsatisfactory, leading to worse performance than the simple *k*-NN with DTW as metric. On the other hand, the best results were obtained testing a state-of-the-art methodology for time-series classification: we adopt the MiniROCKET algorithm, using a Ridge

[20]As done for previous model, we report here the accuracy score for the three classifiers: 0.43 for the gradient descent approach, 0.56 for DT and 0.58 for *k*-NN.

Classifier cross validation for class evaluation, considering 10000 kernels in the training phase, reaching an accuracy level of 0.72, overcoming the baseline's performances in a satisfactory way but with still rather low scores for more static labels.

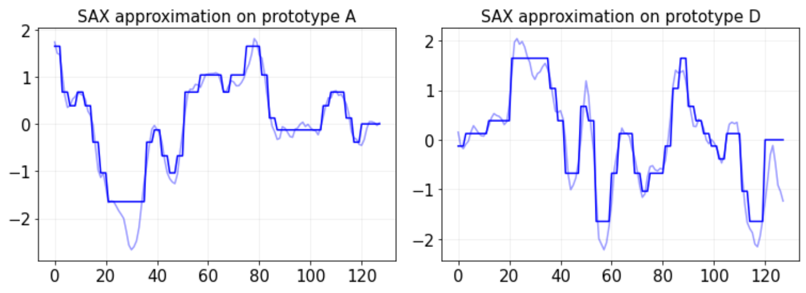
MiniRocket	Walking	Walking upstairs	Walking downstairs	Sitting	Standing	Laying
Precision	0.90	0.74	0.76	0.63	0.66	0.65
Recall	0.79	0.83	0.76	0.55	0.83	0.55
F1-score	0.84	0.78	0.76	0.59	0.73	0.59

4.4 Sequential Pattern Mining

The purpose of the following task is to identify frequent recurring sub-sequences with respect to the original time-series data-set. We first focus our attention on the four prototypical time-series previously isolated, with the aim of identifying common recurring patterns. In this case, due to the absence of original sequences from the data-set, we exploit SAX algorithm as a sequence builder, dividing each 128-time stamps long time-series into 16 successive "windows" of 8 time-stamps each, then applying SAX on each of them, setting an alphabet of 8 symbols and setting 4 as the total length of the resulting segments, in order to use the compressed representation of the time-series' sub-parts as transactions of a single sequence. Therefore, merging together the result of applying SAX on each sub-part of a single time-series allows us to build a sequence of transactions, in this case each including 4 elements corresponding to the starting compressed "window". We further specify that all the four time-series considered are scaled according to a mean variance scaler, considering as parameters a standard deviation of 1 and a mean of 0. We proceed in using GSP algorithm in order to identify the most interesting sequential patterns: with a support threshold of 0.8 we obtain a total of 627 relevant patterns with a lenght higher then 5. Some examples of three of the longest and most interesting patterns, some including transactions with multiple elements, are the following ones: <{5}, {6}, {6}, {7}, {5}, {0 6}, {6}, {4}>; <{5}, {6}, {5}, {6}, {0 6}, {6}, {6}>; <{3}, {0 6}, {6}, {4}, {5}, {6}>.

We underline here that also for all the other not reported patterns, the {0 6} itemset is the only transaction with a cardinality higher then 1 that GSP ends up identifying. We also specify that for other relevant patterns of the same length, SAX symbols going from 0 to 3 tend to appear rather sporadically inside the transactions.

We can then perform a more general and simplified analysis, taking into account the whole time-series data-set but without any splitting of the original series in order to build sequences; we still apply SAX on scaled versions of the original time-series, in order to get a sequence of transactions from each one of them, but this time that the whole sequence of symbols extracted from each time-series will be used as a sequence of itemsets, without any splitting into sub-portions: we report a visual reconstruction of two prototypical time-series using SAX approximation as mentioned above, this time with a total of 10 symbols and 40 segments:



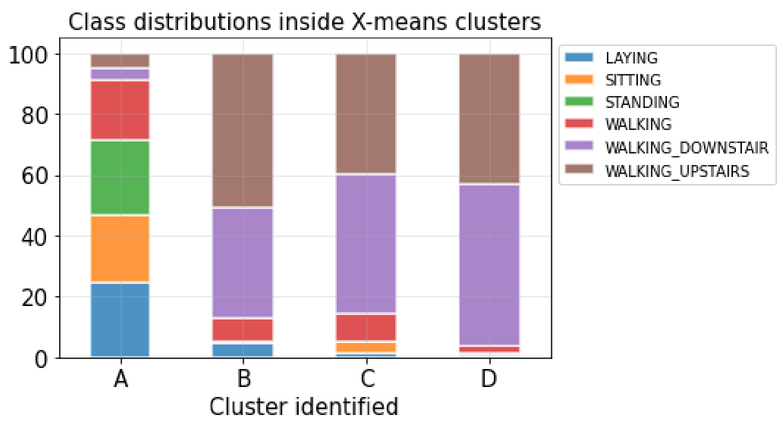
We kept using GSP imposing a threshold support of 0.6: a total of 678 patterns were identified and we report here some relevant sequential patterns found with their respective support measure, with a total of 568 patterns with a length higher then 2. Some examples of frequent patterns of length 3 are reported in the box below, specifying this time their relevant support. We also emphasize that applying threshold constraints that are slightly higher result in a rapid decrease in terms of patterns identified: for 0.7 we have 243 total patterns and for 0.8 just 75.

Frequent patterns for whole time-series data-set	
Frequent patterm	Support
<{4}, {5} , {6}>	5505
<{3}, {2} , {2}>	5436
<{7}, {6} , {2}>	4900
<{9}, {7} , {2}>	4436

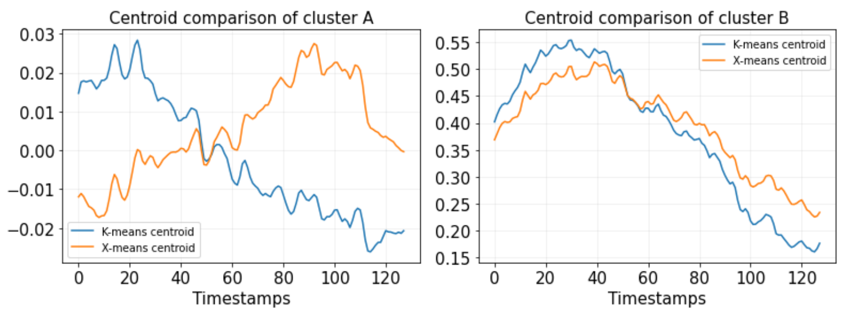
5 Advanced clustering methods

5.1 X-means

We exploit X-means algorithm on the time-series data-set analyzed in the previous section in order to check the purity of clusters with respect to target class distribution. As before, we impose a total number of 4 clusters as final desired result, also checking how centroids identified by X-means present differences from the ones isolated through previous clustering methodologies. Performances are not very satisfactory in this case: we reach a first A-cluster of 5595 points while the other ones encapsulate the remaining elements, with a cardinality of 691 and 675 respectively for B-cluster and C-cluster, and a size of 391 for the D-cluster. Hence, the clusterization results quite imbalanced: as the bar-chart aims to show, the majority of the points belonging to a rather static movement class are isolated inside the biggest cluster, while the other points having labels related to more dynamic movements are scattered inside the other three groups. This is a tendency we have already observed for the previous clustering methods, but in this case it results much more emphasized looking at the difference between clusters in terms of their respective size:



For completeness, we include a comparison of centroids obtained through the application of X-means with the ones established by K-means applying as distance metric the euclidean distance, focusing on the most similar and most different couple of centroids:

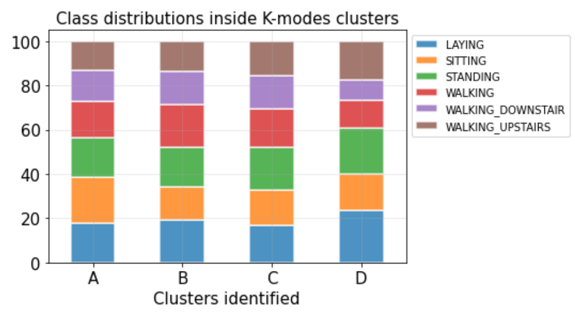


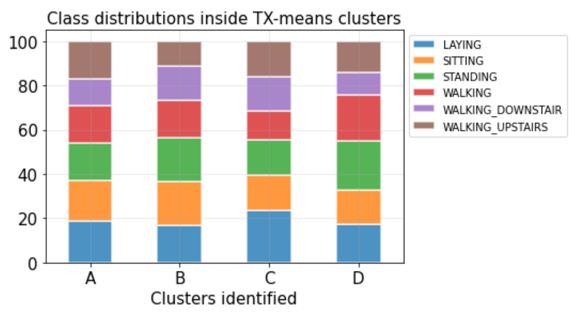
As one could easily guess considering clusters' sizes, it is the centroid of the biggest cluster presenting the highest difference from the one identified through K-means.

On the other hand, we apply the OPTICS clustering algorithm on the original data-set to perform a comparison between noise points identified by OPTICS and the original anomalies previously isolated with different outlier detection techniques. Performing an OPTICS clustering analysis with 10 as minimum number of samples leads to a Silhouette score of 0.476 and leads to a total of 93 anomalies, with a very small percentage in common with the ones identified through previous methods and with not so satisfactory results. We also have a total of 4 clusters grouped through this method, emphasizing different levels of density for the data-set, a result consistent with previous clustering approaches defined.

5.2 K-modes and TX-means

In order to apply *K-modes* and *TX-means* algorithms for a significant analysis, we exploit the previous pattern mining task, using as input the time-series approximated with SAX considering an alphabet of 10 symbols and segments of length 40. In order to keep a continuity with the previous clustering methods, we set 4 as the number of total clusters to be identified by *K-modes*; On the other hand, *TX-means* automatically establishes the final number of clusters for the data-set in input - in this case 4 total clusters were isolated, making the target class distribution comparison easier:





The results are quite distant from clustering outcomes observed with previous methodologies: both *TX-means* and *K-modes* on the approximated time-series seem unable to perform a satisfactory differentiation between groups of elements belonging to different values for the activity class: every cluster presents more or less the same distribution of points coming from all the six values of the target class. We tested *K-modes* algorithm with a k ranging from 4 to 8 included and, from a qualitative perspective, class distribution did not change significantly for higher k values, entailing that in general *K-modes* performs quite badly on the time-series data-set taken into account.

6 Final remarks

The following brief section will be dedicated to a summary of different results coming from the tables and graphs presented above. For the outlier detection task, we emphasize the distance in terms of performance of an LOF approach from the other methodologies applied and we especially praise the performances of Isolation Forest and ABOD as fast ways to identify the top anomalies. Focusing on the classification task, we underline the effectiveness of ensemble methods and deep neural network approaches specifying the importance of gravity related attributes for what concerns classifiers that exploit Decision Tree methodologies, results consistent with the explaination techniques used for individual records. Although some models reached more then acceptable results, we must also consider the fact that in every classifier considered we could identify a weakness considering the correct identification of sitting activities, a class for which the various classifiers seem to show poor performances in terms of precision and recall. For what concerns time-series analysis, we emphasize the satisfactory ability of k-NN in separating dynamic activities from more static ones in the clustering task and the efficacy of MiniROCKET algorithm for classification purposes. We also emphasize how checking prototypes' motifs is revealing of some different tendencies time-series belonging to different clusters seem to present. At last, we underline the relevance of adopting advanced clustering techniques for a comparison with simpler approaches, although results coming from the former does not seem to give much gain in terms of class distribution.