Data Mining - Human Activity Recognition

Giovanni Perri, Marco Recchioni

July 2022

1 Introduction

The experiment records 30 people performing normal activities:

- 1-Walking
- 2-Walking upstairs
- 3-Walking downstars
- 4-Sitting
- 5-Standing
- 6-Laying

The data from such activities have been recorded trough the sensor of a smartphone (Samsung Galaxy S II).
The data have been pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter with 0.3 Hz cutoff frequency was used. From each window, a vector of features was obtained by calculating variables from the time and frequency domain.
In the end two datasets have been provided. One with the time series with 128 feature corresponding to the time steps, and one with all the characteristic of the trasformation of those variable regarding a specific activity and subject, consisting in 561 feature.

2 Module 1

In this first section the 561-feature dataset is studied to investigate it's property and see how basic algorithm performs on it.
The dataset is divided in a 70-30 proportion for the train and test set.

2.1 Data exploration

In order to understand the problem a good knowledge of the characteristic of the data is essential.
Is straightforward to notice that the data are already normalized in a range $[-1, 1]$, but due to the big amount of feature, their singular distribution haven't been calculated focusing more on the general property of the set.
In fact, those feature come from different transformation and analysis of the same time record and hence they are a lot related to each others.
Looking at the linear correlation (Pearson correlation) we identified a series of attributes which share correlation equal to 1. We removed, since they add no information, the redundant feature for a total of 21.
In general we have seen that in the half of the $561x561$ correlation matrix:
-4366 couples of attributes have linear correlation > 0.95
-8292 couples have Spearman correlation > 0.95
We tried to remove such points but it turned out on the classification task to get a worse representation of the problem, that's because correlation does not imply causation.
The signals have been recorded for 6 activities performed by 30 different subjects. We want to inspect how those 2 variable are distributed over the train and test set.
In the following an histogram of the class distribution:

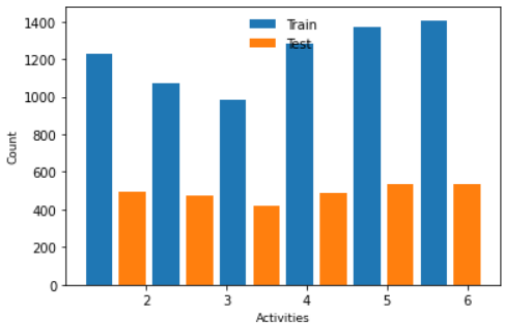


Fig. 1 Histogram for the class among train and test set

Figure 1 shows that the activities are balanced over the train and test set.
An histogram of the subjects instead is plotted below and demonstrates how they are completely separated over this two sets. The data from a person recording will belong to either the train or the test set. This is reasonable since we're averaging over different subjects in the train set to extract important feature which will be use to predict the activity of another random person to make the prediction unbiased towards particular pattern which could characterize the activity performed by a particular subject.

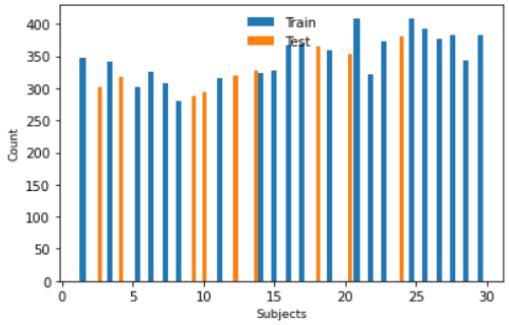


Fig. 2 Histogram for the subjects among train and test set

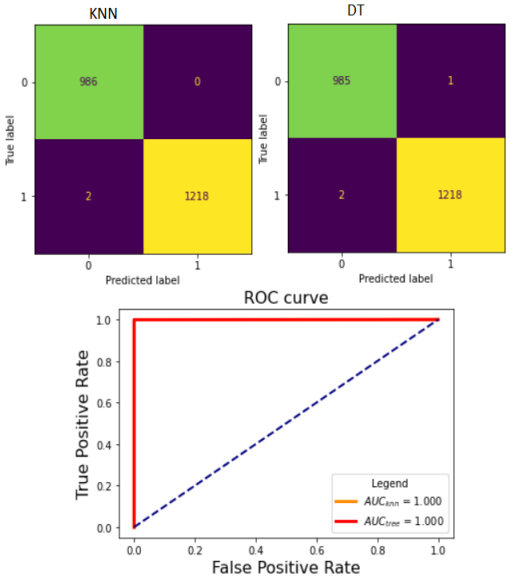


Fig. 3 Confusion matrix and ROC curve for KNN and DT. Classification task: movement or stationary

2.2 Classification task

The classification tasks that could be performed on this dataset are many and have even more applications. Is getting important for the phone to understand in real time the activity performed by the user for example the cellphone function of recording the daily steps.
We have tried many classification task and in this first stage we've extracted 30% of the data from the train set for validation purpose.
The classification have been carried out with two classifier: ClassificationTree and KNN.
The hyperparameters defining those classifiers have been found for each classification with a GridSearchCV and later with a CrossValidation and accuracy and standard deviation are extracted.

2.2.1 Stationary or movement

The first classification task we've checked is the ability of the classifier to distinguish between movement and stationary activities.
This is the basic tasks the classifier should be able to perform on such a dataset and in fact it turned out to be really easy for both KNN and DT.
As a matter of fact both of classifiers gave the same result for every dataset used (full one (X), removing feature with correlation 1 (Xc1), removing correlation > 0.95 (Xc9)) .
In fact for both classifiers, accuracy, F1 and recall score are 1. As well as the ROC curve area which well depict the perfect achivment of the task.

We can notice from the confusion matrix that only 2 points for KNN and 3 for DT have been missclassified.
The KNN took as best number of neighbours 11.
For the DT the optimax max depth is 9.

2.2.2 Activities

We want to see how those algorithms classify the whole dataset in a multiclass classification task in a OVR way (one vs rest). This is easier for the algorithm since allows to create hard constrain for the class in consideration.
Using the DT algorithm a feature importancies bar plot is extracted showing that the majority of the feature don't even partecipate on the decision for the class the data belong to.

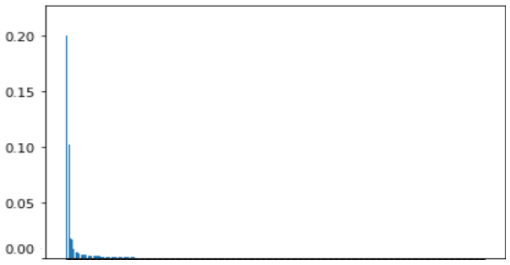


Fig. Bar plot: importancies (y) vs features (x)

The classification have been carried out on 4 dataset:

- full data (X)
- Removing correlation 1 (Xc1)
- Removing correlation 0.9 (Xc9)
- Removing feature from DT importancies (XDT)

We performed a cross validation with 5 folders on the 70% of the training set and used the remaining for validation to create the confusion matrix.
Results:

- X: $accDT = 0.93 \pm 0.01$, $accKNN = 0.962 \pm 0.004$
- Xc1: $accDT = 0.93 \pm 0.01$, $accKNN = 0.964 \pm 0.005$
- Xc9: $accDT = 0.93 \pm 0.02$, $accKNN = 0.960 \pm 0.004$
- XDT: $accDT = 0.93 \pm 0.01$, $accKNN = 0.933 \pm 0.006$

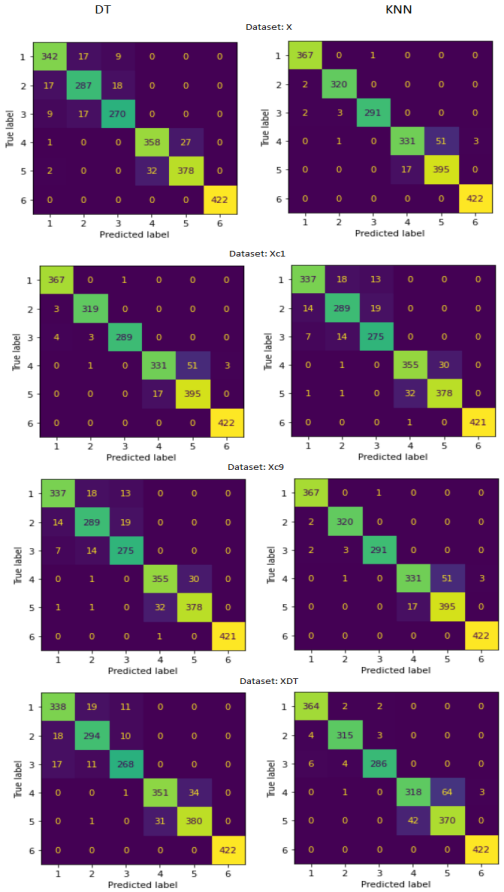


Fig. 5 Confusion matrix for KNN and DT and different datasets

All of them give pretty similar results. However by removing high correlation feature Xc9 presents a slightly worse accuracy for the KNN algorithm and an increase of the standard deviation on the DT.
While the results we get from XDT shows no change in the accuracy of the DT (predictable outcome since those are the feature not used during classification) and a significant lowering on the KNN accuracy.
Even though all of them remain pretty high we decided to keep the Xc1 dataset.

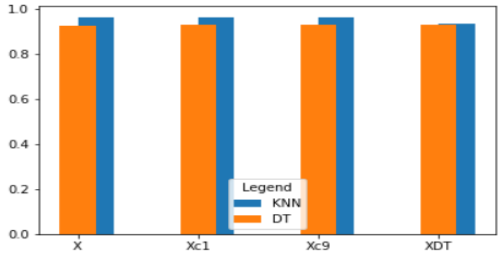


Fig. 6 Histogram of the accuracy reached by the classifiers on different datasets

However we tested the trained algorithm even on the test set giving much lower results ($\sim$ 8% less accuracy). This is an overfitting issue as briefly discussed at the end of section 2.1 for which we tested the data on a validation test containing the same subjects which performed such activity.
Another aspect worth noting is that for the DT the max depth for this classification is 17. So for a better understanding of the problem and a accurate classification not only on the macroclasses (stationary or movement) the tree has to go deeper as expected.
While for the best number of neighbours for the KNN fall to 6 which is reasonable since it has to look for more activities.

2.3 Outliers

Due to the high dimensionality of the data the algoritmh which fit the most the problem should be ABOD and isolation forrest. For completeness we also performed the KNN and the DBSC.
The hyperparameters tuning has been carried out to get the less number of points.
To assign a quality measure on the outliers detection, we train the DT on the dataset without the outliers found, and than tried to classify them.
Real outliers should be hard to predict.
We show for every method the bar plot of the outlier-score (the trand of this score depend on the algorithm for example KNN outliers have higher score while for the isolation have lower one), the class of belonging of the points found and their classification report.
KNN
The algorithm found at minimum 403 outliers with "n neighbours" set to 3.
In the following an histogram of the anomaly score, an histogram of the activities of those outliers and relative classification.

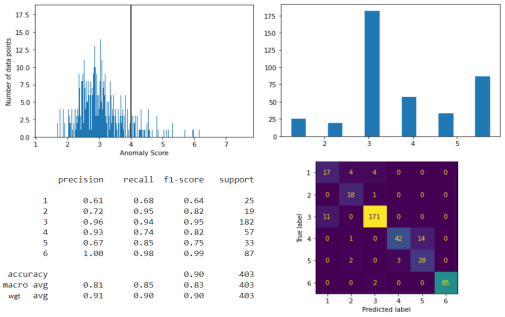


Fig. 7 Anomaly score, class histogram and classification report for outliers found with KNN

DBSCAN

With $eps = 5$, $minsample = 24$, 241 outliers have been found.

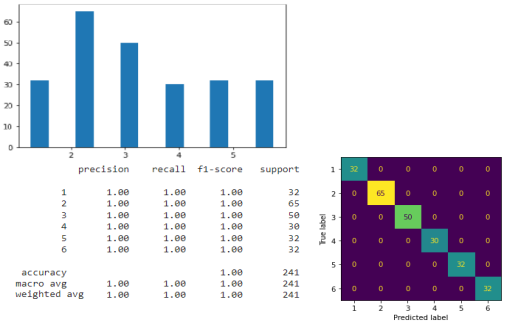


Fig. 8 Class histogram and classification report for outliers found with DBSCAN

ABOD

With a number of neighbours of 35, 724 outliers have been detected with such an algorithm.

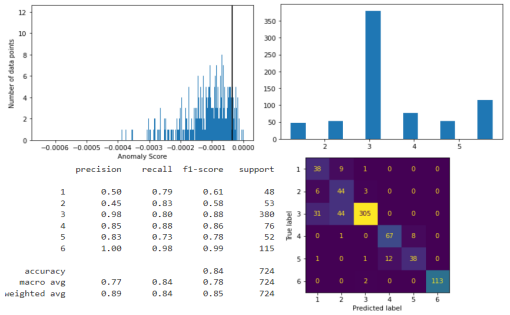


Fig. 9 Anomaly score, class histogram and classification report for outliers found with ABOD

Isolation Forrest

Using 200 estimators and all the feature 637 outliers have been found.

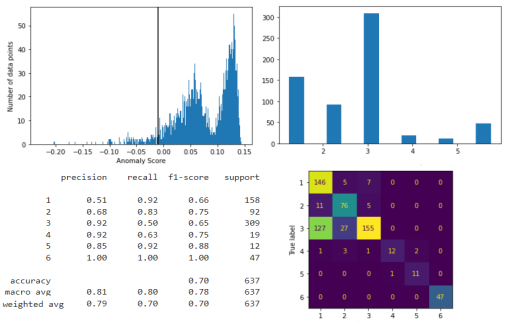


Fig. 10 Anomaly score, class histogram and classification report for outliers found with IF

Considerations

It is clear that most of the outliers are found on the walking downstairs activity (3) although those points are well classified in all the cases.
Not taking into consideration those class 3 points, all the algorithm but the IF found a majority of anomaly points on the stationary activity in contrast to the IF.
The classification is a slightly unbalanced so we can look at the F1 score on the classification report.
The points from the DBSCAN are classified almost exactly as well as the ones from the KNN.
While the other 2 algorithms found outliers which are shuffled over the classes during classification (in fact their F1 score is much lower) meaning that those points could better represents outliers.

2.4 Imbalanced Learning

To fulfill the task of the project we've perform an imbalance learning classification task.
This time we focus on the movement activity, hence the classification will be to distinguish between walking and walking on the stairs. Those data extracted from the train set are not so unbalanced since their ratio is 1/3 walking (class 1) and 2/3 walking on the stairs (class 0). Hence, we've discard 90% of the data belonging to the walking activity, to reach a 5% of the data with class 1.
The classification has been carried out with different algorithms for balancing the data.

2.4.1 Classification

Specifically we've used the following unbalanced learning technique:

Normal

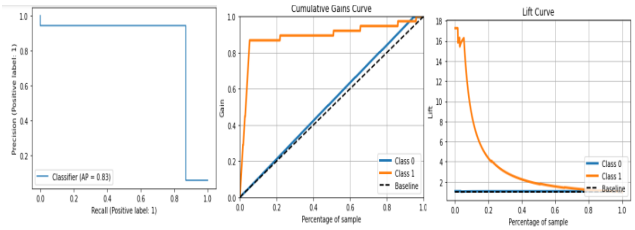


Fig. 11 Prec-Rec, GAIN and LIFT curve with no techniques.

Precision Recall curve: precision is a measure of result relevancy, while recall is a measure of how many truly relevant results are returned. Hence this curve shows a trade off for different threshold of those values. The curve of the model with no method applied falls for high recall values meaning that for low False Negative there will be a dominant number of False Positive.
Cumulative gain curve: shows the ratio of different cumulative positive observation to respect to the total number.
A steep slope at the beginning tells us that the model doesn't perform badly in fact in the first deciles of the most probable prediction it reach a cumulative gain > 0.9, slowing stepping up close to 1 when we consider all the data.
Lift curve: it compares the information gained from the model to a random one. The random curve is represented as a horizontal line. At that point in the chart where the model curve declines below the random curve, the lift factor is smaller than 1. This means that the records of the model contain fewer target values than a random sample with equally distributed target values.
Since the curve reaches this values at the end and it always stays over it, it means that the model is better than a random guess.

Class weight balanced

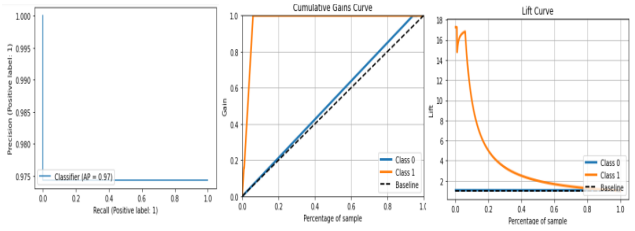


Fig. 12 Prec-Rec, GAIN and LIFT curve, BALANCED.

Those graphs shows how the balanced instance of the classification three outperform the basic classification giving a steeper and more valuable precision recall curve, same for the gain one and the lift value much higher than the normal one.

Random Under Sampling

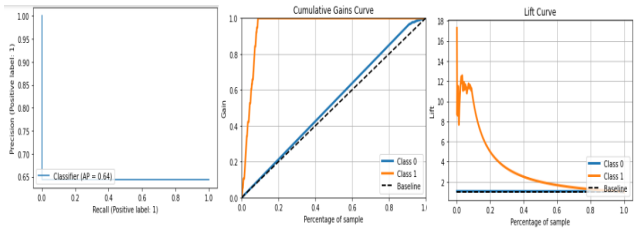


Fig. 13 Prec-Rec, GAIN and LIFT curve, RUS.

Since we have discard $\sim$ 90% of the walking data, undersample the majority class will produce a poor dataset with poor results. That's the case for the Random under sampling for which the curve are a little worse than the normal classification.

CNN

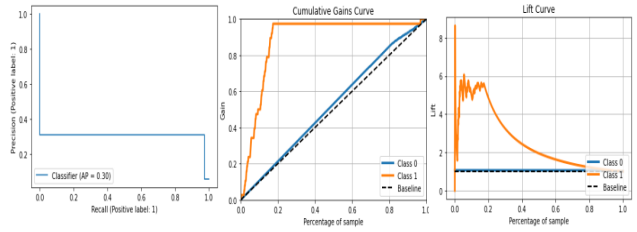


Fig. 14 Prec-Rec, GAIN and LIFT curve, CNN.

Similary to the RUS, this undersample the most uncertain points from the majority class.
This seems to be the worst method for treat this unbalanced problem since all the curve parameters are worse than the normal case. Moreover from the lift chart we can see how it actually perform close to a random model.

Random Over Sampling

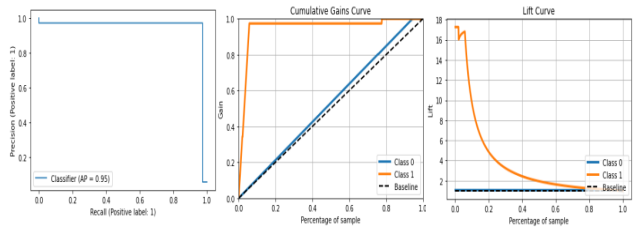


Fig. 15 Prec-Rec, GAIN and LIFT curve, ROS.

With this technique the data basically return to the beginning before we remove 90% of the points. But since it sample from a narrowed distribution the resampled point will be more concentrated on the right class of belonging giving a slightly better results.

SMOTE

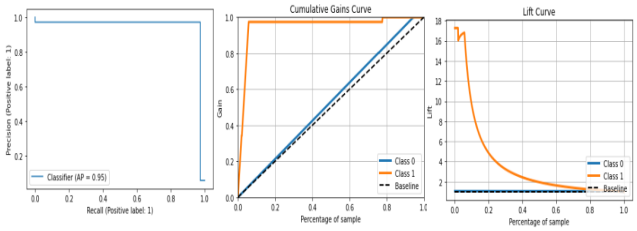


Fig. 16 Prec-Rec, GAIN and LIFT curve, SMOTE.

SMOTE works by selecting examples that are close

in the feature space, drawing a line between the examples in the feature space and drawing a new sample at a point along that line. So it reaches similar results to the ROS.

SUMMARY

In the next image a summary of the performance for all those techniques

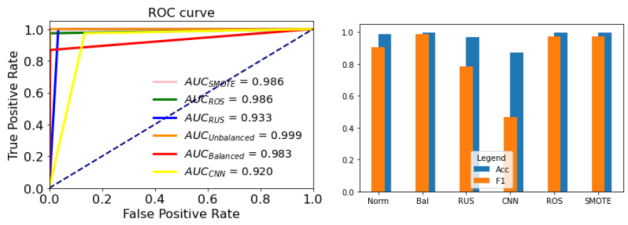


Fig. 17 ROC curve and accuracy reach with every umbalanced method.

We can conclude the "weight class= balanced" attributes of the decision tree gives the best result.

2.5 Dimensionality reduction

Our dataset have a high dimensionality feature space and there are few techniques that can be use to reduce such a problem.

There are two major family of dimensionality reduction: feature selection and feature extraction.

Those procedures are necessary for both visualization and optimization.

2.5.1 Feature selection

Variance Threshold

This is justified by the fact that small variance feature will be mainly constant and give no information.

We removed the feature for different variance threshold and calculate the accuracy by a cross validation using a decision tree.

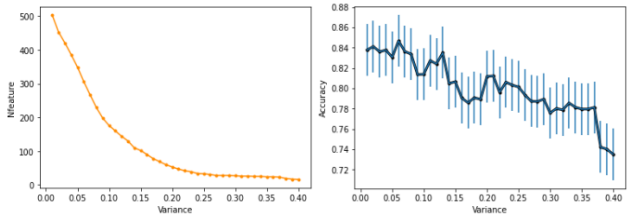


Fig. 18
Left: variance threshold vs number of feature.
Right: accuracy level vs variance threshold

The optimum threshold is a trade off between the number of feature and the accuracy level.

The best accuracy is reached by dropping the feature with variance smaller than 0.06 where the mean accuracy reach 0.85 and the feature are 300. This correspond to the beginning of the elbow of the accuracy-number of feature curve.

Select K best

We varied the number K of best feature ramping from 1 to 50 with two different score function: F1 score and Mutual Information.

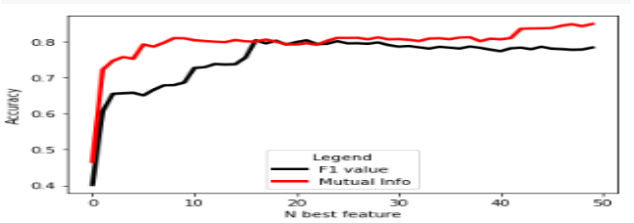


Fig. 19 Accuracy level vs different number of K, for both function score

As we can see the mutual information score function performs better. Accuracy= 0.85 ± 0.09

SelectFromModel

From both DT and RandomForestClassifier.

DT: feature(31), $Acc = 0.86 \pm 0.03$

RFC: feature(120), $Acc = 0.85 \pm 0.02$

2.5.2 Feature extraction

PCA

It's a linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space. It's a good method even for representation

In the next image, left a plot of the PCA components and relative importancy, right, scatter plot in 3 dimensions colored with the different activities label.

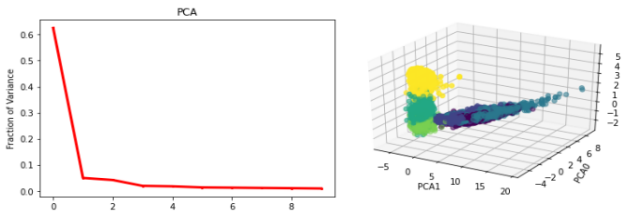


Fig. 20 Left: importancies vs PCA component
Right: PCA 3d scatter plot colored by classes

We can see how different activities remain well separated in this space. Moreover 2 supercluster can be seen, defining two macrocategories of activity: stationary or in movement.

To find the optimal number of components to consider we plot the accuracy.

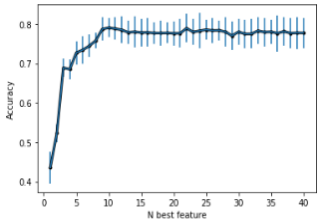


Fig. 21 Accuracy vs number of PCA components

For which there is a maximum followed by a plateau at 10 feature.
Accuracy: 0.79 ± 0.04

ISO

It is a non-linear dimensionality reduction through Isometric Mapping. Number of neighbuours is set to 7.
In this space the classes are well separated.

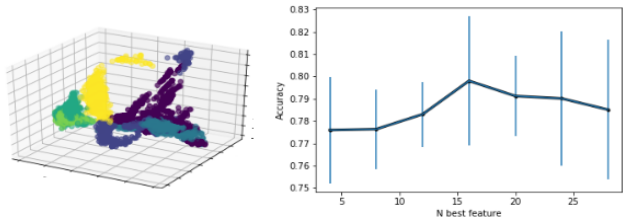


Fig. 22 Left: ISO 3d scatter plot colored by classes Right: importancies vs ISO component

Best number of feature: 16
Accuracy: 0.79 ± 0.03

Gaussian Random Projection

It uses a random gaussian matrix to project the initial space. The dimensions and distribution of random projections matrices are controlled so as to preserve the pairwise distances between any two samples of the dataset.
The separation of the classes is not clear anymore in 3 dimensions as we can see:

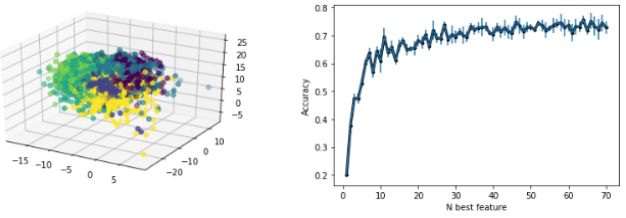


Fig. 23 Left: GRP 3d scatter plot colored by classes Right: importancies vs GRP component

The tradeoff of optimal number of feature and accuracy is reached at 66 feature.
Accuracy: 0.75 ± 0.02

TSNE

Embending technique to visualize high diminsional data.
The number of feature can't go further than 3. Anyway, the optimal number is 2.
Accuracy: 0.86 ± 0.06

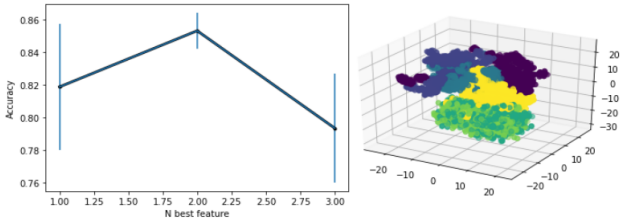
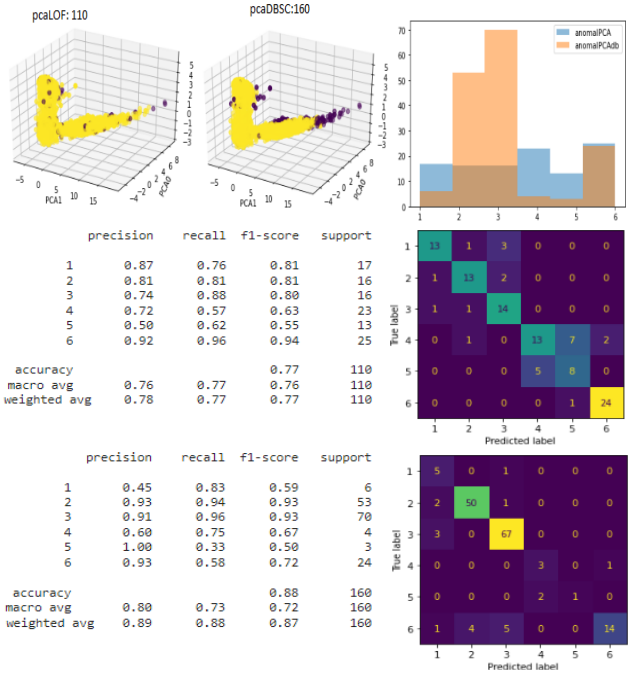


Fig. 24 Left: importancies vs TSNE component Right: TSNE 3d scatter plot colored by classes

2.5.3 Outliers improvement

Due to the lowering on the number of feature, thanks to the feature extraction, we can now use outliers detection algorithms for low dimensions like the Local Outlier Factor and DBSC that now could be more useful.
We have applied those techniques to all the reduced datasets and like in Section 2.3, we tried to classify them.
Hyperparameters are set to found $\sim 3\%$ of the data as outliers.

PCA



	precision	recall	f1-score	support
1	0.87	0.76	0.81	17
2	0.81	0.81	0.81	16
3	0.74	0.88	0.80	16
4	0.72	0.57	0.63	23
5	0.50	0.62	0.55	13
6	0.92	0.96	0.94	25
accuracy			0.77	110
macro avg	0.76	0.77	0.76	110
weighted avg	0.78	0.77	0.77	110

	precision	recall	f1-score	support
1	0.45	0.83	0.59	6
2	0.93	0.94	0.93	53
3	0.91	0.96	0.93	70
4	0.60	0.75	0.67	4
5	1.00	0.33	0.50	3
6	0.93	0.58	0.72	24
accuracy			0.88	160
macro avg	0.80	0.73	0.72	160
weighted avg	0.89	0.88	0.87	160

Fig. 25 LOF and DBSCAN outlier detection and classification for PCA dataset

ISO

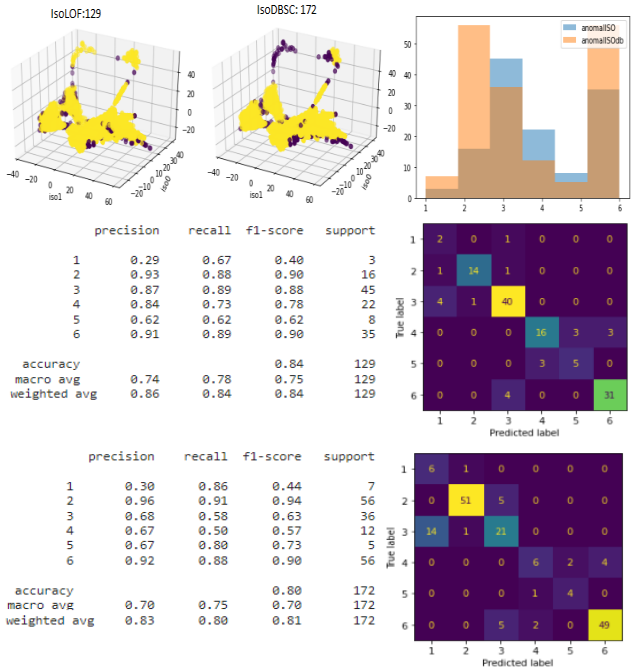


Fig. 26 LOF and DBSCAN outlier detection and classification for ISO dataset

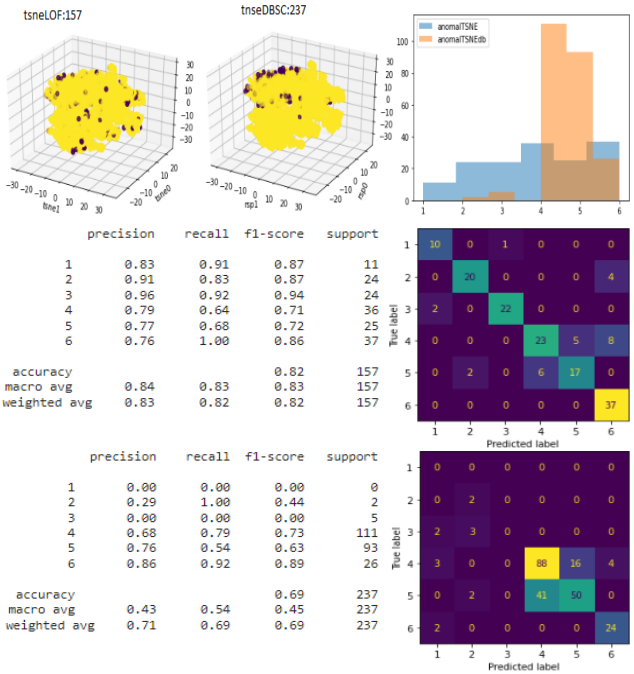


Fig. 28 LOF and DBSCAN outlier detection and classification for TSNE dataset

This time, all the methods return outliers hard to classify (exception for the outliers of the 6 class).
We decided to eliminate those outliers common to most of the algorithms.
We take into account each possible couple of outliers-sets (for example a couple would be outliers from the PCA dataset calculated with the LOF and the LOF outliers from the TSNE dataset or the DBSCAN outliers from the PCA).
The number of couples is the binomial coefficient of the number of outliers-sets and 2.
For what we did in section 2.3, there were 3 methods (DBSC is omitted due to the easy classification of its outliers). That means 3 outliers sets and 3 possible couples.
We found 499 points which are actually shared among these 3 couples but only 215 points are predicted as outliers by all 3 methods as we can see from the left figure of Fig. 29.
In this section, we found 8 more outliers-sets for a total of 55 possible couples.
In total there were 682 points shared by at least by 2 outliers algorithms.
From that we took only those common to at least 20 of the couples, as for majority vote, finding 88 points.

Gaussian Random Projection

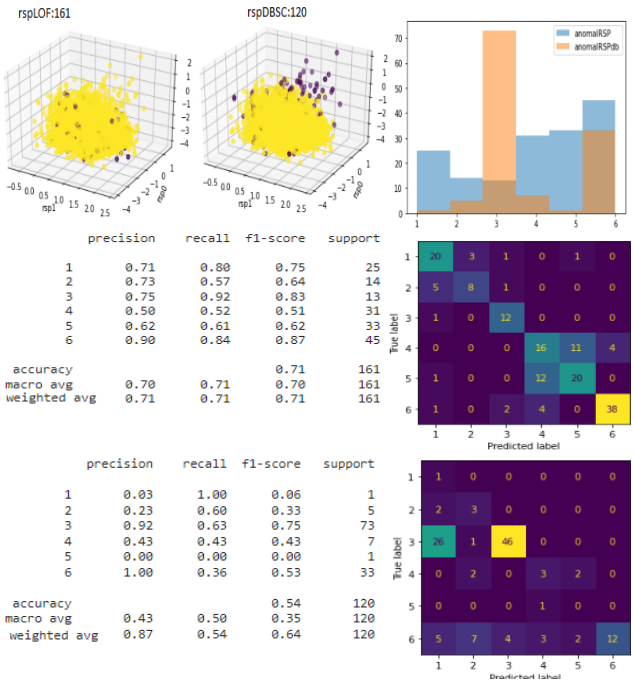


Fig. 27 LOF and DBSCAN outlier detection and classification for GRP dataset

TSNE

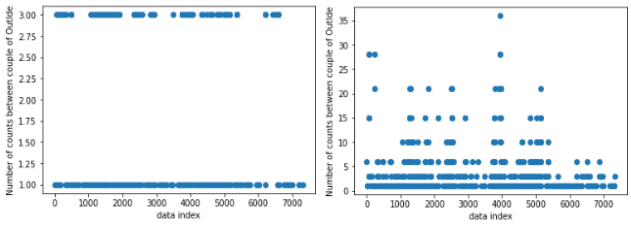


Fig. 29 Left: outlier count among couple of outliers sets in section 2,3 vs outlier index
Right: outliers count among all couple of sets vs outliers index

2.6 conclusion

The dataset feature space has been immediately reduced by eliminating the feature with correlation 1, although we've seen how removing general high correlation feature doesn't necessary mean an improvement.
We've seen how the classification among stationary and movement activity is very easy, confirmed by the 3dimensions PCA plot which shows how those data represent different areas in space.
We wanted to use the 31 feature set, SelectFromModel(DT), since it is the one which the DT gave the best result with less standard deviation, hence more stable.
The 88 outliers chosen from majority vote are removed from the dataset.

3 Module 2

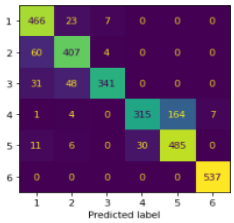
3.1 Advanced Classification Methods

In this section we used the advanced classification methods, tuning the hyperparameters with a GridSearch.
We want to define new classification tasks.
We have seen in the previous section how stationary and movement activity are well separated. Moreover class 6 is always well predicted inside the stationary activity.
Hence for each advanced classifier we classified, beyond the general multiclass classification, every couple in the movement activity and the couple sitting(4)-standing(5) for the stationary ones.
The dataset is balanced over the classes, so the accuracy and the ROC-curve are enough to describe the level of the algorithm.

3.1.1 NaiveBayesClassifier

We assume equal prior to all classes, which is reasonable since they are equally distributed.
The algorithm assumes a gaussian distribution for the likelihood of the feature.
We run the namely NormalTest from the scipystats library, which returns a 2-tuple of the chi-squared statistic, and the associated p-value. Given the null hypothesis that the data came from a normal distribution, the p-value represents the probability that a chi-squared statistic that large (or larger) would be seen. Hence small value of the p-value means a low probability of the data to be Gaussian.
Our 31 feature dataset has no feature with a p-value over 0.05, hence we can assume none of them are normally distributed.
Anyway we run the GaussianNB classifier anyway for which results aren't too bad.
Performing the Normal Test on the full dataset 8 feature with high p value are discovered. However the classification on this dataset gave poor results.
Multiclass classification for our dataset and the gaussian one:

	precision	recall	f1-score	support
1	0.82	0.94	0.88	496
2	0.83	0.86	0.85	471
3	0.97	0.81	0.88	420
4	0.91	0.64	0.75	491
5	0.75	0.91	0.82	532
6	0.99	1.00	0.99	537
accuracy			0.87	2947
macro avg	0.88	0.86	0.86	2947
weighted avg	0.88	0.87	0.86	2947



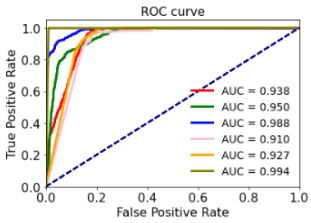


Fig. 30 SelectFromModel dataset
Top: Report, confusion matrix and ROC curve for multiclass classification

	precision	recall	f1-score	support
1	0.32	0.32	0.32	496
2	0.24	0.30	0.27	471
3	0.24	0.30	0.26	420
4	0.35	0.22	0.27	491
5	0.22	0.22	0.22	532
6	0.30	0.28	0.29	537
accuracy			0.27	2947
macro avg	0.28	0.27	0.27	2947
weighted avg	0.28	0.27	0.27	2947

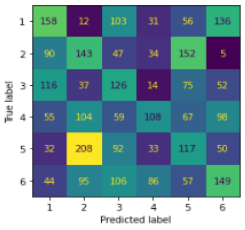


Fig. 31 Gaussian feature dataset
Report and confusion matrix for multiclass classification

While for the binary classification:

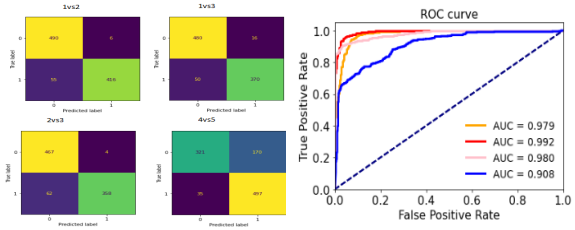


Fig. 32 SelectFromModel dataset
ROC curve and confusion matrix for the couples of classification

3.1.2 SupportVectorMachine

The parameter are:

- Kernel: the type of decision boundary. Best one: Linear
- C: tells how much you want to avoid misclassifying each training example. Is the inverse of the regularization strength. Best one: 15
- gamma: this parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. Best one: scale (which is the inverse of the number of feature times their variance.

In order to visualize the action of the kernel, we plot the decision boundary with different kernels on the 2d PCA components.
From this representation is clear how linear kernel better performs on this dataset giving the best splitting for the data.

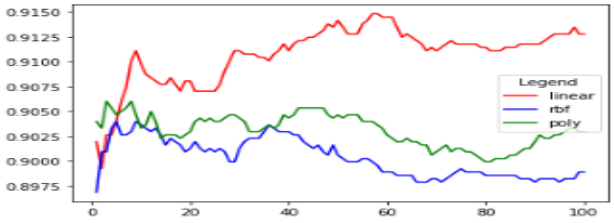


Fig. 33 SVM decision boundary based on different kernel on the 2d PCA components

For the action of the C parameter, we plot the average accuracy reached for each kernel, varying the C parameter (sigmoid kernel is not plotted since the accuracy level was much lower).

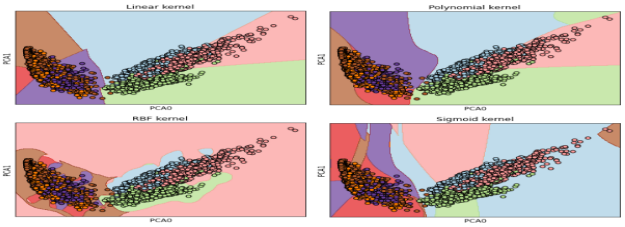


Fig. 34 Accuracy level vs C parameter for different kernels.
Dataset is the usual SelectFromModel set

The shape of those curves reach are pretty similar. They reach a plateau along with they fluctuate and than fall, hence we chose as best values the beginning of the plateau, 15.
The multiclass classification shows:

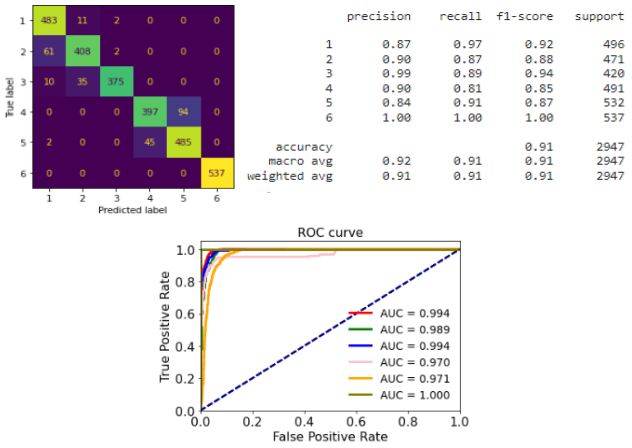


Fig. 35 SVM confusion matrix, report and ROC curves for multiclass.

While the binary classification picking couple of activities:

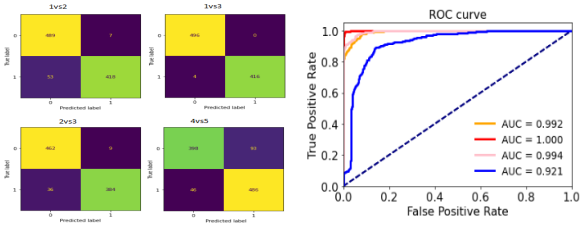


Fig. 36 SVM confusion matrix, report and ROC curves for binary classification.

3.1.3 LogisticRegression

The parameter are:

- Solver: the optimization algorithm to use. Best one: SAGA which is the fastest.
- Penalty: to create soft margin on the boundary of the classes, allowing errors. Best one: L2.
- C: tells how much you want to avoid misclassifying each training example. Is the inverse of the regularization strength. Best one: 2.

Multiclass:

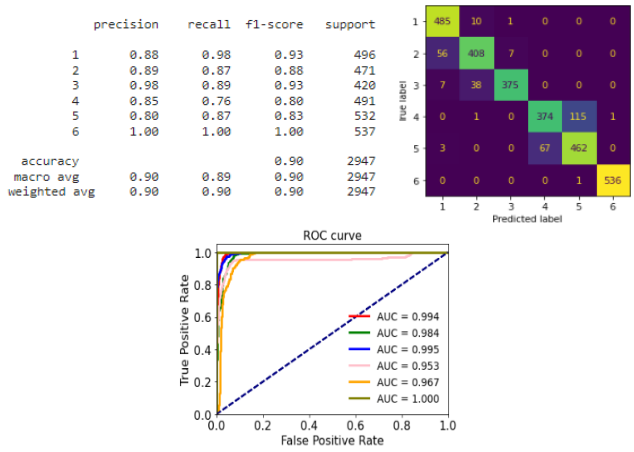


Fig. 37 LogisticRegression confusion matrix, report and ROC curves for multiclassification.

While for the couple of activities:

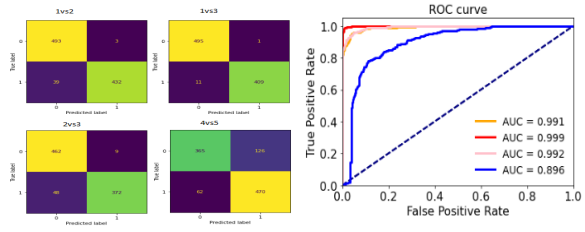


Fig. 38 LogisticRegression confusion matrix, report and ROC curves for binary classification.

3.1.4 RandomForrest

The hyperparameters found with the GridSearch are:

- Criterion: the impurity score for the tree. Best one: gini
- Max feature: random feature to look for splitting in each tree. Best one: log2

The number of estimator is varied manually as well as the depth of the tree to avoid overfitting.

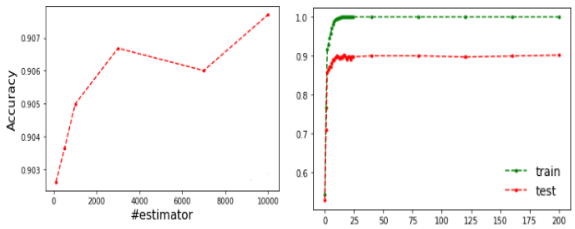


Fig. 39 Random forest
Left: accuracy vs number of estimators
Accuracy vs depth of the tree

From the first plot we can see how increasing the number of estimator the model becomes more accurate and decreases the standard deviation.
From the second plot we can conclude that the overfitting problem is not present maybe because it doesn't matter how deep the tree is, the data are well separated among classes, so that a really deep tree which arrives at accuracy 1 on the train set, manage to correctly classify the data on the test.
So we decided to keep the model as simple as possible, like the Bayes theorem suggest, taking as max depth the beginning of the plateau (i.e. max depth = 20).
We show the feature importancies and the permutation test.

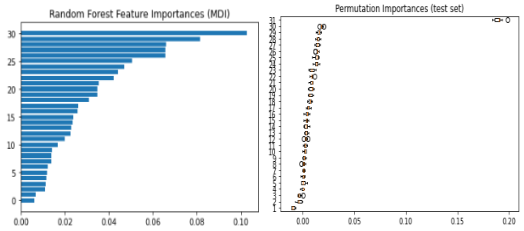


Fig. 40 Random forest
Left: feature importancies
Right: Permutation test

Multiclass:

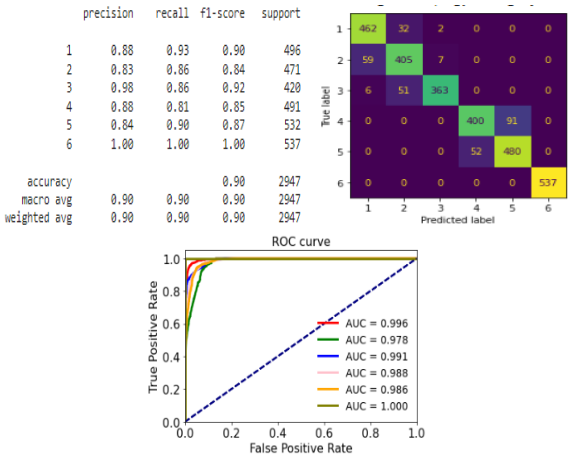


Fig. 41 Random forest
report, confusion matrix and ROC curve, multiclassification

Binary classification:

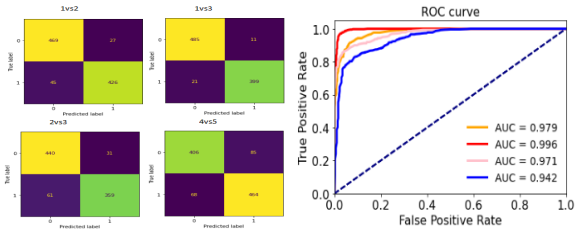


Fig. 42 Random forest
report, confusion matrix and ROC curve, binary classification

3.1.5 Boosting classifiers

This kind of classifiers, together with the RandomForest, belong to the class of ensemble classifiers. We investigated the Bagging, AdaBoost and XGB.
They all use by default the decision tree and the hyperparameters we can tune are the algorithm, the learning

rate and the number of estimators.

In general, increasing the number of estimator increases the time of execution.

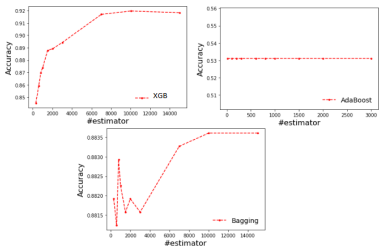


Fig. 43 Accuracy vs number of estimators for different algorithms

We see by plotting the accuracy to respect to the number of estimators that increasing this value correspond to an increase and a stabilization of the accuracy. We notice how AdaBoost is actually really stable even for a low number of estimator in contrary to the Bagging and XGB algorithm.
The learning rate is a trade off in respect to the number of estimators. For the AdaBoost seems actually independet while for the other 2 algorithm, (setting the value of the number of estimators at its plateau level) gives better result with a learning rate of 0.8.
Results are very similar, although a little worse, than the random forest.

3.1.6 Neural networks

We explore 2 kind of neural networks. The one presented by the sklrn library, the MPL classifier, and the Keras one.

MLP
To construct the neural network we start by looking for the best hyperparameters without hidden layers and then investigate the variability of the output in relation of the number of percepetrons and layers.
The grid search, with an early stopping based on the accuracy of the test set, return:

- alpha: the strenght of the L2 regularization. Best one: 0.0001
- batch size: the number of sample to be computed togheter. Best one: 32.
- Learning rate: best one, Invscaling. Meaning that the learning strenght decrease with time.
- Solver: the solver for weight optimization. Best one: Adam
- Activation function: Tanh.

Given the best hyperparameters, we plot the accuracy and loss level in respect to the number of neurons and for different number of hidden layers.

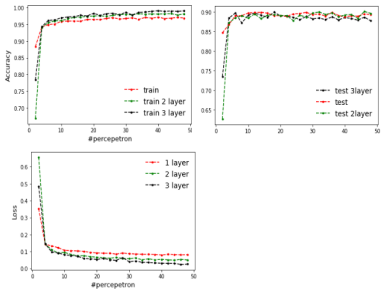


Fig. 44 MLP accuracy and loss vs number of neurons per layer, with different number of layers

Increasing the number of neurons (as for the depth in the random forest) doesn't present the overfitting problem since the test set accuracy reach a plateau. However we can see how the overfitting problem is shift to the number of layers as we can see from the plot where the train set accuracy increases while the test set decreases.
We choose the 1 layer model with 21 neurons.
In the end the model consist in:
-Input layer with 31 neurons.
-1 hidden layer with 21 neurons.
-Output layer with 6 neurons.
The classification results are:

	precision	recall	f1-score	support
1	0.88	0.96	0.92	496
2	0.87	0.87	0.87	471
3	1.00	0.87	0.93	420
4	0.86	0.75	0.80	491
5	0.79	0.89	0.84	532
6	1.00	1.00	1.00	537
accuracy			0.89	2947
macro avg	0.90	0.89	0.89	2947
weighted avg	0.90	0.89	0.89	2947

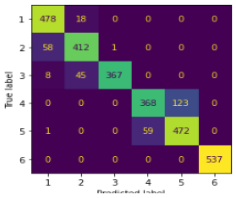


Fig. 45 MLP report and confusion matrix

However since the random weight inizialization this result is not stable and so we averaged over more run to get: accuracy= 0.89 ± 0.01

Keras
We tried on doing the same as before for this kind of network obtaining really poor results.
The first change was to transform the Y label by OneHotEncoder. This function mapped the one dimensional array with 6 values to 6 arrays which values are 0 or 1. Hence the class 1 is mapped to [0,0,0,0,0,1], class 2 to [0,0,0,0,1,0] and so on.
This significantly increases the accuracy swapping the activation function from tanh to softmax. Anyway the results we got were still not enough for a good classification arriving to 0.3 accuracy on the test set at best.
In our trials we decided to change the dataset and surprisingly we got an accuracy of ~ 0.87 using the 11 PCA component and the softmax activation function.
We then plotted as before the response of the model on the accuracy and loss to the number of neurons and layers.

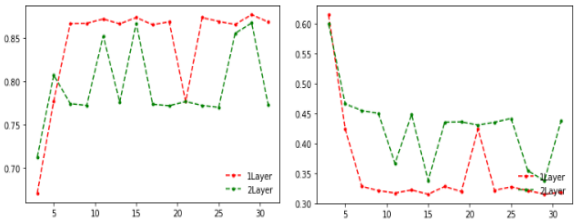


Fig. 46 KERAS neural network, accuracy and loss vs number of neuron per layer and different number of layers

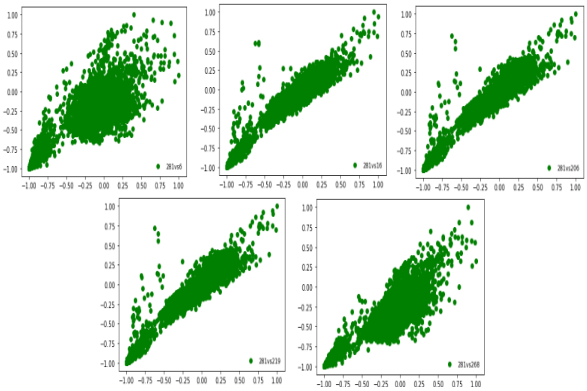


Fig. 47 scatter plot of feature n. 281 on the x axis vs the other high correlated variable

3.1.7 Conclusion

The advanced classification methods shows a solid and stable way for classification.
They all gave good results expecially for the multiclass classification where all the ROC curves reached almost value 1.
For the binary classification we can see how all the methods perform almost perfectly for the movement activities while the hardest task to classify is the difference between sitting and standing where the best classificator was the random forest reaching a ROC curve area of 0.942.

3.2 Linear Regression

In order to solve a linear regression task we first look at the high correlation variables and noticed that many of them shares high correlation with multiple attributes.
We decide to take into account the attribute number 281 namely "fBodyAcc-sma()" which share a correlation over 0.95 with 5 other attributes:

- 6 tBodyAcc-std()-Z
- 16 tBodyAcc-sma()
- 206 tBodyAccMag-sma()
- 219 tGravityAccMag-sma()
- 268 fBodyAcc-mean()-Z

Here the scatterplot between the target attribute 281 and the others:

3.2.1 Univariate linear regression

For this purpose we chose one of the high correlated feature, for example the number 6 and perform an univariate linear regression.
The rectification yield:

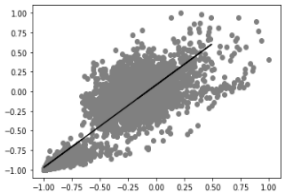


Fig. 48 univariate linear regression

And the prediction based on this inference are quite accurate. For which:
$R2$: 0.856
MSE : 0.029
MAE : 0.116
Graphically:

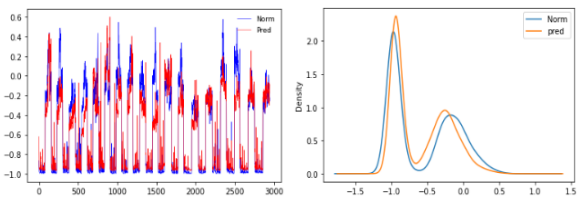


Fig. 49 univariate linear regression
Left: plot of the feature 281 of the test set, real and predicted
Right: Distribution plot for the feature 281, real and predicted

We notice 2 different region which might belong to two macro-classes, hence we split the attributes in the records belonging to the active class and the stationary one.
The result:

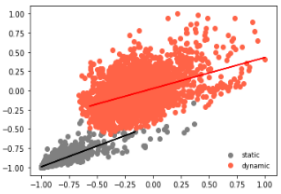


Fig. 48 univariate linear regression splitting stationary and movement activity

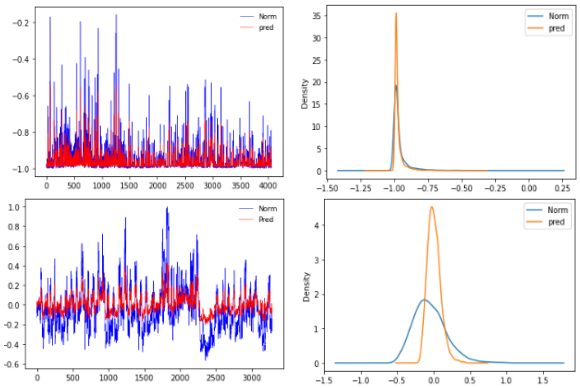


Fig. 49 univariate linear regression splitting stationary and movement activity
Top are stationary, bottom are active.
Left: plot of the feature 281 of the test set, real and predicted
Right: Distribution plot for the feature 281, real and predicted

The error measures in this case:

	Static	Dynamic
R2	0.766	0.528
MSE	0.001	0.021
MAE	0.014	0.118

It seems that splitting the activities lower the overall error since squaring the sum in quadrature of the MSE gives a lower MSE than the total set.
Anyway it won't be done for the multivariate regression since we want to keep this inference as general as possible.

3.2.2 Multivariate linear regression

For a multivariate regression we use Ridge and Lasso algorithm which differ on the objective function to minimize.
We perform the regression for different values of the of the alpha parameter which multiply the l1 and l2 regularization norm (for lasso and rigde respectivly).
In the following the plot of the coefficients and the intercept inferred by the algorithms for smaller values of alpha going from 0.1 to 10^{-8}.

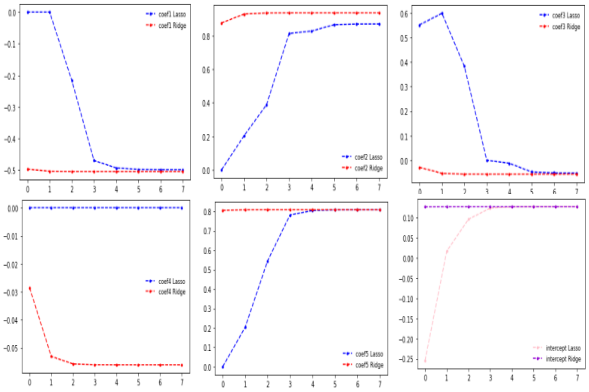


Fig. 50 Coefficients of the other feature inferred and intercept vs alpha values, for RIDGE and LASSO

While for the errors:

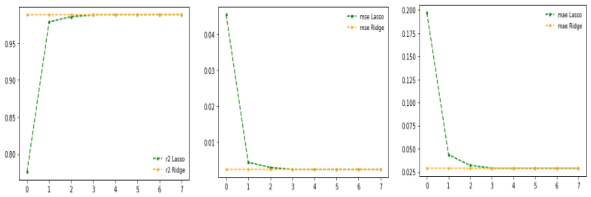


Fig. 51 Error measures vs different values of alpha, for RIDGE and LASSO

For small values of alpha we notice that the two regression become almost the same, exception for the 4th feature considered, which is the number 219, in fact the Lasso algorithm excludes this variable from the regression, setting the weight coefficient to zero.
Anyway the prediction become basically the same so we show only the one from Ridge.

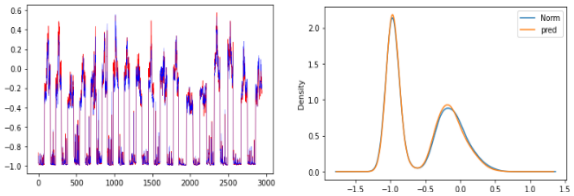


Fig. 52 multivariate linear regression
Left: plot of the feature 281 of the test set, real and predicted
Right: Distribution plot for the feature 281, real and predicted

3.2.3 Gradient boosting regression

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage a regression tree is fit on the negative gradient of the given loss function.
Really important for this algorithm is the number of estimator it is going to use. We plot the prediction for different number of estimator (10,100,1000,10000) and the results are significantly different.

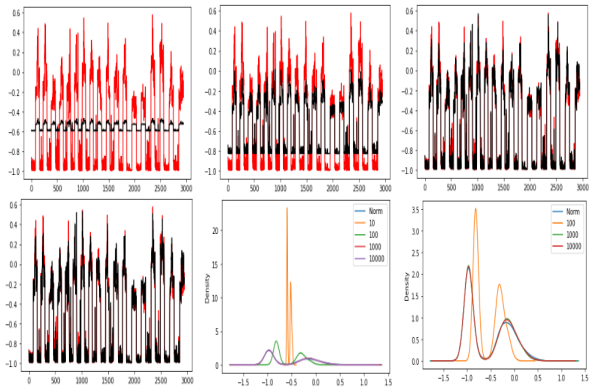


Fig. 53 GB regression plot and distribution using different numbers of estimators

Also we can see from the distribution of the prediction for different loss function with 1000 estimators, that all of them can picture quite well the regression problem but the squared loss is the best one since the R2 is a little higher than the absolute error.

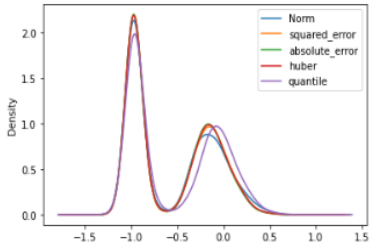


Fig. 54 GB regression distribution plot using different loss functions

4 Time Series Analysis

For this purpose we use the second dataset. This is a collection of the signals from the gyroscope, body acceleration and total acceleration for the 3 different axis. Each of it consists on the time series at equidistant timesteps related to a particular subject performing one of the 6 activities.

4.1 Prepare the dataset

At first approach to understand what we are dealing with, we plot the time series for all the axis of a particular activity for all the kind of signal: body, gyroscope, total acceleration. This is done in order to get the general properties of the various sets.
Afterwards we plot the time series of 5 different realization of the same activity to see the the capture the overall characteristic of one activity.
For example we shows the differences we depict by this analysis looking at the activity 1 and 5.
Here the different sets:

Activity 1:

Fig. 55 Activity 1. TS for all the axis of all the sets

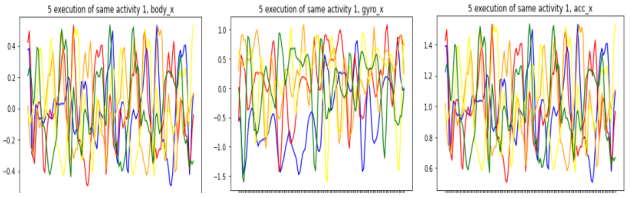


Fig. 56 Activity 1. 5 TSs for the x axe for all the signal sets.

Activity 5:

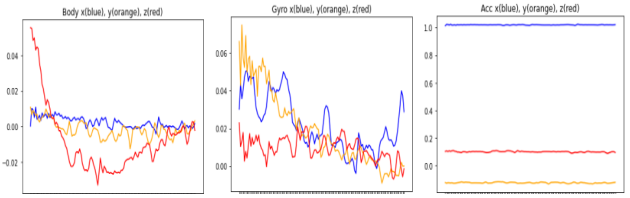


Fig. 57 Activity 5. TS for all the axis of all the sets

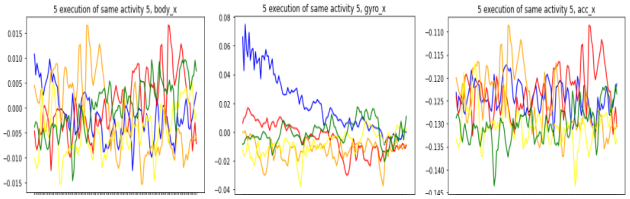


Fig. 58 Activity 5. 5 TSs for the x axe for all the signal sets.

In the activity 1 (which is a motion activity) the signal for all the sets is dominated by big fluctuations.
While in the stationary activity 5, the rumor is significantly less and we can detect more easily a general trend.
For this reason we thought to transform the stationary activity by a noise smoothing and the active ones with a amplitude scaling transformation to better capture the fluctuation.
However that would be unfair since modify the data of different class in a different way would bias the result.
So we thought to noise-smoothing the data belonging to the train set (for a small rolling window of 4 to keep the high fluctuation and discard the small one) and then test it on the untouched test set.
The action of such a transformation is depicted in Fig. 59 which shows the activity 1 (left) and 4 (right) for the body-acc-x signal.

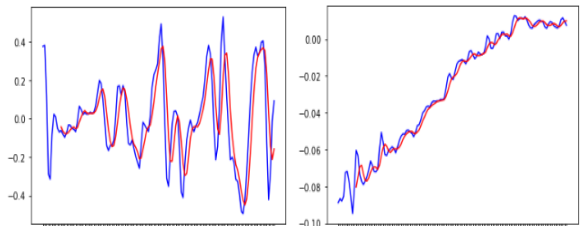


Fig. 59 Acc-body-x
Left: transformed (red) TS activity 1
Right: transformed (red) TS activity 4

To validate such a variable transformation we performed the KNN classifier using the Euclidean and Manhattan distances for both untransformed (blue) and transformed (orange) set.
The optimal number of neighbours depends on the set and we looked for it for every one giving different results.

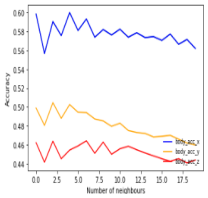


Fig. 60 accuracy vs number of neighbours for different datasets, showing a different best number for each

In the following a bar plot of the accuracy for all sets:

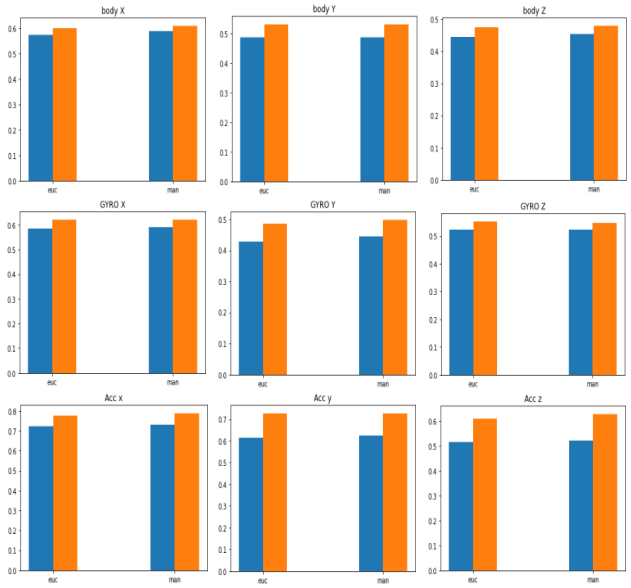


Fig. 61 KNN accuracy for all signal-sets using euclidean and manhattan metrics.
Blue untransformed dataset, orange transformed one.

Manhattan distance is slightly better than the euclidean one, but it still doesn't give good results.
However it is clear how the transformation actually increases the accuracy for all the sets.
By looking at the confusion matrix we see how this transformation allow a better distinction of the two macro-classes. For example we report only the confusion matrix for acc-Body-Y set calculated with the euclidean distance.
Left untouched series, right transformed one.

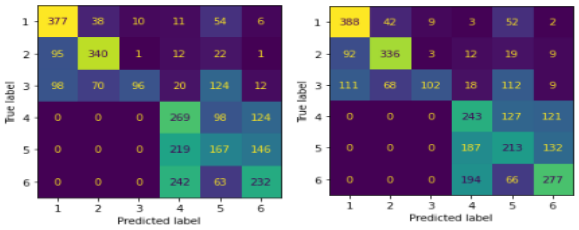


Fig. 62 Confusion matrix
Left: untouched TS
Right: transformed TS.

Now we want to chose the best multivariate time series dataset by combining couple or more signal-sets, increasing the dimensionality of the time series, giving more constrain to the class where each series belong.
In order to do so we should check every combination of the signals sets, however our computer couldn't really handle it, so we picked the sets which gave alone the best accuracy.
Picking the sets with accuracy higher than 0.6 we got the multidimensional dataset joining: body-x, gyro-x, gyro-z and all the total accelerations.
Before clustering it we checked how KNN performed on that.

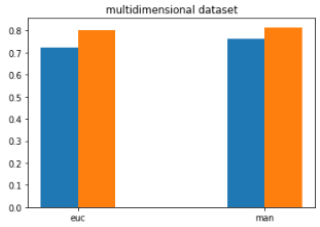


Fig. 63 Bar plot of the accuracy for multidimensional dataset

The results have improved a lot ramping from 0.6 to 0.8 when using multidimensional dataset.
The transformation seems to be useful even in this case.

4.2 Clustering

In order to perform a good cluster we need to choose for the optimal number of clusters. A variable which come at hand for defining how good a clusterization is, is the inertia which is defined as the distance between each data point and its centroid, squaring this distance, and summing these squares across one cluster. A good clustering algorithm has low inertia and a low number of clusters, however this is a trade off since increasing the number of clusters, decreases the inertia.
This can be chosen together with the Silhouette score which give us information on how well separated are the clusters, hence we are looking for a trade off of the highest Silhouette and smallest inertia.
So we used KMeansTimeSeries to cluster our multidi-

mensional dataset and plot the inertia over the number of clusters and the same for Silhouette. We summed the Silhouette of each of the signal set and due to the complexity of the algorithm we ranged from 2 to 8.
Inertia plot give us the expected dynamics, howerver we can notice how the orange line (calculated with the DTW metric) is lower than the blue one (Euclidean) which arroverate the hypothesis that DTW metric is better for TS. (However our computer couldn't handle the clusterization or classification with such algorithm giving errors after hours due to the ending of the RAM space).

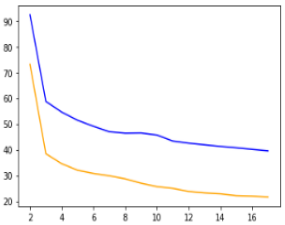


Fig. 63 Inertia vs number of cluster with euclidean and DTW metric

The Silhouette plot is a little more interesting, giving positive values only for 2 and 3 clusters and becoming negative for bigger numbers, meaning that the cluster doesn't manage to classify the points.
We report the plot for 2 clusters and 6:

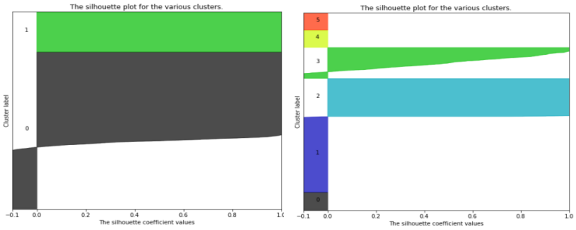


Fig. 65 Silhouette plot for 2(left) and 6(right) clusters.

We thought that with 2 cluster we got the optimal Silhouette score because the KMeans splits the active and stationary activity retrieving this two macroclasses.

2 CLUSTERS

So we plot the 2 centroid to respect to the timestamps for every signal set and plot an histogram of the activity present in each of the two clusters.

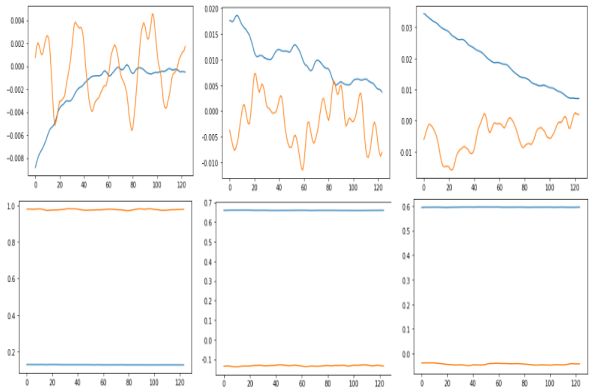


Fig. 66 TS centroid for 2 clusters.

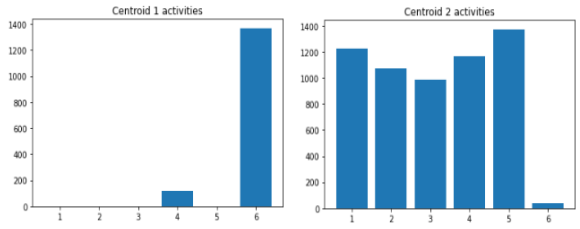


Fig. 67 Histogram of activities within the 2 clusters

So we can actually see how 2 cluster can only distinct what belong to activity 6 (STANDING) and the rest.

6 CLUSTERS

At this point we performed a 6 clusters KMeans to see how the cluster would predict the activity label in this case.
Those are the results for the centroid TS and activity inside the clusters.

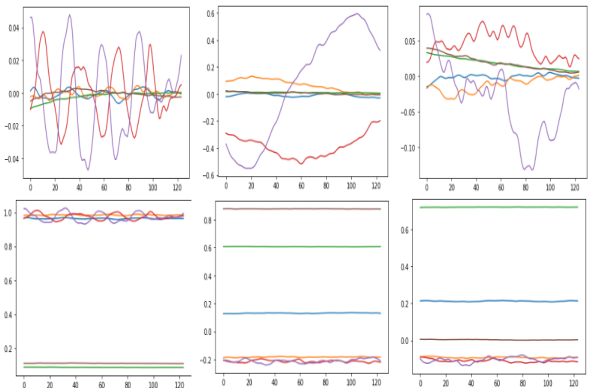


Fig. 68 TS centroid for 6 clusters.

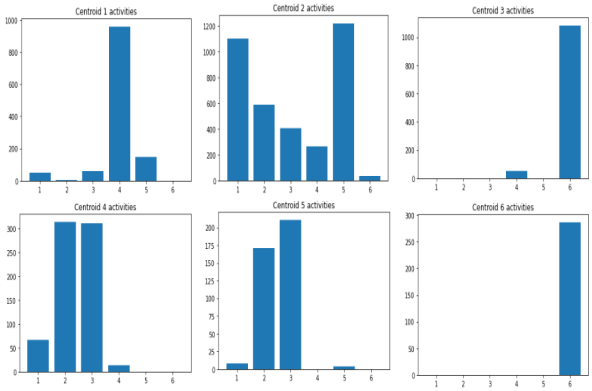


Fig. 69 Histogram of activities within the 6 clusters

We notice how walking on the stairs is completely mixed over the 4 and 5 cluster. The second cluster has a similar behaviour of the 1 cluster in the previous case, recognizing only what's not standing. The cluster 3 and 6 should be merged together since they retrieve only the standing activity.

4.3 Time series approximation

In order to improve and speed up the process it is possible to use some approximation to the time series. We performed the DFT and PAA.

4.3.1 DFT

DFT is very useful for signal understanding but we have to chose the optimal number of coefficient to keep in count when transforming.
Hence we plot the coefficient for all the sets we use and see which is the optimal number to cut the tail of the series.
Top: body-acc-x,gyro-acc-x,gyro-acc-z
Bottom: all axes of total acc

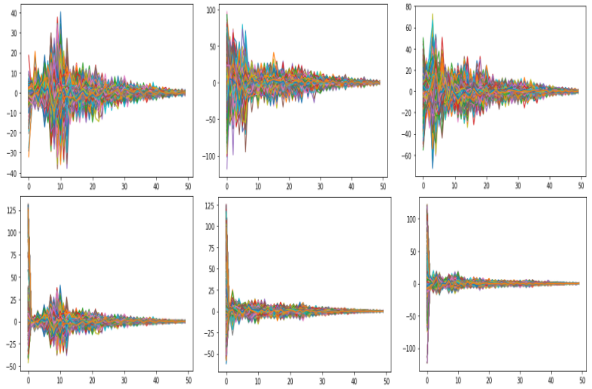


Fig. 70 Fuorier coefficients values vs fuorier components

There are less important components in the total acceleration axis (that was clear even from the normal plot since their were smoother) but we decided to truncate the series at 35 to not discard important feature from the other signals.
Performing the KMeans with 6 clusters as before we get:

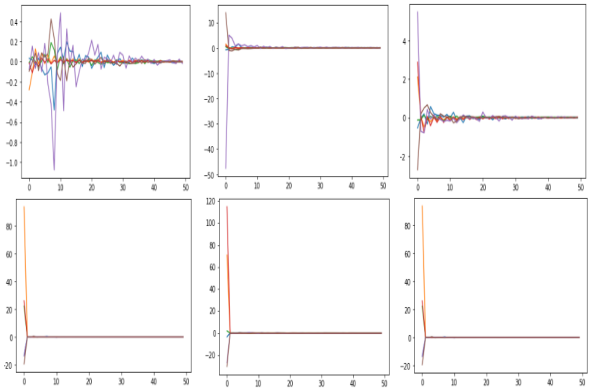


Fig. 71 TS centroid for 6 clusters DFT.

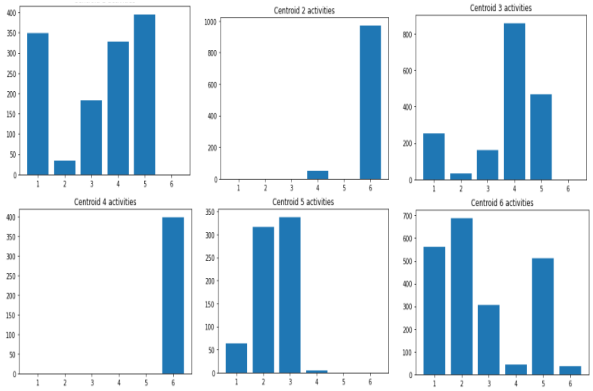


Fig. 72 Histogram of activities within the 6 clusters DFT

From the histogram we can see we achieved pretty similar result to the normal dataset.

4.3.2 PAA

We decided to use 35 segments to represent the PAA transformation. The heuristic reason for that is that if the DFT needs 35 sine and cosine components to well represent the data, the same number can fit for the PAA.
We show one representative example:

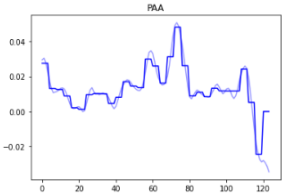


Fig. 73 TS acc-body-x with PAA transformation.

The results of the Kmeans:

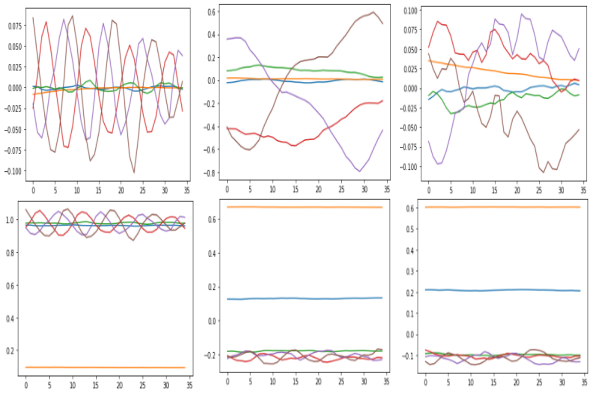


Fig. 71 TS centroid for 6 clusters PAA.

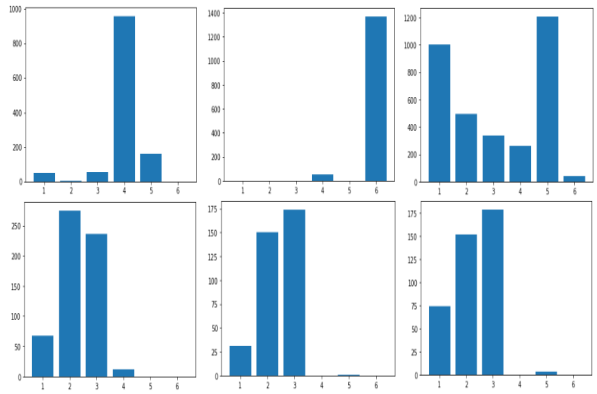


Fig. 72 Histogram of activities within the 6 clusters PAA

This transformation gave the best results since it manage to cluster together almost all class 4 and 6. However the 3rd cluster, top-right, group most of data except a class 4 and 5 which are sparse over the other clusters.

4.4 Motif and anomalies

To better investigate the nature of the time series motif and anomalies are useful tools based on the matrix profile of a time series.
So computing the matrix profile with a STOMP algorithm which complexity in time and space is O(N). We decided to compare the motifs and anomalies investigation on 3 series belonging to the same activity class but performed by different subjects.
The set is body-acc-x, the subjects are the number 1,3,5 and the activities analyzed are the walking activity (1) and the the sitting one (4).

The plots show time series with colored mofits and in violet the anomalies.
The expectation is to find motifs similar for all the subjects which are gonna be representative of the class, hence the shapelets.

4.4.1 Sitting activity

The window size for the matrix profile is set to 10 due to the low and rapid oscillation of this activity.
The motifs are of the same size while the anomalies are set to be of size 3 since we expect that anomalies are short set of data.

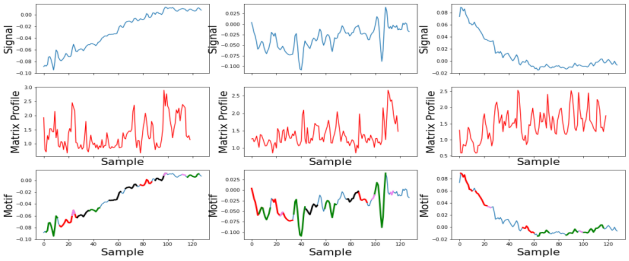


Fig. 73 TS with motifs and anomalies class 4

The signals looks very different and so do the matrix profile. However the Motifs seems to charaterize two kind of oscillation, the big one in green, absent in the 3 graphs, and smaller oscillation in red and black.

4.4.2 Walking activity

For the walking activity the size window is increased to 18 since the peaks we see in the times series are distributed over that time steps.

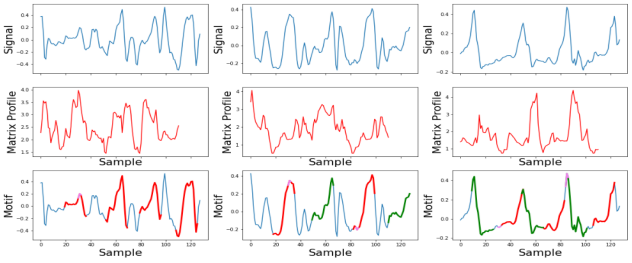


Fig. 74 TS with motifs and anomalies class 1

The red motifs seems to try to depict the shark fin shape while the green one the rise or fall with oscillations.

4.5 Shapelets

Shapelets discovery can be used to infer the class of the test set.
As for the motifs and anomalies (we haven't said that before) the dataset we used is the multidimensional one but without the smoothing process since for the recognition of particular activity the exact shape of the dataset can be important.
In fact, the prediction based on the shapelets on the body-acc-x set gave an accuracy of 56,5 for the transformed one and 58.7 for the untouched.
Next, a plot of the accuracy on the multidimensional dataset respect to the max shapelets length and the number of shapelets to find per lenght.
We first checked the best length. Then we looked for the best number per length, using the best length found. In doing so we are supposing that their relation to the accuracy is monotonic or independent (which is not the case) but due to the long execution time that was the only idea we had.
Left the different maximum length, right different number.

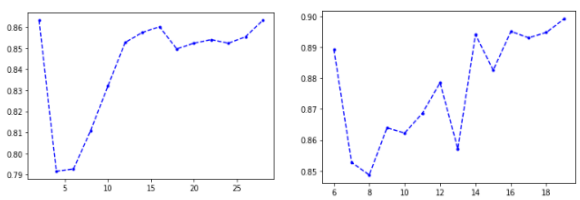


Fig. 75 SHAPELETS
Left: accuracy vs max length of the shaplets
Right : acccuracy vs number per length

Setting the max length at 25 and the number per length at 18 we infer on test set getting the following results.

	precision	recall	f1-score	support
1	0.95	0.95	0.95	496
2	0.91	0.90	0.91	471
3	0.92	0.93	0.92	420
4	0.81	0.77	0.78	491
5	0.79	0.87	0.83	532
6	1.00	0.95	0.97	537
accuracy			0.89	2947
macro avg	0.90	0.89	0.89	2947
weighted avg	0.90	0.89	0.89	2947

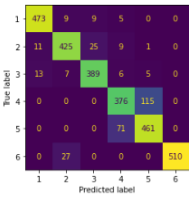


Fig. 76 report and confusion matrix with the shapelets inference

4.5.1 Shapelets transformation

The classification techniques used in section 2 can be applied to our multidimensional time series once we transform the data to respect to the shapelets found. This transformation bring the dataspace from dimension "7352record × 128timesteps × 6 sets" to "7352record × 18 features".

Decision Tree
Setting the hyperparameters with a grid search the accuracy reached is about 0.88.
The classification report:

1	0.90	0.84	0.87	496
2	0.85	0.90	0.88	471
3	0.86	0.90	0.88	420
4	0.84	0.78	0.81	491
5	0.80	0.84	0.82	532
6	1.00	1.00	1.00	537
accuracy			0.88	2947
macro avg	0.88	0.88	0.88	2947
weighted avg	0.88	0.88	0.88	2947

Fig. 77 report DT shapelets dataset.

KNN
Optimizing the number of neighbours which is under 5, the KNN algorithm is used with 3 different metric, euclidean and DTW. The results are slightly different. DTW metric increases the accuracy by 0.07%, from 90.12% → 90.19%
Classification report euclidean (left) and DTW (right).

	precision	recall	f1-score	support
1	0.97	0.94	0.95	496
2	0.93	0.93	0.93	471
3	0.94	0.97	0.95	420
4	0.78	0.78	0.78	491
5	0.80	0.80	0.80	532
6	1.00	1.00	1.00	537
accuracy			0.90	2947
macro avg	0.90	0.90	0.90	2947

	precision	recall	f1-score	support
1	0.97	0.94	0.95	496
2	0.94	0.93	0.93	471
3	0.94	0.97	0.95	420
4	0.78	0.78	0.78	491
5	0.80	0.80	0.80	532
6	1.00	1.00	1.00	537
accuracy			0.90	2947
macro avg	0.90	0.90	0.90	2947

Fig. 78 report KNN shapelets dataset.
Left: euclidean metric
Right: DTW metric.

CNN
To set the hyperparameters for the CNN we moved as in section 2. First with a grid search we looked for the best activation function and optimizer and then checked for how deep it should be.
As in section 2 the class variable has been trasformed with a OneHotEncoder.
We constructed 2 CNN. One for the shapelets transformed space with 18 neurons in input. Another for the full multidimensional dataset with 128 × 6 inputs. In both cases the network has been trained for 10 epochs which were sufficient to reach the plateau. They both needed 1 hidden layers with different number of filters and kernel size but both reached a peak using as activation fuction the 'relu' at the begining and 'softmax' for the last layer.
In the full dataspace CNN manage to outperform the other gaining 2% of accuracy.
The classification report for the full dataset (left, 91% accuracy) and the shapelets trasformed (right, 89% accuracy).

	precision	recall	f1-score	support
0	1.00	0.95	0.98	496
1	0.92	0.91	0.91	471
2	0.86	0.97	0.91	420
3	0.82	0.82	0.82	491
4	0.88	0.83	0.85	532
5	1.00	1.00	1.00	537
accuracy			0.91	2947
macro avg	0.91	0.91	0.91	2947

	precision	recall	f1-score	support
0	0.99	0.86	0.92	496
1	0.84	0.93	0.88	471
2	0.95	0.91	0.93	420
3	0.73	0.90	0.81	491
4	0.93	0.74	0.83	532
5	0.96	1.00	0.98	537
accuracy			0.89	2947
macro avg	0.90	0.89	0.89	2947

Fig. 79 report CNN shapelets dataset.
Left: full dataset
Right: shapelets transformed.

LSTM NN
The same we did for the LSTM NN.
In this case the full dataset is really computational expensive and give worst result than the transformed one.
Full dataset (left, 87% accuracy), and transformed one (right, 89% accuracy).

	precision	recall	f1-score	support
0	0.95	0.83	0.89	496
1	0.94	0.82	0.87	471
2	0.72	0.95	0.82	420
3	0.71	0.94	0.81	491
4	0.93	0.64	0.75	532
5	1.00	1.00	1.00	537
accuracy			0.86	2947
macro avg	0.87	0.86	0.86	2947

	precision	recall	f1-score	support
0	0.98	0.89	0.93	496
1	0.89	0.92	0.91	471
2	0.85	0.96	0.90	420
3	0.78	0.86	0.82	491
4	0.86	0.77	0.81	532
5	0.99	0.96	0.97	537
accuracy			0.89	2947
macro avg	0.89	0.89	0.89	2947

Fig. 80 report LSTM shapelets dataset.
Left: full dataset
Right: shapelets transformed.

ROCKET and MINIROCKET
Those use convolutional kernels to transform the time series.
For both, the dispenser suggested to use the linear RidgeClassifier to classify the transformed data when the set has < 20000 samples and so we did.
Rocket needs at least 15000 kernels to obtain a decent accuracy score (it reaches 69%), same for miniRocket which find the best accuracy (80%) setting the kernel dilatation at 16.
The classification report for both Rocket (left) and MiniRocket (right).

	precision	recall	f1-score	support
1	0.73	0.87	0.79	496
2	0.82	0.50	0.62	471
3	0.69	0.74	0.71	420
4	0.55	0.40	0.46	491
5	0.57	0.88	0.70	532
6	0.76	0.63	0.69	537
accuracy			0.67	2947
macro avg	0.69	0.67	0.66	2947

	precision	recall	f1-score	support
1	0.69	0.82	0.75	496
2	0.68	0.69	0.69	471
3	0.73	0.67	0.70	420
4	0.84	0.77	0.81	491
5	0.84	0.85	0.84	532
6	1.00	0.95	0.97	537
accuracy			0.80	2947
macro avg	0.80	0.79	0.79	2947

Fig. 81 report ROCKET (left) and MINIROCKET (right) shapelets dataset.

5 Sequential Pattern Mining

Our goal is to find frequent sequential pattern which could identify the nature of the activity which they refer to.
Hence we divided each signal set into 6 subsets, one for each class.
So for example body-acc-x will split in 6 subsets body-acc-x-1, body-acc-x-2 and so on.
Since SPM is basen on a item count and our dataset is made of sample of a continuous attribute the data has been transformed by the SAX algorithm.
The hyperparameters are tuned to better represent all the classes, of course they are the same for all datasets for a fair comparison.
The best values we found are:

- segment number: 40
- symbol number: 8

A plot of 3 different time series from body-acc-x of each activity to sketch the problem.

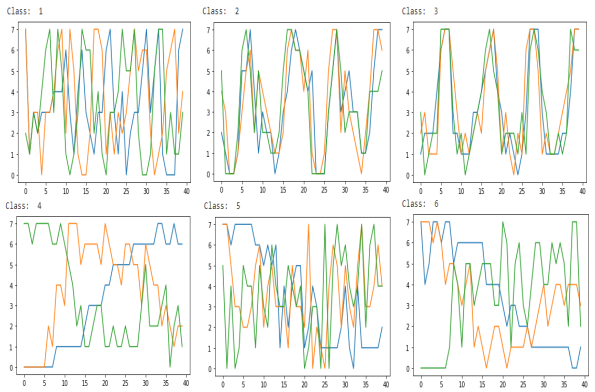


Fig. 82 three TS for every class after SAX transformation

They clearly follows particular paths for each activity.

5.1 Number of patterns

The algorithm we used for SPM is SPAM for frequent patterns and BIDE+ for the closed one.
In the following a graph of the variation of number of patterns vs length and minimum support.

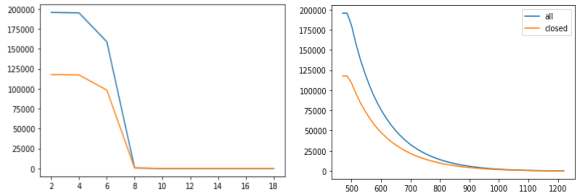


Fig. 83 Number of patterns found vs max length(left) and min-sup (right)

5.2 MAX GAP

From the graphs at the beginning of the section different class have different periodicity.
So we can use the parameter "max-gap" to constrain the search for the frequent patterns to the one which items are close to each other to respect to their periodicity.
By varying the max gap we looked for the patterns with the 90% of support and min-2 length sequences for all sets of class 1.

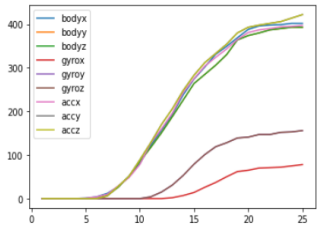


Fig. 84 number of patterns for class 1 and every set varying the max-gap between items in the pattern

Looking at the graph over 20-maxgap the number of patterns reach a plateau since we retrieve all the patterns with no more constrain.
Plotting again the number of patterns per different length we see:

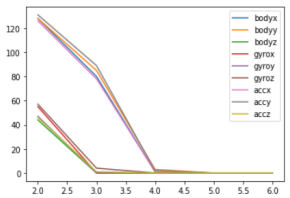


Fig. 85 number of patterns for class 1, every set, max gap set to 20, varying the length.

The sets with more patterns shows even longer patterns. This with Fig. 84 can tell us that different sets can be more predictive to a particular class than others.
In fact doing the comparing the variation of the number of patterns on the same set over the various classes to respect of the max gap as before we see:

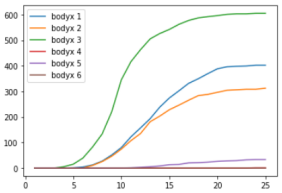


Fig. 86 number of patterns for all classes vs max gap.

This clearly shows the difference in the periodicity of the different activities. In fact the curve start to raise faster for the movement activities which shows high oscillation and hence a low max gap, in contrary with the stationary ones.
The most of the information is before crossing the S shape so we took the best value for each curve.
We show again the min length and min sup graph.

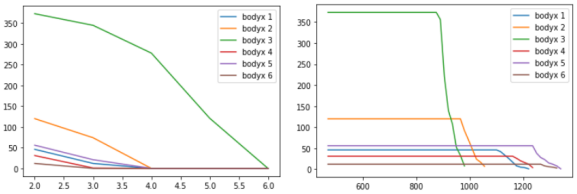


Fig. 87 acc-body-x every class
Left: min length
Right: min sup

The min sup graph shows us the total support of each class range from 900 to 1200.
As we can see the number of frequent patterns for stationary acc-body-x class return ¡50 sequences but those are really stable since they have really high support.
The min-length graph tells us that for the class for which the set is more sensible the pattern lengths increases giving more solid information.
In the following a table with the counts of the same sequences within the frequent one among different classes. In the bracket the total number of frequent sequences.

class	1	2	3	4	5
1 (46)					
2 (120)	41				
3 (373)	35	69			
4 (31)	13	21	11		
5 (56)	16	26	16	30	
6 (12)	7	8	3	12	11

Hence once again the acc-body-x seems to be more representative of class 3 since has the smallest ratio of number of equal pattern and number of sequences.
We show the most common sequences for class 1 and 3 with higher support.

Pattern class 1 (sup)	Pattern class 3 (sup)
$[6, 7, 1], 1162$	$[7, 1, 7], 971$
$[1, 2, 6], 1149$	$[7, 2, 7], 971$
$[7, 2, 6], 1148$	$[2, 2, 7], 970$
$[2, 6, 7], 1148$	$[2, 2, 1], 965$
$[5, 7, 1], 1135$	$[1, 1, 2, 2, 7], 888$

So in case of recording those values within a certain time is more likely to belong to such class.
So for example for our purpose since we're dealing with signals from the cellphone for activity recognition, the device could dynamically store the signals for this time length and infer on the activity being performed.

6 Advanced Clustering

In this section the dataset left in section 2 is retrieved and different clustering algorithm are performed on that.
Comparing the normal KMeans algorithm with the other advanced methods.
Those have been applied even to the 10 componets PCA dataset for a better grafical representation.

6.1 KMeans

Through a grid search the hyperparameters are set as:
-centroid inizializer: random
-algorithm: lloyd
For the best number of neighbours we looked at the Silhouette score.

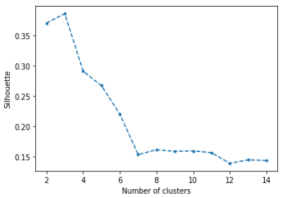


Fig. 88 Silhouette score Kmean

The best silhouette score is reached for number of neighbours set to 3.
Anyway we performed the algorithm even with 6 cluster to see how the points would be grouped.

3 clusters
Reduced dataset:

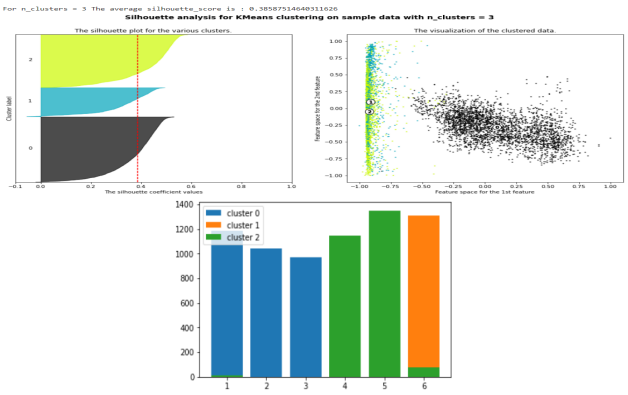


Fig. 89 Silhouette score 3 cluster and class histogram

PCA dataset:

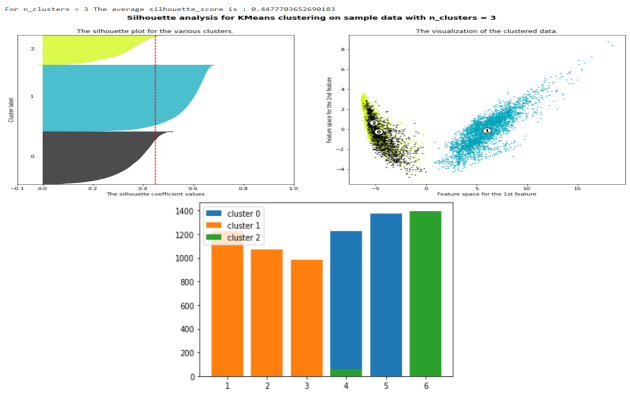


Fig. 90 Silhouette score 3 cluster and class histogram, with PCA dataset

6 clusters

Reduced dataset:

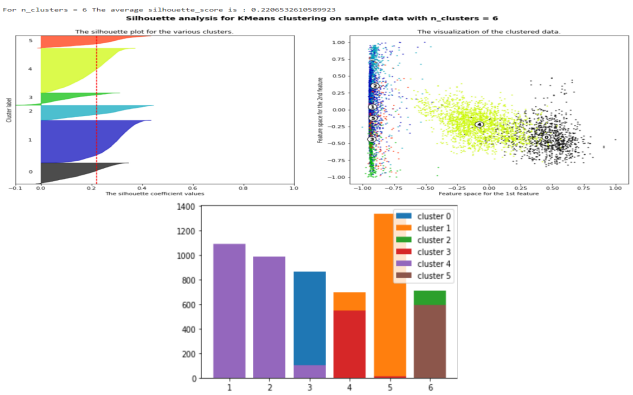


Fig. 91 Silhouette score 6 cluster and class histogram

PCA dataset:

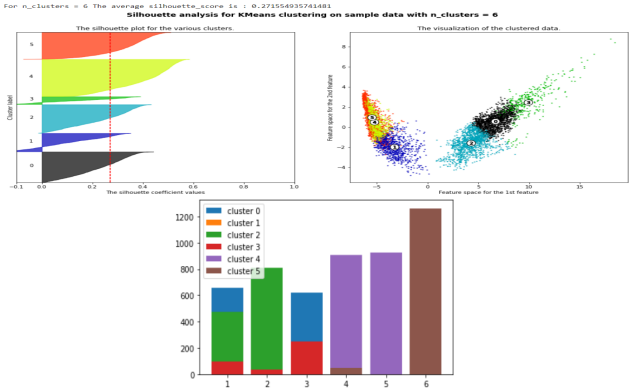


Fig. 92 Silhouette score 6 cluster and class histogram, PCA dataset

Since the PCA represantation offers a better visualization and better clusters for the other algorithms we will show only those graphs.

6.2 XMeans

Silhouette graph for both BIC and MNDL criterior for the information gain.

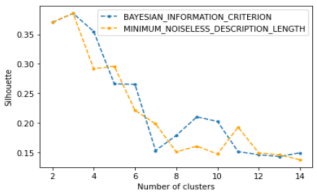


Fig. 93 Silhouette score XMeans with 2 score function

From the graph is not clear which perform better hence we use the BIC criterior.

The Silhouette plot, cluster and histogram for this algorithm:

3 clusters

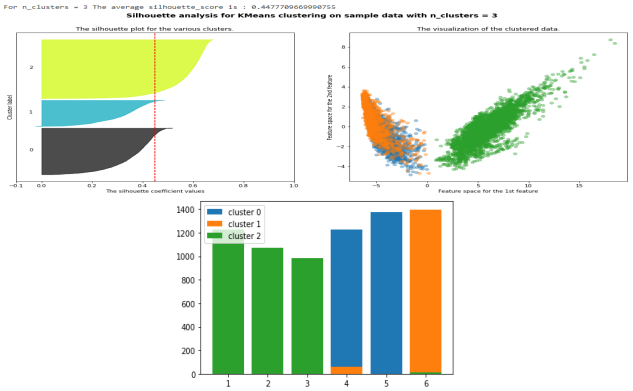


Fig. 94 Silhouette score XMeans and class histogram 3 cluster and PCA dataset

6 clusters

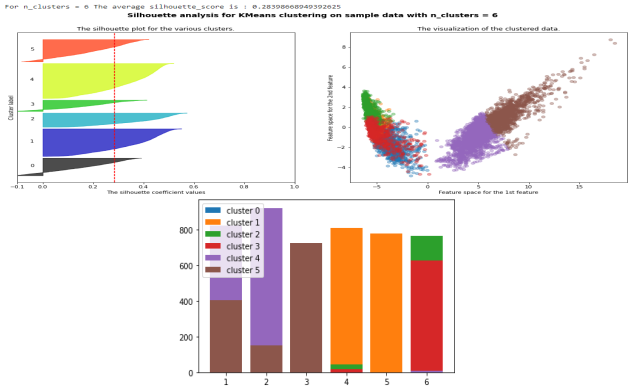


Fig. 94 Silhouette score XMeans and class histogram 6 cluster and PCA dataset

6.3 OPTICS

OPTICS (Ordering Points To Identify the Clustering Structure), closely related to DBSCAN, finds core sample of high density and expands clusters from them. Unlike DBSCAN, keeps cluster hierarchy for a variable neighborhood radius. Clusters are then extracted using a DBSCAN-like method or an automatic technique

like the "xi" method.
The reachability plot shows that the algorithm order the points in such a way that the only possible number of cluster for the DBSCAN algorithm would be 2 setting the eps at 2.

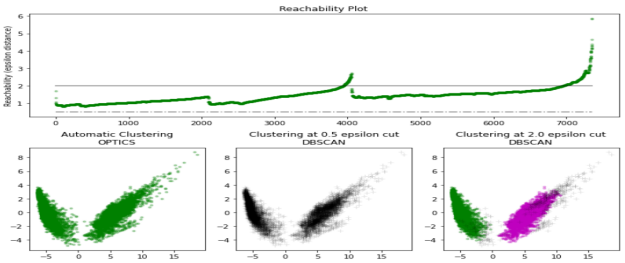


Fig. 95 Reachability plot (top) and clusterification with different eps values (bottom)

We tried to identify more cluster using the "xi" method but for multiple value checked in a range of (0.00001-0.9) the only result we got was 2 miniclusters inside the points, recognizing the majority point as noise.

7 Categorical Clustering

This kind of clustering uses categorical attributes hence our dataset is divided into buckets of equal frequency based on the quantile.
We prepared 2 dataset, the first with 10 buckets and the second with 100 buckets.

7.1 K-Mode

To investigate which one of the dataset would suits better the algorithm proposed we looked at the Silhouette and Cost plot for the two datasets.

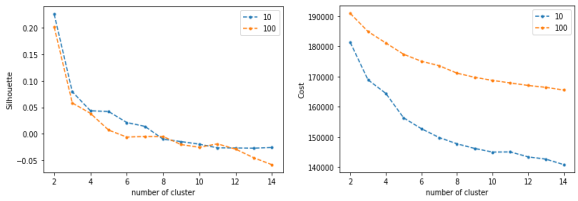


Fig. 96 Silhouette score and cost for K-mode

The Silhouette follows the same path for both 10 and 100 buckets dataset, meaning that the distance between the cluster created are basically the same hence they have the same cluster quality.
Meanwhile the cost for the 10 buckets dataset is lower. That implies that the points are closer to their own centroid.
Hence for the K-Mode we use the 10 buckets dataset.
Plotting the same plot for different centroid inizialization algorithm shows really similar behave, the Cao function seems to be a little more stable on the silhouette score and hence will be the one we use.

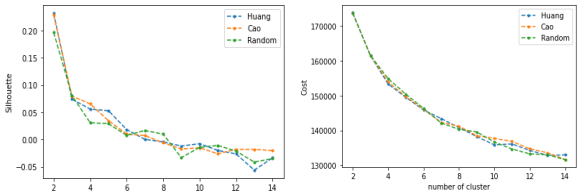


Fig. 96 Silhouette score and cost for K-mode with different centroid inizialization

From this graph we can see how the algorithm doesn't favourite any particular number of cluster since the function appear monotonic, hence we will see how it perform with the usual 6 clusters.
In the following a scatter plot of the bucket dataset to respect to the cluster and the respective initial space with the same cluster.
The histogram shows that the classes are mixed among the cluster and hence those are not good results.

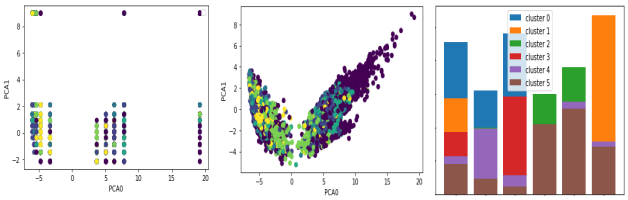


Fig. 97 Cluster scatter plot and class histogram Kmode

8 Explainability

For this section we want to use the LIME and SHAP explainability algorithms for a better feature and motivation understanding of the classification of the decision tree.
We've already checked its optimal depth and accuracy score and through LIME and SHAP we want to get a better undestanding.
The tree is shown at the end of the project.

8.1 LIME

This give an intuitive explanation of the decision of the DT showing which feature give a positive or negative impact on the choice.
We show 2 the explaination of 2 random instances.

Fig. 98 Lime Explainability for 2 instances

The first figure shows a prediction of 80% of probability of success due to the negative contribution of many feature.
While the second image predict class 6 with probability 1 even though there is a strong negative contribution from the 3rd feature. That is clear from the decision tree image, which the first split reach a 0 gini for the class 6 immediately since the only feature required is the

8.2 SHAP

As the lime algorithm it is use to explain the prediction done by the classifier enchanting the relevance of the important feature for the decision.
Infact this return an interactive table. We show the one which sort the data by similarity and shows in blue the positive and red negative contribution of a particular feature to predict such a label.

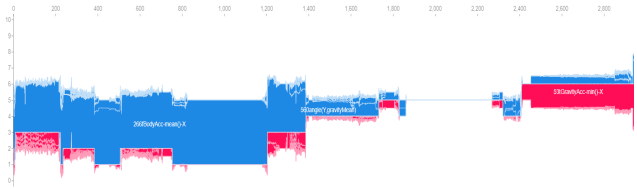


Fig. 99 SHAQ Explainability

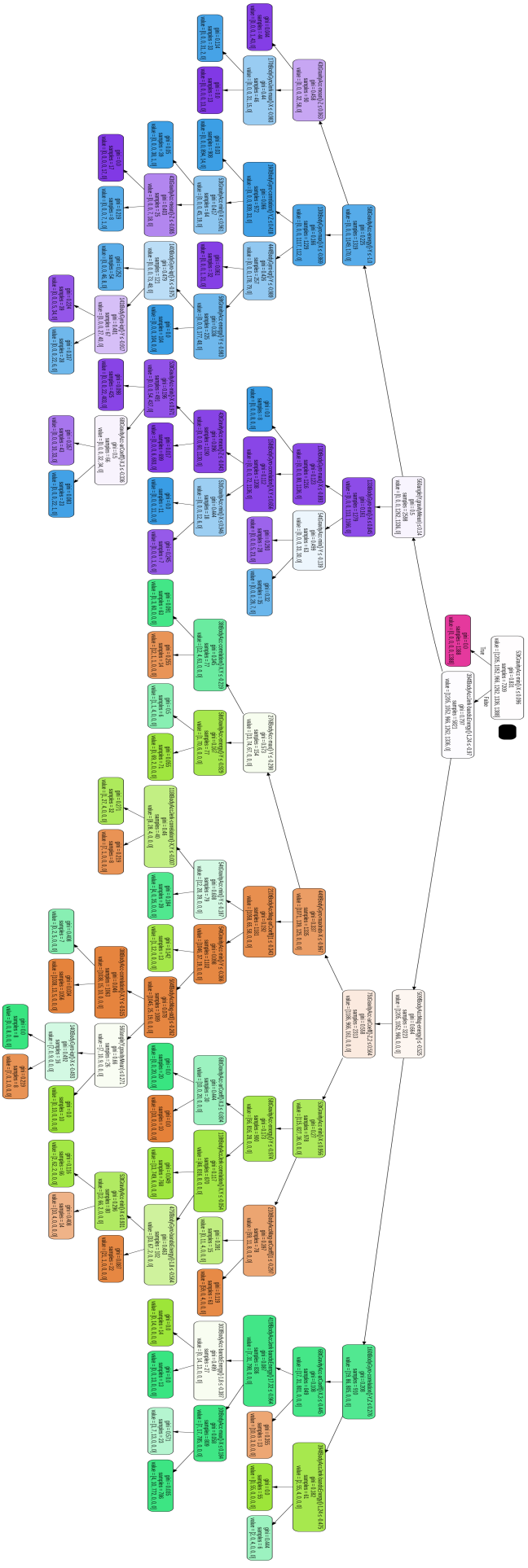


Fig. 100 DT