
UNIVERSITÀ DI PISA

# HUMAN ACTIVITY RECOGNITION

## PROJECT WORK - DATA MINING II

AUTHOR

*LUDOVICO LEMMA*

2021-2022

# Project Work - Data Mining II: Human Activity Recognition

Ludovico Lemma

June 6, 2022

## 1 Introduction

The original experiment through which this database was built regarded the recording of six different daily activities of 30 subjects while carrying a smartphone with inertial sensors (accelerometer and gyroscope) attached to the waist. The six activities recorded were: walking, walking upstairs, walking downstairs, sitting, standing and laying. The 30 individuals which were subjected to the experiment were in a range of 19 and 48 years old, they were divided in two groups, 70% were selected for the training set while the remaining 30% was selected for the test set. The data provided has both the format of a time series and its derived format of a 561-feature vector.

## 2 Data Understanding and Preparation

### 2.1 Data Description

The 561-feature vector has a total of 10.299 records, 7.352 are part of the training set and 2.947 are part of the test set. The values of the feature vector are continuous variables in the [-1,1] range. Of the thirty subjects those with the label 2, 4, 9, 10, 12, 13, 18, 20, 24 were selected for the test set, I will keep this original subdivision when it comes to the classification tasks even though with multiple classification techniques it appeared through cross validation that there may be better splits.

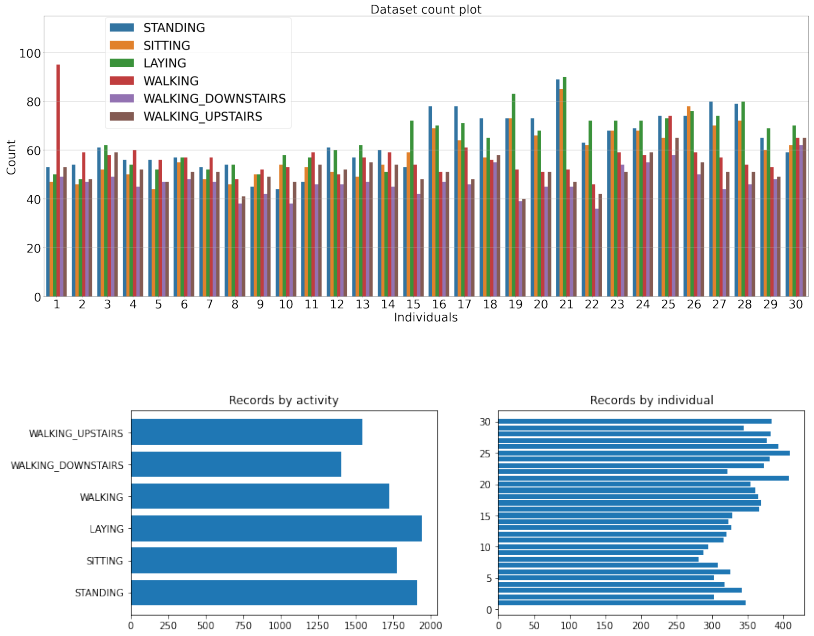


Figure 1: Countplots of the dataset

The 561-feature vector provided is sorted according to the individual who performed the actions and then those actions are kept near each other, because of possible problems with the validation sets

selected I decided to shuffle the dataset when dealing with classification.
Then I checked for missing values and duplicate columns and records, there weren't duplicates horizontally along the rows, there were 21 duplicates along the same columns but, considered the trivial number, there doesn't seem to be any major error in the data.
To select the features to focus on preliminarily during the exploration, I ran an instance of the decision tree algorithm on the unpreprocessed data for each activity label (Figure 7), then I focused the analysis on the top most important features of each activity label. With this methodology, considered also the high performances of the decision tree on the test data, with areas under the ROC curves higher than 0.8 for all the activity labels (the worst predictions regarded "walking upstairs" with 0.85 area under the curve), it was immediately clear that a subset of twelve features divide the data way better than the rest, in particular the 12 feature I selected had more than 0.10 importance for each activity label, in particular with tGravityAcc-min()-X it is possible to predict the laying activity with 100% performances for each metric. I checked these results with cross validation and KNN (Figure 7) and they were comparable. I then checked the distribution of 2 features with a scatterplot in Figure 2 according to a subdivision of the activities I considered should be similar in the measurements, which I will call stationary-state and moving-state.

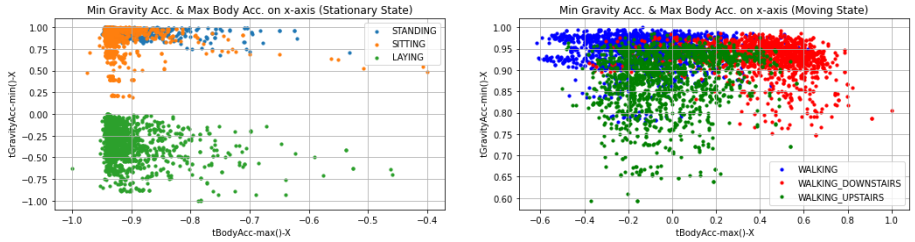


Figure 2: Scatterplot of two variables with activities in stationary and moving state

As it can be seen it is clear that measurements of laying are distant from the other activities, while as expected the values of the moving-state also have a stricter domain on the higher end of the distributions. I then tried to run K-Means on the data to understand how well it captured the original subdivision, thus I initialized 6 centroids. As in Figure 3, K-Means understood that the moving and stationary state are distanced but it didn't understand the separation internal to the groups.

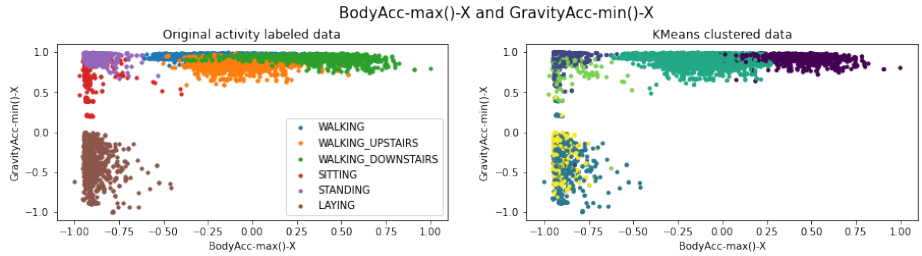


Figure 3: K-Means results on the whole dataset

## 2.2 Dimensionality Reduction / Feature Selection

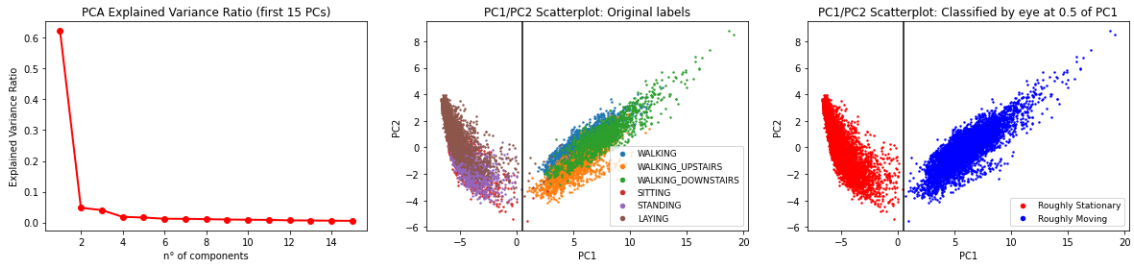


Figure 4: PCA results

I first applied the principal component analysis on the data, the results can be seen in Figure 4. The first component has a variance explained ratio over 0.6, the distance between stationary and moving state was clearly captured by the first component and I projected a line to evidence the distinction. It cannot be seen but I checked that a very limited number of variables (less than thirty) has a high loading on the first component (distinct peaks were observed beyond the 0.085 threshold). Then I tried Isomap (on the left Figure 5), I tried different parameters but considering the expensive computations I sticked with 15 nearest neighbours and 2 principal components. As it can be seen it captured the almost perfect explainability of the Laying class compared to the PCA and it understood somehow the difference between walking upstairs and downstairs, though not clearly distinguishing walking.

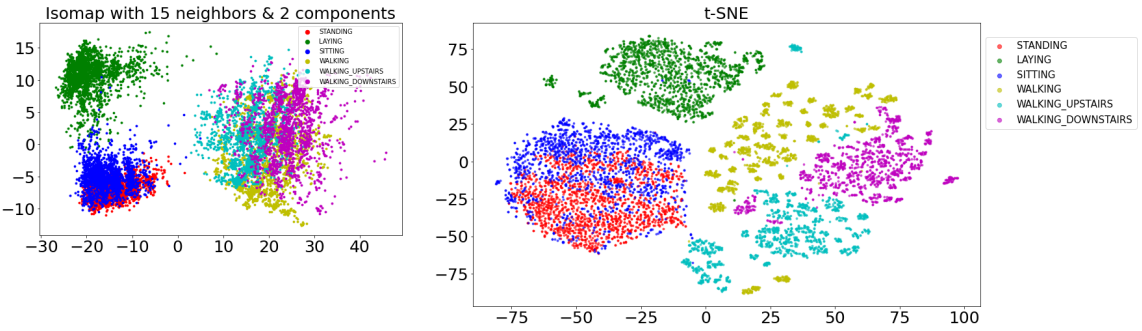


Figure 5: Isomap (left) and T-sne (right) results

Then I also tried t-sne (on the right in Figure 5), here the output of two components was very promising, it captured not only the distance between the various activities, somehow confusing only standing and sitting, but it also captured the differences between individuals walking pretty clearly. It wasn't possible to plot this information without overloading the image, but I verified also considering each subject separately and in the moving state t-sne considered most of the records of the individuals in a single group (without knowing their labels). I considered this may be useful in the anomaly detection task later.

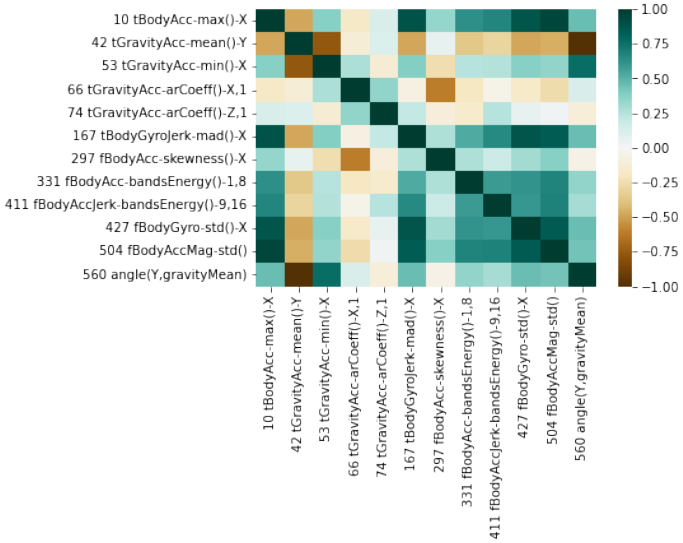


Figure 6: Correlation Heatmap of selected Features

These techniques were the most successful in providing further information, but having to decide to reduce the dataset dimensionality, I considered these approach too much for the type of dataset I have, thus I limited myself to remove all the features accordingly to the decision tree provided feature importances, then in order to understand the results of this preprocessing step I plotted a correlation heatmap in Figure 6, there does seem to be some high correlations in the output, but considered the high performances shown initially I guessed multicollinearity wouldn't be a great problem for the classification task.

## 2.3 Imbalanced Learning

Considering what I presented in Figure 1 the balancing of the activities' labels is the following: Standing: 18.51%, Sitting: 17.25%, Laying: 18.88%, Walking: 16.72%, Walking Downstairs: 13.65%, Walking Upstairs: 14.99%. Indeed, they are not imbalanced, in order to apply some techniques there is also another problem with the high performances as shown in Figure 7, as such I decided to focus on the most difficult class to predict: Walking Upstairs.

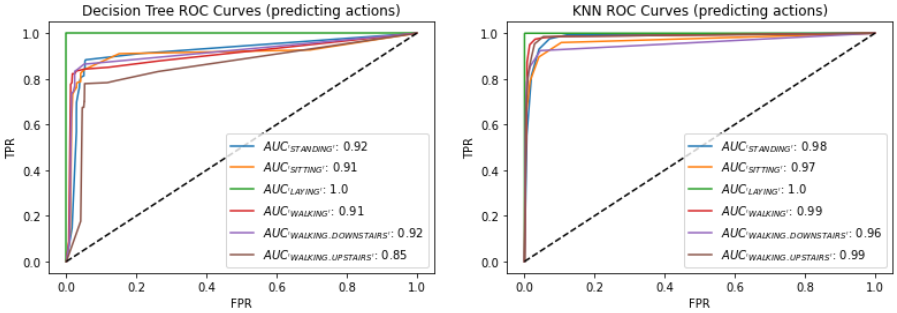


Figure 7: Decision Tree and KNN Roc curves

From this moment forward I used the training set to perform the task. Before anything else, I checked which is the number of record for this activity label where performances started to drastically fall, after some experiments I concluded that that number is between 30 and 50 records (from 1.073 records with this label in the training set), thus I randomly picked 32 as the number of records which represents 0.51% of the dataset (around 14.5% reduction, as a byproduct reducing the records in the training set from 7.352 to 6.279), making the class in question very imbalanced. Then I applied on the resampled data different imbalanced learning techniques, namely Undersampling (reducing the training data to 64 records), Oversampling and SMOTE (increasing in both cases the training data to 12.558 records( as it can be seen in Figure 8 (while applying the oversampling I slightly moved the points for visualization purposes).

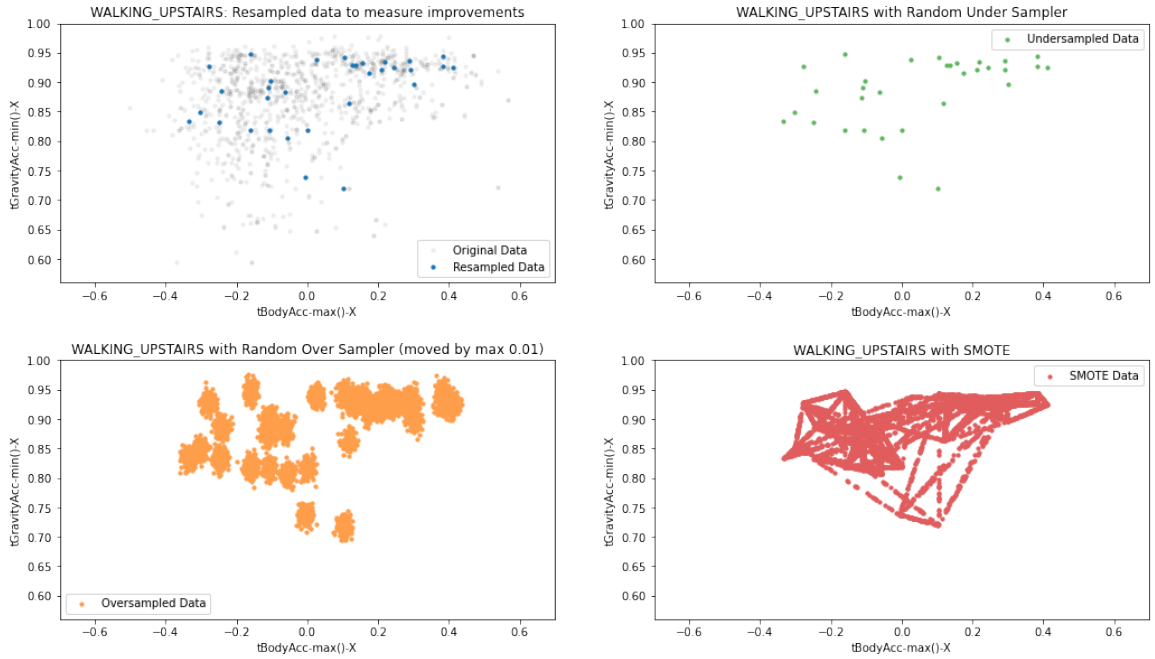


Figure 8: Imbalanced Learning applied techniques

After the preparation I then reapplied the classification algorithms with hyperparameter tuning for each of the new preprocessed training data (looking for the best score of the area under the ROC curve). KNN as before emerged as the best classifier on average while the best imbalanced learning technique was undersampling on both the decision tree and KNN. In Figure 9, on the left I plotted the outlook of the decision tree used with the undersampled data considered its simplicity, while on

the center the ROC curves of the decision tree and on the right those of KNN.
The most relevant parameters selected through the gridsearch for the decision trees where: the increased class weight of 5 for the records with the label "Walking Upstairs", the minimum sample to constitute a leaf equal to 10 for SMOTE and Undersampling, 1 for the imbalanced dataset and the oversampled one and finally the max depth of 2 and the entropy criterion for the imbalanced dataset while on the other three cases it preferred no depth limit and the gini criterion.
As it can be seen with even the imbalanced dataset (with the target class only representing only 0.51% of the dataset it's still possible to reach 80% of AUC), and for the undersampling case it's interesting to notice it needed only to set a threshold on a variable to predict the class in question. KNN showed similar good performances, SMOTE performed better than the imbalanced dataset in this case, the gridsearch was performed on the number of neighbours to consider, for the imbalanced and the undersampled data it preferred 18 neighbours, for SMOTE 19, in the case of oversampling only 1 neighbours. It's interesting to notice that oversampling was the worst technique in each shown case.

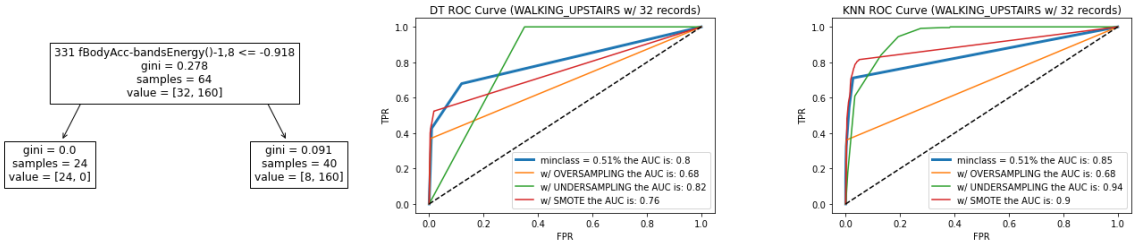


Figure 9: DT and KNN on the rebalanced data

## 2.4 Anomaly Detection

For this task I came back to the original training set reduced to the 12 features I decided to focus on and to a new instance of t-SNE only applied on the training set. Considering the label are known and the fact that through my exploration it didn't seem to me to be the case there were global outliers (at most collective outliers), for all anomaly detection techniques I decided to partition the dataset according to the activity labels and I analyzed the partitions separately looking for contextual outliers. As methodology of detecting the anomalies I decided to proceed as it follows. First I computed the anomalies according to the Boxplot, Silhouette Score, DBSCAN, Local Outlier Factor, Convex Hull and ABOD, everything by keeping the results of each dataset version separate but comparing the results of each method. Then I compared the results between all the methods, by a simple intersection I determined which where the outlier of the dataset. Finally I looked if there were correspondences between the application of all methods to the two dataset versions and I decided I should combine the results through a union of the 2 outliers sets and then if there were too many I would replace their values by the mean of the most appropriate distribution, if not that many (no more than 10) I would just delete the values I doubt more.

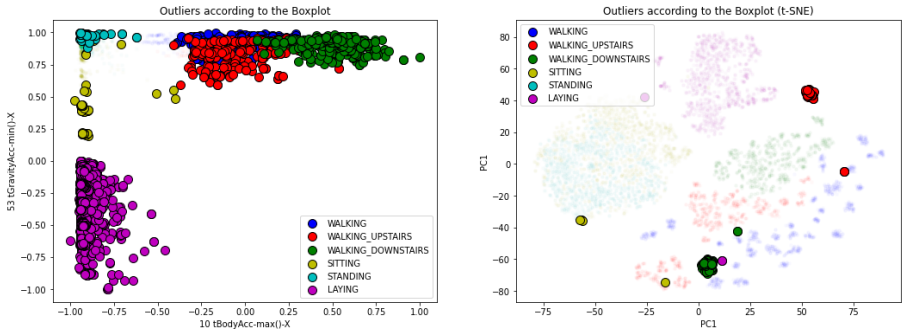


Figure 10: Anomalies according to the boxplot method

As it can be seen in Figure 10 I started with the boxplot, on the left we have the method applied to the 12 features selected, on the right on t-SNE. The results seems more promising on t-SNE even though some individual walking upstairs and downstairs pattern may have been misclassified

as I can see it captured also small groups of values that probably pertain to a subject, but the internal structure of 12 dimensions may be too difficult to understand by only projecting it on two dimensions. The method identified 2338 outliers on the 12-feature dataset and 124 outliers on t-SNE, 61 in common between the two.

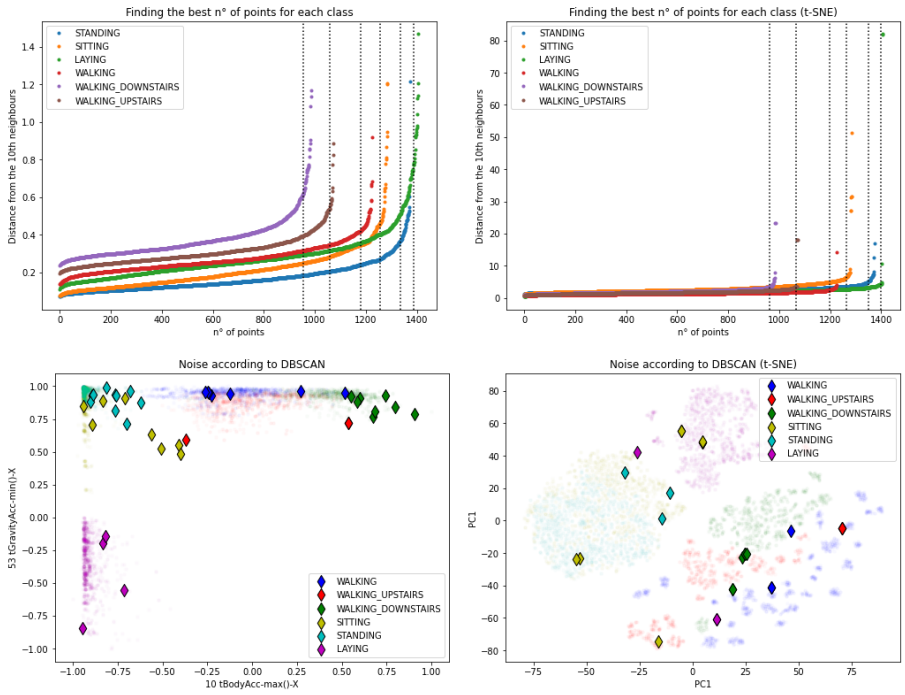


Figure 11: DBSCAN tuning and anomaly detection results

Then I moved to apply dbscan, in this case I also tuned the parameters, in particular I selected the best epsilon by using the maximum curvature point of the line of the distance from the 10th neighbour, by using that value I iterated across all classes and then I applied the algorithm according to those parameter to the respective class. The result can be seen in Figure 11, the number of anomalies detected was 41 for the 12-feature dataset and 26 on t-SNE, with 2 common anomalies detected in both datasets.

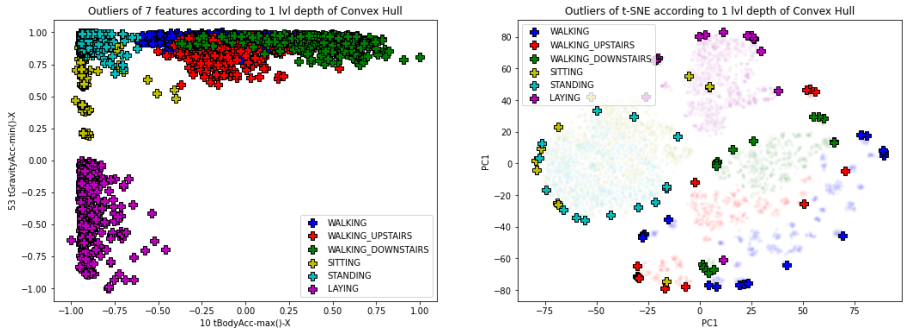


Figure 12: Convex hull vertices

With the convex hull (Figure 12), the efficiency of the algorithm became a problem, not only more features included more outliers as the number of vertices increased but 12 dimensions were too many to get results in a satisfactory short time, thus I decided to reduce the number of features until I got results in less than a minute, the computation was fast enough with 7 features, I selected the features by looking back at those with the highest importance. In the case of t-SNE, having only two components the issue didn't occur, the results of the algorithm was that I got 3128 vertices with 7 features, 87 with the 2 components of t-SNE, 57 appeared in both sets. I decided to focus only on the 1st level of depth (the most external vertices of the classes considered).

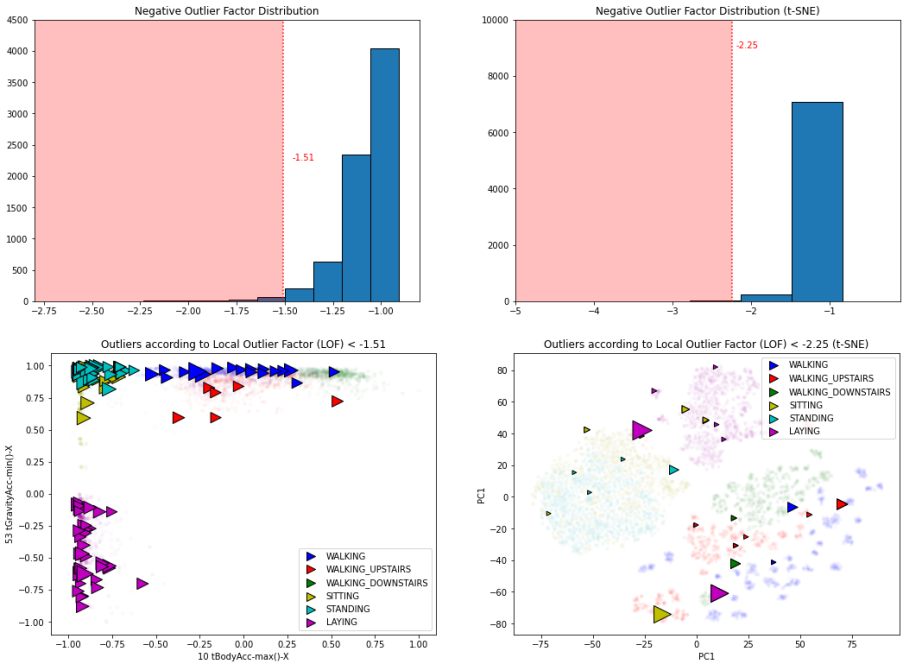


Figure 13: LOF scores distribution, threshold selection and results

Then I used the Local Outlier Factor (Figure 13), I got the score for both the datasets and by dividing the datasets by classes I selected the lowest possible threshold from where the records were sparse, in the case of the 12-feature dataset the negative outlier score of -1.51 was enough, while for t-SNE there where still lots of records which appeared to be well grouped together, thus I reduced the threshold to -2.25, for both the datasets I tried multiple number of neighbours and I decided to stick with 5 neighbours for both of them. I obtained 125 outliers with the 12-feature datasets and 38 on t-SNE, with 4 in common.

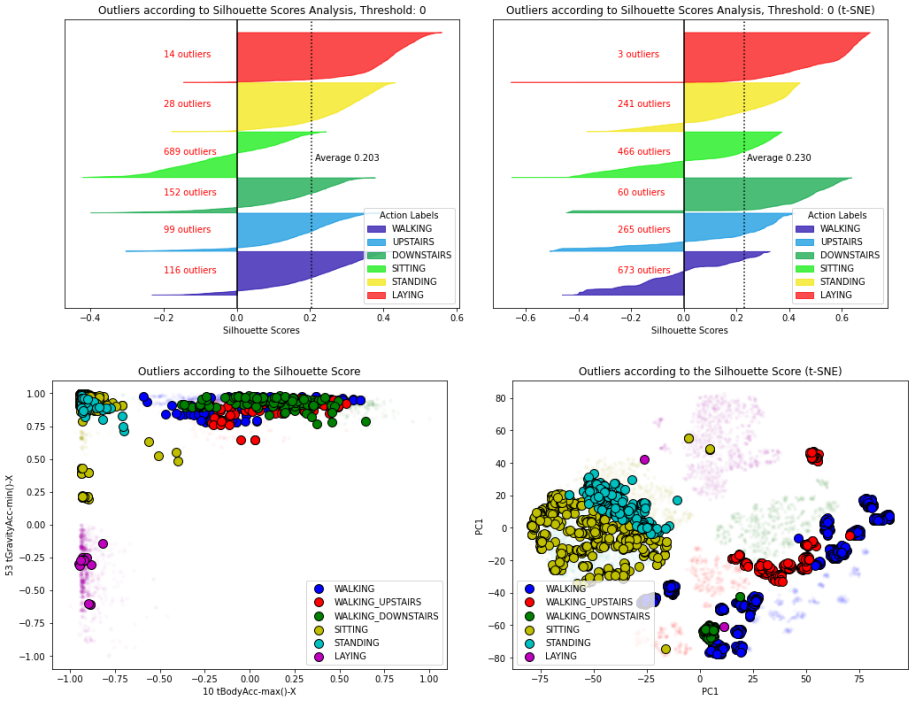


Figure 14: Silhoeutte Score Analysis results

Then I performed a silhouette score analysis and I considered the points with silhouette scores under 0 to be outliers, the results can be seen in the upper half of Figure 14, as it can be seen the only different proportion was with the walking category, which is more spread out, consistently to my previous understanding of it, but that makes this method less effective with t-SNE, complexively I found 1098 outliers in the 12-Feature dataset, 1708 with t-SNE of which 618 where in common.

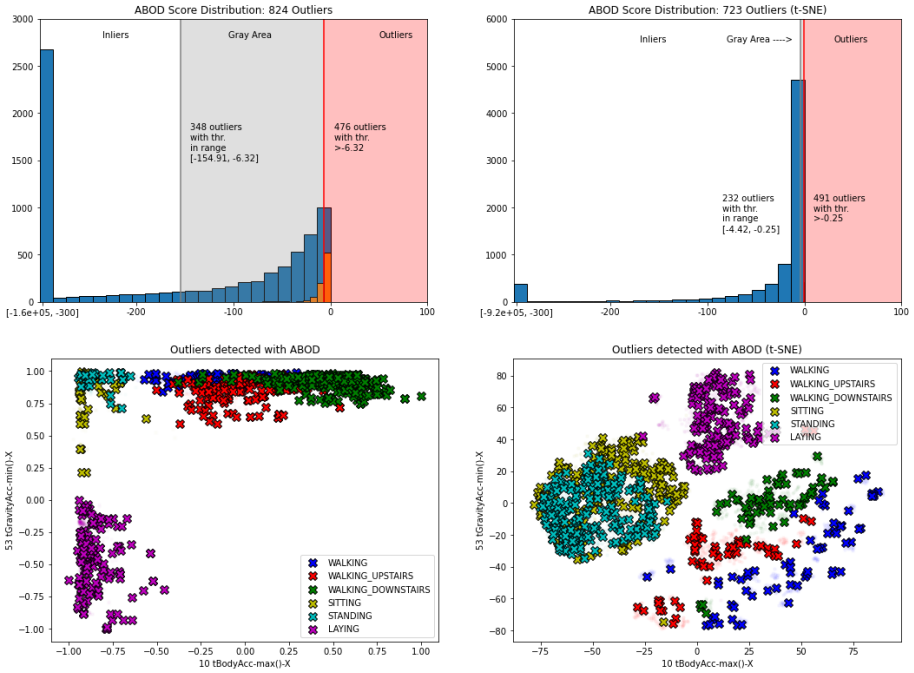


Figure 15: ABOD decision scores and Detected Anomalies

Finally I used the Angle Based Outlier Detection method Figure 15, in this case I analyzed the distribution of decision scores assigned by the algorithm, and which were the threshold the prediction became "outliers-only". It predicted 824 outliers in the 12-feature dataset and 723 on t-SNE, the common outliers with this method were 99.
In the end I intersected the results of all the methods, in the case of the 12-feature dataset 7 outliers were in common with each method, while on t-SNE only 2, which were both in common with those found with the 12-feature dataset. As I anticipated, considered the limited number, I opted to drop the 7 found outliers instead of adjusting their values. The new number of records in the dataset is 10.292.

# 3 Classification and Regression

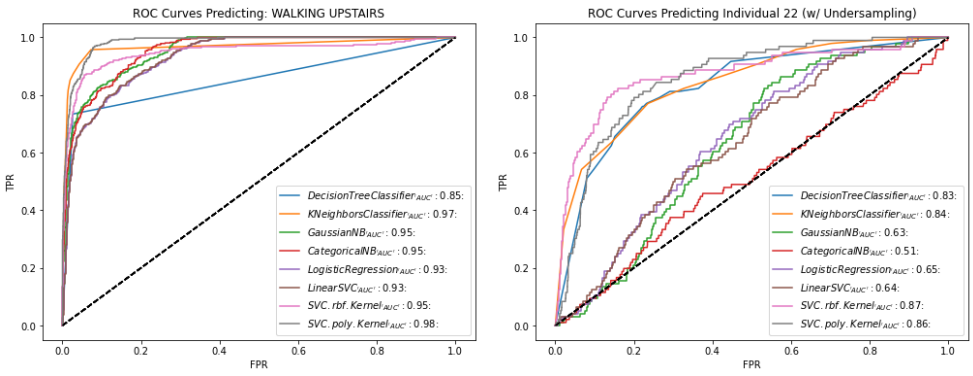


Figure 16: Comparison of the ROC curves of two different target variables

Regarding the classification task I started with the train-test split provided by the author of the experiment, the dataset didn't required either any rescaling for some algorithms as it's already scaled. I looked at the activity labels I considered most difficult to predict (as before the label Walking Upstairs), then I performed a gridsearch with most algorithms to select the best parameters for each of them, the score I optimized was the area under the ROC curve in order to get an idea of the best model regardless of the threshold, then I ran the alogrithms and computed the AUC, the F1 score, Recall, Accuracy and Precision, in order to get reliable results I used 5 fold cross validation to get a measure of the train-test splitting, as an output of cross validation I used the F1 score.
After I tried a substantial number of instances of different algorithms I realized it was very difficult to get bad performances, most of them were indeed comparably very high, with the decision tree getting the poorer results.
Thus as it can be seen on the right of Figure 16 I tried to predict a different variable, namely an individual who was subjected to the experiment. Considered the task I had to concatenate the training and test set and make a new split, the proportion were 0.7 for the training and 0.3 for the test set. I started looking iteratively for the most difficult subject to predict with the decision tree (looking at the worst possible ROC curve), as a result I got that the 22 individual was the less differentiated but even without using any parameter optimization and considering that I didn't included in the computation the different activity labels to help distinguish the task performed, the ROC curve had the pretty high AUC score of 0.7. I considered that to be low enough though to understand better the differences in performances with the different algorithms.
Another preprocessing step needed with the new training set was to apply an imbalanced learning technique, as the number of records pertaining to subject 22 in the whole dataset are 321, 225 in the new training set after the split, which is less than 4% of the number of records. After some experiments I concluded undersampling is the best performing one. Thus I reduced the training set organized for this target variable to 450 records, where half of this number are records pertaining to subject 22.

**PREDICTING WALKING UPSTAIRS**

| | Decision Tree | KNN | GaussianNB | CategoricalNB | Logistic Regression | LinearSVC | SVC (RBF Kernel) | SVC (Poly Kernel) |
|---|---|---|---|---|---|---|---|---|
| ROC AUC | 0.85 | 0.97 | 0.95 | 0.95 | 0.93 | 0.93 | 0.95 | 0.98 |
| Accuracy | 0.93 | 0.95 | 0.87 | 0.84 | 0.91 | 0.91 | 0.93 | 0.95 |
| F1-score (target) | 0.78 | 0.85 | 0.68 | 0.65 | 0.66 | 0.65 | 0.77 | 0.84 |
| Mean F1-score (cv) | 0.88 +/- 0.015 | 0.94 +/- 0.021 | 0.65 +/- 0.013 | 0.63 +/- 0.005 | 0.79 +/- 0.063 | 0.79 +/- 0.068 | 0.95 +/- 0.021 | 0.94 +/- 0.02 |
| Precision (target) | 0.85 | 0.92 | 0.57 | 0.51 | 0.82 | 0.82 | 0.88 | 0.86 |
| Recall (target) | 0.73 | 0.78 | 0.85 | 0.89 | 0.55 | 0.54 | 0.68 | 0.82 |

**PREDICTING SUBJECT 22**

| | Decision Tree | KNN | GaussianNB | CategoricalNB | Logistic Regression | LinearSVC | SVC (RBF Kernel) | SVC (Poly Kernel) |
|---|---|---|---|---|---|---|---|---|
| ROC AUC | 0.83 | 0.84 | 0.63 | 0.49 | 0.65 | 0.64 | 0.87 | 0.86 |
| Accuracy | 0.78 | 0.67 | 0.52 | 0.55 | 0.62 | 0.61 | 0.84 | 0.71 |
| F1-score (target) | 0.17 | 0.14 | 0.08 | 0.06 | 0.09 | 0.08 | 0.24 | 0.16 |
| Mean F1-score (cv) | 0.49 +/- 0.041 | 0.55 +/- 0.043 | 0.005 +/- 0.01 | 0.000 +/- 0.01 | 0.000 +/- 0.01 | 0.000 +/- 0.01 | 0.66 +/- 0.029 | 0.31 +/- 0.079 |
| Precision (target) | 0.10 | 0.07 | 0.04 | 0.03 | 0.05 | 0.04 | 0.14 | 0.09 |
| Recall (target) | 0.76 | 0.82 | 0.71 | 0.44 | 0.60 | 0.56 | 0.82 | 0.84 |

Table 1: Comparison of Performances Metrics

Generally, also by looking at the metrics in Table 1 the best and less computationally demanding algorithm was KNN with both target variables, the performances of the Support Vector Machines are better with some kernels, in particular the polynomial kernel, but it's expensive to tinker with the parameters to find the best ones, while also continue to converge. The worst algorithm instead was clearly the Categorical Naive Bayes classifier in particular with predictions of subjects, in both cases I used the quartiles to determine the categories, but the differences become too subtle with too large categories, the number of records is not a problem because I also tried to oversample the new dataset for this task and I obtained slightly better results with every the algorithms excluded the decision tree, knn and indeed the categorical NB classifier (in this case the highest AUC was that of the rbf kernel of SVM, with 0.90 AUC but the computational time was unreasonable and I preferred to not study further the results oversampling which were more or less comparable) It's also clear there may be better splits in some cases considering the F1 scores of the cross validation, it doesn't seem to be the case the dataset was split to facilitate predictions.
Finally the minimum number of records which guarantee acceptable level of performances is higher than 20 for every classifier while in a binary classification context predicting an activity label, in the case of the predictions of subjects the number of records shouldn't be decreased much more or the performances start to drastically fall with each target class having already few records and being slightly more difficult to predict the different patterns related to a subject, in this second case for many classifier the number is in fact insufficient (especially with the naive bayes classifier, linear support vector classifier and the logistic regression)

## 3.1 Support Vector Machine

The Support Vector Machine classifier gave particularly good results between the classifiers I experimented with until now. For the the classifier to converge it was crucial in this case to have a weak regularization as the parameter C which is inversely proportional needed to be higher than 1 and smaller than 100, in order to visualize and understand how the hyperplane and the support vectors moved with different parameters I reran an instance of this algorithm with the linear default kernel while changing the regularization parameter to see how the convergence compared to not converging results. In Figure 17 to enhance the visualization I only plotted 30 random support vectors, there are many more though, the data points plotted is the whole dataset in this case.

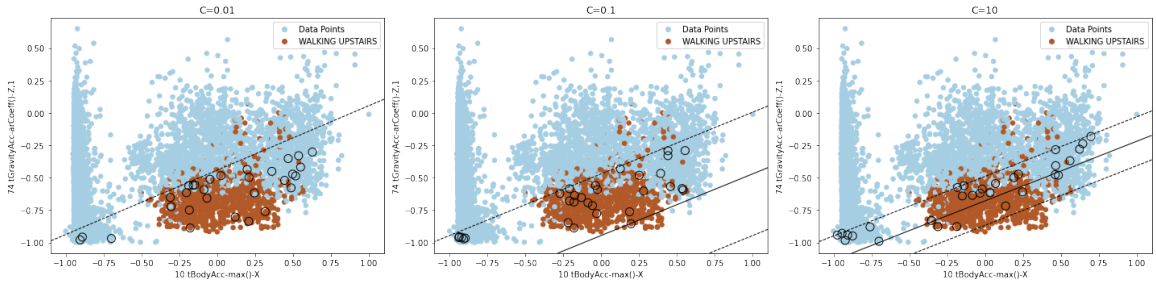


Figure 17: Comparison of effects on convergence with different regularization parameters

In order to understand how different kernel divided the space on two dimensions I also analyzed the hyperplanes of the linear, radial basis function and polynomial kernels, this time only on the test data in order to visualize the prediction's decision surface. It was immediately visible how the second one worked particularly well with the radial shape on the target data.

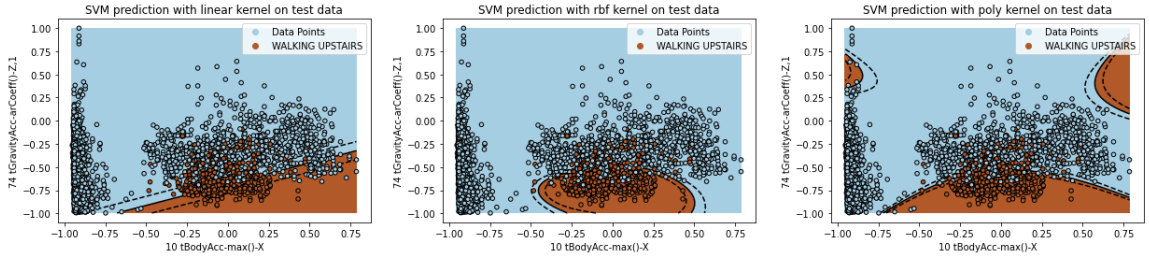


Figure 18: Comparison of different kernels on the test data

## 3.2 Logistic and Linear Regression

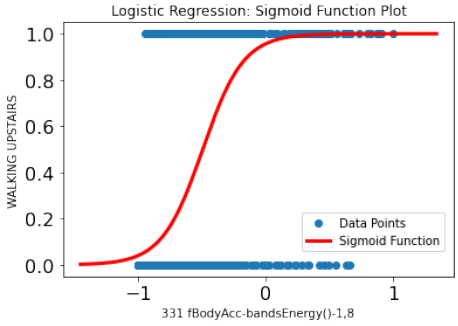


Figure 19: Logistic Regression Sigmoid Function

Regarding logistic regression as it can be seen from the loss function in Figure 19, the sigmoid is almost completely comprised in the distribution of the most predictive variable of the target class (as discovered during the imbalanced learning decision tree in Figure 9) with other variables the sigmoid functions become way less reliable to predict the target class as the curvature is not completely inside

the limits of the distribution.

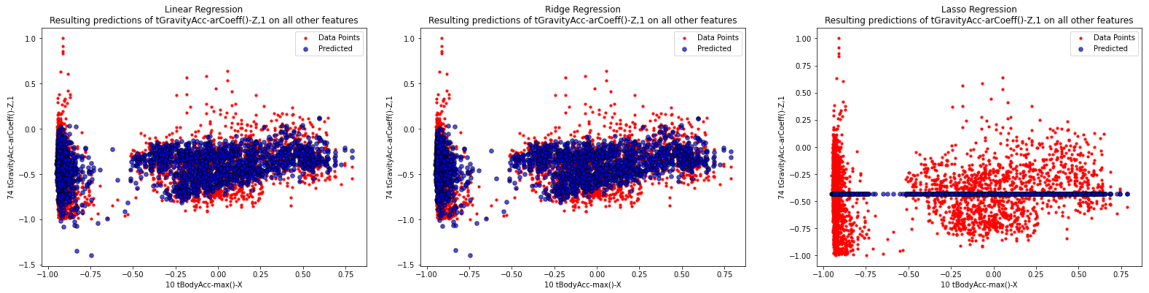


Figure 20: Multivariate problem's Regression Results

Then I also tried to define linear univariate and multivariate regression problems with a continuous target variable. In the case of a multivariate problem as it can be seen in Figure 20, the regression is pretty accurate to capture the variable tGravityAcc-arCoeff()-Z,1 both with the standard linear regression and with the ridge regression, in fact the R squared is close to 0.4 for both of them, while the mean squared error is very low at 0.04 and the mean absolute error at 0.160, the metrics coincide for both of them, thus the indipendent variables explain much of the variance of the dependent variable, even though the dependent variable I selected is not that correlated with the other 11. The Lasso regression instead converged with 0 coefficient on every features, while MSE and MAE are comparable the R squared became -0.02 determining this as the worst regression for this multivariate problem, indeed it simply solved it as a univariate problem as the metrics and the results are identical to the next Lasso application.

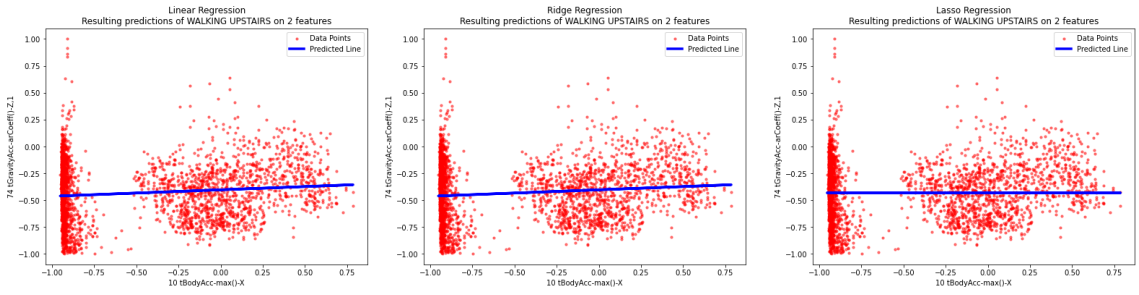


Figure 21: Univariate problem's Regression Results

Regarding the univariate problem instead the results where practically identical across all the three types of regression in question. Again though it seemed that Linear and Ridge regression converged on the same metrics with: R squared equal to 0.01, MSE equal to 0.07 and MAE 0.211 (the last two identical to the Lasso multivariate regression results). Lasso Regression, as said instead, got the identical results as before with an R squared equal to -0.02 and the same MSE and MAE of the other two methods. This may indicate though that the variable selected doesn't explain that much the variance of the dependent variable, indeed the coefficients linear and ridge regression pointed towards a stronger correlation with other variables.

## 3.3 Neural Network

For this classifier I kept the two tasks as before and I tested different activation functions, considering the high complexity of an extensive gridsearch I limited myself to a manual check of few couples of parameters for each activation functions, in particular I tried with constant and adaptive learning rate and with different sizes of the neuron sets of hidden layers, I then choosed to use the 200 neurons for the first hidden layer and 100 for a second hidden layer, I set the learning rate constat to increase my ability to understand the results. I didn't perform any imbalanced learning technique as they reduced the performances for subject 22. As it can be seen in both Figure 22 and 23 the performance were proved to be very high. It can be seen the plot of the Loss Curve in a particular way, I normalized not across the Loss values but across the epochs as I wasn't interested in the loss changes by themeselves but relative to their epochs to keep the shape comparable with the others, I kept the number of epochs in the legend to understand better how the curve was long originally.

Looking at the results it can be seen that the logistic activation function performed almost perfectly with the "walking upstairs" prediction, but it was almost insufficient with the prediction of subject 22, but in the first case the price was that the number of iterations for convergence was 427, which makes it pretty costly, compared to other solutions. In particular the ReLU and Tanh activation functions proved themeselves to be the best, especially the first one, which provided very good results without going to far to converge. The identity activation function instead iterate just a few dozens of times but the results weren't that great with the prediction of subject 22 (with the first prediction it as already easy with less sophisticated classifiers. I presented the metrics in Table 2.

| | **WALKING UPSTAIRS** | | | | **SUBJECT 22** | | | |
|---|---|---|---|---|---|---|---|---|
| | Identity | Logistic | Tanh | ReLU | Identity | Logistic | Tanh | ReLU |
| ROC AUC | 0.93 | 0.98 | 0.97 | 0.98 | 0.60 | 0.59 | 0.89 | 0.92 |
| Accuracy | 0.90 | 0.95 | 0.94 | 0.95 | 0.97 | 0.97 | 0.98 | 0.98 |
| F1-score (target) | 0.67 | 0.84 | 0.80 | 0.82 | 0.0 | 0.0 | 0.36 | 0.45 |
| Precision (target) | 0.82 | 0.85 | 0.83 | 0.85 | 0.0 | 0.0 | 0.85 | 0.81 |
| Recall (target) | 0.56 | 0.83 | 0.79 | 0.79 | 0.0 | 0.0 | 0.23 | 0.31 |

Table 2: Artificial Neural Network Performance Metrics by Activation Function

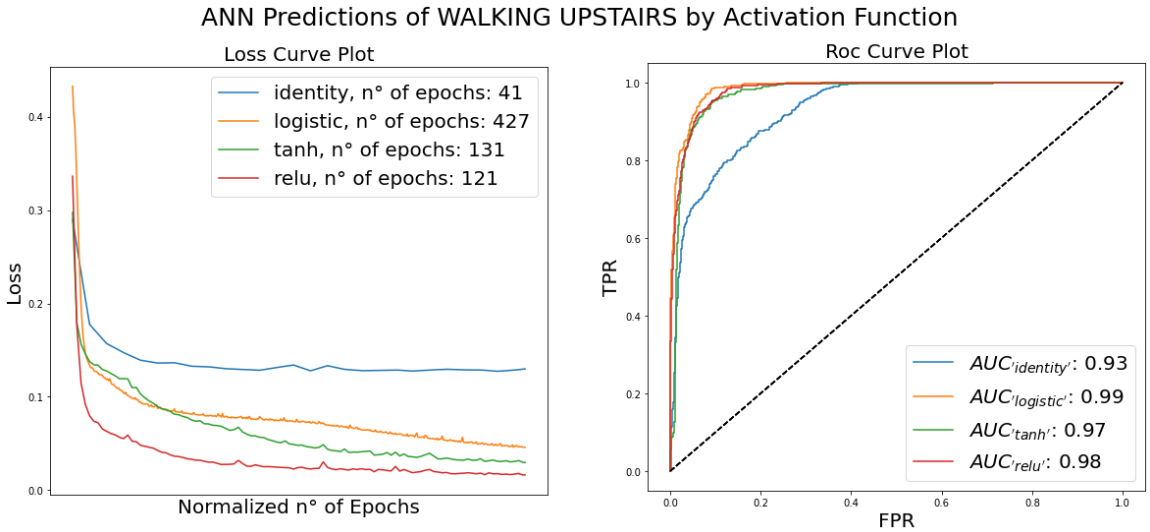


Figure 22: Artificial Neural Network Performances (Walking Upstairs)

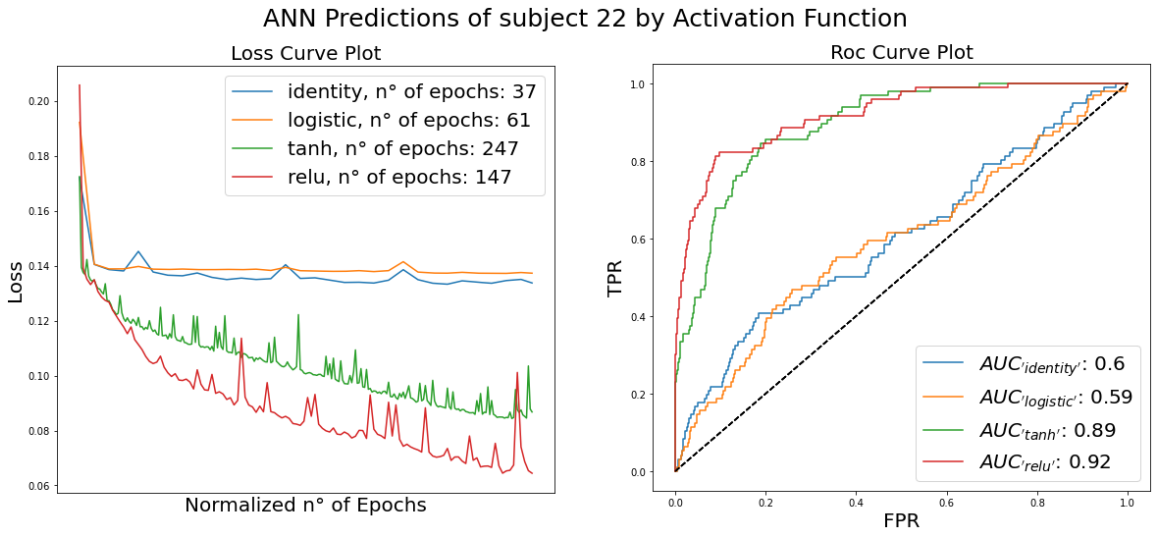


Figure 23: Artificial Neural Network Performances (Subject 22)

## 3.4 Ensemble Methods: Bagging, Random Forest and Adaptive Boosting

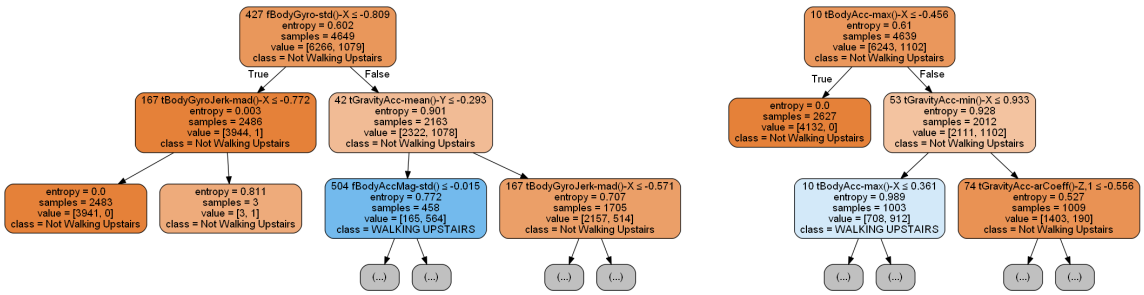


Figure 24: Two DT estimators out 50 of the Random Forest Classifier (Walking Upstairs)

As Ensemble methods I used two classifier based on Bagging and one on Adaptive Boosting. The base estimators I used was the decision tree in the case of AdaBoost and in one case of Bagging (Random Forest) being the classifier with which I got always lower results when trying to predict "Walking Upstairs", in the remaining case of Bagging I preferred as a comparison a Bagging of an unstable but fast classifier, as such I avoided KNN which is too stable and SVM as it is too slow. I opted to try with the artificial neural network with the "identity" activation function to see if sampling features and voting affects the performances (which were the poorest among neural networks), I reduced the hidden layer size only to one layer of 100 neurons to speed up the process, a quick check demonstrated that the performance metrics didn't diverge much.
In the case of the random forest I also performed a randomized search to look for the best combination of parameters. Thus, the hyperparameter selected were 50 number of estimators for Bagging (too slow with 100 estimators) and the RandomForest, for the latter the randomized search also selected the entropy criterion to split, a max depth of trees equal to 11, the minimum number of samples to perform a split equal to 5, it didn't set any minimum weight to be a leaf.

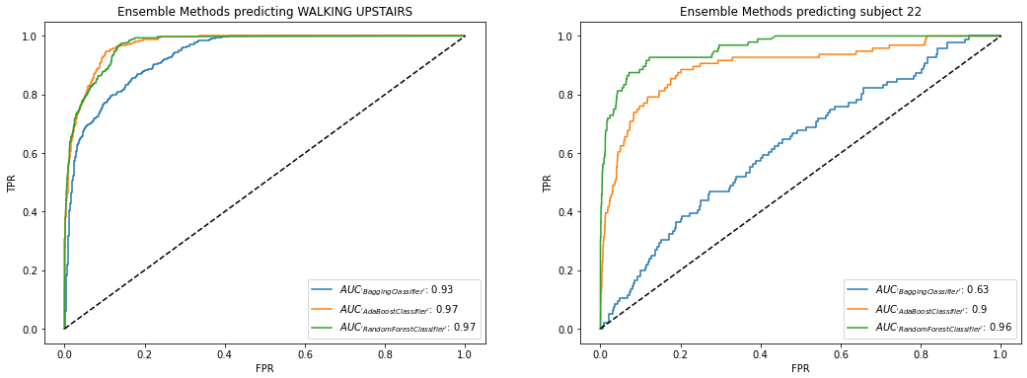


Figure 25: Ensemble Methods Results

As a result I got that bagging the neural network with the identity activation function didn't provide any change in the performance metrics showed before, this means that activation function is too stable as a classifier to reduce its variance through sampling and voting in bagging. Regarding Adaptive Boosting and the Random Forest I got way higher results, providing that it is possible to optimize a classification with multiple random decision trees and decision stumps, the results were comparable, 0.97 for the AUC score, 0.93 for Accuracy, 0.77 for the F1 score of the predicted class, 0.81 precision and 0.72 recall while having in both cases a 0.99 AUC score with cross validation, meaning that there are better splits as evidenced also previously with other classifiers.
The feature importances in this case were interesting, in Figure 9 it was shown that with very imbalanced records, the preferred feature to split the space was the Body Acceleration, here though as shown in Figure 26 that feature was still important but was overtaken by 4 other features, meaning that at least 5 features determined these high differential performances, it may be that the decision surface is way more separable in a five dimensional plane than in a lower dimensional case as it appeared before. By looking also at the permutation though we can see that Body Acceleration is considered more important and with less variance, that means it's indeed specifically a good predictor of Walking Upstairs even though there should be at least 2-3 more features to have a good

prediction.

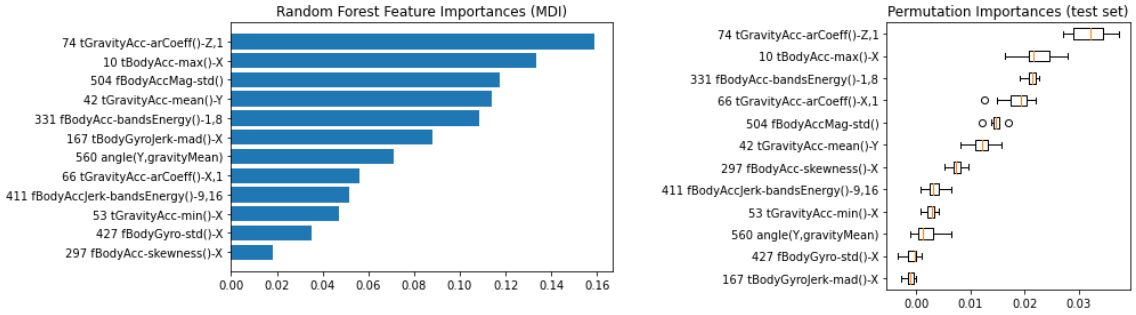


Figure 26: Random Forest Feature Importance of Walking Upstairs class

Another fact I noticed is that by changing the number of estimators to consider in the models the performances changes drastically only if there are less than 10 of them, and the performance until they are more than 3 are reasonably good, with only Adaptive Boosting's performances falling faster.
For predictions about subject 22, I didn't use any undersampling again, because of a lower performance outlook through a quick check. With the Bagging set as before I got a slight but still comparable improvement compared to the previous instance of a single neural network with the identity activation function (only the AUC increased by 0.03, the rest of the metrics without changing thresholds stays at zero with only accuracy high as before because of the imbalance)
Regarding Adaptive Boosting I got slightly poorer results than the Random Forest Classifier, but mainly on AUC score and Precision, it is in fact still comparable with accuracy around 0.97, F1 score around 0.3 and Recall around 0.2, with the Random Forest the precision got the surprising value of 1, meaning that the model fits without false positives. I won't show further in this report the feature importance in this second case, neither I will present an outlook of the trees, as no relevant further information could be extracted.

## 3.5 Ensamble Methods: Gradient Boosting

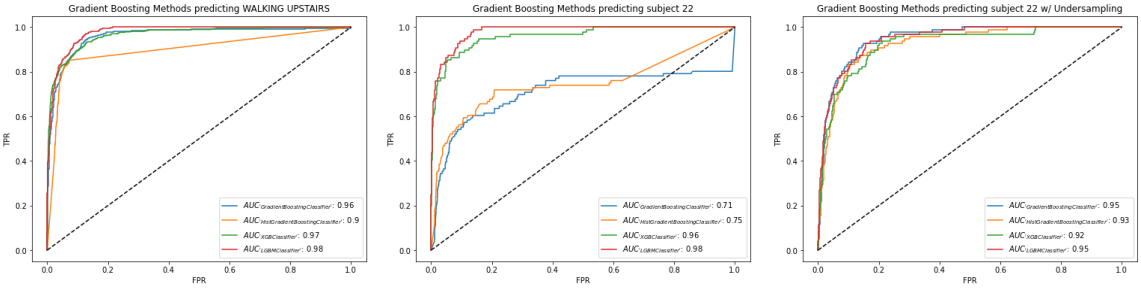


Figure 27: Gradient Boosting ROC curves

With Gradient Boosting I obtained some of the best results of the entire range of classifiers, only comparable to neural networks with the ReLU activation function and the Random Forest, but sometimes as in the case of predicting a very imbalanced class even better results especially with the Light Gradient Boosting Machine and the eXtreme Gradient Boosting classifier (where results reach the limits of 100% AUC score with cross validation), they also proved themeselves to be very fast compared to the aforementioned previous classifiers.

**WALKING UPSTAIRS WITH GRADIENT BOOSTING)**

| | Gradient Boosting | Histogram Based GB | eXtreme GB | Light GB Machine |
|---|---|---|---|---|
| ROC AUC | 0.96 | 0.89 | 0.97 | 0.98 |
| ROC AUC (cv) | 0.979 +/- 0.015 | 0.975 +/- 0.008 | 0.997 +/- 0.003 | 0.998 +/- 0.002 |
| Accuracy | 0.93 | 0.92 | 0.94 | 0.94 |
| F1-score (target) | 0.77 | 0.72 | 0.80 | 0.80 |
| Precision (target) | 0.81 | 0.77 | 0.85 | 0.85 |
| Recall (target) | 0.73 | 0.67 | 0.76 | 0.75 |

Table 3: Gradient Boosting Performances with predicting Walking Upstairs

The hyperparameter were set with a randomized search, in particular for the gradient boosting

classifier it set the Friedman MSE criterion, 100 number of estimators, a learning rate of 1 and the depth of the trees at 3, the same configuration was set for the Histogram-Based Gradient Boosting, while for eXtreme GB I manually set the objective as binary with the logistic function then the randomize search found the learning rate of 1 and trees' depth of 6. Finally the for the LGBM classifier the boosting type I set was "gradient boosting with decision tree", the binary objective and I removed the L1 and L2 weight regularizations, the randomize search removed limits to the maximum depth and it set to 100 the estimators, with 31 number of leaves. As it can be seen from Table 3 and 4 the metrics results are almost consistently higher than those of other classifiers, it can be noticed in particular that the Gradient Boosting classifier and the Histogram based one are more subject to imbalanced data as the prediction of the first fall under the baseline with some lower thresholds. But the most interesting results can be obtained with predictions of imbalanced data with the eXtreme Gradient Boosting classifier and the Light Gradient Boosting Machine, with F1 Scorse around 60%, the highest I got until now with the "very imbalanced" target class of subject 22 to predict.

| | SUBJECT 22 WITH GRADIENT BOOSTING | | | | SUBJECT 22 WITH GRADIENT BOOSTING (w/ UNDERSAMPLING) | | | |
|---|---|---|---|---|---|---|---|---|
| | Gradient Boosting | Histogram Based GB | eXtreme GB | Light GB Machine | Gradient Boosting | Histogram Based GB | eXtreme GB | Light GB Machine |
| ROC AUC | 0.71 | 0.65 | 0.96 | 0.98 | 0.95 | 0.93 | 0.92 | 0.95 |
| ROC AUC (cv) | 0.781 +/- 0.07 | 0.677 +/- 0.058 | 0.973 +/- 0.017 | 0.983 +/- 0.006 | 0.781 +/- 0.070 | 0.677 +/- 0.058 | 0.973 +/- 0.017 | 0.983 +/- 0.006 |
| Accuracy | 0.95 | 0.96 | 0.98 | 0.98 | 0.88 | 0.86 | 0.88 | 0.89 |
| F1-score (target) | 0.28 | 0.24 | 0.60 | 0.64 | 0.31 | 0.28 | 0.28 | 0.32 |
| Precision (target) | 0.28 | 0.31 | 0.78 | 0.84 | 0.19 | 0.17 | 0.17 | 0.19 |
| Recall (target) | 0.29 | 0.20 | 0.49 | 0.51 | 0.85 | 0.86 | 0.79 | 0.85 |

Table 4: Gradient Boosting Performances with predicting Subject 22

Finally, I applied also the Gradient Regressor to the univariate regression problem I presented in the paragraph about Logistic and Linear Regression, in this case the Gradient Regressor converged to a fitted curve which clearly follow the shape of the data, as it can be seen by looking at the predicted points, this result can be observed also by looking at the R squared metric which is 0.043, low but 4 times higher than the equivalent of the previous linear and ridge regressions, by also keeping stable the MSE at 0.067 and the MAE at 0.205 (similar to the previous ones).

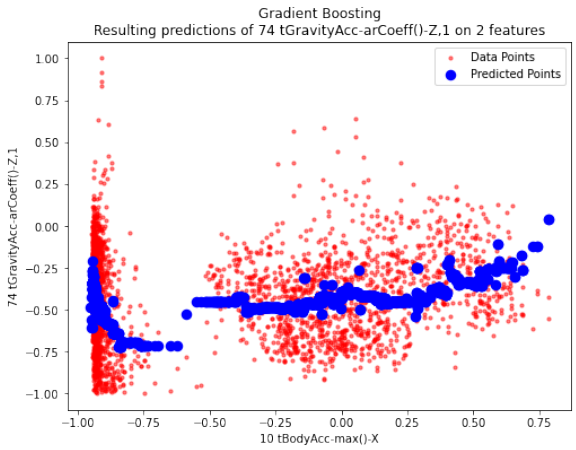


Figure 28: Gradient Regressor predicted points

# 4 Time Series Analysis

Here I moved to analyze the original signals from which the 561-feature vector was extracted. I decided to start the analysis by focusing on the object of most of the previous analysis of those features, namely the target variable "Walking Upstairs". From the analysis performed it was clear that the feature that explained better that action was related to the acceleration on the Z-axis, as for example the feature "tGravityAcc-arCoeff()-Z", which is the autoregression coefficient built with the Burg AR Estimator with a Burg order of 4 from the signal of the total acceleration on the Z-axis, which I identified through a first run of the decision tree importance. Thus I decided to study the time series of that signal.
Thus from the signal files referring to the total acceleration of the Z-axis I got the 10.299 time series (from both training and test set), then in order to work with the time series I decided to randomly sample 350 time series from the training set to reduce the number of records to do computations on,

I made it so that I had at least a time series with the subject 22 walking upstairs to present here in continuity with the previous plots.

## 4.1 Clustering

First I tried clustering the time series with KMeans, initially I approximated the data but then I discarded the idea as I obtained more interesting results with the raw time series and the computation with 350 records were faster anyway. I used both Euclidean and Dynamic Time Warping distances, but the results were slightly more precise with the latter and as such I will present them as in Figure 29. I primarily used KMeans in order to understand if there were some resulting centroids which represented well the mean of the known labels, I considered it useful being an unsupervised method in order to anticipate possible results with a classification method.

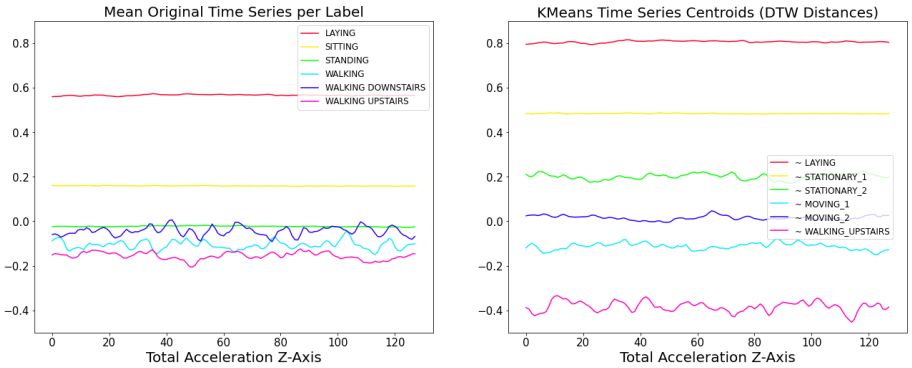


Figure 29: Clustering comparison of centroids to the known label means

As it can be seen the original mean were less spread out, and by comparing the individual clusters to the original labeled data we get that 2 clusters capture better the extreme values "Laying" and "Walking Upstairs" took, while 2 pairs of clusters represent approximately the other "Stationary" and "Moving" state even though one moving state cluster captured more "Standing" time series than it should. The third lowest error rate after is a cluster I called "Stationary_1" which it captured "Sitting" and the rest of "Laying" time series not included in the most representative cluster of that class.

## 4.2 Motif and Anomaly Discovery

As a second step of the time series analysis I wanted to look for motifs and anomalies in the time series. In order to understand better the total acceleration on the Z-axis I decided starting from now to focus more on presenting the results regarding the aforementioned target variable Walking Upstairs of subject 22.
Before finding the motifs and the anomalies I had to compute the matrix profile of the time series as I presented in Figure 30, I selected as a window 5, there as a comparison I also plotted the same time series transformed, the transformation I opted for was the re-scaling of the amplitude and the smoothing of the noise (on a 5 window mean).

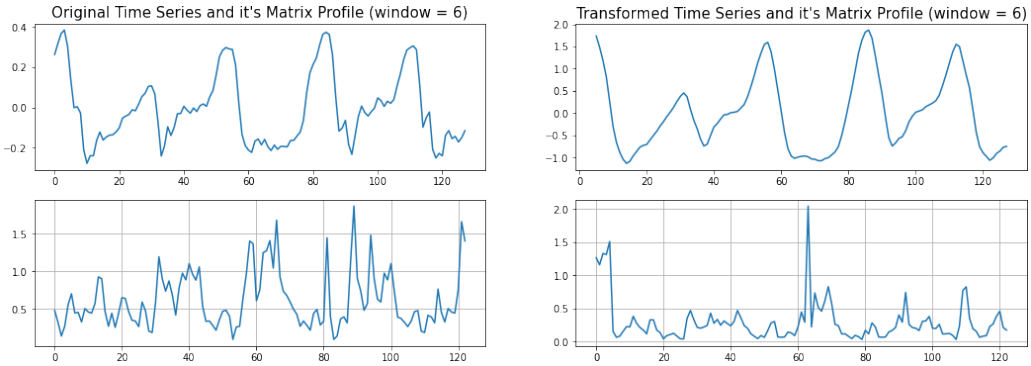


Figure 30: Matrix profile of the same time series, raw and transformed

As it can be seen the two matrix profiles are slightly different, it can also be noted that in the first 3 motifs catch immediately the attention while in the second case an anomaly has more prominence.

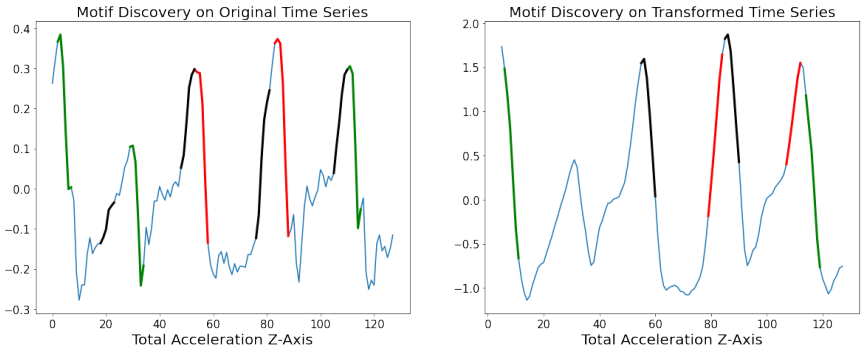


Figure 31: Motifs identified in the two cases presented

By deciding as maximum threshold of motifs to find 6, in Figure 31, I found in the both cases 2 types of descending patterns and one type of ascending, but in the first case the algorithm was a little more precise, probably because the transformation smoothed out some non-noise value and the amplitude scaling increased the focus on anomalies as it can be seen in Figure 32, in the first case it seemed very small descending patterns and sudden volatility of the values were identified as anomalous, while in the second case it's clear even part of a motif identified in the first case was considered anomalous, this made me consider great caution in considering eventual transformations.

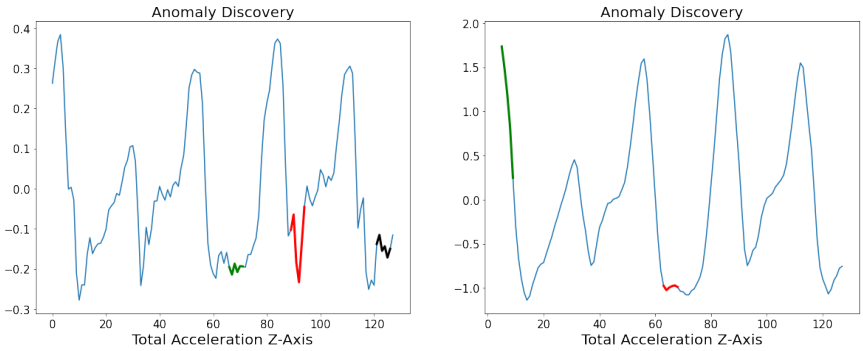


Figure 32: Anomalies identified in the two cases presented

## 4.3 Classification with Transformations and Approximations

Before the classification task I decided first to test some approximation techniques, as already noted in the case of clustering all the following methods gave worst results than the raw data. I first tried the Discrete Fourier Approximation by using 16 coefficients, the Piecewise Aggregate Approximaton with 15 segments and starting from it the Symbolic Aggregate Approximation with 10 symbols, I plotted in Figure 33 the inverse approximation on the original rescaled time series to look at how it reduced the information.

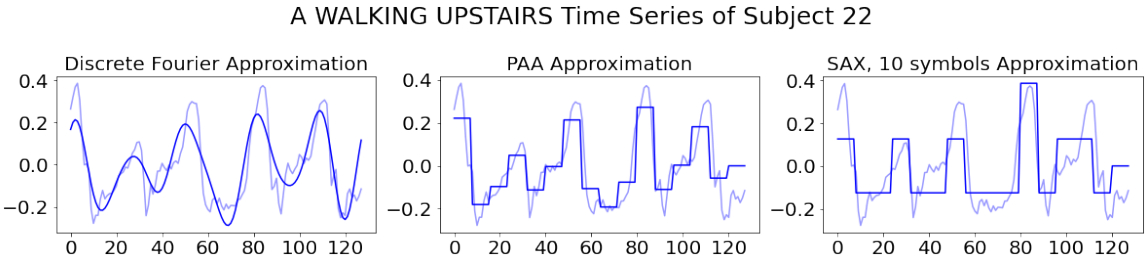


Figure 33: Approximation of one of the time series

I also tried some transformation on the time series before performing the approximation but I considered that the removal of differences in trends, mean, variance, amplitude or the smoothing of the time series was detrimental to the classification task as I guessed that some of the most important predicting features I focused on in previous classification paragraphs included some of those information and the last few considerations in the motif and anomaly discovery paragraphs.

| | **DFT (16 coefficients)** | | | **PAA (15 segments)** | | | **SAX (10 symbols)** | | |
|---|---|---|---|---|---|---|---|---|---|
| Distances | Euclidean | Manhattan | DTW | Euclidean | Manhattan | DTW | Euclidean | Manhattan | DTW |
| Accuracy | 0.914 | 0.917 | 0.914 | 0.914 | 0.914 | 0.914 | 0.911 | 0.911 | 0.911 |
| **Walking Upstairs:** | | | | | | | | | |
| F1-score | 0.694 | 0.707 | 0.694 | 0.694 | 0.694 | 0.694 | 0.680 | 0.680 | 0.680 |
| ROC AUC | 0.852 | 0.846 | 0.875 | 0.854 | 0.856 | 0.886 | 0.847 | 0.742 | 0.872 |

Table 5: Classification results by Approximation Technique

Then for the classification task I considered the Discrete Fourier with 16 coefficient the most apt approximation having the signals such a natural almost sinusoidal shape but to be sure of the choice I classified with all three approximation techniques and the results can be seen in Table 5 (the nearest neighbours selected were 5).

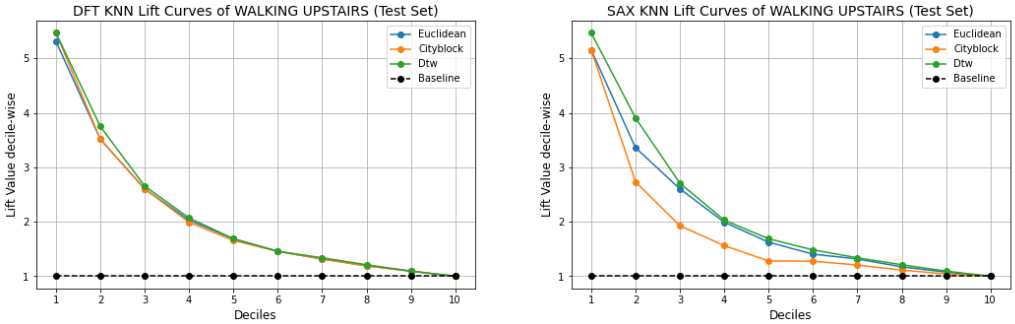


Figure 34: Lift Charts comparison with DFT and SAX approximations and different metrics

The test set I used for this classification will be the one I will also use from now on, I basically sampled the original test set into another 350 time series. Even though slightly my guess was correct and the DFT results were more interesting and higher than the others, further more just as a conceptual exercise I tried to look if the probability of correct guesses was different by sorting out the probabilities of prediction to visualise those results, I opted for a lift chart, as I did in Figure 34, there it can be seen clearly that on average the DFT approximation performed better and similarly with all the metrics while with SAX I got comparable results only with the DTW metrics. Finally, the price for the approximation is not that high, as a comparison with the previous metrics the KNN algorithm fitted with the DTW metric on the raw 350 time series of the training set, on the test set gets 0.95 accuracy (about 0.04 from the highest), while for the metric referred to the label Walking Upstairs gets 0.86 as the F1 score (about 0.16 from the highest) and 0.94 as the AUC score (about 0.07 from the highest), thus if the complexity of the computation won't be high I will prefer the raw time series, otherwise I will choose the DFT approximation if having coefficients is not a problem for the classifier, in which case I will prefer the Piecewise Aggregate Approximation as it is also a good approximation.

## 4.4 Shapelets Discovery and Shapelets-based Classification

For the Shapelet Distance-based classifier I used the raw 350 time series of the training set and the 350 time series of the test set I defined previously. In order to extract the shaplets I used the learning based approach, where I applied a known heuristic (J. Grabocka et al., 2014) to derive the number and size of the shaplets to look for based on the shape of the input data. On a 350 time series dataset with 128 time stamps and the 2 classes (Walking Upstairs or not) the heuristic identified as best parameters 4 shapelets with 12 time stamps.
I also tried to run the heuristic on the approximated dataset with the Piecewise Aggregate Approximation (considered that I needed an approximation which was also a time series) but unfortunately it indicated 3 shapelets with 1 timestamp each and I considered it too few to understand the underlying mechanisms properly, so I decided to extract the shaplets from the raw data, the resulting

accuracy of the learning process was 0.88. The optimizer was the Stochastic Gradient Descent, here I there's the problem that this dataset perform poorly if normalized (as a comparison the F1 score were around 0.10 in contrast to the not normalized data results in Table 6, so I had to avoid normalization risking that some gradients may explode or vanish. The weight regularizer I selected was 1%. Then in order to visualize the shaplets I normalized them (only the shapelets) on the time series shapes with an offset which in the case of the time series presented in Figure 35 was 0.2 (in order to reduce the amplitude of the normalized shapelets) then I also readjusted the normalized shapelets according to the original mean (with a simple sum) multiplied by 0.4 (in order to realign them better on the original time series as the shapelets found weren't themeselves perfectly aligned).

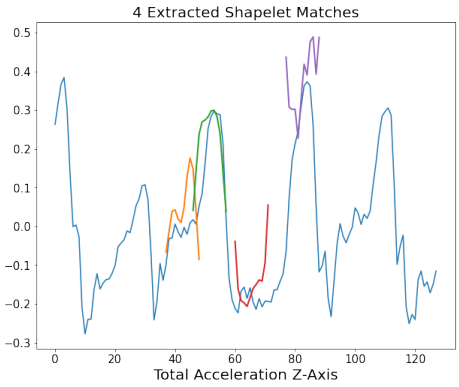


Figure 35: Shapelets extracted from the time series and aligned

As it can be seen the results are promising and the shaplets seem to represent important subsequences even though the violet, and in part the orange, seem to be a little forced on this particular time series.
Then I transformed the time series training and test set on the corresponding shapelet-transformed space using the model just trained and fitted and I ran an instance of the Decision Tree as a benchmark and multiple instances of KNN with 5 nearest neighbours and different distance metrics, namely Euclidean and DTW, with multiple constraints applied to the latter (None, Sakoe-Chiba constraints and Itakura constraints), those results (in Table 6) are grouped indistinctly under KNN as this algorithm's results were totally invariant to the used metrics and constraints with these particular shapelets, I also inserted Precision and Recall here to be more specific but they are almost homogeneous with the ones I got in the previous subsection (with 100% precision), so that it's not shadowed by the constant F1-score influenced by the low average recall. It cannot be said though that the classification gave good results as they were comparable to the ones of the KNN's run after different approximations, which were poorer than those on the raw time series (exception done for the results of the AUC score which suggests better thresholds).

| | Decision Tree | KNN (Euclidean and DTW) |
|---|---|---|
| Accuracy | 0.908 | 0.911 |
| F1-score (Walking Upstairs) | 0.68 | 0.68 |
| Precision (Walking Upstairs) | 0.94 | 1.00 |
| Recall (Walking Upstairs) | 0.53 | 0.52 |
| ROC AUC (Walking Upstairs) | 0.818 | 0.883 |

Table 6: Shaplet Based Classification Results

With different approaches than the learning one the classifiers gave similar and almost identical results even with Piecewise Aggregate Approximation applied (slightly poorer on the Decision Tree)

## 4.5 Convolutional Neural Network and Mini-Rocket Classification

Another classifier I used was CNN, which required a lot more data to work properly, as such I came back to the whole signal total acceleration Z-axis dataset of 10.299 time series dividing it accordingly to the training and test set selected by the authors as a first step I built the convolutional basis which comprised 3 convolutional 1-dimensional layers with the Rectified Linear Unit (ReLU) as activation function and the number of output filters in the convolution I set was in sequence 16, 32 and 64, while the length of the convolution window (kernel size) I set was in sequence 8, 5 and 3. Then I set

a 1-dimensional Global Average Pooling and a Dense Layer at the end with the Sigmoid activation function. During the training I set as metric to optimize the accuracy while I used the sparse categorical cross-entropy as loss function and an adaptive stochastic gradient descent optimizer (Adam). The learning rate the optimizer adapted to was a constant 0.001 during all epochs which I set to 5.

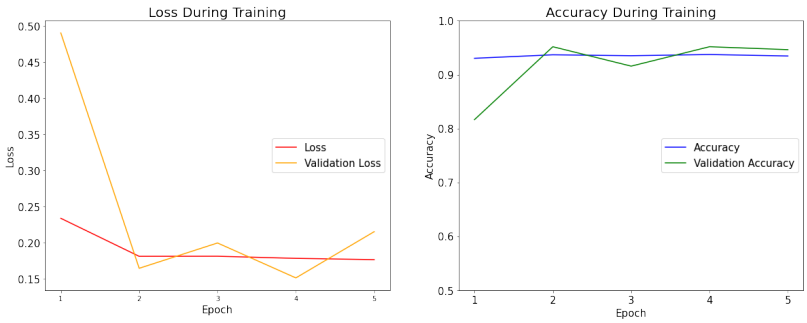


Figure 36: CNN Loss and Accuracy during Epochs

Finally I also tried to run the Mini-Rocket algorithm, I transformed the data accordingly and I ran the Ridge Classifier while normalizing and the results even slightly were the best so far.
I resumed the results of the classifier comparing it to a benchmark 1-NN classifier with the Euclidean distance metric (DTW was computationally too expensive). As it can be seen Mini-Rocket performed better in almost all metrics although just a little

| | KNN (Euclidean) | CNN | Mini-Rocket |
|---|---|---|---|
| Accuracy | 0.90 | 0.92 | 0.93 |
| F1-score (Walking Upstairs) | 0.71 | 0.73 | 0.76 |
| Precision (Walking Upstairs) | 0.78 | 0.98 | 0.98 |
| Recall (Walking Upstairs) | 0.65 | 0.58 | 0.63 |

Table 7: CNN and Mini-Rocket Classification Results

# 5 Sequential Pattern Mining

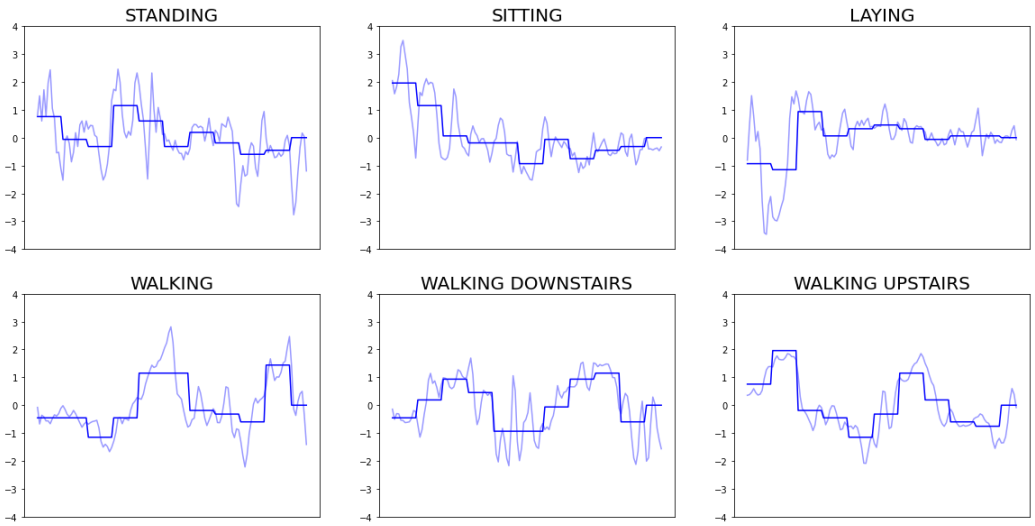


Figure 37: SAX, 20 Symbols Approximation (examples from subject 1)

For this task I extracted the sequential patterns with the Prefix Span algorithm, contrary to the classification task I decided to apply it on the entire training set presented in the original study.
First, this time I approximated the data differently, because here I just needed to understand possible patterns, for a better human understanding I re-scaled the data to mean 0 and standard deviation

1, then I approximated with a SAX transformation, with 10 segments and 20 symbols (Figure 37). Finally in Table 8 I resumed all the important subsequences found, by both considering the highest support and increasing number of length of subsequences for each label. It is interesting to notice that laying and sitting have the same most common subsequence or the fact that walking upstairs had the lowest most common pattern, followed by walking, while the rest have almost all median values (10 on 20 symbols), LAYING is interesting in particular because it shows the most constant patterns of all with the highest possible level of support and the longest possible subsequences.

| label | abs_support | relative_support | subsequences |
|---|---|---|---|
| STANDING | 703 | 51.16% | [[9]] |
| STANDING | 225 | 16.38% | [[11, 10]] |
| STANDING | 56 | 4.08% | [[7, 11, 12], [11, 10, 9]] |
| STANDING | 15 | 1.09% | [[16, 8, 7, 5], [16, 13, 7, 3]] |
| SITTING | 671 | 52.18% | [[10]] |
| SITTING | 220 | 17.11% | [[10, 8]] |
| SITTING | 62 | 4.82% | [[10, 10, 8]] |
| SITTING | 16 | 1.24% | [[7, 9, 12, 12]] |
| LAYING | 825 | 58.64% | [[10]] |
| LAYING | 301 | 21.39% | [[10, 10]] |
| LAYING | 81 | 5.76% | [[10, 11, 12], [11, 9, 10]] |
| LAYING | 20 | 1.42% | [[12, 12, 10, 7]] |
| WALKING | 766 | 62.48% | [[6]] |
| WALKING | 265 | 21.62% | [[7, 5]] |
| WALKING | 71 | 5.79% | [[5, 9, 6]] |
| WALKING | 18 | 1.47% | [[7, 16, 6, 7]] |
| WALKING_DOWNSTAIRS | 538 | 54.56% | [[9]] |
| WALKING_DOWNSTAIRS | 187 | 18.97% | [[9, 8], [9, 10]] |
| WALKING_DOWNSTAIRS | 51 | 5.17% | [[13, 5, 13]] |
| WALKING_DOWNSTAIRS | 16 | 1.62% | [[13, 5, 13, 7]] |
| WALKING_UPSTAIRS | 560 | 52.19% | [[3]] |
| WALKING_UPSTAIRS | 201 | 18.73% | [[5, 5]] |
| WALKING_UPSTAIRS | 58 | 5.41% | [[18, 5, 3]] |
| WALKING_UPSTAIRS | 15 | 1.4% | [[18, 5, 3, 18], [18, 3, 18, 4], [18, 4, 4, 5], [18, 6, 3, 18], [18, 7, 3, 3]] |

Table 8: Most frequent patterns by different lengths for different labels

# 6 Advanced Clustering

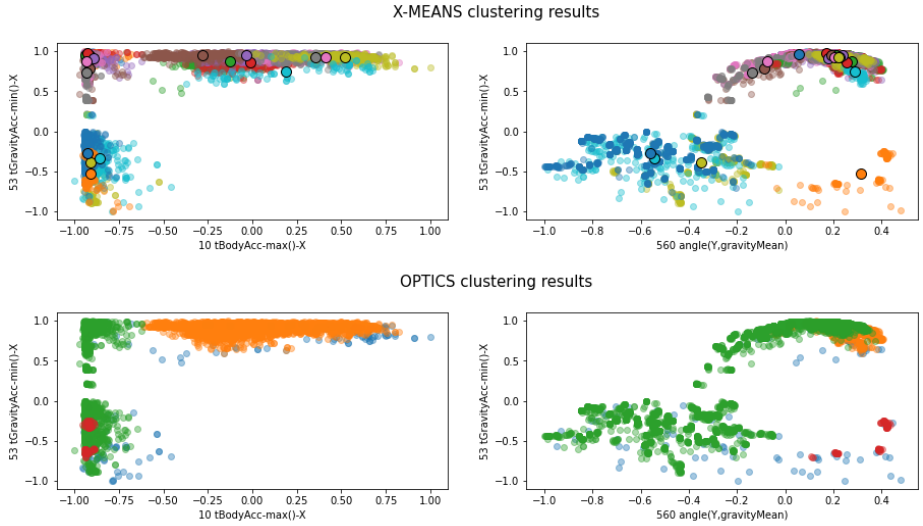


Figure 38: Advanced Clustering techniques

For this task I applied two clustering algorithms, namely X-Means and OPTICS, on the dataset reduced after dimensionality reduction and anomaly detection. I plotted the results in Figure 38, on the left in continuity with the pair of variables I presented during previous sections but I also considered another variable to use on the x axis where the distribution of clusters would be understood more easily (in particular the angle of gravity mean on the y axis). For X-Means I obtained an average silhouette score of 0.23, while the SSE and the distribution of the cluster can be seen in Table 9.

| **Cluster** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **N_Elements** | 383 | 422 | 565 | 488 | 162 | 105 | 342 | 171 | 406 | 229 | |
| **SSE** | 28,3844 | 41,9967 | 43,6245 | 37,5810 | 27,3347 | 20,3975 | 36,9944 | 34,5953 | 101,2948 | 85,0270 | |
| **Silhouette** | 0,2119 | 0,1461 | 0,2147 | 0,1941 | 0,2039 | 0,2177 | 0,1950 | 0,1705 | 0,3056 | 0,2046 | |
| **Cluster** | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | **Total:** |
| **N_Elements** | 692 | 93 | 550 | 351 | 641 | 449 | 404 | 495 | 304 | 93 | 7345 |
| **SSE** | 140,4214 | 44,3440 | 102,8721 | 77,4504 | 150,4746 | 71,6205 | 100,4643 | 102,6079 | 135,0213 | 37,3020 | **1419,8088** |
| **Silhouette** | 0,4549 | 0,3516 | 0,1963 | 0,1136 | 0,1067 | 0,2258 | 0,1249 | 0,1540 | 0,0616 | 0,2488 | **Average: 0,2051** |

Table 9: X-means clusters' SSE and Silhouette scores

For OPTICS I applied the DBSCAN method, I selected through trial and error (to reduce the noise) the number of minimum samples to 10, while the epsilon I set was 0.4, the results is shown in Table 10. I got 0.38 as average silhouette score, I got way less clusters and by also looking at the distribution of the original label in the data understanding part it's clear it divided more or less correctly the 2 large moving and stationary clusters. In my opinion it also identified a more realistic number of noise points, by also looking at the plots for different combination of variables, with respect to the ones I obtained during the anomaly detection part.

| **Cluster** | 0 | 1 | 2 | **Total** | Noise |
|---|---|---|---|---|---|
| **N_Elements** | 3177 | 3906 | 68 | 7151 | 194 |
| **SSE** | 1499,165 | 3786,472 | 22,86987 | 5308,506716 | - |
| **Silhouette** | 0,362242 | 0,366585 | 0,390922 | **Average:** 0.373249 | - |

Table 10: OPTICS clusters' SSE and Silhouette scores

# 7 Explainability

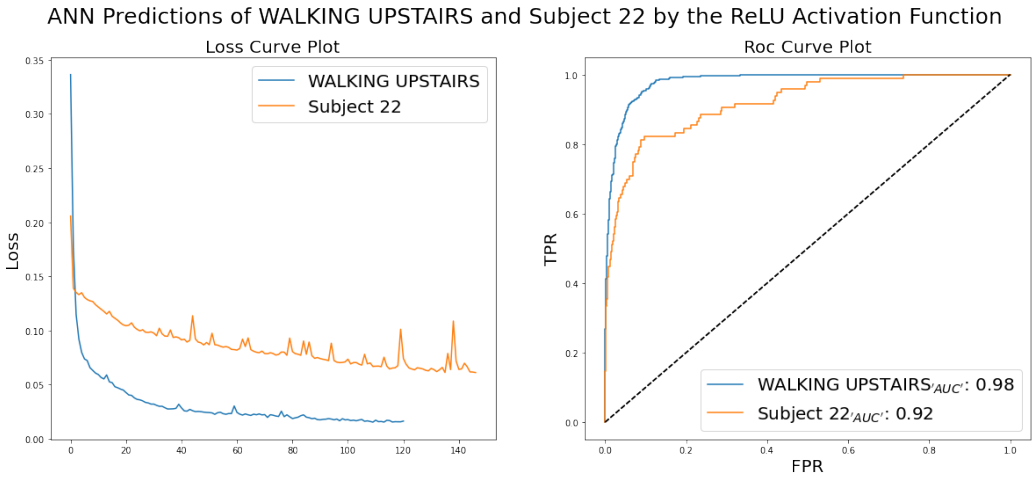


Figure 39: Loss Function Plot and the ROC curves

I decided to present this task by focusing on one of the most successful classifier I applied in the classification section, namely the Artificial Neural Network with the Rectified Linear Unit (ReLU)

as activation function. I decided to present two different predictions as I did during that part, by presenting both the predictions about Walking Upstairs and Subject 22 of the experiment. As it can be seen in Figure 39 I resumed the performances obtained with both target variables in two plots. The dataset I used was the same I used during the classification task, thus without the anomalies identified with the original split but shuffled records in the case of Walking Upstairs prediction and a new split of the entire dataset stratified on subject 22 in order to predict it. As before in both cases I didn't include in the models the categorical variables.

## 7.1 Explaining a correct prediction for Walking Upstairs

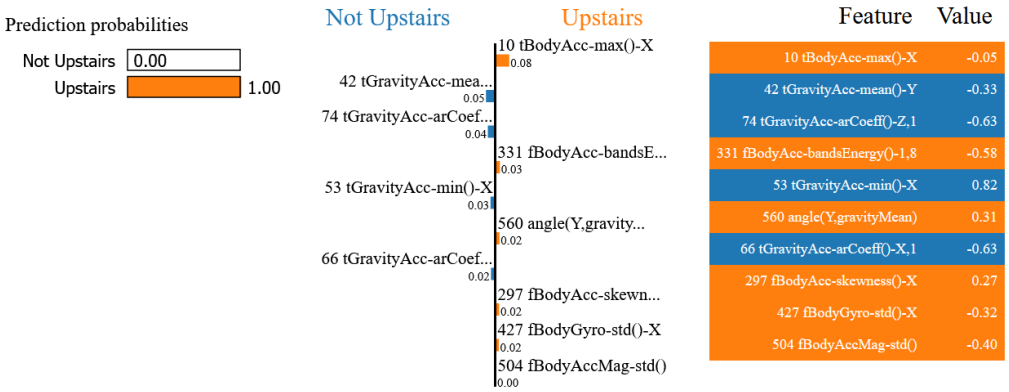


Figure 40: Local Interpretable Model-Agnostic Explanations for a record of Walking Upstairs

The first technique I applied was Lime, I applied it on the Neural Network and I explained the first instance of my dataset which had as label "Walking Upstairs". The results of the prediction of the classifier were perfectly correct in this case. The Explainer provided me the features as shown in Figure 40, as it can be seen some of the important features to predict the action of walking upstairs was the maximum accelaration on the X axis on the time domain which was already used specifically by the random forest while it wasn't considered as an important feature to split on by the decision tree, the one though that captured most my attention was the fact that tGravityAcc-arCoeff()-Z,1 was here considered negatively influential to determine the action.

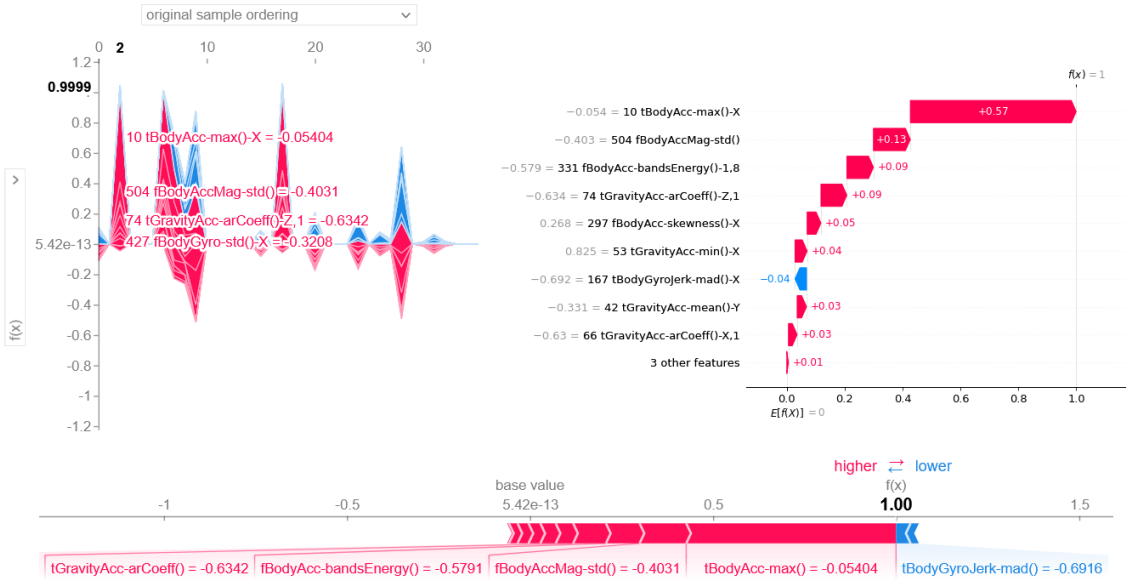


Figure 41: Shapley Additive Explanations for a record of Walking Upstairs

Then I decided to apply the SHAP method, as it can be seen, I analyzed the results it provided me on the same instance of the test set used in LIME, it gave me some results in contrast with the previous ones but it generally confirmed the positive role of the body acceleration into determining

the target variable but it contradicted the result for the tGravityAcc-arCoeff()-Z,1 feature, while both considering it important though. Around another instance, just before the 30th we can see that there is a sudden peak with negatively influential feature, in that case not represented in the graphs (where I preferred to show results to compare to the Lime's ones) it is confirmed by SHAP that the mean gravity acceleration on the Y axis on the time domain is negatively determining the result of the prediction.

## 7.2 Explaining a wrong predictions of Subject 22

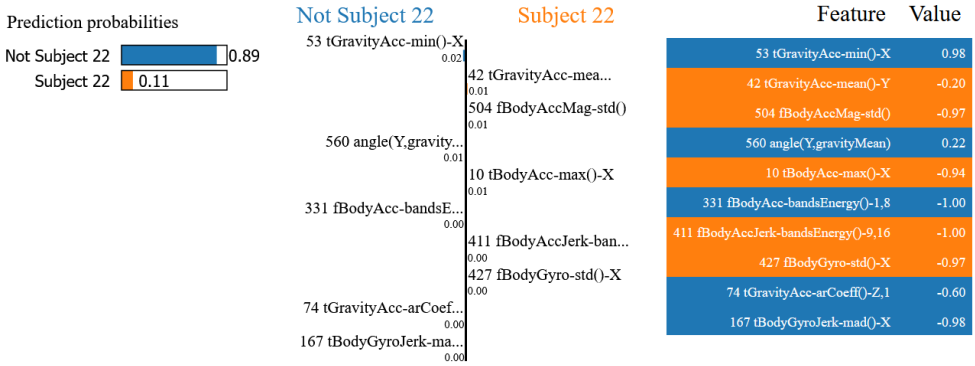


Figure 42: Local Interpretable Model-Agnostic Explanations for a record of Subject 22

Now just as before I moved to the most difficult to interpret original results. The model I am gonna explain reached high levels of good predictions with all metrics, but here I am gonna tell how much emphasis should have been put on some features to reach better results. We have that here with Lime that the mean gravity acceleration on the Y axis weighted positively on the model and tried to move it on the right direction, in this case the wrong prediction was determined by the minimum gravity acceleration registered on the X axis.

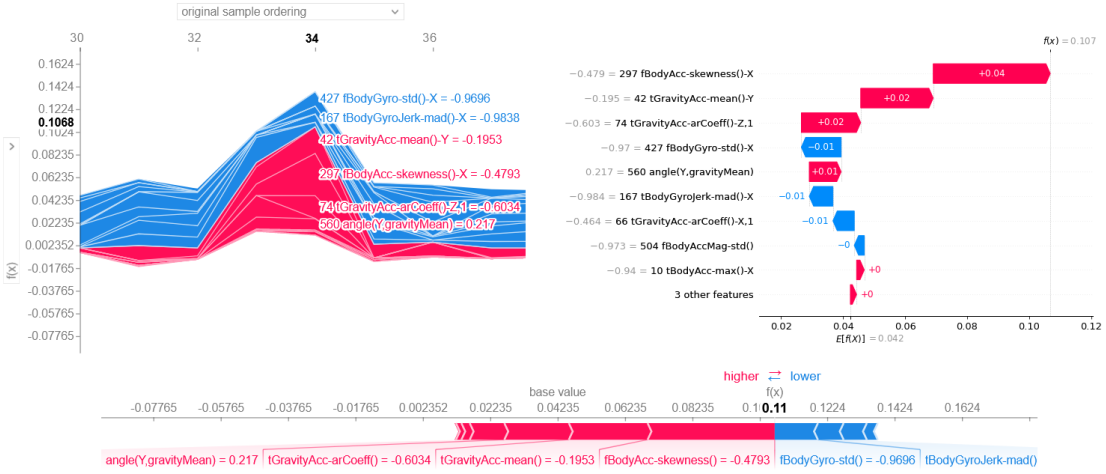


Figure 43: Shapley Additive Explanations for a record of Subject 22

Then with SHAP I discovered that it's probably the skewness of the accelleration on the X axis that helped correctly formulate the very small probability of the correct prediction, followed by the aformentioned feature. These two were probably given low priority and weight with this particular instance, thus the wrong assumption.
Finally, to conclude my report, I also studied a comparison of the weight attributed to the features of multiple instances analyzed by SHAP, as it is possible to look at it in Figure 44.
It's clear how the two dataset used are very differently balanced, as in the first case there is a clear direction of the weight attributed to the features, while in the second case there are sporadic observations with very different weights, while most observations as expected had features which contribute towards more negative assumption of the identity of the individual performing the actions.
It's also possible to see how some features had higher peaks even though the output values were in

a stricter range as it was more difficult to come up to a correct answer.

For this reason it may be better to explain models after applying some balancing technique, even though with this particular case, as shown previously, having fewer records dramatically lower the performances and it's not worth it to just explain the model.
To better understand the model it may be more useful to gather more data in that sense, even though the classification task I designed wasn't obviously expected to be the objective of the experiment.

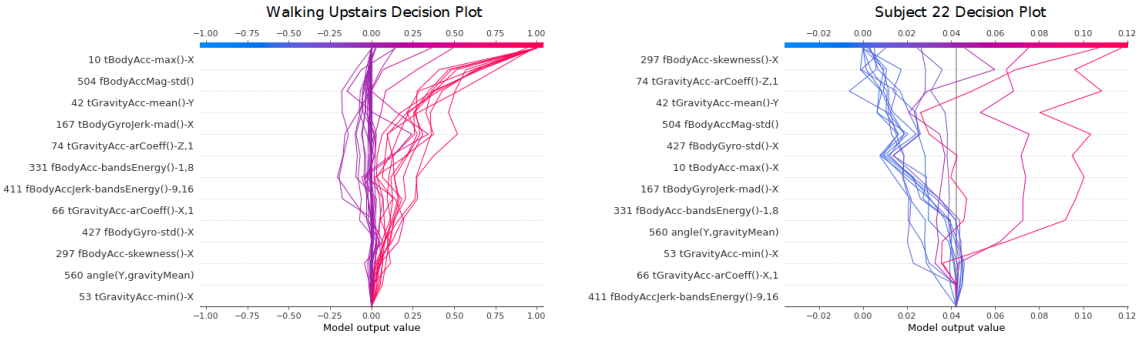


Figure 44: Decision Plots recap for the two predictions