Università di Pisa

Data Mining II
a.a 2021-22

Analisi Dataset
Human Activity Recognition with Smartphones

A cura di
Eleonora Rossi, Gianmario Ercoli, Lorenzo Casarosa

Indice

1 Introduzione

Il dataset, oggetto delle analisi riportate in questo report, è lo "*Human Activity Recognition Using Smartphone*" che consiste in una raccolta di sei attività differenti: *walking, walking upstairs, walking downstairs, sitting, standing, laying.* I valori sono stati ottenuti attraverso un esperimento condotto con uno smartphone dotato di sensori inerziali e trasportato da 30 soggetti diversi.

1.1 Comprensione dei dati e preparazione

Il dataset **Human Activity Recognition Using Smartphone** è composto da due sub-datasets, il train e test set: il primo composto da 7352 valori suddivisi in 563 colonne; il secondo invece da 2947 records ripartiti in 563 colonne. Durante una preliminare fase di preparazione e comprensione del training set, è stata condotta una ricerca finalizzata ad individuare eventuali errori, valori nulli o duplicati: l'esito di tale ricerca è stato negativo.

1.2 Classificatori base

In seguito all'individuazione di *Activity* come variabile target sono stati considerati i due seguenti compiti di classificazione:

- Completo. Per questo task sono state utilizzate le sei categorie presenti nel dataset originale, per verificare la capacità dei classificatori di distinguere le differenti attività. Salvo diversa specificazione, questo sarà il task di predizione che i classificatori considereranno per individuare il corretto valore (della variabile target) da attribuire a ciascun record.
- Binario. Per completezza è stato seguito anche l'approccio binario, categorizzando il dataset in due classi (statiche e dinamiche). Esso tuttavia verrà adottato nei casi in cui il metodo lo richieda necessariamente o permetta una più semplice interpretazione dei risultati.

Nonostante il dataset di partenza fosse composto sia da training che da test set, al fine di rendere più attendibili possibile i risultati conseguiti dal classificatore, il test set è stato utilizzato esclusivamente come mezzo di valutazione delle performances reali e finali del classificatore in oggetto, mentre con la tecnica di cross-validation applicata attraverso Grid-serch è stato suddiviso il training set in un sub-train e validation set.

1.2.1 Decision Tree

Al fine di classificare il dataset originale è stato utilizzato il Decision Tree per il quale abbiamo provveduto in prima istanza a trovarne la configurazione ottimale tramite l'uso del validation set. Successivamente le performances del classificatore sono state testate sui valori reali del test set. Gli iperparamentri sono stati individuati tramite il metodo *RandomizedSearchCV*. La tabella, di seguito riportata, contiene il tuning degli iperparametri. Inoltre dalla Figura 1 è possibile vedere come sia presente una differenza consistente tra lo score ottenuto per il training set e quello per il validation, ciò potrebbe significare overfitting da parte del modello.

Parametri	Descrizione	Valori testati	Miglior valore
Criterio	Misura la qualità dello split	Gini, Entropy	Entropy
Max_depth	Profondità massima dell'albero	Range (1, 10)	9
Min_samples_leaf	Numero esempi necessari per lo split	2, 5, 10	5
Min_samples_split	Numero esempi necessari in un nodo foglia	2, 5, 10	2

Tabella 1: Hyper-parameters tuning per il Decision Tree

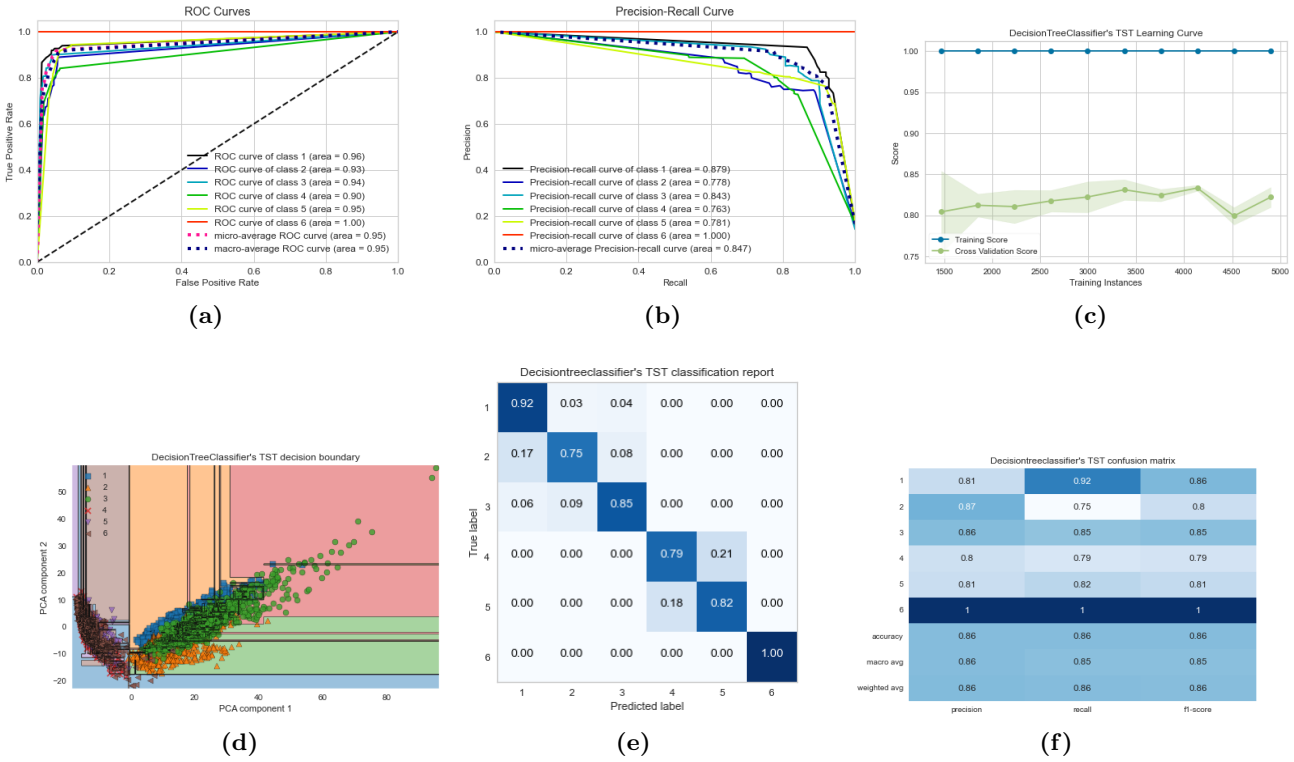


Figura 1: Risultati relativi alla migliore configurazioni del Decision Tree

1.2.2 K-nearest neighbors

Partendo dalla configurazione di cui fatto appena menzione (1.2.1) e successivamente alla normalizzazione dei valori tramite il metodo *StandardScaler*, il medesimo task di predizione svolto dal Decision Tree è stato adottato nuovamente con il KNN. Di seguito la tabella contenente il tuning degli iperparametri. Dalla Figura 2c è possibile vedere una maggiore convergenza delle curve di training e cross validation score a indicare una maggiore capacità del modello di generalizzare rispetto al Decision Tree.

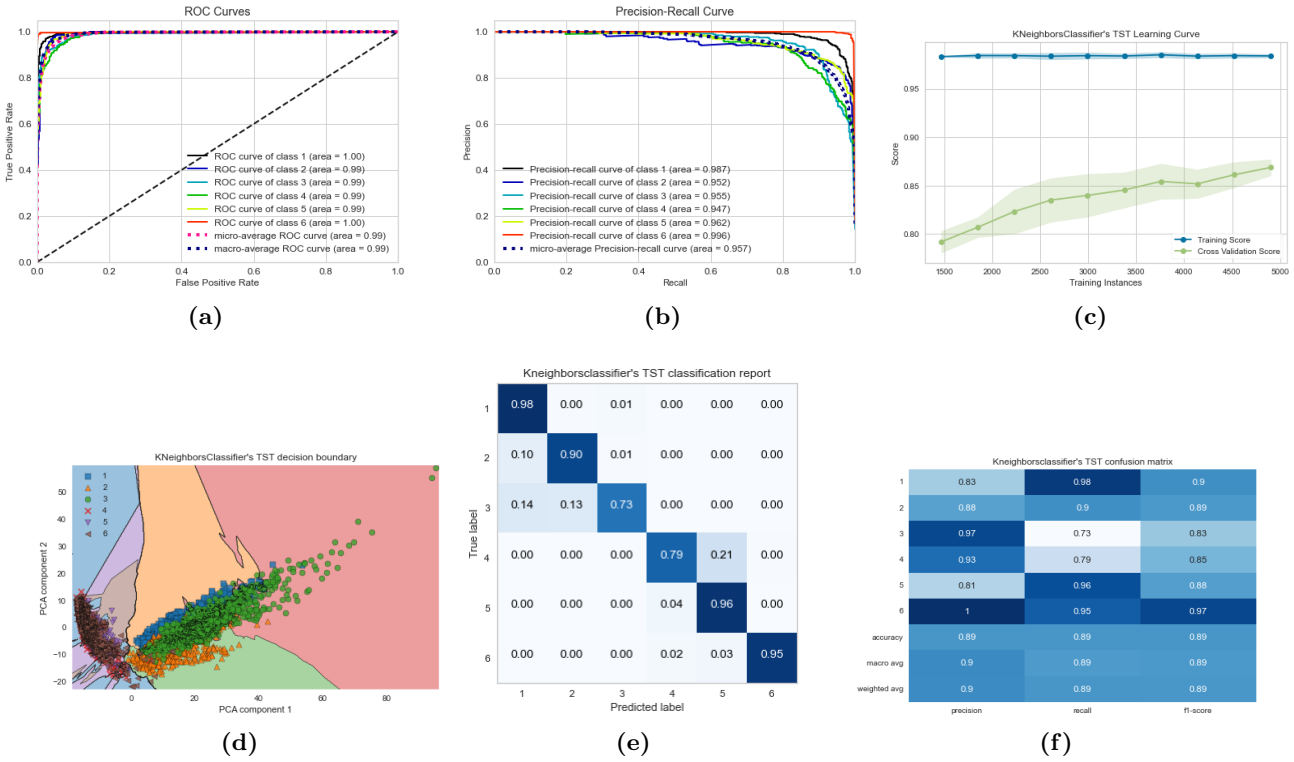


Figura 2: Risultati relativi alla migliore configurazioni del K-nearest neighbors

Parametri	Descrizione	Valori testati	Miglior valore
n_neighbors	numero di vicini	range(2,$\sqrt{n\_features}$+1, 2)	24
weights	funzione di peso utilizzata per la predizione	uniform, distance	distance

Tabella 2: Hyper-parameters tuning per KNN

1.2.3 Comparazione tra i classificatori

Possiamo concludere che il dataset sia facilmente classificabile da entrambi i modelli. Infatti, osservando la confusion matrix, si può notare come sia stato possibile ottenere punteggi sia di F1 score che di accuracy prossimi al 90%. Analizzando le *decision boundaries* generate da entrambi i modelli è evidente che entrambe siano molto simili tra loro, con la differenza che in quella del Decision Tree è presente un dettaglio maggiore. Il modello, infatti, suddivide le zone anche sulla base di pochi esempi, a riprova del fatto che il classificatore applicato a questo dataset tenda all'overfitting. In conclusione, nonostante entrambi i modelli sia in grado di discriminare sufficientemente bene le classi, il KNN ha il vantaggio di evitare maggiormente l'overfitting e per questo è possibile considerarlo, tra i due classificatori base, il migliore.

1.3 Outlier detection

Per classificare determinati records come outliers o meno sono stati utilizzati dei metodi più sofisticati: due approcci *density based* (LOF e DBscan), un metodo *distance-based* (KNN), un metodo *angle-based* (ABOD) e uno *model-based* (Isolation Forest). Di seguito è presente la tabella con il numero di outliers identificati da ciascun metodo.

Metodi	N.Outliers	Top 1%
LOF	9	1
DBSCAN	6742	
KNN	730	7
ABOD (5)	1089	10
ABOD (10)	738	7
Isolation Forest	552	5

Tabella 3: Outliers totali e top 1% individuati dai metodi. Il DBSCAN non associa un'anomaly score ma semplicemente l'etichetta.

Per il LOF e l'ABOD sono stati testati come possibile numero di vicini un range di valori che va da 2 alla radice quadrata del numero di samples presenti nel dataset successivamente raddoppiata. Per il KNN come numero di vicini ottimale è stata scelta la radice quadrata del numero di records.
Per il DBSCAN sono stati individuati un eps=9 e un minsamples=5 come parametri migliori, in quanto tutte le altre configurazioni portavano a considerare la quasi totalità dei records presenti nel dataset come anomalie. Considerazioni più interessanti sono emerse in seguito all'utilizzo del metodo ABOD: infatti si è notata una differenza importante tra 5 e 10 vicini, in quanto, nel primo caso è risultato un numero di outliers uguale a 1089, per poi stabilizzarsi intorno ai 730 all'aumentare dei vicini. Questo risultato può derivare dal fatto che per un numero eccessivamente basso di vicini, ci sia maggiore probabilità che questi siano localizzati tutti nella medesima direzione rispetto al punto di cui si sta valutando se sia inlier o outlier. La scelta del valore 10 è seguita a una riflessione sui risultati ottenuti, considerando che da tale punto in poi otteniamo una certa stabilità nel numero di outlier individuato.

Metodi considerati	Intersezione top 1%
LOF - DBSCAN	1
KNN - ABOD_10N	3
KNN - ABOD_5N	3
KNN - DBSCAN	7
ABOD_10N - ABOD_5N	3
ABOD_10N - DBSCAN	7
ABOD_5N - DBSCAN	10

Tabella 4: Intersezione del top 1% degli outliers

La tabella 4 mostra il numero di outliers che è possibile individuare intersecando L'1% valori anomali con maggiore score, che risulta essere particolarmente basso. Da analisi approfondite è sorto che tutti i metodi utilizzati in questo processo hanno individuato degli outliers non sempre condivisi con gli altri. Si evidenzia dunque una riduzione nell'intersezione tra i vari metodi: ad esempio il KNN condivide con il metodo ABOD soltanto 3 valori anomali (rispetto ai 7 totali). Si noti invece che tutti i top 1% degli outliers individuati dai metodi ABOD, KNN e LOF sono identificati come tali anche dal modello DBSCAN.

Il passo successivo è stato quello di eliminare gli outliers dal dataset: dato il basso numero di valori abbiamo effettuato la selezione a partire dal numero totale di outliers individuati dai vari metodi. In prima istanza, i valori rimossi dal dataset sono stati quelli considerati come outliers da almeno 3 metodi. Dato che l'analisi ha portato all'eliminazione di 605 records, numero eccessivamente alto, è stato ritenuto opportuno considerare come valore soglia l'intersezione di 4 metodi e successivamente di 5 (in modo che il numero di records classificato come outliers diminuisse). In seguito ad ogni step, il dataset è stato ripulito dalle anomalie individuate e poi processato dal Decision Tree e KNN. Dalla tabella seguente, è evidente come non ci siano stati significativi miglioramenti; per questi motivi è stata preferita la soluzione che eliminasse solo 286 outliers.

N. metodi	N. outliers eliminati	Accuracy %	
		KNN	DT
	0	89	86
>= 5	3	89	85
>= 4	286	88	85
>= 3	605	88	86

Tabella 5: Accuracy dei modelli KNN e DT al variare del numero di metodi considerati

1.4 Dimensionality Reduction

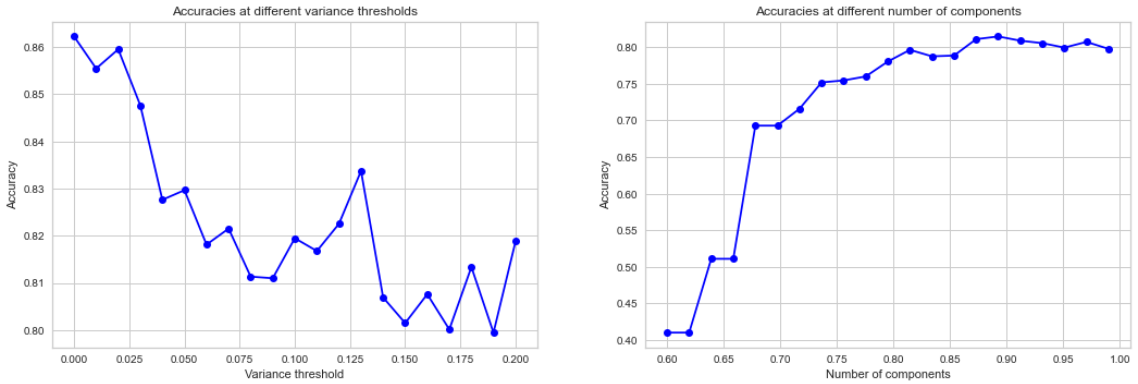
Data l'alta dimensione del nostro dataset abbiamo deciso di operare una riduzione della sua dimensionalità adottando quattro tecniche differenti: *Variance Threshold*, *Principal Component Analysis*, *Univariate Features Selection* e *Recursive Feature Elimination*. Per ognuna delle tecniche utilizzate abbiamo prima operato una ricerca della configurazione ottimale dei parametri e successivamente i dataset ridotti sono stati processati dai due classificatori KNN e Decision Tree (per i quali è stata effettuata la GridSearch ad ogni passaggio). Entrando nel merito del primo metodo è stato selezionato una range per stabilire il valore soglia da 0 a 0.2. Graficamente (Figura 3a) è possibile notare come un aumento del valore soglia produca una diminuzione quasi monotonica dell'accuracy.

Con riferimento al parameter tuning della seconda tecnica (PCA) si è resa necessaria una fase preliminare per stabilire l'andamento dell'accuracy rispetto al valore soglia tollerato della varianza tra i vari records. Anche qui, mediante l'ausilio del grafico sottostante, è possibile osservare come un aumento del valore della varianza comporti un aumento della accuracy e relative performances dei classificatori. Questo aspetto può essere spiegato considerando che una maggior tolleranza sul valore della varianza comporta un maggior numero di componenti utilizzate (per una soglia di 0,90 otteniamo un numero di componenti pari a 34). La Recursive Feature Elimination è una tecnica che si appoggia a un estimatore esterno per assegnare dei pesi

a ciascuna feature con conseguente selezione di quelle più importanti e relativa eliminazione delle altre. Gli estimatori utilizzati in questo progetto sono:

- Decision Tree (Accuracy: 0,87; N. Features selezionate: 33)
- Logistic Regression (Accuracy: 0,83; N. Features selezionate: 187)
- Random Forest (Accuracy: 0,85; N. Features selezionate: 123)

Data la quasi parità di valore per l'accuracy si è stabilito di considerare il Random Forest come il miglior estimatore per questa tecnica in quanto, mantenendo un numero abbastanza elevato di features, limita la perdita di informazioni.
Per quanto riguarda l'ultima tecnica l'attività di best-parameter tuning si è limitata a stabilire il numero di features da mantenere. Si è perciò testato un range 0-500 (la quasi totalità del dataset) ottenendo come valore ottimale 200. Riportiamo di seguito l'andamento dell'accuracy al variare del numero di colonne da mantenere (Figura 4),



(a) **(b)**

Figura 3: Accuracy ottenuta al variare dei parametri di varianza (3a) e componenti (3b)

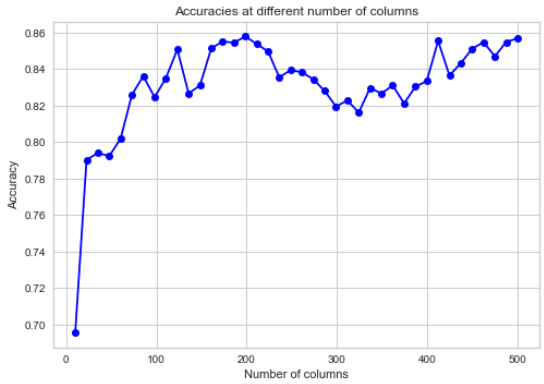


Figura 4: Accuracy ottenuta al variare del numero di colonne considerate per il metodo UFS

Dalla comparazione dei risultati conseguiti da ciascuna tecnica si evince come l'ultima tecnica testata (UniFS) risulti essere anche la migliore per quanto riguarda le performances del Decision Tree e del KNN.

Metodo	**N. features selezionate**	**Accuracy %**	
		KNN	**DT**
Variance Threshold	123	89	84
PCA	34	85	82
Univariate Feature Selection	200	90	86
Recursive Feature Elimination	123	90	85

Segue poi l'utilizzo di diverse tecniche con il solo scopo di visualizzare il dataset in 2 dimensione: *Random Subspace Projection*, *t-Sne*, *Isomap* e *Multi Dimensional scaling* (Figura 5).

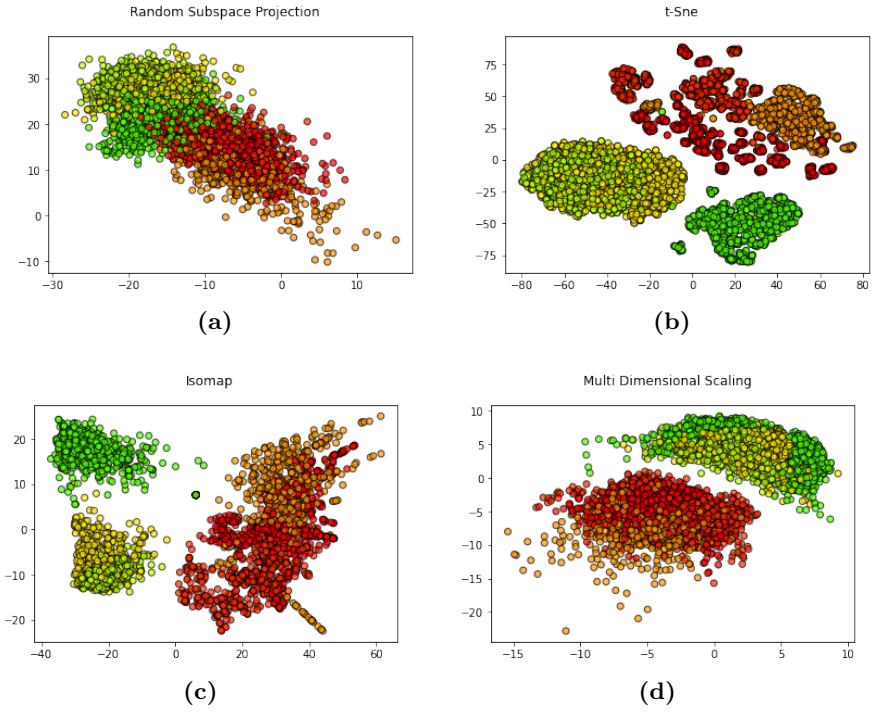


Figura 5: Visualizzazione del dataset attraverso diverse tecniche di Dimensionality Reduction

1.4.1 Improve Outlier Detection

Dopo aver scelto come tecnica ottimale di riduzione del dataset la Univariate Feature Selection, il dataset risultante è stato riesaminato dai metodi di outlier detection descritti precedentemente (sezione 1.3). La suddetta analisi è stata condotta al fine di verificare se un adeguato ridimensionamento del dataset potesse condurre a dei migliori risultati in termini di performance dei classificatori rispetto a quelle raggiunte con il dataset originale o comunque non ridotto. Di seguito riportiamo le metriche ottenute dai due classificatori (ricordiamo che un record è etichettato come un outlier se è riconosciuto tale da almeno n metodi di detenzione delle anomalie).

N. metodi	**Accuracy %**	
	KNN	**DT**
>= 5	90	86
>= 4	90	86
>= 3	90	86

Tabella 6: Accuracy dei modelli KNN e DT al variare del numero di metodi considerati

La tabella mostra come non vi sia alcun significativo miglioramento a seguito dell'applicazione delle tecniche di anomaly detection sul dataset ridotto.

1.5 Imbalanced Learning

Grazie alle analisi condotte abbiamo appreso che il nostro dataset non presenta sbilanciamenti evidenti tra le classi, come è possibile vedere nella figura (Figura 6) e dalla tabella (Tabella 7) seguente che riporta i valori ottenuti dai metodi testati sul Dataset originale.

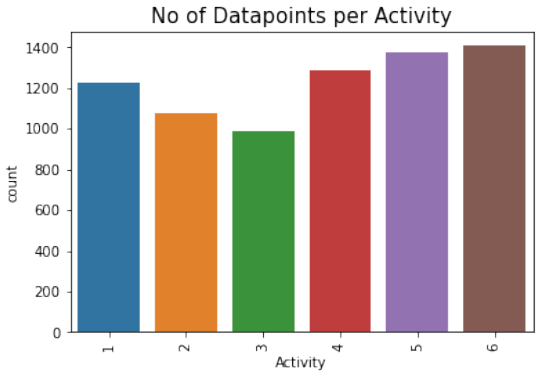


Figura 6: Numero di records per ciascuna classe

Dataset originale			
Undersampling	F1 Walking Down	F1 Laying	Accuracy
RandomUndersampler	0,86	1	0,86
Condensed Nearest Neighbor	0,74	1	0,60
Oversampling	F1 Walking Down	F1 Laying	Accuracy
RandomOversampler	0,85	1	0,87
SMOTE	0,82	1	0,85

Tabella 7: Valori ottenuti applicando metodi di bilanciamento sul dataset originale

Perciò, al fine di affrontare questo task, si è resa necessaria una fase preliminare di sbilanciamento "manuale" del dataset. In particolare sono state prese in considerazione le attività tre e sei (rispettivamente Walking down e laying) di cui sono stati ridotti del 90% i records. Il dataset così formato viene processato dal Decision Tree (previa predisposizione della Gridsearch) per valutarne le relative performances che riportiamo di seguito (Figura 7) e in particolare assistiamo a una forte riduzione dell'accuracy che passa dalla quasi perfezione allo 0,75. L'unico aspetto degno di nota è l'elevato valore del F1 score che rimane molto alto per l'attività 6 nonostante la forte operazione di sbilanciamento attuata.

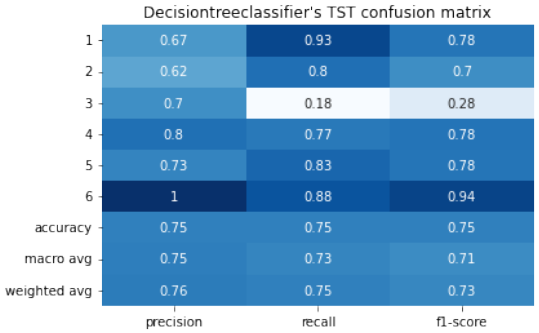


Figura 7: Confusion Matrix del DT su Dataset sbilanciato

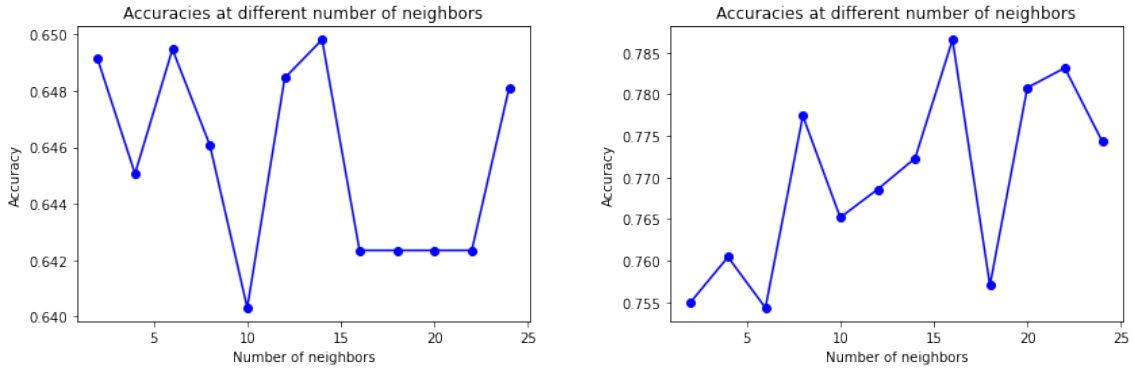
In seguito alle valutazione dei risultati ottenuti con il DT abbiamo deciso di implementare le varie tecniche di bilanciamento per osservare quale tra queste restituissero i più alti valori di Accuracy e F1-score una volta processati dal DT. Le tecniche utilizzate per questo scopo sono state:

- Oversampling: Random Oversampling e SMOTE;
- Undersampling: Random UnderSampler e Condensed Nearest Neighbor;

Per tutte le tecniche sopra riportate è stato necessario testare diversi tipi di parametri:

- Random UnderSampler: *sampling_strategy* = (majority, not minority, not majority, all) con best parameter *all*

- Random Oversampling: *sampling_strategy* = (minority, not minority, not majority, all), con best parameter *not minority*
- SMOTE: *sampling_strategy* = (minority, not minority, not majority, all), con best parameter *not minority*; *k_neighbors* = (2- $\sqrt{n\_features}$), con best parameter 15
- Condensed Nearest Neighbor: *sampling_strategy* = (majority, not minority, not ajority, all), con best parameter *majority*; *k_neighbors* = (2- $\sqrt{n\_features}$), con best parameter 13



(a) **(b)**

Figura 8: Accuracy ottenuta per diverso numero di vicini considerati per Condensed Nearest Neighbor (8a) e SMOTE (8b)

1.5.1 Confronto e valutazione finale

Riportiamo qui di seguito la tabella riepilogativa relative alle prestazioni delle tecniche precedentemente elencate. Si può facilmente evincere come qualsiasi tecnica di bilanciamento produca ottimi risultati per quanto riguarda l'attività laying (che nonostante lo sbilanciamento aveva comunque un F1-score e un'accuracy elevati), mentre trapelano importanti differenze tra le tecniche per quanto concerne l'attività 3, ossia Walking Down. Il peggior metodo risulta essere il Condensed Nearest Neighbor, ma anche il Random Undersampler non produce un gran miglioramento, a dimostrazione che le tecniche di undersampling per questo dataset non sono efficaci. Per quanto riguarda le tecniche di oversampling possiamo giungere alle medesime conclusioni per il metodo Random Oversampling mentre assistiamo ad un miglioramento evidente grazie allo SMOTE, il quale quindi si rivela essere la miglior tecnica di bilanciamento del dataset in esame.

Dataset sbilanciato			
Undersampling	F1 Walking Down	F1 Laying	Accuracy
RandomUndersampler	0,46	1	0,76
Condensed Nearest Neighbor	0,43	0,99	0,65
Oversampling	F1 Walking Down	F1 Laying	Accuracy
RandomOversampler	0,49	1	0,77
SMOTE	0,60	1	0,82

Tabella 8: Valori ottenuti sul dataset sbilanciato

1.6 Conclusioni

Nonostante la riduzione del dataset non abbia apportato miglioramenti dal punto di vista delle performance dei classificatori, nelle successive analisi verrà sempre utilizzato il dataset ridotto per due motivi: evitare la *curse of dimensionality* e in secondo luogo per favorire un minor running time degli algoritmi che verranno implementati. Avendo scelto come tecnica di riduzione la Univariate Feature Selection, il dataset è ora composto da 200 features e 7261 records.

2 Classificatori avanzati

2.1 Naive Bayes

Abbiamo addestrato il classificatore Gaussian Naive-Bayes con il dataset descritto nella sezione precedente. Non è stato necessario effettuare lo scaling dei dati in quanto questo algoritmo non è basato sulla distanza, quindi non avrebbe apportato miglioramento alcuno. I risultati mostrati di seguito (Figura 9) sono stati ottenuti senza applicare alcun processo di Hyper-parameter tuning.
Come è possibile vedere dal grafico della Learning curve (Figura 9c), c'è un miglioramento iniziale del cross-validation score, ad indicare che l'aggiunta di nuove istanze permette una migliore generalizzazione del modello, fino a raggiungere la convergenza da parte di entrambe le curve di score. Tuttavia è possibile notare che a seguito di tale risultato è presente un andamento variabile e decrescente per entrambe le curve. Probabilmente l'aumento del dati non porta ad un effettivo miglioramento se superata una certa soglia, conducendo il modello all'overfitting.

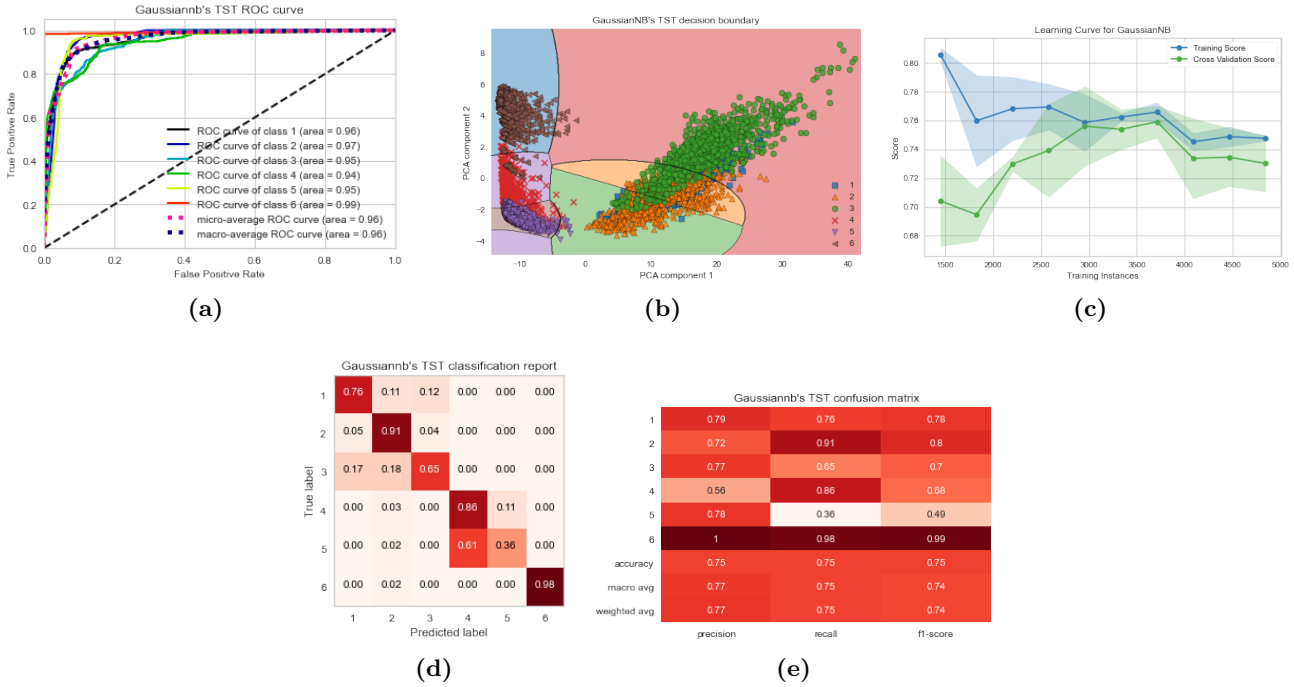


Figura 9: Risultati relativi alla migliore configurazioni del Naive Bayes

2.2 Support Vector Machine

Per quanto riguarda il metodo Support Vector Machine è stato necessario testare tre differenti tipi di kernels: LinearSVM, poly, rbf e sigmoid. Dato che SVM è sensibile allo scaling dei dati, è stato necessario in primis normalizzare i dati. É necessario poi fare una premessa: SVM non supporta implicitamente una classificazione multi-classe, è tuttavia possibile adottare diversi approcci per poter raggiungere tale scopo: l'approccio *one-vs-rest*, *crammer singer* e *one-vs-one*. Tutti questi approcci implicano la suddivisione del problema multi-classe in più problemi binari indipendenti fra loro. Mentre i primi due approcci sono stati testati come possibili parametri durante l'hyperparameter tuning del LinearSVM, per il metodo SVM è stato possibile utilizzare solo l'approccio *one-vs-one*.

Tra i vari kernels testati, il modello migliore è stato ottenuto con il metodo LinearSVM. Ciò potrebbe essere dovuto al fatto che in generale si fa uso del *kernel trick* quando si ha un numero di features superiore rispetto ai samples disponibili al fine di evitare l'overfitting. Questa condizione non si verifica nel nostro dataset, quindi il LinearSVM risulta essere più che sufficiente, nonostante anche gli altri kernels non lineari abbiamo comunque ottenuto alte performances (tra l'80 e il 90%). Nella tabella 9 è presente l'hyperparameter tuning del LinearSVM. Per i kernel non lineari sono inoltre stati testati i seguenti parametri: *degree* (solo per il kernel polinomiale) che controlla la flessibilità della decision boundary (range(2,13), migliore:3) e *gamma* (scale, auto; migliore: scale, auto per sigmoid). Come è possibile vedere dalla figura 10 il modello è in grado di generalizzare sufficientemente bene, non presentando alcun fenomeno di overfitting.

Parametri	Descrizione	Valori testati	Miglior valore
C	inverso della forza di regolarizzazione	0.001, 0.01, 1.0, 10.0, 50.0, 100.0	10.0
multi_class	strategia multi-classe (LinearSVM)	ovr, crammer_singer	ovr
tol	tolleranza per lo stopping criteria	0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0	0.001
class_weight	pesi delle classi	None, balanced	balanced

Tabella 9: Hyper-parameters tuning per LinearSVM

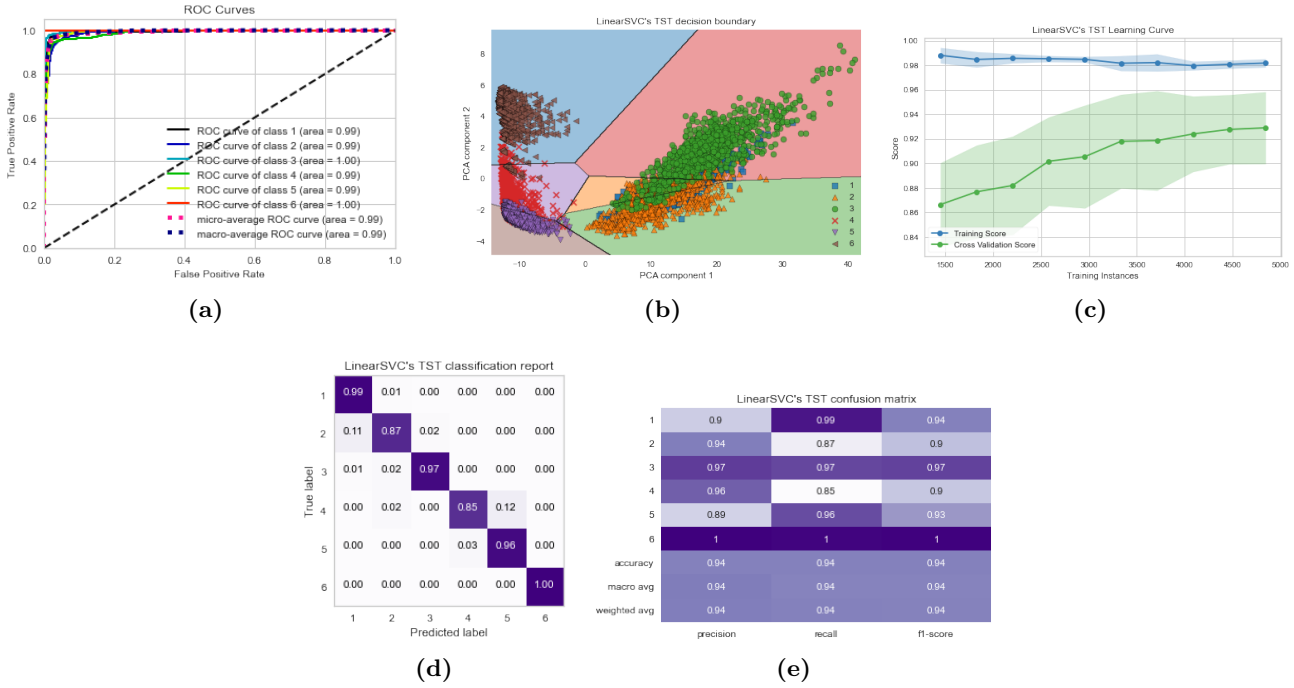


Figura 10: Risultati relativi alla migliore configurazioni del LinearSVM

2.3 Logistic Regression

Abbiamo addestrato due classificatori lineari: Logistic Regression e Ridge Regression. Per entrambi, al fine di utilizzare le tecniche di regolarizzazione, è stato necessario normalizzati i dati: per il Ridge Regression il miglior metodo è stato il RobustScaler, mentre per il Logistic Regression il MaxAbsScaler. Per il Logistic sono stati testati anche i parametri *Penality* (L1, L2 e elasticnet, migliore: L1) e *multi_class* (auto, ovr e multinomial, migliore: auto). Il confronto delle learning curves (Figura 11), prodotte dai due metodi, ci fornisce una chiara interpretazione su quale risulti essere il migliore per questo tipo di task: la curva prodotta dal ridge regression mostra una maggiore stabilità nel suo andamento (sintomo di una buona generalizzazione) e una maggior convergenza tra l'andamento del validation-score e del training score. Tuttavia osservando il grafico del decision boundary, (Figura 11b) è possibile notare regioni non nettamente separate ma anzi quasi sovrapposte. In particolare l'attività 4, che ci fornisce l'interpretazione più evidente, presenta un più basso valore della recall rispetto alle altre attività. Ciò evidenzia l'inadeguatezza che la decision boundary mostra nel dividere i records di tale classe (addirittura alcuni di essi giacciono su aree diverse). Nella tabella 10 è presente l'Hyper-parameter tuning del RidgeRegression.

Parametri	Descrizione	Valori testati	Miglior valore
alpha	forza di regolarizzazione	0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0	0.001
solver	ottimizzatore del modello	auto, saga	auto
tol	tolleranza per lo stopping criteria	0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0	0.1
class_weight	pesi delle classi	None, balanced	balanced

Tabella 10: Hyper-parameters tuning per il RidgeRegression

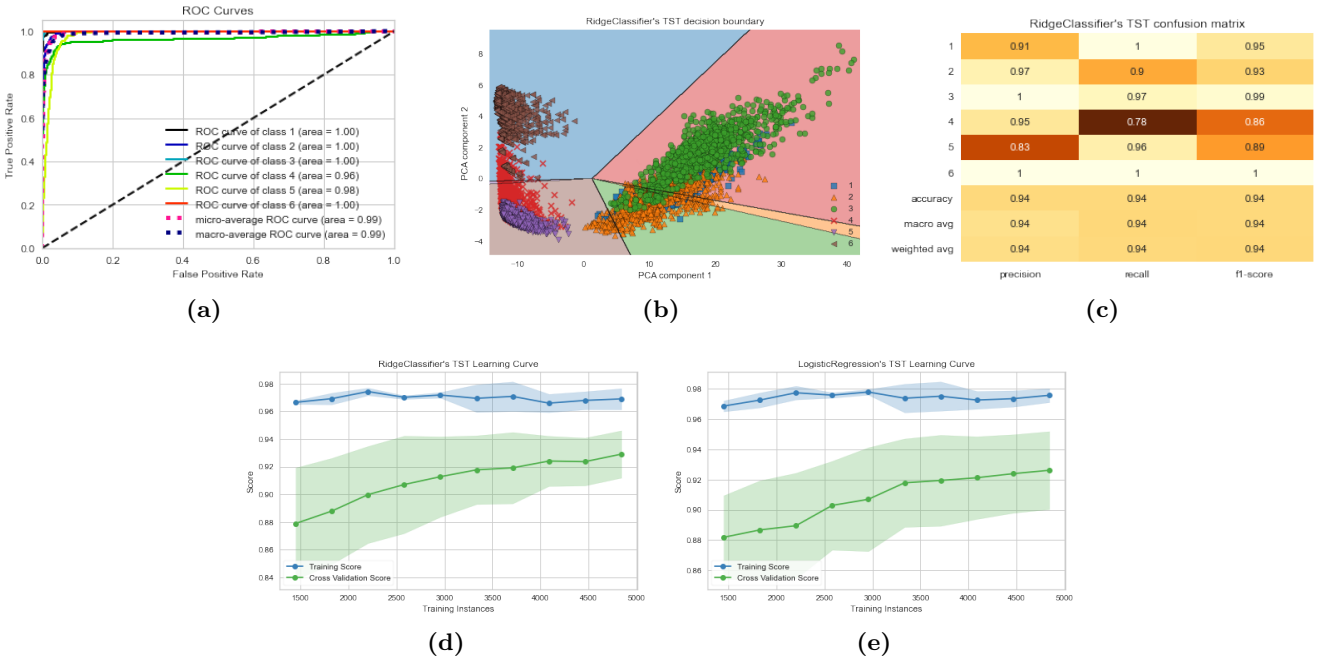


Figura 11: Risultati relativi alla migliore configurazioni del Ridge Regression e 11d learning curve del Logistic Regression 11e

2.4 Neural Networks

Per quanto riguarda il classificatore Neural Networks abbiamo sperimentato due tipologie differenti di reti: Single Perceptron e MLPClassifier. Per il primo non sono stati usati hidden layers, mentre per il secondo è stata testata sia la versione mono che quella bi-hidden layers. Inoltre, per i tre tipi di reti sono state applicate normalizzazioni differenti: per il Single Perceptron e per il MPLClassifier a doppio hidden layer è stato utilizzato lo Standard Scaler, mentre per il mono-hidden layer è stato utilizzato il Robust Scaler. Per le due versioni del MLPClassifier è stato effettuato un ulteriore hyperparameter tuning per quanto riguarda le unità all'interno degli hidden layers testando i valori da 1 al numero delle features. Di seguito si riportano le tabelle (Tabelle 11, 12) relative alle differenti configurazione adottate per la scelta dei parametri ottimali per i due differenti networks studiati.

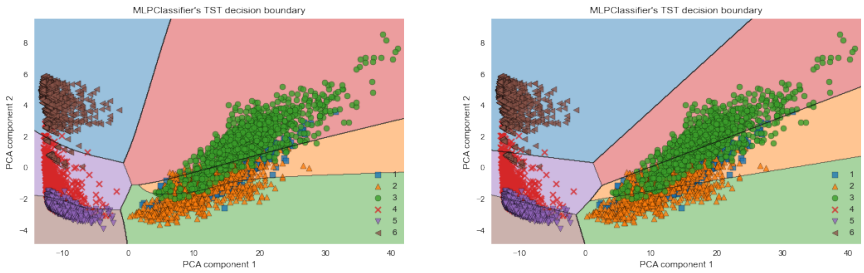
Parametri	Descrizione	Valori testati	Miglior valore
Alpha	Limita dimensione pesi contro overfitting	0.00001,.., 0.1, 1.0	0.01
Tol	Tolleranza per i criteri di arresto	0.00001,.., 0.1, 1.0	0.0001
Learning_rate_init	Tasso appr. iniziale. Aggiorna i pesi.	0.0001,..,0.1, 1.0	0.001

Tabella 11: Hyper-parameters tuning per il Single Perceptron

Parametri	Configurazioni testate	Miglior valore
Activation	'identity', 'logistic', 'tanh', 'relu'	'tanh'
Tol	0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0	0.0001
Learning_rate	'constant', 'invscaling', 'adaptive'	'adaptive'
Alpha	0.00001, 0.0001, 0.001, 0.01, 0.1, 1.0	0.1
Momentum	Range (0.1, 1.1, 0.1)	0.7000000000000001
Early_Stopping	True, False	False

Tabella 12: Hyper-parameters tuning per il 1-hidden layer

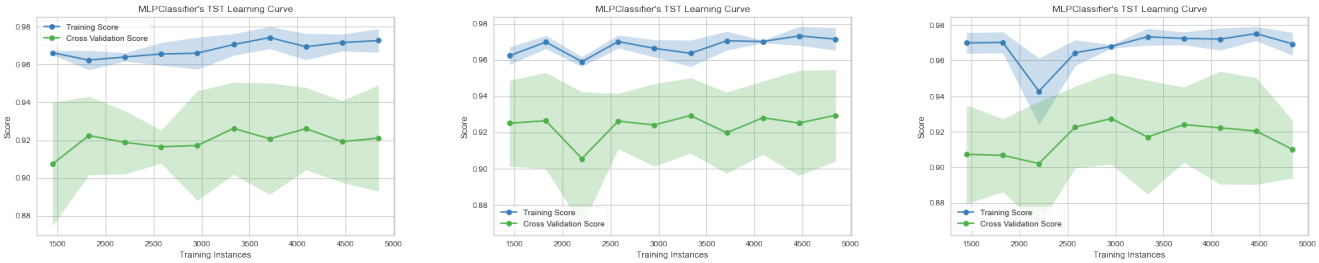
Si nota un'importante analogia tra il single hidden layer e il double hidden layers nell'ambito della separazione delle classi all'interno del decision boundary.



(a) **(b)**

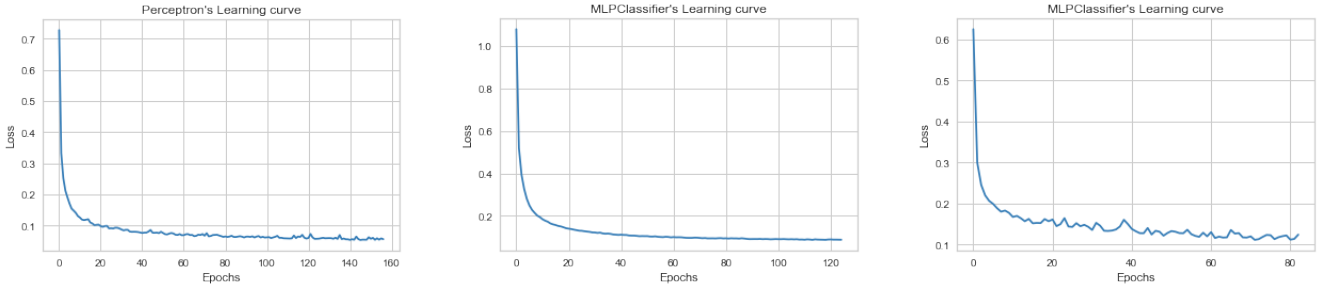
Figura 12: Risultati del decision boundary della versione a 1 hidden layer 12a e della versione a due hidden layers 12b

I risultati ottenuti mostrano una buona performance da parte di tutti i metodi, tanto che è possibile osservare un'accuracy del 94% per il percettrone singolo e un accuracy del 95% per entrambe le versioni di MLP. Tuttavia analizzando più nel dettaglio i plot ottenuti è possibile notare delle carenze per quanto riguarda il MLP a due hidden layer.
Analizziamo ora le learning curve ottenute (Figura 13). Per quanto riguarda le reti a multi hidden layers, nonostante la non convergenza delle curve nella learning curve non sono rilevati grandi cambiamenti nell'andamento del Training Score e del Cross Validation score (entrambe le configurazioni delle networks fanno registrare dei picchi di decrescenza di entrambi gli score, non riuscendo a mostrare una decisa convergenza tra i due). Tuttavia, per quanto riguarda quello a singolo hidden layer l'elevata semplificazione della classificazione porta le due curve ad avvicinarsi, sintomo di una soluzione più ottimale per il nostro dataset; tale avvicinamento è però accompagnato da una maggiore instabilità. L'instabilità risulta ancora più evidente nella versione a doppio hidden layers che mostra dei forti picchi verso il basso di entrambi gli score, giungendo ad un risultato basso del Cross Validation Score (il più basso fra i tre). Questo è probabilmente dovuto al fatto che aggiungere troppi layers conduce a un vero fenomeno di overfitting, motivo per cui abbiamo evitato di testare ulteriori layers per l'algoritmo.



(a) **(b)** **(c)**

Figura 13: Learning curve Single Perceptron 13a, versione a 1 hidden layer 13b e della versione a due hidden layers 13c



(a) **(b)** **(c)**

Figura 14: Learning curve **loss** Single Perceptron 14a, versione a 1 hidden layer 14b e della versione a due hidden layers 14

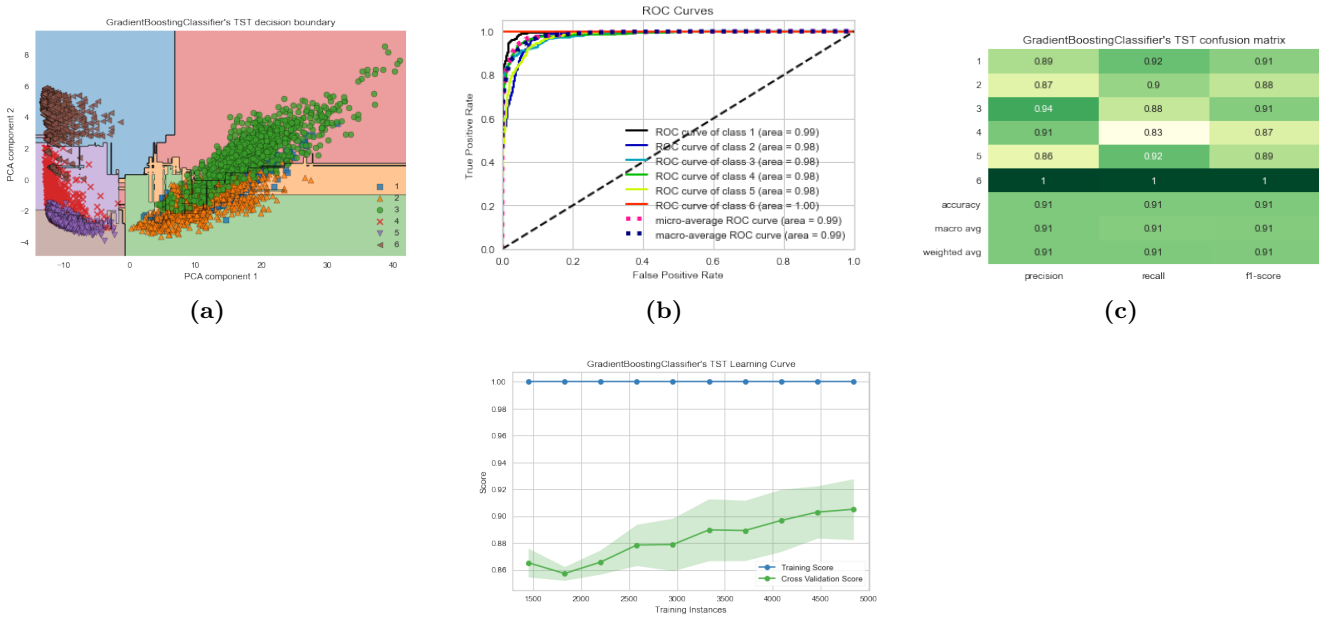
2.5 Gradient Boosting

Proseguendo nel classification task è stato testato il Gradient Boosting estimator, utilizzando i quattro seguenti classificatori: Gradient Boosting base, HistGradientBoost, LightGBM, XGBoost. In primo luogo è stato utilizzato il Gradient Boosting previa costruzione della GridSearch con i seguenti parametri:

Parametri	Descrizione	Valori testati	Miglior valore
learning_rate	Riduce contributo di ogni albero	0.1-0.3	0.15
n_estimators	Numero di alberi nella foresta	1,2,3,...,200	32
criterion	Misura la qualità dello split	Friedman_mse, Squared_error	Friedman_mse

Tabella 13: Hyper-parameters tuning per il Gradient Boosting

Una volta ottenuta la miglior configurazione dei parametri sono stati costruiti i seguenti plot (Figura 15): la decision boundary, la confusion matrix e la roc curve delle sei attività. Nonostante l'area sotto la Roc curve (Figura 15b) tenda a 1, la learning curve (Figura 15d) mette in evidenza la non convergenza tra la curva del Training score e del Cross validation score mostrando quindi come probabilmente questo algoritmo sia stato addestrato in maniera troppo mirata al training senza essere capace di classificare altrettanto bene i dati del validation set.



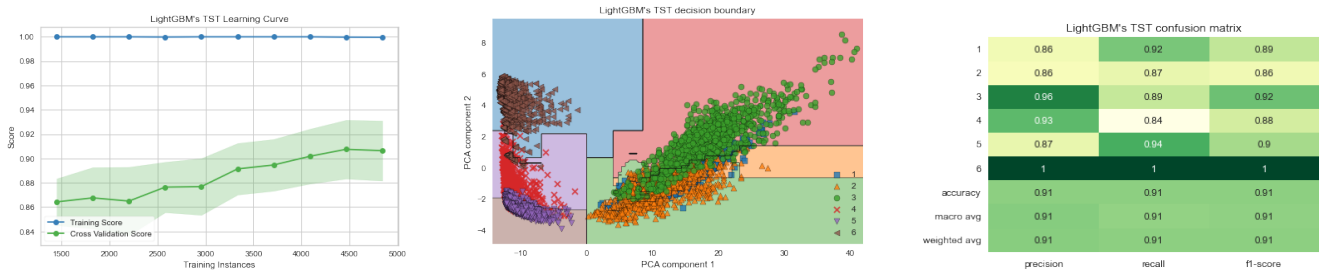
(a) (b) (c)

(d)

Figura 15: Risultati relativi alla migliore configurazione del Gradient Boosting base

2.5.1 Light Gradient Boosting Machine

Per questo classificatore ci siamo avvalsi di un'ampia molteplicità di parametri per la costruzione della relativa GridSearch: oltre ai tre sopracitati è stato usato anche il reg_alpha, il reg_lambda e il num_leaves. I primi due sono termini di regolarizzazione mentre l'ultimo rappresenta il numero massimo di foglie per l'apprendimento del classificatore. Utilizzando la configurazione migliore ottenuta, analizzando i grafici e le metriche che ne scaturiscono (Figura 16) è possibile asserire che abbiamo un miglioramento nell'andamento della curva del Cross validation score. Questa riesce a raggiungere un punteggio finale più alto, mentre notiamo comportamenti molto similari per quanto riguarda la decision boundary (con le classi talvolta sovrapposte) e le metriche relative alle 6 attività (anche accuracy e F1-score restano invariate a 0.91).

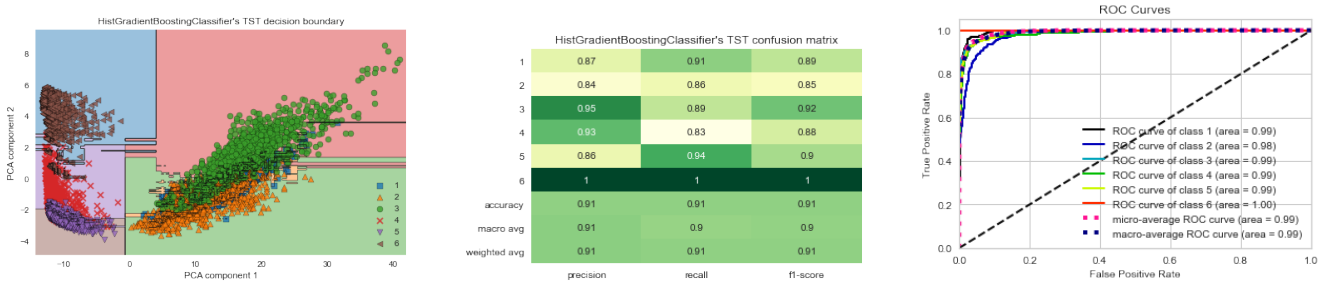


(a) **(b)** **(c)**

Figura 16: Risultati relativi alla migliore configurazione del Light Gradient Boosting

2.5.2 HistGradientBoost

Per il dataset preso in esame questo classificatore risulta essere il migliore rispetto agli altri tre in termini di accuracy ed f1-score ottenuti (0.97 per entrambi), mentre per quanto riguarda la decision boundary è possibile notare come sia presente un'eccessiva separazione delle sei classi presenti, ciò potrebbe comportare una poca capacità di generalizzazione su dati mai visti.



(a) **(b)** **(c)**

Figura 17: Risultati relativi alla migliore configurazione del Hist Gradient Boosting

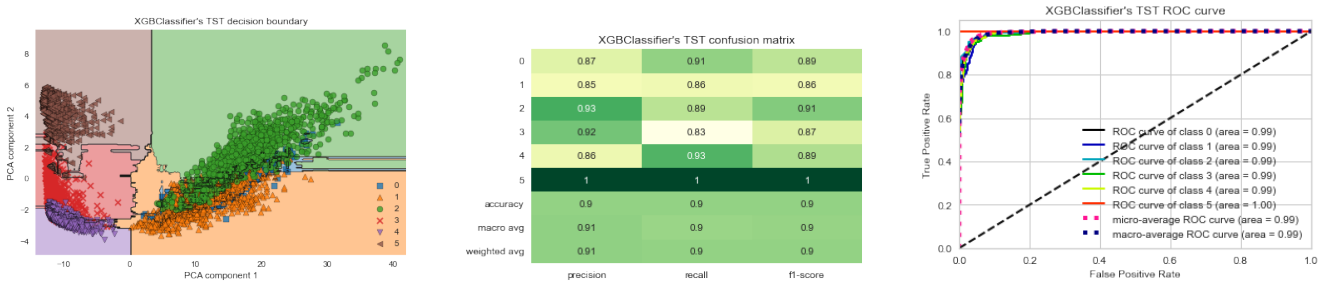
La miglior configurazione dei parametri che dà luogo ai valori appena descritti è la seguente:

Parametri	Valori testati	Miglior valore
Learning rate	0.05,0.1,0.15,0.2,0.25,0.3	0.25
L2 regularization	0.01,0.02,0.03	0.01

Tabella 14: Hyper-parameters tuning per HistGradientBoost

2.5.3 XGBoost

Al fine di permettere il funzionamento dell'algoritmo in questione si è resa necessaria in fase preliminare una codifica nel range delle classi che non andrà più da uno a sei ma da 0 a 5. Fatta questa precisazione e costruita come per gli altri classificatori la GridSearch è possibile dedurre come i risultati conseguiti da questo classificatore siano leggermente peggiori rispetto a quelli raggiunti dagli altri tre sopra descritti. Infatti il valore dell'accuracy e dell'F1-score si attestano intorno a 0.9, molto più basso ad esempio del HistGradientBoosting e la decision boundary si dimostra anch'esso incapace di distinguere in maniera netta e chiara le varie attività. Comunque il XGboost si dimostra un efficace classificatore per il dataset in esame.



(a) **(b)** **(c)**

Figura 18: Risultati relativi alla migliore configurazione del X Gradient Boosting

2.6 Ensamble Methods

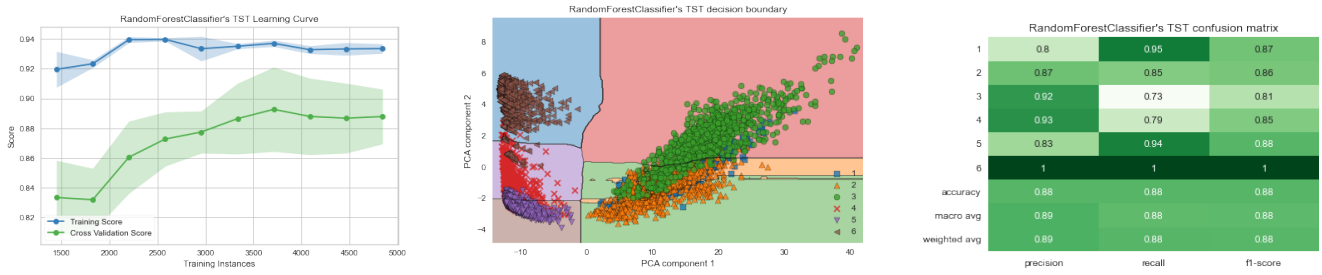
In questa sezione sono stati testati tre differenti tipi di meta-estimators: RandomForest, AdaBoost e Bagging

2.6.1 RandomForest

Il RandomForest, che utilizza il Decision Tree come classificatore base, ottiene risultati pressoché analoghi a quelli ottenuti dal Tuned Random Forest Classifier. Sia l'accuracy che l'F1-score si attestano intorno a 0.88. Il Random Forest con la configurazione ottimale è stato ottenuto mediante l'adozione della GridSearch con i seguenti parametri:

Parametri	Descrizione	Miglior valore
min_samples_split	[2, 5] + list(range(10, 101, 10))	2
class_weight	(None, 'balanced', 'balanced_subsample')	'balanced_subsample'
min_sample_leaf	[1, 2, 5] + list(range(10, 101, 10))	2

Tabella 15: Hyper-parameters tuning per il Random Forest



(a) **(b)** **(c)**

Figura 19: Risultati relativi alla migliore configurazione del Random Forest

Come è possibile notare dalle figure sopra riportate il Random Forest ottiene un buon risultato in termini di convergenza tra le curve del Cross Validation e del Training set ma mostra notevoli difficoltà nella suddivisione dei records nelle varie classi. La Decision Boundary evidenzia infatti come records appartenenti a classi diverse si trovino spesso mischiati tra di loro; non riesce quindi ad operare una netta separazione. In termini di metriche, ottiene un buon risultato, con valori di accuracy che si attestano intorno a 0.88 e F1-score dello stesso valore. Per quanto riguarda l'importanza delle features (Figura 20) il modello si concentra maggiormente su quelle relative all'accelerazione gravitazionale rispetto a quella corporea.

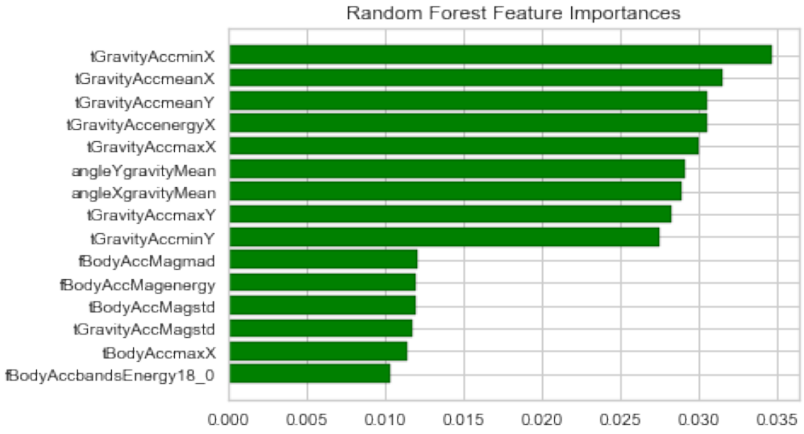


Figura 20: Importanza delle features attribuita dal Random Forest

2.6.2 AdaBoost

Al fine di approfondire le analisi è stato utilizzato l'algoritmo AdaBoost con tre diversi tipi di classificatori: Decision Tree, KNN e lo stesso RandomForest. La configurazione con il decision tree risulta essere di gran lunga la peggiore delle tre sia prima che dopo la costruzione della GridSearch. Si passa infatti da un'accuracy di 0.31 e 0.53 di f1-score a una situazione in cui accuracy e f1-score sono 0.66. Questi risultati sono confermati anche dalla learning curve (di sotto riportata, Figura 21) la quale evidenzia un andamento crescente e decrescente (numerosi picchi) sia per il training score che per il cross-validation score (nonostante quest'ultimo sia molto vicino alla curva del training score). Risultati differenti invece sono stati ottenuti combinando l'algoritmo AdaBoost con l'algoritmo KNN e RandomForest. È possibile dedurre dalle seguenti matrici di confusione e dalle learning curve come il processo di costruzione della gridsearch e la conseguente fase di parameter tuning abbia portato notevoli benefici alle configurazioni base di questi algoritmi.

Parametri	Pre-tuning	Tuned
Accuracy	0.88	0.88
F1-score	0.83	0.83

Tabella 16: Tabella confusion matrix pre e post tuning del Random Forest

L'unico aspetto che permane è la difficoltà che la decision boundary presenta nel distinguere nettamente le varie attività.

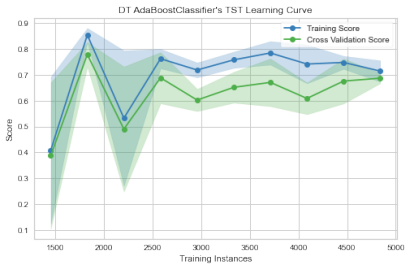


Figura 21: Learning curve della configurazione migliore dell'ensemble method con DT

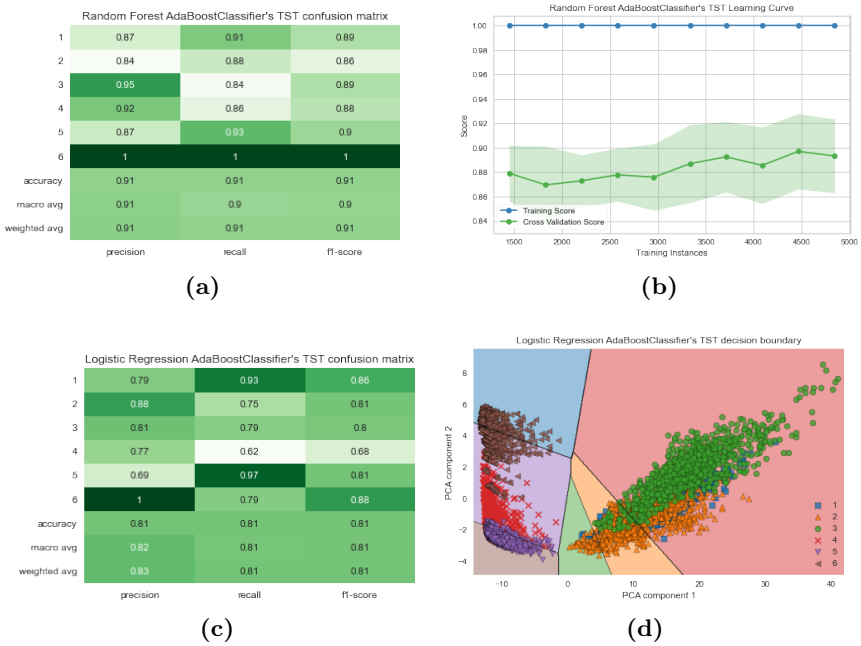


Figura 22: Learning curve 22b e matrice di confusione 22a dell'ensemble con RF e Decision Boundary e Matrice di confusione dell'ensemble con KNN 22c

2.6.3 Bagging

L'ultimo estimatore preso in esame in questa fase è l'Ensamble Bagging con il quale sono state condotte analisi similari a quelle fatte con l'AdaBoost (Decision Tree e KNN). In particolare possiamo notare subito delle forti differenze con il Decision Tree del precedente punto. Infatti in questa configurazione l'accuracy si dimostra già molto alta (0.87) ancora prima di arrivare al Tuned-DecisionTree-Classifier. Tuttavia se ci soffermiamo sulla Learning Curve (Figura 23a) notiamo un andamento instabile (e decrescente in alcuni tratti) in entrambe le curve, fattore sicuramente non positivo che può essere ricondotto quasi certamente all'overfitting. Per quanto riguarda invece il KNN le metriche risultano essere molto buone, ma presenta una forte incapacità nel separare le classi in sede di decision boundary (Figura 23b).

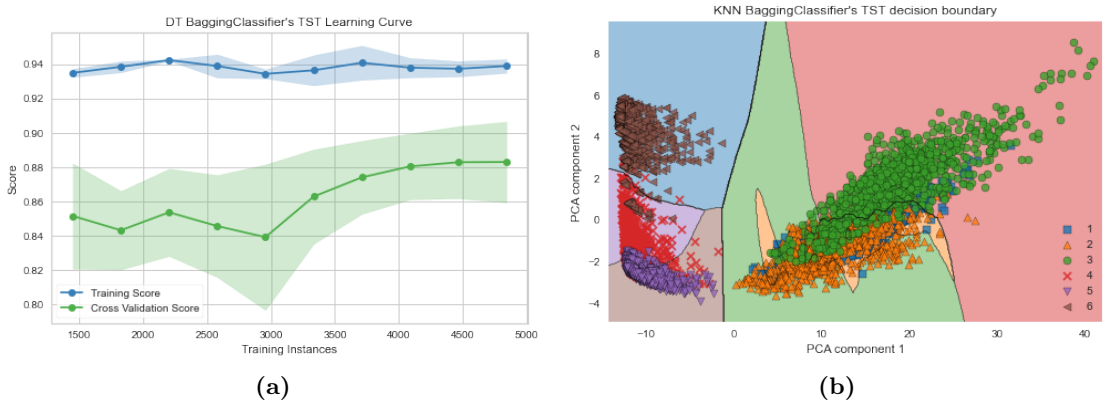


Figura 23: Learning curve del Decision Tree 23a e Decision Boundary del KNN 23b

2.7 Final Evaluation

Sebbene non sia stato possibile trovare un algoritmo che ottenesse risultati ottimali sia dal punto di vista della suddivisione dei records nelle varie classi sia per quanto riguarda i comportamenti del training e cross-validation score nella learning curve, è possibile tuttavia riscontrare performance molto buone nonché similarità tra algoritmi appartenenti a famiglie diverse. A questo proposito si riporta che il Support Vector Machine possiede la più alta stabilità nonché convergenza delle curve di apprendimento (10c). Tuttavia, questo algoritmo non si dimostra molto efficace se consideriamo la Decision Boundary che produce; al

contrario abbiamo riscontrato che il Random Forest testato nell'ambito dell'Ensamble method family giunge a risultati migliori in termini di suddivisione dei records in base alle classi del dataset. A livello generale, tuttavia, è possibile affermare che gli algoritmi di tutte le famiglie presentate si dimostrano molto abili dal punto di vista della classificazione, ottenendo valori di accuracy e F1-score molto alti.

3 Regression

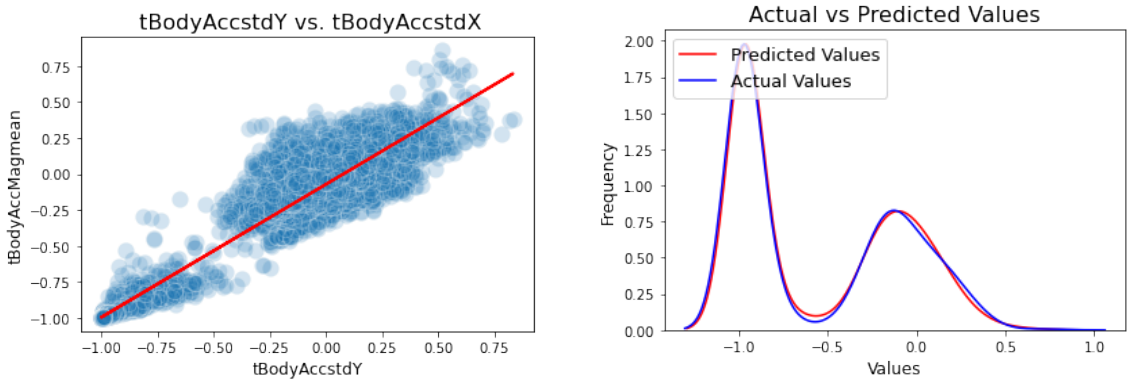
Per entrambi i tipi di regressione lineare (simple e multiple, rispettivamente) sono stati utilizzati i metodi di LinearRegression, Ridge e Lasso.

3.1 Simple Linear Regression

Nella risoluzione di questo task è stato necessario selezionare due features continue dal nostro dataset. In prima istanza è stata osservata la correlazione di coppie di features, criterio che però non si è dimostrato del tutto risolutivo in quanto era presente un elevatissimo numero di variabili con una forte correlazione. Ci siamo quindi avvalsi di un indice per misurare la bontà che la regressione effettua sui valori reali del dataset: cioè il valore della R^2 (Range 0-1). Lo studio di questo indice, accompagnato dal grafico composto da coppie di variabili, ha permesso di selezionare la *'tBodyAccMagmean'* e la *'tBodyAccstdY'* come variabili ottimali per lo svolgimento del simple linear regression task. La seguente tabella (Tabella 17) e l'alto valore di R^2 sembrerebbero avvalorare la correttezza di questa scelta, infatti la variabile indipendente *tBodyAccstdY* è in grado di predire circa il 94% della varianza della variabile dipendente *tBodyAccMagmean*.

	Score su Test Data
R^2	0.944
MSE	0.013
MAE	0.067

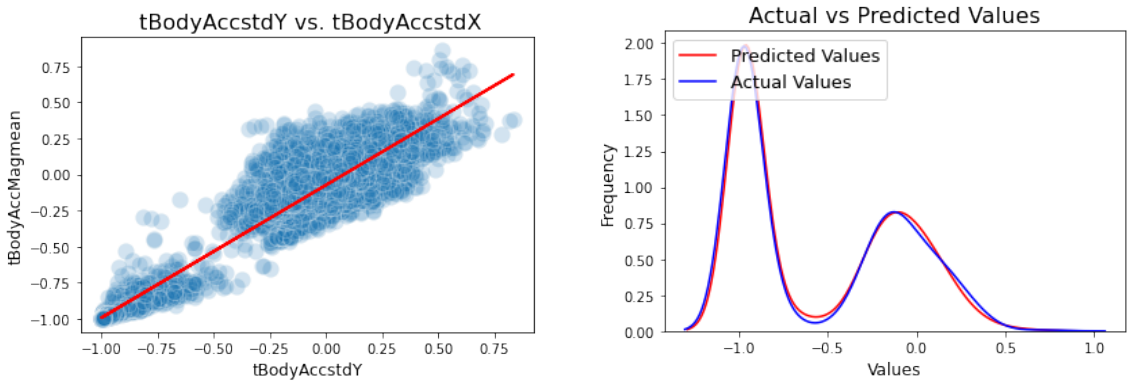
Tabella 17: Metriche ottenute sul test set



(a) **(b)**

Figura 24: Predizione variabile indipendente su quella dipendente nel SLR

Conclusasi questa fase preliminare sono stati applicati a tale modello i metodi Lasso (con e senza ottimizzazione di alpha) e Ridge (con e senza ottimizzazione di alpha). L'applicazione di tali metodi in tutte le loro configurazioni produce risultati pressoché analoghi sia confrontandoli tra di loro sia con quelli ottenuti precedentemente dal Simple Linear regression. Riportiamo quindi per completezza solamente i grafici ottenuti dal Lasso Simple Linear Regression con ottimizzazione di alpha, il cui best score si attesta al 0.94 (valore leggermente più alto rispetto agli altri).



(a) **(b)**

Figura 25: Predizione variabile indipendente su quella dipendente nel LassoSLR

3.2 Multiple Linear Regression

Allo scopo di generalizzare i risultati ottenuti con il Simple Linear Regression è stato affrontato il problema della regressione multipla. Per svolgere tale compito si è reso necessario effettuare una scelta per determinare ulteriori variabili indipendenti. A tale scopo sono state eseguite due prove molto differenti tra di loro ma basate entrambe sul R^2-score. La prova uno consisteva nel fissare come variabile dipendente il *'tBodyAccminX* e studiare quali ipotetiche variabili indipendenti avrebbero avuto una correlazione alta (positiva e negativa) con tale variabile. Con una fortissima correlazione positiva (0.9991) troviamo la variabile *tBodyAccsma* mentre *tBodyAccMagmean* risulta avere una forte correlazione negativa con *'tBodyAccminX* (-0.97). Questo approccio conduce ovviamente a un indice R^2 rasente il valore di 1. Allo scopo di abbassare tale score è stata implementata una strategia opposta: sono state scelte e aggiunte a quella già presente come variabili indipendenti le quattro che risultavano avere la correlazione positiva più bassa con il *tBodyAccMagmean*. Nel dettaglio sono state considerate: *'tGravityAccmaxX'*, *'tGravityAccmeanX'*, *'tGravityAccenergyX'*, *'tGravityAccminX'*. Ciò ha condotto a un leggero abbassamento del R^2 (ora a 0.94). Analogamente alla procedura seguita per il Simple Linear Regression anche per questo problema sono stati applicati i metodi del Lasso e del Ridge. Entrambi sono stati però testati solamente con il valore ottimale di alpha, trovato mediante l'applicazione della gridsearch (per il primo alpha= 0.0002 mentre per il secondo 0.043). In entrambi i casi però R^2 resta molto alto (0.94) e ciò può essere dedotto anche dalla coincidenza della curva dei valori predetti con quella dei valori attuali che viene riportata di seguito:

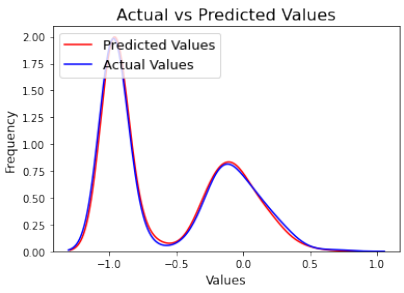


Figura 26: Predizione variabile indipendente su quella dipendente nel MLR

L'ultimo Regressor testato in questo ambito è il Gradient Boost (regressor in quanto variabile target continua) ottenendo in questo caso uno score ancora più elevato (R^2= 0.96).
Quindi è possibile asserire che il Simple Liner Regressor e il Multiple Linear Regressor ottengono in tutte le loro configurazioni score veramente alti.

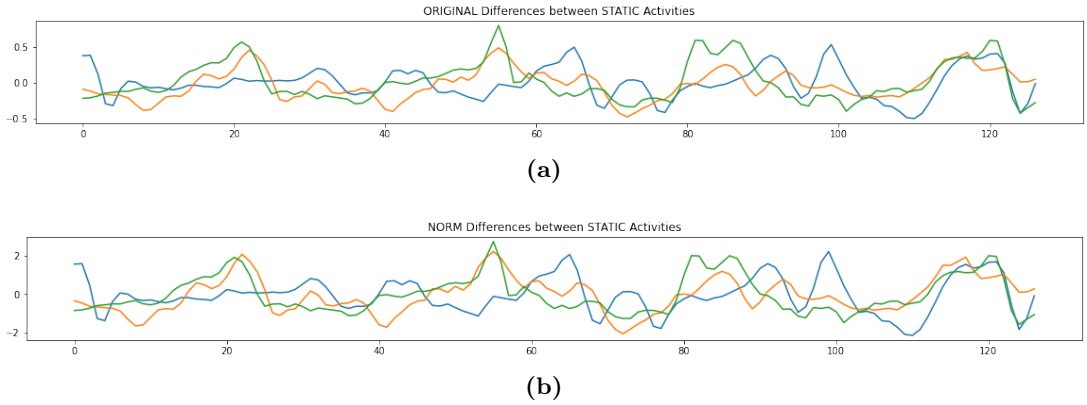
4 Time Series

In questa fase di analisi si è deciso di utilizzare come dataset sequenziale il *Body_acc_x_train* costituito da 128 timestamps e 7352 time series. Tale dataset è stato ottenuto a partire da segnali registrati tramite il

sensore accelerometro, successivamente campionanti utilizzando una finestra scorrevole di lunghezza 2.56 secondi con 50% di overlap.

4.1 Clustering approximations

Nella fase di preprocessing sono stati testati 3 diversi tipi di trasformazione: *Offset Traslation*, *Trend Removal* e *Amplitude Scaling*. Poiché questi tre tipi di trasformazioni hanno generato andamenti molto similari delle Time Series, al fine di confrontare le forme delle stesse è stato deciso di adottare l'amplitude scaling, la quale ha prodotto i seguenti grafici:

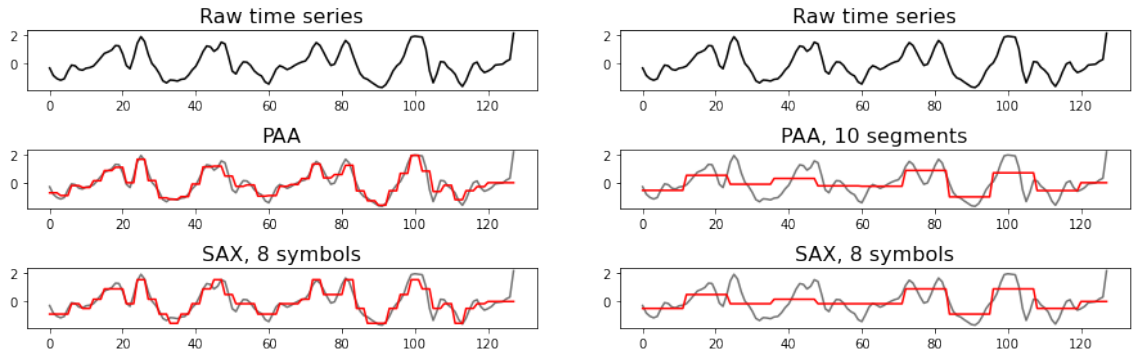


(a)

(b)

Figura 27: Time series originali (27a) e trasformate tramite amplitude scaling (27b)

Conclusasi la fase di scelta della trasformazione da adottare, è stata applicato la *Moving avarage* al fine di rimuovere i rumori presenti nel Dataset. Per quanto riguarda le Time windows sono stati effettuati vari tentativi, il migliore dei quali è risultato essere quello con time window uguale a 2 (anche considerata la scarsità di rumore presente nel dataset).



(a)

(b)

Figura 28: Time series ottenute con PAA e SAX con 40 segmenti 28a e con 20 simboli 28b

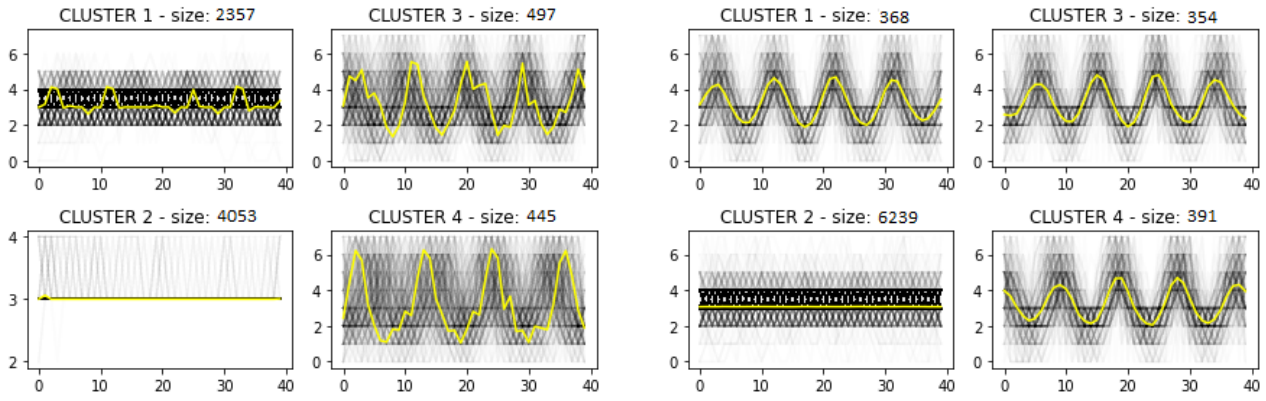
Sono stati dati come input all'algoritmo di clustering k-means tre tipi di dataset: dataset trasformato ma non approssimato (che noi chiameremo RAW), dataset trasformato e approssimato con SAX (con 8 simboli e 40 segmenti) e il dataset trasformato e approssimato con PAA (40 segmenti). Si è deciso di optare per tali parametri per SAX e PAA in quanto permettono di ottenere un giusto compromesso tra una giusta riduzione del dataset senza però perdere troppa informazione (infatti come è possibile vedere dalla Figura 28 il dataset approssimato con solo 10 segmenti porta ad ottenere una time series quasi lineare). Per il k-means su dataset PAA sono stati utilizzati vari tipi di metrica: Euclidean e DTW. Per il k-means su dataset SAX sono stati testati anche i DTW con Itakura e Sakoe-chiba. Per il k-means su RAW sono state adottate la Euclidean e il DTW. Lo scopo dell'analisi è quello di confrontare il risultato migliore per ciascun tipo di approssimazione e successivamente effettuare un confronto tra questi ultimi. Per ciascuna di queste tecniche è stato di volta in volta individuato il valore ottimale di K tramite l'SSE method e la Silhouette score. Il criterio discriminante per stabilire la bontà del clustering sta nell'osservare la distribuzione delle varie attività all'interno dei singoli clusters; in particolare sarà giudicato buono un cluster che raggruppa al

suo interno attività omogenee fra loro (che presumibilmente potranno condurre a SSE più bassa) ma allo stesso tempo maggiore eterogeneità tra attività situate in clusters differenti (quindi Silhouette più alto).

4.1.1 SAX Dataset

Partendo dai risultati raggiunti con il dataset SAX (avendo individuato come valore di K uguale a 4), sono state riscontrate alcune similarità ma anche delle differenze tra i clusters individuati utilizzando le diverse metriche. In particolare Euclidean e DTW formano clusters molto differenti tra di loro.
Partendo dall'analisi della metrica Euclidean è possibile vedere che, se si analizza il numero di records distribuiti tra i vari gruppi (Figura 29b), si ottengono clusters abbastanza omogenei, fatta eccezione per il cluster 2 che contiene 6239 records; considerando invece le metriche ottenute, si osserva un valore abbastanza alto di Silhouette Score (0.62), indice di una buona divisione tra i clusters generati. La DTW invece presenta una miglior vicinanza dei records al rispettivo centroide ma una più bassa Silhouette, inoltre i 4 clusters formati presentano una forte disomogeneità nel numero di records distribuiti nei vari clusters.

Se si analizza la distribuzione delle classi nei cluster generati, nel caso dell'Euclidean assistiamo alla presenza esclusiva di records dinamici in tre clusters, mentre in quello dominante (ovvero quello avente numero maggiore di records) sono presenti entrambi i tipi di attività con una predominanza di quelle statiche. Discorso diverso per il DTW che si dimostra meno abile nel discriminare i vari tipi di attività; infatti il cluster 4 è l'unico che presenta un solo tipo di attività (statiche) mentre gli altri risultano composti da entrambi i tipi di attività. I DTW testati con i vincoli si uniformano uno all'Euclidean (Sakoe-chiba) e l'altro se ne discosta, risultando molto simile al DTW senza vincolo. Di seguito si riporta la configurazione grafica delle situazioni appena descritte.

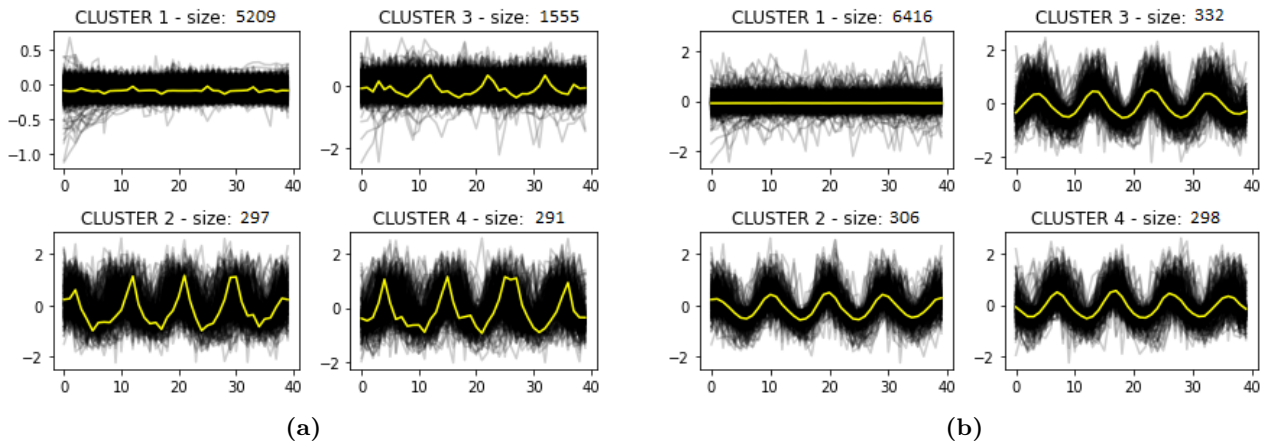


(a) **(b)**

Figura 29: Clusters individuati su dataset SAX con metrica DTW (29a) e su dataset SAX con metrica euclidean (29b)

4.1.2 PAA Dataset

Anche per questa seconda tecnica di approssimazione si è reso necessario individuare il valore ottimale di K, che risulta essere ancora una volta 4. Stavolta però sono state usate solo la Euclidean e le DTW senza vincoli.
Se si considera la metrica euclidean, si evidenzia una forte similitudine tra k-means ottenuto su dataset PAA e dataset SAX sia nei valori di SSE e Silhouette score sia per quanto riguarda la composizione dei clusters. Una differenza che rende leggermente preferibile la PAA Euclidean rispetto a quella della SAX consiste nel fatto che ciascun cluster (2, 3 e 4) avente solo attività dinamiche risulta essere costituito da una predominanza di attività differenti (Tabella 18). Per quanto riguarda la metrica DTW si riscontrano invece maggiori differenze tra PAA e SAX. In particolare nella PAA si ottiene un bassissimo valore della SSE (0.78) a parità di Silhouette score. Ad ulteriore conferma dei valori appena descritti si può riportare la presenza marcata in tutti i clusters contenenti attività dinamiche dell'attività numero tre; lì dove predominano le attività statiche (cluster 1), l'attività predominante risulta essere la numero sei (che è statica). La seguente figura rappresenta quanto appena descritto.

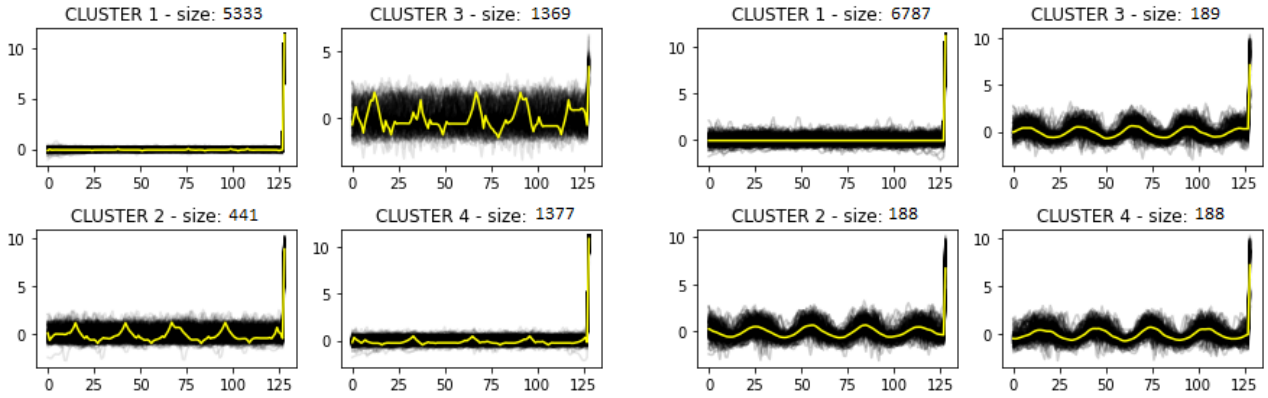


(a) **(b)**

Figura 30: Clusters individuati su dataset PAA con metrica DTW (30a) e su dataset PAA con metrica euclidean (30b)

4.1.3 RAW Dataset

Confrontando le varie metriche Euclidean sopra descritte con la RAW Euclidean, notiamo differenze negli indici ottenuti e similitudini per quel che concerne la distribuzione del numero di records all'interno dei clusters. Rispetto alla PAA Euclidean l'indice di SSE torna a salire mentre la Silhouette si mantiene stabile come valore. Più evidenze si hanno nella composizione dei clusters: con la RAW Euclidean tutti i clusters a prevalenza dinamica contengono una piccola percentuale di records riconducibile all'attività statica numero sei (situazione che non si verificava nelle due precedenti Euclidean TS Metrics). Anche in questo caso è possibile notare un cluster molto maggioritario rispetto agli altri tre, altra similitudine con la SAX Euclidean. La distribuzione descritta viene riportata nella seguente figura.



(a) **(b)**

Figura 31: Clusters individuati su dataset RAW con metrica DTW (31a) e su dataset RAW con metrica euclidean (31b)

L'ultimo confronto effettuato riguarda la DTW. In particolare nel dataset trasformato ma non approssimato (RAW) i valori di SSE e Silhouette tornano ad essere molto simili a quelle configurazioni già descritte precedentemente. Anche la composizione dei clusters a livello di numero di records si discosta dalla RAW Euclidean in quanto presenta due clusters dominanti e due minoritari aventi più o meno lo stesso numero di records. La composizione dei clusters rispetto alle attività risulta essere stavolta meno marcata e definita, poiché solamente il cluster 3 risulta essere composto da attività esclusivamente di un tipo (dinamiche). Tutti gli altri risultano essere formati da attività sia statiche che dinamiche.

4.1.4 Final Evaluation

Risulta difficile stabilire con precisione quale sia la migliore configurazione ottenuta in quanto occorrerebbe tenere in considerazione simultaneamente sia gli score ottenuti che la distribuzione delle attività nei clusters

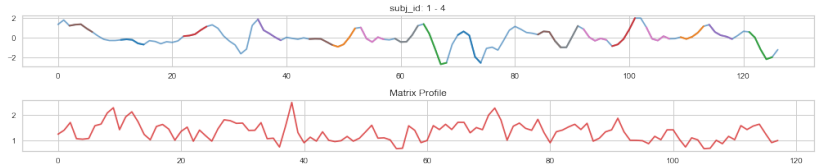
formati. Nello specifico se ci soffermassimo esclusivamente su SSE e Silhouette score saremmo portati a concludere che la PAA DTW produca i migliori risultati. Tuttavia ciò non è confermato dalla presenza del cluster quasi totalitario e dall'incapacità di raggruppare le attività statiche da parte del K-Means. Soffermandoci su quest'ultimo aspetto invece la PAA euclidean risulta essere la migliore in quanto i clusters di sole attività dinamiche che vengono generati presentano ciascuno un'attività dominante diversa. Tuttavia non riesce ad espletare il compito di discriminazione tra attività statica e dinamica.

	Attività					
	1	**2**	**3**	**4**	**5**	**6**
Cluster 1	14%	12%	10%	20%	21%	22%
Cluster 2	35%	32%	33%	0	0	0
Cluster 3	33%	34%	33%	0	0	0
Cluster 4	33%	29%	37%	0	0	0

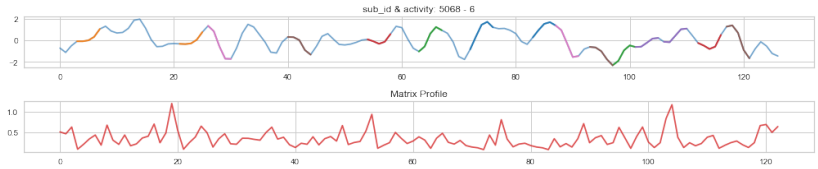
Tabella 18: Distribuzione delle attività nei clusters generati sul dataset PAA euclidean

4.2 Motifs e anomalie

Dopo aver applicato nuovamente la Moving Average al dataset trasformato come descritto nella sezione precedente, è stato deciso di verificare la presenza di Patterns all'interno delle Time Series delle cinque attività svolte da un soggetto. Nello specifico l'analisi si è focalizzata sul soggetto 1 e sul soggetto 25. Testando varie finestre per la motifs-discovery abbiamo notato che con 10 si ottenevano patterns tra loro dissimili e per questo è stata diminuita la finestra portandola a cinque, ottenendo dei significativi miglioramenti.



(a)



(b)

Figura 32: Motifs delle time series del soggetto 1 (34a) e del soggetto 25 (34b)

Questa fase ha evidenziato delle differenze sia nelle Times Series che nei Motifs relativi alle attività svolte dai due soggetti, come era prevedibile. Un ulteriore approfondimento condotto su questo argomento ha portato a risultati interessanti: qui sono state prese in considerazione le tre attività dinamiche in quanto ritenute più interessanti per l'individuazione di patterns ricorrenti. Partendo da una base di 30 soggetti, per ciascuna attività è stato individuato il soggetto che presentava una maggiore similarità tra i motifs. Visti i comportamenti molto simili tenuti dalle tre attività si riporta in dettaglio l'andamento della time series di una delle attività.

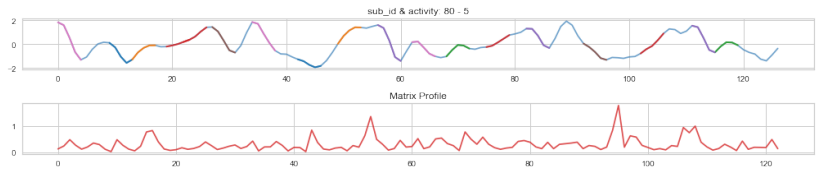
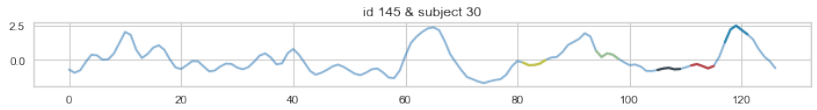
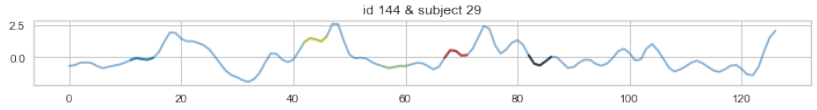


Figura 33: Motifs time series attività 2

Risultato piuttosto prevedibile è stato il non riscontrare una forte presenza di anomalie nella fase di anomaly-discovery per i due esperimenti. Ciò potrebbe essere dovuto al quotidiano svolgersi di queste attività e al fatto che si trattasse di un monitoraggio delle azioni controllato.



(a)

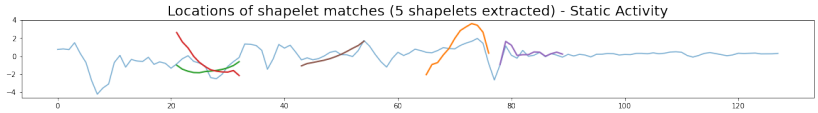


(b)

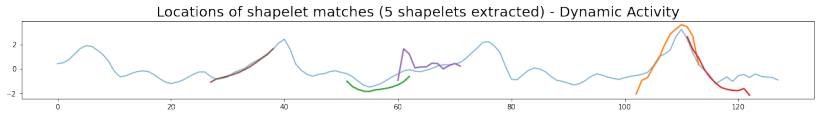
Figura 34: Anomalies delle time series dell'attività 3

4.3 Classificazione

Per lo svolgimento di quest'ultimo task non si è reso necessario effettuare lo split del dataset il quale è stato direttamente trasformato attraverso l'Amplitude scaling nella fase di preprocessing. Successivamente è stato applicato il metodo *grabocka_params_to_shapelet_size_dict* al fine di trovare la lunghezza ottimale per gli shapelets. Ovviamente si è reso necessario effettuare vari tentativi con diversi parametri; nello specifico la percentuale di lunghezza delle time series, ovvero l= 0.1, 0.01, 0.05, 0.08. Come criterio determinante per la scelta è stata considerata l'accuracy ottenuta tramite lo ShapeletModel: perciò i migliori valori risultano essere 0.1 (accuracy=0.94) e 0.08 (accuracy=0.90). Ottenuto ciò abbiamo deciso di utilizzare il parametro l=0.1 che ci ha permesso di ottenere 5 shapelets di lunghezza 12. Il modello utilizzato per l'estrazione è stato addestrato con un parametro di regolarizzazione di 0,01 per un totale di 1500 epoche. Di seguito vengono mostrati gli shapelets estratti dalle Time Series delle attività statiche e dinamiche.



(a)



(b)

Figura 35: Shapelets attività statiche 35a e dinamiche 35b

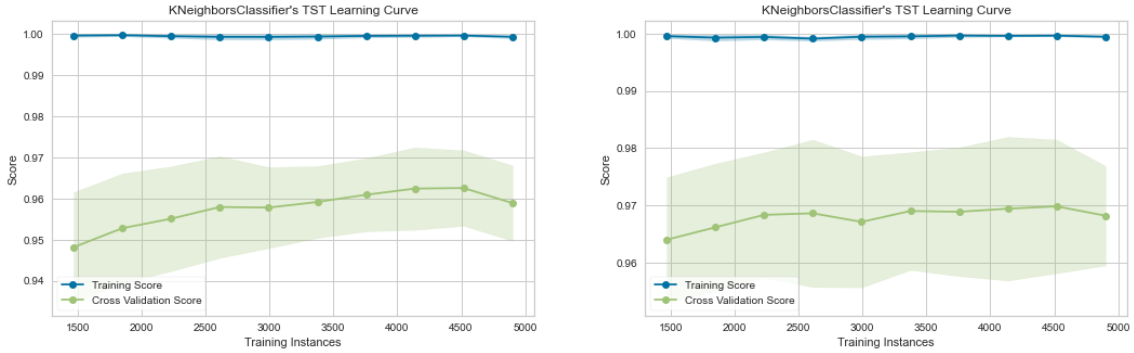
Abbiamo costruito così un nuovo dataset con gli shapelets ottenuti che sono stati utilizzati per addestrare altri due classificatori (Decision Tree e KNN). Dopo aver individuato le migliori configurazioni attraverso GridSearch per entrambi i modelli, osserviamo che il valore dell'accuracy si attesta a 0.96 in entrambi gli scenari. Non si assiste quindi a marcate differenze tra le performances ottenute dai tre classificatori.

4.3.1 KNN nelle Time Series

Il medesimo dataset visto nello studio delle Time Series è stato utilizzato anche per il KNN sempre al fine di studiare il loro comportamento. In particolare il dataset è stato binarizzato classificando come di consueto le attività in statiche e dinamiche. Terminata la breve fase di data preparation è stata avviata la fase di pre-processing nella quale grazie al TimeSeriesScalerMeanVariance abbiamo normalizzato le time series all'interno di questo dataset. Per eseguire il classificatore KNN ci siamo avvalsi di tre differenti metriche: Euclidean, Manhattan distances e DTW fast. In merito a quest'ultima, visti i lunghissimi tempi e le difficoltà di esecuzione dell'algoritmo, si è reso necessario individuare un sottoinsieme del dataset utilizzato: in particolare saranno considerate esclusivamente 200 righe per quanto riguarda X-Train e 100 time series

per X-test. Prima di entrare nel dettaglio di queste tre metriche occorre precisare che è stata svolta una fase di GridSearch al fine di individuare la migliore configurazione possibile per il KNN nei tre casi. Tale GridSearch ha riguardato il numero ottimale di vicini e i pesi (uniformi o pesati in base alla distanza).

Con riferimento alle prime due metriche osserviamo fortissime analogie e praticamente nessuna differenza sostanziale, sia in termini di confusion matrix che di comportamento che il training e il cross-validation score adottano. Entrambi infatti presentano un'accuracy e un F1-score molto alti, rasenti il 100% ma è possibile osservare nella learning curve dei due casi una piccola divergenza tra le due curve appena menzionate.



(a) **(b)**

Figura 36: Learning curve del KNN con metrica euclidean 36a e con manhattan 36b

Un discorso differente invece è quello in riferimento alla DTW fast che ricordiamo però avere meno valenza in quanto fatta su un sottoinsieme del dataset; ciò rende impossibile effettuare un confronto con le due metriche appena descritte. Ad ogni modo il KNN con tale metrica presenta anche in questo caso valori molto alti in accuracy, precision, F1-score e recall ma si osserva una differenza nel grafico della Learning Curve di seguito riportato:

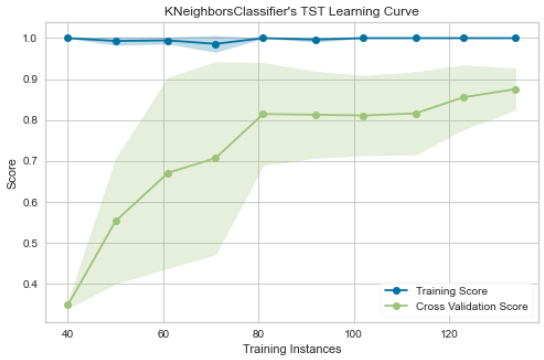
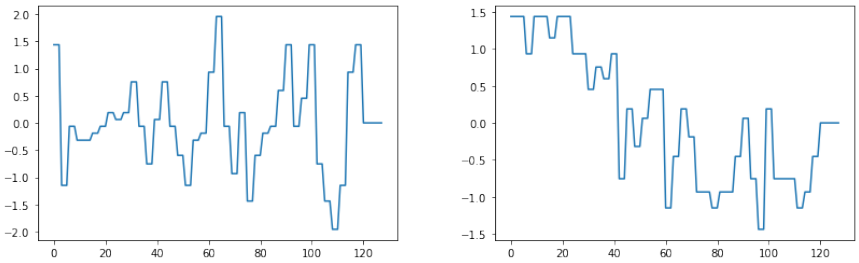


Figura 37: Learning curve del KNN con metrica dtw fast

Nonostante una forte instabilità della curva che rappresenta il cross-validation score si ha una convergenza con la curva del Training Score; fattore sicuramente positivo che non si osservava nelle altre due situazioni. Si può concludere che tutte e tre le configurazioni del KNN ottengono comunque delle ottime performances in termini di confusion matrix dimostrandosi quindi adeguate per la supervised classification dei records in questo dataset.

5 Sequential Pattern Mining

Per agevolare l'analisi abbiamo deciso di binarizzare il dataset ottenendo così attività statiche e dinamiche. A causa delle peculiarità di questo dataset, in particolare l'assenza di variabili categoriche, ai fini dello svolgimento di questo task si è reso necessario adottare la SAX approximation per trasformare le Time Series in sequenze. La configurazione scelta per tale l'approssimazione è di 40 segmenti e 20 simboli; si ottiene quindi una situazione in cui ciascuna riga diventa una sequenza di una transazione ognuna delle quali composta da 40 eventi.



(a) **(b)**

Figura 38: Time series ottenute dopo la trasformazione dell'attività dinamica 38a e statica 38b

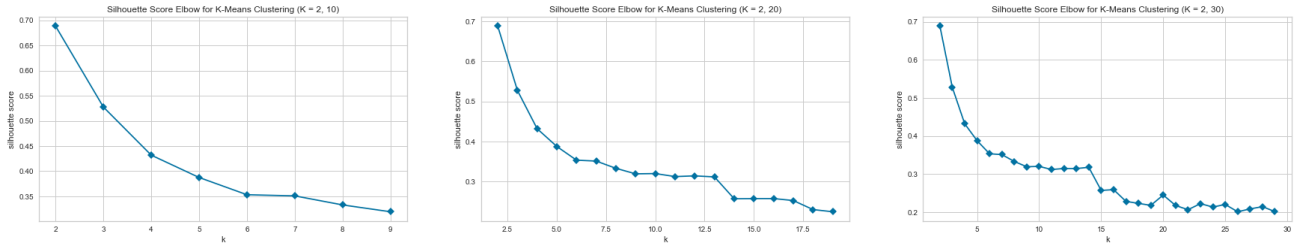
La situazione appena descritta non si dimostra molto interessante ai nostri fini, quindi abbiamo trasformato le Time series in sequenze di 20 transazioni composte di due eventi ciascuna. Lo span così ottenuto per sequenza è di 20 transazioni. La trasformazione del dataset appena descritta ci ha permesso di estrarre le sequenze frequenti prima applicando il sequantial pattern mining alle sole attività dinamiche e successivamente ripetendo la medesima analisi per le attività esclusivamente statiche. Le soglie di min_sup scelte per questi due tipi di analisi parallele sono dapprima il 14% e poi il 10%; con riferimento alle attività dinamiche entrambi questi due valori di soglia minima permettono di estrarre un numero elevato di frequent patterns. L'adozione dei medesimi valori di min support con riferimento alle sole attività statiche evidenza una forte differenza con il comportamento appena osservato con le attività dinamiche. Infatti è possibile notare come in questo secondo tipo di attività un min support del 14% generi un numero molto basso di frequent patterns. Si è reso quindi necessario abbassare la soglia al 10% ottenendo così, anche in questo contesto, un numero elevato di frequent patterns.

6 Advanced Clustering

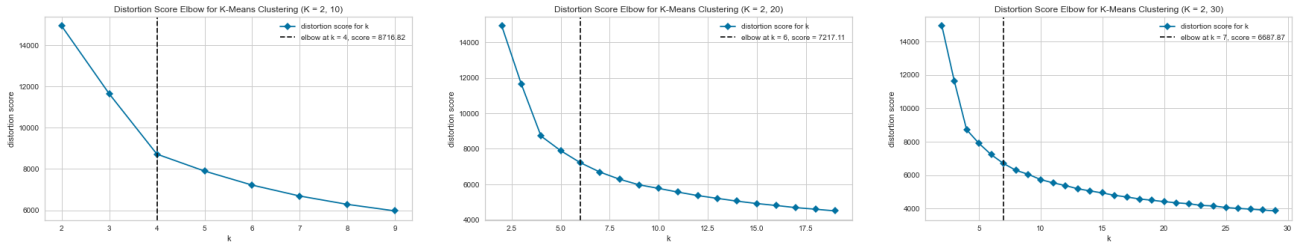
Nell'ambito dell'Advanced Clustering task è stato deciso di implementare l'algoritmo di K-Means, successivamente quello di X-Means per poi proseguire con una comparazione tra i risultati ottenuti. In prima istanza però, prima di tutte queste analisi, è stata fondamentale la fase di preprocessing la quale, partendo da una situazione in cui i valori delle features rientravano in un range da -1 a 1, mediante l'adozione del MinMaxScaler method ha permesso di far variare i valori delle features tra 0 e 1.

6.1 K-means

Come detto, è stato quindi implementato il K-Means nella sua configurazione ottimale; vale a dire: è stato individuato il valore migliore di K (numero di clusters da generare). La scelta di K si è basata su due parametri fondamentali: l'Elbow e Silhouette methods. Dalle seguenti figure 39, che mostrano l'andamento della silhouette score e del SSE score è possibile desumere come il valore di k da scegliere sia 6 (uguale al numero delle attività presenti nel dataset).



(a)



(b)

Figura 39: Plot per la scelta del k ottimale

A tutto ciò, seguono plot (Figura 40) che meglio descrivono la composizione e la natura dei clusters ottenuti ma ciò che è importante mettere in evidenza è la distribuzione che i records vanno ad assumere all'interno dei 6 clusters generati dal K-Means. Le seguenti figure mostrano visivamente tale distribuzione e la dimensione di ciascun cluster.

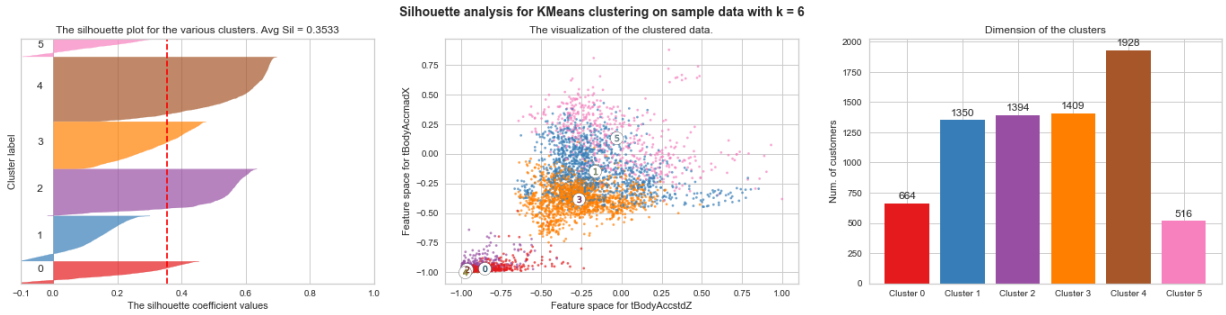


Figura 40: Analisi dei clusters ottenuti con K-means

Legato a tale distribuzione è stato ritenuto fondamentale studiare anche la composizione di ciascun cluster in termini di attività. Come si evince dalla figura seguente (41) il cluster 2 e il cluster 5 sono quelli che presentano una maggiore omogeneità tra le attività che le compongono. Il cluster 2 è formato esclusivamente dalle attività statiche 6 e 4 (fortissima prevalenza dell'attività 6) mentre nel cluster 5 (composto esclusivamente da attività dinamiche) è molto marcata la prevalenza dei records appartenenti all'attività numero 3. Occorre però precisare che in nessuno dei cluster si viene a creare una compresenza tra attività statiche e dinamiche.

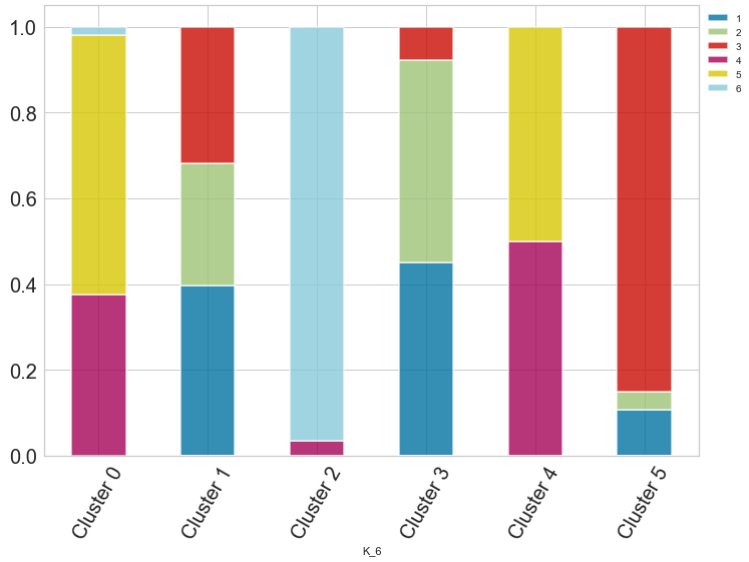


Figura 41: Distribuzione delle classi nei clusters generati da K-means

6.2 X-means

Successivamente come algoritmo di confronto è stato adottato X-Means sullo stesso dataset ottenuto in fase di preprocessing menzionata a inizio paragrafo. Nello svolgimento delle analisi sono state ricercate le condizioni per favorire la comparazione tra i risultati ottenuti dai due algoritmi; in altre parole è stato adottato lo stesso numero di k (6) e sono stati eseguiti gli stessi plot (Figura 42). Confrontando i risultati ottenuti con K-means e X-means possiamo notare che nonostante sia presente una Silhouette score e dimensione dei clusters piuttosto differenti, dal plot di visualizzazione è possibile notare una certa somiglianza nel modo in cui le due variabili sono state raggruppate. Somiglianza poi confermata dall'analisi successiva di distribuzione delle attività.

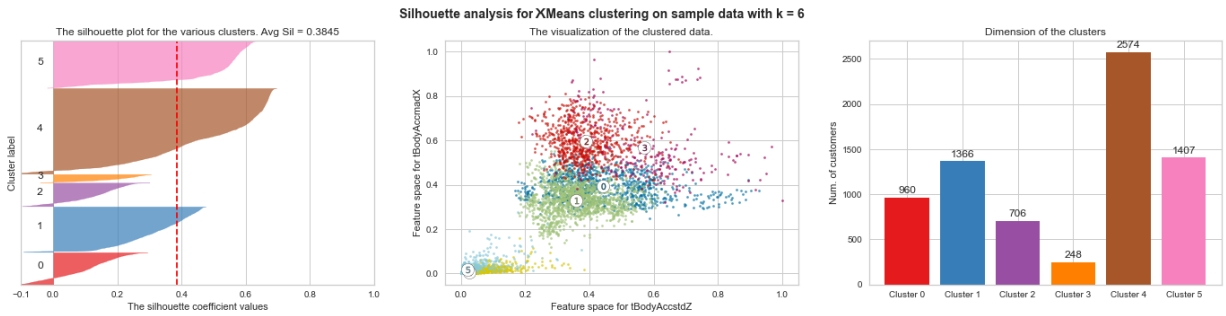


Figura 42: Analisi dei clusters ottenuti con X-means

L'oggetto principale del confronto è stata l'analisi della distribuzione delle varie attività all'interno dei clusters generati. Il altre parole è stato osservato come i due algoritmi riuscivano a discriminare le attività statiche e quelle dinamiche. La figura 43 riportata di seguito mostra come la composizione dei clusters ottenuta dall'algoritmo X-Means sia similare rispetto a quella osservata in precedenza con il K-Means. Anche in questo caso infatti all'interno del medesimo cluster non sussiste una presenza di attività statiche e dinamiche, ma al contrario si osserva una perfetta separazione dei due tipi di attività. A titolo esemplificativo riportiamo per esempio che i primi 4 clusters contengono solo attività dinamiche mentre le ultime due solo attività statiche. Unica differenza rispetto al K-Means consiste nel fatto che X-Means non riesce ad avere una sola attività statica all'interno di un singolo cluster, ma ne ha sempre due. Ripetiamo come entrambe gli algoritmi ottengano una buona distribuzione dei records all'interno dei vari clusters, con il K-Means che riesce anche ad ottenere clusters dalle dimensioni più similari tra di loro.

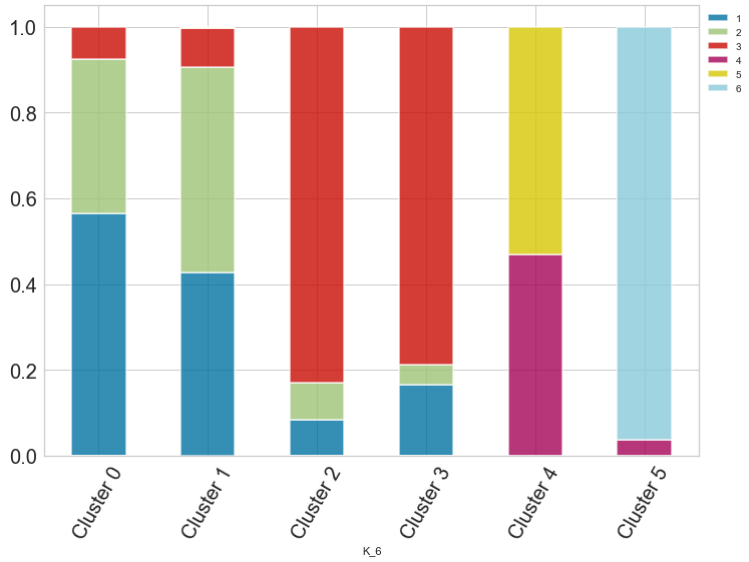
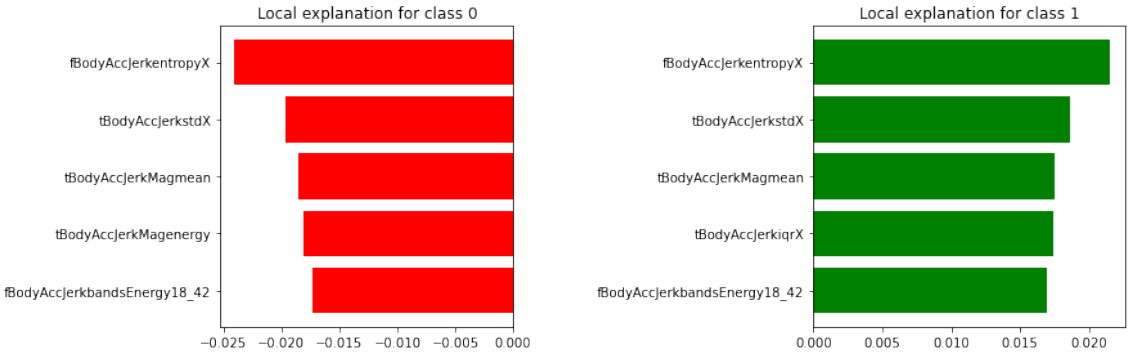


Figura 43: Distribuzione delle classi nei clusters generati da X-means

7 Explainability

Anche per lo svolgimento di quest'ultimo task è stato necessario in prima istanza binarizzare il problema distinguendo in attività dinamiche (valore 1) e attività statiche (valore 0). Il dataset così ottenuto verrà testato dal classificatore Random Forest con la seguente configurazione: max_depth = 9, min_samples_leaf = 15, min_samples_split = 5 e n_estimators = 200. I valori ottenuti di accuracy e f1-measure sono pari a 1. Il dataset così composto è stato passato all'algoritmo del Local XAIs. La procedura seguita è la seguente: selezionare un record statico e uno dinamico che il classificatore era riuscito ad attribuire correttamente alle rispettive classi di appartenenza dopodiché è stato selezionato un record che il classificatore ha predetto come dinamico ma la cui natura è quella di attività statica. Per ciascuna di questi tre records sono state studiate le features che maggiormente hanno "aiutato" il classificatore nella sua predizione. Di seguito si riportano i plot (Figura 44) relativi ai primi due records appena citati (quelli correttamente predetti). La probabilità di predizione delle classi è del 100% per i primi due records.



(a) **(b)**

Figura 44: Visualizzazione del contributo delle features nel classificare un sample statico 44a e dinamico 44b correttamente predetti

Analisi più interessanti emergono durante la classificazione del terzo record predetto come dinamico ma in realtà di natura statica. In questo contesto è stato possibile osservare una somiglianza tra i valori delle probabilità che il classificatore ha attribuito alla natura del record: 47% attività statica e 53% attività dinamica. Anche in questo caso è stato ritenuto opportuno analizzare nel dettaglio il contributo che le varie features hanno dato durante questo compito di classificazione. In particolare è stato notato come il maggior peso sia stato apportato da attività appunto dinamiche. Nella seguente figura è possibile osservare quanto appena descritto.

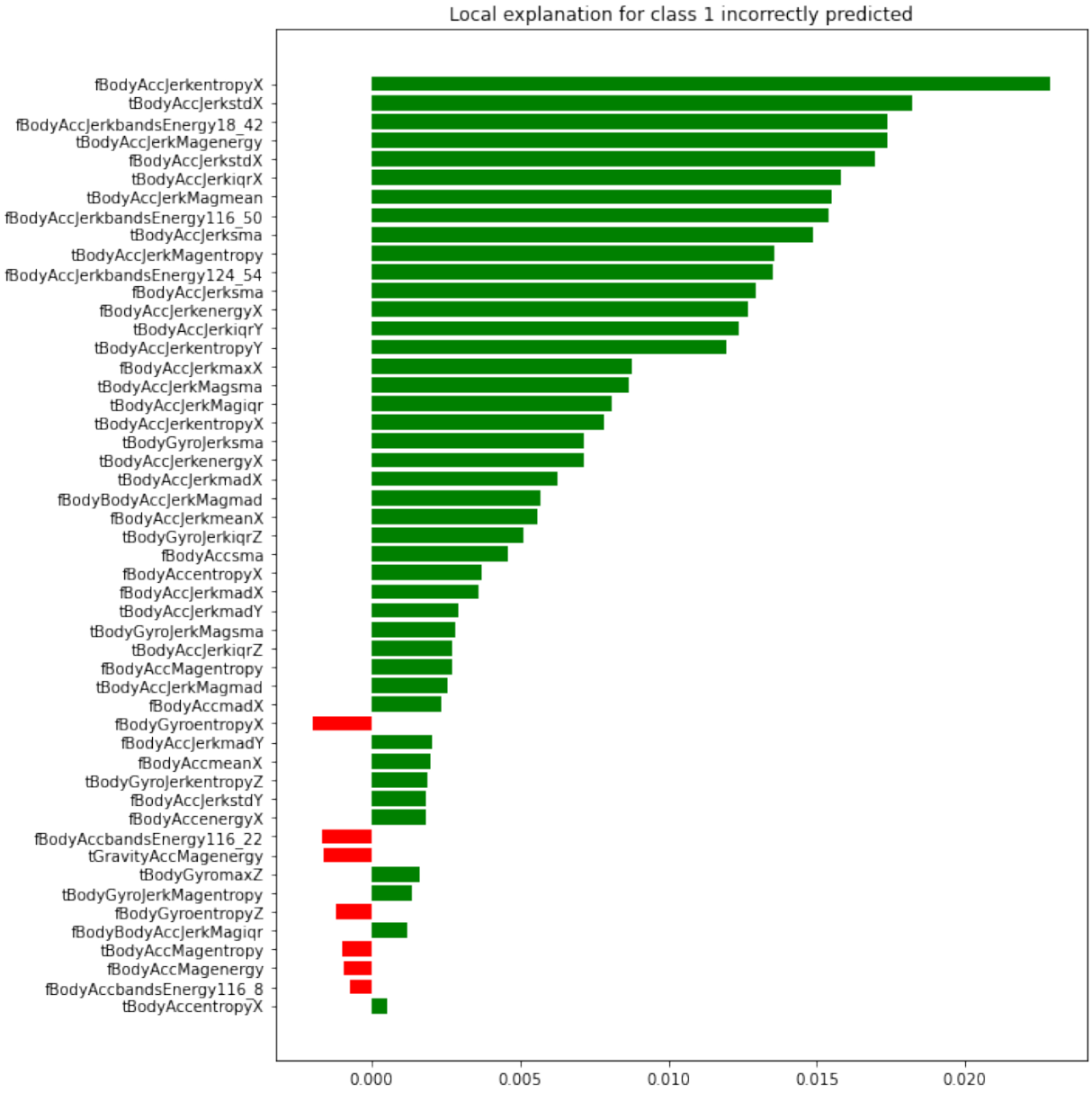


Figura 45: Visualizzazione del contributo delle features nel classificare erroenamente un record come dinamico