Report progetto di

Advanced Topics on Data Mining and Applications

Final Draft

Università di Pisa

16/04/2020

Francesco Salerno (534622)
Gianmarco Di Mauro (587959)
Marco Ciompi (537856)

1. Data Understanding

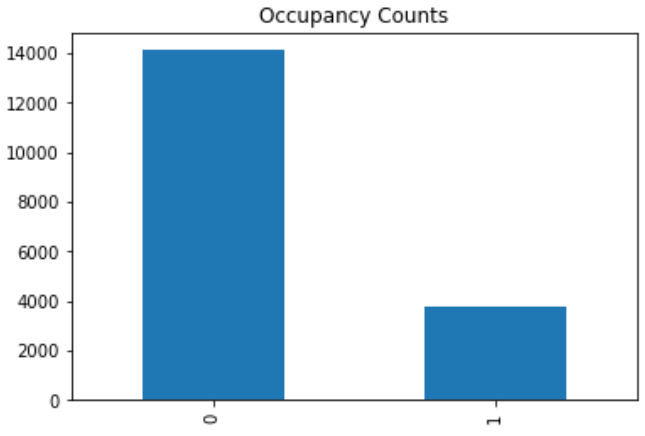
I dati utilizzati per lo sviluppo di questo progetto universitario fanno parte del dataset "*Occupancy Detection*", il cui scopo principale era quello di definire le relazioni tra la presenza di persone all'interno di una stanza e le caratteristiche ambientali di essa, quali luce, umidità, temperatura e percentuale di monossido di carbonio nell'aria. I dati sono stati prelevati nell'arco di due settimane, tra il 4 e il 18 febbraio 2015, all'interno di una stanza d'ufficio regolarmente in uso.

Il dataset si presentava inizialmente suddiviso in tre differenti file: "*datatraining*", "*datatest*" e "*datatest2*". Ai fini della nostra analisi abbiamo deciso di unire "*datatraining*" e "*datatest2*" per il training, raggiungendo così la somma di 17.895, e di utilizzare "*datatest*" come test set finale. Per comodità, d'ora in poi con il termine "dataset" indicheremo esclusivamente l'insieme dei due file utilizzati per lo sviluppo del training.
Gli attributi di cui è composto il dataset sono:

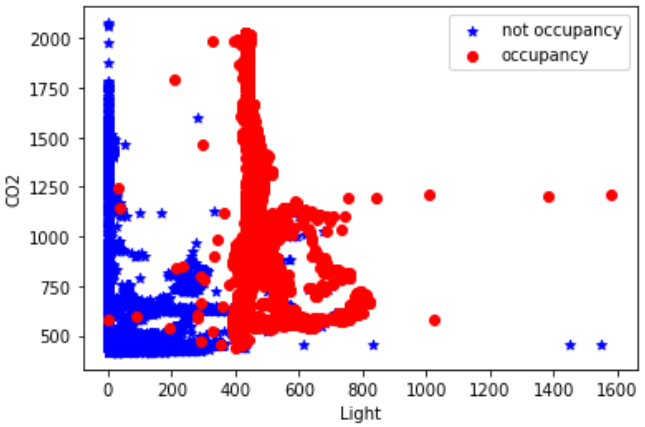
- **date** : ovvero la data del rilevamento, espressa in anno-mese-giorno ora : minuti : secondi. Attributo categorico;
- **temperature** : ovvero la temperatura espressa in gradi Celsius. Attributo continuo;
- **humidity** : ovvero la percentuale di umidità presente nella camera al momento del rilevamento. Attributo continuo;
- **light** : ovvero la luce presente nella stanza al momento del rilevamento, espressa in Lux. Attributo continuo;
- **CO2** : ovvero la quantità di anidride carbonica presente nella stanza al momento del rilevamento, espressa in ppm. Attributo continuo;
- **Humidity Ratio** : ovvero il rapporto tra umidità e temperatura. Attributo continuo;
- **occupancy** : ovvero se la stanza era occupata al momento del rilevamento. Attributo categorico binario;

1.1 Data exploration

La classe di riferimento per la nostra analisi è quindi *occupancy*. Rispettivamente ad essa il dataset risulta molto sbilanciato, con una percentuale di 79% a 21%. Si contano infatti circa 14.117 record con valore "0" di *occupancy*, cioè non sono occupate al momento del rilevamento, mentre i restanti 3778 hanno valore "1". Fortunatamente è presente una netta separazione nella distribuzione di questi valori, riscontrabile in particolar modo in relazione all'attributo *light*.



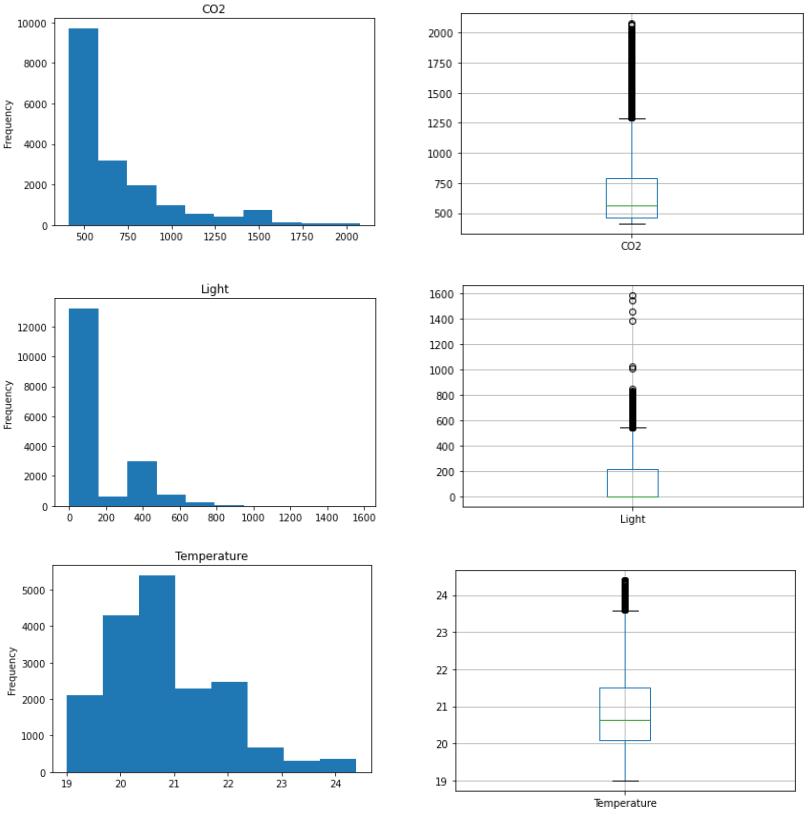
Osservando lo *scatter plot* sulla destra si evince una forte correlazione tra quantità di luce all'interno della stanza e presenza di persone: è chiaramente visibile come sotto il valore di 400 lux il numero di record con occupancy "1" - i puntini rossi - sia estremamente basso.



1.1.1 Outliers detection

Lo studio dei vari attributi presenti sul dataset ci permette anche di fare una prima stima riguardo la presenza di outliers nel dataset.

Come è possibile vedere dagli istogrammi qui presentati, gli attributi *CO2, Light* e *Temperature* non hanno una distribuzione gaussiana, ma anzi presentano una lunga coda sulla destra. Tutti e tre gli attributi presentano record con valori superiori, anche di molto, alla media. Per quanto riguarda i primi due attributi si può anche notare come più del 50% della totalità dei record presenti valori situati sull'estrema sinistra dell'istogramma, quindi valori molto bassi: ovvero 0 per *light*, cioè assoluta oscurità, e 400 per *CO2*. Attraverso l'osservazione dei dati scopriamo anche che il 100% dei record che assumono questi due valori, al momento del prelievo non erano occupate da persone - quindi occupancy 0 -.

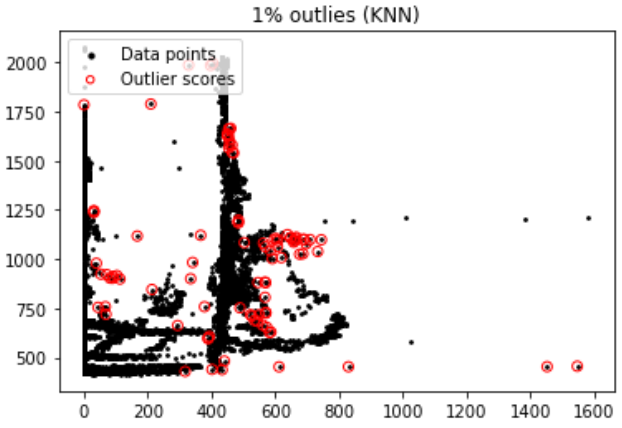


Abbiamo quindi tentato di isolare l'1% degli outliers più "marginali" utilizzando tre diversi approcci: uno basato sulla densità, LOF; uno basato sulla distanza, KNN; ed uno basato sugli angoli, ABOD. Abbiamo ottenuto i risultati migliori con i primi due, mentre il terzo è risultato poco adatto ai nostri scopi, probabilmente perché più funzionale con "*high-dimensional*" dataset.

	KNN	LOF
mean	17.20	-2.40
max	769.46	-29.84

I restanti due hanno restituito risultati più soddisfacenti, evidenziando in molti casi come outliers gli stessi record. In particolare entrambi i metodi hanno indicato come maggiormente atipi i record con bassa concentrazione di CO2 e altissima esposizione luminosa, ovvero i record individuabili a sud est negli

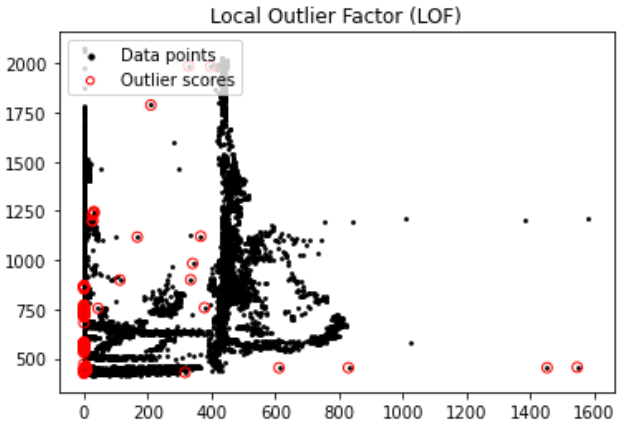
scatter plot qui mostrati. Con entrambe le metodologie questi hanno riportato i punteggi più alti.



1.2 Data Preparation

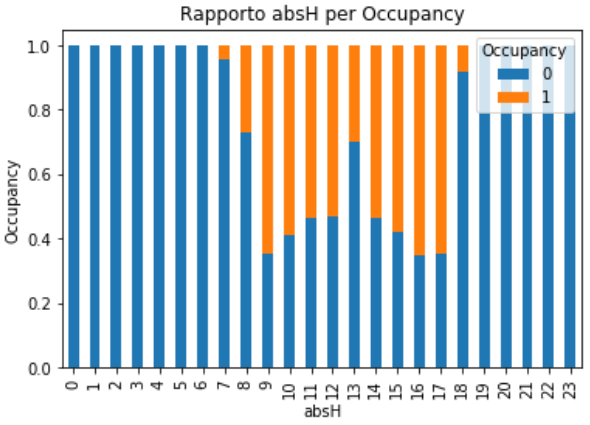
Al fine di facilitare il lavoro con l'attributo date è stato deciso di utilizzato per la creazione di altri attributi deducibili dalla data ma non utilizzabili nel formato stringa:

- **day** : ovvero il giorno in cui è stato effettuato il rilevamento
- **weekend** : ovvero se il giorno in cui è stato effettuato il rilevamento era sabato, domenica o un giorno feriale.
- **absH** : ovvero l'ora in cui è stato fatto il rilevamento
- **orario** : che in riferimento ad absH suddivide la giornata in mattina, pomeriggio, notte.



Di questi, merita particolare attenzione absH. Dall'analisi della distribuzione risulta che le stanze risultano generalmente occupate negli orari che vanno da 9 del mattino alle 19 di sera. Utilizzando questa informazione si è creato un ultimo attributo:

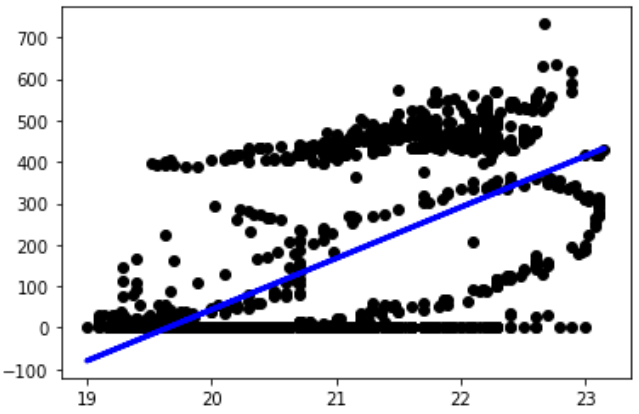
- **out_of_turn** : ovvero un attributo binario che indica se il rilevamento è stato effettuato nell'orario di massima occupazione delle camere o meno.



Humidity ratio, infine, essendo un attributo ottenuto dal rapporto tra Humidity e Temperature, risulta essere ridondante rispetto ai due "genitori", in quanto non portano nessuna informazione aggiuntiva rispetto agli altri. La correlazione tra questo e Humidity è infatti molto alta, 0.94. Per questa ragione abbiamo deciso di mantenere soltanto quest'ultimo parametro e rimuovere Humidity ratio.
Abbiamo inoltre suddiviso il dataset in training e test, con percentuale 70-30% per permettere la formulazione e la valutazione del modello.

1.3 Linear Regression

Per eseguire la regressione lineare abbiamo deciso di mantenere nel dataset esclusivamente gli attributi continui, ovvero *temperature, humidity, CO2 e light*. Come classe da predire abbiamo scelto quest'ultima. Per scegliere l'attributo ideale per la predizione di *light* abbiamo fatto riferimento alla correlazione di questo con gli altri e, con un valore di 0.67, il candidato ideale si è rivelato essere *temperature*.



$$Y = 123.5 * X + 2426.5$$

L'errore quadratico medio e l'errore assoluto medio risultano piuttosto alti, ma dato la non altissima correlazione dei due attributi ci riteniamo soddisfatti. Non si evincono sostanziali differenze tra regressione lineare, Lasso e Ridge.

	MSE	MAE
Linear Regression	20543.4	113.7
Lasso	20550.9	113.6
Ridge	20543.6	113.7

Una lieve riduzione dell'errore si ottiene utilizzando una regressione lineare multipla, sebbene non così rilevante. L'analisi non risulta soddisfacente in quanto ci aspettavamo valori molto più bassi per Lasso e Ridge, algoritmi che di fatto dovrebbero performare meglio in presenza di più variabili indipendenti.

	MSE	MAE
Linear Regression	14984.9	92.7
Lasso	14998.0	92.6
Ridge	14985.0	92.7

2. Classificazione

La task di classificazione è stata svolta sul dataset descritto nel primo capitolo, adeguatamente modificato come appunto visto in precedenza. La classe di riferimento della task è "occupancy", quindi l'obiettivo finale è identificare quali record descrivono l'ufficio in presenza di persone e quali no.

2.1 Basic Classifiers

2.1.1 KNN

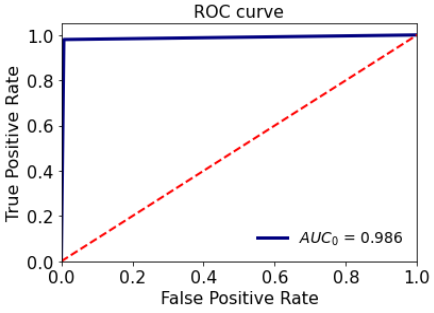
Nella fase preliminare trasformiamo gli attributi categorici in attributi binari, così da facilitare il funzionamento dell'algoritmo.
Eseguiamo dunque l'algoritmo con diverse combinazioni di parametri:

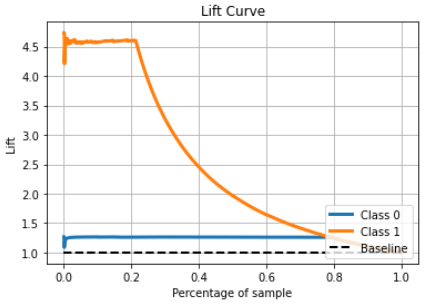
- K: { 1, 10, 50, 130, 250, 500}
  Come punto di partenza per la selezione dei diversi valori di K si è scelto la radice quadrata di N, ovvero 134. Gli altri valori sono stati selezionati per studiare il comportamento dell'algoritmo all'aumento e al diminuire di K.
- Weight: {uniform, distance}
  Si è studiato l'algoritmo sia con peso uniforme per ogni record, sia con valorizzazione dei record più vicini.
- p: {1, 2}
  Rispettivamente distanza manhattan ed euclidea.
- Metrica: Minkowski, grazie alla quale non è stato necessario fare uno scaling dei dati.

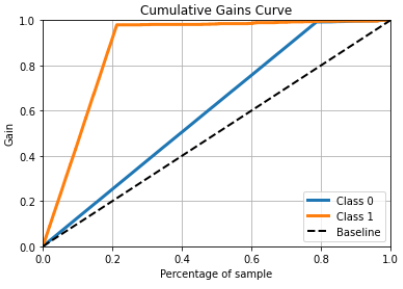
I risultati più alti sono stati ottenuti con K= 130 e distanza manhattan. Abbiamo tuttavia optato come risultato ottimale quello ottenuto con K= 1 e distanza euclidea.

	Accuracy	F1_score	
K=130	0.993	0.995	0.983
K=1	0.990	0.994	0.976

La cross validation ha riportato un valore di accuracy di 0.989 +/- 0.001.



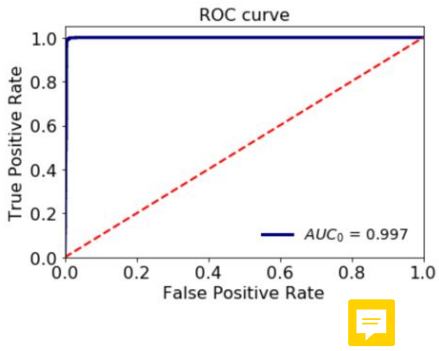


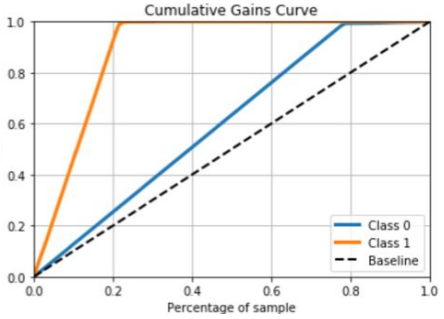


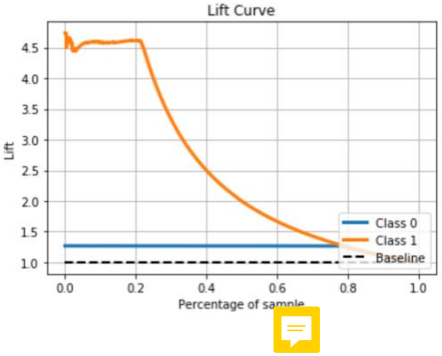
2.1.2 Naive Bayes

Assumendo una distribuzione gaussiana, abbiamo provato ad eseguire l'algoritmo sul nostro training, rimuovendo preventivamente gli attributi legati alla componente temporale, non considerabili come attributi continui e quindi non ottimali per l'algoritmo.
I risultati ottenuto sono estremamente positivi, con accuracy 0.98 e F1-score di 0.99 e 0.96 per i due valori di occupancy.

	Precision	Recall	F1-score	Support
0	1.00	0.98	0.99	2823
1	0.93	1.00	0.97	756



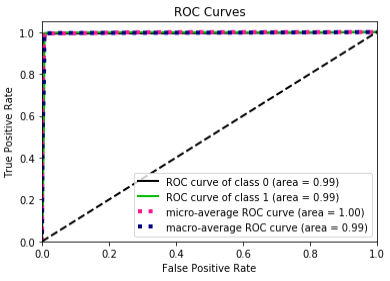


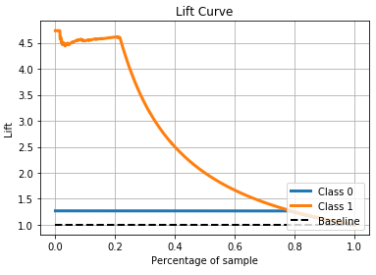


2.1.3. Logistic Regression

Essendo la nostra classe di riferimento un attributo binario, è possibile utilizzare una regressione logistica ai fini della classificazione. L'analisi è stata portata avanti sulla maggior parte degli attributi. Tuttavia di seguito si riportano i due tramite i quali abbiamo riscontrato i risultati migliori, ovvero *Light* e *CO2*.
Con il primo abbiamo ottenuto un valore di accuracy di 0.99 ed F1-score di 0.99 e 0.97:

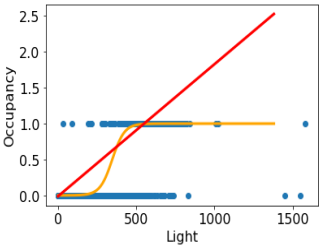
	Precision	Recall	F1-score	Support
0	1.00	0.99	0.99	4235
1	0.95	1.00	0.97	1134

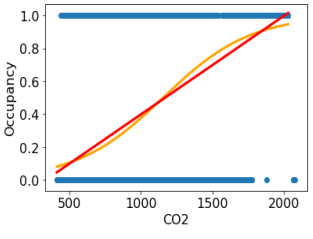




Con il secondo invece accuracy 0.77 ed F1-score di 0.87 e 0.27.

	Precision	Recall	F1-score	Support
0	0.81	0.93	0.87	4235
1	0.42	0.19	0.27	1134

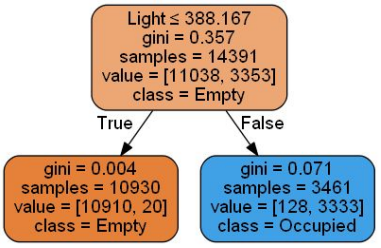




Si può notare come, tenendo in considerazione *CO2,* l'algoritmo faccia più difficoltà a riconoscere la classe minoritaria, ovvero "1".

2.1.4 Decision Tree

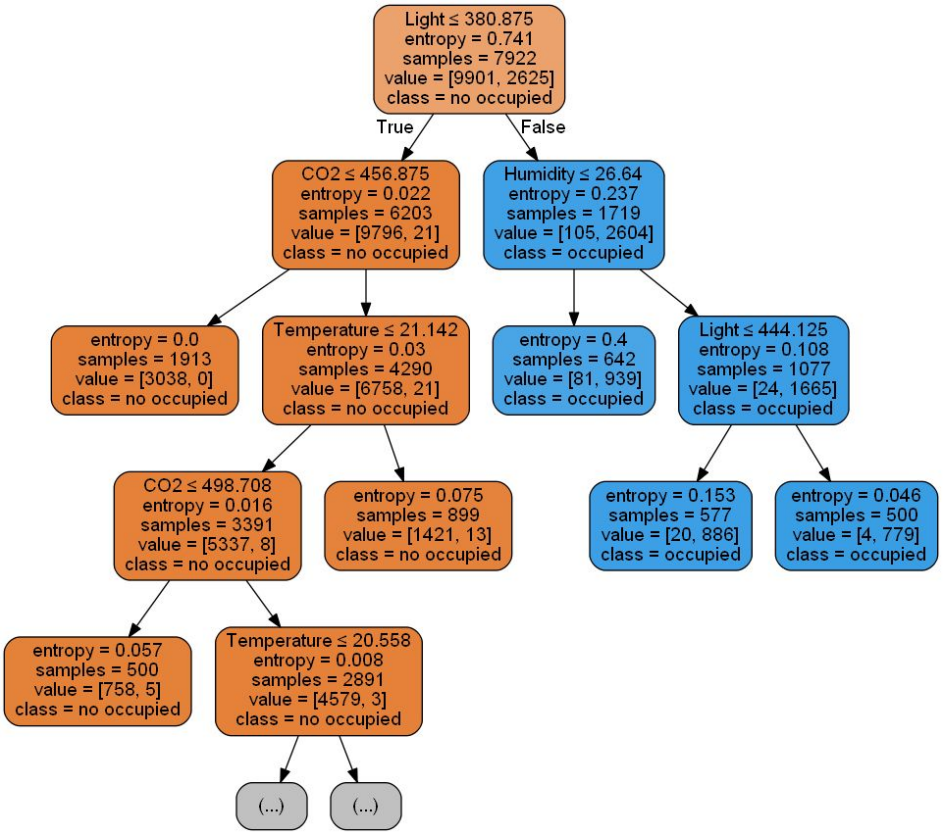
Come già visto in sede di data understanding, una separazione molto netta tra i valori con *occupancy* "0" e quelli con "1" è già possibile utilizzando esclusivamente l'attributo *light:* tutti i record con light inferiore a 400, quindi con la stanza in condizione di luce scarsa se non di buio assoluto, hanno occupancy "0", indicano cioè una stanza vuota. Sarebbe quindi possibile ottenere un risultato eccellente con accuracy 99% semplicemente con un decision stump basato su questo attributo, come mostrato in figura.



Abbiamo tuttavia deciso, al fine di studiare meglio la composizione del dataset e la possibilità di altre regole decisionali, eseguire l'algoritmo anche con i seguenti parametri:

- min_sample_split: {2,5,10,20,30,50,100}
  ovvero il numero minimo di record necessari per eseguire uno split.
- min_sample_leaf: {1, 5, 10, 20, 30, 50, 100}
  ovvero il numero minimo di record per un nodo foglia.

Non abbiamo riscontrato sostanziali differenze a livello di risultati, tuttavia è interessante osservare come gli altri attributi contribuiscono alla classificazione: in particolare il secondo attributo più importante risulta essere *CO2,* in quanto la maggioranza dei record che hanno un valore di *CO2* inferiore alla media risultano essere non occupati.

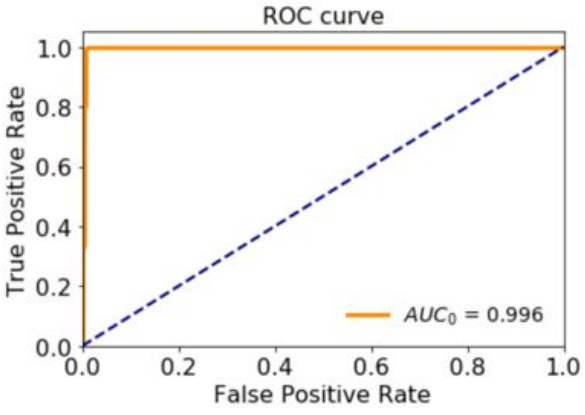


2.2 Advanced Classifiers

2.2.1 Support Vector Machine

Come già evidenziato, in relazione alla classe di riferimento il dataset risulta facilmente separabile. Questa caratteristica lo rende perfetto per l'utilizzo dell'algoritmo SVM, il cui scopo è appunto quello di individuare un iperpiano che massimizzi di divisione tra i due valori della classe di riferimento.
Come azione preliminare abbiamo svolto la Principal Component Analysis in modo da individuare i due principali attributi che maggiormente descrivono il dataset, e poi abbiamo eseguito l'algoritmo con diversi valori di C, ovvero il parametro che regola lo spessore dei margini.
I risultati ottenuti con SVM lineare sono stati tutti ottimi, a prescindere dal valore di C. Per questo abbiamo scelto come ottimale i risultati ottenuti con C=1, che massimizza la grandezza dei margini.

	Precision	Recall	F1-score	Support
0	1.00	0.99	0.99	4235
1	0.95	1.00	0.97	1134



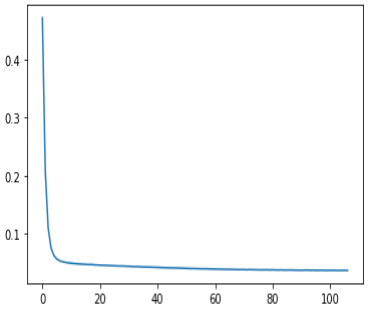
Abbiamo eseguito anche l'algoritmo SVM non lineare, ma i risultati sono stati inferiori rispetto a quelli ottenuti con la lineare proprio per la naturale struttura del dataset divisibile linearmente.

2.2.2 Neural Network

Dopo la normalizzazione del dataset tramite StandardScaler si è iniziato quindi con l'importare la funzione MLPClassifier dalla libreria sklearn.neural_network. Da ciò con parametri si sono ottenuti i seguenti risulati. E la seguente loss curve.

Accuracy 0.9923635686347551
F1-score [0.9951416 0.98216616]

	precision	recall	f1-score	support
0	1.00	0.99	1.00	4235
1	0.97	1.00	0.98	1134
accuracy			0.99	5369
macro avg	0.98	0.99	0.99	5369
weighted avg	0.99	0.99	0.99	5369

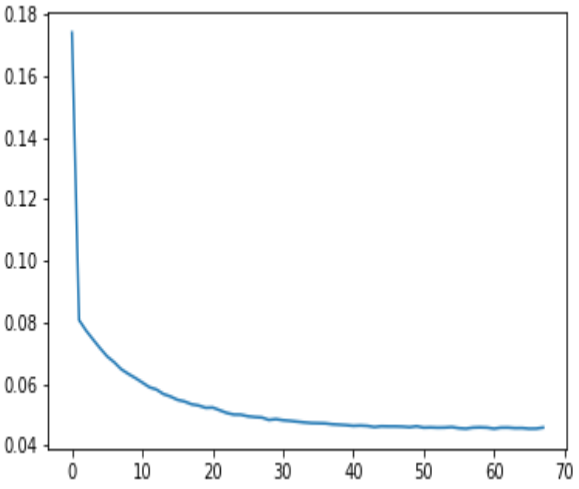


Inoltre è stata applicata la stessa funzione questa volta specificando i seguenti parametri:
hidden_layer_sizes=(128, 64, 32,), alpha=0.1, learning_rate= adaptive, activation = than, early_stopping = False , momentum=0.8, random_state=0)
ottenendo i seguenti risultati:

Accuracy 0.9923635686347551

F1-score [0.99514045 0.98218166]

	precision	recall	f1-score	support
0	1.00	0.99	1.00	4235
1	0.97	1.00	0.98	1134
accuracy			0.99	5369
macro avg	0.98	0.99	0.99	5369
weighted avg	0.99	0.99	0.99	5369

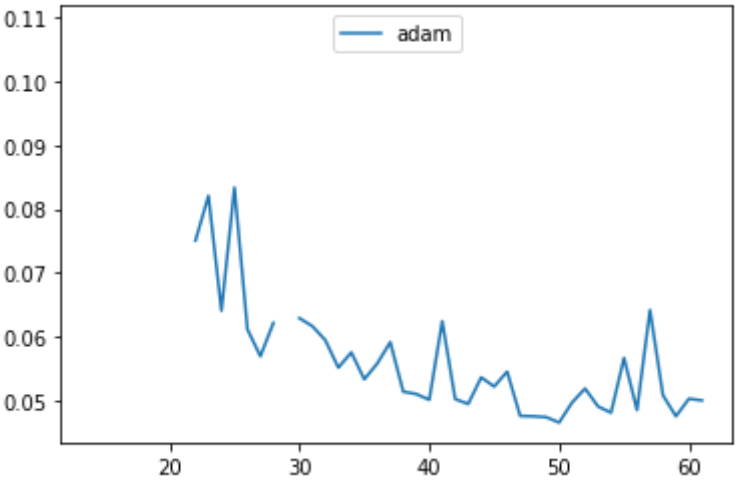


In seguito abbiamo provato nuovi solver con diverse combinazione di parametri: riportiamo graficamente il risultato migliore

adam

training set score and loss: 0.991, 0.050019

2.2.3 KERAS (Deep Neural Network)



Tramite l'utilizzo delle librerie keras.model e keras.layer abbiamo importato le funzioni Sequential e Dense. Abbiamo utilizzato due diverse architetture nel definire la funzione build mode. Il primo con parametri (128,64,1) e il secondo con parametri (35,17,1).

Nella creazione dei modelli abbiamo inoltre abbiamo effettuato diversi tentativi di cui si riportano i risultati (relativi alla sola prima architetture).

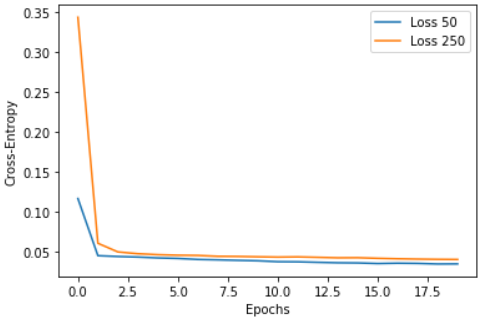
Ad esempio il modello 1 è stato allenato impostando come numero di epoche pari a 20 e batch uguale a 50.

Questi valori come quelli successivi sono stati involontariamente imposti da limiti computazionali dovuti alla non elevata efficienza dei nostri strumenti.

Il secondo modello è stato allenato invece su parametri epoche uguale a 20 e batch uguale a 250. Di seguito vengono riportati le relazioni ottenute graficamente e numericamente.

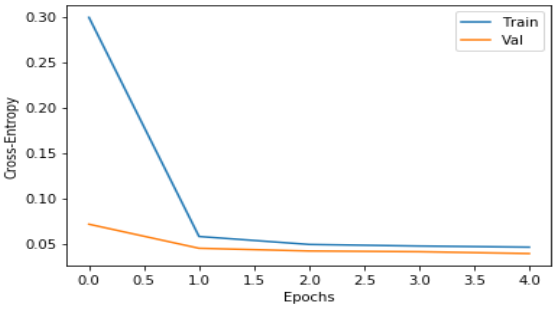
5369/5369 [==============================] - 34s 6ms/step

5369/5369 [==============================] - 40s 8ms/step
Loss 0.031679, Accuracy 0.992177
Loss 0.034968, Accuracy 0.991991



Inoltre, oltre ai risultati cumulativi riportiamo le i dati relativi alle ultime 3 epoche di ogni modello

Modello1:
Epoch 18/20
12526/12526 - 365s 29ms/step - loss: 0.0358 - accuracy: 0.9901
Epoch 19/20
12526/12526 331s 26ms/step - loss: 0.0350 - accuracy: 0.9904
Epoch 20/20
12526/12526 - 328s 26ms/step - loss: 0.0351 - accuracy: 0.9899



Modello 2
Epoch 18/20
12526/12526- 178s 14ms/step - loss: 0.0411 - accuracy: 0.9893
Epoch 19/20
12526/12526 - 182s 15ms/step - loss: 0.0408 - accuracy: 0.9893
Epoch 20/20
12526/12526 - 182s 15ms/step - loss: 0.0407 - accuracy: 0.9893

2.2.4 Ensamble

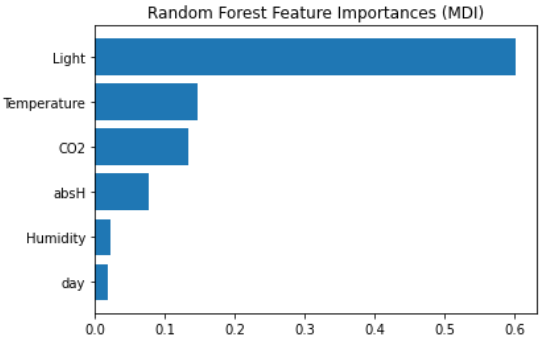
Come già visto il dataset risponde molto bene al decision tree, essendo facilmente separabile. A maggior ragione, risultati ottimali e riconducibili a quanto già detto su questo algoritmo si sono ottenuti con la random forest. Questa è stata eseguita con diverse configurazioni:

- max features = {“sqrt(n)” “log(n)”}
- criterion = { “gini” “entropy”}
- min samples leaf= {1, 500}
  ovvero con pruning o senza pruning

Inoltre tutte le iterazioni sono state eseguite con 100 estimatori.
In generale non abbiamo riscontrato nessuna variazione sostanziale nei valori di accuracy e F1 score ottenuti con i vari run, se non nell’ordine di variazioni di +/- 0.001.
Riportiamo in tabella i risultati ottenuti con “sqrt(n)” e entropy, con e senza pruning.

	Accuracy	F1_score	
Pruning	0.992	0.995	0.980
No Pruning	0.995	0.997	0.987

Come già visto, gli attributi più interessanti alla fine della costruzione dei decision tree sono stati light, CO2 e temperature.
Utilizzando **AdaBoost** con decision tree e random forest come base estimator, otteniamo risultati simili. Con la cross validation otteniamo un valore di 0.99 +/- 0.001.



	Accuracy	F1_score	
Decision tree	0.994	0.996	0.986
Random forest	0.994	0.996	0.986

Riportiamo anche i valori ottenuti con **Bagging** utilizzando come base estimator SVC, Random Forest e KNN.

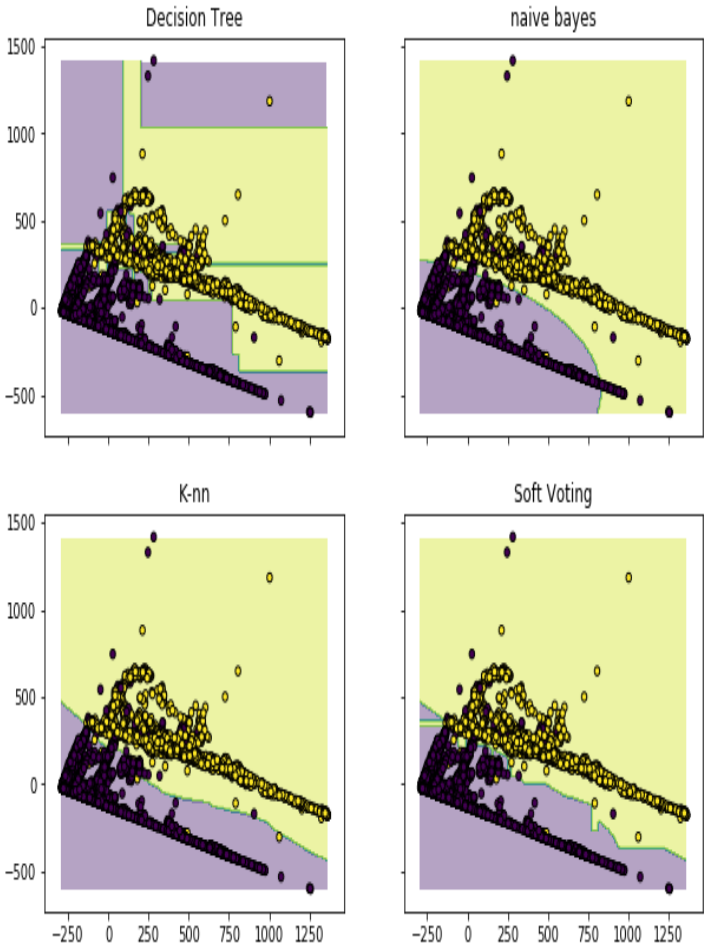
	Accuracy	F1_score	
SVC	0.993	0.995	0.982
Random forest	0.994	0.997	0.988
KNN	0.993	0.995	0.983

2.3 Dimensionality Reduction

Oltre a testare vari algoritmi di classificazione sul dataset iniziale, abbiamo provato ad eseguire sul dataset una "dimensionality reduction", ovvero una riduzione degli attributi, ed eseguire la classificazione su questo dataset modificato. Per applicare la riduzione ci siamo affidati a quattro diverse metodologie: *variance threshold; Univariate features selection; Recursive Features Elimination, PCA*.
La prima consiste nell'eliminare gli attributi con varianza inferiore ad un certo "margine" da noi deciso. Dopo varie prove abbiamo deciso di impostare il margine a 50: questo ha permesso di rimuovere gli attributi day, absH e Humidity. *Univariate features selection* invece assegna ad ogni attributo un valore rispettivamente al suo ANOVA F-value. I tre attributi ad avere un punteggio maggiore sono stati:

1. **Light**: 59909
2. **Temperature**: 4516
3. **CO2**: 3394

Anche in questo caso i risultati della classificazione sono stati molto buoni.
Per quanto riguarda *Recursive Feature Elimination,* abbiamo deciso di eseguirlo in relazione al Decision tree. Come sospettavamo già, questa modalità ha eliminato tutti gli attributi eccetto *Light,* sufficiente da solo per svolgere la classificazione.
La PCA, infine, ha selezionato i due attributi che maggiormente descrivono il dataset, ovvero *Light e CO2*. Sul dataset composto da questi due soli attributi abbiamo svolto diversi classificatori. I grafici a destra propongo in maniera visiva il decision boundary dei vari algoritmi: il colore giallo indica occupancy "1", il viola "0".



	Accuracy	F1_score	
Variance Threshold	0.993	0.995	0.982
Univariate Feature Selection	0.993	0.996	0.984
Recursive Features elimination	0.988	0.992	0.973

2.4 Imbalanced Version

Oltre a testare il dimensionality reduction, abbiamo provato anche a svolgere degli esercizi di classificazione su un dataset altamente sbilanciato. Come visto in fase di understanding, il nostro dataset iniziale presenta una percentuale di 80% a 20%, abbiamo deciso di portarla a 96% a 4% rimuovendo casualmente alcuni elementi della classe minoritaria. Dopo di che, abbiamo eseguito nuovamente alcuni classificatori basilari e avanzati.

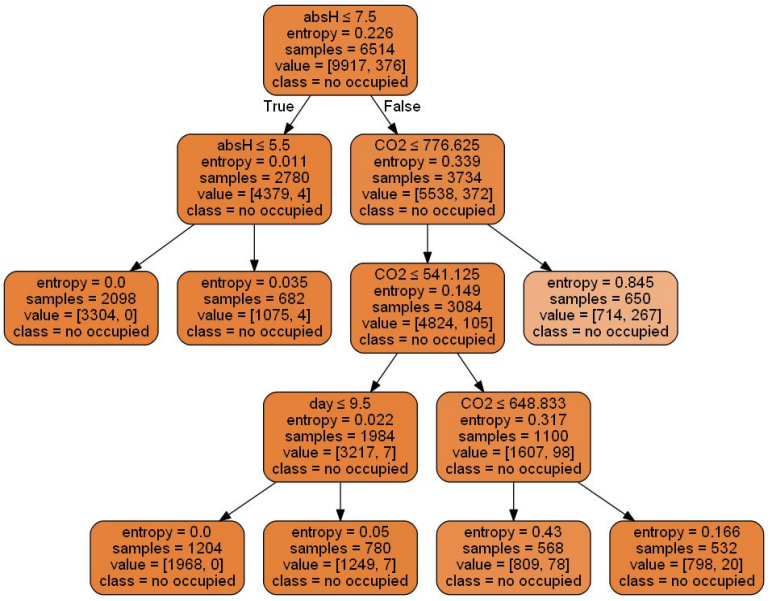
2.4.1 Basic classifiers

I risultati ottenuti con i classificatori basilari sono rimasti molto alti, nonostante il dataset fosse altamente sbilanciato. Con Decision Tree e KNN abbiamo ottenuto una accuracy del 99%: possiamo dire che i due algoritmi non hanno risentito delle modifiche apportate al dataset, in quanto non ha influito sulla

distribuzione dei record che sono rimasti nettamente separabili e quindi distinguibili sia in considerazione al loro vicinato sia in considerazione dell'attributo *Light*.
Diverso il caso del Naive Bayes, che ha restituito un valore di accuracy di 96%, non considerabile positivo in quanto sarebbe il risultato che si otterrebbe assegnando ad ogni record del test valore 0. Siamo riusciti a migliorare le sue prestazioni aumentando il "decision threshold" a 0.8 (97% di accuracy) e con un CNN (98% di accuracy).

2.4.2 Advanced classifiers

Sul dataset sbilanciato abbiamo eseguito anche il Random Forest e SVM (lineare e non lineare). Per ogni classificatore abbiamo eseguito Random Undersampling, CNN, Random Oversampling, SMOTE, abbiamo provato a modificare il "class weight" e il "decision threshold".
Con Random Forest abbiamo inizialmente ottenuto un valore di accuracy di 96.01 che, come visto prima, non può essere considerato positivo, in quanto l'algoritmo ha assegnato il valore maggioritario ad ogni record del test. Abbiamo ottenuto degli interessanti miglioramenti con tutte le tecniche elencate sopra, e in particolar modo con SMOTE:



Accuracy	0.99
F1-score	1 , 0.91

Con SVM lineare abbiamo invece ottenuto ottimi valori già dall'inizio: 98% di accuracy e 0.99 e 0.83 di F1-SCORE. Non abbiamo riscontrato nessun miglioramente con le strategie pensate appositamente per gestire i dataset sbilanciati.
Con SVM non lineare, invece, abbiamo ottenuto un risultato simile a quello del Random Forest. Anche in questo caso SMOTE ha restituito i risultati migliori, mentre il random undersampling è stato il peggiore.

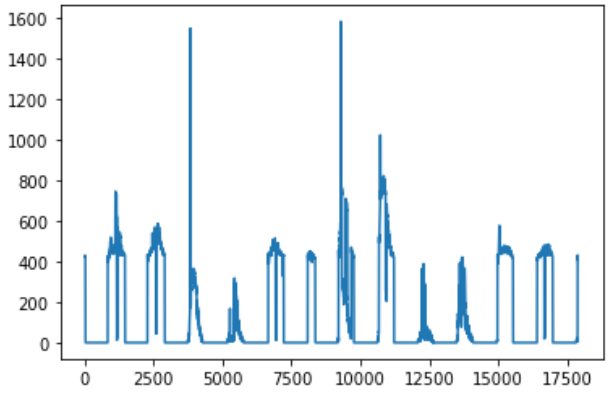
Accuracy	0.98
F1-score	0.99 , 0.71

3. Time Series

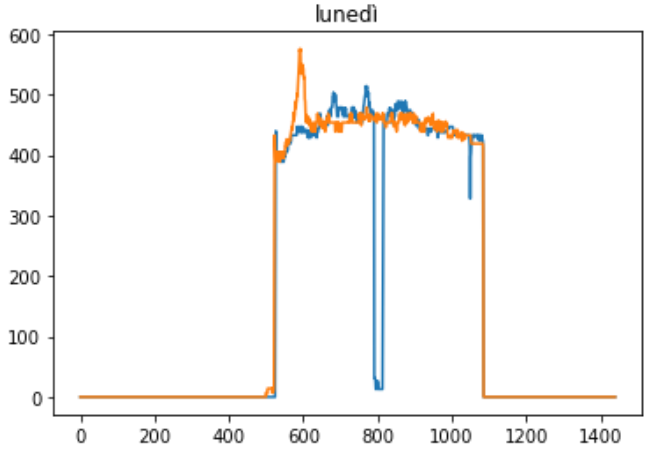
Come già evidenziato, il dataset si compone di diversi record prelevati nell'arco di due settimane in un ufficio regolarmente utilizzato a scopi lavorativi. Ogni giorno sono stati raccolti circa 1400 record distribuiti regolarmente nell'arco delle 24 ore. A questa schema fanno eccezione 4 giorni, in cui ne sono stati raccolti circa 600: il primo, mercoledì 4 Febbraio; l'ultimo, mercoledì 18 Febbraio; e i due giorni 10 e 11 febbraio, rispettivamente martedì e mercoledì. Al fine di studiare i record secondo uno schema temporale, questi 4 giorni sono stati generalmente esclusi.

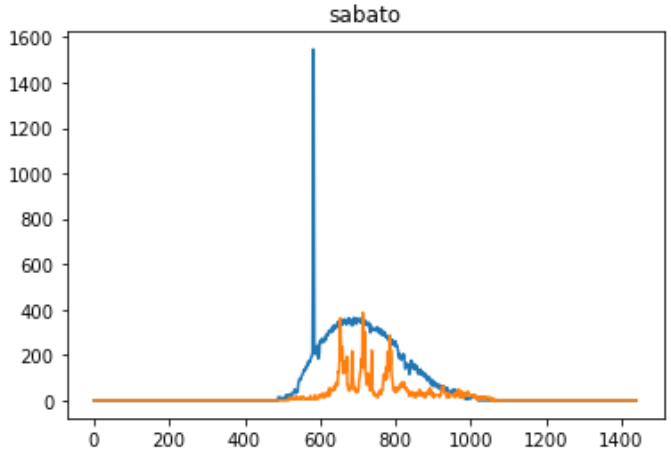
Abbiamo deciso di concentrare la nostra ricerca sui due attributi *light* e *temperature,* ma occasionalmente abbiamo confrontato i risultati anche con l'attributo *CO2.*

Per quanto riguarda l'attributo *Light,* si può osservare come in generale questo abbia un andamento regolare e ripetitivo, che segue ciclicamente l'alternanza giorno notte: le linee orizzontali alla base del grafico rappresentano le notti - il valore di *light* è 0 - mentre durante il giorno le linee si alzano a formare delle forme abbastanza similari.

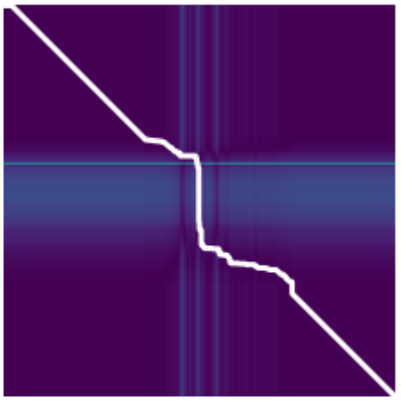


Mentre non è osservabile un **trend** delle linee, è osservabile una certa **stagionalità**: alle forme più squadrate corrispondono i giorni di lavoro settimanale; mentre a quelle meno definite i giorni di riposo settimanale.



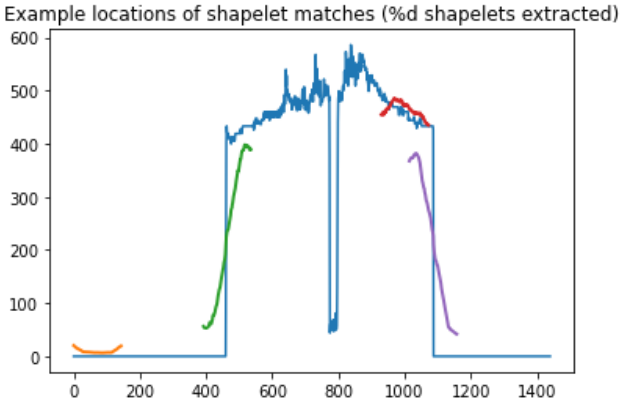


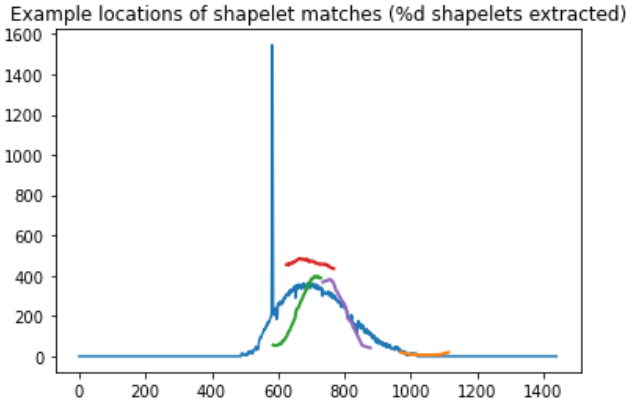
Seguendo questa divisione temporale - cioè confrontando giorni della settimana uguali - le curve risultano sorprendentemente coincidenti, rendendo quindi poco utili i nostri tentativi di **offset translation**, **amplitude scaling** e **noise smoothing**. Anche le distanze risultano relativamente brevi, specialmente utilizzando il Dynamic Time Warping, che permette una comparazione non lineare (a destra DTW delle due TS relative al sabato).



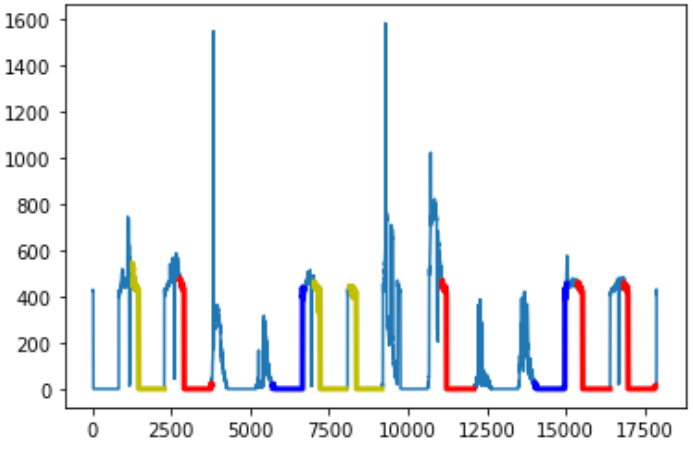
	Lunedì	Sabato
Euclidea	2345	4401
DTW	1925	1892

Sono dunque abbastanza evidenti per queste due diverse tipologie di curve i rispettivi **shaplets**, ovvero quelle forse distintive di una classe:



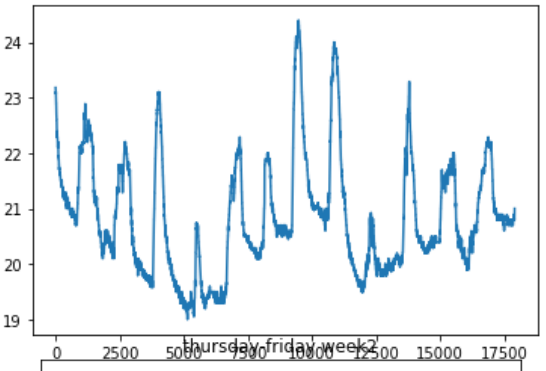


Considerando dunque la TS nella sua interezza è possibile osservare l'alternanza giorno notte come **motif** ricorrente, come è possibile osservare nel grafico mostrato di seguito, mentre risultano essere **anomalie** i picchi che raggiungono il valore di 1600 lux, come evidenziato anche nella sezione relativa agli outliers.

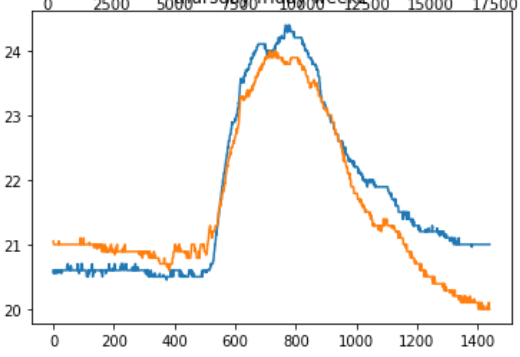


Passando invece all'attributo *Temperature,* l'andamento generale delle linee risulta molto diverso e meno "lineare". È sempre presente una forma di **stagionalità** data dall'escursione termica tipica dell'alternanza giorno notte, ma

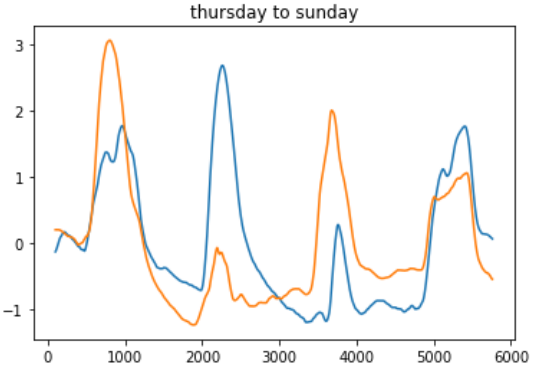
questa è più variabile rispetto a quella data dall'assenza o presenza di luce.
A livello di **trend** si può idealmente dividere il grafico in due parti: è possibile osservare come nella prima metà le temperature tendono a diminuire molto rapidamente, fino a raggiungere un picco a fine della prima settimana, dopo di che tendono a risalire più gradualmente.



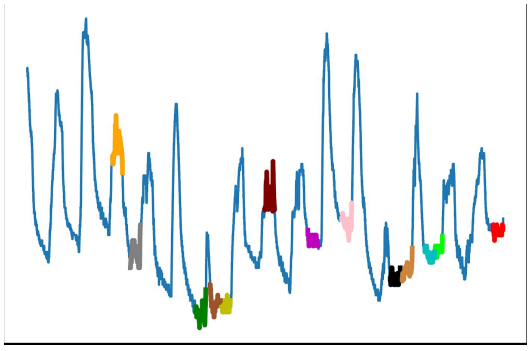
Anche in questo caso abbiamo deciso di mantenere una divisione per giorni in modo da poterli confrontare tra di loro. Ci siamo tuttavia chiesti se avesse comunque senso continuare a paragonare giorni uguali tra di loro: perché due lunedì avrebbero dovuto mostrare un andamento simile nelle temperature? Ci è sembrato quindi più logico provare ad analizzare giorni consecutivi, in modo da avere una visione più completa dei cambiamenti di temperatura. Per avere una visione più chiara abbiamo quindi eliminato dal grafico **noise** e **trend.**



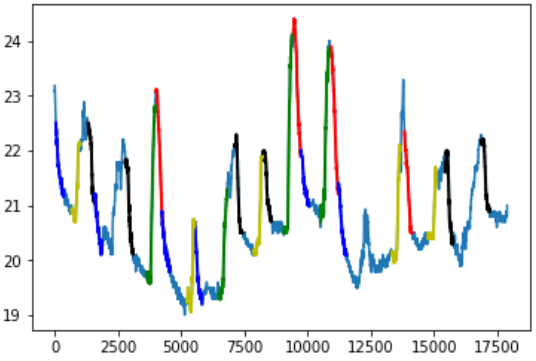
Come ci aspettavamo, giorni consecutivi mostrano andamenti simili. Ma questa visione ci ha in realtà permesso di osservare come, con qualche eccezione, tutti i giorni seguono più lo stesso andamento " a campana", cioè che sale ripidamente fino ad un picco, generalmente individuabile a metà della curva e, quindi, a mezzogiorno, per poi discendere più o meno rapidamente nelle ore più fresche della giornata.



I **motifs** di *Temperature* - ultimo grafico a destra - saranno dunque questi questi cambi di temperatura dati dall'escursione termica, mentre le anomalie - ultimo grafico a sinistra - saranno da ricercare in quelle curve che

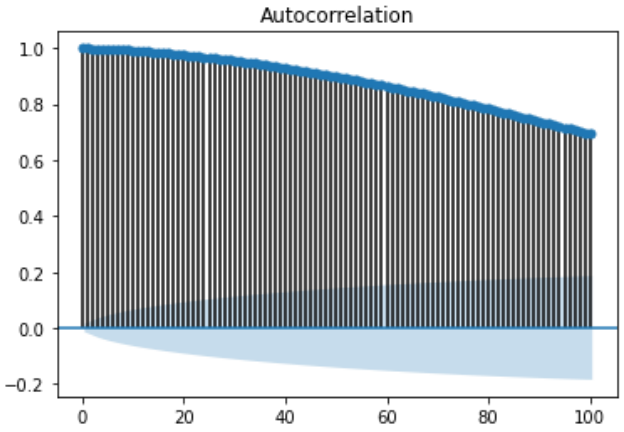


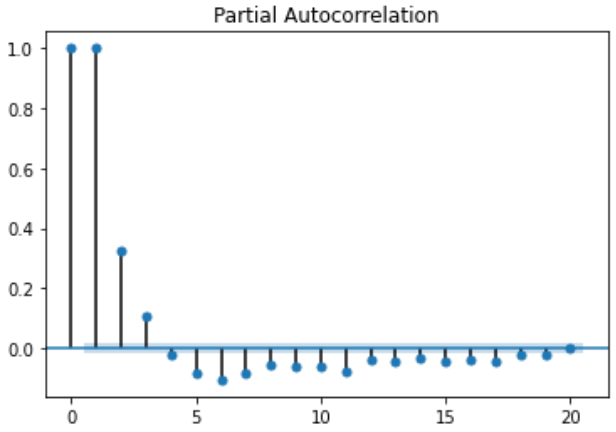
presentano più picchi nell'arco di un'unica giornata.



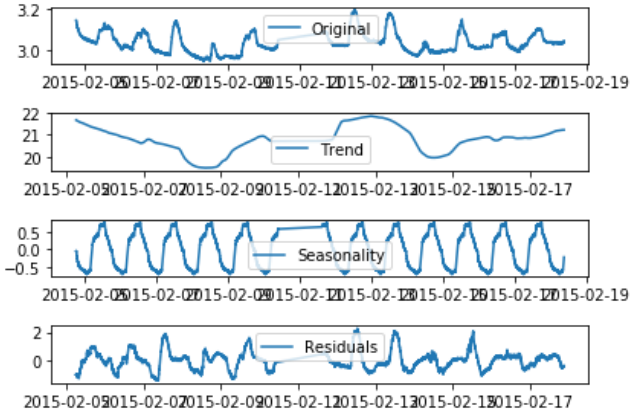
3.1 Forecasting

Date le caratteristiche distintive dei due attributi, abbiamo deciso di riferisci esclusivamente a *Temperature* per la costruzione di una previsione.

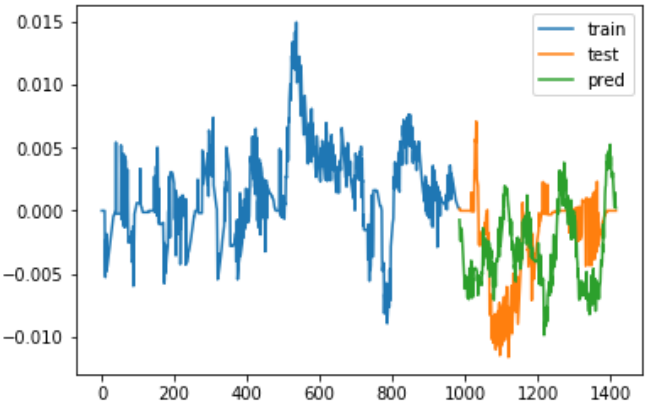




Da grafici relativi all' **ACF -** il grafico a sinistra - e al **PACF -** il grafico a destra - è possibile ricavare alcune interessanti informazioni riguardanti la nostra time series. Questa infatti presenta un alto livello di **autocorrelazione**, che si mantiene molto elevato anche per grandi finestre temporali (**lags**). Tuttavia possiamo osservare tramite **PACF** che le "informazioni" significativa per la costruzione di una predizione si concentrano principalmente nei primi quattro **lags**, e diventano ininfluenti già dopo i primi venti.



Abbiamo dunque provato a fare la previsione utilizzando **Exponential Smoothing**, **ARIMA** e **SARIMAX**, ma per fare ciò è stato prima necessario normalizzare il dataset e rimuovere dalla TS sia il **trend** che la **seasonality** in modo da renderla stazionaria.
Tuttavia i risultati ottenuti risultano essere poco convincenti e si adattano poco alla nostra TS.
Gli algoritmi Exponential Smoothing e Arima non sono stati molto performanti per questa TS, ma abbiamo invece avuto dei buoni risultati da SARIMAX che è riuscito a rappresentare meglio la distribuzione temporale di questo attributo.



MAE	0.004
RMSE	0.005
MAD	0.004

3.2 Clustering

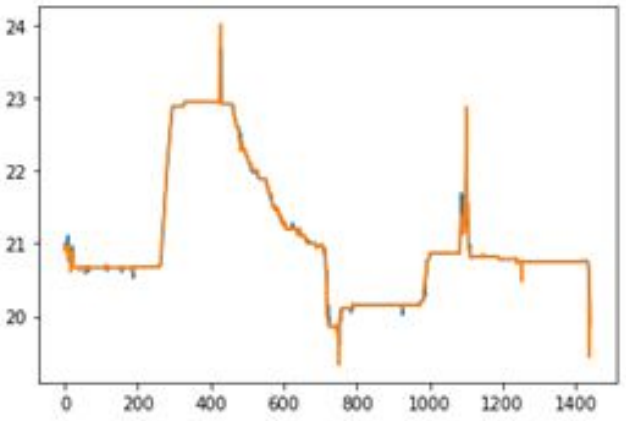
Il clustering riguardante le Time Series segue gli stessi principi del clustering tradizionale, ovvero cerca di raggruppare time series diverse in gruppi diversi secondo la misurazione della similarità tra Time Series. In tale modo è possibile avere una diversa configurazione del dataset in grado di catturare meglio alcuni aspetti non facilmente rilevabili.
Tra i vari metodi che potevamo applicare abbiamo scelto:

- *Shape Based Clustering*
- *Approximated Clustering*

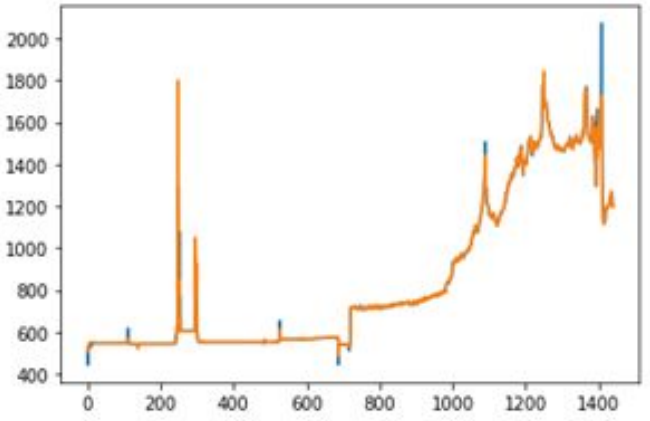
Per poter effettuare l'analisi abbiamo raggruppato le diverse time series scelte in un unico dataset selezionando quelle che a nostro parere di maggiore interesse.
Nel **Shape Based Clustering** abbiamo utilizzato diverse misure di similarità, ovvero: *Euclidean*, *Manhattan* e *Dynamic Time Warping*. Ovviamente il miglior risultato è stato ottenuto tramite la DTW.
Di seguito presentiamo i risultati ottenuti dagli attributi *Light* e *Temperature*.
Un esempio delle distanze ottenute per l'attributo Temperature, ad una minore distanza coinciderebbe un maggiore somiglianza tra i cluster.

Euclidean	507.9
DTW	17.2

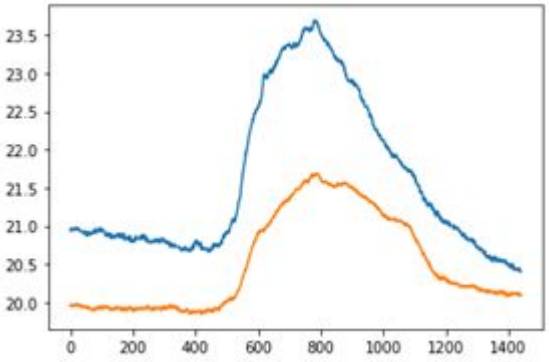
Le immagini si riferiscono ai cluster ottenuti utilizzando Shape Based Clustering con la DTW, la prima è in riferimento all'attributo *Temperature,* la seconda a *Light*.

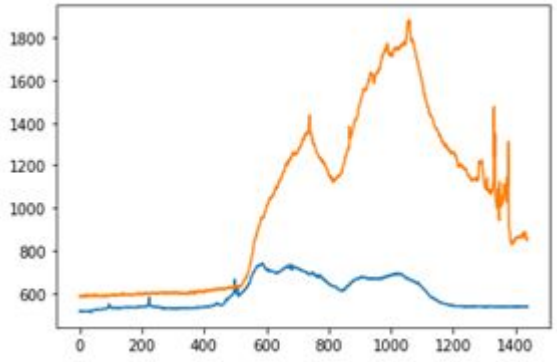


L' **Approximate Clustering** invece ha reso un migliore risultato termini di separazione delle classi, che era ciò che noi tentavamo di ottenere poiché con il metodo precedente abbiamo ottenuto una quasi sovrapposizione totale tra i cluster. Nell'impostare i parametri abbiamo scelto come descritto in precedenza un numero di cluster pari a 2 ed effettuato 20 iterazioni:



Le immagini nella pagina successiva si riferiscono ai cluster ottenuti utilizzando Approximate Clustering con DTW, la prima in riferimento a *Temperature*, la seconda a *Light*.





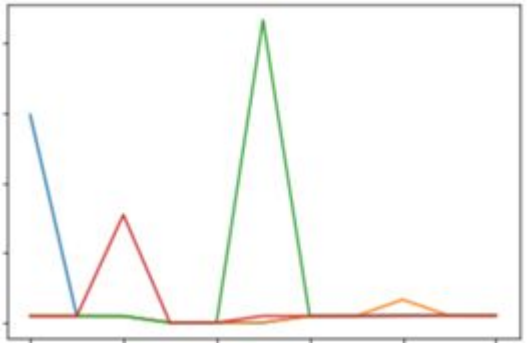
3.3 Classification

3.3.1 Univariate

Nello sviluppo della classificazione sul dataset composto da un unico attributo abbiamo deciso di utilizzare l'attributo Temperature, che è stato suddiviso in 264 time series (una time series per ogni ora che componeva l'intervallo di tempo coperto), avente la seguente forma:

264 *(rows)* **x 60** *(columns)*

In seguito, per dare i label ad ogni time series abbiamo sfruttato vari modelli di clustering (shape-based, feature-based, compression-based clustering), scegliendo tra questi quello che minimizza l'errore (la somma delle distanze al quadrato dei sample rispetto al centro del cluster più vicino).Abbiamo scelto quindi lo shape-based clustering, ottenendo un errore di 23.92 e la seguente individuazione di quattro cluster.



```
TimeSeriesKMeans(dtw_inertia=False, init='k-means++', max_iter=5,
                 max_iter_barycenter=100, metric='euclidean',
                 metric_params=None, n_clusters=4, n_init=1, n_jobs=None,
                 random_state=0, tol=1e-06, verbose=0)
```

Abbiamo quindi partizionato il dataset con la seguente forma: 70% train, 30% test. Prima di svolgere lo shapelet classifier abbiamo estratto le shapelets, che sono sotto sequenze estremamente rappresentative per ogni classe. Tuttavia abbiamo ottenuto risultati non particolarmente soddisfacenti (accuracy 64%), probabilmente perché le time series erano troppo brevi.
L'individuazione di cluster distinti tramite il modello sopra elencato ci ha permesso di classificare ogni time series con una ottima accuracy per quanto riguarda il KNN ed il Decision Tree, raggiungendo il

98%. Il miglior modello che ha performato in questa fase è sicuramente il Decision Tree raggiungendo un accuracy del 99%.
In seguito abbiamo implementato algoritmi di machine learning avanzati, come il recurrent neural network ed il long short-term memory. Tuttavia, i risultati ottenuti da questi modelli non sono stati soddisfacenti. In particolare, l'accuracy raggiunta con il modello LSTM sul validation set è stata particolarmente bassa, segnale interpretabile come probabile overfitting del modello.

3.2 Multivariate

Davanti alla necessità di testare una classificazione su una TS multivariata ci siamo chiesti che cosa effettivamente questo significasse e quali possibilità questo aprisse. Abbiamo quindi pensato a due diversi possibili utilizzi di una classificazione simile e abbiamo provato ad eseguirli.
Nella prima abbiamo utilizzato l'intero dataset come una TS multivariata e ci siamo chiesti se, attraverso le shapelet, l'IA fosse capace di riconoscere l'attributo a cui la curva faceva riferimento. Dopo aver rimosso gli attributi categorici abbiamo quindi modellato l'algoritmo sul training set per poi effettuare un test. Contro le nostre aspettative, i risultati non sono stati interessanti:

	Precision	F1-SCORE
Temperature	0.5	0.67
Light	0.5	0.67
CO2	0	0
Humidity	0	0

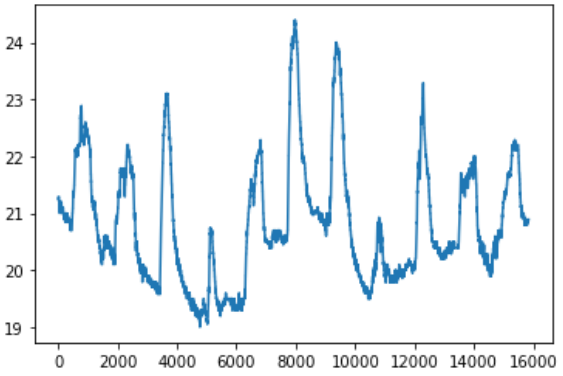
L'algoritmo è riuscito ad individuare gli attributi *Light* e *Temperature,* ma non ha classificato nessuna TS come *CO2* e *Humidity.*
Per il secondo tentativo, abbiamo creato una nuova TS a partire da tutti i record riferiti al giorno 9 febbraio e l'abbiamo suddivisa in altre 24 TS più piccole, ognuna corrispondente ad un'ora della giornata. Ogni TS contiene 59 record, ognuno espresso nei quattro attributi *Light*, *Temperature*, *CO2*, *Humidity*. Abbiamo quindi assegnato un label ad ogni TS a seconda l'orario a cui facesse riferimento fosse mattutino, pomeridiano o serale.

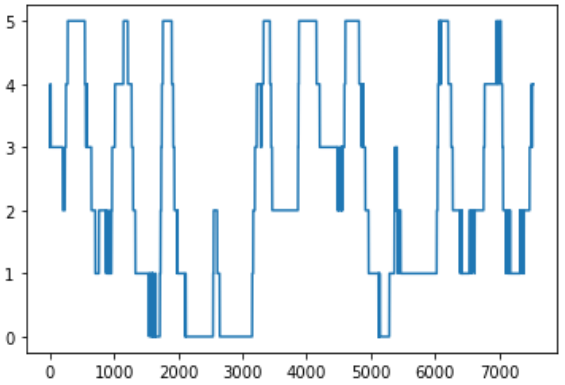
	Precision	F1-SCORE
Notte	0	0
Mattina	0.20	0.29
Pomeriggio	0.67	0.67

4. Pattern Mining

Come ultima analisi abbiamo provato a ricercare i pattern ricorrenti all'interno di una TS. Più specificamente, abbiamo suddiviso la nostra TS per ore in modo da ottenere 264 TS più piccole (24 ore per 11 giorni considerati) e su queste abbiamo poi eseguito l'algoritmo in modo da individuare le combinazioni ricorrenti. L'attributo utilizzato è Temperature.

Per fare ciò è stato necessario approssimare la TS originale utilizzando il metodo SAX: abbiamo ridotto il numero di record a 7530, ovvero poco meno della metà (il numero è stato scelto in modo da essere divisibile per 264), e li abbiamo classificati utilizzando un alfabeto di 6 valori. Dal confronto a destra si può osservare la TS prima e dopo l'approssimazione. Si noti quindi che nella SAX con "2" e "3" vengono classificati i valori che si distribuiscono intorno alla media, ovvero 21; con "5" tutti i valori dal 23 in sù; e con "0" i valori intorno a 19.





È stato dunque eseguito il Sequential Pattern Mining utilizzando diversi valori di **support** e ricercando esclusivamente pattern che fossero **closed.**

Min sup \ Length pattern	<5	5=<x<10	>=10
6	16	13	48
10	9	10	37
20	7	2	9

Data la natura stessa dell'attributo, i risultati ottenuti non sono stati particolarmente interessanti: il pattern mining ha evidenziato il fatto che le temperature non tendano a salire o scendere repentinamente da un valore all'altro, ma tendono invece a cambiare lentamente nel tempo. Per cui tutti i pattern ottenuti sono ripetizioni di temperature uguali a se stesse o, al massimo, con differenze di pochi gradi centigradi. Per cui per esempio per ben 26 volte compare un "pattern" che consiste nella ripetizione di trenta 0 (quindi è stata registrata per un lungo periodo una temperatura tra i 20 e i 19 gradi), et similia.