DATA MINING 2
PROJECT REPORT

2019-2020

Arianna Racioppa MAT. 602046

Valentina Faiman MAT. 543967

Lorenzo Mannocci MAT. 518263

Data Understanding e Data Preparation

L'analisi affrontata in questo report si basa sull' "Occupancy Detection Dataset", composto da tre file distinti:

- *training.txt*: 8143 records,
- *datatest.txt*: 9752 records,
- *datatest2.txt*: 2665 records.

Il dataset è composto da rilevazioni con frequenza di un minuto tramite dei sensori che registrano le condizioni fisiche presenti all'interno di un ufficio a San Jose, California nel periodo di Febbraio 2015.
Gli attributi di cui si dispone sono sette: *date* (tipo data), attributi numerici continui relativi alle condizioni fisiche della stanza in un certo istante (*Temperature*, *Light, CO2*, *Humidity, HumidityRatio),* un label di tipo booleano che indica se la stanza è occupata (1) oppure no (0).
Nella Tabella 1 e nella Tabella 2 sono riportate delle statistiche utili nelle fasi di analisi successive relative al dataset comprensivo di tutti e tre i file.

	Temperature	Humidity	Light	CO2	HumidityRatio
mean	20.906	27.656	130.757	690.553	0.004
std	1.055	4.982	210.431	311.201	0.001
min	19	16.745	0	412.75	0.002674127
max	24.40833333	39.5	1697.25	2076.5	0.006476013

Tabella 1: Statistiche relative agli attributi numerici presenti nel dataset

	Temperature	Humidity	Light	CO2	HumidityRatio
Temperature	1.00	-0.16	0.69	0.45	0.21
Humidity	-0.16	1.00	-0.03	0.30	0.93
Light	0.69	-0.03	1.00	0.45	0.22
CO2	0.45	0.30	0.45	1.00	0.48
HumidityRatio	0.21	0.93	0.22	0.48	1.00

Tabella 2: Matrice di correlazione per gli attributi numerici presenti nel dataset

È stato notato che i valori di *Light*, *Temperature* e *CO2* hanno un andamento simile in ogni giornata ed in particolare *Light* evidenzia momenti di on ed off. Inoltre, si è osservato che nel week-end *Light* e *Temperature* a inizio e fine giornata hanno rispettivamente un aumento e una diminuzione più graduale rispetto ai giorni lavorativi.
Di conseguenza, per poter mantenere queste informazioni nelle task di classificazione, al dataset originale sono state aggiunte due colonne:

- WE: individua se il rilevamento si riferisce ad un giorno della settimana (0) oppure al fine settimana (1);
- hour: indica l'ora in cui è stato registrato il dato, varia nell'intervallo [0,23] e in seguito sarà considerata come ordinale e non numerica.

Le variabili numeriche continue (*Temperature, Humidity, Light, CO2, HumidityRatio*) sono state normalizzate sia secondo la Standard che MinMax normalization. Quest'ultima è stata applicata dopo aver osservato che non vi sono rilevazioni che si allontanano in maniera significativa dalla distribuzione del resto dei dati nel dataset.

Task 1: Basic Classifiers and Evaluation

Classification Task

L'obiettivo della classificazione per questo dataset è di predire se la stanza è occupata (1), oppure no (0).
Gli attributi ed i classificatori utilizzati sono i seguenti:

- Regressione, KNN effettuati sul dataset originale, escludendo l'attributo *date*;
- Naive Bayes, Decision Tree effettuati sul dataset integrato con i nuovi attributi, escludendo *date*.

Per ogni tecnica di classificazione è stato utilizzato il dataset con le due diverse normalizzazioni (Standard e MinMax) in modo da poter confrontare le performance.
Volendo eseguire per i modelli che necessitano di un tuning degli iperparametri una K-fold Cross Validation, si è deciso di unire tutti i file in unico dataset. In seguito, è stata eseguita la divisione tra Training Set e Test Set in percentuale 70-30, con il quale si è ottenuto:

- Training Set: 14392 records,
- Test Set: 6168 records.

I dati all'interno delle due classi sono distribuiti in una percentuale pari al 23.10% per la classe 1, 76.90% per la classe 0: il dataset quindi non presenta un forte sbilanciamento.

K-NN

Per decidere quale fosse il corretto numero di nearest neighbours da tenere in considerazione, è stata fatta una grid search con 5-fold cross validation, per tutti i valori compresi tra 5 e 120 con weighted distance o uniform. Nella Tabella 3 si riporta la configurazione che ha prodotto i migliori risultati medi sul Validation Set in base all'f1-score.

n_neighbors	weight	accuracy	f1_micro	recall	roc_auc	precision
20	distance	0.992148	0.992148	0.99218	0.998634	0.974336

Tabella 3: Configurazione migliore in base alla f1-score e relativi risultati sul Validation Set

Si è notato che all'aumentare dei nearest neighbors, conseguiva un aumento della recall, ma allo stesso tempo una riduzione della precision e l'accuracy rimaneva quasi costante (sia nel caso di peso uniforme, sia nel caso weighted distance).
In generale tutte le configurazioni provate nella grid search danno risultati soddisfacenti, ma i valori migliori vengono ottenuti per k=20 e weighted distance. Nella Tabella 4 viene riportato il risultato della performance del modello sul Test Set, che risulta ottima soprattutto per la classe 0.

	precision	recall	f1-score
0	1	0.99	1
1	0.97	1	0.98
accuracy	0.9925		

Tabella 4: Risultati ottenuti sul test per il KNN

Naive Bayes

Non avendo iperparametri da definire, al dataset è stato applicato il GaussianNaiveBayes che ha prodotto i risultati in Tabella 5. I risultati della classificazione sul dataset normalizzato con Standard e MinMax sono identici. Rispetto agli altri classificatori i risultati sono leggermente inferiori poiché il Naive Bayes necessita dell'indipendenza degli attributi, che in questo dataset sono correlati tra loro.
È stato effettuato un tentativo di classificazione escludendo anche l'attributo *HumidityRatio* (fortemente correlato con *Humidity*), ma non sono stati ottenuti miglioramenti significativi.

	precision	recall	f1-score
0	1	0.95	0.97
1	0.85	1	0.92
accuracy	0.9586		

Tabella 5: Risultati ottenuti sul test per il Naïve Bayes

Decision Tree

Per capire quali siano gli attributi su cui si basa la classificazione è stata calcolata la features'importance, rappresentata nella Figura 2. *Light* è l'attributo risultato di gran lunga più importante, per cui si è deciso di costruire un modello basato su tutti gli attributi eccetto esso per verificare quanto la classificazione dipenda da questa variabile. Nella Figura 1 sono mostrate le features'importance in questo secondo caso: si evidenzia così come in secondo luogo l'attributo più importante risulti essere *Temperature*, ma come anche gli altri, tra cui la nuova feature *hour,* concorrano alla classificazione corretta dei record.

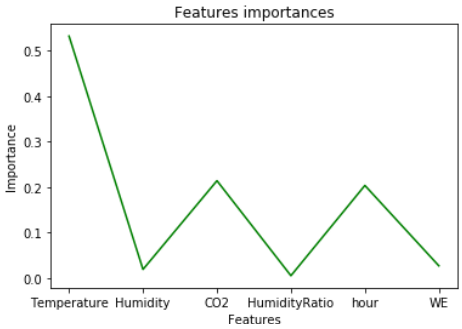

Figura 1: Features'importance degli attributi senza l'attributo Light

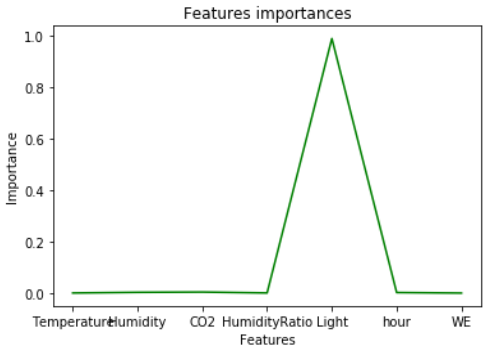

Figura 2: Features'importance degli attributi

È stata eseguita inizialmente una RandomizedSearchCV, per poter scegliere i parametri da testare. Da questa prima ricerca si è potuto constatare come l'iperparametro min_depth non influisca in alcun modo sulle performance del modello, quindi è stata settata a 'none'. Si è proceduto con la grid search 20-cross fold validation provando a variare i seguenti "parametri":

- Entropy/Gini Index;
- Utilizzo dell'attributo *Light*/Senza attributo *Light.*

I risultati ottenuti per la normalizzazione MinMax e normalizzazione Standard sono identici, si è deciso di mostrare dunque solo i risultati per la prima.
I risultati utilizzando il Gini index sono leggermente migliori di quelli ottenuti usando l'Entropy, per tutte le possibili metriche prese in considerazione.
Inoltre, comparando i risultati ottenuti con e senza l'attributo *Light*, è stato osservato che le performance sul Training Set sono migliori nel secondo caso, ma si ottengono risultati leggermente peggiori sul Test Set (Tabella 7), questo significa che in questo caso il modello presenta un lieve overfitting. In conclusione, la miglior combinazione risulta essere quella nella Tabella 6 (MinMax, *Light*, Gini Index).

MinMax, Gini Index, min_sample_leaf = 1, mean_samples = 5						
	Class	**precision**	**recall**	**f1-score**	**accuracy**	**roc_auc**
Training Set	0	0.99	0.99	0.99	0.99826	
	1	0.99	0.99	0.99		
Test Set	0	0.99	0.99	0.99	0.99303	0.99
	1	0.98	0.98	0.98		

Tabella 6: Risultati sul Training Set e sul Test Set per il modello di Decision Tree che utilizza tutti e 5 gli attributi numerici, compresa Light

MinMax, NO Light, Gini Index, min_sample_leaf = 1, mean_samples = 5						
	Class	**precision**	**recall**	**f1-score**	**accuracy**	**roc_auc**
Training Set	0	1	1	1	1	
	1	1	1	1		
Test Set	0	0.99	0.99	0.99	0.98817	0.98
	1	0.97	0.97	0.97		

Tabella 7: Risultati sul Training Set e sul Test Set per il modello di Decision Tree che utilizza attributi numerici, esclusa Light

Infine, si riporta nella Figura 3 una parte dell'albero finale.

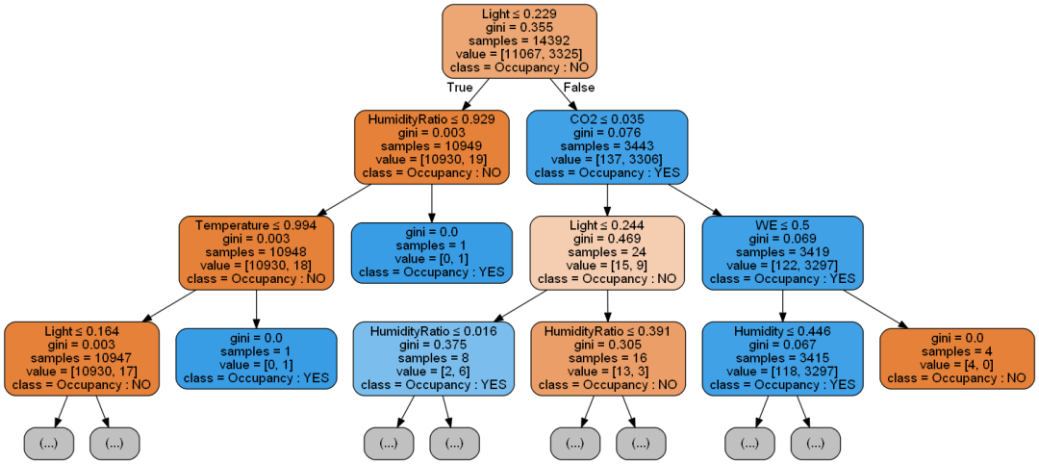

Figura 3: Parte dell'albero costruito dal Decision Tree Classifier

Logistic Regression

L'unico iperparametro da scegliere per questo classificatore è C, tuttavia è stato verificato che al suo variare non sono stati prodotti cambiamenti significativi nel test, di conseguenza è stato mantenuto il valore di default.
Il modello selezionato ha i seguenti valori per i coefficienti ed intercetta:

- coeff: -6.0934e-01, 9.5674e-02, 2.3299e-02, 3.5855e-03, 4.3270e-08;
- intercetta: -0.025.

Nella Tabella 8 vengono riportati i risultati del modello sul test.

Class	**precision**	**recall**	**f1-score**	**accuracy**	**roc_auc**
0	0.99	0.98	0.99	0.9893	0.9944
1	0.95	0.99	0.97		

Tabella 8: Risultati ottenuti sul test per la Logistic Regression

Riduzione della dimensionalità

In questa sezione si andrà ad attuare una riduzione della dimensionalità del dataset con i metodi di Variance Threshold, Univariate Feature Selection, Recursive Feature Elimination e infine PCA. Dopo aver applicato ognuno di questi metodi (ad eccezione del Recursive Feature Elimination che è stato utilizzato solamente per il Decision Tree) è stato costruito un modello per ogni classificatore. Nei casi in cui il classificatore richiedesse una selezione di iperparametri è stata effettuata una grid search (KNN e DT). Vengono riportati i risultati in base al classificatore e alle tecniche utilizzate.

KNN

Gli attributi selezionati per questo classificatore sono diversi a seconda del metodo utilizzato. Nel caso del **Variance Threshold** sono stati mantenuti *Humidity* ed *HumidityRatio,* due attributi altamente

correlati tra loro; mentre, con l'Univariate Feature Selection le colonne rimanenti sono *Temperature* e *Light*. Queste diverse selezioni hanno portato l'**Univariate Feature Selection** ad avere una performance migliore in quanto ha conservato attributi più interessanti per la classificazione. In entrambi i metodi dalla grid search i miglior iperparametri sono risultati: 7 nearest neighbors e weight distance. Nella Tabella 9 e nella Tabella 10 sono riportati i risultati della classificazione sul test.

Variance Threshold (0.04) {Humidity, HumidityRatio}			
	precision	**recall**	**f1-score**
0	0.95	0.97	0.96
1	0.90	0.84	0.87
accuracy	0.9422		

Tabella 9: Risultati sul test per KNN dopo aver utilizzato il Variance Treshold con una treshold uguale a 0.4

Univariate Feature Selection {Temperature, Light}			
	precision	**recall**	**f1-score**
0	0.99	0.99	0.99
1	0.97	0.98	0.97
accuracy	0.9881		

Tabella 10: Risultati sul test per KNN dopo aver utilizzato Univariate Feature Selection

Per quanto riguarda la PCA, il dataset è stato ridotto a due dimensioni. Nella Tabella 11 sono riportati i risultati della classificazione eseguita sul test (parametri utilizzati K=4, weight=distance) e nella Figura 4 viene mostrata la decision surface. Le performance del classificatore sono più basse per la classe 1, questo si può anche dedurre dalla Figura 5 grazie alla presenza di un buon numero di punti rossi (classe 0) tra i punti verdi (classe 1).

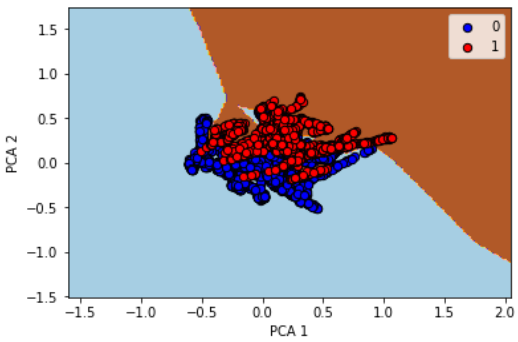

Figura 4: Decision surface e scatterplot del Test Set per il KNN

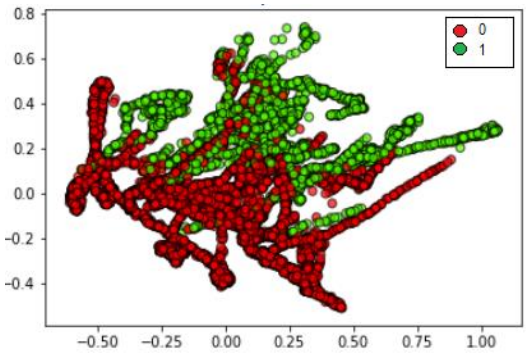

Figura 5: Plot delle due dimensioni della PCA del Training Set

PCA			
	precision	**recall**	**f1-score**
0	0.98	0.98	0.98
1	0.93	0.92	0.93
accuracy	0.9657		

Tabella 11: Risultati sul Test Set del KNN dopo aver utilizzato PCA

Naive Bayes

Il dataset utilizzato è quello integrato con le variabili *hour* e *WE*. In questo caso, sono proprio queste due ad essere selezionate dalla **Variance Threshold**, mentre con **Univariate Feature Selection** gli attributi rimanenti sono *Temperature* e *Light*.
Come si può osservare dalla Tabella 12 e dalla Tabella 13, nel caso dell'Univariate i risultati sul test sono nettamente migliori, questo perché, come già spiegato precedentemente, gli attributi selezionati sono più significativi.

Variance Threshold (0.01) {hour, weekend}			
	precision	**recall**	**f1-score**
0	1	0.24	0.38
1	0.28	1	0.44
accuracy	0.4119		

Tabella 12: Risultati sul test del Naive Bayes dopo aver utilizzato Variance Treshold con una treshold uguale a 0.1

Univariate Feature Selection {Temperature, Light}			
	precision	**recall**	**f1-score**
0	1	0.96	0.98
1	0.87	1	0.93
accuracy	0.9659		

Tabella 13: Risultati sul test del Naive Bayes dopo aver utilizzato Univariate Feature Selection

La PCA ottiene una precision bassa per la classe 1, come si può osservare dalla Tabella 14. La particolare distribuzione dei punti può essere imputata alla presenza delle due nuove variabili *hour* e *WE,* come mostra la Figura 6.

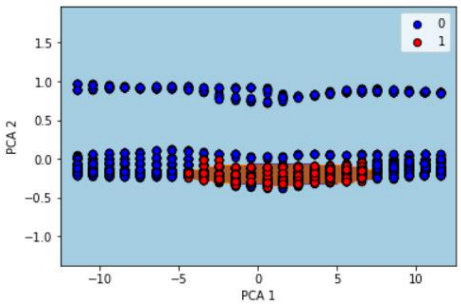

Figura 6: Decision boundaries e scatterplot del Test Set per il Naives Bayes

PCA			
	precision	**recall**	**f1-score**
0	0.97	0.89	0.93
1	0.71	0.90	0.80
accuracy	0.8929		

Tabella 14: Risultati sul test del Naive Bayes dopo aver utilizzato PCA

Decision Tree

Le feature selezionate sono quelle ottenute nel paragrafo precedente (in quanto si sta utilizzando lo stesso dataset). In questo caso però è stato possibile applicare anche la Recursive Feature Elimination, che ha selezionato *CO2* e *Light*. Per i primi tre metodi (VT, RFE, UFS), sono indicate nelle tabelle le feature selezionate e la loro relativa importanza nell'albero di decisione.
Dalla Tabella 15, Tabella 16 e Tabella 17 il peggior metodo risulta essere Variance Threshold, dove la classe 1 ottiene risultati molto più bassi rispetto a tutti gli altri metodi, soprattutto in termini di precision.
Univariate Feature Selection e Recursive Feature Elimination ottengono risultati ottimi anche con solo due feature, riducendo molto la dimensionalità.

	Variance Treshold (0.1) {'hour': 0.78, 'WE': 0.21}			Univariate Feature Selection {'Temperature': 0.0047, 'Light': 0.99}			Recursive Feature Elimination (threshold = 0.012) {'CO2': 0.0038, 'Light': 0.99}		
	precision	**recall**	**f1-score**	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
0	0.95	0.91	0.93	0.99	0.98	0.99	0.99	0.98	0.99
1	0.75	0.85	0.8	0.96	0.99	0.97	0.96	0.99	0.97
accuracy	0.9012			0.98			0.98		

Tabella 15: Risultati sul test del Decision Tree dopo aver utilizzato Variance Treshold con una treshold uguale a 0.1

Tabella 16: Risultati sul test del Decision Tree dopo aver utilizzato Univariate Feature Selection

Tabella 17: Risultati sul test del Decision Tree dopo aver utilizzato Recursive Feature Elimination

Infine, per quanto riguarda la PCA si mostra, oltre alle performance (Tabella 18) che rimangono buone per la classe 0 e in calo per la classe 1, uno scatterplot con i decision boundaries (Figura 7). Si può osservare da questo grafico come risulti più complessa la decisione nella parte centrale, dove confinano

le due classi. Infatti, gran parte degli split presenti nell'albero sono necessari per riuscire a dividere correttamente i punti in questa zona.

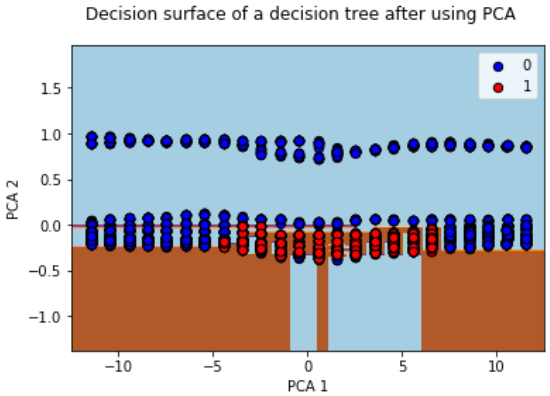

Figura 7: Decision boundaries e scatterplot del Test Set per il Decision Tree

PCA			
	precision	**recall**	**f1-score**
0	0.99	0.99	0.99
1	0.97	0.96	0.96
accuracy	0.98		

Tabella 18: Risultati sul test del Decision Tree dopo aver utilizzato PCA

Logistic Regression

In questo caso è stato utilizzato il dataset non normalizzato, tuttavia gli attributi selezionati dai due metodi sono gli stessi del paragrafo precedente. Anche con la Logistic Regression si può osservare dalla Tabella 20 che la **Univariate Feature Selection** ottiene performance nettamente migliori, soprattutto per la classe 1.

Variance Threshold (0.04) {Humidity, HumidityRatio}			
	precision	**recall**	**f1-score**
0	0.85	0.94	0.89
1	0.70	0.45	0.55
accuracy	0.8271		

Tabella 19: Risultati sul test della Logistic Regression dopo aver utilizzato Variance Treshold con una treshold uguale a 0.4

Univariate Feature Selection {Temperature, Light}			
	precision	**recall**	**f1-score**
0	1	0.98	0.99
1	0.94	1	0.97
accuracy	0.9837		

Tabella 20: Risultati sul test della Logistic Regression dopo aver utilizzato Univariate Feature

A differenza di quanto osservato per il KNN, in questo caso utilizzando la **PCA** si ottengono ottimi risultati (Tabella 21) e questo si può osservare dalla Figura 8, in cui i punti delle classi sono ben divisi fra loro.

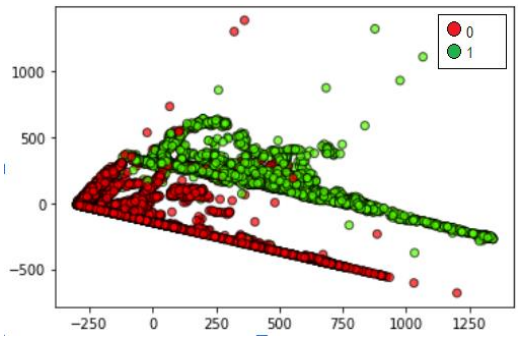

Figura 8: Plot delle due dimensioni della PCA del Training Set

PCA			
	precision	**recall**	**f1-score**
0	1	0.99	0.99
1	0.96	1	0.98
accuracy	0.9899		

Tabella 21: Risultati sul test della Logistic Regression dopo aver utilizzato PCA

Classification su Unbalanced e Re-balanced Dataset

Partendo dal dataset originale, la cui distribuzione dei record nelle classi corrisponde a 23.10% per la classe 1 e 76.90% per la classe 0, è stato effettuato uno **sbilanciamento** attraverso la funzione

make_inbalance della libreria imblearn, la quale trasforma un set di dati nella sua versione non bilanciata con un rapporto specifico, nel nostro caso 96%-4%. Dopo lo sbilanciamento il dataset è passato da un totale di 20560 record a 16467: 15810 per la classe 0 e 658 per la classe 1.

In un secondo momento è stato applicato un metodo di bilanciamento del dataset. Tra le varie tecniche studiate si è scelto di utilizzare l'oversampling della classe minority poiché l'undersampling avrebbe portato ad una riduzione estrema del numero di dati.
In particolare, è stato applicato SMOTE sul Training Set, ottenendo un bilanciamento di 50 e 50. A questo punto il numero di record appartenenti ad ogni classe è di 11065. Sia sul dataset sbilanciato che ribilanciato sono stati utilizzati i vari classificatori per valutare le loro performance.

KNN

Per questo classificatore (n_neighbors=6, weights= distance), rispetto al dataset originale, è possibile notare dalla Tabella 22 una riduzione della capacità di classificare correttamente la classe 1, anche se questa si mantiene comunque piuttosto alta. Dopo il ribilanciamento (n_neighbors=5, weights=distance), è migliorata la recall, ma peggiorata la precision per la classe 1, cioè sono aumentati i false positive ma diminuiti i false negative (Tabella 23).

Sbilanciato			
	precision	**recall**	**f1-score**
0	0.99	0.99	0.99
1	0.94	0.92	0.93
accuracy	0.9949		

Tabella 22: Risultati sul test del KNN con il dataset sbilanciato

Ribilanciato con SMOTE			
	precision	**recall**	**f1-score**
0	0.99	0.99	0.9971
1	0.89	0.97	0.9343
accuracy	0.9945		

Tabella 23: Risultati sul test del KNN con il dataset ribilanciato con SMOTE

Naive bayes

Usando il dataset sbilanciato come training il classificatore ottiene una precision sulla classe 1 molto bassa, infatti il classificatore ha troppi pochi casi positivi ed etichetta erroneamente degli "0" come "1" (Tabella 24). Le performance sul dataset ribilanciato non migliorano, anzi la precision crolla ulteriormente e si ha anche un peggioramento della recall per la classe 0 (Tabella 25).

Sbilanciato			
	precision	**recall**	**f1-score**
0	1	0.95	0.97
1	0.43	1	0.61
accuracy	0.9481		

Tabella 24: Risultati sul test del Naive Bayes con il dataset sbilanciato

Ribilanciato con SMOTE			
	precision	**recall**	**f1-score**
0	1	0.92	0.96
1	0.35	1	0.52
accuracy	0.9275		

Tabella 25: Risultati sul test del Naive Bayes con il dataset ribilanciato con SMOTE

Decisione Tree

Con questo classificatore le performance relative alla classe 0 non variano molto rispetto ai risultati ottenuti sul dataset originale, tuttavia peggiorano leggermente nel caso del dataset sbilanciato (Tabella 26), per poi migliorare leggermente in quello ribilanciato con SMOTE.
Non si può dire lo stesso per la classe 1 in quanto si ha un calo per tutte le metriche. Al contrario, si ha un miglioramento quando si utilizza il dataset ribilanciato (Tabella 27), in particolare per la recall, che inficia anche su un miglioramento della f1-score, mentre cala la precision. Anche in questo caso SMOTE ha apportato i miglioramenti voluti alla classificazione.

Sbilanciato			
	precision	**recall**	**f1-score**
0	0.99	0.99	0.99
1	0.91	0.90	0.91
accuracy	0.9929		

Tabella 26: Risultati sul test del Decision Tree con il dataset sbilanciato

Ribilanciato con Smote			
	precision	**recall**	**f1-score**
0	0.99	0.99	0.99
1	0.89	0.96	0.93
accuracy	0.9943		

Tabella 27: Risultati sul test del Decision Tree con il dataset ribilanciato con SMOTE

Logistic Regression

Confrontando i risultati delle metriche con quelli del dataset originale si può notare che anche in questo caso, dopo lo sbilanciamento, il modello non riesce più a predire in modo accurato la classe 1 (Tabella 28). Dopo il bilanciamento, la recall della classe 1 ha raggiunto il 100%, però la precision è diminuita. In generale, le metriche per classe 0 sono diminuite ma non di molto (Tabella 29). In questo caso quindi SMOTE ha portato risultati soddisfacenti.

Sbilanciato			
	precision	**recall**	**f1-score**
0	0.99	0.99	0.99
1	0.78	0.85	0.81
accuracy	0.9848		

Tabella 28: Risultati sul test della Logistic Regression con il dataset sbilanciato

Ribilanciato con SMOTE			
	precision	**recall**	**f1-score**
0	1	0.98	0.99
1	0.75	1	0.85
accuracy	0.9868		

Tabella 29: Risultati sul test della Logistic Regression con il dataset ribilanciato con SMOTE

Linear regression problem

In questa sezione verranno trattati due problemi di regressione lineare: il primo consiste nel creare un modello che abbia come variabile target *Light*, nel secondo invece la variabile dipendente selezionata è *Humidity*.

Target: Light

In un primo momento è stato utilizzato un modello di regressione lineare semplice, utilizzando come variabile indipendente *Temperature*. Come è possibile osservare dalla Figura 9 e dalla Tabella 30, i risultati della regressione lineare semplice sono mediocri. Utilizzando invece tutti gli attributi come variabili indipendenti, il coefficiente R2 aumenta. Le regolarizzazioni Ridge e Lasso, invece, non hanno migliorato le performance.

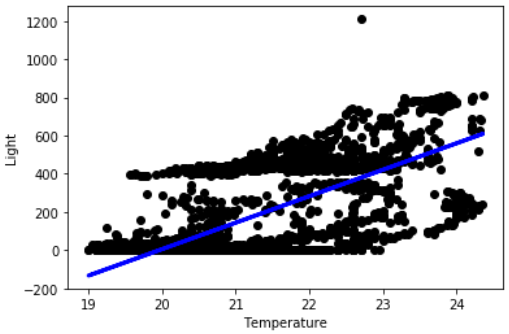

Figura 9: Scatterplot e retta generata dalla regressione lineare

	Coefficiente di determinazione R2
Simple Linear Regression	0.467
Least squares approach	0.494
Lasso	0.494
Ridge	0.494

Tabella 30: Risultati della Simple e Multiple Linear Regression sull'attributo Temperature

Target: Humidity

Per questo target il modello di regressione lineare è stato costruito utilizzando come variabile indipendente: *HumidityRatio*. Essendo i due attributi altamente correlati, i risultati sono soddisfacenti, come si può osservare dalla Tabella 31. Questi migliorano ulteriormente quando nella regressione vengono utilizzati tutti gli attributi, mentre le regolarizzazioni Lasso e Ridge hanno peggiorato nettamente i risultati, quindi non erano presenti problemi di multicollinearità, dato anche il basso numero di feature utilizzate.

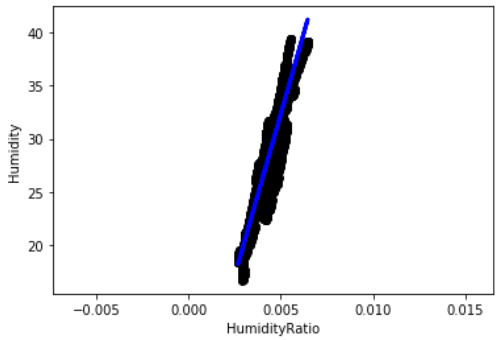

Figura 10: Scatterplot e retta generata dalla regressione lineare

	Coefficiente di determinazione R2
Simple Linear Regression	0.869
Least squares approach	0.997
Lasso	0.124
Ridge	0.208

Tabella 31: Risultati della Simple e Multiple Linear Regression sull'attributo Humidity

Conclusione

Dopo aver testato gli algoritmi di classificazione in diverse condizioni, è possibile concludere quanto segue:

- Sul dataset originale i classificatori hanno tutti buone prestazioni: KNN e Decision Tree ottengono i risultati migliori, mentre per Naive Bayes sono leggermente inferiori;
- Nel caso della dimensionality reduction i classificatori che performano meglio sono Decision Tree con Univariate Feature Selection e la regressione logistica con la PCA;
- Nel caso del dataset sbilanciato KNN è quello che consegue i migliori risultati;
- Nel dataset ribilanciato con SMOTE, i risultati migliori si hanno con KNN e Decision Tree, tuttavia si evidenzia che il classificatore che consegue più vantaggi dal bilanciamento è la regressione logistica, al contrario del Naive Bayes che vede verificarsi un peggioramento dei risultati.

F1-Score per classe 0 e 1				
	KNN	**Naive Bayes**	**Decision Tree**	**Logistic Regression**
Dataset Originale	1	0.97	0.99	0.99
	0.98	0.92	0.98	0.97
Dataset Dimensionlity Reduction	0.99	0.98	0.99	0.99
	0.97	0.93	0.98	0.98
	Univariate	Univariate	Univariate	PCA
Imbala. Dataset	0.99	0.97	0.99	0.99
	0.93	0.61	0.91	0.81
SMOTE Balanced	0.99	0.96	0.99	0.99
	0.93	0.52	0.93	0.85

Tabella 32: Riassunto dei risultati della f1-score (classe 0 sopra e classe 1 sotto) per ogni modello

Si riporta infine un esempio di grafici di ROC Curve, Cumulative Gains Curve e Lift Chart (Figura 11, Figura 12 e Figura 13), solamente nel caso del Naive Bayes sul dataset originale, in quanto i risultati ottenuti sono simili per tutti i classificatori base.

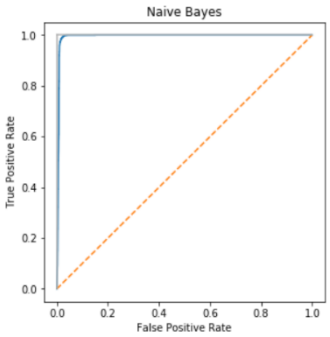

Figura 11: ROC Curve per Naives Bayes

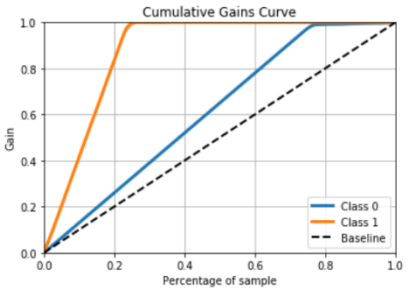

Figura 12: Cumulative Gains Curve per Naives Bayes

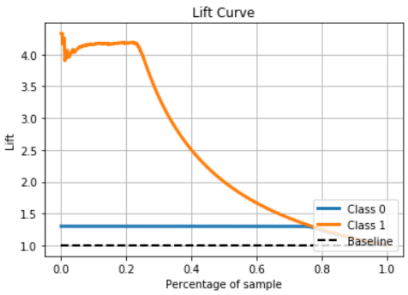

Figura 13: Cumulative Gains Curve per Naives Bayes

Task 2: Advanced Classifiers and Evaluation

Support Vector Machine

Linear SVM

Questo classificatore è stato utilizzato sul dataset normalizzato ad esclusione degli attributi *WE* e *hour*. È stata fatta una grid search (3-fold validation) per individuare il miglior valore di C. I risultati migliori vengono ottenuti per alti valori di C ed in particolare i risultati sul Test Set con C=15 sono descritti nella Tabella 33.

	precision	recall	f1-score
0	1	0.99	0.99
1	0.96	1	0.98
accuracy		0.9907	

Tabella 33: Risultati sul test della miglior configurazione per il Linear SVM

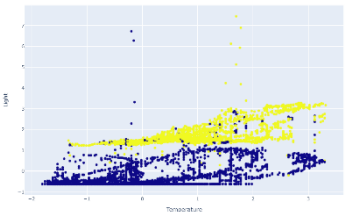

Figura 14: Scatterplot degli attributi Light e Temperature

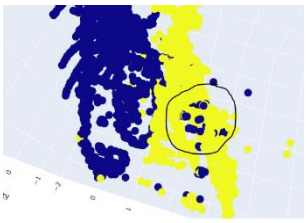

Figura 15: Scatterplot tridimensionale degli attributi

I risultati sono soddisfacenti e per comprendere meglio come un iperpiano lineare riesca a separare bene i dati nelle due classi è stata effettuata un'analisi degli scatterplot (Figura 14 e Figura 15). Con la presenza di *Light* è possibile notare una separazione piuttosto netta tra le due classi, proprietà probabilmente mantenuta nello spazio n-dimensionale.
Attraverso la rappresentazione 3D dei seguenti attributi *Temperature*, *Light* e *Humidity* si può fornire una spiegazione alla bassa precision della classe 1: alcuni punti della classe 0 (blu) sono mescolati con i punti della classe 1 (gialli) e sono proprio questi che costituiscono i "false positive".

Non-linear SVM

Si è effettuata una grid search 5-fold cross validation utilizzando il non linear SVM. L'obiettivo è quello di trovare, scegliendo tra tre diversi kernel poly, rbf e sigmoid, diversi valori di C e di Gamma, una combinazione di parametri che migliorasse le prestazioni del linear SVM.
Nella Tabella 34 si riportano le migliori configurazioni per ciascun kernel con il relativo valore di f1-score nel validation.

	rbf, C: 100, gamma: 1	poly, C: 100, gamma: 1	Sigm, C: 100, gamma: 0.01
f1_score	0.987451	0.975497	0.983329

Tabella 34: Risultati della f1-score per il Validation Set al variare del kernel utilizzato con la migliore configurazione

	precision	recall	f1-score
0	1	0.99	1
1	0.98	0.99	0.98
accuracy	0.9927		

Tabella 35: Risultati sul test con kernel rbf

Il kernel con performance migliore sul validation è rbf ed i risultati della configurazione eseguiti sul test sono contenuti nella Tabella 35. Si può notare un miglioramento della precision per la classe 1 rispetto al linear SVM. In conclusione, nonostante entrambi i modelli presentati producano risultati estremamente positivi, il linear SVM risulta leggermente peggiore.

Neural Network

Single Linear Perceptron

Per questo algoritmo di classificazione sono state valutate le performance variando i parametri eta0 e penalty attraverso una grid search con 3 Cross-Fold Validation.
Mantenendo sempre epochs=100 e class_weight=balanced, nella Tabella 36 sono riassunti i parametri risultanti dalla grid search che hanno i valori migliori di f1-score sul validation.

	Conf.1	Conf.2	Conf.3	Conf.4
penalty	l2	l2	l2	l1
eta0	0.4	0.2	0.3	0.5
f1_score	0.9849	0.9848	0.9844	0.9841

Tabella 36: Parametri risultanti dalla grid search con relativa f1-score

	precision	recall	f1-score
0	1	0.92	0.96
1	0.79	1	0.88
accuracy	0.9385		

Tabella 37: Risultati sul test per la miglior configurazione

Come si può notare, il numero di eta0 è abbastanza piccolo, sempre sotto lo 0.5. La configurazione che presenta i risultati migliori sul Validation Set è la prima. La performance sul test è descritta nella Tabella 37 e, come si può osservare, vi è una bassa precision per la classe 1.

Multilayer Perceptron

Il Multilayer Perceptron Classifier (MLP) necessita del settaggio di una grande quantità di iperparametri. Quindi, per individuare la combinazione più adatta al nostro dataset è stata fatta una grid search con 9-Cross Fold Validation dove sono state considerate diverse combinazioni di iperparametri. In particolare, per hidden_layer_size è stato utilizzato un solo hidden layer contenente un numero di neuroni compresi tra 2 e 6, in quanto il numero ottimale di hidden neuron è solitamente compreso tra la dimensione dell'input e quella dell'output. Anche per il batch_size sono stati considerati diversi valori (6, 12, 24, 32, 64 e 128). I restanti iperparametri sono stati inseriti nella grid considerando tutte le varie possibilità disponibili, ovvero activation: ['tanh', 'relu', 'logistic'], solver: ['sgd', 'adam', 'lbfgs'], momentum tra 0.1 e 0.9 e learning_rate: ['constant','adaptive'].
Sono riportate nella Tabella 38 le configurazioni risultanti dalla grid search con il più alto valore di f1-score sul validation.

batch_size	hidden layer_sizes	learning rate	solver	momentum	f1_score
12	5	constant	adam	0.9	0.9856
12	3	constant	adam	0.8	0.9854
32	5	constant	adam	0.8	0.9854
32	6	constant	adam	0.6	0.9854

Tabella 38: Configurazioni risultanti dalla grid search con relativa f1-score

Come si può notare, nelle migliori combinazioni il learning rate ed il solver sono sempre gli stessi, inoltre il momentum mantiene valori diversi, ma tutti alti.
Il modello applicato sul Test Set è il primo, ovvero quello che ha prodotto i risultati migliori sul validation. L'esito della classificazione è riassunto nella Tabella 39.

	precision	recall	f1-score
0	1	0.99	0.99
1	0.96	1	0.98
accuracy	0.9905		

Tabella 39: Risultati sul test per il Multilayer Perceptron

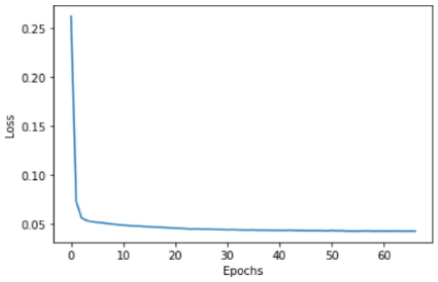

Figura 16: Variazione dalla Loss al variare del numero di epochs

La Figura 16 mostra il variare della Loss al crescere di epochs: la rete neurale sembra apprendere molto velocemente poiché già dopo poche iterazioni la curva raggiunge valori molto bassi, per poi continuare a decrescere più lentamente fino ad ottenere un valore di 0,042.
Il Multilayer Perceptron, rispetto al Single Linear Perceptron, riesce a classificare i dati non linearmente separabili, ottenendo quindi un significativo miglioramento della f1-score su entrambe le classi (il motivo è il medesimo illustrato nella sezione della SVM).

Deep Neural Network

In questo caso è stato utilizzato sia un Validation Set (modello 1), che una k-fold cross validation (modello 2).
Si è creato innanzitutto un modello base, per poi variare il numero di epochs e batch_size, così da verificare preliminarmente quali fossero i valori migliori. I risultati ottenuti in tre casi diversi sono descritti nella Tabella 40.

	Epochs = 50, batch_size = 10			Epochs = 50, batch_size = 50			Epochs = 700, batch_size = 10		
Class	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**	**precision**	**recall**	**f1-score**
0	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99
1	0.96	0.99	0.97	0.96	0.99	0.98	0.97	0.98	0.99
accuracy	0.9905			0.9911			0.9915		

Tabella 40: Risultati sul test del medesimo modello al variare del numero di Epochs e batch_size

Si può osservare che aumentando il numero di batch_size i risultati migliorano leggermente (seconda configurazione). Mentre, incrementando il numero di epochs, si ha un lieve decremento della precision per la classe 1, ma un miglioramento di tutte le altre metriche. Quest'ultimo caso sembra quindi il migliore, tuttavia per motivi di tempo e computazionali utilizzare un così alto numero di epochs sarebbe stato infattibile. Dunque, si è proceduto con parametri settati a epochs=100 e batch_size = 10 per i due seguenti modelli.

Modello 1 senza e con Early Stopping

Si è così proceduto con lo stesso modello, con epochs=100 e batch_size=10, inserendo Early Stopping con patience=10. Si può osservare che, rispetto al modello base (Tabella 41) nel caso dell'Early Stopping (Tabella 42) si ha una f1-score più alta e dunque questo è il miglior modello, soprattutto grazie al miglioramento nella classificazione dei record per la classe 1.

Epochs = 100, batch_size = 10, No Regulation Model			
Class	**precision**	**recall**	**f1-score**
0	0.99	0.99	0.99
1	0.97	0.97	0.97
accuracy	0.989		

Tabella 41: Risultati sul test per Deep Neural Network

Epochs = 100, batch_size = 10, Early Stopping patience =10			
Class	**precision**	**recall**	**f1-score**
0	0.99	0.98	0.99
1	0.96	0.99	0.98
accuracy	0.9912		

Tabella 42: Risulati sul test per Deep Neural Network nel caso di Early Stopping

Si è provato poi ad applicare una **Dropout Regolarization** ed un **Ridge Regularization**, tuttavia in entrambi i casi i risultati sono stati molto scarsi, dunque non sono stati riportati.

Modello 2 con grid search

In questo secondo modello si sono fissate alcune caratteristiche dell'architettura della DNN: activation function ed hidden_layer_size uguale per tutti gli hidden layer.
Dopodiché, si è svolta una grid search variando i seguenti parametri: n_layers = [1, 2, 3], h_dim = [32, 64, 128], activation = ['relu', 'tanh'], optimizer = ['adagrad', 'adam'].
Il miglior classificatore è risultato essere: {'activation': 'tanh', 'h_dim': 32, 'n_layers': 3, 'optimizer': 'adam'}. I risultati sul test sono descritti nella Tabella 43.

	precision	**recall**	**f1-score**
0	0.99	0.98	0.99
1	0.96	0.99	0.98
accuracy	0.9908		

Tabella 43: Risultati sul test per la miglior configurazione ottenuta dalla grid search

Rispetto al modello 1 senza regolarizzazione, il modello 2 ha una recall per la classe 1 più alta, che d'altra parte si era riusciti ad aumentare anche un Early Stopping nell'altro modello. Inoltre, cala lievemente la recall per la classe 0, ma aumenta la precision. In conclusione, rispetto al modello senza Early Stopping, questo risulta essere migliore.

Ensemble Classifiers

Random Forest

Per questo classificatore è stata utilizzata una grid search variando i parametri min_samples_leaf, min_samples_split ed il numero di estimator. Già con due DT i risultati sono ottimi, anche se leggermente inferiori rispetto al singolo Decision Tree. Tuttavia, i migliori risultati si ottengono con un numero di estimator pari a 100.
In Figura 17 si riporta un grafico che mostra le features'importance del Random Forest, in cui nuovamente *Light* ed in secondo luogo *Temperature* sono gli attributi con maggior importanza.

	precision	**recall**	**f1-score**
0	0.99	0.99	0.99
1	0.97	0.99	0.98
accuracy	0.9935		

Tabella 44: Risultati sul test per il Random Forest

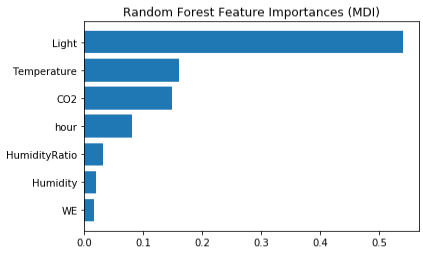

Figura 17: Features'importance nel Random Forest

Dalla Tabella 44, che mostra i risultati sul test, si possono osservare buoni valori di recall e una leggera flessione della precision rispetto al classico Decision Tree (vedi Tabella 6). Il Random Forest risulta essere quindi un buon classificatore.

AdaBoost

L'algoritmo di base scelto è il Decision Tree, in quanto si presta particolarmente bene ad essere utilizzato dall'algoritmo AdaBoost. Gli iperparametri utilizzati per l'albero sono quelli individuati dalla grid search eseguita nella task 1: min samples leaf=1, min samples split=5, max_depth= None. Poiché il Decision Tree generato utilizzando questi parametri presenta già da solo un bassissimo error rate, pari a 0.007, non ci si aspetta che il miglioramento dato dall'ensemble sia particolarmente significativo.
È stata impostata una grid search (3-fold) per trovare i miglior iperparametri: numero di base classifier da utilizzare e learning rate, provando anche combinazioni con un alto numero di base classifier (fino a 1000) ed una bassa learning rate. I migliori parametri trovati (in particolare considerando l'error rate come misura) sono i seguenti:

- numero di base estimator=350;
- learning rate=1.

Nella Tabella 45 sono riportati i risultati sul test e nella Figura 18 si rappresenta la variazione dell'error rate all'aumentare del numero di classificatori base utilizzati. Il valore dell'error rate per il modello finale scelto è di 0.006, quindi inferiore rispetto a quello del singolo Decision Tree.

	precision	recall	f1-score
0	0.99	0.99	0.99
1	0.98	0.99	0.98
accuracy	0.9940		

Tabella 45: Risultati sul test per AdaBoost

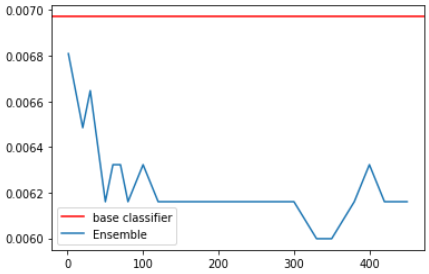

Figura 18: Error rate al variare del numero di base estimator per AdaBoost

Per testare meglio le potenzialità di AdaBoost si è deciso di applicarlo a un caso in cui il Decision Tree ha prodotto risultati leggermente inferiori: il **caso del dataset sbilanciato**. L'albero selezionato è quello risultato migliore nella Task 1 (imbalanced). In questo caso, il numero di base estimator utilizzati è pari a 250 e learning rate uguale ad 1. La Tabella 46 presenta il risultato di questo modello eseguito sul test.

	precision	recall	f1-score
0	0.99	0.99	0.99
1	0.94	0.94	0.94
accuracy	0.9960		

Tabella 46: Risultati sul test di AdaBoost per il caso del dataset sbilanciato

Confrontandolo con il risultato del singolo Decision Tree (Tabella 27) si può notare come la capacità di predizione per la classe 1 sia decisamente migliorata. L'error rate passa da 0.072 con il base classifier (Decision Tree imbalanced) a 0.004 con AdaBoost.

BAGGING

Per prima cosa è stato necessario individuare il numero migliore di n_estimator: attraverso una grid search sono stati provati diversi valori in un range da 10 a 300 ed il parametro migliore è risultato 121. L'utilizzo di classificatori stabili è meno vantaggioso poiché l'ensemble non aiuterà a migliorare le prestazioni di generalizzazione, per cui è stato utilizzato come base_estimator il Decision Tree, in particolare quello risultato migliore nella Task 1. La performance sul test è descritta nella Tabella 47.

Rispetto agli esiti del Decision Tree (vedi Tabella 6) il BAGGING ha aiutato ad ottenere un leggero incremento di tutte le metriche, quindi risulta essere migliore.

	precision	recall	f1-score
0	1	0.99	1
1	0.98	0.99	0.99
accuracy	0.9931		

Tabella 47: Risultati sul test per BAGGING

Task 3: Time Series Analysis, Forecasting and Classification

Motif discovery

Per l'individuazione di motif è stata utilizzata la TS risultante dalle 8100 rilevazioni nel file *training.txt*. L'analisi è stata effettuata sia sulla TS originale, sia su quella sottoposta a trasformazioni quali eliminazione del trend e riduzione del noise, tuttavia sono presentati solo i risultati ottenuti sull'originale essendo più significativi.
Sia per l'estrazione di motif che di anomalie è stata utilizzata la matrix profile con una sliding window uguale a 360, ovvero il numero di minuti in 6 ore. Questo ha permesso di individuare motif più significative rispetto ad altri tentativi con window uguale a 60, inoltre la stessa lunghezza temporale è stata mantenuta anche per la shapelet discovery.
La feature scelta per l'analisi è *Temperature* e, dando un limite massimo di 10, sono stati individuati 7 motif. Le somiglianze riscontrate riguardano soprattutto la struttura dei picchi e degli avvallamenti che si ripetono per via della periodicità della TS, come si può osservare dalla Figura 19.
Utilizzando la medesima matrix profile sono state individuate delle anomalie, in particolare fissando la zona di esclusione pari a 60 le anomalie individuate sono 3, come mostra la Figura 20.

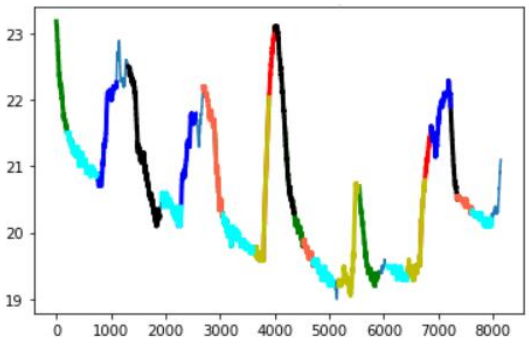

Figura 19: Rappresentazione dei motif individuati

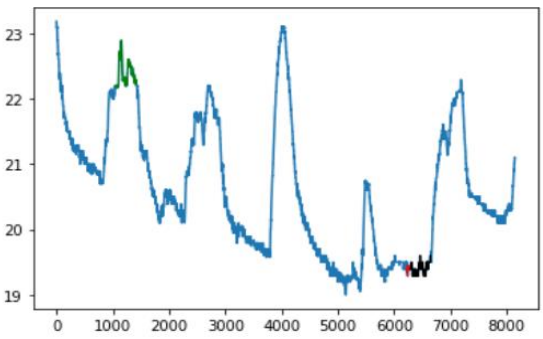

Figura 20: Rappresentazione delle anomalie individuate

Shapelet Discovery

In questa fase è stato creato un dataset di TS a partire dalla TS originata dal file *training.txt*, considerando l'attributo *Temperature*. Il dataset finale ha un numero di TS pari a 22, di lunghezza uguale a 360, ovvero 6 ore. Questa scelta è dovuta alla necessità di ottenere risultati significativi mantenendo allo stesso tempo una sufficiente quantità di dati: utilizzando, ad esempio, una lunghezza pari a 60 si sarebbe ottenuta una quantità più adeguata di TS, ma all'interno della medesima ora non vi sarebbero stati cambiamenti significativi nell'andamento della temperatura per ricavarne analisi interessanti.

Come label si è mantenuta la classificazione occupato/non occupato calcolata come moda dei valori presenti in ciascuna TS.
Le shapelet individuate sono 3, di lunghezza 36, numero risultato ideale per la struttura del nostro dataset.

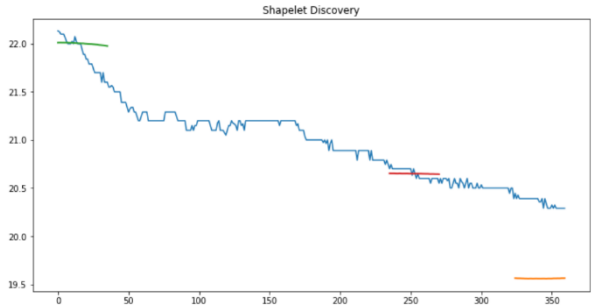

Figura 21: Shapelet individuate e il loro allineamento rispetto a una TS del dataset

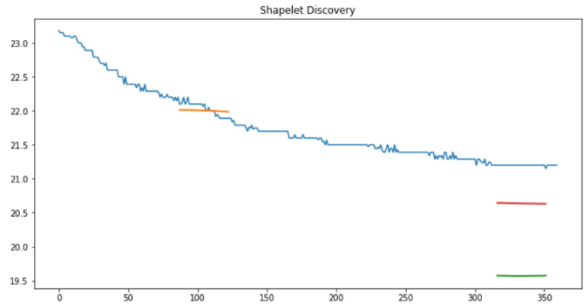

Figura 22: Shapelet individuate e il loro allineamento rispetto a una TS del dataset

Come si può notare nella Figura 21 e Figura 22, le shapelet si differenziano non tanto per la forma, quanto per il loro valor medio. Il motivo per cui sono dunque poco significative è dovuto al fatto che si hanno solamente 22 esempi su cui costruire il modello: una quantità piccola che non porta ad individuare precisamente delle sottosequenze che rappresentino le classi.

Clustering

Per questa fase è stato utilizzato un dataset di TS univariate, di lunghezza pari ad un'ora, originate da feature diverse. La combinazione risultata più significativa, presentata in seguito, è costituita da *Temperature*, *Light* e *CO2*, per un totale di 405 TS.
Dopo aver applicato una normalizzazione MinMax, i tipi di clustering su cui è stata effettuata l'analisi sono: Shape Based, Feature Based, Compression Based ed Approximation Based.

Nel caso dello **Shape Based** attraverso l'Elbow Method si è ricercato il numero di cluster ottimale: utilizzando la distanza euclidea (vedi Figura 23) il numero ideale risulta essere compreso tra 2 e 5, mentre nel caso del Dinamic Time Warping è tra 4 e 6.

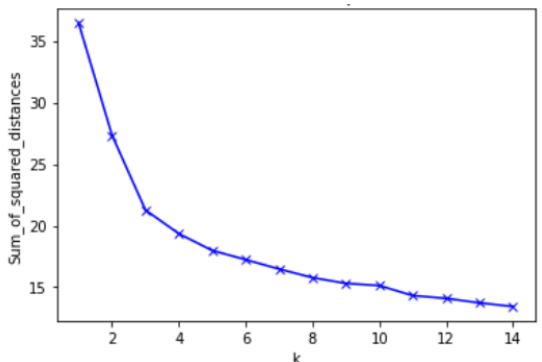

Figura 23: Elbow Method per individuare il miglior k nel caso della distanza euclidea

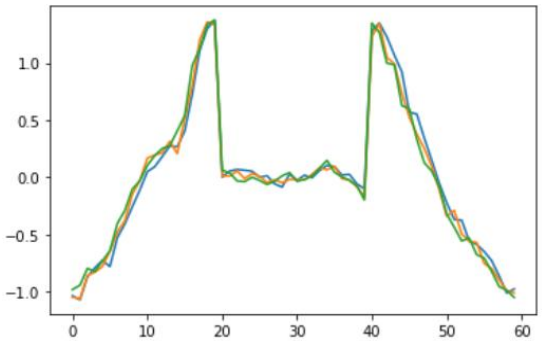

Figura 24: Centroidi individuati per il clustering Shaped Based, distanza euclidea

In entrambi i casi, i cluster generati hanno centroidi simili (Figura 24, caso distanza euclidea), il che significa che la media intracluster è simile tra i diversi cluster, infatti il valore della silhouette è pari a 0.227: questo indica che i cluster non sono ben separati.

Nel caso del **Feature Based** il clustering viene fatto su una lista di "features", calcolate sul dataset senza normalizzazione, la quale viene applicata successivamente sui risultati. In questo caso l'Elbow Method

mostra un numero di cluster ideale più ridotto, ovvero 2 o 3. Con k=3 si ottengono i centroidi mostrati nella Figura 25.

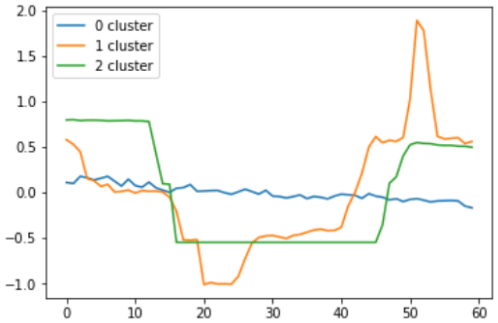

Figura 25: Centroidi per Feature Based clustering

Il contenuto dei cluster risulta essere sbilanciato: infatti molti sono i dati che appartengono al cluster 0 (383) e pochi quelli appartenenti al cluster 1 e 2 (14 e 8). In questo caso, il valore della silhouette è di 0.085 e si può dunque concludere che il metodo non è ottimale. Anche con un numero maggiore di cluster infatti il risultato non cambia.

Nel caso del **Compression Based**, i risultati non sono significativi. Infatti, nonostante siano state utilizzando diverse configurazioni di parametri per l'algoritmo DBSCAN i risultati, sebbene variando nel numero di noise points individuati, consistono sempre in un solo cluster.

Infine, l'**Approximated Clustering** è stato testato con approssimazione PAA variando il numero di segmenti. È stato notato che per un adeguato numero di segmenti (intorno a 20), i centroidi risultano simili tra loro, come già visto per lo Shape Based. Per ottenere centroidi che si differenzino tra loro è necessario ridurre il numero di segmenti a 5, questo tuttavia comporta un'elevata perdita di informazioni. In ogni caso la silhouette del clustering ottenuto è in linea con quella dello Shape Based (0.227).
In conclusione, il clustering su questo specifico dataset non riporta risultati soddisfacenti o significativi.

Forecasting

Il dataset utilizzato è il *datatest2.txt* nel quale sono presenti 9700 rilevazioni. Questa scelta è dettata dal fatto che tra i 3 dataset forniti sono presenti periodi lunghi poche ore in cui non si hanno registrazioni e dunque si è deciso di procedere tenendo in considerazione quello più recente.
Poiché alcune misurazioni sono state effettuate al 59esimo secondo ed altre al 60esimo si è proceduto a regolarizzare la frequenza ad un minuto esatto.
La TS considerata è quella contenente i valori di *Temperature* (Figura 27), la cui decomposizione è rappresentata nella Figura 26. Da una prima analisi del plot è possibile notare la presenza di una componente di stagionalità giornaliera e un avvallamento nella curva del trend.

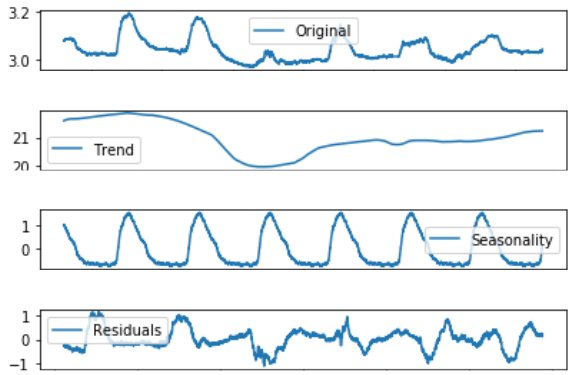

Figura 26: Decomposizione della TS

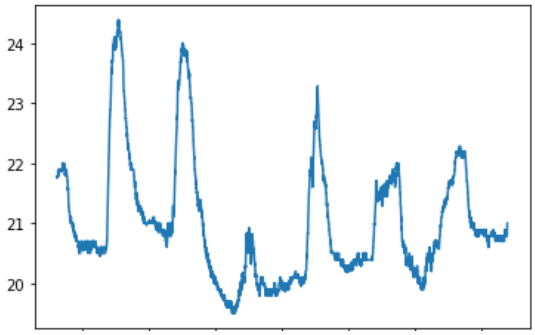

Figura 27: TS per feature Temperature

Per valutare la **stazionarietà** è stato effettuato il Dickey-Fuller test, i cui risultati, mostrati in Figura 28, hanno portato a rigettare l'ipotesi null per il critical value del 5%, ma non dell'1%. Inoltre, poiché l'ACF presenta una graduale discesa e anche la PACF ha valori alti per diversi lag, si è deciso di applicare delle trasformazioni, in particolare la log mean differencing.
Per effettuare il forecasting il dataset è stato diviso in training (70%) e test (30%).
Il Simple Exponential Smoothing ha prodotto risultati non buoni, è stato quindi utilizzato **Exponential Smoothing** senza trend (Figura 29), ma con seasonal period di 1440 (numero di minuti in un giorno).
Valutando il modello tramite varie metriche si può concludere che non è molto significativo, poiché il valore del coefficiente di determinazione è piuttosto basso (0.338), e soprattutto il MAPE è uguale a 1.911.

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.227136
p-value                          0.018467
#Lags Used                      38.000000
Number of Observations Used    9713.000000
Critical Value (1%)              -3.431023
Critical Value (5%)              -2.861838
Critical Value (10%)             -2.566928
```

Figura 28: Risultati del Dickey-Fuller test per la TS originale

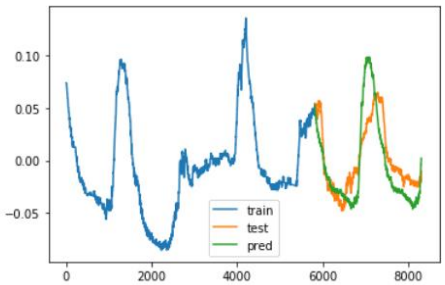

Figura 29: Risultati del forecasting con Exponential Smoothing

Per applicare **ARIMA** e **Seasonal ARIMA** è stato fatto un resampling della TS (a causa dei limiti computazionali), passando da una rilevazione per minuto ad una ogni 30 minuti. La TS risultante ha un problema di trend, ma soprattutto di seasonality, come si può riscontrare dell'ACF plot in Figura 30.
Il Dickey-Fuller test ha riportato risultati analoghi a quello della TS senza resampling: si è deciso quindi di applicare nuovamente la log mean differencing. Dopo questa trasformazione, i risultati del Dickey-Fuller test, mostrati in Figura 31, portano a rigettare l'ipotesi nulla anche per critical value dell'1% e si può dunque concludere che la TS è stazionaria.

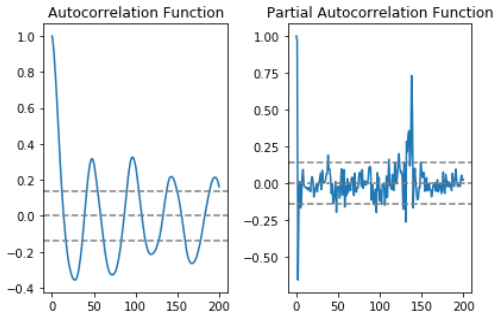

Figura 30: ACF e PACF plot per la TS resampled

```
Results of Dickey-Fuller Test:
Test Statistic                  -4.708793
p-value                          0.000081
#Lags Used                       5.000000
Number of Observations Used    273.000000
Critical Value (1%)              -3.454533
Critical Value (5%)              -2.872186
Critical Value (10%)             -2.572443
```

Figura 31: Risultati del Dickey-Fuller test della TS resampled e trasformata

Poiché ARIMA non ha dato risultati soddisfacenti ma, al contrario produce modelli con R2 negativo, si è deciso di procedere con Seasonal ARIMA.
Per prima cosa sono stati **settati i valori di p, d e q**: dopo varie prove, si è deciso di settare d a zero; per quanto riguarda p e q, osservando il plot della serie differenced (Figura 32 e Figura 33) si può notare che l'ACF ha un andamento sinusoidale e che il PACF ha valori alti fino a lag uguale a 2 e dunque si è deciso di settare p=2 e q=0.
Successivamente, si è passati al **settaggio di P, D, Q**: per quanto riguarda D, poiché è presente stagionalità, si è deciso di settarlo ad 1, cioè di utilizzare un seasonal differencing; il settaggio di P e Q sulla base della stima a partire dai grafici ACF e PACF è stata più difficoltosa, tuttavia dopo varie prove la scelta finale è ricaduta su P=2 e Q=0.

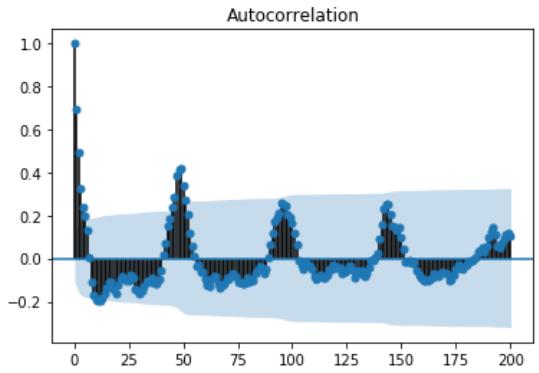

Figura 32: ACF plot della serie differenced

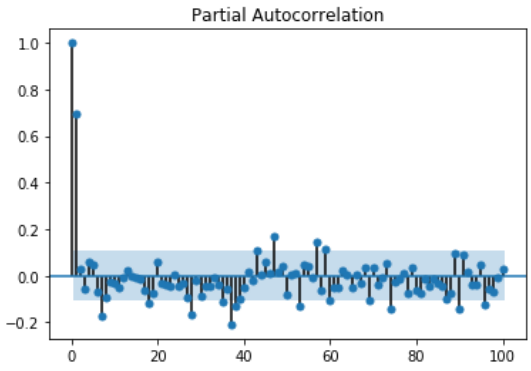

Figura 33: PACF della serie differenced

Il modello finale composto con i seguenti parametri **order=(2, 0, 0), seasonal_order=(2, 1, 0, 48)**, ha dato i risultati sul test rappresentati nella Figura 34 e Figura 35.

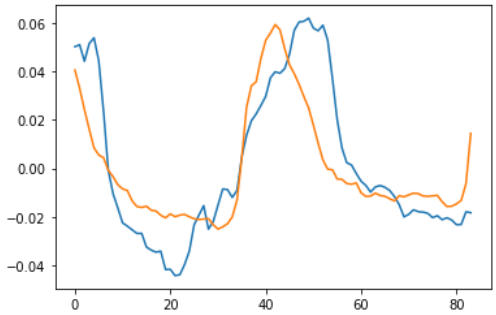

Figura 34: Risultato del forecasting con seasonal ARIMA, test e forecasting a confronto

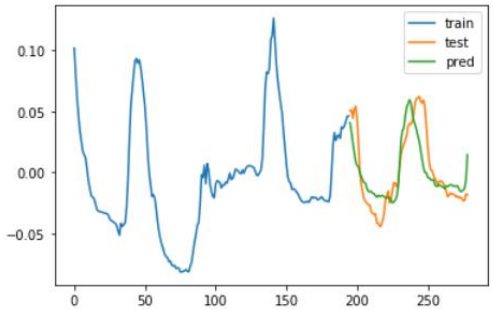

Figura 35: Risultato del forecasting con seasonal ARIMA

Dalla valutazione di questo modello è stato ottenuto un R2 pari a 0.628, che è il risultato migliore per questa metrica rispetto a tutte le configurazioni provate; tuttavia il MAPE è leggermente aumentato (3.873) rispetto al risultato dell'Exponential Smoothing.

Classification

Dal file *training.txt* è stato ottenuto un dataset di 135 TS lunghe un'ora, da utilizzare come training. La stessa cosa è stata effettuata con il *datatest2.txt*, dove il 20% è stato utilizzato come Validation Set (prima parte) e l'80% come Test Set (seconda parte).
L'obiettivo della task è quello di classificare ogni TS come appartenente al giorno oppure alla notte. Quindi, il label creato è *DayTime*, uguale a 0 qualora sia un'ora notturna (orario fra le 7.00 p.m. e le 6.59 a.m.) e uguale a 1 nel caso sia diurna (dalle 7.00 a.m. alle 6.59 p.m.).

Classification univariate Temperature e Light

Sono stati dapprima cercati diversi modelli nel caso di dataset univariate utilizzando l'attributo *Temperature* e poi *Light*. Nella Tabella 48 si riportano i risultati dell'f1-score della classe 0 e 1 per la miglior configurazione di ciascun classificatore utilizzato.
Dunque, si può osservare che il modello migliore non solo in termini di f1-score, ma anche di accuracy, è il **Feature Based Decision Tree** con l'attributo *Light*. Questo risultato non sorprende in quanto si può immaginare che le feature calcolate, come ad esempio il valore medio della luce, anche in ufficio, aiutino a separare nettamente le ore diurne da quelle notturne.
Alcuni classificatori però hanno performance significativamente migliori con l'attributo *Temperature*: lo **Shapelet Classifier** performa meglio soprattutto sulla classe 1; lo **Shapelet Distance Base DecisionTree**, che nel caso dell'attributo *Light* ha portato a pessimi risultati; il **TS Classifier Decision Tree** che,

nonostante non ottenga risultati eccellenti, ha sicuramente performance migliori rispetto alla classificazione con *Light*, che ha riportato pessimi risultati.
Un ulteriore classificatore che ha ottenuto buone performance sull'attributo *Light* è il TSClassifier KNN sia con metrica minowsky che la dtw_sakoechiba.

Model	Temperature		Light	
	Class 0	Class1	Class 0	Class 1
Shapelet Classifier	0.83	0.74	0.83	0.71
Shapelet-distances-based KNN	0.75	0.59	0.94	0.93
Shapelet-distances-base Decision Tree	0.71	0.63	0.00	0.65
Feature Based DecisionTree	0.80	0.61	0.96	0.95
TS Classifier KNN minowsky	0.77	0.51	0.94	0.93
TS Classifier DecisionTree	0.75	0.64	0.00	0.65
TS Classifier KNN dtw_sakoechiba	0.77	0.51	0.94	0.93

Tabella 48: Risultati dell'f1-score della classe 0 e 1 per la miglior configurazione di ciascun classificatore utilizzato

Infine, sono stati applicati due tipi di Neural Networks: CNN e LSTM.
Per la CNN sono stati effettuati vari tentativi variando il numero di convolutional layer e tentando diversi valori di dropout. Quest'ultimo tuttavia ha apportato dei peggioramenti dovuti al fatto che la rete non è in overfitting e dunque comporta solo una perdita di informazioni. Aumentando il numero di convolutional layer da tre a quattro si è notato un incremento della recall per la classe 1, ma una riduzione della medesima per la classe 0, mentre la precision è solo diminuita (Tabella 49 e Tabella 50).
Il modello finale selezionato è composto da tre convolutional layer con un numero di filtri rispettivamente pari a 12, 32 e 64; kernel_size che diminuisce per ogni layer (8, 5 e 3); mini_batch_size uguale a 13; ReLu come activation function per tutti e i tre layers; nessuna Dropout Regularization.

CNN con 3 convolutional layer			
	precision	recall	f1-score
0	0.67	0.99	0.8
1	0.97	0.46	0.63
accuracy	0.7384		

Tabella 49: Risultati CNN con 3 convolutional layer

CNN con 4 convolutional layer			
	precision	recall	f1-score
0	0.67	0.83	0.74
1	0.75	0.54	0.63
accuracy	0.70		

Tabella 50: Risultati CNN con 4 convolutional layer

Nonostante i numerosi tentativi, l'LSTM classifier non ha portato a buoni risultati, probabilmente per il basso numero di record utilizzati come training nella classificazione.

Classification con Multivariate Dataset

Il dataset utilizzato è quello contenente tutti e cinque gli attributi ed è formato da 135 TS nel Training Set, ciascuna lunga 60 (ovvero un'ora).
È stata utilizzata la libreria pyts, che consente di utilizzare i classificatori tradizionali. È stato inizialmente usato il Decision Tree Classifier, che tuttavia classifica correttamente solo le TS con *DayTime*=1. Con il KNN e metric=dtw_sakoechiba la f1-score è risultata pari a 0.75 per la classe 0 e 0.62 per la classe 1. In generale quindi, le performance dei classificatori tradizionali su questo dataset sono risultati peggiori rispetto a quelle ottenute sull'univariate.
Successivamente, sono stati sviluppati nuovi modelli di LSTM Classifier, CNN 1D e CNN 2D, i cui valori di f1-score sono riportati nella Tabella 51.

Model	LSTM		CNN1		CNN2	
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
f1-score	0.83	0.75	0.85	0.80	0.83	0.81

Tabella 51: Risultati dei modelli LSTM Classifier, CNN 1D e CNN 2D

In conclusione, la classificazione migliore si ha con il dataset univariate che utilizza *Light* e sul multivariate il modello alternativo migliore è CNN 1D.

Task 4 - Sequential Pattern Mining

Questa task è stata svolta sul file *datatest2.txt*, essendo il più lungo, da cui è stato ricavato un dataset con 162 TS, ciascuna di 60 rilevazioni e mantenendo tutti e cinque gli attributi numerici.
Si è proceduto con una discretizzazione del dataset utilizzando la SAX con n_paa_segments=20, n_sax_symbols=5. Il motivo per cui è stato scelto un numero di simboli così basso è dovuto al fatto che il numero di sequential pattern significativi sarebbe stato molto basso in quanto la presenza di questi ultimi diminuisce all'aumentare della varietà dei possibili item. Nella Figura 36 si riporta un esempio per le prime due TS con l'attributo *HumidityRatio* nel caso precedente e successivo alla discretizzazione.
È stato dunque creato il dataset di TS dove ogni transazione ha lunghezza 20, ovvero contiene 20 tuple (formato necessario per la libreria PrefixSpan). Ciascuna tupla contiene cinque valori corrispondenti agli attributi nel seguente ordine: *Temperature*, *Humidity*, *Light*, *CO2*, *HumidityRatio*.
Fissato un min_support uguale a 3, sono stati estratti i sequential pattern e sono stati filtrati quelli con lunghezza uguale a 4. Da questa ricerca ne sono risultati 18 di cui solamente uno avente support uguale a 4. Ne riportiamo alcuni nella Tabella 52.

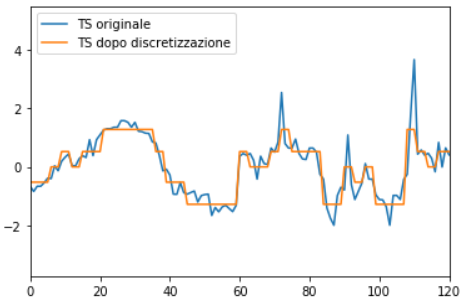

Figura 36: Rappresentazione dell'attributo HumidityRatio con e senza discretizzazione

Support	Sequential pattern
3	(0,0,0,0,0)(0,0,0,0,0)(0,0,0,0,0)(0,0,0,0,0)
3	(3,3,2,2,3)(3,1,2,1,1)(0,0,2,1,0)(0,0,2,4,0)
3	(2,4,2,1,4)(2,4,2,0,4)(2,1,2,1,1)(2,0,2,3,0)
3	(14,2,0,4)(1,3,2,0,3,)(1,1,2,2,1)(2,0,2,4,0)
4	(1,4,2,0,4)(1,3,2,0,3)(1,1,2,2,1)(4,0,2,4,0)
3	(1,4,2,0,4)(1,3,2,1,3)(1,1,2,2,1)(2,0,2,4,0)
3	(0,0,1,0,0)(0,0,1,0,0)(3,4,2,4,4)(4,4,2,4,4)
3	(0,0,1,0,0)(0,0,1,0,0)(0,0,0,0,0)(3,4,2,4,4)

Tabella 52: Sequential pattern con min support = 3

Non risulta semplice fare un'analisi dei risultati, in quanto si sta parlando di item che sono numeri, quindi difficilmente interpretabili. Tuttavia, si può osservare che il primo risultato, mostrato nella Tabella 52, presenta 0 per tutti gli attributi e per tutti i timestamp: ciò potrebbe significare che rappresenta un momento notturno, in cui i valori di *Temperature*, *CO2*, *Light* ecc. sono bassi. Allo stesso modo, si può notare che nel penultimo SP si ha una serie di valori bassi seguiti da timestamp successivi in cui i valori aumentano: probabilmente rappresenta il passaggio dalla notte alla mattina in cui l'ufficio si popola di persone. Infine, si riportano per completezza i Top-10 per supporto, anche se poco significativi essendo di lunghezza uguale a 1 (Tabella 53, SP con lo stesso support sono riportati nella medesima riga).

Support	Sequential pattern
21	(1,1,2,1,1)
20	(2,2,2,3,2)
19	(1,1,2,2,1); (2,1,2,2,1); (2,2,2,2,2)
18	(2,2,2,1,2); (2,3,2,3,3)
17	(2,3,2,1,3); (2,3,2,2,3)
15	(1,2,2,1,2)

Tabella 53: Top-10 sequential pattern ordinati per support

Task 5 - Outlier Detection and Explainability

Outlier Detection

Il dataset utilizzato per questa task è il risultato della concatenazione dei tre file forniti. Sono stati presi in considerazione solo gli attributi numerici (quindi esclusi *date* ed *Occupancy*) a cui è stata applicata la Standard Scaler Normalization. Questa decisione deriva dal fatto che i metodi applicati per la ricerca di anomalie sono basati per lo più su distanze e il nostro dataset presenta dati in diverse scale. I tre metodi utilizzati sono: LOF, ABOD e KNN.
Per ogni algoritmo sono stati analizzati i 12 punti (circa l'1%) con *outlierness score* più alto.
Di seguito si riportano i risultati, riassunti nelle tabelle (tutti i decimali sono approssimati per motivi di spazio).

LOF (Local Outlier Factor)

Per prima cosa, si è definito il numero di nearest neighbors (K) da considerare. Dopo varie ricerche, considerando anche il numero di outliers ricavati al variare di K, la scelta è ricaduta su 20, valore al di sopra del quale il numero di outliers individuati dall'algoritmo rimane costante. Il numero di outliers (label uguale a -1) trovati è 1194 su 20560 punti.
I Top-12 outliers della LOF sono rappresentati nella Tabella 54.

date	LOF_label	LOF_score	ABOD_label	ABOD_score	KNN_label	KNN_score
2015-02-17 08:38:00	-1	-23.702	1	0.00	1	0.483
2015-02-04 09:40:59	-1	-18.53	1	-0.014	1	4.455
2015-02-12 08:19:00	-1	-14.914	1	-0.122	1	1.172
2015-02-04 09:40:00	-1	-12.559	1	-0.071	1	3.235
2015-02-12 03:43:59	-1	-12.309	1	-50.589	1	0.545
2015-02-07 09:42:59	-1	-11.106	1	-0.033	1	4.382
2015-02-12 03:42:59	-1	-11.042	1	-117.67	1	0.484
2015-02-07 09:42:00	-1	-10.601	1	-0.018	1	4.79
2015-02-18 09:10:59	-1	-10.544	1	-2.463	1	1.343
2015-02-07 04:17:00	-1	-10.088	0	-24720.216	0	0.049
2015-02-08 22:02:59	-1	-9.773	1	-0.058	0	0.098
2015-02-07 09:43:59	-1	-8.699	1	-0.001	1	1.759

Tabella 54: Top-12 outliers della LOF a confronto con gli altri metodi di outliers detection

Si può osservare che la maggior parte degli outliers individuati da questo metodo viene ritrovata anche dagli altri, eccetto per i record evidenziati: in particolare, un punto è stato etichettato come anomalia solo da LOF. Infine, molti degli outliers risultanti da questo algoritmo riguardano rilevazioni effettuate di mattina.

ABOD (Angle Based Outlier Detection)

Per questo metodo i parametri scelti sono i seguenti: contamination=0.1, n_neighbors=10, method=fast ed il numero di outliers individuato è 1967.
Nella Tabella 55 sono riassunti i Top-12 outliers, che presentano tutti uno score molto vicino a 0. Come evidenziato dalla tabella, solo 4 dei punti che sono outliers per ABOD lo sono anche per LOF e KNN. In questo modello si evidenzia il fatto che la maggior parte degli outliers individuati fa riferimento al 07-02 e che, in particolare, corrisponde a timestamp consecutivi.

date	LOF_label	LOF_score	ABOD_label	ABOD_score	KNN_label	KNN_score
2015-02-07 06:04:00	1	-0.977	1	0	0	0.006
2015-02-07 05:04:00	1	-0.977	1	0	0	0.006
2015-02-07 06:21:00	1	-0.977	1	0	0	0.006
2015-02-07 06:22:00	1	-0.977	1	0	0	0.006
2015-02-07 06:33:00	1	-0.977	1	0	0	0.006
2015-02-07 06:35:00	1	-0.977	1	0	0	0.006
2015-02-07 07:31:00	1	-0.977	1	0	0	0.006
2015-02-07 07:45:00	1	-0.977	1	0	0	0.006
2015-02-17 08:38:00	-1	-23.702	1	0	1	0.483
2015-02-07 09:43:00	-1	-8.699	1	-0.001	1	1.759
2015-02-07 09:40:00	-1	-5.297	1	-0.001	1	1.014
2015-02-13 09:49:00	-1	-7.872	1	-0.002	1	1.392

Tabella 55: Top-12 outliers dell'ABOD a confronto con gli altri metodi di outliers detection

KNN

I parametri utilizzati per creare il modello sono i seguenti: contamination=0.1, metric='minkowski', p=2, n_neighbors=20 ed il numero di outliers risultato è di 1906.
Dalla Tabella 56, contenente i Top-12 outliers, si può osservare che i medesimi punti sono rilevati anche dagli altri metodi. Anche in questo caso le anomalie individuate fanno riferimento esclusivamente ad orari mattutini, in particolare intorno alle ore nove.

date	LOF_label	LOF_score	ABOD_label	ABOD_score	KNN_label	KNN_score
2015-02-07 09:42:00	-1	-10.601	1	-0.018	1	4.79
2015-02-04 09:40:59	-1	-18.53	1	-0.014	1	4.455
2015-02-07 09:42:59	-1	-11.106	1	-0.033	1	4.382
2015-02-12 09:47:00	-1	-7.264	1	-0.033	1	4.016
2015-02-04 09:40:00	-1	-12.559	1	-0.071	1	3.235
2015-02-12 09:46:00	-1	-6.456	1	-0.073	1	3.157
2015-02-04 09:42:00	-1	-8.079	1	-0.087	1	2.32
2015-02-18 09:19:00	-1	-2.041	1	-0.152	1	2.091
2015-02-07 09:43:59	-1	-8.699	1	-0.001	1	1.759
2015-02-18 09:17:59	-1	-5.861	1	-7.641	1	1.717
2015-02-12 09:48:00	-1	-3.403	1	-0.113	1	1.646
2015-02-18 09:14:00	-1	-6.876	1	-590.582	1	1.624

Tabella 56: Top-12 outliers del KNN a confronto con gli altri metodi di outliers detection

Conclusione

Per valutare la presenza di elementi comuni è stata fatta una intersezione tra le Top-12 di tutti e tre i metodi. Il risultato è un solo dato che è riportato nella Tabella 57.

date	LOF_score	ABOD_score	KNN_score
2015-02-07 09:43:59	-8.699	-0.001	1.759

Tabella 57: Dato risultante dall'intersezione delle Top-12 di tutti i metodi applicati

Infine, per rendere il confronto più evidente, sono stati realizzati degli scatterplot (Figura 37, Figura 38). Come è possibile osservare, per LOF e KNN vengono generati risultati simili, mentre si è notato che quello che si discosta maggiormente, come già visto nelle tabelle riassuntive, è ABOD. Una possibile motivazione di questo comportamento potrebbe essere giustificata dal fatto che i dati sono generati da Time Series e quindi, come è possibile osservare nella Figura 39, non tutti i dati si clusterizzano in una forma globulare. In alcuni casi si può osservare che una serie di punti formano "un path" e dunque i nearest neighbors troveranno angoli simili tra i punti stessi non essendo "circondati" dai vicini: questi verranno considerati da ABOD come outliers nonostante non lo siano.

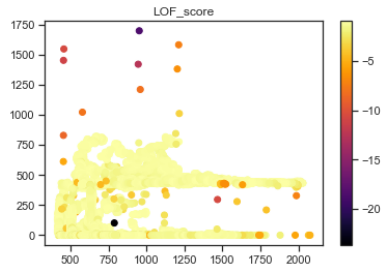

Figura 37: Scatterplot della LOF con gli attributi CO2 e Light

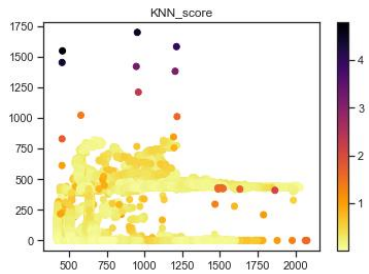

Figura 38: Scatterplot del KNN con gli attributi CO2 e Light

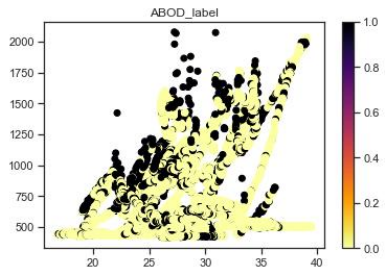

Figura 39: Scatterplot dell'ABOD con gli attributi CO2 e Humidity

Explainability

Per questa task è stato utilizzato un semplice **Inspection Model Explainer** per cercare di analizzare come prendessero le loro decisioni due dei classificatori più complessi: non linear-SVM e Multi Layer Perceptron. Il metodo utilizzato è quello del Partial Dependency Plot.

Per **SVM** i risultati ottenuti sono piuttosto ovvi per quanto riguarda gli attributi *Light*, *CO2* e *Temperature*: come si può osservare dalla Figura 40, la probabilità che la classe predetta sia 1 aumenta all'aumentare di questi valori. Invece, per quanto riguarda *Humidity* e *HumidityRatio* la probabilità è alta per valori bassi e alti, mentre diminuisce per i valori medi (Figura 41).

Per il **MLP** gli attributi presentano un trattamento diverso: solo i plot di *Light* e *CO2* sono crescenti, mentre per *Temperature*, come si può notare dalla Figura 42, la probabilità è decrescente, tuttavia è stato osservato che la sua variazione è minima, quindi molto probabilmente questo attributo non è di grande importanza per questo classificatore.

Infine, è stato notato che per entrambi i classificatori avanzati l'attributo con la variazione di probabilità più alta è *Light*, questo è coerente con tutte le considerazioni fatte finora nell'analisi.

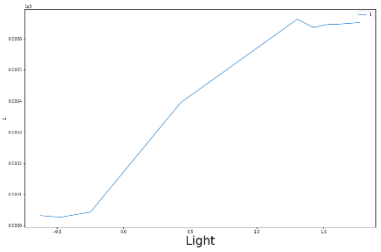

Figura 40: Probabilità di predire la classe 1 per l'attributo Light nell' SVM

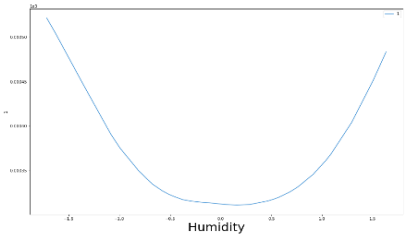

Figura 41:Probabilità di predire la classe 1 per l'attributo Humidity nell' SVM

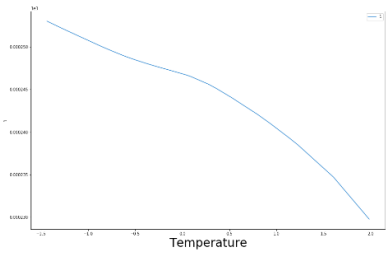

Figura 42: Probabilità di predire la classe 1 per l'attributo Temperature nel MLP