Università di Pisa

Relazione progetto di Data Mining 2

Occupancy Detection

Biviano Matteo
Currao Federica
Scalisi Marta

Anno Accademico 2020/2021

Indice

1 Introduzione

L'obiettivo dello studio è l'analisi di un problema di classificazione binaria, chiamato "Occupancy Detection", in cui attraverso l'analisi di parametri ambientali, acquisiti tramite sensore, si classifica se una stanza è occupata o non occupata, attraverso il valore target *Occupancy*.

2 Data Understanding & Data Preparation

Il dataset è composto da 20560 record divisi in tre gruppi. La tabella seguente descrive le dimensioni dei gruppi di dati, in termini di numero di record e di attributi, e la percentuale di record per cui la stanza è occupata o non occupata, rispetto all'intero dataset.

	Dimensioni	**Occupancy** (%)	**DateTime** (From/To)
Training	(8143, 7)	21% (Occupancy=1)	2015-02-02 14:19:00
		79% (Occupancy=0)	2015-02-04 10:43:00
Test1	(2665, 7)	64% (Occupancy=1)	2015-02-04 17:51:00
		36% (Occupancy=0)	2015-02-10 09:34:00
Test2	(9752, 7)	21% (Occupancy=1)	2015-02-11 14:48:00
		79% (Occupancy=0)	2015-02-18 09:19:00

Figura 1: Descrizione Dataset

Il dataset è composto da 7 attributi che rappresentano le misurazioni, effettuate ogni minuto, di parametri quali: *Date*, *Temperature*, *Humidity*, *CO2*, *Light*, *HumidityRatio*, *Occupancy*.
In particolare, la reale occupazione della stanza, rappresentata dall'attributo *Occupancy*, è stata registrata attraverso l'uso di una fotocamera digitale.
Dal grafico in Figura [2] è possibile notare che nei giorni festivi la stanza non è occupata, mentre nei giorni feriali, tra le 07:00 e le 18:00, risulta invece esserlo. Quando la stanza è occupata, i sensori registrano un aumento di tutte le misurazioni registate, in particolar modo per quanto riguarda *Light* e *CO2* (forti indicatori di occupazione della stanza).
Al contrario, quando la stanza è libera, si registra una loro riduzione graduale, fatta eccezione per l'attributo *Light*. Attraverso l'analisi di quest'ultimo attributo si può dedurre che la stanza è illuminata anche artificialmente, in quanto, se così non fosse, il grafico di *Light* presenterebbe un andamento gaussiano.
Dalle considerazioni fatte è possibile dedurre che la stanza, oggetto dello studio, è un ufficio.

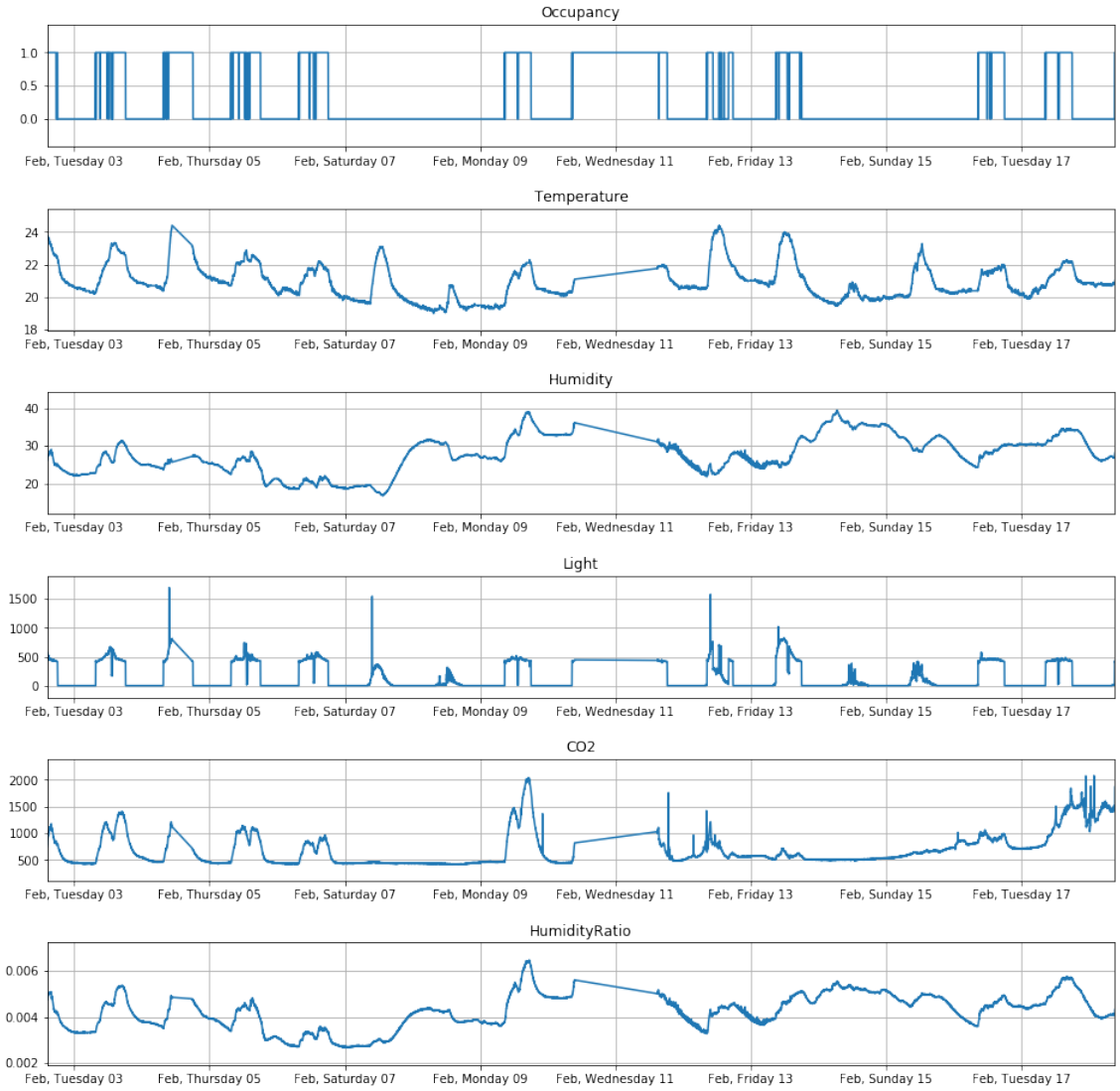


Figura 2: Rappresentazione grafica del dataset

In Figura [3] viene rappresentata la matrice di correlazione degli attributi del dataset. In particolare, si può notare che le variabili *Humidity* e *HumidityRatio* hanno un alto indice di correlazione, pari a 0.96, in quanto quest'ultima è una misura derivata dall'umidità relativa. Di conseguenza, è stato deciso di non considerare l'attributo *Humidity* per le successive analisi.

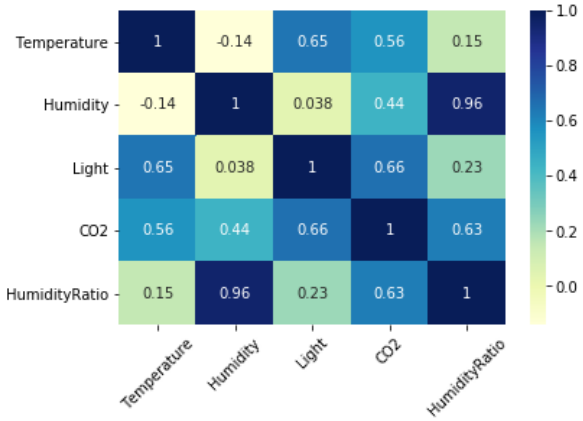


Figura 3: Matrice di correlazione

In seguito all'analisi del dataset, l'attributo *date* è stato sostituito con sei nuove features da esso derivate, ottenute da una codifica di tipo "one hot encoding". Le variabili *Weekday* e *Weekend* indicano rispettivamente se il giorno è feriale o festivo, mentre le altre indicano gli slot

orari della giornata, visibili dettagliatamente in Figura [4]:

Features	Ora di Inizio	Ora di Fine
Day	06:00	12:00
Afternoon	12:00	18:00
Evening	18:00	00:00
Night	00:00	06:00

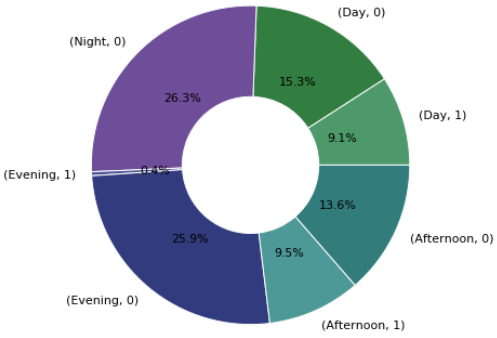


Figura 4: Distribuzione di Occupancy

Inoltre, attraverso la rappresentazione della distribuzione delle variabili (tramite **BoxPlot**) è emerso che l'attributo *Light* presenta alcuni outliers con valori al di sopra di 1000 lx, com'è possibile vedere in Figura [5]. Per evitare che la loro presenza potesse influire nelle analisi successive, sono stati sostituiti attraverso una media mobile a 4 minuti.

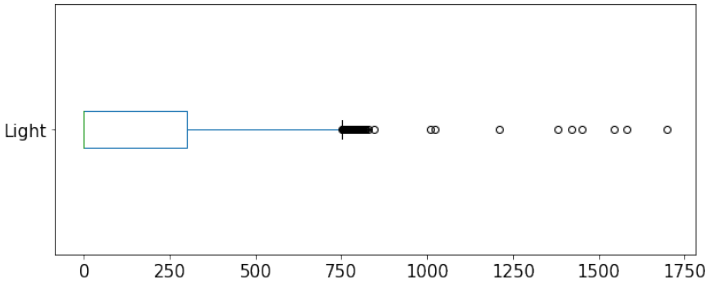


Figura 5: Outlier detection

Infine, i valori degli attributi continui sono stati normalizzati, data la loro differenza di range, con un metodo di tipo **Min_Max**, il quale è stato preferito ad una normalizzazione **Z-scores**, in quanto ha permesso di ottenere un range di valori compreso tra 0 e 1.

3 Basic Classifiers

In questa sezione vengono illustrati gli algoritmi di classificazione e le metodologie utilizzati per prevedere la variabile *Occupancy* e i risultati ottenuti.

Ove possibile, è stata effettuata una fase preliminare di validation, in modo da scegliere gli iperparametri che massimizzassero l'accuracy. In questa fase è stato utilizzato l'algoritmo "GridSearchCV", nel quale il numero di folds è ottenuto tramite una "StratifiedKFold". Per ogni classificatore sono state testate le performance sia con il dataset originale sia con una versione bilanciata attraverso l'*Undersampling* della classe maggioritaria, in modo tale da verificare che l'accuratezza raggiunta non fosse influenzata dall'alta percentuale di record con *Occupacy* pari a 0.

3.1 K-Nearest Neighbors

Per trovare il k migliore, nella grid-search sono stati testati valori di k $\in$ [1,100]. Il miglior risultato ottenuto è stato k = 5. Di fatto, come si può vedere dal grafico a destra, in Figura [6], questo è il valore con media d'errore più bassa.

Il grafico a sinistra mostra invece come la curva di apprendimento ottenuta dal classificatore non

presenta problemi di overfitting/underfitting dato che le performance ottenute per il training e per il test tendono a coincidere.

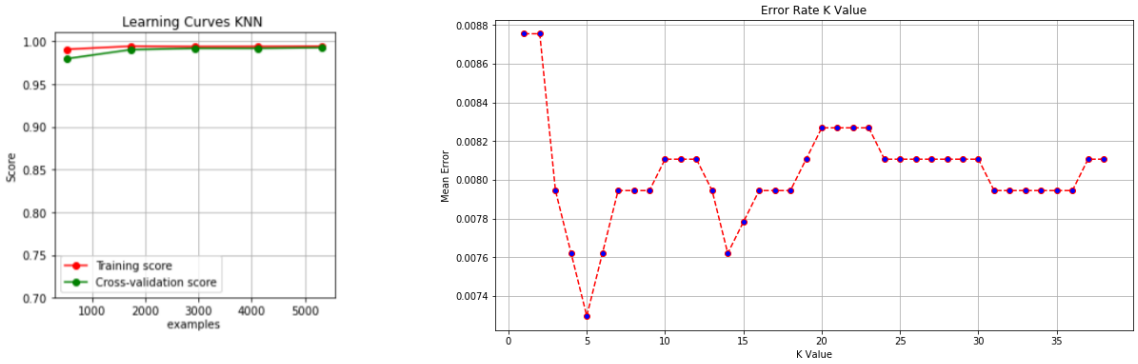


Figura 6: KNN - Learning Curve

I risultati ottenuti dal classificatore sono riportati in Figura [7]. In particolare, la ROC curve mostra che il classificatore basato sul KNN migliora notevolmente il classificatore casuale, presentando un AUC di 0.999. Mentre, dal Lift curve è possibile notare ad esempio che il classificatore è circa 4 volte migliore di quello casuale per il 20% delle previsioni.

Accuracy	0.995
Precision	0.983
Recall	0.994
Specificity	0.995
F1	0.989

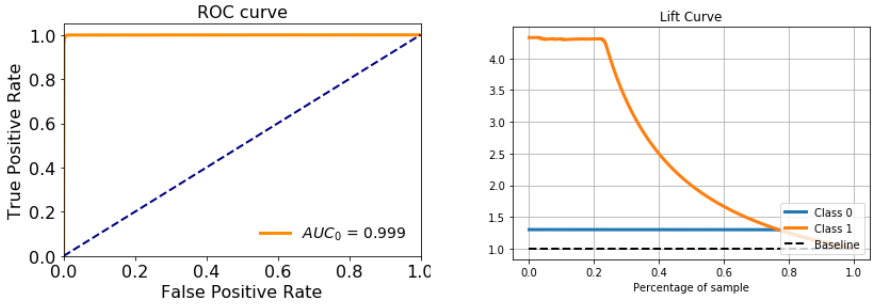


Figura 7: KNN - Risultati

3.2 Naive Bayes

L'algortimo Naive Bayes è stato eseguito utilizzando solamente le variabili *Light*, *CO2*, *HumidityRatio* e *Day*. La selezione di questo sottoinsieme di attributi ha permesso di ottenere un valore di accuracy pari a 97.6%, superiore rispetto al valore ottenibile eseguendo il classificatore con tutti gli attributi.

Tali risultati sono visibili in Figura [8], la quale mostra, a sinistra, le performance di training e test rispetto al classificatore eseguito su tutti gli attributi e, a destra, al classificatore con il sottoinsieme di attributi scelto.

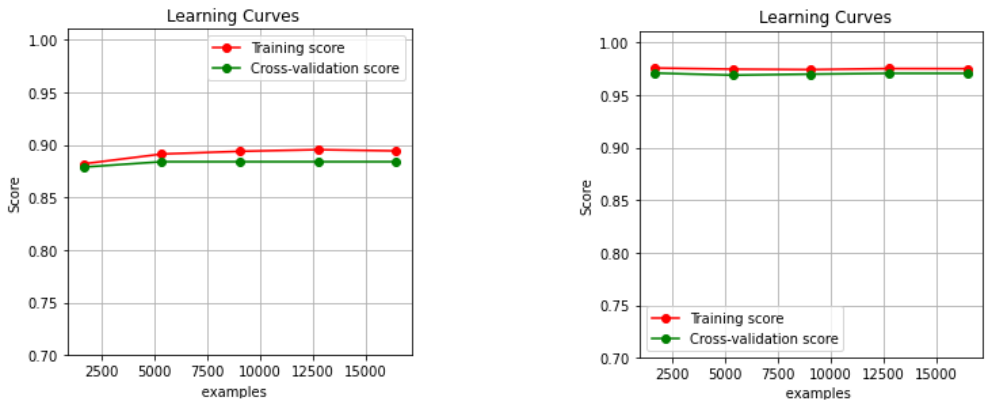
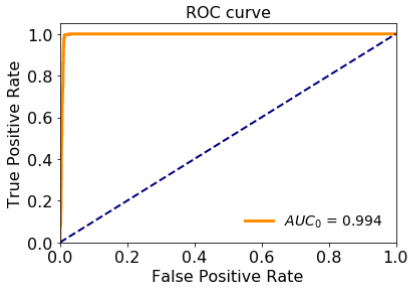


Figura 8: Naive Bayes - Learning Curve

In particolare, la ROC curve, in Figura [9], permette di vedere che il classificatore basato su Naive Bayes migliora notevolmente il classificatore casuale, presentando un AUC di 0.994. Dal Lift curve è possibile, inoltre, notare ad esempio che il classificatore è circa 4 volte migliore di quello casuale per il 20% delle previsioni.

Accuracy	0.976
Precision	0.905
Recall	1.000
Specificity	0.969
F1	0.950



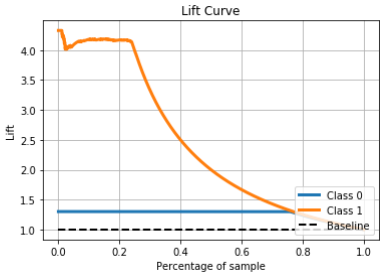


Figura 9: Naive Bayes - Risultati

3.3 Decision Tree

Nella fase di validazione del classificatore, la GridSearch ha permesso di trovare la miglior combinazione dei seguenti iperparametri: criterion, max_depth, min_samples_split.

In Figura [10] viene evidenziato in rosso il risultato che ha permesso di massimizzare l'accuracy, ottenendo un valore pari a 99.2%.

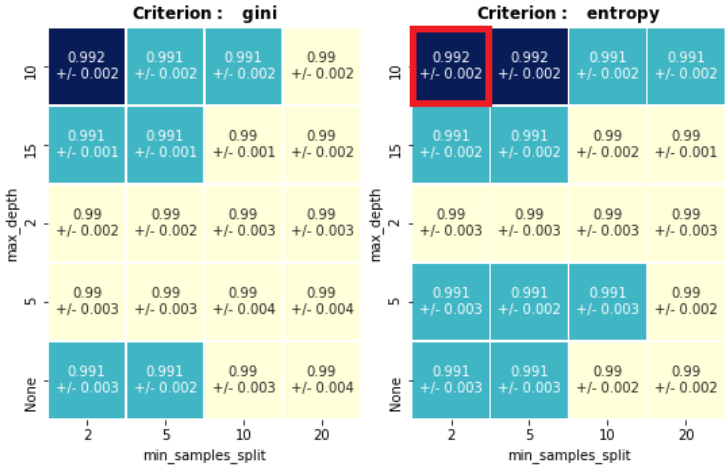
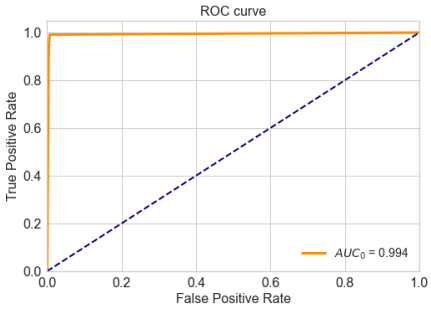


Figura 10: DecisionTree - Parametri

Le performance ottenute sono riportate in Figura [11]. In particolare, la ROC curve mostra che il classificatore basato su DecisionTree migliora notevolmente quello casuale, presentando un AUC di 0.994. La Lift curve evidenzia ad esempio che il classificatore è circa 4 volte migliore di quello casuale per il 20% delle previsioni.

Accuracy	0.992
Precision	0.978
Recall	0.987
Specificity	0.993
F1	0.922



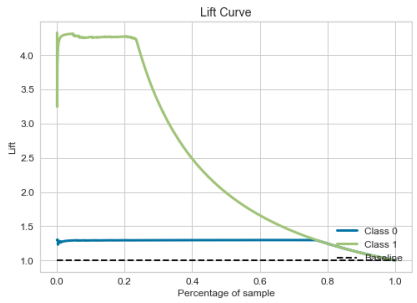


Figura 11: DecisionTree - Risultati

3.4 Logistic Regression

Per la risoluzione del problema della Logistic Regression, durante la fase di validazione è stato scelto solo l'attributo *Light* poiché massimizzava l'accuracy.

La Figura [12] mostra come la regressione logistica sia preferibile rispetto a quella lineare, per il problema preso in esame. I risultati ottenuti dal classificatore sono riportati in Figura [13]. In particolare, la ROC curve mostra che il classificatore basato su Logistic Regression migliora notevolmente il classificatore casuale, presentando un AUC di 0.993. Anche in questo caso, la Lift curve evidenzia ad esempio che il classificatore è circa 4 volte migliore di quello casuale per il 20% delle previsioni.

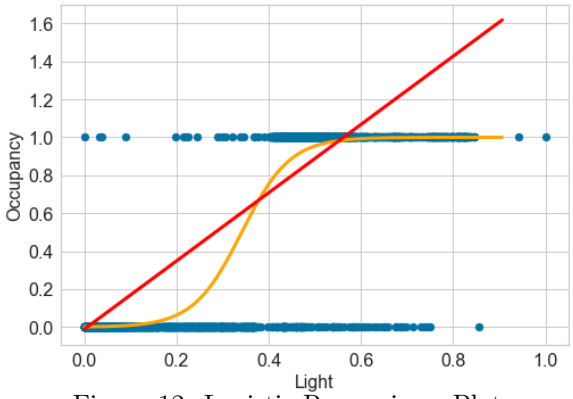
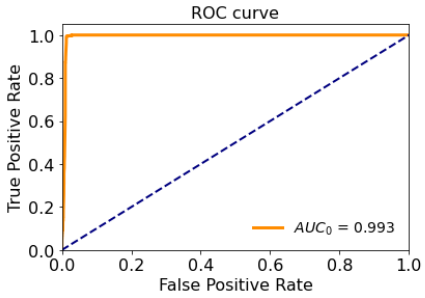


Figura 12: Logistic Regression - Plot

Accuracy	0.984
Precision	0.937
Recall	0.996
Specificity	0.980
F1	0.966



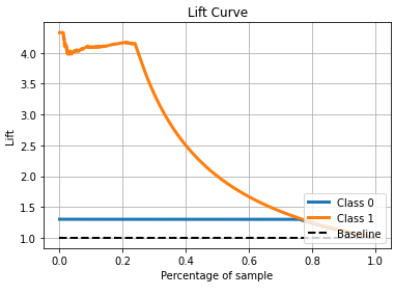


Figura 13: Logistic Regression - Risultati

3.5 Confronto

Tutti i metodi di classificazione testati hanno riportato risultati ottimi, con un'accuracy superiore al 95%. Tra questi, quelli ritenuti migliori, con un'accuracy di circa 99%, sono stati K-Nearest Neighbors e DecisionTree.

4 Dimensionality Reduction

Il problema della riduzione della dimensionalità è stato affrontato attraverso i principali approcci di Feature Selection e Recursive Feature Projection.

4.1 Feature Selection

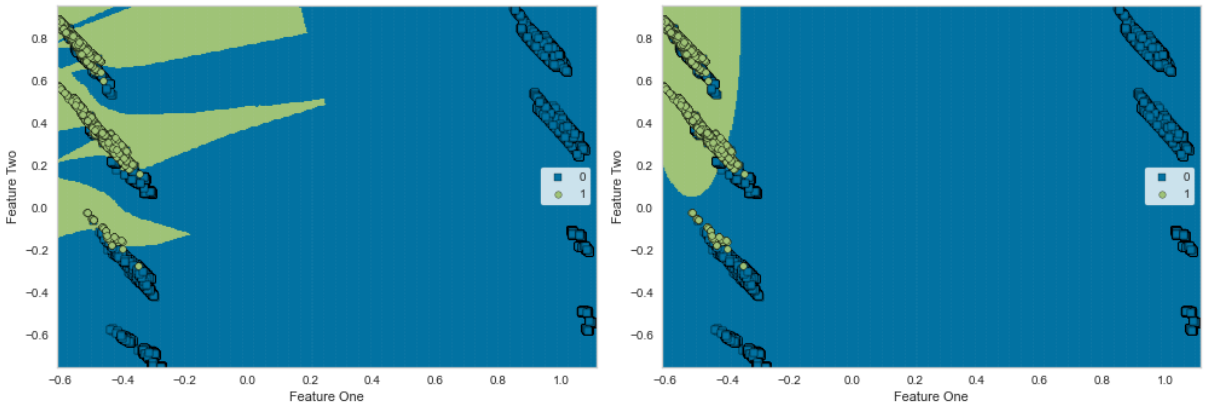
In questo approccio le tecniche utilizzate sono quelle di **Variance Threshold** e **Univariate Feature Selection**. Gli algoritmi sono stati addestrati utilizzando tutti gli attributi presenti nel dataset e, in seguito, le rispettive performance, sono state valutate sul test set. Nel caso della Variance Threshold sono state valutate le performance dell'algoritmo su un range di valori $\in [0.02, 0.2]$. La soglia scelta, pari a 0.04 ha permesso di ridurre le dimensioni del dataset ad 8 dimensioni. É stata ottenuta una buona accuracy per ogni classificatore, 99% per il KNN, 89.2% per Gauss, 98.8% per DecisionTree e LogisticRegression. Nell'Univariate Feature Selection la scelta del k=5 ha permesso di ottenere risultati ottimi con il minimo numero di dimensioni (pari a 5). In questo caso le accuratezze ottenute sono state 99.3% per il KNN, 96.8% per Gauss, 99.2% per DecisionTree e 98.9% per LogisticRegression. Di fatto, aumentando il numero di dimensioni non si otteneva alcun miglioramento nei risultati, mentre riducendolo l'accuracy peggiorava.

4.2 Feature Projection: Principal Component Analysis

Il metodo PCA è stato utilizzato per ridurre la dimensionalità del dataset a due sole componenti, ottenendo i seguenti risultati d'accuracy: 98.3% per il KNN, 95.1% per Gauss, 98.1% per DecisionTree e 94.1% per LogisticRegressione. I decision boundaries ottenuti dall'uso della PCA con due sole dimensioni sono visibili in Figura [14 - 15].

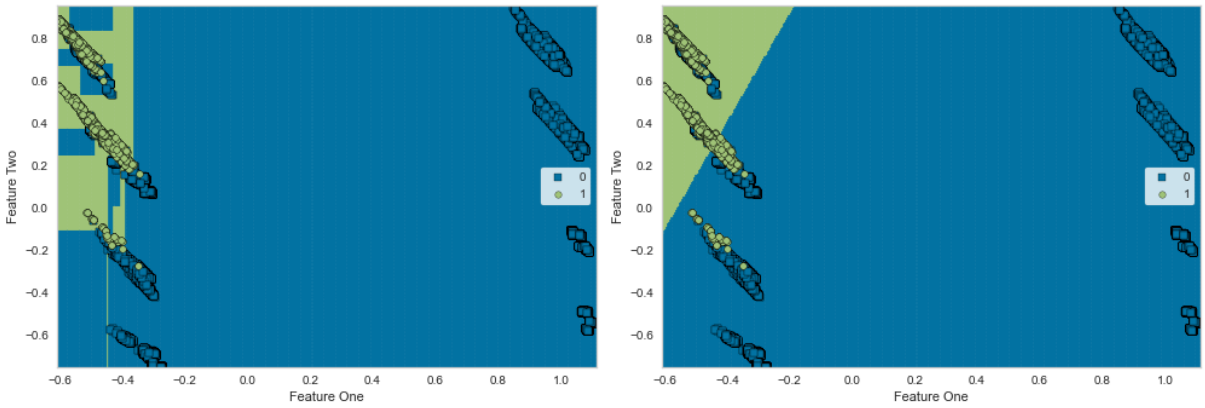
4.3 Confronto

I risultati ottenuti tramite PCA sono leggermente inferiori rispetto ai risultati ottenuti per i corrispondenti classificatori, analizzati in Sezione [3]. Mentre i risultati ottenuti tramite gli algoritmi di Feature Selection sono molto simili a quelli precedenti, nel caso di Logistic Regression otteniamo anche un leggero miglioramento.



(a) KNN (b) Naive Gauss

Figura 14: PCA - Decision Boundaries (KNN - Gauss)



(a) Decision Tree (b) Logistic Regression

Figura 15: PCA - Decision Boundaries (Decision Tree - Logistic Regression)

5 Imbalance

Nel dataset i record, rispetto alla variabile target, risultano essere sbilanciati. Di fatto, il numero di volte in cui la stanza non è occupata è notevolmente maggiore rispetto alle volte in cui lo è. Lo sbilanciamento originale è stato maggiormente accentuato (attraverso l'uso del metodo **make_imbalance**), come visibile in Figura [16], facendo raggiungere alla classe maggioritaria una distribuzione pari a 97.1%.

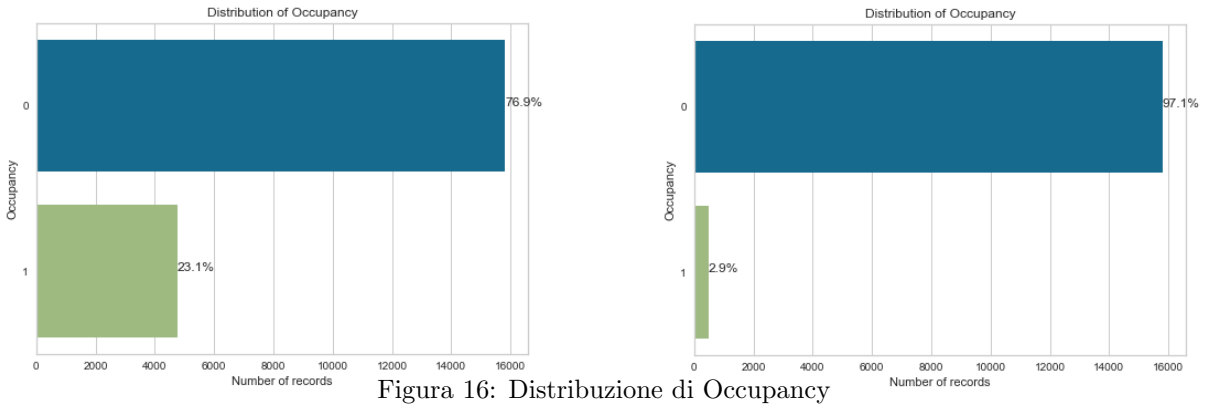


Figura 16: Distribuzione di Occupancy

Sono stati testati tutti i classificatori discussi nella Sezione [3], ottenendo i risultati in Figura [17]. Confrontando i risultati ottenuti con quelli precedenti è possibile notare come le prestazioni siano peggiorate, in particolare con il classificatore Naive Bayes. Ciò dimostra che aumentare lo sbilanciamento del dataset non porta ad un miglioramento delle prestazioni.

	KNN	NaiveGauss	DecisionTree	Logistic Regression
Accuracy	0.995	0.863	0.992	0.987
Precision	0.939	0.176	0.890	0.736
Recall	0.867	1.000	0.846	0.839
Specificity	0.998	0.858	0.997	0.991
F1	0.902	0.299	0.867	0.784
AUC	0.985	0.986	0.985	0.995

Figura 17: Imbalance - Risultati Classificatori

In seguito, il dataset è stato bilanciato attraverso *Oversampling* e, come si può vedere in Figura [18], le prestazioni ottenute sono state decisamente migliori rispetto alle precedenti.

	KNN	NaiveGauss	DecisionTree	Logistic Regression
Accuracy	0.998	0.929	0.998	0.990
Precision	0.996	0.876	0.995	0.987
Recall	1.000	1.000	1.000	0.995
Specificity	0.998	0.858	0.995	0.987
F1	0.996	0.934	0.998	0.991
AUC	0.999	0.985	0.999	0.994

Figura 18: Balance - Risultati Classificatori

6 Regression

Il modello di regressione è stato impostato utilizzando *Light* come variabile indipendente e *Temperature* come variabile dipendente. La scelta è stata effettuata prendendo in considerazione il valore di correlazione dei due attributi. Risolvendo il problema tramite regressione lineare si ha un coefficiente "m" pari a 0.625 e un'intercetta di 0.269, ottenendo il decision boundary presente in Figura [19] (a sinistra).

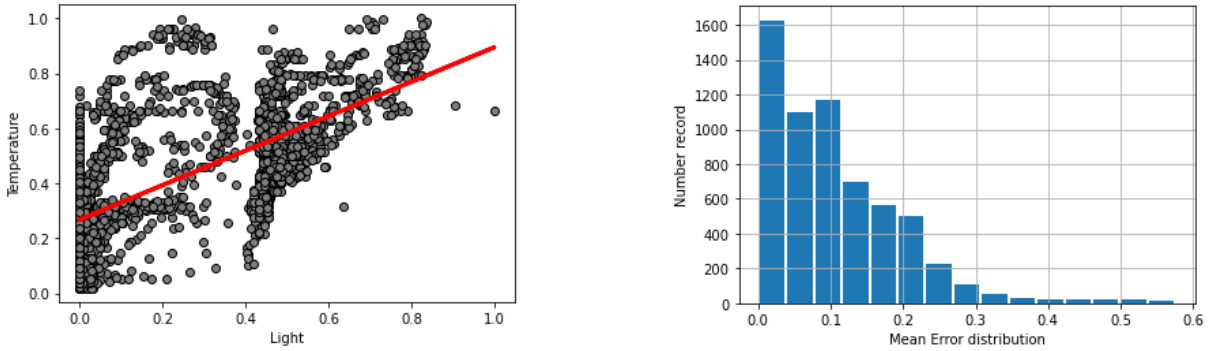


Figura 19: Decision Boundary - Distribuzione errori

Con questi parametri si ottiene un'accuracy e un R2 pari a circa 0.489, mentre un MSE e un MAE pari rispettivamente a 0.019 e 0.106. I valori ottenuti, visibili anche dall'istogramma, indicano che la previsione dei valori della variabile dipendente è buona, benché non ottima, poichè vi è un'elevata presenza di errori, principalmente minori di 0.3. Lo stesso problema è stato risolto con altri tipi di regressione ottenendo però risultati identici o peggiori.

Il problema è stato generalizzato a più variabili dipendenti attraverso la regressione multipla.

Di fatto, l'uso di *Light* e *CO2* come variabili dipendenti ha permesso di ottenere un lieve miglioramento delle metriche con R2 e accuracy pari a 0.515, come mostrato in Figura [20]. Tuttavia, un miglior risultato è stato raggiun-

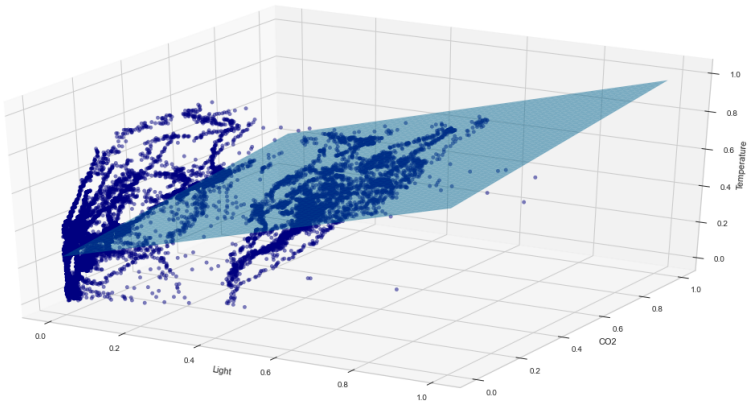


Figura 20: Regressione Multipla

to utilizzando gli attributi *Light*,
CO2, *Day*, *Afternoon*, *Evening*, *Weekday*. In questo caso i valori di R2 e accuracy sono uguali a 0.624, mentre per MSE e MAE si ottengono rispettivamente i valori di 0.014 e 0.087.

6.1 Confronto

In conclusione, è possibile notare come la Regressione Multipla risulta essere più efficace per risolvere il problema di regressione analizzato.

7 Advanced Classifiers

In questa sezione viene descritto l'uso di classificatori avanzati per predire l'occupazione della stanza. Per ognuno di questi, sono stati creati diversi gruppi di iperparametri, come mostrato in Figura [21], a partire dai quali sono stati selezionati i valori "ottimi" che hanno permesso di fornire, per ogni classificatore, il miglior risultato nella previsione della variabile target.

Linear SVM	Non-Linear SVM
tol : 1.0, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6 *C* : 0.001, 0.01, 0.05, 0.1, 1.0, 10.0, 50.0, 100.0	*kernel* : linear, rbg, poly, sigmoid *C* : 0.001, 0.01, 0.05, 0.1, 1.0, 10.0, 50.0, 100.0 *gamma* : 1.0, 1e-1, 1e-2, 1e-3, 1e-4, 1e-5, 1e-6
Single Perceptron	MultiLayer Perceptron
alpha : 0.0001, 0.0003, 0.001, 0.003, 0.01, 0.03, 0.1 penalty: l2, l1, elasticnet *tol* : 1e-1, 1e-2, 1e-3, 1e-4, %1e-5, 1e-6, 1e-7	*learning_rate:* constant, invscaling, adaptive *hidden_layer_sizes:* (12, 23, 11), (23, 43, 32) (128, 64, 32) *momentum:* 0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 0.9
Deep Neural Network	Random Forest
optimizer : SGD, Adagrad, Adam, RMSprop, Adadelta	*n_estimators:* 25, 50, 100, 200, 500, 1000
AdaBoost	Bagging
n_estimators: 5, 10, 25, 50, 100 learning_rate: 0.1, 0.25, 0.5, 0.75, 1	*max_features* : 1, 2, 3, 4 *max_samples* : 0.05, 0.02, 0.01, 0.1, 0.2, 0.5

Figura 21: Hyperparameter Tuning

La scelta dei parametri ottimi è stata effettuata con le stesse metodologie impiegate nella Sezione [3]. Tuttavia, diversamente da quanto fatto in precedenza, si è cercato di ottimizzare l'F1 score, poiché i risultati ottenuti attraverso l'accuracy si sono rivelati poco significativi, in quanto tutti uguali (99%). Fanno eccezione i metodi Ensemble, in cui è stata comunque utilizzata l'accuracy per poter confrontare i risultati presenti in Sezione [3]. I risultati in termini di Roc e Lift, ottenuti per ogni classificatore, vengono mostrati in Sezione [7.5]

7.1 Support Vector Machine

L'SVM è un modello che permette di classificare dati sia linearmente separabili che non. In quest'ultimo caso, realisticamente più probabile, è possibile trasformare i dati in input in uno spazio separabile attraverso un insieme di funzioni matematiche chiamato Kernel. Sia il Kernel che i parametri di regolarizzazione possono causare overfitting, motivo per cui i loro valori sono stai scelti dopo un'attenta selezione .

Utilizzando come features *Temperature*, *Light* *CO2*, *HumidityRatio*, *Evening*, *Weekday* e come parametri d'ingresso **C = 1.0** e **tol = 1.0**, sono stati ottenuti i risultati migliori per il modello Linear SVM, come visibile in Figura [22].
É stata effettuata una prova, similmente a quanto visto in Sezione [6], utilizzando come coppia di features *Temperature* e *Light*. In questo caso sono state ottenute prestazioni leggermente inferiori, seppur molto buone, come anche mostra il margine ottenuto, visibile in Figura [22].

Accuracy	0.990
Precision	0.960
Recall	0.999
Specificity	0.987
F1	0.979
AUC	0.994

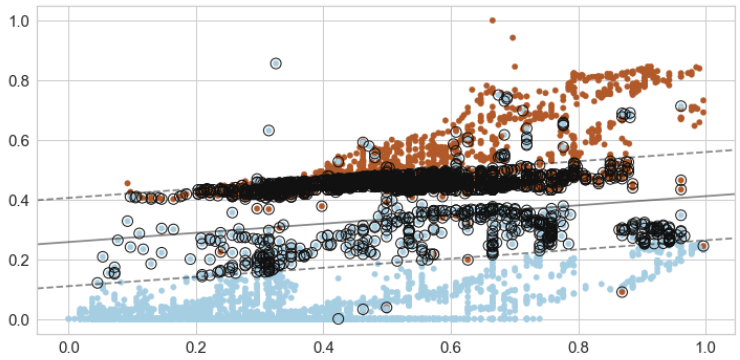


Figura 22: Linear SVM - Risultati e Prova a due Features

Risultati leggermenti migliori sono stati ottenuti con il modello NonLinear SVM, eseguito con la seguente combinazione di parametri: **C = 1.0**, **gamma = 1.0**, **kernel = poly**. A destra della Figura [23] è possibile notare come varia l'error rate, ottenuto come 1 - accuracy, del classificatore al variare dei parametri **C** e **gamma**.
Dato che sono stati ottenuti valori molto simili tra loro, in questo caso il modello linear è prefereribile a quello non linear sia perché ha un costo minore dal punto di vista computazionale sia perché la forte correlazione tra la variabile *Light* e la variabile *Occupancy* permette di predire quest'ultima con elevata precisione.

Accuracy	0.992
Precision	0.973
Recall	1.000
Specificity	0.987
F1	0.985
AUC	0.994

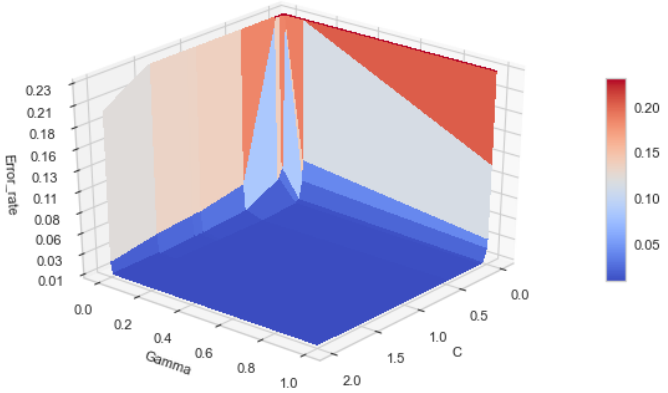


Figura 23: Non-Linear SVM - Risultati

7.2 Neural Network

Uno dei problemi principali delle Neural Networks è l'overfitting. Quest'ultimo può essere evitato non solo controllando il numero di hidden layers, ma anche effettuando una corretta regolarizzazione, motivo principale della scelta degli iperparametri descritti ad inizio capitolo.

Per quanto riguarda il Single Perceptron, in Figura [24] vengono riportati i risultati ottenuti con gli iperparametri scelti, con i quali è stata ottenuta una AUC di 0.994. L'algoritmo è stato eseguito facendo anche Oversampling e Undersampling del dataset. Con quest'ultimo sono state ottenute performance più alte.

	Parameter	Accuracy	Precision	Recall	F1
Single Perceptron	*alpha* : 0.0003, *penalty* : l1, *tol* : 0.001	0.990	0.960	0.997	0.978
Single Perceptron + OverSampling	*alpha* : 0.0001, *penalty* : l1, *tol* : 0.00001	0.993	0.987	0.998	0.993
Single Perceptron + UnderSampling	*alpha* : 0.0001, *penalty* : l1, *tol* : 0.1	0.997	0.995	0.999	0.997

Figura 24: Single Perceptron - Risultati

Nel Multilayer Perceptron i parametri utilizzati sono presenti in Figura [25], i quali hanno portato ad una AUC pari a 0.994. Anche in questo caso è stato testato il classificatore dopo aver bilanciato i dati, ottenendo migliori performance dopo aver effettuato Undersampling.

	Parameter	Accuracy	Precision	Recall	F1
Multilayer Perceptron	*hidden_layer_sizes* : (12, 23, 11), *learning_rate* : adaptive, *momentum* : 0.9	0.990	0.960	0.997	0.978
Multilayer Perceptron + OverSampling	*hidden_layer_sizes* : (128, 64, 32), *learning_rate* : adaptive, *momentum* : 0.9	0.993	0.988	0.998	0.993
Multilayer Perceptron + UnderSampling	*hidden_layer_sizes* : (128, 64, 32), *learning_rate* : constant, *momentum* : 0.9	0.997	0.995	0.999	0.997

Figura 25: Multilayer Perceptron - Risultati

Dopo aver analizzato i risultati ottenuti, molto simili tra Single Perceptron e Multilayer Perceptron, risulta preferibile il primo poiché migliore dal punto di vista computazionale. Si precisa tuttavia, che non sono stati testati altri paramentri (come ad esempio **alpha**) a causa dei costi computazionali della GridSearch.

7.3 Deep Neural Network

Il classificatore è stato implementato delineando due architetture differenti. La prima, chiamata "Model1", è costituita da 8 livelli caratterizzati da 32 neuroni ciascuno, aventi la funzione di attivazione di tipo **tanh**. La seconda, chiamata "Model2", contiene un livello formato da 128 neuroni e 4 livelli da 32 neuroni, tutti aventi come funzione di attivazione **relu**.

	Model1	Model2
Accuracy	0.990	0.990
Precision	0.959	0.959
Recall	1.000	0.997
Specificity	0.987	0.987
F1	0.979	0.978
AUC	0.994	0.994

Figura 26: DNN - Risultati

Con entrambe le architetture sono state ottenute ottime performance, come visibile in Figura [26 - 27], in cui viene anche effettuato il confronto tra le due in termini di Cross-Entropy. Sono state, inoltre, effettuate prove modificando le architetture con parametri di regolarizzazione quali **L2** o **Dropout**, ma con risultati poco significativi, se messi a confronto con quelli già riportati.

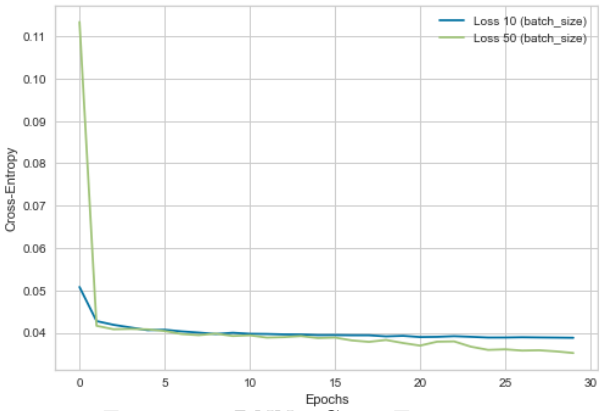


Figura 27: DNN - Cross Entropy

7.4 Ensemble Methods

In questa sezione verranno analizzati i seguenti modelli ensemble: AdaBoost, Random Forest e Bagging. Il modello AdaBoost è stato testato utilizzando, come stimatore base, Naive Bayes in quanto, con quest'ultimo, è stato ottenuto, tra i classificatori presenti in Sezione [3], il valore più basso di accuracy. Di fatto, si è verificato un lieve miglioramento di quest'ultima che da 0.976 è passato a 0.987. I parametri utilizzati, con i quali è stata ottenuta una AUC di 0.996, sono visibili in Figura [28].

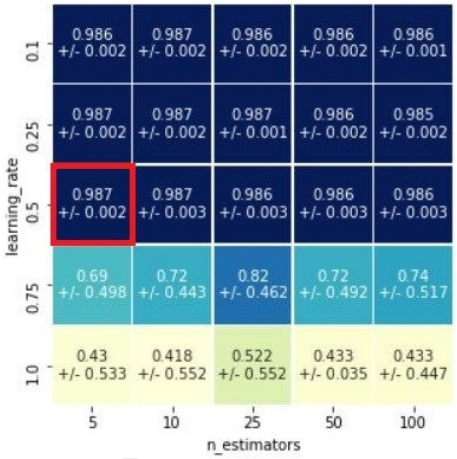


Figura 28: Tuning

Per il modello Random Forest è stato utilizzato lo stimatore di default, DecisionTree. L'algoritmo ha inoltre portato ad una AUC di 0.999 attraverso l'uso dei parametri visibili in Figura [29]. Il grafico in Figura [30] mostra la tecnica **Feature Importance**, che assegna un punteggio alle features in input basandosi sull'importanza che hanno nel predire la variabile target. Come si può notare, nel caso in esame, *Light* è la feature che ha maggiore importanza.

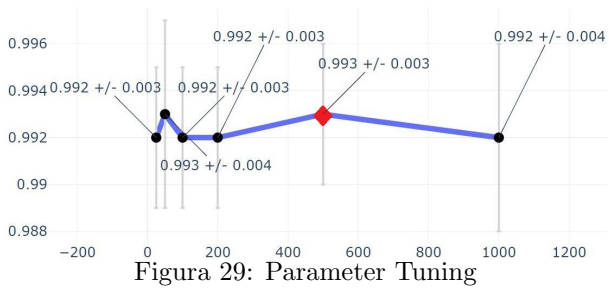


Figura 29: Parameter Tuning

Accuracy	0.993
Precision	0.981
Recall	0.990
Specificity	0.994
F1	0.985
AUC	0.999

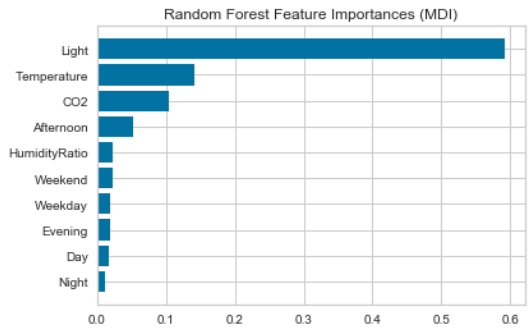


Figura 30: Random Forest - Feature importance

L'algoritmo Bagging è stato testato utilizzando i classificatori presenti in Figura [31]. Confrontando i valori di accuracy con quelli prensenti in Sezione [3], è emerso che con Naive Bayes si ha un lieve miglioramento, passando da 0.976 a 0.981. La AUC ottenuta con i parametri di Naive Bayes è pari a 0.995.

	Parameter	Accuracy	Precision	Recall	F1
Bagging + Naive Bayes	*max_features* = 2, *max_samples* = 0.01, *n_estimators* = 100	0.981	0.925	0.997	0.961
Bagging + KNN	*max_features* = 4, *max_samples* = 0.1, *n_estimators* = 25	0.979	0.955	0.957	0.956
Bagging + Decision Tree	*max_features* = 4, *max_samples* = 0.1, *n_estimators* = 25	0.979	0.952	0.959	0.956
Bagging + Logistic Regression	*max_features* = 4, *max_samples* = 0.1, *n_estimators* = 5	0.964	0.945	0.897	0.921

Figura 31: Bagging - Risultati

In conclusione, è possibile affermare che il Random Forest, tra i metodi ensemble, risulta essere il più efficace per risolvere il problema di classificazione in quanto presenta il valore più alto di accuracy.

7.5 Roc e Lift

Dai grafici in Figura [32] è visibile come per tutti i classificatori precedentemente analizzati, si ottengono risultati simili sia in termini di Roc Curve che di Lift. In particolare, la ROC curve permette di vedere che i classificatori migliorano notevolmente il classificatore casuale, presentando valori di AUC molto simili (come esposto nelle sezioni precedenti). Inoltre, dalla Lift curve è possibile notare come i classificatori sono circa 4 volte migliori di quello casuale per il 20% delle previsioni.

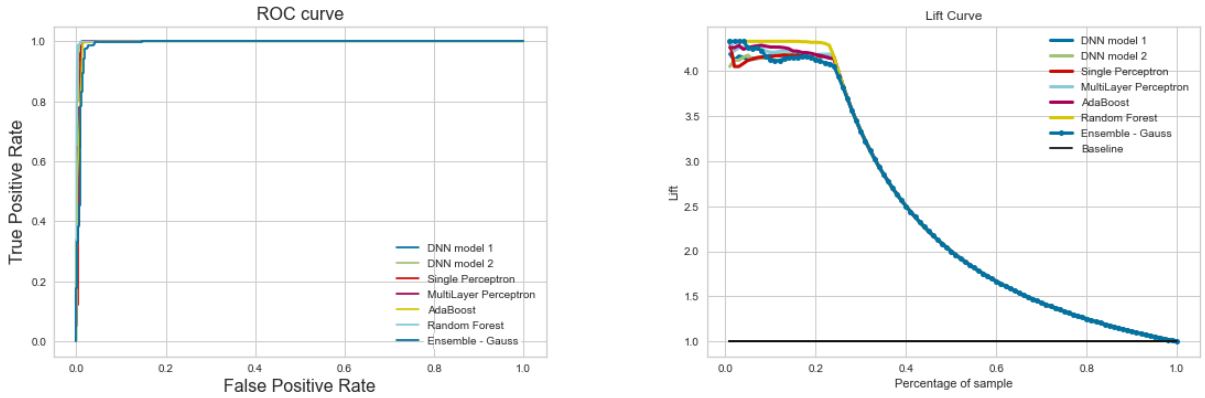


Figura 32: Roc e Lift

8 Time Series Analysis

Per l'analisi delle Time Series è stata scelta come feature la variabile *CO2*, seconda come correlazione con la variabile target *Occupancy*.

8.1 Motifs, anomalies e shapelets

Per la ricerca di Motifs e Anomalies è stato utilizzato un dataset costituito dalla fascia oraria 14:00 - 17:00 di un giorno feriale (Lunedì).
É stata a quel punto calcolata la Matrix Profile della time series, con una finestra temporale di 9 minuti, con la quale sono stati ottenuti, come visibile in Figura [33], diversi valori prossimi allo zero, sinonimo di somiglianza con altre sottosequenze nella time series.

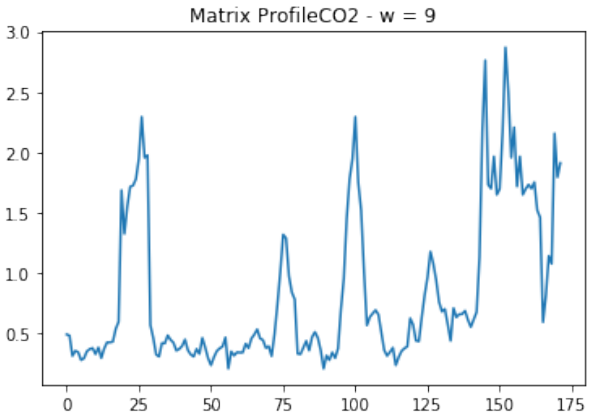


Figura 33: Time series - Matrix Profile

A sinistra della Figura [34], vengono mostrati i motifs ottenuti, dai quali è possibile notare somiglianze in termini di andamento di crescita della *CO2* nella time series, coincidenti con un'occupazione dell'ufficio dopo la pausa pranzo. A destra invece, vengono mostrate le anomalie riscontrate nella stessa serie, coincidenti con possibili rilevazioni scorrette da parte del sensore.

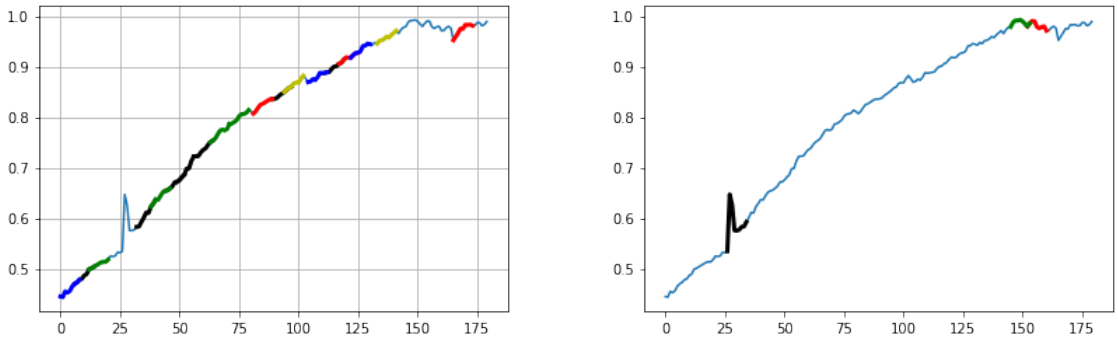


Figura 34: Motifs - Anomalies: Lunedì 14:00-17:00

Sono stati, inoltre, ricercati i motifs, visibili in Figura [35], con una finestra temporale di 25 minuti, in un dataset contenente il giorno feriale dalle 08:00 alle 22:00. Anche in questo caso sono state trovate sequenze simili all'interno della time series, sia in tratti di crescita della *CO2* sia in quelli di decrescita, coincidenti con momenti in cui l'ufficio passa da occupato a non occupato.

In particolare, il colore nero evidenzia il tempo immediatamente successivo all'apertura dell'ufficio, quindi l'incremento repentino di *CO2*, mentre il colore giallo mostra la produzione di *CO2* durante gli orari effettivi di lavoro. Infine, i colori rosso e verde indicano la pausa pranzo o la chiusura dell'ufficio.

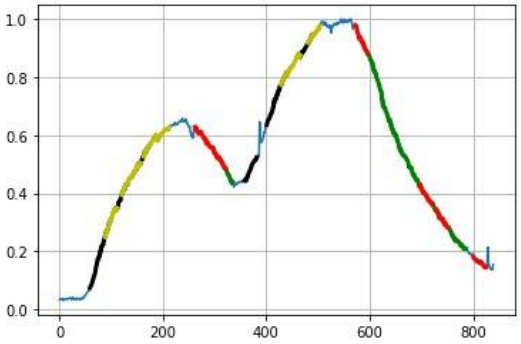


Figura 35: Motifs: Lunedì 08:00 - 22:00

Per la ricerca delle shapelets è stato considerato il datest di 5 giorni utilizzando l'attributo *CO2* e come target *Occupancy*, creando time series di sei ore. In Figura [36] sono visibili le 3 shapelets individuate per due Time Series del dataset, di lunghezza 34. In particolare, le shapelets non si differenziano tanto per la forma, ma per il loro valor medio. Benchè si mostri il miglior allineamento ottenuto per le due serie, le quali identificano il caso in cui la stanza è non occupata (a sinistra) e occupata (a destra), non vengono individuate precisamente delle sottosequenze che rappresentino la classe, probabilmente a causa del dataset considerato che contiene solo 20 time series (4 time series di 6 ore per 5 giorni).

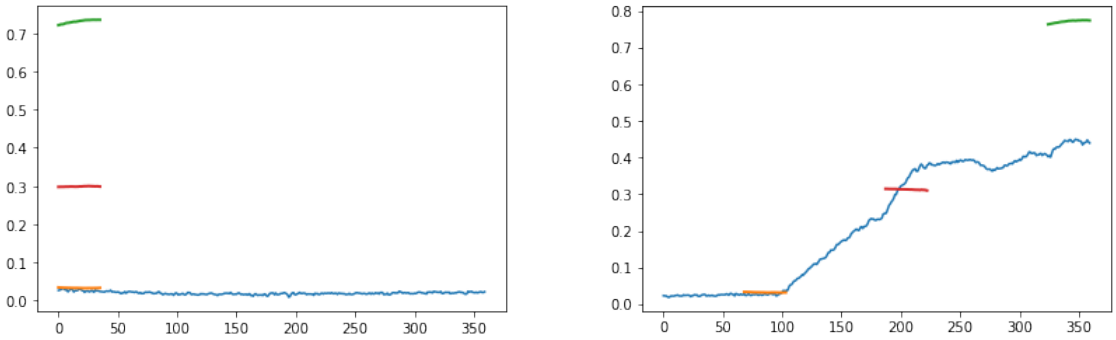


Figura 36: Time series - Shapelets

8.2 Clustering

Sono stati effettuati i confronti tra due giorni feriali della stessa settimana (Lunedì e Giovedì) e tra un giorno feriale e un giorno festivo (Lunedì e Domenica). In entrambi i casi è stata valutata la distanza sia con l'intera time series sia con una fascia oraria ristretta che va dalle 07:00 alle 18:00, in particolare, l'arco di tempo in cui l'ufficio era occupato (in Sezione [2]). I risultati ottenuti, per le diverse distanze, sono riassunti in Tabella [37]. Da quest'ultima è possibile notare come, per tutte le distanze considerate, quelle più elevate si hanno nei confronti feriale-festivo rispetto ai confronti feriale-feriale. Si può quindi affermare che il risultato ottenuto è coerente con quanto analizzato in Sezione [2]. Inoltre, le distanze nella fascia oraria 07.00-18.00 risultano essere di poco inferiori rispetto alle distanze nelle giornate intere; ciò è dato dal fatto che, al di fuori dall'orario lavorativo e nei giorni festivi, la *CO2* mantiene valori prossimi allo 0.

	Euclidea	Manhattan	Cosine	DTW	Sakoe_Chiba	Itakura
Lunedì-Giovedì (00:00 - 24:00)	9.85	256.04	0.08	7.20	9.79	7.59
Lunedì-Giovedì (07:00 - 18:00)	8.15	165.48	0.06	7.08	8.13	7.79
Lunedì-Domenica (00:00 - 24:00)	17.15	454.32	0.52	16.60	17.15	16.94
Lunedì-Domenica (07:00 - 18:00)	15.66	344.33	0.32	15.56	16.66	15.64

Figura 37: Risultati Distanze

La Figura [38] mostra in particolare l'optimal path trovato dal Dynamic Time Warping, con e senza constraints, per quanto riguarda il confronto tra giorno feriale e festivo.

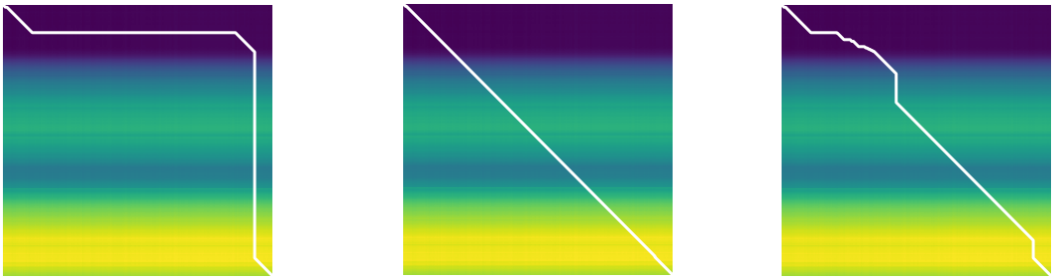


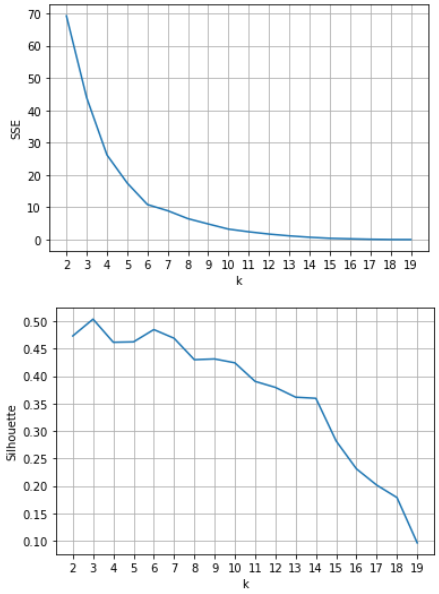
Figura 38: Risultati DTW - Sakoe_Chiba - Itakura

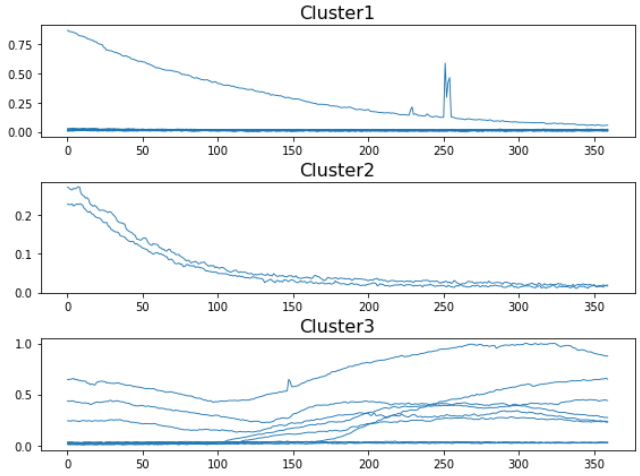
Per effettuare il clustering sono stati utilizzati due dataset. Il primo contenente time series da 6 ore, il secondo contenente time series di un'ora appartenenti solo ai giorni feriali. Il clustering delle time series è stato effettuato mediante diversi approcci:

- K-Means con approccio feature based.
- TimeSeries K-Means con approccio approximation based.

- TimeSeries K-Means con approccio shapelet based.

Per effettuare il clustering "Feature Based" è stato usato il primo dataset dal quale sono state estratte le seguenti statistiche: media, deviazione standard, asimmetria e curtosi. É stato quindi effettuato il clustering utilizzando la distanza euclidea nello spazio delle statistiche. Per calcolare il valore ottimale di K, l'algoritmo K-Means è stato effettuato con k $\in$ [2,20]. Il grafico di SSE e della Silhouette ha mostrato come il valore ottimale di k sia 3.





Come è possibile notare, l'algoritmo ha identificato tre cluster. Il primo contenente 9 time series, tutte tendenzialmente stazionarie, fatta eccezione per una che, invece, evidenzia un andamento decrescente. Quest'ultima, infatti, è "difference stationary" poichè risulta essere stazionaria dal test ADF, ma non da quello KPSS. Il secondo contenente 2 time series che mostrano un andamento decrescente, che rappresenta una diminuzione della CO2 all'interno dell'ufficio. Il terzo contenente invece 9 time series tendenzialmente crescenti, che rappresentano l'intervallo di occupazione della stanza.

Risultati simili ai precedenti sono stati ottenuti anche utilizzando il secondo dataset, le cui time series sono state approssimate attraverso il metodo **Piecewise Aggregate Approximation** con 10 segmenti, poiché l'approssimazione permette di ridurre possibilità di rumore in favore di informazioni più importanti. Sul secondo dataset è stato effettuato anche un clustering di tipo "Shapelet Based". I quattro cluster individuati, sono mostrati in Figura [39]. In particolare, il primo cluster risulta essere molto simile al primo visto nel caso precedente.

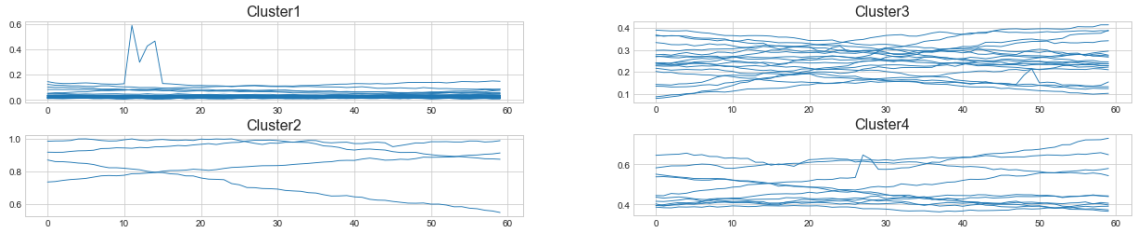
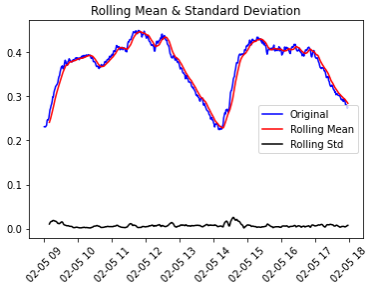


Figura 39: Clustering - Shapelet Based

8.3 Forecasting

Per effettuare il forecasting è stato usato un dataset contenente le rilevazioni di *CO2* in un giorno feriale e nel range orario 09:00 - 18:00. Come visibile in Figura [40], anche se la deviazione standard rimane prossima allo zero, la media subisce grosse variazioni nel tempo, quindi la time series non è stazionaria in generale. Inoltre, dai risultati delle statistiche si nota che la serie è "trend stationary". Infatti, risulta stationary dal KPSS test, poichè il Test Statistic è inferiore ai Critical values si accetta l'ipotesi nulla, secondo la quale "*la time series è trend stationary*"; mentre risulta non stationary dal Dickey-Fuller test, poichè il Test Statistic è superiore ai Critical values non è possibile rifiutare l'ipotesi nulla, cioè che "*la time series ha una radice unitaria*". Prima di effettuare il forecasting, la serie è stata quindi pre-elaborata attraverso *Differencing* e *Trasformation* per ottenere una serie "strict stationary".




```
Results of Dickey-Fuller Test:
Test Statistic                  -2.390322
p-value                          0.144487
#Lags Used                      19.000000
Number of Observations Used    519.000000
Critical Value (1%)             -3.443013
Critical Value (5%)             -2.867125
Critical Value (10%)            -2.569745
dtype: float64
```

```
Results of KPSS Test:
Test Statistic            0.162466
p-value                   0.100000
Lags Used                19.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64
```

Figura 40: CO2 - Statistiche

I risultati dei test statistici ottenuti dopo le trasformazioni effettuate sulla serie sono mostrate in Figura [41]. É possibile notare come la serie risulti effettivamente "strict stationary", infatti si può rifiutare l'ipotesi nulla del DF Test, mentre è possibile accettare quella del KPSS Test.

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.998928
p-value                          0.001416
#Lags Used                      19.000000
Number of Observations Used    529.000000
Critical Value (1%)             -3.442772
Critical Value (5%)             -2.867019
Critical Value (10%)            -2.569688
dtype: float64
```

```
Results of KPSS Test:
Test Statistic            0.159116
p-value                   0.100000
Lags Used                19.000000
Critical Value (10%)      0.347000
Critical Value (5%)       0.463000
Critical Value (2.5%)     0.574000
Critical Value (1%)       0.739000
dtype: float64
```

Figura 41: Statistiche dopo le trasformazioni

In Figura [42] vengono mostrati i grafici *Autocorrelation* e *PartialAutocorrelation* del dataset, spesso utili per la scelta dei parametri di alcuni modelli di Forecasting. É visibile come il PACF ha alti valori fino a lag uguale a 1, per poi seguire un andamento sinusoidale.

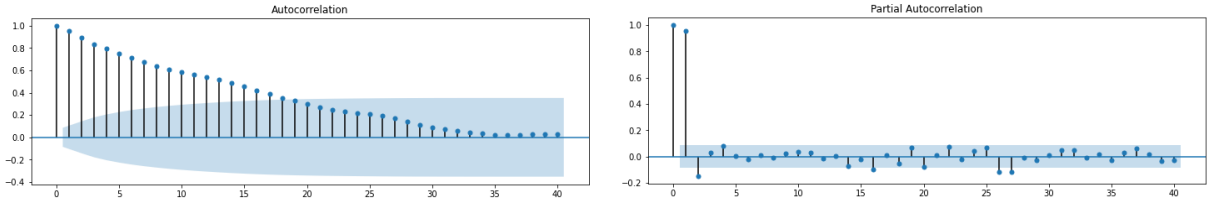


Figura 42: Plot - ACF e PACF

I metodi di Forecasting considerati durante lo studio sono riportati in Figura [43], nella quale si mostrano i confronti tra i modelli e il testset, presente in nero, (ottenuto dividendo il dataset in 70% training e 30% test). A sinistra, vengono confrontati graficamente i risultati di

tre metodi: **Holt Winter's Exponential Smoothing**, **Simple Smoothing** ed **Exponential Smoothing** (eseguito con *seasonal_period* pari a 40 minuti).
Con quest'ultimo sono stati ottenuti i seguenti risultati: MAE = 0.013, RMSE = 0.016, MAPE = 4%, R2 = 0.52, i quali spiegano come il metodo utilizzato è migliore degli altri due metodi con il quale è stato confrontato. Tuttavia, è possibile vedere dai valori precedenti di R2 e MAPE, come il metodo non è molto buono. Non vengono riportati i risultati ottenuti dal modello **ARIMA** poiché ritenuti insoddisfacenti. Tuttavia, in Figura [43] a destra viene riportato il confronto con il modello **SARIMAX**. Attraverso l'analisi dei plot in Figura [42] è stato deciso di settare nel modello $p = 1$, $d = 0$. Per settare gli altri parametri del modello, q, P, D, Q sono state effettuate varie prove, poiché è stato difficoltoso stimarli sulla base dei grafici ACF e PACF. Il modello è stato infine eseguito con i parametri *order=(1, 0, 3)*,*seasonal_order=(3, 1, 3, 10)*, ottenendo i valori: MAE = 0.014, RMSE = 0.017, MAPE = 2%, R2 = 0.48. Mentre l'R2 ottenuto in questo caso è di poco inferiore rispetto all'Exponential Smoothing, il MAPE è decisamente migliore rispetto al precedente.

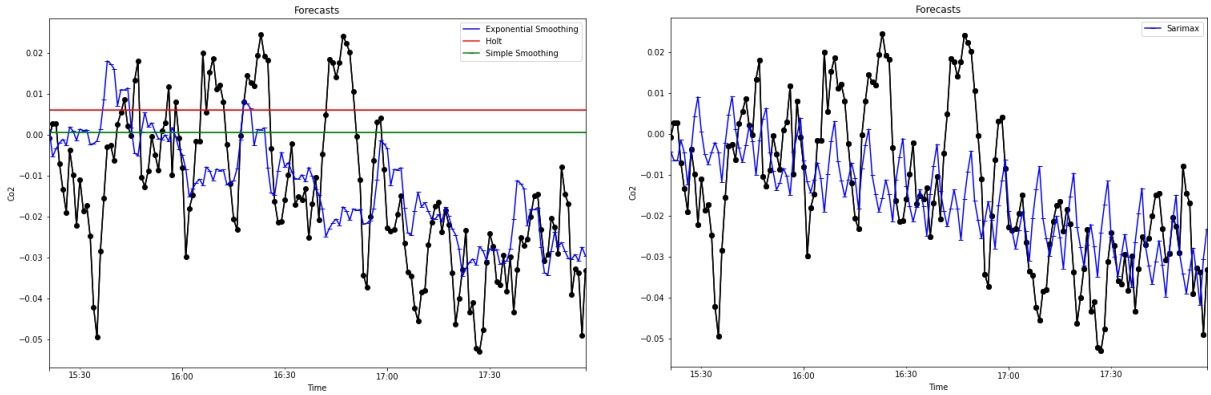


Figura 43: Metodi di Forecasting

8.4 Classificazione

Per effettuare la classificazione sono stati utilizzati due diversi dataset, con lo scopo di predire se la stanza fosse occupata o meno. Il primo contenente time series di sei ore, mentre il secondo, del quale sono riportati i risultati poiché migliori, contenente time series di un'ora. Come fatto in precedenza, per ogni classificatore sono stati scelti i parametri migliori attraverso l'utilizzo della GridSearch.

8.4.1 Univariate data

In questo caso, il dataset è stato creato utilizzando unicamente l'attributo *CO2*, come fatto nelle precedenti analisi. In Figura [44] vengono riportati gli algoritmi utilizzati, con relativi parametri e risultati (in termini di Accuracy e F1 score). Il primo algoritmo provato è stato lo Shapelet classifier, il quale ha individuato 3 differenti shapelets. Successivamente è stato testato lo Shaplet-distances-based Classifier, eseguito utilizzando come classificatori base sia il DecisionTree sia il KNN (con i quali erano stati ottenuti i valori migliori nel Task precedente).

Per eseguire il Feature-based Classifier sono state utilizzate le seguenti statistiche: media, deviazione standard, covarianza, novantesimo percentile e asimmetria; mentre come classificatore base è stato utilizzato il DecisionTree.
Infine, l'ultimo algoritmo testato è stato il CNN. Il classificatore è stato implementato utilizzando un modello composto da quattordici livelli nei quali viene ripetuta per quattro volte la seguente struttura:

- Conv1: filter = 32, kernel_size = 8, activation = 'relu'
- Dense: units = 32, activation = 'relu'
- Dropout: dropout rate = 30%

Gli ultimi due livelli sono costituiti da un livello di tipo "GlobalAveragePooling1D" e un livello "Dense" con funzione di attivazione "sigmoid", la quale assicura che l'output sia compreso tra zero e uno. Il modello creato è stato compilato usando la "binary_crossentropy" come funzione "loss" in quanto i valori del target $\in \{0, 1\}$.

	Parameter	Accuracy	F1
Shapelet Classifier	*optimizer* = "sgd", *weight_regularizer* = .01, *max_iter* = 200, *verbose* = 1	0.805	0.888
Shaplet-distances-based Classifier + Decison Tree	*criterion* = 'gini', *max_depth* = 2, *min_samples_leaf* = 4, *min_samples_split* = 2	0.916	0.945
Shaplet-distances-based Classifier + KNN	*n_neighbors* = 8, *weights* = 'uniform'	0.994	0.993
Feature-based Classifier + Decisicn Tree	*criterion* = 'gini', *max_depth* = 8, *min_samples_leaf* = 4, *min_samples_split* = 2	0.972	0.982
CNN Classifier	*monitor* = 'loss', *factor* = 0.5, *patience* = 5, *min_lr* = 0.0001	0.777	0.875

Figura 44: Time Series - Classificazione univariata

Come è possibile notare dalla Figura [44], i risultati migliori sono stati ottenuti con Shapelet distances-based classifier utilizzando l'algoritmo KNN.

8.4.2 Multivariate data

Per la classificazione multivariata sono state utilizzate le seguenti features: *Light*, *CO2*, *HumidityRatio* e *Temperature*. Per ogni classificatore, in Figura [45] sono riportati i rispettivi valori di accuracy e f1 score. Anche in questo caso per lo Shapelet Classifier sono stati individuati tre differenti shapelets. Lo Shaplet-distances-based Classifier ha restituito risultati migliori utilizzando come classificatore base il KNN, come avvenuto per il caso di classificazione univariata.
Nel Feature-based Classifier sono state utilizzate le seguenti statistiche: media, deviazione standard, covarianza e decimo percentile.
Sono stati, infine, testati sia l'algoritmo LSTM che CNN, utilizzando due diverse architetture.
Il modello utilizzato per l'algoritmo LSTM è costituito da quattordici livelli così definiti:

- Un livello LSTM costituito da quattro neuroni;
- Due livelli BatchNormalization per standardizzare automaticamente gli input dei livelli nella rete;
- Due livelli LeakyReLU;
- Tre livelli Dropout impostati al 30%;
- Un livello LSTM costituito da trentadue neuroni;
- Un livello "Dense" (ultimo livello), avente la funzione di attivazione "sigmoid".

In questo caso, il modello è stato compilato usando la "sparse_categorical_crossentropy". Il CNN è stato invece eseguito utilizzando un modello composto da dieci livelli. I primi due livelli presentano la seguente struttura:

- Conv2: filter = 3, kernel_size = (3, 3), padding = 'same', kernel_initializer = 'TruncatedNormal';
- Dropout: dropout rate = 30%.

Nei successivi quattro livelli si ha la stessa struttura dei primi due, fatta eccezione per il numero di filter = 4 e kernel_size= (4,4). Successivamente sono stati utilizzati un livello Dropout con rate = 30%, un livello Flatten e un livello Dense (avente la funzione di attivazione "sigmoid"). Come per il caso precedente, anche questo modello è stato compilato usando la "sparse_categorical_crossentropy" come funzione "loss".

	Parameter	Accuracy	F1
Shapelet Classifier	*optimizer* = "sgd", *weight_regularizer* = .01, *max_iter* = 200, *verbose* = 1	0.777	0.888
Shapelet-distances-based Classifier + Decison Tree	*criterion* = 'gini', *max_depth* = 2, *min_samples_leaf* = 2, *min_samples_split* = 2	0.916	0.945
Shaplet-distances-based Classifier + KNN	*n_neighbors* = 3, *weights* = 'uniform'	1.0	1.0
Feature-based Classifier + Decision Tree	*criterion* = 'gini', *max_depth* = 2, *min_samples_leaf* = 4, *min_samples_split* = 2	1.0	1.0
LSTM	*monitor* = 'loss', *factor* = 0.5 , *patience* = 80, *min_lr* = 0.0001	0.833	0.903
CNN Classifier	*monitor* = 'loss', *factor* = 0.5, *patience* = 25, *min_lr* = 0.0001	0.972	0.982

Figura 45: Time Series - Classificazione Multivariata

Come visibile in Figura [45] il risultato peggiore (sia in termini di Accuracy che di F1) è stato ottenuto con il tradizionale Shapelet Classifier; mentre i risultati migliori sono stati raggiunti dallo Shapelet-distances-based Classifier usando il KNN e dal Feature-based Classifier utilizzando il Decision Tree.

9 Sequential Pattern Mining

Per l'analisi del Sequential Pattern Mining sono stati utilizzati due dataset di Time Series, il primo di una settimana che comprendeva giorni sia lavorativi che non lavorativi, il secondo invece di un solo giorno lavorativo. In entrambi i casi sono stati utilizzati i quattro attributi principali (*Temperature*, *Light*, *CO2*, *HumidityRatio*). Le serie sono state discretizzate utilizzando il metodo **SAX** con i parametri **n_sax_symbols = 10** e **n_paa_segments = 25**. In Figura [46] viene riportato un esempio del confronto tra la discretizzazione ottenuta ed il valore reale della time series per il giorno lavorativo, relativo alle features *Temperature* e *CO2*.

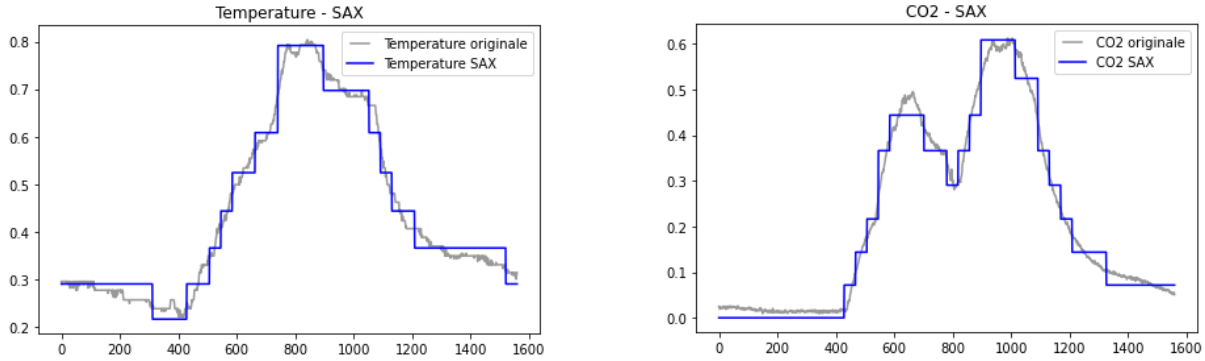


Figura 46: Discretizzazione SAX: Temperature - CO2

La ricerca dei pattern è stata effettuata utilizzando la libreria *PrefixSpan* e quindi rappresentando i dataset di Time Series come transazioni di quadruple (dove ogni elemento rappresenta

il valore di un attributo in un determinato momento temporale) e considerando transazioni di due lunghezze diverse, 30 e 60. In Figura [47] vengono mostrati, per i due dataset considerati (a sinistra quello di una settimana, a destra il dataset composto dal solo giorno lavorativo) i sequential patterns, e relativi supports, per le transazioni lunghe 60. A parità di patterns vengono riportati anche i supporti ottenuti per l'esecuzione dell'algoritmo con transazioni lunghe 30. Sono stati riportati in Figura solo i pattern di lunghezza minima pari a 4.

Support - con transazioni lunghe 60	Support - con transazioni lunghe 30	Pattern [Lunghezza minima = 4]
15	28	(1, 0, 0, 4) (1, 0, 0, 4) (1, 0, 0, 4) (1, 0, 0, 4)
13	26	(1, 0, 0, 4) (1, 0, 0, 4) (1, 0, 0, 4) (1, 0, 0, 4)
11	21	(1, 0, 0, 6) (1, 0, 0, 6) (1, 0, 0, 6) (1, 0, 0, 6)
10	16	(3, 0, 0, 0) (3, 0, 0, 0) (3, 0, 0, 0) (3, 0, 0, 0)
9	19	(1, 0, 0, 6) (1, 0, 0, 6) (1, 0, 0, 6) (1, 0, 0, 6)
9	15	(3, 0, 0, 0) (3, 0, 0, 0) (3, 0, 0, 0) (3, 0, 0, 0)
8	15	(8, 6, 5, 6) (8, 6, 5, 6) (8, 6, 5, 6) (8, 6, 5, 6)
5	8	(5, 6, 6, 8) (5, 6, 6, 8) (5, 6, 6, 8) (5, 6, 6, 8)

Support - con transazioni lunghe 60	Support - con transazioni lunghe 30	Pattern [Lunghezza minima = 4]
6	11	(1, 0, 0, 0) (1, 0, 0, 0) (1, 0, 0, 0) (1, 0, 0, 0)
5	10	(1, 0, 0, 1) (1, 0, 0, 1) (1, 0, 0, 1) (1, 0, 0, 1)
3	7	(2, 0, 1, 2) (2, 0, 1, 2) (2, 0, 1, 2) (2, 0, 1, 2)
3	5	(2, 0, 2, 3) (2, 0, 2, 3) (2, 0, 2, 3) (2, 0, 2, 3)
3	5	(6, 3, 8, 7) (6, 3, 8, 7) (6, 3, 8, 7) (6, 3, 8, 7)
2	4	(0, 0, 0, 0) (0, 0, 0, 0) (0, 0, 0, 0) (0, 0, 0, 0)

Figura 47: Risultati SPM

In particolare, per il dataset di una settimana è possibile osservare come patterns con valori bassi, interpretabili come momenti in cui l'ufficio è chiuso (quindi con valori bassi delle features), hanno un support più alto. Mentre sequenze con valori alti, interpretabili invece come momenti in cui l'ufficio è aperto, hanno un support più basso. Questo risultato, probabilmente, si ottiene perchè nel primo dataset i momenti in cui l'ufficio è chiuso sono maggiori dei momenti in cui è aperto. Alternativamente, è possibile vedere come nel dataset costituito da un solo giorno lavorativo si hanno pattern con valori intermedi, tranne che per l'attributo Light che, in quei casi, ha valore pari a zero. Infatti, quest'ultimo attributo è l'unico che non decresce/cresce gradualmente alla chiusura/apertura dell'ufficio. Inoltre, considerando transazioni di lunghezza inferiore (30 invece di 60) sono stati ottenuti valori di support più alti a parità di stesso pattern.

10 Outlier Detection

L'obiettivo del task è di trovare i Top-10 outliers utilizzando il dataset originale, ma in particolare le features *Light* e *CO2* poiché per le altre non sono stati riscontrati outliers rilevanti. In questa sezione sono stati utilizzati cinque metodi diversi per trovare gli outliers, ricordando che il metodo più semplice (Boxplot) è stato utilizzato in Sezione [2] per identificare i principali outliers della feature *Light*. Sono stati utilizzati due modelli density-based *LOF* e *DBSCAN*, un modello angle-based, *ABOD*, un modello distance-based *KNN* e un approccio convex hull.
La Figura [48] mostra i risultati, ordinati in ordine decrescente per negative_outlier_factor, dei 10 principali outliers ottenuti dall'applicazione del metodo *LOF*, confrontando questi ultimi con i valori ottenuti dall'applicazione dei metodi *ABOD* e *KNN*.

Time	LOF negative_outlier_factor	ABOD decision_scores_	KNN decision_scores_
2015-02-07 09:42:59	-24.102	-2.513	560.551
2015-02-07 09:42:00	-23.976	-1.177	605.457
2015-02-04 09:40:59	-14.634	-1.987	518.895
2015-02-05 13:07:00	-11.592	-2.306	75.912
2015-02-13 09:49:00	-11.562	-4.776	227.089
2015-02-12 09:47:00	-11.466	-4.205	570.514
2015-02-12 09:46:00	-10.565	-1.541	404.379
2015-02-04 09:40:00	-10.337	-1.251	490.358
2015-02-03 13:09:59	-10.137	-1.060	80.799
2015-02-18 01:51:00	-9.370	-8.413	292.189

Figura 48: Top 10 LOF outliers

Per poter paragonare i risultati ottenuti dai metodi, in Figura [49], è stata effettuata l'intesezione tra i Top-10 outliers di *LOF*, *KNN* e *DBSCAN*, ottenendo 5 elementi. In particolare il DBSCAN ha riportato come outliers i 9 visibili nel Boxplot presentato in Sezione [2]. Dall'intersezione sono stati esclusi i Top-10 outliers ottenuti da *ABOD* al fine di non avere intersezione vuota, infatti (come già visibile in Figura [48]) quest'ultimo è il metodo che più si differenzia dai precedenti.

Time	Temperature	Light	CO2	HumidityRatio	Occupancy
2015-02-07 09:42:59	20.745	1451.75	453.0	0.002853	0
2015-02-07 09:42:00	20.7	1546.33	455.3	0.002845	0
2015-02-12 09:47:00	22.79	1581.0	1211.5	0.004338	1
2015-02-12 09:46:00	22.772	1380.0	1202.5	0.004333	1
2015-02-04 09:40:00	22.6	1419.5	945.0	0.004319	1

Figura 49: Intersezione Top 10 Outliers

Queste similitudini sono riscontrabili anche in Figura [50], la quale mostra come vengono generati risultati simili per *LOF* (a sinistra) e *KNN* (al centro), attraverso uno scatterplot delle features *CO2* e *Light*. Vengono, inoltre, mostrati per le stesse features i Top-4 involucri convessi (a destra), i quali sono costituiti dai punti con Depth Score più basso e quindi con maggiore probabilità di essere outliers. In particolare, la profondità massima individuata per queste features è pari a 721.

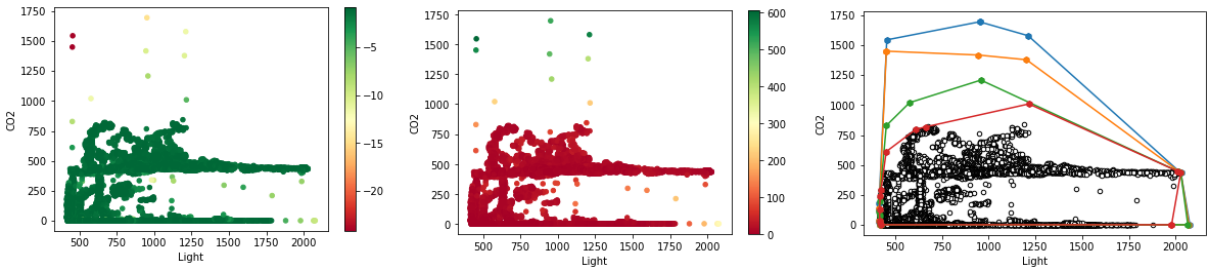


Figura 50: LOF - KNN - ConvexHull

Confronti simili sono visibili anche in Figura [51], nella quale è possibile vedere come l'involucro convesso più esterno, individuato dall'algoritmo ConvexHull (a destra) per le features *CO2*, *Light* e *Temperature*, identifica circa gli stessi outliers, più esterni, del metodo LOF (a

sinistra). Nel plot del LOF gli outliers sono stati identificati in rosso per differenziarli dagli altri punti.

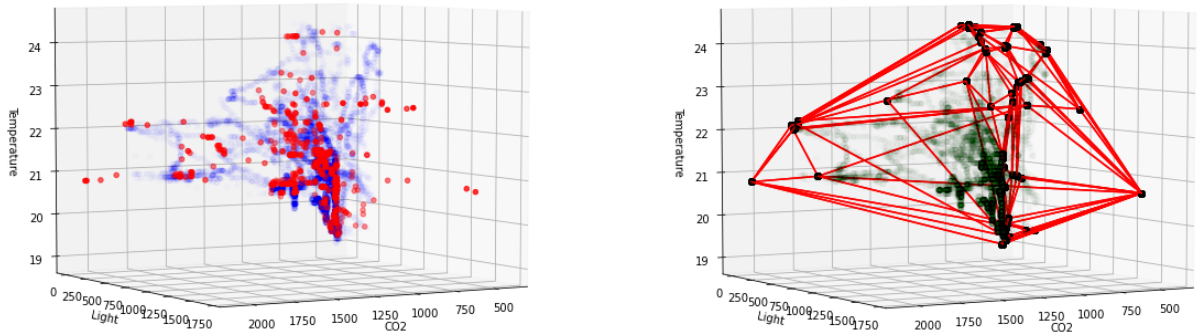


Figura 51: Plot 3D: LOF - Convex Hull

11 Explainability

Per affrontare il problema del "Black Box Explaination" sono stati utilizzati due tipi di metodi, per poter considerare le dimensioni più importanti per l'interpretabilità (globale e locale) ed analizzare come alcuni classificatori, avanzati (Non Linear SVM, Multilayer Perceptron, AdaBoost) e non (kNN), prendessero le loro decisioni.

11.1 Inspection Model Explainer

In questo caso è stato utilizzato il **Partial Dependency Plot**, ottenendo i risultati visibili in Figura [52] (che per motivi di spazio riporta solo i risultati di SVM e AdaBoost), dai quali è possibile notare come un aumento delle features *Light* e *CO2* implica un aumento di probabilità di prevedere la classe "Occupancy = 1"(risultato in linea con quanto visto in precedenza).

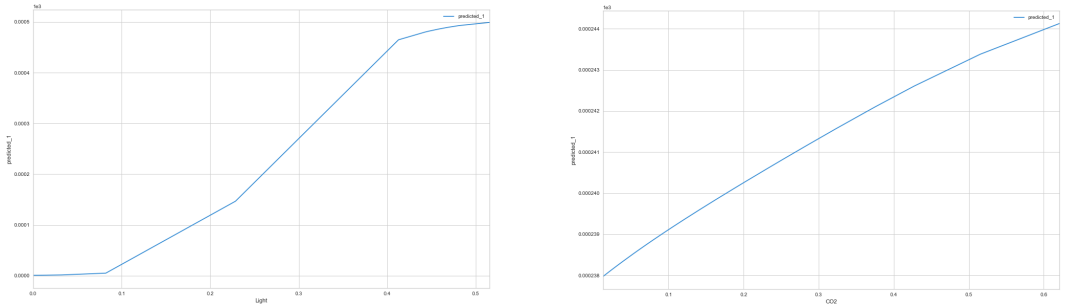


Figura 52: SVM: Light - CO2

Sono stati ottenuti due ulteriori risultati, visibili in Figura [53]. Il primo riguarda la *Temperature*, nel classificatore MLP, in cui la probabilità decresce, ma con una variazione minima, indicando probabilmente che la feature è di poca importanza per il classificatore. Il secondo risultato riguarda le probabilità delle features *Temperature* e *HumidityRatio*, per il classificatore AdaBoost, le quali rimangono costanti avvalorando ulteriormente il motivo per il quale sono stati ottenuti risultati ottimi anche non utilizzando queste features.

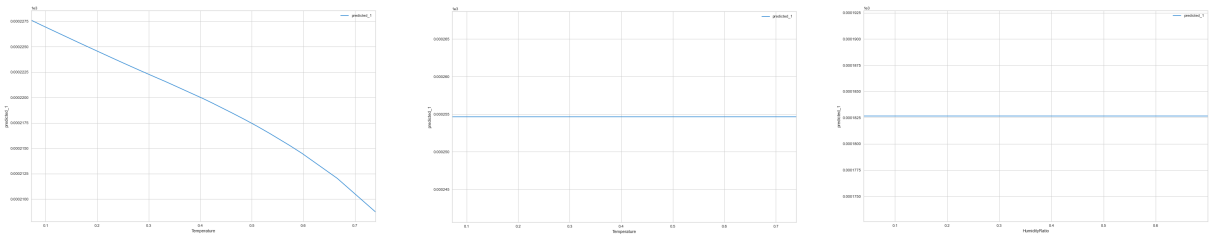


Figura 53: MLP (Temperature) - AdaBoost (Temperature, HumidityRatio)

Per tutti i classificatori è stato ottenuto che *Light* è la feature con la variazione di probabilità più alta, osservazione coerente con le analisi precedenti.

11.2 Outcome Explainer

Come metodo locale per risolvere il problema di Outcome Explanation è stato testato il metodo LORE. Per la corretta creazione delle regole di quest'ultimo, il dataset è stato modificato trasformando la codifica binaria delle features *Day*, *Afternoon*, *Evening*, *Night* e delle features *WeekDay*, *WeekEnd* in due features categoriche *TimeSpan* e *WeekSpan*, le quali hanno come possibili valori *DA*, *AF*, *EV*, *NI* per la prima e *WD*, *WE* per la seconda. In Figura [54] viene mostrato un esempio di possibili regole e rispettivi counterFactor sia per la classe 0 che per la classe 1, avendo come BlackBox i classificatori sopra elencati. In particolare, è possibile notare come per AdaBoost sia le regole che i counter factors sono interamente costituiti dalla feature *Light*.

	Rule	CounterFactual
SVM	{ Light <= 0.15 } --> { Occupancy: 0 }	{ { Light > 0.15, Temperature > 0.74, TimeSpan = EV }, { Light > 0.15, CO2 <= 0.01, TimeSpan = AF }, { Light > 0.15, CO2 > 0.61, TimeSpan = DA } }
	{ Light > 0.16, CO2 > -0.04, TimeSpan = AF, HumidityRatio <= 0.72 } --> { Occupancy: 1 }	{ { Light <= 0.16 } }
MLP	{ Light <= 0.28, TimeSpan != AF, Temperature > 0.03, CO2 <= 0.47, TimeSpan != EV } --> { Occupancy: 0 }	{ { Light > 0.33 }, { Temperature <= 0.03 } }
	{ Light > 0.33, WeekSpan != WE, TimeSpan != NI, Temperature <= 0.74 } --> { Occupancy: 1 }	{ { WeekSpan = WE }, { Light <= 0.30 }, { TimeSpan = NI }, { Temperature > 0.74 }, { Light <= 0.33 } }
AdaBoost	{ Light <= 0.41 } --> { Occupancy: 0 }	{ { Light > 0.41 } }
	{ Light > 0.35 } --> { Occupancy: 1 }	{ { Light <= 0.35 } }
KNN	{ Light <= 0.28, TimeSpan != AF, Temperature > 0.03, CO2 <= 0.47, TimeSpan != EV } --> { Occupancy: 0 }	{ { Light > 0.33 }, { Temperature <= 0.03 } }
	{ Light > 0.25, WeekSpan != WE, HumidityRatio > 0.21, TimeSpan != NI } --> { Occupancy: 1 }	{ { Light <= 0.25 }, { WeekSpan = WE }, { TimeSpan = NI } }

Figura 54: Risultati LORE