

UNIVERSITÀ DI PISA

Anno accademico 2019/2020



**Data Mining II**  
**Project Report**

Gruppo 20

Ilenia Bari - 590445

Gabriele Leone - 563955

Salvatore Lusito - 609000

# Indice

<b>1 Data Understanding and Preparation</b>	<b>1</b>
<b>2 KNN</b>	<b>2</b>
<b>3 Naïve Bayes</b>	<b>3</b>
3.1 NB Categoriale . . . . .	3
3.2 NB Gaussiano . . . . .	4
3.3 Considerazioni . . . . .	4
<b>4 Logistic Regression</b>	<b>5</b>
<b>5 Decision Tree</b>	<b>6</b>
<b>6 Regression</b>	<b>9</b>
<b>7 Imbalanced Learning</b>	<b>10</b>
<b>8 Principal Component Analysis (PCA)</b>	<b>14</b>
<b>9 Support Vector Machine</b>	<b>15</b>
9.1 Linear SVM . . . . .	15
9.2 Non-Linear SVM . . . . .	16
<b>10 Neural Networks</b>	<b>17</b>
10.1 Perceptron (Single and Multi Layer) . . . . .	17
10.2 Deep Neural Network . . . . .	18
<b>11 Ensemble Classifiers</b>	<b>19</b>
11.1 Random Forest . . . . .	19
11.2 Bagging e Boosting . . . . .	19
<b>12 Time Series Analysis</b>	<b>20</b>
12.1 Data Analysis and Preparation . . . . .	20
12.2 Motif e Anomaly Discovery . . . . .	20
12.3 Shapelets . . . . .	21
12.4 Clustering . . . . .	21
12.5 Time Series Classification: Univariate . . . . .	22
12.5.1 Shapelet Classifier . . . . .	22
12.5.2 Shapelet Distance-Based Classifier . . . . .	23
12.5.3 Feature-Based Classifier . . . . .	23
12.5.4 Dynamic Time Warping Classifier . . . . .	23
12.5.5 Convolutional Neural Network . . . . .	23
12.5.6 Recurrent Neural Networks With LSTM . . . . .	24
12.5.7 Conclusioni . . . . .	24
12.6 Time Series Classification: Multivariate . . . . .	24
12.7 Time Series Forecasting . . . . .	25
<b>13 Frequent Pattern Mining</b>	<b>27</b>
<b>14 Outliers Detection</b>	<b>28</b>
<b>15 Explainability</b>	<b>29</b>

# 1 Data Understanding and Preparation

Il Dataset fornito descrive le caratteristiche degli ambienti di lavoro, al fine di evidenziarne lo stato di occupazione da parte dei lavoratori.

Vengono forniti un file di Training e due files di Test. Per rendere il Training più efficace, è stato deciso di creare un unico dataset dal quale, di volta in volta e con tecniche appropriate, estrarre i Training, Validation e Test Set. Unendo i dataset si ottiene un numero totale di record pari a 20560.

Nella seguente tabella (Tabella 1) sono presenti i dettagli di ogni attributo.

Tabella 1: Dataset attributes

Tipo attributo	Attributi
Binario (numerico)	Occupancy (classe target)
Continuo (numerico)	Temperature, Humidity, Light, CO2, HumidityRatio
Datetime	Date

Dall'analisi dei dati è emerso che i secondi relativi all'orario assumono valori pari a 00 o 59. Le rilevazioni avvengono quindi a distanza di un minuto e, per un errore di precisione, a volte non al minuto esatto. Per questo, tutti gli orari sono stati arrotondati al minuto più vicino, di fatto rendendo trascurabili i secondi.

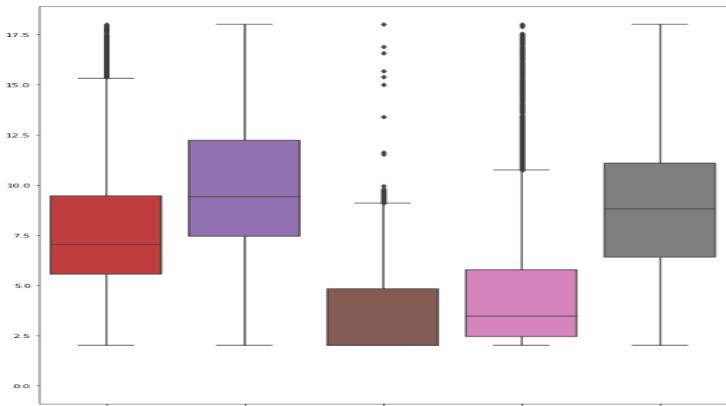
Sono stati creati nuovi attributi: **Day** e **Hour** per isolare giorno e orario dalla data, visto che tutte le date condividono lo stesso mese e anno, e **Weekend**, binario, che indica se la data appartiene al weekend o meno. È stato aggiunto anche l'attributo **TimeSlot**, categoriale con rappresentazione numerica, per associare a ogni orario la fascia oraria di appartenenza, sulla base di una suddivisione delle 24 ore in quattro parti uguali a partire da mezzanotte. Infine, **WorkingHour**, binario, che indica se l'orario corrisponde a un orario lavorativo. Per stabilire quale fosse il range dell'orario lavorativo, sarebbe stato perlomeno necessario conoscere il contesto del Dataset, in modo tale da basarsi sull'orario lavorativo comunemente adottato in quel determinato contesto o cultura. Questa informazione non è presente, per cui il range è stato stabilito in base a un'analisi preliminare dell'andamento della classe target in base all'orario stesso.

In linea generale si è cercato di costruire modelli sulla base delle feature preesistenti, osservando l'eventuale miglioramento delle performance utilizzando anche quelle derivate. In particolar modo, si evidenzia che la feature *WorkingHour* ha dato soltanto la possibilità di effettuare verifiche ulteriori senza la pretesa di essere determinante per la creazione dei modelli. Essendo stata determinata a partire dall'andamento della variabile dipendente stessa, avrebbe rischiato di compromettere la bontà del processo.

Il dataset ha subito un preprocessing tramite una normalizzazione Min-Max su tutti gli attributi continui. Questo è utile per determinati tipi di modelli, in particolar modo se sfruttano le distanze, mentre per altri, come ad esempio il Naive Bayes, la normalizzazione è irrilevante. Tuttavia, per omogeneità, una volta preprocessato il dataset è stato così utilizzato per qualsiasi tipo di modello. Laddove ulteriori modifiche specifiche si rendano necessarie, saranno dettagliate nei capitoli corrispondenti.

Per tutti i modelli illustrati nei capitoli successivi, si è deciso di lasciare da parte una percentuale tra il 20 e il 30% del dataset per il Test Set, in modo tale che fossero dati mai visti prima e non condizionassero il modello, mentre la restante parte dei dati ha composto il Training Set sul quale è stato effettuato il tuning dei parametri tramite le tecniche di RandomSearch/GridSearch con Cross Validation.

Tramite boxplot è stato possibile rilevare potenziali outlier per l'attributo **Light** (Figura 1).



I presunti outlier sono stati estratti e sono risultati pari a 9, ovvero una quantità irrisoria e statisticamente irrilevante rispetto all'intero dataset. È stata confrontata la correlation matrix ottenuta dal dataset preprocessato con quella ottenuta dallo stesso dataset ma con i valori dell'attributo Light dei presunti outlier sostituiti con quelli della mediana, con differenze impercettibili, come previsto. È stato quindi deciso di utilizzare il dataset originale, mantenendo inalterati i pochi suddetti outlier.

Figura 1: Boxplot

## 2 KNN

Per applicare il *K-Nearest Neighbour Classifier*, il valore del parametro  $k$  è stato selezionato sottoponendo il training test all'algoritmo della GridSearchCv con cross validation pari a 5 e un'analisi di valori di  $k$  in un range da 0 a 200. Il valore ottimale di  $k$  restituito è pari a 5. La classificazione con tale parametro sul test set, originato da uno split del dataset pari al 20%, ha restituito i risultati in Tabella 2.

Tabella 2: Classification Report

	Precision	Recall	F1-Score
0	0.998	0.997	0.997
1	0.991	0.993	0.992
<b>Accuracy</b>			0.996

Per un'ulteriore conferma della validità del parametro  $k$  utilizzato, sono stati testati anche altri valori per  $k$  pari a 15, 30, 50 e 150. Nel grafico (Figura 2) vengono messi in relazione i risultati dell'accuracy e F-Measure (asse  $y$ ) e il numero di  $k$  testati (asse  $x$ ). Dal grafico si osserva come, con un valore di  $k$  pari a 5, sia l'accuracy che l'F1 raggiungano i picchi più alti congiuntamente, per poi notare dei decrementi, in particolar modo della F-measure.

Figura 2: Andamento indici

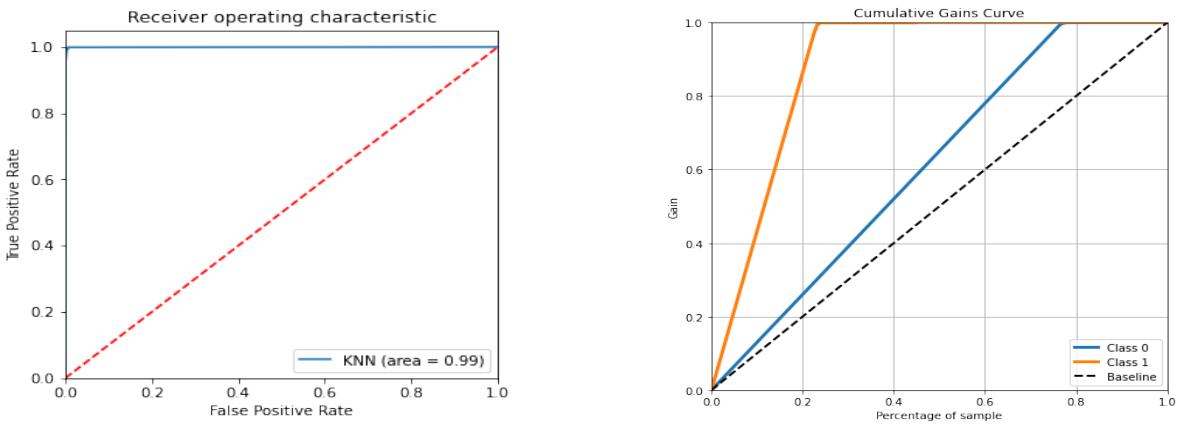
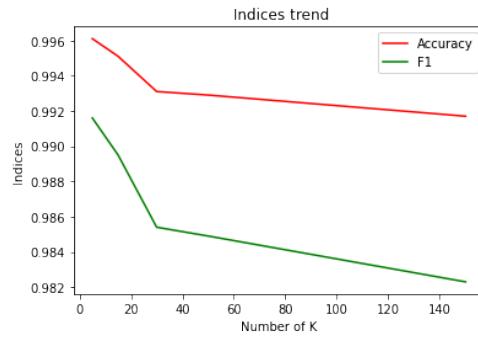


Figura 3

La ROC Curve (Figura 3a) evidenzia che il classificatore agisce in modo ottimale: il picco si presenta subito, per poi mantenersi costante senza ulteriori incrementi.

Un'ulteriore conferma della bontà del classificatore è presente nel *Lift Chart* (Figura 3b). La curva si tiene più alta della baseline, soprattutto per la classe 1 le cui performance sono ottimali utilizzando anche solo il 25% del dataset, mentre la classe 0 con il 75% circa.

### 3 Naïve Bayes

#### 3.1 NB Categoriale

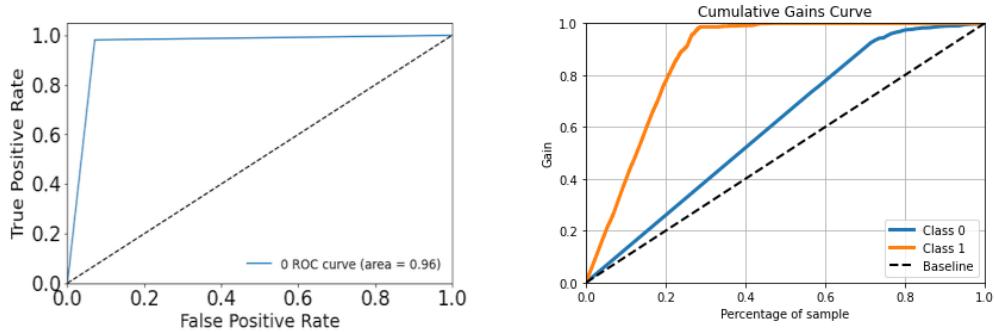
In un primo tentativo sono stati rimossi tutti i campi continui, lasciando dunque il Data Set sprovvisto dei suoi dati originali e contenente solo i nuovi campi generati. Dopo aver applicato il classificatore sono stati registrati i seguenti risultati sul Test Set. Successivamente è stata anche effettuata una Cross-Validation (Tabella 3).

	<b>Accuracy</b>	<b>F1-Score</b>
<b>Test Set</b>	0.9411	0.9604 - 0.8851
<b>Cross Validation</b>	0.9407 (+/- 0.01)	0.9223 (+/- 0.02)

Tabella 3: Categorial NB Results

Il modello si è dunque rivelato molto prestante nel classificare le istanze di test, nonostante abbia subito una notevole riduzione di dati di supporto. Tale assunzione è supportata dalla ROC Curve e dal Lift Chart, in particolar modo per quanto concerne la classe 1 (Figura 4):

Figura 4: Categorial without Continuous Features



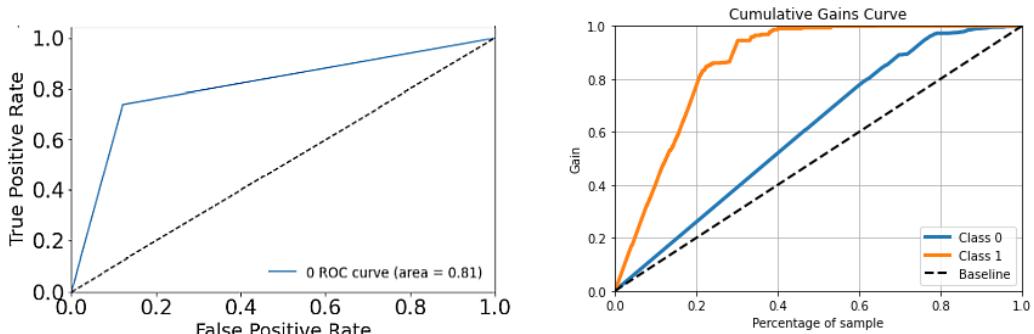
Successivamente è stato applicato un processo di discretizzazione dei dati continui appartenenti al Data Set originale (nbins=10, strategy='uniform'). Il classificatore è stato dunque riapplicato ottenendo le performances in Tabella 4.

	<b>Accuracy</b>	<b>F1-Score</b>
<b>Test Set</b>	0.8463	0.8978 - 0.6893
<b>Cross Validation</b>	0.9159 (+/- 0.02)	0.8853 (+/- 0.02)

Tabella 4: Categorial NB with Discretization Results

Confrontando i due modelli si nota come il primo sia decisamente più performante, fornendo valori di accuratezza nettamente superiori. Anche qui si forniscono Roc Curve e Lift Chart (Figura 5).

Figura 5: Categorial with Discretization



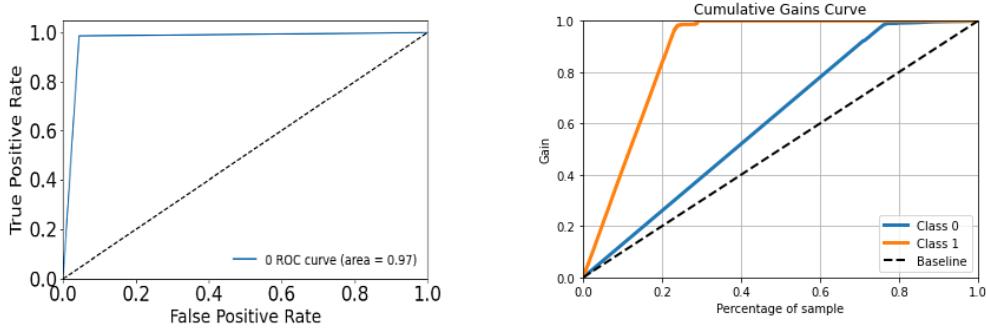
### 3.2 NB Gaussiano

Nel caso di un classificatore a distribuzione Gaussiana, il Data Set è stato classificato prima considerando anche le nuove features generate, poi rimuovendole. In entrambi i casi il classificatore ha ottenuto ottimi risultati. Si riportano di seguito le performances osservate:

	Accuracy	F1-Score
<b>Test Set</b>	0.9632	0.9756 - 0.9253
<b>Cross Validation</b>	0.9601 (+/- 0.09)	0.9491 (+/- 0.10)

Tabella 5: Gaussian NB with all Features

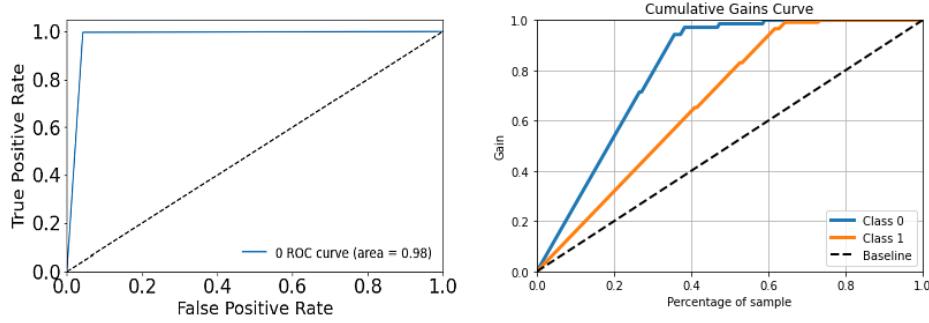
Figura 6: Gaussian with Full Data Set



	Accuracy	F1-Score
<b>Test Set</b>	0.9667	0.9779 - 0.9327
<b>Cross Validation</b>	0.9329 (+/- 0.18)	0.9216 (+/- 0.19)

Tabella 6: Gaussian NB without Discrete Features

Figura 7: Gaussian without Discrete Features



I due classificatori risultano essere abbastanza performanti, seppur il secondo sia calato in accuratezza durante la fase di validazione e il lift chart indichi una curva meno ripida del guadagno ottenuto.

### 3.3 Considerazioni

Confrontando i vari classificatori, si può dedurre come il Naïve Bayes sia una metodologia efficace per la classificazione del Data Set analizzato, con risultati particolarmente ottimali nel caso in cui:

- Si analizzi il Data Set con classificatore Categoriale, rimuovendo le features continue;
- Si analizzi il Data Set con classificatore Gaussiano, considerando l'intero set di features disponibili.

## 4 Logistic Regression

Per creare differenti modelli di regressione logistica in base alle feature di partenza selezionate, sono stati ottenuti ogni volta i parametri ottimali tramite Grid Search sul training set, sfruttando la cross validation con  $k$  pari a 5.

I parametri testati sono stati relativi alle penalità (*Lasso* e *Ridge*) e alla regolarizzazione (con parametro  $C$  pari a 1, 1.2 e 1.5).

La regressione logistica è stata eseguita inizialmente considerando come variabili indipendenti ciascuna feature del dataset presa separatamente.

Si è visto come, in questo modo, quasi tutte le feature abbiano determinato dei modelli che, applicati al test set, hanno portato a risultati per niente o poco soddisfacenti, tranne la feature *Light*, i cui risultati non solo sono stati soddisfacenti ma addirittura ottimali, come da Figura 8 e Tabella 7.

Figura 8: Logistic Regression - ROC Curve and Lift Chart

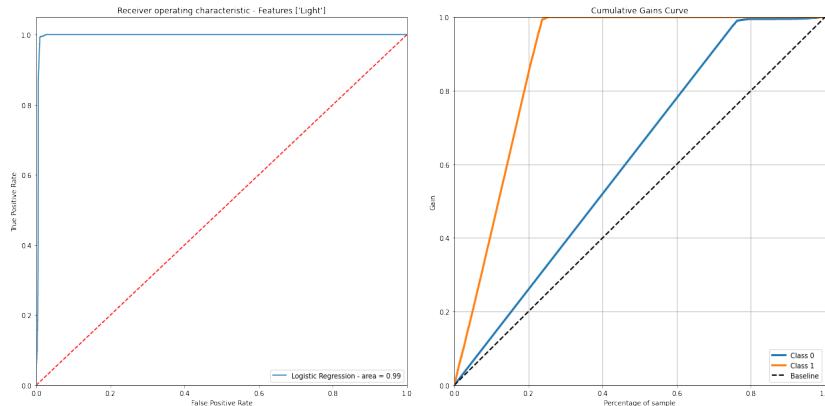


Tabella 7: Classification Report

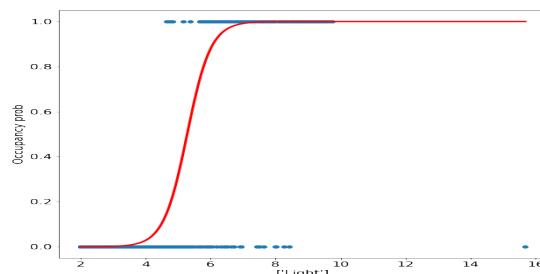
	Precision	Recall	F1-Score
0	1.00	0.99	0.99
1	0.96	0.99	0.98
<b>Accuracy</b>			<b>0.99</b>

Per questo motivo sono stati creati, per completezza e per verificare se questo modello fosse migliorabile, altri due modelli, a partire ovviamente dall'inclusione di questa feature. In uno è stata aggiunta soltanto la feature *Weekend*, nell'altro tutte.

Il secondo modello così creato è risultato migliore rispetto al primo, così come il terzo è risultato migliore rispetto al secondo, di pari passo con l'aumento delle feature. Tuttavia il miglioramento, essendo la performance del primo modello già ottimale, è irrisonio, per cui in un dataset di dimensioni elevate sarebbe opportuno utilizzare la regressione logistica sulla base della sola feature *Light*, ottenendo un buon rapporto tra performance e costi computazionali.

Nella Figura 9 è possibile osservare la curva di regressione per l'attributo *Light*.

Figura 9: Sigmoid logistic curve



## 5 Decision Tree

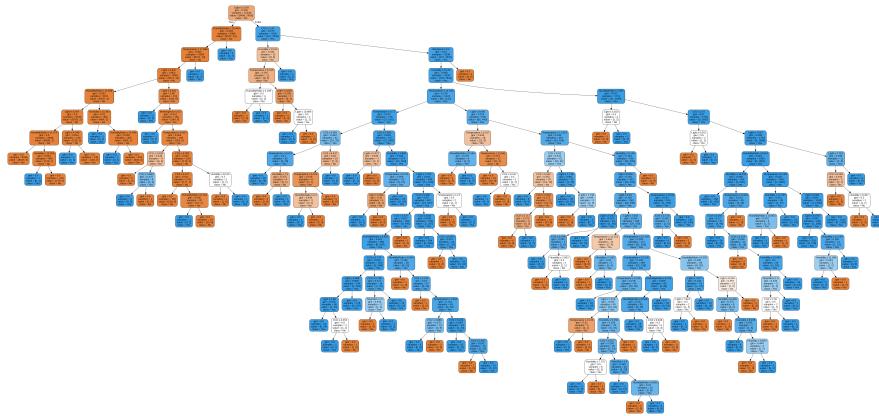
Per applicare il classificatore Decision Tree, è stato innanzitutto valutato il set di features da considerare nel modello: i campi **Date** e **Hour** sono stati rimossi, in quanto non appartengono ad un dominio continuo o discreto e quindi poco utili alla classificazione. La loro binarizzazione è stata scartata a priori, in quanto avrebbe aumentato in modo imponente la dimensionalità del Data Set. Anche il campo **Day** è stato rimosso, in quanto ritenuto troppo legato alla dimensione temporale del fenomeno. Dopo aver adattato il modello ai dati di training, è stata stampata l'importanza data ad ogni feature dal decision tree di partenza (Figura 10):

Figura 10: Features Importance

<b>TimeSlot</b>	<b>0.0004798018468247176</b>
<b>Weekend</b>	<b>0.001696970175045711</b>
<b>WorkingHour</b>	<b>0.00043019625092287226</b>
<b>Temperature</b>	<b>0.011228702460329216</b>
<b>Humidity</b>	<b>0.007884718909810814</b>
<b>Light</b>	<b>0.954087485617062</b>
<b>CO2</b>	<b>0.017788570437608977</b>
<b>HumidityRatio</b>	<b>0.006403554302395836</b>

Si noti come alla feature **Light** venga assegnato un valore di importanza rilevante, considerando meno le altre. Tale dato viene confermato dalla correlation matrix, dove la feature Light ha un grado elevatissimo di correlazione con la classe **Occupancy**. L'albero di decisione ottenuto è esteso e profondo, delineando una complessità del modello generato e una sua scarsa comprensibilità (Figura 11, formato immagine adatto allo zoom digitale):

Figura 11: First Decision Tree



Le prestazioni, inoltre, sembrano indicare un overfitting sui dati di training (Tabella 8):

Tabella 8: Training Set Classification

	Precision	Recall	F1-Score
No	1	1	1
Yes	1	1	1
<b>Accuracy</b>			<b>1</b>

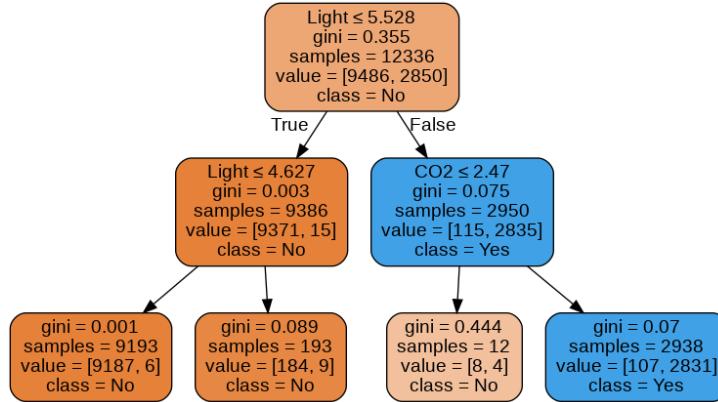
Per validare il modello, oltre alla classificazione sul Test Set, è stata dunque effettuata una cross-validation sull'intero Data Set, settando il parametro *cv* a 10. I risultati hanno rivelato un decremento delle performance del modello, portando i valori di accuracy a circa 0.90 in entrambi i casi, confermando quindi una situazione di overfitting. Per quanto concerne l'ottimizzazione del modello, è stata impiegata una strategia di *Hyperparameter optimization*, utilizzando una *Grid Search* ed una *Random Search*. Il risultato di tale ottimizzazione ha suggerito il seguente set di parametri (Figura 12):

Figura 12: Grid/Random Search Results

<b>Model with rank: 1</b>
<b>Mean validation score: 0.989 (std: 0.009)</b>
<b>Parameters: {'min_samples_split': 5, 'min_samples_leaf': 10, 'max_depth': 2}</b>

Dopo aver ottimizzato i parametri, è stato ottenuto il seguente decision tree (Figura 13):

Figura 13: Second Decision Tree



L'albero generato è di gran lunga più comprensibile e meno complesso del precedente, inoltre riporta delle performance più che buone (Tabella 9):

Tabella 9: Optimized Decision Tree on Test Set

	Precision	Recall	F1-Score
No	1	0.99	0.99
Yes	0.96	0.99	0.98
<b>Accuracy</b>		0.989	

Anche la cross validation, nuovamente effettuata, ha confermato come questo modello sia più efficiente del primo, con un calo dell'accuracy ad appena 0.97 e della metrica F1-score a 0.95.

Per una maggiore comprensione del modello è stato plottato il grafico della ROC Curve, dove l'AUC evidenzia un valore molto elevato, paragonabile all'accuracy del modello (Figura 14).

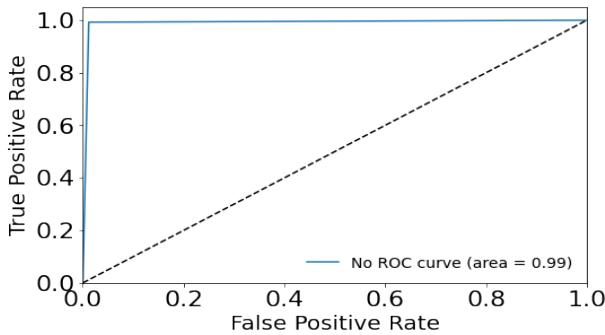


Figura 14: ROC Curve Second Decision Tree

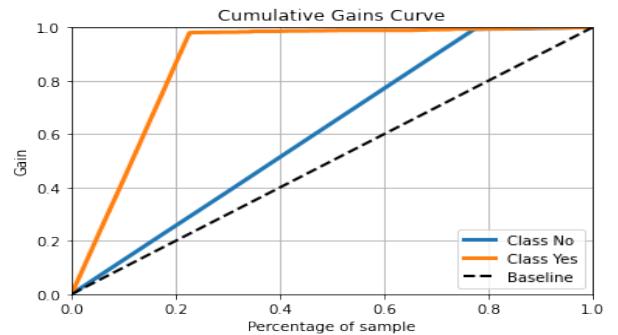


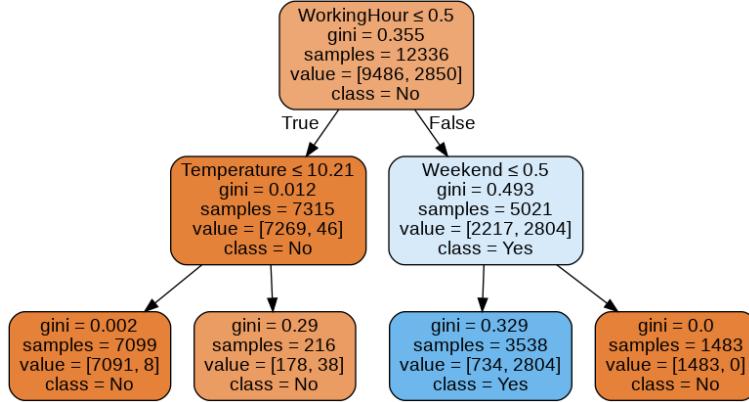
Figura 15: Lift Chart Second Decision Tree

Anche il Lift Chart evidenzia come, considerando una piccola porzione di dati, si abbia un sostanziale guadagno nella classificazione rispetto all'assenza di un modello (Figura 15).

A fini di sperimentazione, è stata rimossa la feature Light dal modello. Il decision tree generato ha attribuito la maggiore importanza ai campi **Weekend** e **Working Hour**, generando un modello che conferma come i campi generati negli step precedenti siano utili alla comprensione del fenomeno presente nel dataset:

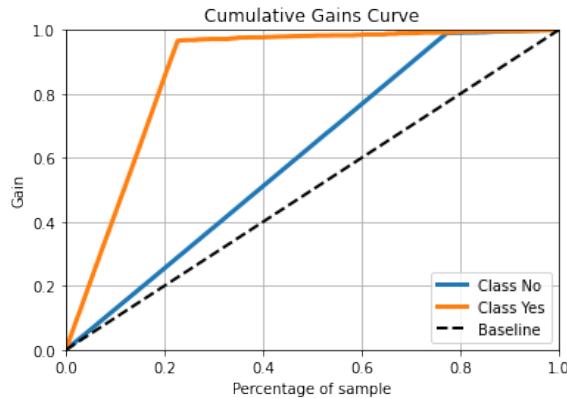
il valore della classe, infatti, non dipende solo dall'intensità di luce ma anche dal momento in cui la rilevazione viene effettuata. Oltre l'orario lavorativo (dalle 18 in poi per assunzione) e nei giorni appartenenti al weekend, infatti, l'albero classifica con la classe "No" i record.

Figura 16: Third Decision Tree



Anche il Lift Chart del nuovo albero evidenzia, come il precedente, un aumento del guadagno seppur leggermente inferiore.

Figura 17: Lift Chart Third Decision Tree



Il nuovo modello ha performance leggermente più basse del precedente, tuttavia risulta più che accettabile e, soprattutto, esplicativo del fenomeno e del task considerati.

## 6 Regression

Il task di **Regressione Lineare Semplice** è stato testato su varie combinazioni di attributi continui appartenenti al Data Set, ma i modelli creati risultavano poco precisi dato il rapporto lineare limitato tra le features considerate. Per tale motivo il task è stato eseguito sulla coppia di features **Humidity - HumidityRatio** che, essendo legate tra loro mediante equazione matematica, produrranno sicuramente un modello di regressione migliore rispetto agli altri.

Il task è stato sviluppato attraverso 4 tecniche di regressione differenti, ovvero Regressione **Lineare Semplice, Ridge, Lasso ed Elastic Net**.

Per le ultime tre tecniche, è stata applicata una cross-validation per determinare quale fosse il valore di **alpha** migliore da utilizzare, così da massimizzare i risultati ottenuti:

	<b>R2</b>	Mean squared error	Mean Absolute error	Best Alfa
<b>Lineare Semplice</b>	0.869	1.370	0.9252	
<b>Ridge</b>	0.869	1.3701	0.9252	1
<b>Lasso</b>	0.8693	1.3701	0.9252	0.0001
<b>Elastic Net</b>	0.8693	1.3701	0.9252	0.0001

Tabella 10: Linear Regression Performances

I risultati tra le tre tecniche sono pressochè identici ad eccezione del coefficiente e dell'intercetta calcolati ma non riportati per non rendere l'esposizione troppo complessa. Si noti come siano stati testati diversi valori di alpha, constatando una diminuzione delle prestazioni ottenute. La funzione di regressione viene riportata in Figura 18.

Per meglio visualizzare i risultati della tecnica Elastic Net è stata plottata la funzione di regressione (Figura 19) applicata su un test set molto ridotto (perc = 0.003).

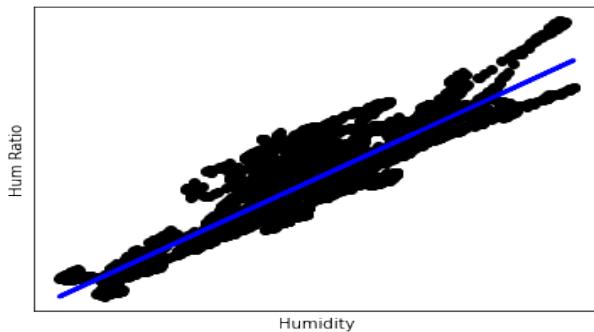


Figura 18: Simple Linear Regression

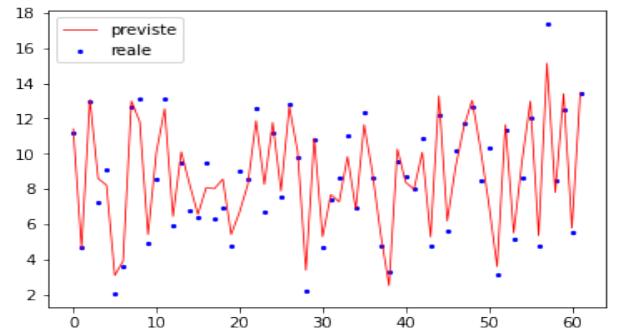


Figura 19: Elastic Net Regression

Successivamente il task di regressione è stato esteso ad una **Regressione Lineare Multipla**, applicando lo stesso procedimento. I risultati ottenuti sono i seguenti:

	<b>R2</b>	Mean squared error	Mean Absolute error	Best Alfa
<b>Lineare Semplice</b>	0.887474	0.0202	0.0844	
<b>Ridge</b>	0.887474	0.0202	0.0844	1
<b>Lasso</b>	0.887462	0.0202	0.0843	0.0001
<b>Elastic Net</b>	0.887470	0.0202	0.0843	0.0001

Tabella 11: Multiple Regression Performances

Com'è possibile notare, il valore di **R2** è il medesimo nell'approccio semplice e Ridge. Il metodo Lasso ha invece il valore più basso, fattore che avrà sicuramente condizionato l'Elastic Net che, essendo una combinazione tra Lasso e Ridge, ottiene un'accuratezza leggermente minore della massima osservata. Si può dunque concludere affermando che il modello di regressione ricreato goda di una precisione più che discreta ma non ottimale, nonostante le varie metodologie ed ottimizzazioni applicate.

## 7 Imbalanced Learning

Nel dataset originale la classe di maggioranza della variabile dipendente *Occupancy* è negativa e corrisponde a circa il 77% dei record.

Per rendere il dataset fortemente sbilanciato con circa 95% dei record per la classe di maggioranza, occorre applicare oversampling sulla stessa oppure undersampling sulla classe di minoranza. È stato deciso di applicare undersampling (con modalità random) per mantenere i dati del dataset inalterati seppur ridotti, rispetto all'oversampling che ne avrebbe artificialmente introdotti di nuovi.

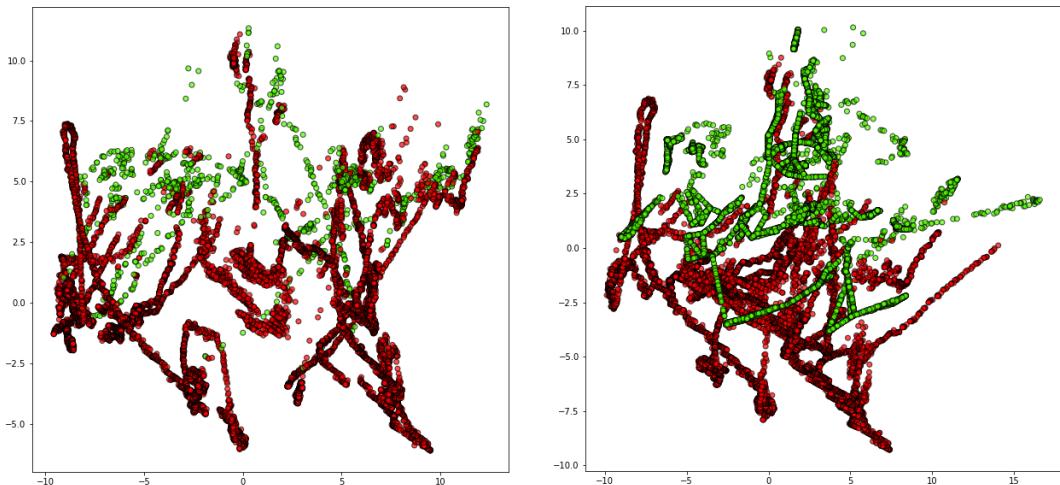
Per ribilanciare il dataset, sono state usate quattro tecniche diverse, creando i rispettivi dataset: **Random Oversampling**, **Random Undersampling**, **ADASYN**, **Near Miss**.

A causa dei calcoli sulle distanze previsti dagli ultimi due algoritmi, non è stato possibile mantenere l'attributo *Date* e i suoi derivati che erano presenti nel dataset preprocessato. Invece è stato possibile mantenerli con l'utilizzo delle tecniche di base di random oversampling e undersampling.

È stato deciso di ribilanciare perfettamente il dataset con una distribuzione delle classi del 50%.

Sono stati osservati, per completezza, sia i risultati dei modelli sui dataset ribilanciati che quelli sul dataset fortemente sbilanciato. Entrambe le tipologie di osservazioni sono state confrontate rispetto a quelle relative al dataset di partenza.

Figura 20: PCA: Imbalanced dataset, Rebalanced dataset with AdaSyn



### KNN

Confrontando i risultati del KNN Classifier sui vari dataset **ribilanciati**, possiamo osservare come i risultati siano abbastanza simili tra loro, con alterazioni nell'ordine del millesimo. Solo il dataset sottoposto a *Nearmiss Undersampling* restituisce un risultato che si discosta dalla media degli altri (Accuracy e F1 pari 0.967).

Tuttavia, la tecnica di ribilanciamento che permette una migliore classificazione con il KNN è *Adasyn Oversampling*, dove il valore di  $k$  ottimale è 1, mentre l'accuracy è pari a 0.998417 e l'F1 a 0.998418.

Rispetto al dataset di partenza c'è un lieve miglioramento negli indici con un aumento di 0.002 per l'accuracy e di 0.004 per la F-Measure. Soprattutto, laddove nel dataset originale c'era una forte differenza per il numero di record necessari alla perfetta classificazione per ciascuna classe, con il nuovo dataset occorrono in maniera equilibrata, sia per la classe 0 che 1, il 50% circa dei campioni.

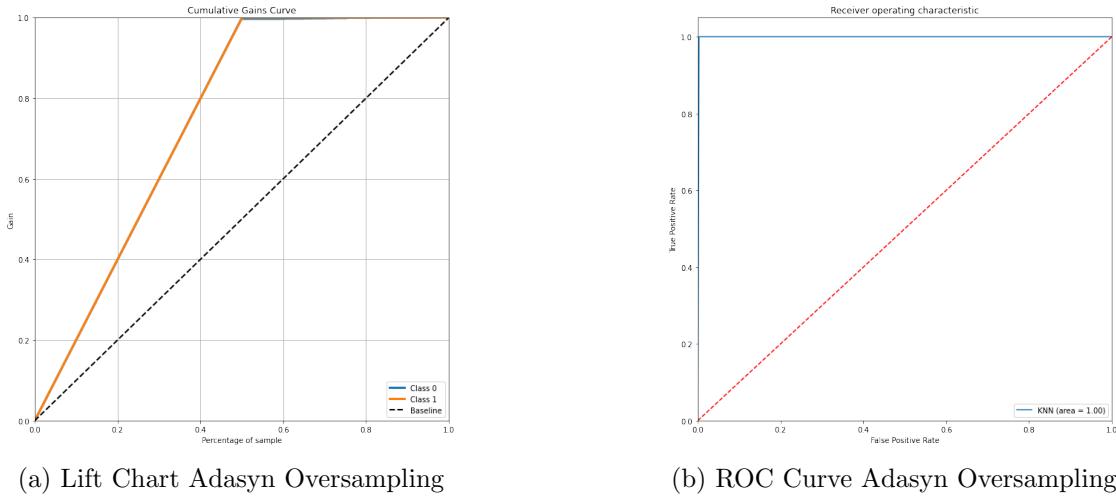


Figura 21

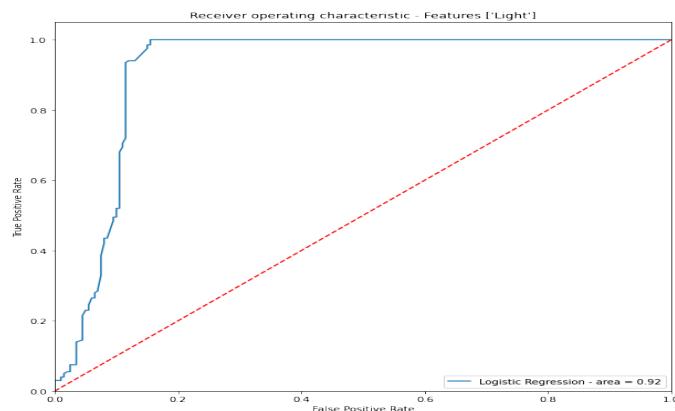
Per quanto riguarda la performance sul dataset **sbilanciato**, la Grid Search ha individuato un valore di  $k$  pari a 3, inferiore rispetto al dataset di partenza. Le metriche sulla classe di minoranza sono calate impercettibilmente, con un AUC Score pari a 0.98, inferiore di 0.01 rispetto a quello del dataset originale. Il modello si conferma quindi robusto in ogni caso.

## Logistic Regression

I risultati ottenuti sui dataset **ribilanciati** tramite random undersampling e oversampling sono pressoché identici tra di loro. Rispetto al dataset originale invece, è possibile osservare dei miglioramenti sui modelli ottenuti su feature singola. Laddove nel dataset originale tutto il test set veniva classificato come classe di maggioranza, in questo caso si ottengono risultati più equilibrati seppur non ottimali. Degno di nota è il modello costruito a partire dalla temperatura, che raggiunge un AUC Score pari a 0.85 e una precisione media pari a 0.83, maggiori rispettivamente di 0.17 e 0.13. Si confermano eccellenti i modelli ottenuti a partire da set di feature che includano almeno l'attributo *Light*.

I modelli basati sul dataset ottenuto con ADASYN Oversampling hanno dato risultati leggermente inferiori, mentre il Near Miss ha determinato i risultati in assoluto peggiori anche rispetto al dataset originale, con un leggero calo anche sulla feature *Light* (Figura 22)

Figura 22: Roc Curve - Near Miss

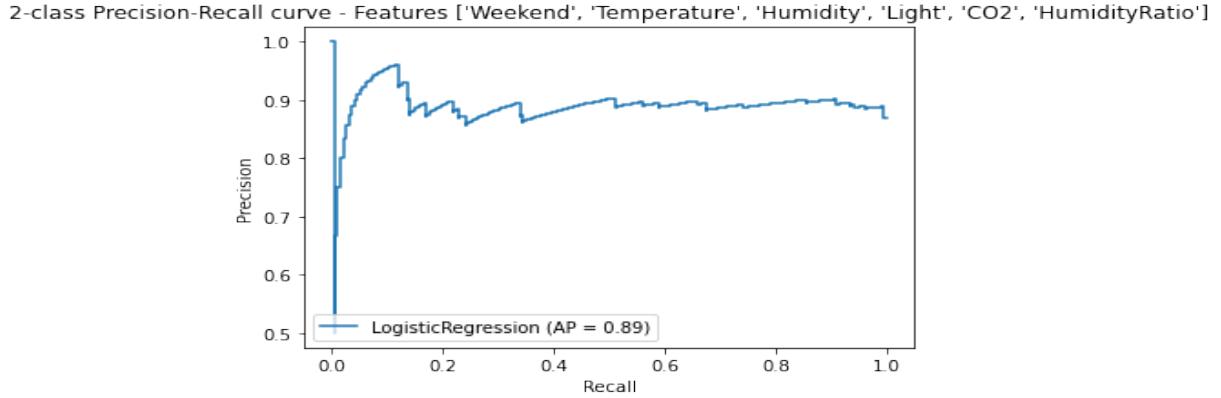


Per quanto riguarda la performance del dataset **sbilanciato**, l'accuracy dei modelli creati a partire da ogni singola feature è aumentata ma, nella maggior parte dei casi, la precision relativa alla classe di minoranza si è azzerata: tutto il test set o quasi, a seconda della feature scelta, viene classificato come classe di maggioranza.

Il modello creato a partire dalla feature *Light* è stato l'unico performante considerando anche la classe di minoranza. Rispetto al modello relativo al dataset originale, c'è stato un azzeramento dei falsi negativi e un leggero aumento, in proporzione alla grandezza del dataset, di falsi positivi.

Si noti a questo proposito la precision-recall curve ottenuta da quest'ultimo modello (Figura 23): riporta un valore di average precision pari a 0.89, contrapposto allo 0.97 del dataset originale . Le altre curve sono pressoché sovrapponibili alle precedenti.

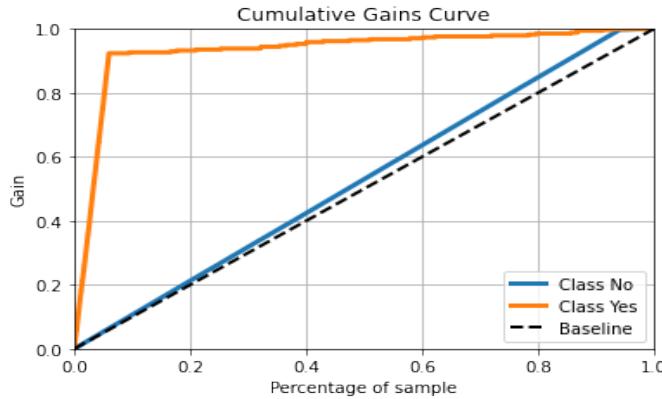
Figura 23: Precision-Recall Curve - Random Undersampling, Imbalanced Dataset



## Decision Tree

L'algoritmo è stato testato considerando i parametri di costruzione dell'albero presentati nel capitolo associato. Innanzitutto è stato provato il Data Set **sbilanciato**, il quale ha prodotto un albero leggermente differente da quello ottenuto in precedenza. Nel secondo livello di profondità, infatti, è stata considerata la feature *Humidity* come criterio di suddivisione al posto di *CO2*. Il modello ha però retto il confronto, ottenendo un'accuracy di 0.988 confermata successivamente dalla cross-validation. Il Lift Chart, tuttavia, ha mostrato come per la classe di maggioranza vi sia un guadagno più rapido come ci si aspettava, diminuendo di molto quello della classe di minoranza.

Figura 24: Lift Chart Embalanced Data Set (Decision Tree)



Successivamente l'algoritmo è stato testato sui vari Data Set **ribilanciati**, ottenendo per le tecniche di Random Oversampling, Random Undersampling e AdaSyn modelli dalle prestazioni identiche a quello originale, seppur con alberi leggermente differenti (ad esempio la tecnica di Undersampling ha considerato solo la feature **Light** come parametro di suddivisione). L'unico metodo che ha riscontrato una differenza è stato il Near Miss, il quale ha ottenuto un'accuracy più bassa rispetto agli altri metodi, ovvero di circa 0.90.

## Naive Bayes

Essendo stati testate due varianti di Naive Bayes, si riportano i risultati ottenuti in maniera distinta. Si noti come le tecniche NearMiss e AdaSyn non sia stato possibile effettuare un test sul set completo di

features, in quanto quelle generate in maniera calcolabile sono state rimosse in questi Data Set per i motivi spiegati pocanzi.

- Naive Bayes Gaussiano:

- Testando il Data Set sbilanciato con tutte le features si sono ottenute le stesse prestazioni di quello bilanciato, con una piccola differenza nel caso del test senza variabili discrete che ha riportato una diminuzione del valore di accuracy della cross validation da 0.96 a 0.93, forse dovuto ad un leggero overfitting sui dati.
- Testando il Data Set ribilanciato mediante AdaSyn, Random Oversampling e Random Undersampling le prestazioni non sono diminuite ma sono aumentate di 0.01 in termini di accuracy.
- Con la tecnica Near Miss è stata invece riscontrata una significativa diminuzione di Accuracy, che è calata fino a 0.86 sul test set e 0.82 nella cross-validation.

- Naive Bayes Categoriale:

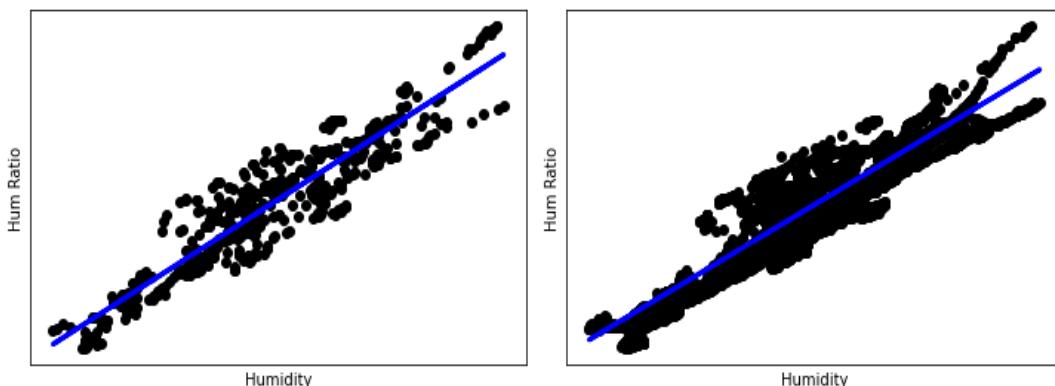
- Testando l'algoritmo sul Data Set sbilanciato le prestazioni si sono mantenute stabili, con un incremento dell'accuracy sul Test Set da 0.84 a 0.91 per quanto concerne il caso in cui le variabili continue sono state discretizzate.
- Le tecniche di Random Undersampling e Oversampling hanno mantenuto, in linea di massima, le prestazioni ottenute sul Data Set originale,
- Le tecniche Near Miss e AdaSyn, invece, hanno ottenuto prestazioni nettamente ridotte, con un'accuracy di 0.51 circa sia nel Test Set che in fase di validazione.

## Regression

Per quanto concerne il task di regressione, in questo report non verranno confrontati i risultati ottenuti con tecniche alternative come Ridge, Lasso o Elastic Search in quanto le performance sono simili tra loro e si creerebbe solo confusione nell'esposizione dei risultati.

- La Regressione applicata sul Data Set sbilanciato ha prodotto un valore di R<sup>2</sup> simile rispetto al Data Set originale, ad eccezione della Regressione Lineare Multipla che ha avuto un abbassamento della metrica da 0.88 a 0.74.
- Le tecniche di Random Oversampling e Undersampling hanno mantenuto prestazioni identiche per la regressione lineare, aumentando però la precisione del modello di regressione lineare multipla portando il valore di R<sup>2</sup> a 0.93. Anche AdaSyn ha ottimizzato la regressione lineare multipla allo stesso modo ma ha perso di prestazioni nella lineare semplice, riportando un valore di R<sup>2</sup> pari a 0.82.
- La tecnica NearMiss ha generato una funzione di regressioni valida per la regressione lineare semplice ( $R^2=0.88$ ), tuttavia ha prodotto una regressione lineare multipla davvero di scarsa qualità ( $R^2=0.58$ ).

Figura 25: Simple Lineare Regression on Undersampling and AdaSyn



## 8 Principal Component Analysis (PCA)

Il Data Set è stato, in una prima fase, convertito nelle sue prime due componenti principali considerando tutte le features, ad eccezione di **Date** e **Hour**, per i motivi già citati in precedenza. È stato ottenuto il plot bidimensionale in Figura 26.

Le due componenti principali sono state poi aggiunte al Data Set come nuove features, ritentando un task di classificazione per la scoperta di un eventuale aumento di performances. L'algoritmo considerato per la classificazione (decision tree) non ha riportato alcuna differenza nel modello proposto, rendendo quindi i due campi aggiunti del tutto inutili. Successivamente è stato creato un nuovo modello di classificazione basato esclusivamente sui due nuovi campi, ma i valori di tale classificatore si sono rivelati meno performanti rispetto al classico basato sul Data Set iniziale:

	Precision	Recall	F1 - Score
<b>0</b>	0.97	0.89	0.93
<b>1</b>	0.72	0.92	0.81
<b>Accuracy</b>	0.90		

Tabella 12: PCA Classification

In una seconda fase, per motivi sperimentali, è stato applicato l'algoritmo PCA ad un Data Set privato delle features generate nelle fasi precedenti, ovvero **Day**, **Weekend** e **WorkingHour**. Il plot è visibile in Figura 27.

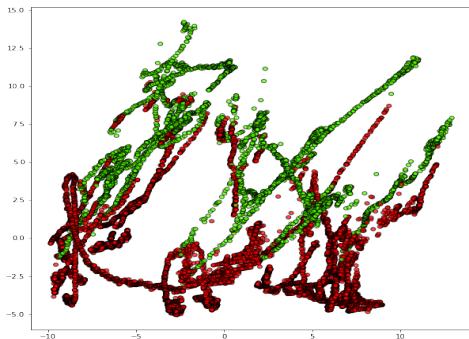


Figura 26: First PCA

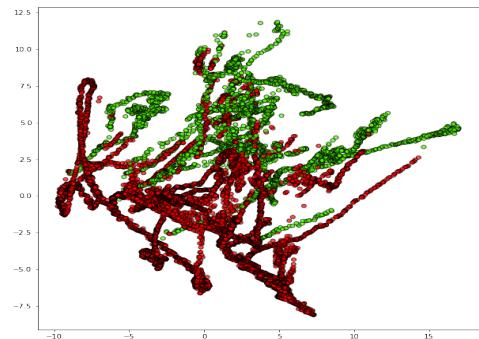
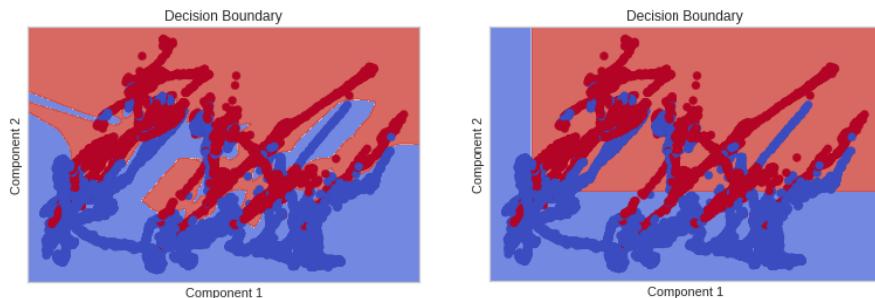


Figura 27: Second PCA

Come si evince dai due plot, i dati sono disposti in modo più comprensibile nel primo di questi, limitando le zone in cui dati appartenenti a classi differenti vengano posti in zone adiacenti. Dopo aver plottato le componenti principali del Data Set, ne è stato stampato il **Decision Boundary** utilizzando prima l'algoritmo K-NN (con k=5) e poi il decision tree già testato precedentemente (Figura 28):

Figura 28: Decision Boundaries (KNN, Decision Tree)



Com'è possibile notare il Decision Boundary del decision tree occupa un'area maggiore che comprende anche una gran quantità di dati appartenenti alla classe opposta, cosa che avviene in misura assai minore per quanto concerne l'algoritmo K-NN.

## 9 Support Vector Machine

### 9.1 Linear SVM

L'applicazione della Linear Support Vector Machine ha richiesto, come primo step, la ricerca del miglior valore del parametro C. A tale scopo è stata effettuata una Grid Search che ha dato come risultato ottimale 10.

Per testare tale risultato, la SVM Lineare è stata applicata utilizzando valori di C differenti, ovvero 0.001, 1, 10, 100. Si riportano i risultati ottenuti:

	<b>Accuracy</b>	<b>F1 - Score</b>	<b>CV Accuracy</b>	<b>CV F1-Score</b>
<b>C = 0.001</b>	0.9894	0.9931 - 0.9776	0.9894 (+/- 0.01)	0.9854 (+/- 0.01)
<b>C = 1</b>	0.9904	0.9937 - 0.9796	0.9899 (+/- 0.01)	0.9859 (+/- 0.01)
<b>C = 10</b>	0.9905	0.9938 - 0.98	0.9899 (+/- 0.01)	0.9859 (+/- 0.01)
<b>C = 100</b>	0.76978	0.8699 - 0.007	0.9701 (+/- 0.11)	0.9454 (+/- 0.23)

Tabella 13: Linear SVM Performances

Per C=100 l'algoritmo di ottimizzazione non ha ottenuto la convergenza nonostante il numero di iterazioni raggiunga il milione. Per tale motivo i risultati ottenuti per tale parametro non sono affidabili.

Si noti, invece, come per C=10 si ottengano risultati leggermente migliori rispetto agli altri, come ci si aspettava dalla Grid Search effettuata. Per tale classificatore si riporta la ROC Curve corrispondente, con un valore di AUC pari a 0.99 (Figura 31).

Per ridurre la dimensionalità del Data Set ed offrire una maggiore visibilità dei dati classificati, è stata nuovamente applicata la PCA, plottando anche i vettori di supporto (Figura 29).

Sulle sole componenti principali è stato nuovamente riapplicato il classificatore, del quale viene riportato il **Decision Boundary** ottenuto (Figura 30).

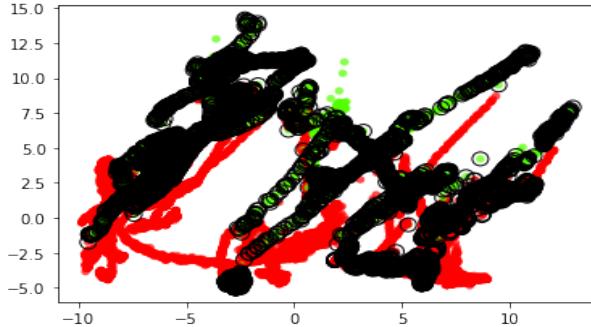


Figura 29: PCA with Support Vectors

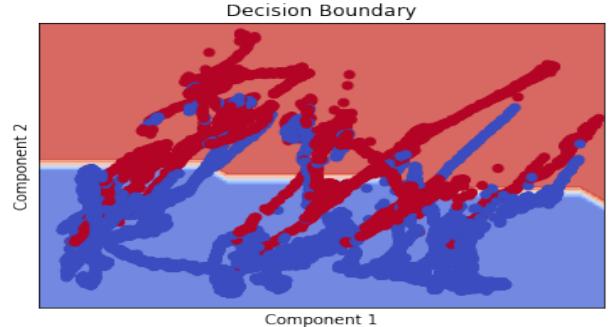


Figura 30: Linear SVM Decision Boundary

È possibile notare come una Support Vector Machine di tipo Lineare non sia adeguata nel marcare un Decision Boundary ottimale, come quello ottenuto ad esempio dal K-NN. Viene infatti delimitata un'area decisionale troppo generica rispetto al modo in cui i dati sono disposti nello spazio bi-dimensionale.

Le prestazioni del classificatore, testato sulle componenti principali, sono infatti calate:

	<b>Accuracy</b>	<b>F1-Score</b>
<b>Test Set</b>	0.8787	0.9232 - 0.7116
<b>Cross-Validation</b>	0.8777 (+/- 0.02)	0.8171 (+/- 0.03)

Tabella 14: Linear SVM Classification on PCA

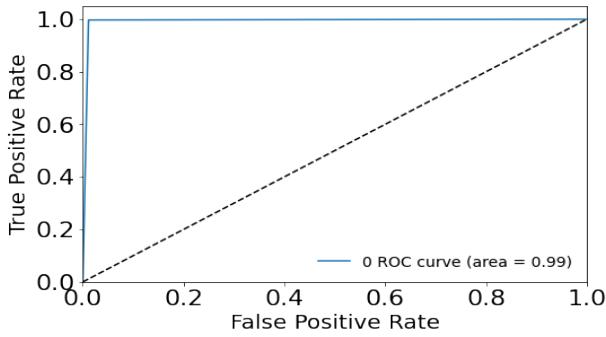


Figura 31: ROC Curve Linear SVM

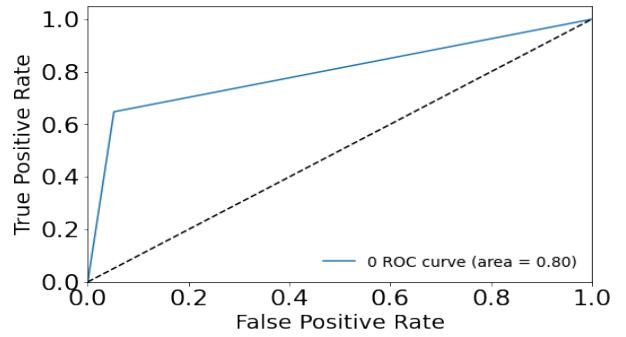


Figura 32: ROC Curve Linear SVM on PCA

## 9.2 Non-Linear SVM

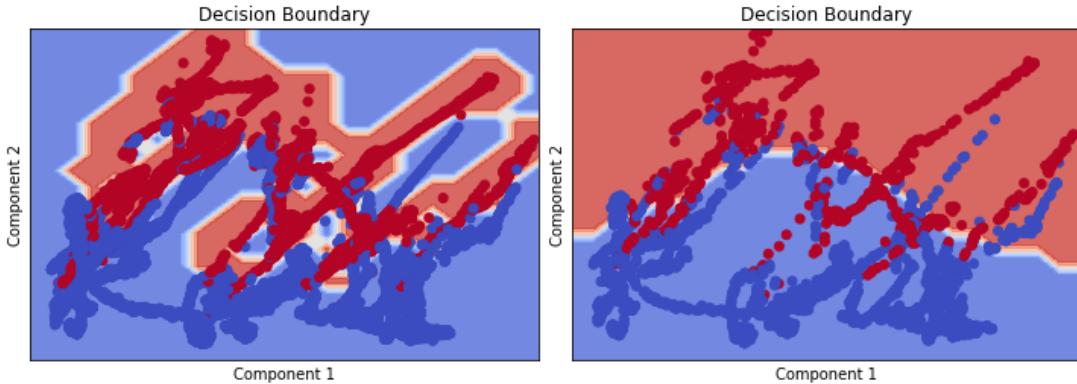
La SVM non Lineare è stata testata utilizzando il kernel **"RBF"**. E' stato testato anche il kernel **"Polynomial"** ma, dato l'elevato carico computazionale richiesto, non è stato possibile ottenere risultati sull'intero Data Set in tempi ragionevoli. Si riportano le performance del classificatore sul Data Set iniziale e sulle componenti principali:

	Accuracy	F1-Score
<b>Data Set</b>		
Test Set	0.9930	0.9954 - 0.9849
Cross Validation	0.9928 (+/- 0.00)	0.9898 (+/- 0.01)
<b>PCA</b>		
Test Set	0.96984	0.9803 - 0.9355
Cross Validation	0.9671 (+/- 0.01)	0.9538 (+/- 0.01)

Tabella 15: Non Linear SVM

Questa volta la riapplicazione di un classificatore SVM sulle componenti principali non ha riportato un calo di prestazioni drastico come per il caso lineare. Sono stati nuovamente plottati i Decision Boundaries ottenuti. Per il kernel Polynomial, la complessità computazionale richiesta non ha permesso una stampa del Decision Boundary sull'intero Data Set, avvenuta invece su una piccola porzione di questo (Figura 33):

Figura 33: RBF and Poly Kernel's Decision Boundary



Il Decision Boundary del kernel RBF si adatta in maniera molto più consona rispetto alla disposizione dei dati nello spazio, marcando zone definite in prossimità di punti appartenenti ad una classe piuttosto che ad un'altra.

Stesso discorso non è applicabile al kernel Polynomial, il quale ottiene un risultato non idoneo alla disposizione dei dati, come per la SVM Lineare. Si può quindi concludere affermando che l'utilizzo di una SVM non lineare con kernel RBF sia la combinazione migliore tra le SVM testate per produrre un Decision Boundary adeguato.

# 10 Neural Networks

## 10.1 Perceptron (Single and Multi Layer)

La creazione di un Perceptron efficiente ha richiesto svariate combinazioni tra **Activation Functions**, **Ottimizzatori** e variabili associate. In un primo momento è stato creato un Perceptron con un hidden layer contenente 30 nodi. La funzione di attivazione utilizzata è stata la **"tanh"** mentre si è usato l'algoritmo di ottimizzazione **"sgd"** con momentum pari a 0.4 e learning rate pari ad 1. La rete è stata addestrata per 500 epoche con un batch di 1000 record per iterata. La rete è stata validata mediante validation set, estratto dal training set. Il risultato finale è stato molto deludente, in quanto l'accuracy è calata drasticamente col passare delle epoche. Inoltre, il grafico della **Loss Curve** ha mostrato come il modello vada in overfitting. Anche l'applicazione del modello sul Test set ha dimostrato tale condizione con un'Accuracy di appena 0.230 (Figura 34).

È stata dunque creata una nuova rete neurale, composta nuovamente da un solo hidden layer con 5 nodi, utilizzando la funzione di attivazione **"sigmoid"**, notoriamente nota per avere buone performance in casi di classificazione binaria. Come ottimizzatore è stato usato **"adam"** con learning rate pari a 0.001. Il numero di epoche, batch e validazione rimangono le medesime. In questo caso la rete è riuscita a convergere verso una classificazione ottimale e non in condizione di overfitting, con un'accuracy pari a 0.99, confermata dalla classificazione del test set (Figura 35).

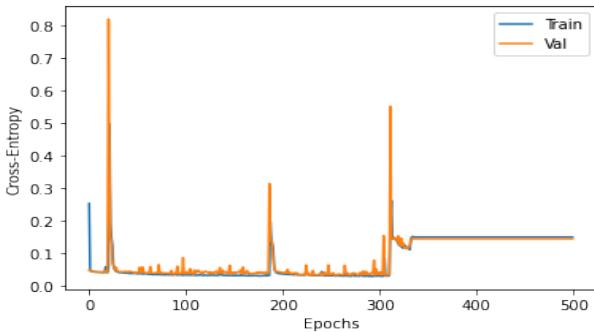


Figura 34: Loss Curve First Perceptron

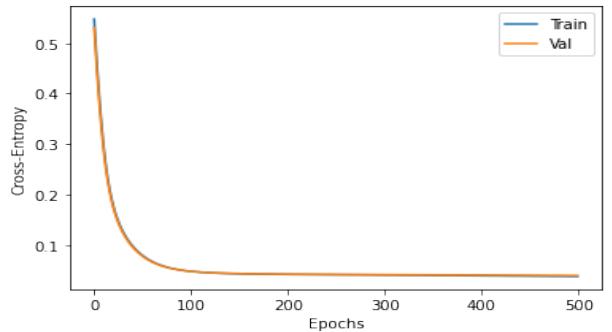


Figura 35: Loss Curve Second Perceptron

Con i medesimi parametri sono stati implementati altri due Perceptron senza strato intermedio, i quali hanno ottenuto performance praticamente identiche alle loro estensioni. Sono stati effettuati altri tentativi con combinazioni differenti tra le varie opzioni disponibili, ma tutte hanno portato il modello in overfitting o in uno stadio di apprendimento meno evoluto di quello riportato. Per tale motivo è possibile affermare come il set di parametri proposto sia ottimale per la generazione di una rete neurale efficace sul Data Set analizzato.

Per ottimizzare ulteriormente il modello sono state applicate tecniche di **"Early Stopping"** e **"Regolarizzazione"**, quest'ultima mediante **"Kernel e Drop Regularization"**. Il criterio di early stopping è stato bilanciato in base al variare della loss curve, interrompendo l'apprendimento dopo che per dieci epoche consecutive il suo valore non fosse variato per un delta pari a 0.05. L'apprendimento della rete è stato interrotto dopo 30 iterate circa, producendo un modello con accuracy pari a 0.987. Si è quindi ottenuto un modello dalle medesime prestazioni ma con una fase di addestramento decisamente più breve, a conferma della bontà del modello stesso.

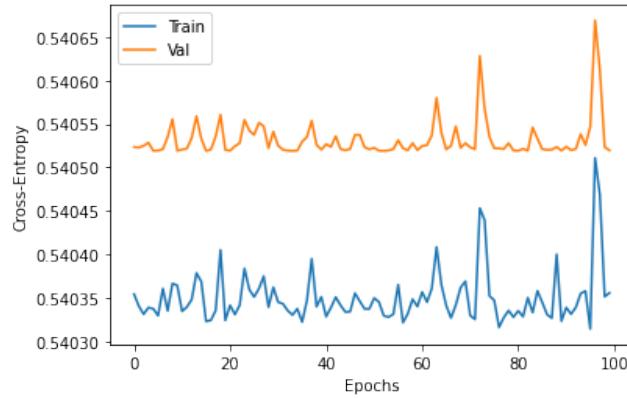
Per quanto riguarda la Kernel Regularization, invece, è stato introdotto nel nodo intermedio un regolarizzatore con valore  $l2=0.00001$ . I risultati ottenuti sono stati i medesimi ma, a fini sperimentali, il valore di  $l2$  è stato ridotto. Si è notato come all'aumentare del regolarizzatore la convergenza sia stata più lenta ma comunque raggiunta.

Con la Drop Regularization, invece, la rete è andata in underfitting, sicuramente a causa della poca complessità della rete la quale, privata iterativamente dei propri nodi, non ha consentito un apprendimento adeguato.

## 10.2 Deep Neural Network

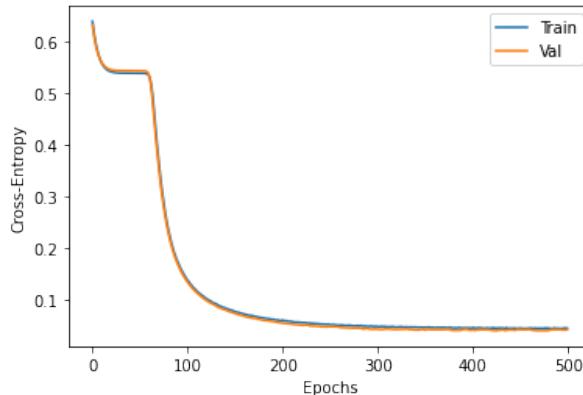
La creazione di una Deep Neural Network si è basata sulle conoscenze derivate dalla creazione del Perceptron. È stata quindi creata una prima rete neurale composta da otto hidden layers, ottimizzatore SGD e funzioni di attivazioni miste, scelte tra sigmoid, softmax e relu. Il modello ottenuto è palesemente in overfitting, risultato dimostrato anche dalla classificazione sul test set, che ha riportato un'accuracy di 0.23 (Figura 36):

Figura 36: Loss Curve First DNN



A questo punto la rete è stata modificata utilizzando come funzione di attivazione la sigmoid e come ottimizzatore adam. Per testare le capacità della rete sono stati inseriti due regolarizzatori Drop Out nei primi layer (Figura 37):

Figura 37: Loss Curve Second DNN



Il grafico mostra come il modello non sia in overfitting, risultato confermato dalla classificazione del Test Set, che ha riscontrato un'accuracy di 0.99. Curiosa è la forma della curva, che ferma la sua discesa per un certo range di epoche per poi riprendere e convergere correttamente. Probabilmente la rete stava convergendo verso un minimo locale, dal quale è poi fuggita grazie all'ottimizzatore adam con momentum, noto per questa sua peculiarità.

Si può dunque affermare quanto detto per il Perceptron, considerando la combinazione sigmoid-adam come ottimale per generare una qualsivoglia rete neurale per la classificazione del Data Set in questione.

## 11 Ensemble Classifiers

### 11.1 Random Forest

Il modello Random Forest è stato prima applicato con un numero di Decision Tree e di profondità massima pari a 3, ottenendo così un'accuracy pari a 0.94 e un F1-score pari a 0.87. Tramite Random Search è stato individuato un modello con parametri diversi, tra i quali un numero di alberi di decisione pari a 25. Le performance sono migliorate, ottenendo valori per l'Accuracy e per l'F1 rispettivamente di 0.99 e 0.98. Essendo le performance del più semplice dei decision tree molto buone, l'applicazione di Random Forest non rende possibile apprezzare miglioramenti significativi.

Nella figura 38 è possibile osservare il ranking degli attributi nel training e nel test set.

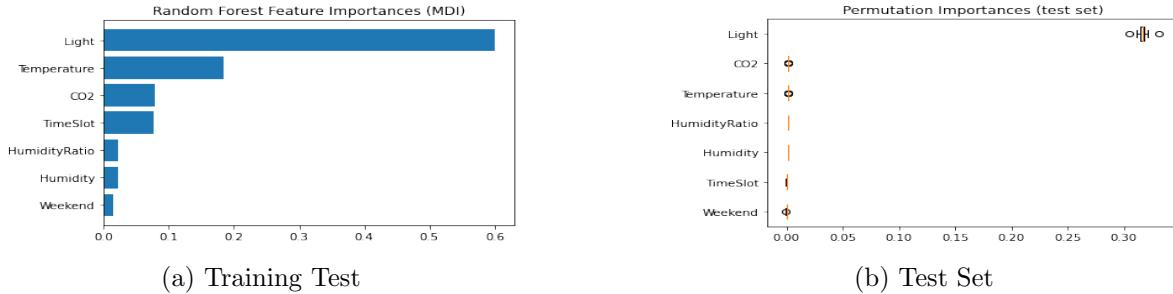


Figura 38: Random Forest Features Importance

### 11.2 Bagging e Boosting

In generale, il Bagging e il Boosting (AdaBoost nello specifico), sono stati eseguiti con un numero di estimatori crescente fino a 20 e con combinazioni diverse di parametri, come il sampling con e senza sostituzione. Non si sono verificati significativi cambiamenti rispetto ai classificatori di base, osservando spesso risultati identici, anche con un **numero esiguo di estimatori**. Si evidenziano, pertanto, soltanto i casi in cui sono emerse delle particolarità.

Applicando bagging o boosting al modello Random Forest non ottimizzato con Random Search, i risultati si allineano a quelli del modello ottimizzato. Applicando boosting al modello già ottimizzato, quest'ultimo migliora ulteriormente, riducendo il numero di record non correttamente classificati da 31 a 17.

Sul Naive Bayes categoriale, ottenuto con discretizzazione dei campi continui, l'applicazione di boosting ha portato a un drastico peggioramento delle performance, con un'accuratezza pari a 0.32. Stessa sorte per il Naive Bayes gaussiano, nel quale l'accuratezza è scesa a 0.74 e l'F1 medio a 0.44. Un netto peggioramento è avvenuto anche applicando il boosting al modello di regressione logistica con feature di partenza WorkingHour: tutto il test set viene classificato come classe di maggioranza.

Lo stesso succede applicando il boosting alla SVM non lineare.

Per applicare i metodi di ensemble su Neural Network, è stata creata una nuova rete basilare che avesse performance migliorabili. È stato utilizzato il Multi-layer Perceptron classifier di scikit-learn impostando un solo layer intermedio composto da un solo nodo. L'accuratezza risultante è stata pari a 0.82 e l'F1 score 0.79. Il miglioramento dei risultati si è ottenuto applicando bagging e boosting con appena 3 estimatori, come da tabella 16, che riporta, a titolo esemplificativo, i risultati ottenuti con AdaBoost.

Tabella 16: Classification Report - AdaBoost su NN

	Precision	Recall	F1-Score
Classe 0	1.00	0.99	1.00
Classe 1	0.97	1.00	0.98
<b>Accuracy</b>	0.99		
<b>AdaBoost n-estimators</b>	2		

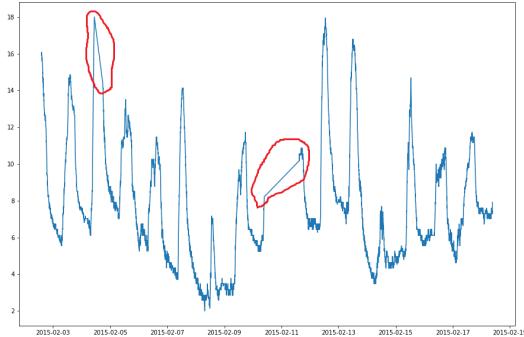
È possibile concludere sottolineando che la maggior parte dei classificatori, avendo già performance ottimali, non beneficiano dei metodi di ensemble. Il modello basato su Neural Network, opportunamente non ottimizzato a priori, è quello che maggiormente si presta a miglioramenti, sfruttando inoltre un numero minimo di estimatori.

## 12 Time Series Analysis

### 12.1 Data Analysis and Preparation

Per i task precedenti è stato utilizzato un Data Set composto dall'unione dei tre proposti. Per l'analisi delle Time Series, tuttavia, ciò non è stato possibile in quanto, tra un Data Set e l'altro, vi è uno sfasamento temporale che non ha permesso di ottenere una Time Series completa. Nel seguente plot viene mostrato questo sfasamento temporale, interpretato dall'ambiente di sviluppo come una linea retta che congiunge i punti tra un gap e l'altro (Figura 39):

Figura 39: Time Series with Gap for "Temperature"

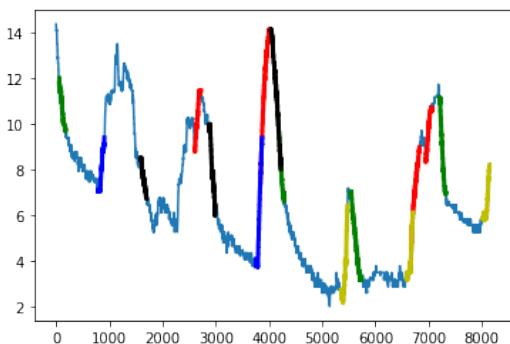


Per risolvere questo problema, semplicemente sono stati riutilizzati i 3 Data Set di partenza, separatamente a seconda del task affrontato.

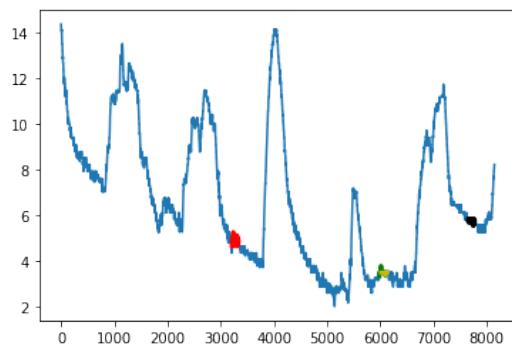
Per l'esecuzione del clustering, della classificazione e per lavorare con le shapelets, è stata effettuata la seguente riorganizzazione dei dati: per ciascuna feature sono stati generati separatamente nuovi Data Set nei quali, in ogni riga, è presente lo stesso numero di misurazioni relative alla feature stessa, ottenendo così una Time Series per riga. Sono state inoltre diversificati i dataset in base alla scelta di lunghezze diverse per le time series (60, 30 e 10 misurazioni). Le misurazioni mantengono un intervallo pari a un minuto, come da dataset originale. Ad ogni Time Series vengono associate diverse label, rappresentanti i valori modali dei campi "TimeSlot", "Weekend", "WorkingHour" ed "Occupancy" relativi ai record da cui sono state formate. Questi campi rappresentano le possibili classi target.

### 12.2 Motif e Anomaly Discovery

Per la scoperta di Motifs e Anomalies è stato considerato il Data Set centrale, contenente le misurazioni tra un gap temporale e l'altro. Per ogni feature è stata calcolato il Matrix Profile, applicando una finestra pari a 120 e considerando, quindi, una finestra temporale pari a 2 ore (essendo le misurazioni effettuate una al minuto). Per riuscire ad individuare più componenti possibili, è stato settato come parametro massimo per i Motifs il valore 1000, mentre per le Anomalies 100. Si riportano di seguito i plot delle varie Time Series con rispettivi Motifs ed Anomalies.



(a) Motifs in Temperature Feature



(b) Anomalies in Temperature Feature

Figura 40: Motifs and Anomalies

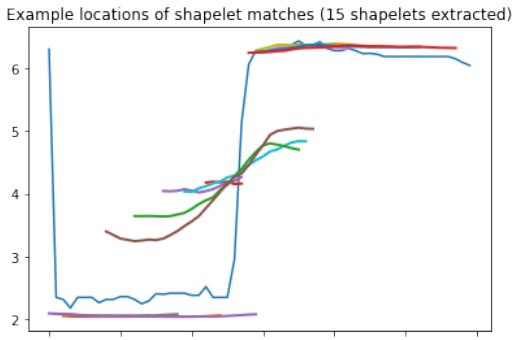
## 12.3 Shapelets

Per l'individuazione delle shapelets sono state utilizzate 4 diverse librerie, focalizzando però l'attenzione su *TensorFlow ShapeletModel* e *PyTS LearningShapelets* in quanto possiedono già i metodi per effettuare predizioni sul test set.

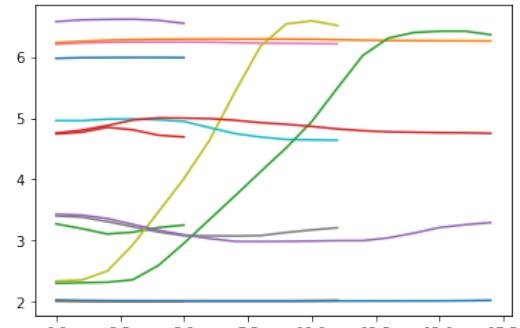
Sono stati effettuati diversi tentativi sulla base di combinazioni diverse di classe target (Occupancy, TimeSlot, Weekend, WorkingHour) e features. I risultati di seguito descritti riguardano la coppia con la performance migliore, ossia Light-Occupancy. Gli algoritmi sono stati applicati su un dataset formato da un insieme di time series aventi 60 misurazioni ciascuna, quindi basate su split temporali di un'ora rispetto all'originale.

Tramite la prima libreria sono state ottenute 15 shapelets, suddivise in 5 gruppi in base alla lunghezza delle shapelets in essi contenuti (multipli di 6 fino a 30). L'accuracy raggiunta con il minimo di epoch (*max\_iter=1*) è stata molto soddisfacente, pari a 0.957, con un F-Score medio di 0.936. Con la seconda libreria si ottiene un'accuracy simile, pari a 0.951, sfruttando lo stesso numero di shapelets. Anche l'F-Score medio si mantiene buono, pari a 0.916.

Nella figura 41 è possibile osservare le shapelets individuate. In particolare, nella 41a è mostrata anche la time series con la migliore corrispondenza.



(a) TensorFlow ShapeletModel



(b) PyTS LearningShapelets

Figura 41: Shapelets

## 12.4 Clustering

Il clustering è stato effettuato sul dataset di timeseries di lunghezza 10 dell'attributo Light. I cluster sono stati ottenuti utilizzando diverse tecniche, quali lo Shape-based, Feature-based, Compression-based e Approximated-based clustering.

Con la prima tecnica di clustering (Shape-based tramite K-Means) sono stati determinati 4 cluster, che presentano un'inerzia pari a 1.582 utilizzando la distanza euclidea come metrica, e di 1.496 con il dynamic time warping.

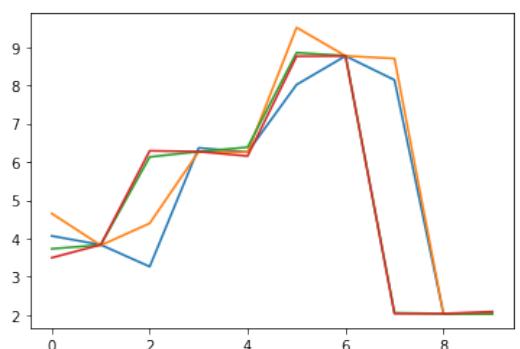
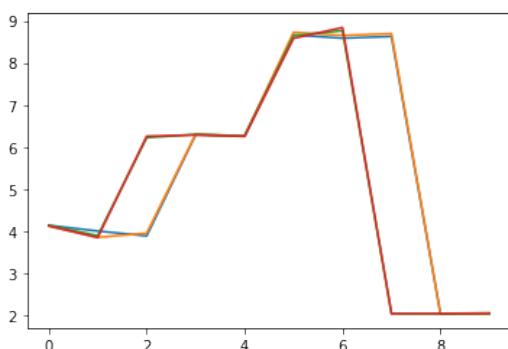


Figura 42: Shape-Based Clusters

Come mostrano i plot, i cluster individuati non appaiono distanti tra loro, fattore determinato anche dalla distribuzione della Feature Light, che si mostra spesso periodica e statica. I bassi valori di inerzia, inoltre, mostrano come i dati interni dei cluster non si discostino dal valor medio relativo a quello stesso gruppo, rendendo il clustering accettabile.

Applicando la stessa configurazione per trovare i cluster approximated-based con un numero di segmenti pari a 5, l'inerzia risultante è pari a 0.743. Il feature-based clustering ha invece prodotto un'inerzia di gran lunga superiore, pari a 794.

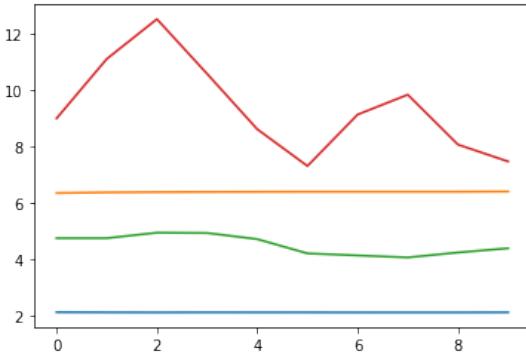


Figura 43: Feature-Based Clustering

In questo caso i cluster individuati sono molto distanti tra loro e l'altissimo valore di inerzia registrato dimostra come i dati raggruppati soffrano di un'elevata dispersione intorno al centroide relativo al cluster stesso. Utilizzare le Features delle varie Time Series al fine di clusterizzare i dati, quindi, non rappresenta una soluzione ottimale.

L'uso del DBScan per un clustering compression-based, con  $\text{eps}=0.5$ ,  $\text{min\_samples}=5$ , ha portato alla creazione di 4 cluster con un coefficiente di silhouette pari a 0.781. Tale valore, quasi prossimo a 1, indica una buona coesione interna e una distanza intercluster accettabile.

## 12.5 Time Series Classification: Univariate

Si riportano i risultati ottenuti considerando Time Series di 30 minuti, relative alle seguenti coppie "Feature-Classe":

- Light - Occupancy (da ora in poi denominata [1])
- Temperature - Working Hour (da ora in poi denominata [2])

Per limitazioni accademiche e chiarezza del report non vengono riportati i risultati di tutte le possibili combinazioni, comunque applicabili data l'avvenuta generazione dei Data Set opportuni. Si procede ora con una rapida panoramica delle tecniche utilizzate e dei risultati ottenuti:

### 12.5.1 Shapelet Classifier

Dopo aver estratto le Shapelets dalle Time Series presenti nei Data Set, è stato applicato un Classificatore su queste, caratterizzato da un ottimizzatore **Adam** con Learning Rate pari a 0.001, per un totale di mille iterate. Si riportano i risultati ottenuti:

	Accuracy	F1-Score
<b>1</b>	0.95609	0.97213 - 0.89655
<b>2</b>	0.81951	0.86346 - 0.73381

Tabella 17: Shapelet Classification

### 12.5.2 Shapelet Distance-Based Classifier

Sulle Shalepets estratte sono stati applicati dei classificatori standard, tra cui il **KNeighborsClassifier** ed il **Decision Tree**. Per il primo è stato settato un valore di k pari a 26 (corrispondente alla radice quadrata del numero di Shapelets), ottenendo i seguenti risultati:

	<b>Accuracy</b>	<b>F1-Score</b>
<b>1</b>	0.96585	0.97832 - 0.91954
<b>2</b>	0.83415	0.87121 - 0.76712

Tabella 18: Shapelet Distance-Based Classification KNeighbors

Per il Decision Tree, invece, è stata settata una profondità massima pari a 5 per evitare Overfitting e per velocizzare il processo di apprendimento:

	<b>Accuracy</b>	<b>F1-Score</b>
<b>1</b>	0.9561	0.97213 - 0.89655
<b>2</b>	0.8	0.84528 - 0.71724

Tabella 19: Shapelet Distance-Based Classification Decision Tree

### 12.5.3 Feature-Based Classifier

Dopo aver estratto un insieme di Features dalle Time Series, a queste è stato applicato nuovamente il Decision Tree:

	<b>Accuracy</b>	<b>F1-Score</b>
<b>1</b>	0.9805	0.98742 - 0.95652
<b>2</b>	0.7707	0.80658 - 0.71856

Tabella 20: Feature-Based Classification

### 12.5.4 Dynamic Time Warping Classifier

Sono stati applicati dei classificatori direttamente sulle Time Series, senza estrazione di valori impliciti o manipolazioni di queste. È stato applicato nuovamente il classificatore **KNeighbors** utilizzando il **Dynamic Time Warping Sakoe Chiba**, ottenendo come risultati:

	<b>Accuracy</b>	<b>F1-Score</b>
<b>1</b>	0.9658	0.97805 - 0.92307
<b>2</b>	0.7317	0.76595 - 0.68571

Tabella 21: Time Series Classification

### 12.5.5 Convolutional Neural Network

Per la classificazione delle Time Series, inoltre, è stata implementata una Rete Neurale Convoluzionale, composta da diversi hidden layers con funzione di attivazione **relu** e da un layer di output con funzione di attivazione **sigmoid**, fondamentale per problemi di classificazione binaria come già accennato nel paragrafo relativo alle Neural Networks. Sono stati ottenuti i seguenti risultati considerando un numero di epoche pari a 5:

	<b>Accuracy</b>	<b>F1-Score</b>
<b>1</b>	0.9707	0.98136 - 0.93181
<b>2</b>	0.76097	0.79497 - 0.71345

Tabella 22: Convolutional Neural Network Classification

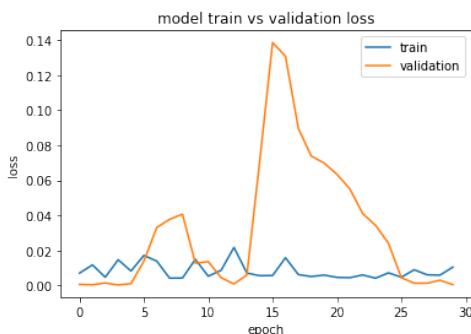
### 12.5.6 Recurrent Neural Networks With LSTM

L'utilizzo delle Recurrent Neural Networks ha riscontrato un Overfitting del modello, con conseguente calo delle prestazioni ottenute sul Test Set rispetto alle altre tecniche. Nonostante i vari tentativi di modificare la Rete Neurale, non sono avvenuti miglioramenti. Tale fenomeno potrebbe essere dovuto al numero di Time Series analizzate.

	Accuracy	F1-Score
<b>1</b>	0.7756	0.87362 - 0
<b>2</b>	0.5951	0.74617 - 0

Tabella 23: Recurrent Neural Networks With LSTM Classification

Figura 44: Overfitting in LSTM Classifier



### 12.5.7 Conclusioni

Com'è possibile notare dai risultati, la coppia Light-Occupancy ha ottenuto risultati nettamente superiori rispetto a Temperature-WorkingHour. Ulteriori tentativi sono stati effettuati con le altre Time Series e labels, ottenendo risultati simili tranne per **Time Slot** la quale, avendo un range di valori più ampio, ha reso i classificatori meno precisi.

Si può quindi concludere sottolineando come, utilizzando la classe di partenza, la Classificazione univariata delle Time Series risulti molto precisa, in particolar modo per le Time Series formate dalla Feature **Light**, probabilmente a causa dell'alta correlazione di quest'ultima con la classe, come mostrato nel capitolo della Data Understanding.

## 12.6 Time Series Classification: Multivariate

Per la classificazione delle Time Series Multivariate è stato nuovamente considerato il Data Set composto dall'unione dei tre di partenza. I record sono stati aggregati in modo da formare una struttura 3D dove, in ogni riga, vi è un Set di Time Series, una per ogni Features del Data Set, della lunghezza di 60 (ovvero una Time Series per ogni ora trascorsa, per ogni Feature).

Successivamente sono stati generati Training, Test e Validation Set rispettando l'ordine temporale delle misurazioni.

Ci si è concentrati sull'utilizzo delle **Convolutional Neural Networks**, nonostante i dati aggregati risultino pochi e con uno sbilanciamento per quanto concerne le classi. I valori di classe di ogni Set, infatti, sono stati calcolati utilizzando la moda dei valori di **Occupancy** nelle 60 misurazioni che compongono ogni Time Series.

Dopo diversi tentativi è stata realizzata una Neural Network composta da molti hidden layer di tipo LSTM, con DropOut e normalizzatori **Batch**, i quali hanno aumentato la stabilità della rete.

Si riportano i risultati ottenuti con un addestramento della rete di cento epoches:

	Precision	Recall
<b>Class 0</b>	0.98	0.94
<b>Class 1</b>	0.85	0.94
<b>Accuracy</b>		0.9420
<b>F1-Score</b>		0.96 - 0.895

Tabella 24: Multivariate Time Series Classification

## 12.7 Time Series Forecasting

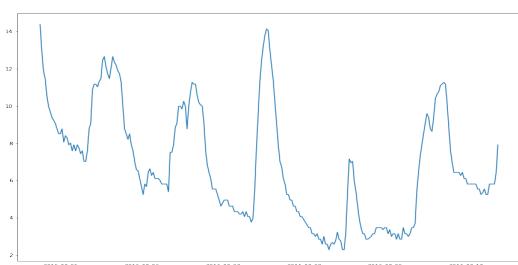
Per applicare il Forecasting sulle Time Series è stato utilizzato uno dei tre split del Data Set principale, così da evitare un cattivo adattamento del modello a causa del gap temporale tra i dati.

Dopo aver tentato di applicare gli algoritmi sulla Time Series completa, ovvero con una misurazione al minuto, ci si è resi conto che la alta frequenza delle misurazioni condizionava negativamente i modelli generati, in quanto rendeva la Series più difficile da gestire e prevedere.

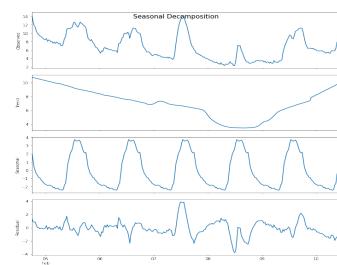
È stata quindi ridotta la frequenza delle misurazioni, considerando una misurazione per ogni mezz'ora trascorsa.

Nel presente report viene mostrato il processo effettuato per la Feature **Temperature**, applicabile ovviamente a tutte le altre.

Sulla Time Series è stata inizialmente effettuata una decomposizione stagionale, per visualizzare al meglio eventuali Trend o Seasonality.



(a) Time Series Iniziale

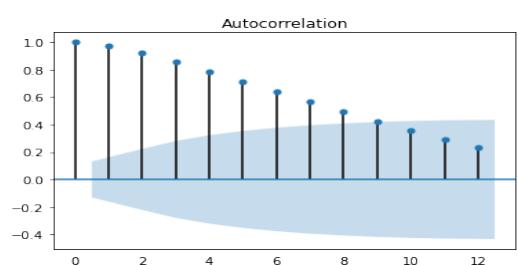


(b) Seasonal Decomposition

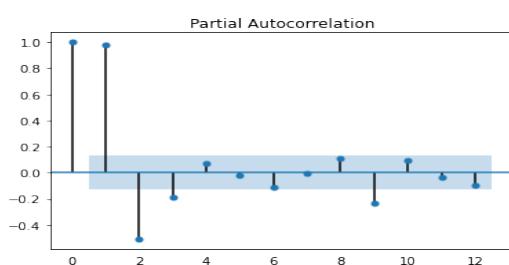
Figura 45: Initial Time Series

Com'è possibile notare, la Time Series è affetta da Trend e Seasonality. Sono state dunque applicate delle trasformazioni per ridurne gli effetti e migliorare il modello finale. E' stata dunque applicata una trasformazione logaritmica e una differenziazione con finestra pari a 48 (24 ore) nel tentativo di renderla quanto più stazionaria e priva di Trend possibile.

Sulla Time Series processata sono stati poi visualizzati i grafici di autocorrelazione, parziale e non, in vista dell'applicazione degli algoritmi.



(a) ACF Plot



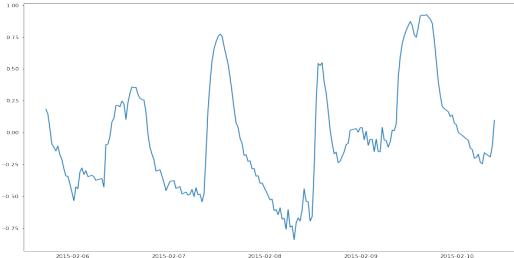
(b) PACF Plot

Figura 46: Autocorrelation plot

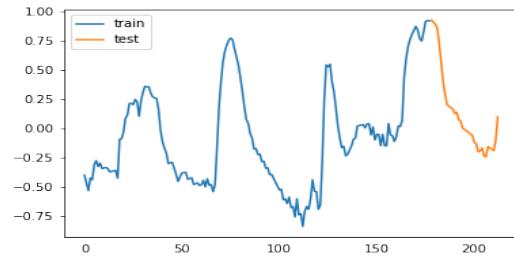
Prima di applicare gli algoritmi, è stato effettuato il **Dickey–Fuller test** per verificare la stazionarietà della Time Series, della quale sono stati poi generati Training Set e Test Set, rispettando l'ordine temporale delle misurazioni.

<b>Test Statistic:</b>	-3.306681		
<b>p-value</b>	0.014582		
	<b>CV 1percentage</b>	<b>CV 5percentage</b>	<b>CV 10percentage</b>
	-3.460154	-2.874649	-2.573757

Tabella 25: Dickey–Fuller Test



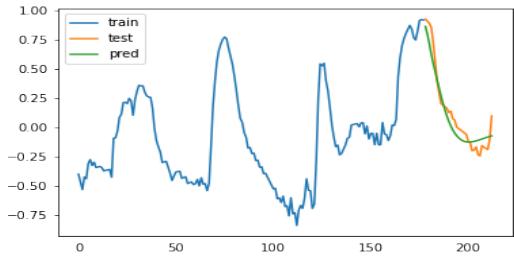
(a) Processed Time Series



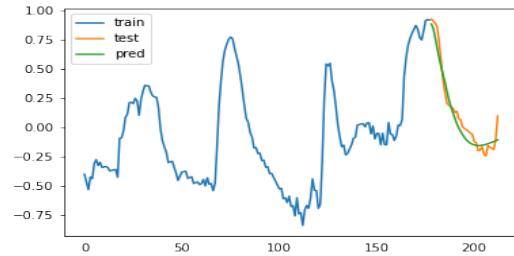
(b) Training/Test Set

Figura 47: Final Time Series

Per la previsione dei dati futuri, sono stati applicati gli algoritmi di Forecasting **ARIMA** e **SARIMAX**, nei quali sono stati impostati i parametri di Auto Regressione e Media Mobile tenendo in considerazione i plot di autocorrelazione (10 per ACF e 4 per PACF), producendo così i seguenti risultati.



(a) ARIMA Forecasting



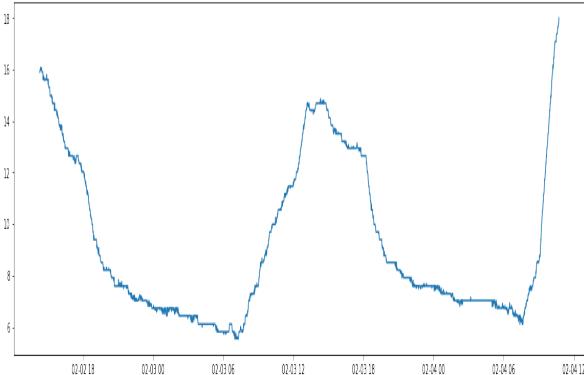
(b) SARIMAX Forecasting

Figura 48: Forecasting

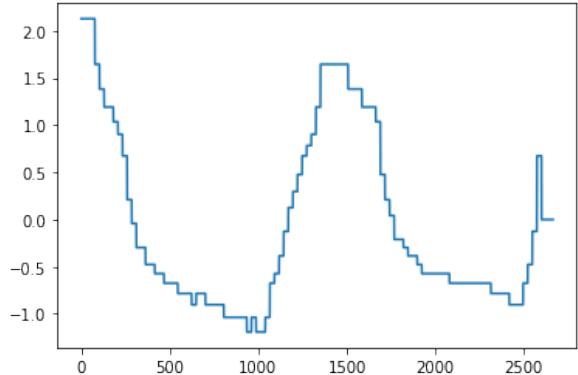
L'algoritmo **ARIMA** ha ottenuto un valore della metrica **R2** pari a 0.938, mentre **SARIMAX** ha ottenuto **R2** pari a 0.955. Si può dunque concludere come, attraverso l'utilizzo di tecniche di trasformazioni adeguate e la validazione della stazionarietà mediante test statistici abbia permesso risultati più che soddisfacenti, avendo ottenuto curve di previsione molto simili ai dati effettivi.

## 13 Frequent Pattern Mining

Prima di procedere con la ricerca dei pattern frequenti, sulle Time Series è stata applicata una trasformazione **SAX**, considerando diverse combinazioni di intervalli e valori possibili. Si riporta un esempio di Time Series trasformata:



(a) Time Series



(b) SAX Transformation

Figura 49: SAX Transformation

Per l'applicazione dell'algoritmo, sono stati considerati i Data Set generati nel precedente capitolo. Sono quindi Data Set di Time Series (dalla lunghezza di 60 misurazioni) relative ad una sola Feature per volta. Dopo aver effettuato la trasformazione SAX sui vari Data Set di Time Series considerando 20 **segmenti** e 10 **valori di mapping**, è stato possibile notare come i pattern frequenti siano monotoni, ovvero formati dallo stesso valore ripetuto più volte.

Tale distribuzione è sicuramente dovuta in primis all'approssimazione effettuata sulle Time Series dai valori di mapping, oltre che alla natura stessa dei dati che, come visto più volte all'interno di questa analisi, tendono ad assumere valori simili nel tempo (**Seasonality nelle Time Series**).

La Feature con sequenze più costanti è **Light**: vi sono infatti molte sequenze con supporti molto alti relative al valore mappato mostrato di seguito. Tale fenomeno può essere associato proprio ai valori assunti da Light, molto costanti nel tempo con qualche variazione sporadica.

- (sup=95, seq=[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])

Vengono riportate di seguito altre sequenze con supporti rilevanti per le altre Features:

- **Temperature:** (sup=52, seq=[8, 8, 8])
- **Humidity:** (sup=51, seq=[8, 9, 9])
- **CO2:** (sup=58, seq=[3, 3, 3])

In generale è possibile notare come, nelle misurazioni effettuate per le varie Features, i valori rimangano spesso simili (tenendo conto dell'approssimazione dovuta alla trasformazione SAX). La Feature Light è quella con stabilità e frequenza più elevata, fattore dovuto forse alla presenza di luce artificiale nella stanza che non determina sbalzi di valori consistenti.

## 14 Outliers Detection

Il dataset originale è composto da 20560 elementi, per cui si è cercato di ottenere un numero di outlier pari a circa 206 (top 1%), laddove i metodi applicati consentissero di determinare a priori il numero di outlier o di assegnare loro uno score, utile al fine estrarre agevolmente il top 1%.

Cinque diversi metodi sono stati applicati: BoxPlot, DBScan, LOF, Isolation Forest e AutoEncoder.

Il BoxPlot ha evidenziato graficamente la presenza di lampanti outlier relativi all'attributo Light, che extrapolati con l'apposita funzione sono risultati pari a 183. I valori di Light dei record così individuati hanno una soglia minima pari a 753.

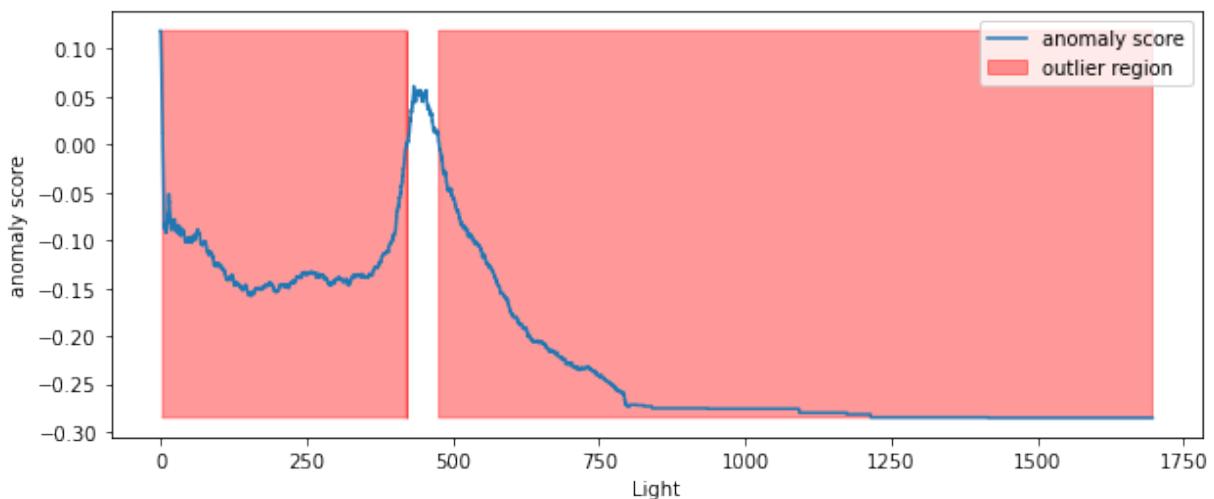
Varie coppie di parametri del DBScan sono state testate al fine di ottenere un congruo numero di outlier, pari a 203 (eps=20, min\_samples=9).

Il parametro del LOF relativo al numero di Nearest Neighbours, è stato ottenuto in base alla regola del logaritmo naturale della dimensione del dataset (10).

L'AutoEncoder è stato eseguito utilizzando un solo nodo intermedio e 10 epochs.

L'Isolation Forest è stato applicato non solo su tutto il dataset, ma anche sui singoli attributi, ottenendo plot specifici con regioni di outliers e relativa curva dell'Anomaly Score, che assume valori tanto bassi quanto più i punti sono ritenuti outliers. A questo proposito, nella Figura 50 è interessante osservare come la curva assuma i valori in assoluto più bassi a partire da un valore di Light pari a circa 750, un risultato perfettamente in linea con gli outlier extrapolati dal boxplot.

Figura 50: Isolation Forest: Anomaly Score in Light



Al fine di confrontare ulteriormente gli outlier rilevati attraverso tutti i metodi, sono stati creati appositi dataframe di outlier e successivamente gli indici di ciascun dataframe sono stati intersecati.

I metodi con i risultati più simili, quasi sovrapponibili, sono stati l'AutoEncoder e l'IsolationForest, con ben il 95% di outlier in comune (196 elementi su 206).

Le altre interserzioni hanno dato un numero esiguo di outlier, con un picco massimo di 34 elementi in comune tra LOF e DBScan.

Intersecando tutti i dataset di outlier si ottengono soltanto i 2 elementi in Figura 51.

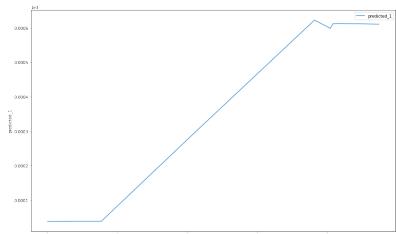
Figura 51: Final outliers (5 methods ensemble)

	Temperature	Humidity	Light	CO2	HumidityRatio	Occupancy
6496	20.700	18.89	1546.333333	455.333333	0.002845	0
6497	20.745	18.89	1451.750000	453.000000	0.002853	0

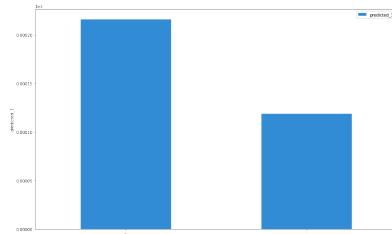
## 15 Explainability

Per testare alcune tecniche di Explainability, sul Data Set completo è stato applicato un classificatore **Random Forest**, utilizzando 20 stimatori. Il classificatore ha ottenuto un livello di **Accuracy** sul Test Set pari a 0.992.

Grazie alla libreria **Skater** è stato possibile plottare i grafici di dipendenza parziale delle varie Features, i quali essenzialmente approssimano la differenza parziale della funzione di previsione rispetto alle variabili passate.



(a) Light Partial Dependence Plot

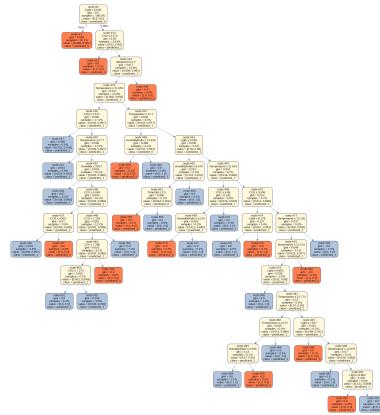


(b) Weekend Partial Dependence Plot

Figura 52: Partial Dependence Plots

È stato poi stampato l'albero di decisione alla base del modello creato.  
(Figura 53):

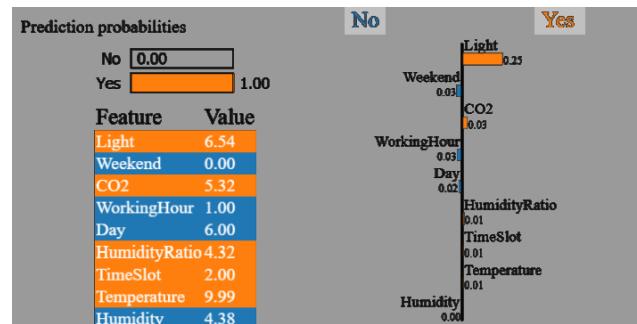
Figura 53: Generated Decision Tree



Tramite la libreria **lime**, sono stati poi generati due resoconti sulla classificazione effettuata su due record, uno con classe "Yes" e l'altro con classe "No". Da notare come la Feature Light abbia sempre più rilevanza rispetto alle altre nella scelta.



(a) "No" Classification



(b) "Yes" Classification

Figura 54: Records Classification Report

Tramite la libreria **SHAP** è stato possibile ottenere ulteriori spiegazioni sui due record classificati, grazie a dei **Waterfall Plots** che mostrano come i valori delle Features abbiano inciso sulla decisione della classe da assegnare ai record. È curioso notare come, per il record classificato con classe "No", la scelta sia stata condizionata prevalentemente dalla Feature **Weekend**, mentre per il record classificato con "Yes", come al solito, Light abbia gioacato un ruolo dominante.

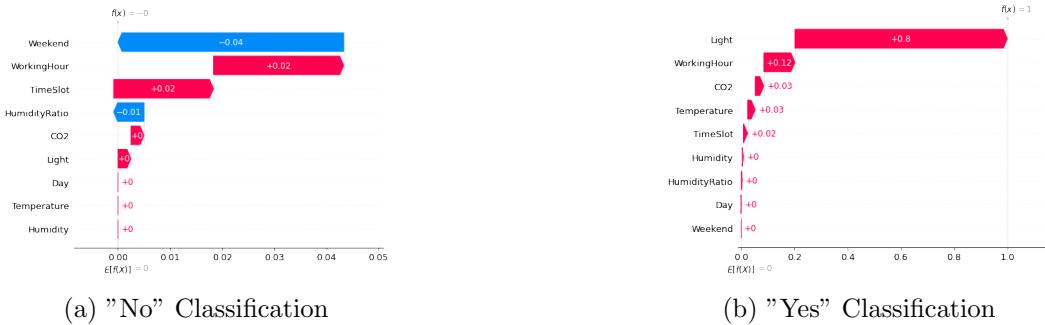


Figura 55: Records Classification Waterfall Plots

Infine, è stato generato un grafico interattivo col quale è stato possibile osservare, per i vari record del Test Set, come le varie Features abbiano inciso sulla scelta delle classi.

Figura 56: Light's Effect on Classification

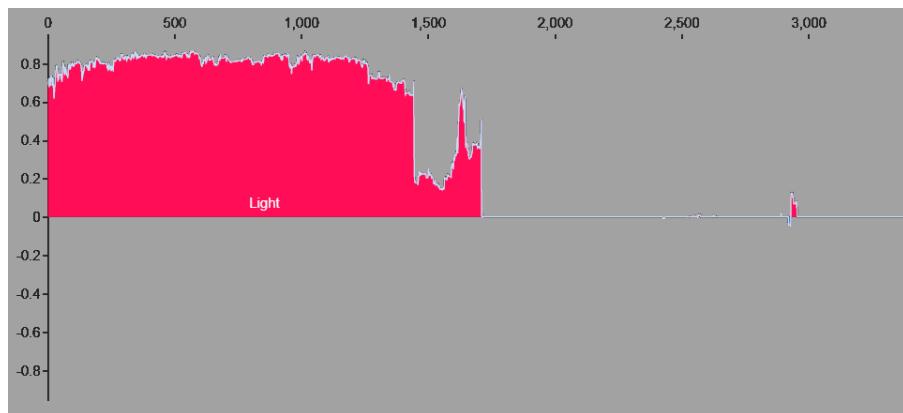


Figura 57: Working Hour's Effect on Classification

