Chiara Baraglia,
Giacomo Pantano,
Gabriele Tenucci.

Data Mining - progetto sul dataset

Occupancy Detection

Anno accademico 2019/2020

Indice

1 Data Understanding **1**
- 1.1 Semantica dei dati . . . 1
- 1.2 Distribuzione delle variabili e statische . . . 1
- 1.3 Qualità dei dati . . . 3
- 1.4 Trasformazione delle variabili . . . 4
- 1.5 Correlazione . . . 4

2 Clustering **5**
- 2.1 K-Means . . . 5
  - 2.1.1 Scelta del numero di centroidi . . . 5
  - 2.1.2 Caratterizzazione dei cluster ottenuti . . . 6
- 2.2 DBSCAN . . . 7
  - 2.2.1 Scelta dei parametri . . . 7
  - 2.2.2 Caratterizzazione dei cluster ottenuti . . . 7
- 2.3 Agglomerative Hierarchical . . . 8
  - 2.3.1 Scelta della funzione di distanza e del criterio di collegamento . . . 8
- 2.4 Valutazioni finali . . . 10

3 Regole di Associazione **11**
- 3.1 Preparazione dei dati . . . 11
- 3.2 Estrazione degli itemset più frequenti . . . 12
- 3.3 Estrazione delle regole di associazione . . . 12
- 3.4 Predire se la stanza sarà occupata usando le regole di associazione . . . 13
- 3.5 Considerazioni finali . . . 15

4 Classificazione **16**
- 4.1 Decision Trees . . . 16
  - 4.1.1 Ricerca dei parametri . . . 16
  - 4.1.2 Analisi del modello . . . 16
  - 4.1.3 Albero di decisione . . . 17
- 4.2 Random Forest . . . 17
  - 4.2.1 Ricerca dei parametri . . . 18
  - 4.2.2 Analisi del modello . . . 18
- 4.3 Bonus . . . 18
  - 4.3.1 Naive Bayesian . . . 18
  - 4.3.2 K-NN . . . 19
- 4.4 Confronto . . . 19

1. Data Understanding

Il dataset analizzato in questo progetto è composto dalle misurazioni di alcuni sensori, con l'obiettivo di creare un modello in grado di prevedere se l'ufficio in cui queste vengono effettuate sia occupato o meno.
Per poter essere efficacemente processati nelle analisi successive, i dati sono stati in primo luogo sottoposti ad un processo di comprensione e preparazione. In particolare abbiamo esaminato il significato di ogni attributo e abbiamo effettuato delle analisi statistiche, in modo da poterne evidenziare la distribuzione e rilevare la presenza di eventuali outliers e valori mancanti. Inoltre è stata misurata la correlazione tra gli attributi numerici e, infine, i dati sono stati preparati per le analisi successive attraverso un algoritmo di scaling.

1.1 Semantica dei dati

Il dataset contiene 8143 righe (misurazioni) e 7 colonne (attributi). La tabella 1.1 riporta, per ogni colonna del dataset, una breve descrizione per spiegarne il significato e l'indicazione del tipo di variabile in questione. L'attributo *Occupancy* è di tipo numerico, ma, poiché indica se la stanza sia o meno occupata, è stato considerato

Tabella 1.1: Descrizione degli attributi del dataset.

Nome	**Tipo**	**Descrizione**
Date	Data	Timestamp che include data e orario
Temperature	Numerica continua	Temperatura in °C
Relative Humidity	Numerica continua	Percentuale di umidità
Light	Numerica continua	Illuminamento espresso in Lux
CO2	Numerica continua	Anidride carbonica in ppm
Humidity Ratio	Numerica continua	Rapporto tra temperatura e umidità relativa
Occupancy	Numerica	Indica se la stanza è occupata o no

come categorico, facendo corrispondere il valore 1 a "Occupato" e il valore 0 a "Non occupato".

1.2 Distribuzione delle variabili e statische

La Tabella 1.2 riporta alcuni valori statistici delle variabili, che sono stati approssimati a tre cifre decimali per facilitarne la lettura. La Figura 1.1 mostra la distribuzione dei valori nelle variabili di input.

Tabella 1.2: Statistiche sulle variabili

	Temperature	**Humidity**	**Light**	**CO2**	**HumidityRatio**
mean	20.619	25.731	119.519	606.546	$3.863 * 10^{-3}$
std	1.017	5.531	194.756	314.321	$0.852 * 10^{-3}$
min	19.000	16.745	0.000	412.750	$2.674 * 10^{-3}$
25%	19.700	20.200	0.000	439.000	$3.078 * 10^{-3}$
50%	20.390	26.222	0.000	453.500	$3.801 * 10^{-3}$
75%	21.390	30.533	256.375	638.833	$4.352 * 10^{-3}$
max	23.180	39.117	1546.333	2028.500	$6.476 * 10^{-3}$

Come si può notare dalla Tabella 1.2 e dalla Figura 1.1, le variabili *Temperature*, *Humidity* e *HumidityRatio* seguono delle distribuzioni simil-gaussiane.
I valori delle variabili *Light* e *CO2*, invece, sono concentrati principalmente in un range ristretto intorno al valore medio. Questo porta ad un numero elevato di outliers, come specificato nella Sezione 1.3.

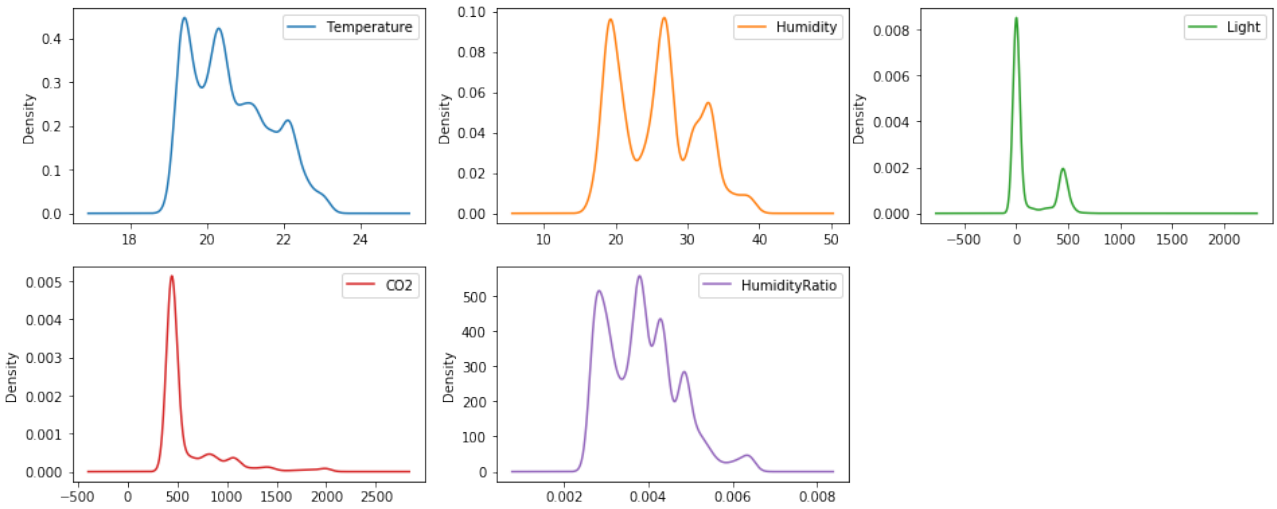


Figura 1.1: Density plot.

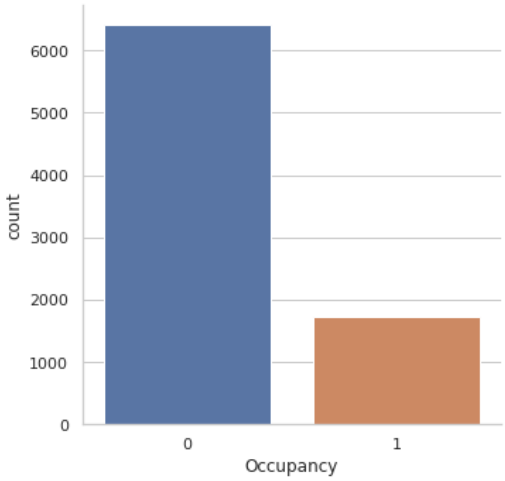


Figura 1.2: Distribuzione della variabile di output Occupancy.

Analizzando la distribuzione della variabile di output *Occupancy*, mostrata in Figura 1.2, è emerso come questa non sia ben distribuita, ma presenti invece uno sbilanciamento verso valori con Occupancy = 0.
Infatti, il numero di samples in cui l'ufficio risulta essere occupato è 1729 (21,23%), contro 6414 istanze per cui l'ufficio non è occupato (78,76%).
La Figura 1.3 mette in relazione ogni coppia di variabili, evidenziando la distribuzione dei valori di Occupancy. Sulla diagonale, invece, è rappresentata la distribuzione dei valori di Occupancy per ogni singola variabile.
Di seguito analizziamo in dettaglio le relazioni più significative.

- **Temperature**: La distribuzione di Occupancy sul parametro Temperature evidenza come, a temperature più basse, l'ufficio tenda ad essere non occupato, mentre a temperature più alte tenda ad essere occupato.
- **Humidity**: I valori di Occupancy risultano ben distribuiti rispetto a quelli del parametro Humidity. Questo non fornisce informazioni significative su eventuali cluster di dati.
- **Light**: Il parametro Light relazionato a tutti gli altri parametri mostra dei cluster ben definiti. In particolare, quando il valore di Light è alto, Occupancy tende ad essere uguale ad 1, mentre per valori bassi tende ad essere 0.
- **CO2**: La distribuzione di Occupancy in base al parametro CO2 mostra una tendenza di Occupancy verso lo 0 quando la CO2 è bassa, e verso l'1 quando la CO2 aumenta.

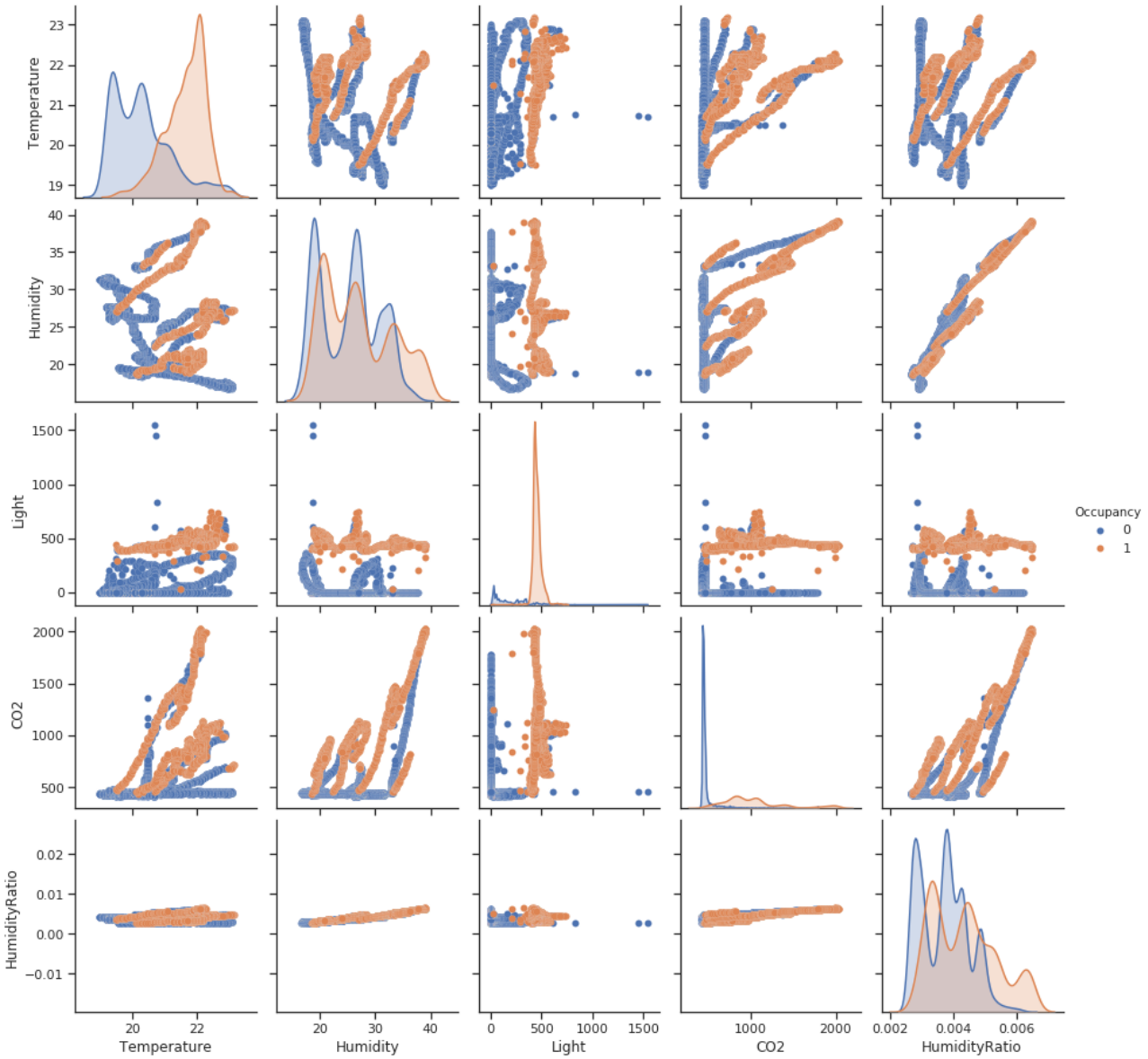


Figura 1.3: Scatter Matrix.

- **HumidityRatio**: I valori di Occupancy in relazione al parametro HumidityRatio risultano quasi completamente sovrapposti; pertanto non risultano informazioni particolarmente interessanti dall'analisi di questo parametro.

1.3 Qualità dei dati

Nella tabella di input non sono stati rilevati valori mancanti.
La figura 1.4 mostra la distribuzione dei valori per ogni parametro, evidenziando in nero gli eventuali outliers.
Secondo le distribuzioni quantili (25%, 75%), non ci sono outliers per Temperature e Humidity, ci sono 15 outliers per Light, 118 per HumidityRatio, e 1063 per CO2.
Gli outliers trovati per il parametro CO2 rappresentano il 13% dei samples totali del dataset, quindi una percentuale non trascurabile. Pertanto abbiamo ritenuto di non poterli eliminare, ma che fosse necessario indagare ulteriormente tramite i risultati degli algoritmi di clustering: questi outliers potrebbero infatti costituire un cluster a sé stante.

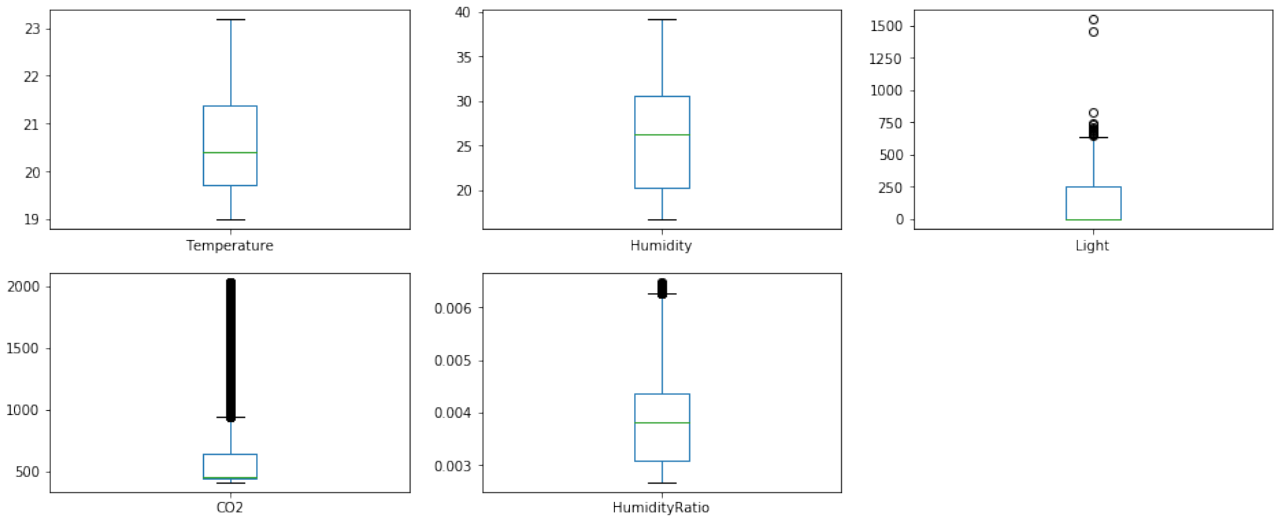


Figura 1.4: Box plot delle variabili di input.

1.4 Trasformazione delle variabili

Al fine di migliorare la leggibilità di questo parametro, abbiamo diviso l'attributo *Date* in due attributi distinti *Day* e *Time*, contenenti rispettivamente il giorno della settimana e l'orario della misurazione. Ritenendo, però, questo parametro poco significativo, abbiamo deciso di non utilizzarlo. Dal momento che abbiamo rilevato molti outliers, abbiamo deciso di scalare le variabili utilizzando Robust Scaler.

1.5 Correlazione

La Figura 1.5 mostra la correlazione tra le variabili calcolata tramite la formula di Pearson.
Come si può notare, ci sono variabili fortemente correlate tra loro. Ad esempio, *Humidity* e *HumidityRatio* mostrano una correlazione lineare positiva superiore al 95%: questo potrebbe far pensare che almeno una delle due variabili possa essere considerata come ridondante.
Valori alti di correlazione positiva si riscontrano anche tra *CO2* e *Light* (0.664) e tra *Light* e *Temperature* (0.65).
Per quanto riguarda la correlazione con il parametro *Occupancy* possiamo notare che i parametri con valori più alti sono *Light* ($\sim 90,7\%$), *CO2* ($\sim 71,2\%$) e *Temperature* ($\sim 53,8\%$). Questo ci porta a considerare di utilizzare soprattutto questi 3 attributi per le analisi successive.

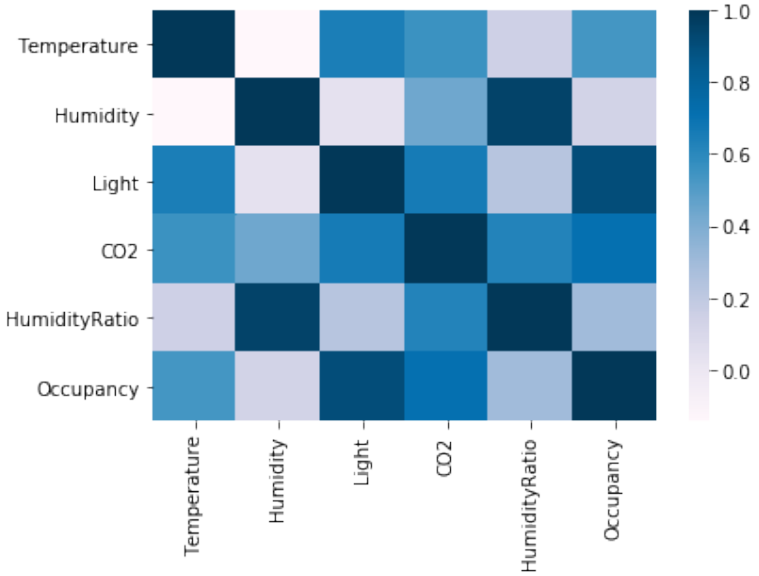


Figura 1.5: Heatmap per la correlazione di Pearson.

2. Clustering

Il dataset è stato analizzato usando 3 diversi algoritmi di clustering: K-Means, DBSCAN e Agglomerative Hierarchical. Nel resto del capitolo sono descritti i metodi usati, le scelte effettuate e i risultati ottenuti.

2.1 K-Means

Il K-Means è un algortimo di clustering center-based, ovvero ad ogni cluster corrisponde un punto rappresentativo, detto centroide, e ogni altro punto è assegnato al cluster con il centroide più vicino.

2.1.1 Scelta del numero di centroidi

Il numero k di clusters che si vogliono ottenere eseguendo l'algoritmo K-Means può essere stimato utilizzando la misura SSE (Sum of Squared Errors): $SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(m_i, x)$.

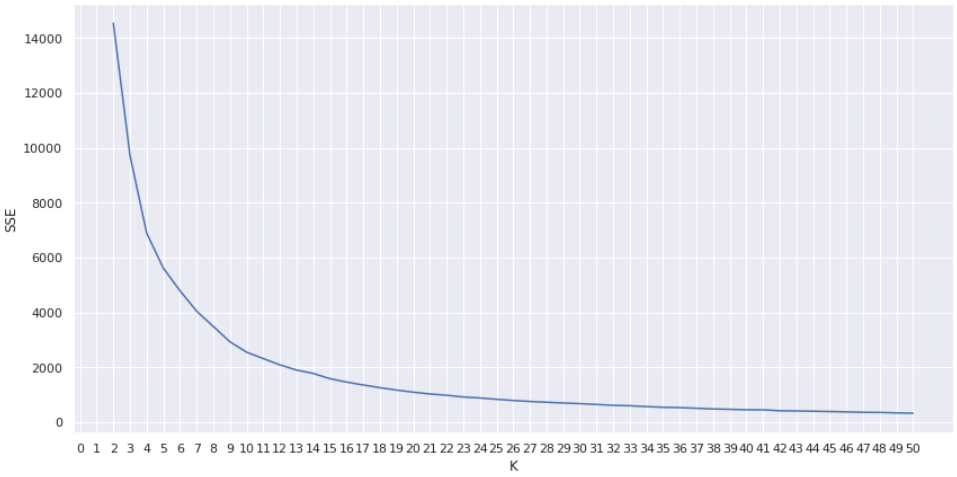


Figura 2.1: SSE.

Per ogni punto, l'errore è visto come la distanza dal cluster più vicino: per ottenere la misura del SSE questi errori vengono elevati al quadrato e sommati tra loro. Nella formula x è elemento del cluster C_i e m_i è il centroide per il cluster C_i. L'algoritmo è stato iterato al variare di k in un intervallo tra 2 e 50 e per ogni iterazione abbiamo salvato il corrispondente SSE: la figura 2.1 mostra i risultati ottenuti. Guardando questi risultati il numero ottimale di cluster risulta essere circa 13, poiché dopo questo valore la curva inizia a decrescere in modo costante.

A questo punto abbiamo eseguito un'analisi della silhouette, in modo da avere un ulteriore riscontro sul numero di cluster da scegliere. La misura della silhouette permette di valutare il clustering sulla base non solo della coesione interna ad ogni cluster ma anche della separazione tra di essi: ha un range di valori compreso tra -1 e 1 e più si avvicina ad 1 e migliore è

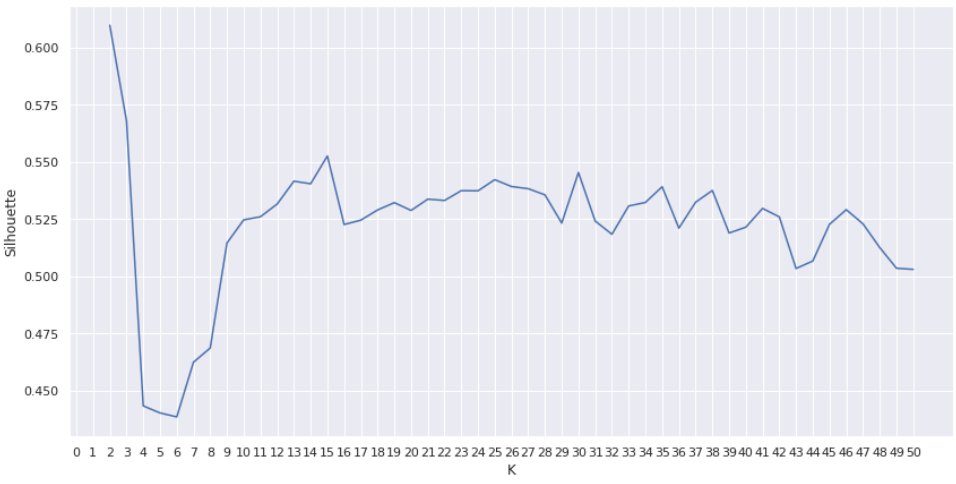


Figura 2.2: Analisi della silhouette.

il risultato del clustering.
Anche in questo caso ab-
biamo eseguito K-Means iterando con valori di k tra 2 e 50 e salvando ogni volta il valore della silhouette. Analizzando i risultati rappresentati dalla curva nella Figura 2.2, abbiamo stabilito che il numero ottimale di cluster per K-Means fosse 15.

2.1.2 Caratterizzazione dei cluster ottenuti

I cluster ottenuti, con i relativi centroidi, possono essere visualizzati nella Figura 2.3, contenente gli scatter plot degli attributi Temperature, Light e CO2 messi in relazione tra loro. La Tabella 2.1, invece, mostra la

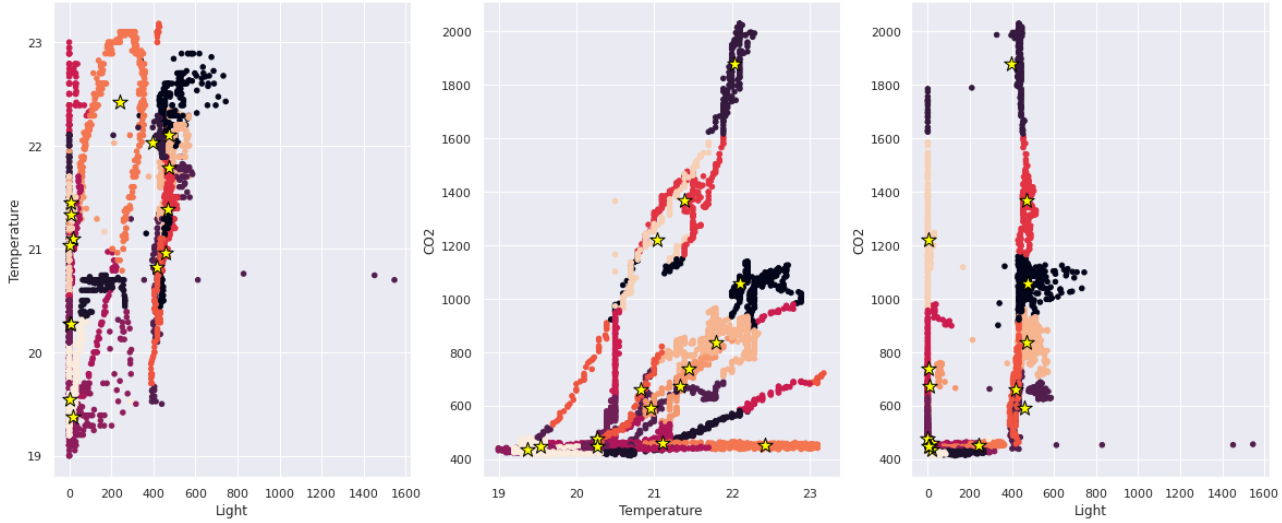


Figura 2.3: Cluster di Temperature in funzione di Light (sinistra), CO2 in funzione di Temperature (centro), e CO2 in funzione di Light (destra). Le stelle rappresentano i centroidi ottenuti.

composizione dei cluster e la distribuzione degli elementi all'interno dei vari cluster, in relazione al valore di Occupancy. Si può notare come, analizzando le percentuali ottenute, siano emersi dei cluster in cui la percentuale di elementi con Occupancy uguale ad 1 risulti molto alta (>=89%), oppure molto bassa (0%). Questo parrebbe indicare che i cluster individuati separino bene il dataset.

Tabella 2.1: Contenuto dei clusters.

Cluster	N elementi	% dataset	N Occupancy = 1	% Occupancy = 1
0	491	6%	462	94%
1	840	10%	0	0%
2	202	2%	183	90%
3	228	3%	204	89%
4	635	8%	0	0%
5	837	10%	0	0%
6	1833	23%	0	0%
7	209	3%	0	0%
8	224	3%	224	100%
9	114	1%	112	98%
10	336	4%	0	0%
11	166	2%	0	0%
12	577	7%	542	93%
13	134	2%	1	0%
14	1317	16%	1	0%

2.2 DBSCAN

DBSCAN è un algoritmo di clustering basato sulla densità: dato un insieme di punti in un certo spazio, raggruppa quelli più vicini la cui distanza è inferiore a un dato ε e permette di separarli da aree a bassa densità che costituiscono il rumore.

2.2.1 Scelta dei parametri

DBSCAN necessita di due parametri: ε e min_samples. Come prima cosa abbiamo fissato il valore di min_samples uguale al logaritmo naturale del numero di elementi nel dataset, che nel nostro caso equivale a 9. Successivamente, per trovare il valore di ε, abbiamo realizzato il grafico delle distanze in Figura 2.4.

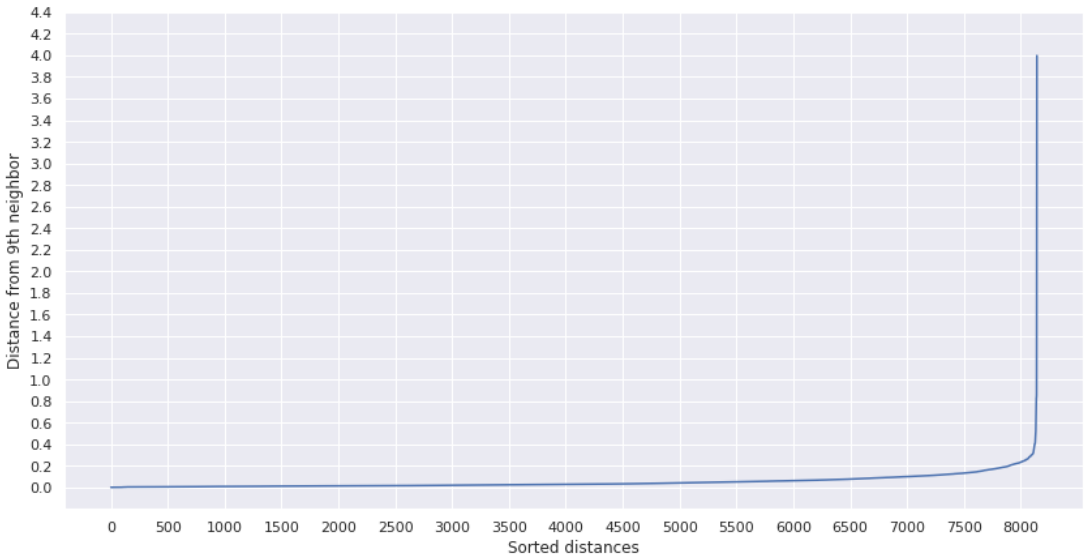


Figura 2.4: Grafico della distanza ε dal decimo elemento più vicino.

Analizzando il grafico, si può dedurre che il valore ottimale di ε sia 0.3, poiché in prossimità di questo valore la curva presenta una variazione di crescita netta.

2.2.2 Caratterizzazione dei cluster ottenuti

L'esecuzione dell'algoritmo DBSCAN con i paramentri precedentemente definiti ha portato alla clusterizzazione osservabile nella Figura 2.5.
La tabella ottenuta è la seguente:

Tabella 2.2: Contenuto dei cluster.

Cluster	N elementi	% dataset	N Occupancy = 1	% Occupancy = 1
0	16	0.2%	16	100%
1	5403	66%	0	0%
2	600	7%	536	89%
3	24	0.3%	0	0%
4	602	7%	585	97%
5	533	6.5%	530	99%
6	23	0.2%	1	4%
7	869	10%	0	0%
8	56	0.7%	54	96%

Inoltre, l'algoritmo ha rilevato 17 elementi che non risultano appartenere a nessun cluster.
Si può notare come, analizzando le percentuali ottenute, siano emersi dei cluster in cui la percentuale di elementi

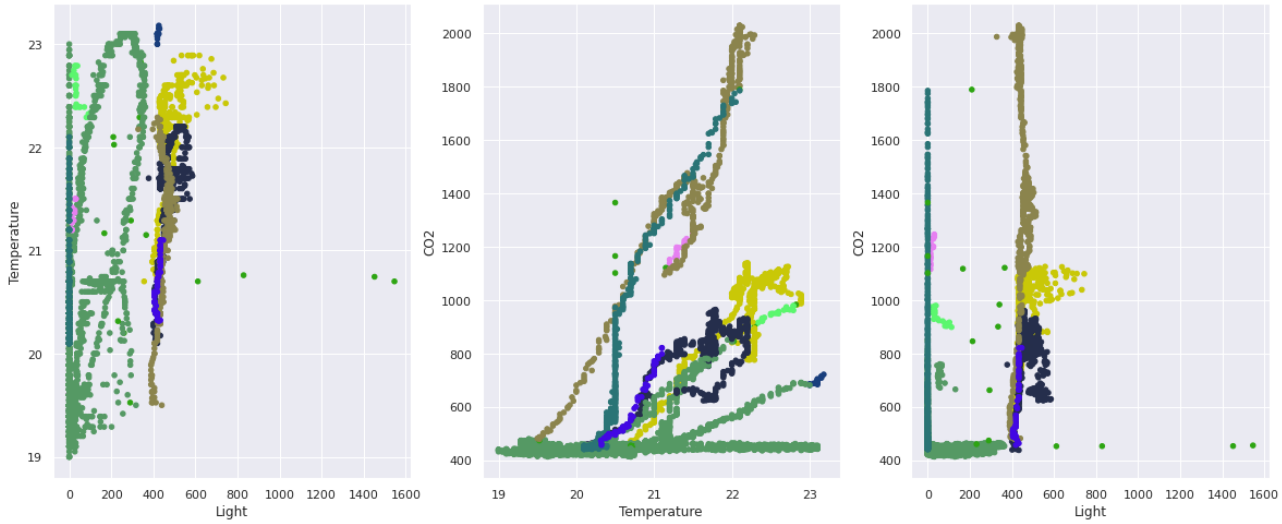


Figura 2.5: Cluster di Temperature in funzione di Light (sinistra), CO2 in funzione di Temperature (centro), e CO2 in funzione di Light (destra).

con Occupancy uguale ad 1 risulti molto alta (>=89%), oppure molto bassa (<=4%). Questo parrebbe indicare che i cluster individuati separino bene il dataset. Inoltre, si può notare che il cluster con etichetta 1 contenga la maggior parte degli elementi del dataset (5403 su 8143), che hanno tutti Occupancy uguale a 0.

2.3 Agglomerative Hierarchical

Il clustering gerarchico è un approccio di clustering che mira a costruire una gerarchia di cluster. In particolare abbiamo esaminato un algoritmo di tipo gerarchico agglomerativo, in cui si parte dall'inserimento di ciascun elemento in un cluster distinto e successivamente si procede all'aggregazione graduale di cluster a due a due.

2.3.1 Scelta della funzione di distanza e del criterio di collegamento

Per stabilire quale fosse la funzione di distanza più adatta e quale criterio di collegamento utilizzare, abbiamo eseguito un'analisi della silhouette per ognuna delle 7 possibili combinazioni di questi due parametri, tenendo presente che il criterio di Ward si può applicare solamente utilizzando la distanza Euclidea.
La Tabella 2.3 mostra i risultati di questa analisi.

Tabella 2.3: Valore della silhouette in relazione alla funzione di distanza e al criterio di collegamento.

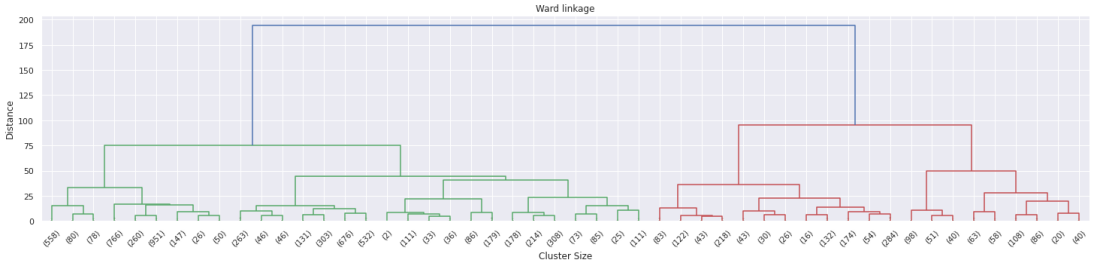
	Single	**Complete**	**Average**	**Ward**
Euclidea	0.6063	0.6291	0.6784	0.5999
Manhattan	0.6063	0.4131	0.6289	-
Cosine	0.1737	0.1979	0.187	-

Risulta evidente che la silhouette migliore, e quindi più vicina al valore 1, si ottenga utilizzando la distanza euclidea e il criterio di collegamento *average*: questo significa che la distanza inter-cluster viene calcolata come la distanza media tra ogni membro di un cluster e ogni membro dell'altro cluster.
Per avere un ulteriore riscontro sul criterio di collegamento da utilizzare, abbiamo creato i relativi dendrogrammi, mostrati in Figura 2.6.
Come si può notare in Figura 2.6a, il criterio *single linkage* non evidenzia divisioni nette nel dataset, in quanto le distanze tra i cluster risultano molto piccole, e crea cluster di dimensione irrisoria. Il criterio *average*, invece, crea cluster più aderenti al nostro dataset. Una volta fissati la funzione di distanza e il criterio di collegamento, abbiamo eseguito un'analisi della silhouette al variare del numero di cluster prodotti dall'algoritmo, come osservabile in Figura 2.7.

(a) Single.

(b) Complete.

(c) Average.



(d) Ward.

Figura 2.6: Dendrogrammi ottenuti con i diversi criteri di collegamento e distanza euclidea.

Tabella 2.4: Contenuto dei cluster.

Cluster	N elementi	% dataset	N Occupancy = 1	% Occupancy = 1
0	7954	97,7%	1540	19%
1	189	2,3%	189	100%

La Tabella 2.4 mostra i cluster ottenuti con il raggruppamento rilevato dall'algoritmo. Si può vedere come il cluster con etichetta 1 sia buono, in quanto comprende solo elementi con Occupancy uguale ad 1. Il cluster 0, invece, ha una percentuale di Occupancy = 1 pari al 19%. Non consideriamo questo un buon cluster, in quanto simile alla percentuale dell'intero training set (21,23%, Figura 1.2).
La Figura 2.8 mostra una rappresentazione dei cluster ottenuti in funzione di Temperature, Light e CO2. Si può vedere come il cluster 1 (blu) risulti significativo, mentre il cluster 0 (arancione) raggruppi zone in cui Occupancy è 0, e zone in cui è 1.

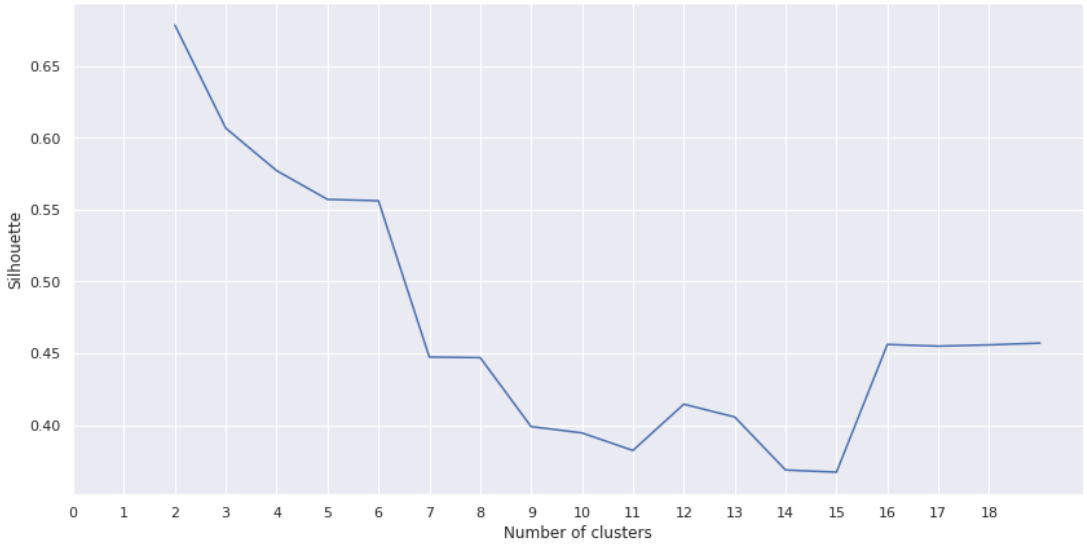


Figura 2.7: Sihouette per il clustering Gerarchico con distanza Euclidea e criterio *average*.

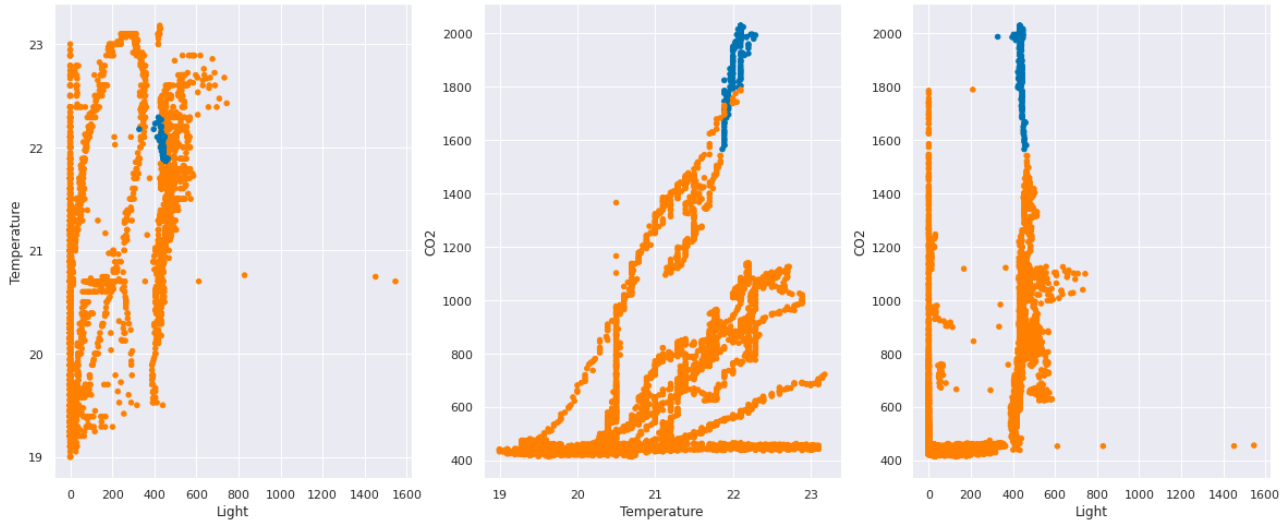


Figura 2.8: Cluster di Temperature in funzione di Light (sinistra), CO2 in funzione di Temperature (centro), e CO2 in funzione di Light (destra).

2.4 Valutazioni finali

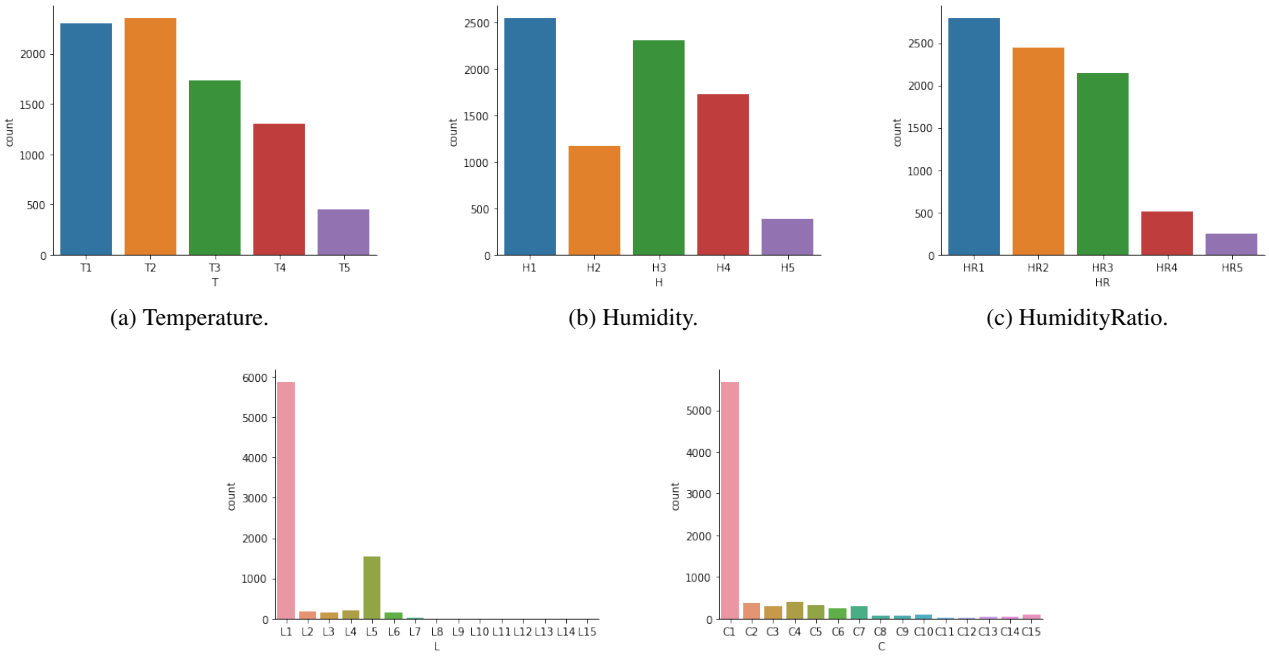
Per i motivi sopra esposti, non abbiamo ritenuto il clustering Gerarchico particolarmente adatto al nostro dataset, preferendo quindi i risultati di K-Means e DBScan. In particolare, abbiamo preferito i risultati ottenuti con DBScan, in quanto ci sono sembrati buoni anche nelle zone in cui sono presenti molti outliers, dove invece K-Means non restituisce risultati soddisfacenti (Figure 2.3, 2.5).

3. Regole di Associazione

Utiliziamo il mining degli itemset frequenti, tramite l'algoritmo *Apriori*, per determinare se esistano valori degli attributi che si riscontrino frequentemente insieme nelle misurazioni presenti nel dataset. Una volta estratti gli itemset frequenti, ne ricaveremo le relative regole di associazione.
In questo tipo di analisi è critica la scelta delle soglie di *minimum support* e *minimum confidence*: è necessario un compromesso tra avere molti itemsets e molte regole poco frequenti ed avere pochi itemsets, e di conseguenza poche regole, ma molto frequenti. Abbiamo quindi deciso di far variare *min_sup* tra 2% e 10% e *min_conf* tra 70% e 95%, in modo da poter scegliere le tresholds più adatte per questo dataset.
Come espresso nella Sezione 1.3 non abbiamo riscontrato valori mancanti, quindi le regole di associazione estratte verrano utilizzate principalmente allo scopo di comprendere quali siano le principali caratteristiche delle misurazioni relative all'avere l'ufficio occupato.

3.1 Preparazione dei dati

L'algoritmo *Apriori* richiede l'utilizzo di un database transazionale e attributi categorici, quindi è stato necessario far sì che i record corrispondessero a transazioni, in cui il valore dell'attributo sia equivalente ad un itemset della transazione. Inoltre abbiamo dovuto discretizzare tutti gli attribuiti numerici e trasformarli in stringhe, in modo da avere itemset categorici. Inizialmente abbiamo utilizzato la regola di Sturges ($k = 1 + \frac{1}{3} * \log_{10} N$, dove N è il



(a) Temperature. (b) Humidity. (c) HumidityRatio.

(d) Light. (e) CO2.

Figura 3.1: Distribuzione delle variabili discretizzate in 5 bins.

numero totale di osservazioni), ottenendo così una discretizzazione in 15 bins. In questo modo, però, abbiamo ottenuto pochi itemset frequenti e un numero insufficiente di regole di associazione. Dopo aver effettuato alcune prove, abbiamo deciso di dividere gli attributi in 5 bins, mantenendo però la divisione in 15 bins per gli attributi *Light* e *CO2*. Queste variabili, infatti, hanno una distribuzione molto poco omogenea e una suddivisione in pochi bins poteva non catturare questa caratteristica, falsando quindi i risultati.
La Figura 3.1 illustra le distribuzioni delle varibili ottenute dopo la suddetta discretizzazione.

3.2 Estrazione degli itemset più frequenti

Per poter stabilire il valore di *min_sup* più adatto per questo dataset, abbiamo effettuato diversi esperimenti con valori crescenti di *minimum support*. Inoltre abbiamo estratto anche gli itemsets massimali, in modo da poter determinare se ci fossero itemsets risultati frequenti solamente perché i valori della variabile relativa fossero troppo sbilanciati. Una volta effettuate queste analisi, abbiamo stabilito che non fosse necessario eliminare nessun item, in quanto, pur essendoci degli sbilanciamenti non trascurabili nelle variabili, questi non risultavano falsare l'estrazione degli itemset frequenti, esaminati anche nella forma compatta di *closed itemsets*.

Tabella 3.1: Numero di itemsets al variare di *min_sup* e del tipo di itemsets prodotti.

min_sup	**frequent**	**maximal**	**closed**
0.02	553	41	239
0.03	446	31	171
0.04	329	17	127
0.05	217	16	104
0.06	158	14	83
0.07	125	12	67
0.08	75	8	48
0.09	73	7	47
0.10	65	7	43

La Tabella 3.1 mostra come il numero di itemsets estratti risulti inversamente proporzionale alla percentuale fissata come soglia di *minimum support*. In base ai risultati ottenuti, abbiamo scelto di impostare *min_sup* al 6%, in quanto forniva il giusto compromesso tra numero di itemsets estratti e significatività degli stessi.

3.3 Estrazione delle regole di associazione

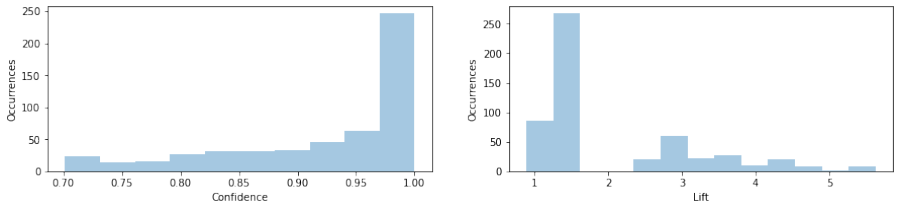
Una volta estratti gli itemsets frequenti e fissato il valore di *min_sup*, ci siamo concentrati sull'estrazione delle regole di associazione, variando il valore di *min_conf* tra 70% e 95% e studiandone anche i valori di lift associati, in modo da poter determinare la soglia più adeguata. Il numero di regole ottenuto per ogni valore di *min_conf* è indicato in Tabella 3.2

La Figura 3.2 mostra la distribuzione di confidence e lift per i diversi valori di *min_conf*. Come si può notare,

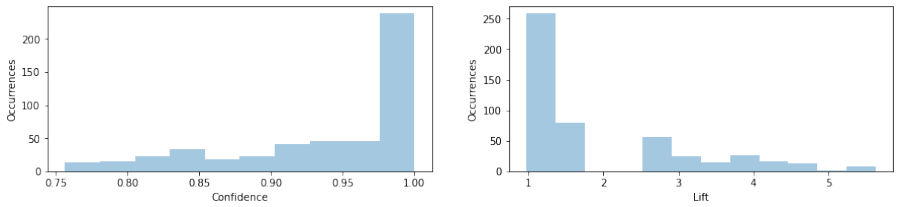
Tabella 3.2: Numero di regole di associazione al variare di *min_conf*.

min_conf	0.70	0.75	0.80	0.85	0.90	0.95
numero di regole	492	475	450	416	348	288

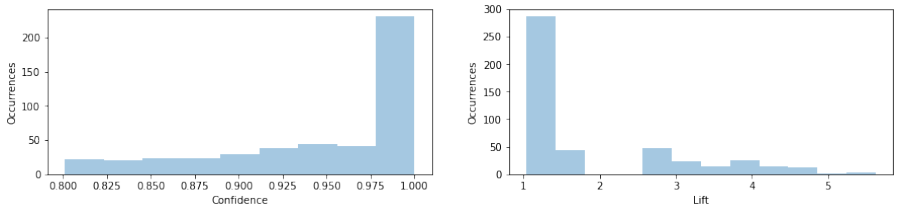
i valori di confidence risultano maggiormente concentrati nell'intervallo [0.98, 1.00]. Per quanto riguarda i grafici di lift, notiamo che aumentando il valore di *min_conf* otteniamo degli intervalli più definiti. In particolare, notiamo che la maggior parte dei valori si trova nell'intervallo [1.2, 1.5], con alcuni picchi tra 2.5 e 5.0.
I valori interessanti di lift sono quelli maggiori di 1, in quanto valori vicini ad 1 indicano che le occorrenze degli items nella testa e nel corpo della regola sono praticamente indipendenti. Tenendo conto di questa proprietà e della distribuzione di confidence, abbiamo deciso di fissare *min_conf* a 90%, in quanto questo valore rappresenta un buon compromesso tra il numero di regole ottenute e le proprietà che presentano.



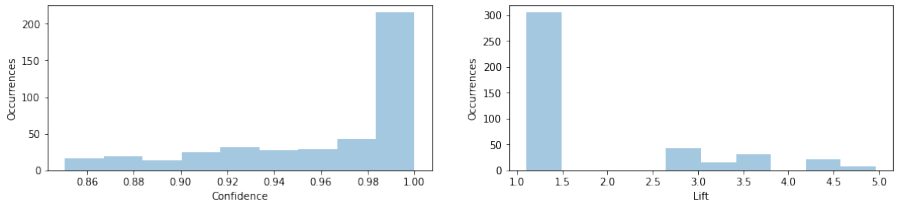
(a) 70%.



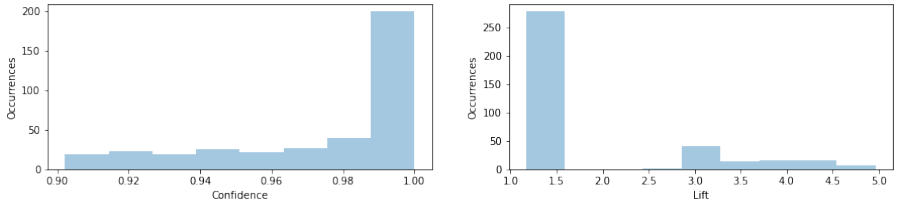
(b) 75%.



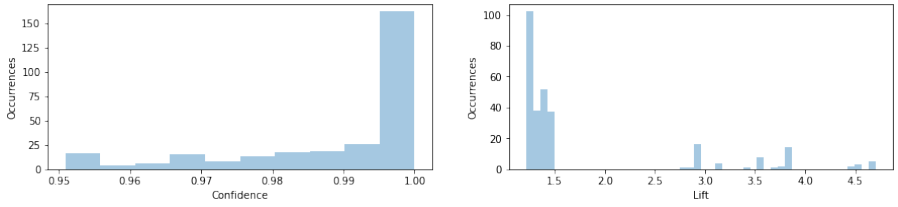
(c) 80%.



(d) 85%.



(e) 90%.



(f) 95%.

Figura 3.2: Istogrammi di *confidence* e *lift* con diversi valori di *min_conf*.

3.4 Predire se la stanza sarà occupata usando le regole di associazione

Per poter predire se l'ufficio risulterà occupato o meno, abbiamo filtrato le regole di associazione ottenute identificando quelle che contengono la variabile target *Occupancy* nella testa, dividendole ulteriormente in regole con Occupancy = 1 e Occupancy = 0.

La Tabella 3.3 mostra le regole ottenute, in ordine crescente di lift, aventi *Occupancy* = 1 come conseguenza e impostando *min_sup* = 0.06 e *min_conf* = 0.9.

Tabella 3.3: Le 4 regole aventi lift più alto per Occupancy = 1.

Rule	**Confidence**	**Lift**
('L5', 'T3') → 'OC1'	0.9918	4.6713
('L5') → 'OC1'	0.9676	4.5572
('T4', 'L5') → 'OC1'	0.9554	4.4999
('L5', 'HR3') → 'OC1'	0.9206	4.3357

Si può notare come l'attributo *Light* dia indicazioni precise della presenza di qualcuno nella stanza: il valore "L5" è individuabile in tutte le regole di associazione che portano ad avere *Occupancy* = 1 e il valore di lift sempre superiore a 4 indica che le variabili risultano positivamente correlate. Inoltre, dalla Tabella si può evincere che anche alcuni valori di *Temperature* ("T3" e "T4") combinati al valore "L5" aumentino la probabilità di trovarsi in una situazione in cui l'ufficio risulti occupato.
Per quanto riguarda le regole ottenute per *Occupancy* = 0, nella Tabella 3.4 vediamo che questo scenario risulta molto meno caratterizzato, dato che il lift più alto risulta comunque molto vicino a 1.

Tabella 3.4: Le 7 regole aventi lift più alto per Occupancy = 0.

Rule	**Confidence**	**Lift**
('H2', 'T3', 'HR2', 'C1', 'L1') → 'OC0'	1.0	1.2695
('H4', 'HR3', 'T1', 'C1', 'L1') → 'OC0'	1.0	1.2695
('H4', 'HR3', 'T2', 'C1', 'L1') → 'OC0'	1.0	1.2695
('T1', 'H3', 'HR2', 'C1', 'L1') → 'OC0'	1.0	1.2695
('T2', 'H1', 'HR1', 'C1', 'L1') → 'OC0'	1.0	1.2695
('C1', 'L1') → 'OC0'	1.0	1.2695
('L1') → 'OC0'	0.9998	1.2693

Sono state rilevate 63 regole a pari valore di lift per *Occupancy* = 0. Avendo ritenuto questo numero troppo elevato, abbiamo cercato di estrarre le regole più significative tra esse. Come si nota in Tabella 3.4, tutte le regole includono C1 ed L1, che corrispondono a valori di *CO2* e *Light* bassi. Inoltre, la regola ("C1", "L1" → "OC0") mostra gli stessi valori di *confidence* e *lift* rispetto alle regole con più elementi, mentre la sola regola ("L1" → "OC0") presenta valori appena inferiori. Questo ci fa capire che caratterizzare maggiormente le regole non migliora la precisione delle stesse. Per questi motivi, le regole che abbiamo ritenuto più significative sono quelle con ("C1", "L1") e "L1".
Per valutare l'accuratezza di queste regole nel predire il valore di *Occupancy*, abbiamo utilizzato uno dei due file di test presenti nel dataset, nello specifico il file `datatest.txt` contenente 2665 entries.

Tabella 3.5: Valori di *coverage* e *accuracy* delle regole di associazione per OC1 (sinistra) e OC0 (destra).

Rule	**Coverage**	**Accuracy**
('L5', 'T3') → 'OC1'	16.15%	93.45%
('L5') → 'OC1'	67.08%	94.22%
('T4', 'L5') → 'OC1'	6.27%	100%
('L5', 'HR3') → 'OC1'	21.29%	93.24%

Rule	**Coverage**	**Accuracy**
('C1', 'L1') → 'OC0'	62.19%	100%
('L1') → 'OC0'	95.45%	100%

In Tabella 3.5 sono riportati i valori di coverage ed accuracy delle regole trovate. Coverage rappresenta la percentuale di elementi presi in considerazione dalla regola analizzata rispetto al numero totale di elementi con il rispettivo valore di Occupancy. Accuracy, invece, rappresenta la percentuale di elementi che presentano effettivamente il valore di Occupancy dichiarato dalla regola analizzata, relativamente al numero totale di elementi individuati dal corpo della regola.
Come si può vedere dalla tabella in questione, per quanto riguarda le regole per Occupancy = 1 la regola

("L5"→"OC0") presenta una copertura di più della metà del dataset, con un'accuracy abbastanza elevata. L'unica regola con accuracy del 100% è quella che aggiunge T4, abbassandone però notevolmente la copertura. Le altre due regole, invece, riducono sia l'accuracy, sia la copertura.
Per Occupancy = 0, invece, si nota subito come i valori di copertura ed accuracy siano notevolmente più alti. In particolare, la regola che comprende solamente "L1" presenta una accuracy del 100%, con copertura quasi totale. Possiamo quindi affermare che questa regola sia più indicata rispetto all'altra considerata, in quanto a parità di accuracy abbiamo una copertura più ampia.

3.5 Considerazioni finali

I risultati ottenuti confermano le ipotesi precedentemente effettuate sull'incisività del parametro *Light* rispetto al valore del parametro target *Occupancy*. In particolar modo, quando abbiamo valori tendenti a 0 della luminosità, l'ufficio risulta sempre vuoto.

4. Classificazione

Il task di Classificazione prevede di costruire dei classificatori, che, in base ai valori di input, stabiliscano se l'ufficio risulterà occupato oppure no. A tale scopo abbiamo utilizzato 4 principali modelli: Decision Trees, Random Forest, Naive Bayesian, e K-NN.

4.1 Decision Trees

Come primo modello abbiamo utilizzato i Decision Trees, che dividono il dataset iterativamente in base alla condizione di test che produce il gain maggiore. Ogni nodo interno indica un test su un attributo, ogni ramo indica il risultato di un test e ogni nodo foglia contiene un'etichetta di classe.

4.1.1 Ricerca dei parametri

Per trovare i parametri ottimali, abbiamo eseguito una Grid Search preliminare sui seguenti range di valori:

- criterion: ['gini', 'entropy']
- max_depth: [None, 2, 3, 4, ..., 20]
- min_samples_split: [2, 5, 10, 20, 30, 50, 100]
- min_samples_leaf: [1, 5, 10, 20, 30, 50, 100]

In base ai risultati ottenuti, abbiamo deciso di eseguire una seconda Grid Search, cercando di affinare ulteriormente i parametri del nostro modello migliore. I range utilizzati per la seconda ricerca sono i seguenti:

- criterion: ['gini', 'entropy']
- max_depth: [5, 6, 7, ..., 15]
- min_samples_split: [50, 100, 125, 150, 175, 200, 225]
- min_samples_leaf: [30, 35, 40, 45, 50, 55, 60, 70, 80, 90, 100]

La ricerca ha restituito un modello avente i seguenti parametri: [max_depth = 5, min_samples_leaf = 40, min_samples_split = 125, criterion='gini'].
Non abbiamo trovato differenze sostanziali tra valutare gli split con il criterio gini e quello entropy.

4.1.2 Analisi del modello

In Tabella 4.1 sono riportati i valori di accuracy ottenuti dal modello migliore sui vari set. Per quanto riguarda Validation, essa è stata calcolata utilizzando 10-fold cross validation, mentre per Training è stata calcolata sull'intero training set, dopo aver allenato il modello su tutti i dati a disposizione. Come si può notare, l'accuracy

Tabella 4.1: Valori di accuracy per Decision Trees

Training	Validation	Test 1	Test 2
98.9%	98.3%	95.4%	95.2%

in fase di training risulta significativamente più alta di quella in fase di test. Questo è generalmente indice di un leggero overfitting.

Tabella 4.2: Indici di performance su test 1, divisi per Occupancy

Occupancy	Precision	Recall	F1-score
0	0.9481	0.9817	0.9646
1	0.9660	0.9064	0.9352

Tabella 4.3: Indici di performance su test 2, divisi per Occupancy

Occupancy	Precision	Recall	F1-score
0	0.9466	0.9956	0.9705
1	0.9794	0.7887	0.8737

Come si può notare dalle Tabelle 4.2 e 4.3, il modello risulta più accurato per valori di Occupancy pari a 0. Infatti, l'indice *F1-score* per Occupancy = 0 risulta maggiore di quello per Occupancy = 1 su entrambi i set di Test. Abbiamo deciso di basarci sulla misura dell'*F1-score* in quanto la distribuzione delle classi risulta irregolare e non ci sono motivazioni tali per dare priorità a *precision* o *recall*.

4.1.3 Albero di decisione

L'albero di decisione ottenuto tramite il training con i parametri ottimali è quello mostrato in Figura 4.1.

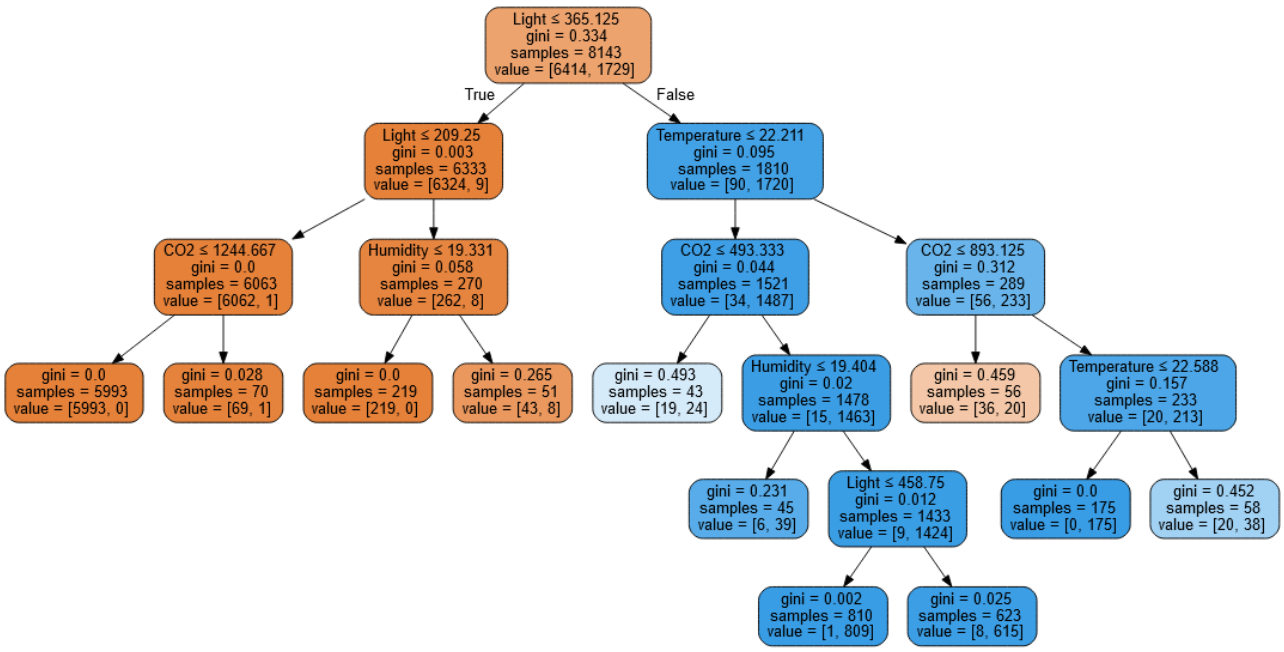


Figura 4.1: Albero di decisione, *Occupancy* = 0 (arancione) e *Occupancy* = 1 (blu)

Come si può notare, già dal primo split la divisione tra samples con *Occupancy* = 0 ed *Occupancy* = 1 risulta nettamente marcata. Degno di nota il fatto che questa divisione sia stata prodotta analizzando il paramentro *Light*, che, come visto in precedenza, risulta essere il più influente fra tutti.
Si può inoltre notare come il sottoalbero che individua i samples con *Occupancy* uguale a 0 (quello di sinistra) abbia profondità minore e presenti divisioni più nette rispetto a quello dei samples con *Occupancy* uguale ad 1 (destra).

4.2 Random Forest

Come secondo modello abbiamo deciso di utilizzare Random Forest, ovvero un tipo di modello ensemble, che si avvale del bagging come metodo di ensemble e dell'albero decisionale come modello individuale.

4.2.1 Ricerca dei parametri

Per trovare i valori ottimali dei parametri, abbiamo eseguito una Grid Search sui seguenti range di valori:

- n_estimators: [100, 300, 500, 800, 1200]
- max_depth: [None, 2, 5, 10, 20, 50, 75, 100]
- min_samples_split: [2, 5, 10, 20, 50, 100]
- min_samples_leaf: [1, 5, 10, 30, 50]

Non abbiamo inserito tra i parametri della ricerca il criterio di valutazione dell'errore (gini, entropy), in quanto in precedenza non sono state riscontrate differenze sostanziali. In questo modo abbiamo potuto dimezzare il tempo di esecuzione della Grid Search.
La ricerca ha restituito un modello avente i seguenti parametri: [max_depth = 5, min_samples_leaf = 1, min_samples_split = 100, n_estimators = 300].

4.2.2 Analisi del modello

In Tabella 4.4 sono riportati i valori di accuracy ottenuti dal modello migliore sui vari set. Si può vedere come il modello appaia costante in fase predittiva, riportando performance consistenti tra loro in fase di training e di test (le performance sul test 2 addirittura superano quelle sul training set).

Tabella 4.4: Valori di accuracy per Random Forest

Training	Validation	Test 1	Test 2
98.9%	98.6%	97.8%	99.3%

Come si può notare dalle Tabelle 4.5 e 4.6, il modello risulta più performante nelle istanze in cui Occupancy è uguale a 0, mentre appare leggermente meno performante nei casi in cui Occupancy è uguale ad 1.

Tabella 4.5: Indici di performance su test 1, divisi per Occupancy

Occupancy	Precision	Recall	F1-score
0	0.9982	0.9675	0.9826
1	0.9463	0.9969	0.9709

Tabella 4.6: Indici di performance su test 2, divisi per Occupancy

Occupancy	Precision	Recall	F1-score
0	0.9988	0.9922	0.9955
1	0.9714	0.9956	0.9834

4.3 Bonus

Come task bonus abbiamo scelto di implementare Naive Bayesian e K-NN, e di fornire un confronto con i precedenti modelli utilizzati in fase di classificazione, dal punto di vista delle performance sui test set e dei tempi di esecuzione.

4.3.1 Naive Bayesian

Per questo modello, abbiamo scelto di utilizzare le tre tipologie di Naive Bayesian solitamente più utilizzate: Gaussian, Bernoulli e Multinomial.

Per quanto riguarda Gaussian e Bernoulli, abbiamo utilizzato le loro implementazioni standard, messe a disposizione dalla libreria `scikit-learn` (GaussianNB, BernoulliNB), mentre Multinomial è stato implementato tramite ComplementNB. Una Complement Naive Bayesian è una Multinomial Naive Bayesian, con l'aggiunta di alcune accortezze che la rendono particolarmente adatta a dataset non bilanciati. Considerata la natura del nostro dataset, abbiamo dunque optato per questa variante, piuttosto che per le classiche Multinomial Naive Bayesian.

4.3.2 K-NN

K-NN è un modello che si basa sulla media degli output dei samples più vicini al valore che vogliamo predire. L'unico parametro da modulare è K, che rappresenta il numero di vicini da prendere in considerazione durante il calcolo della media. Effettuando alcune prove variando K in un range di valori da 1 a 500, abbiamo rilevato una percentuale di accuracy più elevata nel range 80-120. Abbiamo quindi deciso di scegliere K = 90, che è circa la radice quadrata del numero di samples del nostro training set, e pare essere il valore che restituisce l'accuracy migliore.

4.4 Confronto

Per confrontare i modelli abbiamo utilizzato la piattaforma Google Colab, eseguendo tutti i test nella stessa istanza al fine di limitare le fluttuazioni dovute al diverso carico di lavoro dei server. I tempi sono stati approssimati alla quarta cifra decimale, avendo quindi una precisione nell'ordine del millisecondo.
In Tabella 4.7 sono riportati i valori ottenuti in fase di training e test con i vari modelli provati. Come si può notare, il modello che risulta migliore per quanto riguarda l'accuracy è Random Forest. Degno di nota però è anche il modello Naive Bayesian di tipo Gaussian, che riporta valori di accuracy leggermente inferiori, ma con tempi di training e prediction più bassi di diversi ordini di grandezza.
Il peggior modello risulta invece Naive Bayesian di tipo Bernoulli, che, seppur registrando tempi piuttosto bassi ed un valore di accuracy sul test 1 comparabile agli altri modelli, riporta risultati relativamente scarsi sul test 2.
Il modello con il tempo di training più elevato risulta Random Forest, mentre i tempi di prediction maggiori risultano essere, senza sorpresa, quelli di K-NN, a causa del calcolo delle distanze per trovare i samples più vicini (operazione dal costo computazionale generalmente elevato).
Da segnalare, inoltre, il fatto che i parametri ottimali per Random Forest siano stati ottenuti con una Grid Search da circa 1 ora e mezza, e che Decision Tree, invece, abbia richiesto due Grid Search da circa 15 minuti ciascuna, per un totale di mezz'ora circa. Questa operazione non è stata necessaria per gli altri modelli, che non hanno richiesto particolari ottimizzazioni di questo tipo.

Tabella 4.7: Tempi e valori di Accuracy dei migliori modelli

Modello	Tempo training	Tempo test 1	Tempo test 2	Accuracy test 1	Accuracy test 2
Decision Tree	0.0165	0.0021	0.0024	95.4%	95.2%
Random Forest	1.3994	0.0550	0.1486	97.8%	99.3%
Gaussian	0.0038	0.0013	0.0019	97.7%	98.7%
Complement	0.0031	0.0007	0.0008	97.1%	94.0%
Bernoulli	0.0068	0.0019	0.0020	97.0%	82.4%
K-NN	0.0216	0.1177	0.4035	97.7%	96.9%