
**MODELLI PROBABILISTICI PER LE DECISIONI**

UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

DIPARTIMENTO DI INFORMATICA, SISTEMISTICA E COMUNICAZIONE

# Sistema predittivo per il bike sharing: modelli bayesiani in R per inferire il tasso di utilizzo delle bici

*Autori:*
Mattia Pennati - 793375
Luca Virgilio - 794866

Data: 17 giugno 2018

capital bikeshare

*”It’s great exploring a new city by bike,*
*you see things in an entirely different way.”*

A BIKE SHARING USER, WASHINGTON, DC

# Indice

# 1 Introduzione e Motivazioni

I sistemi di bike sharing sono l'evoluzione dei tradizionali servizi di noleggio bici. Essi infatti, non si limitano alla semplice operazione di noleggio quotidiano di una bicicletta, ma permettono sia corse di breve durata che la sottoscrizione ad abbonamenti di diverso genere. La principale novità introdotta dal bike sharing, oltre a un prezzo accessibile a chiunque, è la possibilità di riconsegnare la bici in una postazione differente rispetto a quella di partenza, tramite un sistema semplice e automatizzato.
Anno dopo anno, sono sempre più numerosi i sistemi di bike sharing che si stanno sviluppando in tutto il mondo. Questa moderna idea di trasporto suscita un interesse sempre maggiore con il passare del tempo per le sue ripercussioni nell'ambito dei trasporti, della salute e dell'ambiente.
Differentemente dagli altri sistemi di trasporto, come autobus o metropolitane, tramite i sistemi di bike sharing è possibile registrare con esattezza momento e posizione in cui viene prelevata e restituita una bici. Grazie ai dati forniti dal sistema di "Capital Bikeshare" (CBS) di Washington, D.C, USA (`https://www.capitalbikeshare.com/`), uniti con i dati climatici forniti dal servizio meteo "freemeteo" (`www.freemeteo.com`), sono stati costruiti alcuni modelli probabilistici bayesiani, che verranno presentati nelle sezioni successive, per stimare il numero di bici utilizzate in un determinato momento sotto determinate assunzioni e condizioni.
Modelli di questo genere possono risultare uno strumento di supporto molto utile nelle analisi del servizio atte a prendere decisioni quali il miglior periodo o momento per una manutenzione, la variabilità del costo degli abbonamenti, la disponibilità di biciclette necessarie per una certa giornata e così via. Inoltre, nonostante per il momento sia solo uno spunto per sviluppi futuri, i modelli implementati permetterebbero anche la stima di altri fattori, come quelli climatici o sociali, partendo dal flusso di biciclette utilizzato, come proposto da J. Gama [1], fornitore del dataset utilizzato.
A livello implementativo il lavoro presentato è stato svolo in `R` [2], tramite l'ausilio di alcune librerie per la modellazione e inferenza di reti bayesiane come `bnlearn` [3] e `gRain` [4], basandosi su alcuni libri di riferimento per l'argomento trattato [5][6].

# 2 Dati

In questa sezione verrà descritto brevemente il dataset su cui si è lavorato e le modifiche apportate prima di procedere alla costruzione dei modelli predittivi.

## 2.1 Descrizione del dataset

Il dataset utilizzato è, come anticipato nell'introduzione, un'integrazione ed una rielaborazione dei dati relativi al bike sharing per gli anni 2011/2012 e dei dati climatici per quel periodo.

Esso è stato prodotto, descritto e analizzato da Joao Gama [1] e risulta disponibile nella repository UCI Machine Learning, all'indirizzo `https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset`.
I dati raccolti sono stati organizzati in due modalità differenti, nel file `hour.csv`, i dati registrati sono suddivisi in ore, mentre in quello `day.csv` in base al giorno. Al fine di raggiungere il nostro scopo e per avere un numero maggiore di istanze, si è scelto di lavorare solo sul file `hour.csv`. Esso risulta ovviamente molto più numeroso e permetterà inoltre di evidenziare l'effetto delle varie fasce orarie e delle diverse condizioni climatiche per le quali si potranno utilizzare informazioni più precise anzichè dati medi relativi all'intera giornata.

## 2.2 Struttura del dataset

La struttura del dataset scelto come base per creare e analizzare alcuni modelli predittivi, ovvero il dataset `hour.csv`, è descritta e analizzata sommariamente nella Tabella 1 riportata in seguito. Esso è composto da 17 colonne, sia discrete che continue, possiede 17379 istanze e non contiene nessun valore nullo.

## 2.3 Pre-processing e discretizzazione del dataset

Prima di inferire il modello della rete bayesiana abbiamo eliminato alcuni attributi non ritenuti utili al fine dell'elaborazione perchè ridondanti o ininfluenti per l'obiettivo prefissato. In particolare sono state eliminate le colonne instant, dteday, yr, casual e registered.
Successivamente, per uniformare il tipo delle variabili di cui si dispone, si è deciso di discretizzare le variabili continue basandosi sulla personale conoscenza del dominio e su un processo iterativo di osservazione della numerosità degli eventi nelle diverse discretizzazioni effettuate e dei loro effetti sulle performance, come analizzato nel capitolo sui risultati.
Prima di questa scelta si è provato a eseguire la funzione di discretizzazione fornita dalla libreria `bnlearn`. Essa permette di scegliere fra una discretizzazione a intervalli, una discretizzazione a quantili e la discretizzazione basata sul metodo "Hartemink's pairwise mutual information" [7]. Nonostante le differenti possibilità, le performance dei modelli che saranno presentati successivamente sono risultate poco soddisfacenti e comunque minori di quelle ottenute con una discretizzazione manuale.
Per quanto riguarda le variabili già discrete si è mantenuta la discretizzazione iniziale in quanto ritenuta soddisfacente anche se forse troppo semplicistica per quanto riguarda l'attributo `weathersit`. Essa infatti suddivide le possibili condizioni climatiche in soli 4 gruppi, indica con 1 condizioni di tempo sereno o parzialmente nuvoloso, con 2 situazioni nuvolose o presenza di nebbia, con 3 pioggia o neve leggera e con 4 temporali, grandinate o forti nevicate.
L'attributo ora (`hr`), il cui rapporto con il numero di bici utilizzate è visualizzabile in

| Feature | Descrizione | Continuo o Discreto | Valore minimo | Valore massimo | Valore medio o più frequente |
|---|---|---|---|---|---|
| istant | indice univoco del record | Discreto | 1 | 17379 | // |
| dteday | data | Discreto | 01/01/2011 | 31/12/2012 | // |
| season | stagione | Discreto | 1 (Inverno) | 4 (Autunno) | 3 |
| yr | anno | Discreto | 0 (2011) | 1 (2012) | 1 |
| mnth | mese | Discreto | 1 (Gennaio) | 12 (Dicembre) | 5 e 7 |
| hr | ora | Discreto | 0 | 23 | 16 e 17 |
| holiday | giorno festivo | Discreto | 0 (No) | 1 (Sì) | 0 |
| weekday | giorno della settimana | Discreto | 0 (Domenica) | 6 (Sabato) | 6 |
| workingday | giornata lavorativa (sabato e domenica considerati festivi) | Discreto | 0 (No) | 1 (Sì) | 1 |
| weathersit | condizioni meteorologiche | Discreto | 1 (Sereno) | 4 (Temporali, nevicate fitte, ...) | 1 |
| temp | temperatura in Celsius normalizzata | Continuo | 0.02 (-8°C) | 1 (+39°C) | 0.497 (15.36°C) |
| atemp | temperatura percepita in Celsius normalizzata | Continuo | 0.0 (-16°C) | 1 (+50°C) | 0.476 (15.42°C) |
| hum | valore di umidità % normalizzato | Continuo | 0.0 (0%) | 1 (100%) | 0.63 (63%) |
| windspeed | velocità del vento in $\frac{m}{s}$ normalizzata | Continuo | 0.0 ($0\frac{m}{s}$) | 0.8507 ($56.99\frac{m}{s}$) | 0.19 ($12.73\frac{m}{s}$) |
| casual | numero di bici prelevate da utenti non registrati | Discreto | 0 | 367 | 35.68 |
| registered | numero di bici prelevate da utenti registrati | Discreto | 0 | 886 | 153.80 |
| cnt | numero totale di bici prelevate | Discreto | 1 | 977 | 189.50 |

**Tabella 1:** Analisi del Dataset `hour.csv`

Figura 1, non è stato suddiviso in fasce orarie perchè già discreto e perché, sebbene l'utilizzo delle bici si concentri in particolari fasce orarie, un eventuale discretizzazione portava ad un impoverimento dei modelli e ad un conseguente peggioramento dei risultati.
Le variabili continue per le quali si è invece effettuata una discretizzazione manuale sono:

- `temp` e `atemp`: inizialmente temperatura registrata e temperatura percepita sono state discretizzate in quattro e cinque fasce di valori normalmente distribuite, ma dopo una fase di studio sul modello e sui risultati prodotti, abbiamo deciso di aumentare le fasce di valori rispettivamente a dieci e tredici in modo da avere un intervallo ogni 4°C per la temperatura reale e ogni 5°C per quella percepita.

- `hum`: inizialmente l'umidità è stata discretizzata in 5 intervalli differenti, ma, dopo un'analisi più accurata le fasce sono state portate a 9. Esse, osservando la distri-

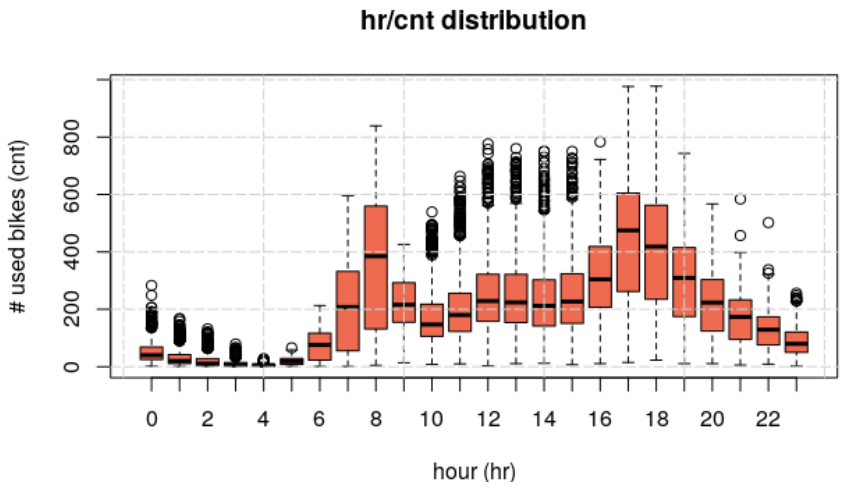



**Figura 1:** Hour plot

buzione dei valori, non avranno la stessa ampiezza; il primo intervallo indicherà infatti un tasso di umidità dallo 0 al 20%, considerandolo un'intervallo di valori raramente registrabile nella realtà, mentre ogni altra fascia rappresenterà un intervallo del 10% di umidità.

- `windspeed`: analogamente a `hum`, la velocità del vento è stata inizialmente suddivisa in 4 intervalli, ma dopo un'osservazione più accurata sulla distribuzione di valori è stata suddivisa in 8 fasce di valori, dove le prime 7 indicano un intervallo di 6,4$\frac{m}{s}$, mentre l'ultima, essendo un'intervallo difficile da raggiungere, rappresenterà i valori compresi fra 44,4$\frac{m}{s}$ e 57$\frac{m}{s}$, massimo valore registrato nel dataset.

- `cnt`: per il numero di bici utilizzate indipendentemente dal tipo di utente, sono stati eliminati gli attributi `casual` e `registered` e si è tenuto il solo attributo `cnt`. Esso è stato discretizzato in quattro intervalli diversi, come riportato in Tabella 2, partendo dalla numerosità delle istanze. Una considerazione importante durante la suddivisione in fasce per questa variabile è data dal fatto che gli anni presi in esame corrispondono al primo e al secondo anno dall'istallazione del servizio, con un numero di stazioni e di bici a disposizione limitato (1300/1500) ed in aumento. Per questo motivo si è ritenuto necessario discretizzare `cnt` con valori differenti per i due anni cercando di concentrarsi sulla percentuale di bici utilizzate e non sul numero.

Nonostante la discretizzazione per `cnt` tenga conto del differente numero totale di bici nei due anni considerati, il modello risulterà limitato dal fatto che un servizio necessita di un paio di anni prima di stabilizzarsi e affermarsi fra la popolazione locale. Nei successivi anni il numero di stazioni e bici ha continuato la sua crescita e si otterrebbero risultati migliori disponendo di dati più stabili o potendo calcolare il fattore di influenza

| Classe | Valore | Percentuale | 2011 | 2012 |
|---|---|---|---|---|
| molte bici | 4 | $cnt > 45\%$ | $cnt > 500$ | $cnt > 700$ |
| utilizzo medio bici | 3 | $25\% < cnt < 45\%$ | 500 | 700 |
| poche bici | 2 | $8\% < cnt < 25\%$ | 250 | 350 |
| pochissime bici | 1 | $cnt < 8\%$ | 80 | 120 |

**Tabella 2:** Discretizzazione numero di bici

dato dalla popolarità del servizio. Non disponendo momentaneamente di questi dati si sono ritocatte verso il basso le soglie di appartenenza ai diversi intervalli di `cnt`, come si può notare dalla tabella. Le percentuali che stabiliscono l'appartenenza ad un determinato intervallo saranno infatti più basse per l'anno 2011.

## 2.4 Analisi dei dati

Per riassumere la distribuzione dei valori delle variabili del dataset, supportare la precedente fase di discretizzazione e fornire spunti per la successiva fase di creazione del modello, sono riportati a seguire diversi grafici relativi a distribuzione e influenza rispetto a `cnt`, come riportato per `hr` nella sezione precedente, per le variabili continue Figura 3, le variabili discrete Figura 2 e le variabili continue dopo la discretizzazione Figura 4.

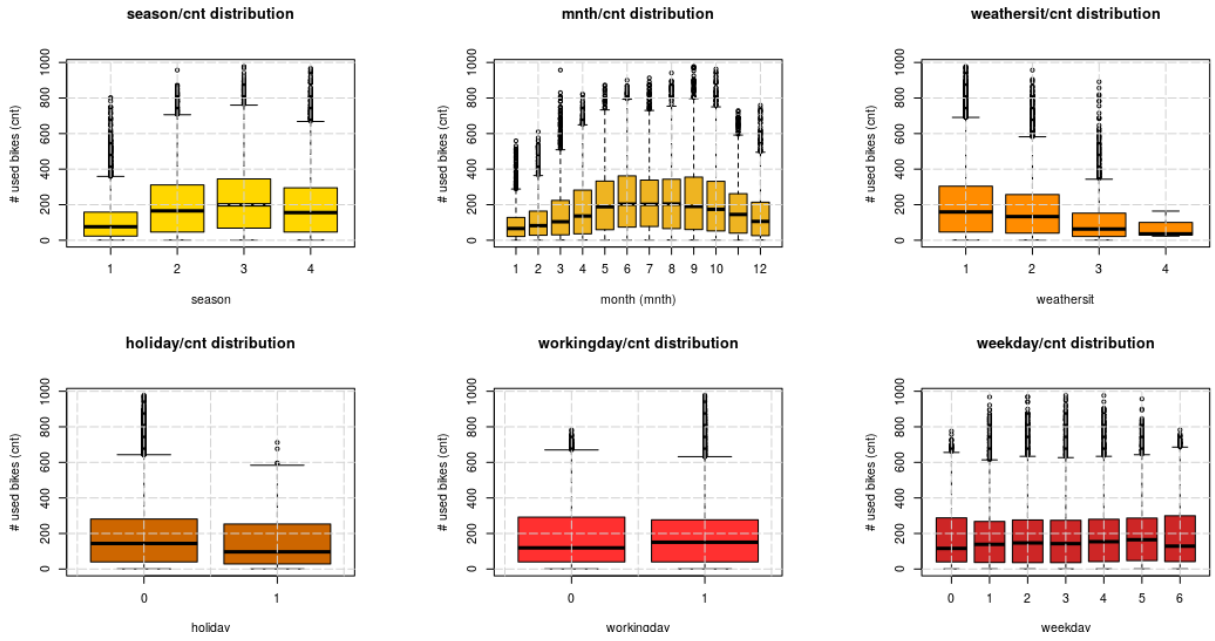



**Figura 2:** Rapporto delle variabili discrete del dataset con il numero di bici utilizzate

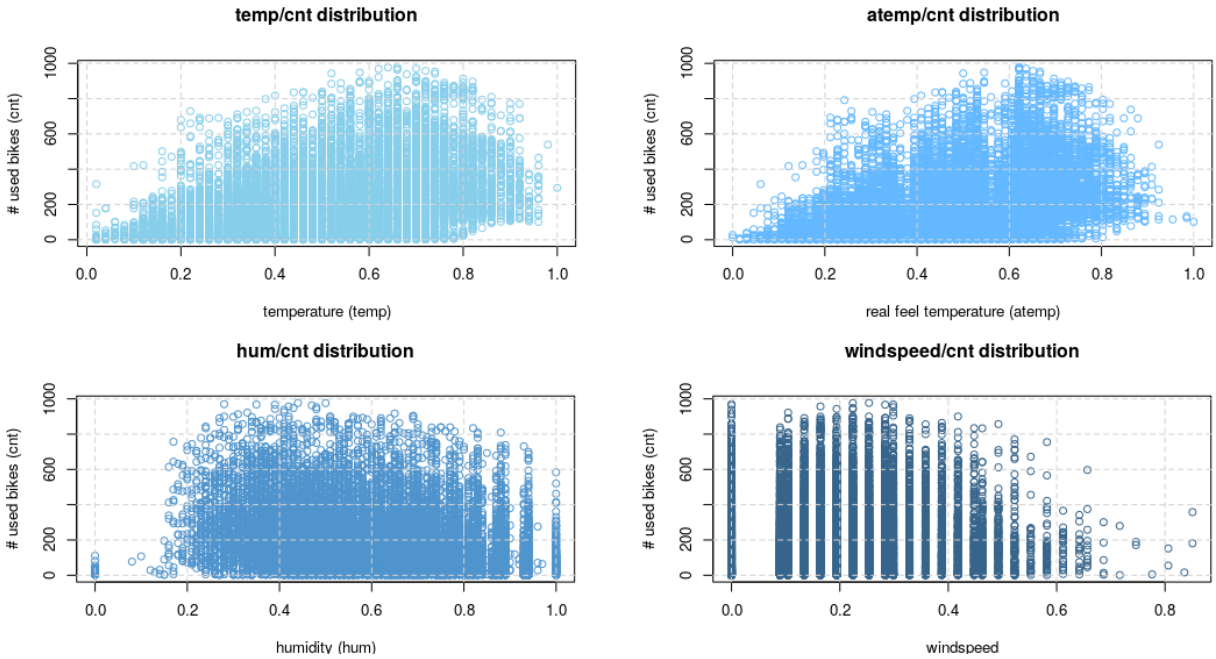



**Figura 3:** Rapporto delle variabili continue del dataset con il numero di bici utilizzate

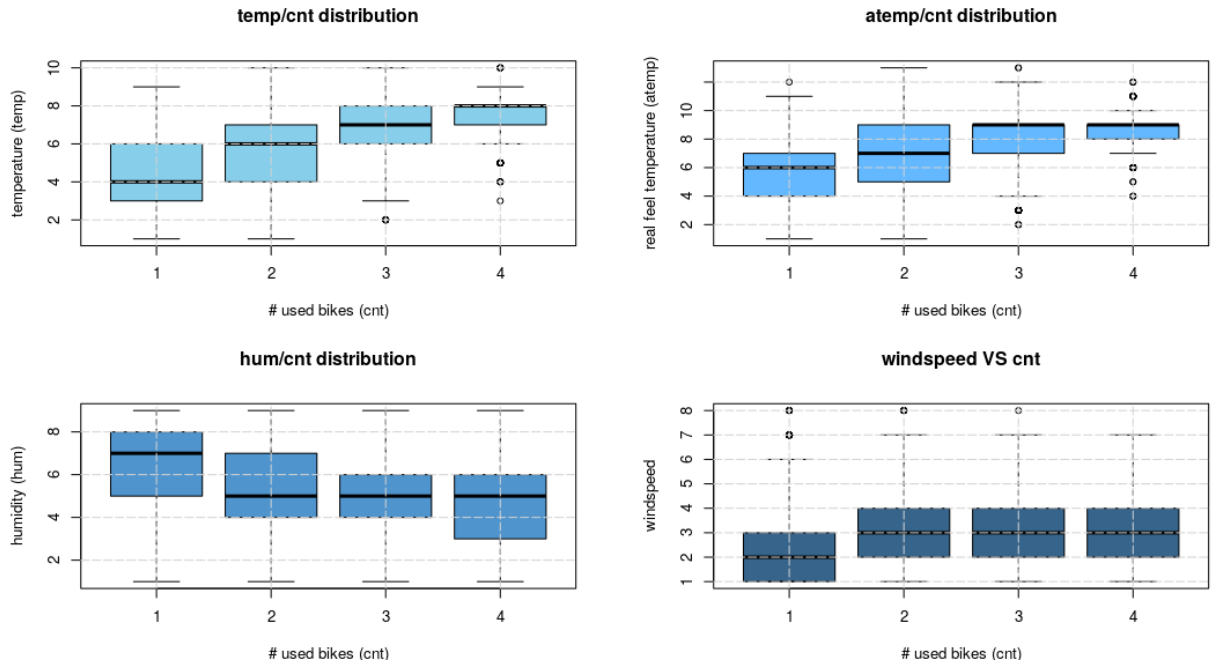



**Figura 4:** Rapporto delle variabili continue del dataset dopo la discretizzazione con il numero di bici utilizzate

# 3 Rete bayesiana

Come riportato da J. Gama i dati su cui si lavora non sono ideali per essere rappresentati da una serie temporale, ma si prestano molto meglio ad una modellazione basata su causalità. Per questo motivo si eviteranno modelli markoviani e ci si concentrerà sulla definizione di modelli bayesiani.

## 3.1 Struttura della rete

Esistono diversi modi per definire il modello di una rete bayesiana, partendo da una definizione manuale e arrivando ad algoritmi in grado di inferire da soli la struttura della rete. Il nostro approccio consiste nell'inferire dai dati il modello della rete tramite appunto l'utilizzo di diversi algoritmi, analizzare i diversi modelli ottenuti e migliorare la rete ottenuta tramite la conoscenza del domino e diverse analisi della correlazione fra le variabili del modello.
La struttura della rete può essere costruita tramite algoritmi di apprendimento di tre diverse categorie [5]:

- *Constraint-based structure learning algorithms*: gli algoritmi di questa classe utilizzano test di indipendenza per verificare coppie di variabili indipendenti. Una volta ottenute tutte le coppie dipendenti, gli archi vengono orientati. Per migliorare le performance degli algoritmi, vengono utilizzate le Markov blanket dei singoli nodi, per semplificare l'identificazione dei vicini.

- *Score-based structure learning algorithms*: questi algoritmi rappresentano l'applicazione di un'euristica generale nella ricerca della migliore struttura della rete bayesiana. A ciascuna rete candidata viene assegnato un punteggio che riflette la bontà dell'adattamento della rete e tramite i diversi algoritmi si cerca di massimizzare tale punteggio. Una delle particolarità di questa famiglia di algoritmi, che verrà sfruttata nella definizione di alcuni modelli, è quella di poter accettare come input una rete già definita cercando di migliorarla e ottimizzarla.

- *Hybrid structure learning algorithms*: Questi algoritmi combinano quelli delle precedenti due categorie per ridurre le loro debolezze e costruire una rete affidabile in una grande varietà di situazioni. Nel primo step, chiamato restrict, si riduce lo spazio di ricerca in uno spazio più piccolo e regolare, mentre nel secondo step, detto maximise, si cerca di massimizzare una data funzione di punteggio.

Per inferire la struttura dei nostri dati, sono stati applicati i seguenti algoritmi:

- Constraint-based [8]:

  - PC: applicazione pratica dell'algoritmo IC [9] che costruisce la rete partendo da coppie di nodi, passando successivamente per le v-structure e concludendo con la ricerca degli archi più complessi per ottenere un DAG parzialmente connesso.

  - Grow-Shrink (GS): algoritmo basato su ”Grow-Shrink Markov blanket algorithm” [10] centrato sulla rilevazione delle Markov blanket.
  - Incremental Association (IAMB): algoritmo basato su ”Incremental Association Markov blanket algorithm” [11] composto da due fasi, una di forward e una di backward, con cui rileva le diverse Markov blanket.
  - Fast-IAMB [12] e Inter-IAMB [11]: varianti dell’algoritmo IAMB.

- Score-based:
  - Hill Climbing [13] e Tabu Search [14]
  - Hill Climbing e Tabu eseguiti partendo da ogni modello constraint-based ottenuto in precedenza

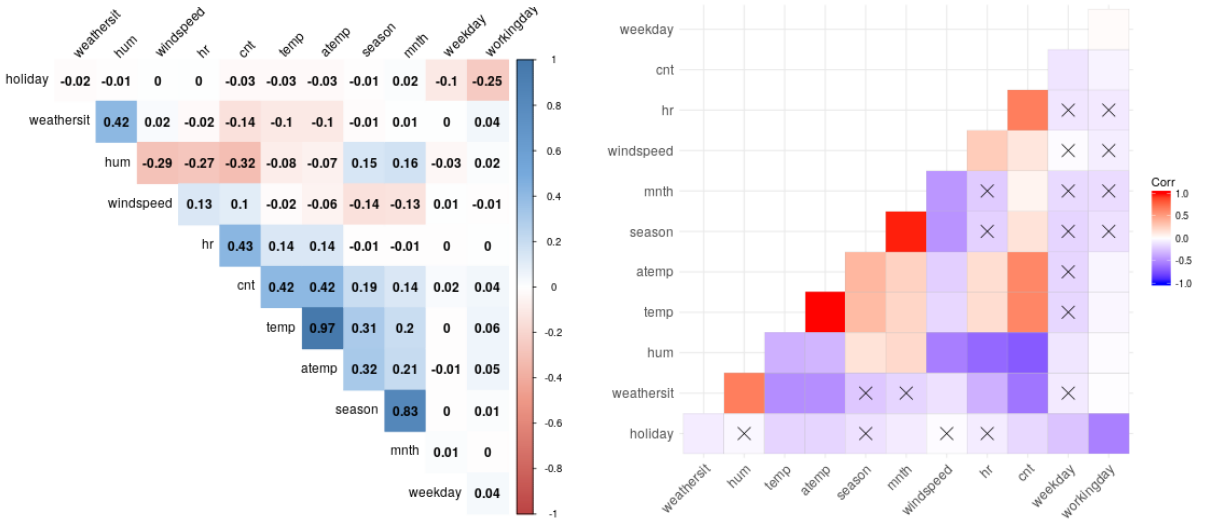
- Hybrid:
  - Max-Min Hill Climbing [15]: algoritmo composto da due fasi, ”restrict”, con la quale riduce il numero di candidati parenti per ogni nodo, e ”maximize”, nella quale applica l’algoritmo score based Hill Climbing.
  - Restricted Maximization: versione più generale dell’algoritmo Max-Min Hill Climbing.

Ognuno di essi è stato eseguito tramite una fase di bootstrap di 500 iterazioni e il modello finale per ogni algoritmo sara la media delle diverse reti ottenute comprendendo solo gli archi che sono presenti in almeno un determinato numero di iterazioni secondo una soglia. Questa fase irrobustisce gli archi ottenuti e riesce ad esplicitarne, teoricamente, la direzione in maniera molto più precisa garantendo che la rete finale sia la migliore ottenibile con quel determinato algoritmo.
I migliori risultati sono stati ottenuti dai seguenti algoritmi:

- il modello ottenuto dall’esecuzione dell’algoritmo inter-IAMB, dai quali combinando correlazione, risultati degli altri algoritmi e conoscenza del dominio verranno ottenuti 3 differenti modelli.

- il modello ottenuto dall’esecuzione dell’algoritmo tabu partendo dai modelli constraint-based. Fra le reti ottenute ne esiste infatti una ricorrente nei risultati dell’esecuzione di Tabu partendo da IAMB, fast-IAMB, inter-IAMB e Grow-Shrink, che verrà presa come riferimento.

Prima di riportare i 4 modelli finali ottenuti si illustra in Figura 5 un’analisi della correlazione. Essa, insieme alla analisi esplorativa del dataset fornita nella sezione precedente, è stata fondamentale nell’affinamento dei modelli. L’osservazione più importante che proviene da questa fase è l’inutilità della variabile `windspeed` nei confronti dell’obiettivo prefissata. Essa, oltre a risultare un nodo singolo nella metà delle reti ottenute dai vari algoritmi, non presenta una correlazione elevata con nessuna altra variabile e, come



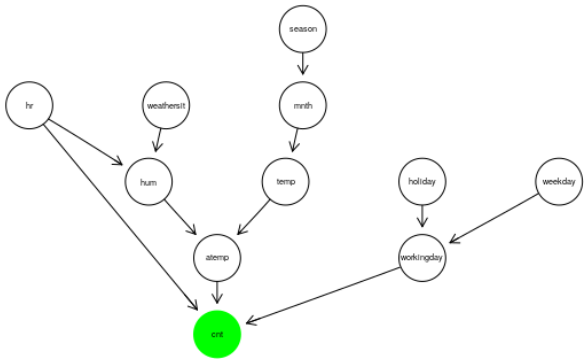

**(a)** Due esempi di matrice di correlazione: nel primo sono riportati i valori della correlazione fra le varie coppie, mentre nel secondo sono barrate le coppie che non presentano una correlazione sufficiente.


**(b)** Matrice di correlazione dove vengono mostrati dei grafici sommari per ogni coppia di variabili e dove i colori indicano il rapporto con l'attributo `cnt`.

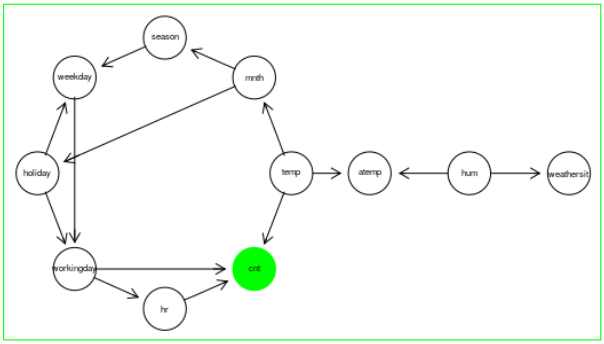
**Figura 5:** Analisi della correlazione delle variabili del dataset considerate nella definizione dei modelli bayesiani.

mostrato in Figura 4, non influenza in nessun modo il numero di bici utilizzate. Per questi motivi verrà eliminata dal dataset e da ogni modello finale. Questa assunzione è stata verificata e confermata in quanto la rimozione della variabile non ha minimamente modificato le performance del modello nella predizione del numero di bici utilizzate.

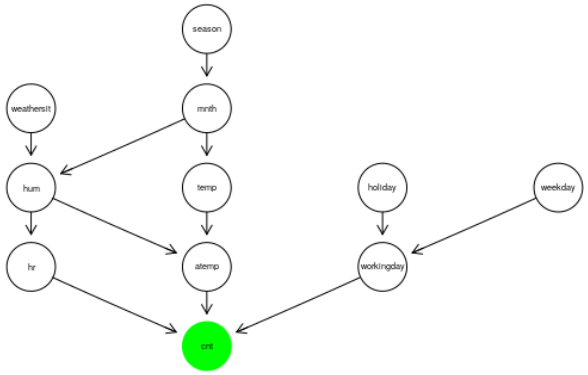
Ad analisi completata, dopo aver invertito la direzione di molti archi basandosi sulla conoscenza del dominio e dopo aver aggiunto alcuni degli archi mancanti fra quelli ritenuti importanti dall'articolo di riferimento per il dataset utilizzato (come temperatura o temperatura percepita, il fatto che sia o meno un giorno lavorativo e uno fra stagione o mese) i 4 modelli finali sono quelli riportati in Figura 6.



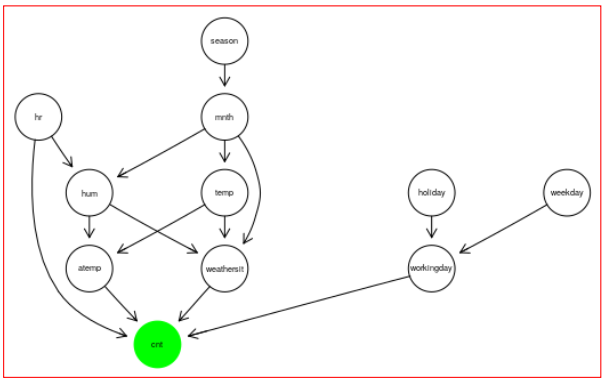

**(a)** Modello 1: modello con le stesse performance del modello 3 in fase di predizione




**(b)** Modello 2: modello con le migliori performance in fase di predizione




**(c)** Modello 3: modello con le stesse performance del modello 1 in fase di predizione




**(d)** Modello 4: modello con il tempo computazionale nettamente maggiore in fase di predizione

**Figura 6:** Analisi della correlazione delle variabili del dataset considerate nella definizione dei modelli bayesiani.

I modelli ottenuti, avendo performance molto simili come si analizza successivamente, verranno considerati alla pari fra loro. L'unico che verrà scartato sarà il quarto in quanto non fornisce risultati migliori degli altri, ma impiega un tempo nettamente superiore nella fase di predizione, mentre il migliore risulta il secondo.

## 3.2 Stima dei parametri

La stima e l'aggiornamento dei parametri di una distribuzione globale viene semplificata grazie all'utilizzo delle Markov Blanket. I due principali metodi per la stima dei parametri sono: *MLE*(maximum likelihood estimation), che non tiene conto delle probabilità a priori, ma stima solo quale sia l'evento più verosimile dai dati, e *Bayesian posterior estimation*, che considera invece anche le probabilità a priori.
Inizialmente abbiamo utilizzato l'approccio likelihood estimation per stimare le CPTs, ma, durante l'analisi dei risultati, abbiamo notato che esistevano un numero non trascurabile di risultati che non venivano stimati, ovvero fornivano stime nulle. Questo poiché tramite *MLE* spesso capita di non riuscire a stimare la probabilità per eventi considerati rari. Per risolvere questo problema abbiamo modificato il metodo di stima dei parametri con Bayesian estimation [16][17] che, utilizzando conoscenza a priori, riesce a stimare una probabilità anche per eventi rari.

# 4 Rete Naive-Bayes

Per poter valutare al meglio la rete illustrata nella sezione precedente, abbiamo deciso di costruire anche una semplice rete Naive-Bayes in modo da poter successivamente confrontare i due modelli.

## 4.1 Struttura della rete

Un modello Naive-Bayes è un particolare modello che prevede la variabile target come causa comune di tutte le altre variabili considerate variabili effetto e assunte come indipendenti fra loro. Questa semplificazione porta alla creazione di un classificatore bayesiano banale che prende infatti anche il nome di *Idiot Bayes Model* [18].
La struttura del modello Naive-Bayes nel nostro caso di studi è riportata in Figura 7 ed è composta dal nodo `cnt` come padre di tutte le altre variabili.

## 4.2 Stima dei parametri

Analogamente al caso di una rete Bayesiana, anche per un modello Naive Bayes, si possono utilizzare i due metodi citati precedentemente, ovvero *MLE*(maximum likelihood estimation) e *Bayesian estimation*. Anche in questo caso la scelta è ricaduta sul secondo metodo per gli stessi motivi elencati nella sezione sulla stima dei parametri per i modelli bayesiani.

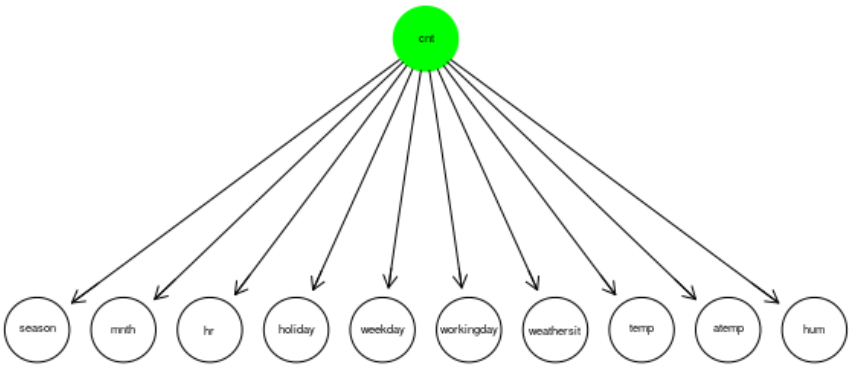



**Figura 7:** Modello Naive-Bayes: la variabile target `cnt` viene considerata come causa comune di tutte le altre variabili del modello.

# 5 Analisi dei Risultati

In questa sezione si andranno ad analizzare e confrontare i risultati dei 3 modelli bayesiani con quelli del modello Naive-Bayes. Dopo una breve analisi delle performance si mostreranno anche le curve ROC per ogni singola classe e la validazione dei modelli avvenuta tramite 5-fold cross validation.

## 5.1 Performance

Per valutare le performance della nostra rete bayesiana, abbiamo suddiviso i dati in diversi trainset e testset in modo da poter utilizzare la rete come classificatore nei confronti della variabile target. Le diverse coppie trainset-testest sono state create prendendo come trainset una percentuale sempre maggiore del dataset partendo dal 50%. La scelta non è casuale, corrisponde infatti all'utilizzo dell'anno 2011 come trainset e dell'anno 2012 come testset, seguendo l'idea proposta nell'articolo di riferimento del dataset.
Per valutare l'affidabilità del modello, per ogni iterazione abbiamo calcolato una matrice di confusione dalla quale ricavare le principali misure di qualità, ottenendo i seguenti risultati:

| Nome | Trainset | Testset | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|
| bnet_test1 | 50% | 50% | 0.821 | 0.824 | 0.747 | 0.784 |
| bnet_test2 | 55% | 45% | 0.829 | 0.807 | 0.784 | 0.795 |
| bnet_test3 | 60% | 40% | 0.835 | 0.797 | 0.782 | 0.789 |
| bnet_test4 | 65% | 35% | 0.827 | 0.793 | 0.771 | 0.782 |
| bnet_test5 | 70% | 30% | 0.833 | 0.795 | 0.778 | 0.786 |
| bnet_test6 | 75% | 25% | 0.827 | 0.793 | 0.791 | 0.792 |

**Tabella 3:** Risultati dei test per il modello 1.

| Nome | Trainset | Testset | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|
| bnet_test1 | 50% | 50% | 0.823 | 0.823 | 0.735 | 0.776 |
| bnet_test2 | 55% | 45% | 0.841 | 0.801 | 0.797 | 0.802 |
| bnet_test3 | 60% | 40% | 0.833 | 0.791 | 0.783 | 0.787 |
| bnet_test4 | 65% | 35% | 0.830 | 0.781 | 0.790 | 0.786 |
| bnet_test5 | 70% | 30% | 0.842 | 0.814 | 0.795 | 0.804 |
| bnet_test6 | 75% | 25% | 0.831 | 0.788 | 0.785 | 0.787 |

**Tabella 4:** Risultati dei test per il modello 2.

| Nome | Trainset | Testset | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|
| bnet_test1 | 50% | 50% | 0.821 | 0.824 | 0.747 | 0.784 |
| bnet_test2 | 55% | 45% | 0.834 | 0.801 | 0.787 | 0.794 |
| bnet_test3 | 60% | 40% | 0.827 | 0.785 | 0.765 | 0.775 |
| bnet_test4 | 65% | 35% | 0.828 | 0.789 | 0.781 | 0.785 |
| bnet_test5 | 70% | 30% | 0.841 | 0.822 | 0.786 | 0.804 |
| bnet_test6 | 75% | 25% | 0.823 | 0.787 | 0.752 | 0.769 |

**Tabella 5:** Risultati dei test per il modello 3.

Come accennato nella sezione introduttiva si sono confrontate le performance dei vari modelli su diverse discretizzazioni provate. I risultati sono riportati in Tabella 6 e mostrano come la discretizzazione utilizzata massimizzi, seppure di poco, le abilità del modello in fase di predizione

| Prova | Discretizzazione | | | | | | Network | Accuracy | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | cnt | hr | temp | atemp | hum | windspeed | | test1 | test2 | test3 | test4 | test5 | test6 |
| 0 | 4 | no | 10 | 13 | 9 | 8 | bayesian | 0.823 | 0.841 | 0.833 | 0.830 | 0.842 | 0.831 |
| | | | | | | | naive | 0.714 | 0.733 | 0.730 | 0.729 | 0.733 | 0.734 |
| 1 | 4 | no | 10 | 13 | 5 | 4 | bayesian | 0.819 | 0.831 | 0.827 | 0.826 | 0.829 | 0.835 |
| | | | | | | | naive | 0.713 | 0.730 | 0.746 | 0.746 | 0.725 | 0.731 |
| 2 | 4 | 8 | 5 | 6 | 5 | 4 | bayesian | 0.752 | 0.776 | 0.775 | 0.780 | 0.774 | 0.778 |
| | | | | | | | naive | 0.668 | 0.690 | 0.695 | 0.694 | 0.700 | 0.691 |
| 3 | 4 | no | 5 | 6 | 5 | 4 | bayesian | 0.810 | 0.820 | 0.819 | 0.817 | 0.813 | 0.814 |
| | | | | | | | naive | 0.722 | 0.732 | 0.746 | 0.749 | 0.754 | 0.741 |

**Tabella 6:** Effetto di diverse discretizzazioni sulle performance del modello 2.

Per concludere l'esposizione dei risultati, si vanno a confrontare le performance dal punto di vista dell'accuratezza al crescere della dimensione dei training set per le reti bayesieane con quelle del modello Naive-Bayes, di cui prima vengono riportati i risultati nella Tabella 7. Le tendenze, come mostrato nelle figure a seguire (Figura 8, Figura 9, Figura 10), sono più o meno le stesse per tutti i modelli, confermandosi attorno ad un 10% di accuratezza. La dimensione del training set ha un'influenza poco determinante, i risultanti tendono a restare comunque stabili e, contrariamente a quanto ci si possa

aspettare, non evidenziano netti miglioramenti al crescere della dimensione dei dati usati in fase di training.

| Nome | Trainset | Testset | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|
| bnet_naive_test1 | 50% | 50% | 0.714 | 0.714 | 0.657 | 0.684 |
| bnet_naive_test2 | 55% | 45% | 0.733 | 0.688 | 0.718 | 0.703 |
| bnet_naive_test3 | 60% | 40% | 0.730 | 0.678 | 0.709 | 0..693 |
| bnet_naive_test4 | 65% | 35% | 0.729 | 0.699 | 0.711 | 0.705 |
| bnet_naive_test5 | 70% | 30% | 0.733 | 0.693 | 0.710 | 0.702 |
| bnet_naive_test6 | 75% | 25% | 0.734 | 0.696 | 0.692 | 0.694 |

**Tabella 7:** Risultati dei test per il modello Naive-Bayes.

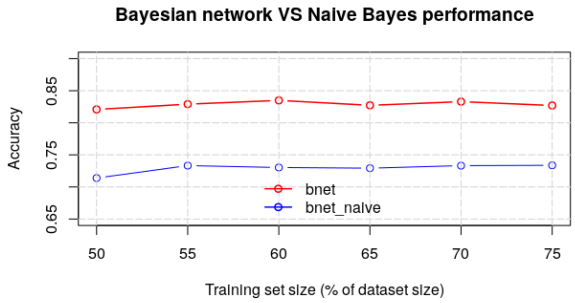



**Figura 8:** Analisi grafica delle performance per il modello 1.

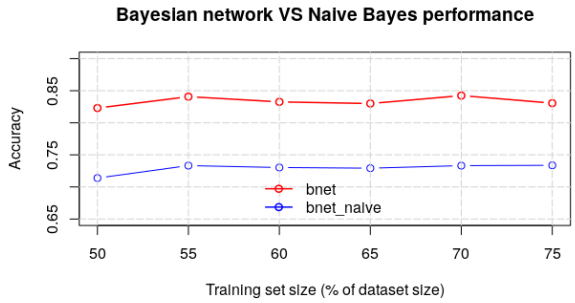



**Figura 9:** Analisi grafica delle performance per il modello 2.

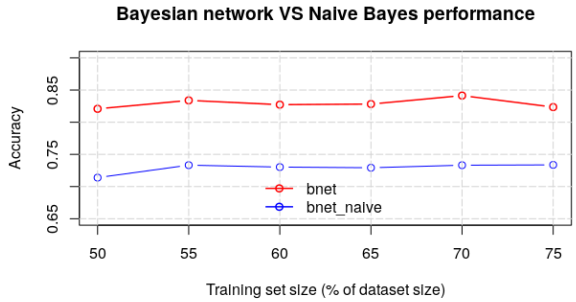
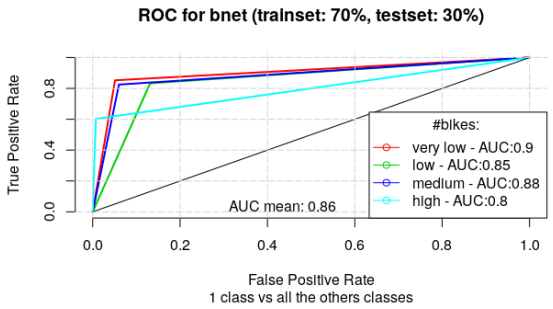



**Figura 10:** Analisi grafica delle performance per il modello 3.

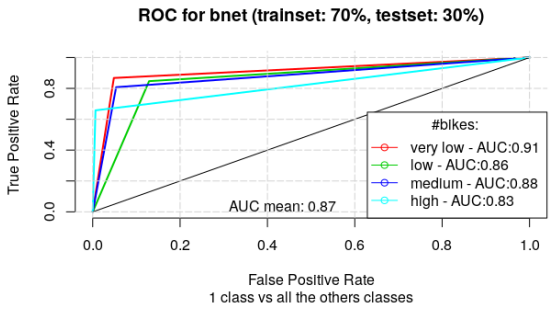
## 5.2 Curve ROC

In questa sezione sono riportate a seguire le curve ROC e l'AUC (area under curve) per ogni modello descritto in modo da poter valutare l'accuratezza dei modelli nei confronti

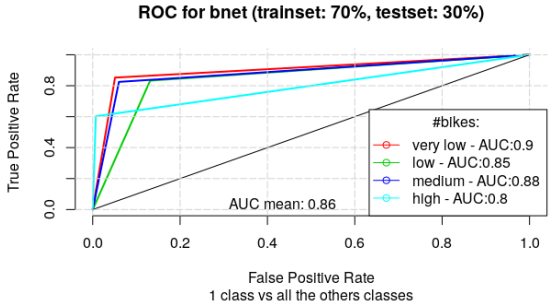
di ogni singola classe predetta. Essendo il problema trattato un problema multi-class ed essendo una curva ROC una curva per classificatori binari, le curve illustrate saranno ottenute in modalità ”1 vs all”, ovvero ogni singola linea, corrispondente ad una classe predetta, sarà ottenuta considerando tutte le predizioni per quella classe come ”1” e tutte quelle per le altre classi come ”0”.



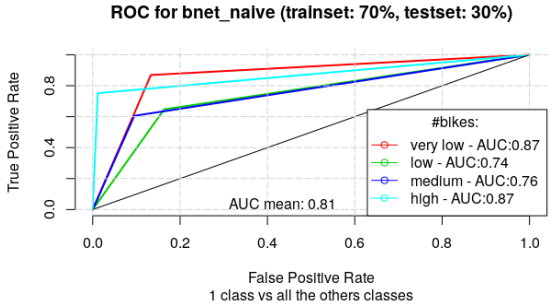

**(a)** Curva ROC per il modello 1.




**(b)** Curva ROC per il modello 2.




**(c)** Curva ROC per il modello 3.




**(d)** Curva ROC per il modello Naive-Bayes.

**Figura 11:** Analisi tramite curve ROC dell’abilità dei vari modelli nel predirre ogni singola classe della variabile `cnt`.

## 5.3 5-fold cross validation

Per validare i modelli ottenuti si è deciso di avvalersi della più nota tecnica di validazione, ovvero la k-fold cross validation.
Inizialmente sii è provato ad utilizzare k=10, ovvero 10-fold, ma i testset risultanti rischiavano di non contenere alcune delle classi da predirre, motivo per il quale si è virati su una 5-fold cross validation.
Questa tecnica permette di stabilizzare le stime delle misure di qualità in modo da fornire un dato molto più attendibile dei precedenti.
Come ci si aspettava sono state conservate le stesse osservazioni precedenti: il modello 2 si conferma leggermente il migliore mentre gli altri presentano gli stessi risultati. I risultati ottenuti sono riportati a seguire in Tabella 8, Figura 12, Figura 13 e Figura 14.

| Nome | Trainset | Testset | Accuracy | Precision | Recall | F-measure |
|---|---|---|---|---|---|---|
| modello 1 | 50% | 50% | 0.817 | 0.760 | 0.781 | 0.768 |
| modello 2 | 55% | 45% | 0.824 | 0.760 | 0.796 | 0.775 |
| modello 3 | 65% | 35% | 0.817 | 0.760 | 0.781 | 0.768 |
| Naive-Bayes | 55% | 45% | 0.707 | 0.673 | 0.670 | 0.668 |

**Tabella 8:** Misure di qualità dopo 5-fold cross validation.

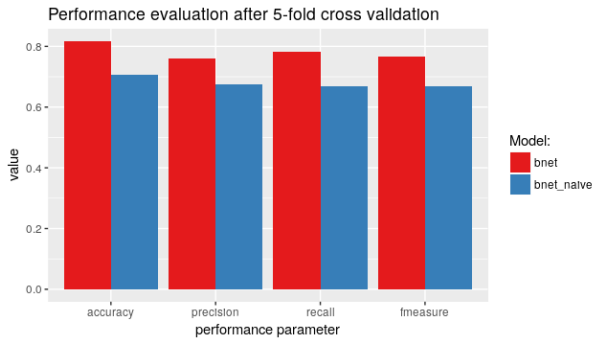



**Figura 12:** Analisi grafica delle performance dopo 5-fold cross validation per il modello 1.

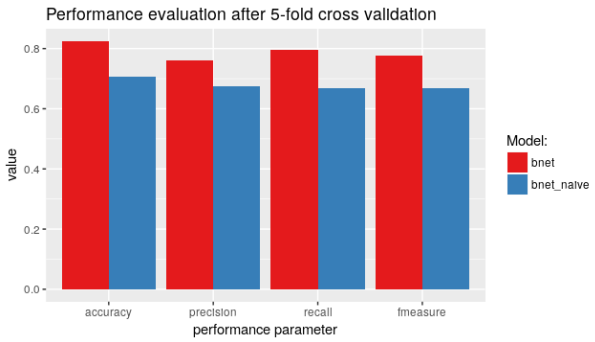



**Figura 13:** Analisi grafica delle performance dopo 5-fold cross validation per il modello 2.

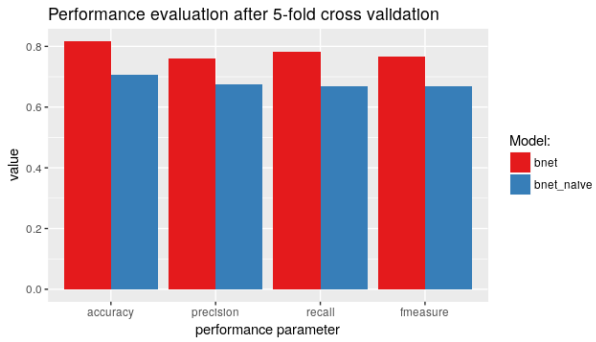



**Figura 14:** Analisi grafica delle performance dopo 5-fold cross validation per il modello 3.

# 6 Codice sorgente e interfaccia

La directory della consegna sarà così strutturata:

- Article: cartella contenente l'articolo di riferimento del dataset
- Correlation: cartella contenente diverse matrici di correlazione. È la cartella dove il codice sorgente esporta i grafici relativi alla correlazione.
- Datasets: cartella contenete i due dataset `hour.csv` e `day.csv`

- Models: cartella contenete i modelli implementati in formato `.RData`. Contiene i quattro modelli differenti dove ogni file include sia il modello bayesiano che il modello naive bayes.
- `project.R`: codice sorgente. Codice utilizzato per l'analisi del dataset, la creazione dei modelli e l'analisi delle performance.
- `projectGUI.R`: codice relativo all'interfaccia grafica.

## 6.1 Codice sorgente

Il codice sorgente potrebbe risultare lungo e di difficile comprensione a primo impatto, ma la sua struttura per blocchi corrispondenti a differenti task (importazione dei dataset, analisi dei dati, algoritmi per la definizione dei modelli, visualizzazione dei modelli, predizione, ...) lo rende più leggibile. Esso è ridondante, in quanto contiene ogni analisi e tentativo di modello effettuati, ma sarà necessario eseguirne solo piccole parti in base agli scopi prefissati.
Se lo si volesse rieseguire si consiglia di non eseguire i blocchi relativi agli algoritmi di inferenza dei modelli e alla costruzione dei modelli finali, ma di caricare direttamente i modelli finali dall'apposita cartella. Quest'operazione è dovuta al non determinismo degli algoritmi utilizzati in fase di inferenza della struttura del modello; essi infatti potrebbero fornire risultati diversi ad ogni iterazione impedendo di riottenere direttamente i modelli finali descritti in precedenza.
Se si avesse interesse a valutare le differenze fra i modelli si consiglia di utilizzare l'interfaccia grafica presentata nella prossima sezione.

## 6.2 Interfaccia

Per concludere il lavoro è stata implementata una semplice interfaccia utente. Essa è stata scritta in R, tramite l'utilizzo della libreria gWidgets2 [19], e fornisce uno strumento semplice e immediato per comparare i modelli descritti in questo elaborato.
L'interfaccia, di cui è riportato un esempio di utilizzo in Figura 15, permette di:

- scegliere quale rete bayesiana utilizzare
- aggiungere, rimuovere o modificare l'insieme delle evidenze;
- visualizzare le reti di entrambi i modelli;
- inferire la variabile target `cnt` su entrambi i modelli e visualizzarne i risultati, sia graficamente che in formato testuale.

Per poterla utilizzare basta aprire R o Rstudio, cambiare la working directory tramite il comando `setwd("directory del progetto")` e eseguire lo script `projectGUI.R` tramite il comando `source("projectGUI.R")`.

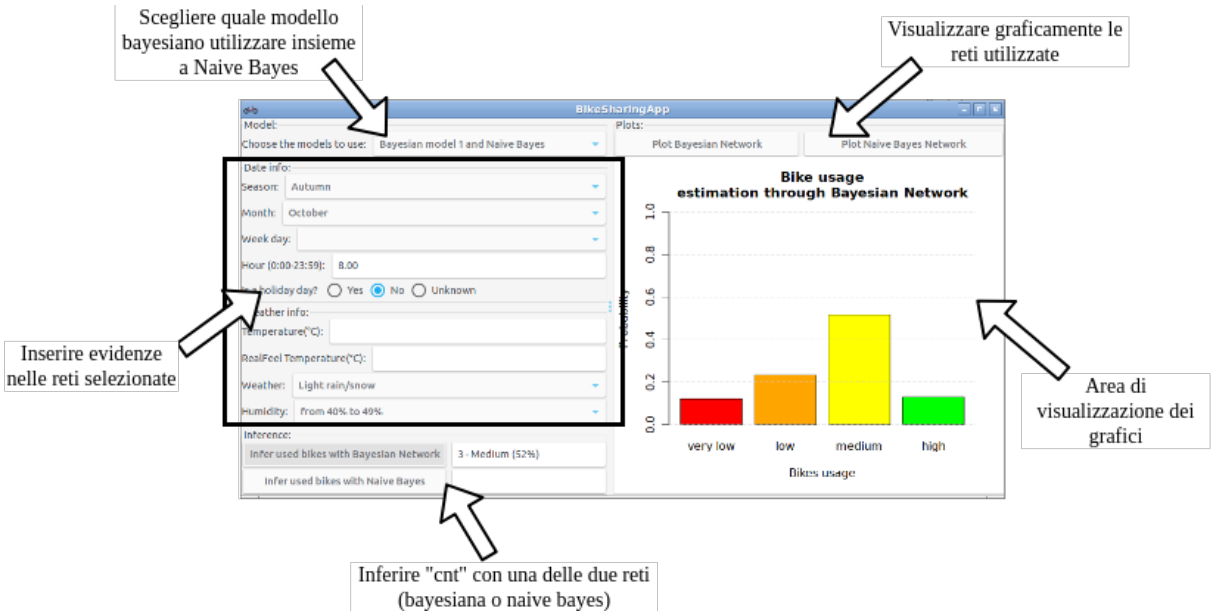



**Figura 15:** Interfaccia ed esempio di utilizzo.

# 7 Conclusioni e sviluppi futuri

Riassumendo i risultati analizzati nel capitolo precedente:

- Risulta possibile inferire la struttura delle reti bayesiane tramite diversi algoritmi ed il fatto che molti di essi forniscano la stessa rete o reti simili garantisce che le strutture dei modelli finali siano da ritenere abbastanza affidabili.

- La dimensione del training set non influenza in maniera determinante nessuna delle misure di qualità. Questo dimostra che le reti utilizzate comprendano le reali tendenze dei dati già con un trainset di dimensioni ridotte.

- Le reti bayesiane descritte presentano performance tra loro simili e migliori di circa il 10% rispetto a quelle ottenute tramite Naive-Bayes per ogni misura di qualità osservata.

- I problemi maggiori si hanno nella predizione della classe "$\texttt{cnt} = 4$". Questo è dovuto alla scarsa numerosità della classe ed al fatto che si stesse lavorando sui primi due anni del servizio, notoramiente anni troppo poco stabili dal punto di vista dei dati.

Da queste osservazioni si conclude che, momentaneamente, le reti osservate rappresentino il miglior risultato possibile. Per incrementare l'accuratezza, in futuro, si potrebbe:

- utilizzare un dataset più ampio

- utilizzare come dataset i dati relativi ad anni del servizio considerati più stabili, ovvero, anni dove la crescita del numero di stazioni e del numero di bici risulta terminata.
- cercare di ottenere il reale fattore di influenza degli anni trascorsi dall'inizio servizio, osservando anche gli anni successivi reperibili dal sito di Capital Bike Sharing.
- provare nuovi modelli ottenuti combinando le reti risultanti dall'esecuzione dei vari algoritmi per inferire la struttura. Questo è un tentativo che può comunque essere fatto, ma, come osservato dai risultati di questo elaborato, non dovrebbe essere possibile trovare una rete che garantisca risultati migliori.

# Riferimenti bibliografici


[1] Hadi Fanaee-T and Joao Gama. Event labeling combining ensemble detectors and background knowledge. **Progress in Artificial Intelligence**, pages 1–15, 2013. ISSN 2192-6352. doi: 10.1007/s13748-013-0040-3. URL [WebLink]. pages 4, 5

[2] R Core Team. **R: A Language and Environment for Statistical Computing**. R Foundation for Statistical Computing, Vienna, Austria, 2013. URL http://www.R-project.org/. pages 4

[3] Marco Scutari. Learning bayesian networks with the bnlearn R package. **Journal of Statistical Software**, 35(3):1–22, 2010. doi: 10.18637/jss.v035.i03. pages 4

[4] Søren Højsgaard. Graphical independence networks with the gRain package for R. **Journal of Statistical Software**, 46(10):1–26, 2012. URL http://www.jstatsoft.org/v46/i10/. pages 4

[5] Radhakrishnan Nagarajan, Marco Scutari, and Sophie Lèbre. **Bayesian networks in R**, volume 122. Springer, 2013. pages 4, 8

[6] Marco Scutari and Jean-Baptiste Denis. **Bayesian networks: with examples in R**. CRC press, 2014. pages 4

[7] Alexander John Hartemink. **Principled computational methods for the validation discovery of genetic regulatory networks**. PhD thesis, Massachusetts Institute of Technology, 2001. pages 5

[8] Marco Scutari. Bayesian network constraint-based structure learning algorithms: Parallel and optimized implementations in the bnlearn R package. **Journal of Statistical Software**, 77(2):1–20, 2017. doi: 10.18637/jss.v077.i02. pages 9

[9] P Bonissone, M Henrion, L Kanal, and J Lemmer. Equivalence and synthesis of causal models. In **Uncertainty in artificial intelligence**, volume 6, page 255, 1991. pages 9

[10] Dimitris Margaritis. Learning bayesian network model structure from data. 2003. pages 9

[11] Ioannis Tsamardinos, Constantin F Aliferis, Alexander R Statnikov, and Er Statnikov. Algorithms for large scale markov blanket discovery. In **FLAIRS conference**, volume 2, pages 376–380, 2003. pages 9

[12] Sandeep Yaramakala and Dimitris Margaritis. Speculative markov blanket discovery for optimal feature selection. In **Data mining, fifth IEEE international conference on**, pages 4–pp. IEEE, 2005. pages 9

[13] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2nd. **Prentice Hall Press, Upper Saddle River, NJ, USA.**, pages 111–114, 2003. pages 9

[14] Remco Ronaldus Bouckaert. **Bayesian belief networks: from construction to inference**. PhD thesis, 1995. pages 9

[15] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. **Machine learning**, 65 (1):31–78, 2006. pages 9

[16] Andre A Rupp, Dipak K Dey, and Bruno D Zumbo. To bayes or not to bayes, from whether to when: Applications of bayesian methodology to modeling. **Structural Equation Modeling**, 11(3):424–451, 2004. pages 11

[17] John K Kruschke, Herman Aguinis, and Harry Joo. The time has come: Bayesian methods for data analysis in the organizational sciences. **Organizational Research Methods**, 15(4):722–752, 2012. pages 11

[18] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2nd. **Prentice Hall Press, Upper Saddle River, NJ, USA.**, pages 481–482, 2003. pages 12

[19] John Verzani. **gWidgets2: Rewrite of gWidgets API for Simplified GUI Construction**, 2016. URL `https://CRAN.R-project.org/package=gWidgets2`. R package version 1.0-7. pages 16